Performance of A Mass Storage System for Video-On-Demand

Jenwei Hsieh, Mengjou Lin, Jonathan C.L. Liu, and **David H.C. Du**
Distributed Multimedia Center[1]
University of Minnesota

Thomas M. Ruwart[2]
Army High Performance Computing Research Center
University of Minnesota

Abstract

Advancements in storage technology along with the fast deployment of high-speed networks has allowed the storage, transmission and manipulation of multimedia information such as text, graphics, still images, video and audio to be feasible. Our study focused on the performance of the mass storage system for a large-scale video-on-demand server. Different video file stripping schemes, such as application level stripping and device driver level stripping, were examined in order to study scalability and performance issues. To study the impact of different concurrent access patterns on the performance of a server, experimental results were obtained on group access on a single video file and multiple group accesses on multiple video files. All of our experiments were conducted on a fully configured Silicon Graphics Inc. Onyx computer system. The Onyx machine was connected to 31 SCSI-2 channels which have 496 Gigabyte disk storage, 20 MIPS R4400 processors, and 768 MByte main memory. From the experimental results, the storage system of Onyx machine can potentially provide about 360 concurrent video accesses with guaranteed quality of service.

Keywords: Multimedia, Video on demand, Mass storage system, Disk array, Data stripping.

Submitted to Journal of Parallel and Distributed Computing:
Special Issue on Multimedia Processing and Technology

[1]Distributed Multimedia Center (DMC) is sponsored by US WEST, Honeywell, IVI Publishing, Computing Devices International and Network Systems Corporation.

[2]This work was partially supported by contract no. DAAL02-89-C-0038 between the Army Research Office and the University of Minnesota for the Army High Performance Computing Research Center and by the grant number 5555-23 from the University Space Research Association which is administered by NASA's Center for Excellence in Space Data and Information Sciences (CESDIS) at the NASA Goddard Space Flight Center.

B

1 Introduction

It is widely agreed among researchers in these recent years that providing a guaranteed video delivery is an essential element to support future *distributed multimedia computing* applications. Solving the storage and retrieval of continuous media, especially video medium, will introduce the integration of traditional text and image data. In order to make these exciting multimedia applications feasible, the common Video-On-Demand (VOD) service needs to be designed. In a VOD system, geographically distributed users can interactively access video files from a remote VOD server. Almost every distributed multimedia application needs this service to store and retrieve the video files for their own purposes.

Although the rapid improvement of storage technology combined with emerging high-speed networks has made the VOD service feasible. To provide a quality-guaranteed video delivery still impose some technical challenges. To support a large number of users[3] in a large-scale VOD system, the following considerations become essential:

- The video medium: video files are different from traditional data files in several ways. *First*, video files are much larger than traditional data files. For example, a two hour long movie in MPEG and MPEG-II formats needs approximately 1.35 Gigabyte and 7.2 Gigabyte amount of storage (based on 1.5 Mbits/sec for MPEG and 8 Mbits/sec for MPEG-II), respectively. *Second*, all video streams must be delivered with a guaranteed quality of service in a continuous fashion during the entire session.
- A mass storage system based on secondary storage devices such as optical tapes/disks, hard disk drives and CD-ROM drives to store video files,
- a powerful machine with sufficient processing CPUs and efficient I/O subsystem which can handle hundreds of concurrent video accesses, and
- a high-speed network which supports multiple simultaneous video transmissions with guaranteed quality of service.

Figure 1 shows a possible VOD architecture which consists of *playback devices* on the end-user sides, a *network infrastructure*, and a *VOD server*. A playback device may consist of a display screen (high-resolution workstation or high-definition television), an interactive control device, and a network connection port. The network infrastructure provides the required interconnectivities between the VOD server and its users (clients). The emerging switch-based high-speed networks such as Asynchronous Transfer Mode (ATM) [4], HIgh Performance Parallel Interface (HIPPI) [3], and Fibre Channel (FC) [2] may be used for such connections since they have the potential to satisfy the huge amount of network bandwidth required for multimedia communications with guaranteed network services.

[3] In this paper, *users*, *clients*, *streams*, and *concurrent accesses* are used equivalently.

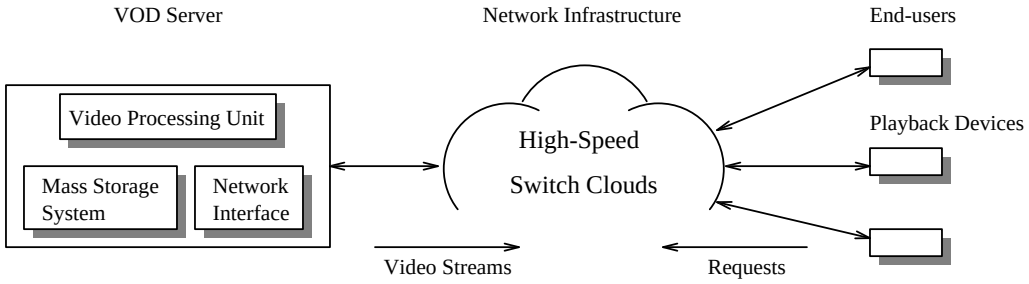


Figure 1: A VOD system architecture

The VOD server contains three major components, an interface to *a mass storage system*, a *video processing unit*, and a *network interface*. The interface to the storage system must be capable of handling multiple concurrent I/O accesses. The video processing unit supports multiple video retrievals. It requires fast processing speed, large memory space, and enormous system bus bandwidth to handle the in and out traffic from storage system to the network. The network interface should support multiple video streams with guaranteed quality of service. Although they are equally important, we mainly focus on the design and performance issues of the mass storage system in this paper.

VOD server design issues have engendered many research activities during the past few years [1, 10, 11, 18, 20, 21, 32, 33, 13, 12, 5, 23, 30, 31, 6, 36, 14, 15]. Related design issues such as real-time support for delay-sensitive retrieval, disk layout strategies for continuous media, admission control for new requests, and disk scheduling schemes have been studied by the researchers. Among these studies, single disk storage system was assumed in [20, 32, 21, 33, 13, 12, 1, 5]. Researchers in the video storage community generally agreed that a single disk system only can provide a limited number of concurrent accesses even with effective buffering and placement schemes. Therefore, in order to provide a large number of concurrent accesses, the large-scale VOD server should be equipped with a mass storage system with multiple disk systems. Some VOD server design based on the multiple-disk systems have been reported in [23, 18, 30, 31, 6, 36, 14, 15] In these studies, multiple disks are usually connected to the same SCSI bus, or 2 SCSI buses in [18]. However, most of the studies are based on analytical models and only verified by simulations.

Another alternative solution to design the mass storage system of a video server is to utilize a RAID (Redundant Arrays of Inexpensive Disks) architecture. There are only few reported paper in [31, 30] that investigated this solution. However, this study only covered the basic stripping method which only represented one solution based on RAID architecture. No experimental results were reported in this study. Unlike previous researchers, we have devoted considerable amount of effort to put together a powerful server with huge amount disk capacity. Experiments have been performed on this real large-scale server to obtain the experimental results with practical value.

Disk arrays have evolved as an effective solution to the I/O bottleneck plaguing many com-

puter systems today [7, 17]. Arrays of fast, small form-factor, low-power drives [26] can provide reliable, cost-effective, high-capacity, high-bandwidth physical I/O devices [16, 19]. Recent studies have focused on the fundamental issues of RAID, including parity placement [17, 29], optimal block sizes [7], and array performance under varying workloads and organizations [7, 8, 22]. However, it is basically still unexplored in the suitability of RAIDs to support the a large-scale VOD server. In this paper, RAID 3 architecture is adopted for the study. One issue is worthy to address is that give a large-scale VOD server with hundreds of video files involved, it is difficult to use a single RAID 3. Basically, a single RAID 3 would be a severe limitation on the application environments where file transfers on the order of hundred of megabytes or more are becoming commonplace [34]. Given the availability of standard interfaces available on the arrays [9] and high-end workstations, a natural extension is to employ multiple RAID 3 devices to achieve the necessary performance and capacity. We shall refer to such a system as array of RAID 3s in this paper.

In order to provide more concurrent accesses from the mass storage system of a large-scale server, it is important to consider different allocation schemes among these RAID 3s. Good allocation schemes will provide more concurrent accesses with guaranteed delivery quality. In a storage system like an array of RAID 3s, video files can be allocated according to different stripping schemes. Each stripping scheme distributes a video file to the storage space according to different arrangement schemes. Three stripping schemes has been studied in this paper. These stripping schemes exploit the parallelism and concurrency of the storage system to different degrees.

First, video files can be placed on a single byte-striped RAID 3 device, which is called *RAID 3 byte stripping*. The RAID 3 byte stripping scheme utilizes the parallelism of multiple disk drives within a RAID 3 to achieve aggregate disk bandwidth, thus providing more concurrent accesses. The advantage of this scheme is its simplicity and feasibility because RAID 3 is commercially available. However, it is still unclear that how well a RAID 3 disk array can support concurrent retrieval of guaranteed video delivery? Our study shows that a RAID 3 disk array can provide about 14 steady concurrent accesses with 512KB request size, which contains 16 video frames based on 32KB per video frame. However, as we pointed out earlier, multiple RAID 3s should be adopted for a large-scale video server because the experimental results justified the limitation of using a single RAID as the mass storage system.

Next, we have investigated the methods to allocate video files on multiple RAID 3 disk arrays. It is found that, particular in SGI ONYX platform, several RAID 3s can be controlled by a single device driver called *logical volume*. A logical volume behaves like a traditional disk partition with each partition in a separate RAID 3. A contiguous stream of data (a video file) can be divided into blocks of data and distributed in a round-robin fashion to several RAID 3s in a logical volume. We call this stripping scheme as *logical volume stripping* in this paper. *Logical volume stripping* offers a simple solution for a server to allocate and transfer video blocks concurrently from multiple RAID 3 disk arrays. The experimental result shows that using *logical volume stripping* with 4 RAID 3 can support around 40 concurrent accesses of guaranteed video delivery, which is about 71% of the achievable aggregated numbers for four RAID 3 disk arrays. And the limitation of fixed already-large buffer size makes the *logical volume stripping* even

worse to around 38% with eight RAID 3 disk arrays. Compared to the *RAID 3 byte stripping*, the *logical volume stripping* suffers a greater latency variation. The reason for this anomaly is because the requested data will not be completed until all the sub-requests are finished from all RAID 3 in the associated logical volume. Since all the multiple disk arrays will be blocked for this parallel transfer, the contention on the device driver and physical devices results in the bottleneck when supporting more concurrent accesses.

Third, apparently a more effective allocation scheme has to be designed to increase the achievable number of concurrent accesses with multiple RAID 3 disk arrays. The *logical volume stripping* suffers the contention on the logical volume device driver. It is identified by us that the concurrent accesses should be distributed among these multiple RAID 3 disk arrays. The load balancing by distributing the concurrent accesses should improve the efficiency. In order to achieve this load balancing, we propose a new allocation scheme called *application level stripping*. The motivation of *application level stripping* came from the following observation. In the application level stripping scheme, the data file was also divided into blocks and stored on the constituted storage devices (logical volumes or RAID 3s) in a round-robin fashion. However, by declustering a video file across multiple RAID 3s, each video retrieval process (i.e., *application*) should retrieve the video blocks in a pipelining manner (i.e., access only one RAID 3 or logical volume at one time). Experimental results show that potentially around 360 concurrent accesses with guaranteed quality can be supported by adopting this novel allocation scheme. This result was obtained with 30 RAID 3 disk arrays, and achieved 86% of the maximal achievable concurrent accesses for this configuration.

The performance study of a large-scale VOD server can not be completed by joint considerations of some related issues. Some video files will become popular such that attracts more concurrent accesses (e.g., a new released movie title) to a video file. Thus, performance of the above three allocation schemes has been measured for the support of concurrent accesses on a single video file. On the other hand, it is a common scenario that different accesses will retrieve different video files concurrently. Therefore, we extend our experiments on the performance comparisons of different allocation schemes to support this multiple files extension. We also believe that building a large-server VOD server might not be fully configured at once. For example, usually a video server will be configured with 2 RAID 3 disk arrays for initial setup. Then more RAID 3s can be added to serve the increasing demands of concurrent accesses once the servicing area has been expanded. Therefore, server scalability is an issue needs to be measured. The server scalability was tested by increasing the number of concurrent video accesses and the number of RAID 3s.

