RJ 10206 (95080) February 16, 2001
Computer Science

Research Report

HANDLING SEEK TIME VARIABILITIES IN SHORTEST ACCESS TIME FIRST DISK SCHEDULING

Spenser Ng

IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

RJ 10206 (95080) February 16, 2001
Computer Science

Handling Seek Time Variabilities in Shortest Access Time First Disk Scheduling

Spencer Ng
IBM Almaden Research Center
650 Harry Road
San Jose, California 95120

Abstract:
Previous studies have already established the superiority of disk scheduling algorithms based on total access time over those that are based only on seek time. The most straightforward algorithm in this class of scheduling policy is oftentimes known as the Shortest Access Time First (SATF) algorithm. However, all previous studies on SATF scheduling use a disk drive model in which the seek time is precisely known and without variation. In practice, due to a number of reasons, there is always some deviation in seek time from its nominal specification. This variation can be problematic for the SATF algorithm. The purpose of this paper is to investigate the effect of seek time variability on the behavior of SATF algorithm, and explore ways for optimizing its performance under this real world condition.

Keywords: *disk scheduling, I/O architecture, optimization, performance, rotational position, scheduling algorithm, seek, shortest access time first*

1. Introduction

For over ten years, numerous papers have been pointing out the widening gap between the performance of CPU and that of mass data storage. The main reason is due to the fact that CPU is all electronics whereas mass data storage (mostly magnetic) involves rotating storage devices. Physicist and mechanical engineers have yet to find a way to make increase in mechanical speed at a rate anywhere close to the rate at which clock speed of silicon circuit is increasing. This I/O performance gap is well known to all in the computer field and needs no further elaboration.

Various methods for narrowing this I/O performance gap has been proposed and are the subject of numerous papers. They include using cache to hide the slowness of the mechanical device [7,13], using multiple disk drives to increase parallelism (striping [8] and disk array [10]), organizing data on a disk drive such that the most frequently used and/or related data are located close together to improve locality of access [1,15], various latency reduction methods [9], and scheduling I/O requests [2,3,14] so as to reduce the average mechanical time of each request. This paper is related to the last method. The remainder of this paper assumes that the reader is familiar with the workings of a magnetic disk drive; if not, Ref. [11] contains an excellent tutorial. Optical disk drives share many similar operating characteristics with magnetic disk drives. Thus, the discussion in this paper is equally applicable to magnetic and optical disks.

Disk drives provide random access by having an actuator that moves the recording head(s) to the desired radial position while the rotation of the disk(s) brings the desired target sector to the head. The time to position radially the head is referred to as the *seek time*, while the rotational time required for the angular positioning is commonly known as the *rotational latency*. Hence, the mechanical access time involved in accessing data in a disk drive consists of two components, viz., the seek time and the rotational latency. One of the first published articles on disk scheduling was by Denning in 1967 [2] which described the shortest-seek-time-first (SSTF) and the SCAN policies as different methods for reducing the average seek time for disk access. Many subsequent papers that proposed and/or evaluated various scheduling policies [3,14] also focused only on reducing the seek time, even though that is only one of two components of a disk drive's access time. In recent years, there has been an explosion in the number of papers dealing with disk scheduling for multimedia applications (e.g. streaming video) where the storage system must meet deadlines in servicing the I/O requests. However, the trend continues that those studies deal exclusively with scheduling of seek only, ignoring rotational latency.

While the concept of a disk scheduling policy that reduces the sum of seek time and rotational latency has been discussed amongst practitioners of disk drive performance for many years, there is a deft of published work that deals with this class of scheduling algorithms. The most straightforward algorithm in this class of scheduling policy is oftentimes known as the *Shortest Access Time First* (*SATF*) algorithm. It is also referred to by other names such as Shortest Positioning Time First (SPTF) and Rotational Positioning Optimization (RPO). It operates by selecting from among all the I/O requests waiting in the queue the one that has the smallest seek time plus latency, i.e., total access time, to be the one to service next. While this approach is not globally optimal, it is computationally relatively simple. The works by Seltzer, et al, [12] and by

Jacobson and Wilkes [6] appear to be the first publications to suggest that the SATF algorithm outperforms all known seek optimization algorithms. A paper by Hwang and Shin [5], and the work of Worthington, et al, [16] using real workload traces, also came to similar conclusion.

