Object Placement in Parallel Tape Storage Systems

Xianbo Zhang, Dingshan He, David H.C. Du and Yingping Lu
Department of Computer Science and Engineering
DTC Intelligent Storage Consortium (DISC)
University of Minnesota, Minneapolis, MN 55455
{*xzhang,he,du,lu*}*@cs.umn.edu*

Abstract

High performance computing and enterprise data center require huge amount of data transfer between disk and tape. With the help of large capacity disk, writing to tape is less problematic than reading from tape. We are investigating how to use multiple tape libraries to build a parallel tape storage system with high aggregated data retrieval bandwidth. The challenge lies in that increasing data transfer parallelism may also increase tape switch time and data seek time that reduce the effective data retrieval bandwidth. We are proposing a novel scheme to place objects across tape drives based on object access pattern to reduce tape switch time and in the meanwhile increase data transfer parallelism. A multiple-tape-library simulator is built to study the proposed scheme. Simulation results show that our scheme outperforms other two previously proposed schemes with a better trade-off between tape switch time, data seek time and data transfer time.

1. Introduction

In the high performance computing cluster environment there are many data intensive applications that feature high I/O bandwidth and huge storage space requirement. Due to the large amount of data generated within the cluster environment [3, 4], moving data between disk and tape is a necessary step to assure the system of a sustained high performance. Thus, high performance data transfer between disk and tape would be extremely important for these cluster file systems. This high performance data transfer between disk and tape is equally important to the backup/archive system for enterprise data centers. For business continuance and regulation requirement, critical data are periodically or even continuously backed up to tapes. If any data loss should occur, the data backup needs to be restored. The total restore time has to be minimized to reduce enterprise financial losses.

Due to decreasing cost and increasing capacity of the ATA/SATA drive, staging disk and disk based virtual tape library (VTL) [5] are widely placed in front of the physical tape libraries to temporarily backup/archive the large amount of data from high performance computing cluster or enterprise data center. These disks work as a high performance buffer with low access latency helping the sequential-access tape drives write in streaming mode with high performance. However, this disk data buffer can not help tape drives retrieve data in high speed. With a common backup/archive process, data are moved onto tapes without data retrieval in mind.

We focus our research on increasing the data retrieval speed for tape storage system within the high performance computing environments and enterprise data centers. The restore operation is needed more frequently than everyone's initial thought. To improve the utilization and fair access of high performance computing cluster resources, users are usually preallocated a time slot to run their applications. When the running application becomes inactive at the end of the allocated time slot, the application and its related data have to be migrated to tape to guarantee enough high performance storage space to other active users. When the user's time slot comes again, the migrated data have to be restored to the cluster to resume the user's work for further computing/analysis. For some numerical simulation program, the whole running time goes from days to months [7]. The running state of the program is backed up to a disk and eventually to a tape at every checkpoint for data recovery in the event of any hardware or software failure. If a failure does happen, the data backup from the latest checkpoint needs to be brought back to the primary disk to restore the running state of the simulation program.

In this paper, we are investigating how to use multiple tape libraries to build a parallel tape storage system with high aggregated data retrieval bandwidth through proper data placement and tape switch strategy. Intuitively, multiple tape libraries working together can increase the aggregate bandwidth between disk storage and tape storage and reduce the tape switch time by introducing parallel load/unload operation. However, as noted in [11, 15, 20], without a proper data placement across drives, increasing data transfer parallelism may also increase tape switch time and data seek time, which delay data transfer and de-

crease the effective data retrieval speed. To make the intuition come true, we are proposing a novel object placement scheme based on object access relationship to place object across tape drives/libraries to reduce tape switch time and increase data transfer parallelism, thus achieving high effective data retrieval bandwidth. Our proposed scheme achieves good trade-off between tape switch operation, data transfer operation and data seek operation and outperforms two other previously proposed placement schemes according to the simulation.

The rest of this paper is organized as follows. An overview of related work on data placement for tape libraries is given in Section 2. The problem is formulated in Section 3. Section 4 briefly describes the object placement trade-offs on parallel tape libraries. Our proposed object placement scheme is presented in Section 5 and evaluated by simulation in Section 6. The conclusions and discussion are presented in Section 7.

