Storage Strategies for Fault-Tolerant Video Servers

E.N. Elnozahy

August 1996
CMU-CS-96-144

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Future versions of this document may be submitted for publication in journals or conferences.

This work is supported in part by a National Science Foundation Research Initiation Award under grant number CCR-9410116, and a National Science Foundation CAREER Award under grant number CCR-9502933, and by the Advanced Research Project Agency under contract DABT63-93-C-0054.

Keywords: Distributed systems, high-availability, MPEG-1, multimedia, QoS, video servers.

Abstract

We consider the problem of providing high availability in cluster-based video servers. The cluster acts as a parallel processor that provides the aggregate I/O and network bandwidths of the component machines. In such an environment, the failure of one server may affect the availability of the video service or its quality. Existing approaches to this problem fall into two categories.

On one hand there are RAID-like schemes that store error correcting code (ECC) in addition to the video data. Should a failure occur, the unavailable data can be computed on the fly using the ECC and the service continues at the same quality. In cluster-based systems, however, the video data are distributed over several servers and there is no convenient point to reconstruct the missing blocks except at the client. Relying on the client for this task is not desirable as it may not have the necessary buffering or processing capacity

On the other hand, some argue that the system could just continue operation without the need for the information on the unavailable host. The underlying assumption is that the human perception may tolerate the reduction in the frame rate that results from the temporary loss of data. This strategy, however, does not work well for compressed video.

In this paper, we argue that substantial improvements can be obtained by taking into account the nature of the compressed video stream. We present two approaches for storing MPEG-1 video that helps in tolerating partial failures in server clusters at the expense of marginal degradation in the quality of service. The first approach restricts ECC storage and computation to reference frames, ignoring low-information frames. The second approach uses the structure of the compressed video stream to substitute unavailable high-information frames with equivalent low-information frames.

1 Introduction

This paper describes strategies for storing video streams in a cluster-based video server. The cluster consists of several servers, each having its own storage subsystem. Such a configuration can be built out of off-the-shelf components, reducing cost and development efforts [10]. Previous research has established the viability of this approach through several analytical studies and prototyping efforts [5]. Server clusters, however, are susceptible to partial failures when one or more component servers fail. A solution to this problem is to use parity encoding in a manner similar to RAID storage systems [2, 7, 13]. In this paper, we argue that although using parity encoding to provide high availability in RAID systems may be a desirable strategy [3], it is not the best alternative for cluster-based servers. In RAID systems, the disk array controller has access to the entire data. Therefore, it is a logical place for efficiently reconstructing the unavailable data should a disk fail. In a cluster-based server, however, no single server has direct access to the entire data. Therefore, either the client has to retrieve the parity information from the cluster and reconstruct the unavailable data, or some designated server has to perform this task. Relying on the client to reconstruct the unavailable data requires extra buffering and processing capacities, which may not be available. On the other hand, relying on a designated server to reconstruct the missing data reduces the network bandwidth available to the cluster by 50%. It also causes load surges on the surviving servers, making scheduling and resource allocation more complicated. We argue that a better alternative is to use storage strategies that take into account the nature of the video stream and video access patterns.

We present three strategies for storing MPEG-1 video streams and compare them to Simple Parity Encoding (SPE). We used MPEG-1 because of the availability of a public domain software codec, and the relevance of this standard in commercial systems. The first storage strategy selectively computes parity blocks for some but not all video frames. We call this strategy MPEG-Aware Parity Encoding (MAPE). Compared to SPE, MAPE has a lower storage overhead, causes less load surges after a failure, and requires less buffering at clients. The quality of the picture is slightly affected after a failure. Fortunately, the effect is hardly noticeable to human perception.

The second storage strategy allows the cluster to continue operation after a failure occurs with a lower Quality of Service (QoS). We call this strategy Quality-of-Service Based Storage (QBS). Unlike SPE and MAPE, QBS does not use parity at all. Therefore, QBS does not require client machines to possess extra resources or participate in recovery actions. QBS uses the characteristics of MPEG-1 streams in some unconventional ways to achieve these goals. Fortunately, the resulting degradation in service that we observed is hardly noticeable [4].

