

It's not fair - evaluating efficient disk scheduling

Alma Riska

Erik Riedel

Seagate Research

1251 Waterfront Place, Pittsburgh, PA 15222

e-mail: {Alma.Riska, Erik.Riedel}@seagate.com

Abstract

Storage system designers prefer to limit the maximum queue length at individual disks to only a few outstanding requests, to avoid possible request starvation. In this paper, we evaluate the benefits and performance implications of allowing disks to queue more requests. We show that the average response time in the storage subsystem is reduced when queuing more requests and optimizing (based on seek and/or position time) request scheduling at the disk. We argue that the disk, as the only service center in a storage subsystem, is able to best utilize its resources via scheduling when it has the most complete view of the load it is about to process. The benefits of longer queues at the disks are even more obvious when the system operates under transient overload conditions.

Keywords: disk scheduling, unfair scheduling, response time distribution, request starvation.

1. Introduction

In practice, storage subsystem designers put efforts on guaranteeing that the storage subsystem does not become the bottleneck of the system. They prefer to keep the queue length at disks small regardless of the load in the storage subsystem and achieve this by setting a queuing threshold at the disk. In cases when the queuing threshold is reached at the disk, the controller(s) of the storage subsystem and/or the device driver of the host operating system queue the incoming requests until the disk-queued requests are processed. Anecdotally, the preferred queuing threshold at the disk is often three or four requests. Storage system designers often come up with such thresholds using microbenchmarks that analyze the disk in isolation and without considering the dynamic environment under which it operates in practice.

Disk performance highly depends on how well its resources, i.e., head movement and platter rotation, are utilized. Originally, disk scheduling algorithms dealt with op-

timizing head movement and platter rotations for better disk utilization [3, 5, 6, 11]. Such optimization might lead to request starvation [14] and consequently to *unfair* scheduling algorithms. The low queuing threshold acts to minimize request starvation at the disk level when such scheduling algorithms are deployed. Nowadays, considerable amount of effort is placed in further utilizing disk operation by better placement of the data on the disk [9] and by scheduling background jobs while disk continues to serve the foreground ones [7].

In this paper, we evaluate the effects that higher queuing thresholds and unfair scheduling algorithms at disks have on the performance of the storage subsystem. We emphasize that by queuing more requests at the disk, the storage subsystem benefits by providing to the scheduling algorithms more information that is used for better disk resource utilization. We argue that even though the unfair scheduling algorithms starve more requests if longer queues build up at the disk, their percentage to the entire storage subsystem load remains small. While previous work [14] has analyzed and quantified starvation at disk under unfair scheduling, we analyze and evaluate request starvation as a function of queuing threshold at disk. We show that if any optimization techniques are applied at any level of the storage subsystem for better utilizing disk resources then the overall request starvation is *independent* from the queuing threshold at the disk. Finally, we emphasize that longer queues at the disk combined with unfair scheduling are the best and only solution to handle short but sharp overload conditions in the storage subsystem.

The rest of the paper is organized as follows: Section 2 overviews popular scheduling algorithms. Section 3 describes the experimental environment. Section 4 and Section 5 present analysis on the effects of the queuing thresholds at disk under random and traced workloads, respectively. Section 6 focuses on implications of deploying efficient scheduling at the device driver (or controller) and higher queuing thresholds at disk. Section 7 analyzes storage subsystem under non-stationary random workload. We summarize our findings in Section 8.

2. Related Work

Disk performance is bounded by its hardware specifications such as the speed of disk platter rotation and the speed of arm movement and it suffers even more if the arm movement and disk platter rotation are not optimized and fully utilized. Optimization of arm movement and disk platter rotation are related to two important components of the service process at disk, namely the seek time and the rotational latency, respectively. The seek time is related to the time needed for the head (arm) to be positioned on the right track and rotational latency is related to the time needed for the right block to come under the reading/writing head. Seek and rotational latency are minimized by serving requests not in the order in which they are received, but based on their position on the disk relative to the direction of the head and/or disk platter movement. Hence, scheduling policies benefit the most from request service time optimization. The disk scheduling algorithms that benefit from optimization of seek and/or rotational latency treat some of the outstanding requests at the disk *unfairly* because these algorithms do not schedule them in the order in which they were received. The unfair scheduling algorithms are distinguished as those that minimize only the seek time (i.e., Shortest Seek Time First - SSTF) [3] or both seek and rotational latency (i.e., Shortest Positioning Time First - SPTF) [11, 6]. Position-based scheduling algorithms achieve the best overall disk performance [6].

