Adaptive Content Delivery for Scalable Web Servers

Rahul Pradhan and Mark Claypool

Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609, USA
claypool@cs.wpi.edu

Abstract

The phenomenal growth in the use of the World Wide Web often places a heavy load on networks and servers, threatening to increase Web server response time and raising scalability issues for both the network and the server. Rather than large response times or connection failures that occur with typical loaded Web servers, clients are often willing to receive a degraded, less resource intensive version of the requested content. We propose and evaluate a server system that is capable of quantifying the load on a Web server and scaling the Web content delivered to a lower quality upon detecting high load, thus satisfying more clients and reducing response times. We experimentally evaluate our adaptive server system and compare it with a non-adaptive server. We find that our adaptive content delivery server system can support as much as 25% more static requests, 15% more dynamic requests and twice as many multimedia requests as a non-adaptive server.

1 Introduction

The number of people on the World Wide Web is expected to reach 320 million by 2002 (*The Internet, Technology, Analysis and Forecast* 1999) with the top 100 most frequently accessed Web sites (*Hot 100 Sites* n.d.) getting more than a million requests per day. When the request rate on a Web server increases beyond server capacity, the server becomes unresponsive and starts rejecting connections. Web servers that do not respond quickly under heavy loads can slow down a network connection, deny service to clients and cause network failures. The inability of Web servers to degrade gracefully under load results in connections being denied, which translates to a loss in revenue for today's Internet economies (*Scaling the Internet Web Servers* n.d.) and loss of interest from customers (Bhatti, Bouch and Kuchinsky 2000).

To reduce server load, (Andersen and McCune 1998, Colajanni, Yu, Cardellini, Papazoglou, Takiazawa, Kramer and Chanson 1998, Colajanni, Yu and Cardellini 1997) propose distributing load across geographically separated servers. (Mourad and Liu 1997) proposes redirection servers to transparently redistribute users' requests. Numerous networking companies (*Nortel Networks* n.d., *Cisco Systems* n.d., *Quickweb: Intel Corporation* n.d.) have commercial products to load balance a server. However, even a load-balanced site can still suffer from a single server being overloaded. And over-provisioning a Web server with twice the normal capacity cannot always prevent overload conditions (Schechter, Krishnan and M.Smith 1998) since Web loads tends to be bursty.

Another approach to reducing server load is to serve reduced Web content. Reducing the number of embedded objects per page can also result in significant additional resource savings (Abdelzaher and Bhatti 1999). Reducing the number of local links, followed manually by administrators of large sites such as CNN (upon overload due to breaking news (Vahdat, Eastham, Yoshokawa, Belani, Anderson, Culler and Dahlin 1998)) can also reduce load. "Thumbnails" of images can further reduce load by only serving large images when a user specifically requests them. While effective, most of these techniques are not automated to respond to server load dynamically.

Server load may be automatically reduced through the use of adaptive content delivery. Current adaptive content delivery techniques primarily consider network bandwidth as a measure of the load on the server (Abdelzaher and Bhatti 1999). However, server overload may also occur due to the saturation of the CPU or the disk on the server. Streaming multimedia applications are often directly affected by CPU and disk utilization. Dynamic Web page generation techniques, too, are also resource intensive, as they often require some processing.

We propose an approach that detects when a server is overloaded and adapts content to serve more clients and reduce response times. Building upon work in (Abdelzaher and Bhatti 1999), we use request rate and network bandwidth as criteria for switching content, and in addition consider CPU and disk utilization when determining server load. We develop a measure of server load based on the CPU utilization, disk utilization, the outgoing network bandwidth and the observed request rate that is effective for both multimedia and dynamic pages.

Our server adapts content *apriori* and stores two copies of content that differ in quality and processing requirements. Building upon (Ma, Bedner, Chang, Kuchinsky and Zhang 2000), we adapt static Web pages by number of embedded images, dynamic Web pages by processing time, and images and video by picture quality. Switching the quality of the content is done transparently; the client always retrieves the original URL, but the actual content is linked in with soft links (or short-cuts) for either high quality or low quality, as appropriate.

