Low-Cost Access Time Model for Serpentine Tape Drives

Olav Sandstå and Roger Midtstraum
Department of Computer and Information Science
Norwegian University of Science and Technology
N-7034 Trondheim, Norway
{olavsa, roger}@idi.ntnu.no

Abstract

When a serpentine tape drive is used as a slow random access device, the I/O performance can be substantially improved by clever re-ordering of the I/O requests. This kind of re-ordering relies on a scheduling algorithm and a model of the access time. In this paper, we propose a low-cost access time model for serpentine tape drives, which is not a trivial task due to the complex data layout of serpentine tape. This model provides a way to estimate the physical positions on the tape for any logical data block, provides cost functions to estimate the seek time between two physical tape positions, and computes the transfer time of a data request. Our experiments show that the mapping from logical address to physical position has to be instrumented once for each tape cartridge. Algorithms are given to do this at a low cost. The accuracy of the model is assessed by measurements on tape drives and by use in scheduling of I/O requests. Experiments show that the model estimates are good enough to facilitate efficient scheduling of I/O requests.

1 Introduction

In modern computing, magnetic tape has mainly been used by applications that access data on the tape sequentially. Today however, there is a growing interest in building computer applications which store vast amounts of digital data, while still wanting relatively fast random access to the data. Due to its high storage density and low cost, magnetic tape can be a relevant storage technology to consider for such systems. The main limitation of magnetic tape is the very long access time, which can easily reach several minutes in unfavorable situations.

Hillyer and Silberschatz [1] have shown that the access times of serpentine tape drives can often be substantially reduced by use of a scheduler, which reorganizes the retrieval order of the tape requests. In order to do intelligent reorganizing, such a scheduler must be able to compute fairly accurate estimates of access times. Because of the complex data layout on a serpentine tape, this is a not a trivial task. Hillyer and Silberschatz have solved this problem for the Quantum DLT 4000 drive [2] and the IBM 3570 Magstar drive [3], by use of two complex, tailor-made models, which require tremendous amounts of analysis for each individual tape cartridge. Johnson and Miller [4] have proposed a simpler model using a piece-wise linear regression model to estimate seek times.

In this paper, we present a general access time model for a serpentine tape drive. This model consists of three main parts. First, we establish a way to estimate the physical position on a tape, given a logical block address. Second, we partition the seek space into eight disjoint seek classes, with regard to the work that is incurred on the tape drive. For each seek class, we provide analytic cost functions to compute the seek time. Third, we provide a way to compute an estimate for the transfer time of a given data request. The access time model is designed to balance the need of accuracy against the need of fast characterization of tapes. We provide several algorithms which achieve such characterization at a fairly low cost. The accuracy of the proposed access time model is validated by measurements on the Quantum DLT 2000 and the Tandberg MLR1 tape drives. Simulation studies and actual measurements on tape drives show that the achieved accuracy is sufficient to facilitate efficient scheduling of random retrievals from tape.

The rest of this paper is organized as follows. Section 2 gives a brief introduction to magnetic tape technology. Section 3 presents the characteristics of the Tandberg MLR1 drive that was used to develop the access time model for serpentine tape drives, which is presented in Section 4. Section 5 proposes strategies to improve the accuracy of the model by characterization of individual tapes. Section 6 validates the proposed access time model by comparing the estimated access times to measured access times using tape drives. In Section 7 we present results from using the model for scheduling of random accesses to tape, and Section 8 gives our conclusions.

2 Technologies for magnetic tape

There are three main tape technologies: helical scan tape, parallel tape and serpentine tape. *Helical scan tape drives*

read/write transverse or diagonal tracks on the tape using a rotating read/write head. The best known standards are 4mm (DAT), 8mm (video), and the analog VHS cassette. Tapes using helical scan technology obtain high data densities and high transfer rates. For some helical scan drives, the rotation of the head can impose severe wear-out on the tape, possibly limiting the number of times a tape can be read. *Parallel tape* is the classical data tape, where the drives read/write all tracks in parallel during one scan of the tape. *Serpentine tape drives* first read/write a track (or a group of tracks) in forward direction, then read/write the next track in reverse direction, and so on, leading to a *serpentine pattern* for the data layout.

In this paper we focus on the *serpentine* tape model. There are three important technologies for serpentine tape drives, QIC, DLT and LTO. QIC – Quarter Inch Cassette – started as a standard for inexpensive tape storage with modest capacity and bandwidth. During the last years specifications have improved, and now QIC is comparable to DLT, regarding both storage capacity and bandwidth. The QIC standard [5] uses tapes which are a quarter inch wide, has cartridges with both wheels inside the cassette, and provides standard tape formats, covering storage range capacities from 60 MB to 25 GB. DLT – Digital Linear Tape [6] – is a technology originally developed by Digital Equipment Corporation. DLT uses a half inch tape which is stored in a cartridge with only one reel, the second reel is part of the tape drive. When inserting a DLT tape into a drive, the tape first has to be mounted onto this reel. LTO – Linear Tape Open [7] – is a new technology proposed by Hewlett-Packard, IBM and Seagate. It is supposed to become an open technology architecture for tape drives and cartridges, making it possible to interchange tape cartridges between drives from several manufacturers. At the moment, no drives using the LTO format are available.

While QIC and DLT drives are slightly different, their access time characteristics are similar and to a high degree dictated by the serpentine data layout. Contrary to parallel and helical scan drives, serpentine drives do not provide a direct relationship between logical block addresses and physical positions on the tape, making it much harder to estimate the access times.

3 Performance characteristics of a serpentine tape drive

To gain understanding of the behavior of a serpentine tape drive, we have used the Tandberg MLR1 tape drive [8]. This drive uses serpentine data layout and is based on the 13 GB QIC standard [5], making it possible to store 13 GB per tape (without compression). The drive can deliver (read/write) a maximum sustained data rate of 1.5 MB/s to/from the host computer. Each tape has 72 logical tracks, 36 in the forward direction and 36 in the reverse direction.

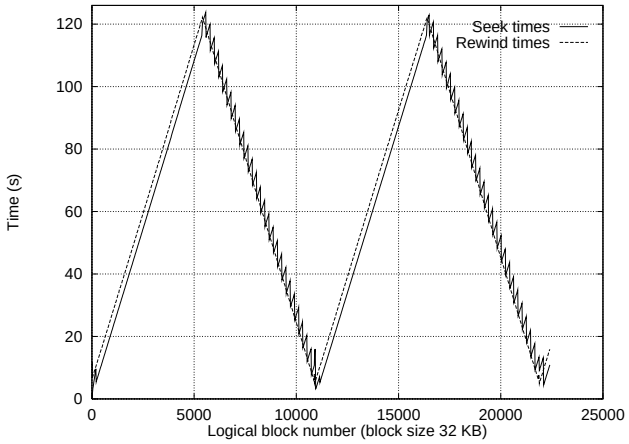


Figure 1: Seek and rewind times for the first tracks of the tape.

