STREAMING SERVER

Inventors: Stergos V. Anastasiadis, 3611 University Dr. Apt. 3Y, Durham, NC (US) 27707; Kenneth C. Sevcik, 99 Harbour Square, Suite 4001, Toronto, Ontario (CA), M5J 2H2; Michael Stumm, 3 Belvole, Toronto (CA), H5X 2A6

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1049 days.

Appl. No.: 10/054,699
Filed: Jan. 22, 2002

Prior Publication Data
US 2003/0074486 A1 Apr. 17, 2003

Foreign Application Priority Data
Jan. 19, 2001 (CA) .................................................. 2331474
Feb. 9, 2001 (CA) .................................................. 2335521
Feb. 9, 2001 (CA) .................................................. 2335540

Int. Cl.
G06F 17/30 (2006.01)

U.S. Cl. .................................................. 707/7; 707/1

Field of Classification Search ....................... 707/1, 707/2, 7, 10, 104.1; 709/203, 219, 221, 226, 709/231; 711/114; 714/4, 5, 11; 719/321
See application file for complete search history.

References Cited
U.S. PATENT DOCUMENTS
5,974,503 A 10/1999 Venkatesh et al. .......... 711/114
6,625,750 B1 9/2003 Duso et al. ............... 714/11

OTHER PUBLICATIONS
Shenoy and Vin, Failure Recovery Algorithms for Multimedia Servers, University of Texas at Austin, pp. 1–34 (undated).
Bolosky, et al., Distributed Schedule Management in the Tiger Video Fileserver, Microsoft Research, SOSIP 97 (undated).
Shenoy and Vin, Efficient Stripping Techniques for Multimedia File Servers, University of Texas at Austin, NOSSDAV 97, pp. 25–36 (undated).
Reddy and Wijayaratne, Techniques for improving the throughput of VBR streams, Texas A&M University, NCN 99 (undated).

(Continued)

Primary Examiner—Apu Mofiz
Attorney, Agent or Firm—Connolly Bove Lodge & Hutz LLP

ABSTRACT
A media server having at least one of a stride-based storage device space allocation scheme, stride-based method of stripping data across multiple storage devices for continuous media streaming, server-based smoothing of variable bit-rate streams, distributed architecture, and fault tolerance.

18 Claims, 20 Drawing Sheets
OTHER PUBLICATIONS


Zhao and Tripathi, Bandwidth-Efficient Continuous Media Streaming Through Optimal Multiplexing (undated).


Biersack and Hamdi, Cost-optimal Data Retrieval for Video Servers with Variable Bit Rate Video Streams, NOSSDAV 98 (Cambridge, UK) (undated).


Gray and Shenoy, Rules of Thumb in Data Engineering, IEEE International Conference on Data Engineering, 2000.


Patterson, et al., A Case for Redundant Arrays of Inexpensive Disks (RAID), University of California (undated).


* cited by examiner
Figure 1
Figure 2

stream stride index

disk strides i i+1 i+2 i+3 i+4 i+5

[Symbols representing requests]

Request j  Request j+1  Request j+2  Request j+3
Figure 4(a)
Figure 4(b)
Group-Grain (Disk 0)

Bytes (x 1E6)

Round

Group-Grain (Disk 1)

Bytes (x 1E6)

Round

Figure 4(c)
Fixed-Grain Striping

![Graph showing number of streams vs load with different Fl values](image)

- Fl = 30
- Fl = 10
- Fl = 1

Figure 5
Figure 6

Fixed-Grain Stripping

Load

Rejected/Accepted

- - F_l = 1
- - F_l = 10
- - - F_l = 30
Fixed-Grain Striping

Number of Streams

Load = 80%
Load = 40%

Block Bf (Bytes x 1000)

Figure 7
Fixed-Grain Striping

- Max Rsrv
- Avg Rsrv
- Avg Diff

Round Disk Time (sec)

Block Bf (Bytes)

Figure 8
Fixed-Grain Striping

- Max Rsrv
- Avg Rsrv
- Avg Diff

Round Disk Time (sec)

Number of Disks

Figure 9
Fixed-Grain vs Variable-Grain Striping

- • Variable-Grain (80%)
- • Fixed-Grain (80%)
- • Variable-Grain (40%)
- • Fixed-Grain (40%)

Number of Streams

Number of Disks

Figure 10
Variable-Grain Striping

Number of Streams

Load

Figure 11
Variable-Grain Striping

Figure 12
Variable-Grain Striping

Max Rsrv  Avg Rsrv  Avg Diff

Round Disk Time (sec)

0.0  0.2  0.4  0.6  0.8  1.0

8  16  32

Number of Disks

Figure 13
Figure 14
Simulated Disk Mode

- **Max Rsrv**
- **Avg Rsrv**
- **Max Meas**
- **Avg Meas**

Round Disk Time (sec)

- Fixed-Grain
- Variable-Grain

2 Disks

Figure 15
STREAMING SERVER

CROSS-REFERENCE TO RELATED APPLICATION(S)

This application is related to and claims priority to Canadian patent application entitled STRIDE-BASED DISK SPACE ALLOCATION SCHEME having serial number 2,331,474, by Stergios V. Anastasiadis, filed Jan. 19, 2001; to Canadian patent application entitled DISTRIBUTED MEDIA SERVER ARCHITECTURE having serial number 2,335,521, by Stergios V. Anastasiadis, filed Feb. 9, 2001; and to Canadian patent application entitled SERVER BASED SMOOTHING OF VARIABLE BIT RATE STREAMS having serial number 2,335,540, by Stergios V. Anastasiadis, filed Feb. 9, 2001.

The entire teachings of the above Canadian patent applications are further incorporated herein by reference.

OTHER REFERENCES

Stergios V. Anastasiadis, Supporting Variable Bit-Rate Streams in a Scalable Continuous Media Server, PhD Dissertation, University of Toronto, public on January 2002.


Stergios V. Anastasiadis, Kenneth C. Sevcik, Michael Stumm, Server-Based Smoothing of Variable Bit-Rate Streams, ACM Multimedia Conference, Ottawa, ON, October 2001 at 147.

The entire teachings of the above references are further incorporated herein by reference.

FIELD OF INVENTION

This invention relates to network servers and, in particular, a streaming server that supports variable bit-rate streams and has at least one of a multitude of storage devices with allocation scheme, stride-based method of stripping data across multiple storage devices, distributed architecture, fault tolerant operation, and server-based smoothing.

BACKGROUND OF THE INVENTION

Spatial and temporal compression have made practical the storage and transfer of digital video streams with acceptable quality. Standardization (for example, through the MPEG specifications) has facilitated widespread distribution and use of compressed video content in a range of applications from studio post-production editing to home entertainment (e.g., Digital Versatile Disks). Although media streams can optionally be encoded at a constant bit rate, it has been shown that equally acceptable quality can be achieved using variable bit-rate encoding with average bit rates reduced by 40%.

As the installed network bandwidth increases, scalable network servers are becoming the dominating bottleneck in the widespread deployment of broadband services. Therefore, the potential scalability of network servers that support variable bit-rate media streams is becoming a fundamental problem.

System design complications coupled with excessive expectations from technological progress previously discouraged the development of media servers efficiently supporting video streams with variable bit rates. Several media server designs either support only constant bit-rate streams, ii) make resource reservations assuming a fixed bit rate for each stream, or iii) have only been demonstrated to work with constant bit rate streams.

It is therefore an aspect of the present invention for providing a distributed continuous-media server architecture, called Exedra, that efficiently supports variable bit-rate streams and reduces the requirements for storage device space, storage device bandwidth, buffer space, and network bandwidth with respect to servers that support only constant bit-rate streams.

Vast storage and bandwidth capacity requirements of even compressed video streams make it necessary to stripe video files across multiple storage devices. Assuming that a media storage server serves requests for several different stream files, appropriate striping makes it possible to scale the number of supported streams to the limit of the server resources, independent of the particular stream file being requested by clients. This is possible by retrieving different parts of each stream file from different storage devices, thus restricting the degree of imbalance in utilization among the storage devices.

However, it has been previously shown that both load imbalance across disks and disk overhead is causing disk striping of variable bit-rate streams to be efficient only on disk arrays of limited size. Therefore, the scalability of network servers that stripe variable bit-rate streams across multiple storage devices is a fundamental problem.

