[Presented at the 7^{th} Workshop on Information Technologies and Systems (WITS '97), Atlanta, Georgia, December 13-14, 1997]

Scalable Video Browsing Techniques for Intranet Video Servers*

Shahram Ghandeharizadeh, Roger Zimmermann, Seon Ho Kim,
Weifeng Shi, Jaber Al-Marri

Computer Science Department
University of Southern California
Los Angeles, California 90089

Abstract

Servers that employ scalable techniques prevent the formation of bottlenecks in order to allow the system to support a higher number of clients as function of additional hardware. They are desirable because they enable a growing organization to satisfy the increased demand imposed on its hardware platform by increasing its size (instead of replacing it with a new one which is almost always more expensive [Sto86]). This study presents the design, implementation, and evaluation of scalable techniques in support of video browsing, i.e., VCR functions such as fast-forward and fast-rewind.

1 Introduction

With the convergence of computers and consumer electronic goods, e.g., digital cameras and camcorders, digital video has become as pervasive as any other data type such as text, records, objects, etc. An Intranet video server provides for storage, delivery, and distribution of video. These servers might contain training video for the employees of an organization, archive broadcasted speeches, video conferences between several people collaborating on a project, etc. In order to be effective, these servers must support browsing techniques for video. Both fast-forward and fast-rewind (with scan) are two such techniques. To illustrate, once a video conference is archived, one of the participants may want to browse the stored clip to recall a specific demonstration or discussion. This is realized using a fast-forward display of a clip to the point of interest. In addition, using this functionality, the user may construct marked indexes for future recall. Even with these marked indexes, browsing continues to be useful because it enables a user to study a segment carefully by moving back and forth in that segment.

*This research was supported in part by a Hewlett-Packard unrestricted cash/equipment gift, and the National Science Foundation under grants IRI-9203389, IRI-9258362 (NYI award), and ERC grant EEC-9529152.

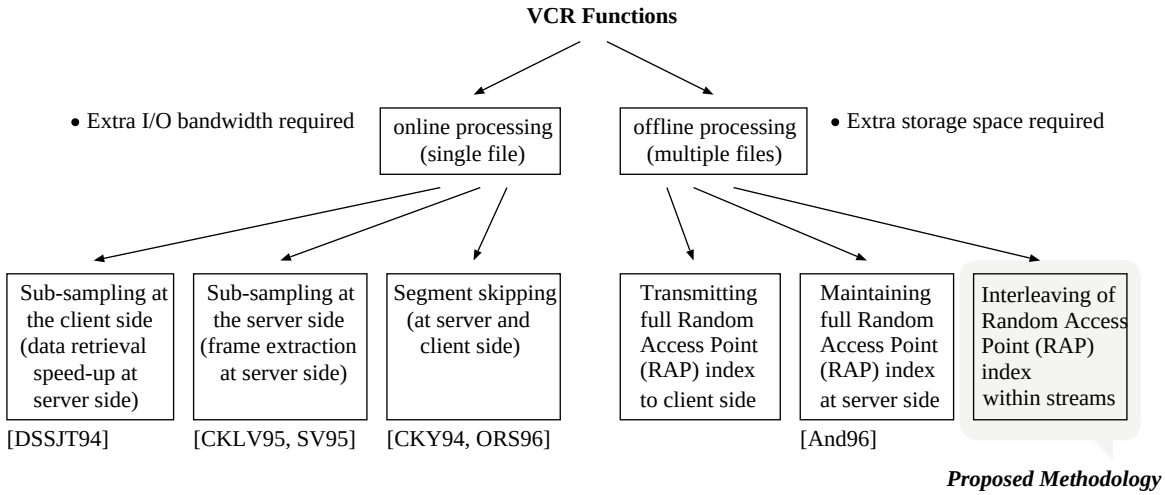

Figure 1: Taxonomy of methods to implement VCR functions in video servers

This study investigates the design and implementation of scalable techniques in support of fast-forward (FF) and fast rewind (FR). By scalable, we imply that there are no inherent bottlenecks and the number of active displays that can invoke FF/FR increases as a linear function of the available bandwidth. These designs minimize: (1) the amount of memory and processing overhead incurred at both the client and the server, (2) the communications overhead between the client and the server, and (3) the delay incurred when VCR functions are invoked. We start with a taxonomy of different approaches to implement VCR functions in order to compare our work with the previous literature. Subsequently, we detail our design.

