Temporally Determinate Disk Access: An Experimental Approach*

Mohamed Aboutabl
Department of Computer Science
University of Maryland at College Park
aboutabl@cs.umd.edu

Ashok Agrawala
Department of Computer Science
University of Maryland at College Park
agrawala@cs.umd.edu

Jean-Dominique Decotignie
Départment d'Informatique
École Polytechnique Fédérale De Lausanne
decotignie@di.epfl.ch

Abstract

Disk drives are the most commonly used secondary storage devices in computer systems. The way operating systems access these devices leads to a wide range of variability in access time. In this paper we study the detailed temporal characteristics of disk drives. We describe a comprehensive set of experiments designed to build a model for the disk drive. Simulation is used to validate the model. This disk model will help design a device driver which can achieve a high degree of temporal determinacy.

1 Introduction

During the recent years, disk drives have tremendously improved in terms of capacity, speed, reliability, and physical size. Even though several other secondary storage technologies have emerged, disk drives remain the dominant choice. However, the traditional method to access a disk has not changed. Applications send a read/write request with the proper disk address to an operating system process called the *device driver*. The application process requesting the disk service is usually suspended until the service is completed. Meanwhile, the device driver relays the request to the disk controller hardware using a standard bus interface such as IDE or SCSI. When the controller is finished serving the request, it sends a hardware interrupt signal to the device driver process. The latter eventually awakens the application process. Throughout this procedure, no statement whatsoever is made about when a disk request is to be completed. In fact, the disk service time is modeled as a random variable with a high variance. This variability is mostly due to the mechanical activity involved. For many applications, this approach is adequate, and they are designed to tolerate such lack of knowledge about the service time.

When we consider *real-time* applications, they have to achieve both functional as well as temporal correctness, and certain time constraints are imposed on the execution of such

*This work is supported in part by ONR and ARPA under contract N66001-95-C-8619 to the Computer Science Department at the University of Maryland. The views, opinions, and/or findings contained in this paper are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Advanced Research Projects Agency, ONR or the U.S. Government.

applications. The unpredictability of the traditional approach for disk service is unacceptable to real-time applications. Consequently, early real-time applications avoided the use of disk drives and operated entirely in main memory. As the complexity of real-time applications increases, it becomes necessary to access disk-resident information. One example of such applications is the video-on-demand services. Clearly, these applications suffer from the temporally unpredictable behavior of disk drives. To alleviate this problem, disk controllers have added large caches to hide some of, but not eliminate, the variability in disk service time. In order to significantly reduce this variability, we should have a better understanding of its cause(s), and use this knowledge in the design of better device drivers. As a starting point, we study the delay components involved in a disk I/O operation such as a "read sector" request. We can identify the following delay components:

(a) A *seek* delay, during which the disk controller moves the head from its current position to the cylinder containing the requested sector. This delay depends on, among other factors, the head position prior to the request, and the distance traveled.

(b) A *rotational* delay, during which the head waits for the requested sector to rotate and arrive under the head. Since the disk rotates continuously, this delay also depends on the head position prior to the request, as well as on the disk rotation speed.

(c) An *off-surface transfer* delay, during which the data is read from the surface and stored in the controller's cache.

(d) A *host transfer* delay, during which the data is transfered from the controller's cache to the host main memory.

Both transfer delays of steps (c) and (d) typically show very small variability. On the other hand, seek and rotational delays have shown variability in the order of 10's of milliseconds.

The key factor to minimize these variability is to continuously keep track of the head position, and to have an accurate understanding of the disk dynamics. For example, seek delays can be precisely determined once we have a better knowledge of the head dynamics. The rotational delay is similarly determined by the disk rotation speed and the current angular position of the head. A device driver which uses this kind of knowledge can accurately predict the response time of a request.

In order to design a device driver which can accurately predict the service time of a disk I/O request, it is necessary that not only the disk dynamics be known, but that the physical layout of the information on the disk be also reflected in the temporal prediction. Further, when a disk controller uses a cache, the cache organization and management has to be incorporated into the device driver design. While general information about these topics may be available in open literature, the details needed for accurate temporal predictions are not easily available. Therefore, in the study presented here, we conducted a series of experiments to determine the necessary information about a disk. Our purpose in these experiments was to ascertain that enough information about a disk drive can be obtained this way. As the primary goal of our study was to determine the physical parameters of the disk, we did not address the effect of using on board controller caches on the prediction of service time. In fact, we disabled the controller cache in all the experiments. Clearly, the methodology used here can be applied to any disk drive.

In this paper, we present an approach for modeling the disk for the purpose of designing a device controller with the ability to accurately predict the service time of a request. The remainder of this paper is laid out as follows: in section 3, we describe the structure of a typical modern disk drive. In section 4 we describe the run-time environment under which our experiments were conducted. The different experiments used to extract the main parameters of the disk are explained in section 5. In section ??, we use of the extracted parameters to build a model for predicting disk I/O service times. This model is validated through a series of simulations in section 7. Finally, the conclusion and our future work are presented at the end of the paper.

2 Related Work

The scheduling of disk I/O requests has been a vital research problem whose results have greatly affected the performance of computer systems. As the design of disk drives change rapidly, there will always be a need for more research to find new disk scheduling algorithms, or at least modify existing ones, to reflect this rapidly evolving technology.