A third strategy that we considered uses striping at a very fine granularity within each video frame. This strategy ensures that if a failure occurs, no frame will be entirely lost. Some frames may have missing portions but the quick successions of video frames would cover the missing portions. While the idea is intresting, our experience with this technique has shown that the resulting degradation in the quality of service is unacceptable. We attribute this result to two factors. First, the relatively large block size within MPEG-1 frames makes it difficult for the human eye to ignore the checkerboard effect that results from missing some of these blocks on the screen. Furthermore, because of the inter-frame encoding nature of the video stream, subsequent frames may compute erroneous picture translations or may not be able to compute sufficient blocks to cover the previously missing portions.

The paper shows that storage strategies that exploit the nature of the video stream are often better than techniques that blindly apply parity encoding. They use less storage overhead, or may impose less requirements on clients, at the expense of marginal degradation in the quality of service during failures.

The paper is organized as follows. Section 2 reviews the salient features of MPEG that are relevant to this paper, and Section 3 discusses availability issues in the context of MPEG video streams. Section 4 describes SPE and MAPE storage strategies. Section 5 describes QBS, and Section 6 describes a Macroblock striping technique. We compare our research to related work in Section 7 and conclude the paper in Section 8.

2 MPEG-1 Compression

MPEG-1 is a standard intended for lossy video compression and decompression with quality comparable to that of VHS video cassette recorders [1]. It uses inter-frame encoding to reduce temporal redundancy and intra-frame encoding to reduce spatial redundancy. This method achieves typical bit rates for compressed video that range between 1.2 Mbit/sec to 2 Mbit/sec for 352×240 picture resolution at 30 frames/sec [6]. A complete review of the compression standard is beyond the scope of this paper. We focus only on the salient features of the standard that affect our work.

2.1 General Description

MPEG-1 classifies the frames in a video stream into three types: Intrapictures (I frames), Predicted pictures (P frames), and Interpolated Pictures (B frames). The names indicate the type of compression each frame type undergoes. The encoder uses only spatial redundancy to compress I frames. The error due to lossy compression for I frame is small since no inter-frame encoding is used. I frames thus have high picture quality at the expense of moderate compression ratios. These frames facilitate random access to the video stream (such as in fast forward and video indexing). They also serve as reference points for inter-frame encoding for P and B frames. P frames use inter-frame encoding based on motion prediction with reference to a previous I or P frame. This technique yields better compression ratio, but the error due to the lossy compression is higher. B frames provide the highest compression ratio by interpolating the difference between a past and a future reference frame (I or P). B frames never serve as reference. Figure 1 shows a typical interleaving of an MPEG-1 video stream. For example, the first frame uses intra-picture encoding and does not depend on any other frame; the second frame encodes the second picture as a function of the difference between the first frame (an I frame) and the fourth (a P frame); and the fourth frame encodes the fourth picture based on the difference with the first one.

The combination of these frame types allows tradeoffs between picture quality and compression performance. Increasing the number of B frames at the expense of reference frames yields better compression efficiency and arguably lower picture quality. This tradeoff is true up to a limit. The correlation between reference frames decreases with the temporal distance between them. A high correlation is important for the interpolation techniques used in compressing B frames to yield efficient compression.

Video compressors usually produce MPEG streams according to a user-specified pattern. A "recommended" pattern that is often used is [6, 1]

$$IBBPBBPBBPBB$$

This pattern represents a reasonable tradeoff between quality, compression ratio, and flexibility regarding random access requirements. It guarantees that there is an intrapicture every 0.4 second, and a reference frame every 0.1 second. The remaining B frames use interpolation for maximum compression efficiency.

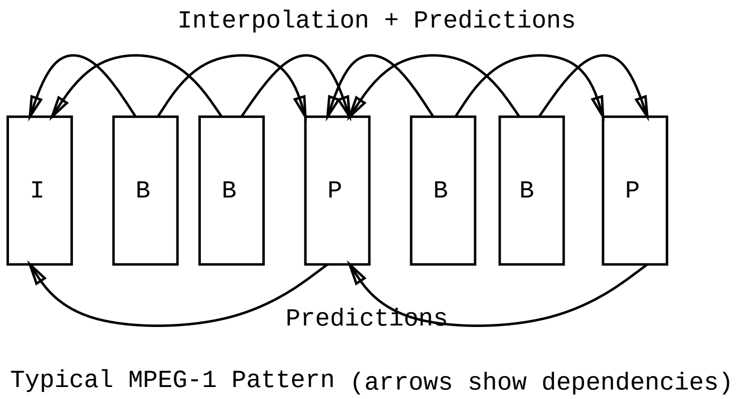

Figure 1 A typical interleaving of an MPEG-1 stream.

