# Data Placement for Tertiary Storage

**Jiangtao Li**
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907 U.S.A.
jtli@cs.purdue.edu
Phone: + 1 765 494-6008
Fax: + 1 765 494-0739

**Sunil Prabhakar**
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907 U.S.A.
sunil@cs.purdue.edu
Phone: + 1 765 494-6008
Fax: + 1 765 494-0739

## 1 Abstract

In this paper we address the important problem of data placement in tertiary storage taking object relationships into account. This work is in contrast to earlier schemes that either focus on specific data types or assume that data objects are accessed independently. Five new data placement schemes are developed. The effectiveness of these schemes is shown through simulation. The proposed schemes, in particular the Edge Merge scheme, give superior performance over schemes optimized for independent access.

We also show that our schemes can easily adapt to variations in the access pattern. This also allows the schemes to be employed when no prior information about the access pattern is available. Interestingly, our results show that the probabilities of object access do not have a big impact on performance. On the other hand, changes to the clustering of nodes have a significant effect. This result underscores the importance of the relationships between objects for placement of data. The use of controlled replication for "free" is also developed and shown to be effective in further improving performance. The study also evaluates the impact of a secondary disk layer and prefetching.

## 2 Introduction

The tertiary storage layer in a hierarchical storage system is characterized by very large data volume and very high random access latency. Both attributes are directly related to the use of numerous cheap removable media sharing a small number of expensive drives and robot mechanisms. The high access latency is typically dominated by media switch time (for certain tape systems, however, the seek time may exceed the media switch time). With ever increasing demands for storing very large volumes of data for applications such as telemedicine, online multimedia document systems, and other large multimedia repositories, large amounts of live data are being stored on tertiary storage systems. Random accesses to data stored on tertiary storage can suffer unacceptable delays as media are swapped on drives. The need for swapping media is dictated by the placement of data. Judicious placement of data on tertiary storage media is therefore critical, and can significantly affect the overall performance of the storage system.

The placement of data for specific domains such as multi-dimensional arrays [1], relational databases [15], and satellite images [21] has been addressed earlier. Research on tertiary storage placement in a more general setting has been addressed under the assumption that the data objects are accessed independently [2]. This assumption is rarely valid in practice – data objects typically are related and this is reflected in the access of the data. For example, online manuals contain hyperlinks to related sections and other manuals, a browsing session in a multimedia repository is typically guided by similarity between objects, and various test results of a given patient are likely to be accessed during diagnosis or treatment. In this paper we address the problem of placement of data on tertiary storage in a general setting without the assumption of independent access. Our approach is to exploit the nature of the access to the data to determine an optimal placement. This work is orthogonal to related issues of data migration and scheduling. The problem of placement of data on tertiary storage can be broken down into two sub-problems due to the significant cost of switching media: i) allocation of data to media; and ii) placement of data within the assigned medium. The problem of placing data within media has received some attention and we employ existing solutions to this problem such as [2]. The focus of our study is on the sub-problem of allocating data to media in order to minimize switching.

We propose and evaluate several placement schemes for tertiary storage systems based upon data access patterns. The schemes can be employed even if the access pattern is not known *a priori*, and can dynamically adapt to changes in access patterns. The study considers the impact of the secondary storage buffer and caching policy on the placement, and effective use of prefetching based upon the placement and access pattern. In an earlier study we demonstrated that for the case of multimedia documents replication of objects is an effective technique for reducing switching and improving performance. We study the use of replication of objects on multiple media for the general case in this study. The effectiveness of the proposed schemes is evaluated using a detailed hierarchical storage simulator. Our results show that significant improvements (as much as 80% reduction in average waiting time) can be achieved with our placement schemes. The remainder of the paper discusses the issues involved, our proposed approaches, and sample experimental results. Further details and results will be given in the full version of the paper.

## 3 Related Work

The placement of data for specific domains such as multi-dimensional arrays [1], relational databases [15], and satellite images [21] has been addressed earlier. Research on tertiary storage placement in a more general setting has been addressed under the assumption that the data objects are accessed independently. Placement schemes based upon independent document access probabilities and no replication have been proposed in [2, 18]. Optimal arrangement of cartridges and file-partitioning schemes for carousel-type systems are investigated in [17]. Placement schemes for data on optical disks are developed in [3]. To the best of our knowledge, our work is the first to address the issues of placement of related objects (in a general setting) and replication.