A common thought that *scheduling policies and buffering schemes are critical to the performance of VOD server* has been agreed in the *multimedia computing* community. However, for our best knowledge, there is no existed experimental results in reporting the degree of performance of degradation by the lack of special scheduling polices for continuous media. We are among the first few groups that perform this large-scale VOD experiments. It is our belief that even without special scheduling polices designed for a VOD server, a fairly good performance should be achieved by a general large-scale VOD server such like SGI's ONYX machine. This speculation has been justified through this experimental study. From the experimental results,

the storage system of Onyx machine can potentially provide about 360 concurrent video accesses with guaranteed quality of service.

The remainder of this paper is organized as follows. We describe our experimental environment and introduce the three levels of video file stripping schemes in Section 2. A simple experiment was also used to show the infeasibility of the conventional file system in the VOD application. We describe the experimental results on supporting concurrent accesses on a single video file in Section 3. Section 4 covers the discussion on the performance results when multiple video files are retrieved concurrently. The performance of different allocation schemes are illustrated and compared in these two sections. We list the related studies in Section 5, and finally, Section 6 concludes this study and discuss our future directions.

2 A Experimental Large-scale VOD Server

In order to conduct our experiments, a Silicon Graphics's (SGI) Onyx symmetric multiprocessor computer was chosen because of its processing power, memory bandwidth, I/O transfer rate, and extensibility. The SGI Onyx computer is connected to a mass storage system consisting of an array of RAID 3s. Each RAID 3 is controlled by Ciprico 6710 [9] controller with 8 data + 1 parity Seagate ST12400N 2.1 Gigabyte (formatted) drives [26]. Onyx machines are characterized as shared memory symmetric multiprocessing computer architectures. Up to 36 processors and 8-way interleaving memory could be connected to the 1.2 Gigabytes/sec system bus. A fully configured Onyx machine could support up to 32 fast-wide SCSI-2 (Small Computer System Interface) channels (each with 20 MBytes/sec I/O bandwidth). Therefore, the I/O subsystem can theoretically provide up to 640 MBytes/sec bandwidth. The array of RAID 3s can provide large storage capacity, aggregated I/O bandwidth, and both high reliability and availability. Since each fast-wide SCSI-2 channel connects to a RAID 3 disk array with 16 Gigabytes storage capacity (8 disk drives, each has 2 Gigabytes space), the total disk storage will be 512 Gigabytes. The aggregated I/O bandwidth of 8 disk drives in each disk array can fully utilize the bandwidth of a SCSI-2 channel.

We currently have a SGI Onyx machine which is connected to a storage system of 8 RAID 3s at the Army High Performance Computing Research Center (AHPCRC), University of Minnesota. We have reported an aggregated 100 MBytes/sec I/O bandwidth for the SGI Onyx machine using RAID 3 disk arrays [25]. However, a fully configured Onyx machine can connect up to 32 RAID 3s via 32 SCSI-2 channels. To study the VOD application, we feel a fully configurated Onyx is necessary. With assistance from Ciprico and SGI we conducted the Maximum Achievable Xfer (MAX) project [24]. The results presented in this paper were obtained by this project. We also would like to experiment the maximum achievable throughput which could be conveyed by the fully configured Onyx machine. An early experimental result has revealed around 510 MBytes/sec sustained concurrent I/O bandwidth [24].

Our Onyx machine connected to eight RAID 3s instead of the maximum 32 RAID 3s. However, the cost of putting together the other 23 RAID 3s (one SCSI-2 is used as the system disk) to push the limits of the I/O bandwidth was expensive and could not be done for this

experiment. Therefore, we chose the approach of using disk array controllers to simulate the operations of real disk arrays. With the help from Ciprico, 23 extra Ciprico RAID 3 controllers with 23 extra fast and wide SCSI-2 channels are added to our system. Each Ciprico controller is programmed to simulate disk operations. An appropriate delay is inserted during the processing of disk requests to simulate the seek and rotational latencies. We examine the performance difference between the real RAID 3s and the hardware-simulated RAID 3s in Section 4.

In order to understand the basic configuration of the Onyx machine, a brief description of its system architecture is given in Section 2.1. The hardware and software components of the storage subsystem are introduced in Sections 2.2 and 2.3. The three level of data stripping schemes are presented in Section 2.4. In Section 2.5, we discuss the problem of accessing video files through a conventional file system. The characteristics and performance differences when accessing video files through the conventional file system and the lower level device drivers directly are compared.

2.1 SGI Onyx System Architecture

The configuration of the Onyx machine is presented in Figure 2. The system bus can sustain 1.2 GBytes/sec bandwidth. Up to 32 RISC processors can be connected to the system bus [27]. With memory interleaving, the memory subsystem can support low-latency read/write transactions. A brief description of the main components are presented as follows:

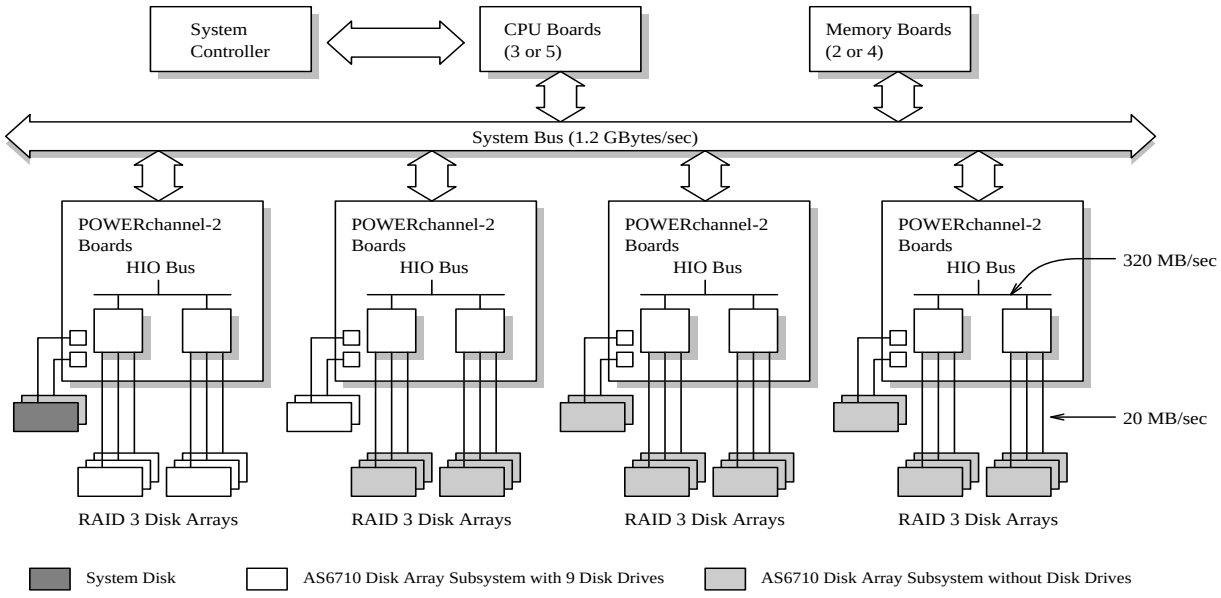


Figure 2: The Onyx hardware components

- *System Bus* (called *POWERpath-2* or *E-bus*): The E-bus supports up to 32 MIPS R4400 processors, a low-latency memory system, high-speed I/O devices, and high-performance graphics subsystems. Read transactions are split to support multiple outstanding reads,

and a wide range of response times are tolerated. The bus is synchronous and uses separate address and data lines. A 256-bit data bus is dedicated to data transfer.

- *Processor Architecture:* The CPU board is populated with 4 MIPS R4400 150 MHz processors. Up to 8 CPU boards can be connected to the E-bus; therefore, a total of 32 R4400 processors could be configured on an Onyx machine. In our Onyx configuration, we installed either 12 processors (3 CPU boards) or 20 processors (5 CPU boards). The reason to reduce the number of CPUs was because we ran out of board slots when configuring the 8-way memory interleaving.

- *Memory subsystem:* The E-bus interleaved memory board is designed to provide low-latency memory response and high bandwidth. Interleaving is a technique for organizing memory into *leaves* (memory banks) that increases the sustainable memory bandwidth. Each leaf can process a memory request for a processor independently. The latency of the DRAM access, which is long compared with the CPU clock rate, is hidden from the processor when overlapped memory access are initiated in multiple memory leaves. Each memory board supports two-way interleaving, and each leaf can transfer an entire cache block, which is the unit of interleaving. Two memory boards support four-way memory interleaving; four memory boards support eight-way interleaving. Both 4-way (with 768 MBytes RAM) and 8-way (with 512 MBytes RAM) memory interleaving were used in this experiment. We found the performance of 8-way memory interleaving better than that of 4-way memory interleaving.

- *I/O subsystem:* The Onyx machine can be configurated with one to four POWERchannel-2 I/O adapter boards (called IO4), which plug into the E-Bus directly. Two daughter boards can be plugged into an IO4 to allow expansion and customization. These daughter boards allow I/O devices to connect to the 64-bit wide 320 MBytes/sec HIO bus. The IO4 contains two native 16-bit fast SCSI-2 controllers. Each controller can operate with a bandwidth of up to 20 MBytes/sec. To accommodate extra disk controllers, each SCSI module contains three 16-bit fast SCSI-2 controllers. At most eight SCSI-2 channels can be supported by an IO4. A fully expanded IO4 can deliver 160 MBytes/sec of SCSI bandwidth. Thus, the maximum configuration, with four IO4 boards, can deliver up to 640 MBytes/sec disk bandwidth theoretically.

2.2 The Mass Storage System

2.2.1 Hardware Architecture

Each 16-bit fast SCSI-2 controller connects to one Ciprico RF6710 RAID 3 which consists of 8 data drives, 1 parity drive, and a disk array controller [9] (see Figure 3). The disks used in our RF6710 are the Seagate ST12400N 2.1 GByte disks[26]. A fully configured Onyx machine can support 32 16-bit SCSI-2 channels. A total of 288 individual disk drives are needed (32×9, 8 data drives and 1 parity drive) to fully populate the 32 disk arrays. Because the number of disks required is more than we could purchase or borrow, there will be no disks on the other 23

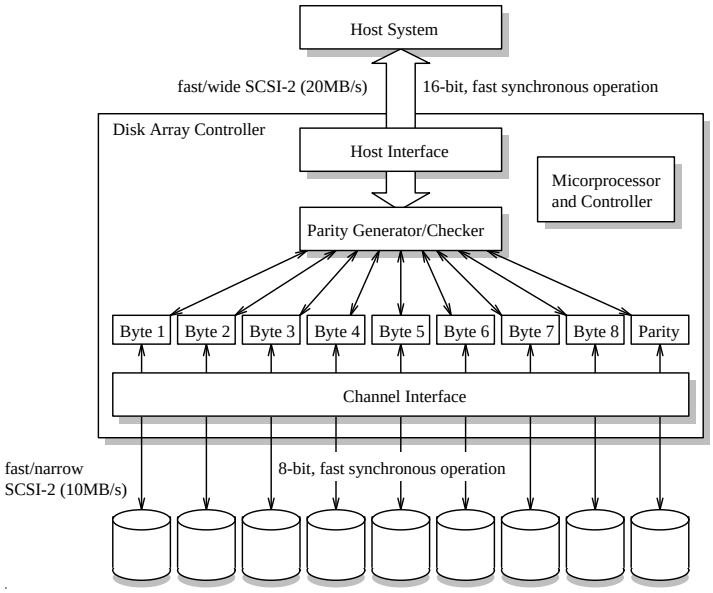


Figure 3: The Ciprico RF6710 disk Array subsystem (byte-stripping RAID 3 storage)

controllers (Our Onyx has 8 real RF6710 RAID 3s and one system disk) but they will perform like real disk arrays. They will read and write data as any disk array would with the exception that data written to the pseudo disk arrays is thrown away and data read from them is always zero. Since this is strictly a performance test, none of the data read or written to the pseudo disk arrays is examined because the data content is irrelevant.

The performance of these disk array controllers depends on the type of access. For purely sequential access the seek and rotational latencies are zero. For any other access that involves a seek, an appropriate delay is inserted in the command processing to simulate the seek and rotational latencies. The seek time is estimated to be proportional to the seek distance and the rotational delay is equal to half a revolution (4.1 milliseconds in this case.) We examine the performance difference between real and pseudo disk arrays in turn of the number of supportable video streams in Section 4.

