A Measurement Based Memory Performance Evaluation of Streaming Media Servers

Garba Isa Ya'u and Abdul Waheed

PEL Technical Report
PEL-2002-03

Computer Engineering Department
College of Computer Science and Engineering
King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia

June 5, 2002

Abstract

Performance of streaming media servers is a key contributor to the success of streaming multimedia applications; either live broadcast of events or video on demand. While according to Moore's law, processor speed has been doubling roughly every eighteen months, memory and disk access speeds increase at the rate of only about 10% per year Therefore, bottleneck in server performance has shifted from processors to memory and disk. Memory hierarchy performance, and cache performance in particular, is the limiting factor in the performance of high throughput streaming servers. We carried out a measurement-based study of the memory performance of two leading streaming media servers: Darwin streaming server and Windows media server. Our goal is to determine the specific conditions under which on-chip cache or main memory becomes a major bottleneck for the performance of these streaming media servers. Our measurements indicate that at large number of client requests, the memory performance degrades significantly, leading to large number of cache misses and page faults. In addition to memory performance, we also compare the CPU usage and throughput of these streaming media servers.

1. INTRODUCTION

Since its introduction in early 1990s, the concept of streaming media has experienced a dramatic growth and transformation from a novel technology into one of the mainstream manners in which people experience the Internet today. Indeed, such growth would not be possible without adequate progress in the development of various core technologies utilized by streaming media software and hardware.

The concept of streaming media came at a time when basic multimedia technologies had already established themselves on desktop PCs. Audio and video clips were

digitized, encoded, and presented as files on the computer's file system. To view the information recorded in such files, PC users ran special software designed to decompress and render them on the screen. The first and most natural extension of this paradigm on the Internet was the concept of downloadable media. Compressed media files from the Web were downloaded on local machines, where they could be played back using standard multimedia software. However, this was not a satisfactory solution for users with limited amounts of disk space, slow connection speeds and/or limited patience. This essentially created the need for streaming media, a technology that enable a user to experience a multimedia presentation on the fly, while it is being downloaded from the Internet.

Streaming servers play a key role in providing streaming services. To offer quality-streaming services, streaming servers are required to process multimedia data under timing constraints and support interactive control operations such as pause/resume, fast forward, and fast backward. Furthermore, streaming servers need to retrieve media components in a synchronous fashion. These servers deliver live or on-demand audio or video content to potentially thousands of clients distributed across the Internet.

Because of the stringent timing and quality-of-service requirements, high-bandwidth demands, and the CPU and memory intensive characteristics of these applications, the performance of the server hardware is critical for efficient performance and delivery of high quality multimedia contents.

In this report, we present an experimental study of the memory performance of streaming media servers. We obtained low-level details of server performance for a number of configurations. We obtained measurements for cache misses and page faults using two leading streaming media servers: Darwin streaming media server and Windows media server. The measurements were obtained for varying number of client requests and two levels of encoding rates and stream distribution. We compare the throughput, CPU utilization, and cache and memory performance of two commercial streaming media servers: Apple's Darwin Streaming Server and Microsoft's Windows Media Server. Our measurements indicate that when the streaming servers are subjected to high number of client requests loads, the cache

misses and page faults become more frequent and performance is significantly affected. In addition, the quality of service (QoS) experienced by the client degrades, resulting in higher packet loss and lower frame rate.

The rest of this report is organized as follows. In section 2, we present background information on streaming media technology and concepts, while in section 3 we discuss some related work in the literature. We present our experimental work in section 4 and discuss the results in section 5. The conclusions and future direction of this research is outlined in section 6.

2. BACKGROUND AND MOTIVATION

Streaming media server performance evaluation requires understanding of the architecture of such a system. In this section, we first present six key areas of a streaming media architecture and then review three widely used streaming media servers. This discussion leads to the motivation for evaluating memory performance of streaming media servers.

2.1 Streaming Architecture

The streaming media technology can be broadly grouped into six key areas, namely: video compression, application-layer QoS control, continuous media distribution service, streaming servers, media synchronization mechanisms and protocols for streaming media. Each of these six areas is a basic building block for streaming media architectures [1].

Raw video and audio data are pre-compressed by appropriate compression algorithms and then saved in storage devices. Upon a client's request, a streaming server retrieves compressed video/audio data from storage devices and then the application-layer QoS control module adapts the video/audio bit-streams according to the network status and QoS requirements. After the adaptation, the transport protocols packetize the compressed bit-streams and transmit the video/audio packets to the Internet. Packets may be dropped or experience excessive delay inside the Internet due to congestion.

