Hindawi Publishing Corporation
International Journal of Stochastic Analysis
Volume 2011, Article ID 390548, 21 pages
doi:10.1155/2011/390548

Research Article

A Stochastic Analysis of Hard Disk Drives

Field Cady, Yi Zhuang, and Mor Harchol-Balter

Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15217, USA

Correspondence should be addressed to Field Cady, field.cady@gmail.com

Received 16 November 2010; Revised 10 February 2011; Accepted 15 February 2011

Academic Editor: Karl Sigman

We provide a stochastic analysis of hard disk performance, including a closed form solution for the average access time of a memory request. The model we use covers a wide range of types and applications of disks, and in particular it captures modern innovations like zone bit recording. The derivation is based on an analytical technique we call "shuffling", which greatly simplifies the analysis relative to previous work and provides a simple, easy-to-use formula for the average access time. Our analysis can predict performance of single disks for a wide range of disk types and workloads. Furthermore, it can predict the performance benefits of several optimizations, including short-stroking and mirroring, which are common in disk arrays.

1. Introduction

Hard disks have been the dominant form of secondary storage for decades. Though they are now joined by solid-state devices, they continue to be the most popular secondary storage medium in the world because of their relatively low cost; hence, it is critical to be able to accurately predict their performance characteristics. This is especially true because the workloads of disks, as well as their underlying technology, are evolving rapidly; even an excellent empirical understanding of disk performance today can quickly become dated. The difficulty of creating or reconfiguring hardware, as well as testing it under a range of realistic conditions, means that we must have tools to predict performance without using the disks themselves; creating the hardware is the last step. Hence, the community relies heavily on simulation and analytical modeling.

Analytical models are particularly desirable for their simplicity and their ability to provide insight. In engineering memory systems, analysis can be used to guide the initial design and to predict how the system will behave in new situations. Formulas can often be used to explain "why" a system behaves the way it does. Simulations are more general, because they do not need to simplify a model to the point of being analytically tractable,

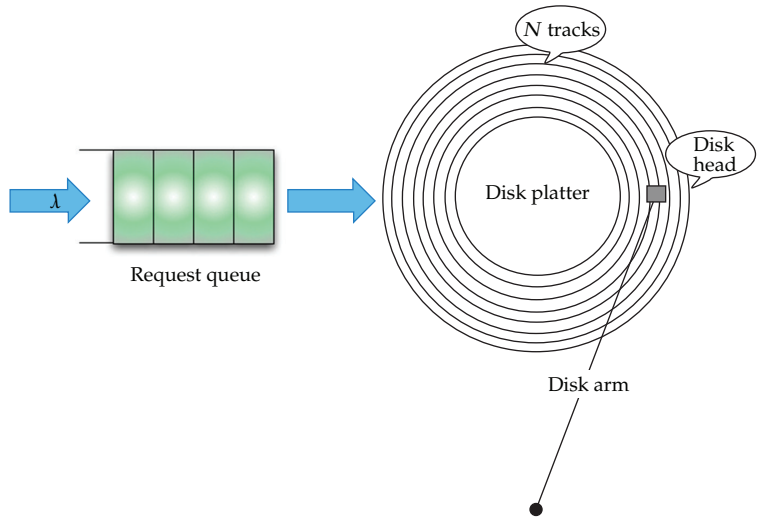

Figure 1: Illustration of a hard disk with a single platter. The disk rotates at a fixed rate and has a head which hovers over the tracks. λ is the arrival rate of memory requests.

but they can be complicated to implement and expensive to run, and they often provide less in the way of insight.

In this paper we present a stochastic model of a modern hard disk and derive a closed-form expression for the average time required to complete a memory request. We also provide several proofs-of-concept that this formula can be used to understand a number of features of modern disks, including their use in disk arrays. We first provide some background on disk technology and an overview of the existing literature on disks, especially theoretical treatments and how they relate to our work. Section 2 presents our mathematical model of a disk. In Section 3 we derive closed-form expressions for the average time to process a memory request under several request scheduling policies. We numerically verify the correctness of our formula, as well as validating our underlying model against real trace data, in Section 4. Section 5 discusses zone bit recording, an ubiquitous feature in modern disks, and shows that our model approximates it excellently. In Section 6, we move from single disks to disk arrays; we show that our model can quantify the benefits of short stroking, alternative memory layouts, and mirroring.

1.1. Background on Hard Disks

Physically, a disk is a circular platter with a magnetic surface, and memory arranged along tracks at different radii from the center. A "read/write head" hovers over the surface, reading or modifying the memory along a track as the disk rotates. Some disks have multiple platters, each with its own head; however, these heads move in sync, so for modeling purposes this is equivalent to one platter with a single head. Figure 1 shows a diagram of a disk with a single platter. Arriving memory requests are stored in a queue prior to servicing.

The most popular metric for disk performance is the average "access time," where the access time of a request is the time between its arrival in the system and its completion. The access time can be broken into three parts.

(1) *Wait time*: the time until the head is over the track to process the request.

(2) *Rotational latency*: the time spent waiting for the first bit of the requested memory to rotate under the head.

(3) *Transfer time*: the time to actually read/write the requested data.

A common term in the literature is "seek time", which typically refers to the time needed to the reposition of the head between servicing two requests. Hence, the seek time for a job would be the last part of the wait time, and "seeking" is used as a synonym for moving the read/write head.

The algorithm used to schedule memory requests is a critical component of disk performance. One of the most popular scheduling algorithms is SCAN. In SCAN, also known as "elevator scan", the head moves between the inner and outermost tracks, processing all requests at intermediate tracks along the way. A related algorithm is C-SCAN [1], which only processes requests while moving from the innermost track to the outermost track; when it reaches the outermost track, it moves directly back to the inner track, ignoring any requests along the way. C-SCAN is often preferred over SCAN because it is thought to be more fair to requests for memory at extremal radii [2]. Both algorithms are valued for their simplicity and relatively short access time, and they have a rich history in the literature and in applications, including being supported in every version of the popular DiskSim simulation program [3]. Other algorithms which have been used include Shortest Seek Time First (SSTF), which chooses the next request to minimize seek time, and Shortest Positioning Time First (SPTF), which also incorporates rotational latency. SSTF and SPTF have lower average access time than SCAN and C-SCAN. However, they also have high variance in access time and are biased against requests at the extremal tracks; these downsides are often felt to outweigh their advantages [4, 5]. All of these algorithms are superior to First Come First Serve (FCFS), which has also been studied.

Prior to the 1990's, every track on a disk contained the same amount of memory; the outer tracks, being longer, had fewer bits per unit of track length. This wasted real estate on the disk, but it was necessary because read/write heads could only read memory at a fixed rate. Modern disks, however, use *zone bit recording*, where the tracks are divided into "zones", and the zones at the outer radii of the disk store more data in each track. This means that at the outer radii of the disk:

(i) more bits are stored in each track,

(ii) the data transfer rate is higher, for a constant speed of rotation,

(iii) more bits are accessible with less seeking.

Thus a request of fixed size incurs less transfer time if placed on the outer tracks, and these tracks contain a disproportionate amount of memory. This observation is the basis of many modern data placement optimizations; however, previous analytical work did not take zone bit recording into account.

Another important development is combining individual disks into high-performance arrays, such as RAIDs. These arrays use many techniques to improve both performance and fault tolerance. One innovation is mirroring, where identical data is duplicated over several disks; while this is mostly done for fault tolerance, incoming requests are often split between the disks, reducing the load on each. New ways have been developed to partition memory among disks and tracks; the organ-pipe arrangement, for example, places memory on tracks according to its access frequency so as to minimize seek time. Another technique

is short stroking, where data is stored only in the outer tracks of disks (the inner tracks are sometimes used for redundant bits). This puts memory where the access time is minimized and gives fewer tracks for the head to scan over. In order to analyze RAIDs, it is useful to have a model for individual disks. So far, theorists have used simplistic models such as FIFO M/G/1 queues, where requests are processed in the order received and no seek time is required.

