Performance Measurements of Tertiary Storage Devices

Theodore Johnson
johnsont@research.att.com
AT&T Labs - Research

Florham Park, NJ 07932

Ethan L. Miller
elm@csee.umbc.edu
CSEE Department
University of Maryland, Baltimore County
Baltimore, MD 21250

Abstract

In spite of the rapid decrease in magnetic disk prices, tertiary storage (i.e., removable media in a robotic storage library) is becoming increasingly popular. The fact that so much data can be stored encourages applications that use ever more massive data sets. Application drivers include multimedia databases, data warehouses, scientific databases, and digital libraries and archives. The database research community has responded with investigations into systems integration, performance modeling, and performance optimization.

Tertiary storage systems present special challenges because of their unusual performance characteristics. Access latencies can range into minutes even on unloaded systems, but transfer rates can be very high. Tertiary storage is implemented with a wide array of technologies, each with its own performance quirks. However, little detailed performance information about tertiary storage devices has been published. In this paper we present detailed measurements of several tape drives and robotic storage libraries. The tape drives we measure include the DLT 4000, DLT 7000, Ampex 310, IBM 3590, 4mm DAT, and the Sony DTF drive. This mixture of equipment includes high and low performance drives, serpentine and helical scan drives, and cartridge and cassette tapes. The detailed measurements of different aspects of tertiary storage system performance provides an understanding of the issues related to integrating tape-based tertiary storage with a DBMS.

Proceedings of the 24th VLDB Conference
New York, USA, 1998

1 Introduction

A tertiary storage system typically refers to a data storage system that uses drives that accept removable media, a storage rack for the removable media, and a robot arm to transfer media between the storage rack and the drives. The media can be disks (usually optical disks) or tapes, though in this paper we concentrate on tape-based tertiary storage. Tertiary storage is used for massive data storage because the amortized per-byte storage cost is usually two orders of magnitude less than on-line storage (e.g., see [23, 8, 33]). Tertiary storage has other benefits, including the removability of the media and fewer moving parts. However, tertiary storage devices have unusual performance characteristics, and access latencies can range into the minutes. Further, there is not much published literature on the performance of these devices. In this paper, we present a detailed performance measurement study of tape-based tertiary storage devices.

Applications that generate and use massive data sets drive the use of and research into tertiary storage. For example, some scientific data sets are extremely large. The NASA EOSDIS project, which supports research into climate change, will collect and archive on the order of ten petabytes of data [19]. Many other scientific projects, such as high-energy physics [21] also have very large data storage requirements. An emerging trend is to integrate databases with tertiary storage. This trend is driven by data warehouses [31, 14], scientific databases [19, 32], multimedia [33, 4], and digital libraries [20]. The database research community has responded to these needs with considerable research into tertiary storage DBMS architectures and optimization.

Systems research (e.g., [25, 14, 30, 36, 26]) makes assumptions about relative cost of various operations.

Performance optimization research (e.g., [3, 5, 20, 35]) use a variety of cost models of the time to fetch media, mount it, and so on. However, these assumptions and models often are either limited, or ignore some significant performance quirks of tertiary storage devices. The development of DBMS technology that incorporates tertiary storage requires a broad-based measurement study of the tape-based tertiary storage devices. In fact, our development efforts [15, 14, 2] were the motivation for this project.

Tertiary storage technology is improving rapidly, with exponential increases in tape capacity and tape drive throughput. The tape drives measured in this study use current technology, but in a few years they will be obsolete. However, we feel that our work makes significant contribution towards understanding tertiary storage performance. First, future tape drive technology will resemble current technology. While seek and transfer rates are likely to change dramatically, the shapes of the curves is not. Second, the measurements described in this paper were accomplished using standard interfaces, such as `mt` and `mtio`. Thus, the measurements described here can be repeated for future products.

2 Related Work

The Sequoia 2000 project [32, 1] is intended to develop a database technology that can be applied to EOSDIS. Satellite images are treated as large objects, which reside on tertiary storage and are cached on secondary storage. The Paradise project has a similar motivation. In [36], a novel query processing technique is proposed for queries that reference tape-resident large objects. The query is executed in two passes. The first pass is a dummy execution intended to collect the sequence of large-object references generated by the query. This sequence is used for scheduling large object fetches in the second execution of the query. In [35], the authors find optimal object sizes for the available tape drives.

Multimedia databases use tertiary storage for large objects (e.g., video clips) [18]. A survey of multimedia database research appears in [4]. Triantafillou and Papadakis [33] observe that multimedia objects can be loaded directly from tape, and provide techniques for increasing capacity at a small cost in buffer storage. Kraiss and Weikum [20] give algorithms for tertiary storage cache and prefetch management in a document database. Christodoulakis, Triantafillou, and Zioga [5] give algorithms for optimal data placement in a robotic storage library.

Several researchers have proposed integrating an SQL database with tertiary storage. Issac [13] proposes an architecture for integrating a general purpose DBMS with tertiary storage, by using a hierarchical storage manager to fetch large objects. Moran and Zak [22] describe an experimental database in which the DBMS fetches individual blocks from tertiary storage.

Sarawagi [30] proposes an architecture by which SQL queries on tape resident data can be processed. A relation is divided into partitions, where a partition is stored contiguously on tertiary media. A database access, such as a join, is transformed into a sequence of operations on disk-resident partitions. In [29], Sarawagi and Stonebraker give query optimization techniques for complex SQL queries. Myllymaki and Livny [23, 24, 25] have investigated disk-to-tape and tape-to-tape join algorithms that do not make use of indices.

In [14], Johnson proposes an architecture for a decision support data warehouse that uses tertiary storage. A method for indexing detail data is given in [15]. Chatziantoniou and Johnson [2] propose a language for a decision support queries on a tape resident data warehouse. The authors show that complex decision support queries can be computed using a few long sequential scans through the detail data. In [3], Chen et al. show how to use an analysis of queries on a scientific data set to lay out a multidimensional array on tape.

While some work has been done to measure, model, and classify the performance of tertiary storage devices, broad based and comprehensive models have not appeared. Comprehensive models of secondary storage (i.e., disk drives) have been published [28]. Many works on systems incorporating tertiary storage include benchmarking studies [30, 36, 24, 3, 7, 12]. Other studies have focused on an aspect of a particular device. Ford and Christodoulakis [6] model optical continuous linear velocity disks to determine optimal data placement. Hillyer and Silberschatz [9] give a detailed model of seek times in a DLT 4000 tape drive, to support a tape seek algorithm [10]. In [11], they take measurements of an IBM 3570.

In this paper, we measure a variety of devices used in tertiary storage systems and present performance characterizations of these devices. The contribution of this work is the scope and detail of our measurements and modeling. We measure aspects of every phase of robotic tape access, including robot arm fetch time, mount time, seek time, transfer rates, rewind time, and unload time. The devices we measure include high, medium, and low performance drives, and large, medium, and small capacity robot libraries. We include measurements that have not previously been published (to our knowledge) but are vital to efficient database implementations, such as short seek times and the effect of delays on transfer rates. Based on our measurements, we provide simple performance characterizations and relate them to issues in building a tertiary storage DBMS.