To improve the quality of video/audio transmission, continuous media distribution services (e.g., caching) are deployed in the Internet. For packets that are successfully delivered to the receiver, they first pass through the transport layers and are then processed by the application layer before being decoded at the video/audio decoder. To achieve synchronization between video and audio presentations, media synchronization mechanisms are required. We briefly describe these six areas as follows.

2.1.1 Video compression

Raw video/audio must be compressed before transmission to achieve efficiency. Video compression schemes can be classified into two categories: scalable and nonscalable video coding. Scalable video is capable of gracefully coping with the bandwidth fluctuations in the Internet [2], hence widely deployed for streaming over the Internet and organizations' intranets. Popular streaming protocols like Microsoft .ASF (active streaming format), Apple .MOV and Real Networks RM (real media) are the most widely deployed.

2.1.2 Application-layer QoS control

To cope with varying network conditions and different presentation quality requested by the users, various application-layer QoS control techniques have been proposed [3], [4], [5]. The application-layer techniques include congestion control and error control. Their respective functions are as follows. Congestion control is employed to prevent packet loss and reduce delay. Error control, on the other hand, is to improve video presentation quality in the presence of packet loss.

2.1.3 Continuous media distribution services

In order to provide quality multimedia presentations, adequate network support is crucial. This is because network support can reduce transport delay and packet loss ratio. Built on top of the Internet (IP protocol), continuous media distribution services are able to achieve QoS and efficiency for streaming video/audio over the best-effort Internet. Continuous media distribution services include network filtering, application-level multicast, and content replication [1].

2.1.4 Streaming servers

To offer high quality streaming services, streaming media servers are required to process multimedia data under timing constraints. A streaming server typically consists of three subsystems: a communicator (e.g., transport protocols), an operating system, and a storage system.

The operating system offers various services related to the essential resources, such as the CPU, main memory, storage, and all input and output devices. Since resources are limited, the server can only serve a limited number of clients with requested QoS. Therefore, resource management is required to manage resources so as to accommodate timing requirements.

2.1.5 Media synchronization mechanisms

Media synchronization refers to maintaining the temporal relationships within one data stream and between various media streams. It is a major feature that distinguishes multimedia applications from other traditional data applications. With media synchronization mechanisms, the application at the receiver side can present various media streams in the same way as they were originally captured.

2.1.6 Protocols for streaming media

Protocols are designed and standardized for communication between clients and streaming servers. Protocols for streaming media provide such services as network addressing, transport, and session control. According to their functionalities, the protocols can be classified into three categories: network-layer protocol such as Internet protocol (IP), transport protocol such as user datagram protocol (UDP), and session control protocol such as real-time streaming protocol (RTSP) [6].

It is important to distinguish between two modes in which video information can be distributed over the Internet, namely, live broadcasting and on-demand streaming. Below, we consider each of these models and the corresponding delivery mechanisms used by modern streaming media systems.

Distribution of Live Video

The source of live video information (such as any standard analog video recorder) is connected to the encoder. The encoding engine is responsible for capturing and digitizing the incoming analog video information, compressing it, and passing the resulting data down to the server. Alternatively, the server can receive such information from a Simulated Live Transfer Agent (SLTA), a software tool that reads pre-encoded information from an archive and sends it to a server as if it has just been encoded from a live source. In the simplest form, the server (or splitter) unicasts the encoded video to each of the clients. In this case, the parameters of the connection between server and each client can be estimated at the beginning of each session and can be systematically monitored during the broadcast. In the case where a network is equipped with multicast-enabled routers, the server needs to send only one multicast stream, which is automatically replicated to all subscribed clients on the network.

On Demand Distribution

One of the major differences between live broadcast and on demand distribution is that there is no direct connection between the encoder and the server in on demand delivery. Instead, a compressed video clip has to be recorded on disk first, and then the server will be able to use the resulting compressed file for distribution. However, server/client communication for delivering on demand content is essentially the same as unicast streaming of live content. Another difference between the two distribution schemes is that in on demand distribution, a user is allowed to rewind and/or fast forward the presentation, while only rewind may be allowed in live broadcast [7].

2.2 Commercial Streaming Servers

Poplar commercially available streaming media servers are: (1) Darwin streaming server/QuickTime streaming server (2) RealSystem server, and (3) Windows media server.

