Adaptive Multimedia Content Delivery for Scalable Web Servers

by

Rahul Pradhan

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

May 2001

APPROVED:

Prof. Mark Claypool, Advisor

Prof. Craig Wills, Reader

Prof. Micha Hofri, Department Head

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. With the advances in the field of optical networking and the increasing use of broadband technologies like cable modems and DSL, the server and not the network, is more likely to be the bottleneck. Many clients are willing to receive a degraded, less resource intensive version of the requested content as an alternative to connection failures. In this thesis, we present an adaptive content delivery system that transparently switches content depending on the load on the server in order to serve more clients. Our system is designed to work for dynamic Web pages and streaming multimedia traffic, which are not currently supported by other adaptive content approaches.

We have designed a system which is capable of quantifying the load on the server and then performing the necessary adaptation. We designed a streaming MPEG server and client which can react to the server load by scaling the quality of frames transmitted. The main benefits of our approach include: transparent content switching for content adaptation, alleviating server load by a graceful degradation of server performance and no requirement of modification to existing server software, browsers or the HTTP protocol. We experimentally evaluate our adaptive server system and compare it with an unadaptive server. We find that adaptive content delivery can support as much as 25% more static requests, 15% more dynamic requests and twice as many multimedia requests as a non-adaptive server. Our, client-side

experiments performed on the Internet show that the response time savings from our system are quite significant.

Acknowledgements

I would like express my deep gratitude and appreciation for Professor Mark Claypool for his continual guidance and support. He has not only been an amazing advisor for this thesis but has also been of help on academic and personal matters. Thank You Mark, for going out of your way to help me and making my stay here at WPI really memorable.

I would also like to thank Professor Craig Wills for being my reader and also for his suggestions and help to improve this work. I would like to thank my best friends, Sooraj and Shivani for being there whenever I needed them. I would also like to thank Akshay for his help in the early part of this thesis.

Last but not the least, special thanks to my parents without whose love, support and encouragement this would not have been possible. I dedicate this thesis to my parents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet has evolved from being just a medium for communication and browsing to a medium for conducting business and selling a myriad of emerging services. The World Wide Web has made the Internet available to a wide range of users. The Web server is at the center of an emerging e-service infrastructure with increasing requirements for reliability, scalability and security guarantees in an unpredictable and highly dynamic environment. This phenomenal growth in traffic and increasingly stringent requirement demands have placed a heavy load on the Web servers.

The Web is emerging not only as an information dissemination mechanism but also as an entertainment medium. The number of people on the World Wide Web is expected to reach 320 million by 2002 [2]. Popular Web sites like Microsoft, CNN, Yahoo which feature in the top 100 [3] most frequently accessed sites, get more than a million requests per day. As Internet usage skyrockets throughout the world this number could easily increase by an order of magnitude as more people get online. However, the hit count does not give an accurate indication of the load on the server - as the richness, perceptual quality, interaction, dynamism and security requirements of the contents increase, the size of the Web pages and the processing time required to serve them increases by an order of magnitude. Personal Web hosting services like geocities.com and nbci.com allow individuals to create and share information easily. With rich media authoring tools readily available more and more Web pages contain multimedia.

The size of the embedded multimedia objects determines the average size of Web pages. This problem is exacerbated by the the increasing popularity of multimedia content. Today the average size of a Web page is 64KB as compared to 32KB in 1997 [4]. As the number of Web-based businesses expands, there is greater competition to capture and hold the attention of viewers. Web-based audio and video streaming capabilities give Web developers a competitive edge in the creation and delivery of captivating applications that attract and retain vast and diverse sets of audiences. Media streaming is becoming popular and widely used because it removes the storage requirement for the client and can be used for real time applications. With media streaming, a server stores the media files and transmits encoded streams to the client. The client then decodes the stream and synchronizes the audio and video and plays it out. Thus the client need not download and store the received multimedia file. This works very well for a live Web cast or for a video on demand server. The client just needs to buffer the media data in memory for a period of time long enough to avoid jitter, and then discard the data after playing. The trend for Web content providers is towards an increasingly multimedia rich experience for the end user. [5] estimates that about 77% of the bytes across the Web are multimedia objects such as images, audio and video clips and 67% of those bytes are images.

With the growth in Internet commerce, dynamic Web pages have become an important tool in the exchange of information. Dynamic Web pages typically require some processing by the server before they can be served. New application technologies like Java, Secure Sockets Layer (SSL) for security, database transactions and sophisticated middleware components increase the processing demands on the server. This means that the delays witnessed by users are directly affected by server performance, and are not simply due to download times. These evolving applications, like continuous media, need high throughput, whereas e-commerce transactions need low response time, even during congested periods. Web servers that do not respond quickly under heavy loads can slow down a network connection, deny service to clients and cause network failures.

Web services need to serve as many users as possible at sufficiently attractive levels of quality and latency to gain and retain their business. Web sites also need to scale under heavy load in order to offer assured service to retain their preferred customers as well as attract potential customers. Flash crowds can overload a popular server leading to sluggish response times or an inability for clients to connect at all [6]. During periods of such high demand, large multimedia objects and computationally expensive dynamic Web pages limit the ability of the server to respond to requests all of its clients.

With network bandwidth increasing due to broadband services like DSL, cable modems and the advances in optical networking, more and more bottlenecks are observed at the server. Thus, there is a need to address the issue of server-side bottlenecks to scale up Internet server capacity to meet networking capacity and service demands. Web servers typically offer poor performance in overloaded conditions, leading to high response times on the most popular servers. When the request rate

Site Types	**Example of Site**	**Impact**
Most Accessed Sites	News Sites, Portals, Top 100	Lost Advertising Revenues
E-Commerce Sites	On Line Catalogs Shopping Sites	Lost Sales Revenues, Fewer Repeat Customers
Streaming Media Sites	Video-Audio on Demand	Lost Event Revenues, Fewer Repeat Business

Table 1.1: The Economic Impact Of Web Performance

on a Web server increases beyond server capacity, the server becomes overloaded and unresponsive. The TCP connection queue of the server's socket overflows resulting in client perceived server outage [7] and the end system can spend more than half of its time processing eventually rejected requests (e.g., protocol stack processing, queuing, socket call processing etc.). Once overloaded, the server starts rejecting connections and cannot offer a graceful degradation in performance.