Worthington, Ganger, and Patt [Worthington94], studied the performance of several disk scheduling algorithms designed for modern disk drives. They examined the impact of complex logical-to-physical address mappings and large prefetching caches on scheduling effectiveness. Joined by J Wilkes, they later described, in [Worthington95], a suite of general-purpose techniques and algorithms for acquiring data on the structure and organization of SCSI disks via the ANSI-standard interface. We have started our work before their work was published. Their ideas were similar to ours. However, we present techniques that measure more disk parameters such as sector rotation time, track skewness, and boundaries of the recording zones. We also present, where applicable, a subjective study of the extracted disk parameters. One of our major contributions is an analytic model to predict the disk I/O service time taking into account the current position of the head. We also presented a 3^{rd}-degree piece-wise polynomial to model the seek time, contrary to the square-root curve suggested by previous researchers [Chen94].

Abbott and Garcia-Molina [Abbott90] provided three algorithms for scheduling disk I/O requests with deadlines. These are Earliest Deadline, ED-SCAN, and Feasible Deadline SCAN.

3 Structure of a Modern Disk Drive

The recording media of a hard disk drive is organized into *platters*. Each platter has two record-able surfaces. The surfaces are further organized into concentric recording paths called *tracks*. The set of vertically aligned tracks is called a *cylinder*. The data is laid out on the tracks in units called *sectors*. The size of each sector is determined by the disk formating utility, and is typically 512 bytes per sector. The data is read from and written to the sectors through a set of read/write *heads*. All the heads are built into one arm assembly, which is controlled by an actuator. At any time, only one head is active. The head assembly itself can move, *seek*, forward and backward to access a specific track. The platters rotate so that the requested sector finally arrives under the head. Figure 1 depicts the structure of a typical disk drive [Anderson95].

3.1 Zone-bit Recording

Since the circumference of a track increases as one moves away from the center, it is possible to accommodate more data on the outer tracks than on the inner tracks without the need to any special heads or medium [Schmidt95]. This is done by partitioning the cylinders into

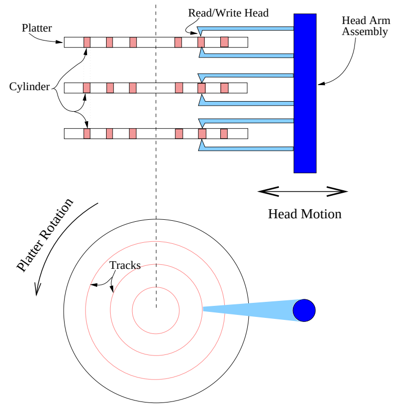

Figure 1: The structure of a hard disk

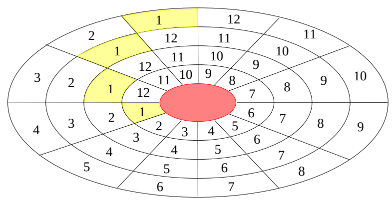

Figure 2: Track Skewness

zones. The outer zones have more sectors-per-track than the inner zones. The published specification for the disk drive we used do not contain such zone details as location and size.

3.2 Track Skewness

In many applications, the disk-resident data is accessed in a sequential order. If the requested data spans more than one track, then, as the read/write head is switching tracks, the next sector may be missed, and the head has to wait for one full revolution. To avoid such performance degradation, the start of a track is not aligned with the start of the previous one, and is skewed. This skewness is large enough to allow for the head switching time or the seek activity, permitting the first sector of the new track to be accessed without waiting for another revolution. Figure 2 shows an example for track skewness. While we believe that the disk we used has skewness built in its low level formatting, exact details are not published.

3.3 Block Addressing

The individual sectors of the disk can be referenced using either of two addressing schemes. These are:

- Cylinder/Head/Sector, or CHS, addressing mode in which the cylinder number, the head number, and sector number are specified. We observed that in the disk we used, the cylinder and head numbers start from 0, whereas the sector numbers in every track start from 1.
- Logical Block Addressing, or LBA, mode in which the disk is treated as a linear array of blocks numbered starting from 0. Each block is the same size as a physical disk sector, with a one-to-one mapping from block numbers to physical sectors.

For a more thorough discussion of the structure and modeling of modern disk drives, the reader is referred to [Ruemmler94]

4 Design Considerations and the Run-Time System

In this section we discuss various issues affecting the design of our experiments. These experiments were conducted using a Pentium 120 MHz computer with 16 MB of RAM and a 1 GB AC21000 Western Digital © hard disk. In order to avoid any interference with the operating system, we conducted our experiments under the Maruti real-time operating system. Maruti is a time-based hard real-time operating system developed by the System Design and Analysis Group at the University of Maryland College Park [Marti94]. Maruti guarantees that the process will run without being preempted for any reason which is helpful when time measurements are taken.

All time measurements were taken with the help of the *maruti_get_current_time* library function which reads the current system time directly from the the clock chip on the motherboard. We conducted several experiments to measure the performance of this function and found out it has an accuracy of ± 3 microseconds.

4.1 The Design of the Disk Driver

As a starting point, we wrote a special purpose device driver that runs under the Maruti operating System. We started from the device driver that is a part of Net-BSD and made the following changes:

- We took out all the code responsible for managing and scheduling the disk requests as well as the functionality to issue multiple-sector requests.
- The Interrupt-driven nature of the device driver was replaced by a busy-waiting loop. This decision was mandated by the need for a higher accuracy in our time measurements.
- Code was added to take time measurement at various phases of disk requests executions.

