Fair Disk Schedulers for High Performance Computing Systems

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

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

Abstract

The design goal of most high-performance computing systems is to reliably provide massive data processing power. An equally important but frequently neglected issue is to ensure that this processing power reaches all the applications executing on the system. From a system perspective, the requirement of matching progress translates to providing fair service to applications.

In this paper, we examine the problem of scheduling disk requests in a fair manner. A new fairness metric is defined and a simulation study is made of the fairness characteristics of a variety of classical disk scheduling algorithms, over a wide range of system loads and scheduling granularities. The experimental results show that these algorithms fail to provide adequate fairness. To address this problem, new algorithms that consistently provide fair service are presented. The tradeoff between fairness and mean response time is also profiled.

1 Introduction

Designers of high-performance computing systems strive to reliably provide massive data processing power. A related issue is to ensure that the benefits of this data processing power are enjoyed by all the applications using the high performance computing system. This is especially required for the cooperative parallel applications that arise in scientific computing and visualization, wherein tasks need to keep pace with each other. In these environments, if the processing power is distributed in an unequal fashion, the benefits of parallel processing are lost to a large extent.

From a system perspective, the requirement of matching progress translates to providing fair service to applications. While processor scheduling has been extensively investigated in this context, the issue of providing fairness in disk scheduling, although considered, has not been analyzed to a similar extent. In this paper, we present a study of the fairness characteristics of a variety of disk scheduling algorithms. We focus on the disk subsystem since the latencies in disk service are usually substantially higher than those encountered at other system components.

Fairness

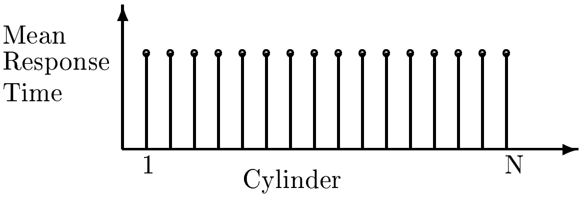
The term "fairness" has been used in several different senses – the interpretation used in this paper is that a resource divides its processing capacity equally among all its current users, that is, it provides "service fairness". Service fairness may be achieved in two different ways: "instantaneous" 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 applications 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 from two factors: First, the context-switch time in disks, that is, the time taken between the completion 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.

---

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


Mean
Response
Time
1
Cylinder
N

the group formation process is re-initiated by making the first arrival form the new group. In STOP, scheduling choices are made only whenever the disk head is stopped. Once a decision has been made and the disk head has started seeking towards the chosen request, that request has to be completely served before the next scheduling choice can be made. Finally, in DYNAMIC, scheduling decisions 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.

2.2 Classical Algorithms

The performance of a variety of disk scheduling algorithms has been evaluated in the literature. The main algorithms include FCFS, SSTF [2], POLL [2] and LOOK [4]. Modified versions of these basic algorithms such as Circular POLL [7] and Circular LOOK [8] 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, wherein 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.

2.3 New Algorithms

For each scheduling granularity, we have developed a new algorithm which exhibits consistent fairness over the entire range of system loading. These new algorithms are called PPOLL, PCPL and NSCP, and they are fair for GATE, STOP and DYNAMIC, respectively. The 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. The new algorithms, however, alter the position of the disk head, as described below, when the disk becomes empty of requests.

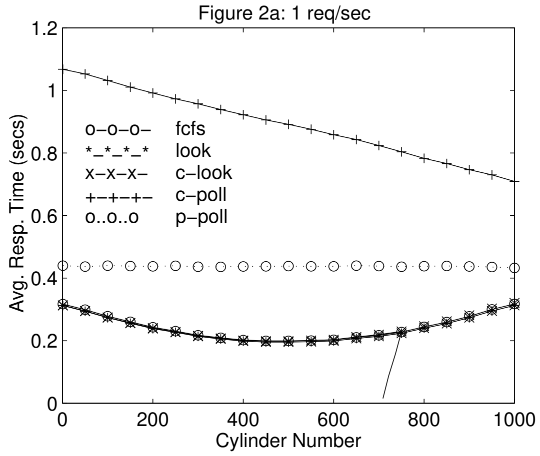
The PPOLL (Park POLL) scheduler is similar to POLL except that at the end of serving each group, the disk head moves to the last cylinder in the current sweep direction and "parks" there.