The incapability 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. Zona research estimates that more than $4 billion in e-commerce revenues are lost each year in the US because of slow pageloading times. Table 1.1 [1] gives the economic impact of overload on different types of sites. It emphasizes the loss for Internet economies on account of poor Web performance. While the consumer E-commerce market is considerable in size, the business-to-business (B2B) E-commerce market is larger and is predicted to grow dramatically. B2B transactions need to be secure, scalable, reliable and instantaneous. Therefore, it is important to consider how to serve the maximum number of clients under constrained conditions.

Improving users' perception of Web page performance is complicated by the fact that the factors that influence Web performance are often interdependent. For example, Web pages that are retrieved faster are judged to be significantly more interesting than their slower counterparts [8], and users may judge a relatively fast service to be unacceptable unless its predictable and reliable. The spikes of traffic that can overwhelm a server have a negative impact on Web users. Users' conception of the Internet is that it provides service on demand, so when they find that at times they are not able to access a site the opinion of the user about the site falls sharply. [8] shows that clients believe that e-commerce companies should possess the capability to have a scalable server which can perform under overloaded conditions. [8] also shows that the inability of an e-commerce site to perform satisfactorily under overloaded conditions affects users' conception of the company's stature and commercial viability. Users do not perceive the network traffic demands, networking infrastructure, ISP's or their own network connection as the cause for the poor performance, but instead they place the blame on the individual businesses represented by the sites [8]. Thus, inevitably if poor response times are regularly observed under loaded conditions, then the opinion of the users about the site falls dramatically. This means that users are less likely to accept delays, or refused admissions to a site.

Zona Research also tracked "bailout rates"(the percentage of people who did not wait for pages to load but instead went to other pages), as shown in Figure 1.1 [1].

The study noted that over 50 percent of the people attempting to download a page in excess of 70KB leave before the page is completed. As might be expected, the larger the page size, the greater the bailout rate: Typically a 40KB page has a 30-percent bailout rate, while a 34KB page has only a 7% bailout rate. What appears to be a mere 6KB difference with 15 percent less information and download time results in a greater than fourfold reduction in the bailout rate. This data demonstrates users' high sensitivity to download times. Thus, the increasing

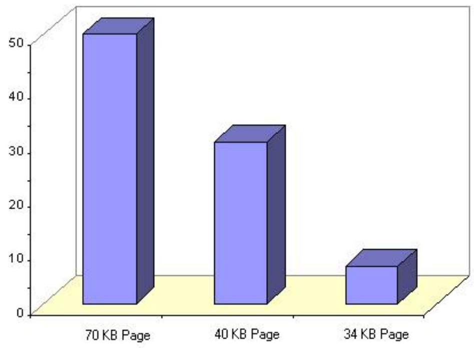

Figure 1.1: Bail Out Rate for Various Web Page Sizes [1]

criticality of Internet applications demand that the servers exhibit a high level of robustness and availability while delivering low response times with a high probability.

There has been significant research to alleviate server load through various load balancing techniques [9, 10]. [11] describes a different approach of adaptive content delivery. It describes modifying Web content to reduce the load on the server. Reducing the number of embedded objects per page can also result in significant additional resource savings. Reducing local links is another way of adapting site content. This approach is sometimes followed manually by administrators of large sites such as www.cnn.com of the Cable News Network (CNN) upon overload due to breaking news [12]. Another approach to content adaptation involves the use of thumbnails to index large images. Thumbnails are typically small images that give a preview of the larger high quality image. Thus, the user can download the entire image only if it is really of interest. [13] describes the benefits of HTTP content negotiation, which uses Apache's [14] built in content negotiation to serve varying content depending on the client/server conditions. However, most of these techniques are not automated to respond to server load dynamically. All the techniques just consider network bandwidth as a measure of the load on the server. This might not necessarily be true as more and more multimedia and dynamic requests are served.

Similar content adaptation techniques can be easily used to adapt server output to client-side resource limitations, or to provide content to more important clients [5]. The processing power, connection bandwidth and the display resolution may vary significantly from client to client especially with the proliferation of mobile computing devices. Content adaptation can provide the most appropriate version of the content to each client in accordance with their resources constraints.

[11] uses request rate and the network bandwidth as their criteria for switching content. Since increasingly a server processes resource intensive multimedia objects and dynamic page generation technologies, the switching should take into consideration those parameters as well. Multimedia places stringent requirements on the system due to its processor and memory intensive operation. The storage and retrieval of multimedia objects are the most resource intensive operations in terms of processing needs and disk access. Dynamic Web page technologies usually involve some processing on the part of the server, generating a Web page on the fly and sending it over the network to the client. Thus, both multimedia and dynamic page generation induce load both to the processor and the disk in addition to the load placed on the network. Thus, in addition to alleviating overload, content adaptation will reduce the amount of server resources wasted on eventually unsuccessful or rejected connections.

For many purposes, the highest reproduction quality is not the aim when pictures are included in a Web page. A client need not wait for a long time for the best quality response from the server, but rather can get a degraded response in a shorter time. This is based on the presumption that as an alternative to connection failures, clients may be willing to receive a degraded, less resource intensive version of the requested contents. This is especially important for multimedia applications where a timely delivery is more important than a guaranteed high quality delivery.

Our solution is designed for a graceful degradation of the server performance under heavy load rather than cope with permanent sustained overload. We concern ourselves with the problem of scaling a dynamic multimedia rich server by switching the content under high load.

This thesis develops a system which dynamically adapts to the content being delivered to the client according to the resources of the server so as to alleviate the load on the server. The dynamic modifications are performed automatically by monitoring the server load.

