[8] A. L. Drapeau and R. Katz. Striped Tape Arrays. In *Proc. of the 12th IEEE Symposium on Mass Storage Systems*, pages 257–265, Monterey, California, April 1993.

[9] A. L. Drapeau and R. Katz. Striping in Large Tape Libraries. In *Supercomputing '93 Proceedings*, pages 378–387, Portland, Oregon, November 1993.

[10] Robert Hyer, Rich Ruef, and Richard Watson. High-Performance Data Transfers Using Network-Attached Peripherals at the National Storage Laboratory. In *Proc. of the 12th IEEE Symposium on Mass Storage Systems*, pages 1–10, Monterey, California, April 1993.

[11] L. Kleinrock. *Queueing Systems, Volume I.* Wiley-Interscience, 1975.

[12] J. D. C. Little. A proof of the queueing formula $l = \lambda w$. *Operations Research*, 9:383–387, May 1961.

[13] David A. Patterson, Garth Gibson, and Randy H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). *ACM SIGMOD Conference*, pages 109–116, 1988.

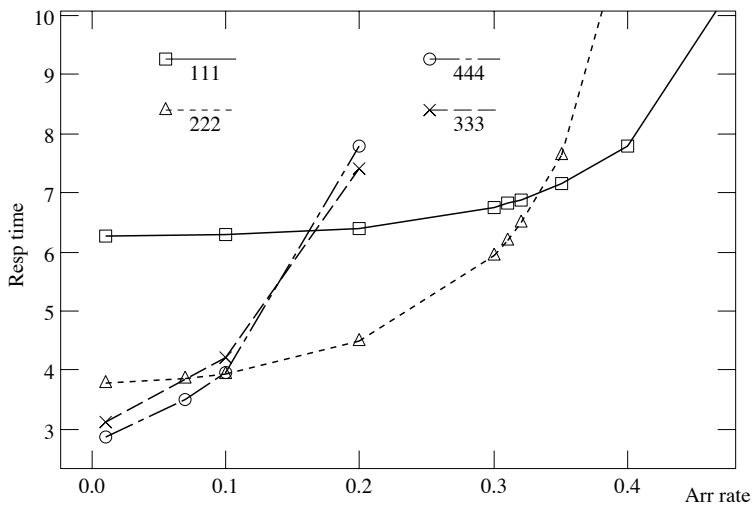


Figure 15: Medium Requests, including policy "333"

References

[1] F. Baskett, K. M. Chandy, R. R. Muntz, and F. Palacios. Open, closed and mixed networks of queues with different classes of customers. *Journal of the ACM*, 22(2):248–260, April 1975.

[2] Steven Berson, Shahram Ghandeharizadeh, Richard R. Muntz, and Xiangyu Ju. Staggered Striping in Multimedia Information Systems. *Submitted to ACM SIGMOD*, 1994.

[3] Edited by Sanjay Ranade. *Jukebox and Robotic Libraries for Computer Mass Storage*. Meckler, 1992.

[4] Personal communication with NSL project at Lawrence Livermore National Laboratory.

[5] R. A. Coyne, H. Hulen, S. Coleman, and R. Watson. The Emerging New Storage Management Paradigm. In *Proc. of the 12th IEEE Symposium on Mass Storage Systems*, Monterey, California, April 1993.

[6] R. Cypher, A. Ho, and P. Messina. Architectural Requirements of Parallel Scientific Applications with Explicit Communications. Technical Report RJ 9079 (80892), IBM, Almaden Research Center, 1992.

[7] Product Description. Ampex DST800 automated cartridge library.

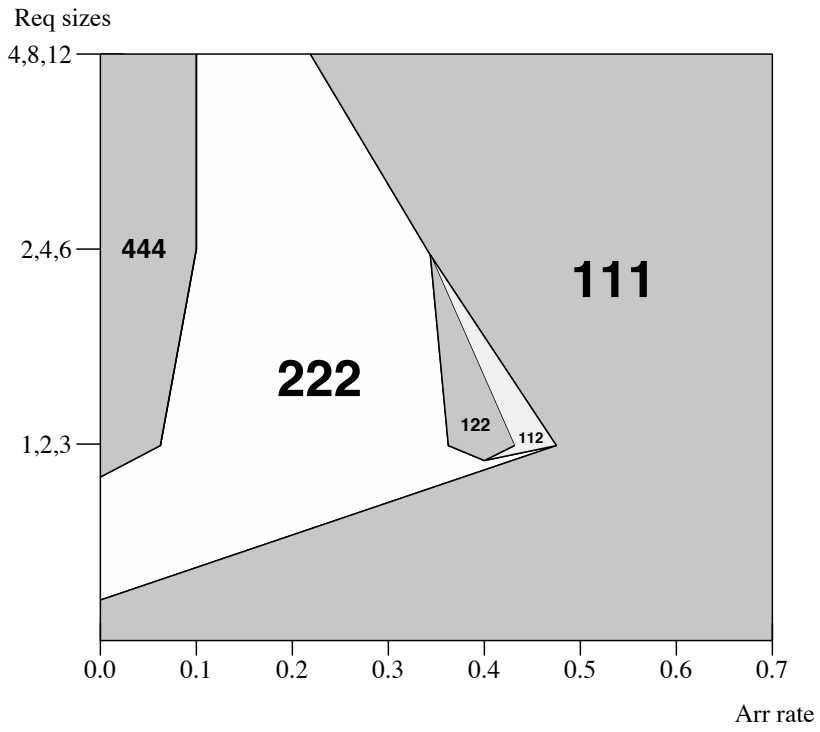


Figure 14: Striping Curves

- prefetching data in order to reduce latency costs
- introducing redundancy into the system (and the model) for reliability purposes
- striping between robotic libraries, rather than within one
- increasing number of drives and/or robot arms and considering the resulting effect on the system
- using *dynamic* striping policies

Acknowledgements

The authors are grateful to Mark Gary for all the valuable discussions.

