

Design and Implementation of the Elvira Video Server

Olav Sandst , Stein Lang rger, and Roger Midtstraum

Department of Computer Systems
Norwegian University of Science and Technology
N-7034 Trondheim, Norway
{olavs, steinl, roger}@idt.ntnu.no

Abstract

Advanced video applications need more support from a video server than most of today's video servers offer. These applications will include both the video, video related information (meta data), and other multimedia data types which will be shared between different applications and users. The video server should give better support for interactive applications and support multiple media formats and virtual video documents.

Based on these requirements, we have designed and implemented the Elvira video server. The architecture is based on a cluster of workstations connected by a high speed network. The goal has been to build a video server which is highly efficient and scalable. To be able to experiment with different strategies for allocation of videos to the nodes in the server, the video server supports striping of videos across nodes and disks. A prototype of the video server is built running on a cluster of UNIX workstations which are interconnected by an ATM switch. In this paper, the design and implementation is presented together with some experimental results from the use of the video server.

Keywords: Video server, storage and delivery of digital video, multimedia database management system.

1 Introduction

An increasing number of computer based services and applications are making use of digital video. Soon it will be technically possible for everybody to watch digitally stored video on their personal computer or television set. This will require complex systems for delivery and storage of huge amounts of video across a hybrid web of computer, telephone, and cable networks.

Most of today's video applications use video in a rather primitive way, where the computer or the TV set top box, more or less, replaces the familiar VCR-recorder. A typical application of this kind is the Video-On-Demand service — VOD. Whereas the main advantages of VOD-services are the availability and ease of access to the video content and the freedom from the programming of the television company, VOD-services are usually based on a simple consumer model of the user interaction. When the appropriate piece of video is chosen — e.g., a movie — it is played out with limited user interaction. Most of the research on video servers has been focused on problems related to delivery and storage of the video contents for these kinds of applications.

More advanced and professionally oriented video applications will emerge in organizations where video content is used in production, for documentation, or in research — e.g., in digital libraries and in television archives. In these kind of applications, the video will be integrated with other data types like text, still images, and graphics and there will be a need to store video related information (meta data) like structure information and annotations. These systems will have to provide mechanisms for the sharing of both the video content and the other data types between different users and applications. The users are going to produce new video content, possibly by using existing footage, and they are likely to update information in the system.

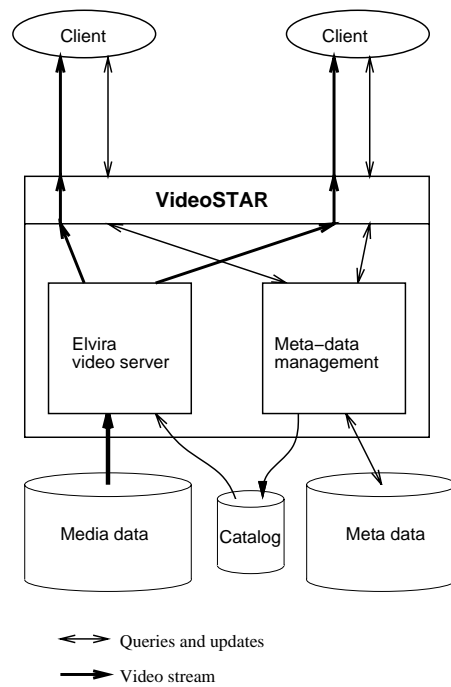


Figure 1: The Elvira video server integrated into the VideoSTAR framework

To provide support for the complex relationships, data consistency and integrity constraints, and to provide concurrent multiuser access to the video data, a multimedia database management system (MMDBMS) should be used as the common database platform for these applications. Supporting complex video applications introduces new requirements on the DBMS. In past research, we have investigated issues related to data modelling and meta data management. We have created the VideoSTAR framework [14] which is a prototype for a video database management system where users and applications can share both media data and meta data. It is based on a generic data model for video information [11, 13]. In addition to the video content itself, this model supports composition, structure information, and user annotations. The purpose of the VideoSTAR research has been to make a model to promote sharing and reuse of video data [15, 16], both media data and meta data.

In the first version of VideoSTAR, the media data — the video and audio — were stored on regular data files which the video player accessed directly. In this paper, we focus on how we have augmented the VideoSTAR video database management system with a video server to be able to administer and deliver huge amounts of video across a network. The requirements that advanced video applications put on the video server have been the basis for the design of the experimental *Elvira* video server. Figure 1 presents how the Elvira video server can be used as a part of the VideoSTAR framework.

Section 2 gives an overview of the architecture of the Elvira video server. The design of the video server is presented in Section 3 together with a model of the catalog used in the video server. Section 4 contains a description of the implementation. Section 5 presents some experimental results, while Section 6 concludes the paper and discusses further work.

1.1 Related work

Video servers for storage and delivery of video across a network is one of the main research areas within multimedia. The attention has mainly been on how to support Video-On-Demand services which allow users to watch movies and videos stored on a digital server [8, 18, 23]. Most of these services have been tailored towards the consumer market where the main goal has been to provide inexpensive video to as many users as possible. Commercial projects like Time Warner's Orlando project have started to deliver video directly to the homes [25].

