Efficient Fair Disk Schedulers

Tulika Pradhan*
Hughes Software Systems
International Trade Tower
New Delhi 110019, India
tulika@hss.hns.com

Jayant R. Haritsa
Dept. of Computer Science and Automation
Indian Institute of Science
Bangalore 560012, India
haritsa@csa.iisc.ernet.in

Abstract

Time-shared systems are usually expected to provide fair service to their users. We recently investigated this issue in the context of **disk** schedulers [HP95] and showed that classical disk scheduling algorithms favor some users over others and therefore fail to provide adequate fairness. We then developed new disk scheduling algorithms that consistently provide fair service. A drawback of these fair algorithms is that they achieved their fairness at the cost of a large increase in user response times. We address this problem in this paper and present an improved set of fair disk schedulers that have significantly better response time performance.

1 Introduction

There is an increasing demand on computer systems to provide service guarantees to end-users with regard to a variety of metrics. Real-time applications, for example, may require meeting of task deadlines to be guaranteed. Time-shared systems, on the other hand, are usually expected to provide **fair service**. While processor scheduling has been extensively investigated in this context, the issue of providing fairness to disk users, although considered, has not been analyzed to a similar extent. In a recent paper [HP95], we studied this issue in detail and showed that classical disk scheduling algorithms favor some users over others and therefore fail to provide adequate fairness. We then developed new disk scheduling algorithms that consistently provide fair service. A drawback, however, of these fair algorithms is that they achieved their fairness at the cost of a large increase in user response times as compared to the classical schedulers. In this paper, we address this problem and present an improved set of fair disk schedulers that have significantly better response time performance.

Fairness

The term "fairness" is used in several different senses in the performance evaluation literature. The interpretation used in our earlier study, and in this paper, is that a fair resource divides its processing capacity *equally* among all its current users, that is, it provides "service fairness". Service fairness may be achieved in two different ways: "instantaneous" and "statistical". In instantaneous fairness, a resource provides fair service at every instant of operation. Statistical fairness, on the other hand, is different in that the resource provides fair service to a group of users when each of them makes a large number of service requests, and the fairness is evaluated over the entire set of requests.

Disk Scheduling

In disk systems, it is *inherently* impossible to achieve instantaneous fairness. This limitation arises due to two factors: First, the context-switch time in disks, that is, the time taken between the completion

*This work was done while the author was with the Dept. of Computer Science and Automation, Indian Institute of Science.

of one request and the start of service of the next request, is not constant. Instead, it is a function of the seek distance between the physical disk locations accessed by the requests. Second, the service process in disks is non-preemptive, that is, once the transfer of data has been initiated for a request, other requests cannot be served until the data transfer is completed. Hence, we can only hope to achieve statistical service fairness in disk drives; that is, the average response time for each concurrent user should be approximately the same when evaluated over a large number of disk accesses.

Performance Metric

There is a large body of literature related to the performance of disk schedulers (e.g. [Fran69, TP72, Oney75, Swar80, GRP87, WGP94, IEEE94]). The metric used to measure fairness in these studies has typically been either the variance or the coefficient of variation of user response times. However, as discussed above, an inherent feature of disk service is that there is significant variation in response times. In this situation, evaluating fairness through the overall variance is perhaps an unsuitable metric. We therefore proposed a different fairness metric in our previous study, based on the following observation: Most operating systems tend to *cluster* the files belonging to a particular user, and therefore, each user's disk requests typically tend to be restricted to a few adjacent cylinders. Given this workload pattern, a better measure of fairness may be to measure the variance of the *cylinder* mean response times. This means that we consider a disk system to be fair if users have approximately the same average response time *independent* of where on the disk their files happen to be clustered. This concept is shown pictorially in Figure 1 for a disk with N cylinders. Accordingly, we used the coefficient of variation of the cylinder mean response times as the fairness metric in our earlier study – the same metric is used in this study also.

Previous Results

In our previous study [HP95], we conducted a detailed simulation-based evaluation of the fairness (as defined above) of schedulers commonly used in commercial disk drives (e.g. SCAN, SSTF, etc.). The results of our experiments showed that these schedulers generally exhibit poor fairness characteristics. In particular, they usually provide better service to users whose files are stored in the neighborhood of the central cylinders, as compared to users whose files are located near the extreme cylinders.

