Performance Model of the Argonne Voyager Multimedia Server

Terrence Disz, Robert Olson, and Rick Stevens
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439
{disz,olson,stevens}@mcs.anl.gov

Abstract

The Argonne Voyager Multimedia Server is being developed in the Futures Lab of the Mathematics and Computer Science Division at Argonne National Laboratory. As a network-based service for recording and playing multimedia streams, it is important that the Voyager system be capable of sustaining certain minimal levels of performance in order for it to be a viable system. In this article, we examine the performance characteristics of the server. As we examine the architecture of the system, we try to determine where bottlenecks lie, show actual vs potential performance, and recommend areas for improvement through custom architectures and system tuning.

1 Introduction

The Argonne Computing and Communications Infrastructure Futures Laboratory (Futures Lab) [1] was created to explore, develop, and prototype next-generation computing and communications infrastructure systems. An important goal of the Futures Lab project is to understand how to incorporate advanced display and media server systems into scientific computing environments. The objective is to create new collaborative environment technologies that combine advanced networking, virtual space technology, and high-end virtual environments to enable the construction of virtual teams for scientific research.

The Voyager multistream multimedia server is one of the cornerstone projects in the Futures Lab. The goal of this project is to develop the next-generation hypermedia server architecture that will enable the construction and rapid deployment of tools for building virtual organizations. Voyager is designed to ultimately replace the types of servers that we currently use for supporting collaborative environments, tools such as ftp servers, Web servers, and document servers. In addition, Voyager will provide an extensible environment for making audio, video, and other stream-oriented recordings available to others on the network. We envision Voyager as the tool that each user in a virtual organization will use to publish his/her information for the rest of the organization's users.

Voyager is being designed to be deployed both at the desktop level and as a large, scalable server for high-performance media-serving applications. We have demonstrated the server at the Supercomputing 95 Conference, at the SuperComputing 96 Conference and at various DOE technology demonstrations. The voyager server is online now for users to view archives. We hope to soon make the server available to our colleagues for recording purposes as well.

2 Multimedia Server Architecture

The general model for a synchronous, scalable multimedia server is shown in Figure 1.

2.1 Client Side

Although we are concerned here mainly with the server, a few words about the client side are appropriate. With the growing presence of multimedia-enabled systems (those with video/audio encode and decode capabilities), we see an integration of collaborative computing concepts into the everyday environments of future scientific and technical workplaces. Desktop teleconferencing is in common use today, while more complex desktop teleconferencing technology that relies on the availability of multipoint (greater than two nodes) enabled tools is now starting to become available on PCs. It is this increasing desktop multimedia presence that motivates the design of a multimedia server. Ideally one would like the ability to capture, record, playback, index, annotate, and distribute multimedia stream data as easily as we currently handle text or still image data.

2.2 Network

Another motivating force for the multimedia server is the growing availability and quality of network presence. Most universities and research institutions have at least T1 capability, and we are seeing increased use

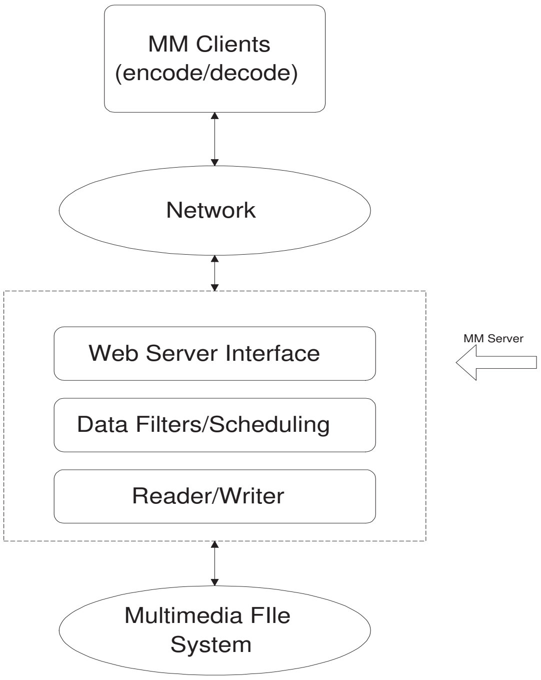

Figure 1: MultiMedia Server

of the Internet Protocol over high-speed ATM (Asynchronous Transfer Mode) networks [2] [3]. The evolution of networks such as the Metropolitan Research and Education Network (MREN), a high-performance ATM network; vBNS (very high speed Backbone Network Service), a national high-performance network devoted to meritorious research projects; and the Bay Area Gigabit Network (BAGNet) [4] are providing everyday access to high-speed networks for researchers and educators, making practical the transmission of multimedia streams. The evolution of the MBONE and regular use of these networks for teleseminars, as showcased by BAGNet, gives us a glimpse of the future of network-based multimedia conferencing.