4.2 The On-board Controller Cache

It is observed that when the controller read-ahead cache is enabled, it is difficult to extract any accurate information about the physical characteristics of a disk drive. The presence of a cache conflicts with our analysis of the disk dynamics. Therefore, we had to disable the read-ahead controller cache[1]. Once we have a sufficient understanding of the physical characteristics of the disk, it would an easy task to incorporate the controller cache in the prediction of a disk request service time.

5 Measuring Disk Parameters

In this section we describe several experiments which we used to measure the disk parameters. We also provide in-depth discussion of the results we obtained using the WD AC21000 disk drive.

[1] Note that not all IDE disk drive manufacturers support the *Set Feature* command option which disables that kind of cache.

5.1 Reading a Disk Block

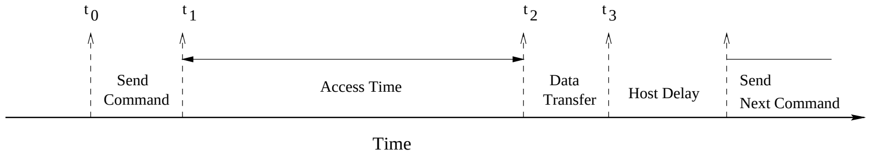

Figure 3: Sending a command to the disk controller

Our experiments used a routine that reads a block from the disk. That routine consisted of the following operations (figure 3):

- Wait for the controller to be READY to receive the next request, then (at time t_0) send the command to the disk controller through a set of on-board controller registers connected to the host's I/O ports. According to the IDE bus standard specifications, the controller will (at time t_1) become BUSY accessing the requested block.

- Wait for the controller's READY signal (at time t_2).

- Transfer the data from the controller's on-board sector buffer to the host's main memory.

Our preliminary experiments showed that both the first and third phases had constant durations, and were independent of the requested sector. A typical duration of the *Send* command phase was 16 microseconds. A one-sector data transfer phase lasted for 80 microseconds.

It is worth noting that, as shown in figure 3, the time period $[t_1, t_2]$ encompasses any necessary seek and/or latency delays, as well as the reading of the sector from the disk surface.

In most of the experiments, we issued two requests to the disk, one immediately after the other. In order to minimize the time before we may issue the next command to the controller, we eliminated the data transfer phase from all our experiments. We have also made use of *in-line* functions to reduce the host delay[2]. We will define $t(k,l)$ as the time elapsed between the completion of reading sector k and the completion of reading sector l, provided that the second request is issued as soon as the first one has been completed (figure 4).

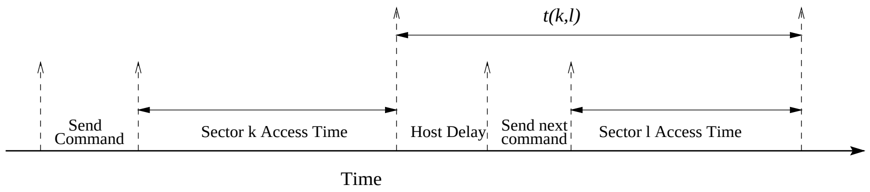

Figure 4: Two successive requests

[2]This is the processing done by the host to prepare for sending the next disk request, e.g. calculation of the block address

5.2 Disk Rotation Time (DRT)

We first measured the rotation speed because it was essential in extracting all other disk parameters. Rotation speed is usually reported by manufacturers as the number of revolutions per minute. The WD AC21000 disk rotates at 5200 revolutions per minute. This corresponds to a disk rotation time of 11538 microseconds/revolution.

We designed the following experiment to measure and study the stability of the actual value of the disk rotation time:

1. Measure the time $t(k, k)$ between two successive completions of reading the same block k. This is equal to one revolution time.

2. Repeat step 1 for different blocks at scattered locations on the disk. Record the maximum, minimum, variance, as well as the average time.

Table 1 shows some descriptive statistics on the measurements taken on the WD AC21000. All times are in microseconds. We measured the disk rotation time reading blocks at more than 2100 different locations on the disk.

It was observed that the disk rotation time had a mean of DRT = 11534 microseconds and a standard deviation of 1.828 microseconds. The experiment was repeated several times measuring the rotation time at the same set of blocks, and, except for few cases[3], the same readings were obtained.

Disk Rotation Time	
Mean	11534.19
Standard Error	0.040
Median	11534
Mode	11534
Standard Deviation	1.828
Minimum	11516
Maximum	11551
Count	2117
95% Confidence Interval	0.0779

Table 1: Descriptive Statistics of the Disk Rotation Time Measurements

5.3 Boundaries of Recording Zones

Our next goal was to determine the exact logical-to-physical mapping of block numbers. This included the number of sectors per track, which normally varies from the outer zones to the inner zones. We also determined the number of cylinders per zone. The experiment proceeded as follows:

1. Sequentially read the blocks of the disk and measure the time $t(k, k + 1)$ between the completion of reading two successive blocks.

[3] In two cases, we observed a disk rotation time of 11396 and 11667 microseconds. These two extreme readings were not repeatable. We suspect that they were due to some random thermal and/or electrical disturbances

(a) If both blocks are on the same track, then $t(k, k+1)$ shall be equal to DRT plus a one sector rotation time,

(b) If the two block are on different tracks, then due to track skewness, the later block will be accessed during the same revolution and $t(k, k+1)$ will be considerably less than DRT.