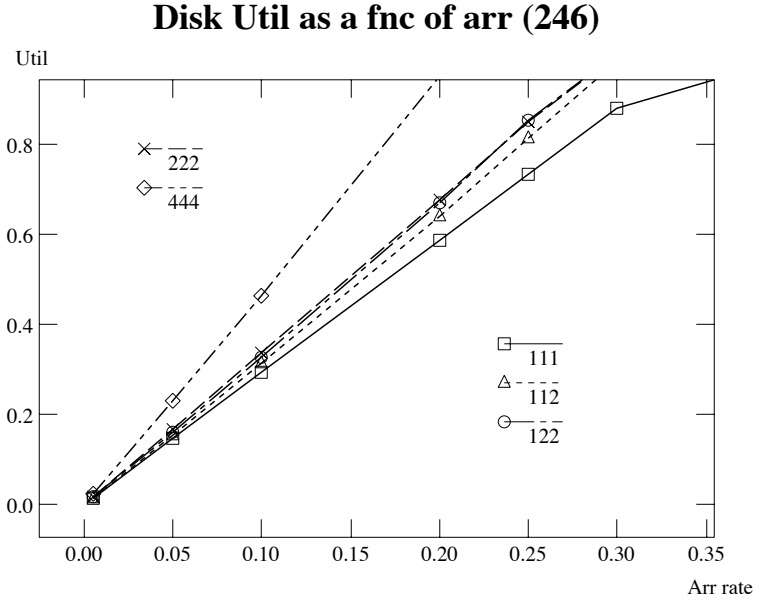


Figure 13: Large Requests Utilization

Other techniques could be used to improve performance, in conjunction with striping, and are the topic of future research. These include, for instance, more sophisticated scheduling of jobs (other than the FIFO scheme used in this paper). An example of the scheduling problem is as follows; suppose there are two jobs in the queue and the first one requires 2 drives and the second requires 1, but there is only one drive available. In this situation, it might be better to schedule the second job ahead of the first one. This is similar to memory allocation problems where algorithms such as "best fit", "first fit", etc., are used. Note that minimization of the robot arm movement can also be considered in a scheduling algorithm. Another way to improve performance is by removing synchronization constraints. Since the receiver of the data might have to store it in a specific order, this may lead to increases in buffer space requirements (either at the sender or the receiver end). The cost of buffer space must be balanced against the resulting improvement in performance.

Future work also includes investigation of the following topics:

- determining the "proper" size of the interleave unit, e.g., block interleaved vs. bit interleaved; advantages and disadvantages of each would depend on the expected access patterns
- including controllers and the communication network, and their limitations, in the model

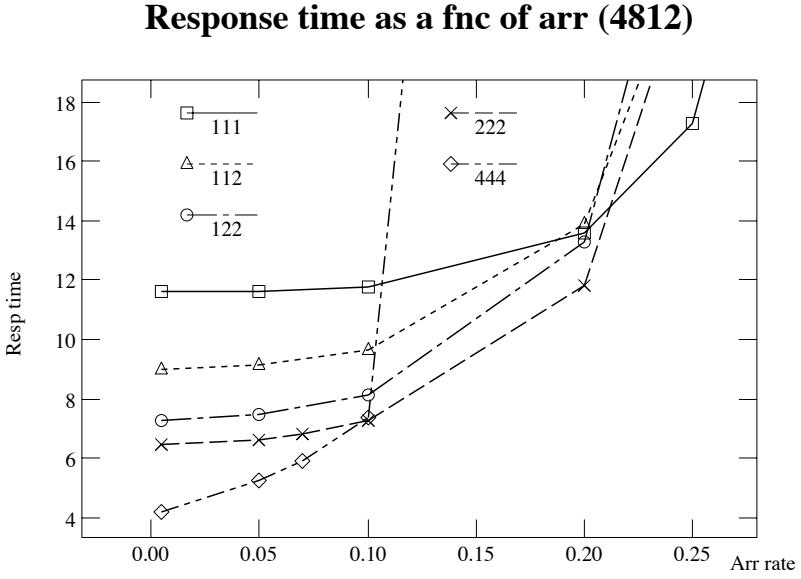


Figure 12: Large Request Sizes

being the best policy, but only for a brief while.

In summary, determining an optimal stripe width for every type of request present in the system is not a simple task and is highly dependent on the system's workload. Our model indicates that striping all classes of requests across the same number of tapes is a good heuristic for minimizing the average system response time (refer to Figure 14), although not always optimal; simulation results illustrate that in most cases equi-striping comes within 10% of an optimal solution. This heuristic significantly reduces the "stripe-width" search space; given a workload for a system with K drives and J classes of customers, we only need to consider $O(K)$, instead of $O(K^J)$, possible stripings. Our simulations also indicate that due to fragmentation problems, certain stripe widths are not worth considering, even in the case of equi-striping.

6 Conclusions and Future Work

We have studied the performance of robotic storage libraries through the use of a simulation model; specifically, we have analyzed the performance of striped libraries as compared to non-striped libraries. We have showed that although striping increases the effective bandwidth of the system, it also introduces overhead into the usage of resources and hence results in increased system utilization. Therefore, striping decisions should depend not only on the bandwidth requirements, but also on the system's workload.

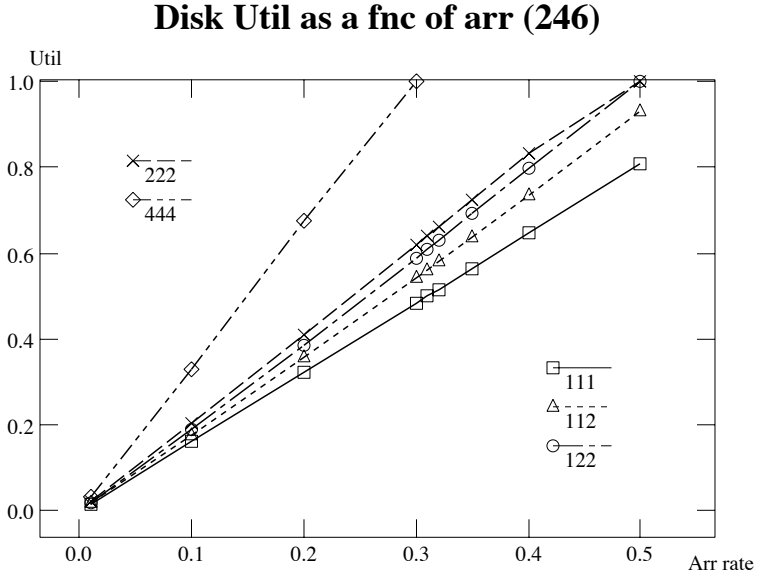


Figure 11: Medium Requests Utilization

time, as compared to the near-by equi-striped regions, is small. In other words, the penalty for "skipping" over such regions should not be high; it is under 10% for most of our simulation runs. This remains true even if the workload is skewed towards smaller requests; the regions might become somewhat larger, but the corresponding improvement in response time remains small.

Also note that region "333" is missing from Figure 14; this omission is intentional and is due to fragmentation problems. Consider Figure 15 which depicts a system with 4 drives and $s_1 = 2, s_2 = 4$, and $s_3 = 6$. There is no region in this figure where policy "333" is best for the following reasons. Striping files across three tapes in a system with four drives insures that there is always a drive idle. When drive utilization is low, improving the transmission time is more important and this is not a serious problem. Hence, "333" performs better than "222", but since striping across four tapes reduces transmission time even further, "444" ends up being the best policy at low utilization. One advantage that "333" has over "444", in addition to having a smaller load-unload overhead, is that it always has one drive idling, which can be loaded with a tape for the next request (as long as there is work in the system); thus, at heavy loads the loading and unloading penalty of "333" is effectively as low as that of "222". As the load increases, "333" begins to perform better than "444". However, by that time, the contention for drives is so high that "222" takes over as the best policy. Given a system with shorter transmission times, such as in Figure 8, there exists a region where the smaller load-unload overhead of "333" outweighs the shorter transmission time of "444" at a low enough utilization; this results in "333"