The hardware configuration of the tested SGI Onyx machine is summarized as follows:

Processors	12 150 MHz R4400s (8-way memory-interleaving) or 20 150 MHz R4400s (4-way memory-interleaving)
Main Memory	512 MBytes (8-way memory-interleaving) or 768 MBytes (4-way memory-interleaving)
POWERchannel-2	4
Fast/wide SCSI-2 Channels	1 (connect to system disk) 8 (connect to real disk arrays) 23 (connect to pseudo disk arrays)
System Disk	2 GBytes
Ciprico RF6710 RAID 3s	128 GBytes (real disk arrays) 368 GBytes (pseudo disk arrays)

2.2.2 Software Architecture

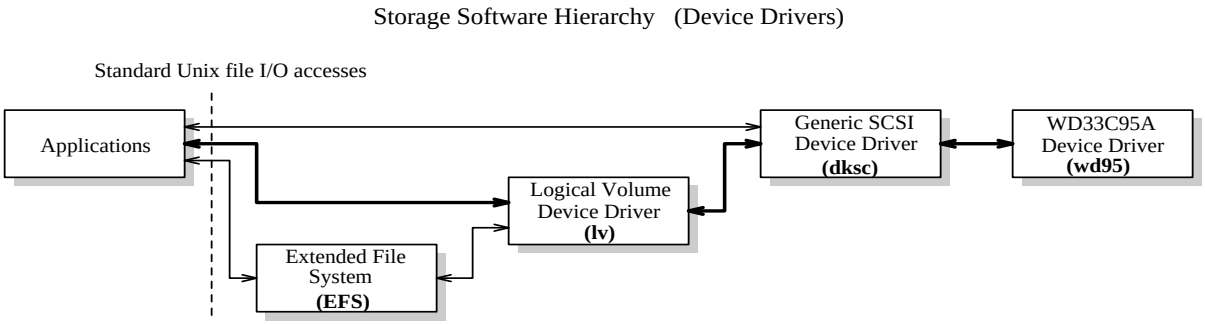


Figure 4: The Onyx software hierarchy of storage subsystem

Our Onyx runs the IRIX 5.2 operating system, a fully symmetric, multiprocessing Unix System V derivative. Figure 4 shows the software hierarchy of the storage subsystem. In this hierarchy, the Extent File System (**EFS**) is a modified standard Unix file system. It contains an enhancement called *extents*. Extents are the data blocks that make up a file. **EFS** sits on top of Logical Volume (**lv**) device driver. **lv** implements a basic stripping algorithm to allow parallel access to multiple disk devices. The **dksc** driver is the generic SCSI disk driver. Applications can access data via either **EFS**, **lv**, or **dksc**. When accessing the **lv** or **dksc** drivers, applications view the disk device as a large continuous file. We will discuss the tradeoff of accessing video files via **EFS** and **lv** in Section 2.5. The **wd95** device driver implements the controller specific interface to the Western Digital WD33C95A fast/wide SCSI-2 controller chip.

An example read I/O request issued via the **lv** device driver would traverse the following path (the highlighted path in Figure 4) through the software hierarchy and hardware components.

1. The **Application** issues a `read()` system call. The operating system passes the I/O request to the **lv** device driver.

2. The **lv** device driver determines the number of devices involved in the operation, starting device (disk or disk array), starting location on each device, and the length of each oper-

ation. For each of the devices involved in the I/O request, **lv** generates a sub-request and calls the **dksc** device driver to start an I/O operation.

3. **dksc** determines the physical starting location on the device and passes each sub-request to the **wd95** driver.

4. The **wd95** driver has two functions:

   - Program the IO4 page mapping registers with the physical memory addresses of the I/O buffer.
   - Program the appropriate host adapter (a Western Digital WD33C95A).

   Once this is done the host adapter is started and the I/O request is sent to a RF6710 disk array.

5. The array controller reads the request information and proceeds to send the appropriate requests to the individual disk drives within the RF6710.

Eventually, the data begins to flow from the disks, through the RF6710 array controller, across the fast/wide SCSI bus, through the WD33C95 host adapter where the IO4 page mapping hardware redirects the continuous data stream to their final physical memory locations. After the data transfer completes, an I/O interrupt is generated by the host adapter which is processed by the **wd95** and **dksc** drivers. At this point, under normal conditions, the I/O request is marked done, and control is returned to the **Application**.

2.2.3 Data Stripping Hierarchy

In the storage subsystem, data is distributed on multiple disk drives or disk arrays in special ways to employ the flexibility and aggregated I/O bandwidth of the storage devices. There are three levels of data stripping schemes for storing video files, *RAID 3 byte stripping*, *logical volume stripping*, and *application-level stripping*. These data stripping schemes comprise the data stripping hierarchy in our storage subsystem.

RAID 3 Byte Stripping

In the RAID 3 byte stripping scheme, data is conceptually interleaved byte-wise over multiple data disks, and a single parity disk is used to tolerate any single disk failure. The bottom left part of Figure 5 illustrates how RAID 3 byte stripping was implemented on a disk array with $8 + 1$ disk drives (8 data drives and 1 parity drive). The stream of data was chopped into bytes and distributed to the 8 disk drives in a round-robin fashion. For every 8 bytes of data, the disk array controller generates one parity byte for fault tolerance. Although the figure only shows the write operation of the RAID 3 disk array, the read operation was performed in a similar way such that the disk array controller constructs the stream of data from all of the disk drives and verifies the integrity of the data by checking the parity bytes.

The RAID 3 byte stripping scheme utilizes the parallelism of multiple disk drives to achieve aggregate disk bandwidth and provide fault tolerance for single disk failure. For each disk operation, all of the constituted disk drives are used. Since each disk drive has its own SCSI channel to connect with the disk array controller (see Figure 3), the disk array provides the aggregate disk bandwidth of multiple disk drives. With the redundancy of disk drives, the disk array can tolerate single disk failure. Any single disk failure was hidden from the user by employing the parity disk to generate the corresponding bytes on the fly from the failed disk drive. The failed disk drive can be replaced during the operation of disk array.

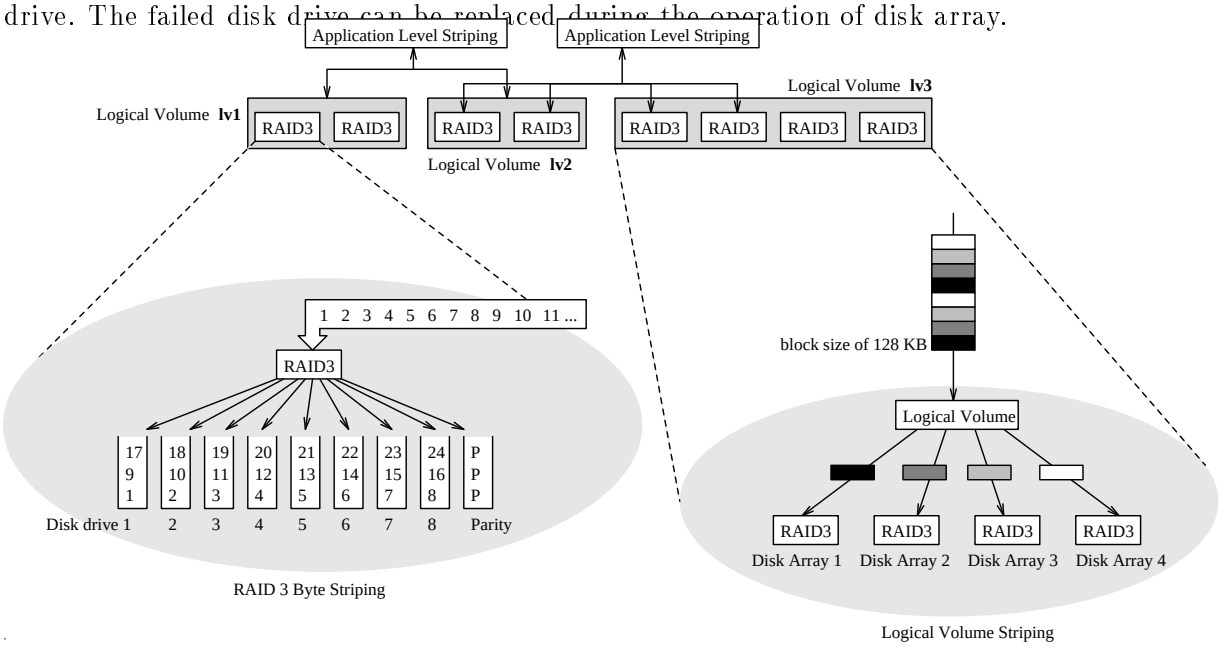


Figure 5: The three level od data stripping schemes.

Logical Volume Stripping

A logical volume is a storage entity which behaves like a traditional disk partition, but its storage may span several physical devices. In our case, the physical devices are RAID 3s. Two parameters are used to define the configuration of a logical volume, *stripping size* and *stripping granularity (step size)*.

- *stripping size*: The value of the stripping size specifies the number of physical storage devices used in the logical volume. For example, there are three logical volumes in Figure 5: **lv1**, **lv2**, and **lv3**. The stripping size of **lv1** and **lv2** is 2, while the stripping size of **lv3** is 4.

- *stripping granularity (step size)*: For a contiguous stream of data that spans multiple disk arrays within a logical volume, the stripping granularity (or step size) specifies the maximum amount of data that is transferred to or from one RAID 3 before switching to the next RAID 3. The logical volume **lv3** in Figure 5 has step size of 128 MBytes. The

stream of data was divided into blocks of 128 MBytes and distributed to the constituent disk arrays.

In the logical volume stripping scheme, a contiguous stream of data was divided into blocks of data and distributed to multiple disk arrays in a round-robin style. Unlike the RAID 3 byte stripping scheme, the logical volume stripping scheme may not employ all the participated disk arrays for every disk operation. The number of disk arrays which are utilized in a disk operation depends on the request size and the stripping granularity. For example, the logical volume **lv3** in Figure 5 uses 4 disk arrays (stripping size is 4) with stripping granularity of 128 KBytes. If the request size sent to **lv3** less than or equal to 128 KBytes only *disk array 1* will be used. All 4 disk arrays will be employed only for those operations which request more than 384 KBytes of data. For request size less than or equal to 384 KBytes, some of the disk arrays remain idle when the logical volume is servicing the request.

Application Level Stripping

The application level stripping scheme implements another level of data stripping on top of logical volumes or disk arrays. The two data stripping schemes mentioned above provide a service abstraction for users and handle the storing and retrieval of data. On the contrary, the application level stripping scheme requires applications to handle the storing and retrieval of data. This means that the applications know where to retrieve and store data. For example, there are two application level stripping entities in Figure 5. The left one employs two logical volumes (**lv1** and **lv2**) and the right one employs 4 RAID 3s to implement the application level stripping.

In the application level stripping scheme, the data file was also divided into blocks and stored on the constituted storage devices (logical volumes or disk arrays) in a round-robin fashion. To retrieve the data, each user accesses one segment of the data file from one of the storage devices at a time. This access method allows other users to access different segments of the same data file with the same application level stripping scheme. This stripping scheme has the following advantages:

- The application level stripping can be implemented on disk arrays or logical volumes. It may employ multiple logical volumes with different stripping sizes and stripping granularities.
- Since the constituted storage devices are not controlled by a single hardware or software component (disk array controller or logical volume device driver). A higher utilization of the storage devices can be achieved by allowing several users to access the devices with the same stripping scheme.
- Storing a video file across multiple devices with this stripping scheme can potentially *increase the number of concurrent accesses to the same video file*. From our experiments, the application-level stripping can sustain more concurrent accesses of one single video file than other two schemes.

2.3 Extended File System vs Logical Volume Driver

As we pointed out previously, the application can access either **EFS** or character device driver (**lv** or **dksc**) directly. Normal **EFS** I/O utilizes a delayed write and read ahead mechanism whereby data to and from disk file are buffered in memory. The IRIX operating system provides an integrated page and data cache, in which the size of the buffer cache can grow and shrink with I/O demands. The cache design favors a usage pattern in which blocks written to disk are frequently read back in, modified, and written back out. However, this caching system requires one additional data copy from user space to kernel space. When reading and writing large sequential files such as video files, the amount of extra copies causes significant performance degradation.

To overcome this problem, SGI provides another **EFS** I/O option, *Direct I/O*, which allows I/O operations to bypass the RAM buffer cache and gain I/O performance when reading and writing large files [28]. One large drawback of the direct I/O is that the data is always written synchronously, i.e., the user process will not return from the read() or write() system calls until the data is transferred. This synchronous behavior will degrade the performance for small I/O request.