2. By observing the readings of step 1, we could determine when the last block in a track is reached.

3. By observing the changes in the number of sectors per track, we detected the end of a recording zone and the start of another.

The partitioning of the WD AC21000 disk drive, together with the sectors-per-track inside each zone, is given in table 2. This table provides the zone boundaries, in terms of cylinder numbers and logical block addresses, as well as the number of sectors per track (SPT). It was observed that a few tracks contained one or two less sectors than the rest in that zone[4]. We also noticed that the last cylinder on the disk is partially available for user data. We also found out that the tracks of the WD AC21000 drive are arranged in a Zig-Zag ordering as shown in figure 5.

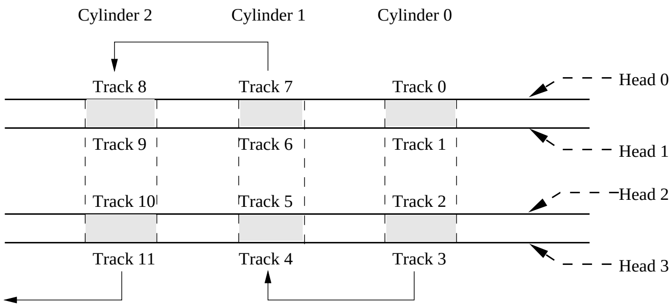

Figure 5: Track Arrangement in the WD AC21000

5.4 Sector Rotation Time (SRT)

The sector rotation time SRT is the time required for one sector to rotate completely under the read/write head. As the number of sectors per track increase as we move towards the center of the disk, the sector rotation time increases.

The controller circuitry, however, does not report the completion of a read/write command immediately after the end of the sector passes by the head. In fact, if the end of a sector n passes under the head at time e_n, there is a *completion overhead* δ_n that elapses before the completion of the operation is reported to the host at time t_n (see figure 6). Therefore, we can define the *effective sector reading time* of block n is equal to $SRT(n) + \delta_n$

However, without actually probing the internal electronics of the controller, we have no way to determine when a sector precisely starts and/or ends. The smallest granularity we can achieve in our experiments is to measure the time that elapses between the completion of reading two physically adjacent sectors on the same track. This time is used to approximate the sector rotation time $SRT(n)$ by measuring

[4] The missing sectors may be due to bad sectors on the surface

Zone no.	Cylinder From	To	Logical Block From	To	SPT
1	0	269	0	184538	171
2	270	374	184539	255098	168
3	375	544	255099	366618	164
4	545	768	366619	509977	160
5	769	1109	509978	720929	155
6	1110	1400	720930	895527	150
7	1401	1725	895528	1078824	141
8	1726	2103	1078825	1279919	133
9	2104	2344	1279920	1401381	126
10	2345	2628	1401382	1537694	120
11	2629	2858	1537695	1642574	114
12	2859	3271	1642575	1824292	110
13	3272	3497	1824293	1918308	104
14	3498	3706	1918309	2001908	100
15	3707	3887	2001909	2069963	94
16	3888	4018	2069964	2116599	89
17	4019	Heads 3&2	2116600	2116777	89
18	4019	Head 1	2116778	2116799	22

Table 2: Recording Zones of the WD AC21000 Hard Disk

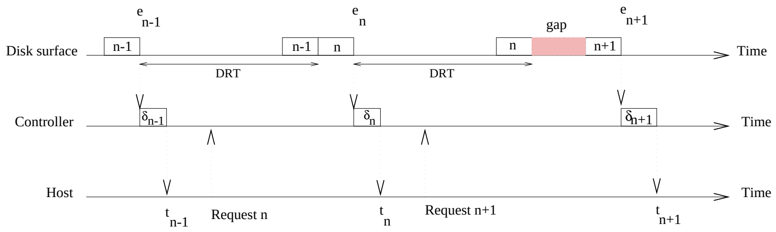

Figure 6: Measuring the Sector Rotation Time

$$\overline{SRT(n)} \stackrel{\text{def}}{=} t(n-1,n) - DRT$$
$$= SRT(n) + (\delta_n - \delta_{n-1}) ,$$

where, $t(n-1,n) = t_n - t_{n-1}$ as defined in section 5.1. More specifically, we first read sector $n-1$, then we immediately (with minimal host delay) issue a request for sector n. Notice that by the time the second request is issued, sector n has just passed the head. We will have to wait for a complete revolution before it comes back. This accounts for the DRT term in the above equation.

We measured the $\overline{SRT}$ at different recording zones on our disk. In each zone, we took measurements for 16300 sample sectors. Figure 7 shows a frequency histogram for the different $\overline{SRT}$ values obtained at zone 1; the outermost zone, zone 9; in the middle, and zone 16; which is very close to the center of the disk. We observed the following:

- The $\overline{SRT}$ values are clustered into two regions. The first cluster, representing "short" sectors, is centered around 40 microseconds and ranges from -10 to 90. The second

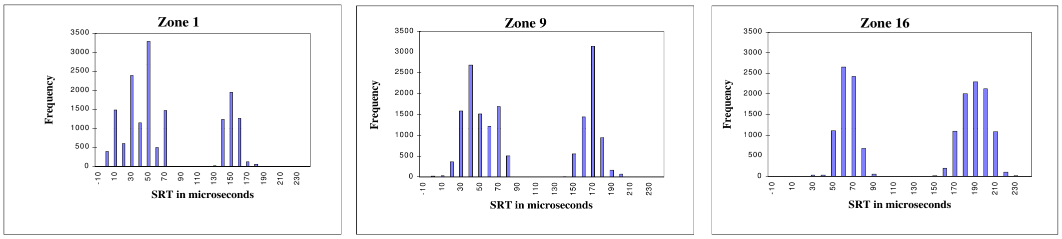

Figure 7: Frequency Distribution of the Sector Rotation Time at 3 Different Zones

cluster, representing "long" sectors, ranges from 130 to 220 microseconds, and is centered around 160.