To determine the characteristics for the Tandberg MLR1, we ran several experiments on the tape drive. First, the tapes were written with fixed length logical data blocks of 32 KB. The number of blocks on each tape varied from 398000 to 400100 blocks. The average write time for a block was 22 milliseconds. To write a full tape takes approximately 2.5 hours. By performing seeks on the tape, we found the access time for one block to vary between 1 and 126 seconds. For seeks starting on the beginning of the tape, the average seek time is 65 seconds. For seeks between two random positions on the tape, the average seek time is 45 seconds.

Figure 1 shows the seek and rewind times for the first four tracks of a tape. The x-axis contains the logical address of data blocks on the tape, and the y-axis gives the number of milliseconds it takes to seek from the start of the tape to each of the data blocks. For every sixteenth logical block address, we measured the time needed to seek from the beginning of the tape to the block, and the time to rewind back to the beginning of the tape. From the figure, we see that for *forward tracks*, the curves for seek and rewind times both are straight and overlap, but for *reverse tracks* the curve for seek times has a sawtooth pattern, while the curve for rewind times is straight.

To explain the sawtooth pattern, we note that each 'tooth' is a straight line covering about 200 logical blocks. The reason we get this pattern on the reverse tracks is that these tracks have to be read in the opposite direction of the forward tracks. When the tape drive tries to locate a position on a reverse track, starting from the beginning of the tape, it first has to seek past the sought block, and then start reading in the read direction until it has found the sought block. Figure 1 indicates that the tape drive uses a set of predetermined points to decide where to stop the seek in forward direction, and start seeking in the opposite direction. These points correspond to the first block in each sawtooth. Hillyer and Sil-

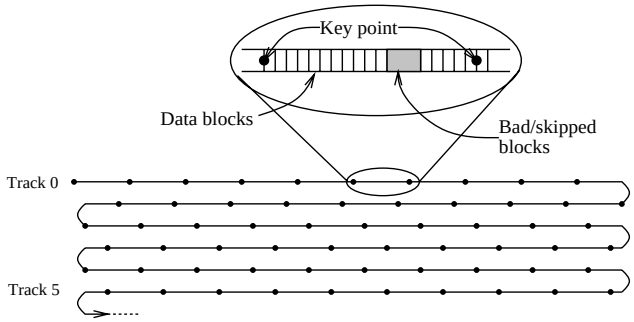


Figure 2: The serpentine layout of the first tracks on a tape with key points.

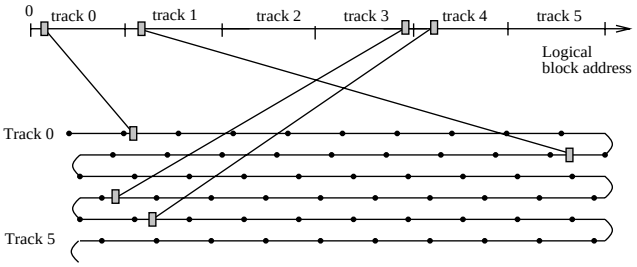


Figure 3: Mapping from logical block addresses to physical positions on the tape. The bullets along the physical tape are the key points of the tape.

berschatz [2] experienced similar sawtooth patterns for the Quantum DLT 4000 drive. They defined the points where the seek time has a large dip from one sawtooth to the next as the *key points* of the tape. Figure 2 shows the serpentine layout of the first tracks on a tape with the key points included. But opposite to what we found, they also experienced sawtooth patterns along the forward tracks. The reason is that the DLT 4000 uses one speed (seek speed) for locating the key point, and a slower speed (read speed) for locating the sought block between two key points. Tandberg MLR1 uses the same speed for both seeking and reading. This suggests that there will be key points along the forward tracks too, and by plotting seek times for seek operations starting on a different position than the beginning of the tape, we get the sawtooth pattern on the forward tracks too.

4 Access time model

The *access time* is the time it takes from the point when a memory device starts execution of an operation, until the data is available to the entity requesting the data, i.e., the sum of the *seek time* and the *transfer time* for the data. For tape operations, there is not much that can be done with the transfer time. As soon as the drive starts reading data from the tape, it will continue reading with a constant transfer rate until it reaches the end of the requested data region. Thus, the transfer time will be proportional to the size of the requested data. Contrary, seek time is essentially wasted time, and should be reduced as much as possible. As a consequence, the main focus of our access time model will be on how to model seek times, since this part of the access time is non-trivial to model, and provides opportunities for substantial optimization of the total access time.

In the presentation of the model, we assume that the tape contains fixed sized blocks. Further, we assume the tape is mounted in the tape drive and positioned at logical block address L_0 when an I/O request arrives. Such an I/O request consists of the logical block address of the first block requested, L_1, and the number of consecutive blocks to be read, N. The purpose of the access time model is to estimate the time the tape drive will use to re-position the tape from the current logical position L_0, to the logical start position L_1, plus the transfer time for the N blocks:

$$accessTime(L_0, L_1, N) = seekTime(L_0, L_1) + transferTime(L_1, N)$$

Hillyer and Silberschatz [2] have proposed an access time model for the Quantum DLT 4000 drive, which relies on locating the address of each key point on the entire tape. This gives a very accurate model, but requires about twelve hours of processing for each single tape. To avoid such problems, we propose a model, that does not depend on knowledge of the exact location of each key point. Our model is based on the following strategy:

1. We estimate the physical position of each logical block on the tape, by use of the logical address of the first block of each track.
2. We estimate the seek time between two physical tape positions by partitioning the possible seeks into disjunct seek classes. For each seek class, we provide a cost function to estimate the cost of the seeks in the class.
3. We estimate the transfer time as the time it takes for the drive to transport the read area of the tape past the drive's read head, plus the time it takes to make the necessary track changes.

4.1 Estimating physical tape positions for logical addresses

Applications access data stored on tapes by using logical block addresses. To be able to establish a cost model for seek and transfer times, we have to find the physical tape positions for the logical block addresses. A physical position on a serpentine tape is given by the *track number*

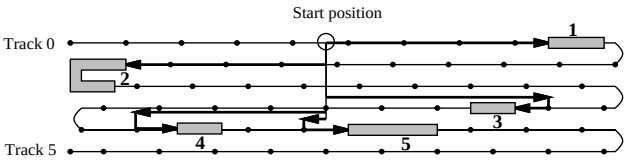


Figure 4: Example of a serpentine tape with the key points marked on the tracks, and possible seek patterns for five data requests.