The main focus has been on the video storage servers [10, 26, 27] and how to optimize these to support as many simultaneous video streams as possible. Different architectures have been

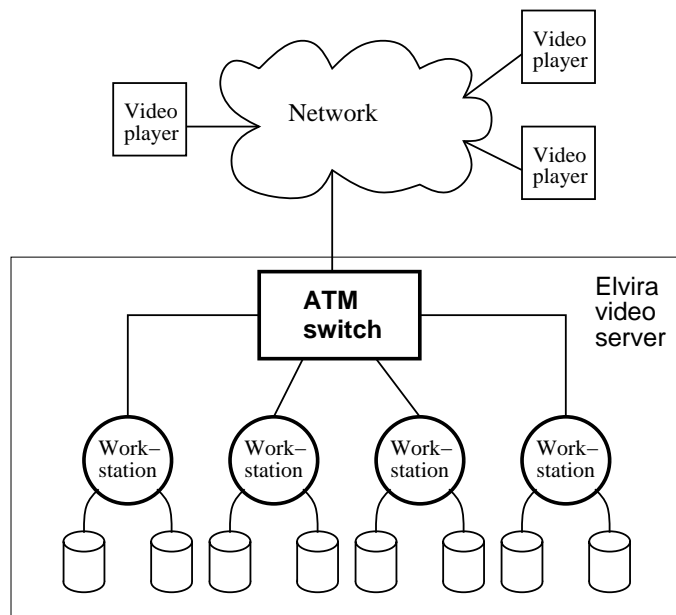


Figure 2: The architecture for the Elvira video server based on a cluster of workstations

proposed for building video servers. Two of the most frequently used architectures for video servers are cluster of workstations and massively parallel multiprocessors. Microsoft's Tiger [1] is based on inexpensive PCs interconnected by an ATM switch. SB-PRAM [9] is an example of a massively parallel processor used as a video server for Video-on-Demand.

Much research has been on disk allocation and disk scheduling [5, 20]. The two main strategies for disk allocation are to store the entire video on one node in the server or to distribute video on several or all of the nodes in the server. An example of the first strategy is SGI's server used in Time Warner's Orlando project [25]. Microsoft's Tiger uses the other strategy where the video is fragmented into fragments consisting of about one second of video [1]. These fragments are then allocated to the nodes in the server in a round-robin fashion. How to allocate video to disks and disk scheduling within one node has also been an important research area [19].

Video delivery across a network has been studied by many researchers due to the problem with jitter in networks and the high bandwidth needed for video delivery [3]. As a result of this research, new video services like Vosaic [4], Xing's StreamWorks [7] and CNN NEWSROOM [6] are starting to appear on the Internet.

2 Architecture for the Elvira Video Server

To deliver digital video from a video server to multiple clients, is resource demanding. The most important resources are network capacity, I/O bandwidth within the server, and storage capacity. An important requirement for a video server is to make optimal use of these resources, while still guarantee to deliver the video and audio frames to its clients on time. Based on these requirements, we wanted to test a video server architecture based on a cluster of workstations connected by a high speed network. A cluster of workstations gives several advantages. The main advantage is that this architecture might provide for scalability. As the demand increases, new workstations can easily be added to the video server to be able to serve more users. Another advantage by using workstations is that we can use commodity hardware and software to build the server.

The architecture used in Elvira is presented in Figure 2. Each workstation is connected to an ATM switch. The ATM network is used for internal communication between the workstations in the server, and for delivering digital video to users. For storing of the video, each workstation has one or more disks.

2.1 Requirements for the Elvira Video Server

During the work with the VideoSTAR prototype, we have defined several requirements which a video server for advanced multimedia applications should support. It has been a goal when designing Elvira to support these requirements and the VideoSTAR data model [13]. In addition to efficient storage and delivery of digital video, the following requirements were the foundation for the design of Elvira:

- **VCR support and frame accurate access** The video server should support full VCR functionality, including single stepping and variable speed fast forward and backward. To support advanced queries against the video DBMS, the video server should support direct access to any specified part of a video document.
- **Interactivity** In advanced video applications, the user will be more active than in video-on-demand applications. The video server has to adapt to these new usage patterns and give better support to interactive applications.
- **Multiple formats** Different applications have different requirements to the format of the delivered video. The video server should be able to store and deliver the same video document compressed by different compression standards e.g., MPEG or Motion JPEG, — and different qualities — e.g., different resolutions and frame rates — while still treating this as *one* logical video document with respect to the rest of the database.
- **Virtual productions** To promote sharing and reuse of the video and meta data, the video DBMS should support *virtual video documents* [24] — i.e., video documents which are composed of sequences from several stored video files. To avoid redundancy, the video server should be able to perform the *materialization* of the virtual video document during the delivery of the video.
- **Editing** Video editing mostly updates the meta data — i.e., the compositional data for the video document — which should be under control of the DBMS. The DBMS should also correspondingly update the catalog information for the video server. During the editing process, the video server should be able to deliver video streams to the client based on *clip lists* delivered from the editing tool.
- **Tertiary storage** Many users of a video DBMS — e.g., television archives — will have a huge and increasing need for video storage. Storing all video on disks will be too expensive. The video server has to be extensible to include other types of cheaper near-line storage like optical storage and tape robots [22].