2. Related work

Tertiary storage libraries are demanded by many multimedia and scientific computing applications due to their huge storage requirements. There is a large body of research on improving tape request response time, which consists of optional tape cartridge switch time, object seek time and data transfer time. In order to reduce the number of tape switches, [20] explores object relationship and puts objects with a strong relationship on the same tape media. The authors assume that the tape switch time dominates the whole request response time, which may not be true for some tape systems, and the potential of parallel data transfer is ignored. Research studies in [24] focus on optimizing object seek time with proper object alignment. Based on independent object access probabilities, the optimal solution within one tape ends up with an organ pipe alignment of access probabilities. To reduce average seek time for multiple tapes, [11] develops different optimized placements for tapes that either rewind to the middle position of the tape or rewind to the beginning position of the tape when they are unmounted. Such optimal solutions do not consider the effects of tape switch and object relationship. [22] shows that an organ-pipe arrangement is also optimal for cartridges and file partitioning under different storage configurations in carousel type mass storage systems. More studies have been conducted on using object-striping techniques to reduce data transfer time [15, 10, 13, 14]. Striping is a well-known technique for improving the effective I/O bandwidth for disk storage systems [21, 8, 6, 23]. Unfortunately, striping on sequential-accessed tapes suffers from long synchronization latencies not faced by random-accessed disks. The optimal striping width on tapes depends on object size, system workload, the ratio of readers and robotic arms. The striping system may perform worse than non-striping system [9, 13, 19, 10]. Thus, in our proposed scheme, we do not consider object striping.

3. Problem formulation

We make the following three assumptions about data retrieval characteristics, which to a large extent reflect the data retrieval characteristics within the high performance computing environment and a backup/restore system [18, 12]: (1) Objects form clusters and a cluster of objects have high chance to be retrieved together; (2) Object clusters have different access probabilities, and an access probability can represent any manually assigned weight or priority of the object cluster; (3) Most data object accesses to the tape storage exhibit whole object sequential access patterns, i.e., the entire object is retrieved from tape.

Figure 1 presents the data transfer architecture we are exploring, where each tape drive has a limited data transfer rate compared to the high performance disk arrays. Tape drives can transfer data to the disk cache in parallel, however the tape load/unload within a tape library is sequential due to the constraint of one robot in a tape library. Robots work independently across tape libraries. We define the tape request response time as the time interval from tape storage system receiving an object request to finishing all requested object transfers. The effective data retrieval bandwidth of a tape storage system is equal to the size of transferred data divided by the tape request response time. For the same amount of data, the longer the tape request response time, the lower the corresponding effective bandwidth.

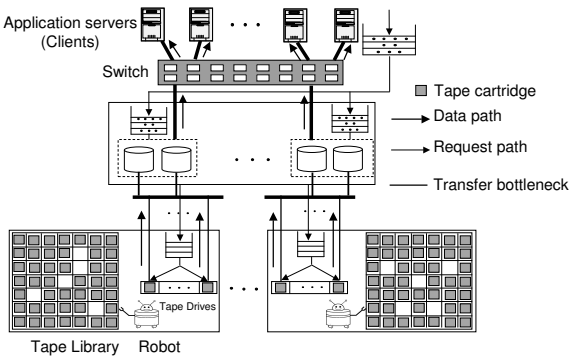

Figure 1. Parallel tape storage system model

The problem we are investigating is: How to place data objects among tape libraries with multiple tape drives to minimize the average tape request response time of N_{req} requests $\sum_{i=1}^{N_{req}}(P(R_i) \times t(R_i))$. Where $P(R_i)$ is the access probability of request R_i while $t(R_i)$ is the request response time for request R_i , which equals to the summation of T_{switch}, T_{seek} and $T_{transfer}$. All three components are variables that depend on the placement and the retrieval scheduling for the requested objects. The placement tradeoffs are discussed in Section 4.

In summary, we are assuming the following:

(1) A set of N_{obj} objects $(O_1, O_2, ...)$ with different sizes.

(2) A set of N_{req} requests: $\{R_1, R_2, ...\}$. The access probability of each request is known; each request can ask for one or more objects.

(3) The service for a request is completed when the last object of a request has been transferred to disk.

(4) The same object can be asked for by different requests.

(5) We have n identical libraries; each tape library has: One robot arm for loading and unloading tapes to drives; d tape drives, $\{D_1, D_2, ..., D_d\}$, each with the same performance properties; t tapes, each with the same capacity C_t; $d << t$.

(6) The bottleneck of data transfer path lies at tape drive, i.e., network or communication channel contention is negligible elsewhere.

We believe that the optimal solution for the problem is NP-hard and will only look for a heuristic solution.

4. Object placement trade-offs

As mentioned before, the request response time is the time period from receiving a data request to finishing all requested data transfers, which is composed of three items: (1) T_{switch}: if the required tape drive is occupied by another tape, it includes the time of tape rewinding, tape unmounting, tape returning to storage cell and fetching the required tape by robotic arm to the drive, and tape mounting. Otherwise, T_{switch} equals to 0 if the tape cartridge is already mounted on a tape drive; (2) T_{seek}: the time to locate the required objects; (3) $T_{transfer}$: the time to transfer the requested objects from tape to disk.

