A Measurement Based Memory Performance Evaluation of High Throughput Servers

Garba Isa Yau and Abdul Waheed
E-mail: {dygarba, awaheed}@ccse.kfupm.edu.sa
Computer Engineering Department
King Fahd University of Petroleum and Minerals
P.O. Box 909 KFUPM
Dhahran 31261, Saudi Arabia
Phone +966-3-860-1489 Fax +966-3-860-2440

Abstract

Performance of high throughput servers largely depends on memory issues like on-chip cache, main and virtual memory, and disk. According to Moore's law, processor speed has been roughly doubling every eighteen months while memory and disk only get faster at about 10% per year. Bottleneck in server performance has shifted from processors to memory and disk. Memory hierarchy performance is a limiting factor in the performance of high throughput servers. In this work, we are conducting measurement-based performance study of key high throughput servers namely: streaming media servers and web servers. Our aim is to identify where on-chip cache and memory becomes bottleneck in the performance of this high throughput servers. We carry out measurement based study of memory performance of key commercial high throughput servers: Darwin streaming server and Windows media server, Apache web server and Microsoft Internet information server (IIS). Our study shows that under stringent clients' demand on this servers, performance degrades due to high cache misses and page faults, rendering the memory hierarchy less efficient, and significant bottleneck itself.

Keywords: streaming media, encoding rate, cache miss, page fault, video on demand, live video broadcast, web cache.

1 INTRODUCTION

Growing demands of the Internet requires high performance servers for such applications like World Wide Web and real-time multimedia applications. Web servers, streaming servers and proxy servers are essentially high throughput servers that normally serve a large number of clients. The continuous explosion of the Internet further makes demand on these servers very high and stringent; hence the performance of these servers must meet up with the demands in today's Internet applications and large number of clients. Due to the growing interest in the development of World Wide Web applications

targeted at commerce and other critical applications, the performance of the Web server is a key aspect in the design of a Web system. The objective is to fulfill the emerging requirement of giving access to an ever increasing amount of requests for information represented with text, image, audio and video coming from a large number of clients distributed across the Internet. Similarly, 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 as the concept streaming media has now been seamlessly integrated with the World Wide Web. Websites now provide link to multmedia contents which will be subsequently served by streaming media servers. This growth is facilitated by progress in the development of various core technologies driving the server software and hardware.

Efficient performance of these servers is a key to the success of any site intending to serve its contents to a large clients with varying demands. Performance of these servers largely depends on hardware and software, especially now that the servers are deployed on high speed networks that would not be bottleneck. Once CPU was a bottleneck, but tremendous progress in technology that leads to explosive growth in CPU speed and capabilities has shifted this bottleneck else where. Memory, I/O and network are now potential bottlenecks on performance of servers that serve a large number of clients especially those with stringent timing requirements like video-on-demand and real-time audio/video broadcast. Memory subsystem in particular is a key factor in performance degradation as memory technology remains far lagging behind processors. This speed disparity leads to the memory hierarchy in computer architecture which depends on data locality of reference. The question we raise is: to which extent is the memory hierarchy, with on-chip cache and main memory beneficial to Internet high throughput servers?

We present an experimental study of the memory performance of some high throughput servers. Our empirical studies focus on two streaming media servers: Darwin streaming server and Windows media server, and also two Web servers: Apache web server and Microsoft Internet Information Server. We collected performance related measurements and low level measurements showing on-chip cache and memory performance, and CPU utilization.

The rest of this paper is organized as follows. In section 2 we outline some background material which forms the bases of our motivation while in section 3 we review some related work on streaming media servers and Web servers. Experimental setup is outlined in section 4 and section 5 discusses the experimental results. We outline our observation on impact of operating system on the servers in section 6. We talk about the conclusion and future work in section 7.

2 BACKGROUND AND MOTIVATION

In this section we shall provide a brief overview of three high throughput servers that are highly used in the Internet, viz: streaming media servers, web servers and web proxy servers. This servers are now the de factor content provision servers on the Internet and their performance is highly significant for efficient

content distribution on the Internet.

2.1 Streaming Media Servers

Streaming media is a technology that makes it possible for a client to experience a multimedia presentation on the fly, that is allowing a client to playback the multimedia content as it is downloaded. The media content is not stored on the disk, but rather played as it is streamed to the client. Streaming media servers accept client requests for specific media objects and stream them from their storage location in the case of on-demand contents or directly from the encoder in the case of live media broadcast. To ensure quality streaming, the servers must process multimedia data under timing constraints that will guarantee acceptable QoS (quality of service).

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. Wu et al [1] presents a detail discussion on these areas.

It is important to mention that two modes are used for streaming media distribution over the Internet, namely, live broadcasting and on-demand streaming. In live broadcasting, users watch live events while on-demand streaming provides access for already stored media.

