Scheduling Non-Contiguous Tape Retrievals

Bruce K. Hillyer, Avi Silberschatz
Bell Laboratories, Rm. 2A-310
700 Mountain Avenue
Murray Hill, NJ 07974
{hillyer,avi}@bell-labs.com
tel +1-908-582-2262
fax +1-908-582-4623

Abstract

Large data installations normally archive relatively inactive data to a near-line tape library. The tape library performs reasonably well for sequential-access retrieval workloads. However, if the retrieval access slices across multiple data sets, or makes retrievals from scattered portions of a large data set, then the performance can suffer drastically.

In this paper, we use modeling and simulation to study several scheduling algorithms for random-access retrievals from tape. This study is based on extensive measurements of the IBM 3570 Magstar MP tape library, which is designed for fast random access. We study the retrieval performance of the Magstar MP as a function of the request size and the queue length (i.e., retrievals per tape switch). The following data point illustrates the performance. Averaged over many trials, the execution time of a schedule of 4 random retrievals from one tape (each retrieval transfers 12 MB) is 98 seconds, including the overhead of rewind, unload, eject, tape switch, and load. This schedule gives an average data rate of 0.5 MB/s, which is comparable to the average data rate of a magnetic hard disk that is fetching randomly-located pages.

1 Introduction

The traditional design of a large-scale storage system is hierarchical, using magnetic hard disks for hot data, and a tape library to archive cold data. This design reflects constraints such as cost, and practical limitations on the number of magnetic hard disks in a system. A hierarchical storage system is well-suited to scientific supercomputing installations that do not have excessive demands for retrievals from the tape archive.

In recent years, we have witnessed the emergence of application classes in which archive-retrieval workloads no longer consist of a few large sequential requests. In commercial settings, on-demand retrievals of archived documents, such as claim forms, check images, and account histories, generate a random-access workload to the tape archive. Web access to public sites can induce large numbers of concurrent accesses to an archive. Even in scientific environments, we see examples of access to widely-scattered portions of large data sets, and slicing across multiple data sets to retrieve relatively small portions. In these cases, staging in all the data to disk may be inefficient. Although tape is not considered to be a random-access medium, these examples suggest that we cannot ignore the problem of numerous, relatively small accesses to a tape library.

In previous work, we modeled the positioning time of the Quantum DLT4700 tape library [1], and studied the performance of scheduling algorithms for random retrievals from the DLT [2]. For random retrievals on the DLT, sophisticated scheduling and very large I/Os are necessary for good performance, because of the large positioning times and tape switch times. (By "tape switch time", we mean the sum of the times required by the tape drive and library to unload and eject the old tape, swap the old tape for another one, load the new tape into the drive, and wait for the drive to become ready to use the new tape.) Even with good scheduling and large I/Os, the random-retrieval performance of the DLT is relatively insensitive to the streaming bandwidth of the drive, and is largely determined by the positioning time.

In this paper, we examine the performance of the IBM 3570 Magstar MP tape library, which is designed for high-speed positioning and tape switches. The Magstar MP uses a modified serpentine track layout. As with other serpentine tape units, it is difficult to predict the positioning time between two blocks on the tape. It is not a simple function of the logical block numbers, and it is not strictly proportional to the physical distance between the blocks.

In section 2, we give a model of locate time for the Magstar MP. In measurements of 55,000 random locates, this model is accurate to within 1 second in 96% of the cases. In section 3, we describe several algorithms to schedule an efficient retrieval order for several requests from one tape. In section 4 we use our locate-time model for the Magstar MP to evaluate the retrieval performance of these algorithms. For uniformly-distributed random requests, we characterize the effective data rate of the Magstar MP as a function of the I/O request size in MB and the number of requests per tape switch. In section 5 we give concluding remarks.

2 A Model of Locate Time

A scheduling algorithm chooses a retrieval order for a set of requests to minimize the overhead of positioning the tape head from one block to the next. To develop such an algorithm, we need the ability to predict the positioning time between arbitrary blocks on the tape. In this section, we develop a formula that predicts the positioning time. Because the `SCSI LOCATE` command is used to position the tape, the model is called the *locate-time model*.

