Design of a Large Scale Multimedia Storage Server *

Milind M. Buddhikot
milind@dworkin.wustl.edu

Guru M. Parulkar
guru@flora.wustl.edu

Jerome R. Cox, Jr.
jrc@hobbs.wustl.edu

Abstract

Large scale multimedia storage servers will be an integral part of the emerging distributed multimedia computing infrastructure. However, given the modest rate of improvements in storage transfer rates, designing servers that meet the demands of multimedia applications is a challenging task that needs significant architectural innovation.

Our research project, called Massively-parallel And Real-time Storage *(*MARS*) architecture, is aimed at the design and prototype implementation of a large scale multimedia storage server. It uses some of the well-known techniques in parallel I/O, such as data striping and Redundant Arrays of Inexpensive Disks (*RAID*) and an innovative* ATM *based interconnect inside the server to achieve a scalable architecture that transparently connects storage devices to an* ATM*-based broadband network. The* ATM *interconnect within the server uses a custom* ASIC *called* ATM Port Interconnect Controller *(*APIC*) currently being developed as a part of an* ARPA *sponsored gigabit local* ATM *testbed. Our architecture relies on innovative data striping and real-time scheduling to allow a large number of guaranteed concurrent accesses, and uses separation of metadata from real data to achieve a direct flow of the media streams between the storage devices and the network. This paper presents our system architecture; one that is scalable in terms of the number of supported users and the throughput.*

1 Introduction

Rapid advances in optical communication, high speed packet switching, data compression, and processor and memory design will make a distributed multimedia computing infrastructure feasible in the near future. Multimedia mail, orchestrated presentations, high quality on-demand audio and video, collaborative multimedia document editing, distributed music rehearsals, browsing remote multimedia archives, and virtual reality environments are a few examples of the exciting new applications that will be available to the users of such an infrastructure. An application in this environment will typically use one or more media streams, such as audio, graphics, video, images and text. Due to the data intensive nature of the component media streams, such applications will demand a large amount of compute, storage and network bandwidth.

*This work was supported in part by the ARPA, National Science Foundation, and an industrial consortium of Ascom Timeplex, Bellcore, BNR, Goldstar, NEC, NTT, SynOptics, and Tektronix.

The client equipment that runs these applications may range from a simple display device such as a TV, to a sophisticated multimedia workstation. A typical application may periodically request data from a remote high bandwidth media storage/archival server. Before the data is sent to the application, it may be sent to a remote compute server for processing. This server might, for example, conduct a content based search on a video stream or create a graphic volume visualization. Subsequently, the application may consume (display) the processed data, redistribute it, and/or write it back to the storage server. Thus, client equipment, high speed data transport networks, storage servers, and compute servers will be essential for future distributed multimedia applications.

It is, however, important to recognize that the requirements of future multimedia applications differ from those of existing applications. The data intensive nature of media streams requires large amount of storage, whereas the bandwidth intensive nature demands high network and storage throughput. A typical application, such as a multiparticipant collaboration, may involve multiple receivers (end-points) and hence will require multipoint communication. Also, due to the periodic, soft real-time nature of multimedia streams, delivering guaranteed end-to-end performance to such applications requires that the network as well as the end-points (the client terminal and the servers) be able to provide Quality of Service (QOS) guarantees.

The problems of designing large capacity, scalable, multicast ATM networks and providing guaranteed bandwidth with desired QOS in such networks have been researched widely and the outcomes of this research are being commercialized steadily. Unfortunately, the clients and servers in their present form suffer from serious network and storage I/O bottlenecks and therefore will not be able to meet the demands of future applications.

In particular, the network I/O bottleneck can be defined as the phenomenon that restricts the application level throughput for network oriented tasks to a small fraction (20 to 30 megabits/sec (Mbps)) of the gigabits/sec (Gbps) network bandwidth. It results from a lack of integration of operating systems, network protocol processing, host architecture and network interface. However, several recent ideas, such as zero-copy or minimal copy host-network interfaces, efficient protocol processing, and real-time support within the operating system, proposed and implemented by various research groups suggest that the network I/O bottleneck can be rectified at the hosts.

The storage I/O bottleneck results primarily from the fact that the rate (the requests per second and the throughput in Mbps) at which storage devices can handle read/write operations is two orders of magnitude smaller than the maximum rate at which an application can issue requests. This performance gap will continue to worsen as the processor, memory and network speeds keep improving faster than that of on-line secondary storage, which is limited largely by the transfer rate of direct access magnetic disks. In spite of the promise of all-optical technologies such as holographic storage, inexpensive availability of such storage in the near future is unlikely. On the other hand, the rate of improvement in the storage density of magnetic disks [29, 13] indicates that magnetic storage will be cost-effective in meeting the storage capacity requirements of future multimedia applications. However, as reported in [6], the rate of improvements in seek time (8% a year), rotational latency (practically no improvements) and transfer speeds (22% a year) will be very modest. This suggests that a large capacity disk storage used naively will not deliver the throughput required by future applications . To make matters worse, the traditional operating systems (such as UNIX) used in existing servers suffer from several drawbacks, such as random placement of data, mixing of the meta-data and data on the storage devices, excessive copying of data between the storage devices and the network interface, and the lack of real-time scheduling support. This mismatch between the requirements of multimedia and the traditional operating system support for secondary storage devices aggravates the storage I/O bottleneck even further. Given these observations, the high performance storage I/O will be critical to successful deployment of multimedia applications. The primary requirements of future network based large scale storage servers are summarized in the following paragraphs.

1. **Large number of clients:** The server may have to support thousands of concurrent clients, each with a number of active media streams and each independently accessing the same or different data.

2. **Large storage capacity:** Given the storage intensive nature of multimedia data, the collective storage requirements for thousands of multimedia documents may exceed tens of terabytes. For example, a movie server with two hundread HDTV quality 20 Mbps [30, 32] 2-hour-long movies will require roughly 3.6 terabytes (TBs) of storage. Similarly, a multimedia storage server that stores a large number of multimedia documents - each composed of multiple media streams, will require a comparable storage capacity.

3. **Large network and storage system bandwidth:** The data intensive and periodic nature of multimedia streams demands large amounts of network and storage system throughput. For example, a movie server that supports one thousand users with HDTV quality movies, will require network and storage bandwidth in excess of 20 Gbps. Storage throughput of this level is two orders of magnitude more than that observed today with the state-of-the art storage technologies such as Redundant Arrays of Inexpensive Disks (RAIDs).

4. **Real-time service:** The periodic nature of the multimedia data necessitates QOS guarantees in the form of guaranteed throughput and bounded latency for all active streams.

5. **Cost-effective:** The architecture of the server must be cost effective to be economically viable. In order to make the future multimedia servers affordable, the total cost per byte of storage must be comparable to that for the present day network based servers.

In addition to high network and storage throughput, a multimedia server may have to provide a lot of compute power to support near real-time media processing. For example, a movie server may have to convert (transcode) a native MPEG compressed movie to H.261 format that some of the customers may use. Such transcoding techniques are normally very compute and memory bandwidth intensive and unless specialized computing power [14] is available their real-time software implementations are difficult.