Server overload may occur due to the saturation of CPU bandwidth, the communication link or the disk bandwidth of the server. Streaming multimedia applications which tend to be resource intensive are dependent upon CPU utilization, network and disk bandwidth and are connection oriented due to their long term streaming nature. The dynamic Web page generation techniques are also resource intensive, as they often require some processing. We develop a measure of the load on the server based on the CPU utilization, disk utilization, the outgoing network bandwidth and the observed request rate.

The remainder of this thesis is organized as follows. Chapter 2 gives a brief overview of the various techniques used to provide a scalable solution to the problem of Web server overload and some related work done in the area of adaptive content delivery. Chapter 3 describes our adaptive content delivery approach in detail, the architecture of our system, and the ways of varying quality for multimedia files. Chapter 4 shows the effect of server performance on the system utilization parameters. Chapter 5 shows the results of tests on both the adaptive and non-adaptive server for static, dynamic and multimedia workloads. Chapter 6 presents possible future work and Chapter 7 concludes the thesis.

Chapter 2

Related Work

Due to the phenomenal growth in the World Wide Web traffic and the consequent load on the Web server, there is currently a lot of ongoing research to improve server performance under load. This chapter, discusses about the various techniques used to improve the performance of a Web server under load including content adaptation techniques which we explore.

2.1 Improving Server Performance

Numerous techniques have been proposed to alleviate server overload. Techniques such as distribution of load across geographically separated servers [10, 9, 15] have been proposed to reduce server load. [16] proposes redirection servers to transparently redistribute users' requests. There has been a lot of research in the field of server load balancing and numerous networking companies have commercial products to load balance a server. This section briefly overviews some techniques used to improve the server performance under load.

2.1.1 Mirrored Architecture

Two web sites are said to be mirrors of each other if a high percentage of paths are valid on two Web sites and the documents linked at those path are similar [17]. These web sites have a complete or a partial replication of content. The selection of the appropriate site is not dynamic, but the clients are either allowed to select a mirrored site or explicitly directed to one.

[17] studies the various mirroring techniques used on the Web and classifies various mirroring techniques on the basis of the similarity in structure and content as:

1. *Level 1 : Structural and Content Identity*

   Every page on host A with a relative path P is byte-wise identical to the page on host B with a relative path P and vice versa.

2. *Level 2 : Structural Identity and Content Equivalence*

   Every page on host A with relative path P, is represented by an equivalent content page on host B with a relative path B and vice versa.

3. *Level 3 : Structural Identity and Content Similarity*

   Every page on host A with a relative path P, is represented by a highly similar page on host B at relative path P and vice versa.

4. *Level 4 : Partial Structural Match and Content Similarity*

   Some pages on host A with relative path P, are represented by a page on host B, with relative path P and vice versa, and these pages are highly similar.

5. *Level 5 : Structural Identity and Related Content*

Every page on host A with relative path P, is represented by a page on host B with relative path P and vice versa. The pages are pair-wise related but in general are not syntactically similar.

2.1.2 Load Balancing

Server load balancing [9] is the process of distributing service requests across a group of servers. Some content-intensive applications have scaled beyond the point where a single server can provide adequate processing power. Both enterprises and service providers need the flexibility to deploy additional servers quickly and transparently to end-users. Server load balancing makes multiple servers appear as a single virtual service by transparently distributing user requests among the servers. The highest performance is achieved when the processing power of servers is used intelligently. Advanced server load-balancing products can direct end-user service requests to the servers that are least busy and therefore capable of providing the fastest response times. Necessarily, the load-balancing device should be capable of handling the aggregate traffic of multiple servers.

Some of the techniques used for Load Balancing are [9]

1. **HTTP Redirection**

   HTTP redirecting distributes a Web site's load among multiple servers by connecting users' browsers directly to the servers. When you select a Web site's URL, you usually connect directly to the computer servicing that URL. For example, type www.wpi.edu, and the server designated to respond to requests for that HTTP address will provide the WPI Web site. However, if the Web site has a replica on a server with the URL www.wpi2.edu, an HTTP-redirecting program can redirect users' browsers to www.wpi2.edu to balance the Web site's load. HTTP-redirecting software automatically directs browsers to a Web site replica if the primary URL's server fails. HTTP redirecting's main disadvantage is that it does not work with all Web browsers.

2. **Packet Redirection**

   Packet redirection is a transparent mechanism as compared to HTTP redirection. The server reached by a request reroutes the connection to another server through a packet rewriting mechanism. The load balancing algorithm can be static or dynamic wherein there is a periodic communication between servers to determine load.

3. **Domain Name Service (DNS) - Based Approach**

   Round Robin (RR) DNS allows a domain name associated with several IP addresses, each of which represents a different web server. For a DNS lookup, the RR DNS server returns a domain name mapping in a round robin fashion, thus each request gets routed to a different server in a round robin manner and distributes the load on the servers.

4. **Load Balancing Switches**

   Load balancing switches such as Cisco's LocalDirector and Nortel Networks' Alteon ACEdirector redirect TCP/IP requests to multiple servers in a server farm, providing a highly scalable, interoperable solution that is also very reliable. These switches sit between the connection to the Internet and the Web farm. All requests come to the switch using the same IP address, and then the switch forwards each request to a different Web server based on various algorithms implemented in the switch. Switches will frequently be able to ping the servers in the farm to make sure they are still up, and to get an estimate of how busy they are so they can make informed decisions about load balancing.

Application load-balancing software distributes a Web site's workload among servers according to the content browsers request. A primary Web server accepts all incoming Web traffic and performs tasks such as static HTML file transmissions. The primary server redirects back- end applications, such as Active Server Pages (ASP) and Common Gateway Interface (CGI) programs, to other computers. Another common algorithm is to load balance based on the content of the request, such as the IP address of the requestor, or some other information in the request. Using the IP address alone does not work well because some Internet service providers (ISPs) and some companies use proxy servers that change the IP address of all of the requestors that go through the proxy to the same address.