2.2 Web Servers

The purpose of a Web server is to provide documents to Web clients as they request for these documents. A Web server operates in the following way. The server listens on a designated port (usually port 80) for a request from a Web client to establish a TCP connection. Once a TCP connection has been opened and the client has made its request, the server must respond to that request. Web servers and clients communicate based on the application layer protocol HTTP (Hypertext Transfer Protocol).

The steps required to process a typical request (i.e. a static GET) are outlined in [2] as follows.

- Step 1: is called to get a new connection.
- Step 2: is called to determine the remote host (for logging purposes).
- Step 3: is called on the socket to get the HTTP request.
- Step 4: is called to disable the Nagle algorithm.
- Step 5: is called to determine the time of the request.
- Step 6: the request is parsed, identifying the appropriate file to send.
- Step 7: is called to obtain the file status and size.
- Step 8 is called on the requested file.

- Step 9: is called on the file descriptor to read the file into the server.
- Step 10: is called on the socket to send the HTTP header to the client.
- Step 11: is called on the socket to send the file to the client.
- Step 12: is called to close the file.
- Step 13: is called to shutdown the connection.
- Step 14: is called on the log file descriptor to log the request.

Web servers are also used for streaming media. Web servers use the HTTP protocol and since the HTTP protocol was originally designed for serving static documents, it was not particularly suited for real-time streaming. Lack of QoS guarantee in the design of HTTP protocol and the use of TCP can cause substantial fluctuation in the delivery times of media, leading to unacceptable media quality at the client end.

Essentially, the performance of proxy servers is very similar to that of Web servers especially if the document requested is in the web cache. When a client makes a request for a web object to a web proxy server, the server examines its web cache and if it finds it in the cache, it returns the object to the client without contacting the origin Web server. This is referred to as cache hit. However, if the object is not available in its cache, it retrieves the object from the origin server, keeps a copy and serves it to the client. When the object is not in the web cache, we say there is a cache miss.

2.3 Memory Hierarchy

There has been tremendous progress in microprocessor technology that leads to high speed CPUs. Also, advances in memory and magnetic disk technology have significantly led 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 [3]. To alleviate the problem of widening performance gap between processor and main memory, computer architecture now incorporates the memory hierarchy system in which data caches are now widely used for hiding memory latency. Memory hierarchy is based on the concept of principle of locality of reference - temporal and spatial. Temporal locality states that recently accessed data are likely to be accessed in the near future while spatial locality says that data whose addresses are near one another tend to be referenced close together in time [3]. Although caches go a long way in improving performance for applications with small working data sets and large amounts of spatial and temporal locality, allowing a small cache to provide enough storage to hold most of the useful data required by the program at any given time, some programs fail to use them effectively because of poor data locality and very large

working data sets that cannot be accommodated in the cache [3]. The consequence is significant lost in performance due to cache misses and processor stall cycles resulting from the misses.

3 RELATED WORK

In this section, we review related work on design and performance of streaming media servers and Web servers.

3.1 Streaming Media Servers

Growing deployment and use of streaming media servers is already drawing the attention of researchers. Performance of a streaming server is a key factor contributing to the quality of the multimedia content for the end-users. Shenoy et al [4] highlighted some fundamental issues arising in multimedia server design. Technical design challenges such as storage and retrieval of multiresolution data, scalability and management were presented. Sohn et al [5] 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. A significant amount of work is reported in the literature on the disk storage performance 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 [6]. The hierarchy consists of a disk that stores the popular movies and a small amount of RAM buffers which store only portions of the movies. 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.

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 by Batatia et al [7] focuses on the design of an I/O subsystem for a continuous media server. They propose several improved architectures based on an existing device: Intel i960RP I/O processor, and evaluate their performance. They report 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 [8]. Various I/O issues in multimedia systems have been discussed in [9].

Our literature search reveals that researchers did not pay much attention to the memory performance of the streaming server itself. An exception is the study conducted by Sohoni et al [10] which rather study the memory system performance of multimedia applications at the client end. Despite poor temporal locality, at the client's application end, high cache hit ratio is reported which is attributed to factors like block partitioning algorithm employed, significant data reuse and excellent spatial locality of continuous multimedia data. However, this study does not address server performance and its high throughput demand.

3.2 Web Servers

Web servers are obviously the key servers in the Internet today since Web traffic accounts for substantial part of the traffic on the Internet. There has been tremendous amount of research activities on Web servers ranging from performance studies, workload characterization [11] and even security issues [12, 13]. But just like the case of streaming media servers, there is no significant work on the performance of processor on-chip cache and memory subsystem. Iyengar et al [14] performance study focused on a method of improving the performance of the Web server in the situation that the CPU becomes the limiting resource. In [15] a new Web server mechanism was reported. Performance results were presented showing the scalability and efficiency of the proposed design. This is an attempt to improve the performance of the Web server itself and not the underlaying hardware.