3 Taxonomy

The technology used to implement a tape drive influences the performance that the user will obtain from the drive. In this section, we discuss the technologies used to build common tape drives. For a deeper discussion of these matters, we refer the reader to the discussions in conferences such as the joint IEEE Mass Storage System Symposium / NASA Goddard conference on Mass Storage Systems and Technologies.

A fundamental characteristic of a tape drive is the layout of data on the tape. To achieve a high density, the tape drive must use as much of the available surface area as possible, and a tape is typically much wider than the data tracks. A helical scan tape writes data tracks diagonally across the tape surface, and packs the diagonal tracks tightly together (e.g., as in a VHS video cassette). A linear tape lays multiple sets of data tracks across the tape. Typically, the data tracks alternate in direction, hence the name "serpentine" (e.g., an audio cassette with autoreverse).

The tape package can be a cartridge (containing 1 reel) or a cassette (containing 2 reels). The tape in a cartridge must be extracted from the cartridge before the tape mount can complete. In addition, the tape cartridge must be rewound before it is unmounted. A cassette can be removed from the tape drive without being rewound. However, the tape must be positioned at a special zone (a "landing zone") to ensure that data is not exposed to contaminants. If the tape drive does not support landing zones, the cartridge must be rewound.

The geometry of a tape makes defining the position of a particular block more difficult that for disk drives. Data storage tapes typically embed some kind of directory to expedite data seeks (this directory is implemented in hardware and is separate from any user-created directory). These directories can be written at the beginning of the tape (or at other special tape positions), in special directory tracks, or in silicon storage devices mounted on the tape package. A precise directory can permit high-speed seeks. In addition, the requirement to read a directory area can increase the mount time, and the requirement to write a directory area can increase the unmount time.

Many tape drives use hardware data compression to increase their capacity and to improve their data rates. However, compressed data is variable sized. Since the location of a block can vary widely, fast seeks can be more difficult to implement. Similarly, a variable size record length increases the flexibility of a tape drive, but can lead to increased seek times.

Some tape drives allow the user to partition the tape into distinct regions. The Ampex DST 310 tape drive allows partitioning. The partitioning simplifies some data management functions, and does not have a significant effect on performance. Some serpentine tape drives that support partitioning can improve seek times within a partition[1]. However, we do not have such a device available for testing.

Other factors that can affect performance are the tape transport implementation and the use of caching. Helical scan tape drives need to wrap the tape around the read/write head. Performing a high-speed seek requires that the tape be moved away from the head to prevent excessive wear – resulting in a large delay in starting the seek. Linear tapes use a simpler transport and do not suffer this problem. Data caches in the tape drive allow the drive to remain in a streaming mode even if the host machine suffers occasional delays in submitting read or write requests. A tape drive will typically read ahead of requested blocks. Some drives will return prefetched data after short block seeks.

There are many other considerations involved in tape drive technology, especially those of reliability and longevity, that we do not address in this paper. Another important consideration is cost. Some of the drives we measure in this paper can have an order of magnitude better performance than others; however, this performance advantage is usually reflected in an order of magnitude higher price tag.

3.1 Summary of the Drives

Table 1 lists the characteristics of the five drives we measured. The helical scan drives use a directory track, while the serpentine drives use a directory area at the beginning of the tape. The Ampex drive allows one to unmount a tape without rewinding it. In this case, the tape is first positioned on a "landing zone" which is closer than the beginning of tape (BOT). The Sony drive also supports unmounts without rewinding. If the end of data (EOD) is closer than BOT, then the tape is advanced to EOD before unmounting.

4 Methodology

Our interest is to measure and develop performance models for the following access characteristics listed below. Taken together, they summarize the end-to-end performance of a tertiary storage device.

Robotic arm access time : This is the time required for the robot arm to move a tape from the shelf to the drive, or from the drive to the shelf.

Mount time : This is the time from when the robot arm has placed the tape into the drive to the time when the tape is "ready". (i.e., the special file for the drive is open and operations can be performed without incurring I/O errors).

Seek time : This is the time from when a seek command is issued to the time when the seeked-to data block can be read into memory (the seek system call might return before the read operation can be initiated). We measure three particular types of seeks:

[1]E.g., the IBM 3570 drive and future models of the IBM 3590. See www.storage.ibm.com.

drive	layout	package	compression	block size	cost	unmount from midtape
4mm	helical scan	cassette	yes	variable	low	no
Ampex DST 310	helical scan	cassette	no	fixed	high	yes
Sony DTF	helical scan	cassette	yes	variable	high	restricted
DLT 4000	serpentine	cartridge	yes	variable	medium	no
DLT 7000	serpentine	cartridge	yes	variable	medium	no
IBM 3590	serpentine	cartridge	yes	fixed	high	no

Table 1: Summary of the drives in the study.

1. **Long seek from Beginning Of Tape** : We measure the time to seek to an arbitrary location in the tape.

2. **Long seek from the middle of the tape** : We measure the time to seek from one arbitrary location on the tape to another arbitrary location. Since this requires $O(B^2)$ measurements (where B is the number of tape blocks), we pick representative locations on the middle of the tape.

3. **Short seek from the middle of the tape** : A seek is expensive to initiate on most tapes. The behavior of a seek for a short distance can be very different from that for a long seek.

Transfer rate : This is the rate (Mbytes / second) at which the tape drive will service read or write requests. This rate can be influenced by the compressibility of the data, the record size, and by the time between successive requests to for tape reads (writes).

Unmount time : This is the time from the request to when the tape can be extracted from the drive by the robot arm.

Compression rate : This is the tape capacity when compression is turned on as compared to its capacity when compression is turned off.

While we have tried to make our measurements as consistent as possible from platform to platform, we have needed to take special measures for some of the devices (e.g., the API for requesting a tape mount was different on each platform). We tested the devices on a wide variety of platforms, each with its own local environment. However, in all cases the tape drive is attached to SCSI bus. Also, some devices have special characteristics (e.g., compression, seek location hints, partitioning, etc.). Finally, we had access to some devices for a limited time only. In all cases, we performed our measurements on multiple tapes. When the equipment was available, we also tested multiple drives. In all cases, the measurement results using different tapes (except for failing tapes) and different drives were almost identical. We show only one chart per measurement study and drive because of space limitations.

5 Comparison

In this section, we present our measurements of a variety of tertiary storage devices. We summarize the measurements of the tape drives in Table 4 for convenience. The subsequent sections will discuss the meaning of individual columns. Blank entries indicate that the data is not available (e.g., tape drives without compression) or that we did not manage to make the measurement during the time that we had access to the equipment.

5.1 Robot Fetch

For the cases in which we have been able to measure robot arm fetch times, we have found the fetch times to be small and nearly deterministic. The location of the tape to be fetched (or returned to) has little effect on the robot arm fetch time. The results are summarized in Table 2. We have measured simple robotic storage libraries. More complex systems involving multiple tape racks, robot arms, and pass-through slots might show a more complex behavior. However, all but the most complex and expensive robotic storage libraries are similar to the ones measured here. The implication of these results is that the placement of tapes on the media shelves has only a minor impact on end-to-end performance, and can be safely ignored for all but the largest systems. Complex optimizations of tape placement [34] are not necessary.