- The number of "long" sectors per track is the same (50 sectors) in all tracks.

- The same $\overline{SRT}$ value was obtained for the same sector every time the experiment was repeated, with very few non-repeatable exceptions.

- The $\overline{SRT}$ values measured at different zones supported the theory that sectors on inner cylinders take longer to rotate than those on outer cylinders.

These observations lead to the following:

1. The sector overhead δ is repeatable and constant for the same sector. However, it varies from 0 to 50 microseconds for different sectors. This accounts for the negative $\overline{SRT}$ values.

2. there are 50 inter-sector gaps of approximate 120-microsecond length at equidistant angular positions. These gaps are probably used for storing embedded servo-control information.

5.5 Controller Overhead Time (COT)

There is a command preprocessing overhead inside the controller before the actuation and/or head switching starts.. Let sector i be requested when sector $i - n$ is currently under the read head. If the value of n is small enough, the preprocessing overhead inside the controller may result in missing sector i in the current revolution. To measure this kind of controller overhead, we conducted the following experiment:

1. For some block k, repeat for all other blocks l on the same track as k,

   - read block k immediately followed by a request to read block l. Measure the time elapsed between the completion of the two read requests, $t(k, l)$.

2. Repeat step 1 for several values of k evenly distributed over the disk, and calculate the minimum. The controller overhead is given by:

$$Controller\ Overhead\ Time\ COT \quad = \quad \min_{\forall k,l} \quad t(k,l)$$

For the WD AC21000 disk drive, the controller overhead was measured to be 1377 microseconds.

5.6 Head Switching Time (HST)

When the requested block is not on the same surface as the most recently accessed block, then the controller has to switch the active read/write head. This time is spent in addition to the controller overhead time discussed in the previous section. For the purpose of predicting the disk I/O service time, however, we consider the controller overhead time as part of the head switching time. To measure this parameter, we conducted an experiment identical to the experiment of section 5.5 except that the second block l was selected from a different track of the same cylinder as block k. Table 3 shows the head switching time to and from the 4 different recording surfaces of the WD AC21000 disk drive. The controller overhead time is also shown on the diagonal. We have observed that, with a precision of ± 50 microseconds, this matrix is symmetric.

Switch from	to Head			
Head	0	1	2	3
0	1379	2301	2540	2541
1	2318	1375	2297	2091
2	2564	2296	1376	2062
3	2554	2147	1994	1375

Table 3: Head Switching Time (in μSeconds) of the WD AC21000 Hard Disk

5.7 Measuring Track Skewness

As pointed out earlier in section 3, the starting locations of adjacent tracks, either on the same or on different cylinders, are not aligned. This improves the performance of the disk when serving sequential requests by taking into account any seek or head-switching activity involved. The track skewness has to be determined in order to calculate the latency time of any disk I/O request. We can measure the skewness of all tracks, denoted by $Skew(k)$ for all tracks k, as the time required to rotate from a common reference point to the first sector in track k. We can arbitrarily choose the common reference point to be the first user accessible block; that is block number 0.

The value of $Skew(k)$ was measured as follows:

1. Let Skew(0) = 0;
2. Repeat for $k = 1$ to Last track number
   (a) Let b1 = First block in track k -1 ;
   (b) Let b2 = First block in track k ;
   (c) Let $t^*(b1, b2)$ be the minimum of 5 different readings of $t(b1, b2)$;
   (d) Let $Skew(k) = (\ Skew(k-1) + t^*(b1, b2)\ ) \bmod DRT$

Table 4 shows some typical values of $Skew(k)$ for the first few tracks. When these values were plotted (figure 8), we observed a regular pattern for the track skewness values. The pattern repeats itself every 19 tracks, except for the first time, where its period is 24 tracks. This regularity helps reduce the amount of information that is required to determine the track skewness.

Track	Skewness	Track	Skewness	Track	Skewness
0	0	10	1007	20	2434
1	2321	11	3314	21	4740
2	4640	12	6096	22	7060
3	6946	13	8402	23	9367
4	9728	14	10720	24	613
5	500	15	1477	25	2919
6	2820	16	4254	26	5238
7	5127	17	6561	27	7544
8	7906	18	8879	28	10312
9	10223	19	11185	29	1084

