A Comparative Analysis of Disk Scheduling Policies

Toby J. Teorey and Tad B. Pinkerton
University of Wisconsin*

Five well-known scheduling policies for movable head disks are compared using the performance criteria of expected seek time (system oriented) and expected waiting time (individual I/O request oriented). Both analytical and simulation results are obtained. The variance of waiting time is introduced as another meaningful measure of performance, showing possible discrimination against individual requests. Then the choice of a utility function to measure total performance including system oriented and individual request oriented measures is described. Such a function allows one to differentiate among the scheduling policies over a wide range of input loading conditions. The selection and implementation of a maximum performance two-policy algorithm are discussed.

Key Words and Phrases: access time, analytical models, auxiliary storage, direct access storage, disk analysis, disk scheduling, performance criteria, peripheral memory devices, real-time systems, response time, rotational delay, scheduling policies, seek time, simulation, storage units, time-sharing systems, waiting time

CR Categories: 3.72, 4.32, 6.34

* Department of Computer Sciences, Madison, WI 53715.

Presented at the Third ACM Symposium on Operating Systems Principles, Palo Alto, California, October 18–20, 1971.

Introduction

A common cause of inefficiency in the implementation of multiprogramming systems is poor use of direct access storage. For fixed head devices (disks and drums), hardware and software techniques [1, 3, 9] have been developed which optimize system efficiency and individual response time simultaneously. When disk arm movement is required, however, no single scheduling policy exists which optimizes both of these performance measures at the same time or, indeed, optimizes a single measure over an entire range of actual operating conditions. The goal of the following analysis is to determine whether such a policy can be formulated.

Several scheduling policies have been proposed in the literature for handling a queue of requests on a movable head disk, e.g. the shortest-seek-time-first (SSTF) policy, the SCAN access method, the N-step scan, and the Eschenbach scheme. These methods all improve the performance of a system over the first-come-first-served (FCFS) policy. We discuss the interrelationship of these policies in terms of disk arm movement and with respect to performance measures such as expected seek time, expected waiting time for individual requests, and variance of individual waiting times. Although past studies have relied on mean queue length as the controlled variable, we prefer mean input rate because it is a better representation of loading conditions.

This paper is concerned entirely with the best design of a basic disk subsystem. It does not consider the nature of the computer processes that generate I/O requests, nor does it include special hardware configurations or file organization techniques. From the following discussion one should have a broad but concise understanding of the considerations necessary in developing future direct access storage devices.

FCFS. A detailed study of the physical movement of the arm of read/write heads on a movable head disk reveals a close relationship between major scheduling policies. The one exception is the FCFS policy, which has a random seek pattern. It does not take advantage of positional relationships between I/O requests in the current queue or of the current cylinder position of the read/write heads.

SSTF. The shortest-seek-time-first policy takes advantage of positional relationships between the read/write heads and I/O requests to improve the throughput, but it is at the expense of discrimination against individual requests [3]. Although there is a finite probability of such discrimination, simulation of SSTF shows that as the load on the disk queuer increases, the difference in performance between SSTF and SCAN (see below) is nearly indistinguishable. To see that this is true we must visualize the disk arm movement for a constant queue length of L requests under heavy loading conditions. Once the disk arm makes progress in a given direction, the density of requests immediately behind the arm's movement is much less than the density immediately in front of it. Even though the total number of requests becomes greater behind the arm's movement, the SSTF policy is only dependent on the request nearest the current arm position. Consequently the arm is not necessarily biased toward the center of the disk. Occasionally a new request occurs which causes the disk arm to change direction, but the density of requests in that vicinity strongly favors a resumption of the original direction. When the queue becomes large, this model bears some resemblance to the classical ruin problem of probability theory [4]. (The probabilities of going right or left in the SSTF model are dependent upon the current locations of requests in the queue. Consequently they are variant with time. In the classical ruin model the probabilities must be constant.) When the queue is small, the SSTF policy results in random arm movement. Examples can easily be contrived to show that this is not an optimal policy under light loading conditions.