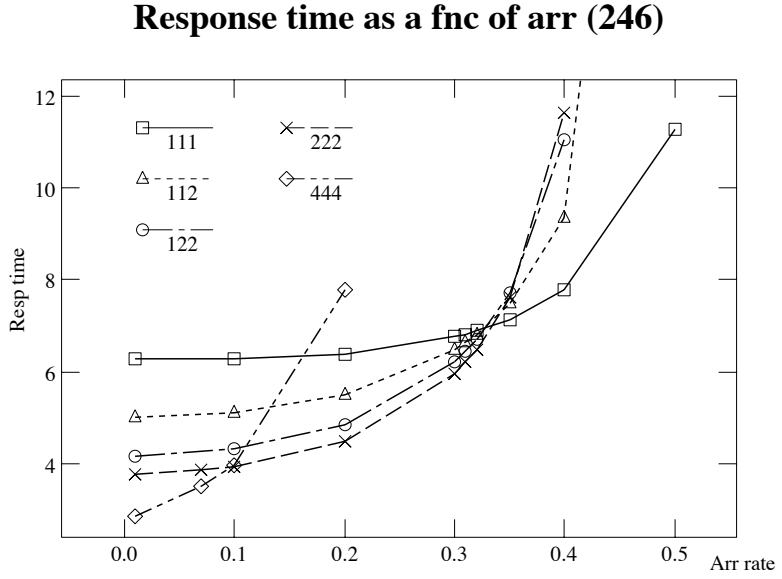


Figure 10: Medium Request Sizes

do exist, e.g., in Figure 8, there is not a great performance penalty associated with disregarding them; we elaborate on this shortly.

Since the workload of the system is a combination of the arrival rate and the amount of work corresponding to each arrival, namely the request sizes, it is more informative to look at a graph that represent how the "best" policy changes as a function of both the arrival rate and the request sizes. Figure 14 illustrates such a graph. Each polygon in this figure represents a region of the workload where a particular striping of requests performs better than any other striping, given the average system response time as the performance metric. When requests are very small, striping is not a good idea under any arrival rate; the overhead of additional loads and unloads outweighs the reduction in transmission time. As the request sizes increase, striping becomes attractive, as long as the arrival rate stays reasonably small. As the arrival rate grows, the delay due to additional contention for resources in the striped system outweighs the benefits of reduced transmission time; it is then advantageous to decrease the stripe width until the load on the system becomes high enough to make striping impractical altogether.

We could make striping decisions based on the graph of Figure 14 by adjusting the stripe width of requests according to the changes in the system's workload. Note, however, that the regions where some classes are striped and others aren't, e.g., "122", are either non-existent or small as compared to the "equi-striped" regions, e.g., "222". Moreover, where these regions do exist, the corresponding improvement in response

the arrival rate[2]. Note that the utilization curve for the non-striped policy is always

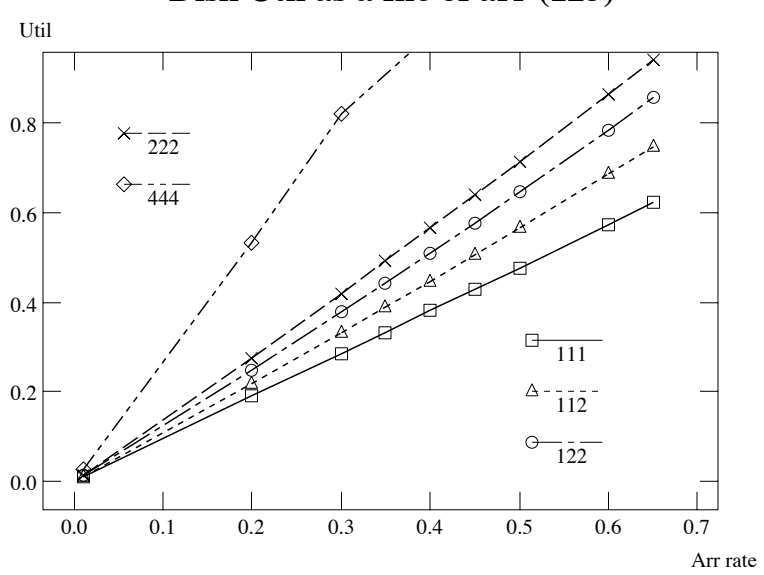


Figure 9: Small Requests Utilization

lower than all other utilization curves; likewise, the utilization curve corresponding to policy "222" is always lower than the one corresponding to policy "444", etc. Given a "heavy enough" load a striped system will always lose to a non-striped one, or, in other words, given a heavy enough load, a "less-striped" system will do better than a "more-striped" system. For instance, consider the response time curves in Figure 8 and the corresponding utilization curves in Figure 9. As the utilization of system "222" gets close to 60%, the load becomes heavy enough to switch to policy "122"; similarly as the utilization of system "112" gets close to 60%, the load becomes heavy enough to switch to policy "111", etc. Thus, what the best policy is changes with changes in the workload.

We are interested to see if the above observations are robust with respect to varying request sizes. Figure 10 presents similar response time curves but for "medium" request sizes, where $s_1 = 2, s_2 = 4$, and $s_3 = 6$; Figure 11 illustrates the corresponding drive utilization curves. Similarly Figure 12 presents response time curves for "large" request sizes, where $s_1 = 4, s_2 = 8$, and $s_3 = 12$, and Figure 13 depicts the corresponding utilization curves. Although the actual numbers change, qualitatively, similar observations made for the "small" request size system hold as the request sizes are increased. Note that in Figures 10 and 12 there are no regions where either policy "112" or policy "122" is best. However, since these regions are small even when they

[2]The robot arm utilization is not shown because it is small in these examples.