Table 4: Track Skewness (in μSeconds) of the WD AC21000 Hard Disk

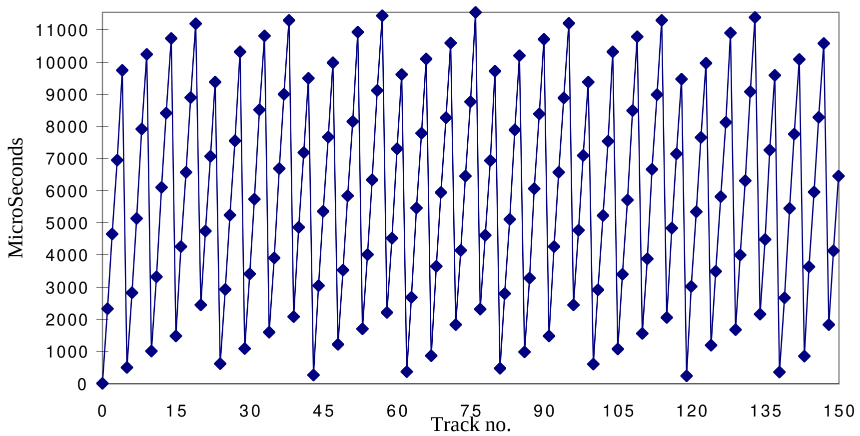

Figure 8: Track Skewness (relative to track 0) in the WD AC21000

5.8 Seek Operation

The objective here is to determine the characteristics of the seek operation and its dependency on the seek distance. We observed that the seek time also depends on other factors such as:

- the origin of the search; that is the current cylinder and sector, and
- the seek direction; forward or backward.

We conducted the following experiment to measure the seek time:

1. For distance $d = 0$ to the full seek distance
   (a) Select a cylinder c from which the seek will start.
   (b) For all $b \in$ { 5 evenly spaced blocks in cylinder c }
      - Find the minimum of $\{t(b,l) - SRT(l)\}$ for all blocks l on cylinder $c + d$ (forward seek), and on cylinder $c - d$ (backward seek). These two minima represent instances of $Seek(+d)$ and $Seek(-d)$, respectively.

(c) Record the maximum of the $Seek(+d)$ and $Seek(-d)$ values obtained above.

2. Repeat step 1 starting from a different cylinder c. It should be noted that the longer the seek distance, the less the number of candidate cylinders c.

The basis of the above algorithm is that the minimum value of $t(b,l)$ is achieved when the latency between blocks b and l is equal to the seek time. In this case, $t(b,l)$ is equal to $SRT(l) + Seek(d)$.

We have noticed, see figure 9, that the seek time varied when starting from different blocks in the same cylinder, even with the seek distance and direction were fixed. The range of this variability was of the order of ± 500 microseconds. As a conservative approach, we decided to record the maximum of these seek times.

Next, we examined the dependence of the seek time on the starting cylinder, given a fixed seek distance. Again, we have noticed, see figure 10, that there was a variability in the range of ± 150 microseconds. Given these variations, we decided to plot the seek curve with the seeks starting from 4 different cylinders, and recorded the maximum seek time for each seek distance. The resulting curve is shown in figure 11.

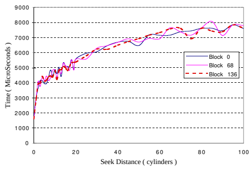

Figure 9: Seeks originating from 3 different blocks on cylinder 0

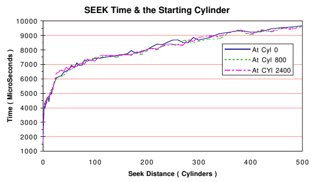

Figure 10: Seeks originating from different cylinders

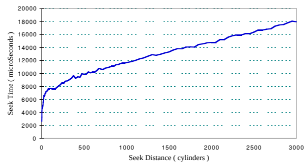

Figure 11: Seek Curve of the WD AC21000

At this point, it is worth mentioning that when we repeated the same experiment three times, with all factors being fixed, the seek time varied in a range of up to 1.2 millisecond in some extreme cases. However, the average variation in the seek time, with all known factors being fixed, was within 100 microseconds.

6 Prediction of the Disk Request Service Time

The objective in this section is to calculate an accurate estimate of the service time of a disk I/O request. We assume that the request has been issued at time t_{req}. For notational simplicity, we will denote time t as $t_{req} + COT$, where COT is the controller overhead time. In figure 12, we assume that the position of the read/write head at time t is denoted by $p(t)$ on track $k-n$. The requested sector s is on track k, which is n tracks away.

The service time may include one or more of the following:

- Seek(n) = time to seek to the target cylinder,
- HST = Head Switching Time to switch the read/write head to the proper surface,
- Latency(p(t) ,s,k) = time for the disk to rotate from the current position *p(t)* and arrive at the target sector s in track k , and
- SRT(s) = time to transfer the data from the surface to the sector-buffer.

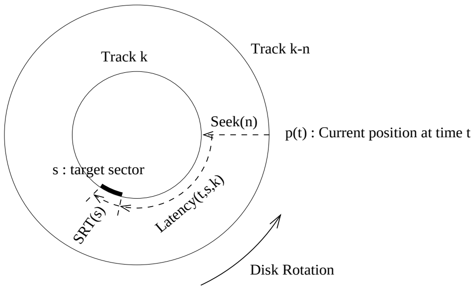

Figure 12: Disk Request Access Time

Typically, the first three components overlap in time. The data transfer component depends on the recording density and the disk rotation speed. In our analysis, we have referred to this component as the sector rotation time (SRT). The total access time of the request can be estimated by the following equations:

$$Revs \;=\; \left\lfloor \frac{Seek(n)}{DRT} \right\rfloor$$
$$L \;=\; Latency(\; p(t)\;, s, k)$$

$$Threshold \;=\; max(\; Seek(n) \;\bmod\; DRT\;, HST\;)$$
$$Service\ Time \;=\; SRT(s) \;+\; Revs \,*\, DRT \;+\; \begin{cases} L & : \text{ if L} \geq \text{Threshold} \\ L + DRT & : \text{ if L} < \text{Threshold} \end{cases},$$

where, DRT is the disk rotation time as measured in section 5.2. $Revs$ is the number of complete revolutions which occur while the head is seeking to the target cylinder. The value of "$Threshold$" determines whether an extra revolution is needed.