Thus, the aforementioned requirements of future storage servers are radically different, and therefore designing such servers is a challenging task that requires significant architectural innovation. To this end, this paper reports our project - the *Massively-parallel And Real-time Storage* (MARS) architecture. Our work uses some of the well known techniques in parallel I/O such as data striping and RAID, and an innovative ATM based interconnect inside the sever to achieve a scalable architecture that transparently connects storage devices to an ATM based broadband network. The ATM interconnect within the server uses a custom ASIC called *ATM Port Interconnect Controller* (APIC) currently being developed. Note that we are interested in supporting an environment that allows media retrieval as well as edits. However, the solutions presented in this paper are aimed at a retrieval environment. We believe that the issue of media edits is tied to the data layout schemes and is orthogonal to the hardware architecture. Therefore, we plan to extend the data layout and the associated scheduling schemes to support a read-write environment.

This paper is organized as follows: Section II discusses the background and the motivation for our work. Section III first describes the basic concept of the Massively-parallel And Real-time Storage (MARS) server and then presents a prototype implementation architecture and associated data layout and scheduling schemes for such a server. It also gives a brief outline of a scalable extension of the prototype implementation for large scale servers. The conclusions from the present status of our work and expected results of the ongoing work are highlighted in Section IV.

2 Background

Given that the storage densities of present magnetic disks are about four orders of magnitude smaller than the theoretically possible maximum, with no insurmountable obstacles in sight, inexpensive magnetic storage devices with capacity in excess of 100 gigabytes (GBs) will become available in the foreseeable future [29, 13]. A large number of such disks can be used to easily meet the terabytes of storage requirement of multimedia servers. However, satisfying the large storage and network throughput requirement will not be as easy.

In existing network based servers, due to lack of integration of the network interface with the operating system's network and disk services, there is no direct path for the flow of data between the disks and the network. This fact coupled with long seek and rotational latencies has restricted throughput of network oriented tasks that use storage facilities to a small fraction of the disk transfer speeds.

A well known way to increase the effective storage throughput is to operate a large number of disks in parallel (an organization commonly called a disk array) and physically distribute (stripe) the data over the disks [24]. However, very large disk arrays suffer from poor reliability. This fact together with the observation that small inexpensive disks outperform expensive high-performance disks in price vs. performance, led Patterson et al. to introduce the concept of Redundant Array of Inexpensive Disks (RAID) [20]. A RAID is essentially an array of small disks with simple parity based error detection and correction capabilities that guarantee continuous operation in the event of a single disk failure in a group of disks. The RAID was expected to perform well for two diverse types of workloads. One type, representative of supercomputer applications such as large simulations, requires infrequent transfers of very large data sets. The other type, commonly used to characterize distributed computing and transaction processing applications, requires very frequent but small data accesses [20]. However, measurements on the first RAID prototype at the University of California, Berkeley revealed poor performance and less than expected linear speedup for large data transfers [7]. The excessive memory copying overhead due to interaction of caching and DMA transfers, and restricted I/O interconnect (VME bus) bandwidth were cited to be the primary reasons of poor performance. Also, it is recognized now that large RAID disk arrays do not scale very well in terms of throughput.

The recent work on RAID-II at the University of California, Berkeley has attempted to use the lessons learned from the RAID prototype implementation to develop high bandwidth storage servers by interconnecting several disk arrays through a high speed *HIPPI network backplane* [15]. The RAID-II operates in two modes: a *standard mode* to support low latency request through a FDDI connection and a *high bandwidth* mode to support large transfers. It's architecture is based on a custom board design called *Xbus Card* that acts as a multiple array controller and interfaces to HIPPI as well as to FDDI networks. Though the measurements on RAID-II have demonstrated good I/O performance for large transfers, the overall solution employs FDDI, Ethernet and HIPPI interconnects and is ad-hoc. Also, it has not been demonstrated to be suitable for real-time multimedia, where the application needs are different than the needs of supercomputer applications. In short, this solution does not satisfy requirements 1, 4, and 5 mentioned before.

Several commercial implementations of RAID [5, 18] have been reported in the literature. Though very high throughputs have been demonstrated under certain access patterns, these commercial high performance implementations are expensive custom solutions for mainframes and supercomputers. Recently, some vendors have announced affordable small capacity RAID products, but most of them use proprietary buses and custom hardware to achieve high performance [1]. In short, none of these RAID implementations cost-effectively satisfy requirements 1, 2, 3, and 5 discussed earlier.

Several recent proposals for filesystems and servers have attempted to address the special requirements of multimedia streams. In particular, Rangan et al. study data layout techniques for multimedia [21, 30]. Daigle et al. [8], Yu et al. [31], Reddy et al. [22] report disk scheduling techniques to support real-time retrieval of multimedia streams. The serious drawback of these proposals is that they use a simple single disk storage model, which assumes that the multimedia data is stored and retrieved with QOS guarantees from a single disk and not from multiple disks or storage devices. This model is clearly unrealistic as the limited throughput of a single disk cannot support hundreds of concurrent clients accessing the same document stored on it. Lougher et al. [17] and Tobagi et al. [25] report a small scale multimedia server based on disk arrays, but they do not attempt to satisfy requirements 1, 2, and 3 mentioned earlier.

Little et al. [16], Miller et al. [19], and Rowe et al. [23] explore issues in the design of video browsing interfaces, authoring systems, metadata indices, and metadata databases. However, none of them address the more pressing problems of supporting large storage throughput and a large number of clients.

To summarize, all the proposals mentioned above address the problems of scheduling, data layout, and metadata design in an isolated fashion. We believe that we are the first to propose a system architecture that is scalable in terms of throughput and number of clients and take an integrated approach that collectively satisfies all the five requirements listed above.

3 Massively-parallel And Real-time Storage Architecture

This section proposes the new idea of *Massively-parallel And Real-time Storage* (MARS) and discusses a prototype implementation architecture.

3.1 Basic Idea

The MARS architecture, shown in Figure 1, consists of a set of independent storage nodes that are connected together by a fast packet based interconnect. The terms "massively-parallel" and "real-time" signify that the system allows real-time retrieval of data streams from a large storage system in a parallel fashion. The interconnect can be a packet switched bus, a ring or even a general purpose multicast switch. The MARS server interfaces directly to a high speed network, such as an ATM network. In our studies, we assume that the interconnect is based on the ATM technology and the server interfaces to one or more ports of an ATM network switch.