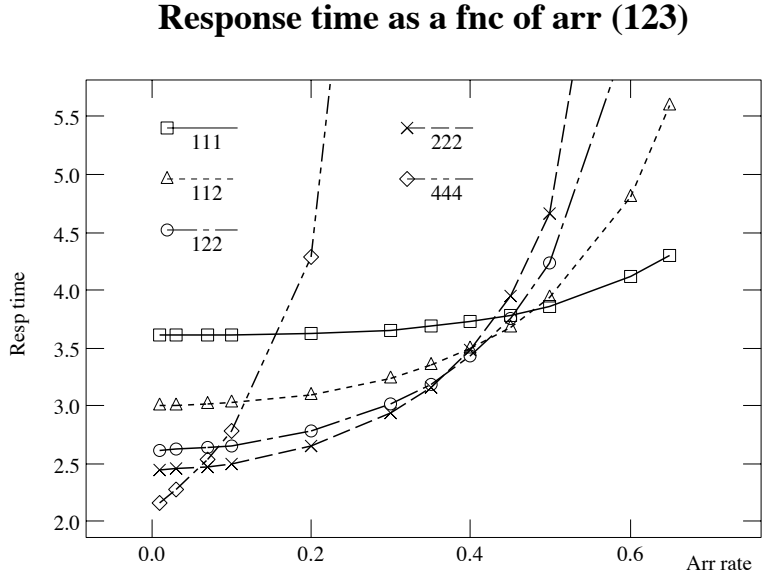


Figure 8: Small Request Sizes

all four drives and holds on to them for the duration of transmission, loads, searches, etc.; at higher arrival rates that time is sufficient to build up a large input queue. The result is that the delay experienced in the input queue eventually outweighs the benefits of the increased bandwidth. As is clear from Figure 8, it does not take a very high arrival rate for the non-striped policy ("111") to outperform the fully striped policy ("444"). As the arrival rate increases, it becomes impractical to stripe the smaller requests; beyond some load it is not practical to stripe at all. Consider Figure 8 again; starting at the arrival rate of 0.1, "222" exhibits the lowest response time. When the arrival rate is sufficiently high (around 0.38 in this example), it is beneficial to stop striping requests of class 1. As the arrival rate increases further it is best to stop striping requests of class 2, until finally, the arrival rate is so high (around 0.48) that it is best not to stripe at all. Note that it is not necessarily best to stop striping the smaller requests first. On one hand, it is reasonable to reduce the bandwidth allotted to the smaller requests since a lesser fraction of their delay is attributed to transmission time, which would be increased due to the reduction in bandwidth. On the other hand, reducing the stripe width of larger requests has a greater effect on (reducing) contention for the drives. The decisions depends on the fraction of each type of request present in the system; for instance, if there are very few large requests in the system, then minimizing the effect on transmission time should be a priority.

Striping *always* introduces additional resource utilization into the system. This can easily be seen in Figure 9, which illustrates drive utilization as a function of

5 Simulation Results

In this section we simulate the queueing model presented in Section 4. All simulations are run until a 95% confidence level and a confidence interval of $\pm 5\%$ surrounding the mean are reached.

Recall the system parameters introduced in Sections 3 and 4. Below we give specific values of these parameter, which correspond to the Ampex storage library; for the remainder of this section we use the Ampex library as an example system.

Parameter	Distribution	Mean
λ	exponential	varied
J	constant	3
$p_j (1 \leq j \leq J)$	constant	$\frac{1}{3}$
μ	exponential	6.00 ($\frac{1}{min}$)
ls	exponential	1.82 ($\frac{1}{min}$)
tr	2 exponential phases of service	
	(transfer)	0.75 ($\frac{GB}{min}$)
	(rewind + eject)	4.29 ($\frac{1}{min}$)
K	constant	4

We assume that for each request, regardless of its class, on the average half of the tape must be searched (and rewound). The amount of data to be transferred and the number of drives necessary for the transfer are the only distinguishing features of each class. Therefore, each class j can be identified by: a) the request size, s_j and b) the request stripe width, w_j. In the following presentation, we consider our model under various workloads, where the load is controlled by first varying the arrival rate and then varying the request sizes. For each load i, determined by the arrival rate λ^i and a set of request sizes, $s_j^i, 1 \leq j \leq J$, we consider all possible striping combinations by varying each w_j^i between 1 and K. However, for ease of exposition we present only a subset of the possible striping combinations; these are sufficient for the illustration of the main results.

Figure 8 presents response time curves for "small" request sizes, where $s_1 = 1, s_2 = 2$, and $s_3 = 3$ (all request sizes are in gigabytes). Each curve represents a different striping policy and is labeled accordingly by $w_1 w_2 w_3$. In this example the requests are large enough such that (in striped systems) there is a sufficient improvement in transmission time, due to the increased bandwidth, as compared to the time spent on additional loads and unloads. When the arrival rate is small, the policy that stripes each request among all four drives has the lowest response time. As the arrival rate increases, so does contention for resources. In the "444" policy, each request acquires

(in the "drive pool" of Figure 7), they are allocated to the request at the front of the queue. Once a drive is allocated, the robot arm loads an appropriate tape on it; the drive then performs the device load and searches to the start of the file. At this point the drive is ready to transmit, but it remains idle until its "buddies", i.e., the other drives participating in servicing the same request, are loaded and ready to transmit as well. As soon as a drive is finished with its transmission, it can rewind and eject the tape. At this point the service of the request is complete, but the tapes must still be unloaded by the robot. In this model we ignore any locality of I/O requests. Since this library is part of a mass storage system, we assume that requests arrive from various parts of the network, and hence there is very little chance of correlation between them; in addition, the number of tapes is very large, as compared to the number of drives. Taking all this into consideration, it is reasonable to assume that the probability of a proper tape already being loaded on a drive is very close to zero, and therefore it is desirable to perform rewind, eject, and unload as soon as the transmission is finished. It is also intuitive that the robot arm should finish all the loading requests before starting on the unloading requests; not servicing a loading request first might, for instance, delay a request that has already acquired all but one drive necessary to start transmission. Hence in our model we represent the robot arm by a simple priority queue, with loads having higher priority over unloads and FIFO scheduling within the same priority class.

Note that in Figure 1, drives do not have dedicated controllers. Therefore, the transmission time of a striped file also depends on the number of independent data paths available for the transfer. The speed of the controllers, relative to the transfer rate of the drives, should be taken into consideration when making striping decisions; for instance, there is little advantage to striping across all the drives in the system, if the controllers are not fast enough to transmit the data from all drives simultaneously. However, we feel that with so few drives in the system, it is reasonable to expect that either each drive has a dedicated controller or that the controllers are sufficiently fast to allow all drives to transmit data simultaneously. We assume that the controllers are not the bottleneck and therefore do not represent them in our model.

The queueing model of Figure 7 does not have product form [1], which suggests that exact analytic solutions are not tractable. Therefore, either approximate solution techniques must be developed, or the queueing network must be simulated. In the following section we present simulation results; analytic solutions are a topic for future work.

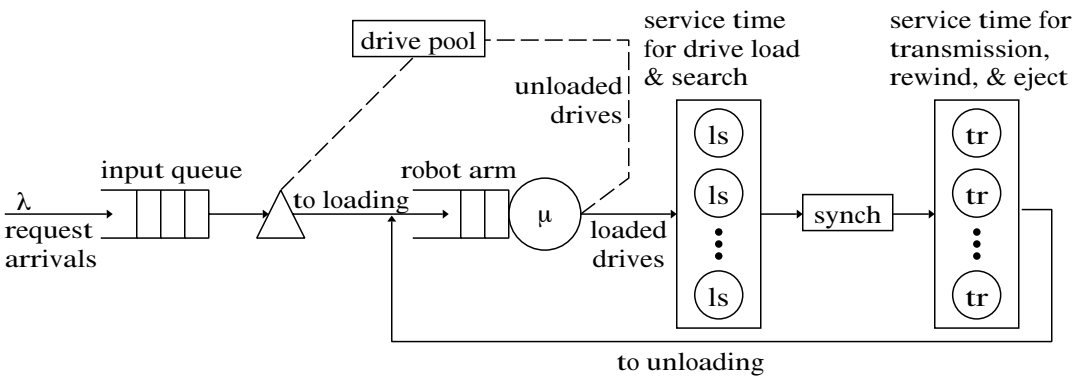


Figure 7: Queueing Model

- $\frac{1}{tr}$ = mean service time of a drive, which includes the drive's
  1. transfer time
  2. rewind time
  3. eject time
- K = number of drives

The different classes in this model represent requests of different sizes, and are distinguished by: a) the request's size and b) the request's stripe width. It is important to look at a multiple class system since striping decisions, as already mentioned in Section 3, depend not only on a request's size, but also on the workload which is composed of the rest of the requests in the system. Since we expect the mass storage system to be used by a mixture of a variety of applications (such as Global Climate Simulation or 3-D Perspective Rendering [6]) with vastly different file access patters, it is difficult to characterize the arrival process for this model. We assume a simple application independent arrival process, namely a Poisson process with a mean arrival rate of λ and produce different workloads by varying both the mean arrival rate and the class mixture of the system.