SCAN. The SCAN access method [3] can be viewed as the basic method of improving the efficiency of a disk queuer. With SCAN the disk arm acts as a shuttle as it sweeps back and forth across all the cylinders, servicing requests. It changes direction only at the inner and outer cylinders. SCAN was proposed as an alternative to SSTF because it does not allow the discrimination against individual requests that occurs with SSTF.

N-STEP SCAN. Another alternative is the N-step scan policy. Its purpose is to decrease the variance of individual waiting times while not significantly degrading throughput. Requests are grouped in sizes of N or less; that is, the minimum of (a) N, and (b) the current queue length L whenever the previous group has been completely serviced. Lower variance results from the fact that once a group has been formed no reordering of requests is allowed. The shortest Hamilton path for these N (or L) requests can be found by various techniques from graph analysis. These cumbersome procedures can be avoided by using a nearly optimal policy suggested by Frank [5]: (a) first scan in the direction for which the farthest request distance is a minimum; (b) then scan back until the remaining requests have been serviced. Under heavy loading conditions this becomes a pure scan from one extreme point on the disk to the other and approaches the optimal scan.

ESCHENBACH. The Eschenbach scheme [11] is a deterministic scanning technique designed for message-switching systems which normally operate under heavy loading conditions. An order E Eschenbach scan is defined as E revolutions per cylinder, with m/E sectors serviced per revolution (m = total number of sectors/track). In an order E scan, $E - 1$ sectors are skipped for each sector serviced. It has the following properties: (a) in order to allow the possibility of servicing all m sectors for a cylinder, the arm must remain in position for at least E revolutions; (b) if the cylinders are labeled $0, 1, \cdots, c - 1$, the arm may spend E revolutions on cylinder 0 before moving to cylinder 1, E revolutions on cylinder 1, and so on until E revolutions are spent on cylinder N. The arm then moves directly back to cylinder 0 without stopping at any of the intermediate cylinders.

We note that (a) and (b) imply that rotational optimization is built into the Eschenbach scheme in the sense that as many sectors as possible are serviced within a given rotation and that service is rendered in the sequence in which sectors are encountered. Rotational optimization is a standard technique for fixed head devices.

In summary, FCFS performs no optimization; SSTF, SCAN, and the N-step scan tend to optimize on seek times; and the Eschenbach scheme optimizes on rotation as well as on seek times. Later we discuss implementing rotational optimization in the other policies. Under heavy loading conditions it is obviously useful, and it is necessary to make a fair comparison with the Eschenbach scheme.

As we can now see, all the proposed improvements over the FCFS policy are variations of a basic scanning technique. Let us look at the differences among these policies for expected seek time T_{sk}, expected service T_s, and expected individual waiting time W. We assume the following:

- Only a single disk module is considered.
- Requests for equal size records are uniformly distributed over the entire disk. Each track is assumed to have m sectors, with each sector containing space for exactly one record.
- There is one movable head per surface, and all disk arms are attached to a single boom so they must move simultaneously. A single position of all the read/write heads defines a cylinder.
- The seek time characteristic is linear (see Figure 1).

Fig. 1. Seek time characteristic: IBM 2314 disk and linear approximation.

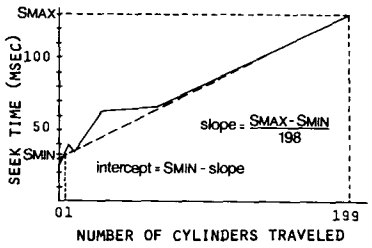


Table I. Expected Seek Time Expressions (T_{sk})

Policy	Expected seek time T_{sk} (conditioned on the fact that the disk arm moves)	
FCFS	$S_{min} + (S_{max} - S_{min}) \frac{(c+1)(c-1)}{3c^2}$ [Weingarten]	(1)
SSTF	$S_{min} + (S_{max} - S_{min}) \frac{(c-1)\beta - 1}{2(c-2)}$ where $\beta = \frac{1}{(L+1)}\left[1 + \frac{1}{L+2}\left(\frac{c}{c-1}\right)^{L+1}\right]$ [see Appendix A]	(2)
SCAN	$S_{min} + \frac{S_{max} - S_{min}}{L' + 1}$ [Denning]	(3)
N-step	$S_{min} + \frac{S_{max} - S_{min}}{L + 1}$ with $2 \leqq L \leqq N$. For $L = 1$ N-step scan reduces to the FCFS policy.	(4)

- No control unit or channel delays are encountered.
- No distinction is made between READ and WRITE requests, and the overhead to decide which request to service next is assumed to have no effect on the efficiency of the scheduling policy.