Unlike quite a few of the existing network file systems, MARS is a stateful server. It stores the state information for every active stream. It follows a connection oriented approach with resource reservations to provide QOS guarantees required by the media streams. Before a client can access a multimedia stream, the MARS server checks if it can reserve resources required to guarantee QOS

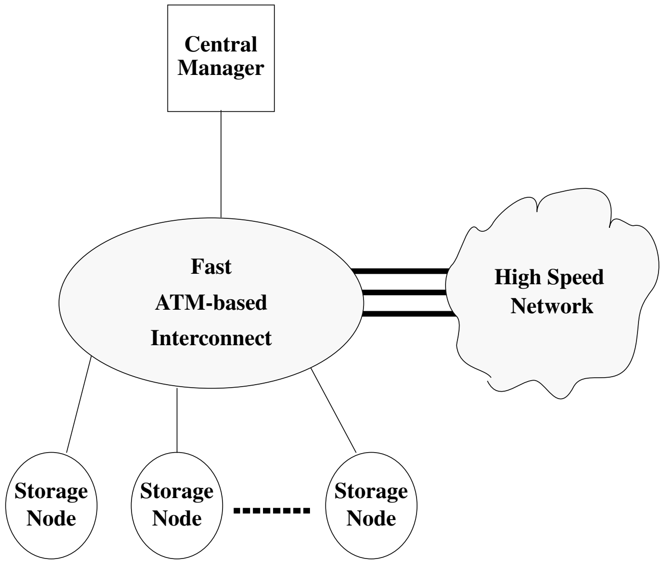

Figure 1: Massively-parallel and Real-time Storage (MARS) system

for the new connection without affecting any of the existing active connections, and accordingly accepts or rejects the connection. This suggests that a MARS server requires an admission control policy. Typically, the central manager receives the connection requests from the remote clients. It uses an admission control procedure, that runs entirely locally or in consultation with the node level admission procedure, to admit or reject the new request. Once admitted, the MARS server reserves resources such as the network bandwidth, storage bandwidth, compute support, and I/O scheduling at each node and provides statistical or deterministic guarantees for the stream retrieval (and/or modification).

Each storage node provides a large amount of storage in one or more forms, such as large high-performance magnetic disks, large disk arrays, or a high capacity fast optical storage. The storage system collectively formed by all storage nodes may be homogeneous or hierarchical. If all storage nodes provide identical storage, such as disk arrays, the system will be homogeneous. On the other hand, a hierarchical storage can be constructed by assigning storage of different types to subsets of storage nodes. A client can be served the required data, either directly from the particular storage node (level) or by a staging mechanism, in which data is first moved to a node with faster storage (higher level) and then served from that device. For example, the nodes that use optical storage and robotically controlled tapes can be considered as off-line or near-line tertiary storage. When a client attempts to access the data stored on these devices, it is first cached on the magnetic disks at the other nodes and then served at full stream rate. Thus, the collective storage in the system can exceed a few tens of terabytes and still, allow a large number of concurrent clients to access documents at the standard stream rate.

In order to ensure a large throughput from the entire system, the MARS server physically distributes the data for a stream among a subset of the storage nodes. The metadata information associated with the data is also distributed among various storage nodes and the central manager. Such data and metadata distribution (striping) depends on various factors such as, length of the document, demand in terms of the number of concurrent clients, degree of interactive behavior, and design of the data layout scheme. Typically, the storage manager decides this distribution at the time the document is stored on the server.

In addition to providing a storage facility, each storage node supports one or more of the resource management functions outlined below.

- **File system support:** Each storage node performs typical file system functions such as data and metadata management, metadata and data cache management, and data buffer management. In addition, it may support advanced database functions, such as the ones proposed by several research groups [16, 23, 19], for efficient browsing and content based searching of multimedia information.

- **Admission Control:** Each storage node keeps track of usage of local resources such as storage and interconnect bandwidth and performs local admission control. The resource reservation and admission algorithms used to accomplish this will be similar to the ones proposed in the context of the ATM networks and the end systems [9, 11, 28].

- **Scheduling support:** Each node, completely manages real-time scheduling of local data read and write functions for each active stream. The scheduling schemes employed by the node are a topic of our on-going research [3] and one such example scheduling scheme is described in subsequent sections of this paper.

- **Compute support:** A large amount of computing power can be provided by using sophisticated media processors such as the MVP [14] at each storage node. Such processors will be embedded on the path between the storage and the network, and can be used to perform media processing functions such as transcoding, speech recognition, image processing, and character recognition required by future multimedia applications.

The range of functionality at the storage nodes decides the complexity of tasks performed by the central manager. For example, the admission control and scheduling can be performed either entirely by the central manager or in a distributed fashion if the storage nodes perform local tasks and cooperate with the central manager. Thus, depending on how sophisticated the storage nodes are and the richness of the ATM interconnect, requirements of large scale servers mentioned earlier can be met to a variable extent. The rest of the discussion in this section further elaborates this architecture and the associated tradeoffs.

3.2 A Prototype Implementation

Figure 2 shows a prototype architecture of a MARS server. It consists of two basic building blocks: the APIC based daisy chained interconnect and the storage node.

The central manager shown in Figure 2 is the resource manager responsible for managing the storage nodes and the APICs in the ATM interconnect. For every document, it decides how to distribute the data over the storage nodes and manages the associated metadata information. It receives the connection requests from the remote clients and based on the availability of resources and QOS required, admits or rejects the requests. For every active connection, it also schedules the data read/write from the storage nodes by exchanging appropriate control information with the storage nodes. Note that the central manager only sets up the data flow to and from the storage devices and/or the network and does not participate in actual data movement. This ensures a high bandwidth path between the storage and the network.

3.2.1 APIC Based Interconnect

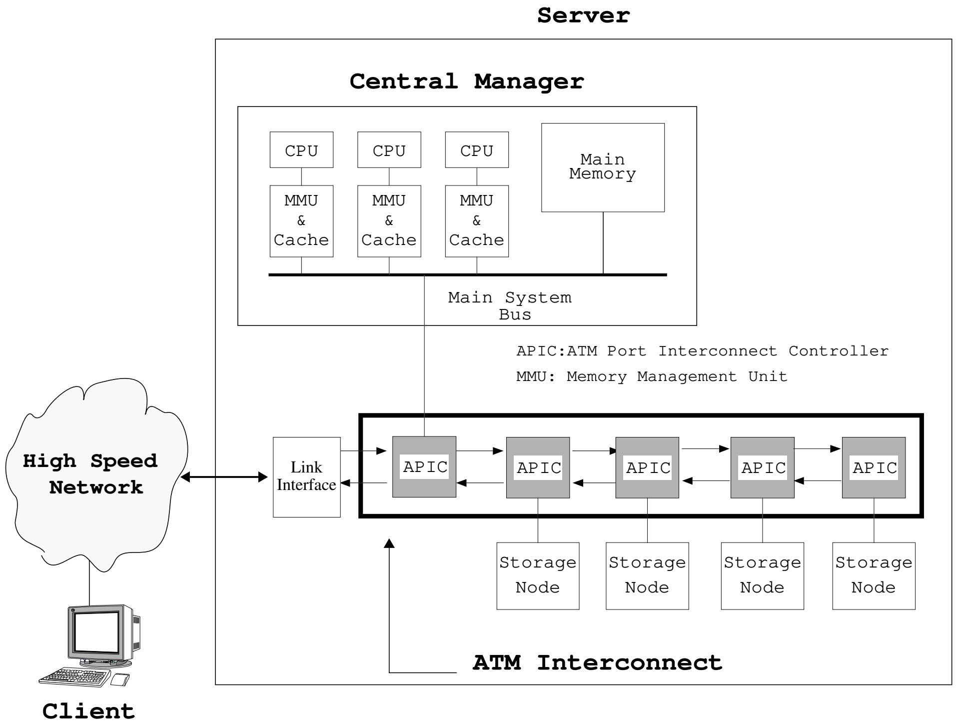

Figure 2: A prototype implementation of a MARS server

Our packet switched interconnect is made out of an ASIC called APIC (ATM Port Interconnect Controller), that is currently being developed as a part of an ARPA sponsored gigabit local area ATM testbed. The APIC chip is the basic building block for a high bandwidth *networked-I/O subsystem*, that provides a direct interface to the network for the host (workstations as well as servers) and a variety of I/O devices.