5.2 Transfer Rates

For each drive, we measured the transfer rate of drive on uncompressed data by writing data to the tape, and then reading it back by issuing `read` system calls in a tight loop. We summarize the transfer rates that we measured in Table 4. We also list the smallest data transfer size we used to obtain the listed transfer rate.

A variable transfer size increases the flexibility of the tape drive, because the block can be sized to hold a desired number of tuples. All of the drives we tested permitted variable transfer size, although the drives with fixed block sizes (the Ampex and the IBM) require that the transfer size be a multiple of the block size. A small minimum transfer size makes some implementation details simpler, for example buffering.

Robot name	fetch		return	
	mean	standard deviation	mean	standard deviation
Grau ABBA/2 (6000 tape)	19.7 sec.	.1	16.4	.2
Storagetek 9710 (404 tapes)	approx. 9 sec	small	approx. 9 sec	small
Ampex 810 (256 tape)	2.9	.3	3.7	.5
Sony DMS-B9 (9 tape)	12.8	2.1	17.2	1.9

Table 2: Robot arm fetch and return times.

5.3 Mount and Unmount

Our measurements of mount and unmount (without a write) times showed that they are nearly deterministic. In every case, the coefficient of variation was .1 or less. We summarize our results in Table 4.

The Ampex drive permits tapes to be unmounted without rewinding. In this case, the tape is moved to the nearest "system zone" and then unmounted. The tape motion is necessary to avoid exposing tape with valid data to the elements. The Sony drive also permits unmounts without rewinding, but the tape is either rewound or moved to the End Of Data (EOD), whichever is closer.

If a tape cartridge is positioned at midtape when it is mounted, one should be able to speed up data accesses because the average seek distances are shortest. In [5], the authors show that tapes that can be unmounted in the middle, such as the Ampex, have a different optimal data layout than tapes that must be rewound before dismount.

We tested the unmount time by seeking to a random location on a tape and then unmounting (see Figure 1). The effect of the system zones can be seen in the sets of two parallel lines, offset by about 6 seconds that appear in the data. The average unmount time is 12.24 seconds with a standard deviation of 3.1 seconds.

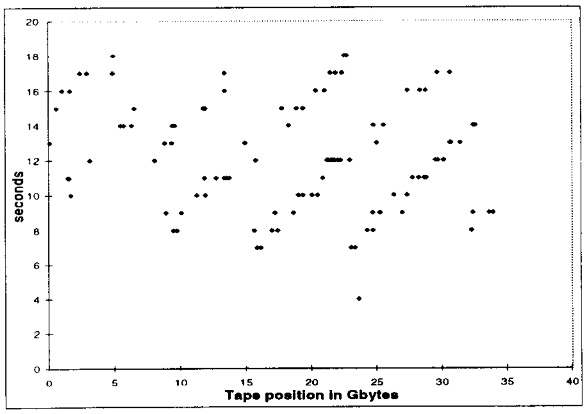

Figure 1: Time to unmount tape, Ampex 310.

If an Ampex tape is unmounted without being rewound, the first seek time increases (as will be shown in Section 5.4.1). Because the seek and rewind times on the Ampex are so fast (as will be shown), rewinding a tape before unmounting reduces access times on average. We ran an experiment of repeatedly mounting a tape, seeking to a random location, reading 1 transfer block of data, then unmounting and returning the tape We collected 60 data points the the case of rewinding before unmounting, and 60 data points for the case of unmounting without a rewind. If we rewound the tape before unmounting, then a fetch/return cycle takes 71 seconds with a standard deviation of 13. If we unmounted the tape without a rewind, the fetch/return cycle takes 85 seconds with a standard deviation of 30. A difference of means test indicates indicates a significant difference between the two quantities.

Unmounting without rewinding can be a valuable optimization in applications such as real-time data recording or backup. However, the technology does not seem to be designed for database applications. One can expect that as tape-resident databases become more common, tape drive manufacturers will make mid-tape dismounts more effective. In the meantime, one should perform careful benchmarks before applying the layout optimization described in [5].

5.4 Seek Time

The large seek times of tape drives cause them to be sequential media. Overcoming seek time delays is a major focus of system optimization research. However, seek times on tapes can exhibit unexpected behavior. In this section we measure and model three types of seeks: the first seek after a mount, a seek from the middle of the tape, and a short seek.

5.4.1 First Seek After Mount

In this section, we mostly study the seek time from the beginning of tape (because most tapes must be rewound before unmounting). The seek times of the helical scan tape drives (4mm, Sony DTF and Ampex 310) have very similar behavior. We plot their seek from BOT and rewind times in Figure 2 (the actual capacity of a Sony DTF and a Ampex 310 tape is 40 Gbytes). The seek from BOT and rewind time for a DLT 4000 is shown in Figure 3, for a DLT 7000 in Figure 4, and for an IBM 3590 in Figure 5. The Ampex drive allows the use of "positioning hints" for block positions that have already been visited. We show the seek times from BOT with positioning hints in Figure 2.

These figures clearly show the difference in seek time between helical scan tapes and serpentine tapes.

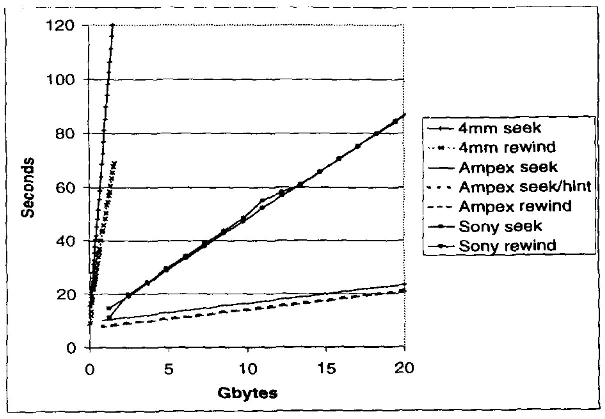

Figure 2: Seek from BOT and rewind times for the 4mm, Sony DST, and Ampex 310 drives.

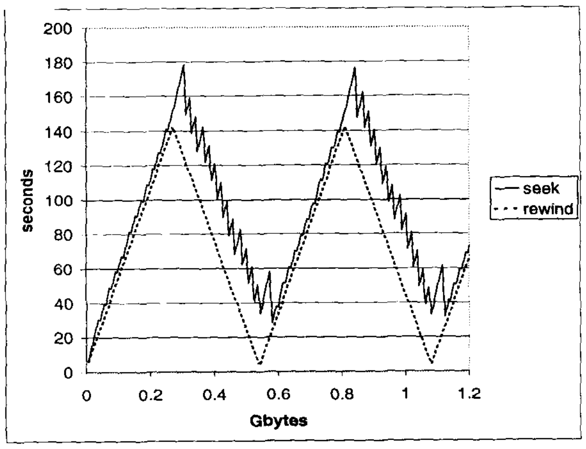

Figure 3: Seek times for a DLT 4000.

