

# Video Server on an ATM Connected Cluster of Workstations

Olav Sandstå, Stein Langørgen, and Roger Midtstraum  
Department of Computer and Information Science  
Norwegian University of Science and Technology  
N-7034 Trondheim, Norway  
{olavs, steinl, roger}@idi.ntnu.no

## Abstract

*Video servers are important for applications which make use of digital video. The video servers should provide better functionality than most of today's video servers offer; – e.g., support of flexible and instant user interactions, delivery of multiple video formats and support of virtual video documents. In this paper we discuss the requirements that video servers should fulfill and we describe the design and implementation of the Elvira video server. The Elvira video server is built on a cluster of standard UNIX workstations interconnected by an ATM switch. The capacity of the Elvira server is evaluated and we show the effects of different strategies for allocation of video data across nodes and disks.*

## 1. Introduction

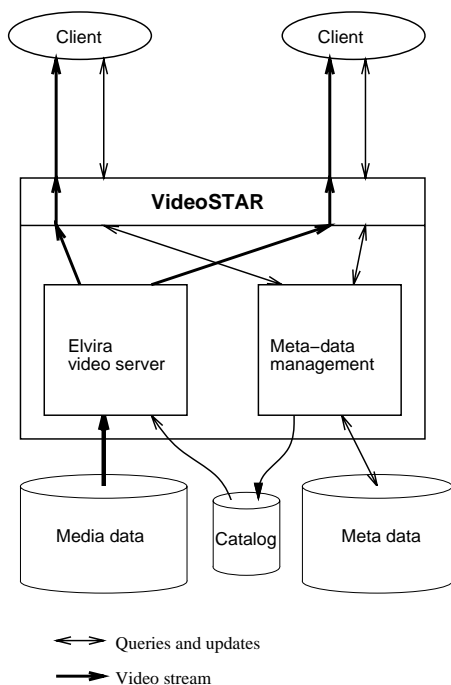
An increasing number of computer based services and applications are making use of digital video. Soon it will be technically possible for everybody to watch digitally stored video on their personal computer or television set. This will require complex systems for delivery and storage of huge amounts 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, in which 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 related information (meta data) like structure information and content 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.

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 [4]. 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 [17] 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 [15]. 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 [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 framework 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



**Figure 1. The Elvira video server integrated into the VideoSTAR framework**

can be used as a part of the VideoSTAR framework.

The rest of this paper is organized as follows. First we present a brief overview of related work. Section 2 then 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 video catalog used in the video server. In section 4, we describe the implementation of the video server and explain the processes involved in delivery of video from the server. Section 5 presents some experimental results which estimates the capacity of the video server, 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 video server [10, 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 [28].

The main focus has been on the video storage servers [13,

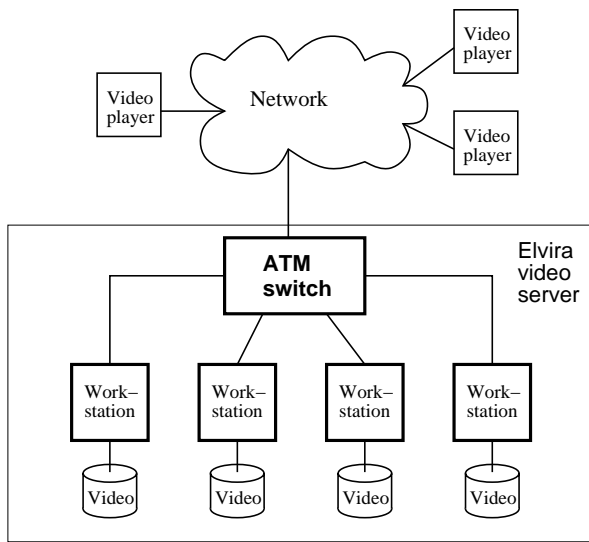
31, 33] and how to optimize these to support as many simultaneous video streams as possible. Different architectures have been proposed for building video servers. Two of the most used architectures for video servers are a cluster of workstations and parallel multiprocessors. Microsoft's Tiger [2] is based on inexpensive PCs interconnected by an ATM switch. SB-PRAM [12] and Oracle Media Server running on a nCube machine [22] are examples of massively parallel multiprocessors used as a video server for Video-on-Demand. Examples of research prototypes of video server systems are Fellini [25] and Berkeley Distributed Video-on-Demand System [11].