It is an aspect of an object of the present invention for providing a new storage device space allocation technique and striping policies for variable bit-rate streams that increase system throughput and improve scalability.

The Internet and online services introduce increasing requirements for quality-of-service guarantees in order to ensure that a failure does not result in denial of service for clients. Hard drives or storage devices continue to be a major source of failure. This is not surprising since disks are essentially the only moving mechanical parts of computers.

It is therefore an aspect of an object of the present invention for providing fault tolerance in storage device arrays and clusters of computer nodes that support variable bit-rate streams.

Variable bit-rate encoding of video streams can achieve quality equivalent to constant bit-rate encoding while requiring average bit rate that is lower by 40%. However, variable bit-rate streams have high variability in their resource requirements, which can lead to low utilization of storage device and network bandwidth in the common case. This occurs because the aggregate bandwidth requirements of concurrently served streams can be significantly higher than on average at particular time instances, and the admission control process bases its decisions on peak aggregate demand when considering new stream requests.

In order to improve resource utilization and the throughput of the system, a number of smoothing techniques have been proposed that can remove peaks in the required transfer bandwidth of individual streams by appropriately prefetching stream data at times of lower bandwidth demand. To date, smoothing schemes always prefetch data into the client buffers. Although such an approach can improve the utilization of both storage device and network bandwidth, it is dependent on the amount of buffer space available at the
client. However, emergence of client devices with widely different hardware configurations make it necessary to reconsider such assumptions.

It is therefore an aspect of the present invention a smoothing method that uses buffer space available in the server and provides efficient striping of variable bit-rate streams across either homogeneous or heterogeneous storage devices.

SUMMARY OF THE INVENTION

According to an aspect of the invention, there is provided a method and system for accessing variable bit-rate streams from one or a multitude of secondary storage devices. The system provides that data for each stream are retrieved according to a prespecified rate that may vary over time. The space of each storage device is managed as a collection of fixed-size chunks with length larger than a given minimum. The data transfers occur in periods (or rounds) of fixed duration. The data of each stream is distributed across a multitude of storage devices in a way that only one storage device is accessed for a stream during one or a multitude of rounds. Detailed accounting is done for the access time of the storage devices, the transmission time of the network transfer devices, and the available memory space in the system for each round. The space of each storage device is managed independently from that of the others. The available memory is allocated contiguously in the virtual space for each access of a storage device, and can be deallocated in units smaller than the length of the original allocation. When storage devices fail, data redundancy and extra reserved device channel bandwidth guarantee uninterrupted system operation.

According to a further aspect of the invention, there is provided a server-based smoothing method that uses only buffer space available at the server for smoothing storage device data transfers. The smoothing method is also extended to support striping of variable bit-rate streams across heterogeneous storage devices. The present invention maximizes the average number of users supported concurrently in continuous-media server systems by applying smoothing techniques and combining them appropriately with storage device striping and admission control policies. In order to prevent excessive smoothing from exhausting the available buffer space, data prefetching is done as long as the proportion of server buffer required by each stream does not exceed the corresponding proportion of the required storage device bandwidth. Thus, the smoothing process is adjusted automatically, according to the total memory and storage device bandwidth available in the server.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be described in detail with reference to the accompanying drawings, in which like numerals denote like parts, and in which

FIG. 1 is a block diagram of a distributed streaming server in accordance with an embodiment of the present invention;

FIG. 2 is a logical representation of a stride-based allocation of disk space on one disk of FIG. 1;

FIG. 3 is a graph of data requirements of twenty consecutive rounds one second each, in an MPEG-2 clip;

FIGS. 4(a) to (c) are graphs of data accesses for the clip of FIG. 3 using alternative striping techniques over two disks;

FIG. 5 is a graph of number of streams versus load under Fixed-Grain Striping;

FIG. 6 is a graph of a ratio of total number of rejected streams over total number of accepted streams versus load under Fixed-Grain Striping;

FIG. 7 is a graph of number of streams versus block size Bf under Fixed-Grain Striping;

FIG. 8 is a graph of round disk time versus block size Bf under Fixed-Grain Striping;

FIG. 9 is a graph of disk busy time per round versus number of disks under Fixed-Grain Striping;

FIG. 10 is a graph of number of streams versus number of disks under Fixed-Grain Striping;

FIG. 11 is a graph of number of streams versus load under Variable Grain Striping;

FIG. 12 is a graph of a ratio of total number of rejected streams over total number of accepted streams versus load under Variable Grain Striping;

FIG. 13 is a graph of round disk access time versus number of disks under Variable Grain Striping;

FIG. 14 is a graph of number of streams versus individual stream types in both Fixed-Grain versus Variable-Grain Striping;

FIG. 15 is a graph of round disk time in Simulated Disk Mode;

FIG. 16 is a server-based system for smoothing of variable bit-rate streams;

FIG. 17 is a functional block diagram of the distributed media server of FIG. 1;

FIG. 18 is a block diagram of a circular vector of dispatch queues that keep track of admitted streams yet to be activated; and

FIG. 19 is shown a block diagram of a fault-tolerant disk array for a media server.

DETAILED DESCRIPTION

Referring to the drawings and initially to FIG. 1, there is illustrated a block diagram of a distributed streaming server 100 in accordance with an embodiment of the present invention. The distributed streaming server 100 comprises storage devices 110 for storing media stream data 114, transfer nodes 120, admission control nodes 130, and a schedule database 140 containing scheduling information. The media stream data 114 are compressed, such as, according to the MPEG-2 specification, with constant quality quantization parameters and variable bit rates. Clients 150 with appropriate stream decoding capability send client/playlist requests 160 and receive stream data 114 via a high-speed network 170. Alternate compression schemes as is known in the art may also be used.

In the streaming server 100, the stream data 114 are retrieved from the storage devices 110 and sent to the clients 150 through the Transfer Nodes 120. Both the admission control nodes 130 and the transfer nodes 120 use of stream scheduling information maintained in the Schedule Database 140.

The streaming server 100 is operated using the server-push model, but other models are possible. When a playback session starts for a client 150, the server 100 periodically sends data to the client 150 until either the end of the stream is reached, or the client 150 explicitly requests suspension of the playback. The server-push model reduces the control traffic from the client to the server and facilitates resource reservation at the server side, when compared to a client-pull model. The data transfers occur in rounds of fixed duration, in each round, an appropriate amount of data is
retrieved from the storage devices 110 into a set of server buffers 122 reserved for each active client 150. Concurrently, data are sent from the server buffers 122 to the client 150 through network interfaces 124. Round-based operation is used in media servers in order to keep the reservation of the resources and the scheduling-related bookkeeping of the data transfers manageable.

Due to the large amount of network bandwidth required for this kind of service, preferably the server 100 is connected to a high-speed network 170 through a multitude of network interfaces 124. The amount of stream data periodically sent to the clients 150 are determined by the decoding frame rate of the stream and the resource management policy of the server 100. A policy is to send to the client during each round the amount of data that will be needed for the decoding process of the next round; any other policy that does not violate the timing requirements and buffering constraints of the decoding client is also acceptable.

The stream data are stored across multiple storage devices 110, such as hard drives or disks, as shown in FIG. 1. Every storage device 110 is connected to a particular Transfer Node 120, through a Storage Interconnect 112, which is either i) a standard I/O channel (for example, Small Computer System Interface), ii) standard network storage equipment (for example, Fibre-Channel), or iii) a general purpose network (as with Network-Attached Secure Disks). Alternate, part of the server functionality can be offloaded to network-attached storage devices.

The Transfer Nodes 120 are computers responsible for scheduling and initiating all data accesses from the attached storage devices 110. Data arriving from the storage devices 110 are temporarily staged in the Server Buffer memory 122 of the Transfer Node 120 before being sent to the client 150 through the high-speed network 170. The bandwidth of the system bus (such as the Peripheral Component Interconnect) is the critical resource within each Transfer Node 120 that essentially defines the number and the capacity of the attached network or I/O channel interfaces.

Playback requests 160 arriving from the clients 150 are initially directed to an Admission Control Node 130, where it is determined whether enough resources exist to activate the requested playback session either immediately or within a few rounds. If a new playback request is accepted, commands are sent to the Transfer Nodes 120 to begin the appropriate data accesses and transfers. The computational complexity of the general stream scheduling problem is combinatorial in the number of streams considered for activation and the number of reserved resources. As the acceptable initiation latency is limited, a simple scheduling approach with complexity linear with the number of rounds of each stream and the number of reserved resources, is used. The admission control nodes are distributed across multiple processors as shown in FIG. 1, and concurrency control issues that potentially arise are also taken into account.