For a specific request, requested objects may or may not all reside on the same tape. They may be across multiple tapes within the same tape library or even across multiple tape libraries. Depending on the object placement, the response time for a request may end up with one of the following three cases, associated with different cost functions.

Case 1: All requested objects are on the same tape. The request response time is determined by the response time of the single tape: $T = T_{switch} + \sum_{i=1}^{N_{req}} (T_{seek,O_i} + T_{transfer,O_i})$. The total seek time can be minimized if the requested objects are located next to each other on the tape and close to the tape drive head position. T_{seek,O_i} and $T_{transfer,O_i}$ are the seek time and transfer time of the $i-th$ object of the request. The pros and cons are summarized below.

Pros: Reduced T_{switch} due to at most one switch per request which reduces contentions for tape drives and the robot

Cons: No data transfer parallelism within a request

Case 2: Requested objects are spread across multiple tapes within the same tape library. Object transfers are in parallel but tape switches are sequential assuming one robotic arm per tape library. The total request response time is determined by the maximum time used by an involved tape. The required time to transfer data from each involved tape is just as described in Case 1. The pros and cons of this placement are generalized as follows:

Pros: Reduced data transfer time if tape drive data transfers happen in parallel

Cons: Potential to increase T_{switch} due to increased contention for tape drives and the robot

Case 3: Objects are further spread across multiple tape libraries. The difference of this case compared to Case 2 is that multiple robotic arms work in parallel. Therefore, we can take advantage of the possible tape switch parallelism between tape libraries. However, within a tape library, the same concerns exist as in Case 2. It has the following pros and cons.

Pros: Reduced data transfer time due to increased parallelism within a request; reduced T_{switch} due to parallel robotic arms

Cons: It may increase contention for tape drives and introduce a wider range of tape drive startup delay

Load balancing among tape drives is important to reduce request response time for all cases. In summary, there are many factors that affect the tape request response time and these factors are affecting each other. A good object placement should provide a good trade-off between tape switch time, data seek time and data transfer time.

5. Proposed object placement

Considering the big penalty associated with tape switches and long transfer time associated with huge request size, we are proposing an object placement scheme to leverage object access probabilities and object relationships to increase tape switch parallelism and synchronize data seek with high probability, and thus increase object transfer parallelism. We focus on reducing tape switches within a tape library, increasing tape switch parallelism across tape libraries and increasing data transfer parallelism among tape drives. To achieve these goals, we introduce the concept of tape batch and a special tape switch strategy to increase the probability of tape switch parallelism and data seek synchronization as well as reduce the number of tape switches.

A tape batch is composed of $n \times i$ tapes to which each library contributes i tapes. To achieve potential parallelism in tape switch, objects within a cluster should be spread within a tape batch with the observation that objects within a cluster have high chance to be requested together. To reduce the number of tape switches within a library, $(d-1)$ tapes should hold objects with accumulated probability as high as possible and be kept mounted all the time. It is mathematically proved in [11] that such a placement combined with the least popular replacement policy minimizes the number of tape switches. However, minimizing the number of tape switches may not minimize the tape switch

time. If a request needs to access objects from two offline tapes and only one drive working as a switch drive, one tape has to wait until the other one has completed the sequential operations of tape switch, data seek and data transfer. On the other hand, if there are two drives working as switch drives, one tape can start to be mounted just after the other one is mounted. The optimal number of switch drives, m, depends on the object probability distribution and the configuration of the tape libraries, and the change of m also changes the accumulated object probability of the always-mounted tape batch. There is no simple formula to get the optimal number. The value of m ranges from 1 to $d-1$, we investigate how to choose a close-to-optimal value of m through the simulation.

With all these considerations in mind, we developed a set of heuristic algorithms to minimize tape request response time, which are described in the following subsections covering object clustering, tape batch and corresponding switch strategy, object placement and tape load balancing.