Table I summarizes the differences in expected seek time for four of the scheduling policies. The expressions are only approximations and do not attempt to be exact descriptions. Expected seek time T_{sk} is computed conditioned on the next request not being at the current cylinder, i.e. a seek is required.

List of Variables

- c total number of cylinders.
- L mean queue length (includes request currently being serviced).
- L' mean number of requests serviced by SCAN in one sweep across the disk surface, given a mean queue length L (see Appendix B).
- λ input rate in requests/second.
- m number of sectors or records per track.
- N upper bound on group sizes for N-step scan.
- P probability that the next request to be serviced is not on the current cylinder [see eq. 5].
- ρ disk utilization ($\rho = \lambda T_s$).
- σ_W^2 variance of waiting time for individual requests.
- S_{max} seek time for distance of $c - 1$ cylinders.
- S_{min} seek time for distance of 1 cylinder.
- T disk rotation time.
- T_s expected service time between two consecutive requests in the queue.
- T_{sk} expected seek time.
- W expected waiting time for individual requests from time of arrival into the queue until completion of service.

All four policies listed in Table I service requests FCFS within the same cylinder, so a single expression for T_s can be derived in terms of the individual T_{sk} expressions. Expected service time for requests that require a seek is given by the well-known equation $T_s = T_{sk} + T/2 + T/m$. Expected service time for requests on the same cylinder as the current request is found by first noting that each cylinder has t tracks and total of mt sectors. If the current request is on sector i, the probability that sector i will be chosen again is $(t-1)/(mt-1)$. The probability that a specific sector, $j \neq i$, will be chosen is

$$[1/(m-1)][1-(t-1)/(mt-1)].$$

The expected number of sectors traversed to service the next request is

$$\sum_{k=0}^{m-1} k \text{ Prob } \{\text{Sector } k \text{ is next}\} + 1$$
$$= \sum_{k=0}^{m-2} k \frac{1}{m-1}\left(1 - \frac{t-1}{mt-1}\right) + (m-1)\frac{t-1}{mt-1} + 1$$
$$= \frac{(mt-2)(m-1)}{2(mt-1)} + 1.$$

Expected service time = T/m (expected number of sectors traversed). Thus we have

$$T_s = P\left(T_{sk} + \frac{T}{2} + \frac{T}{m}\right) + (1-P)\frac{T}{m}\left[\frac{(mt-2)(m-1)}{2(mt-1)} + 1\right] \quad (5)$$

where $P = 1/c$ for FCFS, $P = [(c-1)/c]^L$ for N-step scan, and $P = [(c-1)/c]^{L'}$ for SSTF and SCAN. We

do not require an expression for the Eschenbach scheme because it is useful only under heavy loading conditions, for which the service rate is approximately equal to the input rate.

The equation for expected waiting time, $W = L/\lambda$, was proved by Little [7] to apply to all scheduling policies with a stationary input rate λ, without regard to arrival distributions or service times. For constant T_s we could approximate λ by

$$\rho/T_s \cong [L+1-(L^2+1)^{1/2}]/T_s .$$

The major limitation on these measures is that they are functions of the queue length L and not the input rate λ. To compare two policies with the same queue length is not to compare them under equal loading conditions. For equal loading conditions two algorithms may operate at equilibrium with two very different queue lengths. Given an input rate we can compute L if we know T_s, but T_s cannot be computed unless we know the mean distance between requests, which depends on L. Such a problem can be studied in two ways. One is an iterative process, suggested by Pinkerton [9], that results in convergence to a value of L relating most closely to a given λ for the particular measure being studied. The second technique, simulation, is used here because the necessary output data is already available from the other experiments.