Much research has been done on how to make efficient use of the available computing resources within the video server. An important area which has been studied is how to optimize the use of disk and memory. Several strategies for disk allocation has been proposed. The main strategies are either to store the entire video on a single disc, to stripe the video on several disks on a single machine [1] or on several disks on multiple machines [2, 8, 19, 30]. The main reason for introducing striping has been to achieve better load balancing on the disks. Disk scheduling and how to optimize the use of the disk bandwidth has been studied intensively [6, 32, 36]. Closely related to disk allocation and disk scheduling is how to utilize the available main memory for buffer. Several strategies and algorithms are proposed like BASIC and DISTANCE [29] and *Least/Most Relevant for Presentation (L/MRP)* [26].

Video delivery across a network has been studied by many researches due to the problem with jitter in networks and the high bandwidth needed for video delivery [5, 34]. As a result of this research, new video services like Vosaic [7], Xing's StreamWorks [35] and CNN NEWSROOM [9] are starting to appear on the Internet.

## 2. Architecture for the Elvira Video Server

The purpose of this paper is to present and test a video server architecture based on a cluster of workstations interconnected by a high speed network. Using 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. An important requirement for a video server is to make optimal use of the available computing resources provided by the underlying hardware, operating system and network, while still be able to guarantee non-interrupted delivery of the video and audio frames to its clients. The most important resources in a video server are network capacity, disk and network bandwidth, CPU and memory, and storage capacity. By using a cluster of workstations each of these resources can be added more or less independently of the the other, – e.g., if disk



**Figure 2. The architecture for the Elvira video server**

bandwidth is the limiting resource we can add another disk or a new disk controller. A further advantage by using workstations is that we can use commodity hardware and software to build the server.

The overall architecture of 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 the users. For storing 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 fulfill these requirements and support the VideoSTAR data model [15]. In addition to efficient storage and delivery of digital video, the following requirements were the foundation for the design of Elvira:

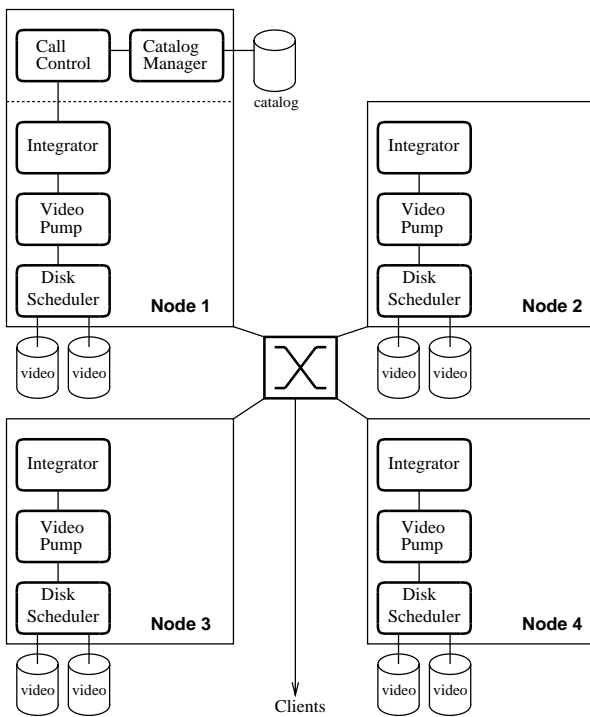
- **VCR support and frame accurate access** The video server should support full VCR functionality including single stepping and variable speed fast forward and fast rewind. To support advanced queries against the video DBMS, the video server should support direct access to any specified part of a video document.
- **Interactivity** In advanced video applications, the user will be more active than in a typical Video-on-Demand

application. The user and the user applications will interact with the video delivery by issuing frequent commands for controlling the video playback. The video server should support such usage patterns by providing immediate response to user commands – i.e., the latency from the user issues a command until the effect of the command is visible on the screen should be small.