The Schedule Database 140 maintains information on the amount of data that needs to be retrieved during each round for each stream and on which storage devices this data is stored. It also specifies the amount of buffer space required and the amount of data sent to the client by the Transfer Nodes 120 during each round. The scheduling information is generated before the media stream is first stored and is used for both admission control and for controlling data transfers during playback. Since this information changes infrequently, it is replicated to avoid potential bottlenecks.

Referring to FIG. 2, there is illustrated a logical representation of a stride-based allocation of storage device space on one storage device of FIG. 1. In stride-based allocation, storage device space is allocated in large, fixed-sized chunks (i, i+1, i+2, i+3, i+4, i+5) called strides 200, which are chosen larger than the maximum stream request size per storage device during a round. The stored streams are accessed sequentially according to a predefined (albeit variable) rate; therefore, the maximum amount of data accessed from a storage device during a round for a stream is known a priori. Stride-based allocation eliminates external fragmentation, while internal fragmentation remains negligible because of the large size of the streams, and because a stride may contain data of more than one round as shown in stride i+3 of FIG. 2. The contents of each stride are tracked in a stream stride index 210.

When a stream is retrieved, only the requested amount of data (one of Request s j+1, j+2, or j+3), and not the entire stride 200, necessary for each round is fetched to memory at a time. Since the size of a stream request per round never exceeds the stride size, at most two partial strides accesses (two seek and rotation delays) are required to serve the request of a round on each storage device. Thus, the stride-based allocation scheme sets an upper-bound on the estimated storage device access overhead during retrieval. This avoids an arbitrary number of actuator movements required by prior allocation methods. This also keeps storage device access head movements to a minimum. For example, while i+3 stride 200 contains parts of Request j+1 and Request j+2, and Request j+3, there is no Request that is stored over more than two strides 200.

While storing the data of each storage device request contiguously, on a disk for example, would reduce the storage device overhead to a single seek and rotation delay (instead of two at most), the overhead for storage management (bookkeeping) of large highly utilized storage devices could become significant. An advantage of the present invention is the reduction of overhead for storage management.

In the sequence definitions that follow, a zero value is assumed outside the specified range.

In the server 100 with D functionally equivalent storage devices, the stream Network Sequence, Sn, of length L defines the amount of data, Sn[i], 1<=i<=L, that the server 100 sends to a particular client 150 during round i after its playback starts. Similarly, the Buffer Sequence Sb of length Lb=L+1 defines the server buffer 122 space, Sb[i], 1<=i<=Lb, occupied by the stream data during round i. The Storage Device Sequence Sd of length Ld=L defines the total amount of data Sd[i], 1<=i<=Ld, 1, retrieved from all the storage devices 110 in round i for the client.

The data are stored on the storage devices 110 in strides 200. The stride size Bs is a multiple of the logical block size Bl, which is a multiple of the physical sector size Bp of the storage device. Both storage device transfer requests and memory buffer reservations are specified in multiples of the logical block size Bl. After taking into account logical block quantization issues, the storage sequence Sd can be derived from the network sequence Sn as follows: If

$$K^d(i) = \sum_{k=0}^{\infty} S_d(k+1)$$

specifies the cumulative number of blocks Bl retrieved through round i, then

$$S_d(i) = (K^d(i) - K^d(i-1)) B_s$$
The Storage Device Stripping Sequence \( S\text{md}(i,k) \) determines the amount of data \( S\text{md}(i,k) \), for \( 0 < i < l, d = 1 \), that are retrieved from the storage device \( 110k \), \( 0 < k < l, d = 1 \), in round i. It is generated from the Storage Device Sequence \( S\text{d} \), according to the striding policy used. Each storage device \( 110k \) has edge to edge seek time \( T\text{fullseek} \), single track seek time \( T\text{trackseek} \), average rotation latency \( T\text{avgrot} \), and minimum internal transmission rate \( R\text{disk} \). The stride-based storage device space allocation policy enforces an upper bound of at most two storage device arm movements per storage device for each client per round. The total seek distance is also limited using a SCAN storage device scheduling policy. Let \( M\text{i} \) be the number of active streams during round i of system operation. Where the playback of stream \( j \), \( 1 < j < M\text{i} \), is initiated at round \( l\text{j} \) of system operation, then, the total access time on storage device \( k \) in round i of the system operation has an upper bound of:

\[
I_{\text{data}}(i,k) = 2T_{\text{fullseek}} + 2M\text{i}(T_{\text{trackseek}} + T_{\text{avgrot}}) + \sum_{j=1}^{M\text{i}} S\text{md}(i-l\text{j},k)/R_{\text{disk}}
\]

where \( S\text{md}(i,j) \) is the storage device striping sequence of client \( j \). \( T\text{fullseek} \) is counted twice due to the storage device arm movement from the SCAN policy, while the factor two of the second term is due to the stride-based method. The first term should be accounted for only once in the storage device time reservation structure of each storage device. Then, each client \( j \) incurs an additional maximum access time of:

\[
T_{\text{data}}(i,k) = 2T_{\text{trackseek}} + T_{\text{avgrot}} + S\text{md}(i-l\text{j},k)/R_{\text{disk}}
\]

on storage device \( k \) during round i, when \( S\text{md}(i-l\text{j},k) > 0 \), and zero otherwise.

If \( R\text{net} \) is the total high-speed network bandwidth available to the server 100, then the corresponding network transmission time reserved for client \( j \) 150 in round i becomes \( T_{\text{net}}(i) = S\text{md}(i-l\text{j})/R\text{net} \), where \( S\text{md}(i-l\text{j}) \) is the Network Sequence of client \( j \) 150. The total server memory buffer reserved for client \( j \) 150 in round i becomes \( B\text{md}(i) = S\text{md}(i-l\text{j}) \), where \( B\text{md}(i) \) is the Buffer Sequence of client \( j \) 150. Although, the above expressions for \( T_{\text{net}}(i) \) and \( B\text{md}(i) \) are sufficient for the needs of the present embodiments, accounting for available network bandwidth and buffer memory within each individual Transfer Node may require them to be split into appropriate sub-expressions.

The reservations of transfer time on each network interface 124 and buffer space 122 on each transfer node 120 are more straightforward, and are based on the Network Stripping Sequence and Buffer Stripping Sequence, respectively.

In traditional storage systems, data access patterns are relatively hard to predict, making it difficult to determine optimal storage device striping parameters. However, with read-only sequential access being the common case for video/media streaming, it is possible to predict to some degree the expected system load requirements during retrieval, making it possible to determine appropriate storage device striping parameters a priori for the storage and retrieval of the data. The present invention includes exploiting this characteristic of stored video/media streams.

Referring to FIG. 3, there is illustrated a graph of data requirements of twenty consecutive rounds (one second each), in an MPEG-2 clip. Referring to FIGS. 4(a) to (c), there are illustrated graphs of data accesses for the clip of FIG. 3 using alternative striping techniques over two storage devices. With Fixed-Grain Stripping, the needed blocks of size \( B\text{f} \) are retrieved round-robin from the two storage devices every round. In Variable-Grain Stripping, a different storage device is accessed in each round, according to the byte requirements of the original clip. In Group-Grain Stripping with \( G = 2 \), stream data worth of two rounds are accessed from a different storage device every two rounds.