3 Design

During the design of the Elvira video server, our goal was to design a flexible and extensible video server, making it easy to experiment with different server configurations. In addition to the requirements presented in the previous section, we wanted to be able to experiment with different allocation strategies for video on the nodes in the video server. The design supports the following allocation strategies:

- **Striping of the video** The video, i.e., the stored media segments, is distributed on several nodes in the server. This is done by dividing the video into smaller fragments. Each fragment can be one or several video or audio frames. The fragments are then allocated to the nodes in the video server in a round-robin fashion.
- **Non-striping of the video** The entire video is stored on one of the nodes in the server.

The purpose of introducing striping of video in Elvira was to achieve better load balancing, and thus a possible higher overall utilization of the server.

The overall design of Elvira is given in Figure 3. One of the nodes in the video server is responsible for handling user requests for playback of videos. This node will have a *Call Control* process running. When this process gets a new request from a user, it has to decide whether there is enough free resources in the server to handle this request or the request has to be rejected. To do this, the *Call Control* process keeps track of the global state of the server, which movies are currently delivered, and the load on all nodes in the server. To help determine where a given video is stored and how much resource it will take to deliver the requested video,

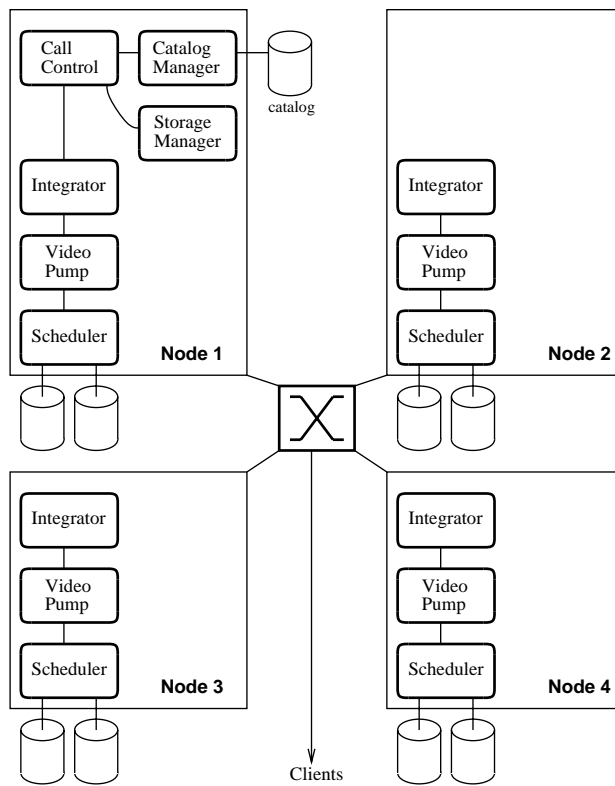


Figure 3: The design of the Elvira video server

a *catalog manager* is used. Based on this information, one of the nodes in the server is selected to be responsible for delivering the video. The catalog manager will be presented in the next subsection.

To support playback of virtual video documents and to support different storage allocation strategies, the responsibility for video delivery has been divided into different tasks. These tasks will, as shown in the figure, be performed by all the nodes in the video server:

- **Integrator** When the *Call Control* has selected the node to be responsible for delivering a video, the Integrator on that node takes over the request and is then responsible for the delivery of the video. It is also responsible for responding to requests from the video client. The user requests can be *Play*, *Pause*, *Fast Forward*, *Fast Rewind* and *Goto Frame*.
- **Video Pump** The Video Pump is responsible for reading video from the disks and deliver the video to the Integrator within the time limits given by the Integrator. If the video is stored entirely on one node, only that node's video pump will participate in the video delivery. If the video is distributed on several nodes, the video pump on each node has to participate in sending video to the integrator. The integrator will then be responsible for integrating the different video fragments delivered by the pumps into one video stream before delivery to the client.
- **Scheduler** The responsibility of the *Scheduler* is to control and optimize the way the disks are accessed. The Scheduler receives disk requests from the video pumps. The Video Pump has assigned a time limit to each disk request. The scheduler has to finish the request within this limit. Based on these time limits the scheduler can optimize the disk usage to ensure that the time limits are met.

A more detailed example on how these processes cooperate to deliver a video will be given in the next section.