- **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 [21].

A goal for developing the video server was to be able to experiment with different server configurations. This requires a flexible and extensible design. To be able to experiment with different video allocation strategies for video on the nodes in the video server, Elvira supports the following video allocation strategies:

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



**Figure 3. The design of the Elvira video server**

- **Non-striping of the video** The entire video is stored on one of the nodes in the server.

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

### 3. Design

The overall design of Elvira is given in Figure 3. We have separated the responsibility for controlling the server and for video delivery into separate tasks. One of the nodes in the video server is selected to be responsible for control of the server. This node will be responsible for handling user requests for playback of videos (admission control) and for controlling the load of the server. As seen in Figure 3 node one has two extra processes which will be responsible performing these tasks.

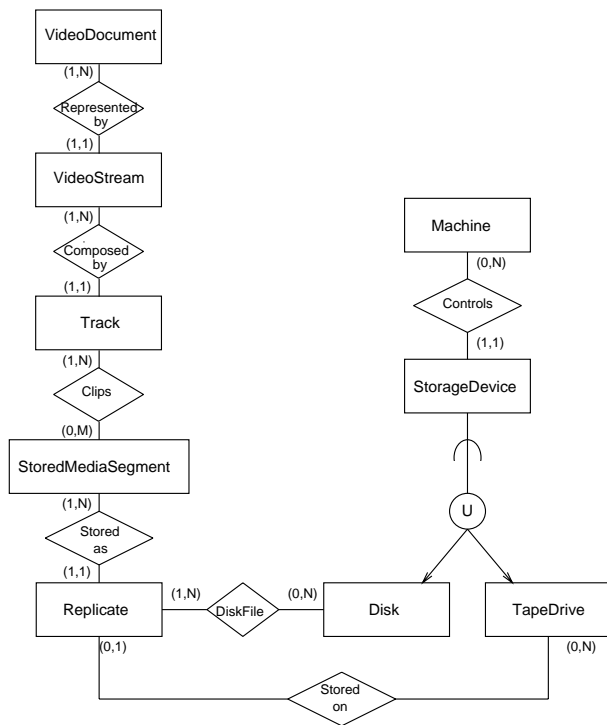
All nodes in the server will participate in delivering of video to the users. To support playback of virtual video documents and to support different storage allocation strategies, the responsibility for video delivery has been further divided into different tasks. These tasks will, as shown in the figure,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

- **Call Control** This process is responsible for handling

new request from the users for playback of a given video. It is also responsible for making sure that none of the nodes in the video server is overloaded. When a new request arrives it has to decide whether there is enough free resources in the server to handle this request, or if 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. Based on this information the *Call Control* process decides which of the nodes in the server to be responsible for the delivery of this video.

- **Catalog Manager** The *Catalog Manager* is responsible for managing the meta-data about the videos stored in the video server – e.g., on which node(s) a video is stored and what video format is used for compressing the video. This information is used by the *Call Control* process to decide how much resources it will take to deliver a requested video and which node(s) has to participate in the delivery of the video. A data model for the video catalog will be presented in the next subsection.
- **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 delivering 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 striped 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.
- **Disk Scheduler** The responsibility of the *Disk Scheduler* is to control and optimize the way the disks are accessed. The *Disk Scheduler* receives disk requests from the video pumps. The video pumps assigns a time limit to each disk request. The *Disk Scheduler* has to finish the disk request within this limit. Based on these time limits the *Disk Scheduler* can optimize the disk usage to ensure that all time limits are met.

A more detailed example on how these processes cooperate to deliver a video will be given in Section 4.