With Fixed-Grain Stripping, data are striped round-robin across the storage devices in blocks of a fixed size \( B\text{f} \), a multiple of the logical block size \( B\text{i} \) defined previously. During each round, the required number of blocks are accessed from each storage device. An example of Fixed-Grain Stripping is shown in FIG. 4(a). In the definition below, \( \text{modD} \) denotes the remainder of the division with \( D \), and \( \text{divD} \) denotes the integer quotient of the division with \( D \). The equation

\[
K_{\text{data}}(i) = \left\lfloor \frac{\sum_{j=1}^{M\text{i}} S\text{md}(i,j)}{B\text{f}} \right\rfloor
\]

specifies the cumulative number of blocks \( B\text{f} \) retrieved through round i for a specific client. When

\[
K_{\text{data}}(i) - \text{modD} > 0
\]

all blocks accessed for the client during round i lie on the same stripe of blocks. Then, the striping sequence \( S\text{md}(i) \) is equal to:

\[
S\text{md}(i,k) = D_{\text{f}}(i,k)/B\text{f}
\]

where

\[
D_{\text{f}}(i,k) = \begin{cases} 
1, & \text{if } K_{\text{modD}}(i-1) < k_{\text{modD}} \leq K_{\text{modD}}(i) \\
0, & \text{otherwise,}
\end{cases}
\]

specifies the particular storage devices that need to be accessed at most once for the stream. When

\[
K_{\text{data}}(i) > K_{\text{data}}(i-1) > 0
\]

the blocks accessed for the client during round i lie on more than one stripe, and the striping sequence becomes:

\[
S\text{md}(i,k) = (K_{\text{modD}}(i) - K_{\text{modD}}(i-1))/B\text{f} \times D_{\text{f}}(i,k)/B\text{f}
\]

where

\[
D_{\text{f}}(i,k) = \begin{cases} 
2, & \text{if } k_{\text{modD}}(i-1) < k_{\text{modD}} \leq K_{\text{modD}}(i) \\
1, & \text{if } k_{\text{modD}} > \max(K_{\text{modD}}(i-1), K_{\text{modD}}(i)) \\
1, & \text{if } k_{\text{modD}} \leq \min(K_{\text{modD}}(i-1), K_{\text{modD}}(i)) \\
0, & \text{otherwise.}
\end{cases}
\]

The first term in the Equation accounts for blocks in stripes fully accessed (i.e., all D blocks, where D is the number of storage devices), while the second term accounts for blocks of stripes partially accessed in round i (i.e., fewer than D blocks).

With Variable-Grain Stripping, the data retrieved during a round for a client are always accessed from a single storage
device round-robin, as shown in FIG. 4(b). The corresponding
stripping sequence becomes:

\[ S_{Bj}^*(f(k)) = K^*(i-1) \cdot B_j. \]

when i mod D = k, with

\[ K^*(i) = \sum_{j=0}^{G-1} S_b(j) \cdot \frac{B_j}{B}. \]

and Smdf(i,k) = 0 when i mod D not equal k. Therefore, the
Storage Device Sequence determines the particular single
storage device accessed and the exact amount of data
retrieved during each round.

Variable-Grain Stripping is a special case (with G=1) of a
method herein called Group-Grain Stripping, where the
amount of data required by a client over G rounds is
retrieved from the Gth storage device that changes round robin (see FIG. 4(c), noting that the y-axis
uses a different scale). The parameter G, G>1, is called
Group Size. The stripping sequence for Group-Grain Stripping is equal to:

\[ S_{Bj}^f(f(k)) = K^*(i+G-1) - K^*(i-1) \cdot B_j. \]

when i mod G = 0 AND (i div G) mod D = k, and Smdf(i,k) = 0
otherwise.

As G increases, the fewer storage devices are accessed to
reduce the storage device overhead (although the access
time per request is increased). On the other hand, the fixed round
spacing between subsequent requests for a stream, basically
divides the server into G virtual servers. The fixed group size
G guarantees that two streams started from the same storage
device at rounds i and j with i not equal j (mod G), do not
have any storage device transfers in a common round. This
is different from increasing Bj in Fixed-Grain Stripping,
where accesses from different streams can randomly coincide
on the same storage device in the same round, resulting
in the system saturating with fewer streams. Increasing G for
a particular round time is advantageous with future expected
changes in storage device technology.

Alternately, aggregation of storage device transfers can
also be achieved with an appropriate increase of round time.
However, this could directly affect the responsiveness of the
system by potentially increasing the initiation latency of
each playback. Longer round time would also increase the
required buffer space.

It will be understood by those skilled in the art that the
stride-based allocation scheme may be based on variable
t-sized strides rather than fixed sized where each of the strides
still satisfy the criteria of containing at least one round worth
of stream.

Referring to FIG. 17, there is shown a functional block
diagram of the distributed media server of FIG. 1. The
architecture 1700 comprises admission control 1710, dis-
patcher 1730, stream manager 1715, buffer manager 1720,
metadata managers 1725, storage device managers 1735,
and storage devices 1740 that store the media data.

The basic functions of the media server include file
naming, resource reservation, admission control, logical to
physical metadata mapping, buffer management, and storage
device and network transfer scheduling.

The admission control module 1710 uses circular vectors
of sufficient length to represent the allocated storage device
time, network time, and buffer space, respectively. On
system startup, all elements of storage device time vectors
are initialized to 2^16, while the network time and
buffer space vector elements are set to zero. When a new
stream request arrives, the admission control 1710 is
performed by checking the requirements of the stream against
currently available resources. In particular, the total service
time of each storage device 1740 in any round may not exceed
the round duration, the total network service time on
any network interface may not exceed the round duration,
and the total occupied buffer space on any transfer node may
be no larger than the corresponding server buffer capacity.

If the admission control test is passed, then the resource
sequences of the stream are added to the corresponding
system vectors managed by the module 1710, and the stream
is scheduled for playback. The dispatcher 1730 is respon-
sible for starting the playback at the appropriate round.

Referring to FIG. 18, there is shown a block diagram of a
vector of dispatch queues that keeps track of admitted streams yet to be activated. Notification records for
the accepted request are inserted into dispatch queues at the
appropriate offset from the current round. When an upcom-
ing round becomes current, the notification records are used
for activating the stream and starting its data transfers.

The metadata manager 1725 for stream metadata man-
agement is organized in a layered above storage device sched-
uling. It is responsible for storage device space allocation
during stream recording, and for translating stream file
offsets to physical block locations during playback. The stream metadata are maintained as regular files in the host
OS (of each transfer node, in the general case), while stream
data are stored separately on dedicated storage devices. The
storage space of the data storage devices 1740 is organized
in strides, with a bitmap that has a separate bit for each
stride. A single-level directory is used for mapping the
identifier of each recorded stream into a direct index of the
corresponding allocated strides. A separate directory of this
form exists for each storage device.

When a stream is striped across multiple storage devices,
a stream file is created on each data storage device. Each
transfer request received by the metadata manager 1725
specifies the starting offset in the corresponding stream file
and the number of logical blocks to be accessed. With the
stream index, each such request is translated to a sequence
of contiguous storage device transfers, each specifying the
starting physical block location and the number of blocks.
From the stride-based storage device space allocation, it
follows that each logical request will be translated into at
most two physical contiguous storage device transfers.

The provision of a separate metadata manager for each
storage device has an advantage of application to general
storage device array organization, including those consisting
of heterogeneous storage devices. Although the handling
of heterogeneous devices (for example disks of different sizes
from different manufacturers) may not be necessary in
limited size traditional storage systems, it might prove
advantageous in the incremental growth and survivability of large
scalable media storage installations.

In order to keep system performance predictable and
unbiased from particular storage device geometry features,
some control is exercised on the storage device space
allocation pattern. In particular, storage device zoning could
possibly lead to excessively optimistic or pessimistic data
access delays, if most allocation were in the outer or inner
cylinders of the storage devices. Similarly, contiguous allo-
cation could lead to lower than expected delays in some
special cases (such as when streams are stored on a single
storage device with a very large on-storage device cache).
However, low-level storage device geometry is generally not
disclosed by the storage device manufacturers, therefore,
The stream manager 1715 is responsible for generating data transfer schedules, where each schedule specifies the amount of data accessed from the storage devices, stored in the server buffer and transmitted over the network during each round. The schedule manager accepts as input actual media files (or their frame size traces), along with the prefetching and striping scheme that can be used. The prefetching schemes can make use of the buffer space that is available at the server or client side, depending on the policy.

Assuming that playback initiation requests arrive independently of one another, according to a Poisson process. The system load can be controlled by setting the mean arrival rate $\lambda$ of playback initiation requests. The maximum possible service rate $\mu$, expressed in streams per round for streams of data size $St$ bytes, is equal to $\mu = \frac{D}{T \text{round/} St}$. Correspondingly, the system load $\rho$, is equal to $\rho = \frac{\lambda}{\mu} = \frac{1}{\lambda} \lambda = \text{max} - \lambda$. The definition of $\rho$ is used in this evaluation.