2.2 Motion compensation

MPEG-1 uses motion prediction in compressing P frames and a combination of prediction and interpolation in compressing B frames. For this scheme to work, each frame is subdivided into *macroblocks* of size 16×16 each. For each macroblock in a P frame, the compression algorithm searches the previous reference frame for a macroblock with a high correlation with the macroblock to be compressed If the search is successful, the compressor computes a suitable motion vector that serves as the compressed representation of the macroblock. If the search is unsuccessful, the macroblock is intra-coded. For B frames, the compression algorithm searches the previous and the following reference frames for macroblocks that have high correlation with the macroblock to be compressed. The compression algorithm computes a suitable motion vector if a high correlation is found in either direction, or if the current macroblock can be expressed as an interpolation between two macroblocks in the previous and following reference frames. Otherwise, the macroblock is intra-encoded. Motion vectors are very compact and are the main factor in obtaining good compression ratios. The search, however, has to use heuristics to limit the processing power required. More exhaustive search can yield better picture quality and compression. It is therefore the case that the compression and decompression efforts are highest for B frames and lowest for I frames, with the compression ratio directly proportional.

2.3 Spatial Redundancy

MPEG uses the Discrete Cosine Transform (DCT) in intra-frame encoding which has two purposes: to eliminate the spatial redundancy in the picture for intra-frame encoded macroblocks, and to compress the motion predictors for inter-frame encoded macroblocks. To eliminate spatial redundancy, each macroblock is subdivided into four 8×8 blocks and the DCT is computed for each. To compress motion predictors, the DCT is computed for the entire macroblock. These two operations produce a frequency-representation of the picture in the form of DCT coefficients. The high frequency component are eliminated since they contribute little to the information content of the picture. The cutoff depends on the desired quality. The processed transform is then run-length encoded using Hoffman compression.

2.4 Client Considerations

The MPEG standard is asymmetric. The compression process is relatively expensive while decompression has low processing and resource requirements. This design has several motivations. It allows clients with varying degrees of capabilities to perform decompression. Decompression could be performed using VLSI chips, and therefore the decompression logic should be simple to allow economic chip implementations. It could also be done in software, and therefore it should be implementable with general purpose processors with acceptable efficiency. The design also recognizes that economic factors may force limitations on the available resources to a client machine such as settop boxes in video-on-demand services. These devices are limited in their buffering and processing capacities [9], and their functionality is limited to implementing the decompression algorithm and a simple user interface.

2.5 MPEG Syntax

MPEG-1 defines a syntax for compliant video streams. Following this syntax guarantees that any compliant decoder would be able to reconstruct the original picture (albeit with some degradation of quality, due to the lossy compression). Two parts of this syntax are relevant to our work. First, the syntax contains overhead information in the form of headers that locate the various frames and can be used for indexing. This overhead information do not contribute to the information within the pictures. It is merely used to guide the decoder in parsing the bit stream. The overhead is relatively small (less than 5% of the total size in all cases studied, and much lower than that on average). Due to the importance of this information, we replicate it on every server in the cluster.

The second relevant part of the MPEG-1 syntax is that it groups macroblocks within a frame into slices. A slice can be as large as to contain all macroblocks within a frame or as small as to contain only one block. The slice serves as a unit for decompression. All motion references are reset whenever a new slice is encountered during parsing. Thus, the beginning of each slice can act as an escape point in case an error occurs during the transmission of the bit stream. Also, it can be used to confine errors within a slice boundary.

3 MPEG and Fault Tolerance

Several researchers have observed that reliability is not a major concern in the transmission of raw video. The servers or the communication system may drop up to 6 frames/sec in a 30 frames/sec video stream before the human eye notices the quality degradation. The same argument applies also to compression schemes that rely on intra-frame encoding, such as the JPEG standard for still picture compression [15]. MPEG-1, however, relies on inter-frame encoding. Thus, the loss of a reference frame renders useless the retrieval and transmission of the frames that depend on it for motion interpolation or prediction. Consider an MPEG-1 video stream consisting of the sequence:

$$I_0 B_0 B_1 P_0 B_2 B_3 P_1 B_4 B_5 I_1$$

The loss of frame P_0 would make the retrieval and transmission of frames B_0, B_1, B_2 and B_3 useless as they depend on frame P_0 for motion interpolation and prediction. Furthermore, P_1 also depends on P_0 for motion prediction, rendering the retrieval and transmission of P_1, B_4 and B_5 useless. Thus, the loss of one reference frame may have a *domino effect* that extends over several frames. The loss of non-reference frames, however, can be tolerated in the sense that this domino effect does not occur.