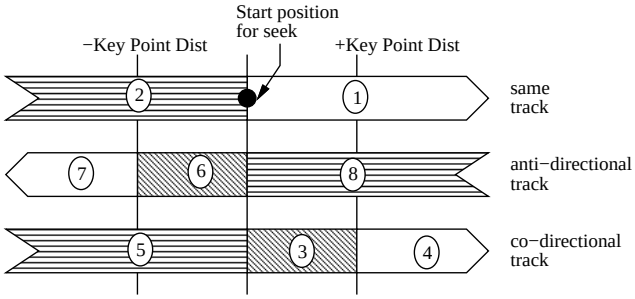


Figure 5: Model used to partition seeks into eight seek classes. It is important to note that this figure is seen from the position the tape drive's read/write head has on the tape when the seek starts.

and the *physical distance* from the beginning of the tape, $(trackno, tapepos)$. Figure 3 shows some examples of how the logical blocks on the first tracks on a tape are mapped to the physical tape.

To establish the mapping from logical block addresses (L) to physical tape positions (p), we use the logical block address of the first block on each track. In this discussion, we assume we have these track addresses available. We will later explain how these addresses can be found. Given these track addresses, it is easy to make a function $track(L)$ which returns the *track number* for any given logical address. Further, assuming the track addresses are stored in the array $trStart[]$, we find the physical distance from the start of the tape as:

$$tapepos(L) = \begin{cases} \frac{L - trStart[track(L)]}{trStart[track(L)+1] - trStart[track(L)]} \\ \text{if } track(L) \text{ is even} \\ 1 - \frac{L - trStart[track(L)]}{trStart[track(L)+1] - trStart[track(L)]} \\ \text{if } track(L) \text{ is odd} \end{cases} \quad (1)$$

This function returns the tape position as a number between 0 and 1. The reason for dividing by the length of the track is, as we will show later, that the length of the tracks vary within a tape.

4.2 Estimating seek times

Figure 4 shows five examples of possible seeks. When a seek starts, the tape drive is positioned at a forward track. For seek number 1, we have to change neither track nor winding direction. Seek number 2 is an example of a seek where we have to change both track and winding direction. For seek number 3, 4 and 5, the tape drive has to seek beyond the start of the requested data area to locate the closest key point. This results in a longer winding distance than the physical distance.

Given the current physical position of the tape drive and the physical position of the start of the requested data item, the model must estimate the time needed by the drive to wind to this position. There are four variables which influence the seek time:

1. the physical distance between the two tape positions,
2. time to change track,
3. time to change winding direction,
4. time to locate the closest preceding key point of the requested data block.

In the remainder of this subsection, we establish an analytical model for how these four cost variables influence the seek time. Every possible seek will be partitioned into one of eight disjunct seek classes based on how the cost variables influences that particular seek. Figure 5 shows how the seeks are partitioned into one of the seek classes based on the relative location (seek distance, track changes and winding direction) of the sought block compared to the physical start position of the seek. Table 1 contains an overview of which cost variables influence each of the seek classes.

Table 1: The different cost variables which influences each of the eight seek classes of the cost model.

Seek class	Distance	Track change	Winding direction	Locating key point	
				Sometimes	Always
1	X				
2	X		X		X
3	X	X		X	
4	X	X			
5	X	X	X		X
6	X	X	X	X	
7	X	X	X		
8	X	X			X

Physical tape distance. As seen in the previous section, the seek time between two logical block addresses is dominated

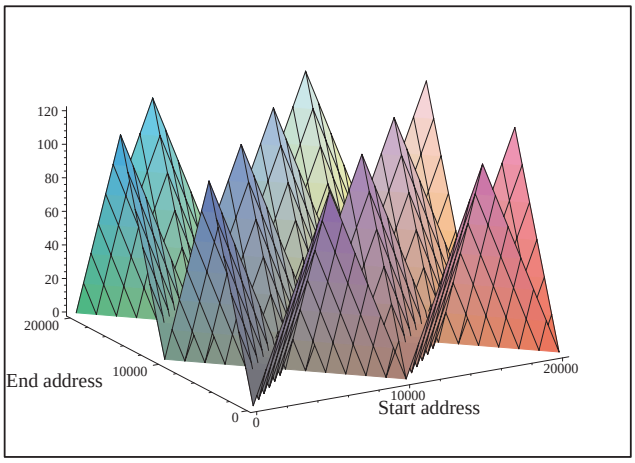


Figure 6: Plot of seek times (in seconds) due to tape winding between positions on the first four tracks on a tape. The seek times are computed using Equation 2.

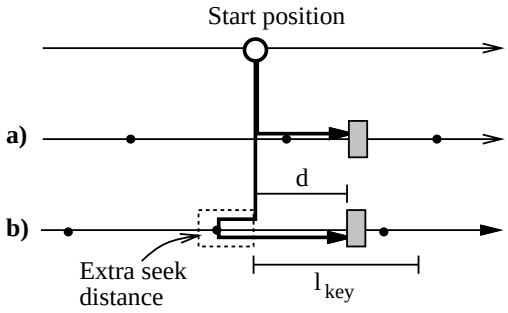


Figure 7: The two possible seek patterns for seeks in seek class 3. **a)** There is a key point between the start position and the sought block, and no extra seek distance is needed for locating the key point. **b)** There is no key point between the start position and the sought block, and the tape drive has to rewind to locate the key point. The extra seek distance needed to locate the key point is marked on the figure.

by the time to wind the tape from the physical start position to the requested position on the tape. As can be seen in Figure 1, the time usage is mostly proportional to the physical distance. Thus, in the model we estimate the seek time due to tape winding between two physical tape positions as:

$$t_{seek}(p_{start}, p_{stop}) = t_{wind} \mid p_{stop} - p_{start} \mid \qquad (2)$$

where the physical positions p_{start} and p_{stop} are found using Equation 1 and t_{wind} is the time the drive uses to wind the tape from the beginning of the tape to the end of the tape. Figure 6 contains a plot of how the seek time due to winding of the tape varies for seeks between logical addresses on the first tracks of a tape.

When the physical distance between the start position and the requested position is large, this function gives a good approximation of the total seek time. For shorter seek distances, the other cost variables have to be included in the model.

Change of track and winding direction. To improve the model, we include the cost of track changes and change of winding direction. Each time the drive has to change from one track to another, we add the track change cost t_{tc}. There are two reasons for approximating this cost with the constant t_{tc}. First, the cost of a track change is mainly a result of having to reposition the tape head and adjust it to a new servo track, not from the physical distance the head has to be moved. Second, the drive changes between *logical* tracks which do not necessarily correspond to the physical movement of the drive's tape head.