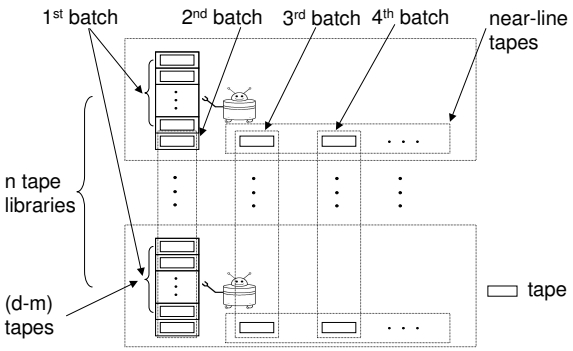

Figure 2. Tape batches across libraries

5.1. Object clustering

We define the similarity among objects as the probability they will be accessed together. The similarity of $P(O_i, O_j)$ between object O_i and O_j is the sum of the probabilities of all requests that contain both objects. The similarity of $P(O_i, O_j, O_k)$ among object O_i, O_j and O_k is the sum of the probabilities of all requests that contain all three objects, and so on. Following the hierarchical algorithm from [17] we can build an object relationship tree with each individual object being a leaf of the tree. Considering the bottom of the tree (leaf level) as level 1, a node of level n represents an access probability among n objects. Once the tree is formed, it is traversed following a depth-first method to get all the clusters based on a preset probability value. Thus, objects with high probability of being accessed together will be grouped into a cluster. Requests information are used to reduce the clustering computation costs. If we are looking for relationship among three objects, objects are chosen from requests with three or more objects and the probability compuation only needs to involve these requests with three or more objects. The quality of the object clustering, which is measured by the probability of objects being accessed together and proper cluster size in terms of the number of objects and the total required space, is vital for the success of the overall placement scheme. The cluster size can be controlled by the preset probability threshold. As a general rule, the number of objects in a cluster from the first sublist should be close to or less than $n\times(d-m)$ and the number of objects in a cluster from other sublists should be close to or less than $n\times m$ for maximum parallelism.

5.2. Tape batch and switch strategy

The tape library system is composed of n tape libraries; each has d identical drives and more than d tapes. Tape mount and unmount within a tape library is performed by one robotic arm in the library. As shown in Figure 2, all $n\times d$ drives are divided into two batches: the batch with tapes always kept mounted and the batch used for tape switches. The always-mounted batch is composed of $n\times(d-m)$ drives to which each library contributes $(d-m)$ drives; the other batch is composed of $n\times m$ drives to which each library contributes m drive. Accordingly, all tapes are divided into two kinds of batches: the first one contains $n\times(d-m)$ tapes composed of $(d-m)$ tapes from each library, and the second and later batches each contains $n\times m$ tapes composed of m tape from each library. The first batch of tapes matches the first batch of drives and is kept mounted all the time, and the second batch is mounted during startup time and will be swapped out if the requested object can not be found within the mounted tapes. We do not force tapes within a batch to be swapped all together, but with the proposed placement in Subsection 5.3, tapes within a batch is likely to be swapped together with high probability.

5.3. Placement algorithm

To reduce tape switch operations, tapes within the first batch should accumulate access probability as high as possible from all available objects to be allocated, tapes within the second batch accumulate probability as high as possible from the remaining unallocated objects, and so on. Equally important, objects within a cluster should be kept within one tape batch. Following these allocation principles, the minimum number of tape switches can be achieved. However, given a set of objects with various probabilities and sizes, how to achieve the tape probability distribution as described can be reduced to the well known 0-1 knapsack problem [16], which is a NP-hard problem. For simplicity and computation efficiency we developed the following heuristic solution.

Step 1. Compute object probability from request probability as $\sum_{O_i \in R_j} P(R_j)$.

Step 2. Sort objects into a decreasing order list based on individual object probability density. The probability den-

sity of object O_i is defined as $P(O_i)/size(O_i)$. The object relationship is not considered in here but in Step 4.

Step 3. Divide the formed object list into multiple sublists: the first sublist has the total object size less than the total capacity of $k \times n \times (d-m) \times C_t$ (the total batch capacity), where k, the tape capacity utilization coefficient, is less than 1; the rest of the sublists each has the capacity of $k \times n \times m \times c_t$ (another batch capacity).

Step 4. With batch capacity constrains, refine sublists based on object clusters formed using the algorithm described in Subsection 5.1. Most likely objects with strong relationship will not be far away from each other within the sorted list, and we only need to move objects between adjacent sublists. The refining goal is to make objects with a strong relationship fall into the same sublist to reduce tape switches while maintaining the skewed tape probability distribution, i.e., tape probability from 1st batch $>$ tape probability from 2nd batch $>$ tape probability from 3rd batch, and so on. Ideally, any request should cause at most one tape switch in a tape library.

Step 5. Allocate objects in each sublist to corresponding tape batches. For maximum data transfer parallelism, objects within a cluster are split to multiple tapes if their aggregate size is big enough. Otherwise, simply putting them on the same tape does not change data transfer time a lot but reduces tape switch time. Tape load balancing algorithm in Subsection 5.4 is used for splitting objects within a tape batch.

Step 6. Align objects within each tape following the organ-pipe alignment based on individual object probabilities [11].