2.2.1 Darwin Streaming Server/QuickTime Streaming Server

This is a technology for delivering media over the Internet. It is developed by Apple Computers. DSS supports a variety of streaming protocol and its native streaming file

format is MOV. It runs on several platforms including Windows 2000, Linux, Solaris, FreeBSD and Mac OS X. It has features for both video on demand and live broadcast.

2.2.2 RealSystem Server

Real Networks developed the RealSystem server, which runs on several platforms including Solaris, Windows and Linux. It uses RSTP and RM streaming file format. It interoperates with Darwin streaming server and supports video on demand and live broadcast.

2.2.3 Windows Media Server

Windows media server is developed by Microsoft and is only supported on Windows platform. It supports only Microsoft streaming protocols (mms) and streaming file format (asf, wma and wmv). Windows media server supports video on demand and live broadcast using Windows media encoder.

There has been tremendous progress in microprocessor technology, which leads to high speed CPUs. Also, advances in memory and magnetic disk technology have lead to improvement in memory density and magnetic disk density much more than access and cycle times. Density of semiconductor DRAM increases by 60% per year, quadrupling in three years, but cycle time has improved very slowly, decreasing by about one-third in 10 years. In a similar fashion, magnetic disk density has been improving by about 50% per year, almost quadrupling in three years. Access time has improved by only one-third in 10 years [8]. It is obvious that memory and disk performance can limit the performance of a busy streaming media server that serves highly popular compressed audio/video contents.

3. RELATED WORK

Growing deployment and use of streaming media servers is also drawing the attention of researchers to this. Performance of a streaming server is a key factor contributing to the quality of the multimedia content for the end-users. Shenoy et al [9] highlighted some fundamental issues arising in multimedia server design. Technical challenges in design; such as storage and retrieval of multiresolution data, scalability and management were presented. Sohn et al [16] looked at the performance of a small-

scale VOD server. They conducted a measurement-based study in which they outlined the predictability of the real-time scheduler and the performance of the VOD server. Results of the performance measurements showed that the network protocol processing is a source of non-predictability. They found that high performance processor should be used to process the network protocol. However the performance of the storage system was not a problem to the VOD service.

A significant amount of work is reported in literature on the disk storage issue for streaming media servers. Due to large volumes of video and other multimedia files, storage and retrieval techniques play an important role in the performance of the server too. A storage hierarchy to design a low-cost cache for a movie on demand (MOD) server was proposed in [10]. The hierarchy consists of a disk, which stores the popular movies, and a small amount of RAM buffers which store only portions of the movies. They reported that due to low cost of disks, the cost of a MOD server based on the proposed architecture is substantially less than one in which the entire movie is loaded into RAM. Another multimedia architecture and data retrieval model for supporting simultaneously multiple clients requesting files of different playback rates is presented in [11]. The performance of the architecture was investigated using a circular SCAN disk scheduling policy in terms of the maximum number of concurrent video streams it can support.

Some studies of multimedia servers pay attention to I/O subsystems due to the high throughput demand of the servers. In fact, streaming media servers are often I/O bound. A study [12] focused on the design of an I/O subsystem for a continuous media server. They proposed several improved architectures based on an existing device: Intel i960RP? I/O processor, and evaluated their performance. They reported that utilization of the I/O processor solved the main memory bottleneck problem, but created a new bottleneck in i960RP? memory. I/O performance in multimedia servers has also been investigated using simulation [13]. Various I/O issues in multimedia systems have been discussed in [14], focusing on disk scheduling, SCSI bus contention and effect of buffer space on the performance of the real-time requests and aperiodic requests.

Rixner [15] proposed the Imagine architecture for streaming media processor, which delivers a peak performance of 20 billion floating-point operations per second. Imagine efficiently supports 48 arithmetic units with a three-tiered data bandwidth hierarchy. At the base of the hierarchy, the streaming memory system employs memory access scheduling to maximize the sustained bandwidth of external DRAM. At the center of the hierarchy, the global stream register file enables streams of data to be recirculated directly from one computation kernel to the next without returning data to memory. Also, local distributed register files that directly feed the arithmetic units enable temporary data to be stored locally so that it does not need to consume costly global register bandwidth. The bandwidth hierarchy enables Imagine to achieve up to 96% of the performance of a stream processor with infinite memory bandwidth from memory and the global register file.