Similarly, we add the cost t_{turn} every time the drive has to change winding direction. Assuming the drive just has finished reading a block (i.e., it is winding in one direction), when it receives a new seek command, the drive has to change winding direction zero, one or two times depending on the relative location of the requested block compared to the current physical tape location:

Case	Cost
seeks forward on the same or a co-directional track (e.g., seek 1 in Figure 4)	$0 \cdot t_{turn}$
changes to an anti-directional track (e.g., seek 2 and 3 in Figure 4)	$1 \cdot t_{turn}$
seeks backwards on the same or a co-directional track (e.g., seek 4 in Figure 4)	$2 \cdot t_{turn}$

This far we have included in the seek time the costs that would incur if the drive was able to seek directly from one position to another without having to go through a key point. Unfortunately, in some cases, the locating of the closest key point incurs extra seek time.

Locating key points. Each time the drive has to seek beyond the start of the requested data area to locate the key point, as in seek number 3, 4 and 5 in Figure 4, this results in a longer winding distance than the physical distance between the start position and the requested block. This extra seek distance depends on the distance between the requested block and the closest key point. The most accurate method for estimating this distance will be to locate each of the key points as done by Hillyer and Silberschatz [2]. Unfortunately, this is too costly for most applications. In our approach, we include the *average* cost of locating the closest key point. Since we do not locate the key points, an important thing to note is that there will be seeks, where we do not know in advance whether the seek to the key point will incur extra seek distance or not. Fortunately, this will only occur for seeks which are shorter than the distance between two key points. An example is seek 5 in Figure 4. If the closest key point for seek 5 is between the start position and

block 5, the drive can wind directly to the block, if not it has to rewind until it gets to the key point, then change winding direction and read until it has reached block 5.

As mentioned earlier, all seeks can be partitioned into eight seek classes as shown in Figure 5. The following table shows how the seek times in each seek class will be influenced by locating the key point:

Seek class	Probability	Cost of locating key point
1, 4, 7	0	0
2, 5, 8	1	$l_{key}t_{wind}$
3	$1 - \frac{d}{l_{key}}$	$(l_{key} - d)t_{wind} + 2t_{turn}$
6	$1 - \frac{d}{l_{key}}$	$(l_{key} - d)t_{wind}$

For seeks in seek class 2, 5, and 8 (see for example seek 3 and 4 in Figure 4), the average extra cost for locating the key point, will be the cost of seeking the length of the distance between two key points, l_{key} (half the distance between two key points to locate the key point, and the same distance to get back to the requested data block). For seeks in seek class 3 and 6 (see for example seek 5 in Figure 4), the formula for the cost will be more complicated since there only is a certain probability that the seek time will be influenced by having to locate the key point. This is illustrated in Figure 7 for seeks in seek class 3. If there exists a key point between the start position for the seek and the requested data block (case a) in the figure), no extra cost will occur. If there is no key point between the start position and the requested data block, the tape drive has to rewind to the closest key point preceding the block as shown in case b) in Figure 7. The situation is similar for seeks in seek class 6. The probability of having to seek extra distance to locate the preceding key point depends on the physical distance between the current position and the requested data block, $P(extra\ cost) = 1 - \frac{d}{l_{key}}$. The extra distance the drive will have to seek is $l_{key} - d$. For seeks in seek class 3, the drive will also have to change winding direction twice.

The complete cost functions for all seek classes are given in Table 2. These are found by adding the cost for each of the cost variables that influence each seek class (see Table 1). In each of the cost functions we have included a constant, $\hat{t}_i$, to account for extra delays due to for example startup delays of the mechanical operations in the drive.

4.3 Estimating transfer times

To estimate the transfer time of a tape access is much easier than estimating the seek time, because the drive reads the tape at a constant data rate. Only when the drive has to change track during the reading of the data segment (as in seek 2 in Figure 4), the model has to include the cost of a track change in the transfer time. For a request for N blocks starting at logical block address L_1, the transfer time is given by:

Table 2: Cost functions for the eight seek classes in the model. In the formulas the seek distance is given as $d = \mid p_{start} - p_{stop} \mid$. l_{key} is the physical distance between two key points given as a fraction of the total tape length. t_{turn} and t_{tc} is the amount of time it takes to change winding direction and change tracks. t_{wind} is the total winding time for a track.

Class	Seek time cost function
1	$t_{wind}d + \hat{t}_1$
2	$(d + l_{key})t_{wind} + 2t_{turn} + \hat{t}_2$
3	$\frac{d^2 - l_{key}d + l_{key}^2}{l_{key}} t_{wind} + 2(1 - \frac{d}{l_{key}})t_{turn} + t_{tc} + \hat{t}_3$
4	$t_{wind}d + t_{tc} + \hat{t}_4$
5	$(d + l_{key})t_{wind} + 2t_{turn} + t_{tc} + \hat{t}_5$
6	$\frac{d^2 - l_{key}d + l_{key}^2}{l_{key}} t_{wind} + t_{turn} + t_{tc} + \hat{t}_6$
7	$t_{wind}d + t_{turn} + t_{tc} + \hat{t}_7$
8	$(d + l_{key})t_{wind} + t_{tc} + \hat{t}_8$

$$transferTime(L_1, N) = N \frac{t_{wind}}{trStart[track(L_1) + 1] - trStart[track(L_1)] + (track(L_1 + N) - track(L_1))\ t_{tc\_read}}$$

It is worth noting, that the constant t_{tc_read} is different from the constant t_{tc} used in the seek time functions. The track change during a read operation always occurs on the end of a track, it always changes to the next track and the drive has to determine the start of the data area on the next track.

4.4 Instrumenting the model to be used with the Tandberg MLR1 drive

To use the model for a given serpentine drive type, we have to determine values for the constants used by the model. The seek time functions given in Table 2 are only depending on the physical seek distance. For all seek classes, except for class 3 and 6, the variable part of the functions is proportional to the physical seek distance between the start and end positions. Thus, for these classes the seek time function will be of the form $\alpha + \beta(\mid p_{start} - p_{stop} \mid)t_{wind}$. So instead of determining values for the constants t_{turn} and t_{tc}, which would be hard to get exact values for, we determine the constants α and β for each seek class. For seek class 3 and 6, the seek time is not a linear function of the physical seek distance. Still, since these functions are for very short seeks, we can approximate these with a linear function without much loss of accuracy. By doing this, the seek time functions in Table 2 can be written as shown in the second column of Table 3.

To use the model with the Tandberg MLR1 drive we have established values for the constants by practical use of the

Table 3: Cost functions for the eight seek classes, with corresponding constants determined for the Tandberg MLR1 drive. These functions return the estimated seek time for a given seek. t_{wind} is the total winding time for a track. For a Tandberg MLR1, this takes 120 seconds.