We build a system which dynamically adapts the content delivered to the client according to the server load. Our system is designed for a graceful degradation of server performance under heavy load rather than cope with sustained server overload. If the server is lightly loaded then high quality is served, otherwise the low quality, less resource intensive version is served.

We experimentally measure the effect of our load monitoring process and the benefits of our adaptive server under conditions of high load. Our analysis shows that our adaptive server can support about 25% more static requests, 15% more dynamic requests and double the number of multimedia requests as a non-adaptive server.

The rest of this paper is organized as follows: Section 2 shows the effect of system load on server performance, providing thresholds for content adaptation; Section 3 shows the results of experiments measuring performance for both the adaptive and non-adaptive server for static, dynamic and multimedia workloads; and Section 4 concludes and presents some possible future work.

2 Web Server Load Monitoring

This section describes our techniques to measure the server load. Our measurement of server load is to determine when the system is loaded such that response times start to degrade. Although our current implementation is specific to the Linux operating system, the techniques we propose are general enough to apply to any Web server system.

Linux provides a virtual file system known as the `proc` file system that acts as an interface to the internal data structures in the kernel. The `proc` file system contains information on every process, the processor utilization, the internal workings of the kernel, the number of bytes read and written from disk, and the number and sizes of packets sent and received over the network interfaces.

To measure CPU utilization we made use of the *top* system command, which obtains statistics from `proc` files. *Top* reports the percentage of CPU time in user mode, system mode, running niced tasks, and idle. We measure the processor utilization as the sum of the user and system processes.

To measure network utilization, we use the `/proc/dev` file which contains statistics such as the number of packets sent or received. We calculate the network utilization as the number of bytes sent or received over the theoretical maximum (10 Mbps on our server).

To measure disk utilization, we use the `/proc/stat` file which contains statistics about the number of blocks read and written. We calculate the disk utilization as the number of bytes read or written over the theoretical maximum (24 Mbps on our server).

To obtain average utilization values rather than bursty instantaneous values we use exponential averaging. We use a weighting of 0.40 as it seemed to be effective in pilot studies for providing an appropriately responsive server. We also use 40 seconds as our timing interval utilization measurements, as it best captured the variation in the utilization values during our pilot studies.

For each measure of load, we use load thresholds at which we determine the server is at high or low load and switch content to high or low quality, as appropriate. We set

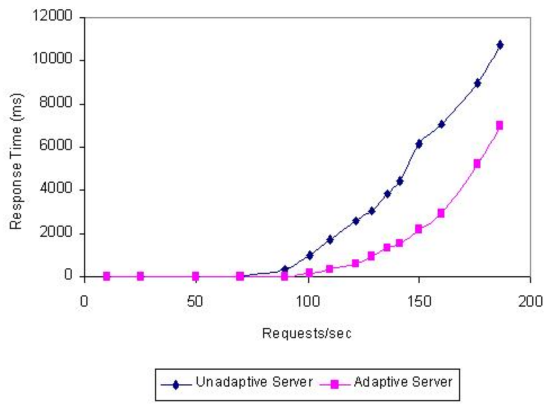

Figure 1: Response Time vs. Request Rate

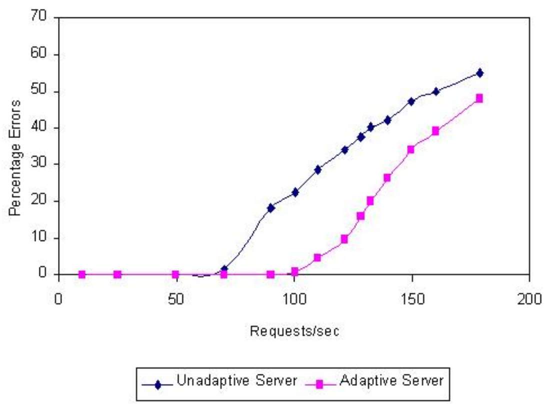

Figure 2: Error Rate vs. Request Rate

two threshold values for each utilization parameter, a high threshold value and a low threshold value, to prevent the system from oscillating from high quality to low quality too frequently. We set the load thresholds based on baseline experiments, but do not present the results here due to space constraints. More details on our load measurement techniques can be found in (Pradhan and Claypool 2001).