Other researchers have addressed the use of hierarchical storage systems for multimedia data. A cache replacement technique for managing secondary storage buffers when multimedia objects are stored on tertiary storage has been developed by Ghandeharizadeh et al [6]. The use of a pipelining mechanism that avoids the need for complete materialization of an object on disk before initializing playback is presented in [5]. We have developed a prefix-caching scheme with low jitter and startup latency for storing continuous media data [14]. Storing video on hierarchical storage has also been studied in [20, 19]. The study addresses I/O bandwidth issues at the various levels of the storage hierarchy. Scheduling schemes for tertiary storage libraries are discussed in [4, 13, 8, 11] – any of these techniques can be applied in conjunction with our research to further improve performance. In [10] a prefetching algorithm based upon Markov-chain prediction of access is developed. Models of tape systems and tertiary storage system parameters can be found in [7, 9].

## 4 Data Placement Schemes

In this section we first explain the nature of access for related objects. This is followed by a description of the proposed tertiary placement schemes that take into account the relationships between objects. Then we discuss the issues of adaptive placement, impact of secondary storage, replication and prefetching.

### 4.1 Access Pattern for Related Objects

For efficient storage and retrieval of data it is critical to take into account the data access pattern. Data objects can be accessed either directly, or through a link from another object. Independent, or direct access to an object can be captured simply by the probability of access. In addition to direct access to objects, users may access objects based upon links from other objects (e.g HTML pages with links to other pages, or hyperlinks between manual pages). Such access is also very common in a browsing scenario whereby users simply follow links of interest. A user would typically begin by accessing an object and then possibly following some number of interesting links. If none of the links are interesting, the user may directly access some other object.

A *Browsing Graph* (BG) can be used to capture such access patterns. The browsing graph consists of labeled nodes and labeled edges. Each node represents an object and the label of the node gives the probability that the node will be accessed independently of the previous visited node. A directed edge between two nodes represents a link from one object to the other and the edge label gives the probability that the edge would be followed.

The sum of the probability of all edges going out of an object is not necessarily 1.0, since it is possible that none of the edges will be followed. We use the term *birth probability* to represent the probability of independent access to objects and *death probability* to represent the probability that once the node is accessed, none of its edges will be followed. The death probability of a node is simply 1 - (sum of outgoing edge probabilities).

## 4.2 Data Placement Schemes

Tertiary storage suffer from high access latency. The access cost in tertiary storage is dominated by the media exchange operation and head position delay. The goal of data placement is to minimize the expected access cost and reduce latency. In [2] it is shown that a placement whereby the objects are placed sequentially in decreasing order of their access probabilities is optimal. We call this the **Birth Probability Scheme**. This result, however, is based upon the assumption that the objects are accessed independently.

**Static Probability Scheme**: The frequency of an object being accessed is usually different from its birth probability. The object birth probability is the probability of the object being accessed directly, while the static probability is the probability of being accessed directly or indirectly. In other words, static probability represents the frequency of the object being requested. Given the user browsing graph, the static probability of an object can be easily computed by simulation. Our static probability data placement scheme is that the objects are placed sequentially in decreasing order of their static probabilities.

**Edge Merge Scheme**: This scheme explicitly takes into account the links between objects. Once an object is requested, it is very likely that objects with high probability links from this object will be accessed next. If such neighbors are placed on the same medium, a medium exchange can be avoided. The main idea of this scheme is therefore to place strongly related objects on the same medium. Ideally, all related objects are placed on the same medium. However, the medium capacity will not allow this. Therefore related objects may have to be spread across multiple media if the "cluster" of related objects is large. On the other hand, if there are small "clusters" then the problem is to pack as many clusters as possible on a single medium.

The basic idea behind edge merge is the following: Not all linked objects can be placed together; therefore, we give priority to higher probability links. To achieve this, we start merging objects that are linked by high probability edges into a new object. We define the new object's birth probability to be equal to the sum of that of the merged objects. Links into and out of the merged objects connect to the new object. Objects are merged in decreasing order of the link probabilities. Merging is not done if the the cumulative size of the resultant object will be larger than the medium capacity. When no further objects can be merged, the cumulative objects are allocated to media. This allocation follows the optimal scheme of [2] in decreasing order of the cumulative static probability.

Note that when two objects are merged, the cumulative birth probability is simply the sum of the birth probabilities of the objects. Similarly, the probability for incoming edges from the same object are merged. For outgoing edges, a weighted sum of the probabilities is used if both merging objects have edges to the same object. The summing is done according to the static probability of the merging objects. The resulting static probability of the merged objects are computed in a manner similar to that explained earlier for the Static Probability scheme.

**Hot Edge Merge Scheme**: This scheme is very similar to the Edge Merge scheme. The only difference is that only edges that have a probability greater than a preset value (i.e. the "hot" edges) are merged. The idea is that this scheme will result in media with very high probability of access which will remain loaded most of the time.

**Birth Hop Scheme**: This scheme presents an alternative technique for combining direct and indirect access patterns. As in the hot edge merge scheme, we hope to use both object access probability and browsing graph information. The birth hop scheme works as follows. We begin by assigning the object with the highest birth probability to a blank medium. Following this step, we place as many objects as possible onto the same medium in decreasing order of either edge probability (from objects already allocated to the medium) or birth probability. Once the medium is full, we assign the object, from those that are unallocated, with the highest birth probability to a new medium and repeat the process. This operation is repeated until all objects are allocated.