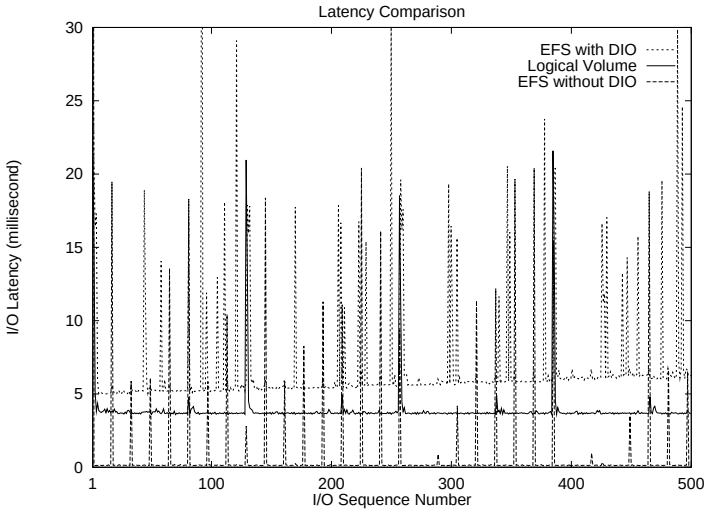


Figure 6: Latency distribution of 500 times of read operation (4 KBytes) for 3 access schemes

To understand the I/O performance of **EFS** and **lv**, we conducted a simple read performance test for **EFS** with direct I/O, **EFS** without direct I/O, and **lv** driver. In our test configuration, the logical volume contains 4 RAID 3s and its stripping granularity is 512 KBytes. The **EFS** is built on top of this logical volume. The request size ranges from 4 KBytes up to 4 MBytes. For each request size, the read operation iterates 500 times. The experiments were conducted exclusively without interference from other processes. The **EFS** operations access a 1.7 GBytes file which was created via the extended file system which has under 20% storage occupied. The

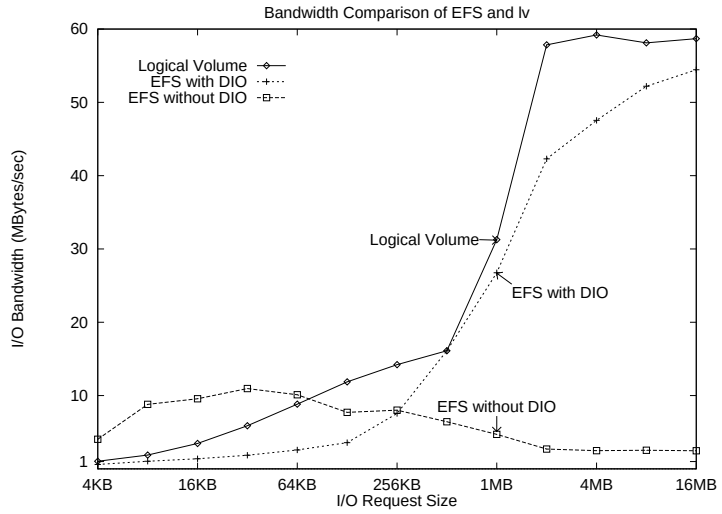


Figure 7: Bandwidth comparison

file system tries to store files as continuous as possible. After analyzing the tested file, we found 31 discontinuities and 26,750 extends has been used to store this file. When accessing **lv**, we assume the file was stored starting from the first byte of the storage. This assumption is reasonable because the logical volume device driver can access any byte in the storage by modifying the file pointer in the file descriptor structure.

The sample minimum, maximum, and mean of the experiments were collected. Figure 6 shows the latency distribution of the 500 read operations (with request size of 4 KBytes) for the three access schemes. **EFS** without direct I/O achieves the lowest latency because of read prefetch. In contrast, the **EFS** with direct I/O has the worst performance (for small request size). The timing jumps observed in Figure 6 were caused by either disk seek time or the discontinuities of the file placement. One interesting behavior is the timing of **EFS** with DIO is increased for larger sequence numbers. This is partially caused by the synchronization operations of direct I/O. Figure 7 shows bandwidth comparisons for the three access schemes. For large file retrievals, the operation **EFS** without direct I/O has the worst performance because of the additional data copying.

Obviously, Our experimental results showed that **EFS** does not perform well for large file retrievals even with direct I/O. One of the main objectives of this paper was to examine several file placement schemes over the large disk farm. However, the **EFS** hides the details of file placement. Therefore, we use either **rdsk** or **lv** device driver to access the underlying storage for the rest of our experiments. The motivation is to explore the maximum capability of the disk storage subsystem.

2.4 Video Retrieval Software

The experiments presented in this paper focus on the concurrent retrievals of video files from a mass storage system to the main memory in a large-scale video file server. The main objective is to investigate how many concurrent video streams with acceptable quality of service can be supported. For each client, the video file server assigns one *retrieval process* to retrieve the video file. The retrieval process reads video frames from the storage system periodically during the whole video retrieval session. The behavior of the retrieval processes is influenced by the buffer management scheme which regulates individual retrieval process. A detailed considerations of the retrieval process and buffer management are described in this section.

2.4.1 Retrieval Process

The retrieval process provides a video retrieval service to the client with a sustained bandwidth. The retrieval processes periodically send read request to the storage system (directly to disk arrays or through logical volumes) and wait for the completion of read requests. Each read request sent to the storage system asks for a segment of video frames. The read requests should be fulfilled by the storage system within a specified time period. The specific time by which the read request must be completed is called the *deadline*. After the video frames are retrieved, the retrieval process sleeps until the next time interval. While the retrieval process is idle, the transferred video block should be delivered to the network through the network interface. With the *retrieve-and-idle* paradigm, a retrieval process provides a video stream with sustained transfer rate to its client.

Retrieval processes only retrieve video frames from the storage system in the memory subsystem of the video file server. The video file server does not deliver the video frames through the network subsystem to the remote clients. In this paper, we primarily focus on the potential of using a mass storage system as a VOD server and evaluate its performance in different configurations. The network subsystem support for a VOD server is beyond the scope of this paper. Therefore, we made an assumption that *the block of video frames is always successfully delivered in time by the network subsystem to the client after retrieval from the mass storage system.*

In our experiments, we did not apply any scheduling mechanism to control the retrieval processes. All processes start around the same time. The processes are activated one by one at the beginning of each experiment. Each retrieval process represents a video retrieval request issued from a remote user. An activated process puts itself into an idle state while waiting for a start signal. All retrieval processes start to execute around the same time when the start signal appears. However, there are timing offset between the first running process and the last one. Since some of the processes will start before others, and some of the processes will finish the retrieval earlier than others, we only measured the performance metrics during the *valid duration*. The *valid duration* is defined as the time period between the time that the last process starts and the time that the first process stops.

To minimize the impact of virtual memory management and process scheduling of the oper-

ating system, the retrieval processes are page locked in the memory to avoid any page fault or program swapping during the experiments. This means that the number of video streams that can be supported by the system is limited by the total physical memory available in the system, since each retrieval process consumes a certain amount of memory space. The retrieval process is also set to a non-degrading, high priority mode to reduce the side effect of process scheduling.

2.4.2 Buffer Management

Buffer management is an essential issue pertaining to a video on demand server. One of the design objectives of the video on demand server is to support as many concurrent accesses as possible, i.e., activate as many retrieval processes as possible. However, the number of clients which can be supported by a server will be limited by several factors including the memory resources available to the memory subsystem. This is because each retrieval process requires a certain amount of memory as buffer space to retrieve and deliver the video frames. As the number of clients increase, the amount of available buffer space for each retrieval process will be decreased, which means the request size sent to storage system has to be decreased.

On the other hand, when the request size becomes smaller, another problem appears. As we will discuss in Section 4, when the request size below some threshold, the disk arrays can not be used efficiently. From the utilization point of view, the request size should be larger than the threshold. This means each retrieval process should have adequate buffer space. There is a tradeoff between increasing the number of clients and allocating adequate buffer space for each client. In all experiments, we allocate two buffers for each retrieval process. The two buffers were used in a pipelined style, one for retrieval from the storage system, and the other for delivering to network interface. The roles of the two buffers will be exchanged in the next time interval.

Without loss of generality, we assume that the video file contains fixed size video frames of 32 KBytes. For a video stream with 30 frames per second playback rate, , the retrieval process should provide 7.864 Mbits/sec bandwidth for the user. The video quality of this bandwidth requirement is closed to MPEG-II video stream. During the video retrieval, the retrieval process periodically sends a read request to the storage system. If the video frames were retrieved before the deadline (which depends on the number of video frames for each retrieval), the process puts itself in a sleep state to simulate the delivery operation until the next time interval. The deadline is determined by the number of video frames retrieved in each request. For example, if video frames are displayed every 33.33 milliseconds (30 frames per second). A retrieval of 16 video frames should be completed and delivered within 533.28 milliseconds ($16 \times 33.33 = 533.28$). The retrieval process will repeat the same operation when it wakes up at the next time interval.

If the video frames are not ready (part of them or all of them) to be delivered by the deadline (this situation is called *miss-deadline delay*), the retrieval process will wait until the video frames are ready, then resume the same read operation again. This means the time line must be adjusted. As shown in Figure 8, the retrieval of segment $i+1$ does not complete before the deadline. The process waits until it completes the I/O request, adjusts the time line, and then issues the read request for segment $i+2$ immediately. In our experiments, we do not drop any video frame caused by the existence of the miss-deadline delays. Therefore, more miss-

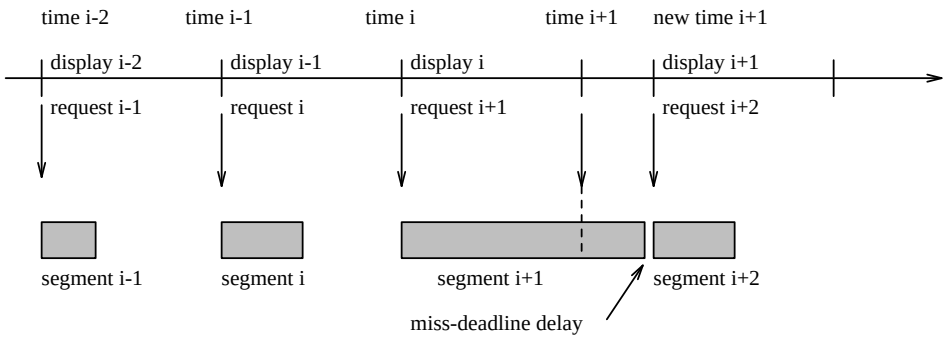


Figure 8: Retrieval and delivery with two buffers

deadline delays occurred during the retrieval session, the longer time required to playback the video files.

2.4.3 Performance Metrics

In our experiments, we use the following QoS parameters to determine the maximum number of concurrent accesses which can be supported by the server with various storage system configurations and allocation schemes.

- *Average in-time retrieval latency*: Average latency of those retrievals which were completed before the deadline.
- *Average miss-deadline retrieval latency*: Average latency of those retrievals which did not complete before the deadline.
- *Number of miss-deadline retrievals*: The total number of retrievals which missed the deadline. The deadline depends on how many video frames were retrieved in each read request.
- *Delay caused by the existence of jitters*: The retrieval processes used in our experiments adjust the time line when miss-deadline retrievals occur. The more miss-deadline delays which occur during the retrieval session, the longer the time required to playback the video files. This parameter can be referenced together with the number of miss-deadline retrievals in order to understand in detail the performance of one video retrieval session.

For each experiment in the following sections, we measured the performance metrics for each active retrieval process. The experiments were repeated with the increment of the number of active processes until the server can not provide acceptable quality of service in average (i.e., the average jitter ratio is greater than 1%). In this way, we evaluate how many concurrent accesses can be provided by the server supporting single and multiple files, data stripping schemes, and storage configurations. The number of retrieval processes which can be supported by the storage

system is determined as follows. For each retrieval process, we calculate the percentage of the miss-deadline retrievals, which is the ratio of the number of miss-deadline requests to the total number of read requests. The *average miss-deadline percentage* was calculated from all the retrieval processes after each iteration of an experiment. We repeat the experiment with the increment of the number of retrieval processes until the average miss-deadline percentage was greater than 1.00%. Thus, we obtain the maximum number of retrieval processes which can be supported by the storage system without any support of scheduling and control mechanism by the modification of software or hardware components.

3 Supporting Concurrent Accesses on Single Video File

In this section, we present the experimental results of simultaneously concurrent accesses to a single video file. The video file is stored on the storage device with different data stripping schemes, RAID 3 byte stripping in Section 3.1, logical volume stripping in Sections 3.2, and application level stripping in Section 3.3. The experimental results give us a fundamental understanding of the performance of the disk arrays and logical volumes.

3.1 RAID 3 Byte Stripping: The Effect of Request Size

The performance of the disk arrays is essential to all of our experiments. The experiments in this subsection study the capability of the disk array regarding to support concurrent accesses with acceptable quality of service. According to the experimental data from Ruwart et al. [25], the throughput of one Ciprico RF 6710 RAID 3 can achieve 17.8 MBytes/sec when the request size is 4096 KBytes. However, it is still not clear what the performance of the disk arrays is when supporting concurrent accesses, which require constant retrieval data rate.