For reference, Table 1 shows a summary of estimates of bandwidth requirements for various video encodings.

Table 1: Bandwidth Requirements

Encoding	Frame Size	Frame Rate	Bandwidth
JPEG	320x240	30 fps	5 Mbps
h.261	352x288	20 fps	128Kbps
h.261	352x288	30 fps	256Kbps
Uncompressed 8 bits per pixel	320x240	30 fps	17 Mbps
Uncompressed 24 bits per pixel	640x480	30 fps	210 Mbps

Audio bandwidth requirements are generally less stringent as shown in Table 2.

Table 2: Audio Bandwidth Requirements

Encoding	Sample Size	Mode	Rate	Bandwidth
PCM	8 bit	mono	8 KHz	64 Kbps
PCM	8 bit	stereo	16 KHz	256 Kbps
PCM	16 bit	stereo	44 KHz	1.4 Mbps

2.3 The Multimedia Server

To effectively serve a wide community, a Web-based multimedia server must be scalable and robust, yet easy to use and easily accessible. At a minimum, it must provide the following functions:

- Web service to manage the client interaction.
- Data filters/transcoders to provide consistent storage formats while maintaining the ability to play back streams in different modes, as requested by users.
- Scheduling mechanisms to ensure correct playback timing and synchronization.
- Ability to store and play continuous-time data.

We believe the server should provide these functions using readily available software and using Internet standards, in order to reach the widest possible community.

2.4 Multimedia File System

Central to the performance of a multimedia server is the filesystem into which the media streams are stored. Normal filesystems are not designed for continuous-time data; under load, a conventional file system may provide lower throughput and higher response times, thereby causing the server to drop incoming data when recording or miss playout deadlines on playback. A multimedia filesystem, on the other hand, is designed to support the demands of real-time storage and playback of continuous-time data streams.

3 The Voyager Implementation

The Voyager server is implemented on an IBM SP2 using a suite of commonly available software tools:

- The IBM Tiger Shark multimedia file system [5]
- The Perl language [6] [7]
- The Nexus run-time communications package [8]
- The ACE communications toolkit [9] [10]

- The LBNL multimedia tools Vic and Vat [11]
- Standard Realtime Transport Protocol (RTP) [12]

3.1 Voyager Hardware

The current Voyager system is implemented on an IBM 9076 SP2 [13].

This is a twelve-node machine. Eight of the nodes are configured as follows:

- SP1 thin node (RS6000/370 planar)
- 256M memory
- 2G disk
- 10Mbps ethernet
- OC3 ATM
- TB2 HPS adapter

The other four nodes are configured as follows:

- SP2 wide node (RS6000/590 planar)
- 256M memory
- 8G local disk
- 2 Fast/Wide SCSI adapters
- 18G Fast/Wide SCSI disk
- 10Mbps ethernet
- TB2 HPS adapter

The connection to the Internet on the ATM-equipped nodes is via OC3 ATM to a Cisco 7513 router. Non-ATM nodes connect to the Internet via ethernet to an RS/6000 970 with an ATM connection to the same Cisco router.

A SPARCstation 20 serves as the Voyager Web and database server.

3.2 Server Software

Voyager relies on the IBM Tiger Shark filesystem [5], now part of the IBM Multimedia Server product, to provide reliable access to the 72 GB of fast/wide SCSI disk that is striped across several nodes.

The Tiger Shark filesystem is present on the eight thin nodes. We use the IBM Virtual Shared Disk to make the fast/wide disk devices, resident on the wide nodes, visible to the filesystem on the thin nodes.

Media streams are played between disk and network with Voyager playback and recording daemons that run on the fileserver nodes. The available content is catalogued by a relational database. The session daemons are instantiated by a set of CGI programs on the Web server that participate in a distributed nPerl server control application.

