UNIVERSITY OF CALIFORNIA, SAN DIEGO

A Space Efficient Disk Drive Rotational Positioning Scheduling Algorithm

A thesis submitted in partial satisfaction of the requirements for the degree Master of Science in Computer Science

by

Pei-Jung Ho

Committee in charge:

Professor Walter Burkhard, Chair
Professor Venkat Rangan
Professor Yannis Papakonstantinou

2000

The thesis of P ei-J ung Ho is approved:

Chair

University of California, San Diego

2000

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF T ABLES

ACKNOWLEDGEMENTS

I would like to thank Geoff Voelker, Jesse Steinberg and Florin Baboescu for providing helpful comments on the early draft of this thesis.

ABSTRACT OF THE THESIS

A Space Efficient Disk Drive Rotational Positioning Scheduling Algorithm

by

Pei-Jung Ho
Master of Science in Computer Science
University of California, San Diego, 2000
Professor Walter Burkhard, Chair

Modern disk drives use rotational positioning algorithms for scheduling. The rotational positioning optimizing (RPO) algorithm is a probabilistic approach based on the rotational positioning algorithm proposed by Heath, which is the major scheduling algorithm in state-of-the-art disk drives. This thesis describes the implementation details of the RPO algorithm and analyzes its scheduling performance by using a disk drive RPO simulator. Two schemes for reducing the storage space of the RPO scheduling information are presented. The first scheme is to change the RPO head positioning data representations, and the second is to use a threshold value to minimize RPO table space. The simulation results examine the trade-off between the storage space and the system performance. Furthermore, the observation of the distribution of relative rotational displacements of successive operations provides a good insight into the behavior of the RPO algorithm.

Chapter I: Introduction

Modern disk drives use rotational positioning scheduling algorithms to select the best queued operation to service. Shortest Access Time First (SATF) [Jacobson et al. 91] uses a greedy algorithm to achieve an approximation to the Optimal Access Time algorithm. Heath describes a method to reduce the time and memory overhead of SATF which improves the scheduling performance [Heath et al. 96]. The Rotational Positioning Optimizing (RPO) algorithm is an implementation of the rotational scheduling algorithm based on Heath's method, and is used extensively in state-of-the-art disk drives.

RPO tables are a central data structure stored on individual disk drives. The drive memory space is costly and consequently minimizing the RPO table size will benefit manufactures and consumers alike. The primary goal of this thesis is to examine the trade-off between memory space for the tables and the performance of the scheduling algorithm. Excellent insight regarding the trade-off is obtained using a disk drive simulation. The simulation results show that the difference in seeking behavior between the heads in a drive are negligible. Therefore, all the heads sharing a common set of RPO tables is more efficient. By adapting a threshold value for each table entry, the size of RPO tables is reduced approximately by a factor of 30 while the performance degradation reported from the simulation is at most 5.94%.

A second goal is to study the distribution of relative rotational displacements of successive operations. The distribution validates the impacts of queue size and seek range and provides insight into the behavior of the RPO algorithm.

I utilize an enhanced disk drive RPO simulator, the prototype simulator was created by Palmer [Palmer 99]. The simulator accurately models the data layout, rotational positioning optimizing schemes, RPO table representation, and seek times.

This study is motivated by the current state of disk drive technology. The design trend of storage systems is being affected by the decreasing cost of memory. The Active Disks system moves portions of an application's processing to execute directly onto disk drives [Riedel et al. 98]. Regarding this intriguing perspective, minimizing the necessary memory space for scheduling still has the benefit of providing additional resources for the computations pushed onto the disks. On the other hand, to provide more accurate scheduling, finer-grained RPO tables may be used in the future and more space will be needed. The conclusions from this study will still be beneficial.

The remainder of this thesis is organized as follows. Chapter II describes the physical components, data layout and scheduling algorithms of modern disk drives. Chapter III presents the specifics of the IBM Ultrastar 18LZX as well as the simulator. Chapter IV presents results regarding the memory space efficient RPO table representation. Chapter V presents observations regarding the distribution of relative rotational displacements. Finally Chapter VI contains my conclusions and suggestions for continued work.

Chapter II: Background

To accurately model a disk drive, it is essential to understand its detailed components. This chapter presents the characteristics of modern disk drives. It begins by describing the physical device, then describes the process of serving a request and the components of the request service time. It discusses some important disk scheduling algorithms, including the classical algorithms and the latest algorithm used in modern disk drives.

2.1 Mechanical Components and Data Layout

A disk contains a collection of platters, rotating constantly on a center spindle. The rotation speed is represented as revolutions per minute (RPM), and is one of the most important considerations in disk performance because a higher rotational speed directly decreases the seek latency and improves the data access rate. The rotational speed of modern disk drives can exceed 10,000 RPM, and the most recent drive from Seagate can rotate as fast as 15,000 RPM. However, the most common rotational speeds in current disk drives are 5,400 and 7,200 RPM.

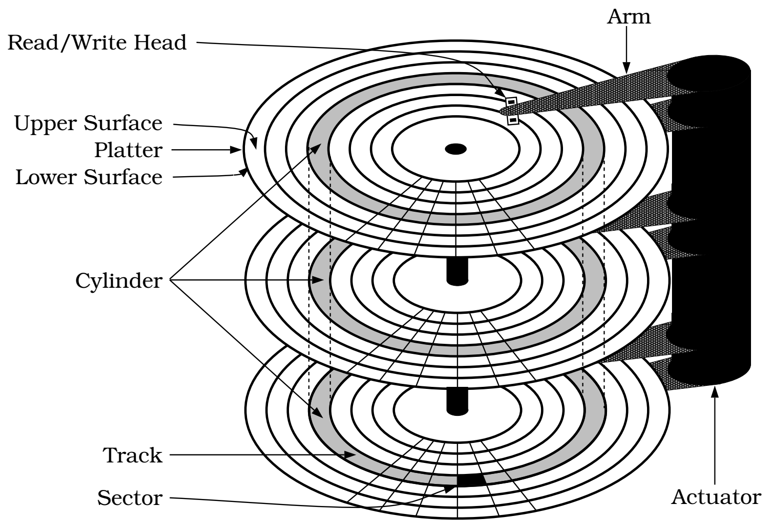


Figure 1: Disk drive internals (Source: [Worthington et al. 95])

The disk mechanism is shown in Figure 1. Data is stored on both sides of each platter, and each surface of a platter has a head responsible for reading and writing data on it. Each of the heads is attached to a disk arm, and all the disk arms are attached to a common rotational pivot and rotated together.

The surface of a platter is divided into a series of concentric circles called tracks. All the tracks with the same radial distance to the spindle in a single disk drive are referred to as a cylinder. Each track consists of a number of sectors, which are the smallest size data layout units on the platters. Server disks allow administrators to specify sector size within a range; typically this is 512 bytes. Client disks have one sector size -- 512 bytes.

In a Small Computer System Interface (SCSI) [SCSI-2 93] drive, the storage space can be viewed as a linear space of addressable blocks, and the disk controller is responsible for mapping each logical block address into the real physical sector.

To maximize the storage capacity of a disk, the outside tracks contain more sectors than inside tracks. This technique is called *zoning*. A zone is a group of adjacent cylinders. All of the cylinders in one zone have the same number of sectors per track. The outermost zone of a platter has the most sectors per track, and the number of sectors per track decreases towards the center. Ten to fifteen zones are very common in modern disk drives. Zoning substantially improves storage capacity as well as data transfer rates of a disk, since more sectors can be visited by the read/write head in one revolution on the outside zones.

2.2 Disk Access

To access the data stored on a platter, the head is moved over the desired track and waits for the exact sector to rotate under it. The service time is defined as the total time for servicing a request and is calculated by the following formula:

service time = seek time + rotational latency + transfer time

+ controller overhead

- Seek time is the time it takes for the read/write head to move from its current cylinder location to the target cylinder. The seek time depends on the seek distance, but is not proportional to the seek distance. For a long seek, the arm first moves with constant acceleration until it reaches the maximum speed, then maintains full speed. When close to the target, the arm slows down, then the head is settled to the exact location by the disk controller. A short seek follows the same procedure except that the arm starts to slow down before reaching its maximum speed, so the settle time dominates the entire seek time. Long seeks take time proportional to the distance plus the settle time, and short seeks take time approximately proportional to the square root of the distance plus the settle time.

- Rotational latency is the time it takes for the target sector to rotate to the position under the read/write head, and is determined by the rotational speed of the disk.
- Transfer time is the time used to transfer data, and depends on the block size, rotational speed, recording density and bus rate.
- Controller overhead is the time that the disk controller needs to process an I/O access.

As an example, the advertised average seek time of IBM Ultrastar 36LZX 3.5-inch server disk is 4.9 ms [IBM 00]. The rotational speed is 10,000 RPM, so the average rotational latency is 2.99 ms. The data transfer rate is 35 to 56.5 MB per second, and the controller overhead is approximately 1 ms.

2.3 Scheduling Algorithms

A disk's workload is characterized by bursts [Ruemmler et al. 93]. When the queue of pending commands becomes long, the scheduler is responsible for reordering the waiting commands to maximize the total throughput of requests. In addition to minimizing the seek time between every pair of commands, the scheduling algorithm may also be responsible for maintaining fairness to avoid starvation.

The first and simplest form of disk scheduling is First Come First Served (FCFS). This algorithm simply serves requests in their arrival order. The advantages of this approach are that it is easy to implement and eliminates the starvation problem. However, it performs poorly even under light workloads.

Seek time scheduling algorithms outperform FCFS by using the seek distance as the scheduling parameter, and are introduced in Section 2.3.1. Improving on the seek time based algorithms, rotational scheduling algorithms consider both seek time and rotational latency in scheduling. Seek time and rotational latency combined make up the access time, and access time based algorithms are discussed in Section 2.3.2.

2.3.1 Seek time based scheduling algorithms

Seek time is used as the scheduling parameter in algorithms such as SSTF, SCAN and LOOK. As described earlier, the seek time is computed as a function of the seek distance, which is the number of cylinders between the current location of the head if it is idle, or the end location of the executing command and the location of the next command.

Shortest Seek Time First (SSTF) is the first scheduling algorithm based on head positional information, in which the request in the command queue closest to the current position will be serviced next. Compared to FCFS, SSTF dramatically decreases the expected waiting time since less time is spent in head movement. However, starvation becomes a problem when most of the waiting requests are located within a small area.