To address the above problem, we developed a new suite of disk schedulers that consistently provided high degrees of fairness. These algorithms are variants of the classical disk schedulers with the following principal difference: In the classical algorithms, when no further requests exist in the queue, the disk head remains stopped at the cylinder of the last request served. In contrast, the new algorithms *position* the disk head, whenever the disk becomes idle, in a manner that results in fairness being achieved.

A drawback of the above fair schedulers is that they achieved fairness at the expense of a large increase in user response times. We address this issue in this paper and present a new set of fair disk schedulers that have significantly improved response time performance.

2 Disk Scheduling Algorithms

In our previous study [HP95], we had considered a range of "scheduling granularities". The scheduling granularity limits the set of time instants at which scheduling decisions can be made, and is determined by the disk controller implementation. For example, in the DYNAMIC scheduling granularity, a new scheduling decision can be made at any time. That is, even during the process of moving towards a selected request, the scheduler can change its decision about which task to serve next based on the arrivals of new requests. In contrast, for the STOP scheduling granularity, decisions can be made only after the disk head has finished serving the currently scheduled request. Apart from STOP, we also considered GATE, where scheduling is done on a group basis. For each of these granularities, a fair algorithm was identified in our previous study.

In this study also, we have considered all the above scheduling granularities. However, due to space constraints, we limit our attention in the rest of this paper to only the DYNAMIC scheduling granularity. The results for other granularities are available in [Prad95].

In this section, we present the salient features of the disk scheduling algorithms evaluated in our experiments. We first describe the classical scheduling algorithms and then the fair algorithms.

2.1 Classical Algorithms

The performance of a variety of disk scheduling algorithms has been evaluated in the literature. The main algorithms include FCFS, SSTF [Denn67], POLL [Denn67] and LOOK [Mert70]. Modified versions of these basic algorithms such as Circular POLL [SLW66] and Circular LOOK [TP72] have also been studied.

The **FCFS** (First Come First Serve) scheduler is a simple algorithm which serves requests in the sequence of their arrival to the disk system. In contrast, the **SSTF** (Shortest Seek Time First) scheduler serves the request that is nearest to the current disk head position. Its goal is to reduce the mean response time by minimizing the seek distance traversed in servicing requests. At high arrival rates, SSTF tends to become localized in a small set of cylinders in an attempt to exhaust all requests to that region, thereby greatly increasing the response time for requests at other cylinders. The **POLL** scheduler sweeps the disk head, back and forth, from one end of the disk to the other, serving requests as it passes by each cylinder. POLL usually exhibits higher response times for extreme cylinders as compared to the centrally-located cylinders. This is because the extreme cylinders see a sequence of alternate short and long gaps in the inter-visit times of the disk head as compared to the inter-visit times of the central cylinders where it is almost periodic. The **LOOK** scheduler implements a slight modification of the POLL algorithm. Instead of reversing the direction of the head movement at the disk extremities, it reverses the direction of the head movement whenever no further requests exist in the current sweep direction.

The **CPOLL** (Circular POLL) scheduler is a "unidirectional" variant of POLL. The basic feature here is that the disk head serves requests in only one sweep direction, say, inside-out. In the opposite direction, CPOLL strokes directly to the innermost cylinder before commencing the sweep again. The **CLOOK** (Circular LOOK) scheduler is a similar variant of LOOK wherein the disk head, on the return path, strokes directly to the innermost cylinder having a request.

2.2 Fair Algorithms

In our previous study [HP95], we had developed, for each scheduling granularity, a new algorithm which exhibits consistent fairness over the entire range of system loading. The new algorithms achieved their fairness by appropriately positioning the disk head, whenever the disk becomes idle, such that the inherent variation in disk response times is evenly distributed across all cylinders. In particular, for the DYNAMIC scheduling granularity, an algorithm called **NSCP** (Non-Stop Circular Poll), provided fair performance. The NSCP scheduler is identical to CPOLL except that the disk head continuously sweeps the disk surface *even* when the system is empty of requests. However, the response time performance of NSCP was considerably worse than that of the classical algorithms, especially at low arrival rates of disk requests.