Request starvation is addressed by several variations of the position- and seek-based disk scheduling algorithms [14]. Among variations of seek-based scheduling algorithms, we mention SCAN or ELEVATOR [3], LOOK [8], (C-SCAN) [10], and VSCAN(R) [5]. Several variation of the position-based scheduling policy SPTF exist as well. In [6], an *aged* shortest positioning time (ASPTF) and a *batched* shortest positioning time (B-SPTF) scheduling algorithms are proposed and analyzed. More advanced disk scheduling frameworks that take into consideration both application and storage subsystem needs are discussed in [12].

Since the most popular disk scheduling algorithms such as SSTF and SPTF introduce requests starvation, i.e., highly variable request response times, system designers are reluctant to apply them over a large set of outstanding requests at the disk and for that reason apply the queuing threshold at the disk. The queuing threshold reduces response time variability since the scheduling algorithms apply over a small set of requests. In the following sections, we analyze disk and storage subsystem performance under the most popular scheduling algorithms (FCFS, SPTF, SSTF) as a function of the disk queuing threshold and argue over the performance benefits of allowing more requests to be queued at the disk.

3. Experimental environment

Our experimental results are obtained via simulation-based analysis. We use DiskSim 2.0 as a detailed disk simulator [4] and appropriately modify it to fit our needs. The single disk that drives the simulation is Cheetah 9.2LP, i.e., 10000 RPM and 9.1 GB. In our experiments, we focus on storage subsystems with high level architecture, as depicted in Figure 1. The outstanding requests are queued at the disk and at the device driver in a single disk storage subsystem and at the disks and the controller(s) in a multiple disk storage subsystem.

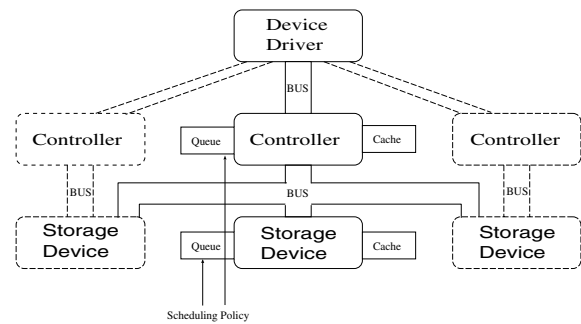


Figure 1. Storage subsystem architecture.

Figure 1 suggests that there are queues at various levels of the storage subsystem. However, there is only one service center in the storage subsystem and that is the disk. This means that the queue at the storage subsystem, although physically stored and administered at different levels, is logically only one, as depicted in Figure 2. Such queue structure indicates that the queuing threshold critically affects, most of all, the performance of the entire storage subsystem and not only that of the individual disk.

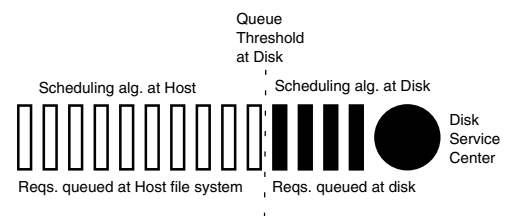


Figure 2. Requests are queued at two levels, i.e., at the disk and at the device driver (or controller). However, there is one one long logical queue.

Our experiments are driven by both random and traced workloads. The random workload is generated using the tools provided by Disksim 2.0, while the traced data are measured at HP Laboratories [1]. In the following sections, where we present our results, we assume that the workload is random and the storage system with only one disk unless otherwise stated. We assume the queue structure depicted in Figure 2 and FCFS scheduling at the device driver or the controller, unless otherwise specified.

4. Higher queuing thresholds at disk under random workload

The performance of disk scheduling algorithms is sensitive to the changes in the storage subsystem load. The higher the load the larger the gap between the performances of different scheduling algorithms. The fair and simple FCFS yields the longest average request response time. The shortest average response time is achieved under position-based scheduling algorithms [6]. Our focus is to quantify the benefits of queuing more requests at the disk rather than at the device driver. We evaluate such benefits by increasing the queuing threshold at the disk for a given system load, which we measure as the average number of outstanding requests in the storage subsystem.