The SCAN algorithm was proposed in [Denning 67]. It uses both the head position and the head movement direction in scheduling. If the head is currently moving toward cylinder number 0, it serves the waiting requests in descending order according to their cylinder numbers. When the head reaches cylinder 0, it reverses direction and starts serving requests in ascending order. The head moves back and forth like an elevator, so this algorithm is also called the elevator algorithm. This algorithm has a fairness problem. The requests located in the center of the disk have less average waiting time than the requests at the edges. Cyclical SCAN (C-SCAN) is proposed to address this fairness problem [Seaman et al. 66]. It only moves the head in one direction when serving requests, and moves the head back to the starting point when it reaches the last cylinder.

LOOK is a variation of the SCAN algorithm which changes the full stroke movement of the head [Merten 70]. Instead of scanning all cylinders, the head stops and reverses direction when no more waiting requests are located in the down stream of the current head position. LOOK improves the performance by removing the unnecessary head movement. Cyclical LOOK (C-LOOK) to LOOK is the same as C-SCAN to SCAN, serving requests with one head direction.

[Geist et al. 87] proposed a continuum of algorithms between SSTF and LOOK, called V(R). The R parameter denotes how strongly biased the scheduler is towards maintaining the current head direction. [Geist et al. 87] suggests that V(0.2) provides a good balance between the average response time and the starvation resistance.

2.3.2 Rotational positioning scheduling algorithms

In addition to seek time, rotational positioning algorithms also utilize rotational latency in the disk scheduling decision. The Shortest Access Time First algorithm (SATF) [Jacobson et al. 91], one of the most important rotational positioning algorithms, uses access time as the scheduling parameter. The method of SATF is presented in the remainder of this section.

Access time is a function of the seek time and rotational speed. Seek time is a function of seek distance in number of cylinders as described in Section 2.2, and rotational speed can be represented as a number of sectors per second if the numbers of sectors per track are known. Given a head position, the access time for any location on the disk can be calculated from this function. Figure 2 shows an example of access time as a function of cylinder and sector offsets. Such an access time pattern is used for scheduling in SATF, and different head positions present different access time patterns.

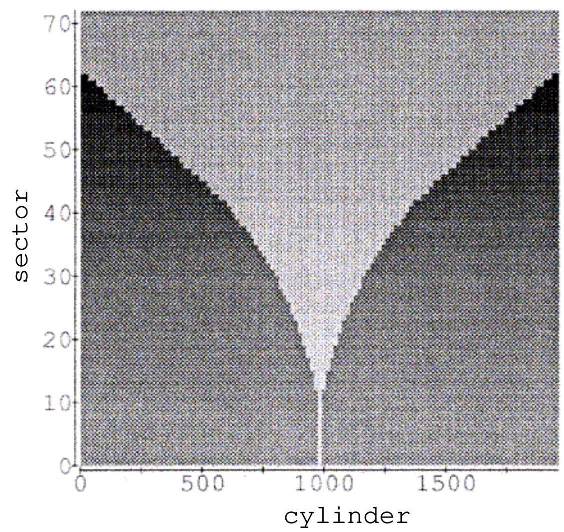


Figure 2: Access time as a function of cylinder and sector from a given head position, which is at sector zero and around cylinder one thousand. The times are uniform horizontally in each region. The lighter shaded region is accessible in one revolution time. (Source: [Jacobson et al. 91])

In this method, the disk is divided up into a number of bins, and commands are grouped based on their locations (bins) in the command queue. When deciding which command to serve next, the bins are scanned in an order depending on the access time graph which is determined by the head position. Figure 3 shows a scheduling example. The scanning orders of bins are pre-computed and stored in static arrays. Each head position determines a scanning order so an array indexed by head position is needed.

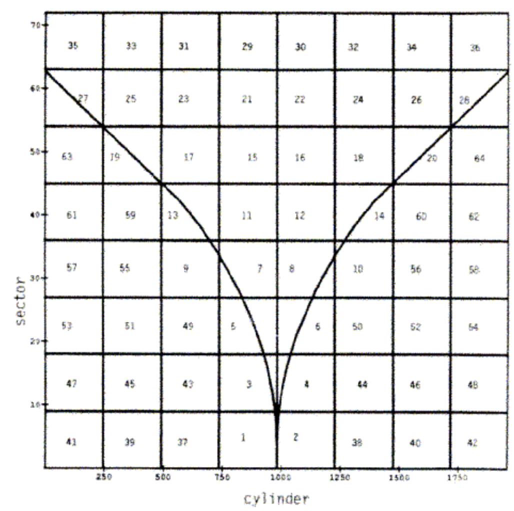


Figure 3: The disk is divided into bins with an example ordering . (Source: [Jacobson et al. 91])

As shown in the Figure 3, bins may be partly in both the region which can be accessed in one revolution and the region for which more than one revolution is needed. In order to deal with the scenario that the first non-empty bin contains only commands outside the one revolution region but the second non-empty bin contains commands which can be accessed in one-revolution, a threshold value is pre-computed and stored for each bin. The scanning terminates when it finds a command which is within the threshold value of the bin in which it is located.

The performance of SATF is superior to all of the seek time based algorithms . However , it needs a significant amount of memory space to store the scanning orders in arrays.

2.3.3 Method for reducing rotational latency

A rotational positioning scheduling algorithm was proposed by [Heath et al. 96] to reduce the storage and computing overhead of SATF. The basic idea is to service commands located in the area that can be reached in one revolution time, as shown in Figure 4.

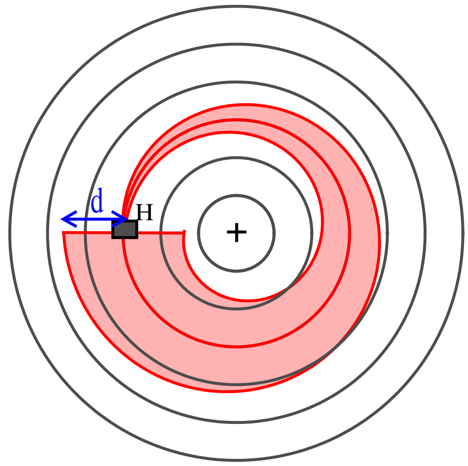


Figure 4: A rotating disk configuration. The disk is rotating in a counter-clockwise direction. The head is shown as H, and d is the distance that the read/write head can move in a single revolution time. The red area can be reached by the head in one revolution time.

The method is based on dividing the disk into a plurality of discrete angular regions, each of which is identified by a servo identifier (sid) at the beginning of each region. The angular regions are illustrated in Figure 5.

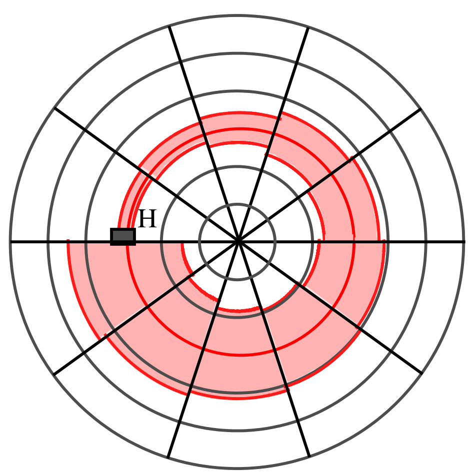


Figure 5: The disk is divided up into ten angular regions as an example.

Given a specific head position, each angular region has an area which can be traversed by the head in one revolution time and is specified by the seek range of the head in number of cylinders. The location of each request is converted into a head, angular region, and cylinder address for scheduling. When deciding which command to serve, the scheduler starts from the angular region rotationally closest to the head, and searches for a command in the queue within the seek range of one revolution time. If no command can be found, the scheduler moves to the next angular region and scans the command queue again. After all the angular regions are considered, the scheduler adds one more revolution to expand the seek range and repeats the searching again until the next command is found.

Determining whether a command is inside the seek range is an important implementation issue. [Heath et al. 96] proposes two approaches. The first is to pre-compute the seek range for each region and build a static array indexed by region number containing the seek range in number of cylinders. The size of the array is the number of angular regions on a platter. The region number is determined by the position of the head. The region rotationally closest to the head has region number zero, and increases clockwise when the disk platters rotate counter-clockwise. When searching the queue, the scheduler can simply check the array and compare the seek distance of the command with the value in the array.

Another approach is to compute the seek time of the command when scheduling and compare the seek time with the rotational latency determined by the rotational distance of the command from the head. The command is served if the seek time is less than the rotational latency. To convert the seek distance into the seek time efficiently, a static array is built for short seek distances and an affine equation is used to compute long distances. This method requires less computation time and performs better than the first method.

Variants of this scheduling algorithm are used in state-of-the-art disk drives, including the model disk in this study. However, the implementation of the RPO algorithm used in this study uses a more "realistic" approach for determining whether a command is inside the seek range. The following chapter describes the algorithm implementation and its approach to this problem in more detail.

Chapter III: Simulator Specifications

The disk simulator accurately models the IBM Ultrastar 18LZX disk drive including the physical components, data layout, and the scheduling algorithm which is based on the rotational scheduling algorithm introduced in Section 2.3.3. The mechanical model includes revolution time, number of heads and platters, number of cylinders, number of zones, sectors per track, and tracks per cylinder. Also, the model maps virtual block addresses into physical addresses made up of cylinders, heads, sectors and angular regions as specified by the scheduling algorithm. The primary purpose of the simulator is to examine the performance of the rotational algorithm for the model disk. The throughput, or disk operating rate, is the metric used to evaluate the performance of the scheduling algorithm.

Since the main purpose is to understand the role of rotational latency in scheduling, some modern disk features are simplified, such as on-board caches, prefetching etc. Starvation is not considered by this simulator, but can be accommodated utilizing a timing mechanism as the real disk does. However, this issue is beyond the scope of this thesis.

3.1 Device Specifications

The simulator models an IBM Ultrastar 18LZX disk drive whose specifications are summarized in Table 1 [IBM 2000].