We present a new fair algorithm here called **PCLOOK** (Park CLOOK) whose response time performance is significantly better than that of NSCP. The PCLOOK scheduler is similar to CLOOK except for the following difference: Whenever the system is emptied of requests, the disk head is *parked* at the nearest disk extremity. If the head is parked at the inner end, the service direction for future requests is inside–out. However, if the head is parked at the outer end, the service direction for future requests is outside–in. Therefore, although the service is *always* unidirectional, the direction of the sweep may change dynamically.

3 Performance Model

A simulation model of a disk system was used to study the fairness characteristics of the various disk scheduling algorithms described in the previous section. The disk head seek times are computed using the common linear approximation formula [TP72]

$$T_s = \alpha + \beta \times n_c \quad (n_c \geq 1) \tag{1}$$

where n_c is the seek distance measured in number of cylinders; α represents the startup time for the disk head to start seeking; β is the seek-distance multiplier. The values of α and β are constant for a specific disk and are set to 30.0 and 0.5 milliseconds, respectively, in our experiments. These values are the same as those used in [TP72]. The rotational latency and the transfer time are assumed to be the same for all requests and is set equal to 8.0 milliseconds. The time taken for the disk controller to make a scheduling decision is assumed to be negligible.

User requests for disk access arrive in a Poisson stream and the data requests are uniformly distributed over the entire disk. The fairness metric is computed as the coefficient of variation of the cylinder mean response times, that is,

$$Fairness = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(RT_i - ORT)^2}}{ORT} \tag{2}$$

where N is the number of cylinders, RT_i is the average response time for cylinder i and ORT is the overall average response time. In this formulation, a scheduler with a fairness value of 0.0 is completely fair. As a practical matter, we consider fairness values that are <u>less than 0.05</u> to be satisfactorily fair.

4 Results

The performance of all the algorithms described in Section 2 was evaluated in our simulation study. However, to ensure graph clarity and to meet space constraints, the performance of only a representative set of algorithms are shown here. This set includes the FCFS, SSTF, LOOK and CLOOK classical schedulers, apart from the fair algorithms, NSCP and PCLOOK. The results of the experiments are shown in Figures 3a through 3h: the first four graphs show the cylinder response time behavior for disk request arrival rates ranging from light load to heavy load; the next three graphs show the global mean response time, the global coefficient of variation, and the cylinder fairness, respectively, as a function of arrival rate; the last graph characterizes the tradeoff between fairness and mean response time.

From these figures, we observe first that the response time performance of FCFS saturates much earlier than that of the other algorithms (Figure 3e). Also, it is quite unfair under light loads, favoring the central cylinders over the extreme cylinders (Figures 3a, 3b). This is because at light loads, new requests typically arrive to an empty system. The disk head will be stopped at the cylinder of the last request served and since the requests are uniformly distributed, the head position is also uniformly distributed. Therefore, new requests arriving at the central cylinders are, on the average, only $N/4$ cylinders away from the current disk head position, whereas requests at the extreme cylinders are, on the average, $N/2$ cylinders away from the current disk head position (where N is the total number of disk cylinders).

Moving on to SSTF, we observe that although its response time is significantly better than that of the other algorithms, its fairness characteristics are extremely poor. This is especially so at higher arrival rates where the central cylinders receive significantly better treatment than the extreme cylinders. The unfairness of SSTF is due to its "greedy" approach which tends to get the disk head localized in the central cylinders.

Turning our attention to LOOK, we observe that its fairness performance is almost as poor as that of SSTF. The unfairness of CLOOK stems from the fact that extreme cylinders see a sequence of