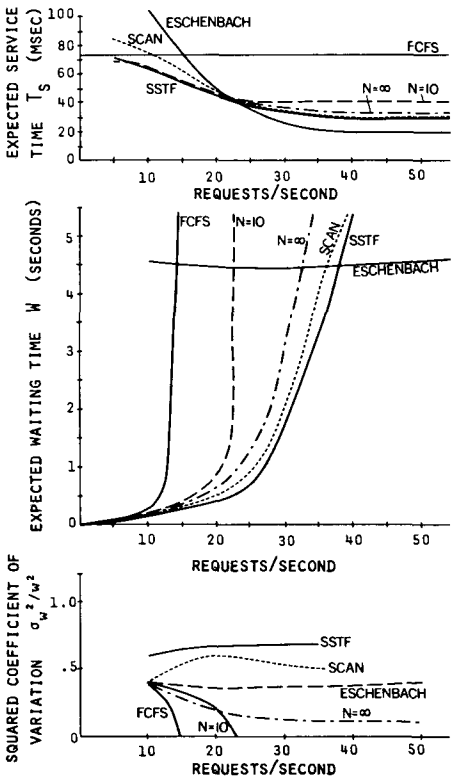
Simulation Results

Each of the disk scheduling policies was modeled in SIMULA [2] on the UNIVAC 1108 in order to verify our analytical results and to study the relations among the policies for various input rates and queue lengths.

Arriving request times were generated from a Poisson distribution. Seek times were computed from a linear approximation of the IBM 2314 seek time characteristic [6] (see Figure 1). Other hardware characteristics, $c = 200$ cylinders, $T = 25$ msec, were consistent with the IBM 2314. Variable parameters included the mean input rate λ, record size, and modifications to the standard algorithms such as rotational optimization. Standard simulation techniques were used to obtain stable output results, including mean queue length, mean service time, and mean and variance of waiting time.

Figures 2 and 3 depict three measures of performance: T_s, W, and the squared coefficient of variation for waiting time. T_s is a measure of system (disk queuer subsystem) performance and is equivalent to the reciprocal of system throughput. W is a measure of individual service, such as for a time-sharing terminal user or for a page request in a demand paging environment. The squared coefficient of variation σ_W^2/W^2 is a dimensionless ratio which is useful for relative comparisons of variances and mean values. It only has a useful meaning when accompanied by the actual value of the mean.

Fig. 2. Performance of disk scheduling policies for m = 50 records/track.

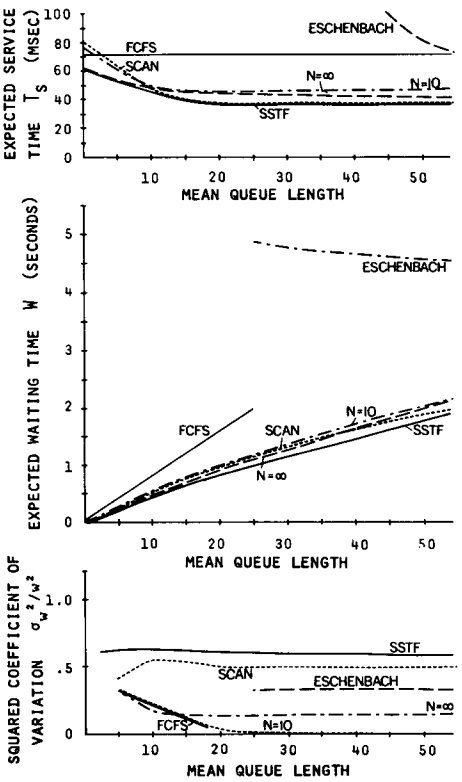


In Figures 2 and 3 these measures of performance are shown for different input rates and queue lengths. The value $m = 50$ records per track was chosen to decrease the effect of data transmission and increase the effect of seek time on total service time. Results for $m = 4$ are compared with those for $m = 50$ in the next section.

These results indicate that the FCFS policy is good at low input rates, but it reaches saturation quickly (at different points depending on record size). Sensitivity to this low saturation point makes FCFS undesirable under most real operating conditions.