Table 1: Specifications of IBM Ultrastar 18LZX

Form factor	3.5 inch
Capacity	18.3GB
Platters	5
Heads	10
Rotation rate	10,000 RPM
Servo identifiers per cylinder	90
Rotational latency (avg)	3.0 msec
Average seek	4.9 msec
Areal density (max)	3,534 Mbits/inch^2
Recording density (max)	260 KB/inch
Track density	13,595 tracks/inch
Sector size	512 to 688 bytes
Number of cylinders	11,712
Number of zones	15
Sectors per track	195 to 382
Media data rate	23.3 to 44.3 MB/sec

3.2 Workload and Parameters

3.2.1 Workload

The simulator uses random workloads that assumes that requests are uniformly distributed across the LBA space. Real-world trace workloads are left for future work. Each operation is created and issued by specifying its logical block address generated by a random number generator with a user-specified seed. Only small accesses are modeled in the experiments. More precisely, each operation only accesses data within a single logical block, and also a single physical sector. Thus, the starting logical block address of an operation will also be the address of the head position at the end of this operation. Long seeks are not considered because this study is only concerned with the access time between pair of requests and data transfer rate is not included as a factor.

3.2.2 Parameters

The disk simulator has the following configurable parameters.

- Queue length: The maximum number of commands that can be in the command queue at one time.
- Number of operations: The total number of commands being executed.
- Logical Block Address (LBA) range: This constrains the location of the operations. Two numbers, low LBA and high LBA, are used to specify the range. LBA addresses of operations are located between low LBA and high LBA.
- Type of operation: Whether the operation is read or write.
- Random number seed: This is used by the random number generator which is used to create the logical block addresses.

The queue is filled with operations at the beginning of the simulation, and maintained full until the number of remaining operations are less than the queue size. During the simulation, one operation is generated and put into the queue when one finishes executing. The simulator creates and services the total number of operations to complete the simulation.

A minimum operating time is necessary to fulfill each request that includes all the controller overhead. The default minimum operating times are 1.06 msec for read operations and 1.6 msec for write operations.

The LBA range determines the longest possible seek distance between consecutive operations. The effects of the seek distance in system performance is explored by setting up experiments involving different seek ranges.

3.3 RPO Scheduling Algorithm

The algorithm used in the simulator is based on the model disk and the rotational positioning algorithm described in Section 2.3.3. The disk is divided into 90 discrete angular regions, and each region encompasses a four degree “pie-shaped” area on each platter.

In Figure 5, the area which can be reached in a single rotation time is separated by a line from the area which cannot be reached. However, in practice it is difficult to determine which area borderline commands are in, and a more sophisticated strategy is desirable to estimate whether an operation can be reached. The strategy of deciding if a location can be reached or not is essential in the implementation. The cost of making a wrong decision is one additional revolution, so the performance can be even worse than picking the next command at random. A conservative strategy is to only consider the commands that can reliably be served in one revolution time, in other words, abandoning all the commands near the border of the shaded area in Figure 5. The problem is that being too conservative reduces the benefits of the scheduling algorithm. On the other hand, if the strategy is too aggressive, the performance is more likely to suffer from the missed revolution.

The approach in the model is to use statistical information to make the scheduling decisions, and the statistical data are measured, pre-computed, and stored on the drive for access by the scheduling algorithm. RPO (Rotational Positioning Optimization) tables are the data structures used to store this information, and are a key component of the simulator .

3.3.1 RPO tables

RPO tables are pre-computed and stored in the disks for estimating the probability of successful seeks .

Cylinder groups are used as the index into RPO tables , and are defined from the relationship between seek time versus seek distance shown in Figure 6. As described in Section 2.2, it shows a square root curve for short seeks , and for longer seeks it becomes an affine relation. Seek time T in the figure denotes the time to tra verse all the cylinders in the disk. There are 227 cylinder groups in the model. The first cylinder group contains the seek distance from 0 to the seek distance corresponding to seek time T/227 in the figure . The second cylinder group contains the seek distance corresponding to seek time range T/227 to 2T/227, and so on.

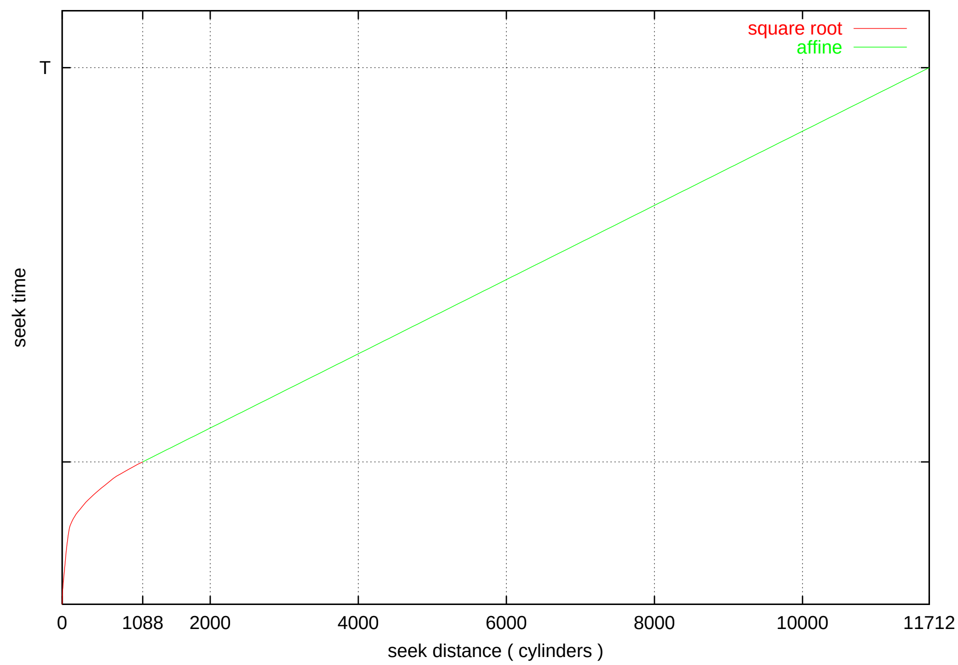


Figure 6: The relationship between seek time versus seek distance . There are 11,712 cylinders in the model disk and for seek distances greater than 1,088 cylinders there is an affine relationship .

For each cylinder group , an RPO table contains the expected rotational latency values measured from the model disk drive and the expected access time values are the average of a significant amount of measurement. The values are measured in number of angular regions (sids), which can be converted to relative time by multiplying the rotational speed and the elapsed time . In other words , the rotational distance in this context means the number of angular regions between the current location of the read/write head and the position of the next command. In particular , each cylinder group entry includes a higher bound value , a lower bound value and some expected access time values between. For a specific cylinder group , if the rotational distance is higher than the higher bound value , the head can traverse this seek distance with probability 1, if it is under the lower bound value , the probability is 0. For the rotational distances between and including the higher bound and the lower bound, the probability is between 0 and 1, and the expected access time values in the table determine the estimated access time .

The algorithm computing the estimated access time for each command in the queue is presented in Figure 7:

Step 1.	[CALCULATE D] $D \leftarrow$ *sid number of* N - *sid number of the current head position*
Step 2.	[TEST] if $D > 0$, go to Step 4
Step 3.	[PLUS ONE REVOLUTION] $D \leftarrow D$ + *number of regions in one revolution*
Step 4.	[TEST] if D > *minimum operating time* , go to Step 6
Step 5.	[PLUS ONE REVOLUTION] $D \leftarrow D$ + *number of regions in one revolution*
Step 6.	[TEST] if D > *higher bound* , go to Step 10
Step 7.	[TEST] if D < *lower bound* , go to Step 9
Step 8.	[CALCULATE D] $D \leftarrow EAT(D)$, go to Step 10
Step 9.	[CALCULATE D] $D \leftarrow D$ + *number of regions in one revolution*
Step 10.	[FINISHED] estimated access time of N is D

Figure 7: Algorithm for computing the estimated access time for command N.

From Step 1 to Step 3, the rotational difference in number of angular regions from the current head position [1] to the candidate command is calculated. Step 4 is to check if the rotational difference is less than the minimum operation time. If it is, one more revolution is needed, so we add the number of angular regions in one revolution to the estimated access time, which is 90 in the model disk. In Step 6, the RPO table is consulted and if the estimated access time is greater than the higher bound in the table, the computation is done. Otherwise, Step 7 compares it with the lower bound. If

1. In the simulation, only single sector disk accesses are considered and a constant queue length is maintained. The current head position denotes the cylinder associated with the last data accessed. This notion is used throughout the remainder of the thesis.

it is less than the lower bound, one more revolution is needed, so we add 90 to the estimated access time again, and it is assumed that all the locations on the platters can be reached at this point. For the case that the estimated access time is between the lower and higher bound, the RPO table is consulted again to get the expected access time (EAT), and it is processed in Step 8.

For example, a command has seek distance s cylinders and rotational distance r sids from the current head position. According to Figure 6, s corresponds to cylinder group g, so the entry indexed by g of the proper RPO table is used. This RPO table entry contains the higher and lower bound values h and l sids respectively, as well as the expected access time values t_1, t_2... $t_{(h-l)+1}$ sids. There are three possible cases:

- $r > h$: This command can be reached in r, so the estimated access time is the rotational latency r.
- $l \leq r \leq h$: If $r = i$, and $1 \leq i \leq (h-l)+1$, the estimated access time for this command is t_i.
- $r < l$: This command can not be reached in r, and at least one more revolution is needed. If r + *one revolution* is greater than l, the estimated access time is r + *one revolution.* If it is less than l, the estimated access time will be r + *two revolutions*, since any location can be traversed at any head position at this point. If r + *one revolution* is within the lower and the higher bound, r + *one revolution* is compared with expected access time values and which follows the step of the second case.

When considering which command to service next, the command queue is scanned and the access times for all the waiting commands are computed. The commands in the queue are maintained in their arrival order without being sorted. The logical block address of each command is converted into head, cylinder, sector and angular region values for calculating the seek distance and the rotational distance, and then the RPO table is used to obtain the estimated access times. The command with the minimum estimated access time will be served next. After serving each request, the estimated access time for every command in the queue is re-computed for the next schedule.

For the purpose of accuracy, there are a set of RPO tables in a disk drive, R-U, R-D, W-U, and W-D, each of which is used for different operation types. Two parameters are used to determine the operation type and specify four types of tables. The first parameter specifies whether the operation is reading data from the disk (R) or writing data to the disk (W). The second parameter specifies whether the head is moving toward the center (U) or the edge of the platters (D). To estimate the approximate storage space needed for the RPO tables in a drive, each cylinder group includes around 30 values on average, and one table contains 227 cylinder groups. There are four tables in a set, so the total size is around 27,240 bytes. A scheme of reducing the RPO table space is discussed in next chapter.