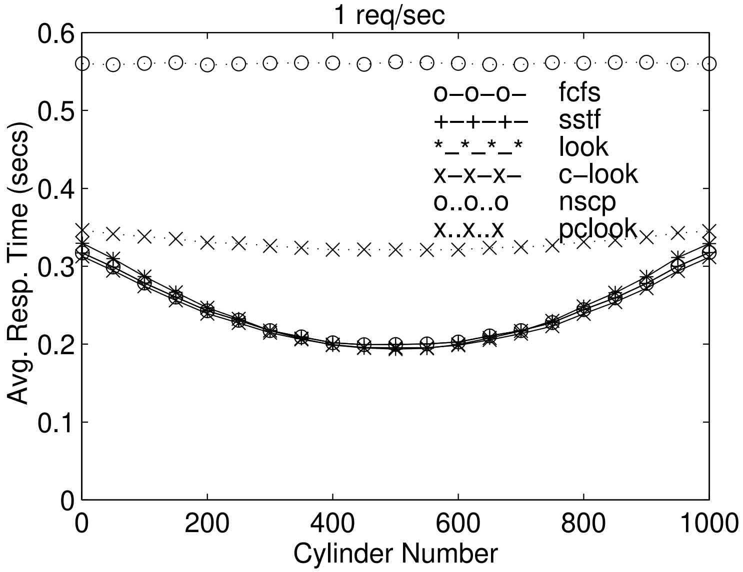

Fig. 3a

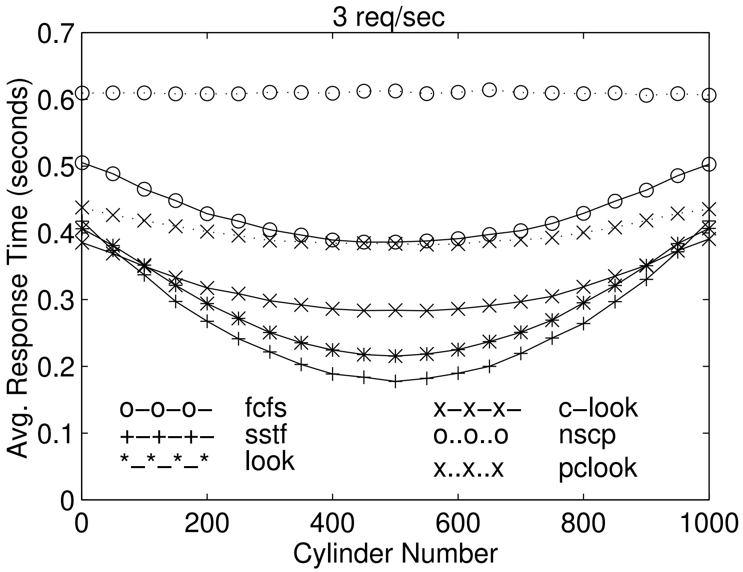

Fig. 3b

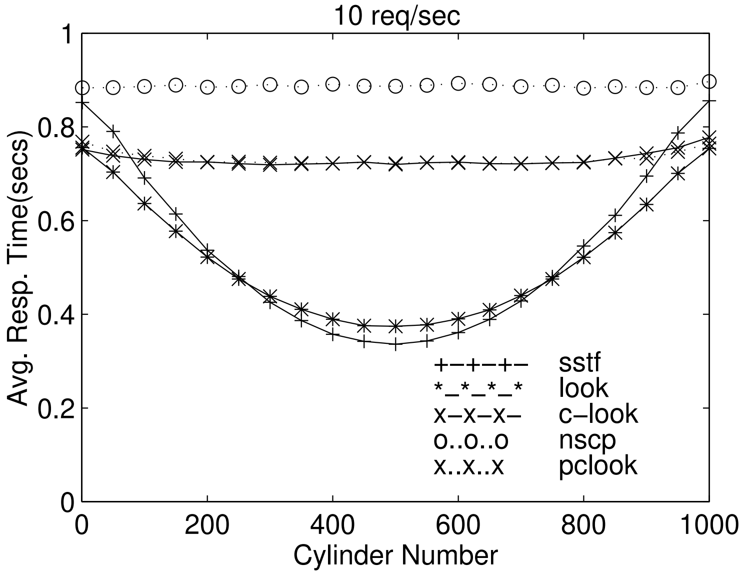

Fig. 3c

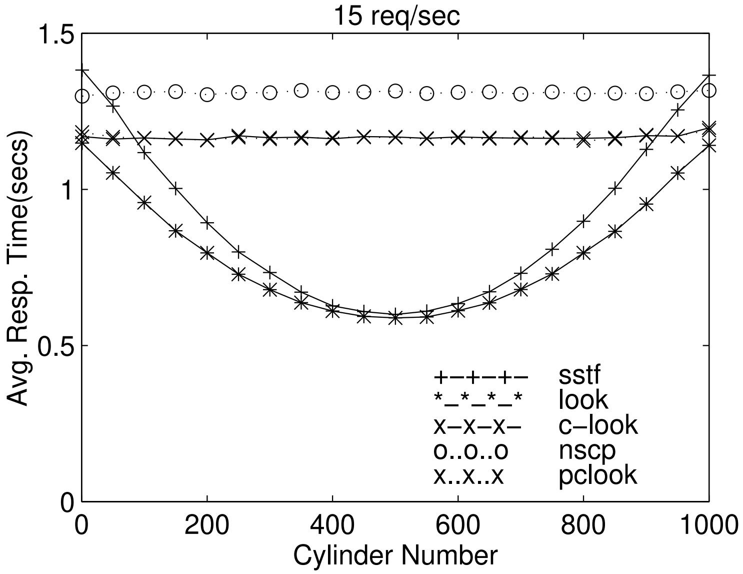

Fig. 3d

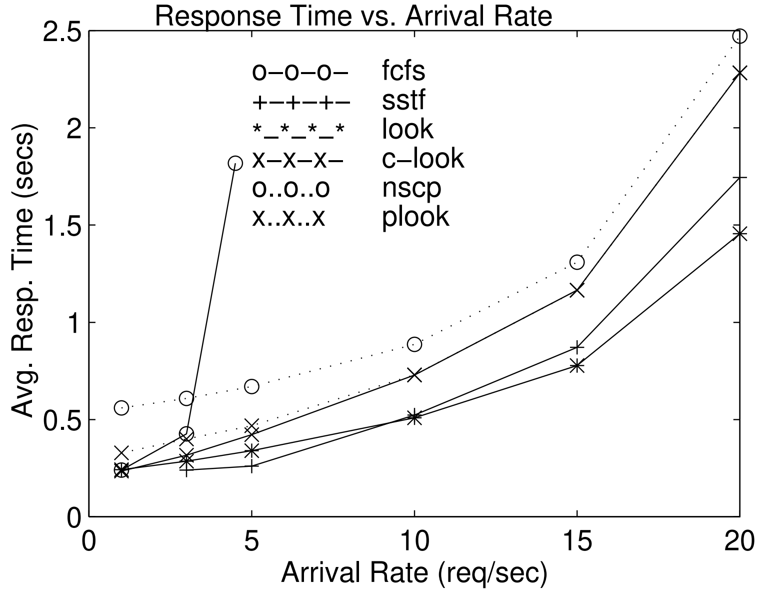

Fig. 3e

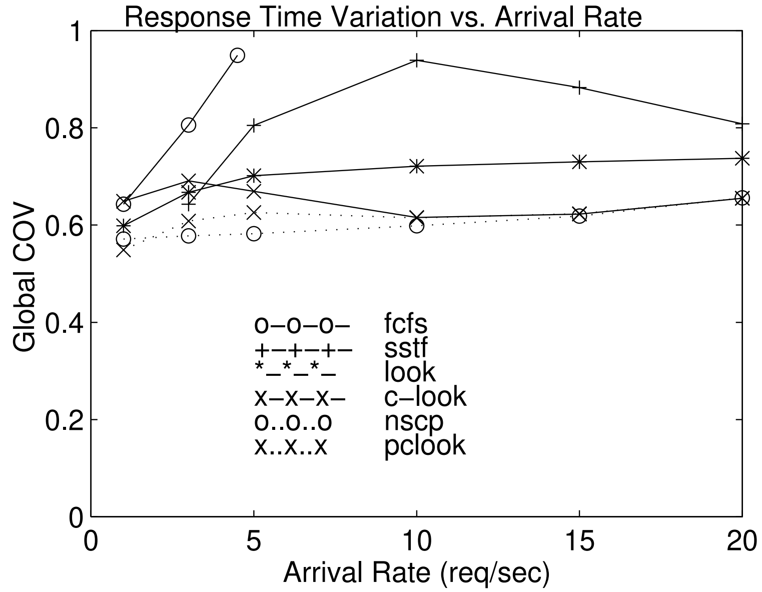

Fig. 3f

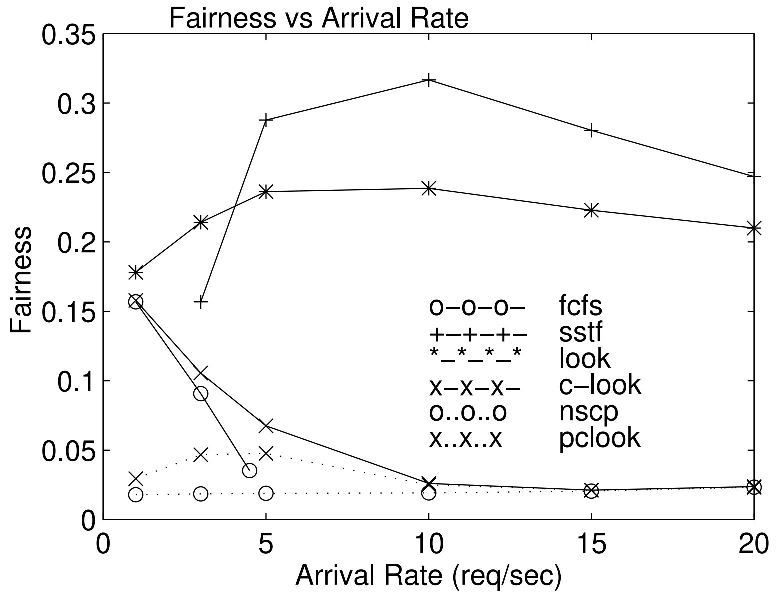

Fig. 3g

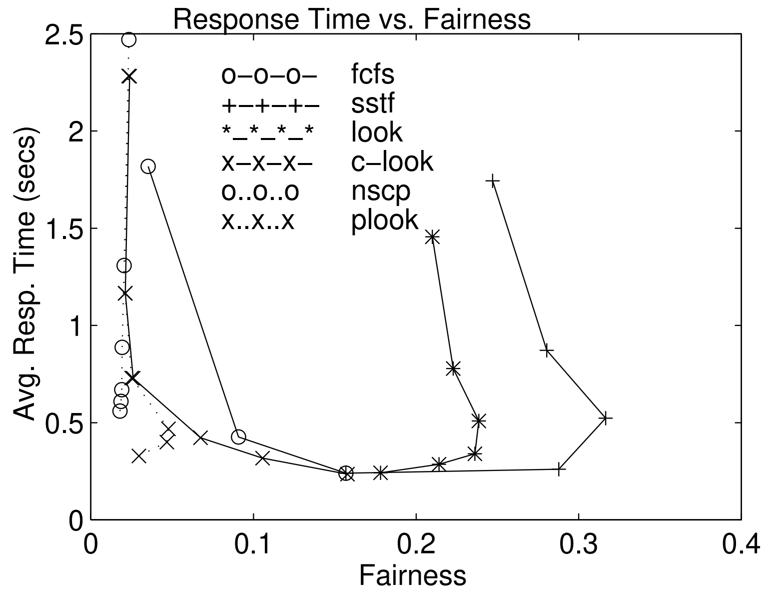

Fig. 3h

alternate short and long gaps in the inter-visit times of the disk head as compared to the inter-visit times of the central cylinders where it is almost periodic. This results in long queues developing at the extreme cylinders and relatively smaller queues at the central cylinders.

For the CLOOK scheduler, we observe that it is as unfair as LOOK at light loads. This is to be expected because, at light loads, CLOOK and LOOK are virtually identical since most of the sweeps serve only a single task. At heavy loads, however, CLOOK is quite fair due to its unidirectional service pattern.

In contrast to the above classical schedulers, the NSCP scheduler is extremely fair across the entire loading range (Figure 3j). The reason for its good performance is the following: Although the CPOLL scheduler performance is not shown in the graphs, it exhibited a marked bias against the inner cylinders at light loads in our experiments. This is because requests for the inner cylinders typically have to wait for the disk head to move to the outer end of the disk and then stroke back to the inner end before commencing the sweep. However, requests for the outer cylinders will usually be served directly since the disk head will tend to be located behind them. The NSCP scheduler, although closely resembling CPOLL, eliminates this bias as follows: Since the disk head is always moving, the bias against the inner cylinders due to always having to wait for the full stroke return time is absent as the disk head may have completed some or all of the return stroke at the time of arrival of the new request.