**Static Hop Scheme**: This scheme is similar to birth hop scheme, except static probability instead of birth probability. The idea of this scheme is to allocate an object to a medium, we can either choose an object with highest static probability, or we can choose an object that has high probability edges with objects already on that medium.

## 4.3 Adaptive Placement

A key component of the proposed data placement schemes is knowledge of the access pattern. Although it is useful to know this a priori, it is not critical to the success of the proposed approach. Such information can easily be gathered from the system by keeping track of object requests. Based upon the observed access pattern, the data placement on tertiary storage can be tuned. In Section 5 we show the effectiveness of this adaptive placement in response to changes in the access pattern. In the complete absence of access information, the placement can begin with an initial guess for the access patterns followed by progressive refinement as user requests are serviced and the actual pattern is discovered.

## 4.4 Impact of Secondary Storage

In hierarchical storage systems, the secondary storage disks typically serve as a cache for data on tertiary storage. Depending upon the size of the disk layer and the caching (or migration) policy, some of the requests for objects are serviced directly from disk without impacting tertiary storage. The effect of the disk cache can be translated into a change in the effective access pattern observed at the tertiary level. An adaptive strategy for tertiary storage can exploit this change in access pattern to generate a placement better suited for the available secondary storage cache.

## 4.5 Replication

Data objects that have strong links to objects in different media are likely to cause excessive swapping of media. While such situations will hopefully not arise often, it is possible that an object may have strong links to objects in different clusters. These two clusters may be placed on separate media due to their size. To overcome this, we propose to selectively replicate objects on multiple media based upon their edge probabilities to objects in various media. Furthermore, for schemes that place related collections of objects, it is possible that there are segments of media that not filled - these can be used to replicate objects for "free" since the extra space is not large enough for a cluster and would otherwise be empty.

## 4.6 Prefetching

Schemes that place collections of related objects together aim to avoid swapping of media for a sequence of requests from a user. It is possible, however, that in order to service the requests of other users, the media may be swapped. This could result in thrashing between the users and expensive swapping. To avoid this situation we investigate the use of prefetching of related objects from a medium before ejecting a loaded medium. Prefetching further delays pending requests and also uses up disk space. It is therefore important to make a good judgment about when and how much to prefetch.

## 5 Experimental Results

In this section we demonstrate the effectiveness of our new data placement schemes towards reducing average response time. The results are based upon a detailed CSIM [16] simulation model of the system. The tape library is modeled on the Ampex DST tape library configured with Ampex DST 310 drives [9]. Further details of the tape simulator can be found in [12]. The Secondary storage is configured with four 5GB disks, totaling 20 GB of disk storage. The tertiary storage component is modeled on a robotic tape library with four Ampex DST drives. Some of the important parameters for the tape simulation are provided in Table 1. The experiments were conducted on a synthetic collection of 10,000 objects of size 100 Megabytes each. The tape library is configured with 2000 tapes each of size 2GB, giving a total of 4TB of tertiary storage.

The set of objects and the access pattern is generated as follows. The birth probability of objects follows a Zipf distribution. In order to capture the effects of links between objects, we introduce the notion of edges between objects. To determine the edges, the objects are divided into clusters. The number of objects in a cluster is uniformly distributed between 5 and 20. Some (5%) of the objects are considered to be outliers that do not belong to any cluster. For each object, a *death probability*, $p_d$, is picked uniformly distributed between

| *Parameter* | *Value(s)* | *Meaning* |
|---|---|---|
| TAPE SIMULATION PARAMETERS | | |
| RWD_OVHD | 0.0006 seconds | Rewind Overhead |
| SEEK_OVHD | 0.0006 seconds | Seconds |
| SEEK_SPEED | 110 MB/s | Tape seek rate |
| EJECT_TIME | 4 seconds | Time to eject a tape |
| LOAD_TIME | 10.1 seconds | Time to load a tape on a drive |
| PICK_TIME | 3.7 second | Time for robot to grab a tape |
| PUT_TIME | 1 second | Time for robot to drop a tape |
| MOVE_TIME | 1.9 second | Time for robot to move |
| XFER_SPEED | 14.2 MB/s | Tape transfer speed |
| NUM_TAPES | 2000 | Total number of tapes |
| TAPE_CAP | 2 GB | Tape cartridge capacity |
| NUM_DRIVES | 4 | Number of Drives |
| DISK SIMULATION PARAMETERS | | |
| ROT_SPEED | 4002 | Rotational speed RPM |
| SEC_TR | 72 | No. of sectors per track |
| CYLINDERS | 1962 | No. of cylinders |
| TR_CYL | 19 | No. of tracks per cylinder |
| TRKSKEW | 8 | Track skew in sectors |
| CYSKEW | 18 | Cylinder skew in sectors |
| CNTRL_TIME | 1.2 | Controller overhead (ms) |
| CAPACITY | 5 GB | Disk storage capacity |

Table 1: Table of Parameters

0.05 and 0.2. This is the probability that the user does not follow any of the links from this object. Edges to other objects within the cluster are created and assigned probabilities that are uniformly distributed so as to add to 1 - $p_d$.