4 Parity-Based Solutions

4.1 Simple Parity Encoding (SPE)

Parity encoding is a standard technique for providing high availability in disk arrays [3]. The Zebra file system extends this technique to cluster-based file servers [7]. We study here the various tradeoffs of using parity encoding in video servers. A straighforward solution to the problem of providing highly available video streams is to use parity encoding as in the Zebra file system. This solution treats video streams as ordinary data. However, care must be taken while striping video streams across the server cluster to reduce the overhead in the case a failure occurs [8]. Simple parity encoding schemes increase the communication and disk access loads on each server during failure recovery by a 100% [8]. Fortunately, the servers can exploit the sequential nature of video access to eliminate this load surge. To do so, each parity group should consist of video data that are contiguous in temporal order. If a server fails, the only surge in communication and disk loads will be due to reading and transmitting the parity blocks. The client reads the remaining data within the parity group anyway because of the sequential access. A system consisting of 8 servers [11], for instance, would experience about a 14.3% load surge if one server fails. Parity encoding can be extended in a straightforward manner to tolerate more than one failure in the server cluster.

Evaluation

SPE is simple to implement and maintain. For a cluster of N servers, SPE requires a storage overhead of $100/(N-1)\%$. Each server must also have spare capacity in disk and network bandwidths of $100/(N-1)\%$ to avoid performance failures if one server fails in the cluster. SPE, however, has one undesirable property when deployed in a server cluster.

When a server fails, SPE allows the reconstruction of the missing blocks from data that are distributed over several servers. This problem does not occur in RAID systems since the entire parity group is available to the disk controller. In a server cluster, relying on the servers to reconstruct the missing blocks after a failure is not acceptable. In such a setup, a server would have to receive over the network all the blocks in a parity group to reconstruct the missing blocks. Assuming perfect load balancing, this setup increases the communication load on each server by $(N-1)*100/(N-1) = 100\%$ during failure recovery. Relying on the client is thus the natural choice for reconstructing the missing blocks since it would not require any additional communication overhead. However, it would require that the client multiply its buffering capacity by N, and the client software would be complicated by the need to recognize the parity scheme and compute the missing blocks. Some client platforms, such as portable computers, may not possess the buffering capacity or the processing power needed for recovery. Furthermore, commercial video-on-demand systems use settop devices as clients [9]. These devices are constrained in processing power and resources for economic reasons. Thus, the need to rely on the client for reconstruction may not be always desirable.

4.2 MPEG-Aware Parity Encoding (MAPE)

The first solution that we present in this paper is a simple extension of SPE that recognizes the nature of the MPEG-1 video stream. This solution recognizes that I and P frames contribute most to the quality and information content of a picture. Also, these being reference frames, special care must be taken to avoid their loss. B frames, on the other hand, can be lost without causing domino effects.

Each MPEG-1 video stream is divided into two substreams. The first contains the I and P frames in the same order as in the original stream. This substream is striped over the servers with parity blocks as in SPE. The second substream contains the B frames. No parity encoding is used for this substream. Furthermore, the B frames are distributed over the cluster servers in a round robin manner, such that the i^{th} server contains all frames B_j where $j \bmod i = 0$. Figure 2 shows the general scheme for 8 servers, and Figure 3 details the distribution of the B frames across the servers. Thus, MAPE uses parity to preserve the availability of the I and P frames. Some B frames will be unavailable, however the round robin allocation of B frames across the servers prevent this unavailability from appearing as the loss of several contiguous frames, which would be noticeable to the human eye.

MAPE can use standard RAID techniques such as distributing the parity blocks among all servers, or parity declustering [8]. Data also should be stored to exploit the maximum disk bandwidth. As such, the substreams should share the disk units of retrieval. In this manner, no additional buffering would be required compared to SPE.

Evaluation

The behavior of this solution depends on the nature of the video stream. To make the evaluation concrete, we consider the "recommended" pattern [1]:

$$IBBPBBPBBPBB$$

Assume that the ratios of the frame sizes are $9:3:1$ for I, P and B frames in that order. These ratios are consistent with what we observed in many streams. Thus, for the above GOP the relative sizes of the IP-substream to the B-substream would be 18:8.