Load balancing with admission control [18] has often been used for overload protection. The traditional method of scaling Web servers has been to do load balancing across a server farm using a front-end load distributor. Such a solution is inherently non-scalable. If the front-end operates at the transport layer or below, scalability suffers because of contents in the memory cache of each server while a content-aware load distributor itself becomes a bottleneck because of the amount of work expected of it [19]. Admission control on the other hand is used to improve the average latency of admitted client requests by rejecting a subset of clients. It is based on the premise that consistent rejection of all requests from a subset of clients may be better than indiscriminate connection failures affecting all clients alike in the absence of admission control. Though admission control can improve server performance by preventing overload, it offers no service to rejected connections, and cannot recover the significant resources wasted in the communication protocol stack on client requests rejected by the server. This wasted kernel overhead may be significant at overload. As an alternative to connection rejection by admission control, Web server load can be reduced by using multicast to distribute commonly requested pages [20]. Other techniques include [10] dynamically scheduling HTTP requests across clusters of server to optimize the use of resources. Rent-a-server, a technique for server replication on demand is presented in [12] to replicate servers on overload.

All the above mentioned techniques essentially propose solutions based on load balancing among multiple servers. However, they do not handle the problem of scalability of a single server, as even a load-balanced server farm may have a problem of sudden degradation of a single server performance. We concern ourselves with the problem of managing an overloaded individual server. Our technique is designed for alleviating peak load rather than cope with sustained load. If the server is loaded for a significant proportion of its uptime then, its calls for upgrading of the server platform. To cope with server overload, servers are either over provisioned or use admission control. When overprovisioning is used often twice the normal capacity is allocated to the Web server [21]. This approach does not always prevent overload conditions. This is because during the evolution of Web applications, there has been a steep growth in the client demand curve that makes provisioning difficult and not conducive to static resource allocation approaches. Thus, brute force server resource provisioning is not fiscally prudent since no reasonable amount of hardware can guarantee predictable performance for flash crowds.

2.1.3 Content Adaptation

A different approach called content adaptation is described in [7] to reduce overload.

Content adaptation involves adaptation of content served to satisfy the server, network or the client conditions. GIF and JPEG images constitute 67% of the total bytes surveyed [5]. These images can be significantly compressed without a proportional decrease in quality. [22] shows ways of adapting the content according to the client requirements. [7] talks about content adaptation to balance server load. To support multiple versions of the same content the path to a particular URL in a given content tree is the concatenation of the content tree name and the URL name, prefixed by the name of the root service directory of the web server. It applies to static as well as dynamic content. Their load measurement is based upon the request rate and the aggregate delivered bandwidth.

Intervention of the content provider may be required to authorize or fine tune certain types of adaptation during the off line pre processing stage. For adaptation technology to be cost effective, the intervention must be minimal and should not change the way content providers have traditionally created Web sites. The cost of adapting content should be less than the cost of alleviating load by upgrading the server's machine and /or its network connection. Content authoring tools allow Web content developers to annotate parts of the content with the specific adaptation tags (for example, expendable, degradable etc.). These tags are preprocessed by content management tools to create separate standard-HTML versions of the site. The appropriate version will be served at run time depending on load conditions. Since the created content versions contain only standard HTML and image formats, no modification is required to the browsers. Default adaptation actions may be used by the preprocessing tools on those parts of the content that have not been tagged. These defaults will reduce the need for explicit adaptation tags thus substantially reducing the effort of utilizing adaptation technology by content providers.

Many Web developers today use an *alt* clause to define an alternate text for cosmetic items like icons, bullets, gifs etc. Adaptation tools may make use of this clause, when available, to replace cosmetic items in less resource intensive versions of content. Content providers may tag objects that should not be removed by default treatment.

Another way of adapting content is to reduce local links. This reduction will affect user browsing behavior in away that tends to decrease the load on the server as users access less content. Reduction of local links may be automated, eg., by limiting the web site's content tree to a specific depth from the top page. Content providers may indicate, using special tags, subtrees that should be preserved beyond the default depth during the reduction process.

There has been numerous research to adapt Web content to account for client variability [23, 22]. Universal access is a concept raised by the research community to address technical issues for enabling information access in a heterogeneous network environment, by accommodating the special needs of users and the constraints of client devices and network characteristics. The goal of universal access is to provide the necessary Internet infrastructure to allow users to access any information over any network from anywhere through any type of client device [24]. To provide universal access, Web content may need to be transformed into an appropriate representation before being delivered to the client.

2.2 Web Content Adaptation

Web content Adaptation involves adapting the content to be delivered according to the bottleneck resource. This section gives an overview of the various ways of adapting Web content.

2.2.1 Multimedia Compression Techniques

Transcoding is a technique to dynamically customize multimedia objects to prevailing conditions. Transcoding can be performed along a number of different axes and the specific transcoding technique used depends on the type of the multimedia object [5]. In this section, we feature transcodings that vary the quality resulting in file size savings. This section gives a brief overview of some of the most widely used multimedia compression techniques.

1. **JPEG**

   JPEG [25] is a standardized image compression mechanism. JPEG stands for Joint Photographic Experts Group, the original name of the committee that wrote the standard. JPEG is designed for compressing either full-color or gray-scale images of natural, real-world scenes. It works well on photographs, naturalistic artwork, and similar material; not so well on lettering, simple cartoons, or line drawings. JPEG handles only still images, but there is a related standard called MPEG (described later) for motion pictures. There are two advantages of using JPEG over other color storage formats - the reduction of file size and fact that it stores 24 instead of 8 bit-per-pixel color data.