In its simplest form, the APIC behaves like a 3×3 switch, two of whose ports can be treated as ATM ports and the remaining one as a non-ATM port. Using one of the ATM ports and a line interface, APIC can be directly interfaced to the input port of an ATM switch. As shown in Figure 2, using the ATM ports, multiple APIC chips can be connected in a bidirectional daisy chain. Since each port is designed to operate at full 622 Mbps (SONET) rate, the aggregate data rate on the interconnect is ≈ 1.2 Gbps. The non-ATM port of an APIC can be looked upon as a read/write port to a data source, such as a memory or an I/O device. The APIC performs incremental AAL5 segmentation and reassembly of ATM cells through this port.

Each APIC manages a set of connections by maintaining certain state information for each active connection in a Virtual Circuit Translation Table (VCXT). The APIC also performs a simple single parameter (such as average bandwidth) based rate control or source pacing for each connection, which is important in scheduling multimedia streams from storage node to the network. The discussion of APIC internals is beyond the scope of this paper, but can be found in [10].

3.2.2 Data Layout and Scheduling

Before we discuss the details of the storage node, we will briefly describe the data layout and the scheduling schemes. It must be noted that the architecture, data layout and scheduling interact very strongly and detailed discussion of the same can be found in [2]. Here, we outline only one of the many possible data layout schemes that satisfy real-time playout requirements and allow a large number of concurrent accesses to the same or different data in a retrieval environment.

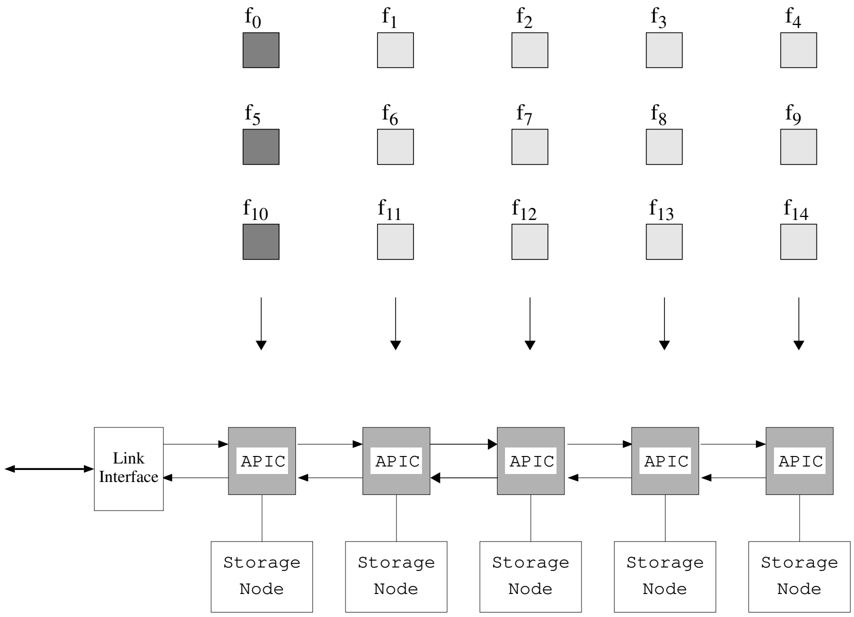

Figure 3: Layout example

The periodic nature of multimedia data is well suited to spatial distribution or striping. For example, a video stream can be looked upon as a succession of logical units repeating periodically at a fixed rate. A logical unit for video can be a single frame or a collection of frames, and the period of repetition can be the frame period, say 33 msecs/frame, or an integral multiple thereof. Each such logical unit or the parts of it can be physically distributed on different storage devices and accessed in parallel. In our architecture, we use this property of multimedia data to stripe it in a hierarchical fashion.

Figure 3 illustrates an example system with five storage nodes numbered 0 to 4 from left to right. The frames f_0, f_1, f_2, f_3, f_4 are assigned to nodes N_0, N_1, N_2, N_3, N_4 respectively. Frame f_5 is again assigned to node N_0, thus following a frame layout topology that looks like a ring. Given that there are N storage nodes, the ring topology of data distribution has a property that at any given node, the time separation between the successive frames of the stream is $N \times$ *Interframe time*. This facilitates prefetching of data to mask the high rotational and seek latencies of the magnetic disk storage.

If the storage device used at the node is a RAID, the blocks of a frame assigned to the node are further striped on the disks in the RAID. If the storage device is just a set of high capacity disks, the entire frame is stored on a single disk. In both cases the actual data layout on the surface of the disk may follow a constrained allocation policy, similar to the one discussed in [21], to ensure bounded seek and rotational latencies in retrieving consecutive blocks.

In addition to the data layout scheme, appropriate scheduling policies must be used at all levels in the path of the data retrieval and transmission. In particular, data reads and writes from multiple storage nodes must be synchronized for each active connection. Also, at each storage node, the read operations for all connections must be scheduled over multiple disks in the array. Last but not the least, at each individual disk, the head movement must be scheduled to ensure predictable latencies for the read and write accesses for all requests.

In order to satisfy overall delay guarantees, the APIC at each node has to ensure that the data retrieved from the devices is transmitted to the network as per the rate specification of the connection. This requires that the APIC perform rate control or pacing on a per connection basis.

The detailed discussion of our multilevel scheduling schemes is beyond the scope of this paper. However, we present one simple scheme for scheduling data retrieval from storage nodes.