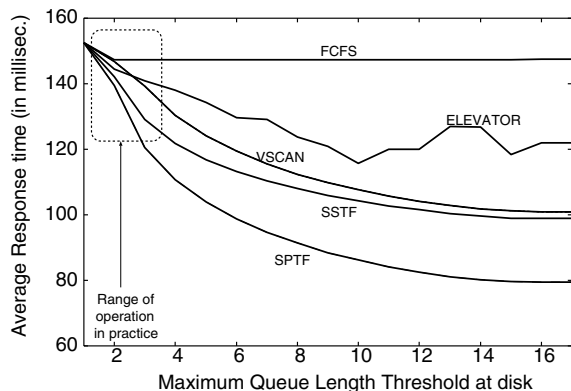


Figure 3. Average response time as function of maximum queue length threshold at disk for 16 outstanding requests in the system

Figure 3 plots the average response time in the storage subsystem as function of the queuing threshold at the disk for the average load of 16 outstanding requests. The average response time is measured for different scheduling algorithms and the best performance improvement, i.e., reduction in average response time, when increasing the queuing threshold at disk is obtained under the position-based scheduling at disk. Note that in Figure 3, we highlight the

area that corresponds to queuing thresholds of up to four, i.e., the common range of disk queuing thresholds today.

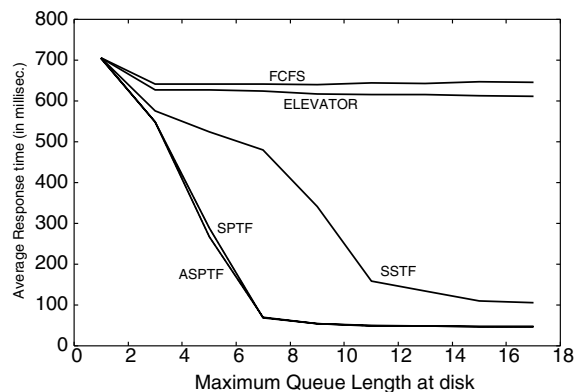


Figure 4. Average response time in a system with four disks, one controller, and an average of 64 outstanding requests.

We also evaluate the average response time of a storage subsystem with four disks and one controller, when the average load is 64 outstanding requests i.e., average of 16 outstanding requests per disk (see. Figure 4) Note a reduction in response time by a factor of more than seven when the queuing threshold at each disk is increased from one to 16 and the applied scheduling algorithm is position-based.

4.1. Response time distribution

The average response time is reduced when more requests are queued at the disk because optimizing among more requests allows the disk to better utilize its resources. The disk resources utilization is achieved by scheduling for service the most favorable request, i.e., the one whose position yields the shortest seek time and rotational latency. Such criteria might treat unfairly some requests positioned away from the most common head position resulting in request starvation and higher variability in the request response time. In this subsection, we evaluate and quantify the request starvation under the unfair scheduling algorithms as a function of the queuing threshold at the disk.

Figures 5 and 6 plot the tail of the response time distribution at the disk for queuing thresholds of 8 and 16, respectively, under several disk scheduling algorithms. The load in the storage subsystem is 16. The crossover point in both figures, clearly, indicates that the majority of requests under FCFS scheduling algorithm exhibits long response times, while under the position- and seek-based scheduling algorithms the majority of requests exhibits short response times. These plots suggest that more than 90% of the re-

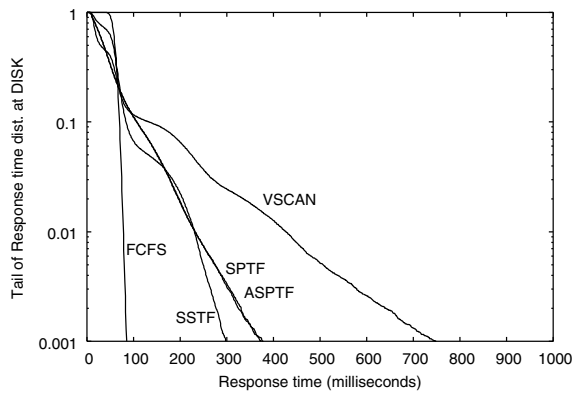


Figure 5. Tail of response time distribution at disk in a system with an average load of 16 outstanding requests and threshold of 8