3.1 Catalog for Elvira

To be able to support the requirements presented in the previous section, the video server must have a catalog describing the media data stored in the server. VideoSTAR [13] gives a

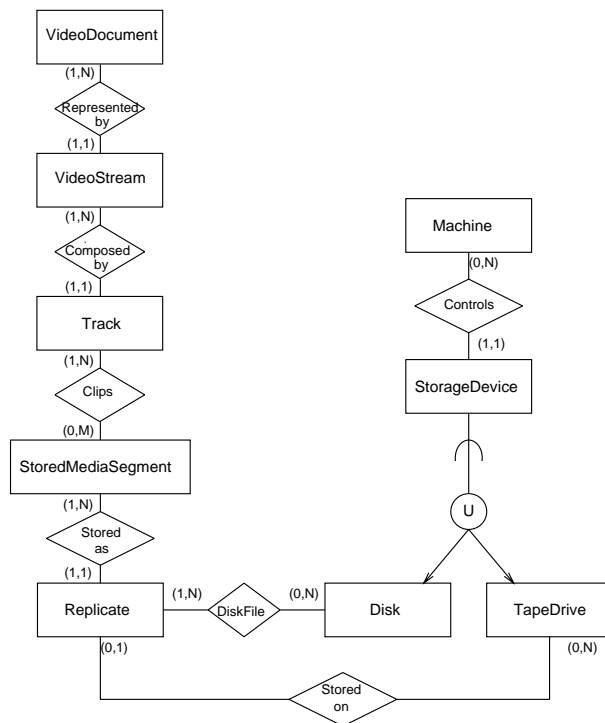


Figure 4: Data model for the Elvira video server catalog

conceptual data model for video and video information. In Elvira, we have extended the parts of this model relevant to the video server with necessary information for storage and delivery of virtual video documents.

An ER-diagram for the catalog is presented in Figure 4. Two of the main concepts the model incorporates is support for multiple media formats and virtual video documents. The *Video Document* is the entry into the model of the catalog and represents a video, – e.g., *The Evening News of 29.5.96* or the movie *Casablanca*.

To support representation of multiple media formats for every logical *Video Document*, there can exist one or more *Video Streams*. A *Video Stream* represents a particular media format of the *Video Document*. We can thus have the *Evening News of 29.5.96* in both MPEG-1 and JPEG compressed format for supporting different video players while still regarding this as one video document. Each video stream can be further divided into several tracks, usually one video track and one or two audio tracks.

To support *virtual video documents* [24] and reuse of footage, each track can consist of one or several *stored media segments* — e.g., the *Evening News* can consist of several sequences which are stored separately and assembled by the video server during playback.

Each stored media segment represents a contiguous sequence of either audio or video footage which has to be stored on a storage medium. The data model contains information on how these are mapped onto different storage devices — e.g., magnetic disks or tape. To support experimenting with different storage strategies, each *stored media segment* can have several replicas. The reason for supporting several replicas can be to increase the performance or to support storage of the media segment on different media technologies — e.g., both in a disk cache and on tape. It also gives us the abilities to experiment with different distribution strategies for the replicas — e.g., a replica can be stored on one disk, or it can be striped over disks on several nodes.

4 Implementation of Elvira

The first prototype of the Elvira video server was built on a cluster of four Sun workstations running SunOS 4, interconnected by an ATM switch [21]. We have tried not to make use of special hardware or software optimizations, to make it easily portable. The video server

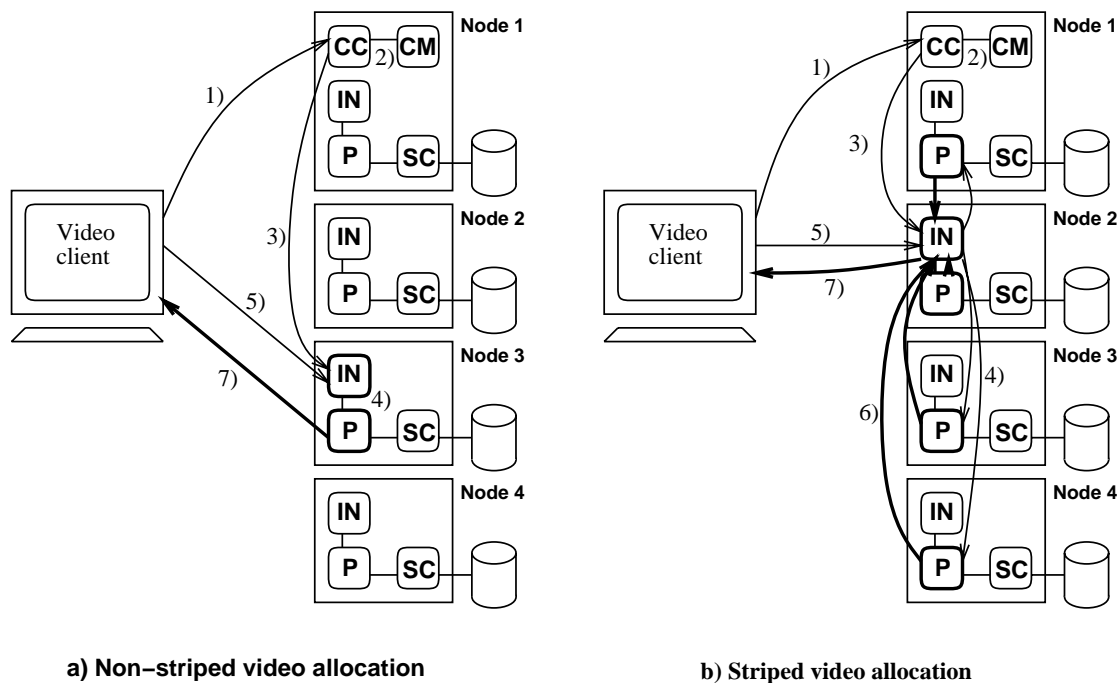


Figure 5: Delivery of a video distributed on 4 machines

is implemented in C/C++. Standard TCP/IP and UDP is used for network communication. This makes it easy to move the video server to new platforms. In addition to SunOS 4, it is also running on SunOS 5 machines and on SGI's IRIX operating system. It is also possible to run the video server on a single machine or on machines connected by a non-ATM network.