To support concurrent accesses on a RAID 3 disk array, the most important design parameter is the requested video block size. Because the total amount of memory space is limited, it is not cost effective to allocate a very large buffer space for each client for achieving the peak throughput of the disk array. On the other hand, if the video block size is too small, the playback duration will be too short to accommodate the retrievals of all concurrent accesses.

Therefore, it is critical to find out the most suitable request size, which is equal to the buffer size in our experiments. In this experiment, we stored one video file on a single disk array and issued multiple retrieval processes to access the video file. We conduct the experiments with different request sizes. For each request size, we repeat the experiment and increase the number of retrieval processes until the quality of service can not be maintained. During the experiment, each retrieval process retrieves the video file at the rate of 30 frames per second. The session lasts around 18 minutes, which is sufficient for us to observe the performance of the storage device. Since we used fixed size video frames, 32 KBytes, the total number of retrieval operations depended on the request size. For example, it takes 32000 read operations to retrieve 18 minutes video file with request size of 32 KBytes (retrieving one video frame at a time), while only 2000 read operations are required to retrieve the same video file with request size of 512

Table 1: Number of clients which can be supported by using different request sizes.

Request size	Streams	Loops	Miss-deadline	$\frac{Miss}{Loops}$(%)	Latency (ms)		Delay
					In-time	Miss	(sec)
32 KB	**1**	32000	14	0.04%	5.35	125.98	1.30
32 KB	**2**	32000	339.50(158-521)	1.06%	13.77	64.00	7.57
64 KB	**3**	16000	35.33(32-37)	0.22%	19.63	149.45	2.91
64 KB	**4**	16000	10190.75(8000-10583)	63.69%	30.47	70.92	43.35
128 KB	**8**	8000	28.75(26-36)	0.35%	35.99	218.11	2.44
128 KB	**9**	8000	353.67(275-584)	4.42%	42.36	167.30	12.02
256 KB	**11**	4000	21.27(13-29)	0.53%	106.21	379.58	2.30
256 KB	**12**	4000	286.58(246-408)	7.16%	117.92	298.22	9.09
512 KB	**14**	2000	16.43(7-35)	0.41%	193.18	658.64	2.51
512 KB	**15**	2000	771.47(188-1349)	38.57%	224.44	611.94	72.74
1 MB	**14**	1000	3.78(2-5)	0.38%	371.57	1392.61	1.23
1 MB	**15**	1000	35.40(30-41)	3.54%	541.94	1151.85	2.99

KBytes (retrieving 16 frames for each read). The results are summarized in Table 1.

In Table 1, *Streams* is the total number of concurrent video accesses. *Loops* is the total number of read requests sent to the disk array from each retrieval process. *Miss-deadline* is the average number of read requests which did not complete before the deadline (the two numbers in the parentheses are the minimum and maximum number of miss-deadline retrievals). $\frac{Miss}{Loops}$(%) is the percentage of the average miss-deadline read requests. For timing concerns, we also calculated the average latency of in-time requests and the average latency of miss-deadline requests. The extended delay caused by the existence of miss-deadline retrievals can be referred in conjunction with the $\frac{Miss}{Loops}$(%).

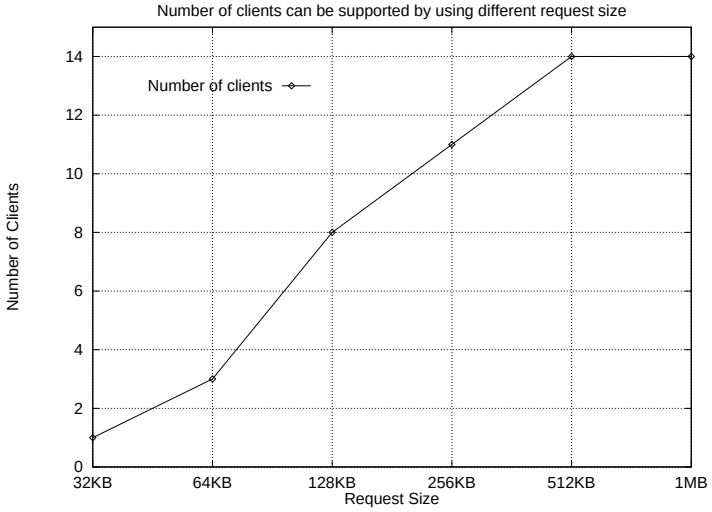


Figure 9: Number of clients which can be supported by different request sizes.

For each request size, we provide two sets of experiment results. One for multiple accesses

Table 2: Performance of pseudo disk arrays

Streams	Wide	Loops	Miss-deadline	$\frac{Miss}{Loops}$(%)	Latency (ms)		Delay
					In-time	Miss	(sec)
15	1	2000	0.00(0-0)	0.00%	108.51	0.00	0.00
16	1	2000	763.94(145-1560)	38.20%	258.27	582.68	37.35

with acceptable quality of service, the percentage of the average miss-deadline was less than 1.00%. The other one for unacceptable quality of service. As shown in Table 1, the difference between these two set of results is apparent. The results show that the performance of the storage device degrades when it is saturated. The content of Table 1 is also illustrated in Figure 9. The experiment results clearly suggest that the performance of the disk array does not scale linearly as the request size becomes larger than 512 KBytes. With a large video server as the SGI Onyx which is equipped with 512 MBytes memory or more, buffers of 512 KBytes are a reasonable size for each client. Therefore, in the remaining experiments presented in this paper, we use 512 KBytes as the buffer size and request size.

3.2 Logical Volume Stripping: The Effect of Stripping Granularity

3.2.1 Pseudo Disk Array

There are 8 real RAID 3 disk arrays, one system disk, and 23 pseudo disk arrays on the fully configured Onyx machine. The pseudo disk arrays are only equipped with a disk array controller without any attached disk drive. The pseudo disk array controller simulates the real disk array by inserting the appropriate delay for the seek and rotational latencies. The resemblance between the real and the pseudo disk arrays is critical to later experiments which employ a large number of real and pseudo disk arrays.

The performance of the pseudo disk array in terms of the supportable concurrent accesses is examined in this experiment. We use the request size of 512 KBytes as the experiments in Section 3.1. The performance metrics of a pseudo disk array are listed in Table 2. The pseudo disk array has a slightly better performance than the real disk array. In Section 4.4, we further study the performance difference between real and pseudo disk arrays when they are used in logical volumes.

3.2.2 The Effect of Stripping Granularity

The logical volume consists of multiple RAID 3 disk arrays, which provides a larger storage space and improves the I/O throughput by using data stripping across multiple physical disk arrays. The data stripping in the logical volume is accomplished by distributing the data with unit of *step size* among the disk arrays in a round-robin style. The step size specifies the amount of data that is transferred to or from one RAID 3 before transferring it to the next RAID 3.

Table 3: Number of clients which can be supported by a 2-wide logical volume using different step sizes.

Step size	Streams	Loops	Miss-deadline	$\frac{Miss}{Loops}$(%)	Latency (ms)		Delay
					In-time	Miss	(sec)
4 KB	**1**	2000	4.00(4-4)	0.20%	234.55	668.58	0.54
4 KB	**2**	2000	1991.00(1990-1992)	99.60%	514.09	624.06	180.75
16 KB	**4**	2000	6.00(5-7)	0.30%	117.09	699.93	0.97
16 KB	**5**	2000	523.80(521-526)	26.22%	303.05	715.33	95.36
64 KB	**10**	2000	12.30(10-15)	0.62%	346.93	814.79	3.41
64 KB	**11**	2000	38.55(34-43)	1.93%	252.84	695.33	6.17
256 KB	**25**	2000	11.60(7-22)	0.58%	204.79	640.13	1.30
256 KB	**26**	2000	27.65(19-37)	1.38%	216.69	643.53	3.00
512 KB	**26**	2000	9.15(4-18)	0.46%	206.56	802.14	2.12
512 KB	**27**	2000	24.07(3-55)	1.20%	193.43	604.03	0.82

The performance of the logical volume depends on the appropriate selection of the step size and the request size issued to the logical volume. We study the effect of stripping granularity in this subsection and the impact of stripping size in Section 4.4.

In this experiment, a 2-wide logical volume (consists of two disk arrays) is examined to find out the most suitable step size (stripping granularity). We used 512 KBytes as the fixed request size and tested the performance of the logical volume with different step sizes, from 4 KBytes to 512 KBytes. Table 3 shows the number of concurrent accesses which can be supported by a 2-wide logical volume using different step sizes. Figure 10 is the corresponding graph of Table 3. For step sizes are less than 256 KBytes, the performance of a 2-wide logical volume is worse than that of a single disk array, which can support up to 14 clients (from Section 3.1). The main reason is the improper selection of the step size. When the logical volume received a read request from the retrieval process, it will check its configuration and issue sub-requests to the constituted disk arrays. The number of sub-requests and their size will depend on the request size, stripping size (2 in this case), and stripping granularity. After issuing the sub-requests, the logical volume waits for the completion of all subsequent. With smaller step sizes, the logical volume needs to issue more sub-requests to the disk arrays and their size is far less than the threshold for obtaining good performance from disk arrays.

The logical volume achieves a quite good performance by using a step size of 256 KBytes. In this case, each read request received by the logical volume is converted into two separate sub-requests and each sub-request was sent to each disk array. This configuration exploits the parallelism of multiple disk arrays with the minimum number of sub-requests even though the size of sub-request is still less than the threshold. On the other hand, when step sizes are greater than the request size only one disk array was activated to serve the read request and other disk arrays remain idle. In this case, the disk arrays within the logical volume was not fully utilized.

For the remaining experiments related to logical volume, we set up the step size according to the number of the constituted disk arrays and the request size to achieve the maximum degree of parallelism.

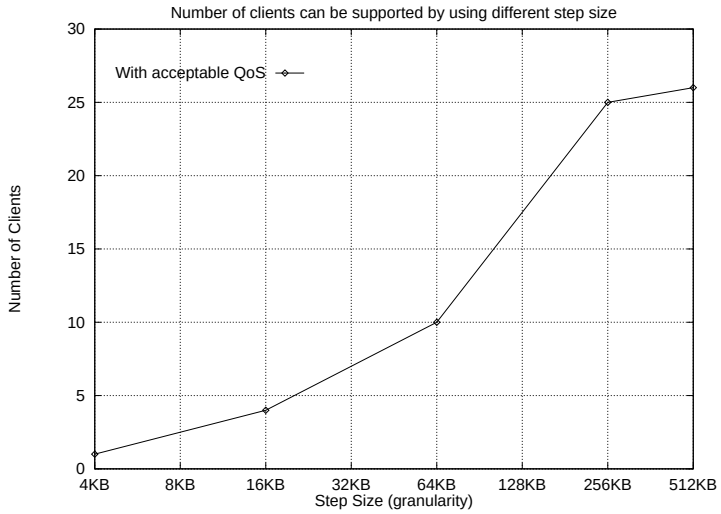


Figure 10: Number of clients which can be supported by a 2-wide logical volume using different step sizes.

3.2.3 Performance of Logical Volume Stripping

Two kinds of logical volumes are studied in the experiments, logical volumes consisting of real disk arrays and logical volumes consisting of pseudo disk arrays. A series of experiments are conducted for each stripping size, 2-wide, 4-wide, and 8-wide logical volumes are tested. A fixed request size of 512 KBytes is used in each retrieval operation. For each stripping size, we also repeat the experiment with the increment of the number of retrieval processes until the quality of service can not longer be guaranteed. The performance of logical volumes is shown in Figure 11, which also includes the results of a single disk array from the previous sections. The figure can be used to compare the difference between real and pseudo disk arrays when used in the logical volumes.

As shown in Figure 11, the performance of the real and pseudo disk arrays are quite similar on a single disk array, 2-wide, and 4-wide logical volumes. There is a large difference between them on 8-wide logical volumes, 56 supportable clients for pseudo disk arrays and 42 for real disk arrays. The main reason for the diversity is the different performance characteristics between pseudo and real disk arrays with smaller sub-request sizes. For the 8-wide logical volumes, the size of the sub-requests issued to each individual disk array is 1/8 of the read request size. The performance of the pseudo disk arrays does not degrade as dramatically as that of the real disk arrays. The experimental results suggest the avoidance of using pseudo disk arrays in logical volumes with large stripping sizes.