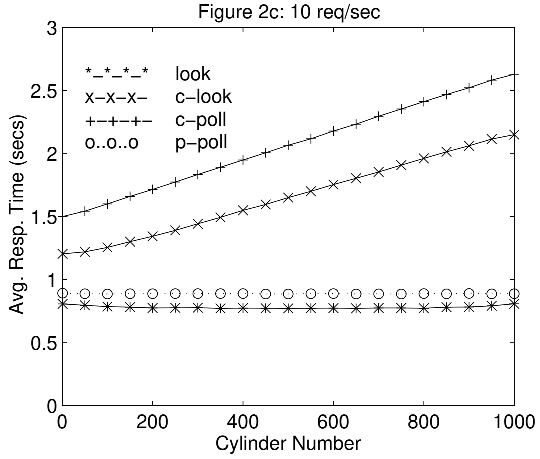
The PCPL (Park Circular POLL LOOK) scheduler is based on both CLOOK and CPOLL with two modifications: First, whenever the system becomes emptied of requests, the disk head moves to the nearest disk extremity and parks there. If the disk head is parked at the inner end, the service direction when new requests arrive is inside-out. However, if the head is parked at the outer end, the service direction when new requests arrive is outside-in. Second, when no further requests exist in the current sweep direction, the disk head is stroked back to the extreme cylinder.

The NSCP (Non-Stop Circular POLL) scheduler is identical to CPOLL except that the disk head continuously sweeps the disk surface even when the system is empty of requests.

3 Experiments and Results

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. In this model, the disk head seek times are computed using the common linear approximation formula $T_s = \alpha + \beta \times n_c \quad (n_c \geq 1)$ 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 [8]. 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,

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

where N is the number of cylinders, RT_i is

Figure 2e: Response Time vs. Arrival Rate

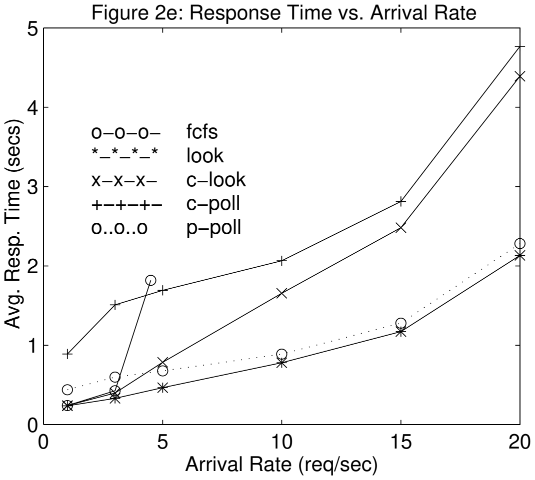


Figure 2f: Response Time Variation vs. Arrival Rate

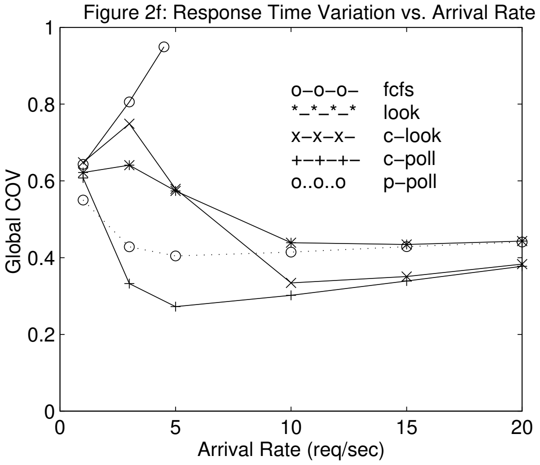


Figure 2g: Fairness vs. Arrival Rate

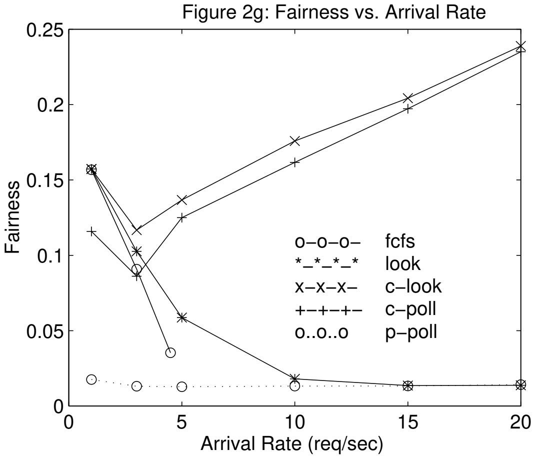
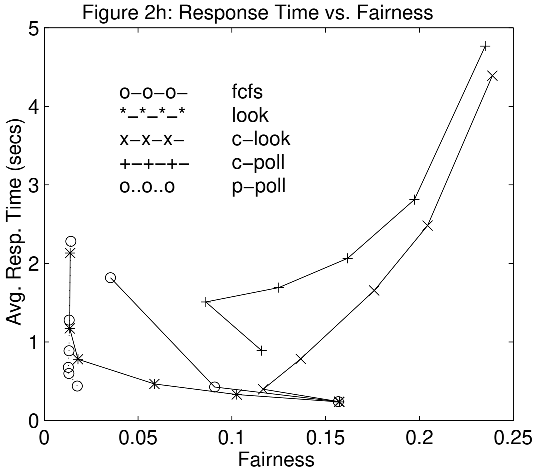