   JPEG works by first transforming the image into brightness and color components which are separately encoded - with different compression parameters

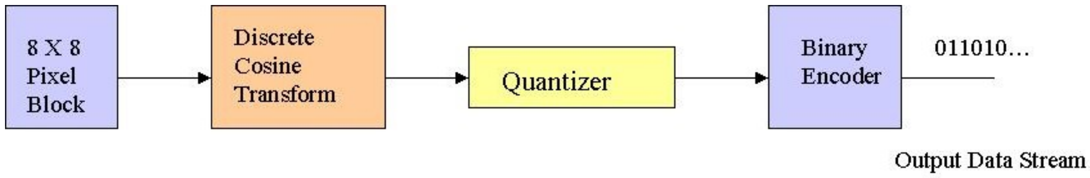

Figure 2.1: JPEG Compression

for each element. The image is then grouped into 8x8 blocks, each of which is transformed using a discrete cosine transform (DCT), which results in a map containing the average value for the block with successively higher-frequency changes within the block. DCT provides a good approximation to decompose an image into a set of waveforms, each with a particular spatial frequency. Hence frequency components which are less perceptible to the human eye can be dropped. Thus, JPEG is intended for compressing images that will be looked at by humans. These values are then *quantized* and rounded to integers (this is the lossy step). Finally, the reduced coefficients are encoded using Huffman variable word length coding and stored in the file together with the compression parameters used.

2. **GIF**

The GIF [26] format is best used for graphical or artistic images. Because GIFs are saved by changes in color information rather than by actual colors, they are usually smaller in file size. The 8-bit color scheme used by GIFs is another reason that the file size is usually so small (as compared to the JPEG 24-bit scheme). When compressing with the GIF format, there is no image degradation involved. GIF is a data stream oriented file format used to define the transmission protocol of a variable length code LZW - encoded bitmap data. Another advantage of the way that GIFs deal with colors is that they can be interlaced so that they slowly appear in greater and greater resolution until they are fully loaded. This is especially helpful when loading a large image as the image can be seen while it's loading instead of waiting until it's fully loaded.

3. **MPEG**

MPEG [27] is a compression standard for audio, video, and data established by the International Telecommunications Union and International Standards Organization.

MPEG-2 is targeted for coding broadcast-quality video signals, hence it is necessary to digitize the source video at its full bandwidth, resulting in both even and odd field pictures in the sequence. The MPEG-2 standard provides a means of coding interlaced pictures by including two field-based coding techniques: field-based prediction and field-based DCT.

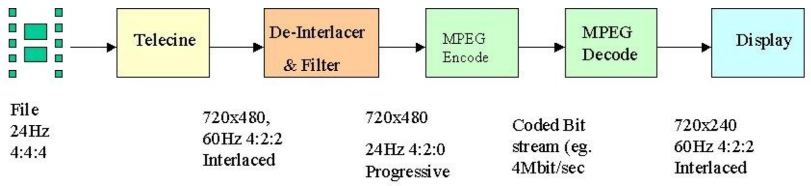

Figure 2.2: MPEG Compression

MPEG distinguishes four frame types of image coding for processing; I-frame, P-frame, B-frame and D-frame. The I-frames are Intra-coded images and are self contained. They are compressed without any reference to other images . I-frames can be treated as still images and are used for random access. The compression rate of the I-frames is the lowest within MPEG. The P-frames are the Predictive-coded frames. The encoding and decoding of P-frames requires the information of previous I frames and/or all previous P-frames. Compression rates for P frames are higher than I-frames. B-frames are Bi-directionally predictive-coded frames. The encoding and decoding of B-frames requires the information of the previous and following I- and/or P-frame. A B-frame is defined as the difference of a prediction of the past image and the following P- or I-frame. The highest compression rate can be attained by using B frames. The D-frames or the DC-coded frames are intra-frame encoded. They can be used for fast forward or fast rewind. D-frames consist only of the lowest frequency of an image.

2.2.2 Content Adaptation Techniques

[23] discusses some of the content adaptation technologies useful for adaptive content delivery. These techniques result in a smart delivery of content, adapting to the existing resource bottlenecks.

1. **Information Abstraction**

   The goal of information abstraction is to reduce the bandwidth and processing requirement for delivering the content by compressing the data, while preserving the information that has highest value to the user. Examples of information extraction include text summarization, image thumbnail generation, and video highlighting and key-frame extraction. Such algorithms can also be used to improve the user's browsing experience by providing a preview of the content. In this way, users are able to quickly browse though more information even though the server/network resources may be constrained. Moreover, information abstraction can be very useful when the client device has limited display capability, such as on palmtops and smart phones. For example, summarizing each paragraph by a few words and shrinking the size/resolution of each image in a Web page will help to fit this page on the small screens of those devices.

2. **Modality Transformation**

   Modality transform is the process of transforming content from one mode to another so that the content can become useful for a particular client device. For instance, most handheld computers are not capable of handling video data because of both hardware and software constraints. In order to make the information contained in the video accessible on these devices, video can be transformed into sets of images and audio into closed caption texts. In this way, users will be able to receive useful information in whatever form that their devices can handle. Other examples of modality transform include speech-to-text and text-to-speech transform and table-to-plain-text or table-to-list transform for HTML. The primary goal of modality transform is to adapt the content representation to client device capabilities. In some cases it may even reduce data volume and, thus save bandwidth in delivery.

3. **Data Transcoding**

   Data transcoding is the process of converting data format according to resource requirements and client device capability. For example, a server which is loaded can serve a lower quality version without as much strain on its resources. Also, some client devices may not be able to display color GIF images because of the lack of viewing or rendering software or the constraint of hardware display capability, such as a black-and-white screen. In such cases, there is a need to transcode the original images into another appropriate format, such as GIF-to-JPEG or color-to-grayscale transformation, so that they can be viewed on the client device in a shorter time. Other examples of data transcoding include video format conversion (such as MPEG-to-QuickTime), audio format conversion (such as WAV-to-MP3), and document format conversion (such as Postscript-to-PDF).