Seek class	Cost functions	Values for	
		α	β
1	$\alpha_1 + \beta_1(\mid p_{start} - p_{stop} \mid)t_{wind}$	0.814	0.984
2	$\alpha_2 + \beta_2(\mid p_{start} - p_{stop} \mid)t_{wind}$	8.805	0.983
3	$\alpha_3 + \beta_3(\mid p_{start} - p_{stop} \mid)t_{wind}$	8.285	−0.573
4	$\alpha_4 + \beta_4(\mid p_{start} - p_{stop} \mid)t_{wind}$	1.036	0.975
5	$\alpha_5 + \beta_5(\mid p_{start} - p_{stop} \mid)t_{wind}$	8.636	0.979
6	$\alpha_6 + \beta_6(\mid p_{start} - p_{stop} \mid)t_{wind}$	7.633	0.307
7	$\alpha_7 + \beta_7(\mid p_{start} - p_{stop} \mid)t_{wind}$	2.068	0.975
8	$\alpha_8 + \beta_8(\mid p_{start} - p_{stop} \mid)t_{wind}$	7.760	0.979

drive. The constants were found by performing 2000 seeks on three different tapes. The seek positions were selected such that the number of seeks of each seek class was approximately the same. We measured the seek time for each seek, and determined the constants for the seek time functions in each seek class by using linear regression. The resulting constants are given in the third column of Table 3.

To estimate transfer times for the MLR1 we have to determine the constants t_{wind} and t_{tc_read}. t_{wind} is the time the tape drive needs to wind from the start of the tape to the end of the tape. For the Tandberg MLR1, the manufacturer states that the maximum rewind time is 120 seconds. This is consistent with our experiences, as we have measured maximum rewind times between 119.9 and 121.2 seconds.

To estimate the time used to change from one track to the next during continuous reading, we measured the time used by the drive to read 32 MB data segments from the three tapes. By computing the difference in transfer time between those data segments which included a track change during the read operation, and those which did not, we found the average value for t_{tc_read} to be 2.9 seconds

5 Characterizing individual tapes

In the previous section, we explained how to estimate the *physical position* of each *logical block address*. This mapping requires knowledge of the logical address of the first block on each track. In this section we present four strategies for estimating/finding these track addresses. It should be obvious that the better the estimate of the track addresses is, the more exact will the estimated access times be.

The first of the strategies is generic, and can be used for all MLR1 tapes. The three other strategies improve the accuracy of the estimated track addresses by characterizing each individual tape.

Average Tape-Length. The first strategy assumes that each tape has the same number of data blocks, and that the data blocks are evenly distributed on all the tracks. Unfortunately, the number of blocks per tape varies rather much. For the tapes we have used, the number of blocks written has been between 398082 and 400055 blocks per tape. The average number of blocks per tape has been 398637, giving an average value of 5537 blocks per track. We use this as the first approximation of the track addresses. Since it is based on an average tape, we call the strategy *Average Tape-Length*.

The problem with the average tape-length strategy is that the estimates for physical positions get worse as we get farther out on the tape. The reason is that we do not know the exact number of blocks per track, and the error in each track length is added as we increase the track number. To make a model without this drifting problem, we need to characterize each individual tape. The straightforward way to characterize a tape completely would be to perform a seek from the start of the tape to each block on the tape. Unfortunately, this is not feasible, since it would take more than a year to perform this for a single tape. Another way to improve the accuracy of the model is to find better estimates for the number of blocks per track on each tape.

Exact Tape-Length. A first approximation of the number of blocks per track can be found by dividing the exact number of blocks written to the tape by the number of tracks on the tape. This can only be done if the entire tape is filled up by fixed sized blocks. We call this strategy *Exact tape-length*.

To further improve the model, we can try to identify the address of the first block on each track. These addresses will vary from tape to tape, due to varying numbers of bad blocks and blocks skipped during writing of the tape. We have tested two different strategies for estimating the start address of each track. The first strategy is based on the write times of the tape, while the second strategy finds the end of the tracks by performing read operations on the tape.

Write-Turn. If we have control of the writing of the tape, and the tape is written block by block, we can measure the writing time for each block. Writing a 32 KB block to the tape takes on average 22 milliseconds, but every time the tape reaches the end of a track, the tape drive has to stop the tape motion before it can start writing in the opposite direction. By studying the writing times, we have found this change of direction to take about three seconds for the Tandberg MLR1 drive. We use this to get a rather accurate estimate for the start address of each track. Since most tape drives use a write buffer, the addresses found during analyzing of the write times have to be adjusted to compensate for this buffer. The reason is that the write times will stay at the average write time after we have reached the end of a track until the write buffer is full. We call this strategy *Write-Turn*.

Table 4: Results from testing the model on a Tandberg MLR1 drive using the Average Tape-length, Exact tape-length, Write-Turn, and Read-Turn algorithms for instrumenting the model. The table contains the average difference between measured and estimated access times for 2000 random block accesses.

	Average error $[seconds]$			
Strategy	All	Tape 1	Tape 2	Tape 3
Average tape-length	10.0 s	11.5 s	11.1 s	7.49 s
Exact tape-length	6.21 s	12.2 s	4.50 s	1.92 s
Write-turn	1.69 s	1.74 s	1.85 s	1.49 s
Read-turn	1.71 s	1.74 s	1.86 s	1.54 s

Read-Turn. If the tape is already written by someone else, or by an application which does not let us have access to the write times for each data block, we can locate the end of the tracks by performing read operations on the tape. One way to do this is to position the tape head on a block close to the end of a track and then start reading contiguous blocks while measuring the read time of each block. As long as the drive reads blocks from the current track, the time for reading one block should be about 20 milliseconds. When the drive reaches the end of the track, it has to change read direction. This change of direction takes about five seconds, and is easily detectable by measuring the time to read each of the blocks. We can use this to detect the block address of the first block on a track.

To reduce the total time it takes to find the end of the tracks, we do this only for the 36 reverse tracks. This saves us from a complete wind/rewind of the tape and from the work of locating the end of the 36 forward tracks.

6 Validation of the model

In this section we validate the model by comparing access times estimated by the model to measurements of access times on tape drives. We also compare the accuracy of the model that can be achieved using the four different strategies for characterizing the tapes presented in the previous section. The access times were obtained by measuring the time used from the time that the computer sends a request for a 32 KB block to the tape drive, until the block is available in main memory. On the completion of one request, a new request was executed without any pause.