When a playback request arrives, the admission control 1710 module checks whether available resources exist for every round during playback. The test considers the exact data transfers of the requested playback for every round and also the corresponding available storage device transfer time, network transfer time and buffer space in the system. If the request cannot be initiated in the next round, the test is repeated for each round up to $\lceil 1/\lambda \rceil$ rounds into the future, until the first round is found, where the requested playback can be started with guaranteed sufficiency of resources. Checking $\lceil 1/\lambda \rceil$ rounds into the future achieves most of the potential system capacity. If not accepted, the request is discarded rather than being kept in a queue.

Fault Tolerance for Media Servers

Referring to FIG. 19, there is shown a block diagram of a fault tolerance storage device array for a media server. The array 2000 is configured for single storage device failure and comprises four disks 2010 to 2013 where content of each of the disks is replicated across the other three disks. Each unit of replication corresponds to data retrieved by a client during one round of playback. Each storage device has primary data for a stream and one backup replica of the primary data is stored spread over the other storage devices.

The methods of replication include deterministic replica placement and random replica placement. In deterministic replica placement, data of a media file stored consecutively on disk 2010 and retrieved during different playback rounds are replicated round-robin across the other disks. The primary data of the other disks are replicated in a similar way. In random replica placement, data of a media file stored consecutively on disk 2010 and retrieved during different playback rounds are replicated on randomly chosen disks 2011 to 2013 where data for each round is replicated on a randomly disk chosen from disks 2011 to 2013. The primary data of the other disks are replicated in a similar way.

An aspect of an objective is to allocate resources in such a manner that service to accepted stream requests are not interrupted during (single) storage device failures. Retriev ing backup replicas of data stored on a failed storage device requires extra bandwidth to be reserved in advance across the surviving storage devices. Thus, the array 2000 normally operates below full capacity. Alternatively, when a storage device fails and no extra bandwidth has been reserved, service will become unavailable for a number of active users with aggregate bandwidth requirements no less than the transfer capacity of one storage device, assuming that data have been replicated as described previously.

The load that is normally handled by a failed storage device is equally divided among the D-1 surviving storage
devices. Thus, tolerating one storage device failure requires that extra bandwidth be reserved on each storage device equal to \(1/(D-1)\) the bandwidth capacity of one storage device. In order to achieve this, the access time of the backup replicas stored on one storage device are accumulated separately for every storage device that stores the corresponding primary data. When a storage device fails, the access load incurred on every surviving storage device is approximately known. In fact, the additional access time that has to be reserved on a storage device in each round is equal to the maximum time required for retrieving backup replicas for another storage device that has failed as shown by data 2020 of FIG. 19.

For each storage device, \(D\) separate vectors indexed by round number are maintained. One of the vectors accumulates access delays for retrieving primary data. The remaining \(D-1\) vectors accumulate access delays for retrieving backup replicas that correspond to primary data stored on each of the other \(D-1\) storage devices. Thus, in each round, the sum of the primary data access time and the maximum of the backup data access times is reserved on each storage device. This method is herein referred to as Minimum Reservation.

The minimum reservation scheme requires maintaining the number of vectors equal to the square of the number of storage devices. Each vector is accessed in a circular fashion and has minimum length equal to that of the longest stream expressed in numbers of rounds. When using large storage device arrays, this might raise concerns regarding the computational and memory requirements involved. In practice, the reduction in unused bandwidth is diminishing as the number of storage devices increases beyond sixteen. Therefore, it makes sense to apply the data replication within storage device groups of limited size, when the storage device array size becomes larger. This keeps the bookkeeping overhead limited and preserves the scalability of this method when stream data are striped across large storage device arrays.

It will be understood by those skilled in the art that the Minimum Reservation method may be applied to different computer nodes when primary data of each computer node is replicated across other computer nodes. Thus, when one node goes down, the primary data from all its storage devices is still accessible through replicas available on the other computer nodes. In servers consisting of multiple nodes, failure of an entire node can be handled gracefully, by keeping each storage device of a node in a separate storage device group and limiting the replication within each group.

When a node fails, inaccessible data for each of its storage devices can be retrieved using replicas available on other storage devices of the corresponding groups.

It will be understood by those skilled in the art that the Minimum Reservation method may also be applied for handling multiple storage device failures which only requires storing multiple backup replicas, and making bandwidth reservations for more than one failed storage device accordingly.

Server-Based Smoothing Referring to FIG. 16, there is provided a system 1600 for server-based smoothing of variable bit-rate streams. The system 1600 comprises a server 1610 having storage devices 1620 for storing stream data, server buffers 1630 for buffering the content prior to transmission, and network interfaces 1640 for interfacing to a network 1650; and clients 1660 to receive the data over client network interfaces 1670 where client buffers 1680 buffers the data before decoding by decoders 1690.

The system 1600 operates according to a server-push model such that when a playback session starts, the server 1610 periodically sends data to the client 1660 until either the end of the stream is reached, or the client 1660 explicitly requests suspension of the playback. Data transfers occur in rounds of fixed duration \(T\) round. In each round, an appropriate amount of data is retrieved from the storage devices 1620 into a set of server buffers 1630 reserved for each active client. Concurrently, data are sent from the server buffers 1630 to the client 1640 over the network 1650.

The amount of stream data periodically sent to the client 1660 is determined by the decoding frame rate of the stream and the resource management policy of the server-based system 1600. A policy is to set to send to the client 1660 during each round the amount of data that is needed for the decoding process at the client in the next round; any other policy that does not violate the timing requirements and buffering constraints of the decoding client would be also acceptable.

The streams are compressed according to the MPEG-2 specification, or any other encoding scheme that supports constant quality quantization parameters and variable bit rates. The stream data are stored across the multiple storage devices 1620.

Playback requests arriving from the clients 1660 are initially directed to an admission control module 1615, where it is determined whether enough resources exist to activate the requested playback session either immediately or within a limited number of rounds. The admission control module maintains a schedule (scheduling information) on how much data needs to be accessed from each storage device of the storage devices 1620 in any given round, the amount of server buffer space required from the server buffers 1630, and how much data needs to be transferred to the client 1660.

This scheduling information for each of the stream files is generated when the media stream is first stored and is generated by scheduling to prefetch stream data so that storage device bandwidth peaks are smoothed out. By smoothing out the peaks, a greater number of streams may be accessed from a set of storage devices at any one time.

One crucial issue with storage device prefetching is how to maintain an appropriate balance between storage device bandwidth and server buffer space usage. Too aggressive prefetching can limit the number of concurrent streams that can be supported because of excessive server buffer usage.

In accordance with an embodiment of the present invention, there is provided a stream scheduling procedure that specifies for each stream both the variable server buffer and storage device bandwidth requirements over time. A storage device block \(b\) originally scheduled for round \(i\) is prefetched in a previous round \(j\) only if: i) the storage device bandwidth requirement in round \(j\) with the prefetched block does not exceed the original storage device bandwidth requirement of round \(i\), and ii) the proportion of server buffer required in each of the rounds \(j\) up to \(i-1\) after prefetching block \(b\) does not exceed the proportion of storage device bandwidth required in round \(i\) without \(b\).

The first condition is necessary in order for the prefetching to have a smoothing effect on the storage device bandwidth requirements over time. The second condition is a heuristic that is applied in order to prevent exhaustion of the server buffer. Both conditions are applied to individual streams, and to multiple streams concurrently.

Thus, the stream smoothing method prefetches data into server buffers, and has several important advantages: ability to support clients with minimal memory resources (such as inexpensive mass-produced specialized devices) and still benefit from smoothing, optimizing for storage device bandwidth...
width that is estimated to increase at rates an order of magnitude slower than network link bandwidth, reduced complexity in admission control processing by not having to support a separate transfer schedule for each individual client type, and reduced stream replication with storage device striping policies that are based on a specified client configuration and retrieval sequence.

A "smoothness" criterion that is based on Majorization Theory is used. For any x=(x1, ..., xn) in R^n, let the square bracket subscripts denote the elements of x in decreasing order x[1]=x[n]. For x,y in R^n, x is majorized by y, x≤y, if:

\[ \sum_{i=1}^{n} x_i \leq \sum_{i=1}^{n} y_i, \quad k = 1, \ldots, n-1 \]

and

\[ \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \]

Then, consider x smoother than y, if x≤y. Finally, we call a vector x in R^n majorization-minimal if there is no other vector z in R^n such that z≤x.