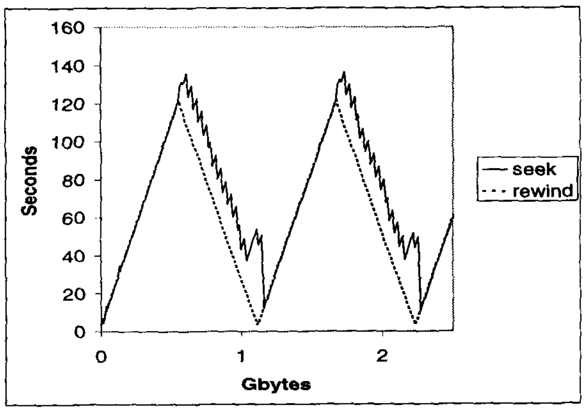

Figure 4: Seek times for a DLT 7000.

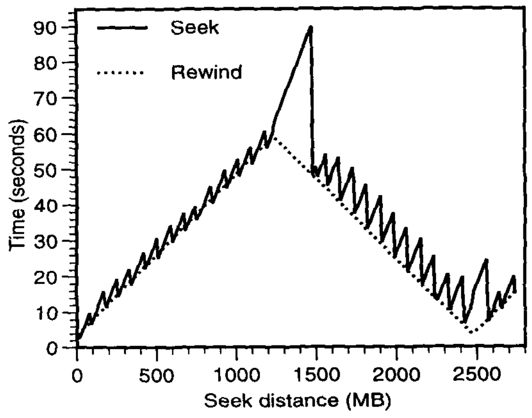

Figure 5: Seek and rewind times for a IBM 3590 (partial tape).

For helical scan drives, the seek time (and rewind time) fit well to a linear function. The seek time function of the serpentine tapes fit to a piece-wise linear function. At a first cut, the seek time to each track of the serpentine tape fits a linear function. The DLT 4000 and 7000 have 64 tracks, 32 in the forwards direction and 32 in the reverse direction. Figures 3 and 4 show only the first four of these tracks. The IBM 3590 has 8 data tracks, Figure 5 shows seek times to the first two of these tracks (the other three pairs show a similar behavior)[2].

The serpentine tape drives (DLT 4000, DLT 7000 and IBM 3590) use the two-dimensional topology of the tape as the primary mechanism for implementing a high-speed search. The fast seek speed is only 1.5 times, 1.1 times and 2 times faster than the read speed for the DLT 4000, the DLT 7000, and the 3590, respectively. The drives store the location of particular blocks in the tape's directory. To implement a distant seek, the tape moves to the last known block position that occurs before the desired block using the high-speed movement, then performs reads the tape until the desired block is located.

[2]The very large seek time at the beginning of the reverse track is probably caused by a firmware bug

It is possible to derive a simple empirical model of seek times on both helical scan and serpentine drives, by using linear regression to fit linear functions. Serpentine tapes require piece-wise linear functions, fitted to the nearly-linear regions in the seek time curve. A summary is given in Table 4. One can derive similar models for rewind times, see [17] for details.

The Ampex 310 can be unmounted without a rewind (as discussed in Section 5.3). When the tape is mounted again, it is positioned at the middle of the tape and therefore is closer to the desired first seek position. In Figure 6, we also measured the time to perform the first seek after a mount. The block positions for the first seek were randomly selected. We recorded the block position at unload and the block position for the first seek, as well as the seek time. However, we did not find any correlation between the distance between the block positions and the seek time (as is required by the optimization discussed in [5]). Instead, the seek time seems to be correlated with the seek block position. Figure 6 shows the result of the experiment. The time to seek to a block if no rewind is done before a mount is considerably larger (i.e., 25 seconds larger) than the time to perform a seek on a tape that has been rewound. Since most rewinds take less than 25 seconds (64% of the tape), we found that it is faster on average to rewind tapes after use.

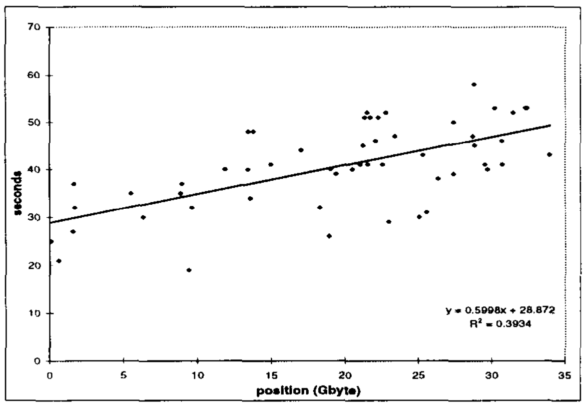

Figure 6: Seek time from mount, Ampex 310.

5.4.2 Seek times from Mid Tape

For the helical scan tapes, seek times between distant blocks in the middle of the tape fit well to a linear function. Figure 7, which shows seek times to and from a fixed block position for a 4mm tape drive, is representative of helical scan drives. It is easy to fit a linear regression model to mid-track seek times, the coefficients derived for the BOT seek times are sufficiently accurate.

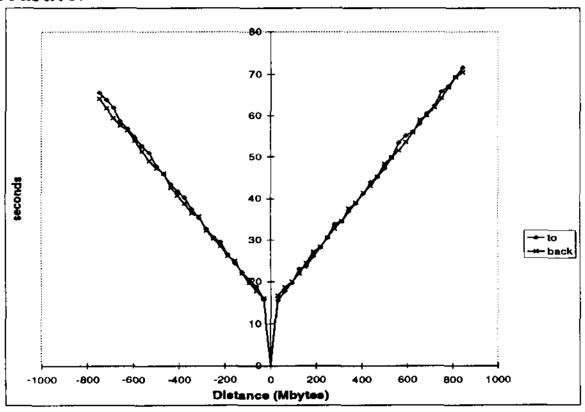

Figure 7: Seeks from the middle of the tape on a 4mm drive.

Seek times on a serpentine tape follow a more complex topology. The starting position divides the tape into four regions : 1) after the starting point on a same-direction track, 2) before the starting point on a reverse-direction track, 3) after the starting point on a reverse-direction track, and 4) before the starting point on a same-direction track (see Figure 9). An approximate model for serpentine tape drives can be found by fitting a piece-wise linear regression model to each of the four regions. Let L be the length of a data track (the actual length in bytes depends on the compression ratio), and let S be a seek position. Let $T = \lfloor S/L \rfloor$ be the track of S, and let $p = (S - L*T)/L$ be the position of S on the track. An example model of the DLT 4000 drive is shown in Table 3 (p_0 and T_0 are the position and track of start position, and p' and T' are the position and track of the destination). Figure 8 shows mid-track seek times for a DLT 4000 tape drive, along with the model predictions. As in the BOT seek models, the coefficients do not necessarily match the fast seek rates.