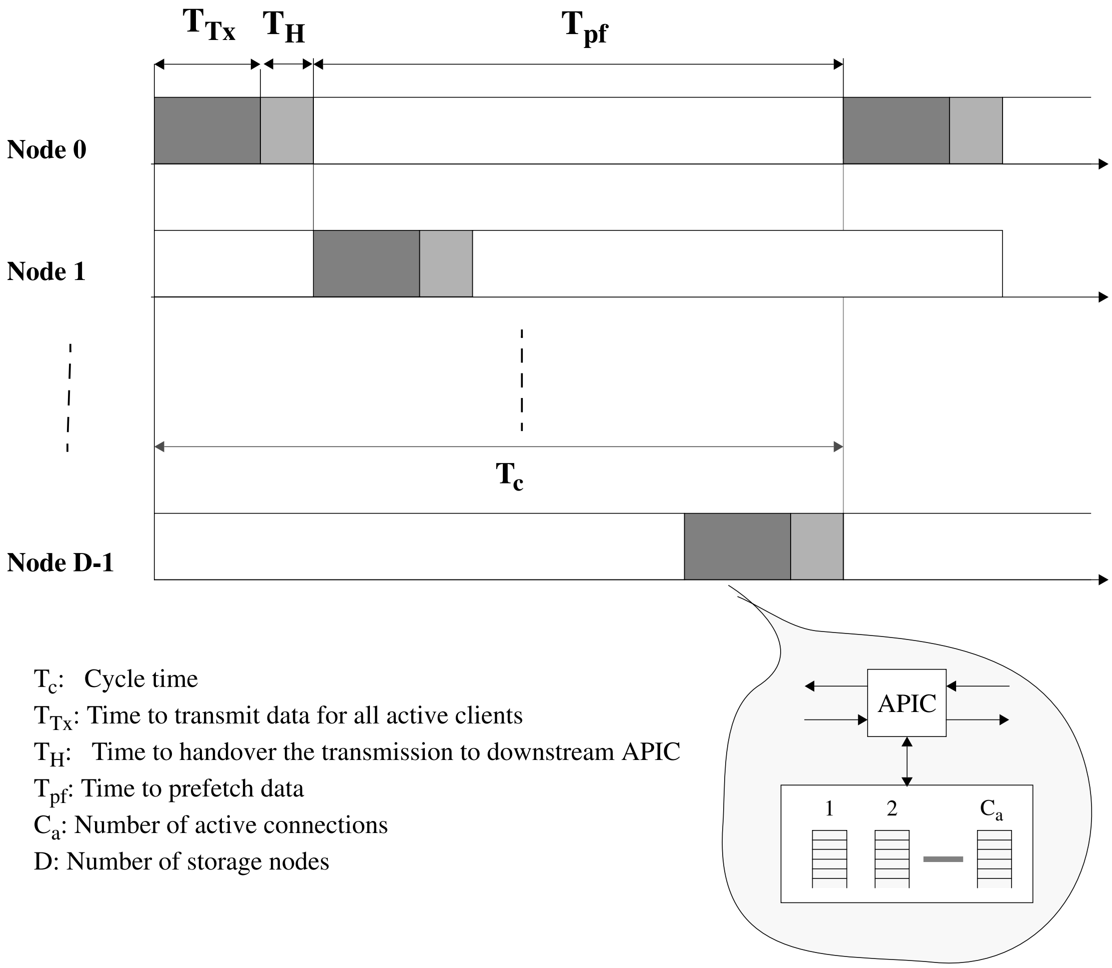

Figure 4: A simple scheme for reads

As shown in Figure 4, each storage node maintains C_a buffers, one for each active connection. In a retrieval environment, the data read from the disks at the storage node is placed in these buffers and read by the APIC to transmit it on to the interconnect and subsequently to the network. At the time of connection admission, every stream experiences a playout delay required to fill the corresponding buffer, after which the data are guaranteed to be periodically read and transmitted as per a global schedule.

The global schedule consists of periodic cycles of time length T_c. Each cycle consists of three phases: *data transmit*, *handover*, and *data prefetch*. During the *data transmit phase* (T_{Tx}), the APIC corresponding to a storage node reads the buffer and transmits it over the interconnect to the network. Once this phase is over, the APIC sends control information (control cell) to the downstream APIC, so that it can start its data transmit phase. The last phase in the cycle, namely the *data prefetch phase* (T_{pf}) is used by the storage node to prefetch data for each connection that will be consumed in the next cycle.

The cycle length T_c determines the buffer and the network and storage bandwidth requirements. T_c depends on the natural interframe period T_f of the active streams. In case of heterogeneous connections, the smallest interframe period T_f among all connections decides the value of T_c. The three cases that arise are as follows:

- $\mathbf{T_c = (T_f + T_H) \times N}$: Since each node is assigned equal share of the cycle, the transmit phase length for this case is T_f. This implies that in every cycle only a frame worth of data is prefetched for each connection.

- $\mathbf{T_c = \frac{((T_f+T_H)\times N)}{k}}$ **and k is an integer:** In this case, less than a frame worth of data is prefetched and transmitted for each active connection in every cycle. This can help minimize the buffer requirement at each storage node. However, there are two overheads associated with this scenario: first, as the cycle time is reduced the aggregate handover overhead can become a sizable fraction of the cycle time, and second, a smaller prefetch buffer makes it difficult to mask the seek, rotational, and transfer latency overhead.

- $\mathbf{T_c = k \times ((T_f + T_H) \times N)}$ **and k is an integer:** This case implies that more than one frame worth of data is fetched and transmitted for each active connection in every cycle. Such a case allows more time to prefetch and mask the seek and rotation latencies.

As an example, consider a prototype with 15 storage nodes ($N = 15$) serving a fixed number of video connections which have frame period of $T_f = 33$ msecs (ms). If the APIC interconnect bandwidth is 622 Mbps, the effective data bandwidth available is 563 Mbps, excluding the bandwidth lost due to the ATM header overhead in each cell. Using these values, if the cycle time is set to $T_c = 33 \times 15 \approx 495$ ms, the length of the transmit phase can be at most $T_{Tx} = 33$ ms. Thus, each storage node has to prefetch a frame worth of data for each connection every $((495 - 33) - T_H) = 462 - T_H$ ms. Assuming that the T_H is of the order of 1 ms, the prefetch time is 461 ms. A simple RAID constructed using disks with maximum rotational and seek latencies of 10 ms can deliver 5 MBps. Thus, a storage node with such a RAID can prefetch 2.3 megabytes of data in 461 ms. A buffer of this size can store approximately 27 frames, each of 84 KB - the average size of a MPEG encoded 20 Mbps HDTV video stream. Since each frame belongs to an independent connection, ≈ 27 independent HDTV connections are possible. The total bandwidth requirement of these connections is 27×20 Mbps $= 540$ Mbps, which is less than the effective interconnect bandwidth of 563 Mbps. Thus, ≈ 27 compressed HDTV clients can be supported simultaneously. Similar calculations show that approximately 110 MPEG compressed NTSC quality clients can be supported in this setup.

It must be noted that the disk and array level scheduling policies [8] are required to guarantee retrieval of data for all active connections during the prefetching period. To summarize, the two advantages that accrue from our data layout and scheduling schemes are as follows:

- First, the successive frames of a multimedia stream stored at a storage node are separated in time by a duration much longer than the normal frame period and thus, the effective period of the stream seen at each node is longer. This allows masking the disk rotational and seek latencies and in turn facilitates other connections that access different data to be served between two successive frame retrievals of any stream.

- Since the frames of a given multimedia stream are physically distributed on multiple storage nodes, multiple numbers of clients can independently access different frames in the same stream by accessing different nodes. This allows a large number of concurrent accesses to the same data.

3.2.3 Storage Node

The architecture of the storage node described here assumes the storage in the form of a RAID, however it can be extended easily to accommodate other forms of storage such as a set of independent high capacity optical and magnetic disks. A RAID storage node is illustrated in Figure 5. The disk array at the node is constructed out of a set of *Small Computer System Interconnect* (SCSI) strings, with a fixed number of disks per string. The multiple SCSI strings are controlled by an array controller, which interfaces to the APIC through a dual ported memory, such as a video RAM (VRAM).