5.4. Tape load balancing

Tape load balancing and maximum data transfer parallelism are the goal of allocating objects in a sublist to the corresponding tape batch. For each object cluster, we first decide how many tapes should be used based on cluster composition and available tapes. Then we split objects in the cluster to multiple tapes based on object load and tape load information. Load of object O_i is computed as $P(O_i) \times Size(O_i)$ while tape load is computed as the summation of object loads on the tape. To achieve load balancing and highly parallel data transfer we use a greedy algorithm shown in Figure 3 for placement within a tape batch.

6. Performance evaluation

We have built a multiple-tape-library simulator composed of more than 4,000 lines of code to study the performance of our proposed object placement scheme and compare it with two other previously proposed schemes briefly introduced in Section 2. For reference convenience, we call these two proposed schemes *object probability placement* and *cluster probability placement* respectively and present their main points as follows:

```
variable i, flag, ndrv
initialize workload of all considered tapes to 0
place all clusters of a sublist into cluster list C_set
for each cluster C in C_set do
    sort objects in C into increasing order based on load
    sort m tapes in decreasing order based on workload
    assign ndrv a proper value based on info of C and tapes
    i ← 0
    flag ← 0
    for each object O in the sorted list
        if (flag==0) then i←i+1 else i←i-1
        if (i==ndrv) then {flag←1; i←i-1}
        if (i==-1) then {flag←0; i←i+1}
        assign object O to tape T_i
        increase workload of T_i by P(O) * size(O)
for each tape do
    align objects based on organ-pipe alignment
    write objects to the tape
```

Figure 3. Greedy load balancing

Object probability placement Individual object access probabilities are assumed to be known. Within a tape, average object seek time is minimized when objects are placed in an organ pipe arrangement. The optimal placement scheme is schematically shown in Figure 4 for a library with 3 tapes assuming each tape can hold 5 equally sized objects. The proposed scheme optimizes object seek time based on independent object access probabilities [11].

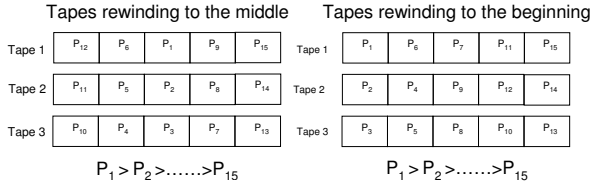

Figure 4. Object probability placement

Cluster probability placement Assuming the access cost in tape library is dominated by the media switch operation and head positioning delay. With the observation that objects have access relationship, the scheme places objects with strong relationship onto one tape to reduce tape switches[20]. It proposes several algorithms to form object clusters.

For convenience, our placement scheme is called *parallel batch placement*.

Simulator The simulator simulates multiple drives transferring data in parallel based on assumptions described in Section 3. Integrated with the simulator is an indexing database that stores object locations as well as other object properties such as object size information. Given a request, the corresponding tapes are identified based on the object indexing database. Requested object(s) residing on mounted switching tapes will be served before they are unmounted while the tape switch operation happens to any tape drive containing no requested objects. After a mounted switching tape finishes transferring the requested object(s), it is rewound and unmounted by a robot, and an

offline tape containing any requested object(s) is mounted by the robot to serve its contained object(s). Tape drives are working independently and robots across tape libraries are also working independently. The objects retrieving order within a tape is optimized to reduce the data seek time based on object location information retrieved from the indexing database. The servicing time of each tape drive, including the time waiting for robot, tape switch time, data seek time and data transfer time, is recorded for the request being served. The longest servicing time recorded with a tape drive is the tape response time for the request. The simulator models robotic arm mount/unmount operations as constant time values for a given tape library, and computes object seek/tape rewind time using a linear positioning model as described in [18], i.e, the tape drive head positioning time is proportional to the distance between the head start position and the end position. We also assume the tape drive reads an object in streaming mode after the tape drive head is positioned to the beginning of the object. Considering all the tape switch operations and data location operations for data read, the tape drive hardly works in streaming mode most of the time and the effective data retrieval rate can not reach the manufacturer's rated speed. As mentioned before, robots and tape drives work in parallel without forced synchronization.

Simulation Settings: We are using the following parameters to set up our simulation, as shown in Table 1. These parameters follow the specifications of IBM LTO Gen 3 tape drive and StorageTek L80 tape library [1, 2].