To begin, we describe the data layout used by a Magstar MP tape, and show a graph of locate-time measurements. We observe that the slope of this graph is discontinuous, and we describe algorithms to determine the "key points" at which these discontinuities occur. These key points are the input parameters of a formula that predicts the locate time between any two blocks on a Magstar MP tape. We describe the formula for the locate time of the Magstar MP, and show the results of experiments that validate the accuracy of this model.

Before ejecting a tape, most drives rewind to the beginning of the tape. The IBM

Magstar MP is different. It is designed to rewind to the middle of the tape, to a position called the *load point*.

The Magstar MP uses a modified serpentine layout. This layout first covers one half of the tape with 32 serpentine “half tracks”, called *wrap halves*, that run between the load point and one end of the tape. Then the head crosses over the load point to the other half of the tape, where 32 more wrap halves are written in a serpentine pattern.

The nominal capacity of a Magstar MP tape is 5 GB. (In this paper, the terms KB, MB, and GB denote 2**10, 2**20, and 2**30 bytes, respectively.) The effective capacity of a tape depends on the compression ratio of the data, and also on the number of defects on that particular tape. For our experiments, we filled tapes with blocks of size 32 KB. Each block contained pseudorandom bytes with the high bit of each byte masked off. The drive’s data compression hardware should be able to reduce this kind of data to about 7/8 of its original size. Over a sample of 4 tapes, the number of 32 KB blocks per tape ranged from 170878 to 180418 (5.6E9 - 5.9E9 bytes). The time to write a tape ranged from 45m58s (45 minutes + 58 seconds) to 50m37s. Thus, the effective writing rate ranged from 1.79 to 1.95 MB/s (i.e., slightly more than 2 million bytes per second).

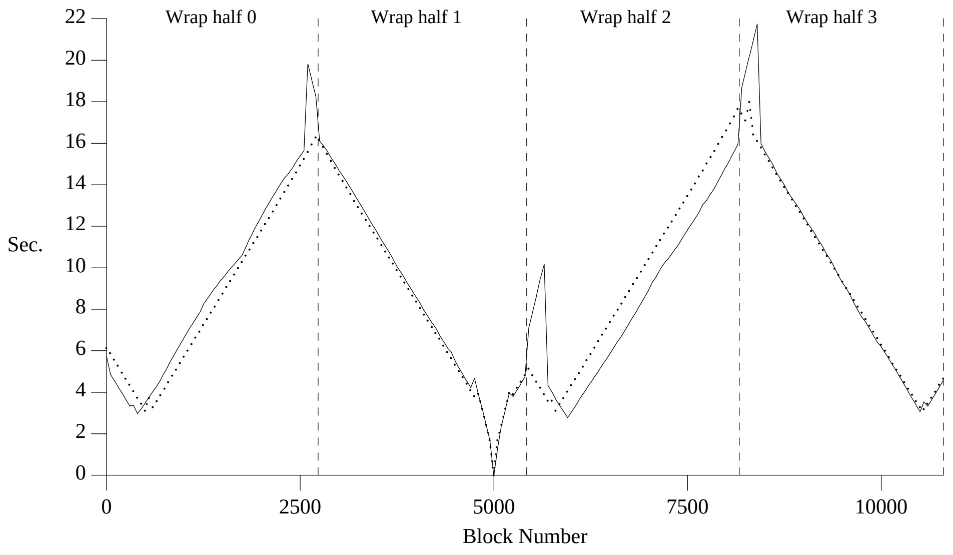

Figure 1: Locate Time from Block 5000 to Blocks 0–10,800.

Figure 1 shows a typical graph of locate-time measurements, taken on an IBM 3570 Magstar MP tape library running microcode version D1I3_27D. The host computer is a Sun Sparc-20 model 61 running Solaris 2.3. We perform the `SCSI LOCATE` command via the Solaris `USCSI ioctl()`, which enables a sufficiently privileged application to send an arbitrary SCSI command to a target. For each measurement in this graph, the starting position is logical block 5000, which is in wrap half 1, about 2/3 of the distance from the outer end of the tape back toward the load point. An experiment measured the locate time from block 5000 to blocks 0, 50, 100, 150, ..., 10800. These measurements are shown by the solid line. The reverse measurements, returning back to block 5000, are shown by the dotted line. The graph shows several interesting aspects of the Magstar MP locate time.