The media streams are transported by using RTP, the Realtime Transport Protocol as specified in RFC 1889 [12], and RFC 1890 [14]. Video is encoded by using either Motion JPEG [15] or h.261 [16]. Audio typically is encoded by using PCM.

3.3 Client Hardware

The hardware that we have used at ANL includes

- RS/6000 41T workstations, with the IBM Ultimedia video and audio adapters and Turboways OC3 ATM adapters. This platform supports hardware JPEG compression and decompression with analog video output.
- RS/6000 43P workstation, with the onboard audio, Parallax video capture adapter, and Cheetah PCI ATM adapter. This platform also supports hardware JPEG compression and decompression.
- PCs running Windows95 and Windows NT. We currently do not have video capture available in these machines, but audio capture and audio/video playback are operational.
- Other Unix workstations, including Sun SPARCstation and SGI Onyx, Indigo, and Indy.

For playback, no specific hardware is required.

3.4 Client Software

A client needs the following software to view and record media sessions in Voyager:

- A Web browser that supports forms
- RTP-compliant video and audio clients

We use the Vic [11] video client and the Vat audio clients from LBNL on the workstation platforms. Ports of these tools are also available for Microsoft Windows. We have also used the RTP tools from Precept [17].

4 Theoretical Voyager Performance Limitations

In this section we discuss the performance limitations of the Voyager system that are dictated by the architecture of the system we are using. Figure 2 is a detailed schematic our SP system.

For each interconnect in the system we can determine (by reading hardware specifications or by other