There are several performance studies of multimedia servers in the literature. A timed Petri-Net model of distributed multimedia database architecture was reported in [17]. The model can handle both static and dynamic media, and can be used to analyze the transient performance of the database server as seen by a client workstation over a broadband network. Performance issues could also be considered based disk scheduling policy in terms of the maximum number of concurrent video streams that can be supported [11], and scheduling on the real-time performance guarantees provided by the server [18].

Our work focuses on performance issues relating to on-chip cache and memory of streaming servers. We employed measurement-based technique to study the effect of memory on performance of streaming servers while they are loaded by request from clients. Though several work on performance evaluation of streaming media servers are reported in the literature, no specific attention was paid to cache and memory issues.

4. MEASUREMENT-BASED EXPERIMENTS

For this performance evaluation study, we employed measurement-based techniques because of the availability of the systems to be evaluated.

4.1 Experimental Design

We conducted some initial experiments for our experimental design to determine the effect of factors and variation explained by each of the factors. Using $2^k r$ experimental design with replication, where k = 3 (number of client requests, encoding rate and stream distribution) and r = 2 (two replications), we computed the variation explained by each of these experimental factors. The number of client requests explains the highest variation with 62.29% of total variation. Encoding rate explained 19.33% while stream distribution explained only 4.94%. All interactions explain negligible variation while experimental error explained a significant 12.87%. High variation explained by experimental error could be attributed to random attributes in the load simulators, which make experiments not exactly repeatable.

4.2 Experimental Testbed

Our experimental testbed comprises of a server machine running the streaming servers and four client machines running load simulators. The setup consists of a closed-LAN with a 3Com 100Mbps switch. The streaming media servers ran on a PC with 166 MHz Pentium, 96MB RAM, and 100Mbps Ethernet NIC. The clients run on PCs with 166 MHz, 64MB RAM and 100Mbps NIC. The load simulator could generate a large number of client requests from a single client computer. Figure 1 depicts our experimental test bed.

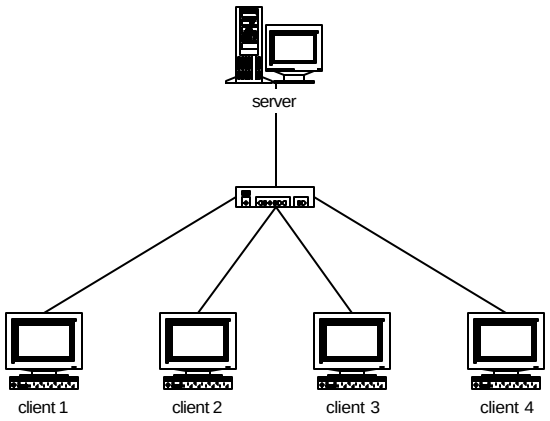


Fig. 1: Experimental test bed

4.3 Tools

We used software tools for simulating large number of clients and collecting system information in our experiments. The following tools were used for the experiments:

- Load simulators: a network-based load simulation and test tool that allows users to emulate multiple number of clients requesting streaming media
  - **Streaming load tool** – used as clients for Darwin streaming server
  - **Windows media load simulator** – for Windows media server

- Performance: a Windows 2000-based performance measurement tool for accessing CPU on-chip performance counters.

- Rabbit: a performance counters library for Intel/AMD processors and Linux [19].

- Ethereal: a software tool for capturing and analyzing network traffic. It has capability for measuring traffic flow through the network interface card. Ethereal runs on both Linux and Windows platforms.

- Intel Vtune performance analyzer: performance and profiling tool. Runs on Linux and Windows platform and provides interface for accessing on-chip performance counters.

- Other tools used were vmstat, netstat, and iostat

4.4 Metrics and Factors

For the performance evaluation of the servers, we used the following metrics:

- ?? Server throughput
- ?? CPU utilization
- ?? On-chip cache misses
- ?? Memory page faults

And the following experimental factors were verified for our experiments:

- ?? Number of client requests; at three levels (4, 100 and 400)
- ?? Encoding rates; at two levels (56k and 300k)
- ?? Stream distribution; at two levels (single and multiple streams)