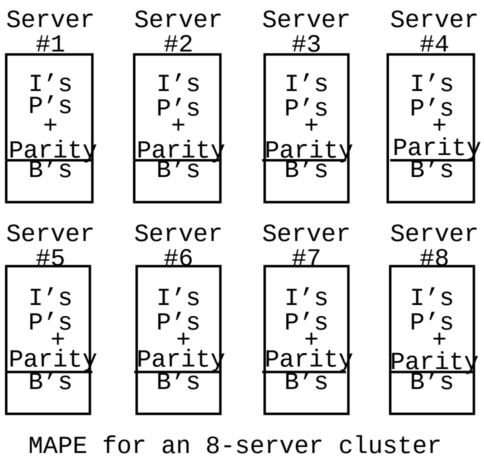

Figure 2 The MAPE storage strategy.

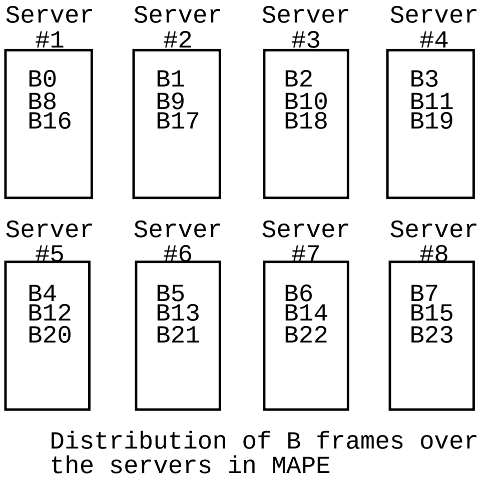

Figure 3 Distribution of B frames in MAPE storage strategy.

Assuming the above figures, the storage overhead required for a cluster of N servers would be $0.7 * 100/(N-1)\%$. If a failure occurs, the surge in disk and communication loads for each surviving server would be $0.7*100/(N-1)\%$. Unlike SPE, however, the failure in one server causes a degradation in picture quality since some B frames will not be transmitted. The frame rate is reduced from 30 frames/sec to $30 - 20/N$ frames/sec. Furthermore, the round robin allocation of B frames ensures that no perceivable gaps will result from missing some of these frames due to a failure. Thus the resulting frame rate with one server failure is 25 frames/sec for $N = 4$, and 27.5 frames/sec for $N = 8$. There is ample evidence that this slight loss in the frame rate will not be noticeable to the viewer [4].

MAPE requires the same amount of buffering as SPE, provided that the substreams are allowed to share the same disk blocks. MAPE, however, requires that the client reconstruct the missing blocks in the IP-substream after a failure. Table 1 summarizes a comparison between MAPE and SPE.

5 A QoS-Based Solution

The MAPE storage strategy improves over SPE by exploiting the nature of the video stream. The client, however, still needs extra buffering and processing capacity to reconstruct missing blocks if a server fails. We take the concept of exploiting the nature of the video stream one step further to eliminate the need for parity altogether. The idea is to allow the Quality of Service (QoS) to degrade if a server fails. Ideally, the degradation should be sufficiently low that the end user would not perceive the difference. Less than ideally, the strategy would be considered successful if the level of operation during failure is acceptable to the end user.

	SPE	MAPE
Server buffering	1 disk retrieval unit	1 retrieval disk unit
Storage overhead	$100/(N-1)\%$	$70/(N-1)\%$
Failure Overload	$100/(N-1)\%$	$70/(N-1)\%$
Client involved in recovery	required for I,P and B	required for I and P only

Table 1 Comparison between MAPE and SPE. Assume $I : P : B$ size ratio to be $9 : 3 : 1$

5.1 Description

The idea behind this strategy is to provide alternative B frames for all reference frames. Each alternative frame is computed using the previous and following reference frames. Also, each alternative frame is stored on a different server than the reference frame it shadows. If a reference frame is unavailable due to a server failure, the alternative B frame is retrieved and sent to the client. The alternate B frames replace parity blocks as a way to providing high availability. Thus clients are no longer required to process parity blocks, or have extra buffering capacity. Moreover, since B frames have the best compression ratio, they incur low storage overhead. On the other hand, the quality of the video stream degrades when a failure occurs since the compression of a B frame loses more information on average than a corresponding reference frame. Furthermore, since the alternate frame is computed out of two more distant reference frames, the degree of correlation between the two frames is lower. Thus, either the size of the alternate frame will be higher because more macroblocks will be intra-encoded, or the quantization factor has to be more coarse, degrading the picture quality even more. Fortunately, the average distance between every other reference frame is 5 frames in the "recommended" pattern. This relatively small distance mitigates the problems that we mentioned in the movies that we experimented with.