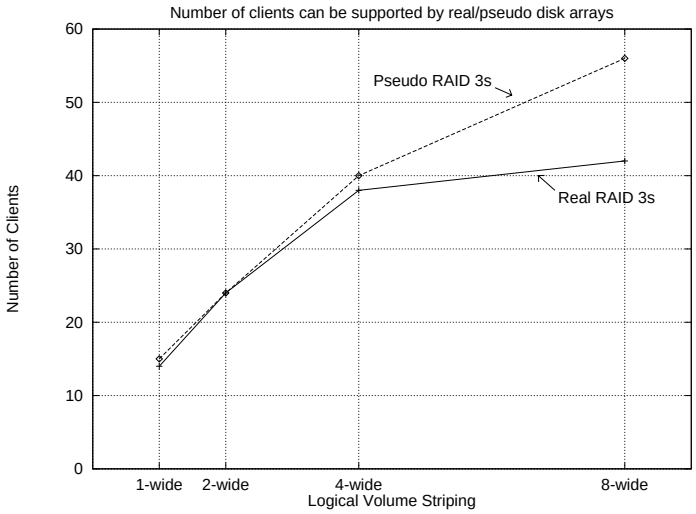


Figure 11: Number of clients which can be supported by real and pseudo disk array with different logical volume stripping.

3.3 Application Level Stripping

One of the performance bottleneck with *logical volume stripping* is that all the constituted disk arrays will be blocked until all sub-requests are completed. To distribute the load of concurrent accesses is the key to improve the performance. In this section, a novel allocation scheme called *application level stripping* is proposed, and the performance improvement has been observed. Application level stripping is the topmost layer in our data stripping hierarchy which might include two other layers, logical volume stripping and RAID 3 byte stripping. Application level stripping employs multiple disk arrays or logical volumes to store and retrieve the video files. Video files are stored across the participating storage entities in the same round-robin style as the logical volume. With application level stripping, the participated storage entities were accessed in turn by the retrieval process one by one. All retrieval processes follow the same access behavior. In this case, other storage entities can still be accessed by other retrieval processes. Therefore, the application level stripping exploit the maximum degree of concurrency and provides more concurrent accesses on a single video file than logical volume stripping.

In this section, we study the performance of the application level stripping on disk arrays or logical volumes with different stripping sizes. We evaluate multiple accesses on a single video file which is stored and accessed on RAID 3s, 2-wide, and 4-wide logical volumes with application level stripping scheme. According to the experimental results of the logical volumes consisting of pseudo disk arrays in Section 4.4, we avoid using 8-wide logical volumes in large scale experiments. During the experiment, all retrieval processes access the same video file which was stored on the storage entities using application level stripping scheme. For example, in the case of 16 1-wide disk array, the video file was stored and retrieved across the 16 disk arrays in

Table 4: Number of clients which can be supported by application level stripping.

No. of lv	Streams	Loops	Miss-deadline	$\frac{Miss}{Loops}$(%)	Latency (ms)		Delay
					In-time	Miss	(sec)
1-wide disk arrays							
8	**100**	2000	12.42(0-37)	0.63%	148.81	625.28	0.98
8	**105**	2000	27.80(6-56)	1.40%	179.74	619.69	2.16
16	**200**	2000	16.88(0-70)	0.84%	125.09	713.12	2.10
16	**210**	2000	34.36(6-120)	1.72%	162.93	648.36	3.14
31	**210**	2000	7.19(0-46)	0.36%	94.98	2179.96	3.96
31	**240**	2000	38.91(1-245)	1.95%	156.49	919.23	6.92
2-wide logical volumes							
4	**96**	2000	8.92(0-19)	0.45%	169.56	652.80	1.04
4	**100**	2000	41.23(21-71)	2.07%	164.33	643.06	4.22
8	**180**	2000	14.49(1-31)	0.72%	156.12	679.43	1.53
8	**190**	2000	96.44(58-139)	4.82%	186.77	745.71	20.16
15	**280**	2000	11.13(0-128)	0.56%	145.09	328.29	1.37
15	**300**	2000	33.64(1-586)	1.68%	145.64	1541.69	9.44
4-wide logical volumes							
2	**70**	2000	7.96(3-19)	0.40%	124.57	691.90	1.12
2	**72**	2000	287.28(227-329)	15.09%	232.74	585.55	14.85
4	**150**	2000	8.19(2-19)	0.41%	152.28	744.52	1.63
4	**160**	2000	377.81(296-515)	18.89%	215.78	674.06	52.60
7	**245**	2000	18.55(6-56)	0.93%	152.69	694.43	2.72
7	**260**	2000	77.96(2-327)	3.90%	150.25	689.46	10.07

round-robin fashion.

The experimental results were summarized in Table 4 and displayed graphically in Figure 12. The comparison is based on the total number of participating disk arrays which is used on the x-axis in Figure 12. The total number of logical volumes used in the experiments can also be calculated from the total number of disk arrays. For example, 30 disk arrays may comprise 15 2-wide logical volumes. We use two lines to present the performance and scalability of application level stripping in Figure 12. The bottom one of the two lines represents a number of concurrent accesses that can be supported by the logical volume with which we have experimented. The area below the bottom line is the *feasible range*. For any point (x, y) in the feasible range, y concurrent video accesses can be supported by x RAID 3s, or $\frac{x}{s}$ s-wide logical volumes with acceptable quality of service.

The upper line denotes the number of concurrent accesses that can not be supported by the logical volumes with which we have experimented. The area above the upper line is the *infeasible range*. The exact number of clients which can be supported is within the range embraced by these two lines. Due to the availability of equipment, it is not feasible for us to conduct a comprehensive experiment in order to find out the exact number.

Application level stripping on 2-wide logical volumes and 4-wide logical volumes shows substantial scalability. With 2-wide logical volumes, the application level stripping can support more than 280 clients accessing the same video file, which is beyond the limitation of any logical volume and disk arrays. Application level stripping on a 1-wide disk array does not exhibit similar scalability when the total number of disk arrays exceed 16.

The experiment in this subsection is further expanded upon in order to study the effect

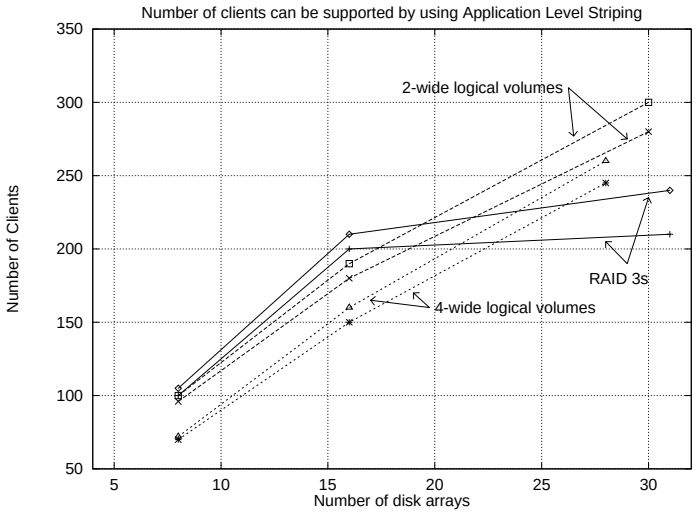


Figure 12: Number of clients can be supported by application level stripping.

Table 5: Effect of memory interleaving.

Streams	Wide	Loops	Miss-deadline	$\frac{Miss}{Loops}$(%)	Latency (ms)		Delay
					In-time	Miss	(sec)
20 processors, 768 MB 4-way interleaved memory							
280	2	2000	11.13(0-128)	0.56%	145.09	328.29	1.37
300	2	2000	33.64(1-586)	1.68%	145.64	1541.69	9.44
12 processors, 512 MB 8-way interleaved memory							
300	2	2000	11.00(0-67)	0.55%	85.14	1629.66	3.95
330	2	2000	15.74(1-69)	0.79%	84.43	1243.48	5.43
360	2	2000	9.78(1-51)	0.49%	86.49	1873.09	4.28

of memory interleaving. As mentioned in Section 2.1, the Onyx machine can support high bandwidth and low latency memory access by using interleaved memory. In this experiment, application level stripping is tested on 15 2-wide logical volumes using two kinds of server configurations. The first configuration is the same as the ones used in previous subsections, 20 processors and 768 MBytes 4-way interleaved memory. The second one uses only 12 processors and 512 MBytes 8-wide interleaved memory. We reduce the processors in order to provide more board slots to configure the 8-wide interleaved memory.

The experimental results are listed in Table 5. As shown in Table 5, the video file server does not require much computational power. In the second experiment, 12 processors are capable of supporting more than 360 concurrent accesses. The number of processes which can be activated is limited by the available physical memory. In the first experiment, even though there are more processors and larger memory space, the server can only handle more than 280 retrieval processes. It is the data movement capability which makes the difference between these two experiments.

3.4 Summary

We summarize the experiments discussed in this section as follows:

- To support concurrent accesses on a RAID 3 disk array, the key design parameter is the request size. In order to achieve adequate throughput, the request size sent to the physical storage device should be larger than the threshold. A suitable request size depends on the storage device.
- To support the concurrent accesses using *logical volume stripping*, the design of stripping granularity is critical. Proper selection of the stripping granularity can exploit the parallelism of multiple disk arrays and reduce the number of sub-requests generated by the logical volume. The reduction of the number of sub-requests also decrease the overhead caused by the interaction between the logical volumes and physical devices.
- Application level stripping is a flexible data stripping scheme which can employ multiple disk arrays or logical volumes. This scheme exploits the degree of concurrency and provides maximum number of accesses on one single video file.
- To support a large number of concurrent video accesses, the video server must have adequate system bandwidth to satisfy the data movement requirement.

4 Supporting Concurrent Accesses on Multiple Video Files

The experimental results in the previous section provide fundamental understanding about the performance of storage devices and the impact of different data stripping schemes. Based on this knowledge in the previous section, we conduct a series of large scale experiments to study the scalability of the video file server and the effect of user access patterns. To investigate the scalability of the video file server (our Onyx machine), we activate multiple groups of users to access multiple video files at the same time. The *group access* pattern is defined as a group of users which access the same video file at the same time. Up to 30 RAID 3 disk arrays are used in the following experiments. We conduct the group access experiments in Sections 5.1 and 5.2. Group access on multiple video files using application level stripping scheme is presented in Section 5.3

4.1 Logical Volume Stripping

4.1.1 Single File on Each Device

Group access occurs when multiple retrieval processes send read requests to the same video file, or the higher level server requesters (e.g. logical volume) issue sub-requests to the same

service provider which can be a software component (e.g. SCSI device driver) or a hardware component (e.g. disk array controller). Group access can appear at different levels. At the application level, multiple retrieval processes which access the same video file cause the read requests to be sent to the same logical volume. Similar group access can appear at the physical disk array level when multiple sub-requests were sent to one disk array controller from multiple SCSI device drivers.

The experiments in Section 4 present group access on a single video file. In this section, we activate multiple group of users, each group accesses their own video file which is stored on one storage device. The experiments evaluate how many concurrent video accesses can be supported by a storage device with different stripping sizes when there is only one video file stored in that device. Since the storage space of a disk array or logical volume is much larger than the space required by a single video file. A single video file only occupies a small portion of the storage space. For example, a 1.35 gigabytes video file only uses around 8.4% (1.35/16) of the storage space of a disk array. The time required to retrieve the video frames can be represented by the following equation:

$$T_{request} = T_{os\_fixed} + T_{cmd} + T_{seek} + T_{rot} + T_{xfer},$$

where T_{os_fixed} is the fixed operating system overhead, T_{cmd} is the time required by the storage subsystem to process the request, T_{seek} is the time it takes the RAID to seek from one end to the another, T_{rot} is the time to complete one rotation of the RAID, and T_{xfer} is the time it takes to transfer the requested video frames across the SCSI I/O channel. In the worst case, the T_{seek} seek time latency is the dominating term of the above equation. Note that we always place the video files contigously as much as we can. Therefore, the worst case seek time of the disk arrays or logical volumes is reduced by employing multiple disk drives or disk arrays. For example, if we use a 16 gigabytes disk array to store a 1.35 gigabytes video file (a two hour movie) sequentially from the beginning of the disk array, the worst case seek time latency will be reduced to 1/8 of that of a single disk drive. The reduction will be even larger if we use a logical volume of multiple disk arrays to store a single video file.

In the experiment, the 30 RAID 3 disk arrays are used to constitute 2-wide, 4-wide (only use 28 RAID 3s), and 8-wide (only use 24 RAID 3s) logical volumes. A video file is stored on each logical volume from the beginning of the device without any duplication. During the experiment session, a group of retrieval processes are activated and assigned to a video file. Each video file is accessed by the same number of retrieval processes. All processes are started around the same time by the start signal as mentioned before. The configurations of the experiment are illustrated in Figure 13. In Figure 13, the shaded area in each device represents the space occupied by the video files. As the stripping size increases, which means there are more physical devices in the logical volume, the shaded area in each physical device decrease. Therefore, the worst case seek time latency is also reduced.