Based on the results from (Pradhan and Claypool 2001), we selected the following threshold values: for CPU utilization, we chose 75% and 60% as the high and low thresholds; for network utilization, we use 55% and 35% as the high and low thresholds; and for disk utilization we used 75% and 60% as the high and low thresholds.

3 Adaptive Web Server Performance

To evaluate our Web server system, we use three client machines simulating multiple Web clients issuing uniform sized HTTP GET requests via *httperf*. The clients were connected to the network with a standard 10 Mbps Ethernet link. The hardware platform that we used for our experiments was an Intel Pentium III, 500 MHz with a SCSI disk, 128 MB of main memory, and a standard 10 Mbps Ethernet card. The Linux version was 2.2.14. The Web server software used was Apache (*The Apache Web Server Project* n.d.) version 1.3.12.

We evaluate the performance for our system based on the throughput, response time, error rates and the number of concurrent multimedia and dynamic clients the Web server supports. Throughput is the responses rate (responses/sec) from the server. Response time is the time it takes from the sending of a request until the arrival of the last byte of the response at the client. Error rate is the rate at which a failure occurs in the client in an interaction with the server, typically from an overflow in the connection queue of a busy server.

For the adaptive server system and the non-adaptive server, we compare the perfor-

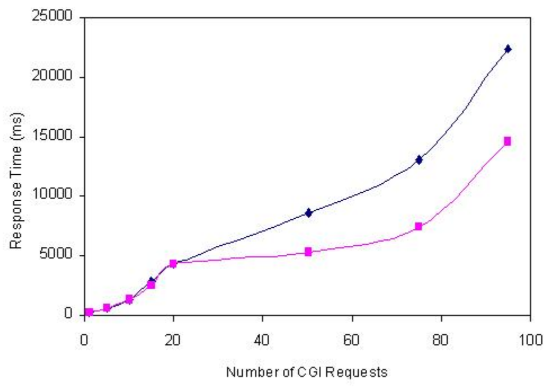

Figure 3: Response Time vs. Number of CGI Clients

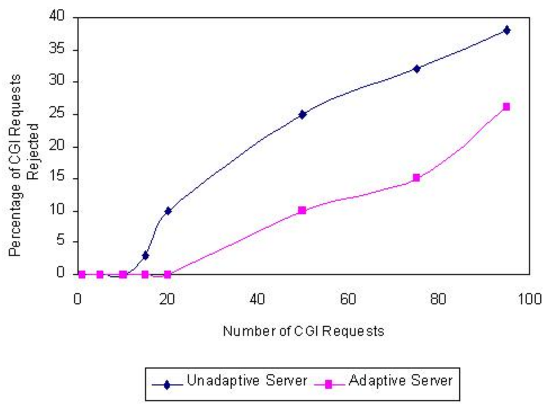

Figure 4: Failure Rate vs. Number of CGI Clients

mance for a static HTTP workload, a continuous media workload and a dynamic workload and analyze the overhead incurred by our system.

Figure 1 shows the average response time for each request. The response time starts out at a minimum of 4.8 ms and then increases slowly as more requests are received until it reaches the maximum server capacity after which it increases exponentially. The response time graph for the adaptive server is shifted to the left indicating that acceptable response times are obtained for more requests per second.

Figure 2 shows a graph of the error rates as the percentage of requests that failed. As the request rate increases, the server gets saturated and the number of requests that are rejected increases. The adaptive server has a lower error rate than the non-adaptive server under heavy loads.

Figure 3 shows a graph for dynamic requests. The horizontal axis represents the number of concurrent CGI requests while the vertical axis represents the response time in milliseconds. As shown in the graph, the response time increases almost linearly in both cases until about 20 concurrent CGI requests. After 20 requests, the non-adaptive server performance degrades considerably, while the adaptive server performance remains almost constant, since the server is now serving smaller, less resource intensive files.