The "actual" B frames are still stored in a manner similar to the MAPE storage strategy. Replicating these frames is not likely to improve the video stream's quality, and would waste disk space needlessly. Note that the MPEG decompressor must be modified to recognize the replacement B frames. This modification is very simple and affects only the parsing of the input. This way, it would be possible for the decompressor to adjust the reference frames appropriately.

5.2 Storage Algorithm

A storage algorithm to implement the QBS strategy has to satisfy the following constraints:

- Any three consecutive reference frames should be stored on non-overlapping servers.
- Backup B frame must placed on a different server than the corresponding reference frame.

The first constraint is necessary so that if a failure occurs, the backup B frame can use the other two reference frames to compute the missing one. The second constraint should be obvious.

Round-Robin storage of reference frames among the available servers would satisfy the above two constraints. Care must be taken, however, to ensure load balancing. If the number of reference frames in a group of picture is a multiple or a divisor of the number of servers, then a situation may occur where I frames may cluster on a subset of the servers available (see Figure 4). Therefore, a staggered allocation would be preferrable in this case (see Figure 5).

5.3 Evaluation

QBS relieves clients from the need to reconstruct missing frames or participate in recovery actions. Thus, constrained and limited-capacity clients would prefer this method over either MAPE or SPE. The cost of this advantage is additional storage overhead, slightly higher load surge after a failure, and a degradation in the quality of service. It is interesting to note that QBS's storage overhead is independent of the number of servers. Instead, it depends on the video stream. See [4] for a table containing the relative sizes of the alternative B frames for a video portion taken from the movie "Jurrassic Park."[1]

6 Macro-Block Based Striping

We have also explored another strategy for relying on QoS reduction as a means for providing high availability. MAPE and QBS strategies deal with the MPEG-1 stream at the frame level. To explore the potential of considering storage policies at a lower level within the MPEG-1 syntax, we investigated a fourth strategy that deals with the MPEG-1 stream at a finer granularity than a frame. This strategy treats the MPEG-1

[1]Including all the information here is not possible for size limitations.

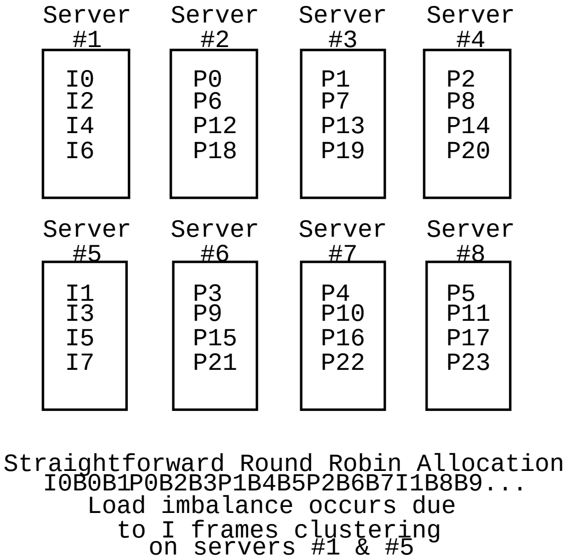

Figure 4 Clustering of I frames in simple Round Robin allocation.

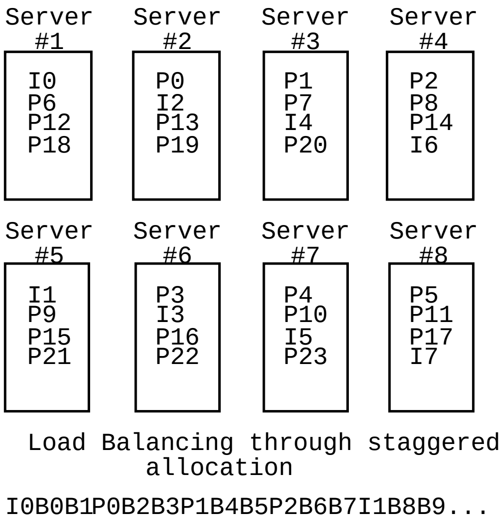

Figure 5 Load balancing through declustering of I frames in staggered allocation.

bit stream at the stripe level within each frame. The stripes are stored on the cluster servers in round robin order. Thus, the first server gets the first stripe in the first frame, the second server gets the second stripe, and so on. This allocation continues until all stripes of the first frame have been allocated, then the stripes of the second frames are considered until the entire video stream is stored.