1.2. Prior Work

Disks have been studied using many tools, including theoretical analysis, trace-based simulation, and model-based simulation. Theoretical work initially focused on the access time of single disks, but later branched out to a range of topics. Traces from real disks have been used to study disk access patterns and to validate hypotheses from theory or model-based simulation. A large proportion of the disk literature uses model-based simulations, which can be very flexible and convenient.

The early theoretical works [4, 6–8] focused on SCAN and related algorithms. They modeled disk tracks as queues, and assumed that memory requests arrived as a Poisson process. These papers focused on the access time of requests; primarily the mean access time, but also its variance and fairness. Some of these treatments [4, 8] gave approximate answers or gave implicit solutions rather than closed-form formulas. Others [6, 7] included subtle mathematical errors, which rendered them only approximations. Later works [9–11] corrected these errors, but their derivations were very complicated and, more importantly, answers had to be evaluated numerically by solving systems of equations. In fact, the time required to solve these systems is polynomial in the number of tracks, which limits their practicality for the dense disks used today. Coffman and Gilbert [10] included a closed form for the fluid limit of his model, but the result is only valid when individual requests have vanishingly small service requirements. The model we use is similar to those in early papers, in particular Coffman et al. [6], but our analysis differs significantly.

Later theoretical work has expanded beyond access time under SCAN-like scheduling policies to include a range of topics. Some papers [12–14] have addressed more modern scheduling policies, though they have only done so in ways that are approximate or that give loose asymptotic bounds. Others [15–18] have discussed zone bit recording, but mostly in the context of other problems such as disk arrays or data placement. To our knowledge no paper has explicitly addressed the effect of zone bit recording on average access time. Probably the most extensive recent theoretical work is on disk arrays [17–24]. This includes analyses of several types of RAIDs, as well as a number of optimizations they use, and focuses on throughput, access time, and reliability. However, the literature standard is to model a disk array as a queueing network, where individual disks are FIFO M/G/1 queues [17, 19, 25–31]. Besides being only an approximation to real disk arrays, modeling disks with queues obscures how disk internals, such as speed of seeking and the scheduling algorithm used, impact the overall performance.

Papers based primarily on disk traces form a smaller portion of the literature. Much of the trace-based literature [32–34] has focused on characterizing the access patterns of memory systems, building models based on them, and comparing them to the (often implicit) assumptions used in simulations and analytical models. In addition, many papers which are based primarily on simulation or analysis validate their conclusions on a small set of real disk traces.

Because of their versatility, model-based simulations play a central role in the disk literature. Many papers [1, 35–40] have attacked the old problem of comparing request scheduling algorithms. There has also been extensive simulation-based study of more modern techniques, including zone bit recording [16, 41, 42] and disk arrays [43, 44]. Other papers [42, 45, 46] have examined less mainstream topics, such as power consumption and algorithms specific to video file servers.

Many important aspects of disks have not been adequately addressed by theory. For instance, we are not aware of any theoretical work which explicitly addresses the effects of zone bit recording on average access time. The models used in analytical treatments of disk arrays abstract away from all disk internals and obscure their impact on performance. Even the statistics of access time *without* zone bit recording have not been provided in a form that is readily usable. This paper is a partial solution to these problems. We use a similar model to the first papers, in particular Coffman et al. [6]. Our analysis is straightforward, and provides a closed form solution rather than a system of equations; on the other hand, we do not calculate moments of the access time beyond its mean. Though zone bit recording is not strictly covered by our analysis, we show that for practical purposes it can be excellently approximated. Finally, we show that our model can quantify the benefits of a number of modern disk optimizations, including those used in disk arrays.

2. Mathematical Model

We assume that the tracks are dense enough to form a continuum, which is reasonable as modern disks have many thousands of tracks, and let $r_{\min}$ and $r_{\max}$ denote the radii of the inner and outermost tracks of the disk. The disk rotates at a constant speed, and the read/write head moves with speed σ cm per unit time when it is scanning. Under SCAN, it maintains this speed in both directions. For C-SCAN, however, it is only scanning when moving from the inner track to the outer track; we assume that jumping back to the inner track takes a fixed time τ_0.

Memory requests arrive as a Poisson process with rate λ, and each request is for a contiguous piece of memory. We do not distinguish between reads and writes, because they both reduce to moving the read/write head over the requested memory. Each request consists of a location on the disk, indicating where the requested memory begins, and the size of the request measured in physical track length. While in reality a request may extend over several tracks, we assume the tracks are so closely spaced that each request is effective for memory at a single radius. We denote the radius for a request by the random variable R, whose probability density is given by $f_R(\cdot)$. The R values for different requests are i.i.d.

For analysis of C-SCAN and SCAN, we combine the rotational latency and transfer time for a memory request into a single quantity, which we call a "job". As soon as the head arrives at a track for which there is a pending request, it immediately begins work on the associated job and resumes motion when the job is finished. In this way the problem reduces to the head moving along a 1-dimensional path and pausing to complete jobs it encounters along the way. We denote the size of a job by the random variable S, and let its probability density be $f_S(\cdot)$. We assume that S is independent of R for a given request. We also assume that separate requests are i.i.d.

Other effects can be captured by our model. For example, processing a request which spans several adjacent tracks might require nonnegligible time to switch between tracks, in addition to the rotational latency and transfer time; this amounts to just an additional

Table 1: The symbols used in our mathematical model and derivation.

$r_{\min}$	Radius of the innermost track
$r_{\max}$	Radius of the outermost track
σ	The speed of the read/write head
τ_0	In C-SCAN, the time to backtrack from the outer to the inner track
λ	Arrival rate of the requests
R	r.v. denoting radius of the track specified by a memory request
$f_R(\cdot)$	pdf for R
S	r.v. denoting job size; rotational latency + transfer time for a request
$f_S(\cdot)$	pdf for S
n	The (large) number of jobs in a hypothetical request sequence
$E[T]$	The average access time of a job
T^l	The time access time of label l
M^l	The part of T^l during which the head is moving
P^l	The part of T^l during which the head is processing a job
N_Q	r.v. denoting the number of jobs in the system not being processed at a random time
ρ	The load of the system
S_e	r.v. denoting the remaining time on a job being processed
$f_H(\cdot)$	The pdf of the location of the head, given that it is not backtracking

contribution to the job size S, and our analysis holds if we modify $f_S(\cdot)$ so that these other time penalties are included in S. We can even partially account for non-Poisson arrival sequences; if several requests for contiguous memory arrive in quick succession, they can be treated as a single large request, and our analysis discusses when the last of these requests will be completed.

The key assumptions we make are the Poisson arrival of requests, the independence of different requests, and the independence of R and S for a given request (though the last of these can be treated approximately); these assumptions are standard in the theoretical disk literature. How well these assumptions are satisfied depends on the application; they are probably very accurate for file servers hosting files to many users, but very poor for a disk booting up an operating system.

Our analysis applies to general $f_R(\cdot)$ and $f_S(\cdot)$. However, for calculation we usually assume $f_R(r) \propto r$, which normalizes to $f_R(r) = 2/(r_{\max}^2 - r_{\min}^2)r$. This reflects the fact that under zone bit recording the capacity of a track increases with its radius; a random bit of memory is more likely to be in an outer track than an inner track.

3. Derivation

For simplicity this section will focus C-SCAN; most of the analysis also applies to SCAN, and the part where they differ (which is straightforward but more tedious for SCAN) is covered in the appendix. For reference, Table 1 summarizes our notation. We denote the access time of a random request by the r.v. T, and seek to calculate $E[T]$.

A common tool in modeling disks is the M/G/1 First Come First Serve (FCFS) queue, which is also a classic application of stochastic analysis. Although SCAN and C-SCAN do not process jobs in the order of their arrival, we would still like to leverage this tool. To this

end, we use a technique we call "shuffling" to break $E[T]$ into two parts, one of which is equivalent to an FCFS queue.