Chapter IV : Reducing Storage Space of the RPO Tables

The values in the RPO tables play a key role in the implementation of the rotational scheduling algorithm, so several efforts have been made to make the values more accurate and decrease the probability of making wrong decisions. At the same time, saving storage space for RPO tables is beneficial considering the expense of the memory space. This chapter discusses and examines two schemes concerning the trade-off between the RPO table space and the performance of the scheduling algorithm:

- Changing the RPO head positioning data representations: Regarding the mechanical components, the head movement is controlled by the arms and disk arms are attached to a common actuator in different positions. When the actuator rotates, the strength received by each arm is not precisely identical, and this slight variance may cause different movements for different heads. To improve the accuracy of RPO tables, it has been suggested to use a particular set of RPO tables for each of the heads in a disk drive, and each table characterizes a particular head. This approach increases the necessary memory space from 27KB to 270KB for the model disk. However, it is still not clear how important this mechanical effect is, and it increases the storage overhead in proportion to the number of heads.
- Minimizing the space of RPO tables: A strategy for minimizing the storage used for predicting the head movement is to find out the best threshold value between the lower bound and the higher bound, then the number of

values needed can be dramatically decreased to only one threshold value for each cylinder group. The challenge is to find out the appropriate threshold, or the best degree of the aggressiveness of the scheduling algorithm in terms of system performance.

The approaches used to simulate both schemes are described in detail in the following section, and the trade-off between storage space and the scheduling performance is analyzed from the system throughput reported by the simulator.

4.1 Approaches

4.1.1 Head positioning data representations

The model disk has five platters and ten read/write heads. From the top to the bottom, the heads can be numbered from 0 to 9. Head 0 is on top of the first platter, and its arm is independently attached to the actuator. The arms of head 1 and head 2 are joined and attached to the actuator together, the arms of head 3 and head 4 are joined together, and so on. Head 9 is under the last platter, and it is attached to the actuator independently, the same as head 0. It is reasonable to assume that head 0 and head 9 have the same behavior because these two heads are symmetrical on the two ends of the actuator. Similarly, head 1, head 2, head 7 and head 8 have the same behavior, and head 3, head 4, head 5, and head 6 have the same behavior. One set of RPO tables is used for each group to save storage space. In this experiment, the set of RPO data measured from head 0 is used for the group containing heads 0 and 9, the data measured from head 2 is used for the group containing heads 1, 2, 7 and 8, and the data measured from head 5 is used for the group containing the rest of the heads.

4.1.2 Optimizing the data representations in RPO tables

Figure 8 depicts the rotational distance values in an entry of an RPO table .

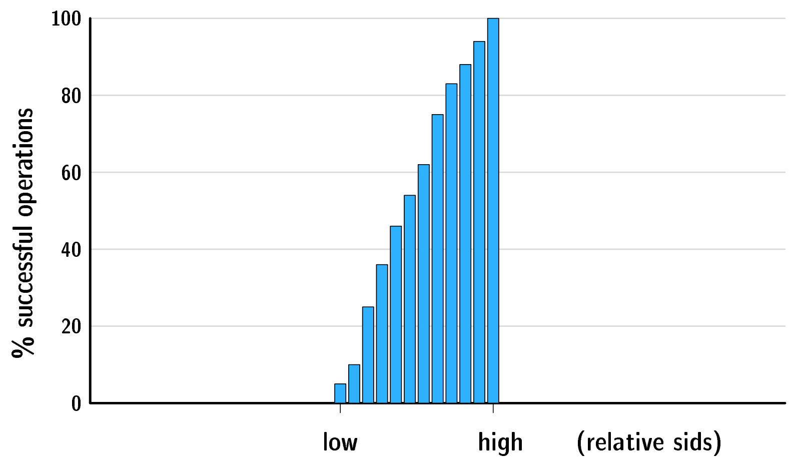


Figure 8: Probability of successful operations versus rotational latency . The graph represents one cylinder group entry in an RPO table , and it shows the probability of successful accesses according to each rotational latency . Low on the x-axis is the lower bound value , and high is the higher bound value in the RPO table .

The goal as mentioned earlier is to find out the best value between the lower and the higher bound to use as the threshold. The value k in Figure 9 denotes the threshold, and the scheduling algorithm only serves commands with rotational latencies higher than k. A parameter α is defined to specify the aggressiveness of the scheduling strategy , and k is determined from α .

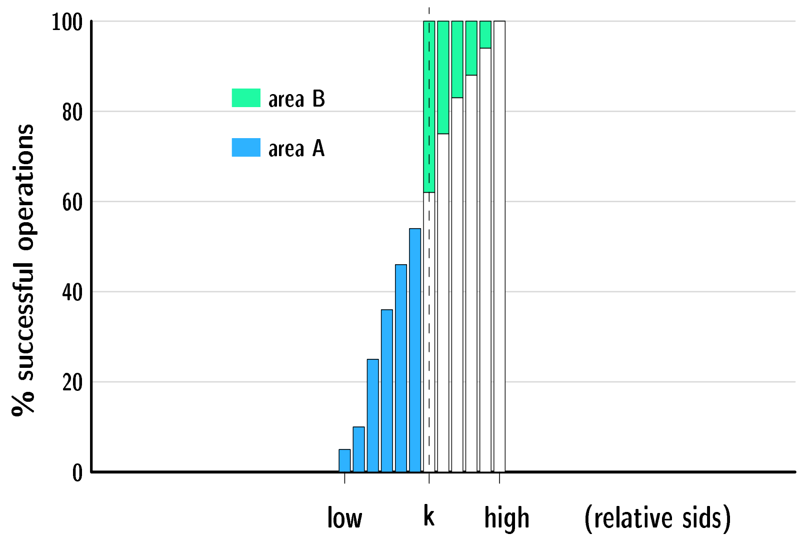


Figure 9: The definition of α. Area A and area B are determined by the threshold value k, and α is defined as *(area A/area B)* to specify the degree of aggressiveness.

α is defined as $\frac{areaA}{areaB}$ in the figure, and

$$\text{area A} = \sum_{i = low}^{k-1} f_i$$

$$\text{area B} = \sum_{i = k}^{high} (100 - f_i)$$

where f_i is the percentage of successful operations for the i sids rotational latency.

The smaller α is, the more aggressive the scheduling strategy is. How α affects the disk performance is discussed later.

4.2 Simulation Specifications and Results

Five different LBA ranges listed in Table 2 are used in the experiments.

Table 2: LBA ranges in the experiments

LBA Range	LBA range	Zone number
R1	0 to 200,000	0
R2	0 to 4,000,000	0
R3	0 to 12,000,000	0 to 1
R4	0 to 24,000,000	0 to 6
R5	0 to 36,000,000	0 to 14

The LBAs of operations are created randomly within the specified LBA range. R1 represents small seeks so the head movement is constrained to a small range. R2 represents slightly larger distance seeks, however, as in R1, destination addresses only reside in zone 0, the outermost part of the platters. The operation addresses in R3 can reside in either zone 0 or 1, so the head can move across two zones during a seek. Seeks in R4 can be as large as half of the full stroke distance. In R5, the operation addresses can be anywhere on the platters, so seek distances can be any possible value, from very small seeks to full-stroke seeks.

In this experiment, the queue size is fixed at 32, the standard specification of the model disk. The effects of queue sizes are examined in next section. Several experimental groups of head positioning data representations are simulated for comparison, and the groups are summarized as follows:

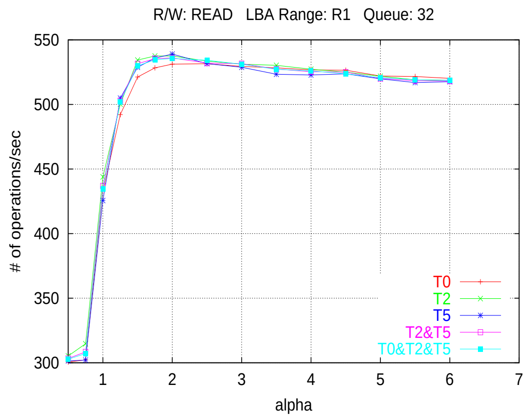
- T0: Table 0 is used for head 0 1 2 3 4 5 6 7 8 9.
- T2: Table 2 is used for head 0 1 2 3 4 5 6 7 8 9.
- T5: Table 5 is used for head 0 1 2 3 4 5 6 7 8 9.

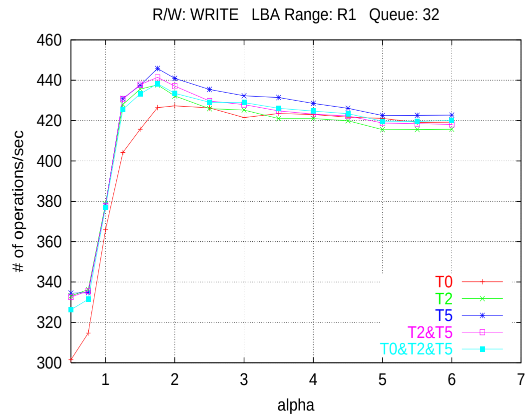
- T2 & T5: Table 2 is used for head 0 1 2 7 8 9; Table 5 is used for head 3 4 5 6.
- T0 & T2 & T5: Table 0 is used for head 0 9; Table 2 is used for 1 2 7 8; Table 5 is used for head 3 4 5 6.

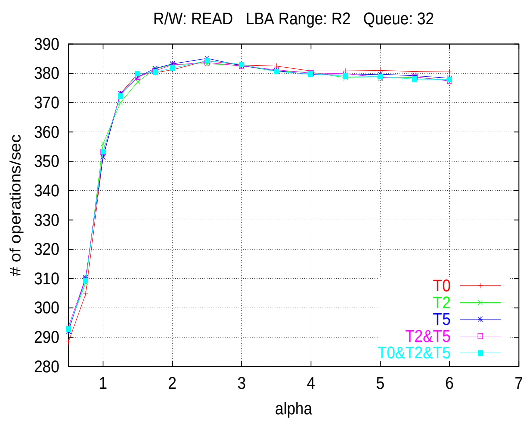
Figure 10 shows the throughput in number of operations per second as a function of α for these experimental groups for different simulation parameters. Both read and write operating rates decrease when the seek ranges increase. The operating rate for write operations is significantly higher than read operations, and the differences are very similar for different seek distances.