6.1 Validation using Tandberg MLR1

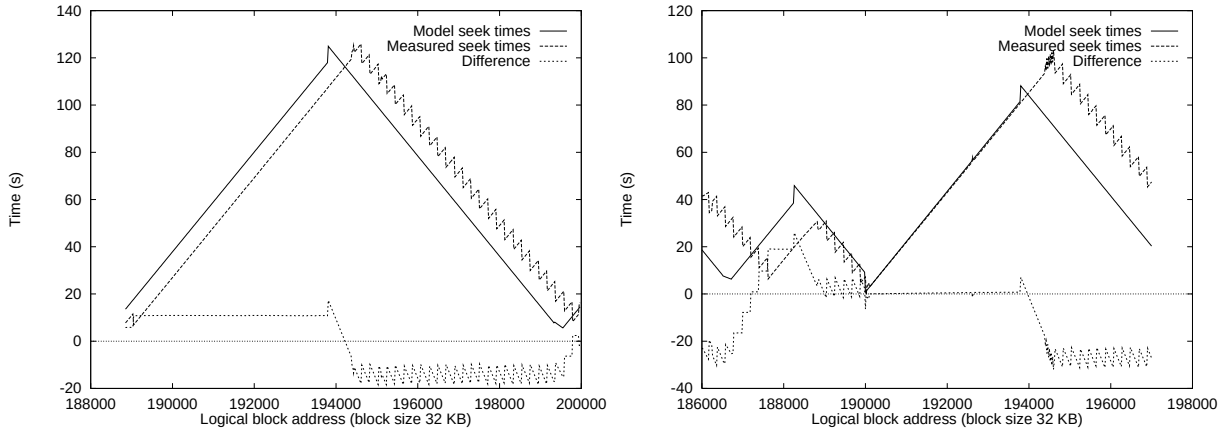
Three tapes, which were not used during instrumentation of the model constants, were used in the validation of the model. These were filled with 32 KB data blocks. During the writing of the tapes, we logged the write time for each block. From the log of write times, we got the exact number of blocks on each tape, and by analyzing the write times with the *Write-Turn* strategy we found the start address of each track. We also ran the *Read-Turn* algorithm on each of the tapes to find the start address of each forward track.

To compare access times estimated by the model with measured access times using the Tandberg MLR1 drive, we performed 2000 random block accesses on each of the three tapes. For each access, we measured the access time and compared it to the corresponding access time estimated by the model. Table 4 contains the average difference between the measured and estimated access times for each of the four strategies for characterizing the tapes.

Before we comment on these numbers, it is worth noting that without a tape model all that can be said about the access times is that they are in the interval from 1 to 126 seconds with an average of 45 seconds. By studying the table, we see that the model performs worst when we use the *Average Tape-Length* strategy, with an average difference between estimated and measured access times of 10 seconds. This is as expected, since the varying tape sizes lead to bad estimates for the start address of each track. As a result the estimated seek times will drift away from the measured seek times as we get further out on the tape. An example can be seen in Figure 8. Figure 8a shows a plot of measured and estimated seek times for seeks starting at the beginning of the tape to every 20th block on two tracks on the tape, together with the difference. Figure 8b contains a similar plot for the same two tracks when we use a fixed position about 1/3 out on the tape as the start position for the seeks. These two figures show that the estimated access times do not model the measured seek times very well. The reason is the use of fixed, average track length, in the model.

As Table 4 shows, the results are much better when we use one of the three strategies which characterize each tape. We also note that the two strategies which estimates the length of the individual tracks perform better than the strategy where we use a constant track length based on the total length of the tape. The reason is that even though the *Exact Tape-Length* strategy gives a correct estimate for the average track length, the track lengths can vary within a tape. As a result, the average difference between estimated and measured access times can vary rather much from tape to tape when using the *Exact Tape-Length*, e.g., compare the results for tape 1 and tape 3 in Table 4.

If we compare the two strategies for detecting the ends of the tracks, we see that they perform almost identically, with the *Write-Turn* strategy performing slightly better. In our experiments, the average difference between estimated and measured access times for random accesses was 1.7 seconds when using the *Write-Turn* strategy for finding the track addresses. There are two reasons why the results when using the *Write-Turn* strategy differ from the results when using the *Read-Turn* strategy. First, using *Read-Turn* we only



(a) Seeks start at beginning of tape.

(b) Seeks start at block address 190000.

Figure 8: Measured and estimated seek times for accesses to data blocks on two tracks using the tape model instrumented by the *Average Tape-Length*.

localize half of the track addresses. Second, the strategies may not make the exact same decision about what is the first block on each track, due to the use of a buffer during the writing of the tape. In Figure 9a, we have plotted the measured and estimated seek times for seeks starting at the beginning of the tape together with the difference for the same two tracks as shown in Figure 8a using the *Write-Turn* strategy. This time we observe that the two curves overlap much better, leading to better estimates. Figure 9b shows the corresponding curves for seeks starting at a fixed position 1/3 out on the tape. This figure should be compared to Figure 8b.

In Figure 10a we have plotted the distribution of the difference between estimated and measured accesses times for random accesses when using the *Write-Turn* strategy. This figure shows that most of the estimated access times are within 5 seconds from the measured access times. Figure 10b compares the distribution of the difference between estimated and measured times for three of the strategies. For *Write-Turn*, 90 percent of the measured access times are within 5 seconds from the estimated access times, while for *Average Tape-Length* this has increased to almost 25 seconds.

6.2 Validation using Quantum DLT 2000

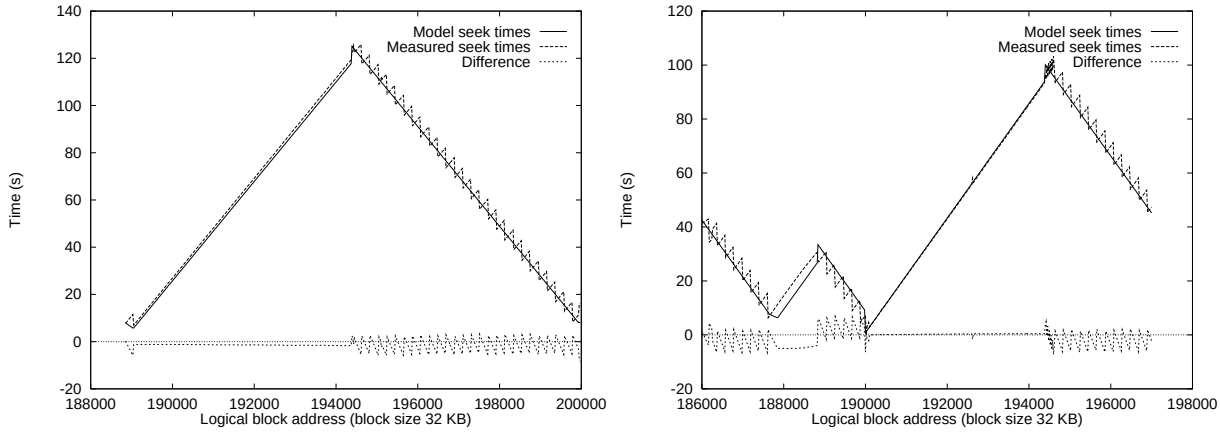
The model was developed using the Tandberg MLR1 drive. To test how the model performs for a different tape drive, we tested it using one of the department's old Quantum DLT 2000 drives. Just as for the MLR1, we found the values for the constants in Table 3 by performing seek operations on three tapes, and using linear regression on the seeks within each of the seek classes to determine the constants.