Figure 2h: Response Time vs. Fairness



2a, 2b). 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 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 LOOK, we observe that under light loading (Figures 2a, 2b) it favors the central cylinders over the extreme cylinders. However, at higher arrival rates, it becomes quite fair. In contrast, the circular algorithm CLOOK is unfair throughout the entire loading range, favoring the inner cylinders at the expense of the outer cylinders. This is because requests which arrive during a sweep are only served in the next sweep and the direction of the head while servicing requests is inside-out. Hence the outer cylinders wait longer for service than the inner cylinders.

The CPOLL scheduler shows an interesting behavior. At very low arrival rates (Figure 2a), new requests typically arrive to an empty system. Since the position of the disk head is uniformly distributed, requests for the inner cylinders typically have to wait while the disk head moves to the outermost cylinder and then strokes back to the innermost cylinder before recommencing the sweep. However, requests for the outer cylinders will usually be served directly since the disk head will tend to be located behind them. Therefore, the outer cylinders have a better response time than the inner cylinders. For moderate to heavy loads (Figures 2b – 2d), however, the inner cylinders are favored, and this is for the same reasons as described above for CLOOK.

Finally, turning our attention to the new algorithm PPOLL, we find that its cylinder response time distribution is nearly flat over the entire loading range

(Figure 2g). This can be explained as follows: At low arrival rates, when requests tend to arrive to an empty system, the parking feature of PPOLL ensures that the average distance of the new request from the current head position is always $N/2$, unlike FCFS (or classical POLL), where the average distance ranges from $N/4$ to $N/2$ as described above. Therefore, there is no discrimination against any cylinder group. At higher loads, when the disk head is moving inside-out the inner cylinders get a better response, whereas when the disk head is moving outside-in the outer cylinders are benefited. Since the requests are uniformly distributed over the disk and since each inside-out sweep is followed by an outside-in sweep, on average the response time is the same for all cylinders. At high arrival rates, LOOK is almost identical to PPOLL (as should be expected) since the requests in each group span most of the disk surface and disk idle periods are rare.

In Figure 2f, the overall coefficient of variation of response time is plotted against the arrival rate. Note that all the algorithms exhibit significant variation. More interestingly, the variation of PPOLL is greater than that of the circular algorithms for most of the loading range. This clearly shows that PPOLL is achieving fairness, not by minimizing the variation inherent in disk scheduling, but instead by "evenly distributing" the variation.

In Figure 2h, the tradeoff between mean response time and fairness is plotted. This clearly shows that PPOLL is satisfactorily fair in that its fairness level is always below 0.05 at all arrival rates. In addition, for medium to heavy loading, its response time is better than that of most of the other algorithms. Under light loading, however, its fairness is purchased at the expense of a significant increase in mean response time.

3.1 Summary of Other Results

The above simulation experiments showed that fair disk schedulers can be constructed for GATE scheduling granularity. The experiments for STOP and DYNAMIC scheduling granularities exhibited similar characteristics: the classical algorithms were unfair at one point or the other in the loading range, whereas the new algorithms, PCPL and NSCP, were fair throughout [6].

A drawback of the above fair algorithms is that they achieve fairness at the expense of significant increases in response times, especially at light loads. We have recently developed improved versions of these algorithms that provide better response time performance without sacrificing fairness [6].

4 Conclusion

In this paper, the problem of ensuring fair service in disk schedulers of high-performance computing systems was addressed. We defined a new fairness metric, the coefficient of variation of the cylinder mean response times. For this metric, the classical scheduling algorithms were found to be unfair for all the scheduling granularities. To address this problem, a new scheduling algorithm was proposed for each scheduling granularity and the experimental results showed these algorithms to be completely fair over the entire loading range in that their fairness value never exceeded 0.05. The new algorithms achieve 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 the cylinders.

References

[1] Coffman, E., Denning, P., *Operating Systems Theory*, Prentice-Hall, 1973.

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

[3] Geist, R., Daniel, S., "A Continuum of Disk Scheduling Algorithms", *ACM Trans. on Computer Systems*, February 1987.

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

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

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

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

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

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