Retrieving Multimedia Objects From Hierarchical Storage Systems

Philip K. C. Tse

Macquarie University,
Marsfield, NSW 2109, Australia.
philip@ics.mq.edu.au
tel +612 9850-9578
fax +61 2 9850-9551

Clement H. C. Leung

Victoria University,
PO Box 14428, Melbourne,
Victoria 8001, Australia.
clement@matilda.vu.edu.au
tel +613 9688-5020
fax: +61 3 9688-4050

Abstract

A multi-level hierarchical storage system (HSS) is able to provide large storage capacity for multimedia data at a more economical cost than disk only systems. The retrieval of multimedia objects from the tertiary storage however imposes substantial performance constraints. As many multimedia applications access objects sequentially following a predetermined order, we have designed a concurrent striping method that store cold objects following this order on the media units. We have carried out simulations to study its performance, and it is observed that this method can improve the system throughput, stream response time, and disk buffer usage.

1. Introduction

Large multimedia systems need to capture, process, store, and maintain a variety of information sources [1]. These applications always include a multimedia database management system whose performance rely on that of the underlying storage system. Although most information systems store all their data on online magnetic disks, the huge amount of multimedia data makes this storage architecture neither practical nor economical. A multi-level hierarchical storage system (HSS) provides sufficient storage capacity at a more economical cost than disk only systems [2-4], but it invariably includes the long access latency of data held in tertiary storage devices deteriorating the performance of the storage system [5, 6].

Multimedia objects can either be staged or pipelined from tertiary storage devices. In the staging method, a stream waits for the entire object to be retrieved before it starts to display. In the pipelining methods, a stream starts to display after the first slice of data is retrieved [7-10]. Pipelining methods reduce stream response time of single stream for low-end tertiary drives, but they cannot be applied to concurrent streams for high-end drives. The traditional striping methods [11-16], the time-slice scheduling [17], and multiple readout algorithm [4] may reduce the stream response time and buffer space, but the system throughput drops rapidly due to extra overheads in exchanging media. Hence, new technique is required for high end tertiary drives serving concurrent streams on HSS.

As many multimedia applications, such as video-on-demand, access objects sequentially following a predetermined order, we have designed a concurrent striping method to store objects following this order to achieve high storage efficiency. We first describe this concurrent striping method and the concurrent streaming in the next Section. Next, we compare the HSS performance using this method and other striping methods in Section 3.

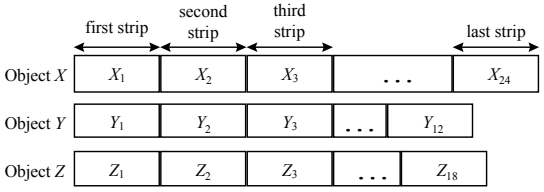

Figure 1. Data Striping

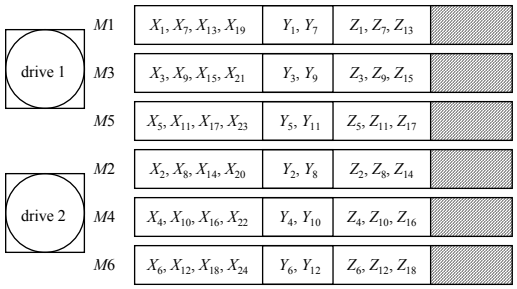

Figure 2. Placement of Data Strips

2. Concurrent Striping Method

A typical HSS consists of memory at the first level, magnetic disks at the secondary level, and robotic storage library at the tertiary level. The robotic storage library is composed of many media units, several independent tertiary drives, and one or more robot arms. Data are stored on the removable and economical media units. When data on a media unit are required, the robot arm exchanges the media unit onto a tertiary drive. Data on the currently loaded media unit become available for access.