The idea behind this strategy is to scatter the stripes of each frame over several servers. (See Figure 6). Thus, if a server fails, no single frame will be entirely lost. Some frames may be partially available, missing those stripes that were stored on the failed server. The partial data available would be used to reconstruct the frame, albeit at a lower quality. This design exploits the inherent properties of the MPEG-1 stream. Stripes are specifically designed as units of compression and escape points if a failure occurs in communicating the stream. Our approach described here is a straightforward application of this philosophy.

Frames that do not have any stripes on the failed server will be decompressed and displayed at the normal quality. This scheme also allows some flexibility in defining the granularity of the picture portions that are vulnerable to losses. By controlling the number of macroblocks in each stripe, we can control the granularity of the missing portions within a picture. At one extreme, we can assign one macroblock per slice. The picture would then miss several blocks at random locations. Alternatively, a number of consecutive blocks could be assigned to a single slice. This would result in stripes of unavailable blocks within the picture (see Figure 7). Two movie clips are available in public showing the quality of the picture that could be obtained under these two slice allocations [4].

While the idea seems interesting, the results were very disappointing. The size of a macroblock is too large for the eye to miss the checkerboard effect that the missed stripes introduce to the picture during a server failure. Furthermore, inter-frame encoding can make the loss of one macroblock in a reference frame affect the quality of the following frames in unpredictable ways. A partial solution for these problems is to look at the motion vectors within the MPEG stream and allocate storage such that inter-dependent macroblocks would be all allocated on the same server. Thus, if this server fails, all the related macroblocks are lost and no bogus translations may occur. Unfortunately, this solution requires a very complex allocation strategy. Furthermore, it is not clear that the resulting picture will get rid of the checkerboard effect. We therefore dismissed this strategy.

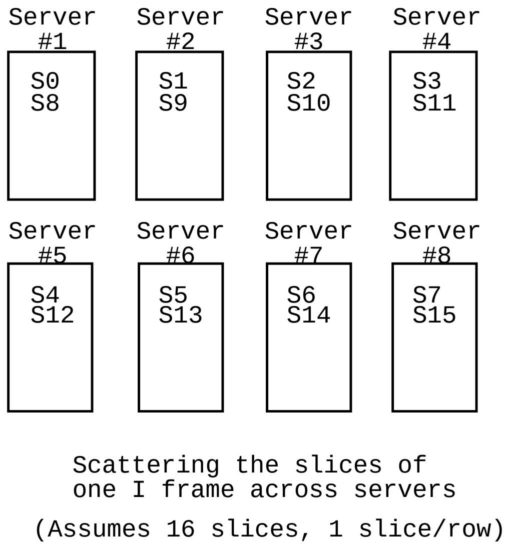

Figure 6 Scattering slices of a single frame across servers.

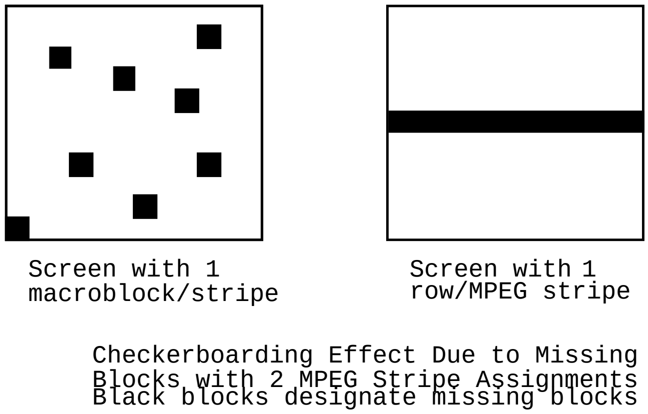

Figure 7 Effect of slice allocation on picture quality during failure.