4. **Data Prioritization**

   The goal of data prioritization is to distinguish the more important part of the data from the less important part so that different quality of service levels can be provided when delivering the data through the network. For example, less important data can be dropped under overload condition or can be progressively delivered to send out the more important data first (such as low-resolution images) and then deliver the less important data to enhance the information later (such as reconstruction of high-resolution images). In this way, we can improve the user's browsing experience by efficiently utilizing available resources. Data prioritization can be achieved within a single media type by using special encoding schemes such as layered coding [28] and multi-resolution compression [29]. It can also be done across multiple media types by giving audio higher priority than video and text higher priority than other types of media.

5. **Purpose Classification**

   A typical Web page contains a lot of information and media objects that are redundant or may not be of interest to a user. For example, an e-commerce Web site may have multiple images for linking to the same product site on the top, bottom and the side of the page. A portal site usually contains many images of banners, logos and advertisements. These data often consume a good deal of network bandwidth and, therefore, decrease the efficiency of information delivery. The purpose of each media object in a Web page can be classified to improve the efficiency of information delivery by either removing redundant object or prioritizing them according to their importance.

Purpose classification of a media object can be done using content analysis techniques. It can also be achieved to some extent by matching URL strings with a pre-established database or via heuristics for associating meanings with certain text contained in the URLs. For example, advertisement images can be detected and blocked by matching URL strings with a list of keywords like "ad", "banner", "advertisement", "promotion", or a list of known advertising web hosts. Objects with names or "alt" tags containing "bullet" and "logo" are deemed less important, or even redundant.

The W3C and the IETF have existing standards and on-going discussions on facilitating server/proxy decision making on the mechanisms of content adaptation and content delivery. One such protocol is the Synchronized Multimedia Integration language (SMIL)[30, 31]. SMIL is a markup language that enables the synchronized delivery of multiple video streams, audio streams and images. It provides conditional constructs to switch tasks (eg. request different content) based on bandwidth conditions. The Extensible Markup Language (XML) [32] describes the logical representation of data and can be utilized to facilitate serving content to different types of clients under varying conditions. The logical representation of data can be converted to an appropriate representation for display using Extensible Style Sheet Language (XSL).

The HTTP/1.1 content negotiation capability [33] and the CC/PP [34] are mechanisms for the client to convey along with its request its preferred version of content and its user agent information. In HTTP/1.1 content negotiation, a user agent can specify in the HTTP header that, for example, English documents are preferred over JPEG, or that JPEG images are preferred over GIF images. [13] shows how HTTP content negotiation can be used to provide an adaptive delivery of web content. CC/PP specifies client capabilities and user preferences as a collection of URI's and RDF (Resource Description Framework Text) [35], which is sent by the client along with a HTTP request. The URI's point to a RDF document which has the details of the client capabilities. RDF provides a way to express metadata for a web document. The CC/PP scheme allows proxies and servers to collect information about the client, from directly the client, and to make decisions based on this information for content adaptation and delivery.

We make use of some of the content adaptation techniques described in order to generate different versions of the Web content. We use content adaptation techniques like Data Transcoding and Purpose Classification to generate varying levels of quality. We use multimedia compression techniques for the quality scaling of multimedia images and videos.

Chapter 3

Adaptive Content Delivery System

This chapter describes in detail our adaptive content delivery system and describes the techniques used to generate varying quality of content in order to adapt to the prevailing server load.

3.1 Adaptive Content Delivery Architecture

Adaptive content delivery is a system technology that transforms Web content and Delivery schemes to optimize the browsing experience for a client. The goal of content delivery in our context is to take into consideration various parameters affecting server performance and then serve the best possible content over all users. We concern ourselves with the problem of alleviating server load hence, Web content must be adapted in a way that preserves essential information but yet reduces the server resource requirements of content delivery. When the entire system is overloaded introducing an extra stage of computation, such as data filtering and compression will further increase the load on the server. We try to follow the approach described in [7] of pre-processing the content *apriori* and storing multiple copies that differ in quality and processing requirements such that at overload the server can switch to a pre-existing less resource intensive version of the content.

Web server performance depends upon several factors: hardware platform, operating system, server software, network bandwidth and workload. The main factor in server-initiated content adaptation is the parameter based on which the content selection is made. The way [11] calculates the parameter makes it infeasible to use it for serving streaming multimedia or dynamic Web pages as it does not take into consideration the CPU and disk utilization while calculating the load. We take into consideration those parameters as well as the available network bandwidth and the observed request rate. These utilization parameters form a basis for determining the need for content adaptation. Our proposed architecture provides dynamic (on the fly) content adaptation as each request comes in. Based upon the current server utilization and the observed request rate, we determine the type of content to be served.

The framework for our system is as shown in Figure 3.1. It consists of the following modules:

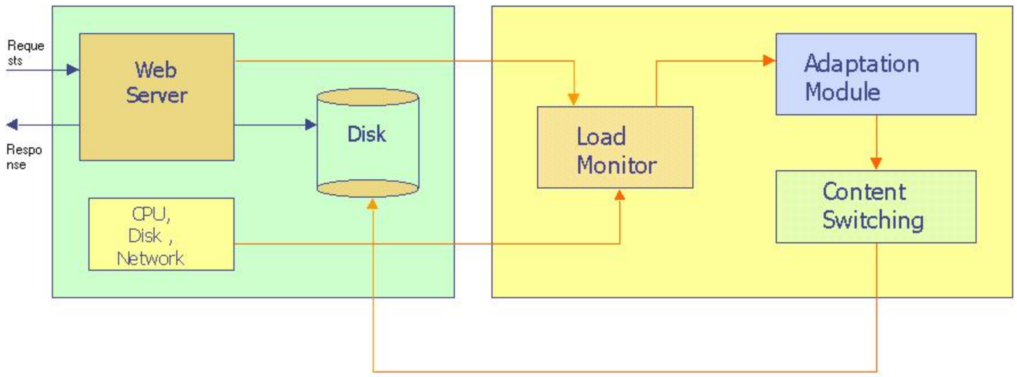

Figure 3.1: Adaptive Content Delivery Framework

The dashed lines represent adaptive framework data, the solid lines represent Web data

1. **Load Monitoring Module:**

   This module continuously monitors the Web server to determine the load on the server and detect overload conditions. The measure of the utilization of a server serving multimedia content can be quantified by taking into consideration the processor, disk and the network utilization along with the observed request rate for the server. These parameters summarize the resource consumption of the server. Chapter 4 describes this module detail including the determination of thresholds for the utilization parameter.