Because the first prototype was built on SunOS 4 which does not support threads, the implementation of Elvira is based on processes. Each of the entities in Figure 3 are implemented as separate processes. During video delivery there will be a separate *Integrator* and *Video Pump* process for each video to be delivered. When the main *Integrator* and *Video Pump* process receives a request for delivering a new video, a subprocess is created responsible for deliver this particular video to the user.

The current version of the video server supports storage and delivery of Motion JPEG and MPEG-1 compressed video and audio.

4.1 Video Delivery

Figure 5 shows two scenarios for delivery of videos from the server to the clients. When a client wants to start playback of a new video, it contacts the *Call Control* (CC) process (1). The Call Control looks up the video document in the catalog (2). Based on the information from the catalog and the current load on the nodes in the server, it decides if there is enough capacity to deliver this document. If the request is accepted, the Call Control process selects one of the machines in the server to be responsible for delivering the video to the user. The request is then handed over to the *Integrator* (IN) running on that machine (3).

When the Integrator process gets the request from the Call Control, it creates a subprocess which will be responsible for the delivery of the video stream to the client. This process contacts the *Video Pump* processes (P) on the machines which have to participate to fulfill the request (4), and instructs them to set up connections for video delivery. Figure 5a shows the process and communications connections when delivering a video which is stored on *one* of the nodes in the video server. As an optimization to reduce data transfer within the server, the video delivery goes directly from the Video Pump process to the client (7). This is not possible when the video is striped across several nodes in the server, as shown in Figure 5b. The Video Pumps deliver the video fragments to the Integrator (6), which is then responsible for delivering the video stream to the user (7).

The current implementation of the video server uses TCP connections for sending commands

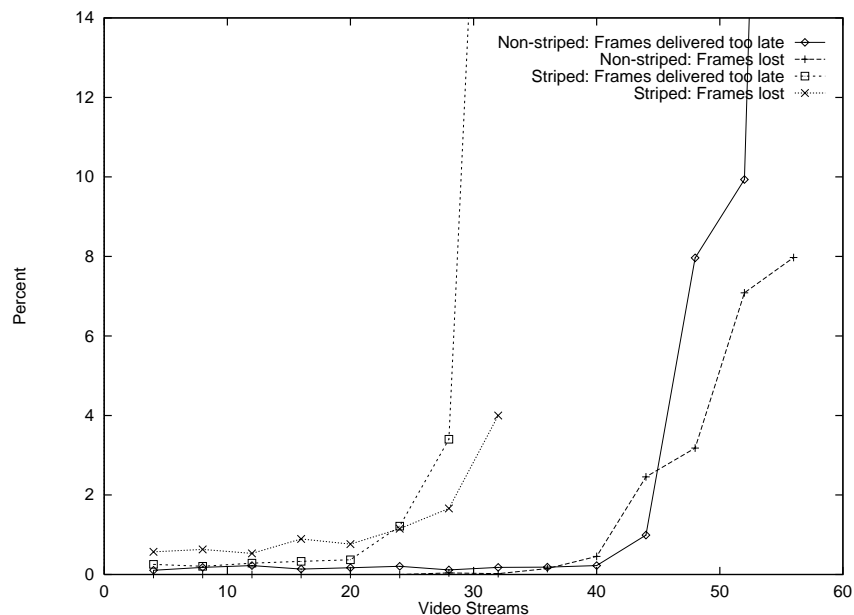


Figure 6: Delayed and lost video frames for non-striped and striped video allocations

and UDP on top of ATM for sending video and audio. The video server uses a *request driven* protocol for delivery of video. After the initial setup has been performed, the client sends requests to the Integrator every time it wants more video frames to be delivered (5). Such a request message consists of the *frame number* for the first frame to be delivered, the *speed* it wants to use for replay the video, and the *number of frames* it wants the server to send in response to this command. A *time limit* for when the client must have the first frame delivered is also given. A typical request will be for a few seconds of video.

By analyzing this request, the Integrator decides which of the Video Pump processes that have to be contacted in order to get the requested part of the video. When the pumps start to deliver data to the Integrator, the Integrator assembles the different fragments and delivers the resulting video stream to the client. The video server has to deliver the first frame of video within the time limit given in the request. After the first frame is delivered, the video server will deliver the following frames as an isochronous stream until the number of requested frames are delivered.