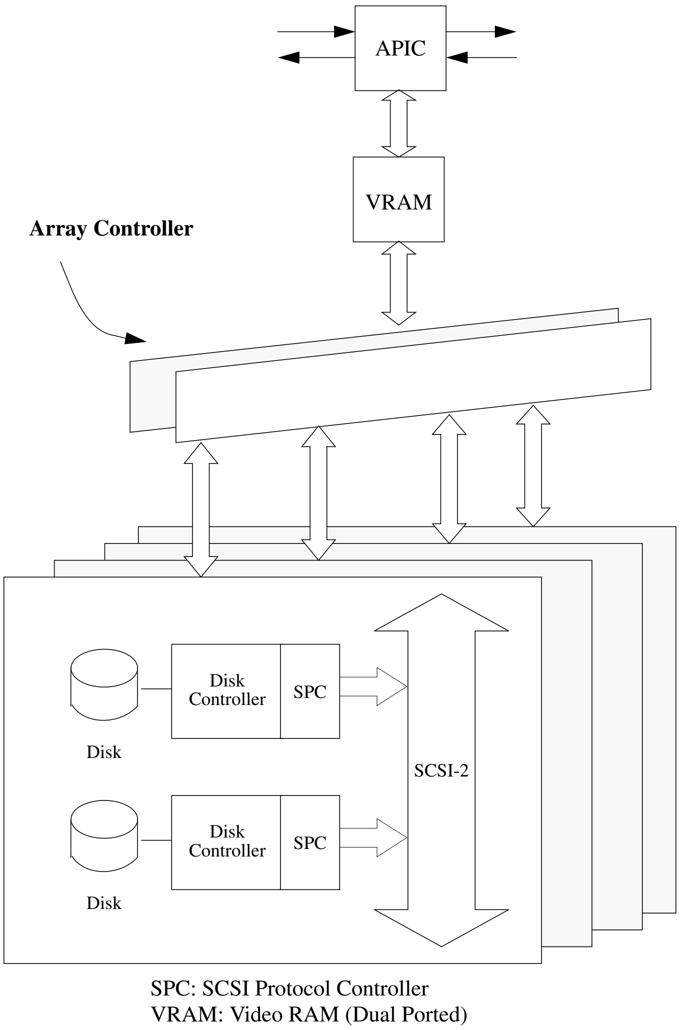

Figure 5: Storage node architecture

As shown in Figure 6, in a read-only environment, the array controller asynchronously writes the data in the VRAM and the APIC consumes it to transfer it to the network. The VRAM is shared by the following buffers:

- **Buffers for periodic streams:** A set of circular buffers serve as the prefetching buffers for the data from periodic streams retrieved off the disk array. These buffers can be statically reserved on a per connection basis at the connection setup time or dynamically allocated during every cycle of the scheduling scheme described earlier. The size of allocated buffers depends on the media type as well as the playback rate. For example, the buffer requirement of a video stream is much larger than that of an audio stream. Also, a video stream played at 15 frames/sec needs much smaller buffers than the one played at the standard 30 frames/sec.

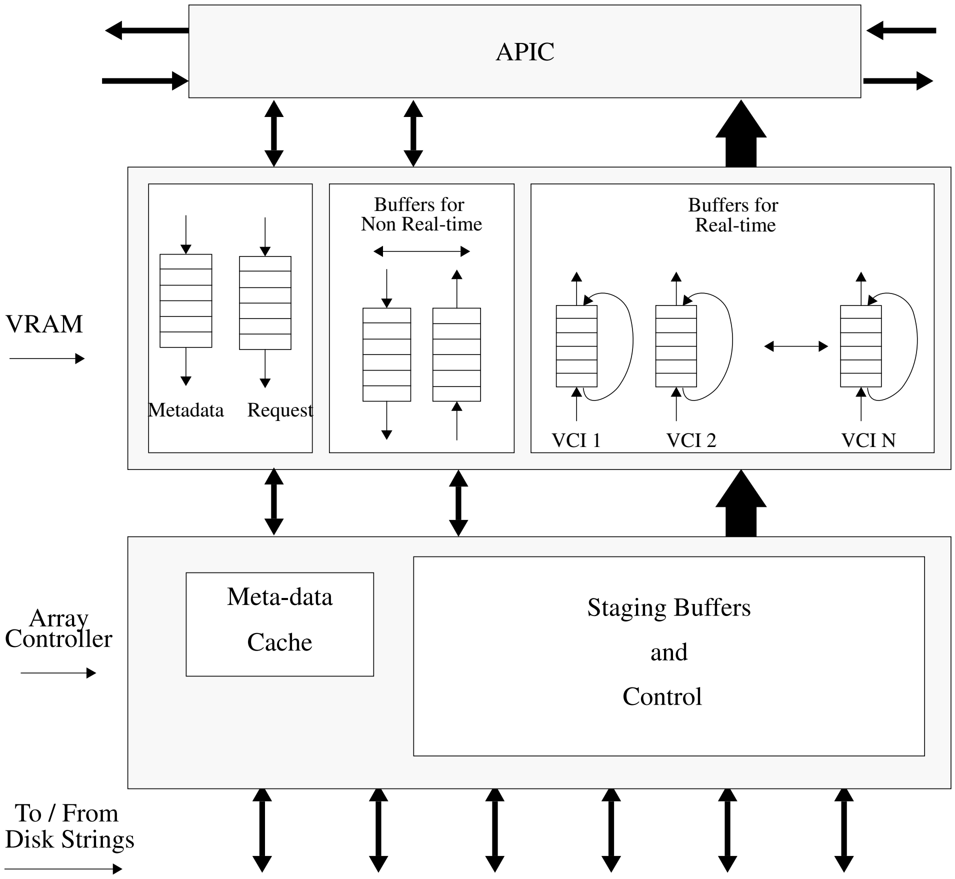

Figure 6: Details of node-APIC interface

The data compression techniques have implications for the buffer allocation policy as well as the buffer requirements. For example, the buffer required for a JPEG encoded stream is significantly larger than its MPEG encoded counterpart. Another characteristics of the compressed streams is that the size of successive frames in the media streams varies significantly, causing a frame-to-frame variability in the buffer requirements. For example, empirical evidence has shown a factor of four to five variability in the data content of single frame of MPEG compressed video streams. Thus, the static allocation of buffers, though simple, requires more memory because the size of the buffer allocated at the connection set up time has to be the maximum expected size of a frame.

- **Buffers for non-real time tasks:** A set of buffers are used for read/writes of non real-time streams, such as still images, text, and progressive image transmissions.

- **Request buffers:** The central manager shown in Figure 2 writes control commands for read and write schedule management in a request queue in the VRAM (Figure 5). The array controller reads these commands to initiate appropriate SCSI transactions to retrieve data from the storage devices. The request queue is partitioned based on the connection identifier or the VCIs.

- **Positional metadata buffers:** The positional metadata is the information about the location of the data on the storage devices. This data is provided by the central manager and used by the array controller to read (write) disks in the array. Like the request queue, the metadata queue is also partitioned based on VCIs.

The array controller is responsible for managing the local storage and providing one or more of the resource management functions outlined earlier. It maintains small staging buffers similar to the ones maintained in the VRAM. In order to minimize accesses to the metadata queues in the VRAM, the array controller maintains a metadata cache. The control block, the heart of the array controller, is a finite state machine or an embedded processor that executes the SCSI protocol for disk read/writes and controls the staging buffer and metadata caches. Due to space constraints, the detailed design of the array controller is not presented here.