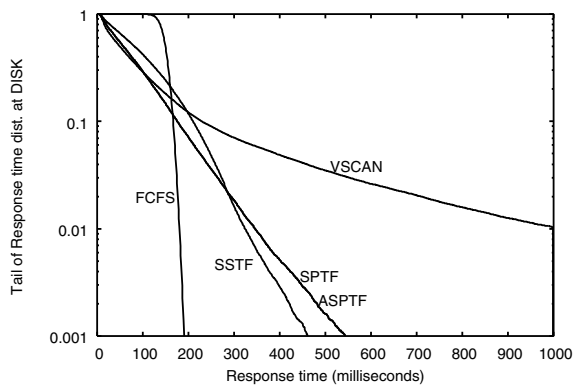


Figure 6. Tail of response time distribution at disk in a system with an average load of 16 outstanding requests and threshold of 16

quests under the position-based scheduling algorithms (Figure 5) exhibit shorter response times in comparison to FCFS policy and only 1% of the requests exhibit response times of up to twice the response time under FCFS scheduling algorithm. Observe that, while the amount of starvation introduced from the position-based scheduling algorithms for both queuing thresholds is the same when compared with the respective performance under FCFS, the overall performance is better when the queuing threshold at disk is 16. This emphasizes that queuing more requests improves disk performance without introducing more request starvation.

Although position- and seek-based scheduling policies do increase the variability in the request response time at the disk level, controlling it by limiting the number of requests queued at the disk level is not necessarily the correct solution. The variability is an inherent characteristic of the

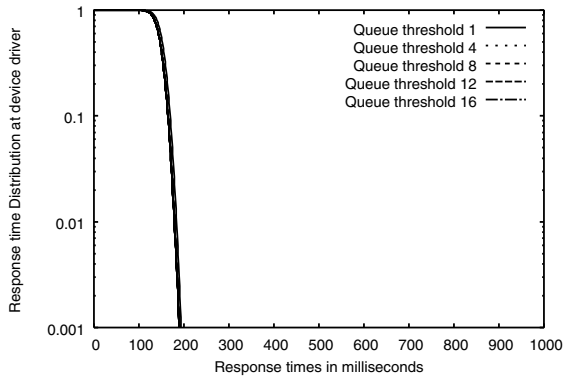
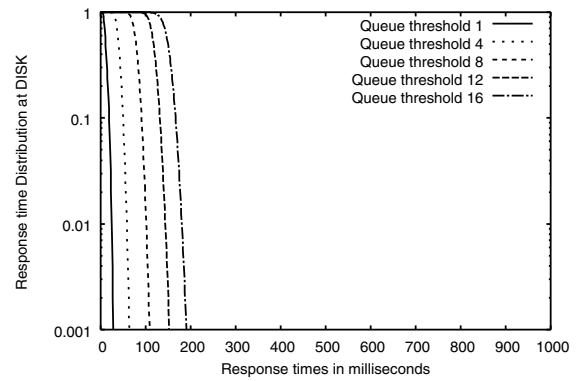


Figure 7. Tail of response time distribution at disk (upper) and device driver (lower) for FCFS. There are in average of 16 outstanding requests in the system.

scheduling algorithm and not of the queuing system. The requests that are not queued directly at the disk, are queued first at the device driver or at the controller of the storage subsystem (as depicted in Figure 2) and wait for the requests queued at the disk to be served. Hence, the variability in response time experienced by the requests scheduled at the disk via unfair scheduling algorithms affects the variability of the response time of the requests queued at the device driver/controller as well.

We support this claim with the results presented in Figures 7 and 8, where we illustrate the tail of the response time distribution at the disk and at the device driver for FCFS and SPTF, respectively. The upper graphs in Figures 7 and 8 represent the tail of response time distribution at the disk for different queuing thresholds and average load of 16 outstanding requests. The lower graphs in Figures 7 and 8 represent the tail of the response time distribution at the device driver. Observe that for the FCFS (Figure 7), the queuing threshold does not affect the overall performance of the storage subsystem. For the SPTF (Figure 8), the queuing threshold does affect the distribution of the response time at