**Figure 4. Data model for the Elvira video server catalog**

### 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 gives a conceptual data model for video and video information [15]. 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 Entity-Relation diagram for the catalog is presented in Figure 4. Two of the main concepts are support of multiple media formats and support of 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 Motion 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 on disk 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 [20]. We have tried not to make use of special hardware or software optimizations, to make it easily portable. The video server is implemented in C/C++. Standard TCP/IP and UDP are used for network communication. This makes it easy to port the video server to new platforms. In addition to SunOS 4, it is also running on machines using SunOS 5 and on SGI's IRIX operating system. It is also possible to run the video server on a single workstation or on workstations connected by a non-ATM network.

Because the first prototype was built using 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 a separate process. During video delivery there will be a separate *Integrator* and *Video Pump* process for each video currently being delivered. These processes are created when the main *Integrator* and *Video Pump* receive a request for delivering a new video, and will from then on be responsible for handling client commands and deliver the video frames for this particular video playback.

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

### 4.1. Video Protocol

The current implementation of the video server uses TCP connections for sending commands and UDP on top of ATM for sending video and audio. The video server uses a *request driven* protocol for delivery of video, opposed to most other

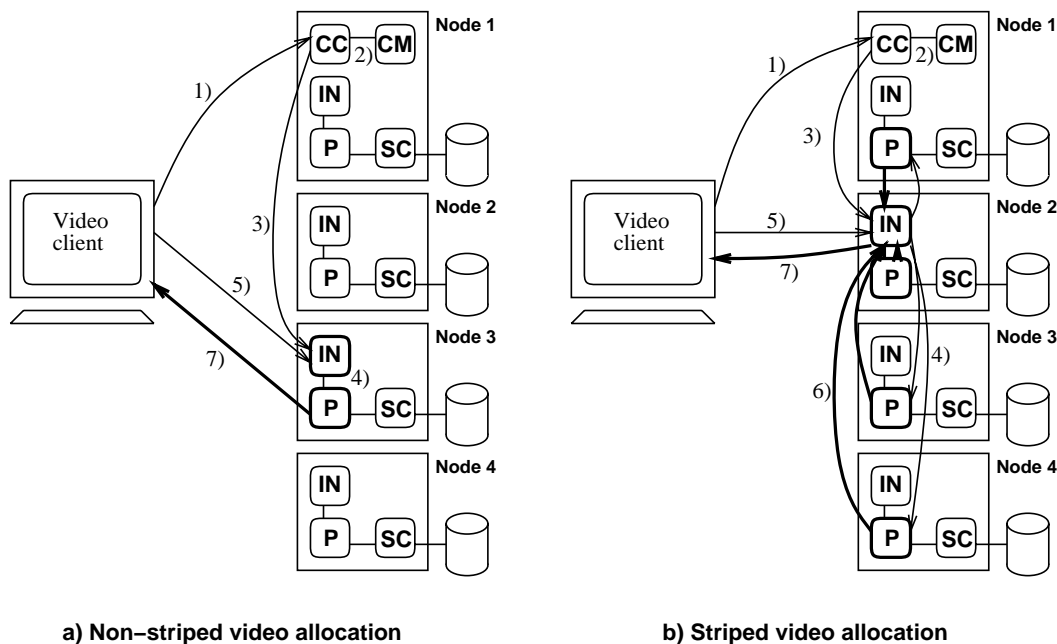


Figure 5. Delivery of a video distributed on 4 machines

video servers which uses a push protocol for video delivery. After the initial setup has been performed, the client sends requests to the Integrator every time it wants more video frames to be delivered. Such a request message consists of the *frame number* for the first frame to be delivered, the *speed* it wants to use for replay of 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 delivery. User interactions through frequent use of VCR functions and jumps within the video document are well supported because all delivery of video is synchronized from the client.