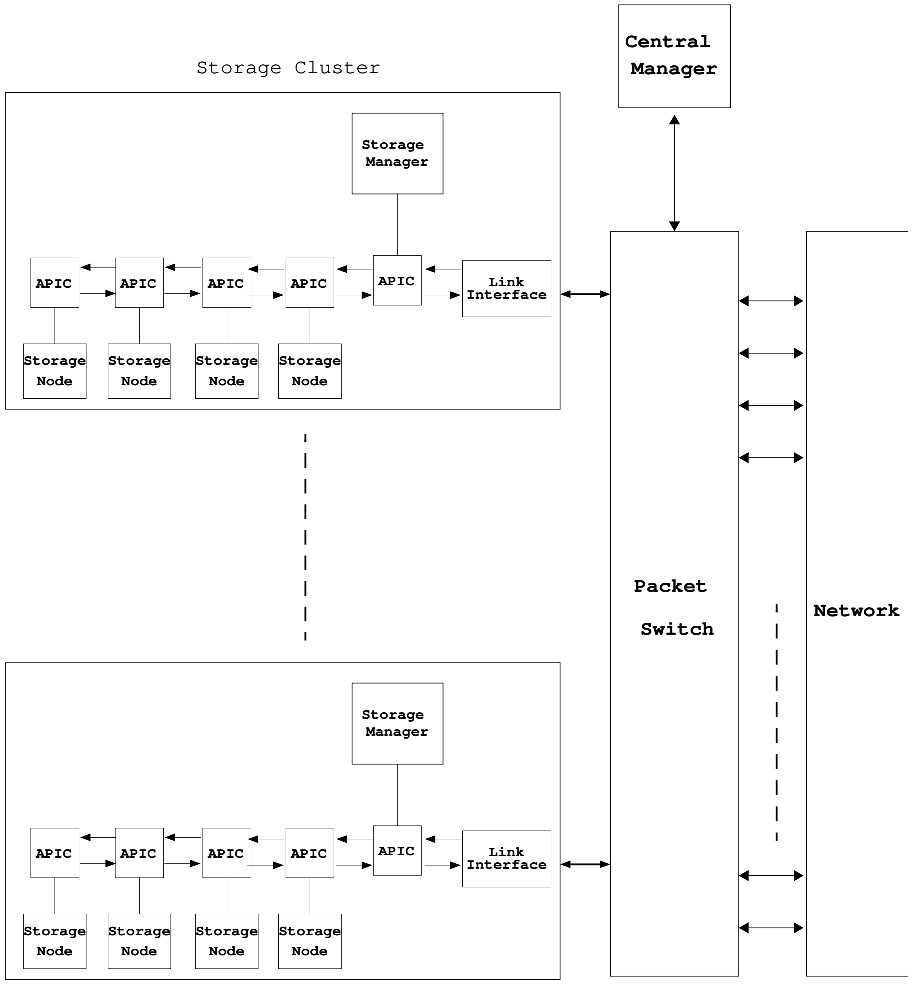

Figure 7: Cluster Based Storage (CBS) architecture for MARS

3.3 A Scalable Extension

The Cluster Based Storage (CBS) architecture, which is a scalable extension of the prototype implementation discussed earlier, is illustrated in Figure 7. It consists of a set of independent clusters interconnected through a fast multicast ATM switch, such as [26, 27]. Each of the clusters resembles our prototype implementation architecture. The APIC at the end of the daisy chain in a cluster transparently interfaces to one port of an ATM switch. Each cluster has a local cluster manager. The central manager in this architecture, manages the switch, performs the network signaling operations, and co-operates with the cluster level storage managers to perform admission control.

Two types of information flows between the CBS server and the external network: one type is the control information such as the client requests for multimedia streams, stream manipulation commands such as fast forward, reverse, pause etc., server responses, and the network signalling information. The other type of information is the actual multimedia data for all active connections. Depending upon the implementation, the CBS architecture may reserve certain switch ports for control information and the rest of the ports for data traffic.

The scalability of this architecture accrues from increasing the number of ports and the bandwidth per port of the ATM switch, a trend supported by advances in the field of ATM switching. Note that the availability of multiple clusters provides a greater number of options for data distribution. For example, popular documents can be replicated in multiple clusters and incoming requests for them can be assigned to one of these clusters. If the number of documents is very large, a subset of them can be assigned and served from each cluster. Also, the documents of very long duration can be conveniently split into smaller parts, each of which can be stored independently on a separate cluster. Note that the packet switch in this architecture allows a connection from any input port to be switched and re-mapped to a different network level connection at any output port. This allows multimedia data to be striped not only over the storage nodes in a cluster, but over multiple clusters as well. The increased parallelism that results from greater degree of data distribution in turn, can support larger number of concurrent accesses to the same document. In short, the CBS architecture is truly scalable in terms of number of clients, storage capacity and storage and network throughput.

4 Conclusions and Work in Progress

In this paper, we addressed the challenging problem of designing large scale multimedia storage servers that will be an integral part of the emerging distributed multimedia computing infrastructure. We presented the five basic requirements of such large scale servers and proposed the concept of *Massively-parallel And Real-time Storage* (MARS) architecture to meet them. We described a prototype implementation of a MARS that uses an innovative APIC based daisy chain ATM interconnect and provides a direct path for data from storage devices to the network.

In conjunction with this architecture, we illustrated a simple distributed layout scheme and an associated scheduling scheme. We showed that an example system, that uses a 622 Mbps ATM interconnect and fifteen storage nodes, each using a simple disk array, can support approximately 27 compressed HDTV clients or 110 MPEG compressed standard NTSC video clients in normal playout mode. Also, using a packet switch to interconnect multiple clusters, each of which resembles our prototype implementation, we get a truly scalable architecture. We believe that ATM will become a pervasive interconnect technology in near-future, and the VLSI chips such as APIC will be inexpensive and available off-the-shelf. Thus, we satisfy all our primary requirements of a large scale server.

In our ongoing work, we are developing multilevel data layout, real-time scheduling, and admission control algorithms to guarantee QOS requirements to a large number of clients. These algorithms will provide such QOS guarantees during normal playout as well as a full spectrum of stream control operations, such as fast forward, rewind, slow play, slow rewind, frame advance, pause, stop, and stop-and-return. We will demonstrate these ideas in an implementation of our prototype architecture. We also plan to extend our ideas to a server with a hierarchical storage system that comprises of a large on-line and near-line robotically controlled optical juke box storage.

5 References

6 *

References

[1] —-, "Storage Concepts, Concept 810-SW real-time RAID," Product Description, Storage Concepts, Inc. Irvine, CA.

[2] Buddhikot, M., Parulkar, G. and Cox, J., "Scheduling, Data Layout and Playout Control in a Large Scale Multimedia Storage Server," Technical Report in preparation, Department of Computer Science, Washington University in St. Louis, Sept. 1994.

[3] Buddhikot, M., Parulkar, G. and Cox. J. "Distributed Layout, Scheduling, and Playout Control in a Multimedia Storage Server," To appear in the Proceedings of the Sixth International Workshop on Packet Video, Sept. 1994.