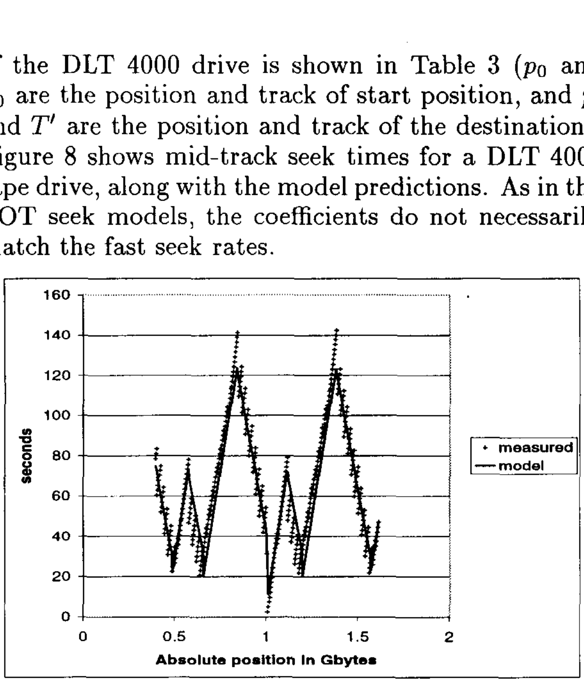

Figure 8: Seeks from the middle of the a reverse track on a DLT 4000 drive.

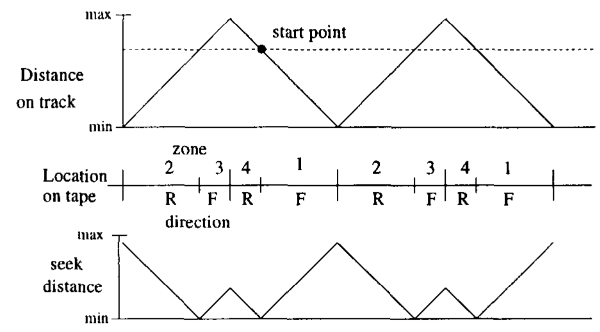

Figure 9: Seek time topology for serpentine tape.

5.4.3 Short Seeks

The increasing interest in building databases that query directly from tape [25, 15, 14, 33] points to the need investigate the performance of short seeks on tape. Data placement, indexing, and sizing algorithms have been developed with the assumption that seek time is directly proportional to seek distance. Often this is not the case. Furthermore, the seek time function over a short distance can be significantly different that the seek time function over a long distance.

Because a tape seek often incurs a substantial delay, an important characterization is the distance at which it is faster to seek to the desired block position than it is to read in the unnecessary blocks. We ran a set of experiments where we would repeatedly read in K blocks, and another set of experiments where we should seek past $K - 1$ blocks and read in one block. The results are shown in Figure 10 for a DLT 4000, Figure 11 for an IBM 3590, Figure 12 for an Ampex 310, and Figure 13 for a Sony DTF.

DLT 4000	$seek(p',T') =$	$21 + 154 * \lvert p_0 - .04 - p' \rvert$	$T' + T_0$ is even, $p' > p_0 - .04$
		$31 + 132 * \lvert 1 - (p_0 - .04) - p' \rvert$	$T' + T_0$ is odd, $p' \leq 1 - (p_0 - .04)$
		$21 + 154 * \lvert 1 - (p_0 - .04) - p' \rvert$	$T' + T_0$ is odd, $p' > 1 - (p_0 - .04)$
		$31 + 132 * \lvert p_0 - .04 - p' \rvert$	$T' + T_0$ is even, $p' \leq p_0 - .04$

Table 3: Seek time functions from the middle of the tape.

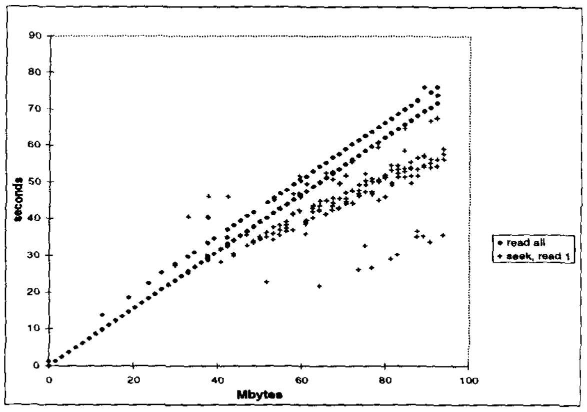

Figure 10: Short seek times versus read times, DLT 4000.

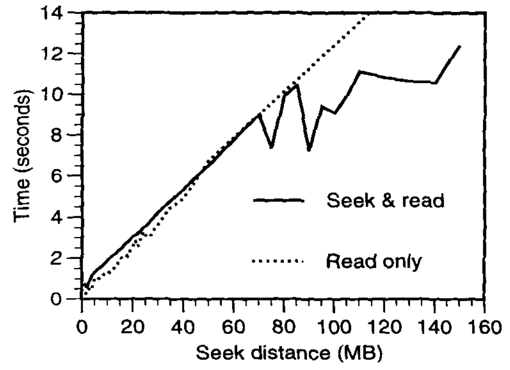

Figure 11: Short seek times versus read times, IBM 3590.

A DLT 4000 does not incur a penalty for performing a short seek, but there is no benefit of performing a seek less than 43 Mbytes. The IBM 3590 shows a similar behavior, with a transition point at 100 Mbytes. A DLT 7000 (not shown here) has a fast seek rate that is only slightly faster than the read rate, with a crossover point at about 150 Mbytes.

It is faster to read up to 1.6 Mbytes of data than it is to do a seek on a 4mm DAT (not shown). For very short seeks, the drive returns cached blocks. We note that the seek time function is different for short seeks than for long seeks, being

$$seek(X \text{ Gbytes}) = \quad 436X + 4.61 \text{ seconds}$$

The Ampex 310 and the Sony DTF do not have short seek time behavior that is radically different than the long seek behavior. However, the Sony DTF will attempt to perform a read instead of a seek for sufficiently short seek. The Ampex 310 must seek past 110 Mbytes and the Sony DTF must seek past 100 Mbytes of data before performing a seek is faster than performing reads.

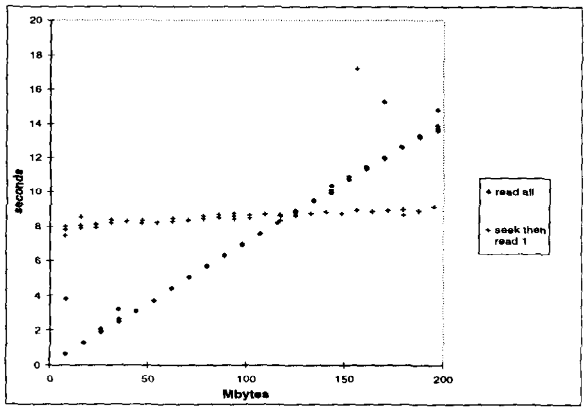

Figure 12: Short seeks versus sequential reads, Ampex 310.