It is important to note that although the access pattern is an input to the placement algorithm, it is not crucial that this pattern be accurate. As mentioned earlier, if the access pattern is unknown or changes after the placement, the system can adapt by reorganizing the data according to the new observed access pattern. Experimental evidence to support this claim is presented in Subsection 5.2.

In each experiment, we run a stream of requests. The stream begin by requesting a starting object identified using the birth probability for that object. As soon as this object is retrieved, the user chooses to either follow one of the edges from this object, or to pick another object independently. This choice is based upon the edge probabilities and the death probability of the currently accessed object. In each test, we run 1000 requests based upon which we compute the average response time.

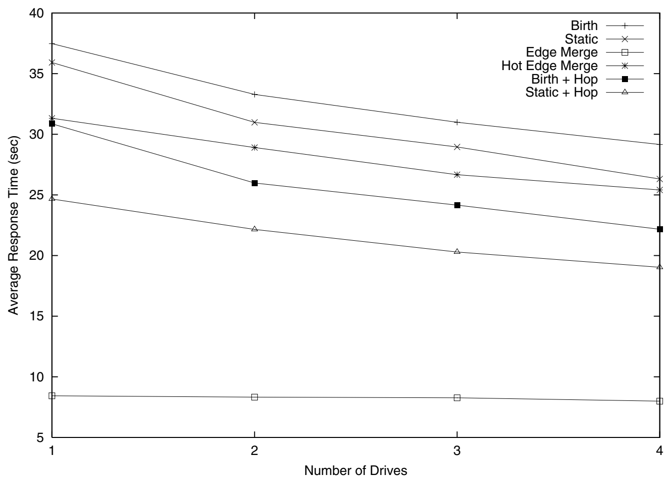


Figure 1: Average Response Time for Different Data Placement Schemes

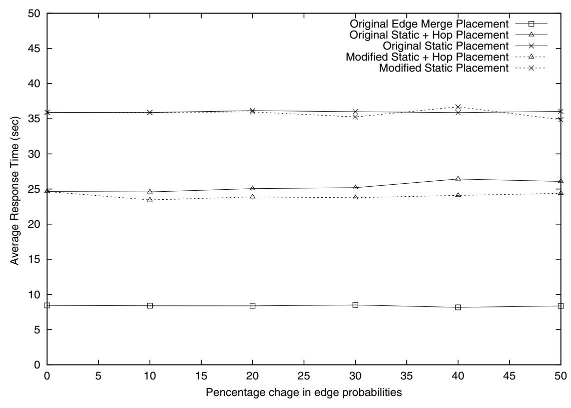
## 5.1 Different Data Placement Schemes and Performance

We begin by studying the relative behavior of the different schemes in reducing average response time. Figure 1 shows the average response time by different schemes. The number of drives was varied from 1 to 4. As can be seen from the graph, the $Edge\ Merge$ scheme gives the best performance, and the $Birth$ scheme has the worst performance. The $Static$ scheme has less average response time than $Birth$ scheme. The Edge Merge scheme reduces the average access time by $77\%$ compared to the Static scheme for a single drive. We can also observe that as the number of drives increases, the average response time reduces for all schemes. The superior performance of *Edge Merge* was observed in all our experiments. The scheme that does not consider the relationships between objects (Birth) has the poorest performance. Similarly, the Static scheme has poor performance since it does not use the link information effectively.

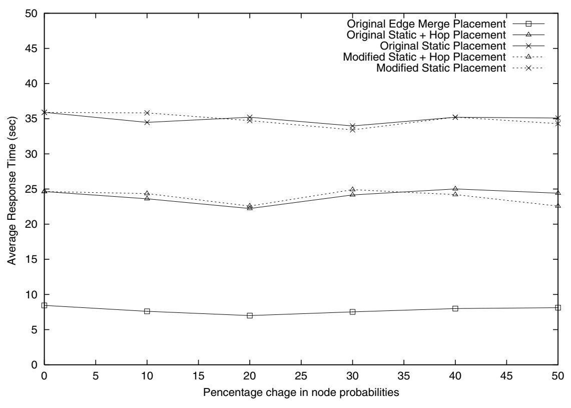
## 5.2 Adapting to Variations in Access Pattern

In the preceding experiment it was assumed that the access pattern is known a priori. This information is used to generate the placements. If the access pattern is unknown or changes after the placement, the placement may be less beneficial. The actual access pattern can easily be discovered by recording the requests for objects. Based upon this input, a more effective placement can be achieved. Note that through observation, it is not possible to distinguish between direct and indirect access to an object. When object $j$ is requested following a request for object $i$, it is not clear whether or not $j$ was accessed due to a link from $i$ to $j$. Consequently, the schemes based upon birth probability would not be applicable. We now investigate the impact of these variations.