SCAN and SSTF provide the greatest throughput and least mean waiting time at the expense of higher variance, or discrimination. SCAN and SSTF are nearly identical for T_s and W, and differ only in variance. For this reason we can easily make the choice of SCAN over SSTF at all but the lowest input rates. At low input, SCAN is inefficient because of the necessity to sweep entirely across the disk surface regardless of the size of the queue. The most practical way to implement SCAN on a real system would be to make a trivial modification suggested by Merten [8], called LOOK.

Fig. 3. Performance of disk scheduling policies for m = 50 records/track.



LOOK allows the disk arm to change direction when there are no further requests to service in the current direction. At low utilization it is as good as or better than the other policies, and at high utilization its performance is nearly equivalent to that of SCAN.

The N-step scan is interesting because it offers reasonable performance in throughput and mean waiting time and has a lower variance than either SSTF or SCAN. However, it is difficult at this point to say whether the lower variance of N-step scan offsets the higher T_s and W. The simulations indicate that much could be gained by not imposing a limit N. We let the upper bound on N be ∞, so that whenever the server is ready to chain a new group of requests, it chooses all the requests in the queue at that point. We will henceforth refer to the N-step scan in terms of the $N = \infty$ case.

The Eschenbach scheme performs well for $m = 50$ under heavy loading conditions. This is due mainly to its rotational optimization feature.

As stated above, one purpose of the simulation was to verify the analytical approximations given in Table I. The results corresponded very closely with two exceptions. With a small queue, the expression for SCAN (3) does not take into account the inefficiency of going to the extremes of the disk regardless of the locations of requests. With a large queue, the N-step approximation appeared to fail because seek time was no longer the dominant factor in total service time.

Criterion for Overall Performance

Taken individually, none of the previously mentioned measures of performance would be adequate to select the best disk scheduling policy for a given situation. If we assume the existence of a utility function that allows us to adjust the relative importance of T_s, W, and σ_W, we will have a measure with which to differentiate overall performance among scheduling policies. One such function could be chosen as follows.

$$\theta_i = T_s{}^a(bW + c\sigma_W) \tag{6}$$

where θ_i, T_s, W, and σ_W are functions of the ith scheduling policy and a given mean input rate λ. a, b, and c are adjustable weights. For the function θ_i no single measure of performance dominates, and each measure can be weighted independently. A linear combination of all three measures is not meaningful because T_s is in milliseconds and both W and σ_W are in seconds. However, we use a linear combination of W and σ_W because the units are the same and the simulations show that their magnitudes are comparable. Taking the product of all three was rejected because it would degenerate for $\sigma_W = 0$. By varying a, b, and c in (6) we can modify the relative importance of any measure.

Our problem is to find the scheduling policy that minimizes over the family of functions θ_i for mean input rates within some specific range:

$$\text{Find } \theta = \text{minimum } [\theta_1, \theta_2, \cdots, \theta_n] \quad \text{subject to } \lambda_1 \leqq \lambda \leqq \lambda_2 \tag{7}$$

where n is the number of policies being considered. We use $\lambda_1 = 0$ and $\lambda_2 = 50$. We will see that θ is an envelope under the family θ_i and cannot be solved by a unique policy for the entire range of λ. We will then see if modifications can be made to these policies so that a unique optimal policy can be designed.

The application of (6) to the simulation results for $m = 4$ sectors/track yields the relationships shown in Figures 4 and 5. Figure 6 shows the $m = 50$ case. Values used for the weights are shown on the graphs. Figures 4 and 5 show that the relative performances of the policies are rather insensitive to nominal changes in the weights. This indicates that (6) is stable as a performance measure and not subject to discontinuities.

An Algorithm for Overall Performance

A study of the Eschenbach scheme versus SCAN with rotational optimization reveals that the superiority of Eschenbach at heavy loading ($m=50$) is due to the circular action of the disk arm from cylinders 1, 2, $\cdots$, $c-1$, c and back to 1. In this way the maximum wait, assuming that no sector queues occur, is one complete cycle of c possible stops; the minimum wait is essentially zero. For SCAN the maximum wait is also a complete cycle, but the shuttle action may result in $2c-2$ stops ($c=200$ in our examples). Clearly SCAN has a much greater variance in waiting time and only a slight improvement in the mean W. Consequently a modification to SCAN can be designed so that the disk arm will sweep in a circular pattern to take advantage of this property [10]. This new policy will be referred to as C-SCAN, and its real implementation will be called C-LOOK.