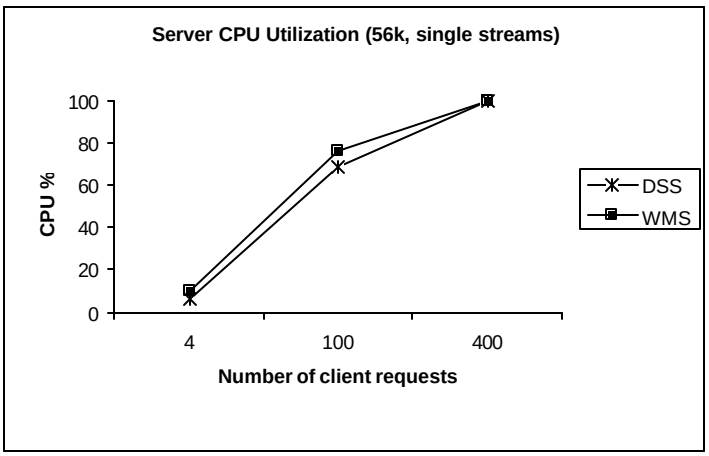
We setup a video on-demand scenario were the clients make request for stored video streams from the server. The measurements tools were used to collect throughput, CPU utilization, cache misses and page faults.

5. COMPARISON OF SERVERS

In this section, we analyze the results obtained from our experiments and discuss the comparative performance of the two servers. We compare the two servers, Darwin streaming server and Windows media server, in terms of CPU utilization, cache miss rate, page fault rate and throughput.

5.1 CPU utilization

Figure 2 shows the CPU utilization of the servers for three levels of the number of client requests (4, 100 and 400) at 56kbps encoding, single stream distribution and for 300kbps encoding with multiple streams distribution. In both cases, Darwin streaming server has lower CPU utilization compared to Windows media server. The CPU utilization also increases with the number of client requests.



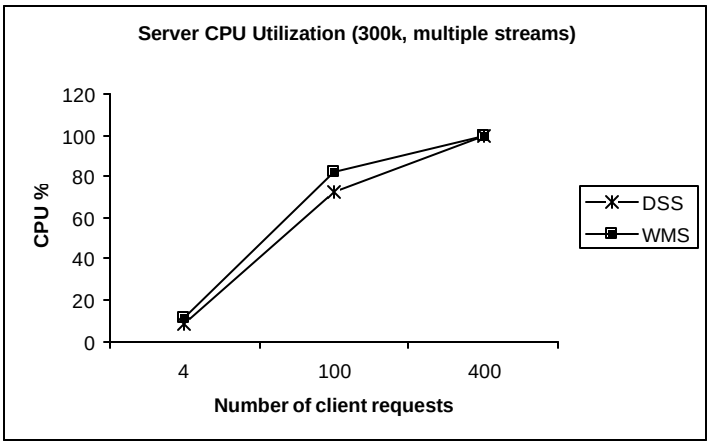
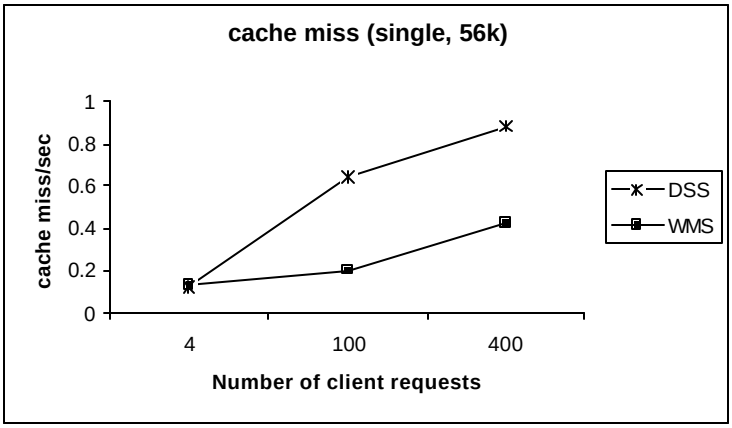
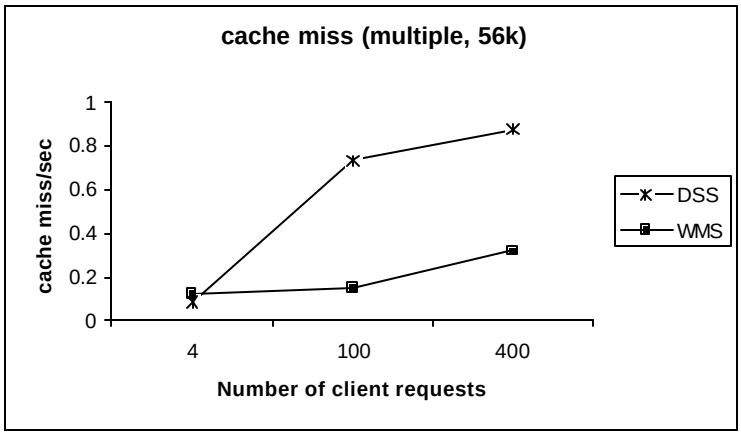