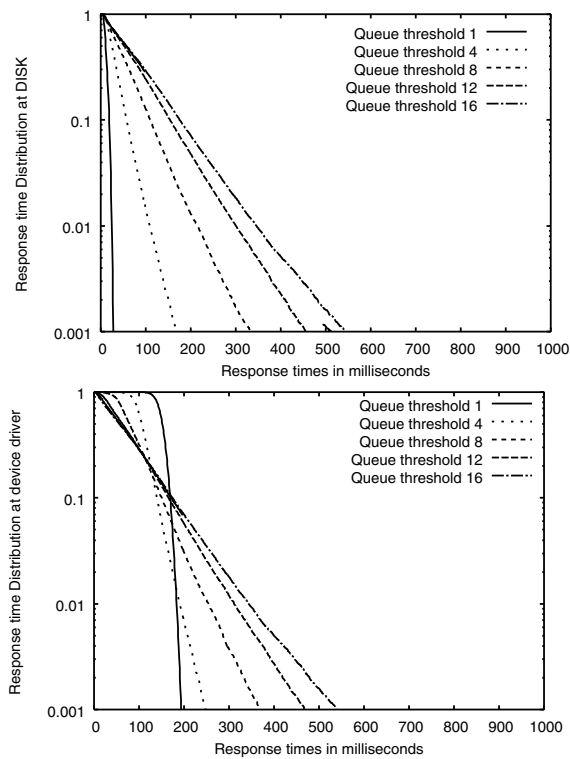


Figure 8. Tail of response time distribution at disk (upper) and device driver (lower) for SPTF. There are in average of 16 outstanding requests in the system.

the disk level. However at the device driver, SPTF scheduling penalizes only a very small portion of the requests, i.e., 1% of requests experience twice the response time under SPTF when compared with the response time under FCFS. The crossover points in these plots indicate the improvement in response time for the group of requests that benefit from SPTF.

5. Higher queuing thresholds at disk under traced workload

In order to understand the effects of longer queues at the disk level under various scheduling algorithms, we repeated most of the previous experiments with traced workloads. We use the HPL CELLO'96 workload [1] and picked approximately a quarter of a million requests to drive our simulation. The average load in the system is 25 outstanding requests. In order to understand the characteristics of the trace that we use in our analysis, we plot in Figure 9 the distribution of the seek distance in the stream of requests of the traced workload. The seek distance is defined as the

number of tracks the head has to move in order to position in the right track for the next scheduled request. To simplify the presentation, we grouped requests in classes whose seek distance is in increments of 500 tracks. In Figure 9, we present the distribution of seek distances under the FCFS scheduling policy. Observe that most of the requests (almost 90%) experience a seek distance of less than 500 tracks, which indicates that unlike the random workload, the traced workload exhibits more locality in the stream of requests. We note that the seek distance distribution remains similar for other scheduling policies such as SPTF and SSTF. The seek distance distribution does not change for this particular workload even when we increase the queuing threshold at the disk.

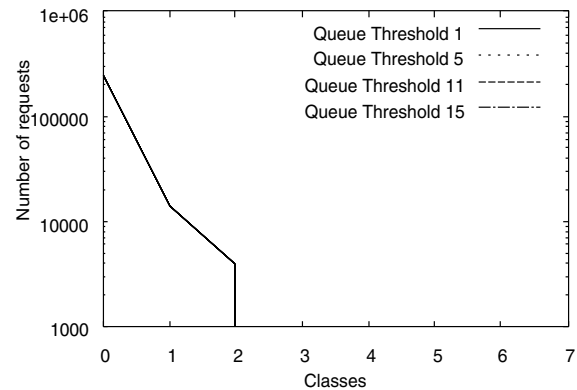


Figure 9. Number of requests per class of request seek distance under FCFS for the traced workload.

In Figure 10, we plot the average response time for FCFS and SPTF¹ under the traced workload as a function of the queuing threshold at disk. Similar to the performance improvement quantified in the previous section, under the position-based scheduling algorithm, SPTF, the average response time reduces by a factor of two when almost all of the outstanding requests are queued at the disk rather than at the device driver.

Similar to random workloads, under traced workload, the storage subsystem exhibits higher request starvation at the disk under position- and seek-based scheduling and higher queuing thresholds. In Figure 11, we present the tail of response time distribution at the device driver under FCFS and SPTF at disk for several queuing thresholds. Observe that for this particular trace, the tail of the response time distribution at the device driver under the SPTF scheduling al-

¹ We focus only on FCFS and SPTF scheduling algorithms because previously we established that SPTF performs the best among scheduling algorithms and FCFS is the only fair one.

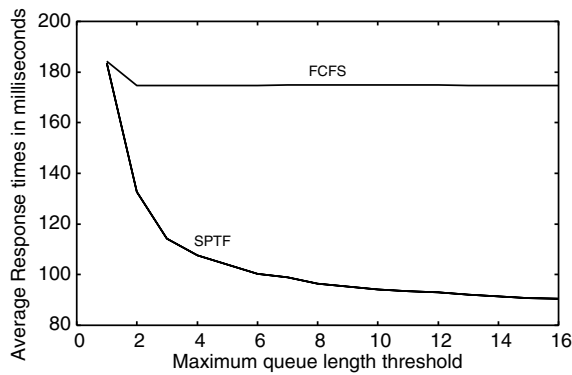


Figure 10. The average response time under SPTF and FCFS as a function of the queuing threshold for the traced workload.

gorithm at disk is not worse than under the FCFS scheduling algorithm at disk. This result emphasizes that specific workload characteristics are essential for the efficiency of the scheduling algorithms.