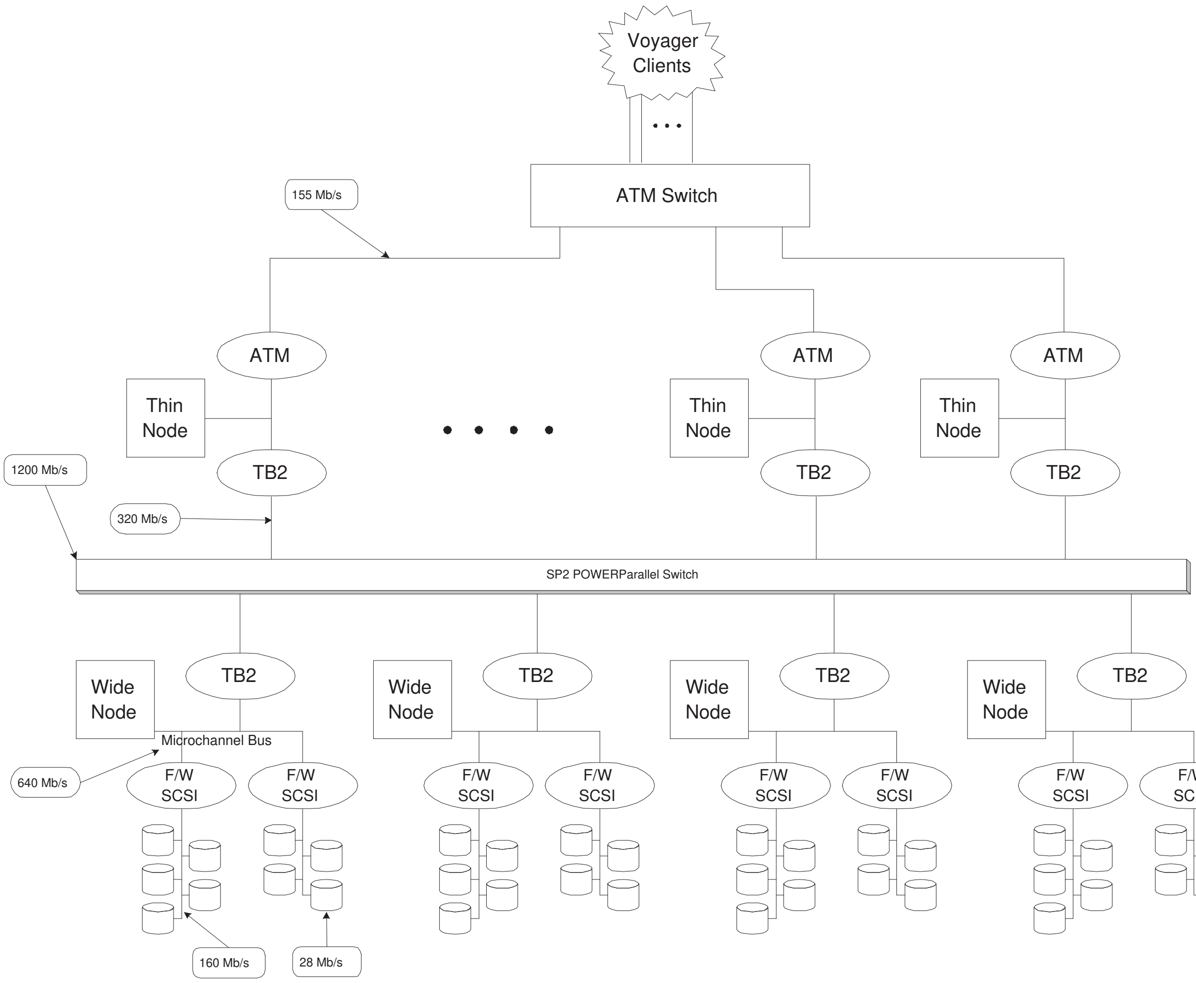

Figure 2:

Table 3: Ideal Component Bandwidths

Component	Bandwidth (Mb/s)	Max Streams
OC3 ATM	155	31
Microchannel Bus	640	128
TB2 Adapter	320	64
SP Switch	1200	240
Fast/Wide SCSI Bus	160	32
SCSI Disk	28	5

means) the best possible bandwidth. Given a maximum bandwidth on a connection, we then compute the maximum number of 5 Mb/s streams that we can transfer on that connection. Table 3 summarizes the bandwidth limits and the resulting stream capacities.

We can first draw some conclusions about the absolute maximum number of streams supportable by the server. There are eight 155 Mb/s OC3 ATM connections coming into the server. Hence, the ATM network imposes a maximum of 8 nodes $\times$ 31 streams per node $=$ 248 streams.

Other absolute maximums include

- TB2 adapter bandwidth imposes a 256-stream limit
- Total SCSI bus bandwidth imposes a 256-stream limit
- Total disk bandwidth imposes a 201-stream limit

We can see that from a theoretical standpoint the system is fairly well balanced. The limiting factor in the total bandwidth is the SCSI disk bandwidth, limiting the server to 201 streams. However, we know that we will not achieve in practice the bandwidths that we have laid out in this section. In order to optimally configure the server, we must empirically determine the bottlenecks in the system.

5 Experiments

We have been running a Voyager server in the Futures Laboratory for roughly two years as a resource for the development of the server itself and for intermittent demonstration and production use. We are currently upgrading the SP hardware on which Voyager runs and plan on making Voyager a solid part of the Futures Laboratory infrastructure. Toward that end, we wish to examine the performance of the Voyager system on the hardware we have in place in order to more completely understand the system, optimize the configuration, and plan for expansion.

We have performed several experiments to probe the actual performance of our SP hardware. These experiments exercise three of the potential bandwidth chokepoints in the system: the ATM network interface at the filesystem nodes, raw disk bandwidth and scalability, and performance of VSD-extended raw disk devices. We also probe the performance observed when running both the ATM network and the Tiger Shark file system.

The benchmarks use two basic application programs: a simple stream source and a flexible event-driven stream sink. Each is implemented in C++ and uses an ACE Reactor [9, 10] object to handle the demultiplexing of multiple streams and the invocation of timer callbacks.

The stream source is invoked with a desired bandwidth, block size, and target host and UDP port. It computes the packet transmission frequency

$$F = \frac{bandwidth}{blocksize}$$

and sends UDP datagrams of size *blocksize* at that rate to the specified host. Each datagram is tagged with a stream identifier and a sequence number. The sender logs the number of packets it sends.

The stream sink listens on a given UDP port for data streams from the sender. It demultiplexes multiple streams based on the stream identifier. For each stream, it gathers statistics on the first and last sequence numbers received and the number of packets received. These statistics are logged at the end of the run. The packets received from the network can optionally be routed to disk, one file per stream.