7 Related Work

To our knowledge, most previous research has used replication or parity-based schemes in designing highly-available video servers [2, 8, 11, 13]. One exception is the work of Vin et al [14]. They described a system called Self-Recovering Array of Disks (SRAD) to provide highly available video access in the context of RAID storage systems. SRAD modifies the JPEG compression algorithm to split the picture into several subpictures in the frequency domain. The subpictures are stored on different disks such that if a failure occurs, a client can reconstruct the missing subpicture by extrapolating the available subpictures. The resulting quality is lower than could be obtained during failure-free operation, but it appears that the difference can hardly be noticeable. The reconstruction is integrated with the decompression algorithm. The SRAD work has a similar philosophy to our QBS strategy in exploiting the nature of the video stream to provide high availability instead of using parity or replication. Although it is presented in the context of RAID storage systems, applications of SRAD to server clusters are straightforward. There is an important difference though. SRAD handles only intra-frame encoded pictures. It is not clear if such a scheme could be applied with the same simplicity to inter-frame coded pictures. In contrast, our work handles both kinds of encoding by relying on the intrinsic characteristics of MPEG.

Extensive research has been devoted to understanding the behavior of RAID systems during failures and improving their availability [2, 8, 11, 13]. None of this work uses the nature of the data in improving high availability, although Patterson *et al* have presented a system that uses information about data access pattern to reduce access latency in general RAID systems [12].

8 Conclusions

We have presented two successful strategies for storing highly-available MPEG-1 video streams in server clusters. These two strategies exploit video access patterns and MPEG-1 encoding characteristics to reduce storage overhead and load surges during failures. The first strategy extends simple parity encoding to exclude B frames from pertaining to parity groups. This strategy results in lower storage overhead and load surges during a failure than SPE, at the expense of marginal degradation in picture quality. The second strategy uses additional highly-compressed frames to replace reference frames during a failure. Slight modification to the video decoder is required to recognize this unusual use of B frames. The storage strategy eliminates the requirements of having to involve clients in failure recovery. These stratgies support our thesis that exploiting the structure of the compressed video stream can yield high availability at low overhead.

References

[1] MPEG-1 VIDEO. http://www.crs4.it/OLD/LUIGI/MPEG/mpeg1-v.html. Available through the world wide web.

[2] S. Berson, L. Golubchik, and R.R. Muntz. Fault tolerant design of multimedia servers. In *SIGMOD Record. Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 364–375, May 1995.

[3] P. Chen, E. Lee, G. Gibson, R. Katz, and D. Patterson. RAID: High-perforamance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–186, June 1994.

[4] E.N. Elnozahy. Degradation in quality of video streams under different storage scenarios. http://cairo.sp.cs.cmu.edu/. Available through the world wide web, contains video clips and corresponding decoders to show picture degradation.

[5] C.S. Freedman and D.J. DeWitt. The SPIFFI scalable video-on-demand system. In *SIGMOD Record. Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 352–363, May 1995.

[6] D. Le Gall. MPEG: a video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, April 1991.

[7] J.H. Hartman and J.K. Ousterhout. The Zebra striped network file system. In *Proceedings of the 14th ACM Symposium on Operating Systems Principles*, pages 29–43, December 1993.

[8] M. Holland and G.A. Gibson. Parity declustering for continuous operation in redundant disk arrays. In *ASPLOS-V Proceedings. Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 23–37, September 1992.

[9] A. Laursen, J. Olkin, and M. Porter. Oracle media server: Providing consumer based interactive access to multimedia data. In *SIGMOD Record. Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, pages 470–477, May 1994.

[10] Microsoft. The tiger video server. Microsoft Press Release, April 1994.

[11] M.N. Nelson, M. Linton, and S. Owicki. A highly available, scalable ITV system. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, pages 54–67, December 1995.

[12] R.H. Patterson, G.A. Gibson, E. Ginting, D. Stodolsky, and J. Zelenka. Informed prefetching and caching. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, pages 79–95, December 1995.

[13] R. Tewari, D.M. Dias, R. Mukhrejee, and H.M. Vin. High availability in clustered multimedia servers. Technical Report RC 20108, IBM T.J. Watson Research Lab, June 1995.

[14] H.M. Vin, P.J. Shenoy, and S. Rao. Efficient failure recovery in multi-disk multimedia servers. In *Proceedings of the 25th International Symposium on Fault-Tolerant Computing*, pages 12–21, June 1995.

[15] G.K. Wallace. The JPEG: still picture compression standard. *Communications of the ACM*, 34(4):30–44, April 1991.