Table 1. Tape drive/library specifications

Average cell to drive time	7.6s
Tape load and thread to ready	19s
Data transfer rate, native	80 MB/s
Maximum/average rewind time	98/49s
Unload time	19s
Average file access time (first file)	72s
Number of tapes per library	80
Tape capacity	400GB
Tape drives per library	8
Number of tape libraries	3

Since the data restore request is submitted one by one within our considered environment with long time interval between two requests, the request queuing time in the request queue is zero and does not go into the object bandwidth calculation. The simulation objects and requests are generated as follows. The total number of objects is fixed as 30,000 and the total number of pre-defined requests is 300. The object size follows a power law distribution within a pre-defined range. The number of objects contained in a request also follows a power law distribution ranging from 100 to 150. The objects in a request are randomly chosen. The same object may be included in several different requests. The access pattern of request follows a $Zipf$ distribution. Request popularity $P_r = c \cdot r^{-\alpha}$, where r is the request rank starting from 1, c is a constant value and α controls request popularity distribution. We get uniform popularity if α=0 and most skewed popularity if α=1.

Metrics: To have a deeper understanding of the behavior of each scheme, in addition to the effective data retrieval bandwidth, we also use the following four metrics to evaluate each scheme: average response time, average tape switch time, average data seek time and average data transfer time. By analyzing tape switch, object seek and data transfer time we can have a better understanding of the request response time. After the object allocation is done, based on the probability distribution a request is chosen from the 300 pre-defined requests and submitted to the simulator to get these four metrics. This repeats 200 times to get the average value for each metrics. The tape switch time is defined as the time associated with the operations of a tape switch, including tape rewind time, tape unmount time, tape mount time and a possible time waiting for the robotic arm when it is performing tape switches on other drives. Considering all the parallel operations of all involved tape drives, it is convincing to use the transfer time and the seek time of the drive, which finishes serving a request at the last place, as the transfer time and the seek time for the request time respectively. Accordingly we compute the tape switch time as the time difference between the request response time and the data seek-and-transfer time. Depending on the actual object placement for the request, a drive may be involved in multiple object seeks and transfers in serving just one request.

Simulation Results *Effect of the number of switching tape drives m*: As shown in Figure 5, when m changes from 1 to 2, there is a jump in the effective data retrieval bandwidth (for convenience we call it effective bandwidth or just bandwidth). As discussed in Section 5, the contention for tape drive and the waiting time for tape switch are most likely very high when m = 1. A maximum bandwidth exists when m is changing from 2 to 4. The actual highest point depends on the value of α. After m goes beyond 4, the bandwidth decreases. When the number of switch drives increases, the always-mounted tape capacity decreases. The decrease of the always-mounted tape capacity causes more objects to be served by offline tapes and increases the contention for the robot and the number of tape switches. There is a trade-off between the contention for the tape drive and the contention for the robot. As expected, the bandwidth usually increases when α increases, since always-mounted tapes can accumulate more probabilities and serve more requests with a more skewed popularity distribution. In the rest of the simulation, the value of m is set to 4.

Effect of request popularity skewing factor α: As shown in Figure 6, with an average request size of around 213 GB, a more skewed request popularity distribution favors our proposed scheme, *parallel batch placement*, and *object probability placement*. However, a more skewed request popularity distribution does not have a big impact on the effective data retrieval bandwidth from *cluster probability placement*. When α increases, less number of

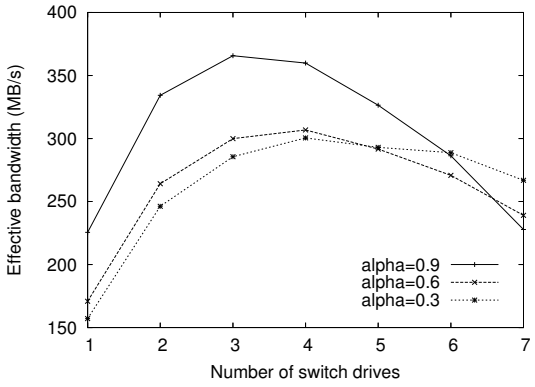

Figure 5. Bandwidth vs. number of drives used for tape switch

tapes can accumulate larger probability and serve more requests resulting less number of tape switches in general, which favors *parallel batch placement* and *object probability placement*. Our proposed scheme, *parallel batch placement*, always outperforms the other two. In the following simulation, α is set to 0.3, reflecting the fact that data are not equally popular, which does not favor our scheme.