2.1. Data Striping and Placement

In the concurrent striping method, each multimedia data object is partitioned into a number of data strips (or segments) of similar length (Figure 1). We assume that each data strip is a logical unit that is displayed for a period of time after the previous one. We divide the media units equally among the tertiary drives so that each media unit is accessed by only one drive to avoid contention of media units by different drives. Then we arrange the media units into a fixed sequence. The data strips are placed into the media units following the predetermined sequence of the media units (Figure 2). One data strip is stored on one media unit until the last media unit is reached. The next data strip is then stored to the first media unit and so on. The data strips are placed within each media unit according to the access frequency of the objects to which the data strip belongs. Each object should have all its data strips placed together so that the sequences of data strips belonging to the same object may be preserved on all media units.

2.2. Concurrent Streams Management

When multimedia objects are accessed, new streams are initiated to access data objects. These streams are placed in a stream queue of the concurrent stream controller. When accepted to display, the concurrent stream controller creates a new stream object which then sends two requests to every tertiary drive and waits. After a drive finishes a request, it sends back an access notification back to the stream object. The stream object then sends the next request to the same drive.

Since the tertiary drives access data independently, an accepted stream starts to display data at the completion of at least one request from each drive. After the first data strip is displayed, the stream checks to see if the next data strip has been retrieved or not. If the next data strip has been retrieved, the stream then displays the next data strip. Otherwise, the stream starves and waits until the next data strip has been retrieved from tertiary storage. After a stream finishes displaying all data strips, a notification is then sent back to the concurrent stream controller.

The SCAN policy is chosen to order the service of requests because it can achieve the highest system throughput. Every stream sends requests to the tertiary drives accessing data in the same fixed sequence of the media units. The tertiary drives serve the requests in groups according to the media unit on which the data strips reside. A group of requests is made up of one request per stream that access data on the same media unit. These requests are hence served together as a group after one media exchange.

3. Storage System Performance

We have compared the object retrieval performance using non-striping, parallel striping, and concurrent striping methods. A number of streams that retrieve MPEG compressed video objects arrive randomly at the HSS. In the non-striping method, all data strips of one object are stored on one media unit until the free space of the media unit is consumed. In the parallel striping method, the data strips spread across several media units according to the number of tertiary drives.

The maximum system throughput in Figure 5 shows that the concurrent striping method always outperforms the parallel striping method and the non-striping method. Thus, it can serve streams quickly to clear off the waiting queues at the arrival of a burst of streams. The parallel striping method cannot achieve as high maximum throughput as the others because of extra overheads in exchanging media at several drives at the same time.

The response time of a new stream significantly affects the quality of service to users. It is composed of the queue waiting time and the start up latency. We have observed that the response time in all methods deteriorates at high stream arrival rates (not shown), but the concurrent striping method deteriorates slowly at a higher stream arrival rate due to its higher maximum throughput.

Disk buffer is needed to store the objects between the time when they are copied from HSS and the time when they are consumed. The disk buffer per stream of the two traditional methods are almost unaffected by the presence of concurrent streams (Figure 6). The total disk buffer consumption hence would increase linearly with more concurrent streams. As more requests are served per media exchange at the presence of more concurrent streams in the concurrent striping method, data strips for each stream are retrieved discontinuously. Hence, the disk buffer size per stream reduces gradually resulting in efficient usage of resources.

