Multimedia Storage Servers

Prashant J. Shenoy and Harrick M. Vin

Rapid advances in computing, communication, and compression technologies coupled with the dramatic growth of the Internet has led to the emergence of a wide variety of multimedia applications—such as distance learning, interactive multi-player games, online virtual worlds, and scientific visualization of multi-resolution imagery. These applications differ from conventional applications in at least two ways. First, they involve storage, transmission, and processing of heterogeneous data types—such as text, imagery, audio, and video—that differ significantly in their characteristics (e.g., size, data rate, real-time requirements, etc.). Second, unlike conventional best-effort applications, these applications impose diverse performance requirements—for instance, with respect to timeliness—on the networks and operating systems. Due to these differences, techniques employed by conventional file systems for managing textual files do not suffice for managing multimedia objects. In this chapter, we discuss storage and retrieval techniques employed by multimedia storage servers.

Storage Techniques

Digitization of audio yields a sequence of samples and that of video yields a sequence of frames. A *media stream* refers to a continuously recorded sequence of audio samples or video frames. Due to the large sizes and data transfer rates of media streams, multimedia servers are founded on *disk arrays*. To effectively utilize a disk array, multimedia servers *stripe* each media stream across disks in the array. The striping unit, also referred to as a *media block*, denotes the maximum amount of logically contiguous data stored on a single disk. Each media block may contain either a fixed number of media units (e.g., video frames or audio samples), or a fixed number of storage units (e.g., bytes). If a media stream is compressed using a variable bit rate (VBR) compression algorithm, then the storage space requirement may vary from one media unit to another. Hence, a server that constructs a media block using a fixed number of media units will be required to store variable size media blocks on the array. On the other hand, if media blocks are of fixed size, then they will contain a variable number of media units. Thus, depending on the placement policy, accessing a fixed number of media units for a stream will require the server to retrieve either a fixed number of variable-size blocks, or a variable number of fixed-size blocks. Due to the sequentiality of media stream playback, the variable-size block placement policy yields predictable access patterns for the disk array, and thereby simplifies disk bandwidth management. This, however, comes at the expense of increased complexity of storage space management. The fixed-size block placement policy simplifies storage space management at the expense of more complex disk bandwidth management algorithms. Hence, the variable-size blocks are suitable for predominantly read-only environments (e.g., video-on-demand (VOD) servers), while fixed-size blocks are better suited for environments that involve frequent creation, deletion, and modification of media stream (e.g., multimedia file systems).

To maximize the throughput of a disk array, load on disks should be balanced. Multimedia servers attempt to balance the load across the disks by using a combination of static and dynamic load balancing techniques.

- *Static load balancing*: A multimedia server attempts to balance load across disks by selecting for each media stream an appropriate (1) stripe unit size, (2) degree of striping, and (3) the amount of replication.

Conventional file systems select stripe unit sizes that minimize the *average* response time while maximizing throughput. In contrast, to decrease the frequency of playback discontinuities, a multimedia server should select a stripe unit size that minimizes the *variance* in response time while maximizing throughput. Whereas small stripe units result in a uniform load distribution among disks in the array (and thereby decrease the variance in response times), they also increase the overhead of disk seeks and rotational latencies (and thereby decrease throughput). Large stripe units, on the other hand, increase the array throughput at the expense of increased load imbalance and variance in response times. To maximize the number of clients that can be serviced simultaneously, the server should select a stripe unit size that balances these tradeoffs.

The degree of striping for media streams is dependent on the number of disks in the array. In relatively small disk arrays, striping media streams across all disks in the array (i.e., wide-striping) yields a balanced load and maximizes throughput. For large disk arrays, however, to maximize the throughput, the server may need to stripe media streams across subsets of disks in the array, and replicate their storage to achieve load balancing.

The amount of replication for each media stream depends on the popularity of the stream and the total storage space constraints.