The notion of "average access time" is only meaningful when the system is ergodic, so that an equilibrium distribution over its states exists. We assume the system is ergodic, which is equivalent to assuming $\lambda E[S] < 1$, and we seek to calculate $E[T]$ for the equilibrium distribution. Physically this means that jobs arrive slowly enough that, on average, they do not pile up endlessly in the queue. Imagine running the disk for a very long time, during which n jobs arrive and are processed, and assume the system begins and ends empty. If n is large enough the average access time for this sequence will approach $E[T]$, and in our derivation we may assume that the system is at equilibrium. We will derive the average wait time for the arrival sequence under the assumption that n is extremely large.

3.1. Shuffles and Invariance of Average Access Time

Shuffles are best understood with a heuristic. Imagine that when a job arrives there is a physical label (like a sticky note) attached to it, and whenever a job is finished the label associated with it leaves the system. In this case there is a one-to-one correspondence between jobs and labels, and average time a label l spends in the system, which we will call it "access time" T^l, is clearly the same as the average access time of a job. We will for the rest of the derivation discuss the access time of labels, rather than of jobs. In a "shuffle" we allow the labels in the system to be permuted among the jobs in the system, in any way and at any point in time. When a job finishes its associated label is still discarded, but this may not be the label it had when it entered the system. The access time of a given label will in general be quite different depending on the shuffling protocol employed, but the average access time across all labels will be invariant under shuffling. This can be seen by observing that every instant of time contributes to $\sum_l T^l$ in proportion to how many jobs are in the system at that time, which is independent of how they are labeled.

3.2. Breakdown into Processing and Moving

The access time T^l for a label l can be broken down into its "processing time" P^l and its "moving time" M^l:

$$T^l = P^l + M^l. \tag{3.1}$$

By "processing time" we mean the part of the access time for the label during which the head is processing some job (perhaps the job to which the label is associated at a point in time). By "moving time" we mean the part of access time during which the head is moving. Let T, P and M be random variables denoting T^l, P^l, and M^l for a randomly chosen l. Averaging (3.1) we see

$$E[T] = E[P] + E[M]. \tag{3.2}$$

We mentioned above that $\sum_l T^l$ is invariant under shuffling, because each instant contributes to it based on how many jobs are in the system, regardless of their labels. Similarly, each instant contributes to $\sum_l P^l$ and $\sum_l M^l$ based on how many jobs are in

the system and whether or not the head is moving. Hence $E[P]$ and $E[M]$ are also invariant under shuffling.

The key idea of our derivation is to pick a convenient shuffling protocol to calculate $E[P]$ and a different shuffling protocol to calculate $E[M]$. Adding these results will be the average access time $E[T]$.

Note that so far our derivation has not used the scheduling policy or any assumptions about the request arrival sequence. These assumptions are only used in calculating $E[P]$ and $E[M]$.

3.3. *Calculating* $E[P]$

The derivation of $E[P]$ is identical for both SCAN and C-SCAN. To calculate $E[P]$, shuffle the labels so that they are processed in First Come First Serve order; the instant before the head arrives at a job, switch the label on this job with whichever label has been in the system the longest (if the job to be processed happens to already have the oldest label in the system, do nothing) so that it gets processed next. Jobs in general are not processed in the order they arrive, but this shuffling protocol ensures that at least the labels are. Then for a given label l, P^l will have three components:

(1) the time to finish the job being processed when l arrives, if there is one;

(2) the time to process any additional labels that are waiting when l arrives;

(3) the time to process l itself.

We calculate these using the PASTA (Poisson arrivals see time averages) property of Poisson arrivals.

To calculate component (1) of P^l, note that the head is processing a job a fraction $\lambda E[S]$ of the time, so when l arrives there is a job to finish with probability $\lambda E[S]$. We define $\rho = \lambda E[S]$, which is often called the load of the system. If the head is processing a job when l arrives, the amount of work left on the current job will be given by a random variable S_e, the equilibrium distribution of job size. It is a standard result that $E[S_e] = E[S^2]/2E[S]$, so (1) will just be $\rho E[S_e] = \lambda E[S^2]/2$.

Let the random variable N_Q denote the number of jobs in the system, which are not being processed, at a random time. Component (2) of P^l will just be $E[N_Q]E[S]$. If we apply Little's Law to the set of labels not being processed, we see that $E[N_Q] = \lambda(E[T] - E[S])$, so

$$E[P] = \rho E[S_e] + E\left[N_Q\right]E[S] + E[S] = \rho E[S_e] + \rho E[T] + (1-\rho)E[S]. \tag{3.3}$$

In deriving component (2) we have used our assumption that R and S for a given job are independent. N_Q is associated with the radius of the head (in C-SCAN, e.g., N_Q will generally be larger when the head is near $r_{\min}$), which is in turn associated with the locations of the waiting jobs (the area right behind the head will tend to have relatively few jobs). So if S and R were related there would be a very complicated dependency between N_Q and the sizes of the jobs that l would have to wait for.

3.4. Calculating $E[M]$

We will present two derivations of $E[M]$. The first is more direct, but only applicable to C-SCAN, whereas the second is more computationally intensive but generalizable to SCAN.

The simplest way to calculate $E[M]$ is to condition on whether the head is busy processing a job when a random job j arrives at a point R. The head spends $\tau_0 + (r_{\max} - r_{\min})/\sigma$ time moving between visits to R, so on average half of that time will remain giving

$$E[M \mid \text{moving}] = \frac{\tau_0 + (r_{\max} - r_{\min})/\sigma}{2}. \tag{3.4}$$

If the head is processing another job when j arrives, then because R and S are independent, the head's location H will be distributed identically to R; H and R will be i.i.d. points in the interval. The moving time for the head to go from point H to point R, then from R back to H, will be $\tau_0 + (r_{\max} - r_{\min})/\sigma$. But since R and H are i.i.d the average time to go $R \to H$ will be the same as the average to go $H \to R$, meaning

$$E[M \mid \text{processing}] = \frac{\tau_0 + (r_{\max} - r_{\min})/\sigma}{2}, \tag{3.5}$$

the same as the previous case, so that

$$E[M] = \frac{\tau_0 + (r_{\max} - r_{\min})/\sigma}{2}. \tag{3.6}$$

The more general derivation is analogous to the one in [6]. We use the trivial shuffle; a label is always associated with the job it enters the system with. Let us take a random label l; the idea is to calculate M^l conditioned on the radius for l's job and the state of the head when it arrives, then average appropriately. Let r_l denote the radius of l's job. If the head is backtracking when l arrives then M^l will be the time to finish backtracking (on average $\tau_0/2$) plus $(r_l - r_{\min})/\sigma$. If the head is not backtracking let its radius be r_h. If $r_h < r_l$ then M^l will be $(r_l - r_h)/\sigma$. Finally, if $r_h > r_l$ then M^l will be $(1/\sigma)(r_{\max} - r_h + r_l - r_{\min}) + \tau_0$. We then need only calculate the probability that the head is backtracking when l arrives and, in case the head is not backtracking, the distribution of r_h.

The average time between visits of the head to any one radius will be a constant τ, which we call the cycle time. Because a fraction ρ of the time is spent processing jobs, we have

$$\tau = \frac{\tau_0 + (r_{\max} - r_{\min})/\sigma}{1 - \rho}, \tag{3.7}$$

and the head will be backtracking when l arrives with probability τ_0/τ.