- If the destination of the locate is on the same wrap half as the starting point, and fairly close (i.e. within about 1/20 of the length of the tape), the locate happens at the speed of tape reading, i.e., about 2 MB/s.

- If the destination of the locate is on the same wrap half as the starting point, but further away than 1/20 of the tape, the locate happens at a higher tape transport speed.

- If the destination of the locate is physically very close to the starting point, but on a different wrap half, the time required to switch tracks and locate to the destination is nearly 3 seconds.

- If the destination is on the far side of a wrap turn, but within about 1/20 of the length of the tape, the locate requires an extra 2 seconds. It appears that the drive locates at high speed to the wrap turn, does some housekeeping for about 2 seconds, and then reads into the next wrap half. In the worst case, this behavior takes about 4 seconds longer than would be required for a high-speed locate directly to the destination.

- The graph of locate time is nicely linear, except for the extra overhead beyond each wrap turn, and the transition from normal speed to high-speed positioning. Broadly speaking, the locate time is proportional to distance traversed. (Please do not read too much into this statement. The locate time is not a simple linear function of the distance between the source and destination. In particular, given 1000 pairs of randomly-chosen logical blocks `A` and `B`, our measurement of `locate_time(A,B)` differs from `locate_time(B,A)` by more than 30% in 100 cases, and by more than 50% in 50 cases.)

From the graph, it is clear that the key points that parameterize the locate-time model will be the block numbers of the wrap turns, and the block numbers of the spikes about 200 blocks to either side of each wrap turn. Since the Magstar MP has 64 wrap halves, a total of 192 points (3 * 64) characterize the entire tape. Unfortunately, Magstar MP microcode D1I3_27D provides no documented way to ask the drive to state the logical block numbers of these key points. Nevertheless, the wrap turns can be found relatively easily, because the `SCSI REQUEST SENSE` command for this drive is documented to return the wrap half number in a field in the sense data. Thus a simple program can binary search for the exact block number of each wrap turn. First we find all the wrap turns near the load point. Next we find all the wrap turns at one end of the tape. Finally, we find all the wrap turns at the other end of the tape. A program to do this runs in about 20 minutes. Our initial attempt at a locate-time model for the Magstar MP was parameterized only by the 64 wrap turns. We were disappointed with the inaccuracy of this model: If we guess incorrectly about whether the destination is in the slow region beyond a wrap turn, we can incur an error of 4 seconds.

The locate-time model that we have developed is also parameterized by the logical block numbers of the spikes to either side of each wrap turn. To find these blocks, we measure locate times from the other side of the wrap turn, looking for the precipitous drop in locate time beyond the spike. It takes between 1.5 and 2 hours to run a program that characterizes a tape by measuring all 192 key points. C code (580 non-commentary source lines) is available at http://www.bell-labs.com/ hillyer/papers/analyze_3570.c for the body of a completely unsupported program that determines the key points of a 5 GB Magstar MP tape filled with 32 KB blocks.

The locate-time model is an ad-hoc collection of cases that describe a behavioral model of how the drive could be positioning the tape between any two logical blocks. We do not claim that the drive actually performs the tape motions specified in the model, but we do claim that the model predicts most locate times well. The C code for a completely unsupported implementation of this model, specialized for the 32 KB blocksize that we used, is available in http://www.bell-labs.com/ hillyer/papers/locate_model_3570.c (393 non-commentary source lines). A general outline of the model is given in the next paragraph—a full discussion of all the special cases is not worthy of the space that would be required.