- *Dynamic load balancing*: Load across disks within a multimedia server may become unbalanced, at least transiently, because of the arrival pattern of requests. To smooth-out this load imbalance, multimedia servers employ dynamic load balancing techniques. If multiple replicas of the requested media stream are stored on the array, then the server can attempt to balance load across disks by servicing the request from the least loaded disk containing a replica. Further, the server can exploit the sequentiality of audio and video retrieval to prefetch data for media streams to smooth-out variation in the load imposed by an individual media stream.

Fault-tolerance is another issue that arises with increase in number of disks in a multimedia server. Multimedia servers should provide mechanisms to recover from a disk failure without losing data or taking the system off-line. Conventional disk arrays either replicate data on separate disks, or use error-correcting codes (e.g., parity encoding) to recover from a disk failure. These approaches, however, can significantly increase the load on the surviving disks in the event of a disk failure, resulting in deadline violations in the playback of media streams. To prevent such a scenario, with conventional fault-tolerance techniques, servers must operate at low levels of disk utilization during the fault-free state. Multimedia servers can reduce this overhead by exploiting the characteristics of media streams. In particular, there are two general techniques that a multimedia server may use. First, a multimedia server can exploit the sequentiality of audio and video access to reduce the overhead of on-line recovery in a disk array. Specifically, by computing parity information over a sequence of blocks belonging to the same media stream, the server can ensure that media blocks retrieved for recovering a block stored on the failed disk would be requested by the client in the near future. By buffering such blocks and then servicing the requests for their access from the buffer, this method minimizes the overhead of the on-line failure recovery process. Second, since human perception is tolerant to minor distortions in media playback, a multimedia server can reduce the overhead of failure recovery by approximating data stored on the failed disk using spatial and temporal redundancies inherent in media streams. This method integrates the media decoding process with failure recovery, and hence distributes the task of failure recovery among client sites.

Retrieval Techniques

A multimedia server needs to choose from two fundamentally different paradigms to schedule retrieval of media streams: *server-push* or *client-pull*. Due to the periodic nature of media playback, a multimedia server using the server-push architecture can service multiple streams by proceeding in *rounds*. During each round, the server retrieves a fixed number of media units for each stream. To ensure continuous playback, the number of media units accessed for each stream must be sufficient to sustain its playback rate, and the service time (i.e., the total time spent in retrieving media units during a round) should not exceed the duration of a round. In the client-pull architecture, on the other hand, the server retrieves media units for a client only in response to an explicit read request. While the server needs to maintain client states in the server-push mode, the client-pull mode is inherently stateless. The primary advantage of server-push architecture is that they optimize the array utilization by batching read/write operations. Furthermore, it is easier for such servers to enforce quality of service guarantees provided to clients.

In both of these architectures, a multimedia server employs admission control algorithms to ensure that the resources required by a new request do not affect the real-time requirements of streams already being serviced. An admission control algorithm determines the resource requirements of a request by estimating its bit rate and the disk access times. Depending upon the nature of this estimate, admission control algorithms can be classified into three categories:

- *Deterministic* admission control algorithms make worst-case estimates of the bit rate and disk access times of user requests, and are used when clients can not tolerate any deadline violations.

- *Statistical* admission control algorithms use probability distributions to estimate the bit rate and disk access time variations of user requests to guarantee that deadlines will be met with a certain probability. Such algorithms achieve much higher utilization than deterministic algorithms, and are used when clients can tolerate some, but only a bounded number of, deadline violations.

- *Measurement-based* admission control use past variations in bit rate and disk access times of media streams as an indicator of future variations. They achieve the highest disk utilization but provide the weakest guarantees.

In addition to admission control algorithms, a multimedia server needs to employ a disk scheduling algorithm that can meet the performance requirements of multimedia applications. Typically, disk requests issued by multimedia servers have deadlines. Conventional disk scheduling algorithms—such as SCAN and Shortest Access Time First (SATF)—schedule requests based on their physical location on disk (so as to reduce disk seek and rotational latency overheads) and ignore other requirements such as deadlines. To service requests with deadlines, several disk scheduling algorithms have been proposed. The simplest of these scheduling algorithms is Earliest Deadline First (EDF). EDF schedules requests in the order of their deadlines but ignores the relative positions of requested data on disk. Hence, it can incur significant seek time and rotational latency overhead. This limitation has been addressed by several disk scheduling algorithms, including Priority SCAN (PSCAN), Earliest Deadline SCAN, Feasible Deadline SCAN (FD-SCAN), SCAN-EDF, and Shortest Seek Earliest Deadline by Order/Value (SSEDO, SSEDV). These algorithms start from an EDF schedule and reorder requests without violating their deadlines such that the seek time and rotational latency overhead is reduced.