Finally, considering the new fair algorithm, PCLOOK, we observe first that its response time performance is significantly better than that of NSCP throughout the loading range and especially so at light loads. The reason for PCLOOK being better than NSCP is that PCLOOK is based on CLOOK while NSCP is based on CPOLL, and the basic response time performance of CLOOK is better than that of CPOLL due to reduced motion of the disk head. With regard to fairness, we observe that although PCLOOK is not as good as NSCP (Figure 3h) at light loads, yet it is always satisfactory in that its fairness level is below 0.05 over the entire loading range.

The reason that PCLOOK has slightly worse fairness than NSCP at light loads is the following: If, after the service of a request, there are no further requests in the system, the head parks at one of the extremities. In this case, the average seek distance for all cylinders is $N/2$. On the other hand if, after the service of a request, there is a request pending (at low arrival rates, this will typically be the only request in the queue), the disk head moves straight to serve this request. In this case, the average seek distance for the central cylinders is $N/4$ as compared to $N/2$ for the extreme cylinders. The combination of these two effects result in a slight bias of PCLOOK towards the central cylinders under light loads. Under heavy loads, the algorithm behaves like CLOOK which is fair.

In Figure 3f, the overall coefficient of variation of response time is plotted against the arrival rate. Note that all the algorithms exhibit significant variation and that the variation of NSCP and PCLOOK are only slightly less than that of others. This clearly shows that the fair algorithms achieve fairness, not by minimizing the variation inherent in disk scheduling, but instead by "evenly distributing" it.

The experiments for GATE and STOP scheduling granularities exhibited similar characteristics in that the improved algorithms for these granularities were found to reduce the response times and still remain satisfactorily fair (i.e. below 0.05) throughout the loading range. The details of these algorithms and experiments are available in [Prad95].

5 Conclusion

In this paper, the problem of simultaneously providing fair and efficient service in disk schedulers was addressed. A new fair scheduler, PCLOOK, was presented and through a simulation-based study, it was shown that PCLOOK has significantly improved response time performance as compared to NSCP, our earlier fair scheduler, while maintaining satisfactory fairness performance. PCLOOK achieves its response time improvement by reducing the movement of the disk head.

References

[Denn67] Denning, P., "Effects of Scheduling on File Memory Operations", *Proc. AFIPS SJCC*, 31, 1967.

[Fran69] Frank, H., "Analysis and Optimization of Disk Storage Devices for Time-Sharing Systems", *Journal of ACM*, 16(4), October 1969.

[GRP87] Geist, R., Reynolds, R., and Pittard, E., "Disk Scheduling in System V", *Proc. of ACM Sigmetrics Conference*, May 1987.

[HP95] Haritsa, J. R., and Pradhan, T., "Fair Disk Schedulers for High Performance Computing Systems", *Proc. of International Conf. on High Performance Computing*, December 1995.

[IEEE94] *IEEE Computer, Special Issue – The I/O Subsystem: A Candidate for Improvement*, March 1994.

[Mert70] Merten, A., "Some Quantitative Techniques for File Organization" *Ph.D. Thesis*, Univ. of Wisconsin, 1970.

[Oney75] Oney, C., "Queueing Analysis of the Scan Policy for Moving-Head Disks", *Journal of ACM*, 29(3), July 1975.

[Prad95] Pradhan, T., "Fair Schedulers for Disk Drives", *Master's Thesis*, Dept. of Computer Science and Automation, Indian Institute of Science, June 1995.

[SLW66] Seaman, P., Link, R., and Wilson, T., "An Analysis of Auxiliary Storage Activity" *IBM System Journal*, 5(3), 1966.

[Swar80] Swartz, G., "Polling in a LOOP system", *Journal of ACM*, 27(1), January 1980.

[TP72] Teorey, T., and Pinkerton, T., "A comparative Analysis of Disk Scheduling Policies", *Comm. of ACM*, 15(3), March 1972.

[WGP94] Worthington, L., Ganger, G., and Patt, Y., "Scheduling Algorithms for Modern Disk Drives", *Proc. of ACM Sigmetrics Conf.*, June 1994.