In Figures 2 (a) and (b) we study the impact of random changes in the object access probabilities and the edge probabilities respectively. In each experiment the placement is



(a)



(b)

Figure 2: Impact of Changes in (a) Edge; and (b) Node probabilities

generated based upon an initial access pattern. Next, a random subset of 10% of the nodes (edges) are chosen and their probabilities are altered to varying degrees. The performance is tested using this altered access pattern. The frequency of access to documents based upon this altered graph is captured and a new placement is made based only upon these observed frequencies (with no other knowledge of the changed access pattern). Using this adapted placement, the performance is again measured. This is repeated for varying degrees of changes from the original access pattern. From the graphs we observe that changes in edge and node probabilities have very little impact on the data placement schemes. These experiments show the impact of changes in the distribution of the node and edge probabilities while keeping the structure of the access pattern fixed. In other words, the results showed that if we know the groups of objects that are related, exact knowledge of the probabilities is not critical.

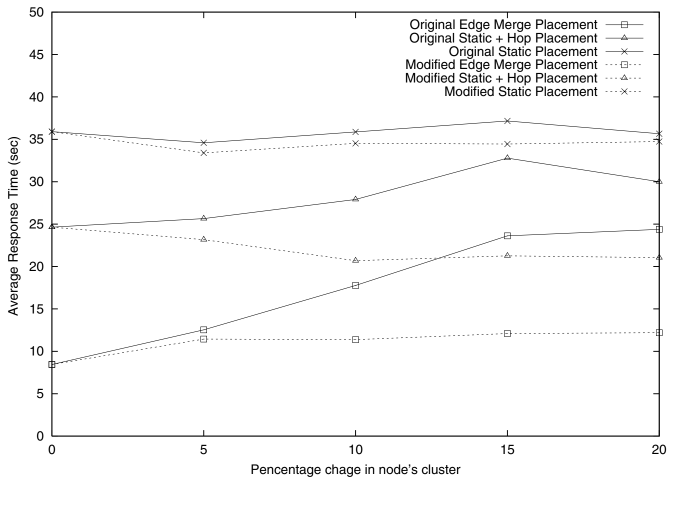


Figure 3: Impact of Changes in Node's cluster

In this experiment we study the impact of poor knowledge (or lack of knowledge) about the grouping of related objects. In Figures 3 we study the impact of limited random changes in the object cluster composition. The placement is generated based upon an initial access pattern. Next, a random subset of 5%, 10%, etc of the nodes are chosen and the node's cluster membership is changed. The performance is tested using this altered access pattern. We also measure the performance of an adapted placement based upon the observed access pattern. As can be seen in the graph, changes to cluster composition result in an increase in the average response time for both placement schemes. However, we see that after adapting to the new pattern, we are able to reduce the response time. The response time is reduced sharply in $Edge\ Merge$ scheme, it drop to same level as no change to the access pattern. We can also notice that even without adapting to the new placement, the $Edge\ Merge$ scheme still performs better than $Static$ scheme.

From these three graphs we see an interesting result: information about the clustering or grouping of related objects is more critical than exact information about the probabilities of access. This is good news since these relationships are generally easy to discover statically based upon the application semantics (e.g. urls in a given web page). The results also underscore the importance of not making the assumption of independent access.

## 5.3 Impact of Secondary Storage

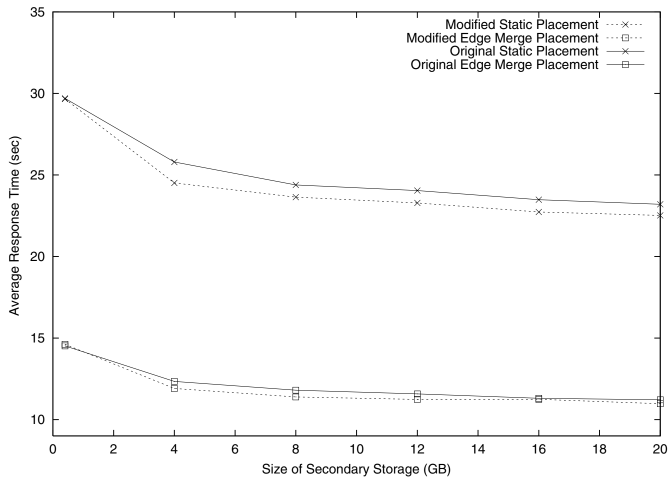


Figure 4: Impact of Secondary Storage

In this experiment, we study the impact of the size of the disk buffer. In hierarchical storage system, the secondary storage disks typically serves as a cache for data on tertiary storage. User requests for data cached in the buffer are served without any access to tertiary storage. If the requested object is not in the disk cache, the object is copied from tertiary storage to buffer, then from the buffer to the user. A buffer replacement policy is used to create space when the buffer becomes full. In our experiments, we use the popular Least Recently Used (LRU) cache replacement policy.