By running 2000 random block accesses on three other tapes, and comparing the measured access times with the access times estimated by the model, we got the average difference as given in Table 5. These results should be compared to the results given in Table 4 for the MLR1 drive.

Table 5: Results from testing the model on a Quantum DLT 2000 drive using the Average Tape-length, Exact Tape-length, Write-Turn, and Read-Turn algorithms for instrumenting the model. The table contains the average difference between measured and estimated access times for 2000 random block accesses.

	Average error $[seconds]$			
Strategy	All	Tape 1	Tape 2	Tape 3
Average tape-length	24.0 s	24.9 s	24.0 s	23.0 s
Exact tape-length	13.8 s	8.69 s	7.27 s	25.5 s
Write-turn	6.84 s	6.39 s	6.59 s	7.54 s
Read-turn	6.89 s	6.64 s	6.83 s	7.20 s

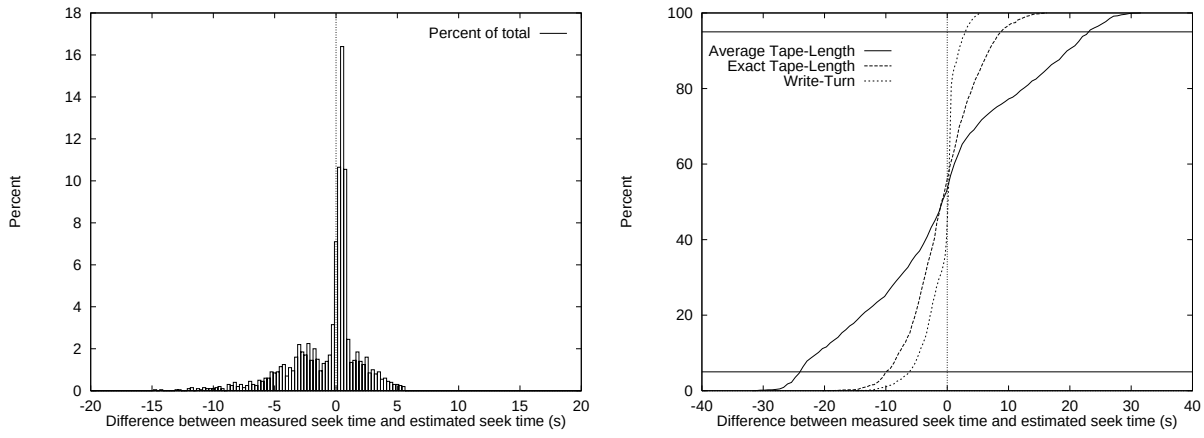
As can be seen from the table, the average error is about four times higher for the DLT 2000 drive than for the MLR1 drive when using the Write-Turn and the Read-Turn algorithms for characterizing the tapes. There are three main reasons for this. First, the model was developed for the Tandberg MLR1. While they are both serpentine tape drives, the DLT 2000 behaves somewhat differently when locating tape positions [2]. Second, the DLT 2000 has fewer key points on each track. As a result, the distance between the key points is longer. This will add to the average error for seeks where the cost functions include extra costs to locate the closest key point (i.e., all seek classes except 1, 4 and 7 in Table 2). Third, the DLT 2000 uses one speed for seeking, and a lower speed for reading. The maximum speed is used for locating



(a) Seeks start at beginning of tape.

(b) Seeks start at block address 190000.

Figure 9: Measured and estimated seek times for two tracks using the tape model instrumented by the *Write-Turn* algorithm.



(a)

(b)

Figure 10: **a)** Distribution of the difference between measured and estimated access times for random accesses by using the tape model instrumented by the *Write-Turn* algorithm. **b)** Cumulative distribution of the differences between measured and estimated times for random accesses for the *Average Tape-Length*, *Exact Tape-Length* and *Write-turn* strategy.

the closest key point. The drive then changes to the reading speed to seek to the requested block, which increases the average error for all seeks.

Of the three causes for the larger errors mentioned above, it should be easy to improve the model to handle different seek and read speeds while still maintaining a generic model. The other two points are more difficult to improve. To include the difference in behavior when locating tape positions between MLR1 and DLT 2000 into the model, would make it more complex and less generic. To reduce the effect of the longer distance between key points would require us to include the positions of key points into the model, and worse, it would make the characterization process much more time consuming.

The average access time for a DLT 2000 is about 60 seconds. Compared to not using an access time model, being able to estimate access times with an average error of about 7 seconds is still a large improvement. Thus although the model was developed using a MLR1 drive, it is also useful for other serpentine tape drives.

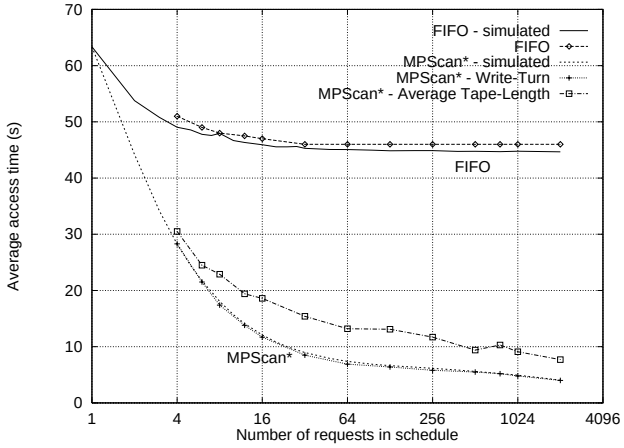


Figure 11: Average access time for each I/O request for different problem sizes, using the FIFO and MPScan* scheduling approaches. The figure shows both access times from simulation studies and results from real experiments using the Tandberg MLR1 drive. The size of each I/O request is one logical block of 32 KB.

6.3 Cost of establishing the model

It is important to be aware of the cost of getting the better results by using the model. The cost of establishing the serpentine tape model is low. The cost functions for each of the eight seek classes in Table 2, can be established once for each tape drive type. Finding the start addresses of the tracks has to be done once for each tape because these vary from tape to tape. The *Exact tape-length* strategy only requires that we get the total length of the tape when it is written. The *Write-Turn* strategy requires that we are able to measure the writing times of the blocks on the tape. If these writing times are available, the cost of finding the track addresses is virtually zero. The *Read-Turn* algorithm requires that each tape is run through the process of finding the end of the tracks before the model can be used. On average, we have measured the time usage for the algorithm to be about 13 minutes per tape. This is still worth the extra cost because of the much better estimates provided by the model.