Each request arriving to the system requires between 1 and K drives[1] and with probability p_j belongs to class j, where the class determines the amount of data to be transferred for this request. The arriving request joins the input queue of requests waiting for drives. As the drives are unloaded one at a time and become available

[1]We do not consider striping wider than K, since it is not possible to read more than K tapes at once.

Moreover, the striped system saturates at a lower arrival rate than a non-striped system. For a system to be stable, it must have the characteristic that $\lambda_i < k_i\mu_i$. Given that $\lambda_s = \lambda_{ns}$ and Eq. (5), the non-striped system can sustain a higher arrival rate than a striped one. Consider again the Ampex system transferring 1 GB files using four drives but this time under heavy load. Given $K = S = 4$ and $N >> K$ customers already in each system, we can compute, using Eqs. (3) and (4),

$$R^s(N) = \frac{N+1}{\frac{K}{S}\mu_s} = 147(N+1)$$

and

$$R^{ns}(N) = \frac{N+1}{K\mu_{ns}} = \frac{147(N+1)}{4}.$$

Furthermore, the non-striped system, in this example, can sustain an arrival rate of up to $k_{ns}\mu_{ns} = \frac{4}{147}$ (1GB) requests per second, whereas the striped system becomes unstable when the arrival rate reaches $k_s\mu_s = \frac{1}{147}$ (1GB) requests per second. In general, the ratio of $k_{ns}\mu_{ns}$ to $k_s\mu_s$ is a function of the request size and does not have to be as high as in the above example; for instance, given 4GB requests, $\frac{k_{ns}\mu_{ns}}{k_s\mu_s} \approx 2.14$.

We hope that by now the reader is convinced that, in general, striping decisions are not trivial. Now that we have acquired the proper intuition about striping, we are ready to construct a more accurate model of the storage library, which is the topic of the following sections.

4 Model

A model of the robotic tape library (refer to Figure 1) is illustrated in Figure 7. This model represents the different phases of a striped I/O request service time and has the following parameters:

- λ = mean arrival rate of requests
- J = number of job classes
- p_j = probability of arrival being from class j, $1 \leq j \leq J$
- $\frac{1}{\mu}$ = mean service time of the robot arm (either loading or unloading)
- $\frac{1}{ls}$ = mean service time of a drive, which includes the drive's:
  1. load time
  2. search time

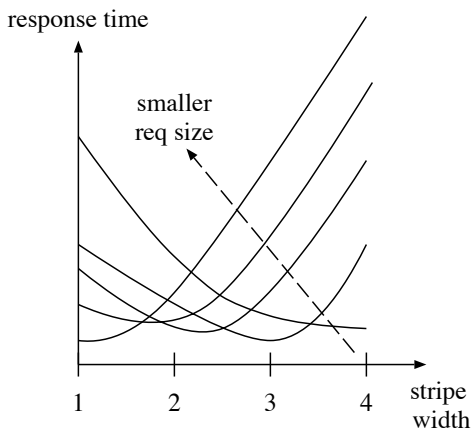


Figure 6: Response Time Under the No-load Condition

Given that there is contention for the drives and that $N^i >> k_i$, i.e., the heavy load case,

$$R^i(N^i) = \frac{N^i - (k_i - 1)}{k_i \mu_i} + \frac{1}{\mu_i}$$

and therefore

$$R^{ns}(N^{ns}) = \frac{N^{ns} + 1}{k_{ns}\mu_{ns}} = \frac{(N^{ns} + 1) * (C + L + T)}{K} \quad (3)$$

$$R^{s}(N^{s}) = \frac{N^{s} + 1}{k_{s}\mu_{s}} = \frac{(N^{s} + 1) * (C + LS + T/S)}{K/S} \quad . \quad (4)$$

Suppose now that $N^s = N^{ns} = N$, i.e., that a tagged customer finds N jobs already waiting, whether it arrives to a striped or a non-striped system. Then, given that $S > 1$, we can conclude from Eqs. (1) and (2) that

$$k_{ns}\mu_{ns} > k_s\mu_s \quad (5)$$

and therefore, using Eqs. (3) and (4)

$$\frac{(N + 1) * (CS + LS^2 + T)}{K} > \frac{(N + 1) * (C + L + T)}{K} \quad .$$

Thus, the response time experienced by this tagged customer is smaller in the non-striped system. In other words, the striped system takes longer (than the non-striped one) to get rid of a backlog of N customers. Given that the two systems have the same arrival rates, intuitively, this indicates that under heavy load, the average number of customers is greater in the striped system (than in the non-striped system), since it is less efficient at getting rid of backlogs. According to Little's Result [12], this implies that the average response time is also greater in the striped system.

Each arrival to a non-striped system requires a single server, but an arrival to a striped system requires S servers. To simplify matters, let us assume that: a) K is a multiple of S and b) the time to service a request, striped across j drives, is the same on all j drives. Then we can model the non-striped system as a multiple server with

$$k_{ns} = K \qquad \text{and} \qquad \mu_{ns} = \frac{1}{C + L + T} \tag{1}$$

and the striped system as a multiple server but with K/S *super* servers, i.e.,

$$k_s = \frac{K}{S} \qquad \text{and} \qquad \mu_s = \frac{1}{C + LS + T/S} \quad . \tag{2}$$

We consider two extreme situations: a) the *no load* case, i.e., the case where requests are sent so seldom that there is no contention for drives in the system and b) the *heavy load* case, i.e., the case where the system is so busy that there is always someone waiting to receive service. In the case of no load in the system, $W^i(N^i) = 0$ and $N^i < k_i$. Then

$$R^{ns}(N^{ns}) = \frac{1}{\mu_{ns}}$$

$$R^{s}(N^{s}) = \frac{1}{\mu_{s}}$$

and a striped system is an improvement over a non-striped one when

$$\mu_{ns} < \mu_s$$

or

$$T + L + C > \frac{T}{S} + LS + C$$

If our measure of performance is response time (in the no-load case), then given a file size, we can determine an optimal stripe width by minimizing $LS + T/S$, such that $1 \leq S \leq K$ and S is an integer. This is (qualitatively) illustrated in Figure 6, where each curve corresponds to a fixed size request. For example, consider the Ampex system servicing a 1 GB request, first without striping and then with a stripe width of 4. Then,

$$\mu_{ns} = \frac{1}{C + L + T} = \frac{1}{47 + 20 + 80} = \frac{1}{147}(\frac{1}{sec})$$

$$\mu_{s} = \frac{1}{C + 4L + T/4} = \frac{1}{47 + 80 + 20} = \frac{1}{147}(\frac{1}{sec})$$

which suggests that any request greater than 1 GB is worth striping (across four drives) in the Ampex system, under the no-load assumption.