This request driven protocol puts great responsibility on the clients for buffer management and synchronization of the video presentation. In most other systems, the video server is responsible for delivering a highly isochronous video stream. In our protocol, the client has to send a new request for more video to the video server before the delivery of the current request is finished to make sure it does not run out of video frames.

The advantage of this request driven approach is that it puts more relaxed requirements on the server for synchronization of the video deliverance. User interactions through frequent use of VCR functions and jumps within the document are well supported because all delivery of video is synchronized from the client.

5 Experiments and Results

In this section, we present some results from using the video server. During these tests, we have used a video server running on a cluster consisting of four SparcStation 20 each with a 125 Mhz HyperSparc processor. Each workstation has 64MB main memory and between 4GB and 14 GB of disk for video storage. The workstations are running SunOS 5.4 and are connected by a Fore ASX-200 ATM switch.

To measure the capacity and scalability of the video server, we started a number of video clients sending requests to the video server. We used two different video clients. The first was an instrumented version of the video player used in VideoSTAR which writes a log of the arrival time of each video/audio frame, and compares it to the display time of that video/audio frame.

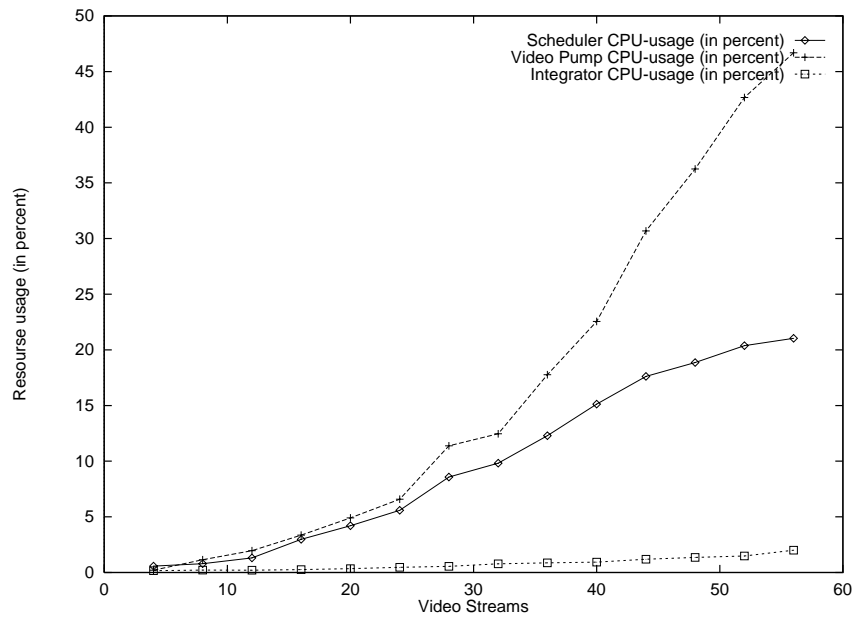


Figure 7: The CPU load measured on one of the nodes using non-striped video allocation.

The second version was a *dummy* video player which sends requests to the server, but drops the frames which are returned. This was used for creating a steady load on the server and because we did not have enough client workstations to run the full video player.

During the experiments we used a 20 minutes TV news program from TV2 Norway which we compressed into a 1.8Mbit/s Motion JPEG video stream with 20 frames pr. second. We made 50 duplicates of this video file which were distributed on each of the nodes in the server. For storing the videos each node had one 4GB Seagate Barracuda disk.

Non-striped Video Allocation

The results from the experiment with non-striped video allocation is given by two of the curves in Figure 6. The number of video frames which are delivered too late is insignificant until it reaches a load of about 40 simultaneous video streams. When we increase the number of simultaneous video streams to more than 44, the number of delayed and lost video frames increases rapidly. The limiting resource is the bandwidth of the disk. With 44 video streams, each disk must deliver on average 2.5MB/s and the total network load generated by the server output is about 80Mbit/s. The CPU usage for the main processes in *one* of the nodes in the server is given in Figure 7.

Striped Video Allocation

Figure 6 also contains the number of delayed and lost frames when striping the video on all nodes in the server. When using this allocation strategy, the server manages to deliver about 20 simultaneous videos. The main reason is that the communication within the cluster is nearly doubled because the video stream has to go through the Integrator process (see Figure 5b) and each video stream has to go through the communication interface three times instead of just one. Five processes participate in the video delivery instead of just one as in the non-striped example. This introduces more jitter into the video stream. The main memory used for buffer is also increased because the Integrator might need to buffer video frames it gets from the Video Pumps before they are delivered to the user. This can also be seen in Figure 8. Compared to Figure 7, the CPU resources used by the Integrator are much higher when the video is striped on several disks, and has to be integrated before delivered to the user.