6. Better scheduling at the disk and at the host file system

Requests in a storage subsystem are queued in two different physical levels, i.e., at the disk and, if the queuing threshold is reached, at the device driver in a single-disk storage subsystem or at the controller in a multi-device storage subsystem. There is only one service center in the entire storage subsystem and that is the disk, which means that regardless of where queuing occurs, all requests are to be served by the disk. Such hierarchical queuing at the storage subsystem allows for better scheduling algorithm at both the device driver and the disk.

In addition to the FCFS scheduling algorithm, one can deploy at the device driver level other scheduling algorithms such as the seek-based ones. Seek-based scheduling algorithms work optimally if the *physical block numbers - PBNs* are used, and yields similar results if *logical block numbers - LBNs* are used. Hence, at the device driver level, one can deploy algorithms such as SSTF and VSCAN using as reference the LBNs of each request. The utilization of the disk resources is by far optimal in such scenario, as SPTF scheduling algorithm is the one that best optimizes the use of disk resources. Another reason why seek-based scheduling at the device driver level only partially improves the performance of the storage subsystem is that the most accurate information (head position and direction of arm movement) that is used in optimal disk scheduling algorithms is available only at the disk level. This is why SPTF and other

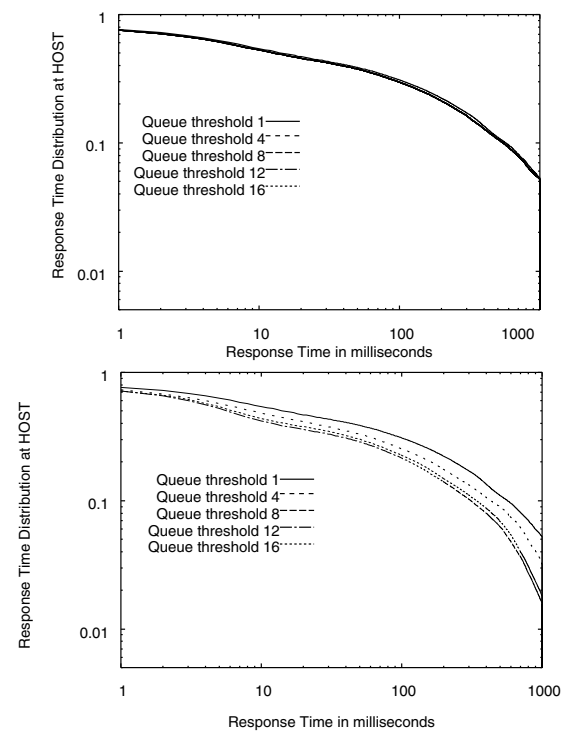


Figure 11. The tail of the response time distribution at device driver under FCFS (upper graph) and SPTF (lower graph) under the traced workload. There are in average 25 outstanding requests in the storage subsystem.

position-based scheduling algorithms are not deployable at other levels of the storage subsystem but the disk.

Figure 12 illustrates the average response time from a storage subsystem as a function of the queuing threshold at the disk, when FCFS, SSTF, and VSCAN are deployed at the device driver and SPTF is deployed at the disk. The average system load is 16 outstanding requests. Observe that the improvement in the average response time for the combination of seek-based scheduling at host and SPTF scheduling at disk is only 50% as the queuing threshold is increased at the disk. However, note that the best performance is achieved only if all requests are queued at the disk and SPTF scheduling is implemented there.

Figure 13 illustrates the tail of the response time distribution in the case when SSTF scheduling algorithm is applied at the device driver and SPTF scheduling algorithm applied at the disk. Observe that the tail of the response time distribution does not depend on the queuing threshold. If any optimization is done at the device driver, extending it further at the disk does not affect the amount of overall starvation experienced by the requests in the storage subsystem.