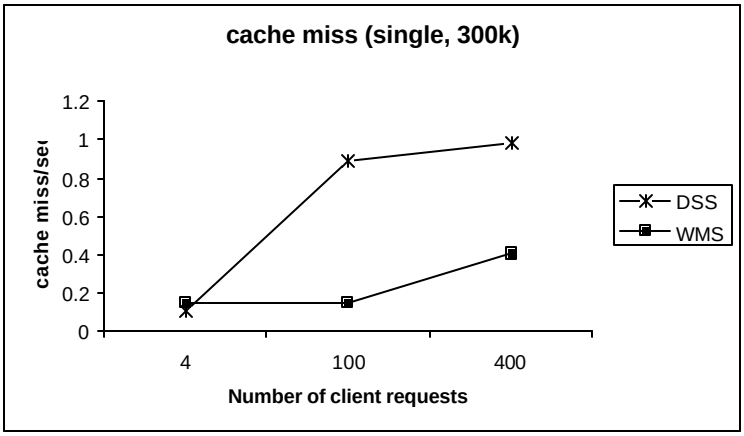
Fig. 2: CPU utilization

5.2 Cache miss

We compare the results of cache miss rate in Figure 3. At 56kbps encoding, Windows media server has lower cache miss rate at 100 and 400 number of client requests while for 300kbps encoding, Windows media server has a high cache miss for 400 client requests; much higher compared to Darwin streaming server. At 56kbps encoding, Windows media server has better cache performance, while at 300kbps encoding, Darwin streaming server is better.


cache miss (single, 56k)
cache miss/sec
1
0.8
0.6
0.4
0.2
0
4
100
400
Number of client requests
DSS
WMS


cache miss (multiple, 56k)
cache miss/sec
1
0.8
0.6
0.4
0.2
0
4
100
400
Number of client requests
DSS
WMS


cache miss (single, 300k)
1.2
1
0.8
0.6
0.4
0.2
0
4
100
400
Number of client requests
DSS
WMS

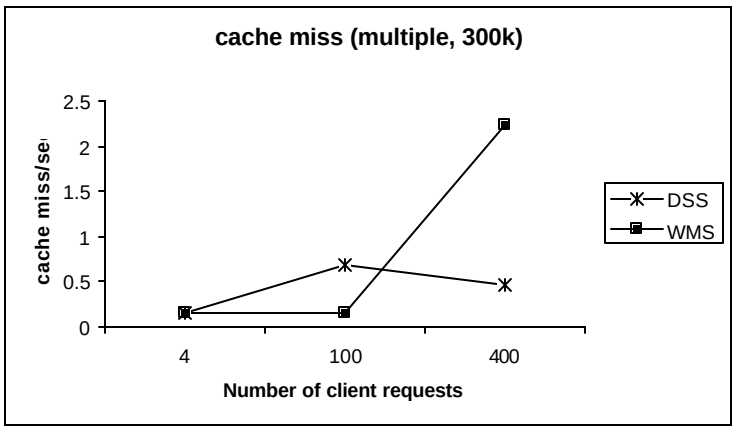
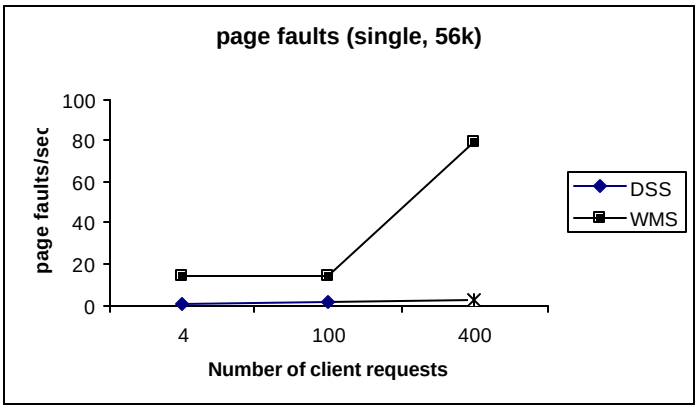
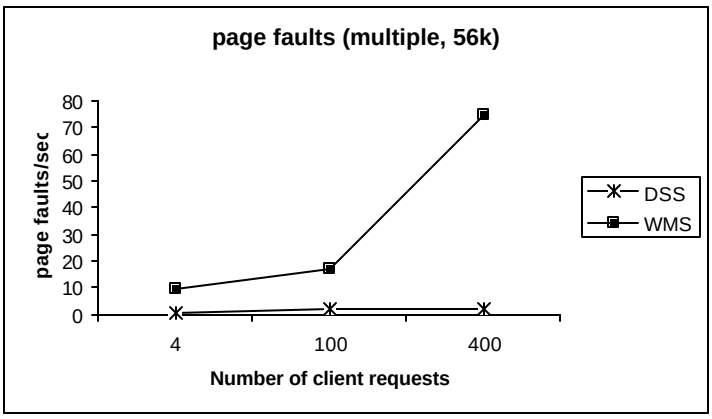