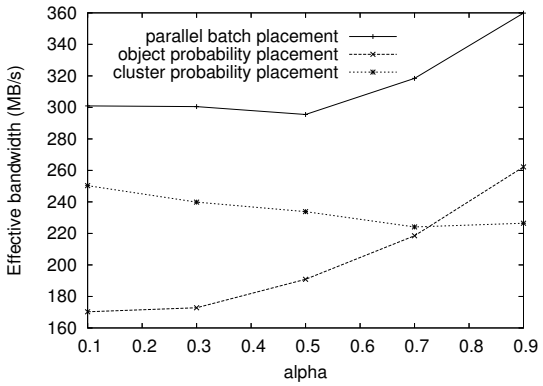

Figure 6. Bandwidth vs. alpha (α)

Effect of request size: Figure 7 shows that the bandwidth increases but not dramatically when the average request size increases. During this simulation, the request size is changed by changing the object size. Thus, the data transfer accounts for more percentage of the total response time while the time for tape switch and data seek may not be changed. Our proposed scheme still does a better job than the other two schemes within the tested range. For an extreme test case, we reduce the object size lower bound to such a point that the $n \times d$ tapes can hold all the objects and are kept mounted all the time. Thus, object requests are all served by mounted tapes and the request response time does not contain any switch time. We found *object probability placement* has the lowest response time with a lowest seek time while *cluster probability placement* and *parallel batch placement* happen to have a similar response time. But data transfer time of *cluster probability placement* accounts for 62% of the toatal response time while data transfer time of *parallel batch placement* only accounts for 19%. This finding agrees with our expectation from the design goals of each scheme.

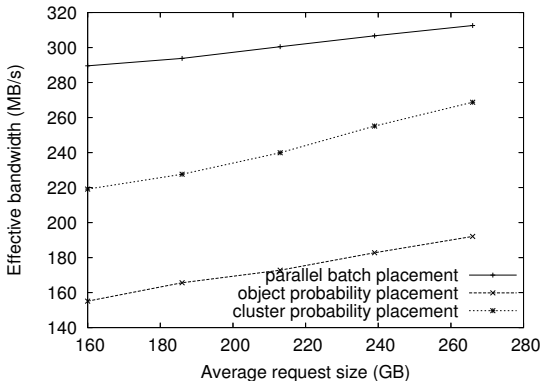

Figure 7. Bandwidth vs. average request size

Scheme scalability comparison: When the average request size is about 240 GB, we get the read performance of the three schemes as shown in Figure 8. When the number of tape libraries increases, both *parallel batch placement* and *object probability placement* scale well while *cluster probability placement* does not scale since it does not have any data transfer parallelism. However, when the number of tape libraries increases from 1 to 3, the data retrieval bandwidth from *cluster probability placement* increases reflecting the reduced robot contention. It is clear that our scheme consistently provides the best performance out of the three schemes.

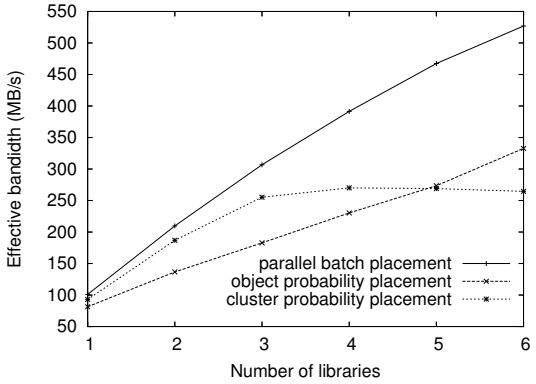

Figure 8. Bandwidth vs. number of tape libraries

Request response time components: Figure 9 compares the three schemes in terms of average tape switch time, average data seek time and average request transfer time when the average request size is about 160GB. It shows

that *object probability placement* takes the longest switch time since it does not take object relationship into consideration and requires more tape switches. The average data seek time from three schemes does not play a big role here compared to the tape switch time. The *object probability placement* does the best in data tranfer time. However, its tape switch time is the worst and dominates the response time. When the average request size changes, three schemes show a similar trend as shown in Figure 9. All the simulation results show that our proposed scheme achieves a better trade-off among tape switch time, data seek time and data transfer time than the other two, and has the best response time most of the time.

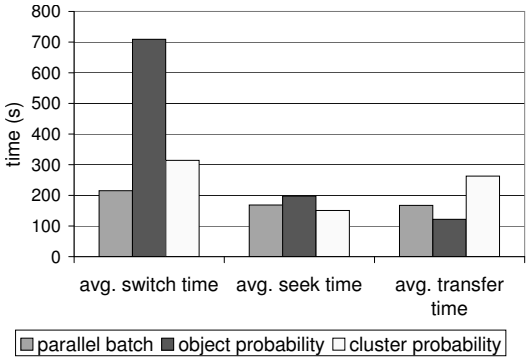

Figure 9. Response time component comparison

Due to page limitations, we will not show the performance of different schemes when tape library technology improves, e.g., increased data transfer speed and tape capacity. In general, our scheme improves more than the other two schemes for these cases. We have varied the total number of objects, the number of pre-defined requests and the number of simulated requests, and found they do not change the relative performance of the three schemes.