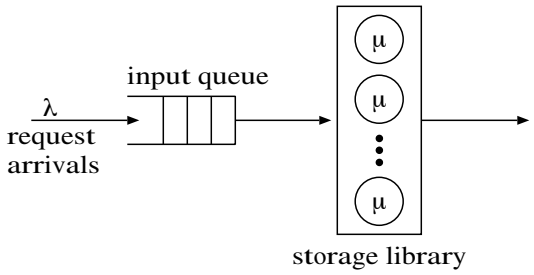


Figure 5: Simple Model

- μ_i = mean service rate of each server in system i, $i = s, ns$
- N^i = number of customers in system i when a tagged customer arrives, $i = s, ns$
- $R^i(N^i)$ = expected response time of a tagged customer arriving to system i with N^i customers already there, $i = s, ns$
- $W^i(N^i)$ = expected waiting time of a tagged customer arriving to system i with N^i customers already there, $i = s, ns$
- k_i = number of servers in system i, $i = s, ns$
- K = number of drives in the storage library
- T = transfer time on a single drive
- C = time to load drive, search, rewind, and eject
- L = robot load or unload time
- S = stripe width

In this model, all operations performed by the storage library (search, rewind, load, transfer, etc.) are represented by a multiple server, where the response time of a tagged customer can be represented by the following simple equation (assuming that the service time is exponentially distributed, with mean μ_i) [11]:

$$R^i(N^i) = W^i(N^i) + \frac{1}{\mu_i} \qquad i = s, ns$$

where

$$W^i(N^i) = \begin{cases} \frac{N^i - (k_i - 1)}{k_i \mu_i} & N^i \geq k_i \\ 0 & N^i < k_i \end{cases}$$

their transfer time accounts for only a small portion of the total response time, so it is likely that the additional tape switching time will outweigh the improvement in transfer time. Also, intuitively, large files could benefit from striping, since the data transfer time accounts for a large fraction of the response time.

Note, however, that we can not make striping decisions strictly based on the file's size. Since our goal is to improve the overall response time of the system as opposed to improving the response time of any individual job, the effects of striping through increased contention must be taken into consideration. For instance, another disadvantage of striping is that it increases contention for drives, as well as other resources, which results in an "artificial" increase in resource utilization. The larger a file is or the larger the stripe width, the more contention there is for resources; this is (qualitatively) illustrated in Figure 4. Since the system response time increases

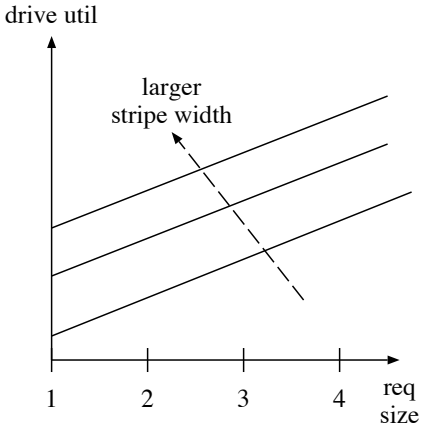


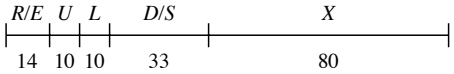
Figure 4: Drive Utilization

with the contention, it might not be advantageous to stripe even very large files under heavy loads. At light loads, the effect of an increase in drive utilization is not significant, but at heavy loads, it becomes the dominant factor. Note that search and rewind could account for a significant fraction of the drive utilization; reducing these times would reduce contention for drives and increase the range of loads for which striping is advantageous. Once again, we should point out that *load* is an important factor in striping decisions.

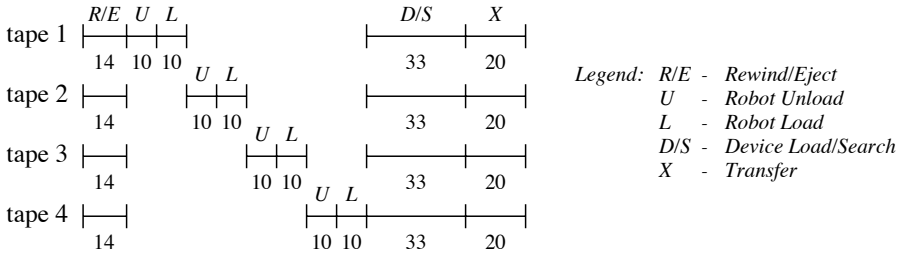
To illustrate these points and to gain some insight into the problem, let us consider a very simple queueing model of the system, illustrated in Figure 5, with the following parameters:

- λ_i = mean arrival rate to system i, $i = s, ns$

for a completion of one tape switch before it can start on another one. Therefore, such functions as rewind, search, etc., can be overlapped, but, due to a single (in our example) robot arm, all the robot loads and unloads have to be performed sequentially. Hence, the latency penalty for striping across W tapes amounts to additional $W-1$ loads and unloads. This is illustrated in Figure 2(b) for an Exabyte system retrieving a 1 GB file striped across 4 tapes, where loading and unloading the 3 additional tapes adds a little over 2 minutes to the total response time. Hence, by striping the 1 GB file across 4 tapes, on an Exabyte system, we win about 26 minutes during the transfer, and lose about 2 minutes to tape switching. In this example, the net effect of striping is an improvement, of 24 minutes, in the total response time; however, given a smaller file, or a system with faster transfer rates, this might not be the case. For instance, the Ampex system can transmit the same 1 GB file in less than 2 minutes, but its tape switching time is also much lower (a little over a minute). Striping a 1 GB file across 4 tapes in the Ampex system does not improve response time because the reduction in transmission time is cancelled by the latency penalty due to additional loads and unloads. The non-striped and striped transfers for the Ampex system are illustrated in Figures 3(c) and 3(d), respectively. Note that improving the speed of the robot



(a) Transmission (in sec) of a 1 GB file (no striping)

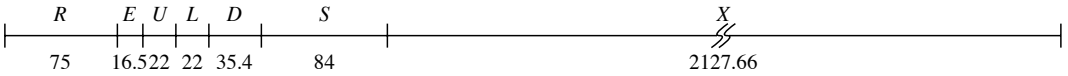


(b) Transmission (in sec) of a 1 GB file (striped on 4 tapes)

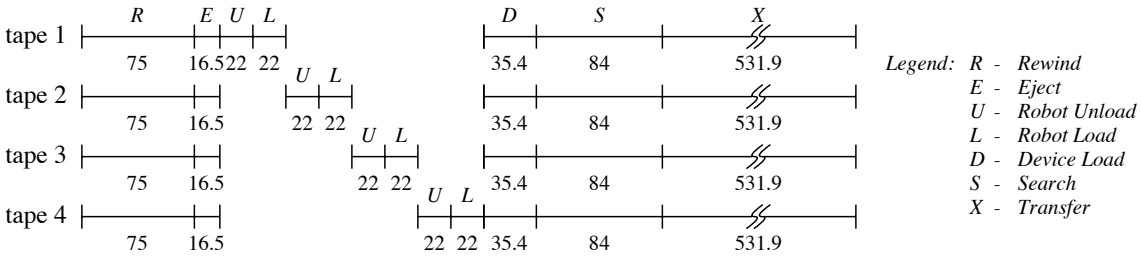
Figure 3: Timing Diagram for Ampex DST800

arm movement or having more than one arm could make striping advantageous over a wider range of request sizes.