In accordance with another embodiment of the present invention, there is provided a method that, given a stream network sequence Sn and a target server configuration, generates a smoothed storage device sequence Sd. The generated storage device sequence is majorization-minimal under the specified constraints. The generated storage device sequence is subsequently transformed into a stripe sequence Sn according to some storage device striping method, such as the Variable-Grain Striping.

The storage device time reservation for a block of X bytes is approximately equal to:

\[ T_{disk}(X) = 2(T_{network} + T_{server}) + X/R_{disk} \]

Definition 1: Let the Disk Time Proportion of X bytes, Pd(X), be the fraction of the round time Tround that the disk time reservation Tdisk(X) occupies: Pd(X)=Tdisk(X)/Tround. Further let the Buffer Space Proportion of X bytes, Pb(X), be the fraction of the buffer space for each disk, Bdisk, (Bdisk is the total buffer space divided by the number of disks D) that X bytes occupy in a round: Pb(X)=X/Bdisk. Then, the Maximum Resource Proportion in round i, is the maximum of the corresponding disk time and buffer space proportion in that round: \( \text{max}(\text{small max}(Pd(Sd)), Pb(Sb)) \).

The above utilization definitions refer to resource reservations within a round. However, for conciseness of the following presentation, the words Round and Reserved are dropped and are referred to as Disk Utilization, Buffer Utilization, Maximum Utilization, and respectively.

Definition 2: The Deadline Round for a block is the latest round at which the block can be accessed from the storage device without incurring a real-time violation at the network transfer. Then, with respect to a specific block, all rounds before the deadline round are considered Candidate Rounds and the one actually chosen for prefetching is called the Prefetch Round. All the rounds between the deadline and a prefetch round are called Shift Rounds. These definitions only affect the number of blocks accessed in each round, since stream block accesses are done sequentially during playback.

Definition 3: The Maximum-Propotion Constraint is defined as the requirement that the maximum resource proportion of the deadline round is no less than the maximum resource proportion of the corresponding (if any) prefetch and shift rounds.

The Server Smoothening method is as follows:

0. processServerSmoothing
1. input: Lr, Sd(0) (\text{outside [1..Lr]}), B
2. output: Lr, Sd[i], Lr, Sb[i]
3. begin
4. blockQuantize(Lr, Sd[i], B) (* see App. B *)
5. for l=0 to Lr-1
6. if Sd(l) \text{ or } Sb(l) \text{ or } Sd(l+1)
7. repeat
8. lmin := l
9. Pmax := \text{max}(Pd(Sd(l)), Pb(Sb(l)))
10. lstep := l
11. while (lstep < Lr AND Pmax) := false
12. Pmax := \text{max}(Pd(Sd(l)), Pb(Sb(l)))
13. lstep := l
14. Pmax := \text{max}(Pd(Sd(l)), Pb(Sb(l)))
15. (*check for max proportion decrease*)
16. if (Tmax < Tmax)
17. lmin := l
18. else if (Pmax < Pmax)
19. end-if
20. lstep := lstep - 1
21. end-while
22. if lmin < l (\text{update vectors } *)
23. Sd(l) := Sd(l) + B
24. Sb(l) := Sb(l) + B
25. for l=1 to lstep - 1
26. lstep := lstep - 1
27. Sd(l) := Sd(l) + B
28. Sb(l) := Sb(l) + B
29. end-for
30. Sd(lstep) := Sd(lstep) + B
31. end-if
32. until lmin := l (\text{prefetch search failed})
33. end-if
34. end-for
35. end

This method initially invokes the blockQuantize procedure.

The BlockQuantize Procedure

0. process blockQuantize
1. input: Lr, Sd(0) (\text{outside [1..Lr]}), B
2. output: Lr, Sd(i), Lr, Sb(i)
3. begin
4. Sd(1) := 0, Sb(1) := 0
5. l := Lr, l := l + 1
6. totSn := 0
7. for l=0 to Lr-1
8. prevSn := totSn, totSn := totSn + Sd(l+1)
9. (* to use function ceil() for the [] operation *)
10. Sd(l) := Sd(l) + ceil(r(Sd(l)))
11. Sb(l) := Sb(l) + 1
12. end-for
13. end
This implies that the storage sequence can be smoothed out without incurring buffer space proportion peaks exceeding the storage device time proportion of the current round. Otherwise, the block prefetching operation will not have any positive smoothing effect overall, and the corresponding search fails.

It has traditionally been assumed that storage device arrays consist of homogeneous storage devices (for example identical disks from the same manufacturer), presumably in order to keep the system complexity manageable. With the scalability of stream striping demonstrated and the sequential access of stored video making things somewhat simpler, systems with different storage devices types may be provided that may be scaled incrementally with the newest storage device technology. Newer storage device models typically achieve higher transfer rates and have larger storage capacities.

The objective is to maximize the number of active streams by increasing the storage device bandwidth utilization across all the storage devices. This might lead to suboptimal storage capacity utilization, assuming it is affordable given the current technology trends. In order to maximize the storage device bandwidth utilization, the Server Smoothing method is extended to handle heterogeneous storage devices. In particular, the disk time Tdisk(X) and the disk time proportion function P(X) were redefined to accept a second disk type parameter k that specifies the particular disk parameters to be used P(X, k) = Tdisk(X, k) / Trounda. During the operation of the Server Smoothing method, the disk type k assumed in each round can be derived using a simple rule, such as k = t mod D, where D is the total number of the disks.

The above disclosure generally describes the present invention. A more complete understanding can be obtained by reference to the following specific Examples. These Examples are described solely for purposes of illustration and are not intended to limit the scope of the invention. Changes in form and substitution of equivalents are contemplated as circumstances may suggest or render expedient. Although specific terms have been employed herein, such terms are intended in a descriptive sense and not for purposes of limitation.

**EXAMPLES**

The examples are described for the purposes of illustration and are not intended to limit the scope of the invention.

A Media Server System

A media server system or stream server system was built in order to evaluate the resource requirements of the different striping techniques. The modules were implemented in about 17,000 lines of C++/Pthreads code on AIX4.1, and ran on a single node. The code was linked either to the University of Michigan DiskSim disk simulation package, which incorporated advanced features of modern disks such as on-disk cache and zones for simulated disk access time measurements, or to hardware disks through their raw device interfaces. The indexing metadata were stored as regular Unix files, and during operation were kept in main memory. The MPEG-2 decoder from the MPEG Software Simulation Group was used for stream frame size identification.

The basic responsibilities of the media server included file naming, resource reservation, admission control, logical to physical metadata mapping, buffer management, and disk and network transfer scheduling.

With appropriate configuration parameters, the system operated at different levels of detail. In Admission Control mode, the system receives playback requests, does admission control and resource reservation but no actual data transfers take place. In Simulated Disk mode, all the modules are functional, and disk request processing takes place using the specified DiskSim disk array.

The system was primarily used in Admission Control mode (except for our validation study, where the system was used in Simulated Disk mode). The Admission Control module used circular vectors of sufficient length to represent the allocated time of each disk, the network time, and the buffer space respectively. On system startup, the disk time vectors are initialized to 2-1 fullSeek, while the network time and buffer space are initially set to zero. When a new stream request arrived, admission control is performed by checking against current available resources. In particular, the total service time of each disk in any round may not exceed the round duration, and the total network service time may also not exceed the round duration, while the total occupied buffer space may be no longer than the server buffer capacity. If the admission control test was passed, the resource sequences of the stream are added to the corresponding vectors of the module, and the stream is scheduled for playback.

**Performance Evaluation Method**

The playback initiation requests arrived independently of one another, according to a Poisson process. The system load was controlled through the mean arrival rate \( \lambda \) of playback initiation requests. Assuming that disk transfers form the bottleneck resource, in a perfectly efficient system there is no disk overhead involved in accessing disk data. Then, the maximum arrival rate \( \lambda \) was chosen to equal \( \lambda_{\max} \) of playback initiation requests, that corresponds to system load 100%, to be equal to the mean service rate with which stream playback would complete in that perfectly efficient system. This makes it possible to show the performance benefit of arbitrarily efficient data striping policies. Subsequently, the mean service rate \( \mu \), expressed in streams per round, for streams of data size \( Soty \) bytes becomes: \( \mu = D \cdot Rdisk \cdot Trounda / Soty \). Correspondingly, the system load \( \rho \) was set equal to: \( \rho = \lambda / \mu = 1 \), where \( \lambda = \lambda_{\max} / \mu \).