#### 4.2. Delivery of Non-Striped Video

Figure 5 shows two scenarios for delivery of videos from the server to the clients. The first of these shows how a video is delivered to a client in the case where the entire video is stored on one node in the server. When a client wants to start playback of a new video, it contacts the *Call Control* (CC) process (1). The Call Control asks the *Catalog Manager* (CM) to look 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 the machine which has the video stored 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) (4), and instructs it to set up connections for video delivery. The client sends its requests for video to be delivered to the Integrator

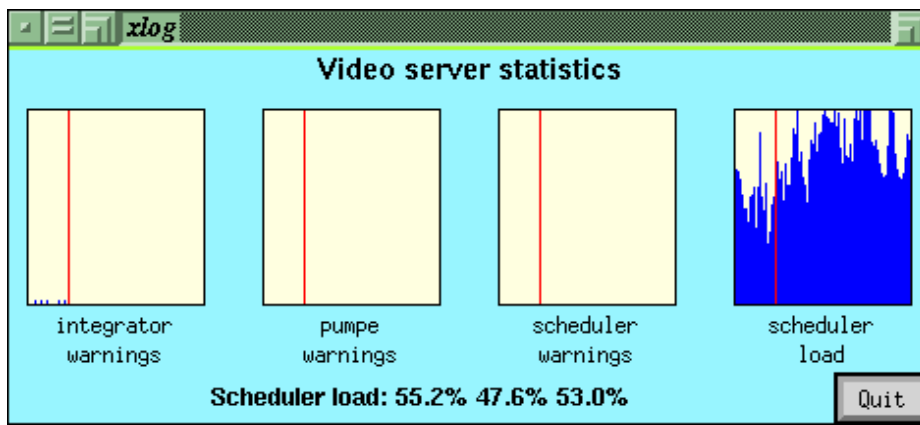


Figure 6. A graphical interface for monitoring the current status of one of the nodes in the video server

which then instructs the video pump to do the delivery of the video. As an optimization to reduce data transfer within the server, the Video Pump sends the video directly to the client (7) without sending it through the Integrator.

Figure 6 shows a graphical window which shows status information about the current load and error conditions on one of the nodes in the video server.

### 4.3. Delivery of Striped Video

Figure 5b shows how the video is delivered in the case the video is stored on more than one of the nodes. Just as in the case where the video was stored on one node, the client sends a request to the the Call Control(1). The Call Control selects on of the nodes in the server to be responsible for this video delivery, and instructs the Integrator on that node to be responsible for the delivery(3).

When the Integrator process gets the request from the Call Control, it contacts the *Video Pump* processes (P) on all the machines which have to participate to fulfill the request (4), and instructs them to set up connections for video delivery.

Since the video now is partitioned into small parts stored on different nodes, it has to be integrated into one stream before it is sent to the client. The Video Pumps deliver the video fragments to the Integrator (6), which assembles the video fragments and deliver the video stream to the user (7). For the video client it is indifferent whether the video is striped or not.

### 4.4. Support for VCR-operations

The video server supports normal playback, fast forward, fast rewind and random seek of the videos stored in the

server. For both Motion JPEG and MPEG-1 we store the video and audio on separate media files. For each media file we have a corresponding index file which contains pointers to the individual frames in the media file.

Audio is only delivered during normal video playback. To support fast forward and fast rewind for Motion JPEG we do frame skipping in the server. If the user request fast forward with a speed of five times the normal playback speed, the server will only send every fifth video frame to the client.

For MPEG-1[27] it is not as straightforward to support fast forward and fast rewind. We have implemented two different algorithms for doing fast forward and fast rewind [18]. The first method do frame skipping. Depending of the requested speedup we send I- and P-frames (skipping the B-frames) or only the I-frames (skipping both P- and B-frames). The second method we have implemented uses separate files for doing fast forward and fast rewind. These files are made by picking every  $n$  frames from the original file, decode the frames and coding them into a new MPEG-1 video file. If this file is played with normal speed it will look like the original file with a speedup of  $n$ . When the user requests fast forward the server will change from the original MPEG-1 file to the fast forward file.