To calculate $f_H(\cdot)$, let dr be small enough that, at any point in time, we can assume there is at most one job whose associated radius is in $(r, r + dr)$. Now $f_H(r)dr$ will be proportional to the average amount of time the head spends in the interval $(r, r + dr)$ during one cycle. This will be the travel time to traverse the interval plus the time to

process the job which might have arrived since the head was last there. Hence $f_H(r)dr \propto (1/\sigma)dr + \lambda\tau f_R(r)drE[S] = (1/\sigma)dr + \rho\tau f_R(r)dr$, which normalizes to

$$f_H(r) = \frac{1}{\tau - \tau_0}\left(\frac{1}{\sigma} + \rho\tau f_R(r)\right). \tag{3.8}$$

Now recalling that when l arrives the head can be at a larger radius, at a smaller radius, or backtracking, we see that

$$\begin{aligned} E[M] = {} & \frac{\tau_0}{\tau}\left(\frac{\tau_0}{2} + \frac{1}{\sigma}\int_{r_{\min}}^{r_{\max}} (r_l - r_{\min}) f_R(r_l)dr_l\right) \\ & + \frac{\tau - \tau_0}{\tau}\int_{r_h=r_{\min}}^{r_{\max}}\int_{r_l=r_{\min}}^{r_h} \frac{1}{\sigma}(r_h - r_l) f_H(r_h) f_R(r_l)dr_l dr_h \\ & + \frac{\tau - \tau_0}{\tau}\int_{r_h=r_{\min}}^{r_{\max}}\int_{r_l=r_h}^{r_{\max}} \left(\frac{1}{\sigma}(r_{\max} - r_h + r_l - r_{\min}) + \tau_0\right) f_H(r_h) f_R(r_l)dr_l dr_h. \end{aligned} \tag{3.9}$$

After much calculation, this reduces to (3.7).

The derivation of $E[M]$ for SCAN, which is a straightforward variant of the one for C-SCAN, is summarized in the appendix.

3.5. The Final Answer

Adding $E[M]$ and $E[P]$ we see

$$\begin{aligned} E[T] &= E[M] + E[P] \\ &= E[M] + \rho E[S_e] + \rho E[T] + (1-\rho)E[S], \\ (1-\rho)E[T] &= E[M] + \rho E[S_e] + (1-\rho)E[S]. \end{aligned} \tag{3.10}$$

So the total average wait time for C-SCAN will be

$$\begin{aligned} E[T] &= \frac{E[M]}{1-\rho} + \frac{\rho}{1-\rho}E[S_e] + E[S] \\ &= \frac{\tau_0 + (r_{\max} - r_{\min})/\sigma}{2(1-\rho)} + \frac{\rho}{(1-\rho)}\frac{E[S^2]}{2E[S]} + E[S]. \end{aligned} \tag{3.11}$$

4. Validation

We validated this work in two ways. First, we empirically tested the correctness of our formulas by simulating our model under a wide range of parameters (different values of λ, σ and $r_{\max}$, and different functional forms and parameter values for f_R and f_S) and comparing the empirical access time to the theoretical predictions. In all cases, when run long enough,

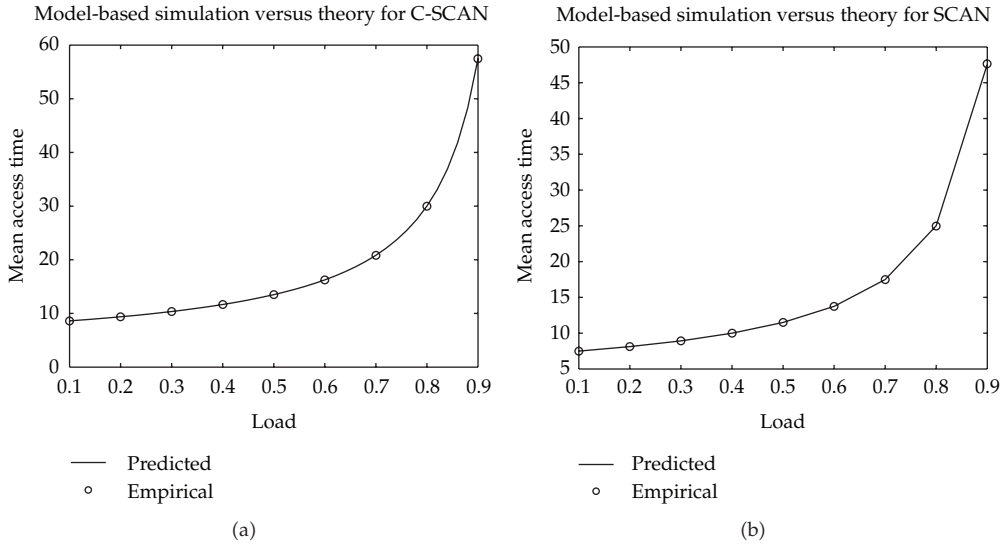

Figure 2: Predicted average access time compared to simulated access time. Request stream and disk behavior are generated according to model assumptions. Note that the curves for theoretic and simulated results are not visibly discernible for either SCAN or C-SCAN. In each case the radii were uniformly distributed between 0 and 1, σ was 3 cm/sec, every job had a fixed size of 5 sec, and 2 million requests were simulated. Similar agreement was seen across a range of parameters and distribution choices.

simulation agreed with theory to many decimal places; a typical example of this agreement is shown in Figure 2.

Secondly, we wished to test how well the underlying assumptions of our model were satisfied by real disks. Or more specifically, we wished to see whether real-world deviations from the model caused the performance to be substantially different from our predictions. We obtained trace data from a web server and simulated the performance of our modeled disk (i.e., the head still moved at its idealized constant speed) when receiving this real-world request sequence. The sizes and locations of the memory requests were taken from the trace; arrival times were randomized according to a Poisson process, so that we could explore a range of load scenarios by varying the arrival rate. For predicting the access time we assumed $f_R(r) \propto r$; this is reasonable when one is aware of zone bit recording but has no more detailed knowledge of memory layout. Comparing our predictions to the simulation results tested the following assumptions: (1) R is independent for separate requests, (2) S is independent for separate requests, (3) R and S are independent for a given request, and (4) $f_R(r) \propto r$. The results are shown in Figure 3. Across all points tested for both algorithms, theoretical and simulation results differed by less than 15%, and generally agreement was much closer. This even includes loads greater than .85, which is quite high for most systems. The fact that our formulas gave an accurate prediction indicates that either these assumptions are approximately satisfied by the system, or our formula is robust to violations of these assumptions.

The trace data (available at http://www.leosingleton.com/gt/disktraces/) was collected by Singleton et al. and used in [47]. The trace was collected using a Linux 2.6 web server with an ext3 journaling filesystem. The data consists of a sequence of locations (measured by starting block number) and the amount of memory blocks requested. We

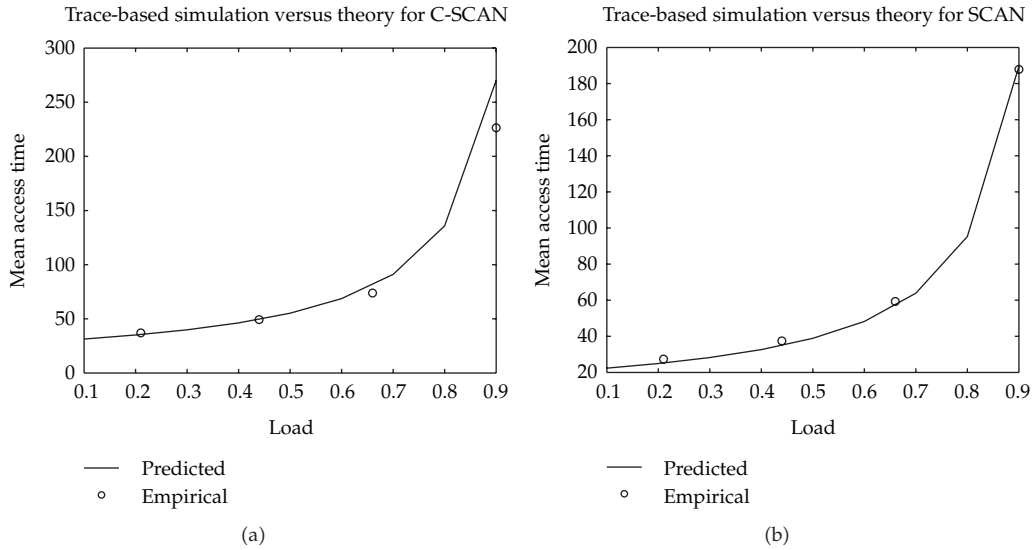

Figure 3: Predicted average access time compared to simulated access time using a real-world request stream. Size and location of memory requests were taken from a web server trace, and interarrival times were randomized to simulate Poisson arrivals under a range of system loads.

mapped block numbers to locations on a disk, assuming blocks are indexed starting from the outermost track. Consecutive requests for adjacent memory were treated as a single request; we discussed this technique in Section 2 as a way to adapt our derivation to non-Poisson arrival sequences.