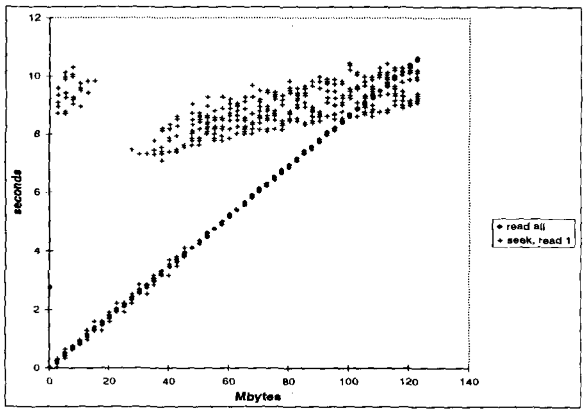

Figure 13: Short seeks vs. sequential reads, Sony DTF.

5.5 Delays in Performing Reads

It is commonly accepted that buffering is required to obtain a high data throughput for tape drives. Data transfer requests (e.g., reads) must be quickly delivered to the tape drive to keep the drive in a "streaming" mode. The manuals on the helical scan tape drives note that if no data transfer requests are made within a timeout period, the tape is disengaged from the tape head to reduce tape wear.

To test the effect of delays on transfer rate, we ran the following experiment. We repeatedly requested the read of 1 block of data, and then performed a timed delay when the read system call returned. This experiment models data processing by repeatedly loading a block of tuples, then processing each tuple in the block. We recorded the response time of the system call, and computed the transfer rate from tape to be the Mbytes per block divided by the response time plus the delay. The results are shown in Figure 14 for the DLT 4000 drive, and in Figure 15 for the Sony DTF.

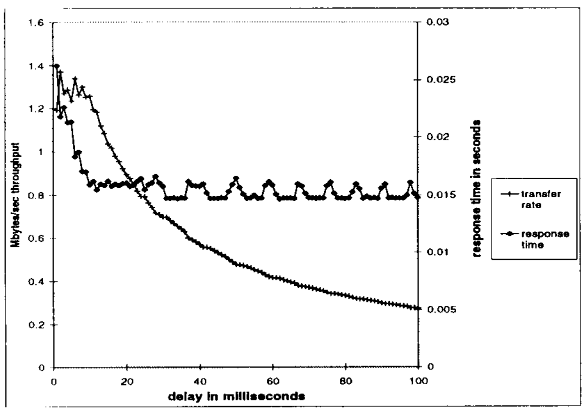

Figure 14: Read throughput versus delay in submitting read requests, DLT 4000.

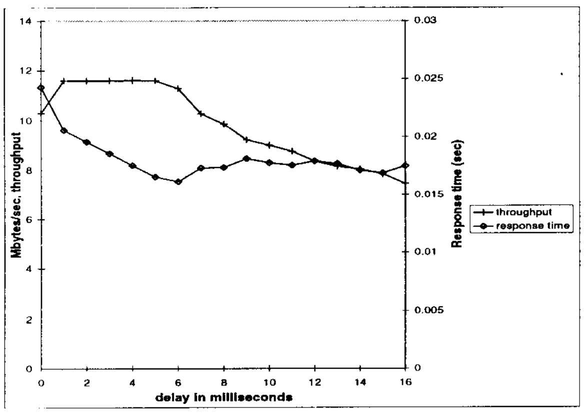

Figure 15: Read throughput versus delays in read requests, Sony DTF.

Each of the drives can tolerate a small delay without a reduction in the transfer rate. The response time for a read request actually decreases with an increasing delay in submitting requests, most noticeable in the DLT 4000. Each of these drives incorporates a sizable internal buffer, and performs read-ahead caching of data to keep the tape in streaming mode. The DLT 4000 will tolerate a delay of 10 milliseconds, the Ampex 310 (not shown) a delay of 6 milliseconds, and the Sony DST a delay of 6 milliseconds without a decrease in read throughput. We did not find that even 10 second delays increased the response time for reading one block of data. Modern drive incorporate a sufficient amount of cache to hide tape disengagement delays for small reads.

As a further test, we read a large block of data then performed a timed delay. This experiment models data processing by loading a data set, and then processing it. Our hypothesis is that asking to read more data than is cached will force a the tape drive to incur a start-up delay before completing the data transfer. For the DLT 4000 tape drive, we observed no startup delays even for reads of 100 Mbytes and inter-request delays of 100 seconds. For the Ampex drive, we observed startup delays for reads of 70 Mbytes and larger, and inter-request delays of 32 seconds and larger. We note that the Ampex drive has a 64 Mbyte cache. The startup penalty is approximately 7 seconds.

A tape-based database must obtain the maximum throughput possible from its tape drives. Our measurements of read throughput versus delays in submitting reads verifies the "common knowledge" that high-performance tape drives need a multi-threaded buffering data reader to ensure that transient delays in processing a block of data (busy disk, cache miss on a join, etc.) does not slow down tape data access. However, the situation is not as dire as has sometimes been reported [25]. Even in an access pattern that can incur startup delays (repeatedly reading a large block then processing it for a long time), one can avoid penalties by reducing the transfer size to fit within the tape drive cache size.

5.6 Compression

Many tape drives incorporate hardware compression to increase the capacity of tapes. A side benefit can be to increase the transfer rate. Compression is easier to implement in tape drives than for disk drives because tape drives are inherently sequential media, and are therefore can more easily handle the variable sized compressed blocks. However, the hardware compression introduces uncertainty – tape capacity, seek times, and transfer rates are all dependent on the tape compression ratio.

We measured the compressibility of a collection of data warehouse files (including binary as well as ascii files)[3] by compressing the file with gzip, and computing the compression ratio to be the size of the uncompressed file divided by the size of the compressed file. We computed the compression ratio of the tape drive to be the capacity of a tape with compression turned off divided by the capacity of the tape with compression turned on. We also measured the transfer rate to and from tape. The results are shown in Figures 16 and 17.

We found that the the compression ratio achieved by the tape drives fits well to a linear regression on the gzip compression ratio. The compression ratio achieved by the drives is about half that achieved by gzip. The precise linear regression models are shown

[3]We measured the Sony DTF and the DLT 4000 at different sites. Local security concerns prevented us from transferring files and using the same test suite at both sites. We did test files from the Canterbury Corpus at both sites (http://corpus.canterbury.ac.nz).

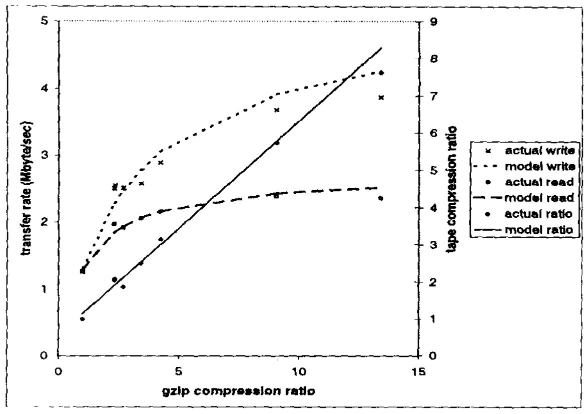

Figure 16: Tape compression performance, DLT 4000.