## 5. Experiments and Results

In this section, we present some results from using the video server. The goal is to evaluate the design and implementation of the video server and to be able to compare the different storage strategies used by the video server.

During these experiments, 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 a 4GB Seagate Barracuda disk

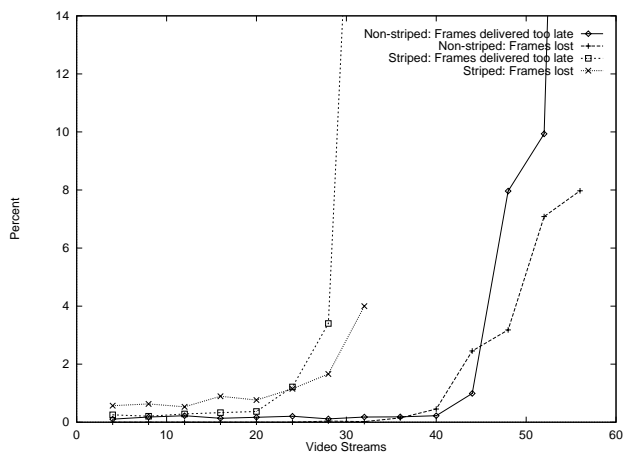


Figure 7. Delayed and lost video frames for non-striped and striped video allocations

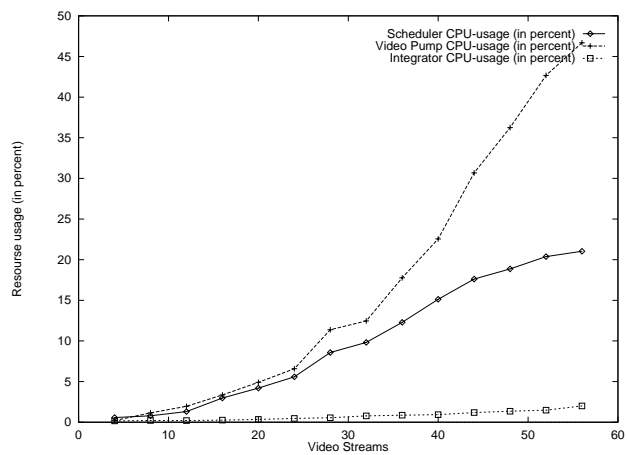


Figure 8. The CPU load measured on one of the nodes using non-striped video allocation

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. 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.

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.

### Non-striped Video Allocation

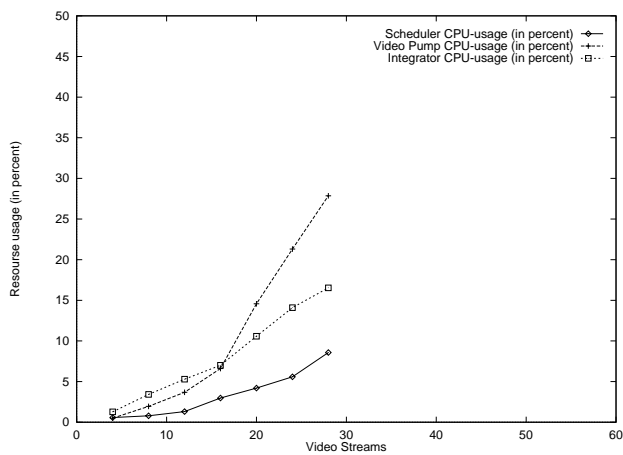
The results from the experiment with non-striped video allocation are given by two of the curves in Figure 7. The number of video frames which are delivered too late is insignificant until it reaches a load of about 40 simultaneous video streams. When we increase the number of simultaneous video streams to more than 44, the number of delayed and lost video frames increases rapidly. The limiting resource is the bandwidth of the disk. Unfortunately the current design does not utilize more than one disk efficiently because the disk scheduler process is single threaded and do not make use of asynchronous I/O. 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 8.