5. Access Time under Zone Bit Recording

In disks with zone bit recording (ZBR) the transfer rate is highest for the outer tracks, meaning that S tends to be smaller for the outer tracks. However, the derivation in Section 3 assumes that job size is independent of track, so the derivation does not strictly apply to disks with ZBR. In this section we discuss how ZBR can be approximated using a variant of our model. We also discuss the average access time conditioned on R, a subject not discussed above, and how it can be approximated.

The most natural way to model ZBR is to assume that the length of track accessed by a memory request is independent of R, but the transfer time is proportional to $1/R$ (the statistics of the rotational latency would not change). In this more general case we let $S(r)$ be a random variable denoting the size of a job whose track is at radius r; S is the size of a job not conditioned on its radius. Our entire derivation, except the calculation of $E[P]$, can be easily modified to work for general $S(r)$, and we will show that $E[P]$ can in practice be exceptionally well approximated. There are two things which we would like to approximate: the average access time of the system, $E[T]$ and the average access time for jobs conditioned on R, $E[T \mid R = r]$.

As in Section 3, we only include the analysis for C-SCAN; SCAN is a straightforward extension.

5.1. Incorporating S(r)

We keep the definitions of M and P from Section 3. In addition, we define $M(r)$ and $P(r)$ to be M and P conditioned on $R = r$. The average wait time of a job at radius r will then be $E[T(r)] = E[M(r)] + E[P(r)]$.

In calculating $E[T] = E[M] + E[P]$, $E[M]$ can be calculated exactly, and $E[P]$ can be very well approximated. The only change to our derivation of $E[M]$ is that the equilibrium density of the location of the head, $f_H(r)$, must be modified. Using the argument from Section 3.4

$$f_H(r)dr \propto \frac{1}{\sigma}dr + \lambda\tau E[S(r)]f_R(r)dr \tag{5.1}$$

which normalizes to

$$f_H(r) = \frac{1}{\tau - \tau_0}\left(\frac{1}{\sigma} + \lambda\tau f_R(r)E[S(r)]\right). \tag{5.2}$$

The problem is our derivation of $E[P]$; we assumed that the number N_Q of labels in the system was independent of the sizes of the jobs waiting to be processed. But in the ZBR setting these quantities are not independent, because they both correlate with the location of the head. Our approach is to ignore any such correlation and simply plug $f_S(\cdot)$ (the weighted sum of the distributions across all radii) into our formulas above.

To validate this approximation to $E[T]$ we simulated our mathematical model, but using a variety of dependencies between R and S, and comparing it to the behavior predicted by our approximation. Figure 4(a) shows the results. For calculating the overall average access time our approximation works exceptionally well in all cases we tried: so well in fact that we suspect our formula may still be exact in this more general case, though we have not been able to prove it.

In calculating $E[T(r)]$, $M(r)$ can still be calculated exactly. We use the modified $f_H(\cdot)$ and condition on the location r_h of the head when a tagged job arrives:

$$\begin{aligned} E[M(r)] = {} & \frac{\tau_0}{\tau}\left(\frac{\tau_0}{2} + \frac{(r - r_{\min})}{\sigma}\right) \\ & + \frac{\tau - \tau_0}{\tau}\int_{r_h=r_{\min}}^{r_{\max}} \frac{1}{\sigma}(r_h - r)f_H(r_h)dr_h \\ & + \frac{\tau - \tau_0}{\tau}\int_{r_h=r_{\min}}^{r_{\max}} \left(\frac{1}{\sigma}(r_{\max} - r_h + r - r_{\min}) + \tau_0\right)f_H(r_h)dr_h. \end{aligned} \tag{5.3}$$

However, we do not have a rigorous approach to $P(r)$. Instead, we approximate $E[P(r)]$ by the approximate $E[P]$ discussed above, uniformly for all r. Figure 4(b) compares this approximation to simulation, and shows that this approach yields a reasonable estimate of the real performance. Our approximation is an overestimate for some tracks and an underestimate for others; these appear to balance out making our approximation to the overall $E[T]$ quite accurate.

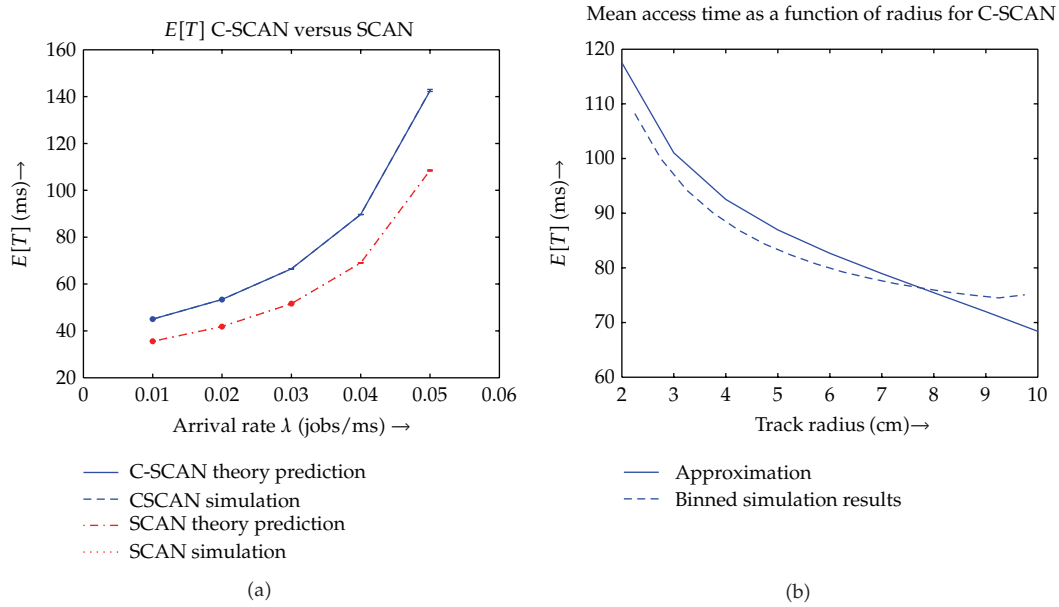

Figure 4: Our approximations to zone bit recording. (a): our approximation to the overall average access time under SCAN and C-SCAN. The difference between prediction and simulation is not visually discernible for either SCAN or C-SCAN. (b): our approximation to $E[T(r)]$ compared to simulation results under C-SCAN. Agreement is reasonable, but not as impressive as for the overall average access time. For both plots $r_{\min} = 2\,\text{cm}$, $r_{\max} = 10\,\text{cm}$, $\sigma = .33\,\text{cm/ms}$, and the disk spins at 7200 RPM.

5.2. Error Bounds

It is possible to loosely bound the errors of our approximation for the overall average access time. Recall that in Section 3.3 we saw $E[P] = \rho E[S_e] + E[N_Q]E[S]$. In the case of general $S(r)$ we bound this contribution by the extrema of $E[S(r)]$ over all r. This yields

$$\rho E[S_e] + E\left[N_Q\right]\min_r E[S(r)] \leq E[P] \leq \rho E[S_e] + E\left[N_Q\right]\max_r E[S(r)] \tag{5.4}$$

which implies

$$\frac{E[M] + \rho E[S_e]}{1 - \lambda \min_r E[S(r)]} \leq E[T] \leq \frac{E[M] + \rho E[S_e]}{1 - \lambda \max_r E[S(r)]}. \tag{5.5}$$

The quality of the bounds depends only on $\max_r E[S(r)]$ and $\min_r E[S(r)]$, rather than on higher moments.