To summarize, an advantage of striping is an improved transfer time, where (basically) the wider we stripe a file, the shorter is its transfer time. One disadvantage of striping (others will be exposed shortly) is an increase in number of tape switches. Therefore, we have to balance the amount of time we gain by lowering the transfer time against the amount of time we lose by having to load multiple tapes for a single request. Intuitively, small files should not be good candidates for striping, since



(a) Transmission (in sec) of a 1 GB file (no striping)



(b) Transmission (in sec) of a 1 GB file (striped on 4 tapes)

Figure 2: Timing Diagram for Exabyte 120

is that it improves the effective bandwidth of the system and therefore reduces the file transfer time.

Given that files are striped, the transmission procedure changes as follows:

1. identify the n tapes containing requested file
2. for each tape i:
   - wait for a drive to become available
   - if the proper tape is not already loaded in this drive (perform a tape switch):
     - rewind and eject the tape in the drive
     - return the old tape to its proper location in the library (robot unload)
     - fetch the new tape, i, from the library and load it into the drive (robot load plus device load)
   - search to the beginning of the portion of the file stored on tape i
3. transmit the data from all n drives

One negative side effect of striping is an increase in the number of tape switches performed per request, which further contributes to an already high latency due to tape switching. Note, that the robot is free to perform other duties, as soon as it's done fetching and loading a tape into a drive, i.e., it does not have to wait

Search Time ($\frac{1}{2}$ tape) (sec)	84	20-30
Transfer Rate (MB/sec)	0.47	14.5
Robot Load/Unload	22 sec	$<$ 10 sec

The basic operation of servicing an I/O request can be described as follows:

1. identify the tape containing requested file
2. wait for a drive to become available
3. if the proper tape is already loaded in that drive, go to step 5
4. otherwise, perform a *tape switch*:
   - rewind and eject the tape in the drive
   - return the old tape to its place in the library (robot unload)
   - fetch the new tape from the library and load it into the free drive (robot load plus drive load)
5. search to the beginning of the file
6. transmit the data

Although robotic storage systems vary in size, speed, and cost, one characteristic remains certain; the number of robot arms and the number of drives is small as compared to the number of tapes. Due to the small number of drives, it is highly probable that a new request will not find the proper tape already loaded and will generate a tape switch. As pointed out in [9], the Exabyte120 system takes more than 4 minutes to perform a tape switch, which is very slow and therefore undesirable. The tape switches are expected to occur frequently and are largely responsible for the high latency experienced by users of robotic storage libraries. Depending on the file size, transmission time could also be a significant fraction of the total response time. For instance, Figure 2(a) illustrates a tape switch and a transmission of a 1 GB file on an Exabyte system. The entire procedure requires over 39 minutes, approximately 35 of which is spent on transmission. One way to improve the bandwidth of a system is to involve several drives in a single transfer. In other words, if a file is *striped* across several tapes, then fractions of it could be transmitted simultaneously by several drives, thus increasing the effective bandwidth of the transfer. For instance, if the same 1 GB file is striped evently across 4 tapes, i.e., 250 MB per tape, then the transfer time is reduced to approximately 9 minutes. Hence, the advantage of striping

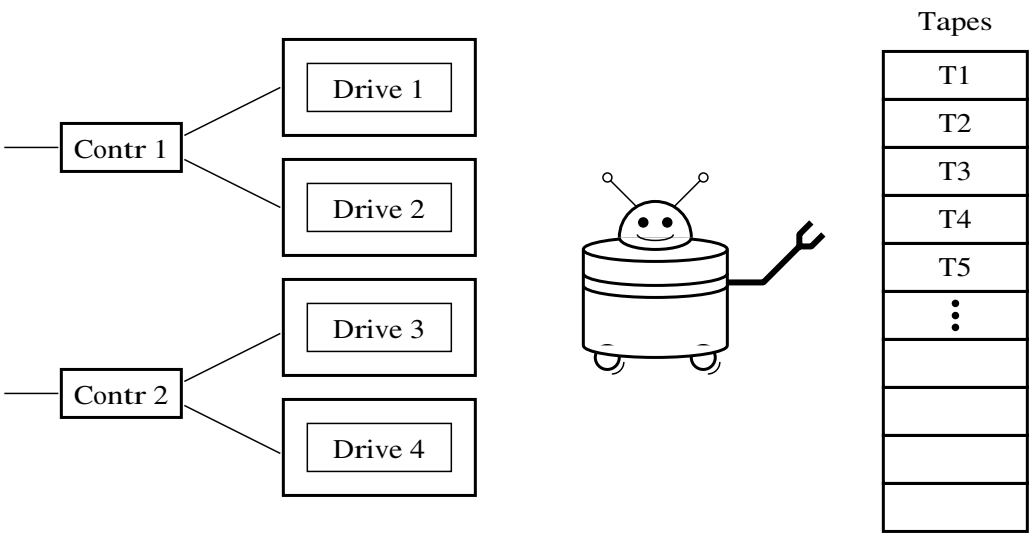


Figure 1: Robotic Storage System

(As will be explained later in this paper, the number of controllers and the number of tapes in the library is not significant in our model.)

The actual size of the library, e.g., the amount of data it can hold, can vary greatly. For instance, the following are typical numbers for two commercially available storage libraries, one developed by Exabyte, using 8mm tapes and the other by Ampex, using D2 tapes.

	Exabyte	Ampex
Number of Robot Arms	1	1
Number of Drives	4	4
Number of Tapes	116	256
Tape Capacity	5GB	25GB

The speeds of the various parts of the storage system, e.g., drives' transmission rate, robot loading time, etc., also vary greatly from one system to another. For instance, the following are typical parameters for the Exabyte120, as reported in [9], and the Ampex DST800 [4].

	Exabyte	Ampex
Mean Drive Load (sec)	35.4	5
Mean Drive Eject (sec)	16.5	4
Rewind Time ($\frac{1}{2}$ tape) (sec)	75	12-13

simulation results and Section 6 describes future work and presents concluding remarks.