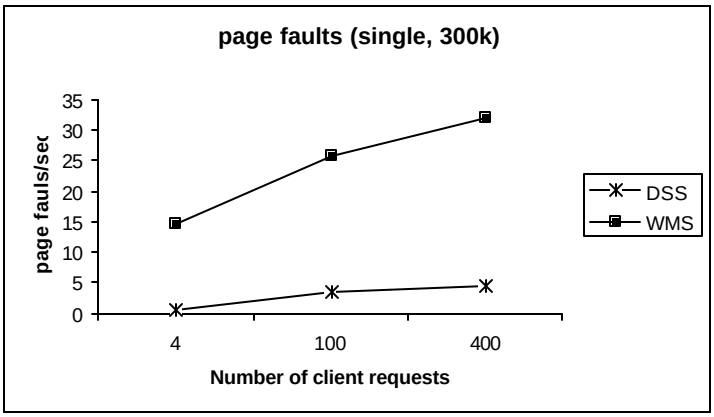
Fig. 3: Cache miss rate

5.3 Page faults

As shown in Figure 4, for both encoding rates: 56kbps and 300kbps, and stream distribution: single and multiple stream, Windows media server has much higher page fault rate compared to Darwin streaming server.







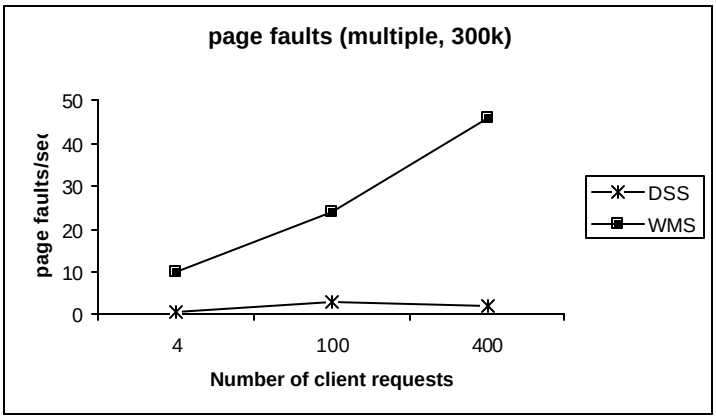


Fig. 4: Page fault rate

5.4 Throughput

We present the throughput comparison in Figure 5. At all levels of number of client requests, Windows media server has higher throughput compared to Darwin streaming server

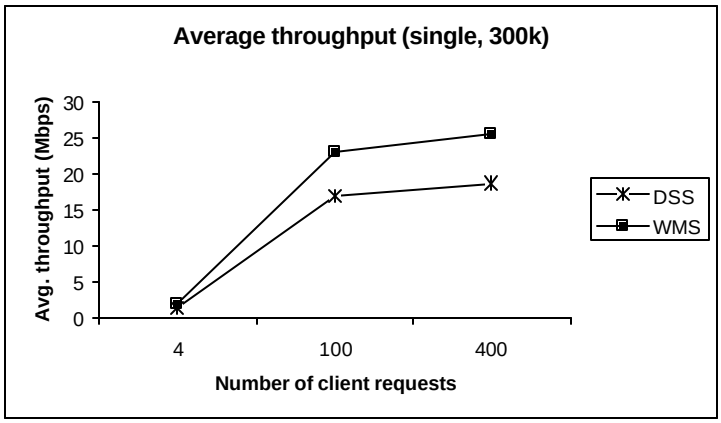


Fig. 5: Average throughput

6. CONCLUSION AND FUTURE DIRECTION

In this report, we have presented the results of measurement-based evaluation of the memory performance of streaming media servers. We conducted experiments on Darwin streaming server and Windows media server under identical workload conditions. Our measurements show that Darwin streaming server has less CPU demand but also less throughput compared to Windows media server. The large number of cache misses and page faults leads to significant wastage in CPU cycles and high memory latency, hence a bottleneck on performance.