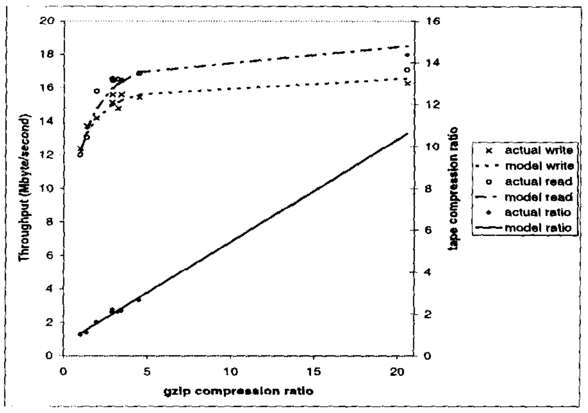

Figure 17: Tape compression performance, Sony DTF.

in Table 4.

The read and write transfer rates do not scale with the compression ratio, and instead seem to approach an asymptote. Let TH be the throughput and let C_{gzip} be the gzip compression ratio. Then, we hypothesize the that TH and C_{gzip} are related by $TH = 1/(a + b/C_{gzip})$ Mbytes/sec. That is, the transfer time (for a fixed block size) is a fixed cost plus a transfer cost, and the transfer cost decreases with the compression ratio. The throughput is the inverse of the transfer time. We used linear regression to fit the data to this model and got the following results (with R^2 values of .963, .971, .933, and .903, respectively): $a_{dlt4000,read} = .367$, $b_{dlt4000,read} = .404$, $a_{dlt4000,write} = .192$, $b_{dlt4000,write} = .572$, $a_{DST,read} = .0526$, $b_{DST,read} = .0293$, $a_{DST,write} = .0593$, $b_{DST,write} = .0219$.

The two tape drives in this study have significantly different compression performance. In addition to the absolute differences, we note that the DLT 4000 drive performs compressed writes faster than compressed reads, while for the Sony DTF drive the opposite is true. However, the compression performance of both drives fits well to simple linear regression models. While these models are not perfectly predictive, they provide a good first approximation to the performance of the drive on a new data set.

The performance of tape drive compression leads to the following conclusion. While one should always use tape compression (turning off compression did not improve performance even for gzipped files), it is best to compact or compress data to the greatest extent possible before writing it to tape. The primary reason is that transfer rates do not scale with compression. One needs to pre-compress the data to maximize the tuples (frames, etc.) per second transferred. However, the pre-compression can be a simple and fast compression [27], the tape compression will make the pre-compression efficient. In addition, pre-compression can increase the volume of data that can be written to tape.

The caveat to using tape drive compression is the increase in the unpredictability of tape drive performance. In particular, one is no longer able to use a-priori complex models of seek times for seek scheduling [9, 10]. Alternatives are to use crude but insensitive models of seek times (e.g., Table 3), or to turn off tape compression.

5.7 Performance Summary

Our performance measurements permit us to characterize the drives in this study. The rows in Table 4 summarize the measurements in this section. To summarize long seek performance we use two numbers, the fixed overhead of performing a seek and the additional time to seek across a full tape. For helical scan tapes, these values are easily derived from the seek time function. However, these values are also a good characterization of serpentine tapes. The fixed overhead is one half of the penalty of a seek to a block on an opposite track, and the full tape seek time is the full track seek time.

One example application of these summaries is to determine how important is optimal file placement (see [5]). On the Ampex DST, file placement is not very important because the seek startup cost is 40% of the full tape seek cost. So, one needs to seek almost halfway across the tape before doubling the seek time. For the Sony DTF, however, file placement is important because the seek startup cost is only 7% of the full tape seek cost.

For another example, we build a simple model of how much data should be fetched per tape mount. Let us assume that we want the time to transfer the data from tape to be at least equal to the time to fetch a tape, mount it, and seek to the file (we assume that the tape is unmounted and returned to the shelf during an idle period). To simplify the study, we assume that we are fetching only one file from tape, and that the time to perform the seek is the seek startup time plus half of the full tape seek time (this assumption expresses the idea of seeking to a random location on the tape

measurement	4mm	Ampex DST	Sony DTF	DLT 4000	DLT 7000	IBM 3590
mount time (seconds)	50	10.1	51	40	39	16.5
unmount time (seconds)	21	4	18	21	12	23
transfer rate (Mbyte/second)	.325	14.2	12.0	1.3	4.3	8.9
transfer size (Kbytes)	16	1024	512	32	32	512
seek startup (seconds)	13	9.6 (7.4)	10	21	13	6
seek full tape (seconds)	110	26.2	144	147	119	60
min. seek (Mbytes)	1.6	110	100	43	150	100
max. read delay (ms)		6	6	10		
compression rate			$.58 + .488 * C$	$.56 + .577 * C$		
max. compressed read transfer rate (Mbyte/sec)			19.0	2.7		
max. compressed write transfer rate (Mbyte/sec)			16.9	5.2		

Table 4: Performance summary.

for both helical scan and serpentine tape). We assume that the time to fetch a tape is 10 seconds, except in the case of the Ampex which was mounted in a considerably faster tape robot.

These overheads are summarized in Table 5 for each drive. To find a minimum reasonable file size to load, we multiple the length of time to reach the start of the file by by the transfer rate. Finally, we show the minimum reasonable file size as a fraction of the total tape capacity. This table shows that high-throughput drives need to read large amounts of data per tape to be effective. A corollary is that high-performance drives should have high-capacity tapes.

6 Conclusion

We have taken detailed measurements from six common tertiary storage tape drives. These drives include low, medium, and high performance drives, linear and serpentine data layouts, and cassette and cartridge tape packages. Based on the measurements, we derive simple empirical performance models of common tape access activities.

The problem of integrating databases with tertiary storage is the subject of much recent research activity. The measurements that we have taken indicate a number of implementation considerations, such as the problem of short seeks, parameters for buffering tape reads, and issues in handling compressed data. Without a detailed performance measurement study, these issues would not be obvious.

Tape drive technology continues to improve, and new drives will have different performance characteristics. For example, recent technology includes the partitioning of serpentine tapes, and technology for faster tape mounts (see recent reports at http://www.thic.org/thic/). Hillyer and Silberschatz [11] show the seek response time of an IBM 3570, a serpentine tape drive with some of these features.

However, we expect that a performance summary such as Table 4 can still be applied, and that new drives can be benchmarked using experiments similar to the ones in this study. We refer the interested reader to [16] for a discussion of benchmarking issues. Some of the programs we used for the benchmarking study can be found at `http://www.csee.umbc.edu/~elm`.

7 Acknowledgments

We'd like to thank all of the people who gave us access to the tape drives and helped get the experiments set up. They include Jim Ohler and Jim Finlayson at the Department of Defense, P. C. Hariharan, Jeanne Behnke, and Joel Williams at the Mass Storage Testing Laboratory at NASA Goddard, and Gary Sagendorf at AT&T Labs - Research.