The calculation of the Seek and Latency time components is discussed in the following subsections

6.1 Prediction of the Seek Time

By closely inspecting the seek curve of figure 11, we could find a piece-wise polynomial approximation function, see figure 13.

For short seek distances $d \leq 140$ cylinders away from the current position, we could find the following least-squares third degree polynomial:

$$3677.09 \;+\; 128.202d \;-\; 1.37628d^2 \;+\; 0.004767d^3$$

For medium-length seeks, $140 < d \leq 1000$, the seek curve could be approximated by the following quadratic polynomial:

$$6528.4 \;+\; 8.53328d \;-\; 0.003381d^2$$

Finally, long seeks, $d > 1000$, were linearly proportional to the seek distance, and could be approximated by:

$$8643.65 \;+\; 3.14242d$$

Figure 13 also shows the fitting function as well as the original seek curve. This fitting function could now be used to predict the seek time of any disk request.

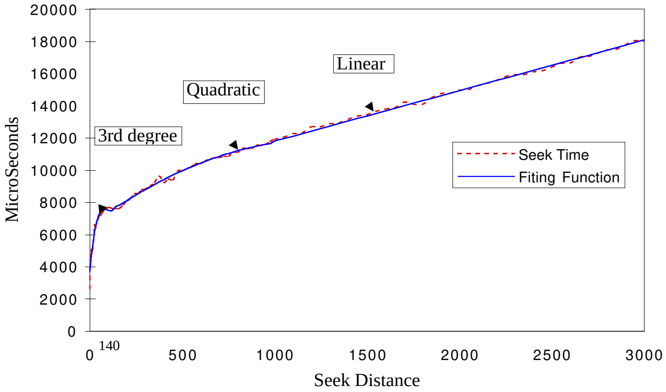

Figure 13: Piece-wise Polynomial Fit for the Seek Curve

6.2 Prediction of the Latency Time

In the this section, we develop a methodology to accurately predict the latency component, i.e. $Latency(\ p(t)\ , s, k)$, of a disk I/O service time. We assume that the request has been issued at time t_{req}. For notational simplicity, will will denote time t as $t_{req} + COT$, where COT is the controller overhead time. We first introduce the following notations:

1. $gap(s, k)$: The number of long ($\approx$ 110 microseconds) inter-sector gaps between the first sector of track k and sector s in the same track.
2. $rot(s, k)$: The time to rotate from the beginning of the first sector in track k to the beginning of sector s on the same track.
3. $p(t)$: The rotational position of the read/write head arm at any time instance $t \geq 0$. At time $t = 0$, the head is initialized at sector 1, head 0, of cylinder 0. The function $p(t)$ is periodic with a period of DRT.

Next, we show how the above quantities are computed.

Let $LONG$ denote the total number of long inter-sector gaps in a whole track. On the WD AC21000, $LONG$ was found to be 50. In general,

$$gap(s, k) \quad = \quad \left\lfloor \frac{s}{SPT(k)} * LONG \right\rfloor \quad ,$$

where SPT(k) is the number of sectors in track k. The rotation time $rot(s, k)$ can thus be computed as follows:

$$rot(s, k) \quad = \quad 120 \ * \ gaps(s, k) \quad + \quad \frac{DRT - 120 * LONG}{SPT(k)} \, (s - 1).$$

The calculation of $Latency(p(t), s, k)$ depends on the current position $p(t)$ relative to the target sector as follows (figure 14):

Case a If the target sector is still ahead in the current revolution, then

Latency(p(t) , s, k) = Skew(k) + rot(s , k) - (t mod DRT)

Case b If the target sector has just passed, then

Latency(p(t) , s, k) = Skew(k) + rot(s , k) - (t mod DRT) + DRT

$Skew(k)$ was defined in section 5.7 as the time to rotate from $p(0)$ to the beginning of sector 1 of track k. This value is measured and saved in a look-up table for all the tracks on disk.

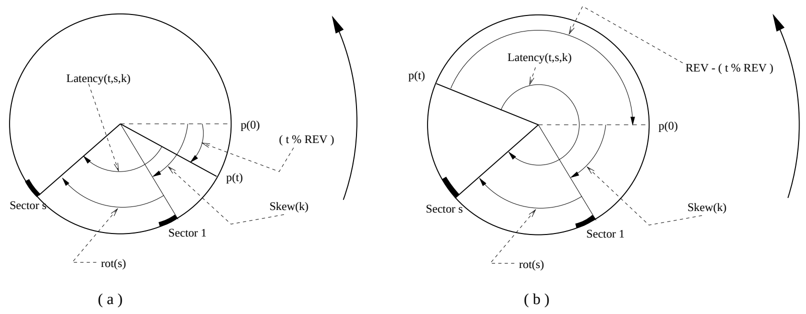

Figure 14: The Latency Time

7 Simulation Results

In order to validate our model of the disk drive, we have conducted a comprehensive set of experiments. In all these experiments, we recorded the actual service time of a read request and compared that value with the our estimate of the service time that was based on our model.