### Striped Video Allocation

Figure 7 also contains the number of delayed and lost video frames during delivery of video striped on all nodes in the server. When using this allocation strategy, the server manages to deliver about 20 simultaneous videos. This is about half of what the server managed when we used non-striped allocation of the 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 IP-protocol and ATM 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 9. Compared to Figure 8, the CPU resources used by the Integrator are much higher when the video is striped on several nodes, and has to be integrated into a single video stream before being delivered to the user.

If we compare the results from these measurements we see that 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. The main reason is that the design of the server using the Integrator doubles the internal communication load when using striped allocation of videos.



**Figure 9. The CPU load measured on one of the nodes using striped video allocation**

On the positive side, 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 this 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.

### 5.1. Practical Use of the Video Server

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

#### VideoSTAR

The video server is integrated into the VideoSTAR [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 [14].

### Television News on the Internet

We are participating in the LAVA<sup>1</sup> project to make television news from the Norwegian Broadcasting Corporation available on the Internet[3]. Currently, we make the daily news program Nordnytt (News from the North) available to Internet users in Norway. 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 [9].

## 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 of multiple video 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 for this is that our design uses a separate process to integrate the fragments from each of the video pumps into one stream before sending the video to the client. This is resource demanding.

Based on the experiences with the current version of Elvira, we are now working on an improved design based on the experiences with the current version of Elvira. Because of the big overhead using the Integrator to synchronize the video delivery, we are removing it from the design. 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 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 will replace today's processes with threads. To better utilize the available disk bandwidth we will use several disk threads per node. To get better control

<sup>1</sup>LAVA is project carried out in cooperation between Norwegian Computing Center, NORUT IT, Department of Computer and Information Science, NTNU, and is funded by the Norwegian Research Council.

of the disk scheduling and video layout on the disks we will use raw disks instead of going through the Unix file system. Further we will increase the unit of striping from being individual video and audio frames to be about one second of video.

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 of writing, we are extending Elvira to use magnetic tape as near-line video storage. The disks will be used as a video cache to the tape storage. Movies which are not stored on disk will automatically be read from tape to the disks when they are requested by users.