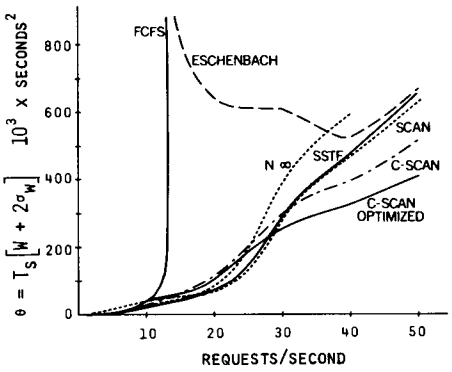
Two simulations were constructed, one with the circular sweep and no other change, and one with both the circular sweep and rotational optimization. The results of these simulations are displayed on Figures 4–6. They indicate that in both cases C-SCAN has mediocre performance at low utilization, but it always becomes the most superior policy of its class (rotational policy used) at high utilization.

A similar test was attempted with the N-step scan, but it showed very little sensitivity to these changes (see Figure 6). In the N-step scan the variance is fixed by the automatic grouping of requests and is insensitive to shuttle or circular action. Also, as expected, the rotational optimization does not improve N-step as much as it does SCAN. The N-step algorithm does not allow new requests to be serviced as they pass under the read/write heads because they are not in the prespecified group, and thus it cannot take advantage of this prospect for increased efficiency.

Our conclusion is that the best overall disk scheduling policy should be a two-part modification of the SCAN access method. At low utilization, use the LOOK policy; at high utilization, use the C-LOOK policy. The decision to implement rotational optimization or not depends on the frequency of heavy loading, constraints of a particular system, and whether or not the cost of additional software development is worthwhile. Rotational optimization could be implemented independently of whatever scheduling policy was used.

A two-part (or two-policy) algorithm may not be necessary if the system load is almost always in some specific limited range. In this case one should choose either LOOK or C-LOOK, depending on the range of load to be handled.

Fig. 4. Overall performance, m = 4 records/track.



Implementation

To consider how to implement a two-policy algorithm, we first examine Figure 4. The crossover point in total performance between SCAN (or LOOK) and C-SCAN (or C-LOOK) occurs at $\lambda = 30$ requests/second. However, the performances for these policies are almost indistinguishable in the interval of 28–32 requests/second. In this interval either policy could be used, so instead of defining a single switching point between the two policies, we define two switching points at the ends of the interval. They can be specified by the equilibrium queue lengths of C-LOOK and LOOK at the lower end (L_1) and the upper end (L_2), respectively. By applying the simulation techniques used here, one could determine a satisfactory pair L_1, L_2 to handle his own particular input characteristics.

To illustrate the concept of two switching points, we define the following operation regions (see Figure 7).

Region	Range of input rates	Scheduling policy used	Crossover to the other policy
1	0–32 requests/sec.	LOOK	$L_2 \simeq 140$ (queue length of LOOK at $\lambda = 32$)
2	28–50 requests/sec.	C-LOOK	$L_1 \simeq 100$ (queue length of C-LOOK at $\lambda = 28$)

The LOOK policy should be used initially. When the counter of queue length exceeds L_2, C-LOOK should be used. If the queue length should then drop below L_1, the LOOK policy should be resumed.

A two-policy algorithm can be implemented quite

Fig. 5. Overall performance, m = 4 records/track.

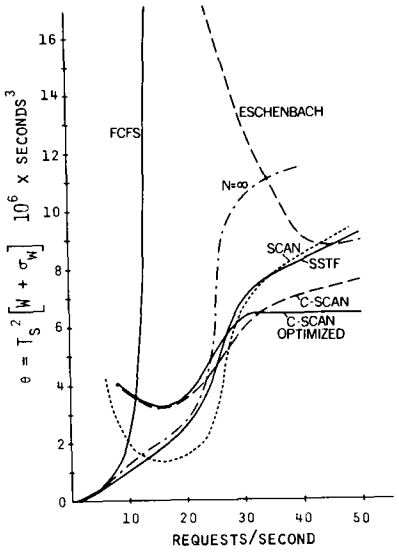


Fig. 6. Overall performance, m = 50 records/track.