Figure 4 shows a graph of the failure rate. The horizontal axis represents the number of concurrent CGI requests while the vertical axis represents the percentage of CGI requests rejected. The adaptive server rejects fewer requests as compared to the non-adaptive server which reaches saturation earlier due to its resource intensive content.

This section evaluates the benefits of the adaptive content delivery system for continuous media requests. We built a streaming MPEG server to serve MPEG-1 files with quality various content for adaptation. We built a MPEG client which connected to our streaming server. We varied the number of clients and noted the corresponding frames per second sent by the server.

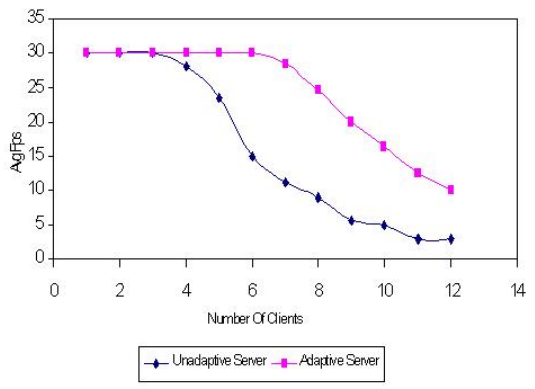

Figure 5: Frame Rate vs. Number of Multimedia Clients

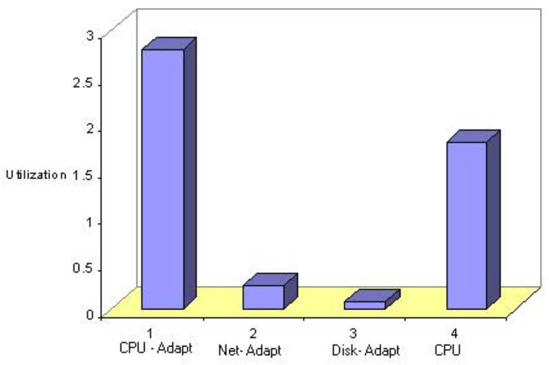

Figure 6: Overhead for the Adaptive Content Delivery System

Figure 5 shows the frames per second sent to the client versus the number of clients requesting the file. As observed from the graph, as the number of clients increases the processing load on the server increases and it cannot maintain a 30 frames per second rate. For the non-adaptive server, the frames per second sent falls below 30 after the 3rd client and it falls below 15 frames per second after 5 concurrent clients. On the other hand, the adaptive server can handle 6 concurrent multimedia clients and it falls below 15 frames per second only after 10 concurrent clients. The adaptation leads to the server sending a smaller video frames which significantly reduces the load on the server resulting in a better frame rate.

Lastly, we examine how much load our adaptive system itself induces on the server. Figure 6 shows the overhead for the content adaptation system. The utilization values shown are the average values for 1000 seconds run on an idle system. The graph shows the CPU utilization for both the adaptive and non-adaptive server, and the network and disk utilization values for the adaptive server.

The values for the CPU utilization for a non-adaptive server show the minimal overhead for our system. We could not measure network and disk utilization without our load monitoring module in place, but these numbers with our system itself are already extremely low.

4 Conclusions

The phenomenal growth in the Internet traffic has placed a heavy load on Web servers. Heavy bursts of traffic overload servers leading to sluggish response times or rejection of requests. Current content adaptation techniques are not particularly well suited to multimedia traffic and dynamic Web pages. Both multimedia and dynamic Web pages have stringent processing, storage and delivery requirements. Hence, serving a variety of requests in a timely manner albeit at a lower quality is of primary concern.

In this paper, we propose an adaptive content delivery system that is light-weight, stand-alone and capable of quantifying the load on the server and transparently adapting the delivered content depending upon the server load. Our system measures the load on the server in terms of its CPU, disk and network utilization to determine the utilization values beyond which to switch content. The content switching is done transparently to both the client and the server.