Performance characteristics were also studied based on comparative studies using Web server benchmarking tools. [16] conducted measurement based performance study of Apache and Microsoft Internet Information server. Their study focused on comparative performance on same hardware, but no attention was paid on the impact of the underlaying hardware. Trecordi and Verger [17] studied the main component affecting the performance and scalability of Web servers. They take into account the impacts of the server software architecture, operating system and the underlaying server hardware. They reported numerical results that reveals that the performance and scalability of WWW servers heavily depend on a lot of parameters that should be properly tuned. Though this study discusses the processor on-chip cache and virtual memory system, no measurement on any metric related to the cache and virtual memory was reported.

4 EXPERIMENTAL SETUP AND TEST BED

We conduct measurement based performance evaluation of streaming media servers and Web servers. In all the cases, our primary focus is the on-chip cache and memory behavior and the impact on throughput. In this section we describe our hardware, setup, tools, metrics and factors, and experiments.

4.1 Hardware

For all the experiments, our test bed comprises of a dual boot server machines, booting either Red Hat Linux 7.2 (kernel 2.4.10) or Windows 2000 server. We also have dual boot client machines that run same OS as the server machine. The setup consists of a closed-LAN with a Cisco 1 Gbps multilayer switch (catalyst 3550). Our server hardware comprises of Pentium IV 2.0 GHz, 256 MB SDRAM, single 40 GB EIDE hard drive (Western Digital WD400) and 3Com 1 Gbps Ethernet NIC. Each of the client machine consists of Pentium III 300 MHz, 96 MB RAM and 100Mbps NIC.

4.2 Streaming Media Servers

The experimental setup is a LAN in which clients submit request for streaming media objects to the server. The streaming server returns the object to the client.

4.2.1 Metrics and factors

Our metrics of interest are:

- L1 cache miss
- L2 cache miss
- page fault rate
- server aggregate throughput
- server cpu utilization

To observe our metrics, we vary the following factors:

- number of streams (streaming clients)
- media encoding rate (56kbps and 300kbps)
- stream distribution (unique or multiple media)

We setup a video-on-demand scenario where clients request for stored compressed video streams from the server.

4.2.2 Tools

We collect measurements for our metrics using a number of software tools that run on the server machine in a non-intrusive way. Some tools run on both platforms (Windows and Linux) while others run only on one platform. However, for platform specific tools, we ensure that such tools exhibit very similar overhead (generally minimally intrusive) on the operating system in question. The following tools were used:

- Streaming Load Tool - for Windows media server we use microsoft streaming load simulator while for Darwin streaming media server, we use streaming load simulator. Both simulators operate in similar manner by making requests for media object through launching a large number of clients.
- Intel VTune performance analyzer (6.1): performance and profiling tool that we use to access CPU on-chip performance counters. It is available for both Windows and Linux
- Windows 2000 'performance': a Windows platform performance tool.

- Netstat: a tool for measuring bandwidth and observing connection status. It is available on both Windows and Linux platforms.
- Linux tools: vmstat, iostat and sar.

4.3 Web Servers

For our experiments on Web servers, only the server OS is changed to accomodate the appropriate Web server. The clients and webmaster all run on Linux, hence, we only needed to dual boot the server machine. Our experimental testbed comprises of the hardware we described earlier. We used five client machines and a Webmaster machine for control of clients using Webstone benchmarking tool [18]. Webstone simulates a large number of clients accessing web documents. Webstone considers the Web server as a black-box, hence to obtain low level measurement on the server itself, we run our other tools on the server machine. With this, we were able to collect memory related measurements and CPU utilization on the server.

In our experiments on Web servers, we only consider document transfers. We did not include requests that require processing at the server side like CGI (common gateway interface), Microsoft ASP (active server page) or any server-side processing.

4.3.1 Software tools

- WebStone: a configurable benchmark tool that allows performance measurement of Web servers. WebStone is originally developed by Silicon Graphics. WebStone 2.5 is Mindcraft's enhancement to WebStone 2.0.1 to improve reliability and portability as well as to make tests more reproducible. WebStone creates a load on a Web server by simulating the activity of multiple clients, which are called Web clients and which can be thought of as users, Web browsers, or other software that retrieves files from a Web server. In other to create large loads on a Web server, WebStone is able to distribute Web clients among client computers. The Webmaster is the program that controls all of the testing done by WebStone. With WebStone, we can measure average and maximum connect time (delay), average connection rate, average and maximum response time and data throughput rate.
- Intel VTune Performance Analyzer
- Platform specific tools: vmstat, sar, iostat and Windows 2000 'performance'.

4.3.2 Metrics and Factors

We have the following metrics:

- L1 cache miss

- L2 cache miss
- Page fault rate
- Aggregate server throughput
- Number of transactions/sec (connection rate)
- Average latency (delay)
- CPU utilization

We observe our metrics while varying the following factors

- Number of Web clients
- Web document size

5 RESULTS AND DISCUSSIONS

In this section, we present the results of our measurement based performance study on two streaming media servers and two Web servers. For the sake of clarity of these discussions, we present the results on streaming servers in a different section from the Web servers. We report measurements obtained using tools described in previous sections.