References

[1] P. Brown and M. Stonebraker. BigSur: A system for the management of earth science data. In *Proc. 21st VLDB Conf.*, pages 720-728, 1995.

[2] D. Chatziantoniou and T. Johnson. Decision support queries on a tape-resident data warehouse. In submission, 1998.

[3] L. Chen, R. Drach, M. Keating, S. Louis, D. Rotem, and A. Shoshani. Efficient organization and access of multidimensional datasets on tertiary storage systems. *Information Systems*, 20(2):155 - 183, 1995.

[4] S. Christodoulakis. Multimedia databases. In *Intl. Conf. on Very Large Data Bases*, 1997. tutorial.

[5] S. Christodoulakis, P. Triantafillou, and F. Zioga. Principles of optimal data placement in tertiary storage libraries. In *Proc. 23rd Very Large Database Conf.*, pages 236 - 245, 1997.

[6] D. Ford and S. Christodoulakis. Optimal placement of high-probability randomly retrieved blocks ion CLV optical disks. *ACM Tans. on Information Systems*, 9(1), 1991.

[7] L. Golubchik, R. Muntz, and R. Watson. Analysis of striping techniques in robotic storage libraries. In *Proc. 14th Conf. IEEE Mass Storage Systems*, pages 225-238, 1995.

drive	File location overhead	transfer rate	minimum file size	% of tape
4mm	128	.325	42	2.6
Ampex DST 310	35.7	14.2	507	1.2
Sony DTF	143	12.0	1720	4.2
DLT 4000	145	1.27	184	1.0
DLT 7000	121	4.3	520	1.5
IBM 3590	61	8.9	543	5.3

Table 5: Performance of file fetches.

[8] J. Gray and G. Graefe. The five-minute rule ten years later, and other computer storage rules of thumb. *SIGMOD Record*, 26(4):63–68, 1997.

[9] B. Hillyer and A. Silberschatz. On the modeling and performance characteristics of a serpentine tape drive. In *ACM SIGMETRICS*, pages 170–179, 1996.

[10] B. Hillyer and A. Silberschatz. Random I/O scheduling in on-line tertiary storage systems. In *Proc. ACM SIGMOD*, pages 195–204, 1996.

[11] B. Hillyer and A. Silberschatz. Scheduling non-contiguous tape retrievals. In *Proc. NASA Goddard Conf. on Mass Storage Systems and Technologies*, pages 113–124, 1998.

[12] G. Hull and S. Ranade. Performance measurements and operational characteristics of the Storagetek ACS 4400 tape library with the Cray Y-MP EL. In *Proc. NASA Goddard Conf. on Mass Storage Systems and Technologies*, pages 229–240, 1993.

[13] D. Issac. Hierarchical storage management for relational databases. In *Proc. 12th Symp. on Mass Storage Systems*, pages 139–144, 1993.

[14] T. Johnson. An architecture for using tertiary storage in a data warehouse. In *Proc. IEEE Conf. on Mass Storage Systems / NASA Goddard Conf. on Mass Storage Systems and Technologies*, 1998.

[15] T. Johnson. Coarse indices for a tape-based data warehouse. In *Int'l Conf. on Data Engineering*, 1998.

[16] T. Johnson and E. Miller. Benchmarking tape system performance. In *Proc. IEEE Conf. on Mass Storage Systems / NASA Goddard Conf. on Mass Storage Systems and Technologies*, 1998.

[17] T. Johnson and E. Miller. Performance measurements of robotic storage libraries. In preparation, 1998.

[18] M. Kienzle, A. Dan, D. Sitaram, and W. Tetzall. Using tertiary storage in video-on-demand servers. In *COMPCOM '95*, pages 225–233, 1995.

[19] B. Kobler, J. Berbert, P. Caulk, and P. Hariharan. Architecture and design of storage and data management for the NASA earth observing system data and information system (EOSDIS). In *Proc. 14th IEEE Mass Storage Systems Symp.*, pages 65–78, 1995.

[20] A. Kraiss and G. Weikum. Vertical data migration in large near-line document archives based on Markov chain predictions. In *Proc. 23rd Very Large Database Conf.*, pages 246 - 255, 1997.

[21] L. Lueking. Managing and serving a multiterabyte data set at the Fermilab D0 experiment. In *Proc. 14th IEEE Mass Storage Systems Symp.*, pages 200–208, 1995.

[22] S. Moran and V. Zak. Incorporating Oracle on-line space management with long-term archival technology. In *Proc. 5th NASA Goddard Conf. on Mass Storage Systems and Technologies*, pages 209–228, 1996.

[23] J. Myllymaki and M. Livny. Disk-tape joins: Synchronizing disk and tape access. In *ACM SIGMETRICS*, 1995.

[24] J. Myllymaki and M. Livny. Efficient buffering for concurrent disk and tape I/O. *Performance Evaluation*, 27:453 - 471, 1996.

[25] J. Myllymaki and M. Livny. Relational joins for data on tertiary storage. In *Proc. Intl. Conf. on Data Engineering*, 1997.

[26] T. Nemoto, M. Kitsuregawa, and M. Takagi. Simulation studies of the cassette migration activities in a scalable tape archiver. In *Proc. 5th Intl. Conf. on Database Systems for Advanced Applications*, 1997.

[27] M. Roth and S. V. Horn. Database compression. *SIGMOD Record*, 22(3):31–39, 1993.

[28] C. Ruemmer and J. Wilkes. An introduction to disk drive modeling. *IEEE Computer*, 27:17 - 28, 1994.

[29] M. S. S. Sarawagi. Reordering query execution in tertiary memory databases. In *Proc. 22st Very Large Database Conference*, 1996.

[30] S. Sarawagi. Query processing in tertiary memory databases. In *Proc. 21st Very Large Database Conference*, pages 585 - 596, 1995.

[31] D. Schneider. The ins and outs (and everything inbetween) of data warehousing. In *Proc. 23rd Intl. Conf. on Very Large Data Bases*, pages 1–32, 1997. in Tutorials.

[32] M. Stonebraker. Sequoia 2000: A next-generation information system for the study of global change. In *Proc. 13th IEEE Symp. on Mass Storage Systems*, pages 47–53, 1994.

[33] P. Triantafillou and T. Papadakis. On-demand data elevation in a hierarchical multimedia storage server. In *Proc. 23rd Very Large Database Conf.*, pages 226–235, 1997.

[34] C. Wong. *Algorithmic Studies in Mass Storage Systems*. Computer Science Press, 1982.

[35] J. Yu and D. DeWitt. Processing satellite images on tertiary storage: A study of the impact of tile size on performance. In *Proc. 5th NASA Goddard Conf. on Mass Storage Systems and Technologies*, pages 460–476, 1996.

[36] J. Yu and D. DeWitt. Query pre-execution and batching in paradise: A two-pronged approach to the efficient processing of queries on tape-resident data sets. Technical report, University of Wisconson, Madison, 1996. Available at http://www.cs.wisc.edu/ jiebing/tape.ps.