Another important decision had to do with the admission control process. When a playback request arrived, it is checked to determine if available resources existed for every round during playback. The test considered the exact data transfers of the requested playback for every round and also the corresponding available disk transfer time, network transfer time and buffer space in the system. If the next round failed this test, it is repeated until the first future round is found, where the requested playback can be started with guaranteed sufficiency of resources.

The lookahead distance \( H_l \) was defined as the number of future rounds that are considered as candidate rounds for initiating the stream for each request before it is rejected. Playback requests not accepted were turned away rather than being kept in a queue. Practically, a large lookahead distance allowed a long potential waiting time for the initiation of the playback. It cannot be unlimited in order for the service to be acceptable by the users. On the other hand, setting the lookahead distance too small can prevent the system from reaching full capacity.

The basic lookahead distance \( H_l \) was set to be equal to the mean number of rounds between request arrivals, \( H_l = 1 / \lambda \). Setting \( H_l = H_\alpha \) allows the system to consider for admission control the number of upcoming rounds that will take (on average) for another request to arrive. More generally, a lookahead factor \( F_\alpha \) is defined as the fraction \( F_\alpha = H_\alpha / H_l \).

As the basic performance metric, the expected number of active playback sessions that can be supported by the server was chosen. The objective was to make this number as high as possible.
Setup

Six different VBR MPEG-2 streams of 30 minutes duration each were used. Each stream had 54,000 frames with a resolution of 720x480 and 24 bit color depth, 30 frames per second frequency, and an IBPBPBPBPBPBP 15 frame Group of Pictures structure. The encoding hardware that was used allowed the generated bit rate to take values between 1 Mbps and 9.6 Mbps. Although the MPEG-2 specification allows bit rates up to 15 Mbit/sec, there is a typical point of diminishing returns (no visual difference between original and compressed video) at 9 Mbit/sec. The DVD specification sets a maximum allowed MPEG-2 bit rate of 9.8 Mbit/sec. Statistical characteristics of the clips are given in Table 1, where the coefficients of variation (CoV) lie between 0.028 and 0.383, depending on the content type. In the mixed basic benchmark, the six different streams were submitted in a round-robin fashion. Where necessary, the results from individual stream types are also shown.

<table>
<thead>
<tr>
<th>TABLE 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Content Type</strong></td>
</tr>
<tr>
<td><strong>Science Fiction</strong></td>
</tr>
<tr>
<td><strong>Music Clip</strong></td>
</tr>
<tr>
<td><strong>Action</strong></td>
</tr>
<tr>
<td><strong>Talk Show</strong></td>
</tr>
<tr>
<td><strong>Adventure</strong></td>
</tr>
<tr>
<td><strong>Documentary</strong></td>
</tr>
</tbody>
</table>

For the measurements, Seagate Cheetah ST-34501 SCSI disks were assumed, with the features shown in Table 2. Except for the storage capacity, which can reach 73GB in the latest models, the rest of the performance numbers are typical of today’s high-end drives. The logical block size BI was set to 16,384 bytes, while the physical sector size Sp was equal to 512 bytes. The slack used to store 8MBs in the disk space allocation was set to 1,572,864 bytes. The server memory was organized in buffers of fixed size BI=16,384 bytes each, with a total space of 64 MB for every extra disk. The available network bandwidth was assumed to be infinite.

<table>
<thead>
<tr>
<th>TABLE 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Seagate Cheetah ST-34501</strong></td>
</tr>
<tr>
<td><strong>Data Bytes per Drive</strong></td>
</tr>
<tr>
<td><strong>Average Sectors per Track</strong></td>
</tr>
<tr>
<td><strong>Data Cylinders</strong></td>
</tr>
<tr>
<td><strong>Data Surfaces</strong></td>
</tr>
<tr>
<td><strong>Zones</strong></td>
</tr>
<tr>
<td><strong>Buffer Size</strong></td>
</tr>
<tr>
<td><strong>Track to Track Seek (read-write)</strong></td>
</tr>
<tr>
<td><strong>Maximum Seek (read/write)</strong></td>
</tr>
<tr>
<td><strong>Average Rotational Latency</strong></td>
</tr>
<tr>
<td><strong>Internal Transfer Rate</strong></td>
</tr>
<tr>
<td><strong>Inner Zone to Outer Zone Burst</strong></td>
</tr>
<tr>
<td><strong>Inner Zone to Outer Zone Sustained</strong></td>
</tr>
</tbody>
</table>

The round time was set equal to one second. A warm up period of 3,000 rounds was used and calculated the average number of active streams from round 3,000 to round 9,000. The measurements were repeated until the half-length of the 95% confidence interval was within 5% of the estimated mean value of the active streams. Example on Fixed-Grain Stripping

In respect of Fixed-Grain Stripping, an important feature of this method is the ability to control the disk access efficiency through the choice of block size BI. As the block size is increased, a larger part of each access is devoted to data transfer rather than mechanical movement overhead. When a stream requests more than one block from a particular disk during a round, a maximum of two contiguous accesses is sufficient with the stride-based disk space allocation used. Referring to FIG. 5, there is shown a graph of number of streams versus load under Fixed-Grain Stripping where the number of active streams with sixteen disks and the mixed workload increases linearly as the load, P, increases from 10% to 50%. At loads higher than 50%, the number of streams that can be supported no longer increases. Since increasing the lookahead factor (F1) from 1 to 30 improves the number of streams that can be supported only marginally, for the rest of the experiments, the lookahead factor F1 was set to 1. This corresponds to a lookahead distance of less than 10 rounds, for a system of sixteen disks operating at load P=80%, and half-hour clips of 1 GByte each.

Referring to FIG. 6, there is shown a graph of a ratio of total number of rejected streams over total number of accepted streams versus load under Fixed-Grain Stripping, where the additional load beyond 50% translates into a corresponding increase in the number of rejected streams.

Referring to FIG. 7, there is shown a graph of number of streams versus block size BI under Fixed-Grain Stripping.

For load values p=40% and p=80%, the number of active streams were measured as the block size increased from BI=32,768 to BI=1,048,576 bytes at steps of 32,768. As can be seen from FIG. 7, at load 80% the number of streams initially increases until BI becomes equal to 327,680 and then drops. A similar behavior is noticed at 40%, although the variation in the number of streams is much smaller across different block sizes.

The Admission Control mode that was used for the above experiments allowed the gathering of statistics on system resources reserved for each round during the admission control process.

Referring to FIG. 8, there is shown a graph of round disk time versus block size BI under Fixed-Grain Stripping. In particular, FIG. 8 depicts the maximum and average access time Tdisk(i,k) that was reserved during the measurement period 3,000<=c<=9,000 for a particular disk (k=0) in a sixteen disk configuration with load P=80%. While the maximum time remains close to 100% across different block sizes, the average time drops from about 90% at BI=32,768 to less than 50% at BI=1,048,576.

With the round time set to 1 sec, the average time (normalized by the round time) corresponds to the expected disk utilization and varies depending on the number of disks accessed for a stream every round. Part of it was actuator overhead and decreased as the block size becomes larger. On the other hand, the maximum difference in reserved access times in a round (Avg Diff in FIG. 8) increased on average from almost zero to above 60%, with increasing block size BI. This could be another reason for the decrease in the average reserved time for larger block sizes.

It was also found that the average reserved time (shown in FIG. 8 only for Disk 0) remains about the same (typically within 2%) across a disk array. Thus, the access load, on average, was equally distributed across the disks, despite variations from round to round.

Referring to FIG. 9, there is shown a graph of round disk time versus number of disks under Fixed-Grain Stripping. As the number of disks increases, the average time drops only slightly from 59% with 8 disks to 67% with 16 and 66% with 32 disks. It is anticipated that the capacity of the system will increase almost linearly as more disks are added.
Refrerring to FIG. 10, there is shown a graph of number of streams versus number of disks under Fixed-Grain Stripping. The measurements were repeated varying the number of disks from 4 to 64. The block size Bf, that maximized the number of streams, was found to remain at Bf=327,680. At 80% load, the number of streams that could be supported increased from 39.17 with 4 disks to 143.57 with 16 disks and 550.23 with 64 disks. This is within 9–14% of what perfectly linear scalability should achieve. With the load at 40%, the number of streams increased from 30.31 with 4 disks to 504.79 with 64 disk, thus reflecting the improved capacity of the system with increased number of disks at low loads.