All previous work on SATF scheduling [4,5,6,12,16] assume a model of the disk drive in which the seek time is precisely known and without variation. However, in practice, there is always some deviation in seek time for any given seek distance from its nominal specification. As explained in Section 3, this variation can be problematic for the SATF algorithm. The purpose of this paper is to investigate the effect of seek variability on the behavior of SATF algorithm and explore ways for optimizing its performance under this real world condition.

2. Method of Study

2.1. Metrics and Workloads

The average mechanical access time, as defined in the Introduction, is used as the obvious metric for evaluating the performance of disk scheduling algorithms, since minimizing the access time is the goal of such algorithms. Inclusion of overheads and data transfer time when making comparison would only dilute the difference in performance between different schemes.

A random access pattern where every sector on the disk has equal probability of being accessed is synthetically generated as the workload for this paper. Such an access pattern is generally likely to be encountered in a server shared by multiple users where SATF scheduling will do the most good. One parameter that affects the performance of any scheduling algorithm is the queue depth, Q. In this study, we simulate a heavily loaded system so that the queue is always full. The queue is initialized to have Q requests pending at the beginning of the simulation. When one request is serviced from the queue, another request is immediately generated and added to the queue. This methodology is identical to the simulation work of [12].

2.2. Simulation

A simple simulator which keeps track of the radial and angular positions of the disk and of all the I/O requests in the queue is written for the study. Each data point is generated by the simulation of one million I/O requests; this is at least one order of magnitude longer than the studies of [5] and [12]. The simulator also validates extremely well against the exact analytical modeling solution for a queue depth of 2.

2.3. Disk Parameters

Since our study is only concerned with seek time and rotational latency, only two disk characteristics need to be input to the simulator, viz., the seek profile of the disk drive and its rotational speed. For this study, a state-of-the-art high performance disk drive is simulated. The rotational speed is chosen to be 10,000 rpm. 15K rpm drives are now starting to appear in the market, but the vast majority of drives being used in a server environment still run at 10K rpm.

For seek characteristics, we employ a commonly used model of constant acceleration and deceleration. For such a model, the seek profile can be characterized by the simple equation of

$$T = A + B\sqrt{x} \qquad (1)$$

where **T** is the seek time, **A** is the settling time, **B** is the acceleration factor, and ***x*** is the seek distance represented as a fraction of the maximum seek distance. Thus x is a number between 0 and 1. It is well known that the average seek distance for random access , with any seek model, is 1/3 the maximum seek distance. For the seek model used here, it can also be easily shown that the average seek time is A + 8B/15. In this paper, we assume A = 0.5 msec. and B = 9 msec., representing a state-of-the-art disk drive with an average seek time of 5.3 msec.

2.4. Fairness

While previous studies on SATF algorithm have established that it is a more effective algorithm than algorithms based only on seek time, they have also shown that it is highly susceptible to request starvation, exhibiting a high degree of variation in response time [6,12,16,]. However, those studies also investigated minor modifications to the SATF algorithm and showed that they can achieve similar performance as the SATF and still ensure fairness. Therefore, fairness will not be the subject of investigation in this study, with the assumption that it can be successfully addressed using the techniques given in those studies.

3. Seek Time Variation

Basically, the seek time of any disk drive increases monotonically with the seek distance, regardless of the exact characteristics of its seek profile. Hence, for disk scheduling algorithms, such as Shortest Seek Time First (SSTF) and C-Look [14], which are based only on seek time, exact knowledge of the seek profile is not required. These algorithms can perform their scheduling by operating on seek distances. Furthermore, the effectiveness of such algorithms is not compromised by deviations in the disk drive's actual seek behavior from its assumed model.

A major difference between SATF algorithm and algorithms which are based only on seek time is that it requires a very precise profile of the seek characteristics of the drive. This is because whether a scheduled request can actually access its target sector in the estimated time may depend on whether the actual seek is completed within the seek time estimated by the model or not. (The seek model used in a disk drive for scheduling can be a mathematical equation, or it can be a table.) If an actual seek time exceeds the estimated seek time, but is less than the total estimated access time, the request will still be completed in the time estimated and the scheduling algorithm works as expected. However, if the actual seek time exceeds the total estimated access time, as would likely happen if the latency component of the total access time is small, then by the time the recording head arrives at the desired cylinder the target sector of the request would have already gone by. In this case, the drive must wait for roughly another revolution before the target sector rotates under the head again. We will refer to this event as a

missed revolution. When this occurs, the performance of the disk drive is significantly degraded since the time of one revolution (6 msec. for a 10,000 rpm drive) is added to the service time of the I/O request.