The stream sink application has the additional capability of determining precise CPU utilization for the duration of the run. The IBM AIX operating system maintains a set of counters that contain cumulative counts of the number of clock ticks spent in idle, user mode, kernel mode, and wait states. The stream sink can be configured to probe the counters at the start and finish of the run and at periodic intervals during the run. We use this information to determine the amount of CPU loading induced by the various experiments.

5.1 ATM Network Performance

The network performance benchmark measures the number of fixed-bandwidth streams that an SP node can source or sink without losing packets. We tested the capacity of the node both to send multiple streams and to receive multiple streams. The sending experiment placed multiple stream sources on one SP thin node, and distributed stream sinks across the other seven thin nodes and three workstations. The receiving experiment placed a single stream sink on one SP thin node and stream sources on the other thin nodes and the same three workstations. For each run we logged the CPU utilization and packet loss rates.

Figure 3 is a plot of the CPU utilization and packet loss rate versus the number of streams for one of the runs. Note that the sum of user and kernel CPU utilization is roughly linear with respect to the number of streams, up to full utilization. Hence, we can compute a best-fit line for the CPU utilization and determine a value for the percentage CPU utilization per stream. Note also that the packet loss rate begins to rise when full CPU utilization is reached. The point at which the packet loss begins to rise defines the maximum number of streams a node can sustain. We summarize these results in Table 4.

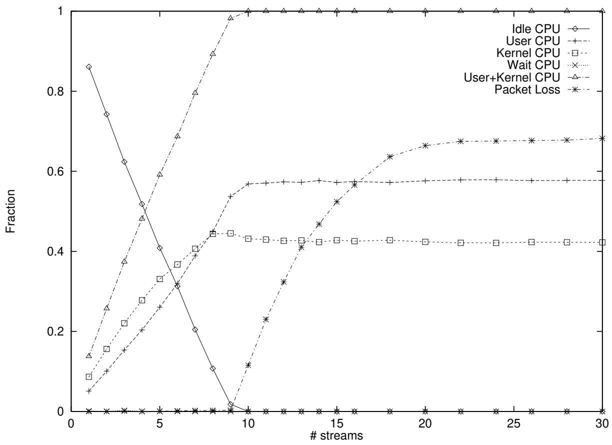

Figure 3: Node network-only performance

5.2 Raw Disk Performance

The next benchmark measures the performance of the disks used in the Voyager multimedia filesystem.

Table 4: Node network performance

Operation	Bandwidth	Block Size	%CPU per Stream	Max Streams
Receive	5Mb/s	4096	10.5	9
Receive	5Mb/s	8192	6.6	14
Receive	128Kb/s	512	2.0	50
Send	5Mb/s	4096	12.6	8
Send	5Mb/s	8192	8.2	12
Send	128Kb/s	512	2.7	37
Send	128Kb/s	1024	1.5	66

This experiment is somewhat different from the others in that it does not compute a maximum number of fixed-bandwidth streams; rather, it measures the maximum bandwidth a single writer can obtain to a disk or set of disks. The experiment used the AIX `dd` command to write to the raw disk device. When testing multiple disks multiple copies of `dd` were run, each writing to a different disk device.

We ran two versions of this test. The first was run on a SP wide node writing to locally attached disks. The results of this test are plotted in Figure 4. The second test was run on a SP thin node, accessing a set of disks residing on one of the wide nodes via VSD. The results of both runs are summarized in Table 5, where we have computed the aggregate and average per-disk bandwidths.

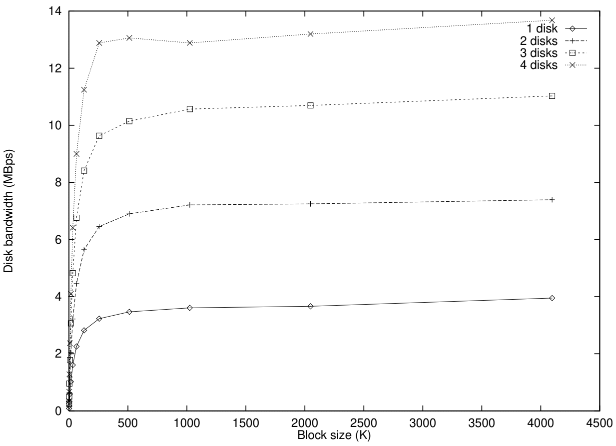

Figure 4: Raw disk bandwidth

Significant in the results of this experiment is the fact that the bandwidth to locally attached disks scales fairly well, showing that we have not yet saturated the SCSI bus. However, the per-disk performance of the VSD disks is disappointing in two regards. Single-disk performance is degraded significantly from the locally attached disk, and scaling is