5.1 Practical Use of the Video Server

The video server is used in the following two practical experiments:

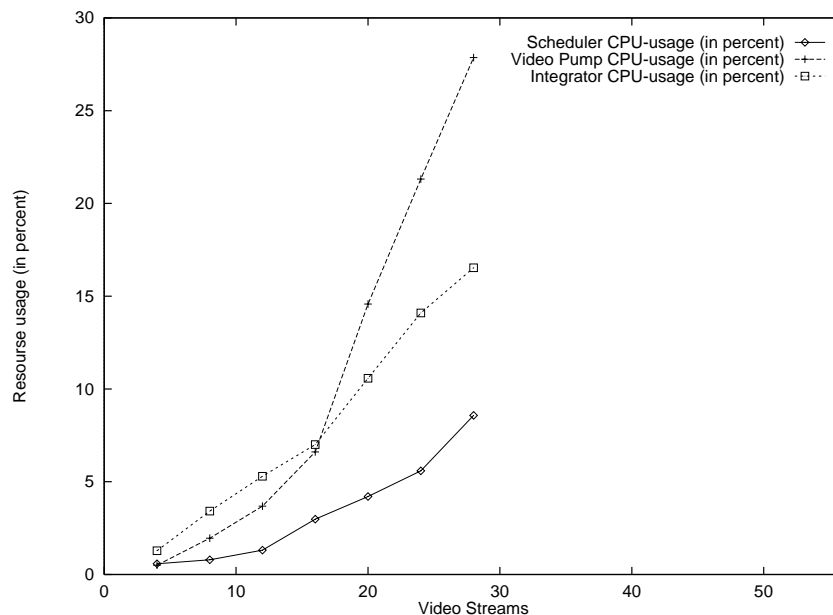


Figure 8: The CPU load measured on one of the nodes using striped video allocation.

VideoSTAR

The video server is integrated into the VideoSTAR [12, 17] database framework as seen in Figure 1. It is responsible for storing and delivering the video managed by VideoSTAR, and responding to requests from the VideoSTAR tools.

Television News on the Internet

We are participating in the LAVA project to deliver television news on the Internet. This project is carried out in cooperation with the Norwegian Broadcasting Corporation, Norwegian Computing Center and NORUT IT. The purpose is to make television news available on the Internet [2]. Currently, we make the daily news program Nordnytt available to Internet users. After the news program has been broadcast on ordinary television, it is digitized and compressed into low quality MPEG-1 video and audio, and stored in the video server. The video is then made available through a World Wide Web interface. This experiment has many similarities to the CNN NEWSROOM service on the Internet [6].

6 Conclusion and Further Work

In this paper, we have presented the architecture and design of the Elvira video server. The design of Elvira is based on new requirements to support advanced video applications. These applications will need better support from the video server than most of today's Video-On-Demand-like services offer. The most important of these requirements is to give better support for interactive applications, as well as support multiple formats and virtual video documents.

We have implemented Elvira on a cluster of workstations connected by an ATM network. Elvira supports different storage allocation strategies for video to nodes in the server. Non-striped allocation of videos to the nodes gives the highest number of simultaneous delivered videos. Using striped video allocation, the server only manages about half the number of simultaneous videos in our tests with a server consisting of four workstations. The main reason is that today's design doubles the internal communication load when using striped allocation of videos. We achieve better load balancing when using striped allocation and in certain situations when there is a non-uniform access to the videos, this might increase the number of users Elvira will be able to serve. In the experiment presented in the paper, the non-striped allocation can not guaranty more than 10 videos delivered if all requests are for videos stored on *one* of the nodes. In the case of striped allocation, the server will be able to deliver the maximum number

of videos independent of the request pattern.

In a server using non-striped allocation of videos, all nodes deliver the video independent of each other. Thus, the number of simultaneous videos Elvira is able to deliver scales linearly with the number of nodes in the server. Using striped allocation of videos to nodes, the server will not scale linearly due to the internal communication overhead in the server.

Based on the experiences with the current version of Elvira, we plan to improve the design. Because of the overhead using the Integrator to synchronize the video delivery, in the next version the Video Pumps will deliver the video directly to the clients even when the video is striped onto several nodes. To be able to do this requires better synchronization of the Video Pumps between nodes. We plan to introduce more strict timing in the server. Today the synchronization of the video delivery is influenced by the process scheduling of the operating system. We plan to replace today's processes with threads.

We are also working on extending the video server with tertiary storage. Storage of huge amounts of video will cost too much if it is based on disks only. At the moment, we are extending Elvira to use magnetic tape as near-line video storage