2. **Adaptation Module:**

   This module decides on the quality of content that can be served. When the server utilization is low all clients receive the best available quality of the content. However, when the server utilization approaches saturation and the observed request rate exceeds a threshold a fraction of clients will be served degraded quality content in order to avoid server overload and connection failures.

3. **Content Selector:**

   Based on the adaptation made by the adaptation module the content selector selects the appropriate content to be served. For example, if the load monitor shows that the server is lightly loaded then the adaptation module informs the content selector to serve the highest quality version available, otherwise to serve a less resource intensive version. This content switching is done transparently. The client always types the original URL and the page displayed also has the same address, but the content varies with the load. We use soft or symbolic links to switch the content from high quality to low. Symbolic links are special files in UNIX, that do not contain any data, but instead are just "pointers" or "shortcuts" to other files. The symbolic links are used as pointers to point to the different files with varying quality. Since symbolic links are just pointers we can make the entire content switching process transparent to the client. For example, consider a file, *index.html* which the client requests. The client, always asks for *index.html,* but internally on the server, file name is mapped to either *indexhigh.html* or *indexlow.html,* which are the two variations of the file. Depending on the server load the symbolic link is made to point to either the high or the low content. This content switching is not only transparent to the user but is also transparent to the server. It does not need any modification of the server and also adds minimal load to it.

4. **Web Server:**

   We used the Apache [14] Web server (version 1.3.12) for our experiments. It is the most widely used server on the Internet today. We ran the Web server in a stand alone mode. In standalone mode, the server is started only after a pool of HTTP processes are spawned and waiting to service incoming requests. On startup, a predefined number of HTTP processes are spawned. Once running, the server increases or decreases this number depending on its load. A master process manages this pool of processes by periodically checking the number of idle child processes and dynamically adjusting this number to the current load. There is a limit on the total number of simultaneous requests that can be supported; no more than this number of child server processes will be created. Each HTTP child process has a finite lifetime, limited by the number of requests it can handle. This helps reduce the number of processes when the server load decreases.

3.2 Adapting Web Content

Web content must be adapted in a way that preserves essential information yet reduces the resource requirements of content delivery. The feasibility of adapting content dynamically depends upon the availability of a varying quality of content so that the server can choose the correct content depending on the overload condition. Web content quality can be easily varied by changing the transcoding of the multimedia objects. By their very nature, multimedia objects are amenable to soft access through a quality-versus-size tradeoff. In order to increase content accessibility and improve the users' online experience, many media processing techniques can be used to enable an intelligent information delivery. Several existing content adaptation techniques apply image processing techniques to adapt the embedded images of a Web page according to the characteristics of the bottleneck resource. Multimedia content can be adapted by changing the quantization factor of the object leading to significant file size savings. We dynamically vary the quality of the multimedia objects according to the server load characteristics. We used some of content adaptation techniques discussed in Section 2.2 to generate varying quality of content on the server. For our system we use two quality levels, high and low. High quality corresponds to the best quality Web content, the low quality content is the less resource intensive version of the best quality content.

For a static web page the best quality is the actual web page along with the graphics, while the low quality corresponds to the text version of the same. The improvements in the server performance here are obtained by the reduction in the size of the objects and the fewer number of requests to the servers due to the reduction in the number of embedded objects in the page.

For dynamic Web pages the scaling to different qualities is slightly complicated as the page is generated on the fly. Dynamic applications exacerbate the server performance by overloading the server and increasing the client perceived delay. With the phenomenal growth in E commerce and B2B transactions. Dynamic web pages are computationally expensive since they require some processing and disk accesses. [36] shows the effect of disk utilization on the disk and CPU utilization. A typical CGI request consists of accessing a large database, processing it and then generating the dynamic page on the fly. Such a resource intensive script adds a lot of overhead to the server. Hence, it is necessary to scale under heavy load. The script can be modified by reducing the computations involved and the data output to the client. Thus, a dynamic Web page can be varied in quality by not only altering its static parts but also by executing a different version of the script which is less resource intensive. A lower quality script may look for fewer matches thus reducing the processing time and also the data to be delivered to the client and in turn reducing the overhead on the server to a minimum. We scale a dynamic request by varying its processing capability. Our high quality script performs processor intensive computations and returns a large output file to the client. The processor intensive computations include threads that count to infinity to keep the CPU busy and program to read a large file. On the other hand, a less resource intensive or lower quality script was one which did minimal computation and returned a smaller output file.

In order to determine the effect of the output file size on the response time, we performed tests during which the script returned increasingly large amount of data. Figure 3.2 shows the effect of the output file size on the response time. The horizon-

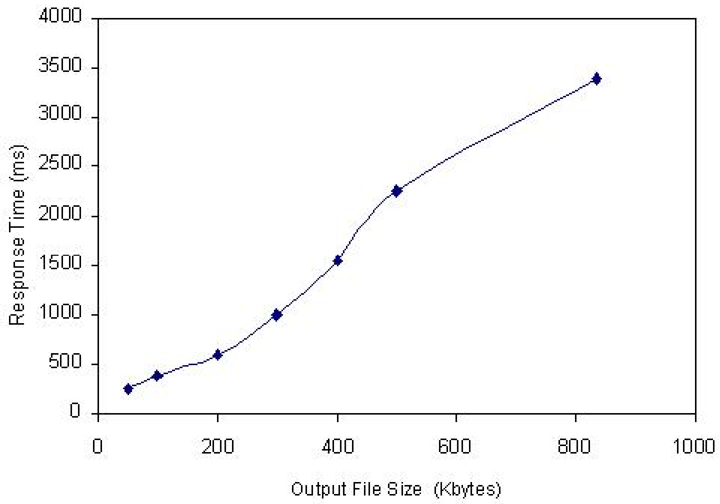

Figure 3.2: Response Time(ms) vs CGI Output File Size

tal axis represents the output file size in kilobytes and the vertical axis represents the response time in milliseconds. The graph clearly shows that as the output file size increases the response time increases significantly for higher file sizes.