6. Applications to Disk Arrays

This section argues that, though our analysis is based on a single disk, it is still useful for understanding disk arrays. As a proof of concept, this section shows how our formulas can be used to quantify the benefits of the following optimizations: mirroring, specialized memory

layouts, and short stroking. In order to be as realistic as possible, we assume that the disks use zone bit recording and apply the approximation from the previous section.

Mirroring is the practice of storing identical data on multiple disks. While mirroring is done partly for redundancy, which is outside the scope of this paper, it also brings performance benefits. Under mirroring, while a write request still goes to all disks, a read request can be serviced by any single disk. Assuming perfect load balancing, the effect of mirroring is captured in our model by simply reducing the request arrival rate λ, where the amount of reduction is dependent on the read-write ratio in the workload under consideration. Mirroring is an especially simple example of how our closed form formula can be applied to evaluate performance quickly. Memory layout and short stroking are somewhat more involved.

6.1. Alternate Memory Layouts

In Section 2 we discussed using $f_R(r) \propto r$ to reflect the fact that, when using zone bit recording, the storage capacity of a track is roughly proportional to its radius. However, $f_R(\cdot)$ is the density of incoming *requests*, not of memory itself. On a single disk a logical unit of memory can be placed on any of the tracks (or several adjacent tracks if it is large); if the choice of track is based on its access frequency, $f_R(\cdot)$ can be quite different. In a disk array, memory can also be placed on any of several disks. This flexibility has allowed for many optimizations of memory layout.

One memory placement optimization, which can be used for single disks or disk arrays, is the organ-pipe arrangement. It is intended for use with the SCAN algorithm, in which case it can reduce the average seek time. The organ-pipe arrangement places the most accessed memory near the middle tracks, and it can be modeled by using an $f_R(\cdot)$ which peaks at $(r_{\max} + r_{\min})/2$. We show such an experiment in Figure 5; use of the organ-pipe arrangement does reduce average access time, but only by a small amount, even at high load. This is as we would expect, because at high load $E[T]$ is dominated by $E[P]$ rather than $E[M]$.

For an example which uses a disk array, imagine that an array has two disks with different seek speeds and has its memory split equally between them. Imagine also that the array serves a request stream of rate λ. If the memory is partitioned randomly each disk will receive requests with rate $\lambda/2$, and the total average access time is found by averaging $E[T]$ for the two disks. But if the more frequently accessed memory is placed on the higher performance disk, the arrival rates to the two disks will be $\gamma\lambda$ and $(1-\gamma)\lambda$ for some γ, and the total average access time will be a weighted average of $E[T]$ for the disks.

6.2. Short Stroking

In short stroking, data is stored only in the outer tracks of a disk and the head only scans over these tracks. This is done in order to take advantage of the decreased transfer time and reduce the interval over which the head must scan. There is also the benefit of fewer requests going to any one disk, since it contains less memory than it would otherwise.

To quantify these benefits, imagine we have one disk worth of data and requests arriving as a Poisson process with rate λ. The data can be stored on a single disk of type D, using its full capacity, or on two identical disks of type D'. A D' disk can be made from

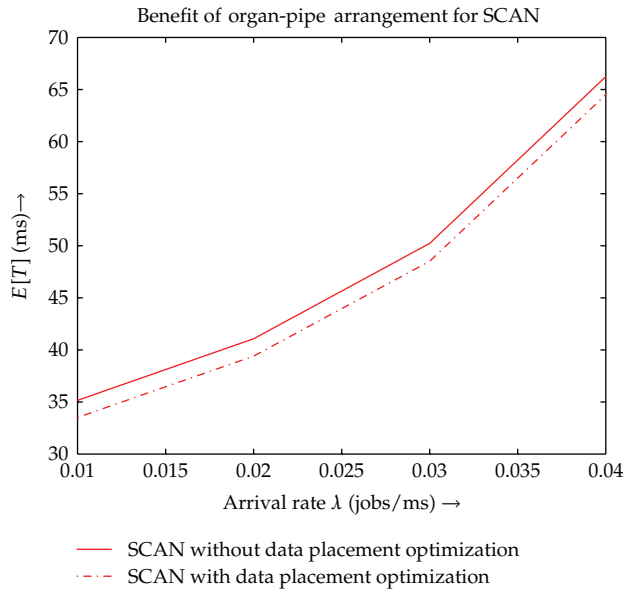

Figure 5: Performance of SCAN scheduling for the organ-pipe arrangement of data and random allocation of memory among tracks. The benefits of the organ-pipe arrangements are mild but consistent. For the plot $r_{\min} = 2\,\text{cm}$, $r_{\max} = 10\,\text{cm}$, $\sigma = .33\,\text{cm/ms}$, the disk spins at 7200 RPM, and each request is for 50 cm of physical track length. For $f_R(\cdot)$ under organ-pipe we use a quadratic which peeks at $(r_{\min} + r_{\max})/2$ and vanishes at the extremal tracks.

a D disk by storing memory only in its outer tracks; they are identical in all parameters except $r_{\min}$. In order that two D' disks have as much memory as a single D disk:

$$r'_{\min} = \sqrt{\frac{1}{2}r_{\max}^2 + \frac{1}{2}r_{\min}^2}. \tag{6.1}$$

Assuming that each request is only for data on one disk, each D' will receive requests at rate $\lambda/2$. The average access time for requests in the striped system will be given by (3.11), except with $\lambda \to \lambda/2$, $r_{\min} \to r'_{\min}$, and the statistics of S are modified to condition on a job's radius being $> r'_{\min}$. Figure 6(a) shows the predicted benefits of short stroking as a function of load.

There are several ways that short stroking improves access time: faster reads/writes, fewer tracks to scan over, and less load on each disk. The technique above shows how much benefit is given by all of these factors together, but we can actually quantify how much each of them individually contributes to the improved performance. If we do not change the distribution of S in going from the D disk to the D' disks, the performance of the striped system will not include the benefit of having faster reads/writes. Similarly, dividing σ by $(r_{\max} - r_{\min})/(r_{\max} - r'_{\min})$ in the striped system will remove the benefit of having fewer tracks to seek over. If we use the original value of λ in the striped system, this removes the benefit of reduced load on each disk. Figure 6(b) shows an example of this breakdown of benefits.

Our original disk D could have been striped over any number n of disks; Figure 7 shows the performance as a function of n, and the diminishing returns are evident. We show

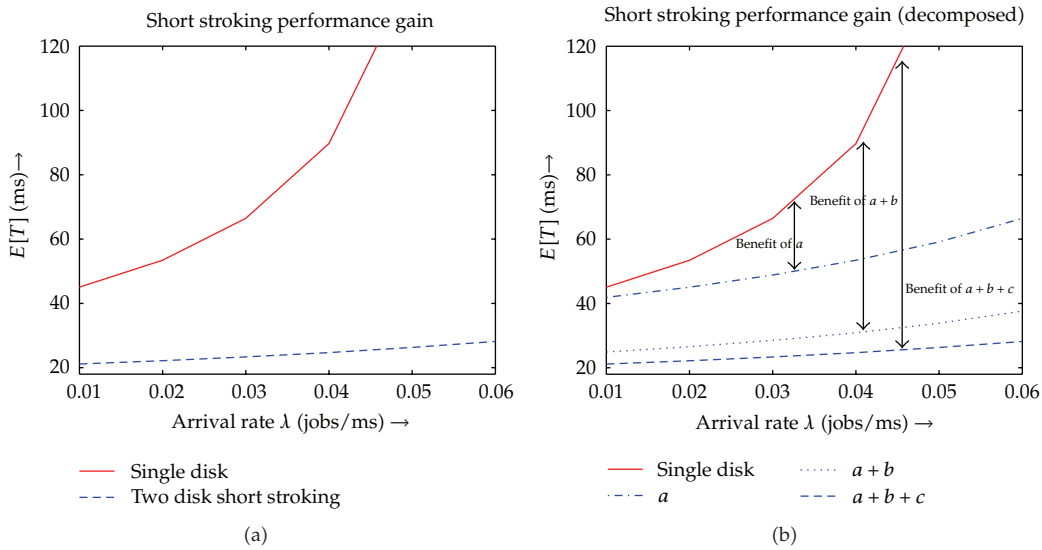

Figure 6: (a) The benefits of short stroking. (b) The benefits of short stroking, decomposed into components: (*a*) reduced arrival rate into each disk; (*b*) less seek time since fewer tracks are used; (*c*) faster data access at larger radii. For this plot $r_{\min} = 2\,\text{cm}$, $r_{\max} = 10\,\text{cm}$, $\sigma = .33\,\text{cm/ms}$, and the disk spins at 7200 RPM disk. Each request is for 50 cm of physical track length, and the scheduling policy is C-SCAN.