To maximize the number of clients that can be serviced simultaneously, multimedia servers also rely on *caching* and *batching* techniques.

- Caching involves using the buffer space at the server to store recently accessed data and subsequently using the cached data to service new requests. Traditional caching techniques based on the *least recently used (LRU)* policy yield poor hit ratios for sequential data accesses. In fact, because of the periodic and sequential nature of media streams, the buffer allocated to a block can be reassigned immediately after use. However, if another user is accessing the same data, within a short time interval, the block can be retained in the buffer for the later user. Algorithms such as *Interval Caching* and *Distance Caching* exploit this observation; they cache *intervals* formed by consecutive accesses to the same media stream. The two requests that form such a consecutive pair are called the *writer* and the *reader*; the writer causes media data to be written into the cache while the reader reads the cached data.

- Batching delays a new user request, for a duration referred to as the *batching interval*, with the expectation that more requests for the same media stream may arrive within that duration. If multiple requests are indeed received for the same media stream within the batching interval, then the server can service all the requests by retrieving a single stream from the disks. The longer the batching interval, the larger is the probability of receiving multiple requests for the same media stream within the interval, and hence the larger is the gain due to batching. However, the larger the batching interval, the larger is the startup latency for requests. Batching policies attempt to find a balance between this tradeoff. One approach for reducing the startup latency is called *piggybacking* (or *catching*). In this technique, requests are serviced immediately upon their arrival. However, if the server began servicing a request for the same media stream sometime in the recent past, then the server attempts to service the two requests in a manner such that the servicing of the new request catches up with that of the previous request. If this is successful, the server then services both of these requests as a single stream from the disks.

Finally, a user may access a media stream in fast-forward or rewind modes. Hence, a multimedia server is required to support these VCR control operations. The main challenge in supporting VCR control operations is that fast-forward or rewind operations generally impose higher disk bandwidth requirements than normal retrieval. Hence, if a multimedia server performs admission control based on the bandwidth requirement for normal playback, then on receiving requests for fast-forward or rewind, the server may not be able to meet the real-time performance requirements of clients. To address this issue, a server may employ techniques that range from degrading the quality of media streams (possibly by using multi-resolution encoding of media streams) while maintaining roughly the same bandwidth requirement during VCR control operations to optimistically reserving some fraction of the total disk bandwidth for servicing such VCR control requests.

Case Studies

Over the past several years, several academic and commercial institutions have developed multimedia servers. Commercial multimedia servers range from low-end PC based multimedia servers designed to serve small work groups to high-end large scale servers that can serve thousands of video-on-demand users.

The low-end servers are targeted for a local area network environment and their clients are personal computers, equipped with video-processing hardware, connected on a LAN. They are designed for applications—such as on-site training and information kiosks—and the multimedia files consist of short audio and video clips. They are also used as small-scale video servers for environments such as hotels and conference centers.

High end servers are targeted for applications such as video-on-demand, in which the number of simultaneous streams is expected to be in the 1000s, and the distribution system is expected to be cable based, or telephone-wire based. In order to provide a large collection of videos in a cost-effective solution, these servers generally are based on a hierarchy of storage devices, ranging from high-cost, high-bandwidth semiconductor memory through disk storage to low-cost, high-capacity tape/CD/DVD libraries.