5.1 Streaming Media Servers

5.1.1 Cache Performance

Figures 1 and 2 show the L1 cache behavior under different configurations of clients, encoding rate and stream distribution. Measurements for both servers are reported. both figures show increase in cache misses as the number of clients increases. Thought not to a large extend, the stream distribution and encoding rate also affect the miss rate. We observe the worse case cache misses in both L1 and L2 when there is a large number of clients requesting multiple streams at 300kbps encoding rate. For this case, we started observing clients being refused connection by the server, which eventually makes the client to time out. We show L2 cache behavior for 300kbps encoding in Figure 3. For all these cases, Windows media server exhibits lower L1 and L2 cache misses. It becomes obvious that as a result of large working data set and lack of data reuse by the server, thrashing occurs in the CPU cache which renders the cache ineffective for this type of application.

5.1.2 Memory Performance

We consider main memory performance in terms of page fault rate. Any data referenced in the memory that is not available would have to be fetched from the disk, an expensive and slow process. Disk access

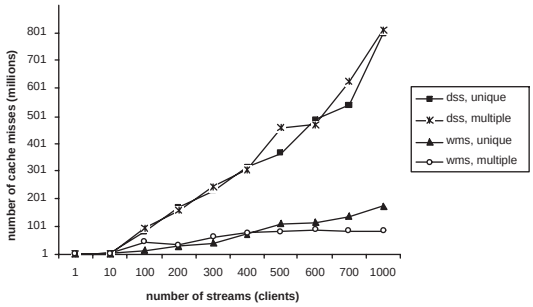

Figure 1: L1 cache misses for 56kbps encoding rate

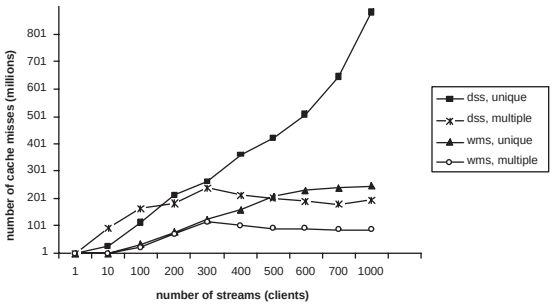

Figure 2: L1 cache misses for 300kbps encoding rate

is very slow especially if there are lots of disk references within a short duration. This is what we observe when we have clients requesting different media files. Figure 4 shows the page fault rate for clients requesting multiple streams at 300kbps. For cases where a unique stream is requested, this can be served from the memory and we observe a fairly constant page fault rate, which indicates low disk activity. However, as the clients request multiple streams, the page fault rate steadily increases with the number of clients. This is due to larger volume of data that have to be fetched from the disk into the main memory as the number of requesting streams grow larger.

The consequence of this high page fault rate is clients' timeout. It is intuitive that as more disk activity is involved, the server responds to clients' request very slowly leading to some of the clients to timeout after waiting for response for a long period of time .

5.1.3 Throughput and CPU Utilization

We show the throughput for 300kbps encoding rate and the corresponding cpu utilization. In Figure 5, the throughput for unique streams increases with the number of clients for both Darwin streaming server and Windows media server. However, for multiple streams, the throughput hardly increases beyond 200 streams. In fact, at 400 clients, we started observing clients timeout in Windows media server while we observe timeout in Darwin streaming server at 1000 streams. In all these cases, Windows media server has higher throughput. The corresponding CPU utilization for this configuration is shown in Figure 6.

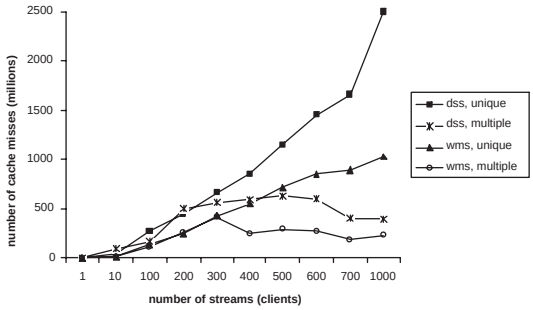

Figure 3: L2 cache misses for 300kbps encoding rate

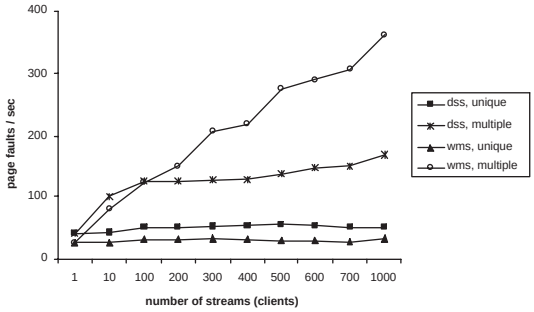

Figure 4: Page fault rate for 300kbps encoding rate

There is increase in CPU utilization as the number of clients increases. For unique stream access, the CPU utilization increases all the way up to 1000 streams. CPU utilization begins to fall when the number of clients timing out increases, thereby reducing the effective number of clients requesting media streams. Although this appears strange in the case of Darwin streaming server in which clients timeout begins at 1000 multiple streams, we still attribute the drop in cpu utilization after 300 multiple streams to low volume of data served to the clients.