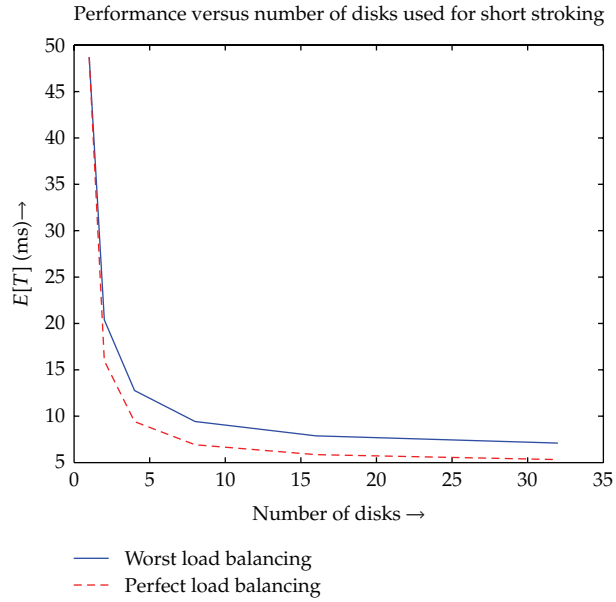

Figure 7: The benefit of short stroking versus the number of disks used, shown for perfect balancing of requests between disks and for the case that all requested memory is on one disk. For this plot, $r_{\min} = 2\,\text{cm}$, $r_{\max} = 10\,\text{cm}$, $\sigma = .33\,\text{cm/ms}$, and the disk rotates at 7200 RPM. Each request is for 1 cm of memory and the scheduling policy is C-SCAN.

performance both for perfect load balancing, where each disk receives requests at a rate λ/n, and worst-case load balancing, where all the memory which actually gets accessed is placed on one disk. In designing disk arrays, this point of diminishing returns can be compared to the marginal cost of adding additional disks.

7. Conclusion and Future Work

This paper fills the longstanding need for an analytical model of disk performance which is versatile, but simple enough to be easily applied. We show that it accurately predicts the performance of individual disks and can be used to quantify the benefits of various optimization techniques, without the need for simulations or numerical methods. This makes it valuable in designing disks and as a tool for predicting the performance of existing disks in novel situations. For example, it can quantify the benefits of increasing the speed of seeking, predict how performance will change when the average file size on a disk changes, and even identify the point of diminishing returns for striping in a disk array.

The next step is to leverage our model to attain a greater understanding of disk arrays, building on the preliminary results of Section 6. In particular, we desire a multidisk model in which the activities of the heads can be correlated; for example, if the disks were mirrored, incoming requests could be assigned to whichever head was closest. The problem of correlated heads is a significant mathematical challenge and will require techniques beyond the current treatment. Another avenue is to use our techniques to solve problems outside of disk scheduling. "Shuffling" applies to any system with one continuous degree of freedom, so long as the system is adjusted along a fixed path and "jobs" can only be processed when it is at a particular fixed value.

Disks will continue to play a prominent role in computer hardware for the foreseeable future, and this work presents a useful tool for their design and implementation.

Appendix

Derivation for SCAN

In SCAN we still have $E[T] = E[P] + E[M]$, and the derivation of $E[P]$ still applies; it is only in calculating $E[M]$ that SCAN and C-SCAN differ.

In C-SCAN the average time between visits to any track is a well-defined cycle time τ. But for SCAN it depends on which direction the head was moving when it last visited the track and on which track is being observed. We define the more general functions:

$$\begin{aligned}\tau_o(r) &= \text{the average time for the head to return to radius } r \text{ after it passes it going outward,}\\ \tau_i(r) &= \text{the average time for the head to return to radius } r \text{ after it passes it going inward.}\end{aligned} \tag{A.1}$$

The average time for the head to return to the same location *and direction of motion* must be a constant $\tau^* = (2(r_{\max} - r_{\min})/\sigma)/(1-\rho)$. We calculate $\tau_o(r)$ by observing that in one sweep

from r to $r_{\max}$ and back, the average amount of work done must be equal to the total amount of work that arrives with a radius $> r$ during τ^*. Hence

$$\tau_o(r) = 2\left(\frac{r_{\max} - r}{\sigma}\right) + (\lambda\tau^*)E[S]\int_r^{r_{\max}} f_R(x)dx, \tag{A.2}$$

and similarly for $\tau_i(r)$.

The $f_H(\cdot)$ from our C-SCAN derivation must also be changed. Instead we define $f_H^o(r)$ such that $f_H^o(r)dr$ is the probability that the head is in $(r, r+dr)$ *and moving outward*, and similarly $f_H^i(r)$. Then

$$\begin{aligned} f_H^o(r)dr &\propto \frac{1}{\sigma}dr + (\lambda\tau_i(r))E[S]f_R(r)dr, \\ f_H^i(r)dr &\propto \frac{1}{\sigma}dr + (\lambda\tau_o(r))E[S]f_R(r)dr, \end{aligned} \tag{A.3}$$

and we normalize by setting $\int f_H^o(r)dr + \int f_H^i(r)dr = 1$. Finally, we must condition on whether r_l is more or less than r_h and on the direction of travel of the head:

$$\begin{aligned} E[M] = &\int_{r_{\min}}^{r_{\max}}\int_{r_{\min}}^{r_l} \frac{1}{\sigma}(r_l - r_h)f_H^o(r_h)f_R(r_l)dr_h\,dr_l \\ &+ \int_{r_{\min}}^{r_{\max}}\int_{r_l}^{r_{\max}} \frac{1}{\sigma}(2r_{\max} - r_l - r_h)f_H^o(r_h)f_R(r_l)dr_h\,dr_l \\ &+ \int_{r_{\min}}^{r_{\max}}\int_{r_{\min}}^{r_l} \frac{1}{\sigma}(r_l + r_h - 2r_{\min})f_H^i(r_h)f_R(r_l)dr_h\,dr_l \\ &+ \int_{r_{\min}}^{r_{\max}}\int_{r_l}^{r_{\max}} \frac{1}{\sigma}(r_h - r_l)f_H^i(r_h)f_R(r_l)dr_h\,dr_l. \end{aligned} \tag{A.4}$$

While perhaps daunting, this formula can easily be evaluated using Mathematica or other symbolic math software.

References

[1] M. Seltzer, P. Chen, and J. Ousterhout, "Disk scheduling revisited," in *Proceedings of the USENIX Technical Conference*, pp. 313–324, 1990.

[2] M. Andrews, *Scheduling techniques for tacket routing, load balancing and disk scheduling*, Ph.D. thesis, Supervisor-Goemans, Michel X., 1997.

[3] J. S. Bucy and G. R. Ganger, "The disksim simulation environment version 3.0 reference manual," Tech. Rep., 2003.

[4] T. J. Teorey and T. B. Pinkerton, "Comparative analysis of disk scheduling policies," *Communications of the ACM*, vol. 15, no. 3, pp. 177–184, 1972.