In the ideal case, a disk drive's servo system should perform consistently and very close to the model specified by the designer. However, in reality, due to a variety of reasons, there is always some deviation in the actual seek time from its nominal specification (some drive designs may exhibit more variance, some may have less). The variation may be due to:

- Direction of seek - seeking from the outside of a disk towards the inside may be different than seeking for the same distance from the inside towards the outside.
- The location on the disk where the seek takes place - seeking from cylinder 1 to cylinder 1000 may have a different seek time than from cylinder 2001 to cylinder 3000.
- Changes in the operating temperature of the disk drive.
- Unit to unit variation because of manufacturing tolerance.
- For any given unit, a mechanical device always have some degree of variability.

Therefore, missed revolution is a fact of life for SATF scheduling algorithm, and will be the subject of study in the remainder of this paper.

To understand the effect of seek time variation on the performance of SATF, we define a simple model here for characterizing the variation. It is not the purpose of this paper to find an accurate characterization model - in fact, each manufacturer's drives may exhibit different characteristics and therefore require different models. Rather, a simple model is used here as a means to illustrate and understand the dynamics behind the relationship of seek variation and SATF. In this model, for any average seek time T, let us define a random variable ***d*** to be the deviation from the average seek time, expressed as a fraction of T. In other words, the actual seek time t = (1+d)T. Let the probability density function f(d) of d be

$$f(d) = \begin{cases} (D + d) / D^2 & \text{if } -D < d < 0 \\ (D - d) / D^2 & \text{if } 0 < d < D \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

where **D** is the maximum amount that an actual seek time may deviate from the average, again expressed as a fraction of the average seek time T. Its distribution function F(d), which is the probability that a seek is completed in time (1+d)T or less, is

$$F(d) = \begin{cases} (D^2 + 2dD + d^2) / 2D^2 & \text{if } -D < d < 0 \\ (D^2 + 2dD - d^2) / 2D^2 & \text{if } -D < d < 0 \end{cases} \qquad (3)$$

We define D to be the *degree of seek variation* of a disk drive. These functions are graphically illustrated in Fig. 1.

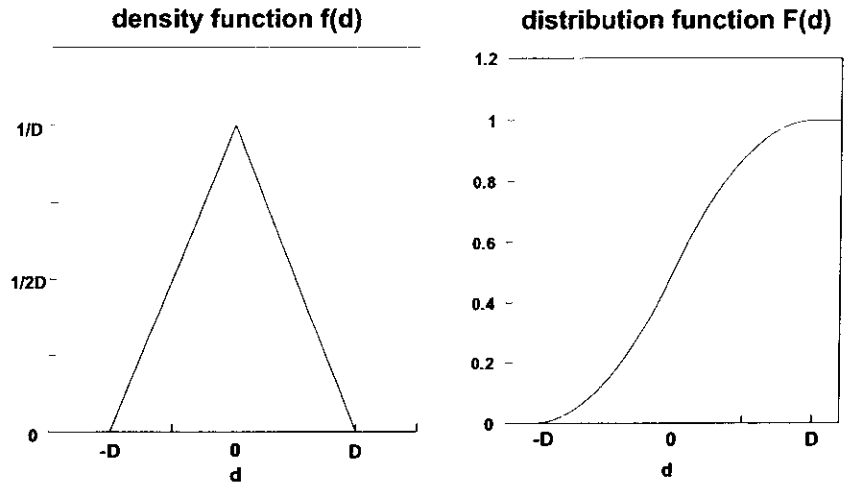

Figure 1. Density and distribution functions of deviation d

Using the average seek time T of equation (1) for scheduling, Tables 1a and 1b show the impact of various degree of variation D on the effectiveness of SATF. Table 1a compares the average access time actually achieved against the ideal case of no seek variation (D = 0). Table 1b shows the corresponding percent of times that missed revolutions occur. As expected, the results clearly show that for any given queue length Q, degradation from the ideal case goes up as the degree of variation increases. It is also observed that for a given degree of variation D, the percent of missed revolutions increases with Q. This is because the longer the queue, the more likely there will be an I/O request with very short rotational latency, which is more susceptible to variation in seek time. Lastly, it can be seen that the percentage of degradation in access time is small for small Q, as shown in Fig. 2. This is due to a combination of fewer missed revolutions and larger average access time for the no variation case.