References

- [1] W. J. Bolosky, J. S. Barrera, III, R. P. Draves, R. P. Fitzgerald, G. A. Gibson, M. B. Jones, S. P. Levi, N. P. Myhrvold, and R. F. Rashid. The Tiger video fileserver. In *Proceedings of the 6th International Workshop on Network and Operating System Support for Digital Audio and Video, NOSSDAV'96*, pages 97–104, Zushi, Japan, April 23–26 1996.
- [2] H. Bryhni, H. Lovett, E. Maartmann-Moe, D. Solvoll, and T. Sørensen. On-demand regional television over the internet. In *To appear in Proceedings of ACM Multimedia '96*, Boston, Massachusetts, November 18–22. 1996.
- [3] S. Cen, C. Pu, R. Staehli, C. Cowan, and J. Walpole. A distributed real-time MPEG video audio player. In *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 151–162, Durham, New Hampshire, April 1995.
- [4] Z. Chen, S.-M. Tan, R. H. Campbell, and Y. Li. Real time video and audio in the World Wide Web. In *World Wide Web Journal, Fourth International World Wide Web Conference Proceedings*, pages 333–348. World Wide Web Consortium, O'Reilly & Associates, Inc., December 11-14 1995.
- [5] A. L. Chervenak, D. A. Patterson, and R. H. Katz. Choosing the best storage system for video service. In *Proceedings of the ACM Multimedia'95*, pages 109–119, San Francisco, California, November 1995.
- [6] C. L. Compton and P. D. Bosco. Internet CNN NEWSROOM: A digital video news magazine and library. In *Proceedings of the Second IEEE International Conference on Multimedia Computing and Systems*, Washington DC, May 1995. IEEE.
- [7] X. T. Corporation. Streamworks technical description.
- [8] D. Deloddere, W. Verbiest, and H. Verhille. Interactive Video On Demand. *IEEE Communications Magazine*, 32(5):82–88, May 1994.
- [9] J. Friedrich, T. Grün, and J. Keller. Video-on-Demand on the SB-PRAM. In *Proceedings of the 6th International Workshop on Network and Operating System Support for Digital Audio and Video, NOSSDAV'96*, pages 105–111, Zushi, Japan, April 23–26 1996.
- [10] D. Gemmell et al. Multimedia Storage Servers: A Tutorial. *IEEE Computer*, pages 40–49, May 1995.
- [11] R. Hjelsvold. Video Information Contents and Architecture. In *Proceedings of the 4th International Conference on Extending Database Technology*, pages 259–272, Cambridge, UK, March 1994.
- [12] R. Hjelsvold, S. Langørgen, R. Midtstraum, and O. Sandstå. Integrated Video Archive Tools. In *Proceedings of the ACM Multimedia'95*, pages 283–293, San Francisco, California, November 1995.
- [13] R. Hjelsvold and R. Midtstraum. Modelling and Querying Video Data. In *Proceedings of the 20th VLDB Conference*, pages 686–694, Santiago, Chile, September 1994.

- [14] R. Hjelsovold and R. Midtstraum. Databases for Video Information Sharing. In *Proceedings of the IS&T/SPIE Symposium on Electronic Imaging Science and Technology, Conference on Storage and Retrieval for Image and Video Databases III*, pages 268–279, San Jose, California, February 1995.
- [15] R. Hjelsovold, R. Midtstraum, and O. Sandstå. A Temporal Foundation of Video Databases. In J. Clifford and A. Tuzhilin, editors, *Recent Advances in Temporal Databases. Proceedings of the International Workshop on Temporal Databases*, pages 295–314, Zürich, Switzerland, September 1995. Springer Verlag.
- [16] R. Hjelsovold, R. Midtstraum, and O. Sandstå. Searching and Browsing a Shared Video Database. In *Proceedings of the 1995 International Workshop on Multi-Media Database Management Systems*, pages 90–98, Blue Mountain Lake, New York, August 1995.
- [17] R. Hjelsovold, R. Midtstraum, and O. Sandstå. Searching and browsing a shared video database. In K. C. Nwosu, B. Thuraisingham, and P. B. Berra, editors, *Multimedia Database Systems*, chapter 4, pages 89–122. Kluwer Academic Publishers, 1996.
- [18] W. W. Hodge. *Interactive Television*. McGraw-Hill, 1995.
- [19] K. Keeton and R. H. Katz. Evaluating video layout strategies for a high-performance storage server. *ACM Multimedia Systems*, pages 43–52, 1995.
- [20] T.-G. Kwon and S. Lee. Data placement for continuous media in multimedia DBMS. In *In proceedings of the International Workshop on Multi-Media Database Management Systems*, pages 110–117, Blue Mountain Lake, New York, August 1995.
- [21] S. Langørgen. Elvira - EksperimentelL VIdeotjeneR for Atm. Master's thesis, Norwegian Institute of Techology, Trondheim, Norway, December 1994.
- [22] S. Lau, J. C. Lui, and P. Wong. A cost-effective near-line storage server for multimedia system. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 449–456, Taipai, Taiwan, March 1995. IEEE.
- [23] T. Little and D. Venkatesh. Prospects for Interactive Video-on-Demand. *IEEE Multimedia*, 1(3):14–24, Fall 1994.
- [24] W. Mackay and G. Davenport. Virtual Video Editing In Interactive Multimedia Applications. *Communications of the ACM*, 32(7):802–810, 1989.
- [25] M. N. Nelson, M. Linton, and S. Owicki. A highly available, scalable ITV system. In *Proceedings of Fifteenth ACM Symposium on Operating Systems Principles*, pages 54–67, Copper Mountain Resort, December 3-6 1995. ACM.
- [26] R. Rooholamini and V. Cherkassky. ATM-Based Multimedia Servers. *IEEE Multimedia*, pages 39–52, Spring 1995.
- [27] B. Tierney et al. Distributed Parallel Data Storage Systems: A Scalable Approach to High Speed Image Servers. In *Proceedings of ACM Multimedia 94*, pages 399–405, San Francisco, California, October 1994.