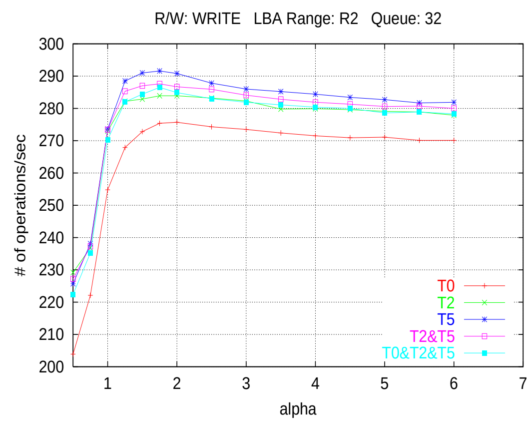
Operating rates increase dramatically when α increases from 0.5 to around 2, and decrease slightly after 2, and the decrease is more significant for write operations. The highest throughput for read operations can be achieved when α is 1.75 to 2.5, and 1.5 to 1.75 is best for write operations. The curves are approximately the same for different LBA ranges.

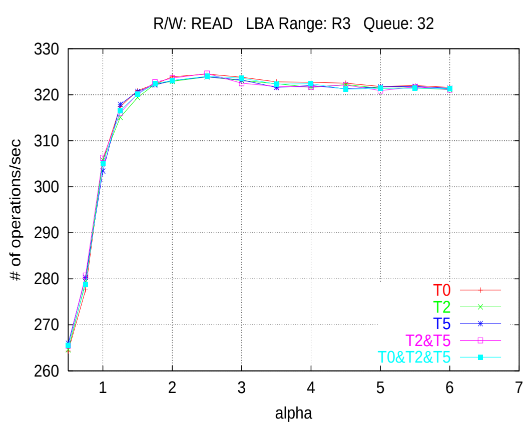
It is interesting that the head positioning data representations actually have very little effect on the performance of the read operations. The effect is more evident for write operations, but it is not significant. As expected, the experiment group T0 performed poorly. Other groups have very similar performance. Surprisingly, the group of T0 & T2 & T5 doesn't show the best performance. On the other hand, T5 outperforms the others especially for short seeks. The possible reason for this is that the experimental error of the RPO data measurement is higher than the differences of the moving behaviors of different heads. At the same time, I cannot exclude the possibility that the RPO tables have errors in measurements.

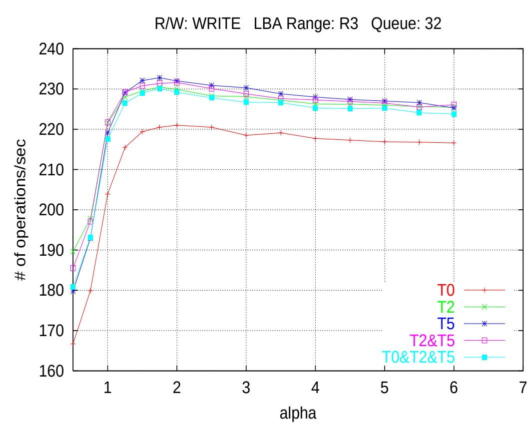

R/W: READ LBA Range: R1 Queue: 32
# of operations/sec
alpha
T0
T2
T5
T2&T5
T0&T2&T5


R/W: WRITE LBA Range: R1 Queue: 32
# of operations/sec
alpha
T0
T2
T5
T2&T5
T0&T2&T5


R/W: READ LBA Range: R2 Queue: 32
# of operations/sec
alpha
T0
T2
T5
T2&T5
T0&T2&T5


R/W: WRITE LBA Range: R2 Queue: 32
# of operations/sec
alpha
T0
T2
T5
T2&T5
T0&T2&T5


R/W: READ LBA Range: R3 Queue: 32
# of operations/sec
alpha
T0
T2
T5
T2&T5
T0&T2&T5


R/W: WRITE LBA Range: R3 Queue: 32
# of operations/sec
alpha
T0
T2
T5
T2&T5
T0&T2&T5

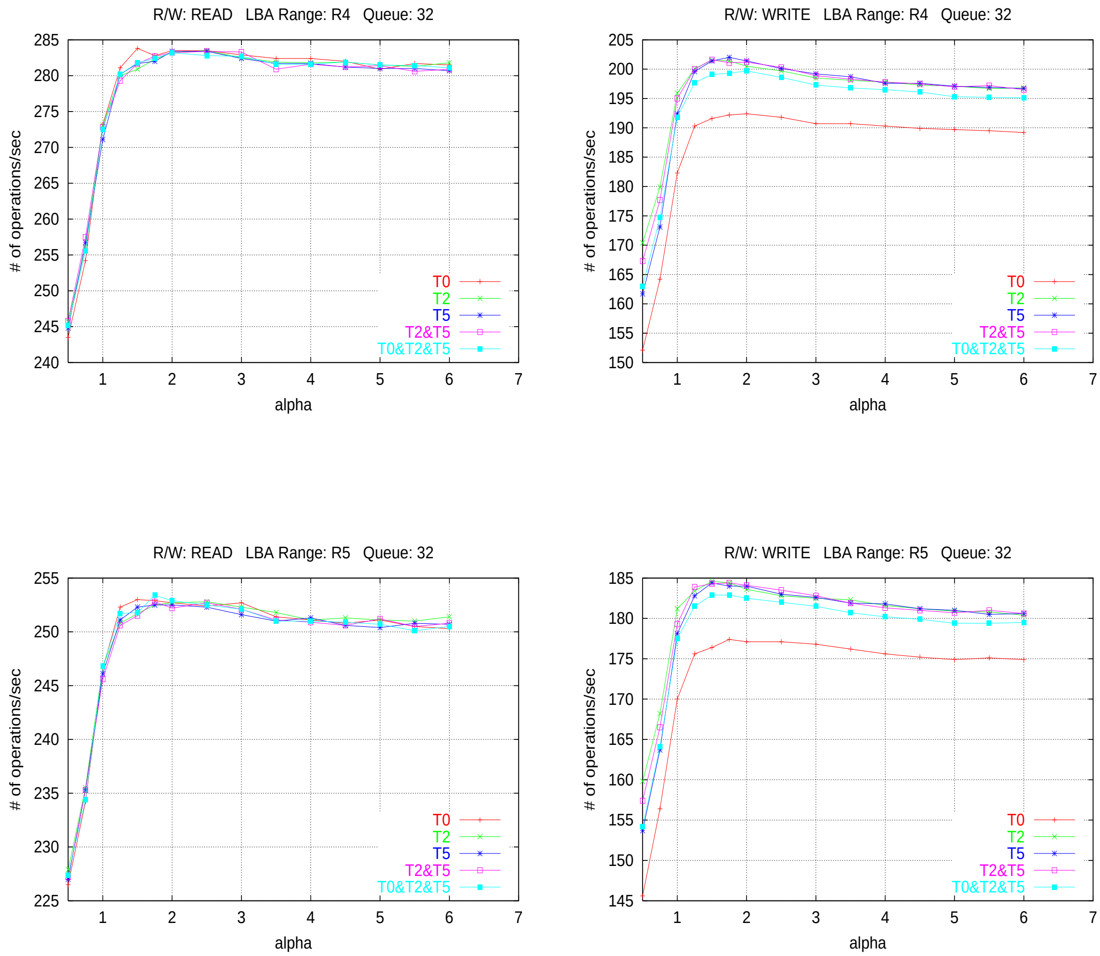


Figure 10: Performance of different head positioning data representations .

Now let's only look at the simulation results for head positioning data group T2, in which all the heads use the set of RPO tables measured from head 2. Results for other groups are very similar .

Figure 11 compares the performance of different seek ranges for read and write operations with queue size 32.

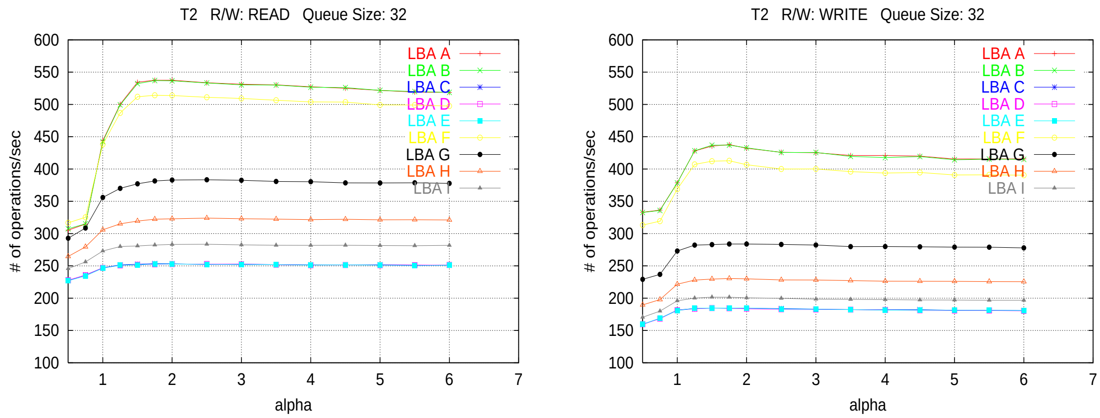
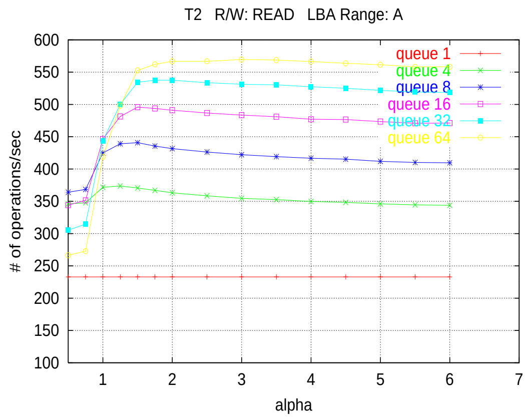
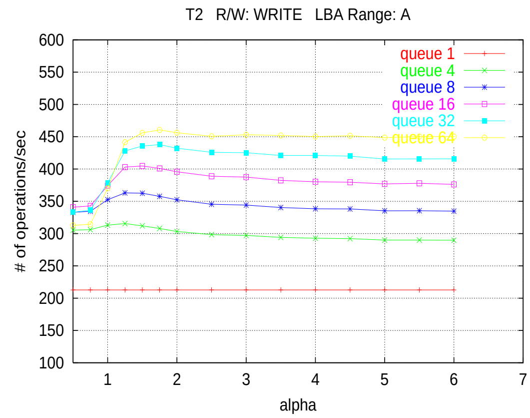


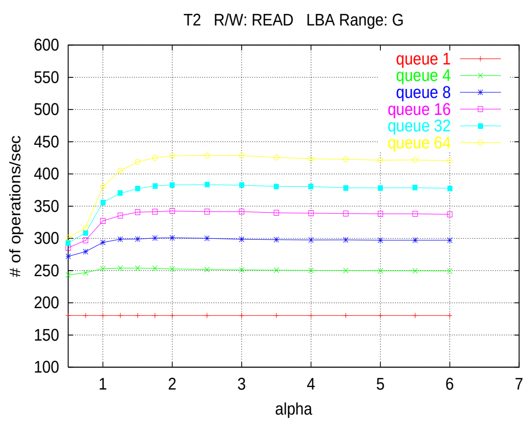
Figure 11: Performance of different seek ranges .