Table 5: Raw disk performance

Number and Type of Disks	Aggregate Bandwidth	Per-disk Bandwidth
1 local	3.58	3.58
2 local	7.04	3.52
3 local	10.42	3.47
4 local	13.14	3.28
1 VSD	2.80	2.80
2 VSD	4.43	2.22
3 VSD	4.81	1.60
4 VSD	5.56	1.39

poor. It is not immediately obvious from looking at the architecture of the system that this should be the case. There are a number of configuration and tuning parameters in the AIX network interface, TB2 adapter and VSD software; though we have already performed some tuning of the system, we suspect that the poor performance of VSD may be due to a misconfiguration of one or more of these parameters.

5.3 ATM to Tiger Shark Performance

The final experiment involved a precise model of a Voyager recording daemon. We ran a stream sink on a SP thin node, configured to write the stream data to a Tiger Shark filesystem. A varying number of stream sources were placed on other nodes and the workstations. We gathered the same data as in the ATM network performance benchmark: CPU utilization and packet loss rate.

We also varied the configuration of the Tiger Shark file systems into which the streams were written. We tested file systems that consisted of one-, two-, and three-node stripes. In each case a single disk was configured on each node.

Figure 5 is a plot of the CPU utilization and packet loss rate versus the number of streams for a representative run. We again see that the sum of the user and kernel CPU utilization is roughly linear with respect

Table 6: Node network/filesystem performance

Number of Disk Nodes	Bandwidth	Block Size	%CPU per Stream	Max Streams
1	5Mb/s	4096	18.6	5
1	5Mb/s	8192	13.3	7
1	128Kb/s	512	7.5	37
2	5Mb/s	4096	18.3	5
2	5Mb/s	8192	11.8	8
2	128Kb/s	512	2.7	37
3	5Mb/s	4096	18.2	5
3	5Mb/s	8192	11.2	8

to the number of streams; we summarize the results of performing the best-fit calculations for this data in Table 6.

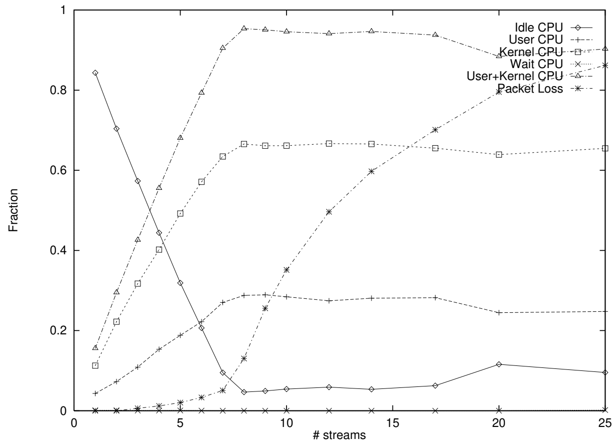

Figure 5: Node network/disk performance

5.4 Analysis

The clearest result of this set of experiments is the severe penalty paid in CPU use for driving streams to or from the ATM network interface. This penalty is due to the processing that the UDP and IP protocols require: checksum calculations, segmentation and reassembly, context switching, and data copying. Relieving the system CPU of the responsibility of this processing will increase the capacity of the node by making more CPU time available for the Voyager server application daemons.

Zero-copy ATM adapter technology is one solution to this problem. Researchers at IBM have built a prototype ATM adapter called *Cheetah* which uses DMA to transfer data between the ATM network and main memory [18, 19]. The system CPU is only involved in the setup of packet transfers. We have demonstrated the use of this adapter in a Voyager client machine, where it proved to work very efficiently.

Such technology would prove very useful in the server itself. Unfortunately, we cannot currently utilize the Cheetah technology in the server: Cheetah is restricted to use on the PCI bus, while the nodes in the Voyager SP are based on a microchannel bus. However, we can make use of the newly-available raw AAL5 ATM interface on the SP nodes, which bypasses the UDP and IP protocol stacks. We will be experimenting with this technology after the upgrade of the Voyager SP hardware, performing another set of benchmark experiments to determine the new balance of bandwidth chokepoints in the upgraded hardware.