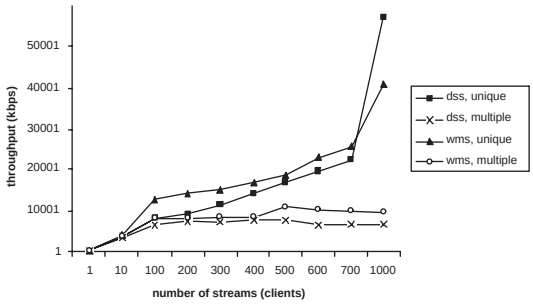

Figure 5: Server throughput for 300kbps encoding rate

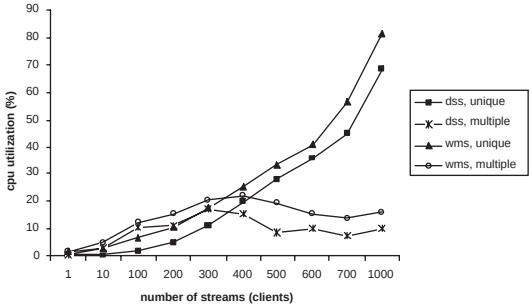

Figure 6: Server CPU utilization for 300kbps encoding rate

5.2 Web Servers

In this section, we discuss the measurement based performance study results we obtain for Apache and IIS servers.

5.2.1 Web Transactions

To make our discussion clearer, we would start with looking at the relation between web document size and number of transactions. The size of the document requested by a web client is highly significant on the performance of the Web server in terms of both hardware and software. Looking at Figure 7 we see that as the document size increases, the number of transactions become smaller. For a large document size, the server and client must maintain a connection for a longer time to transfer the file to the client, hence at any instance, the effective number of clients hooked to the server becomes significantly low, thereby leading to the low activity and high latency for waiting clients as we would show when we discuss latency. Generally, the two servers perform poorly for very small web documents. This observation is further supported by another study on Web servers by Hu et al [15].

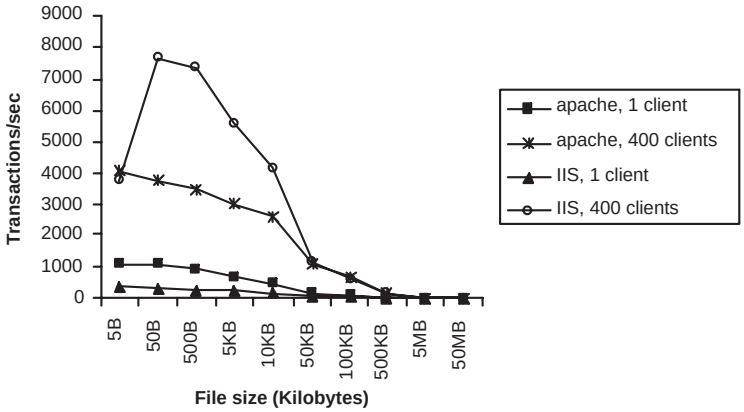

Figure 7: Web server transactions per seconds

5.2.2 Cache Performance

As shown in Figures 8 and 9, both L1 and L2 caches perform poorly for small documents. Though this is surprising, but as we mention the effect of document size on number of transactions; if the document size is small, more connections are established and released within a short time. This creates more activity in the processor cache as it even shows in the CPU utilization. The CPU utilization of the server is shown in Figure 13. Apache performs worst in terms of cache misses. Apache is a process-based server, forking several processes that serially accept new connections. Although Apache server tries to minimize the overhead of forking new processes by pre-forking a pool of processes at initialization, however, during heavy loads, the server resorts to forking a new process for every request. This highly involves the CPU and leads to excessive cache activity leading to high cache misses. Generally, the two servers have poor cache performance for small documents.