Table 1a. Average access time (in msec.) for various degrees of variation D

	Q = 4	Q = 8	Q = 16	Q = 32	Q = 64	Q = 128	Q = 256
No variation	5.82	4.86	4.01	3.30	2.75	2.31	1.96
D = 0.05	5.91	4.97	4.12	3.43	2.88	2.45	2.11
D = 0.10	6.00	5.06	4.23	3.54	3.00	2.58	2.25
D = 0.15	6.07	5.15	4.33	3.65	3.11	2.69	2.37
D = 0.20	6.14	5.23	4.43	3.75	3.21	2.80	2.48
D = 0.25	6.21	5.31	4.51	3.83	3.31	2.90	2.58

Table 1b. Percent of missed revolutions for various degrees of variation D

	Q = 4	Q = 8	Q = 16	Q = 32	Q = 64	Q = 128	Q = 256
D = 0.05	1.46%	1.74%	1.93%	2.06%	2.19%	2.32%	2.48%
D = 0.10	2.87%	3.35%	3.67%	3.94%	4.18%	4.44%	4.74%
D = 0.15	4.15%	4.85%	5.32%	5.71%	6.02%	6.39%	6.83%
D = 0.20	5.42%	6.28%	6.87%	7.35%	7.72%	8.14%	8.65%
D = 0.25	6.61%	7.59%	8.33%	8.79%	9.22%	9.76%	10.39%

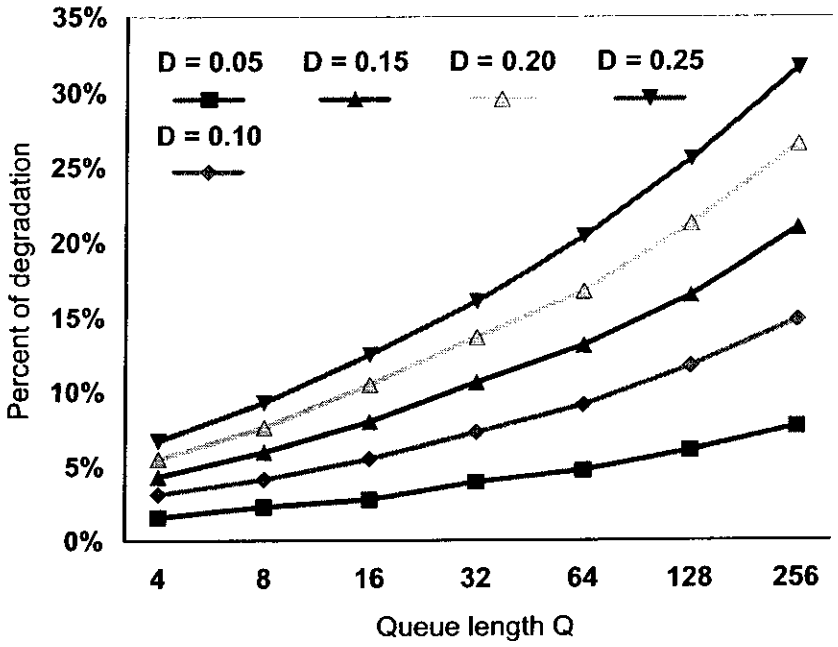

Figure 2. Percent of degradation due to seek time variation

4. Optimizing for SATF

Since the penalty for a missed revolution is high (it adds the time of one extra revolution of the disk to the service time), it may seem intuitively obvious that the best performance will be attained if we ensure that missed revolution never happens. This is achieved by using a conservative seek time estimate of (1+D)T for scheduling, rather than the average seek time T. By the definition of D in section 3, 100% of all seeks are completed within the time (1+D)T.

To test this hypothesis, we introduce a new parameter S, called the *seek scheduling factor*. S is a fraction such that when multiplied to the degree of variation D defines the seek profile to be used in scheduling. In other words, (1+SD)T is the estimated seek time. Note that S can be a negative number and can vary from -1 to 1. S = 1 is the most conservative seek profile and guarantees that missed revolutions will never occur.