The model is given a source logical block number and a destination logical block number. For each, the model searches the table of 192 key points and interpolates to determine which wrap half contains the block, and whether the destination is in the expensive region just beyond a wrap turn. If the source and destination are in the same wrap half, then the locate time is given by one of two linear functions: one models the reading speed used for nearby locates (slope 0.015 seconds per block, y-intercept 0.95 seconds), and one models high-speed tape positioning (slope 0.006 seconds per block, y-intercept 2.11 seconds). The model states that high-speed positioning is used for tape motion further than 230 blocks within one wrap half. Note that all of these constants depend on our 32 KB block size and the particular compression ratio of our data, so they would need to be scaled for general use.

If the source and destination are not in the same wrap half, the model uses the interpolated positions derived from the logical block numbers to calculate the time to traverse the distance at high speed, and adds appropriate amounts of time if the destination is in the expensive region just past a wrap turn, and also if the tape head crosses the load point. In the model, the time cost for a destination in the wrap-turn region is the high-speed time to the wrap turn near the destination, plus 1.6 seconds, plus the slow-speed time from that wrap turn to the destination block, and the time penalty for crossing the load point is 2.4 seconds. In addition, we can see an asymmetry in Figure 1. If the destination is in an in-bound half wrap, i.e., an odd-numbered half-wrap, an extra 0.75 seconds are required.

The model was developed to fit a random walk consisting of a sequence of 3000 locates on one particular tape. After we were satisfied with the accuracy of the model, we tested it by comparing its predictions with locate-time measurements on 4 tapes. We tested a sequence of 10,000 locates on one tape, 5000 locates on a second tape, 20,000 locates on a third tape, and 20,000 locates on a fourth tape. In all four tests, 96% of the locate times were predicted with an accuracy of 1 second or better. For each test, 6 or fewer locates had errors between 2 and 10 seconds. The remainder had errors between 1 and 2 seconds.

3 Scheduling Random Retrievals from Tape

This section first describes a collection of algorithms to schedule a set of retrievals from a single tape, and then mentions a way to schedule a set of retrievals distributed over multiple tapes. The single tape scheduling algorithms are as follows.

One no-effort schedule is FIFO, which simply retrieves the requests in whatever order they are given.

Another no-effort schedule is READ, which reads the entire tape sequentially, discarding blocks that are not in the request list. For the Magstar MP, we found the READ schedule to be the best choice when more than 1200 blocks of size 32 KB are to be retrieved from a single tape.

The SORT schedule is formed by sorting the requested logical block numbers into ascending order. This algorithm works well for a helical-scan tape or a 9-track tape, because the drive will satisfy the requests in a single sweep over the requested blocks. For a modified serpentine tape, such as the IBM 3570 Magstar MP, the sorted schedule will first satisfy all the requests on one side of the load point, and then all the requests on the other half of the tape. For a small number of requests, this approach is a little better than FIFO. For a large number of requests, the SORT schedule converges to the READ schedule.

The SCAN schedule sorts the requests by estimated physical position on the tape, using the table of key points (in particular, the logical block numbers of the wrap turns). The retrieval pattern starts at the load point, sweeps toward one end of the tape in a single pass, then reverses to sweep toward the other end of the tape, and finally back to the load point. In our experiments, we only retrieved blocks when the tape head was moving from the load point outward. A slight variation that we did not test would partition the requests into out-bound (i.e. odd-numbered) wrap halves and in-bound wrap halves. The head would follow the same path as described above, but out-bound requests would only be retrieved when the head is moving outward from the load point, and in-bound requests would be retrieved while the head is moving toward the load point.

The shortest latency time first (SLTF) schedule is the simple greedy schedule. It starts at the load point. From the set of all outstanding requests, it proceeds to the one having the shortest predicted locate time from the current position. Repeat until all requests are satisfied. For some graphs, the SLTF algorithm is too shortsighted: it greedily chooses the shortest edge now, even if that forces the use of a much longer edge in a later step.

OPT is the optimal schedule. We calculate it by using the model of locate time to predict the schedule execution time for every possible ordering of the requests: the fastest one is chosen. The number of possible orderings is exponential in the number of requests, so it is impractical to calculate the OPT schedule for more than 10 or 12 requests.