## References

- [1] S. Berson, S. Ghandeharizadeh, R. R. Muntz, and X. Ju. Staggered striping in multimedia information systems. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, pages 79–90, Minneapolis, Minnesota, May 1994.
- [2] 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 1996.
- [3] H. Bryhni, H. Lovett, E. Maartmann-Moe, D. Solvoll, and T. Sørensen. On-demand regional television over the Internet. In *Proceedings of ACM Multimedia '96*, pages 99–107, Boston, Massachusetts, November 1996.
- [4] S. T. Campbell and S. M. Chung. The role of database systems in the management of multimedia information. In *Proceedings of International Workshop on Multi-Media Database Management Systems*, pages 4–11, Blue Mountain Lake, New York, August 1995.
- [5] 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.
- [6] S. Chaudhuri, S. Ghandeharizadeh, and C. Shahabi. Avoiding retrieval contention for composite multimedia objects. In *Proceedings of the 21st VLDB Conference*, pages 287–298, Zurich, Switzerland, September 1995.
- [7] 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 1995.
- [8] A. L. Chervenak, D. A. Patterson, and R. H. Katz. Choosing the best storage system for video service. In *Proceedings of ACM Multimedia '95*, pages 109–119, San Francisco, California, November 1995.
- [9] 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*, pages 296–301, Washington DC, May 1995.
- [10] D. Delodderre, W. Verbiest, and H. Verhille. Interactive Video On Demand. *IEEE Communications Magazine*, 32(5):82–88, May 1994.
- [11] C. Federighi and L. A. Rowe. A distributed hierarchical storage manager for a video-on-demand system. In *Proceedings of Storage and Retrieval for Image and Video Databases II, IS&T/SPIE Symposium on Electronic Imaging Science and Technology*, pages 185–197, San Jose, California, February 1994.
- [12] 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 1996.
- [13] D. Gemmel et al. Multimedia Storage Servers: A Tutorial. *IEEE Computer*, pages 40–49, May 1995.
- [14] R. Hjelmsvold, S. Langørgen, R. Midtstraum, and O. Sandstå. Integrated Video Archive Tools. In *Proceedings of ACM Multimedia '95*, pages 283–293, San Francisco, California, November 1995.
- [15] R. Hjelmsvold and R. Midtstraum. Modelling and Querying Video Data. In *Proceedings of the 20th VLDB Conference*, pages 686–694, Santiago, Chile, September 1994.
- [16] R. Hjelmsvold, 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. Hjelmsvold, R. Midtstraum, and O. Sandstå. Searching and browsing a shared video database. In K. C. Nwosu, B. Thuringham, and P. B. Berra, editors, *Multimedia Database Systems*, chapter 4, pages 89–122. Kluwer Academic Publishers, 1996.
- [18] R. Koteng. Video server with VCR-functionality for MPEG-1. Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway, December 1996. In Norwegian.
- [19] T.-G. Kwon and S. Lee. Data placement for continuous media in multimedia DBMS. In *Proceedings of the International Workshop on Multi-Media Database Management Systems*, pages 110–117, Blue Mountain Lake, New York, August 1995.
- [20] S. Langørgen. Elvira – Experimental video server for ATM. Master's thesis, Norwegian Institute of Technology, Trondheim, Norway, December 1994. In Norwegian.
- [21] 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, Taipei, Taiwan, March 1995. IEEE.
- [22] A. Laursen, J. Olkin, and M. Porter. Oracle Media Server: Providing Consumer Based Interactive Access to Multimedia Data. In *Proceedings of the 1994 ACM SIGMOD*, pages 470–477, Minneapolis, Minnesota, May 1994.
- [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] C. Martin, P. S. Narayanan, B. Özden, R. Rastogi, and A. Silberschatz. The Fellini multimedia storage server. In S. M. Chung, editor, *Multimedia Information Storage and Management*, chapter 5, pages 117–146. Kluwer Academic Publishers, 1996.
- [26] F. Moser, A. Kraiß, and W. Klas. L/MRP: A buffer management strategy for interactive continuous data flows in a multimedia DBMS. In *Proceedings of the 21st VLDB Conference*, pages 275–286, Zurich, Switzerland, September 1995.
- [27] MPEG-1. *Coding of moving pictures and associated audio for digital storage media up to about 1,5 Mbit/s*. ISO 11172. International Organization for Standardization, 1992.
- [28] 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 1995.
- [29] B. Özden, R. Rastogi, and A. Silberschatz. Buffer replacement algorithms for multimedia databases. In S. M. Chung, editor, *Multimedia Information Storage and Management*, chapter 7, pages 163–182. Kluwer Academic Publishers, 1996.
- [30] B. Özden, R. Rastogi, and A. Silberschatz. Disk striping in video server environments. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 580–589, Hiroshima, Japan, June 1996. IEEE.
- [31] R. Rooholamini and V. Cherkassky. ATM-Based Multimedia Servers. *IEEE Multimedia*, pages 39–52, Spring 1995.
- [32] S. Sahu, Z.-L. Zhang, J. Kurose, and Towsley. On the efficient retrieval of VBR video in a multimedia server. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 46–53, Ottawa, Canada, June 1997.
- [33] 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.
- [34] S. L. Vieira, N. R. Gomes, and A. Teixeira. A distributed MPEG player with jitter control. In *Proceedings of XVI International Conference of the Chilean Computer Science Society*, pages 155–164, Valdivia, Chile, November 1996.
- [35] Xing Technology Corporation. Streamworks technical description, <http://www.xingtech.com/>.
- [36] P. S. Yu, M.-S. Chen, and D. D. Kandlur. Design and analysis of a grouped sweeping scheme for multimedia storage management. In P. V. Rangan, editor, *Proceedings of Third International Workshop on Network and Operating System Support for Digital Audio and Video*, number 712 in Lecture Notes in Computer Science, pages 44–55, La Jolla, California, November 1992. Springer.