Figure 1 distinguishes two basic paradigms[1] to provide VCR functions from a video server: either (1) a single, normal-speed movie is processed accordingly in real-time during playback time[2] (this paradigm is termed *online* processing), or (2) a clip is pre-processed by the content provider with separate files for FF and FR viewing (termed *offline* processing). The tradeoff between these two paradigms is I/O processing versus storage space. While online techniques require extra bandwidth (disk, network), offline techniques require additional disk storage. Each method has other advantages and disadvantages that might make it more appropriate for a specific application. We describe each method in turn.

[1] We do not consider *near* VoD systems for this taxonomy because they usually do not provide full VCR functionality ([BHH+94]).

[2] The processing may be done at the server side, the client side, or both.

• **Online Processing**

Sub-sampling at the client side [DSSJT94] allocates n times the playback bandwidth for n times fast viewing. This approach is associated with statistical quality-of-service (QoS) guarantees. With a low system load, its probability of providing immediate access to full-resolution FF and FR is high. When bandwidth is scarce due to a high system load, service is either delayed or provided by sacrificing resolution. This approach is conceptually simple but may need a complex decoder for high speed decompression or real-time frame reduction. Moreover, it may waste network and disk bandwidth.

Sub-sampling at the server side is conceptually similar to sub-sampling at the client side, except that the real-time frame reduction is performed at the server side. This approach does not increase the network bandwidth, however it does waste disk bandwidth. More sophisticated variations of this method can reduce this overhead (see [CKLV95, SV95]).

Segment skipping [CKY94, ORS96] skips a fixed number of blocks to achieve the desired playback rate by controlling the placement of blocks on disk drives. For example, to provide five times fast forward, this method displays one block out of five consecutive blocks. The advantage of this approach is its scalability because it requires no extra network and disk bandwidth to support various fast rates. However, it results in unusual previewing because it skips blocks, not frames.

• **Offline Processing**

[And96] proposed a method to implement separate fast-forward and fast-rewind files by selecting and pre-processing frames, either before or after compression. For example, to create a five times fast-forward file, every fifth frame is selected from the original movie before compression. This collection of frames is encoded in the regular manner (e.g., using MPEG) and stored in a separate file. Cross-references among the different files are maintained to enable a user to switch between versions. This method requires neither additional network nor extra disk bandwidth. Moreover, video quality is not degraded. However, it requires extra storage space for additional files. In Section 2 we detail our design and state our contributions.

The rest of this paper is organized as follows. In Section 2 we introduce the design of our technique, which is then detailed for Motion JPEG compressed videos in Section 3. Section 4 extends our approach for MPEG compressed videos. The performance of the proposed scheme is described in Section 5. We conclude with the future research directions in Section 6.

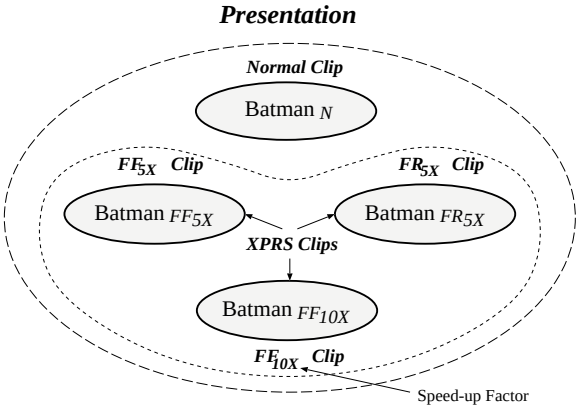

Figure 2: Terminology and notation shown with the example of movie **Batman**

2 Design Overview

The design presented here is based on an offline approach, i.e., it implements the fast forward and fast rewind functionalities by maintaining separate, pre-processed versions of the normal-speed clip. Figure 2 shows the terminology that is used throughout the rest of this paper. A *presentation* consists of one *normal clip* and several *eXPResS* (XPRS) *clips*. Each XPRS clip corresponds to a specific speed-up factor and direction (forward or reverse). If a user invokes fast-forward during an active display, the system switches from the normal clip to the corresponding XPRS clip to provide the effect of on-demand fast-forward. For example, a presentation (say **Batman**) might have a fast-forward version with five times speed ($\text{Batman}_{FF_{5X}}$), a fast-forward version with ten times speed ($\text{Batman}_{FF_{10X}}$), and a fast-rewind version with five times speed ($\text{Batman}_{FR_{5X}}$), as shown in Figure 2.