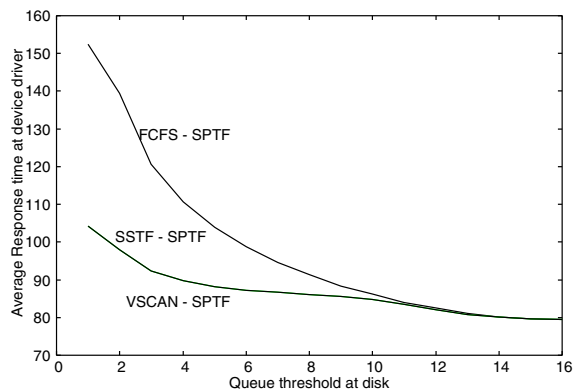


Figure 12. Average response time at the device driver under FCFS, SSTF, and VSCAN at the device driver and SPTF at the disk. Average load is 16 outstanding requests.

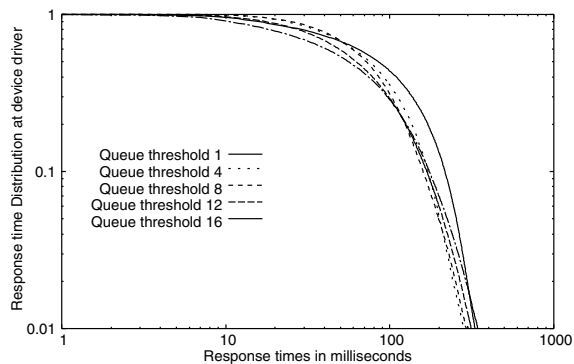


Figure 13. The tail of response time distribution at the device driver under SSTF scheduling algorithm at the device driver and SPTF scheduling algorithm at the disk. There are in average 16 outstanding requests in the storage subsystem.

7. Non-stationary random workload

With the new set of application that are being deployed and supported by the Internet infrastructure, new patterns have emerged on the workload seen by the systems. One of the most important characteristics is the brief and sharp fluctuations in the intensity of the arrival process [13]. The sharp fluctuations in the load are characteristics of the storage subsystem as well as of other subsystems. Critical in such sharp and short overload conditions is that the system handles the overload without performance degradation or complete collapse. The key on handling short overloads is

to fully utilize the resources of the storage subsystem and, for the request scheduling, this means that the best scheduling algorithm should be deployed and the entire picture of the storage subsystem load should be available to the disk. By allowing the disk to have knowledge of the situation, i.e., queue the entire set of outstanding requests, it optimizes its resources better and handles the overload.

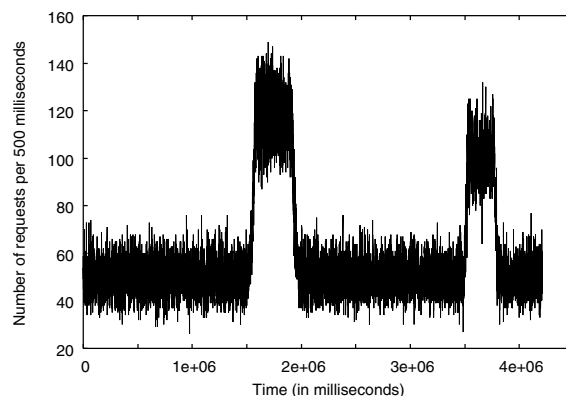


Figure 14. Non-stationary arrival process.

We evaluate overload conditions by generating a random traced workload with a non-stationary arrival process depicted in Figure 14. Each point in the figure represents the number of requests arriving in time intervals of 500 milliseconds. The overload conditions are two, the first one lasting for about 8 minutes and the second one for less than 5 minutes, together accounting for approximately half the number of requests in the entire trace. If, under such an arrival process, the scheduling algorithm of choice is FCFS then the overload condition is not sustained and the system collapses (the average response time reaches the minute mark level and is not tolerated by the system). If SPTF scheduling is applied at disk then the overload condition is sustained (Figure 15). Nevertheless, how well the overload condition is handled, highly depends on the number of requests queued at the disk. Figure 15 represents the response time of requests as a function of the arrival time under the SPTF scheduling algorithm at disk. In the case of a queuing threshold of 4 (the line that performs worse) only one overload condition is sustained, while with a queuing threshold of 1024 (the line that performs the best) both overload conditions are sustained, yielding a response time that is more than twenty times better for the first overload condition. Observe that once the arrival intensity is high and the storage subsystem in overload, the unfairness of the scheduling algorithms such as SPTF is surpassed many times by the benefits of optimizing disk resources utilization.