Tables 2a and 2b show the effect of using various seek scheduling factor S on SATF scheduling, assuming a degree of variation D of 0.2. The average access times attained for different queue lengths are tabulated in Table 2a, and the corresponding percentage of missed revolutions are listed in Table 2b. In Table 2a, the shortest average access time for each queue depth Q is highlighted. The access times are also graphically illustrated in Fig. 3 for queue depths of 8, 16, 32 and 64, while Fig. 4 plots the percentage of missed revolutions for scheduling factors of -0.2, 0, 0.2, 0.4 and 0.6.

The results, perhaps somewhat surprisingly, clearly show that adopting a conservative seek profile (S=1) for scheduling is not the best strategy. This is because, while missed revolution is completed avoided, it loses out on a lot of opportunities for scheduling requests with shorter rotational latency that the disk drive could have serviced in time without missing revolutions. For the queue depths of interest studied, the best performance is obtained by using a more aggressive seek profile than S=1. A second observation that can be drawn from the results is that the smaller the queue depth, the more aggressive (smaller S) the seek profile should be in order to achieve optimum performance. This is because, with a small number of requests to choose from, the expected service time of the second best (less aggressive) choice for shortest access time is on the average longer than if the queue depth is long and there are many requests to choose from. Therefore, the penalty for not choosing the shortest candidate (because of using a more conservative seek profile) is larger for small queue depths. This suggests that one can dynamically adjust what seek profile to use for scheduling based on how deep the command queue is.

Table 2a. Average access time (in msec.) for D=0.2 and various scheduling factors S

	Q = 4	Q = 8	Q = 16	Q = 32	Q = 64	Q = 128	Q = 256
S = 1.0	6.45	5.49	4.60	3.82	3.18	2.67	2.27
S = 0.9	6.39	5.43	4.54	3.77	3.14	2.63	2.24
S = 0.8	6.33	5.37	4.49	3.73	3.09	2.60	2.21
S = 0.7	6.29	5.32	4.44	3.68	3.06	2.58	2.19
S = 0.6	6.23	5.27	4.40	3.65	3.04	2.56	2.18
S = 0.5	6.19	5.23	4.36	3.62	3.02	2.55	2.18
S = 0.4	6.16	5.20	4.34	3.61	3.02	2.56	2.20
S = 0.3	6.14	5.19	4.33	3.61	3.04	2.59	2.23
S = 0.2	6.12	5.18	4.34	3.64	3.07	2.64	2.29
S = 0.1	6.12	5.19	4.37	3.68	3.13	2.70	2.37
S = 0.0	6.14	5.23	4.43	3.75	3.21	2.80	2.48
S = -0.1	6.17	5.29	4.49	3.84	3.32	2.92	2.62
S = -0.2	6.22	5.36	4.59	3.95	3.46	3.07	2.78

Table 2b. Percent of missed revolutions for D=0.2 and various scheduling factors S

	Q = 4	Q = 8	Q = 16	Q = 32	Q = 64	Q = 128	Q = 256
S = 1.0	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
S = 0.9	0.00%	0.00%	0.01%	0.01%	0.01%	0.01%	0.01%
S = 0.8	0.04%	0.05%	0.05%	0.06%	0.07%	0.06%	0.07%
S = 0.7	0.14%	0.17%	0.19%	0.20%	0.20%	0.23%	0.24%
S = 0.6	0.34%	0.41%	0.45%	0.48%	0.51%	0.53%	0.57%
S = 0.5	0.67%	0.78%	0.86%	0.93%	0.96%	1.04%	1.11%
S = 0.4	1.16%	1.34%	1.49%	1.60%	1.70%	1.78%	1.88%
S = 0.3	1.85%	2.14%	2.35%	2.54%	2.67%	2.83%	2.98%
S = 0.2	2.73%	3.19%	3.52%	3.81%	3.95%	4.19%	4.48%
S = 0.1	3.94%	4.53%	5.00%	5.37%	5.60%	5.96%	6.37%
S = 0.0	5.42%	6.28%	6.87%	7.35%	7.72%	8.14%	8.65%
S = -0.1	7.20%	8.33%	9.13%	9.71%	10.20%	10.84%	11.48%
S = -0.2	9.22%	10.75%	11.74%	12.47%	13.18%	13.82%	14.75%