[4] Buddhikot, M., Parulkar, G., and Cox, Jerome, R. Jr., "Design of a Large Scale Multimedia Storage Server," Proceedings of the INET'94/JENC5, Conference of the Internet Society and the Joint European Networking Conference, Prague, June, 1994.

[5] Cao, Pei, et al., "The TickerTAIP Parallel RAID Architecture," Proceedings of the 1993 International Symposium on Computer Architecture, May 1993.

[6] Chen, P., et al., "RAID: High-performance, Reliable Secondary Storage," submitted to ACM Computing Surveys.

[7] Chervenak, A, "Performance Measurements of the First RAID Prototype," Technical Report, Department of Computer Science, University of California, Berkeley, 1990.

[8] Daigle, S., "Disk Scheduling for Continuous Media Data Streams," Master's Thesis, Carnegie Mellon University, Dec. 1992.

[9] Dailianas, A., and Bovopoulos, A., "Real-time Admission Control Algorithms with Delay and Loss Guarantees in ATM Networks," Proceedings of IEEE INFOCOM'94, Vol. 3, pp. 1065-1072, June, 1994.

[10] Dittia, Z., Cox., J., and Parulkar, G., "Catching up with the networks: Host I/O at gigabit rates," Technical Report WUCS-94-11, Department of Computer Science, Washington University in St. Louis, Apr. 1994.

[11] Golestani, J., "Congestion-Free Communication in High-Speed Packet Networks," IEEE Transactions on Communications, Vol. 39, No. 12, pp. 1802-1812, Dec. 1991.

[12] Haskins, R., "The *Shark* Continuous-Media File Server," Proceedings of IEEE Compcon'93, San Francisco CA, Feb. 1993.

[13] Hylton, Todd., Coffey., K., Parker, A., and Kent, H., "AdStaR Scientists Detect Giant Magnetoresistance in Small Magnetic Fields, Using Easy to Make Sensor," SCIENCE, August, 1993.

[14] Guttag, K., Gove, R., and Aken, V., "A Single-Chip Multiprocessor for Multimedia: The MVP," IEEE Computer Graphics and Applications, pp. 53-64, Nov. 1992.

[15] Lee, E., et al., " RAID-II: A Scalable Storage Architecture for High-Bandwidth Network File Service," Technical Report UCB//CSD-92-672, Department of Computer Science, University of California at Berkeley, Oct., 1992.

[16] Little, T., D., et al., "A Digital On-demand Video Service Supporting Content-based Queries," Proceedings of ACM Multimedia'93, Anaheim, CA, pp 427-436, Aug. 1993.

[17] Lougher, P., and Shepherd, D., "The Design of a Storage Server for Continuous Media," The Computer Journal, Vol. 36, No. 1, pp. 32-42, 1993.

[18] LoVerso, S., Isman, M., et al. "sfs: A Parallel File System for the CM-5," Proceedings of USENIX Summer Conference, June, 1993.

[19] Miller, G., Baber, G., and Gilliland, M., "News On-Demand for Multimedia Networks," Proceedings of ACM Multimedia'93, Anaheim, CA, pp. 383-392, Aug. 1993.

[20] Patteson, D., et al., "A Case for Redundant Arrays of Inexpensive Disks (RAID)," Proceedings of the 1988 ACM Conference on Management of Data (SIGMOD), Chicago IL, pp. 109-116, June 1988.

[21] Rangan, V., and Vin, H., "Designing File Systems for Digital Video and Audio," Proceedings of the 13^{th} Symposium on Operating System Principles, Operating Systems Review, pp. 81-94, Oct. 1991.

[22] Reddy, N. and Wyllie, J., "Disk Scheduling in a Multimedia I/O System," Proceedings of ACM Multimedia'93, Anaheim, CA, pp. 225-233, Aug. 1993.

[23] Rowe, L., Boreczky, J., and Eadds, C., "Indexes for User Access to Large Video Databases," IS & / SPIE Symposium on Electronic Imaging Science and Technology, Conference 2185, 1994.

[24] Salem, K., and Garcia-Molina, H., "Disk Striping," IEEE International Conference on Data Engineering, 1986.

[25] Tobagi, F., Pang, J., Baird, R., and Gang, M., " Streaming RAID - A Disk Array Management System for Video Files," Proceedings of ACM Multimedia'93, Anaheim, CA, pp. 393-400, Aug. 1993.

[26] Turner, J., "An Optimal Nonblocking Multicast Virtual Circuit Switch," Proceedings of IEEE INFOCOM94, Vol. 1, pp. 298-305, June, 1994.

[27] Turner, J., "A Gigabit Multicast Switch: System Architecture Document," Applied Research Laboratory, Washington University in St. Louis, Feb., 1994.

[28] Turner, J., "Bandwidth Management in ATM Networks Using Fast Buffer Reservation," IEEE Networks Magazine, Vol. 6, No. 5, Sept. 1992, pp. 50-58.

[29] Venkatramani, C., and Chiueh, T., "Survey of Near-Line Storage Technologies: Devices and Systems," Experimental Computer Systems Laboratory, Technical Report #2, Department of Computer Science, SUNY Stony Brook, NY, Oct. 1993.

[30] Vin, H., and Rangan, V., "Design of a Multi-user HDTV Storage Server," IEEE Journal on Selected Areas in Communication, Special issue on High Definition Television and Digital Video Communication, Vol. 11, No. 1, Jan. 1993.

[31] Yu, P., Chen., M., and Kandlur, D., "Grouped Sweeping Scheduling for DASD based Storage Management," Multimedia Systems, Springer-Verlag, pp. 99-109, Dec. 1993.

[32] Zou, W., Y., "Digital HDTV Compression Techniques for Terrestrial Broadcasting," High Definition (HD) World Review, Vol. 3, No. 3, pp. 4-10, 1992.

Author Information

Milind Buddhikot received the Bachelor of Engineering (B.E.) degree in electrical engineering with Distinction from the University of Bombay, India in August, 1987 and the Master of Technology (M.Tech.) degree in communications engineering from the Indian Institute of Technology, Bombay in December, 1988. He is presently a D.Sc. candidate in the Department of Computer Science at Washington University in St. Louis. His research interests are in the area of high speed networking and multimedia computing.

Dr. Guru Parulkar is an Associate Professor of Computer Science at Washington University in St. Louis. He holds the Ph.D. in computer science from the University of Delaware (1987), the M.Tech. in electrical engineering from the Indian Institute of Technology, Bombay (1983), and the B.E. in electronics and communications from the University of Indore (1981). His research emphasis has been on the design, specification, and implementation of protocols to support distributed computing applications involving visualization, multimedia, and collaboration as components on high speed networks.

Dr. Jerome R. Cox, Jr. received an Sc.D. in electrical engineering from the Massachusetts Institute of Technology in 1954. He is the Welge Professor of the Department of Computer Science at Washington University, St. Louis, Missouri, and served as a chairman of the department from 1975 to 1991. He is currently director of the Applied Research Laboratory (ARL) and research director of the Electronic Radiology Laboratory. His research interests are digital communication, computer systems design and electronic radiology. He is a Senior member of the Institute of Medicine of the National Academy of Sciences and is a Fellow of the Institute of Electrical and Electronics Engineers and the Acoustical Society of America.