In order to switch between the different clips of a presentation based on user activities, a method of jumping to the appropriate location in each clip must be provided. These locations are referred to as Random Access Points (RAPs) [And96]. Their cross-references, which map between the frames of different clips, must be maintained. In the example of Figure 2, four mappings are necessary. One maps the frames of Batman_N to those of $\text{Batman}_{FF_{5X}}$, $\text{Batman}_{FF_{10X}}$, and $\text{Batman}_{FR_{5X}}$. This enables a user to switch to any one of the available FF or FR versions during normal display. The remaining three map the frames of each XPRS clip to one another and the normal clip. An example use of these mappings is to enable a user to switch to fast-rewind directly from fast-forward.

The focus of this study is how a Presentation Manager (PM), which controls the display at

the client side (e.g., an extended TV set-top box or a PC-TV) gains access to RAPs. Previous designs have suggested to maintain RAP *index files*, presumably at the server side [And96]. There are several disadvantages to this approach. First, the server must maintain a separate database of RAP index files that can be accessed very quickly (e.g., a real-time or main memory database). This resource could become a bottleneck and limit the scalability of a server. Second, because of client buffering and networking delays the server may not know exactly which frame the client is displaying at the very instant a VCR function is invoked. Therefore, the transition to the new clip may not be very smooth[3].

Our design and implementation techniques enable a server to scale to thousands of streams by avoiding the formation of bottlenecks. The two key ideas that facilitate scalability are as follows. First, the RAP mappings are not maintained centrally (e.g., at the server). Instead, they are interleaved between frames of clips in support of distributed processing. The mappings are organized such that the relevant information is available when the user invokes a VCR functionality. The existence of these mappings does not impact a hardware decoder (e.g., MPEG, MJPEG) to display a presentation: e.g., with an MPEG-2 decoder that utilizes the standard, the interleaved records can be represented as user data which is ignored by the decoder. Alternatively, a simple process (either hardware or software based) can filter these mappings from the data stream and redirect them to the VCR functions circuits, thereby avoiding their transmission to the decoder. This means that the system reads mappings on behalf of a display even though the user might not invoke a VCR functionality. The wasted bandwidth (and storage space) attributed to retrieving this mapping information is negligible, less than 0.2% in our prototype implementation, see Table 3(a).

The second key idea is as follows. A PM interprets the interleaved mappings on demand when a user invokes a VCR function. This off-loads the processing of mappings to the PMs in order to avoid the formation of bottlenecks at the server. Furthermore, the transition between different clips will be precise because the client can determine precisely the last decoded frame when a VCR function is invoked. **In essence, with our technique, a small fraction of the mapping information is staged at the right place (a PM) at the right time (when the user invokes a VCR functionality) to facilitate distributed information processing to realize a scalable server.**

[3]This problem may be avoided by encoding frame numbers into each stream or counting the frames at the client side and submitting this information together with a VCR-command to the server.

3 Methodology with Motion JPEG

The design described in Section 2 depends, to some extent, on the data format assumed to encode a digital video clip. In this section, we will first outline our design and implementation based on a Motion JPEG data format[4]. In Section 4 we outline our technique for the popular MPEG standard.

MJPEG Data Format: The *Motion JPEG* (MJPEG) format as implemented by the Parallax XVideo700™ system can support a video stream based on the Joint Photographic Experts Group (JPEG) image compression standard. Each frame is encoded based on the JPEG algorithm with no inter-frame dependencies. The basic structure of the Parallax MJPEG movie format starts with a fixed length header that describes critical attributes of a clip such as height and width of video frames, quality factor of JPEG images, desired playback rate, audio specifications, etc. A sequence of frames follows the header data structure. These frames are played sequentially. Each individual frame starts and ends with a one-byte delimiter and contains variable-length video and/or audio data.

Interleaved RAP Mappings: In our approach, the RAP mapping information is represented as *records*. Each record consists of a set of pointers to its corresponding RAPs. We interleave records between the frames of clips to enable the system to switch between clips of the same presentation. This section details the structure, content, and location of these records.