7. Conclusions and future work

Reducing tape request response time is a complex research issue and involves many conflicting factors. In this paper we proposed a novel object placement scheme combined with a special tape switch strategy to increase tape switch parallelism and data transfer parallelism among multiple tape libraries. Simulation results show that our proposed scheme achieves the design goal and has a better throughput than two previously proposed schemes [11, 20].

In a real system, objects are moved to tapes periodically. When we place objects on tapes, we only have the local knowledge of object probability and relationship. How to make an optimal or near-optimal solution for the long-term backup/retrieve operations remains to be solved.

References

[1] Product specifications, International Business Machines Corporation (IBM). http://www.ibm.com.

[2] Product specifications, Storage Technology Corporation (StorageTek). http://www.storagetek.com.

[3] Lustre: A scalable, high-performance file system. Whitepaper, Cluster File System, Inc. http://www.lustre.org.

[4] Sgs file system rfp. Technical report, DOE NNCA and DOD NSA, April 2001.

[5] Data protection - backup has never been simpler. Whitepaper, HP, July 2005.

[6] S. Berson, S. Ghandeharizadeh, R. Muntz, and X. Ju. Staggered striping in multimedia information systems. *SIGMOD Rec.*, 23(2):79–90, 1994.

[7] L. T. Chen, R. Drach, M. Keating, S. Louis, D. Rotem, and A. Shoshani. Optimizing tertiary storage organization and access for spatio-temporal datasets. In *NASA Goddard Conference on Mass Storage Systems*, March 1995.

[8] P. M. Chen and D. A. Patterson. Maximizing performance in a striped disk array. In *Proceedings of the 17th annual international symposium on Computer Architecture*, pages 322–331. ACM Press, 1990.

[9] A. L. Chervenak, D. A. Patterson, and R. H. Katz. Storage systems for movies-on-demand video servers. In *Proceedings of the 14th IEEE Symposium on Mass Storage Systems*, page 246. IEEE Computer Society, 1995.

[10] T. Chiueh. Performance optimization for parallel tape arrays. In *Proceedings of the 1995 International Conference on Supercomputing*, pages 385–394, July 1995.

[11] S. Christodoulakis, P. Triantafillou, and F. Zioga. Principles of optimally placing data in tertiary storage libraries. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 236–245. Morgan Kaufmann Publishers Inc., 1997.

[12] J. N. Daigle, R. B. Kuehl, and J. D. Langford. Queuing analysis of an optical disk jukebox based office system. *IEEE Trans. on Computers*, 39(6):819–828, 1990.

[13] A. Drapeau and R. Katz. Striped tape arrays. In *12th IEEE Symposium on Mass Storage Systems*, pages 257–265, 1993.

[14] A. L. Drapeau and R. H. Katz. Striping in large tape libraries. In *Proceedings of the 1993 ACM/IEEE conference on Supercomputing*, pages 378–387. ACM Press, 1993.

[15] L. Golubchik, R. R. Muntz, and R. W. Watson. Analysis of striping techniques in robotic storage libraries. In *Proceedings of the 14th IEEE Symposium on Mass Storage Systems*, page 225. IEEE Computer Society, 1995.

[16] E. Horowitz and S. Sahni. *Fundamentals of Computer Algorithms*. Computer Science Press, 1978.

[17] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 2:241–254, 1967.

[18] T. Johnson and E. L. Miller. Performance measurements of tertiary storage devices. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 50–61. Morgan Kaufmann Publishers Inc., 1998.

[19] K. Keeton, A. Drapeau, D. Patterson, and R. H. Katz. Storage alternatives for video service. In *Proceedings of the Thirteenth IEEE Symposium on Mass Storage Systems*, pages 100–105. IEEE Computer Society, 1994.

[20] J. Li and S. Prabhakar. Data placement for tertiary storage. In *10th NASA Goddard Conference on Mass Storage Systems and Technologies/19th IEEE Symposium on Mass Storage Systems*, pages 193–207, April 2002.

[21] D. A. Patterson, G. Gibson, and R. H. Katz. A case for redundant arrays of inexpensive disks (raid). In *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, pages 109–116. ACM Press, 1988.

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

[23] P. Triantafillou and C. Faloutsos. Overlay striping and optimal parallel I/O for modern applications. *Parallel Computing*, 24(1):21–43, 1998.

[24] C. K. Wong. *Algorithmic Studies in Mass Storage Systems*. W. H. Freeman & Co., 1983.