[5] N. C. Wilhelm, "An anomaly in disk scheduling: a comparison of fcfs and sstf seek scheduling using an empirical model for disk accesses," *Communications of the ACM*, vol. 19, no. 1, pp. 13–18, 1976.

[6] E. G. Coffman, L. A. Klimko, and B. Ryan, "Analysis of scanning policies for reducing disk seek times," *SIAM Journal on Computing*, vol. 1, no. 3, pp. 269–279, 1972.

[7] W. C. Oney, "Queueing analysis of the scan policy for moving-head disks," *Journal of the Association for Computing Machinery*, vol. 22, pp. 397–412, 1975.
[8] C. C. Gotlieb and G. H. MacEwen, "Performance of movable-head disk storage devices," *Journal of the ACM*, vol. 20, no. 4, pp. 604–623, 1973.
[9] E. G. Coffman, Jr. and M. Hofri, "On the expected performance of scanning disks," *SIAM Journal on Computing*, vol. 11, no. 1, pp. 60–70, 1982.
[10] E. G. Coffman, Jr. and E. N. Gilbert, "Polling and greedy servers on a line," *Queueing Systems*, vol. 2, no. 2, pp. 115–145, 1987.
[11] E. Coffman and M. Hofri, "Queueing models of secondary storage devices," in *Stochastic Analysis of Computer and Communication Systems*, Elsevier, New York, NY, USA, 1990.
[12] T. S. Chen, W. P. Yang, and R. C. T. Lee, "Amortized analysis of some disk scheduling algorithms: SSTF, SCAN, and *N*-StepSCAN," *BIT. Numerical Mathematics*, vol. 32, no. 4, pp. 546–558, 1992.
[13] M. Andrews, M. A. Bender, and L. Zhang, "New algorithms for the disk scheduling problem," in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pp. 550–559, Burlington, Vt, USA, 1996.
[14] T.-H. Yeh, C.-M. Kuo, C.-L. Lei, and H.-C. Yen, "Competitive analysis of on-line disk scheduling," in *Proceedings of the 7th International Symposium on Algorithms and Computation (ISAAC '96)*, Lecture Notes in Comput. Sci., pp. 356–365, Osaka, Japan, 1996.
[15] S. Ghandeharizadeh, D. J. Ierardi, D. Kim, and R. Zimmermann, "Placement of data in multi-zone disk drives," 1996.
[16] R. Van Meter, "Observing the effects of multi-zone disks," in *Proceedings of the Usenix Technical Conference*, 1997.
[17] A. Thomasian and J. Menon, "RAID5 performance with distributed sparing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 6, pp. 640–657, 1997.
[18] A. Thomasian and M. Blaum, "Higher reliability redundant disk arrays: organization, operation, and coding," *ACM Transactions on Storage*, vol. 5, no. 3, article no. 7, 2009.
[19] A. Thomasian and J. Menon, "Performance analysis of raid5 disk arrays with a vacationing server model for rebuild mode operation," in *Proceedings of the 10th International Conference on Data Engineering*, pp. 111–119, 1994.
[20] E. K. Lee and R. H. Katz, "An analytic performance model of disk arrays," in *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '93)*, pp. 98–109, 1993.
[21] M. Uysal, G. A. Alvarez, and A. Merchant, "A modular, analytical throughput model for modern disk arrays," in *Proceedings of the 9th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS '01)*, pp. 183–192, 2001.
[22] E. Varki, A. Merchant, J. Xu, and X. Qiu, "Issues and challenges in the performance analysis of real disk arrays," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 6, pp. 559–574, 2004.
[23] A. Thomasian, B. A. Branzoi, and C. Han, "Performance evaluation of a heterogeneous disk array architecture," in *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS '05)*, pp. 517–520, September 2005.
[24] A. S. Lebrecht, N. J. Dingle, and W. J. Knottenbelt, "A performance model of zoned disk drives with I/O request reordering," in *Proceedings of the 6th International Conference on the Quantitative Evaluation of Systems (QEST '09)*, pp. 97–106, September 2009.
[25] S. Z. Chen and D. Towsley, "The design and evaluation of raid 5 and parity striping disk array architectures," *Journal of Parallel and Distributed Computing*, vol. 17, no. 1-2, pp. 58–74, 1993.
[26] J. Menon, "Performance of RAID5 disk arrays with read and write caching," *Distributed and Parallel Databases*, vol. 2, no. 3, pp. 261–293, 1994.
[27] A. Merchant and P. S. Yu, "Analytic modeling of clustered RAID with mapping based on nearly random permutation," *IEEE Transactions on Computers*, vol. 45, no. 3, pp. 367–373, 1996.
[28] A. Thomasian, C. Han, G. Fu, and C. Liu, "A performance evaluation tool for RAID disk arrays," in *Proceedings of the 1st International Conference on the Quantitative Evaluation of Systems (QEST '04)*, pp. 8–17, September 2004.
[29] H. Takagi, *Queueing Analysis: A Foundation of Performance Evaluation*, Elsevier Science, 1991.
[30] L. Kleinrock, *Queueing Systems*, Wiley Blackwell, New York, NY, USA, 1975.
[31] S. Lavenberg, *Computer Performance Modeling Handbook*, vol. 4 of *Notes and Reports in Computer Science and Applied Mathematics*, Academic Press, New York, NY, USA, 1983.
[32] C. Ruemmler and J. Wilkes, "Unix disk access patterns," in *Proceedings of the USENIX Technical Conference Proceedings*, 1992.

[33] W. W. Hsu and A. J. Smith, "Characteristics of I/O traffic in personal computer and server workloads," *IBM Systems Journal*, vol. 42, no. 2, pp. 347–372, 2003.
[34] A. Riska and E. Riedel, "Long-range dependence at the disk drive level," in *Proceedings of the 3rd International Conference on the Quantitative Evaluation of Systems (QEST '06)*, pp. 41–50, September 2006.
[35] R. Geist, R. Reynolds, and E. Pittard, "Disk scheduling in system v," in *Proceedings of the 1987 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '87)*, pp. 59–68, 1987.
[36] R. Geist and S. Daniel, "A continuum of disk scheduling algorithms," *ACM Transactions on Computer Systems*, vol. 5, no. 1, pp. 77–92, 1987.
[37] M. Hofri, "Disk scheduling: Fcfs vs. sstf revisited," *Communications of the ACM*, vol. 23, no. 11, pp. 645–653, 1980.
[38] D. M. Jacobson and J. Wilkes, "Disk scheduling algorithms based on rotational position," Tech. Rep., Hewlett-Packard, Palo Alto, Calif, USA, February 1991.
[39] A. Thomasian and C. Liu, "Some new disk scheduling policies and their performance," in *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '02)*, pp. 266–267, 2002.
[40] A. Thomasian and C. Liu, "Performance evaluation for variations of the satf scheduling policy," in *Proceedings of the Internationl Sympium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS '04)*, pp. 431–437, 2004.
[41] B. L. Worthington, G. R. Ganger, Y. N. Patt, and J. Wilkes, "On-line extraction of scsi disk drive parameters," in *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, pp. 146–156, 1994.
[42] S. Chen and M. Thapar, "Zone-bit-recording-enhanced video data layout strategies," in *Proceedings of the 4th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 29–35, February 1996.
[43] A. Thomasian, "Priority queueing in raid disk arrays with an nvs cache," in *Proceedings of IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS '95)*, 1995.
[44] A. Thomasian, "Multi-level RAID for very large disk arrays," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 4, pp. 17–22, 2006.
[45] A. Riska and E. Riedel, "Its not fair—evaluating efficient disk scheduling," in *Proceedings of the 11th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS '03)*, pp. 288–295, 2003.
[46] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang, "Modeling hard-disk power consumption," in *Proceedings of the USENIX Conference on File and Storage Technologies*, 2003.
[47] L. Singleton, R. Nathuji, and K. Schwan, "Flash on disk for low-power multimedia computing," in *Proceedings of the 14th Annual Multimedia Computing and Networking 2007*, February 2007.