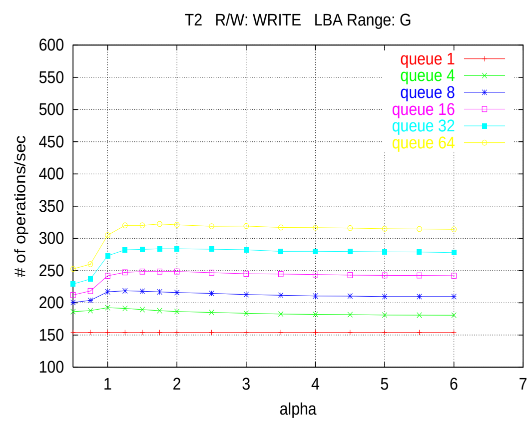
Figure 12 compares the throughput as a function of α for different queue sizes with the same LBA ranges . The size of the queue is an important factor of scheduling performance . The larger the queue is, the more we can benefit from the scheduling . In the experiments , queue size 1 is the same as FCFS scheduling and used for comparison. As shown from the plots , queue size 64 has the most significant improvement when the value of α is increased from 0.5 to approximately 2, and the improvement reduces with smaller queue sizes . It is also noticeable that the best α value changes with the queue size . For read operations , the best α values are from 1.25 to 3.0 for queue size from 4 to 64. For write operations , the best α values are from 1 to 1.75. The best α values are summarized in Table 3 and Table 4.

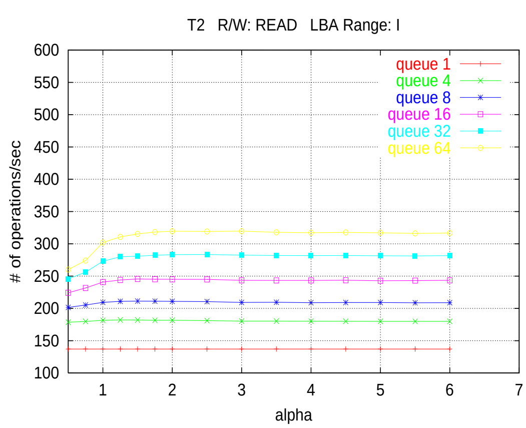
In terms of the best throughputs , when doubling the queue size the throughput rate grows constantly , about 30 to 50 operations per second for read operations , and a little less for write operations .

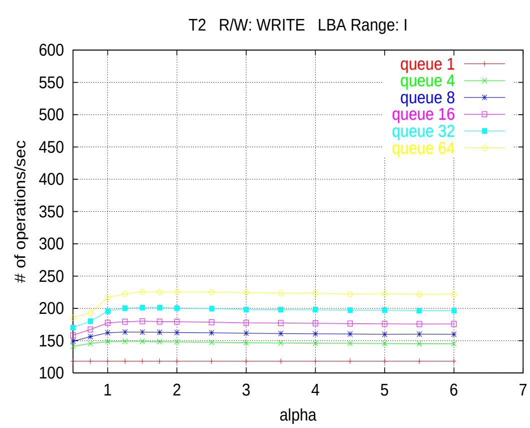

T2 R/W: READ LBA Range: A
# of operations/sec
alpha
queue 1
queue 4
queue 8
queue 16
queue 32
queue 64


T2 R/W: WRITE LBA Range: A
# of operations/sec
alpha
queue 1
queue 4
queue 8
queue 16
queue 32
queue 64


T2 R/W: READ LBA Range: G
# of operations/sec
alpha
queue 1
queue 4
queue 8
queue 16
queue 32
queue 64


T2 R/W: WRITE LBA Range: G
# of operations/sec
alpha
queue 1
queue 4
queue 8
queue 16
queue 32
queue 64


T2 R/W: READ LBA Range: I
# of operations/sec
alpha
queue 1
queue 4
queue 8
queue 16
queue 32
queue 64


T2 R/W: WRITE LBA Range: I
# of operations/sec
alpha
queue 1
queue 4
queue 8
queue 16
queue 32
queue 64

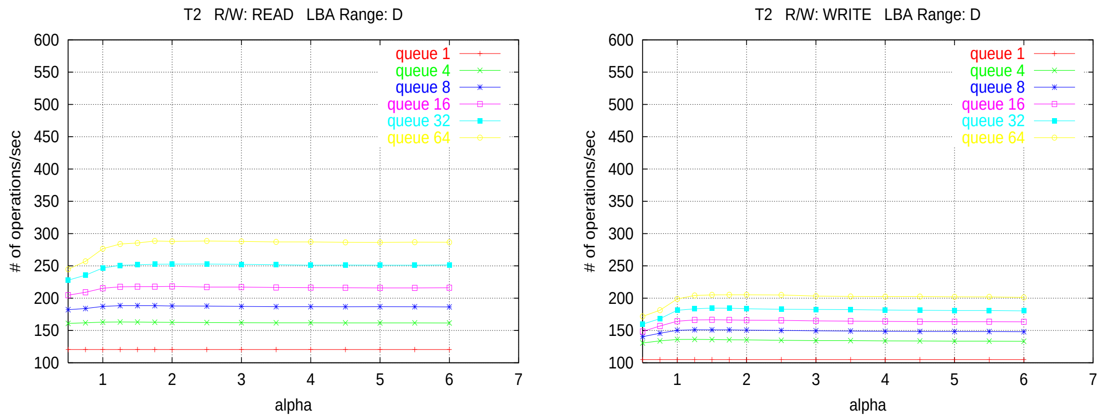


Figure 12: Performance of different queue sizes with the same LBA ranges.

Table 3 and Table 4 summarize the best α value for each experimental condition. The throughput is compared with the original model to observe the performance degradation by using a threshold to decrease the RPO table space. In the tables, the throughput marked "real" is reported from the original model, and the differences show the percentage the throughput degraded when using the threshold. The degradation is more significant for smaller seek ranges and larger queue sizes.

Table 3: Best throughput (no. of operations / sec) T2-READ

	Queue Size = 1	Queue Size = 4	Queue Size = 8	Queue Size = 16	Queue Size = 32	Queue Size = 64
R1	-- / 233.0 real: 233.0 diff. 0.00%	1.25 / 373.7 real: 378.8 diff. 1.35%	1.5 / 440.7 real: 449.0 diff. 1.85%	1.5 / 495.8 real: 507.2 diff. 2.25%	1.75 / 537.5 real: 557.4 diff. 3.57%	3.0 / 569.5 real: 592.9 diff. 3.95%
R2	-- / 180.4 real: 180.4 diff. 0.00%	1.25 / 253.8 real: 256.3 diff. 0.98%	2 / 300.9 real: 305.4 diff. 1.47%	2 / 342.5 real: 349.8 diff. 2.09%	2.5 / 383.3 real: 398.1 diff. 3.72%	2.5 / 428.6 real: 447.1 diff. 4.14%
R3	-- / 156.7 real: 156.7 diff. 0.00%	1.25 / 209.6 real: 211.1 diff. 0.71%	1.75 / 246.2 real: 249.2 diff. 1.20%	2 / 285.3 real: 289.9 diff. 1.59%	2.5 / 323.9 real: 331.1 diff. 2.17%	2.5 / 360.7 real: 372.2 diff. 3.09%
R4	-- / 137.1 real: 137.1 diff. 0.00%	1.25 / 182.2 real: 183.0 diff. 0.44%	1.5 / 211.3 real: 213.0 diff. 0.80%	1.5 / 245.6 real: 249.3 diff. 1.48%	2.5 / 283.5 real: 288.3 diff. 1.66%	3.0 / 319.6 real: 325.8 diff. 1.90%
R5	-- / 120.4 real: 120.4 diff. 0.00%	1.25 / 163.2 real: 164.0 diff. 0.49%	1.25 / 188.4 real: 189.9 diff. 0.79%	2 / 218.2 real: 220.6 diff. 1.09%	2.5 / 252.8 real: 256.1 diff. 1.29%	1.75 / 288.6 real: 294.3 diff. 1.94%

Table 4: Best throughput (no. of operations / sec) T2-WRITE

	Queue Size = 1	Queue Size = 4	Queue Size = 8	Queue Size = 16	Queue Size = 32	Queue Size = 64
R1	-- / 212.7 real: 212.7 diff. 0.00%	1.25 / 315.5 real: 320.5 diff. 1.56%	1.25 / 363.0 real: 372.1 diff. 2.45%	1.5 / 404.7 real: 419.9 diff. 3.62%	1.75 / 437.7 real: 460.5 diff. 4.95%	1.75 / 460.5 real: 489.6 diff. 5.94%
R2	-- / 153.9 real: 153.9 diff. 0.00%	1 / 192.4 real: 197.3 diff. 2.48%	1.25 / 218.6 real: 228.4 diff. 4.29%	2 / 248.6 real: 260.7 diff. 4.64%	1.75 / 283.9 real: 299.5 diff. 5.21%	1.75 / 322.4 real: 338.6 diff. 4.78%
R3	-- / 133.4 real: 133.4 diff. 0.00%	1 / 165.0 real: 169.4 diff. 2.60%	1.25 / 182.2 real: 189.4 diff. 3.80%	1.5 / 203.6 real: 213.5 diff. 4.64%	1.75 / 230.5 real: 241.5 diff. 4.55%	1.75 / 260.5 real: 273.7 diff. 4.82%
R4	-- / 118.2 real: 118.2 diff. 0.00%	1.25 / 149.1 real: 151.7 diff. 1.71%	1.25 / 163.4 real: 168.8 diff. 3.20%	1.5 / 180.2 real: 187.6 diff. 3.94%	1.5 / 201.5 real: 211.0 diff. 4.50%	1.5 / 225.8 real: 237.3 diff. 4.85%
R5	-- / 104.8 real: 104.8 diff. 0.00%	1.25 / 136.2 real: 137.9 diff. 1.23%	1.25 / 151.0 real: 154.7 diff. 2.39%	1.5 / 166.6 real: 172.4 diff. 3.36%	1.5 / 184.6 real: 191.6 diff. 3.65%	1.75 / 205.2 real: 215.0 diff. 4.56%