The experimental results are summarized in Figure 14. The comparison is based on the total number of participated disk arrays which is used on the x-axis in Figure 14. In Figure 14, we use two lines to present the performance and the scalability of the video file server. Three sets of line

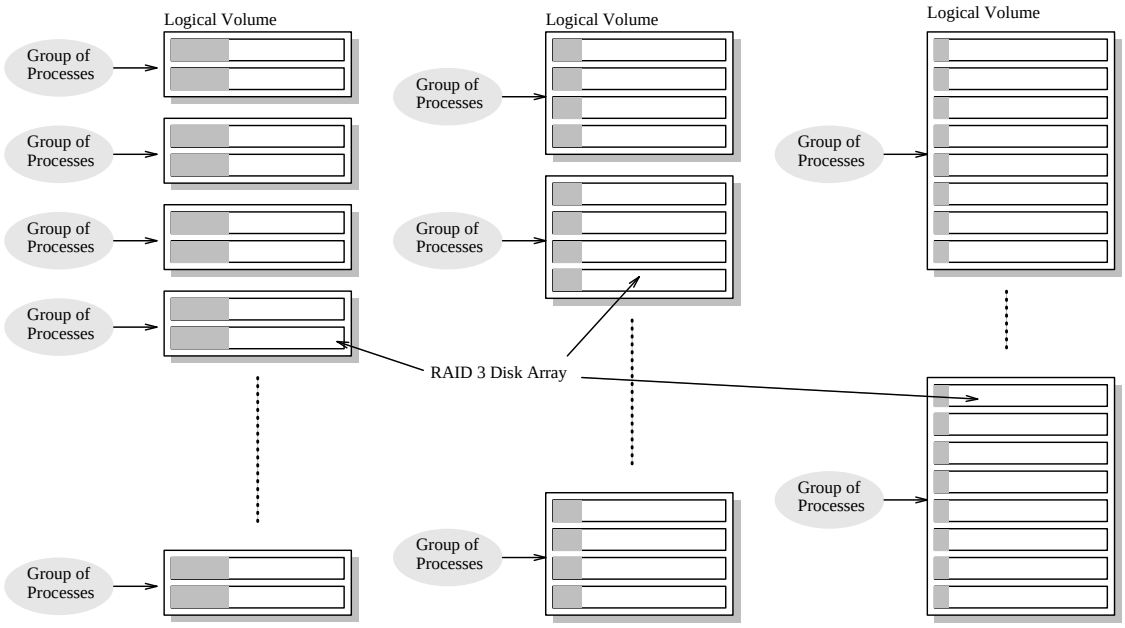


Figure 13: Group accesses on multiple video files.

pairs show the performance of the video file server when using 2-wide, 4-wide, and 8-wide logical volumes. The area below the bottom line is the *feasible range*, and the area above the upper line is the *infeasible range*. The exact number of clients which can be supported is within the range embraced by these two lines. As illustrated by Figure 14, the Onyx computer system shows a high degree of scalability by using 2-wide and 4-wide logical volumes. The system did not achieve linear scalability with 8-wide logical volumes. The main reason is the heavy interactions between logical volume device drivers and the underlying SCSI device drivers. Even though the 8-wide logical volumes provide the smallest worst-case seek time latency.

4.1.2 Multiple Files on Each Device

In the previous group access experiments, we focused on the impact of multiple service requests on a single service provider. However, it is an extreme case when each disk array or logical volume only stores one single video file. A 16 gigabytes disk array is capable of storing more than 10 full length video files. It is very likely that multiple clients will request different video files within the same disk array or logical volume. In this scenario, the worst case seek time latency would be as large as that of a single disk drive. The worst case happens when two consecutive retrievals access the same disk array or logical volume, one of them accesses the video frames on the most inner cylinder of the first video file, and the other one accesses the video frames on the most outer cylinder of the last video file.

When multiple group access on different video files within the same storage device, a certain

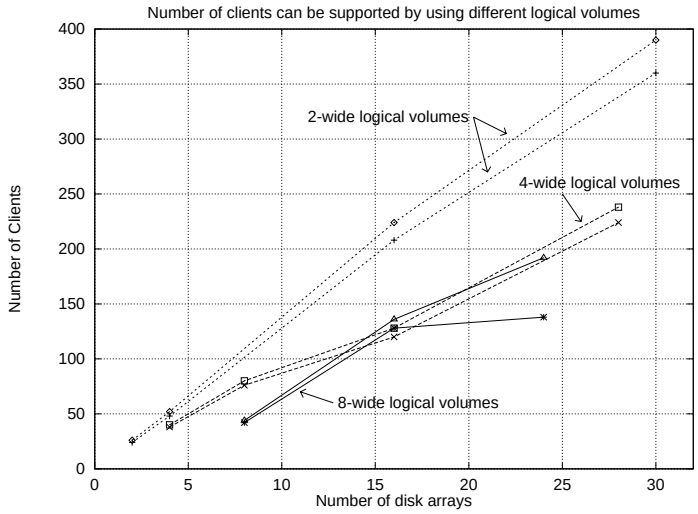


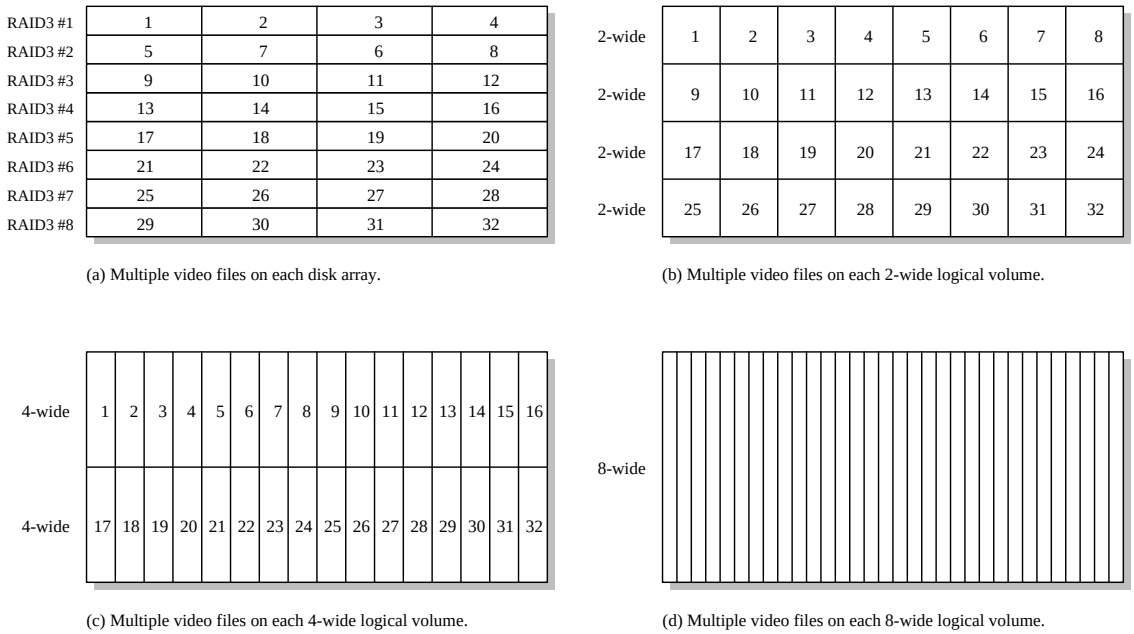
Figure 14: Number of clients which can be supported by multiple logical volumes with different stripping sizes.

amount of performance degradation is expected. Without proper access scheduling schemes enforced by the disk arrays or logical volumes (e.g. C-SCAN), the read requests will have longer worst case seek time latency. In this section, we investigate the degree of performance degradation when multiple group access on different video files within one storage device. In the experiments, multiple video files are stored in each disk array or logical volume as shown in Figure 15. We use 8 RAID 3 disk arrays as an example to show how the video files are stored. For example, Figure 15(a) shows four video files are stored uniformly on each disk array, Figure 15(c) shows each logical volume has 16 video files.

We evaluate the multiple group access on the following three storage configurations:

- 30 RAID 3 disk arrays, each disk array stores 4 video files evenly across the whole storage space.
- 15 2-wide logical volumes, each logical volume stores 8 video files evenly through the whole storage space. The step size (granularity) is set to 256 KBytes for the request size of 512 KBytes.
- 7 4-wide logical volumes, each logical volume stores 16 video files. The step size is set to 128 KBytes for the request size of 512 KBytes.

The experimental results were listed in Table 6 and the corresponding graph in Figure 16. In the experiments, the retrieval processes are assigned uniformly to access the video files. For example, when 360 clients access 30 1-wide disk arrays which can store 120 video files, each individual video file will have $360/120 = 3$ retrieval processes and each disk array will have



(a) Multiple video files on each disk array.

(b) Multiple video files on each 2-wide logical volume.

(c) Multiple video files on each 4-wide logical volume.

(d) Multiple video files on each 8-wide logical volume.

Figure 15: Video file allocation for multiple group access on each device.

$3 \times 4 = 12$ retrieval processes. After comparing Table 6 (multiple groups on each device) and Figure 14 (single group on each device), we observe a certain degree of performance degradation. For 2-wide logical volumes, the storage system can support more than 360 concurrent video accesses with single group access on each device. However, less than 330 clients can be supported by the storage system with multiple group accesses on each device. The performance degradation for this randomness is about 9%. The experimental results are visualized in Figure 16.

Table 6: Multiple access on each storage device.

Streams	**Wide**	**Loops**	**Miss-deadline**	$\frac{Miss}{Loops}$(%)	**Latency (ms)**		**Delay**
					In-time	**Miss**	**(sec)**
30 1-wide disk arrays							
330	1	2000	8.30(0-28)	0.41%	227.47	673.01	1.58
360	1	2000	893.06(767-1103)	44.65%	119.22	658.61	112.48
15 2-wide logical volumes (Step=256 KB)							
300	2	2000	9.26(0-50)	0.46%	217.89	490.67	1.91
330	2	2000	131.84(0-666)	6.59%	252.14	643.17	17.93
7 4-wide logical volumes (Step=128 KB)							
140	4	2000	1.63(0-10)	0.08%	92.27	228.03	0.40
175	4	2000	161.57(0-1215)	8.08%	160.42	171.85	11.86

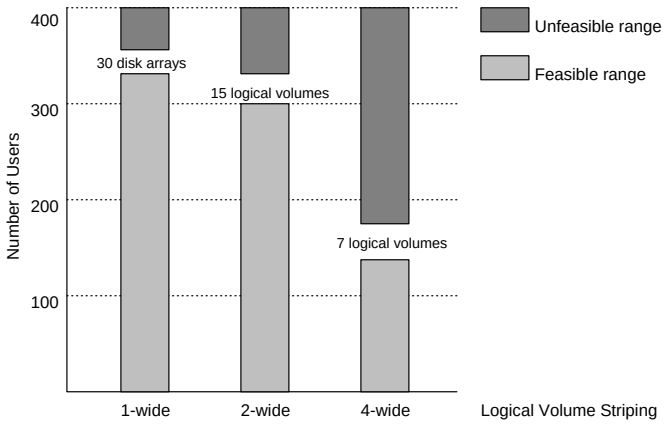


Figure 16: Multiple group accesses on devices with different logical volume strippings.

4.2 Application Level Stripping

In the last experiment, we studied the performance of multiple group accesses on multiple video files using the application level stripping scheme. We divide the whole storage system of 30 RAID 3 disk arrays into two partitions. Each partition consists of 4 real disk arrays and 11 pseudo disk arrays. We stored one video file on each partition using the application level stripping scheme. Two groups of retrieval processes are activated to access the video files. Each group accesses its own video file. With the Onyx machine's powerful I/O adapters and parallel processing, the two partitions can be treated as two independent large storage devices.

We conduct the experiment with the server configuration of 512 MBytes 4-way interleaving memory and 20 processors. Table 7 shows the experimental results. The experiment only activated up to 360 clients, 180 clients per partition. As shown in the table, the server is capable of providing acceptable quality of service with 360 retrieval processes. However, due to the limitation of the available physical memory and the memory throughput, we can not activate more processes.

4.3 Summary

We summarize the experiments in this section as follows:

- Up to 30 RAID 3 disk arrays were used for a series of large scale experiments. In each experiment, these RAID 3s are used to constitute storage devices (logical volumes) with different stripping sizes. For each storage device, we store a video file and issue a group

Table 7: Application level stripping on multiple partitions.

Partition	Streams	Wide	Loops	Miss-deadline	$\frac{Miss}{Loops}$(%)	Latency (ms)		Delay
						In-time	Miss	(sec)
Each partition has 4 real, 11 pseudo disk arrays								
1	150	1	2000	12.31(1-78)	0.62%	207.96	1101.62	4.32
2	150	1	2000	8.30(0-53)	0.41%	206.51	1052.94	3.43
1	165	1	2000	11.82(2-56)	0.59%	220.12	989.26	4.59
2	165	1	2000	13.76(2-60)	0.69%	220.54	950.16	5.12
1	180	1	2000	11.63(2-26)	0.58%	261.92	1027.06	4.34
2	180	1	2000	16.94(2-39)	0.85%	264.69	909.19	5.56

of retrieval processes to access the video file. Multiple groups of retrieval processes are activated around the same time to evaluate the scalability of the video file server.