In the first experiment, we considered several sets of disk blocks. For each set, the blocks were randomly selected such that they belonged to the same track. The objective was to validate our modeling of the sector rotation times, the distribution of the long sector gaps, and the controller overhead time. These parameters were the basis for our estimation of the rotational displacement of each block relative to the beginning of its track.

Next, we relaxed the selection criterion so that blocks of the same set could now be on any track, but in the same cylinder. Our objective here was to validate our modeling of the track skewness as well as the head switching time.

Finally, we removed all restrictions and the blocks were randomly selected from any location on the disk. In this experiment, we were able to validate our modeling of the seek curve

In each experiment, we recorded an average of 15000 readings. The results that we obtained were very impressive. In 96% of the cases, our model predicted the service time within ±200 microseconds of the actually measured value. In 3% of the cases, our prediction deviated from the actual service time by one revolution. According to our model, if our predicted value of the service time misses the requested block by as low as 1 microsecond in the current revolution, then we will have to wait for the block to rotate up to one extra revolution before we can catch it. The pie chart of figure 15 shows the distribution of the absolute value of the prediction error in microseconds. The same figure applies to the results of all the three experiments. We also observed that our model was consistent in predicting the service time as the same block was repeatedly requested. The prediction errors were stable, which suggests that we can further minimize them if we could get more in depth understanding of the disk electromechanical characteristics.

Based on the results of our experiments, we conclude that our model of the disk drive can achieve a high degree of temporal accuracy in 96% of the cases.

Error in microsec	Legend	Frequency	Average	Percent
up to 100		14649	46	76%
100 to 200		3866	148	20%
200 to 300		225	219	1%
Missed by One DRT		494	11539	3%
Total		19234		

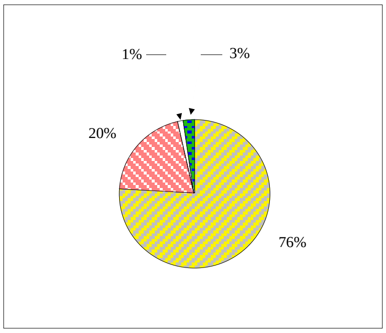

Figure 15: Error in The Prediction of Service Time

8 Conclusion and Future Work

In this paper, we proposed several experiment to extract vital disk parameters necessary to build a model for the disk drive. There are still some phenomena that need to be explained. The variability in the sector rotation times is one example for such phenomena. However, we have noticed that the temporal behavior of the disk is repeatable, even if not fully understandable without access to the internal design of the drive.

The parameters that we have collected so far enabled us to proceed with building a disk drive model. The model was verified by comparing simulation results with actual measurements. An accuracy of up to 200 microseconds was achieved when predicting the service time for 96% of the disk requests.

Being verified, the model will be used to simulate the performance of several real-time disk scheduling policies, and hopefully propose some enhancements to existing policies.

The effect of the on-board controller cache organization and replacement algorithms will be studied. The model will be expanded to accommodate for the existence of such cache.

Finally, a real-time disk scheduler will be built, and a special-purpose file system will be embedded into the Maruti real-time operating system. As an application, MPEG video playback will be tested on the final product.

References

[Worthington95] Bruce L. Worthington, Gregory R. Ganger, Yale N. Patt, and John Wilkes. "On-Line Extraction of SCSI Disk Drive Parameters." In *Proceedings of the 1995 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems,* pages 146-156, May 1995.

[Worthington94] B. Worthington, G. Ganger, and Y. Patt. "Scheduling Algorithms for Modern Disk Drives." In *Proceedings of the 1994 ACM SIGMETRICS*, pages 241-251, May 94.

[Ruemmler94] Chris Ruemmler and John Wilkes. An Introduction to Disk Drive Modeling. *IEEE Computer,* pages 17-28, March 1994.

[WD96] Western Digital Corporation, "Caviar AC21000 EIDE Disk Drive Technical Reference Manual", 1996.

[Schmidt95] Friedhelm Schmidt, "The SCSI Bus and IDE Interface Protocols, Applications, and Programming", Addison-Wesley 1995.

[Anderson95] D.T. Anderson and M. Tribble, "The Hard Disk Technical Guide", 11^{th} edition, Micro House 1995.

[Abbott90] Robert K Abbott and Hector Garcia-Molina. "Scheduling I/O Requests with Deadlines: a Performance Evaluation." In *Proceedings of the 1990 Real-Time System Symposium* pages 113-124, 1990.

[Jacobson91] D. M. Jacobson and John Wilkes. "Disk Scheduling Algorithms Based on Rotational Position." HP Laboratories Technical Report HPL-CSP-91-7rev1, Feb. 26, 1991.

[Chen94] Shenze Chen and Don Towsley. "Scheduling Customers in a Non-Removal Real-Time Systems with an Application to Disk Scheduling." *Real-Time Systems*, vol 6, pages 55-72, 1994.

[Marti94] M. Saksena, J. da Silva and A. K. Agrawala, "Design and Implementation of Maruti-II." *Principles of Real-Time Systems* Sang Son (ed.), 1994. Also available as UMD CS-TR-3181, UMICAS TR-93-122.

[Geist87] Robert Geist and Stephen Daniel. "A Continuum of Disk Scheduling Algorithms." *ACM Trans. on Computer Systems*, vol 5, No. 1, February 1987, pages 77-92.