To approximate OPT for a larger number of requests, we can use the LOSS algorithm, which is a classical heuristic algorithm for the asymmetric traveling salesman problem. (The traveling salesman problem seeks to find the shortest route that passes through a set of cities: here each request is a city, and the locate-time model gives the "distance" between any two cities. The distance is asymmetric in that `locate_time(A,B)` is often significantly different from `locate_time(B,A)`.) The LOSS algorithm is presented in [3], and is also described in some detail in [2], where it is used to schedule retrievals from a Quantum DLT4000 tape. The results in section 4 will show that the LOSS algorithm for the Magstar MP does not generate significantly better schedules than SLTF or SCAN. For this reason, we only give a sketch of the algorithm here.

The basic idea of the LOSS algorithm is as follows. The algorithm processes a graph consisting of cities with directed edges between them. The major cycle of the algorithm searches for one pair of cities to be connected in the schedule. The major cycle is repeated until all the cities are connected. Within a major cycle, the algorithm considers every remaining city, and calculates the difference between the closest and second closest neighbor (separately for inbound edges and outbound edges). This difference, which is called the "loss" for the city, is a lower bound on how much worse the schedule will become if the schedule doesn't use the edge to the closest neighbor. The algorithm finds the city having the greatest loss value, and chooses the edge to its closest neighbor to be part of the schedule. When an edge is chosen, all the other edges going out of the source are removed from the graph, and all the other edges going into the destination are also removed. As soon as a city has only one inbound edge and one outbound edge, that city is fully scheduled, so it need not be considered further.

For the Magstar MP, we have not quantitatively studied the scheduling of requests that are distributed over multiple tapes. But we can mention a result from our unpublished study of another tape jukebox. An intuitively appealing technique to schedule retrievals from a tape jukebox is as follows. Whenever a drive becomes idle, iterate the following three steps. Step 1: for each tape, calculate the best schedule to retrieve all outstanding requests for that tape. Step 2: switch to the tape whose schedule gives the highest effective transfer rate (bytes divided by time). Step 3: execute the schedule for that tape. The interesting result is that for uniformly-distributed random requests, the performance of this algorithm is very nearly equaled by a much simpler algorithm: switch to a tape having the greatest number of outstanding requests, and execute the best schedule for that tape. If it is necessary to prevent starvation of requests to unpopular tapes, we can process the tapes in rounds, such that all the tapes for all current outstanding requests must be processed before any new request is added to the schedule.

4 Random-retrieval Performance of the IBM 3570 Magstar MP

The model of locate time that is described in section 2 gives an accurate estimation of the locate time between any two logical blocks on a Magstar MP tape. Using this model, it is easy to estimate the total tape locate time for a schedule of requests: just sum the locate time from the load point to the first block, and from the end of each block in the schedule to the beginning of the next one. In this first experiment, we do not include the time to rewind to the load point after the schedule is completed, and we do not include time for a tape switch.

The basic experiment evaluates the effectiveness of the scheduling algorithms by calculating the average locate time per request, as a function of the choice of scheduling algorithm and the number of requests in the schedule (denoted N). For one trial, we generate a set of N random block numbers on the tape, use each algorithm to calculate a schedule, and then calculate the total locate time of each schedule. We divide the total locate time by N to obtain the average locate time per request. We repeat this process and calculate the average over many trials. For the OPT algorithm we average over 10,000 trials for N taking each value from 1–9, we average over 1000 trials for N = 10, and we average over 100 trials for N = 12. For FIFO, SORT, SCAN, SLTF, and LOSS, we use 10,000 trials for N ranging over 1–10, 12, 16, 24, 32, 48, 64, 96, 128, and 192; we use 8000 trials for N =

256; 4000 trials for N = 384; 2000 trials for N = 512; 1000 trials for N = 768; 500 trials for N = 1024; 250 trials for N = 1536; and 125 trials for N = 2048.

To confirm that the averages are calculated over sufficiently many trials to be repeatable, the above experiments have been repeated with three different pseudorandom number generator starting seeds. Changing the starting seed alters the average locate time by less than 1% in all cases.