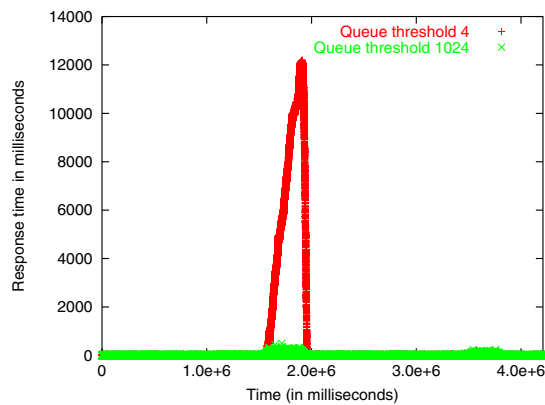


Figure 15. Response time of individual requests at the host file system under SPTF scheduling at disk as a function of time.

8. Conclusions

In practice, the designers of storage systems use microbenchmarks to evaluate disk performance and prefer to allow disks to queue only a few requests in the hope of maintaining low response times and avoiding request starvation at the disk level. However, the disk is often the only service center of a much larger storage subsystem, and by queuing only a few of the current outstanding requests at the disk, one prevents the disk from having a good picture of the entire workload that it has to serve and best utilizing its resources. The requests that are not sent to the disk are still queued elsewhere in the system, and users must still wait for them, so overall response time is lower if the disk is less efficiently used.

In this paper, we evaluate these design choices and conclude that the queuing threshold at the disk does not affect negatively the performance of the overall storage subsystem. On the contrary, higher queuing thresholds at the disks improve the overall response time by a factor of two for low-to-medium system load and by a factor as high as twenty under overload conditions.

We show that the unfair scheduling algorithms at the disk such as SPTF, SSTF, treat unfairly only a small portion of the entire set of requests in the storage subsystem. Most importantly, the amount of starvation introduced by the unfair scheduling algorithms is almost independent of the queuing threshold at the disk, especially when any type of disk resource optimization happens in the upper levels of the storage subsystem. We conclude that the best performance by a disk is achieved when efficient, and unfair, scheduling algorithms such as SPTF are applied and all the outstanding requests are queued at the disk.

References

- [1] CELLO 1996 Traces. Storage Systems Program HP Laboratories, URL: <http://tesla.hpl.hp.com/public-software/>.
- [2] E. G. Coffman and M. Hofri. On the expected performance of scanning disks. *SIAM Journal of Computing*, 10(1):60–70, 1982.
- [3] P. J. Denning. Effects of scheduling on file memory operations. In *Proceedings of AFIPS Spring Joint Computer Conference*, pages 9–21, 1967.
- [4] G. R. Ganger, B. L. Worthington, and Y. N. Patt. The DiskSim simulation environment, Version 2.0, Reference manual. Technical report, Electrical and Computer Engineering Department, Carnegie Mellon University, 1999.
- [5] R. Geist and S. Daniel. A continuum of disk scheduling algorithms. *ACM transactions on Computer systems*, 5(1):77–92, 1987.
- [6] D. M. Jacobson and J. Wilkes. Disk scheduling algorithms based on rotational position. Technical Report HPL-CSP-91-7rev1, HP Laboratories, 1991.
- [7] C. R. Lumb, J. Schindler, G. R. Ganger, D. F. Nagle, and E. Riedel. Towards higher disk head utilization: extracting free bandwidth from busy disk drives. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 87–102, Oct. 2000.
- [8] A. G. Merten. Some quantitative techniques for file organization. Technical report, Ph.D. Thesis, Technical Report Number 15, Department of Computer Science, University of Wisconsin, 1970.
- [9] J. Schindler, J. L. Griffin, C. R. Lumb, and G. R. Ganger. Track-aligned extents: matching access patterns to disk drive characteristics. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, pages 259–274, Jan. 2002.
- [10] P. H. Seaman, R. A. Lind, and T. L. Wilson. An analysis of auxiliary-storage activity. *IBM System Journal*, 5(3):158–170, 1966.
- [11] M. Seltzer, P. Chen, and J. Osterhout. Disk scheduling revisited. In *Proceedings of the Winter 1990 USENIX Technical Conference*, pages 313–323, Washington, DC, 1990.
- [12] P. J. Shenoy and H. M. Vin. Cello: A disk scheduling framework for next generation operating systems. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 44–55, June 1998.
- [13] M. Welsh and D. Culler. Adaptive overload control for busy Internet servers. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems, (USITS'03)*, Seattle, WA, 2003.
- [14] B. L. Worthington, G. R. Ganger, and Y. N. Patt. Scheduling for modern disk drives and non-random workloads. Technical Report CSE-TR-194-94, Computer Science and Engineering Division, University of Michigan, 1994.