The implementation of the model consists of about 400 lines of C++ code. For each characterized tape, we must store the track addresses, i.e. one integer per track, when using the *Write-Turn* and *Read-Turn* strategies, or the total length of the tape when using the *Exact tape-length* strategy.

7 Scheduling of I/O requests

The goal of scheduling concurrent requests is to produce a retrieval order, which will result in a minimum execution time when the requests are executed by the tape drive. In this section, we evaluate the usability of the access time model by using it for scheduling random data accesses. This is done by comparing the estimated execution time of schedules with the measured retrieval times, and by investigate the improvement that can be obtained by using the access time model for scheduling of the requests.

To evaluate the access time model, we consider two scheduling approaches, MPScan* and FIFO. A *MPScan** (Multi-Pass Scan Star) scheduler uses the access time model to reorder the requests to make the most out of the streaming capability of the tape drive. The main idea is to organize the requests into one or several full scans of the tape such that the drive avoids frequent changes of winding direction and is able to stream as much of the time as possible. In contrast, FIFO is a model-independent approach. A *FIFO* scheduler executes the requests in the initial order. A more thorough discussion of scheduling algorithms can be found in [9].

Figure 11 shows the results from simulation studies and actual experiments using a Tandberg MLR1 drive. The figure gives the average access time per retrieved object as a function of the scheduling approach and the number of requests in the schedule. To compare the effect of characterizing the tape, we have included results for MPScan* using both the *Average Tape-length* and the *Write-Turn* strategy for instrumenting the tape model. From the figure, one should make three important observations. First, there is a relatively good correspondence between estimated access times (from simulations) and measured access times (except when using the Average Tape-length strategy). For most schedules, the difference between estimated and measured access times is less than +/-5 percent. Second, for schedules of lengths from 2 to 2048 requests, there are substantial savings to be collected by use of a model-dependent algorithm like MPScan*. The maximum gain is for a schedule of 196

requests, where a MPScan* schedule executes in 20 minutes and 47 seconds, compared to an execution time of 2 hours, 5 minutes and 15 seconds for the corresponding FIFO schedule, saving more than 1 hour and 45 minutes. Third, characterization of each tape by using, e.g., Write-Turn, gives much better results than using the Average Tape-length strategy for instrumenting the model.

From these results, we make the conclusion that the accuracy of the proposed access time model is sufficient to serve as a basis for efficient scheduling of random I/O requests against the Tandberg MLR1 tape drive. To get the best utilization of the tape drive, each tape has to be characterized before using the model.

8 Conclusion

In this paper we have proposed an access time model for serpentine tape drives. By studying the behavior of a Tandberg MLR1 tape drive, we have partitioned every possible seek into one of eight distinct seek classes. This partition is based on which operations the drive has to perform in order to go from the current position it has on the tape, to the physical position of the requested data block. For each of these cases, we have established cost functions. These cost functions use physical tape positions for estimating access times. To map from logical block addresses used by applications, to physical positions, the model uses estimates for the logical address of the first data block on each track.

Experiments show that the length of each track varies between tapes and within a single tape. As a result, to improve the accuracy of the estimated access times, it is necessary to characterize each tape by estimating the address of the first block on each track. The paper presents several algorithms with varying costs to perform this characterization. By using the best characterization algorithm, *Write-Turn*, the model is able to estimate access times with an average difference between estimated and measured times of 1.7 seconds for the Tandberg MLR1 drive.

The proposed model balances the need of accuracy with the time needed to characterize each individual tape. One of the strengths of the model is the low cost of characterizing individual tapes. If we have control of the writing of the tape, the cost of performing the characterization of the tape is virtually zero by using the *Write-Turn* strategy. If we are not able to log the writing of the tape, the address of the first block on each track can be found by using the *Read-Turn* strategy. The *Read-Turn* algorithm uses 13 minutes compared to twelve hours for the algorithms suggested by Hillyer and Silberschatz [2].

Although the model is made using a specific tape drive, it is generic enough to be easily adjusted to other serpentine tape drives. This is shown by testing and evaluating the model using a Quantum DLT 2000 drive. The utility of the model is demonstrated by using it for scheduling of random accesses against a tape. In our research on databases for storage and retrieval of digital images and videos, we have used the access time model as a basis for scheduling of concurrent accesses for multimedia objects stored on magnetic tape [10].

To improve the model, the key points have to be included in the model. As shown by Hillyer and Silberschatz [2], it is too time consuming to locate these for each tape. It would take even more time to do this on the Tandberg MLR1, since this drive has about twice as many key points per track as the Quantum DLT 4000. If we should include the key points in the model, information about the location of the key points has to be made available to applications by the producer of the tape drive.

References

[1] B. K. Hillyer and A. Silberschatz. Random I/O scheduling in online tertiary storage. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 195–204, Montreal, Canada, June 1996.

[2] B. K. Hillyer and A. Silberschatz. On the modeling and performance characteristics of a serpentine tape drive. In *Proceedings of the 1996 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 170–179, Philadelphia, Pennsylvania, May 1996.

[3] B. K. Hillyer and A. Silberschatz. Scheduling non-contiguous tape retrievals. In *Proceedings from Fifteenth IEEE Symposium on Mass Storage Systems*, pages 113–123, College Park, Maryland, March 1998. In cooperation with Sixth NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies.

[4] T. Johnson and E. L. Miller. Performance measurements of tertiary storage devices. In *Proceedings of the 24th VLDB Conference*, pages 50–61, New York, USA, August 1998.

[5] QIC Development Standard. *QIC-5010-DC, Serial Recorded Magnetic Tape Cartridge For Information Interchange, Revision E*. Quarter-Inch Cartridge Drive Standards, Inc., December 1994.

[6] D. Lignos. Digital linear tape (DLT) Technology and product family overview. In *Proceedings of Fourth NASA Goddard Conference on Mass Storage Systems and Technologies*, Maryland, USA, March 1995.

[7] Linear Tape–Open (LTO) Technology. White paper, April 1998. http://www.lto-technology.com/.

[8] Tandberg Data, Oslo, Norway. *Tandberg MLR1 Series Streaming Tape Cartridge Drives Reference Manual*, 1st edition, August 1996.

[9] O. Sandstå and R. Midtstraum. Improving the access time performance of serpentine tape drives. *To be presented at the 15th International Conference on Data Engineering*, Sydney, Australia, March 1999.

[10] O. Sandstå and R. Midtstraum. Analysis of retrieval of multimedia data stored on magnetic tape. In *Proceedings from 1998 IEEE International Workshop on Multi-Media Database Management Systems*, pages 54–63, Dayton, Ohio, August 1998.