For each algorithm, figure 2 shows the average number of seconds per locate, as a function of N, for a schedule that starts at the load point, and ends with the tape head just beyond the last request in the schedule. We see that for the Magstar MP, the SLTF and SCAN algorithms both work well. We also see that if more than 1200 randomly distributed blocks of size 32 KB are to be retrieved from one tape, the fastest option is to read the entire tape. (Recall from section 2, that for nearly noncompressible data, a tape holds approximately 175,000 such blocks.)

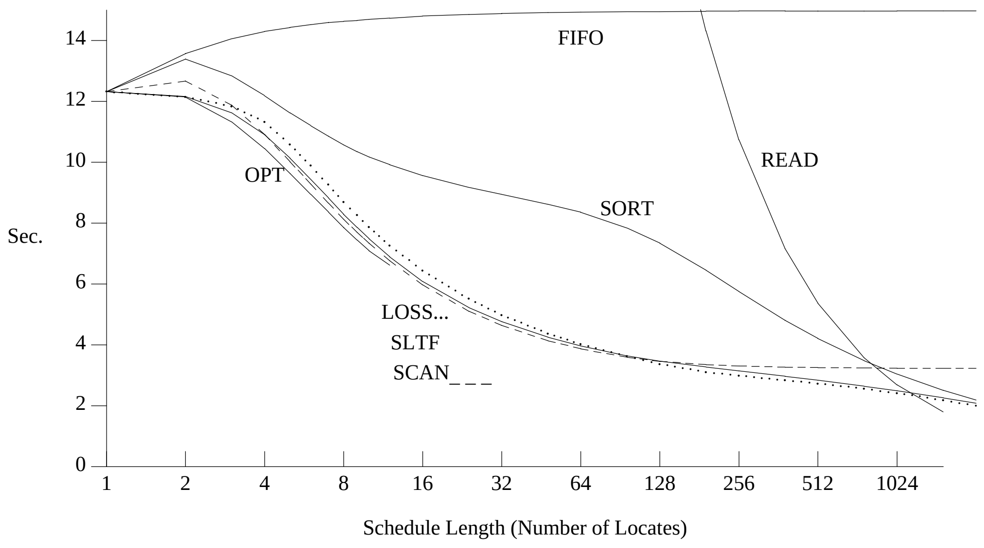

Figure 2: Average Locate Time as a Function of Schedule Length.

From the fact that the performance is nearly identical for SCAN, SLTF, and OPT, we conclude that on the Magstar MP, the speed of a schedule is largely a consequence of where the requests lie on the tape—an obvious order is likely to be a good order. This result is different from the results previously obtained for the Quantum DLT4700 tape library, for which the complex locate time enables a more sophisticated algorithm (LOSS) to give a substantial speedup [2].

The next experiment examines the additional time required when the schedule makes an excursion from the load point, through all the requested blocks, and back to the load point. For this experiment, the OPT schedule is calculated in three ways for comparison. The `OPT_no_rw` schedule is identical to the OPT schedule shown earlier: it starts at the load point, and ends just beyond the last block in the schedule, with no rewind. The second schedule, called `OPT_append_rw`, simply appends a rewind to the `OPT_no_rw` schedule. Since `OPT_no_rw` finds a minimal-time schedule through the last request, one may reasonably suspect that `OPT_append_rw` maximizes the distance of the final rewind. The third schedule, called `OPT_sched_rw`, actually calculates a minimal-time retrieval schedule that starts at the load point, proceeds through all the requests in any order, and then rewinds to the load point. Figure 3 shows the average locate time for these three variations of OPT, for schedules ranging from 1 to 10 requests. The averages are calculated over 10,000 trials for N = 1–6, 2000 trials for N = 7, 1000 trials for N = 8–9, and 100 trials for N = 10. As before, this experiment has been repeated with three different starting pseudo-random number generator seeds. For N = 10 the result varies by 2%; for the other values of N, changing the seed varies the result by 1% or less.