Figure 4 presents the performance for the various schemes for different buffer sizes. The buffer size is varied from 400MB upto 20GB. We can observe from the graph that as the size of the buffer increases, the average response time decreases for all schemes. We also observe that the presence of a disk cache does not change the relative performance of the $Edge\ Merge$ scheme and the $Static$ scheme.

Since we have secondary storage as cache. The effect of the disk cache can be translated into a change in the effective access pattern observed at the tertiary level. The hot objects (objects with high static probability) may not be hot at the tertiary level since these objects may always be cached on disk. In order to account for this change in the access pattern, we can adapt the placement based upon the observed access pattern at the tape level as was done in Section 5.2. In Figures 4, we study our new data placement based on observed access pattern. As we can seen from the graph, the new adapted data placement slightly better than original data placement.

## 5.4 Impact of Replication

In our original model, each object only has one copy in tertiary storage. To replicate objects on tertiary storage, there are two approaches. The first approach is to replicate some frequently requested objects. We can use this approach with the $Birth$ and $static$ schemes. However, disk caching will reduce the effectiveness of this approach because most of hot objects will reside in cache. The second approach is to replicate related objects when free space is available on a medium. This approach works for $Edge\ Merge$ scheme and $Hot\ Edge\ Merge$ scheme. In our experiment, we mainly study the $Edge\ Merge$ scheme with the second approach due to its superior performance. Unused segments on a medium are filled using the following rules. First objects that have strong connections with objects already in the medium are replicated. If space still remains after considering such objects, hot objects are replicated. The results of the experiment are shown in Figure 5. It can be seen that the free data replication results in a noticeable improvement in performance.

## 5.5 Prefetching Issues

As stated in the last section, prefetching related objects can be beneficial. The disadvantage is that prefetching delays pending requests further and uses up disk space. In this subsection, we study the impact of prefetching for the proposed schemes. In order to see the impact of the amount of prefetching performed, we tested our six schemes with different prefetching sizes. In this experiment prefetching is performed whenever possible. When a new tape is loaded onto the drive, any object not in the disk cache may be prefetched. The total amount of data prefetched from a single medium is varied from 0 to 300 MB. The results of the experiment are shown in Figure 6. As can be seen, most schemes benefit from prefetching when the prefetch size is 100 MB. For larger sizes, only the Edge Merge scheme benefits – the average response time is reduced by 13%. This is explained by the fact that the Edge Merge scheme is based on the relationship between objects. When one object is retrieved, the most connected objects are likely to be in the same medium, so