Since it is obvious from this study that memory is a major bottleneck in the performance of streaming servers, a direction in future research work could be to alleviate this bottleneck. Streaming media servers could be designed to bypass the memory hierarchy by incorporating techniques such as memory-to-I/O transfer of data with poor spatial and temporal locality that leads to significant cache misses and page faults.

REFERENCES

[1] D. Wu et al, "Streaming Video over the Internet: Approaches and Directions," IEEE, Transactions on circuit and systems for video technology, Vol. 11, NO. 3, March 2001.

[2] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in Proc. ACM SIGCOMM '96, Aug. 1996, pp. 117-130.

[3] A. Eleftheriadis and D. Anastassiou, "Meeting arbitrary QoS constraints using dynamic rate shaping of coded digital video," in Proc. 5th Int. Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV'95), Apr. 1995, pp. 95–106.

[4] X. Wang and H. Schulzrinne, "Comparison of adaptive Internet multimedia applications," IEICE Trans. Commun., vol. E82-B, no. 6, pp. 806–818, June 1999.

[5] Q. Zhang, G. Wang, W. Zhu, and Y.-Q. Zhang, "Robust scalable video streaming over Internet with network-adaptive congestion control and unequal loss protection," in Proc. Packet Video Workshop, Kyongju, Korea, April 2001.

[6] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," Internet Engineering Task Force, RFC 2326, Apr. 1998.

[7] G. J. Conklin et al, "Video Coding for Streaming Media Delivery on The Internet," IEEE Transactions on Circuits and System for Video Technology, Vol. 11, March 2001.

[8] J. L. Hennessy and D. A Patterson, "Computer Architecture: A Quantitative Approach," Morgan Kaufmann Publishers, Inc., 1996.

[9] P. J. Shenoy, P. Goyal and H. Vin, "Issues in Multimedia Server Design," ACM Computing Surveys, Vol 27, No. 4, December 1995.

[10] B. Ozden, A. Biliris, R. Rastogi and A. Silberschatz, "A Disk-Based Storage Architecture for Movie on Demand Servers," Information Systems Vol. 20, No. 6, pp. 465, 1995.

[11] B. Sonah, M.R. Ito and G. Neufeld, "The Design and Performance of a Multimedia Server for High-Speed Networks", Proceedings of IEEE International Conference on Multimedia Computing and Systems, ICMCS 1995.

[12] M. Weeks, H. Batatia and R. Sotudeh, "Improved Multimedia Server I/O Subsystems," Euromicro98, 24th Conference Proceedings, Vasteras, Sweden. 1998.

[13] M. Weeks and C. Bailey, "Continuous Discrete-Event Simulation of a Continuous-Media Server I/O Subsystems," Euromicro 2000, Workshop on Multimedia and Telecommunications, Maastricht, Netherlands, September 2000.

[14] A. L. Reddy and J. Wyllie, "I/O Issues in a Multimedia System," IEEE Computer, vol. 27, no. 3, pp. 69--74, Mar. 1994.

[15] S. Rixner, "A Bandwidth-efficient Architecture for a Streaming Media Processor," PhD Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, February 2001.

[16] J. M. Sohn, G. Y. Kim and T. G. Kim, "Performance Measurements of a Small-Scale VOD Server Based on the UNIX," The Third IEEE Symposium on Computers and Communications ISCC'98 Athens, Greece June 1998.

[17] T. Yeap and A. Karmouch, "Performance Evaluation of a Distributed Multimedia Database System Over a Broadband Network,"

[18] R. Tewari, R. Mukherjee, D. Dias and H. Vin, "Design and Performance Tradeoffs in Clustered Video Servers," International Conference on Multimedia Computing and Systems, Hiroshima, June, 1996.

[19] D. Heller, "Rabbit: A Performance Counters Library for Intel/AMD Processors and Linux," Scalable Computing Laboratory, Ames Laboratory, Iowa State University.