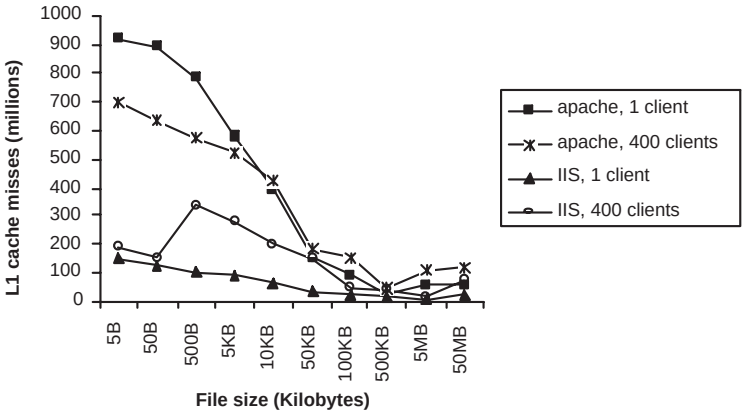

Figure 8: Web server L1 cache misses

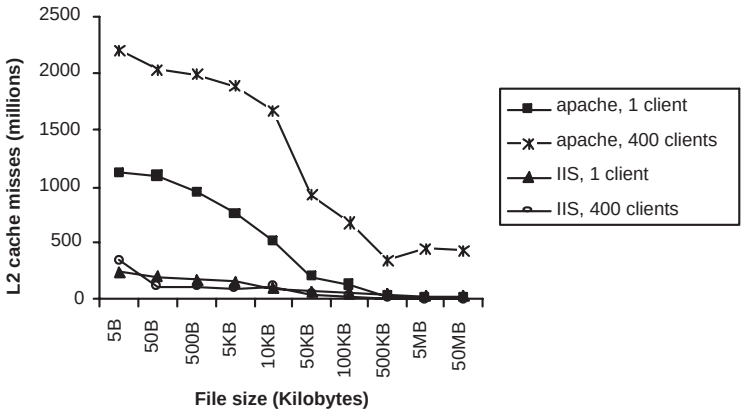

Figure 9: Web server L2 cache misses

5.2.3 Memory Performance

We observe high page fault rate when the document size is large. This is quite obvious as large documents must be continuously fetched from the disk to the memory and to the network. For a small document, this might be served from the memory since the entire file can reside in the memory. The page fault rate is shown in Figure 10.

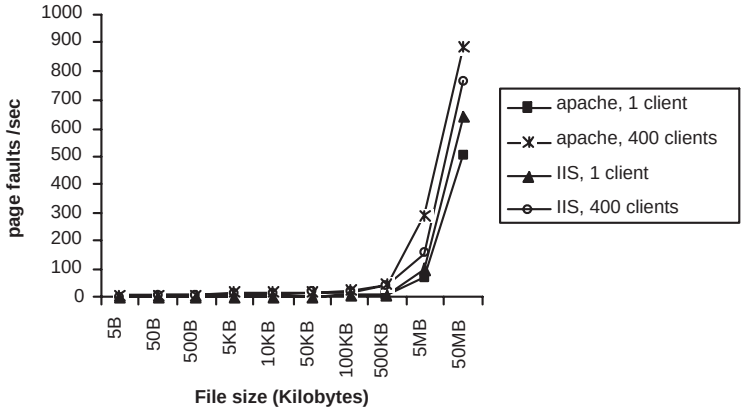

Figure 10: Web server page fault rate

5.2.4 Other Metrics

Other metrics are latency, throughput and CPU utilization. We observe the peak throughput at 500KB and 5MB document size. This is shown in Figure 11. For both single client and 400 clients, Apache server delivers more throughput. Generally, throughput scales with document size. For small document

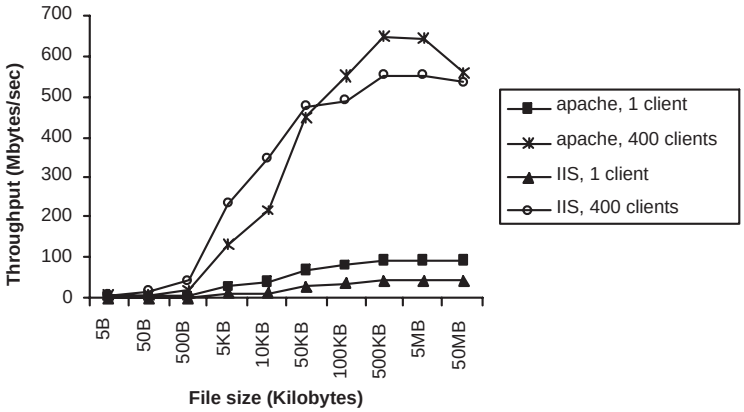

Figure 11: Web server throughput

size, clients generally experience low latency. If the document is large, 5MB and 50MB, the latency becomes too high and clients might even timeout. It is not surprising as it takes more time to deliver

the large file. Relation of document size to latency is shown in Figure 12. We also show CPU utilization in Figure 13. It is easy to saturate the server (100% CPU utilization) when the requested document is small. As we explained earlier, when the requested document is small, the number of transactions per second (connection rate) becomes high, more connections are setup and torn, hence more CPU activity.