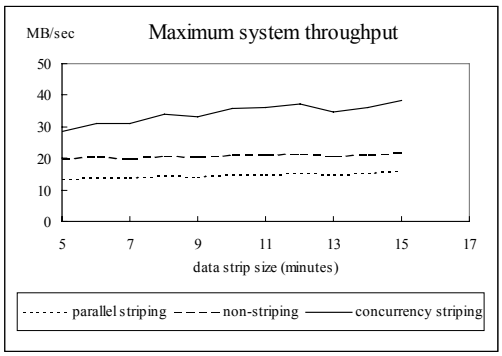

Figure 5. Maximum System Throughput

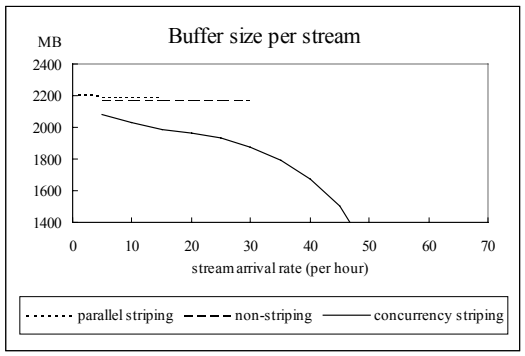

Figure 6. Disk Buffer Size

4. Conclusions

The use of HSS is inevitable in very large multimedia systems. The main concerns in using these systems are their relatively poor response characteristics and large resource consumption. We have described a concurrent striping method that makes use of the predetermined sequence in retrieving multimedia objects from HSS. We have shown that this method has several advantages. Firstly, its system throughput exceeds that of existing methods. It can serve streams quickly to clear off the waiting queues. Secondly, it uses less disk buffer space and handles more streams to reduce the resource contention on disk space. Thirdly, the response time of new streams are less deteriorated under heavy load conditions. These advantages make the concurrent striping method the most suitable approach for storage of multimedia objects from HSS.

5. References

[1] C.H.C. Leung (Ed.), *Visual Information Systems*, LNCS 1304, Springer-Verlag, 1997.

[2] M.G. Kienzle *et al.*, "Using Tertiary Storage in Video-on-Demand Servers," *IEEE COMPCON*, (1995) 225-233.

[3] M.H. Lee *et al.*, "Storage Hierarchy Design in Video-On-Demand Servers," *SPIE Storage and Retrieval for still image and video databases IV* (1996) 2670:300-307.

[4] H. Suzuki *et al.*, "Storage hierarchy for video-on-demand systems," *SPIE Storage and Retrieval for Image and Video Databases II* (1994) 2185:198-207.

[5] A.L. Chervenak *et al.*, "Storage systems for movies-on-demand video servers," *IEEE Sym on MSS* (1995) 246-256.

[6] Y.N. Doganata and A.N. Tantawi, "A cost/performance study of video servers with hierarchical storage," *IEEE MCS* (1994) 393-402.

[7] S. Ghandeharizadeh *et al.*, "On multimedia repositories, personal computers, and hierarchical storage systems," *ACM Multimedia* (1994) 407-416.

[8] W. Tavanapong *et al.*, "A framework for supporting previewing and VCR operations in a low bandwidth environment," *ACM Multimedia* (1997) 303-312.

[9] P.K.C. Tse and C.H.C. Leung, "A low latency hierarchical storage system for Multimedia Data," *IAPR Intl Workshop MINAR* (1998) 181-194.

[10] J.Z. Wang *et al.*, "SEP: a space efficient technique for managing disk buffers in multimedia servers," *IEEE MCS* (1996) 598-607.

[11] T.C. Chiueh, "Performance Optimization for Parallel Tape Arrays," *ACM Supercomputing* (1995) 375-384.
[12] S. Christodoulakis *et al.*, "Principles of Optimally Placing Data in Tertiary Storage Libraries," *VLDB Conf.* (1997).
[13] A.L. Drapeau and R.H. Katz, "Striped Tape Arrays," *IEEE Sym on MSS* (1993) 257-265.
[14] A.L. Drapeau and R.H. Katz, "Striping in Large Tape Libraries," *ACM Supercomputing* (1993) 378-387.
[15] L. Golubchik *et al.*, "Analysis of Striping Techniques in Robotic Storage Libraries," *IEEE Sym on MSS* (1995) 225-238.
[16] P. Triantafillou and T. Papadakis, "On-Demand Data Elevation in a Hierarchical Multimedia Storage Server," *VLDB Conf.* (1997) 1-10.
[17] S.W. Lau *et al.*, "A Cost-effective Near-line Storage Server for Multimedia System," *IEEE Data Engineering*, 449-456, 1995.