We evaluated the performance of our system for static, dynamic and multimedia workloads, including a streaming MPEG server and client that can react to the server load by scaling the quality of frames transmitted. We compared the performance of the adaptive content delivery system with a non-adaptive system under similar conditions. We find our adaptive system can serve as many as 25% more static clients. Our system also serves 15% more dynamic clients and almost twice the number of multimedia clients at acceptable frame rates than a non-adaptive server. The adaptive system also shows significant savings in response times for the client.

The main benefits of our approach include: measurement of server load based upon network and disk utilization, in addition to CPU utilization; alleviating server load by a graceful degradation of server performance; transparent content switching for content adaptation; and no requirement of modification to existing server software, browser or HTTP protocol.

For future work, our measure of response time could be used to predict service times. A function to predict service times could be the basis of an admission control technique used to admit only those clients which can be served within a pre-defined response time limit. This can be of great advantage to ISP's and video servers who have clients who require guaranteed response times.

Also, our content adaptation approach can be combined with client resource determination techniques to deliver the quality tailored to each client. Client browsers could explicitly request various quality levels from the server based on client side resources or personal preferences. Our transparent content switching technique can be used to serve different quality content to heterogeneous clients such as desktops, mobile phones or PDA's.

References

Abdelzaher, T. and Bhatti, N.: 1999, Adaptive Content Delivery For Web Server QoS, *International Workshop On Quality of Service.*

Andersen, D. and McCune, T.: 1998, Towards a Heirarchial Scheduling System for Distributed WWW Server Cluster, *Proceedings of The Seventh International Symposium on High Performance Distributed Computing.*

Bhatti, N., Bouch, A. and Kuchinsky, A.: 2000, Integrating User Perceived Quality Into Web Server Design, *Technical report*, HP Laboratories, Palo Alto.

Cisco Systems: n.d. Internet site `http://www.cisco.com`.

Colajanni, M., Yu, P. and Cardellini, V.: 1997, Scheduling Algorithms For Distributed Web Server, *Proceedings of 17th International Conference on Distributed Computing Systems*.

Colajanni, M., Yu, P., Cardellini, V., Papazoglou, M., Takiazawa, M., Kramer, B. and Chanson, S.: 1998, Dynamic Load Balancing in Geographically Distributed Heterogeneous Servers, *Proceedings of The 18th International Conference on Distributed Computing Systems*.

Hot 100 Sites: n.d. Internet site `http://www.100hot.com`.

Ma, W., Bedner, I., Chang, G., Kuchinsky, A. and Zhang, H.: 2000, A Framework for Adaptive Content Delivery in Heterogeneous Network Environments , *Hewlett Packard Labs*.

Mourad, A. and Liu, H.: 1997, Scalable Web Server Architectures, *Proceedings of The Second IEEE Symposium on Computer and Communications*.

Nortel Networks: n.d. Internet site `http://www.nortelnetworks.com`.

Pradhan, R. and Claypool, M.: 2001, Adaptive Multimedia Content Delivery for Scalable Web Server, *Technical Report WPI-CS-TR-01-21*, Worcester Polytechnic Institute.

Quickweb: Intel Corporation: n.d. Internet site `http://www.intel.com/quickweb`.

Scaling the Internet Web Servers: n.d. Internet site `http://www.cisco.com/warp/public/cc/pd/cxsr/400/tech/scale_wp.htm`.

Schechter, S., Krishnan, M. and M.Smith: 1998, Using Path Profiles to Predict http Requests, *Proceedings of Seventh International World Wide Web Conference*.

The Apache Web Server Project: n.d. Internet site `http://www.apache.org`.

The Internet, Technology, Analysis and Forecast: 1999.

Vahdat, A., Eastham, P., Yoshokawa, C., Belani, E., Anderson, T., Culler, D. and Dahlin, M.: 1998, Web OS: Operating System Services for Wide Area Applications, *Proceedings of Seventh International Symposium on High Performance Distributed Computing*.