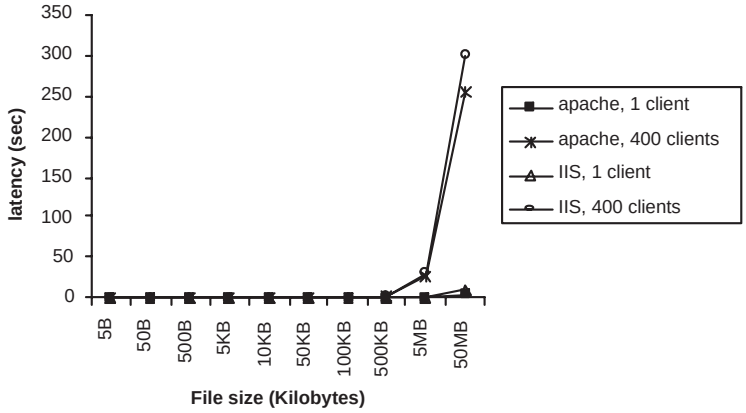

Figure 12: Web server latency

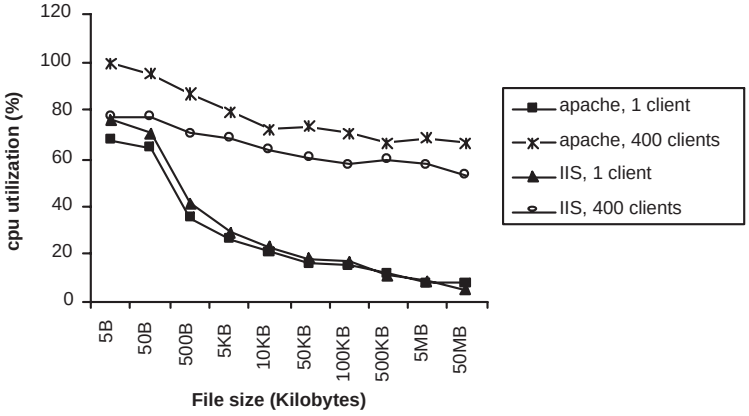

Figure 13: Web server CPU utilization

6 OPERATING SYSTEM IMPACT ON PERFORMANCE

Though it is not our goal to benchmark the performance of the different servers in our study. However, we suspect that since the servers were run on different platforms, there might be differences in memory behavior due to the operating system. To observe this effect, we made a quick memory subsystem assessment. ECT (extended copy transfer) memperf [19] is a method to characterize the performance of memory systems. It captures two aspects of the memory hierarchy: its behavior with temporal locality by varying the working set size (block size) and the spatial locality by varying the access pattern (strides).

The calculated value is the transfer bandwidth (for a large amount of data). We present the extended copy transfer characterization for load sum test. The load test measures the memory load performance for all the block-sizes and access patterns. Figure 14 shows our memperf microbenchmark result for Linux and Windows running on the same hardware. Both operating systems show similar memory performance, with the memory bandwidth decreasing as the block size increases; that is not fitting into cache. The worst case is when the block size is beyond 512kB which is the size of the level 2 cache. We run test only for stride = 1. Based on these results, we conclude that difference in memory performance by the individual servers is inherent in the servers themselves and not their host operating system.

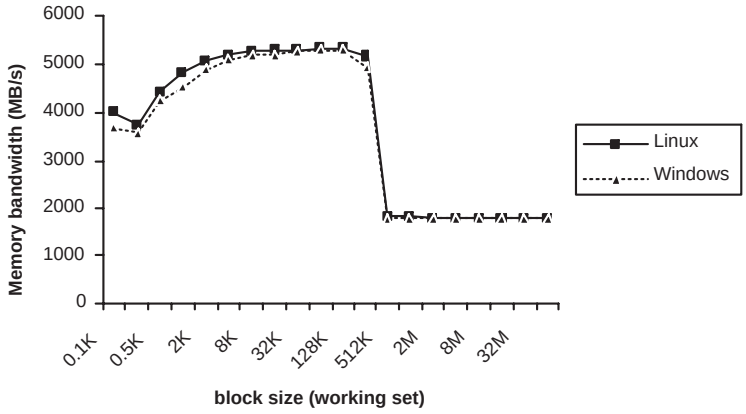

Figure 14: Extended copy transfer characterization (stride = 1)

7 CONCLUSION AND FUTURE WORK

In this paper, we have presented the measurement-based evaluation of memory performance of two streaming media servers and two Web servers. We carry out measurement based performance study to observe memory performance of these Internet high throughput servers and specifically identify where the on-chip cache and memory subsystem become bottleneck. For streaming media servers, our measurements show that cache and memory subsystem performs worst when there is a large number of clients requesting multiple media streams at 300kbps encoding rate. Both servers exhibit similar characteristics in cache/memory performance under identical workload. 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.

Similarly, for the Web servers, we observe that on-chip cache performs worst when the document is small. Small documents might fit into cache, but the high number of connection setup and tear-off significantly involves the processor which leads to high number of cache misses and high CPU utilization. On the other hand, large document size leads to higher page fault rates since the requested file must be continuously retrieved from the disk until it is all served to the requesting client.