With Fixed-Grain Stripping, the mixed workload the number of streams is maximized at Bf=327,680 across different number of disks and system load values.

Example of Variable Grain Stripping

Refrerring to FIG. 11, there is shown a graph of number of streams versus load under Variable Grain Stripping. As shown, the performance of Variable-Grain Stripping on sixteen disks is shown as the load increases from 10% to 100%. The number of streams grows linearly as the load increases up to 70%. This is significantly higher than the 50% load, where Fixed-Grain Stripping flattened out (FIG. 5). As before, a lookahead factor value of Fl=1 attains more than 95% of the system throughput, and that is the value that is used.

Refrerring to FIG. 12, there is shown a graph of a ratio of total number of rejected streams over total number of accepted streams versus load under Variable Grain Stripping. As shown, only loads higher than 70% with Variable Grain Stripping increases the number of rejected streams.

Refrerring to FIG. 13, there is shown a graph of round disk access time versus number of disks under Variable Grain Stripping. As the number of disks increases, the average reserved time increases from 83% with 8 disks, to 84% with 16 disks, and 85% with 32 disks. The maximum number of sustained streams with 4 to 64 disks was also measured. At a load of 80%, the number of streams increases from 48.11 with 4 disks, to 202.69 with 16 disks and 786.05 with 64 disks. Thus, as the number of disks increases, the number of streams remains within 3% of what perfectly linear scalability should achieve. In addition, the advantage of Variable-Grain Stripping over Fixed-Grain Stripping increases from 23% with 4 disks to 43% with 64 disks.

Refrerring to FIG. 14, there is shown a graph of number of streams versus individual stream types in both Fixed-Grain versus Variable-Grain Stripping. As the content type changes from Science Fiction to Documentary and the variation in data transfers correspondingly drops, the block size has to be larger in order to maximize the performance of Fixed-Grain Stripping. However, the performance remains about the same for the five stream types, and increases only with the Documentary stream. In contrast, Variable-Grain Stripping manages to transform even minor decreases in data transfer variation into improved performance. Overall, Variable-Grain Stripping maintains an advantage over Fixed-Grain Stripping between 11% and 50%.

Validity is Simulated Disk Mode

In order to keep the computation time reasonable, the previous work were conducted with the system in Admission Control mode, where playback requests arrive leading to corresponding resource reservations, but without actual time measurement of the individual disk transfers. The statistics of the disk time resource reservations is compared with the statistics gathered over the access times of all individual data transfers involved, using the DiskSim representation of the Seagate Cheetah ST-34501 disk. A two-disk array model is used with each disk attached to a separate 20 MB/sec SCSI bus, and no contention assumed on the host system bus connecting the two SCSI buses. The statistics are gathered during 6,000 rounds after a warmup period of 3,000 rounds, as before. The mixed workload is used with average number of active streams 21.23 and 23.27 for Fixed-Grain and Variable-Grain Stripping, respectively, corresponding to 80% load.

Refrerring to FIG. 15, there is shown a graph of round disk time in Simulated Disk Mode. As can be seen from FIG. 15, in both the average and maximum case, the reserved disk time is no more than 8% higher than the corresponding measurements using the DiskSim. The difference can be attributed to the fact that the reservation assumes a minimum disk transfer rate and ignores on-disk caching.

Effect of Technology Improvements

To project disk technology improvements for the foreseeable future, the compound growth rates from the past are extended linearly into the future (Table 3). In particular, a 30% increase in internal disk transfer rate per year, and 23% decrease in seek distance is used. The full seek time depends linearly on seek distance, so the decrease is also 23%. However, a decrease of 12% per year for the track seek time is also assumed, which is dependent on the square root of the seek distance (among other factors more complex to project including settle time). Finally, a rotation speed increase of 12% per year is assumed. The stream types and sizes remaining the same.

<table>
<thead>
<tr>
<th>Disk Parameter</th>
<th>Today</th>
<th>2 Years</th>
<th>5 Years</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min Transfer Rate (MB/sec)</td>
<td>11.3</td>
<td>19.10</td>
<td>41.92</td>
</tr>
<tr>
<td>Max Seek Time (msec)</td>
<td>18.2</td>
<td>10.74</td>
<td>4.91</td>
</tr>
<tr>
<td>Avg Track Seek Time (msec)</td>
<td>0.98</td>
<td>0.76</td>
<td>0.51</td>
</tr>
<tr>
<td>Avg Rotation Latency (msec)</td>
<td>2.09</td>
<td>2.38</td>
<td>1.70</td>
</tr>
</tbody>
</table>

The above compared Fixed-Grain Stripping to Variable-Grain Stripping, which is a special case of Group-Grain Stripping at G=1. With current disk technology, having G=1 maximizes the number of streams. But as the disk access time drops, it is beneficial to increase G, so that G rounds worth of stream data are transferred in a single round. Specifically, when using the mixed workload, it is anticipated that two years into the future, the number of streams that could be supported with Group-Grain policy at G=2 increases by 35% when compared to Fixed-Grain Stripping. Five years into the future, the corresponding benefit of Group-Grain Stripping at G=3 remains at 29%. Thus, under reasonable technological improvements, there are significant performance improvements when using Group-Grain Stripping instead of Fixed-Grain Stripping.

Although preferred embodiments of the invention have been described herein, it will be understood by those skilled in the art that variations may be made therefer without departing from the scope of the invention. While this invention has focused on the stripping problem for the common case of sequential playback of video, it will be understood by those skilled in the art that variations beyond the playback of video, such as, the download of a file or any other material or any stream data, may be made thereto without departing from the scope of the invention.

Although preferred embodiments of the invention have been described herein, it will be understood by those skilled
in the art that variations may be made thereto without departing from the scope of the invention or the appended claims.

What is claimed is:

1. A method of storage device space management for storing stream data on storage devices of a streaming server where stream data transfers occur in rounds of fixed duration, comprising
   dividing the stream data into segments of stream data where each of the segments comprises an amount of stream data retrieved from the storage space of one storage device in response to one retrieval request;
   allocating storage space of the storage devices in strides where each of the strides comprises a contiguous chunk of storage space and that each of the strides is sized for storing at least one segment; and
   storing the segments in the strides.
2. The method of claim 1, wherein the strides are of a predetermined fixed size.
3. The method of claim 1, wherein the strides are of variable sizes.
4. The method of claim 1, wherein stream data is stored on the disks using fixed-grain striping.
5. The method of claim 2, wherein stream data is stored on the disks using fixed-grain striping.
6. The method of claim 3, wherein stream data is stored on the storage devices using fixed-grain striping.
7. The method of claim 1, wherein stream data is stored on the storage devices using group-grain striping.
8. The method of claim 2, wherein stream data is stored on the storage devices using group-grain striping.
9. The method of claim 3, wherein stream data is stored on the storage devices using group-grain striping.
10. An array of storage devices for a streaming server having storage space management for storing stream data, where stream data transfers occur in rounds of fixed duration, the storage device space management comprising
    dividing the stream data into segments of stream data where each of the segments comprises an amount of stream data retrieved from the storage space of one storage device in response to one retrieval request;
    allocating storage space of the storage devices in strides where each of the strides comprises a contiguous chunk of storage space and that each of the strides is sized for storing at least one segment; and
    storing the segments in the strides
    wherein at least one segment of stream data is retrieved for transfer during one of the rounds.
11. The storage device space management of claim 10, wherein the strides are of a predetermined fixed size.
12. The storage device space management of claim 10, wherein the strides are of variable sizes.
13. The storage device space management of claim 10, wherein stream data is stored on the storage devices using fixed-grain striping.
14. The storage device space management of claim 11, wherein stream data is stored on the storage devices using fixed-grain striping.
15. The storage device space management of claim 12, wherein stream data is stored on the storage devices using fixed-grain striping.
16. The storage device space management of claim 10, wherein stream data is stored on the storage devices using group-grain striping.
17. The storage device space management of claim 11, wherein stream data is stored on the storage devices using group-grain striping.
18. The storage device space management of claim 12, wherein stream data is stored on the storage devices using group-grain striping.

* * * * *