Assume that a content provider desires to offer a presentation (say Batman) with n different VCR functionalities (say FF_{5X}, FF_{10X}, and FR_{5X}; $n = 3$). Our algorithm to construct the necessary clips and interleave records between the frames operates in two passes. Figure 3 illustrates the process. During the first pass, the desired frames from the original MJPEG clip are selected to construct a normal clip (termed N for short) and the XPRS clips that contain interleaved records with empty fields. During the second pass, the fields of these records are updated to contain the proper offsets (i.e., RAP pointers) that facilitate the cross-reference mappings.

The detailed steps of the **first pass** are as follows.

- The number (n) of XPRS clips is determined ($n = 3$ in our example). This fixes the size of the interleaved RAP records to n times the size of a RAP pointer. Each pointer will map this location to the corresponding point in one of the scan functionalities. The order of the

[4]Motion JPEG is not as well standardized as MPEG. We base our discussion on the format specified by Parallax Graphics Inc. [Par95].

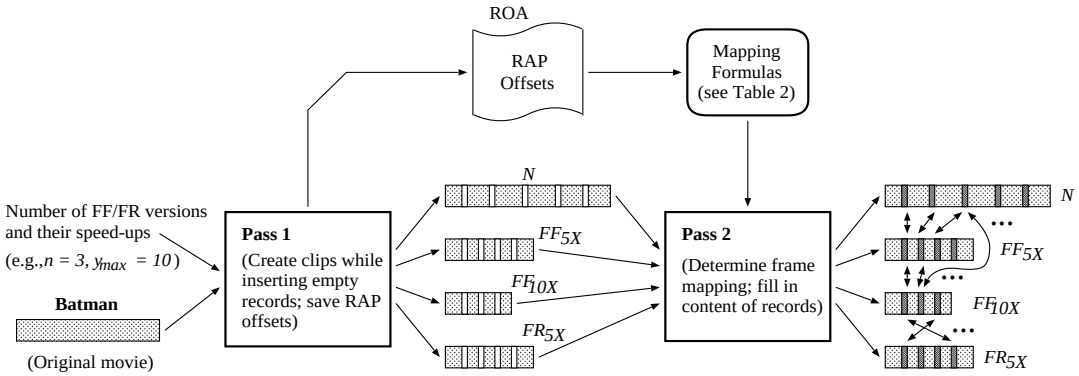

Figure 3: Two pass algorithm for inserting RAP mappings[5]

pointers within a record is $\langle \mathrm{Ptr}_{FF_{aX}}, \ldots, \mathrm{Ptr}_{FF_{bX}}, \mathrm{Ptr}_{FR_{cX}}, \ldots, \mathrm{Ptr}_{FR_{dX}} \rangle$, with $a < \ldots < b$ and $c < \ldots < d$. In XPRS clips, the pointer slot that would map to itself is filled with pointers to the normal version.

- The maximum scan speed (y_{max}) is determined ($y_{max} = 10$ in our example due to FF_{10X}) among the n speeds.
- $n+1$ clips with empty interleaved records are generated. A record precedes each frame of an XPRS clip. In the revised version of the normal clip (Batman_N), a record precedes the first frame and a record is inserted after every y_{max}^{th} frame (10^{th} in our example). The records are left empty because filling them for any reverse clip requires a second pass.
- An intermediate file, termed a RAP Offset Array (ROA) is generated to temporarily store all offsets of RAP records as they are allocated in the newly created clips.
- Each clip header is extended to maintain: (1) the size of a record (n pointers), and (2) the number of frames that appear between two consecutive records for this version (e.g., 1 or 10).

The first pass is complete when the last frame of the normal clip has been processed.

The **second pass** takes the clips generated by the first pass as input and fills in the contents of the RAP records. Recall that the necessary offset information for every frame was stored by the first pass in the intermediate ROA file. In addition, it is necessary to determine which frame of one version of a clip maps to which frames of the other clips. Table 2 shows the equations used to determine this correspondence. As an example, assume Batman_N consists of 4000 frames with 10 frames between two records. Given frame 99 in Batman_N, its corresponding frame number in Batman_{FF5X} is 19, in Batman_{FF10X}, it is 9, and in Batman_{FR5X}, it is 780. At the end of pass two the construction of all the clips is completed. For a detailed discussion of all the issues involved please see our extended technical report [GZK+97].