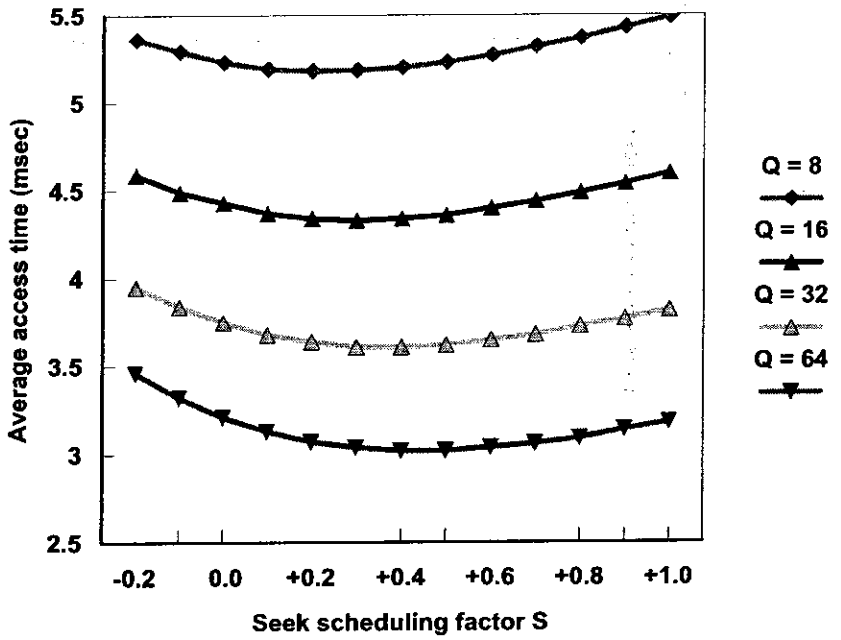

Figure 3. Average access time for D=0.2 and various Q and S

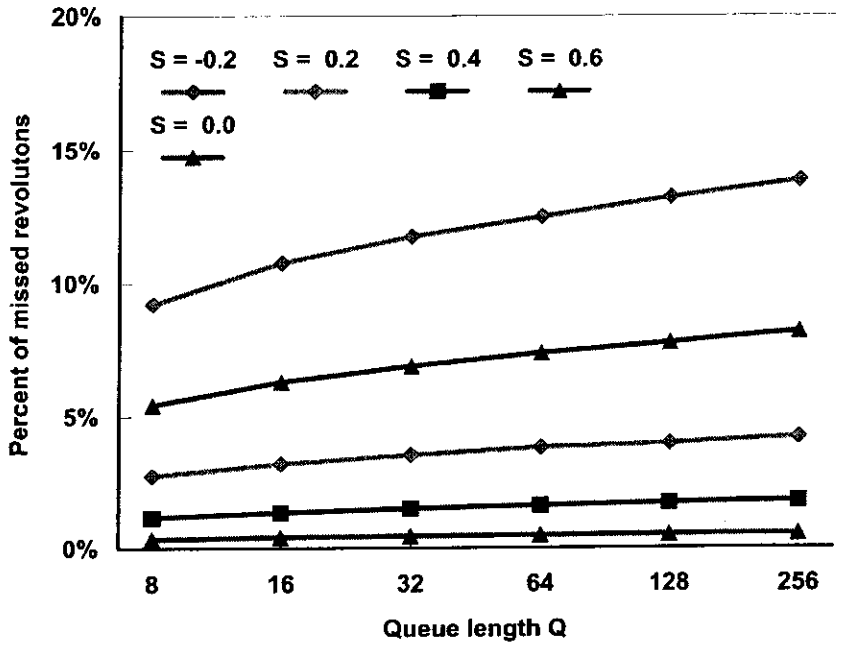

Figure 4. Percent of missed revolutions for D=0.2 and various Q and S

5. A Strategy for Handling Missed Revolution

The previous section has shown that optimum performance for SATF algorithm is obtained by using seek scheduling factors S that are more aggressive than 1.0 despite some amount of missed revolutions in the requests serviced. Since missed revolutions still occur, a simple idea of further improvement is proposed here. The proposed enhancement consists of the following steps.

1. When a missed revolution condition is detected, instead of waiting idly for one disk revolution, servicing of the selected (*primary*) request is terminated. A disk drive operating with the SATF scheduling algorithm can detect that a missed revolution condition is encountered when on arrival at the destination track the angular position indicates that the rotational latency will be substantially larger (by roughly one revolution) than the estimated latency calculated during the scheduling.
2. The disk drive immediately starts to service an alternate (*secondary*) request.
3. The secondary request has already been selected by the controller of the disk drive during the mechanical access time of servicing the primary request.
4. The secondary request is selected also using SATF scheduling, using as the starting reference point the start position of the primary request plus an angular displacement θ, where θ is the angle covered by the rotation of the disk during time (1-S)D T', where T' is the estimated average seek time of the primary request.