6 Concluding Remarks

In this article, we have presented a technical description of a scalable multimedia server. We have shown theoretical limits in our implementation and measured actual limitations through a series of experiments. Through these experiments, we have sought to determine the sources of loss of performance in the server. We have investigated sources of contention and overhead and have identified at least two actionable sources.

We have discovered that file system overhead is more than expected and does not scale as well as we had expected. More investigation into file system tuning for Tiger Shark and the IBM Virtual Shared Disk is needed. We will continue these experiments and seek to improve the file system performance.

Secondly, we have discovered a source of overhead in the protocol stack driving the ATM connection. We have some evidence that the newly available zero-copy ATM driver from IBM or a raw AAL5-ATM interface will work well to reduce this overhead.

Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

References

[1] Terrence L. Disz, Remy Evard, Mark W. Henderson, William Nickless, Robert Olson, Michael E. Papka, and Rick Stevens, "Designing the future of collaborative science: Argonne's futures laboratory," *IEEE Parallel and Distributed Technology Systems and Applications*, vol. 3, no. 2, pp. 14–21, Summer 1995.

[2] D. McDysan and D. Spohn, *ATM: Theory and Application*, McGraw Hill, 1995.

[3] M. Laubach, "IP over ATM and the construction of high-speed subnet backbones," *ConneXions*, vol. 8, no. 7, July 1994.

[4] D. Wiltzus, L. Berc, and S. Devadhar, "BAGNet: Experiences with an ATM metropolitan-area network," *ConneXions–The Interoperability Report*, vol. 10, no. 3, March 1996.

[5] R. Haskin and F. Schmuck, "The Tiger Shark file system," in *Proceedings of the IEEE Computer Conference, 1996*. IEEE, March 1996.

[6] Randall Schwartz, *Learning Perl*, O'Reilly and Associates, 1993.

[7] Larry Wall, Tom Christiansen, and Randall Schwartz, *Programming Perl*, O'Reilly and Associates, 1996.

[8] I. Foster, C. Kesselman, and S. Tuecke, "The Nexus approach to integrating multithreading and communication," *JPDC*, vol. 37, pp. "70–82", 1996.

[9] Douglas C. Schmidt, "The adaptive communication environment an object-oriented network programming toolkit for developing communication software," in *Proceedings of Sun Users Group Conference*, December 1993.

[10] Douglas C. Schmidt, *Pattern Languages of Program Design*, chapter Reactor: An object behavioral pattern for concurrent event demultiplexing and event handler dispatching, Addison-Wesley, 1995.

[11] S. McCanne and V. Jacobsen, "Vic: A flexible framework for packet video," in *ACM Multimedia 95*. ACM, November 1995, pp. 511–522.

[12] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobsen, "RTP: A transport protocol for real-time applications," January 1996, Network Working Group, RFC 1889.

[13] T. Agerwala, J. L. Martin, J. H. Mirza, D. C. Sadler, D. M. Dias, and M. Snir, "SP2 systems architecture," *IBM Systems Journal*, vol. 34, no. 2, 1995.

[14] H. Schulzrinne, "RTP profile for audio and video conferences with minimal control," January 1996, Network Working Group, RFC 1890.

[15] L. Berc, W. Fenner, R. Frederick, and S. McCanne, "RTP payload format for JPEG-compressed video," October 1996, Network Working Group, RFC 2035.

[16] T. Turletti and C. Huitema, "RTP payload format for H.261 video streams," October 1996, Network Working Group, RFC 2032.

[17] Inc. Precept Software, *Precept IP/TV Viewer Users Manual*, Precept Software, Inc., initial release edition, June 1996, Part Number 201.

[18] Ronald Mraz, Douglas Freimuth, Edward Nowicki, and Gabriel Silberman, "Using commodity networks for distributed computing research," Tech. Rep., IBM T.J. Watson Research Center, 1995.

[19] Lucas Womack, Ronald Mraz, and Abraham Mendelson, "A study of virtual memory MTU reassembly (VMMR) within the PowerPC architecture," in *Proceedings of the Fifth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '97)*. IEEE MASCOTS97, January 1997.