[5]File sizes are not shown to scale. A FF_{5X} version of a clip would be approximately 20% the size of the N version (or less if the audio portion is removed).

Parameter	Definition
X	Frame number for the current version
Y	Frame number for the new version
S_X	Speed of the current version
S_Y	Speed of the new version
F_N	Number of frames in the normal clip
F_Y	$\lfloor F_N / S_Y \rfloor - 1$
F_I	Number of frames between two records in the normal clip

Table 1: Parameters and their definitions for Table 2

Mapping	Equation
from normal clip to FF clip	$Y = \lceil X/S_Y \rceil$
from FF clip to normal clip	$Y = \lceil X \cdot S_X / F_I \rceil \cdot F_I$
from normal clip to FR clip	$Y = F_Y - \lfloor X/S_Y \rfloor$
from FR clip to normal clip	$Y = \lceil ((F_N - 1) - X \cdot S_X)/F_I \rceil \cdot F_I$
between two similar[a] XPRS clips	$Y = \lceil X \cdot S_X / S_Y \rceil$
between two dissimilar[b] XPRS clips	$Y = F_Y - \lfloor X \cdot S_X / S_Y \rfloor$

[a] Similar XPRS clips have the same type but different speeds, e.g. FF_{5X} and FF_{10X}.
[b] Different XPRS clips have different types, e.g. FF_{5X} and FR_{5X}.

Table 2: Mapping equations between frames among different clips

4 Extension to MPEG

Our approach can be easily applied to other video formats such as MPEG (Motion Picture Experts Group). The main difference between MPEG and MJPEG is that MPEG provides for the inter-frame as well as intra-frame compression. MPEG's inter-frame compression results in several different frame types in the compressed domain: I, P, and B frames (for details see [ISO92]). Only I frames can be decoded independently. This makes it impossible to start a display at an arbitrarily chosen frame. Therefore, an exact mapping between different frames in XPRS files may not be possible with MPEG. However, a simple adjustment in the placement of RAPs enables the implementation of our approach with MPEG video without any noticeable visual problems: the interleaved records are always placed before I frames. Depending on the frame pattern in MPEG

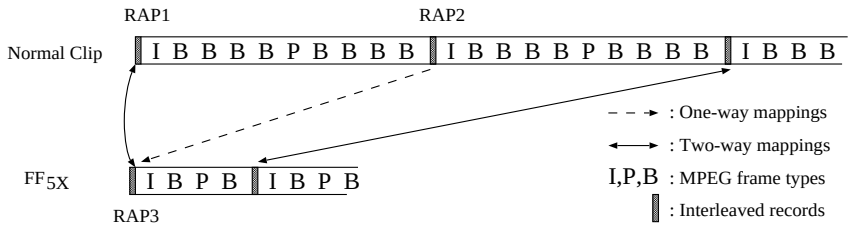

Figure 4: Interleaved RAPs method adapted to MPEG encoded streams

Versions	Presentations		
	Disney	*Top Gun*	*Hawaii*
Normal	0.010 %	0.012 %	0.009 %
FF_{5X}, FR_{5X}	0.121 %	0.156 %	0.109 %
FF_{10X}, FR_{10X}	0.121 %	0.156 %	0.109 %

(a) Wasted storage space and network bandwidth attributed to RAP mapping information

Switching Time [sec]	System Load			
	45%	60%	70%	80%
Minimum	0.391	0.485	0.486	0.480
Maximum	3.621	5.204	5.803	7.541
Average	1.587	1.675	2.056	2.361

(b) Clip switching time for different system loads

Table 3: Performance evaluation results

encoding, several RAPs may have the same value. For example, in Figure 4, both RAP1 and RAP2 are mapped onto RAP3 even though RAP2 is supposed to be mapped onto the third frame (P) in FF_{5X}. This discrepancy may result in a fraction of a second difference in viewing, which is tolerable for browsing. Note that this difference varies according to the frame patterns chosen during MPEG encoding. These details are discussed in [GZK+97] and eliminated here due to space restriction.

5 Performance Evaluation

To evaluate our design, we implemented it for MJPEG compressed videos in Mitra[6] [GZS+97]. For each presentation, there were 5 clips in the system: Normal, FF_{5X}, FR_{5X}, FF_{10X} and FR_{10X}. We first measured the bandwidth (and storage space) attributed to the RAP mapping information. The results in Table 3(a) show that the overhead of interleaving RAP records is negligible (less than 0.2%).