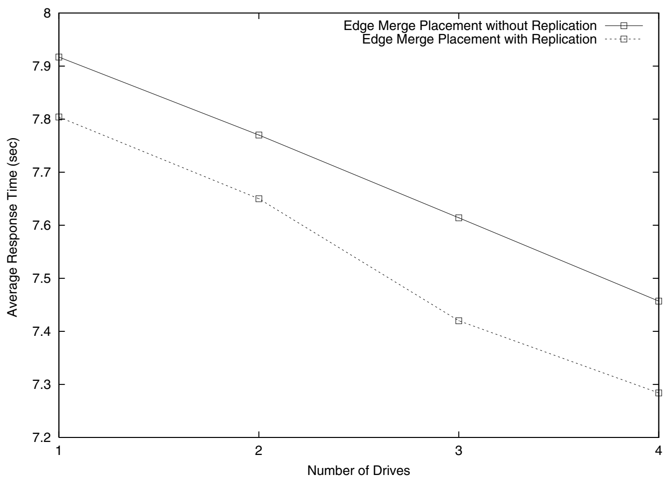


Figure 5: Impact of Replication on Edge Merge Scheme

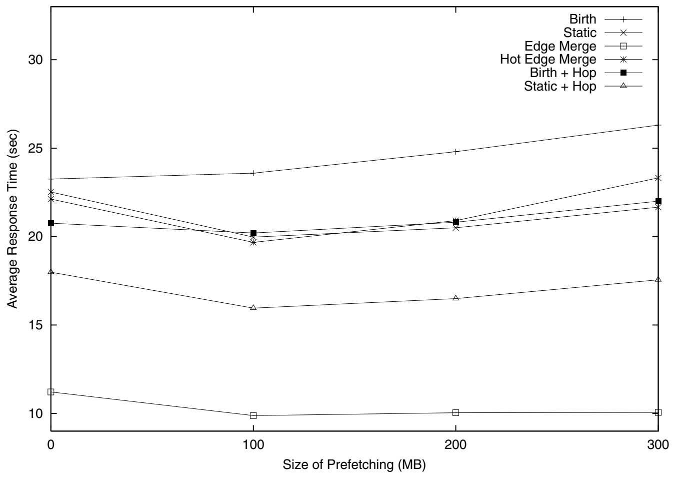


Figure 6: Impact of Prefetching on Different Schemes

prefetching is beneficial. Prefetching is not good for the Birth Scheme because under this scheme related objects are scattered in different media. In fact the penalty of prefetching larger than 100MB of data is higher than the benefit.

Next we study the choice of when to prefetch with the Edge Merge scheme in order to make prefetching most effective. In the last experiment we prefetched blindly. In this experiment, we prefetch only if there is a suitable object. There are two kinds of candidates for prefetching when a medium is loaded for retrieving object $O_i$: i) objects with strong links from $O_i$; and ii) objects with a large static probability. The experiment is controlled by two parameters: a minimum edge probability (say $min_{ep}$) and a minimum static probability (say $min_{sp}$). If an object in same medium has edge probability greater than $min_{ep}$ or static probability greater than $min_{sp}$, that object will be a prefetching candidate. Since we cannot prefetch all candidates the amount of data prefetched is limited. The results are shown in Figure 7. Only Edge Merge is studied, with several prefetching policies. We study 4 policies: i) $min_{ep}$ = 0.5, $min_{sp}$ = 0.005, this is most restrictive policy; ii) $min_{ep}$ = 0.3, $min_{sp}$ = 0.0003, preference is given to high static probability objects; iii) $min_{ep}$ = 0.05, $min_{sp}$ = 0.005, preference is for high edge probability objects; and iv) $min_{ep}$ = 0.05, $min_{sp}$ = 0.0003, this is the most liberal policy. As we can observed from graph the most liberal policy gives better performance than the most strict policy. The two schemes that have a low threshold for the edge probability give better performance for small prefetch sizes, but their performance degrades for larger prefetch sizes.

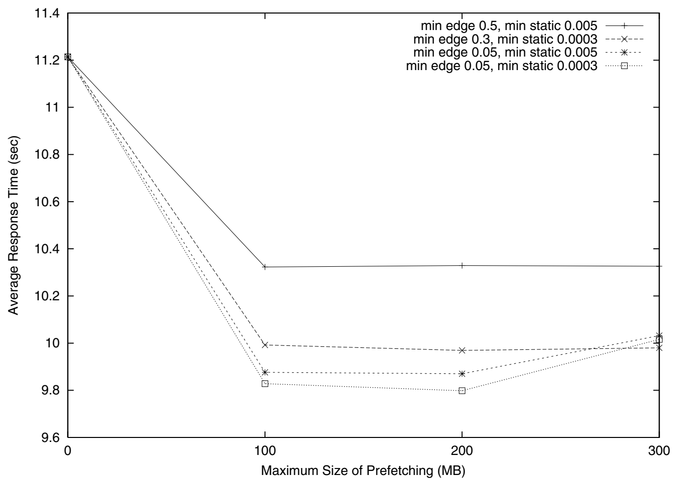


Figure 7: Impact of Different Prefetching Policies on Edge Merge Scheme

## 6 Conclusion

In this paper we address the important problem of data placement in tertiary storage taking object relationships into account. We also study the advantage of limited replication in this setting. This work is in contrast to earlier schemes that either focus on specific data types or assume that data objects are independently accessed. To the best of our knowledge, this is the first study to explore these issues. We propose five new data placement schemes. The effectiveness of these schemes in reducing average response time is shown through extensive experimentation using a detailed simulator. We find the Edge Merge scheme has best performance. The performance of placement schemes that are known to be optimal under the assumption of independent access is not as good as that of the proposed schemes.

We also show that our schemes can easily adapt to variations in the access pattern. In fact this allows the schemes to be employed when no prior information about the access pattern is available. The schemes progressively adapt to give good performance as the access pattern is learned. Capturing the access pattern is easily achieved at the tertiary storage level. In all cases, adjusting the placement to the new observed pattern resulted in significantly improved performance. Interestingly, our results show that the probabilities of access (node and edge) do not have a big impact on our Edge Merge scheme. Changes to the clustering of nodes, on the other hand, has a greater effect. This goes to show the importance of the inter-relationships between objects. The use of controlled replication for "free" is also developed and shown to be effective in improving performance further. The impact of disk caching is easily handled in a manner similar to that of variation in access patterns. The effective access pattern at the tertiary layer is measured and used to place the data, rather than the overall access pattern. The techniques are coupled with prefetching which is found to be beneficial for the Edge Merge scheme.