Concluding Remarks

Multimedia storage servers differ from conventional storage servers to the extent that significant changes in design must be effected. These changes are wide in scope, influencing everything from the selection of storage hardware to the choice of disk scheduling algorithms. This chapter provides an introduction to the problems involved in multimedia storage server design and to the various approaches towards solving these problems. The chapter consist of six recent articles. The first article titled “Multimedia Storage Servers: A Tutorial” provides an overview of the architectures and algorithms required for designing multimedia servers. The second article titled "Random RAIDs with Selective Exploitation of Redundancy for High Performance Video Servers" deals with placement, fault-tolerance and performance issues. The article shows how redundant information stored by a multimedia server for fault-tolerance can also be employed for improving response times in the absence of faults. Specifically, the proposed technique treats an overloaded disk on the server akin to a failed disk and uses redundant information stored on the remaining disks to reconstruct data requested from the overloaded disk (which reduces the load on this disk and improves response times).

The next two articles deal with retrieval issues. The first of these two articles is titled "Disk Scheduling in a Multimedia I/O System" and discusses a scheduling algorithm that takes deadlines of real-time disk requests into account while optimizing disk seek overheads. This is achieved by combining the earliest deadline first (EDF) algorithm with the SCAN disk scheduling algorithm; the resulting algorithm is referred to as SCAN-EDF. The second article titled "A Statistical Admission Control Algorithm for Multimedia Servers" deals with admission control issues. It shows how probability distributions of the client load and disk service times can be employed to provide statistical performance guarantees to clients in a multimedia server.

The final two articles deal with techniques for maximizing the number of clients that can be supported by a multimedia server. The first of these articles is titled "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Environments" and discusses techniques for efficiently utilizing the buffer cache at the server. The final article titled "On optimal piggyback merging policies for video-on-demand systems" discusses batching policies for improving the number of clients supported by the server.

We conclude with a bibliography of other recent articles in this area for readers interested in further study.

- **Case Studies**

  - D. Anderson, Y. Osawa and R. Govindan, "A File System for Continuous Media," *ACM Transactions on Computer Systems*, 10(4):311-337, April 1994.
  - W. Bolosky et. al., "The Tiger Video File Server", *Proceedings of the Sixth International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'96)*, pages 97–104, April 1996.
  - M. Buddhikot, G. Parulkar and J. Cox, "Design of a Large Scale Multimedia Storage Server", *Journal of Computer Networks and ISDN Systems*, pages 504–524, December 1994.
  - R. Haskin, "Tiger Shark–A Scalable File System for Multimedia", *IBM Journal of Research and Development*, 42(2):185-197, March, 1998.
  - C. Martin, P. Narayan, B. Ozden, R. Rastogi, and A. Silberschatz, "The Fellini Multimedia Storage Server," *Multimedia Information Storage and Management,* Editor – S. M. Chung, Kluwer Academic Publishers , August 1996.
  - D. Pegler, D. Hutchison, P. Lougher and D. Shepherd, "A Scalable Architecture for Multimedia Storage", *High-Speed Networking for Multimedia Applications*, W. Effelsberg, O. Spaniol, A. Danthine, D. Ferrari (eds), Kluwer Academic Publishers, Ch. 16, pp. 127-263, 1996.
  - P J. Shenoy, P. Goyal, S S. Rao and H M. Vin, "Symphony: An Integrated Multimedia File System", *Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking (MMCN'98), San Jose, CA*, pages 124-138, January, 1998.

- **Retrieval Issues**

  - P. Barham, "A Fresh Approach to File System Quality of Service", *Proceedings of the Seventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97), St. Louis, MO*, pages 119–128, May 1997.

- E. Chang and A. Zakhor, “Cost Analyses for VBR Video Servers”, *Proceedings of Multimedia Computing and Networking (MMCN) Conference*, pages 381–397, January 1996.
- A. Dan, D. Sitaram and P. Shahabuddin, “Scheduling Policies for an On-Demand Video Server with Batching”, *Proceedings of the Second ACM International Conference on Multimedia, San Francisco, CA*, pages 15-23, October 1994.
- S. Sahu, Z Zhang, J. Kurose, and D. Towsley, “On the Efficient Retrieval of Variable Bit Rate Video in a Multimedia Server”, *Proc. of IEEE Conference on Multimedia Computing Systems (ICMCS)*, June 1997, pages 46-53, Ottawa, June 1997.
- P Shenoy and H M. Vin, “Cello: A Disk Scheduling Framework for Next Generation Operating Systems”, *Proceedings of ACM SIGMETRICS Conference, Madison, WI*, pages 44-55, June 1998.