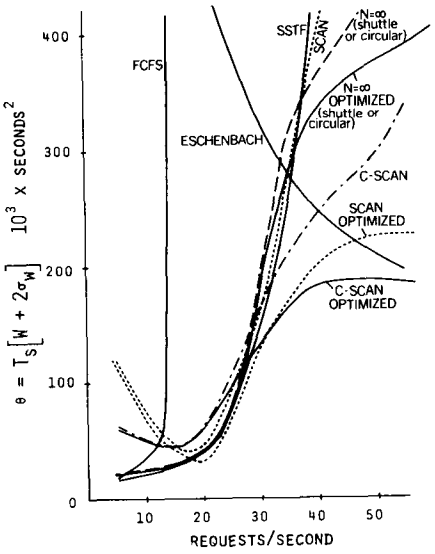
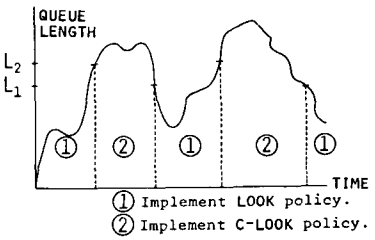


Fig. 7. Concept of two switching points.



simply: Locate the request queue in main memory. Rank I/O requests by cylinder number, sector, and track, respectively; insert arriving requests in a doubly linked circular list. A 2-bit status code is required to differentiate among LOOK (sweep forward), LOOK (sweep backward), and C-LOOK. Also, the current queue length and values for L_1 and L_2 must be readily available. Such a design could be realized in a software subsystem, a peripheral processor, or an all-hardware disk queuer.

Conclusion

We defined total performance in terms of a utility function which combines system throughput with mean and variance of waiting time for individual requests. Cost of implementation was also considered. Based on these factors we found that under light loading conditions the maximum performance disk scheduling policy was LOOK, a variation of the SCAN access method. Under heavy loading conditions C-LOOK, combining the best characteristics of LOOK and the Eschenbach scheme, had maximum performance.

Acknowledgments. The authors would like to express appreciation to Ron Otto for his helpful suggestions. The referees also provided several helpful comments.

Appendix A

SSTF Expected Minimum Seek Time

The expected minimum seek time for the SSTF policy was derived by Denning [3]. His result, assuming a seek was required, was stated as

$$T_{sk} = \frac{1}{2} + S_{min} + \frac{S_{max} - S_{min}}{2(L+1)}\left[1 + \frac{1}{L+2}\left(\frac{c}{c-1}\right)^{L+1}\right] \quad \text{(A.1)}$$

where T_{sk}, S_{min}, S_{max}, L, and c are defined in Table I. This equation appears to be in error, as pointed out by Merten [8], because the number ½ was unchanged in the transformation from distance units to time units, and thus it is unitless in (A.1). Merten's correction was based on an interpretation that the ½ term should be separate from the other term in the distance equation. A more reasonable interpretation is that the distance expression contains exactly one term, which transforms in the following manner to a time expression. Let δ represent Denning's distance expression:

$$\delta = \frac{1}{2} + \frac{c-1}{2}\beta \quad \text{where}$$

$$\beta = \frac{1}{L+1}\left[1 + \frac{1}{L+2}\left(\frac{c}{c-1}\right)^{L+1}\right]. \quad \text{(A.2)}$$

The time required to move the disk arm a positive distance δ, conditioned that the disk arm moves, is

$$T_{sk} = \left(S_{min} - \frac{S_{max} - S_{min}}{c-2}\right) + \frac{S_{max} - S_{min}}{c-2}\delta \quad \text{(A.3)}$$

See Figure 1 for interpretation of this linear approximation of seek time. By substituting (A.2) into (A.3) and using simple algebra, we obtain

$$T_{sk} = S_{min} + (S_{max} - S_{min})\left[\frac{(c-1)\beta - 1}{2(c-2)}\right]. \quad \text{(A.4)}$$

Equation (A.4) satisfies both the physical concept and the dimensional analysis.