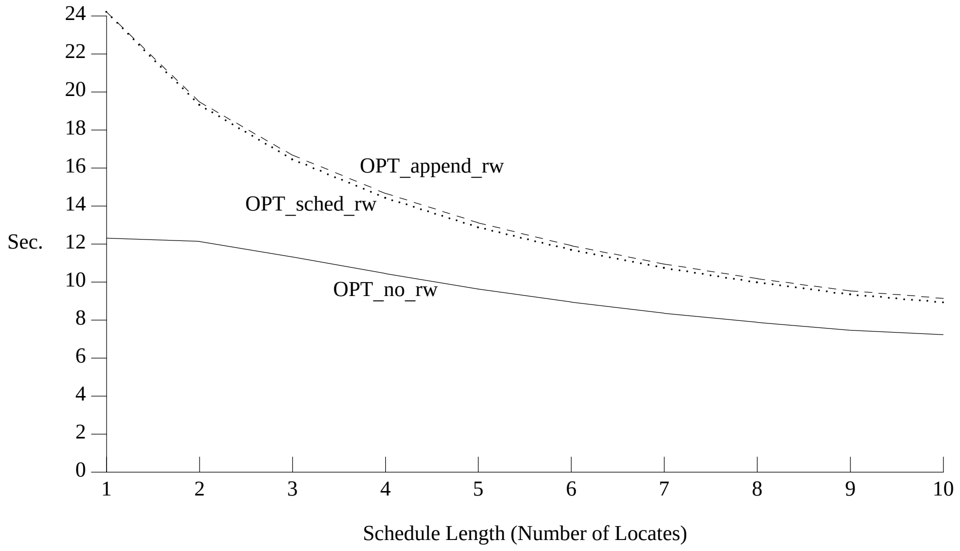

Figure 3: Average Locate Time when Appending or Scheduling a Rewind to Load Point.

From this figure, we see that appending a rewind to the schedule gives very nearly the same result as explicitly minimizing a schedule that ends with a rewind. Our explanation for this observation is that any schedule will need to traverse that subset of the tape that contains the requests—on the Magstar MP, the near-linearity of the locate-time function causes a good schedule to traverse that portion of the tape in a way that approximates a linear sweep. Whether that sweep performs I/O during the outbound trip or during the rewind is unlikely to have a large performance impact. The total rewind time increases with the schedule length, because with a larger number of random requests, there is a greater likelihood that some request will be near an outer edge of the tape, causing the rewind distance to approach its maximum.

The final topic that we study is the effective transfer rate (and hence drive utilization). We determine the relationship between the utilization and two factors: the length of a schedule (i.e., number of requests per tape), and the size of each request (i.e. number of bytes transferred by each retrieval in the schedule). The effective transfer rate is the total number of bytes retrieved, divided by the total retrieval time (the sum of the times to eject the previous tape, switch to the new tape, load that tape into the drive, locate and read through the blocks in the schedule, and rewind to the load point.) The drive utilization is the effective transfer rate divided by the streaming bandwidth. The locate time as a function of schedule length N is obtained from the best schedules in figure 2. Thus we use OPT for schedules of 1–12 retrievals per tape, SCAN for 16–96, LOSS for 128–1024, and READ for 2048–16384. The transfer time is calculated from the streaming transfer rate of 1.93 MB/s that we measured for our slightly-compressible data. The expected rewind time is obtained from the simulation output that supports figure 3. The rewind time ranges from 11.9 seconds for a schedule of length 1 through 17.1 seconds for a schedule of length 10. For longer schedules we use an approximation of rewind time that increases from 17.3 seconds through 22 seconds, which is the largest rewind time that we have measured. For these longer schedules, the rewind time is a small fraction of the total schedule time, so our approximation of the rewind time introduces an error of 2% or less. For the tape switch overhead (eject, switch, load) we use the value 16.3 seconds, as reported in [4].

Figure 4 presents a family of utilization curves for the Magstar MP, using our best scheduling algorithm for each schedule length, and including the overhead of a rewind and a tape switch for each tape schedule. A utilization of 100% would mean that the effective transfer rate equals the streaming data rate of the drive, about 2 MB/s for noncompressible data. The 25% utilization curve shows (for example) that the tape library has an effective transfer rate of about 0.5 MB/s when executing a random access workload that retrieves a single block of size 26 MB after each tape switch. The effective transfer rate is also 0.5 MB/s for a schedule that does a tape switch, retrieves 8 random blocks of size 8 MB, and rewinds.