- **Placement Issues**
  - S. Berson, S. Ghandeharizadeh, R. Muntz and X. Ju, “Staggered Striping in Multimedia Information Systems”, *Proceedings of the ACM SIGMOD conference*, 1994.
  - T. Chiueh and R. Katz, “Multi-Resolution Video Representation for Parallel Disk Arrays”, *Proceedings of the ACM Multimedia’93, Anaheim, CA.*, pages 401-409, August 1993.
  - J R. Santos, R. Muntz and B. Rabeiro-Nato, “Comparing Random Data Allocation and Data Striping in Multimedia Servers”, *Proceedings of ACM Sigmetrics 2000, Santa Clara, CA*, pages 44-55, June 2000.
  - R. Tewari, D. Dias, R. Mukherjee and H. Vin, “Design and Performance Tradeoffs in Clustered Multimedia Servers”, *Proceedings of IEEE Conference on Multimedia Computing Systems (ICMCS), Hiroshima, Japan*, June 1996.
  - R. Tewari, R. King, D. Kandlur and D. Dias “Placement of Multimedia Blocks on Zoned Disks”, *Proceedings of the Multimedia Computing and Networking (MMCN) Conference, San Jose, CA*, January 1996.
  - P J. Shenoy and H M. Vin, “Efficient Striping Techniques for Multimedia File Servers”, *Proceedings of the Seventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV’97), St. Loius, MO*, pages 25-36, May 1997.

- **Fault Tolerance**
  - S. Berson, L. Golubchik and R. Muntz, “Fault Tolerant Design of Multimedia Servers”, *Proceedings of SIGMOD Conference*, pages 364-375, 1995.
  - P. Shenoy and H M. Vin. “Failure Recovery Algorithms for Multimedia Servers”, *ACM/Springer Multimedia Systems Journal*, 8(1) pages 1-19, January 2000

- **Miscellaneous Issues**
  - A L. Drapeau and R H. Katz, “Striping in Large Tape Libraries”, *Proceedings of Supercomputing’93*, November 1993.
  - W. Feng, B. Krishnaswami, A. Prabhudev, “Proactive Buffer Management for the Delivery of Stored Video Across Best-Effort Networks”, *Proceedings of the ACM Conference on Multimedia, Bristol, UK*, pages 285-290, September 1998.
  - L. Golubchik, J. C. S. Lui and R. R. Muntz, “Reducing I/O Demand in Video-On-Demand Storage Servers”, *Proceedings of SIGMETRICS ’95, Ottawa, Canada*, May 1995.

- B. Ozden, R. Rastogi and A. Silberschatz, "Buffer Replacement Algorithms for Multimedia Storage Systems", *Proceedings of the IEEE International Conference on Multimedia Computing Systems (ICMCS)*, pages 172–180, June 1996.
- J. Rexford, S. Sen, J. Dey, W. Feng, J. Kurose, J. Stankovic and D. Towsley, "Online Smoothing of Live, Variable-Bit-Rate Video", *Proceedings of the Seventh International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '97), St. Louis, MO*, pages 249-258, May 1997.
- R. Tewari, H. Vin, A. Dan and D. Sitaram, "Resource Based Caching for Web Servers" , *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN), San Jose, CA*, January 1998.
- J. Salehi, Z. Zhang, J. Kurose and D. Towsley, "Supporting Stored Video: Reducing Rate Variability and End-to-End Resource Requirements through Optimal Smoothing", *Proceedings of ACM SIGMETRICS, Philadelphia, PA*, pages 222-231, May 1996.
- P. Shenoy and P. Goyal and H M. Vin, "Architectural Considerations for Next Generation File Systems", *Proceedings of the Seventh ACM Multimedia Conference, Orlando,FL*, November 1999.
- D. Venkatesh and T.D.C. Little, "The Use of Media Characteristics and User Behavior for the Design of Multimedia Servers", *Multimedia Information Storage and Management*, S.M. Chung, ed., Kluwer Academic Publishers, pages 95-116, 1996