Appendix B

Number of Requests Serviced by SCAN

The SCAN policy defines L' as the sum of the mean queue length L (the initial set of requests to be handled in one scan) plus all new requests that fall in the path of the disk arm in one full sweep of the c cylinders. If we assume that the distribution of requests yet to be serviced is uniform, then L' can be computed in terms of a recurrence formula for the expected number of requests remaining to be serviced. Let the disk arm proceed from left (cylinder 0) to right (cylinder $c-1$). Let $DR_i = DR_{i-1} - DR_{i-1}/(NR_{i-1}+1)$ denote the expected number of cylinders from the disk arm position to cylinder $c-1$ after i requests have serviced. Let $NR_i = NR_{i-1} - 1 + qDR_i/DR_0$ denote the expected number of requests currently to the right of the disk arm after i requests have been serviced. q = expected input rate/expected service rate, or the expected number of requests arriving in the ith service interval. The boundary conditions are $NR_0 = L$ and $DR_0 = c - 1$.

To derive L' we first substitute for DR_i in the equation for NR_i to obtain

$$NR_i = NR_{i-1} - 1 + q\frac{DR_{i-1}}{DR_0}\left(\frac{NR_{i-1}}{NR_{i-1}+1}\right).$$

By substituting recursively for DR_{i-1}, and then for NR_{i-1}, we reach

$$NR_i = NR_0 - 1 + q\sum_{k=0}^{i-1}\prod_{j=0}^{k}\frac{NR_j}{NR_j+1}. \quad \text{(B.1)}$$

Equation (B.1) is a function only of L and q. In particular it is independent of c. If we stop the recurrence computation when $NR_i \leqq 0.5$, we obtain

$$L' = i = \left[L - 0.5 + q\sum_{k=0}^{i-1}\prod_{j=0}^{k}\frac{NR_j}{NR_j+1}\right] \quad \text{(B.2)}$$

smallest integer greater than.

For practical values of q, i.e. $0.8 \leqq q \leqq 1$, we obtain $1.53 \leqq L'/L \leqq 1.72$.

References

1. Abate, J., and Dubner, H. Optimizing the performance of a drum-like storage. *IEEE Trans. Comput. C-18*, 11 (Nov. 1969), 992–997.
2. Dahl, O.J., and Nygaard, K. SIMULA—A language for programming and description of discrete event systems: Introduction and users' manual. Norwegian Computing Center, Oslo, 1966.
3. Denning, P.J. Effects of scheduling on file memory operations. Proc. AFIPS 1967 SJCC, Vol. 30, AFIPS Press, Montvale, N.J., pp. 9–21.
4. Feller, W. *An Introduction to Probability Theory and Its Applications*, Vol. 1, 3rd ed. Wiley, New York, 1968.
5. Frank, H. Analysis and optimization of disk storage devices for time-sharing systems. *J. ACM 16*, 4 (Oct. 1969), 602–620.
6. Introduction to IBM System/360 direct access storage devices and organization methods, C20-1649-4, IBM.
7. Little, J.D.C. A proof for the queuing formula: $L = \lambda W$. *Oper. Res. 9*, 3 (1961), 383–387.
8. Merten, A.G. Some quantitative techniques for file organization. Ph.D. Th., Tech. Rep. No. 15, U. of Wisconsin Comput. Center, 1970.
9. Pinkerton, T.B. Program behavior and control in virtual storage computer systems. CONCOMP Proj. Rep. No. 4, Ph.D. Th., U. of Michigan, Apr. 1968.
10. Seaman, P.H., Lind, R.A., and Wilson, T.L. An analysis of auxiliary-storage activity. *IBM Syst. J. 5*, 3 (1966), 158–170.
11. Weingarten, A. The analytical design of real-time disk systems. Proc. IFIP Congr. 1968, North Holland Pub. Co., Amsterdam, pp. D131–D137.