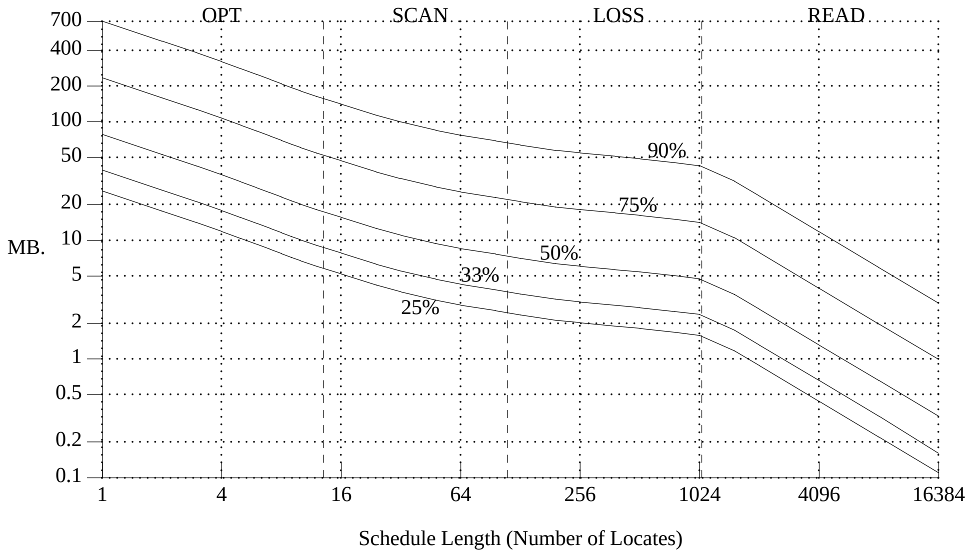

Figure 4: Drive Utilization as a Function of Schedule Length and Transfer Size.

5 Conclusion

For a workload consisting of random retrievals in a robotic tape library, a large component of the service time consists of switching tapes and the fast-forward and rewinding activity needed to access the desired data. For this sort of workload, a traditional figure of merit is the number of retrievals served per hour. We like to think of alternative metrics, namely the drive utilization (`data_transfer_time / (access_time + transfer_time)`), and the effective data rate (`bytes_transferred / (access_time + transfer_time)`). The drive utilization indicates how efficiently the tape library is being used, and the effective data rate indicates whether the library is producing data fast enough to keep the CPU and application busy.

We have studied the performance of the IBM 3570 Magstar MP tape library, which is designed for fast tape switching and positioning times. We have developed a way to predict the positioning time between any two logical blocks on a tape that is filled with fixed-size blocks. Based on this model of locate time, we have studied the performance of several scheduling algorithms for a random retrieval workload on the Magstar MP, and have produced estimates of the effective transfer rate as a function of the two factors: the number of retrievals scheduled from a tape, and the number of bytes transferred by each retrieval. Our estimates are conservative, in that they are based on uniformly-distributed random requests. If the requests for a tape exhibit spatial locality (i.e., if the blocks are somewhat clustered, rather than randomly scattered across the tape), the overhead of tape positioning may decrease.

6 Acknowledgment

The measurements reported in this paper have been conducted on an IBM 3570 Magstar MP tape library loaned to us by IBM. We thank John Gniewek, of IBM Tucson, for arranging this.

References

[1] 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*, Philadelphia, PA, May 23–26 1996.

[2] B. K. Hillyer and A. Silberschatz. Random I/O scheduling in online tertiary storage systems. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data,* pages 195–204, Montreal, Canada, June 3–6 1996.

[3] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys. *The Traveling Salesman Problem*. Wiley, Chichester, 1985.

[4] Strategic Research Corporation, 350 So. Hope Ave., Suite A-103, Santa Barbara, CA 93105. *Demystifying tape performance, http://www.sresearch.com/search/105420.htm*, 1996.