5. The secondary request uses a seek scheduling factor of S=1 to ensure that it will not have missed revolution.

We call the above technique *SATF with missed revolution retargeting*. The effectiveness of this technique is simulated for the case of D = 0.2 and shown in Table 3. Again, the shortest average access time for each queue depth Q is highlighted. Comparing Table 3 with Table 2a, it can be seen that some reduction in the average access time is indeed achieved, especially for larger values of Q.

Figure 5 graphically illustrates the comparison for queue depth Q = 64 as an example. In addition to showing that retargeting gives a better performance, a few other observations can be made here. First, as the scheduling factor gets closer to 1, there is little difference between the two algorithms. This is because when scheduling is done with a conservative seek profile, missed revolution does not happen very often and, hence, retargeting hardly occurs. Conversely, the more aggressive a seek profile is used for scheduling, the greater the improvement retargeting shows, due to the increased frequency of missed revolutions. Third, when retargeting is used, optimum performance is attained using a slightly more aggressive seek profile (lower value of S) than when no retargeting is used. This is because the penalty for estimating wrong is now lessened, so one can afford to more aggressively go after candidates with shorter total estimated time. Lastly, performance is somewhat less sensitive to the seek scheduling factor S when retargeting is used, because the penalty for guessing the seek time wrong is mitigated by the retargeting action. This means when the retargeting algorithm is used, one can afford to be a little bit less precise in selecting the seek profile.

Table 3. Average access time for D=0.2 using missed revolution retargeting

	Q = 4	Q = 8	Q = 16	Q = 32	Q = 64	Q = 128	Q = 256
S = 0.8	6.34	5.37	4.49	3.72	3.10	2.60	2.21
S = 0.7	6.28	5.32	4.44	3.68	3.06	2.57	2.18
S = 0.6	6.23	5.27	4.39	3.63	3.02	2.54	2.16
S = 0.5	6.19	5.22	4.35	3.60	2.99	2.52	2.14
S = 0.4	6.15	5.19	4.32	3.57	2.98	2.51	2.13
S = 0.3	6.10	5.15	4.29	3.55	2.97	2.50	2.13
S = 0.2	6.08	5.13	4.27	3.54	2.96	2.51	2.13
S = 0.1	6.06	5.12	4.27	3.55	2.97	2.52	2.15
S = 0.0	6.05	5.11	4.27	3.56	3.00	2.54	2.18
S = -0.1	6.04	5.12	4.29	3.59	3.03	2.58	2.22
S = -0.2	6.04	5.14	4.32	3.63	3.08	2.63	2.27