Overall, we see that the proposed techniques are very effective in placing data on tertiary storage. The techniques perform much better than schemes that are optimal under the assumption of independent access. In our experiments the Edge Merge scheme achieved as much as 77% reduction in average access time over the state-of-the-art scheme (Static).

**Acknowledgment** This work was supported by the National Science Foundation under CAREER grant IIS-9985019, and Research Infrastructure Grant 9988339-CCR.

## References

[1] L. T. Chen, R. Drach, M. Keating, S. Louise, D. Rotem, and A. Shoshani. Efficient organization and access of multi-dimensional datasets on tertiary storage systems. In *Information Systems*, volume 20, pages 155–83. Elsevier Science, 1995.

[2] S. Christodoulakis, P. Triantafillou, and F. Zioga. Principles of optimally placing data in tertiary storage libraries. In *VLDB'97, Proc. of Intl. Conf. on Very Large Data Bases, 1997, Athens, Greece*, pages 236–245, 1997.

[3] D. A. Ford and S. Christodoulakis. Optimizing random reterievals from clv format optical disks. In *Proceedings of the Int. Conf. on Very Large Data Bases*, pages 413–22, Barcelona, Spain, September 1991.

[4] C. Georgiadis, P. Triantafillou, and C. Faloutsos. Scheduling and performance of robotic tape libraries in video server environments. Technical report, Multimedia Systems Institute of Crete (MUSIC), Tech. Univ. of Crete, Crete, Greece, 1997.

[5] S. Ghandeharizadeh, A. Dashti, and C. Shahabi. Pipelining mechanism to minimize the latency time in hierarchical multimedia storage managers. *Computer Communications*, 18:170–184, march 1995.

[6] S. Ghandeharizadeh and C. Shahabi. On multimedia repositories, personal computers, and hierarchical storage systems. In *Proc. of ACM Int. Conf. on Multimedia*, 1994.

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

[8] B. K. Hillyer and A. Silberschatz. Random I/O scheduling in online tertiary storage. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, Canada, 1996.

[9] T. Johnson and E. L. Miller. Performance measurements of tertiary storage devices. In *Proc. of 24rd Intl. Conf. on Very Large Data Bases*, pages 50–61, New York, 1998.

[10] A. Kraiss and G. Weikum. Vertical data migration in large near-line document archives based on markov-chain predictions. In *Proceedings of 23rd International Conference on Very Large Data Bases*, pages 246–255, Athens, Greece, August 1997.

[11] S. More, S. Muthukrishnan, and E. Shriver. Efficiently sequencing tape resident jobs. In *Proc. ACM Symp. on Principles of Database Systems*, 1999.

[12] S. Prabhakar. An overview of current tertiary storage technology and research. Master's thesis, University of California, Santa Barbara, 1998.

[13] S. Prabhakar, D. Agrawal, A. El Abbadi, and A. Singh. Scheduling tertiary I/O in database applications. In *Proc. of the 8th International Workshop on Database and Expert Systems Applications*, pages 722–727, Toulouse, France, September 1997.

[14] S. Prabhakar and R. Chari. Minimizing latency and jitter for large scale multimedia repositories through prefix caching. Technical Report CSD 01-018, Department of Computer Sciences, Purdue Univeristy, September 2001.

[15] S. Sarawagi. Database systems for efficient access to tertiary memory. In *Proc. of 14th IEEE Symp. on Mass Storage Systems*, pages 120–6, Monterey, California, 1995.

[16] H. D. Schwetman. CSIM: A C-based, process-oriented simulation language. In *Proceedings of the 1986 Winter Simulation Conference*, pages 387–396, December 1986.

[17] S. Seshadri, D. Rotem, and A. Segev. Optimal arrangements of cartridges in carousel type mass storage systems. *The Computer Journal*, 37(10):873–887, 1994.

[18] P. Triantafillou, S. Christodoulakis, and C. Georgiadis. Optimal data placement on disks: A comprehensive solution for different technologies. Technical report, Multimedia Systems Institute of Crete (MUSIC), Tech. Univ. of Crete, Greece, 1996.

[19] P. Triantafillou and T. Papadakis. On-demand data elevation in hierarchical multimedia storage servers. In *Proc. of 23rd Intl. Conf. on Very Large Data Bases*, pages 226–235, Athens, Greece, August 1997.

[20] P. Triantafillou and T. Papadakis. Exploiting tertiary storage for performance improvement in video-on-demand servers. Technical report, Multimedia Systems Institute of Crete (MUSIC), Technical University of Crete, Crete, Greece, 1998.

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