2 Related Work

Striping is a well known technique for improving the effective bandwidth of an I/O system. It has successfully been used in general purpose disk array systems, such as the RAID technology [13], to increase the bandwidth of inexpensive disks, as well as in application specific systems, such as multimedia storage servers [2], to meet the high bandwidth demands of display stations. Therefore, it is natural to attempt to extend this idea to a tape storage system.

In [8], the authors examine the performance of striped magnetic tape libraries. They consider a closed simulation model with a workload consisting of fairly small (under 1 GB) reads (75%) and writes (25%) of a constant size, within an individual simulation run. The striping configuration is analogous to RAID with block interleaving. In [9], the authors continue their work by considering bit interleaved systems and larger (than 1 GB) requests. Since in this case the striping configuration is analogous to RAID3 [13], the reads are equivalent to $K-1$-wide striped requests and the writes are equivalent to K-wide striped requests, where K is the number of drives (see Section 5 for a discussion on disadvantages of using $K-1$-wide requests in a system with K drives).

In this paper we consider an *open* model of a storage library with *multiple* sizes of requests, within an individual simulation run, and various stripe widths (in addition to K and $K-1$). This is more appropriate for a mass storage system which is expected to be used by a mixture of a variety of applications executing I/O requests from various parts of the network. Given this multiple class open model, we illustrate that the workload, which is a function of the rate at which requests arrive as well as their size, is a major factor in striping decisions.

3 System

A robotic storage library is a complex system that, for the purposes of our discussion, can be described in terms of the following main components (refer to Figure 1):

- R = number of robot arms
- K = number of drives (or transports)

tape switches, which involve: a) unloading the tape currently occupying the device, b) placing it back in the library, c) fetching the new tape, and d) loading the new tape in the device. As pointed out in Section 3, these can be quite time consuming.

In this paper we analyze the performance of robotic storage libraries and study techniques for improving it. Specifically, we consider *striping*, a technique successfully used in disk arrays [13], multimedia servers [2], etc., as an approach to decreasing the response time of a robotic storage library. Striping usually refers to partitioning a file among several disks (or tapes, in our case) for the purpose of increasing the system's bandwidth by transmitting the file from several devices at once and hence reducing its transfer time. One downside of striping is that each file transmission results in loading of several tapes, instead of one, which increases the already high latency of tertiary storage. Another downside of striping is the "artificial" increase in the effective workload on the devices. Since several devices are used for each transfer, instead of one, there is a greater degree of contention for each device, which generally results in increased response time of the system. Hence the decrease in transfer time must be balanced against the increased latency due to additional tape loads and the increased waiting time due to higher contention for devices.

The question we would like to answer is "how many tapes should participate in each file transfer", i.e., what should be the file's stripe *width*. This is not a simple question since the "optimal" stripe width depends on many parameters, including system load, media and device types used, system configuration, etc; the load, which is a combination of both the arrival rate of requests to the system and the size of these requests, is a particularly important parameter to consider. For instance, *intuitively*, we expect a large file to be a good candidate for striping, since it would have a long transfer time. However, if this file is so large that it occupies devices long enough to create a high degree of contention in the system, then striping such a file could result in very long delays for other requests. It may be preferable that such a large request uses only a single drive and allows many other smaller requests to get through the system quicker by using the remaining drives. Even though this approach increases the response time of the very large request, it could accomplish the goal of decreasing the average system response time. Hence, the matter of striping is not a trivial one and further analysis is required. In this paper, we develop a simulation model of a large robotic storage library; we use this model to compare performance of systems under different striping policies. Our long term goal is to use the model in making design decisions in the development of robotic storage libraries for the National Storage Laboratory (NSL) project.

The remainder of the paper is organized as follows. Section 2 briefly summarizes the background work in this area. Section 3 describes the system under consideration, and Section 4 presents the corresponding queueing model. In Section 5 we discuss

1 Introduction

Until recently, improvements in computational speed have been the main focus of research and development in high performance computing. These efforts resulted in GFLOP machines, but have also increased the gap between processor and I/O performance; although, there has been significant growth in the capacity of secondary and tertiary storage media, the transfer rates of I/O devices have not kept up with the unprecedented growth in processor speeds. Today, scientists spend hours retrieving and storing data required by their supercomputing applications. The technology needed for improving this situation, e.g., high speed networks and I/O channels as well as high capacity storage media, exists, and therefore storage of vast amounts of information and its fast transfer over high speed networks have become major challenges in high performance computing. For example, the National Storage Laboratory (NSL) [10], an industry-led collaborative project currently under development at the Lawrence Livermore National Laboratory, has been organized to investigate technology for storage systems that will be the future repositories for our national information assets.

Since storage requirements are growing rapidly and since networked computing has become the norm in most computer centers, there is a great demand for *mass storage* servers [5], an integrated system consisting of a multitude of network attached storage devices whose function it is to provide cheap and rapid access to vast amounts of data. The economics of storage devices are such that there is always a tradeoff between access speed and device and media costs. Therefore, storage devices are usually organized in a hierarchy, with more frequently accessed data stored on fast but expensive devices, towards the top of the hierarchy. Tapes are a form of cheap but slow media which are usually used for storing either infrequently accessed files or files that are too large to fit on secondary storage; however, when such information is accessed there is still a demand for rapid response time. Fast but expensive devices are available to satisfy these response time requirements (for instance the drives used in the Ampex DST800 systems [7]), but in order to be economically viable, their costs should be amortized over a multitude of tapes. Hence, jukeboxes or robotically controlled tape libraries should be components of a mass storage system [3]; these libraries can vary in size and speed but can usually store terabytes of information.

Such large data repositories should be prepared to handle large file transfers, for the following reasons: 1) many scientific applications request data in large chunks, and 2) large retrievals are a customary approach to amortizing high latency costs, typically associated with tape storage systems. Note that the high latency is a result of the sequential nature of the tape media as well as the lack of dedicated devices in robotic libraries; since tapes share the devices, a certain amount of latency is associated with

Analysis of Striping Techniques in Robotic Storage Libraries

Leana Golubchik*

3436 Boelter Hall, Graduate Student Office
UCLA Computer Science Department
Los Angeles, CA 90024-1596
(310) 206-1803, leana@cs.ucla.edu

Richard R. Muntz
UCLA Computer Science Department

Richard W. Watson
Lawrence Livermore National Laboratory

March 22, 1994

Abstract

In recent years advances in computational speed have been the main focus of research and development in high performance computing. In comparison, the improvement in I/O performance has been modest. Faster processing speeds have created a need for faster I/O as well as for storage and retrieval of vast amounts of data. The technology needed to develop these mass storage systems exists today. Robotic storage libraries are vital components of such systems; however, they normally exhibit high latency and long transmission times. In this paper we analyze the performance of robotic storage libraries and study striping as a technique for improving response time. We show that striping, which improves the effective bandwidth, introduces overhead into the usage of the library's resources, and hence conditions under which it is advantageous are highly dependent on the system's workload.

*This work was partially done while the author was a summer student at the Lawrence Livermore National Laboratory.