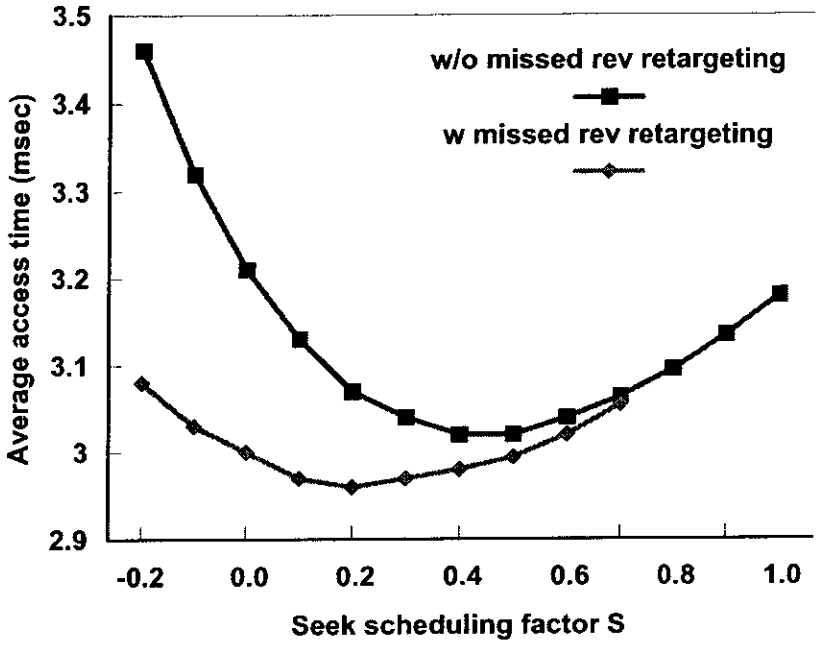

Figure 5. SATF with missed revolution retargeting for Q=64

6. Summary

In this paper we have discussed a practical issue with implementing the SATF scheduling algorithm in a disk drive. Because of the inevitability of variation in a disk drive's mechanical servo system, missed revolutions may occur in the actual execution of the algorithm, affecting its performance. We examined how to optimize the performance of the SATF algorithm in the face of this reality of life. It was shown that the best performance is not achieved by scheduling conservatively to completely avoid missed revolution. Instead, a somewhat more aggressive seek profile allowing for some missed revolutions to occur will actually result in a better overall performance. Furthermore, the shorter the command queue, the more aggressive the seek profile should be. This suggests that one can dynamically adjust what seek profile to use for scheduling based on how deep the command queue is. A simple enhancement to the SATF algorithm was also proposed in this paper, and its effectiveness and behavior were investigated. One benefit of using the proposed method is that one can be less precise in selecting the seek profile used for scheduling and still achieve near optimal perforamnce. With the knowledge described in this paper, and following the methodology given, a disk drive designer will be able to tune his implementation of the SATF algorithm to attain maximum performance.

References

1. S. Aky and K. Salem, "Adaptive Block Rearrangement", *ACM Trans. on Computer Systems*, Vol. 13, No. 2, May 1995, pp. 89-121.
2. P. Denning, "Effects of Scheduling on File Memory Operations", *AFIPS Spring Joint Conference*, April 1967, pp. 9-21.
3. R. Geist and S. Daniel, "A Continuum of Disk Scheduling Algorithms", *ACM Trans. on Computer Systems*, Vol. 5, No. 1, February 1987, pp. 77-92.
4. M. Heath, D. C. Pruett, and B. Nguyen, "Method for reducing rotational latency in a disc drive", US Patent 5570332, issued Oct. 29, 1996.
5. K. Hwang and H. Shin, "New Disk Scheduling Algorithms for Reduced Rotational Latency", Proc. of the 3rd International Symposium on Database Systems for Advanced Applications, Taejon, South Korea, April 1993, pp. 395-402.
6. D. Jacobson and J. Wilkes, "Disk Scheduling Algorithms Based on Rotational Positioning", Technical Report, Hewlett Packard Laboratories, HPL-CSP-91-7, February 1991.
7. R. Karedla, J. S. Love, and B. Wherry, "Caching Strategies to Improve Disk System Performance", *IEEE Computer*, March 1994, pp. 38-46.
8. M. Kim, "Synchronized Disk Interleaving", *IEEE Trans. on Computers*, Vol. C-35, No. 11, November 1986, pp. 978-988.
9. S. Ng, "Improving Disk Performance via Latency Reduction", *IEEE Trans. on Computers*, Vol. C-40, No. 1, January 1991, pp. 22-30.
10. D. Patterson, G. Gibson, and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)", ACM SIGMOD conference proceedings, Chicago, IL., June 1988, pp. 109-116.
11. C. Ruemmler and J. Wilkes, "An Introduction to Disk Drive Modeling", *IEEE Computer*, March 1994, pp. 17-28.
12. M. Seltzer, P. Chen, and J. Ousterhout, "Disk Scheduling Revisited", *Winter USENIX*, Washington, DC, January 1990, pp. 313-324.
13. A. Smith, "Disk Cache - Miss Ratio Analysis and Design Considerations", *ACM Trans. on Computer Systems*, Vol. 3, No. 3, August 1985, pp. 161-203.
14. T. Teorey and T. Pinkerton, "A Comparative Analysis of Disk Scheduling Policies", *Commun. ACM*, Vol. 15, No. 3, March 1972, pp. 177-184.
15. P. Vongsathorn and S. Carson, "A System for Adaptive Disk Rearrangement", Software Practice and Experience, Vol. 20, No. 3, March 1990, pp. 225-242.
16. B. Worthington, G. Ganger and Y. Patt, "Scheduling Algorithms for Modern Disk Drives," *Sigmetrics*, Santa Clara, CA, May 1994, pp. 241-251.