Chapter V : The Distribution of Rotational Displacements of Successive Operations

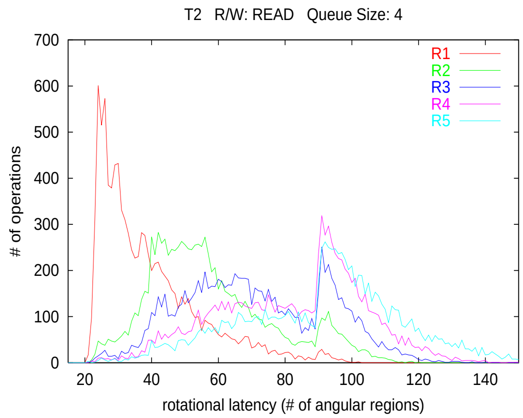
The distribution of rotational displacements of successive commands gives an insight into the behavior and the performance of the RPO algorithm. It provides another point of view to study the roles of the factors such as queue size and seek range in this algorithm. This chapter discusses the distribution of the rotational displacements of successive commands and how the RPO scheduling algorithm causes this distribution.

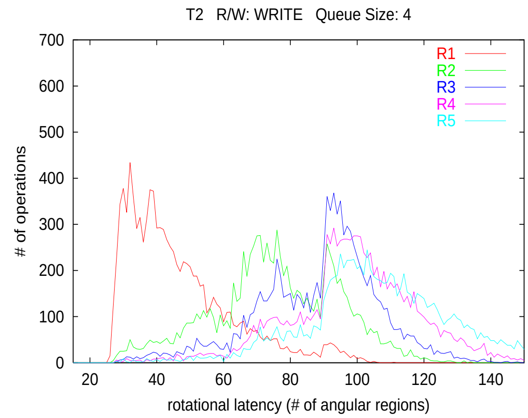
5.1 The Distribution

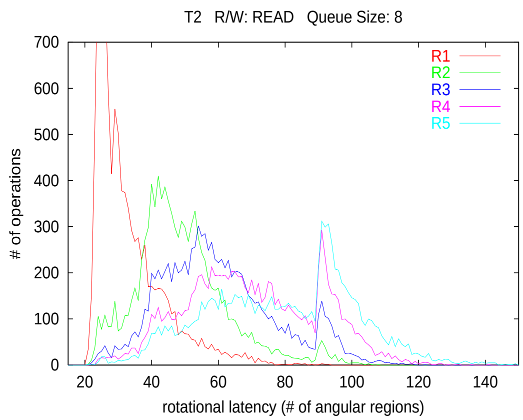
Figure 13 shows the distribution of the rotational displacements for all successive commands. Each set of plots shows the distribution for a queue size and an operation type for different LBA ranges.

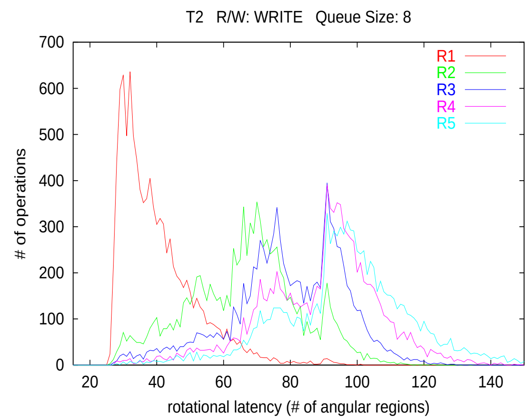
The average access time of write operations is longer than that of read operations. The effects of the seek range and the size of the queue are validated and correspond to my observation in previous chapter. Rotational delay is apparently less for smaller seek ranges. When the queue size increases, the distribution is significantly concentrated to the lower rotational latency.

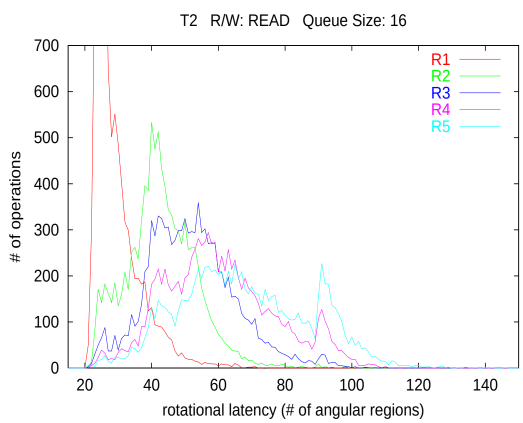
An interesting common characteristic is noticed in the distribution for all the experiments that there is always a blip right after 90 angular-region time. More precisely, the distribution shows a local maximum when the access time is 91 angular regions.

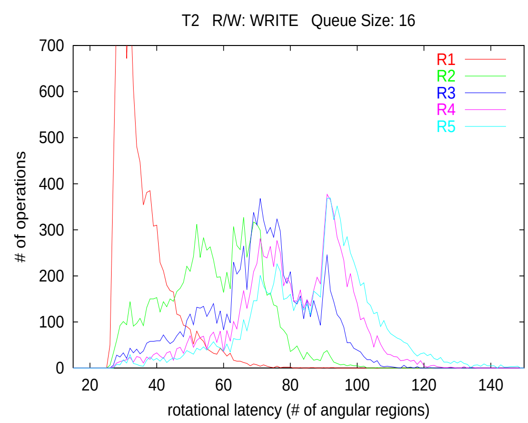

T2 R/W: READ Queue Size: 4
R1
R2
R3
R4
R5
700
600
500
400
300
200
100
0
20
40
60
80
100
120
140
# of operations
rotational latency (# of angular regions)


T2 R/W: WRITE Queue Size: 4
R1
R2
R3
R4
R5
700
600
500
400
300
200
100
0
20
40
60
80
100
120
140
# of operations
rotational latency (# of angular regions)


T2 R/W: READ Queue Size: 8
R1
R2
R3
R4
R5
700
600
500
400
300
200
100
0
20
40
60
80
100
120
140
# of operations
rotational latency (# of angular regions)


T2 R/W: WRITE Queue Size: 8
R1
R2
R3
R4
R5
700
600
500
400
300
200
100
0
20
40
60
80
100
120
140
# of operations
rotational latency (# of angular regions)


T2 R/W: READ Queue Size: 16
R1
R2
R3
R4
R5
700
600
500
400
300
200
100
0
20
40
60
80
100
120
140
# of operations
rotational latency (# of angular regions)


T2 R/W: WRITE Queue Size: 16
R1
R2
R3
R4
R5
700
600
500
400
300
200
100
0
20
40
60
80
100
120
140
# of operations
rotational latency (# of angular regions)

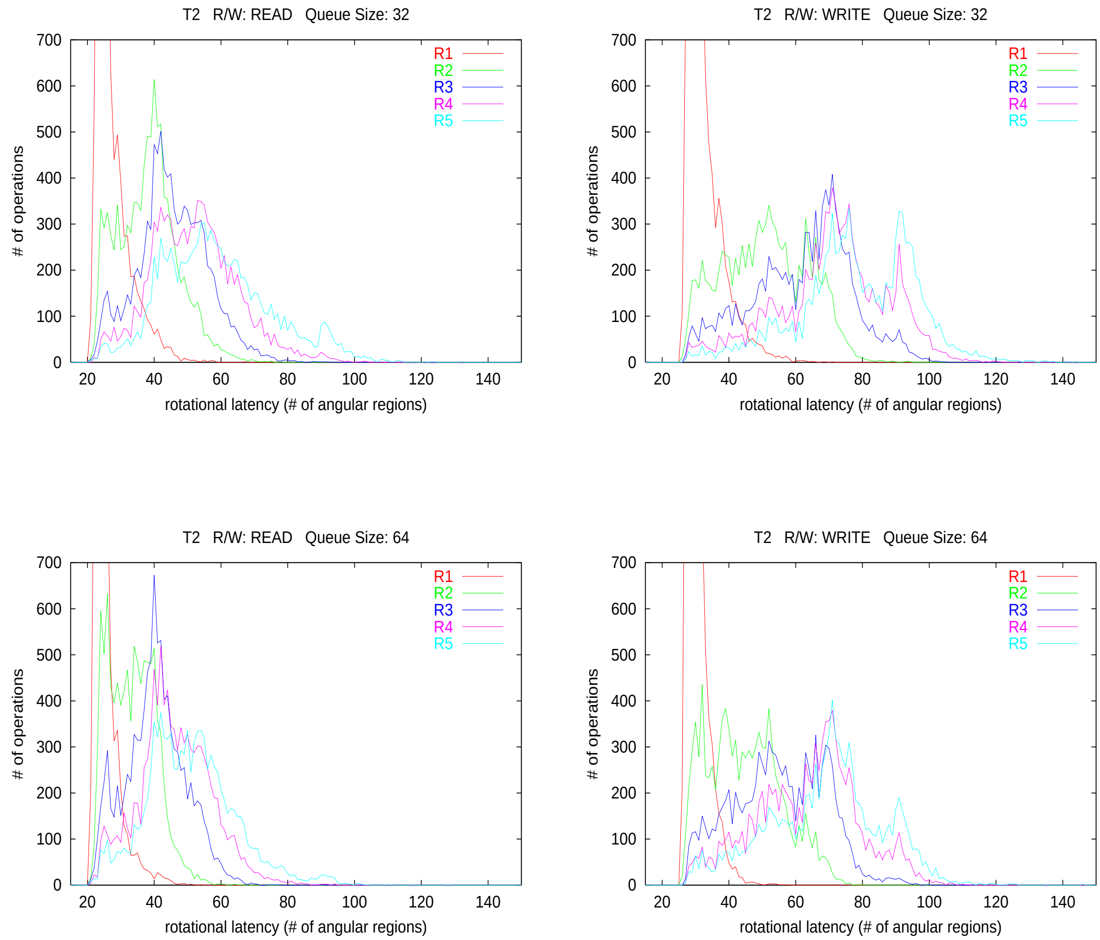


Figure 13: Effects of changing the queue size for different seek ranges. The head positioning data for head 2 is used for all the heads, and all seek ranges are considered. The total number of operations is ten thousand.