Then we proceeded to evaluate the video quality of our approach. In each experiment, a synthetic workload was imposed on the server with the arrival rate of user requests following a Poisson distribution. Four experiments were conducted with system loads of 45%, 60%, 70%, and 80%. The PM was modified to automatically generate requests for VCR functions. The normal play time was randomly chosen between 10 and 40 seconds and the FF/FR time between 10 and 30 seconds. In all cases, the switching between different clip versions was observed to be precise. Particularly, when a display was in the process of FF or FR, the pictures appeared to be continuous when compared with segment skipping. This is because the display is provided exactly every y^{th} frame. In these experiments, we also measured the switching time when VCR functions were invoked. The latency is a function of the system load (see Table 3(b)) because the system must

[6]Mitra is a research prototype of a VoD server implemented at the database laboratory of the University of Southern California.

startup a new stream. The minimum latency was approximately 0.5 seconds, while the average was 2.4 seconds with a heavy system load of 80%. Those numbers can be significantly improved using a variety of optimization techniques. For example, a PM may display its buffered data at a faster rate while issuing a request to the server to switch to an XPRS file. This would hide the latency observed by a user. The tradeoffs associated with this technique are detailed in [GZK+97].

6 Conclusion and Future Research Directions

This study presented the design and implementation of scalable browsing techniques for Intranet video servers in support of high quality playback with low processing overhead. Using a prototype implementation of these techniques in Mitra [GZS+97], we validated the feasibility of our method as well as its performance. In the near future, we intend to implement our technique for MPEG.

References

[And96] D. Andersen. A Proposed Method for Creating VCR Functions Using MPEG Streams. In *Proceedings of the 12th International Conference on Data Engineering*, Feb. 1996.

[BHH+94] Robert O. Banker, Jeffrey B. Huppertz, Michael T. Hayashi, David B. Lett, Voytek E. Godlewski, and Michael W. Raley. Method of providing video on demand with VCR like functions, October 18, 1994. U.S. Patent No. 5,357,276.

[CKLV95] H.J. Chen, A. Krishnamurthy, T.D.C. Little, and D. Venkatesh. A Scalable Video-on-Demand Service for the Provision of VCR-Like Functions. In *Proceedings of the 2nd Intl. Conference on Multimedia Computing and Systems*, pages 65–72, Washington, D.C., May 1995.

[CKY94] M. Chen, D.D. Kandlur, and P.S. Yu. Support for Fully Interactive Playout in a Disk-Array-Based Video Server. In *Proceedings of the ACM Multimedia*, Oct. 1994.

[DSSJT94] J.K. Dey-Sircar, J.D. Salehi, J.F.Kurose, and D. Towsley. Providing VCR Capabilities in Large-Scale Video Servers. In *Proceedings of the ACM Multimedia*, Oct. 1994.

[GZK+97] S. Ghandeharizadeh, R. Zimmermann, S.H. Kim, W. Shi, and J. Al-Marri. Scalable Video Browsing Techniques for Intranet Video Servers. USC CS Technical Report USC-CS-TR97-659, University of Southern California, 1997.

[GZS+97] S. Ghandeharizadeh, R. Zimmermann, W. Shi, R. Rejaie, D. Ierardi, and T.W. Li. Mitra: A Scalable Continuous Media Server. *Kluwer Multimedia Tools and Applications*, 5(1):79–108, July 1997.

[ISO92] ISO. *Coding of moving pictures and associated audio - for digital storage media at up to about 1.5 Mbit/s*, 1992. ISO Standard IS 11172.

[ORS96] B. Özden, R. Rastogi, and A. Silberschatz. The Storage and Retrieval of Continuous Media Data. In V.S. Subrahmanian and S. Jajodia, editors, *Multimedia database systems*. Springer, 1996.

[Par95] Parallax Graphics. *Video Development Environment^{TM}: Reference Guide for Sun Solaris, HP/UX and IBM AIX*, 1995.

[Sto86] M. R. Stonebraker. The case for Shared-Nothing. In *Proceedings of the 2nd International Conference on Data Engineering*, 1986.

[SV95] P. J. Shenoy and H. M. Vin. Efficient support for scan operations in video servers. In *Proceedings of the ACM Multimedia*, November 1995.