- A similar experiment is also tested with a different configuration which stores multiple video files evenly across the entire storage space of a device. Thus, multiple groups of retrieval processes are accessing the same storage device. We study the performance degradation of the storage devices when the access pattern incurs longer seek time latencies.

- The server is potentially capable of supporting more than 360 concurrent video streams each of 7.864 Mbits/sec. This can be achieved by using 15 2-wide logical volumes or application level stripping scheme on 30 RAID 3 disk arrays.

5 Related Studies

There are many research effort [35, 20, 32, 21, 33, 13, 12, 1, 5] on the storage and retrieval of continuous media on a single disk computer systems. There are some research reports, both from industrial and academic work, on the design and analysis of multiple disk systems to support continuous media.

Reddy and Banerjee's simulation work [22] on the evaluation of multiple-disk I/O systems measured the performance on both file/transaction system and scientific application workload. The real-time constraint for multimedia data has not been addressed until recently in [23] using a variation of SCAN disk scheduling algorithm. However, multiple disks with a shared SCSI bus was studied and the results were based on simulation instead of experiments.

Lougher and Shepherd's work [18] on the design of a storage server for continuous media was designed for a multiple-disk system. The multiple disk architecture was described as a two disks accessed individually by two SCSI buses. These two disks are assumed transfer concurrently, and sequential allocation method is adopted. Therefore, this allocation scheme was similar to the *logical volume stripping* in our paper. However, no experimental results were reported in this study.

Tobagi and Pang [30, 31] designed and implemented a video server based on personal computer platform and RAID technology. The multiple disk system also using multiple disks on a shared SCSI bus architecture, and N_d consecutive video segments was distributed evenly to N_d

synchronized disks. Pre-sorting disk scheduling was used to reduce the disk I/O. However, this server can only supported tens of concurrent accesses and no experimental results were reported in this study.

The recent proposed multimedia disk scheduling mechanisms by Chen, Kandlur and Yu in [6, 36] assumed that multiple cylinders were retrieved for each continuous media stream, and sequential allocation method was adopted. They also extended their analysis on multiple synchronized disks. The buffering requirement of this scheme was cylinder-based for multi-platter disks, thus requiring large buffer size. Since all the multiple disks were assumed synchronized, this is similar to the *logical volume stripping* in our study. Again, only simulation results were reported.

Ghandeharizadeh and Ramous [14, 15] used replication strategies to support parallelism for continuous retrieval of continuous media. Their results demonstrated the superiority of the replication approach based on their simulation model. These disks are assumed independent, and sequential allocation method is adopted. Since replication introduce the inefficiency of disk utilization, we do not consider any replication strategy in this paper.

6 Conclusion and Future Works

A series of experiments are conducted to study the design issues of a mass storage system for a large-scale video-on-demand server. All experiments are tested on a SGI Onyx machine whose configuration includes 20 MIPS R4400 processors, up to 768 MBytes interleaving memory, and 31 RAID 3 disk arrays. Related issues which have been studied include data stripping schemes, video file placement, user access patterns, and the scalability of a symmetric parallel processing computer system as a video on demand server. Our experimental performance study results in the following design guidelines for a video on demand server:

- A RAID 3 disk array appears to be a good candidate for constructing the storage system of a large-scale VOD server. A RAID disk array provides large storage space, large aggregated I/O bandwidth, and fault tolerance by using multiple disk drives.
- Different data stripping schemes support different degrees of parallelism and concurrency. The RAID 3 byte stripping exploits the parallelism of multiple disk drives. Logical volume stripping provides a moderate degree of parallelism and concurrency using multiple disk arrays. Application level stripping uses storage devices in a flexible way and allows the maximum number of concurrent accesses on a single video file.
- Video file placement and user access pattern affect the performance of the video file server. The video files must be arranged properly on the storage devices in order to reduce longer worst case seek time latencies which is the dominant factor of the retrieval latency.
- A large scale video on demand server requires superior data movement capabilities. This can be achieved by using a memory system with a high transfer throughput, efficient I/O subsystem, and high speed network subsystem.

- The video file server should provide fast multiprocessing, a large amount of memory space, and enormous system bandwidth to handle requests from hundreds of users.

A symmetric multiprocessing computer system equipped with a cluster of disk arrays is potentially capable of supporting hundreds of video accesses concurrently even without any control or scheduling mechanism. The experimental results verify the potential of using a mass storage system as a VOD server and also form the basis for our further studies. This experimental study also validated the common belief on the need of new scheduling polices and buffering schemes for a large-scale VOD server. The large-scale VOD server based on the ONYX machine theoretically can provide more than one thousand concurrent accesses. Because of the limitation on the available memory space, the reported concurrent accesses only achieve about 30% of the theoretical limit. Therefore, lots of issues need to be solved to provide a large-scale server with close 100% performance. Our future work will focus on the following related issues:

- Network subsystem support: A natural extension of this study is the design of efficient mechanisms to deliver video frames from the memory subsystem through the network interface to remote clients. The network subsystem should be capable of supporting hundreds of video streams with guaranteed bandwidth and latency.
- Different server configurations: A single server configuration has been evaluated in this paper. There are other interesting configurations such as massively parallel processing machines and a cluster of servers connected by high-speed switch based networks.
- Scheduling schemes: Efficient scheduling schemes and resource reservation mechanisms are required to utilize various resources of the video on demand server. This includes scheduling access to processors, storage devices, and network interfaces.
- Admission control: To maintain guaranteed quality of service for a large number of users, an admission control mechanism must be implemented to accept or reject additional requests to access the video files based on the server's current workload.
- Buffer management: Buffering is very effective for providing guaranteed quality of service. The targets include cache space in the storage device, user buffers in the memory subsystem, and I/O buffers in the network interfaces. Group sharing of buffer spaces can potentially reduce the requirement of memory space.

Acknowledgment

Ciprico, Inc provided the disk array controllers and engineering support for the pseudo disk arrays. Silicon Graphics, Inc provided the hardware required to attach the disk arrays to the Onyx machine. The authors wish to express their sincere gratitude to James MacDonald at Army High Performance Computing Research Center, James Schnepf, Jeff Stromberg, and Rose Tsang at the University of Minnesota for their valuable comments and support.

References

[1] D.P. Anderson, Y. Osawa, and R. Govindan. A File Ssytem for Continuous Media. *ACM Transactions on Computer Systems*, November 1992.

[2] ANSI X3T9.3. *Fiber Channel - Physical and Signaling Interface (FC-PH)*, 4.2 edition, November 1993.

[3] ANSI X3T9.3. *High Performance Parallel Interface: Mechanical, Electrical, and Signalling Protocol Specification*, 8.2 edition, March 1993.

[4] ATM Forum. *ATM User-Network Interface Specification*, 3.0 edition, September 1993.

[5] H. Chen and T. Little. Physical storage organization for time-dependent multimedia da ta. *To appear at the 4th intl. Conf. on Foundations of Data Organization And Algorithms (FODO'93)*, Oct 1993.

[6] M. Chen, D. Kandlur, and P. Yu. Optimization of the grouped sweeping scheduling (gss) with heterogeneous multimedia streams. *Proceedings of ACM Multimedia'93 Conference*, pages 235–242, Aug 1993.

[7] P.M. Chen and D.A. Paterson. Maximizing Performance in a Striped Disk Array. In *Proc. 1990 Int. Symp. on Computer Architecture*, pages 322–331, 1990.

[8] A.L. Chervenak and R.H. Katz. Performance of a Disk Array Prototype. In *Proc. SIGMETRICS*, pages 188–197, May 1991.

[9] Ciprico Inc. *RF6700 Controller Board Reference Manual*, August Number 21020236 A, 1993.

[10] J.K. Dey, C.S. Shih, and M Kumar. Storage Subsystem in a Large Multimdeia Server for High-Speed Network Environments. In *IS&T/SPIE Symposium on Electronic Imaging Science and Technology*, February 1994.

[11] G.R. Ganger, B.L. Worthington, R.Y. Hou, and Y.N. Patt. Disk Arrays: High-Performance, High-Reliability Storage Subsystems. *IEEE Computer*, March 1994.

[12] J. Gemmell. Multimedia netwok servers: Multi-channel delay sensitive data retrieval. *Proceedings of the ACM Multimedia'93 Conference*, pages 243–250, Aug 1993.

[13] J. Gemmell and S. Christodoulakis. Principles of delay-sensitive multimedia data storage and retr ieval. *ACM Transaction of Information Systems*, 10(1):51–90, Jan 1992.

[14] D. Ghandeharizadeh and L. Ramos. Continuous retrival of multimedia data using parallelism. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):658–669, Aug 1993.

[15] D. Ghandeharizadeh and L. Ramos. An evaluation od three virtual replication strategies for continuous retrieval of multimedia data. *Proceedings of the Intl. Symposium on Next Generation Database Systems and Their Applications*, pages 188–197, Sep 1993.

[16] G. Gibson. *Redundant Disk Arrays: Reliable, Parallel Secondary Storage*. MIT Press, 1992.

[17] E.K. Lee and R.H. Katz. Performance Consequences of Parity Placement in Disk Arrays. *IEEE Transactions on Computers*, pages 651–664, June 1993.

[18] P.K. Lougher and D. Shepherd. The Design of a Storage Server for Continuous Media. *Computer Journal*, 36(1), Feb. 1993.

[19] D.A. Paterson, P.M. Chen, G. Gibson, and R.H. Katz. Introduction to Redundant Arrays of Inexpensive Disks (RAID). In *Proc. IEEE Compcon Spring*, 1989.

[20] P.V. Rangan and H.M. Vin. Designing File Systems for Digital Video and Audio. *Operating Systems Review*, October 1991.

[21] P.V. Rangan and H.M. Vin. Efficient Storage Techniques for Digital Continuous Multimedia. *IEEE Transactions on Knowledge and Data Engineering*, August 1993.

[22] A.L.N. Reddy and P. Banerjee. An Evaluation of Multiple-Disk I/O Systems. *IEEE Transations on Computers*, December 1989.

[23] N. Reddy. Disk scheduling in a multimedia i/o system. *Proceedings of the ACM Multimedia'93 Conference*, pages 225–233, Aug 1993.

[24] T.M. Ruwart. M.A.X.: The Maximum Achievable Xfer Experiment. Technical report, Army High Performance Computing Research Center, University of Minnesota, May in preparation, 1994.

[25] T.M. Ruwart and M.T. O'Keefe. Performance Characteristics of a 100MB/second Disk Array. Preprint 93-123, Army High Performance Computing Research Center, University of Minnesota, December 1993.

[26] Seagate Technology Inc. *ST12400 Family: ST12400N/ND, ST11700N/ND*, Publication Number 77767451, Rev. A, 1993.

[27] Silicon Graphics Inc. *Symmetric Multiprocessing Systems*, 1993.

[28] Silicon Graphics Inc. *Pipeline*, January 1994.

[29] D. Stodolsky, G. Gibson, and M. Holland. Parity Logging: Overcoming the Small Write Problem in Redundant Disk Arrays. In *Proc. 20th Annual Symposiumon Computer Architecture*, pages 64–75, 1993.

[30] F. Tobagi and J. Pang. Starworks - a video application server. *Proceedings of IEEE COMCON'93*, pages 4–11, Feb 1993.

[31] F. Tobagi, J. Pang, R. Baird, and M. Gang. Streaming raid(tm) - a disk array management system for video files. *Proceedings of the ACM Multimedia'93 Conference*, pages 393–400, Aug 1993.

[32] H. Vin and V. Rangan. Admission control algorithm for multimedia on-demand servers. *Proceedings of the Third International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 56–69, Nov 1992.

[33] H. Vin and V. Rangan. Designing a multiuser hdtv storage server. *IEEE Journal on Selected Areas in Communications*, 11(1):153–164, Jan 1993.

[34] P.R. Woodward. Interactive Scientific Visualization of Fluid Flow. *IEEE Computer*, pages 13–26, October 1993.

[35] C. Yu, S. Wei, D. Bitton, Q. Yang, R. Bruno, and J. Tullis. Efficient placement of audio data on optical disks for real-time applications. *Communications of the ACM*, pages 862–871, Jul 1989.

[36] P. Yu, M. Chen, and D. Kandlur. Design and analysis of a grouped sweeping scheme for multiimed ia storage management. *Proceedings of the Third International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 44–56, Nov 1992.