In order to understand this behavior, the procedures in the scheduling algorithm are analyzed and traced. Surprisingly, most of the successive operations at 91 angular regions or higher result from Step 5 in Figure 7. This effect is shown in Figure 14. This means that the original rotational differences of these operations with the current head position are less than the minimum operating times, and one more revolution was added to their estimated access times. In the simulator, the default minimum operating times are 16 angular-region time for read operations and 24 angular-region time for write operations. In Figure 14, the operations which had 90 angular regions added in Step 5 dominate the distribution higher than 90 angular regions, especially for read operations. Out of the total number of operations, 5.4% of successive read operations and 27.6% of successive write operations are originally less than minimum operation times.

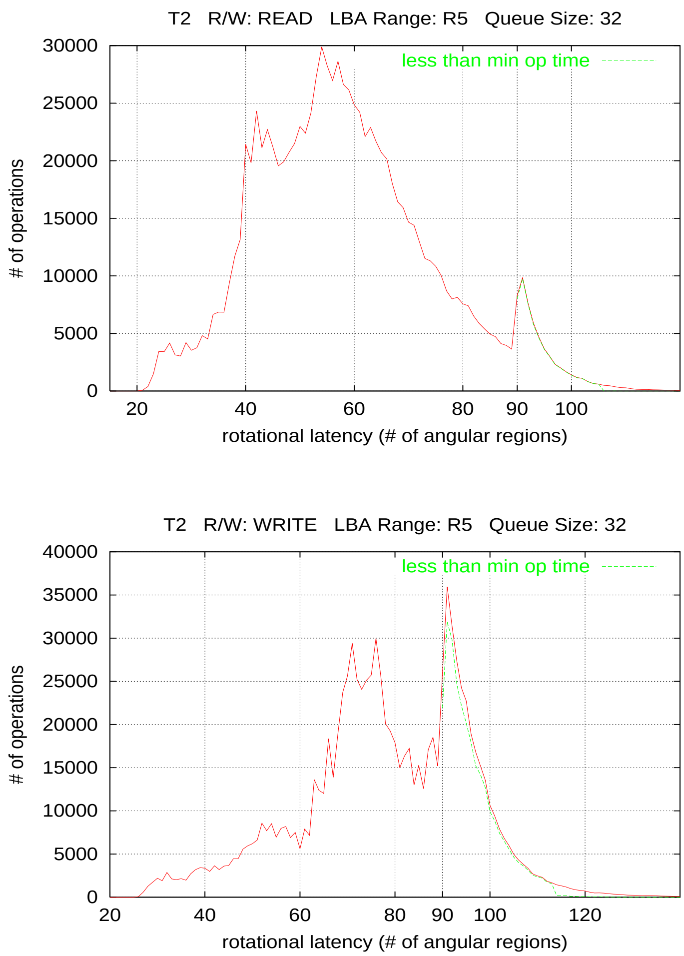


Figure 14: Effects of the minimum operating times. The operations whose rotational differences are originally less than minimum operation time, shown as a dashed line, dominate the distribution higher than 90 angular regions. The total number of operations is one million.

To explore this effect, different minimum operating times are simulated and Figure 15 shows the performance changes when increasing the minimum operating time of read operations from 16 sids to 24 and 32 sids, and write operations from 24 sids to 36 and 48 sids. Table 5 summarizes the percentage values of the distribution in Figure 15. The minimum operating times are specified in number of angular regions, and a single angular-region time is 0.067 msec.

The numbers have noticeable increases when the minimum operating times increase.

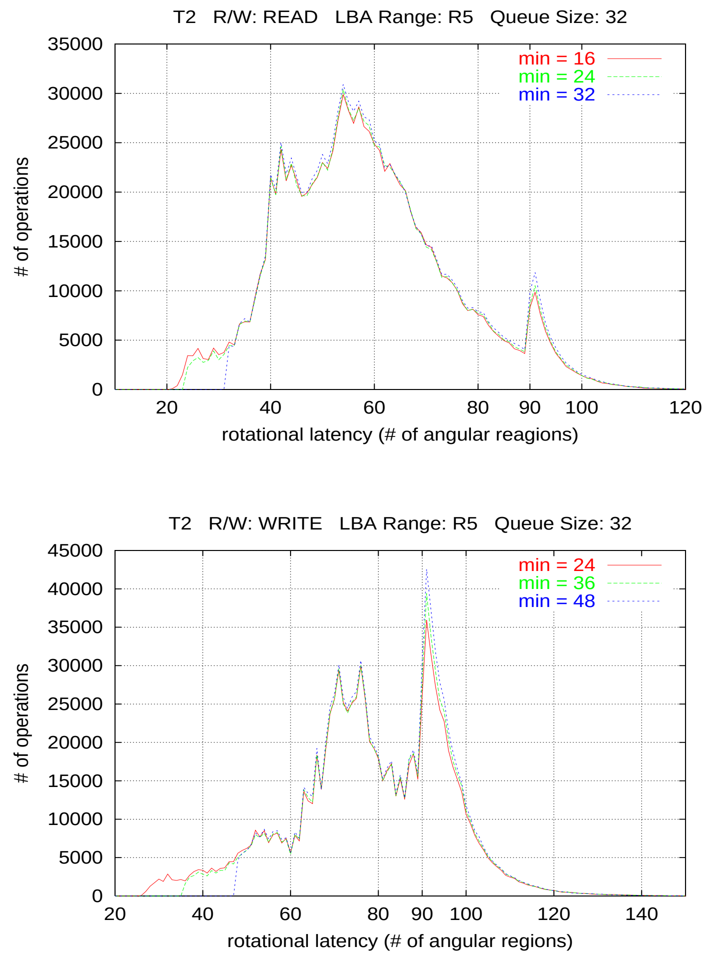


Figure 15: Effects of changing the minimum operating times . The total number of operations is one million.

Table 5: Effects of changing the minimum operating times.

Read / Write min. operating time (sids)	16 / 24	24 / 36	32 / 48
percentage of operations with less than min. operating time	5.4% / 27.6%	5.9% / 30.3%	6.6% / 32.6%

Chapter VI: Conclusions and Future Work

Both implementation and performance of rotational scheduling algorithms have been researched extensively. The RPO scheduling algorithm was developed based on this previous research. The simulator models the IBM Ultrastar 18LZX disk features and is custom-made for studying the performance and the scheduling behavior of the RPO algorithm.

The simulation shows that using one head's positioning data for all the heads in a disk drive is accurate enough. The central heads can be used to characterize the behavior of all the heads, and the differences from other heads are little.

The queue size and the seek ranges are important factors in scheduling performance. Shorter seek ranges result in shorter access times as expected. Throughput grows constantly when the queue size doubles. Using a threshold value instead of several expected access time values in an RPO table can approximately reduce the necessary storage space by a factor of 30. For disks with queue size 32, the best throughput can be achieved when alpha is 1.75 to 2.5 for read operations and 1.5 to 1.75 for write operations depending on the seek range and the workload. However, the performance degradation is generally higher than 3% for read operations and 4% for write operations, and it is unfortunately not negligible.

The distribution of relative rotational displacement shows the behavior of the scheduling algorithm. The effects of the queue size and the seek range are validated by observing the distribution. Besides, the minimum operation times show an unexpected effect that a significant amount of operations are originally within the area which cannot be traversed in less than the minimum operating times.

The most important improvement in the RPO algorithm is considering probability when deciding whether a request can be served or not, and it is implemented in RPO tables by pre-computing the expected access times for each rotational distance. This probabilistic approach provides better scheduling decisions than the strictly "yes-no" approaches adopted in other rotational scheduling algorithms. Examining how much it outperforms other scheduling algorithms and validating the results for other disk drives is my next plan.

Some modern disk features can affect the scheduling performance such as on-board caching. To exploit these feature to improve the RPO algorithm is one of the final goals.

Finally, synthetic workloads modeling burst and locality attributes, as well as traces from real-world systems, will be utilized in the future work, and it can provide a more realistic analysis of system performance.

References

[Denning 67] P. J. Denning, "Effects of scheduling on file memory operations", *AFIPS Spring Joint Computer Conference*, pages 9-21, April 1967.

[Geist et al. 87] R. Geist, S. Daniel, "A continuum of disk scheduling algorithms", *ACM Transactions on Computer Systems*, pages 77-92, February 1987.

[Heath et al. 96] M. A. Heath, D. C. Pruett, B. C. Nguyen, "Method for reducing rotational latency in a disk drive", United States Patent No. 5,570,332, Oct, 1996.

[IBM 00] IBM hard disk drives, http://www.storage.ibm.com, 2000.

[Jacobson et al. 91] D. M. Jacobson and J. Wilkes, "Disk scheduling algorithms based on rotational position", HP Laboratories technical report HPL-CSP-91-7rev1.

[Merten 70] A. G. Merten, "Some quantitative techniques for file organization", Ph.D. Thesis, Technical Report No. 15, University of Wisconsin Computer Center, 1970.

[Palmer 99] J. Palmer, "RPO simulation code for IBM ULTRASTAR 18LZX disk drive", IBM Almaden Research Center, Summer 1999.

[Riedel et al. 98] E. Riedel, G. Gibson, C. Faloutsos, "Active storage for large-scale data mining and multimedia", *24th VLDB Conference*, New York, USA, 1998.

[Ruemmler et al. 93] C. Ruemmler, J. Wilkes, "UNIX disk access patterns", *Winter 1993 USENIX*, January 1993.

[SCSI-2 93] Small Computer System Interface-2, draft revision 10k edition, March 1993.

[Seaman et al. 66] P. H. Seaman, R. A. Lind, T. L. Wilson, "An analysis of auxiliary-storage activity", *IBM System Journal*, Vol. 5, No. 3,1996, pages 158-170.

[Worthington et al. 95] B. L. Worthington, G. R. Ganger, Y. N. Patt, J. Wilkes, "On-line extraction of SCSI disk drive parameters", *ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 146-156, May 1995.