Since it is obvious from this study that memory could be a major bottleneck in the performance of Internet high throughput servers, we are looking into ways to alleviate this bottleneck. We are particularly interested in improving the performance of streaming media servers. Streaming media servers could be designed to take advantage of the excellent spatial locality in continuous media by incorporating techniques like lookahead prefetching and demand prefetching. Such techniques can improve cache hit rate and reduce page fault rates. Our work continues to explore these issues.

Acknowledgement The authors would like to thank King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia for supporting this research effort.

References

[1] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, "Streaming video over the internet: Approaches and directions," *IEEE, Transactions on circuit and systems for video technology*, vol. 11, pp. 282–300, March 2001.

[2] E. Nahum, T. Barzilai, and D. D. Kandlur, "Performance issues in www servers," *IEEE/ACM Transactions on Networking*, vol. 10, February 2002.

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

[4] P. J. Shenoy, P. Goyal, and H. Vin, "Issues in multimedia server design," *ACM Computing Surveys*, vol. 27, December 1995.

[5] 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*, June 1998.

[6] 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, p. 465, 1995.

[7] M. Weeks, H. Batatia, and R. Sotudeh, "Improved multimedia server i/o subsystems," *Euromicro98, 24th Conference Proceedings*, 1998.

[8] M. Weeks and C. Bailey, "Continuous discrete-event simulation of a continuous-media server i/o subsystems," *Euromicro 2000, Workshop on Multimedia and Telecommunications*, September 2000.

[9] A. L. Reddy and J. Wyllie, "Io issues in a multimedia system," *IEEE Computer*, vol. 27, pp. 69–74, March 1994.

[10] S. Sohoni, R. Min, Z. Xu, and Y. Hu, "A study of memory system performance of multimedia applications," *Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 206 – 215, 2001.

[11] M. F. Arrit and C. L. William, "Web server workload characterization: The search for invariants," *ACM SIGMETRICS 96*, 1996.

[12] K. Krishna and M. Prasant, "Architectural impact of secure socket layer on internet servers," *International conference on computer design (ICCD)*, September 2000.

[13] A. Goldberg, R. Buff, and A. Schmitt, "Secure web server performance dramatically improved by caching ssl sessions keys," *Workshop on Internet Server Performance, SIGMETRICS '98*, June 1998.

[14] A. Iyengar, E. MacNair, and T. Nguyen, "An analysis of web server performance," *In Proceedings of the IEEE 1997 Global Telecommunications Conference (GLOBECOM '97)*, November 1997.

[15] J. C. Hu, I. Pyarali, and D. C. Schmidt, "Measuring the impact of event dispatching and concurrency models on web server performance over high-speed networks," *In Proceedings of GLOBECOM '97*, 1997.

[16] A. O. Sala-Alada and A. Waheed, "Performance comparison of apache and microsoft iis web server," *In Proceedings of International Arab Conference on Information Technology*, December 2002.

[17] V. Trecordi and A. Verga, "An experimental study on the performance of www servers," *Global Telecommunications Conference, GLOBECOM '96*, pp. 22–27, November 1996.

[18] G. Trent and M. Sake, "Webstone: The first generation in http server benchmarking." Available: http://www.sgi.com/Products/Web-FORCE/WebStone.

[19] C. Kurmann and T. Stricker, "Characterizing memory system performance for local and remote accesses in high end smps, low end smps and clusters of smps," *7th Workshop on Scalable Memory Multiprocessors held in conjuction with the 25th Annual International Symposium on Computer Architecture ISCA98*, June 1998.

Authors biography

Garba Isa Yau received B.Eng (Honours) electrical and electronic engineering from Abubakar Tafawa Balewa University Bauchi, Nigeria in September, 1998. He joined department of computer engineering, King Fahd University of Petroleum and Minerals as research assistant in January 2001. Currently pursuing Masters Degree in computer engineering. His research interests focus on systems performance analysis and evaluation, computer networks and security, and mobile computing. Garba is student member of IEEE.

Abdul Waheed is an assistant professor in Computer Engineering Depatrment at KFUPM. Before joining COE, he was working at Inktomi Corporation in Foster City, California, USA as a performance engineer in network products division. He was a research staff member at NASA Ames Research Center, Moffett Field, California, USA from May 1997 until July 2000. He held a summer position in Concurrent Computing Division at Hewlett-Packard Research Laboratories in Palo Alto, California, USA in 1994. He received the BSc degree with honors in Electrical Engineering from University of Engineering and Technology, Lahore, Pakistan in 1991. He received the MS degree in 1993 and the PhD degree in 1997, both in Electrical Engineering from Michigan State Univeristy, East Lansing, Michigan, USA. His current research interests include performance evaluation, high-performance computing and networking systems, and multimedia systems. He has written over twenty refereed conference and journal papers on related topics. Dr. Waheed is a member of the IEEE Computer Society.