For multimedia objects we used quality scaling to generate objects of varying quality. A useful property of JPEG is that the degree of lossiness can be varied by adjusting compression parameters. The quantization factor in a JPEG encoding determines the quality of the file. The higher the quantization factor, the better the quality of the file and consequently greater its file size. Conversely, a smaller quantization factor corresponds to a lower quality and thus a smaller file size. The highest value of the quantization factor is 100. This means that the image maker can trade off file size against output image quality. This property of JPEG is used to generate varying levels of quality of a given web content. We degrade the image file to various levels of quality by varying the quantization resulting in significant savings in the file size compared with the drop in perceived quality. Figure 3.3 shows the file size savings achieved by varying JPEG quality. The horizontal axis shows the percentage savings in file size while the vertical axis gives the JPEG quality factor. The graph shows an increase in the file size savings with a decrease in the JPEQ quality factor.

Continuous media files can be scaled using temporal scaling or quality scaling. Temporal scaling involves intelligently dropping frames (frames which are dependent) before sending the multimedia file. This adds some processing overhead to the server. This is because even though the frames are dropped the server still has to read through the file to get to the next frame. Thus temporal scaling reduces network load but not the processing load for the server. Quality scaling on the other hand involves reducing the quality of all the frames uniformly but sending all the frames. Video files like MPEG can be encoded with a different quality factor in order to generate a low quality file resulting in an appreciable reduction in file size without a significant change in the perceived quality. [37] shows that with a significant reduction in file size within limits there is hardly any difference in the perceived quality. Quality scaling thus allows web services to transmit variations of the same multimedia object at different sizes, allowing some control over the resources consumed in transmitting the file to the client. We chose quality scaling for scaling multimedia files.

A similar approach is adopted for scaling of MPEG files. In MPEG compression, the quantization factor or the Q scale factor determines the quality of the MPEG file. It represents a tradeoff between quality and compression. The Q scale factor is varied for the three types of frames; I, B and P individually. A large Q scale factor gives better compression but a worse quality, while smaller values give better quality

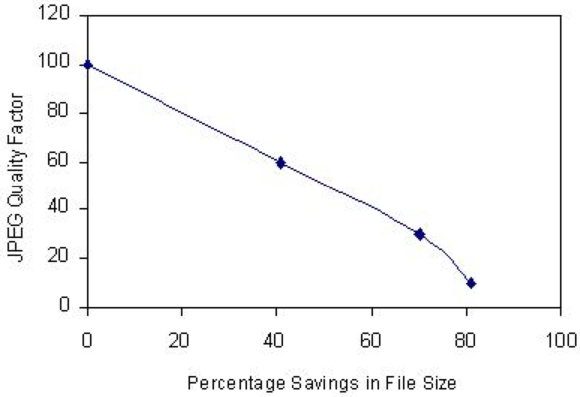

Figure 3.3: JPEG Quality vs File Size

but less compression. The Q scale factor can be varied from 1 to 32. A Q scale factor of 1 corresponds to the best quality and the Q scale factor of 32 corresponds to the worst quality or best compression.

Figure 3.4 shows the file size savings achieved by varying the Q scale factor. The horizontal axis shows the percentage savings in the file size while the vertical axis has the MPEG Q scale factor. The percentage file size savings increase almost exponentially with an increase in the quality the Q scale factor. As mentioned, the encoding with Q scale factor 1 has the highest quality factor but the least file size savings. We take the file size obtained by such an encoding as a reference. The encoding of 30 shows the maximum file size savings of about 93%.

For scaling multimedia files, we re-encode the MPEG file with a different quantization value. To decode a given MPEG file we use *mpeg_play* [38]. Mpeg_play is an

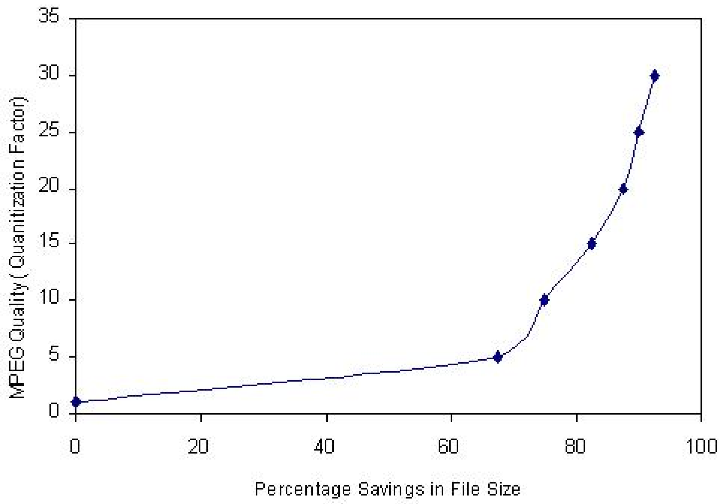

Figure 3.4: MPEG Quality vs File Size

MPEG player written in C. By default it uses X11 to display the decoded movies. It can optionally produce PPM files. We use mpeg_play to produce *ppm* files by using its -*dither ppm* option. These ppm files are then re-encoded using *mpeg_encode* [39]. Mpeg_encode produces an MPEG video stream from a parameter file and a set of ppm images. The parameter file is used to specify the quantization factor for the MPEG stream. The quantization scale values (Q scale) give a trade-off between quality and compression. The Q scale values can be set differently for I, P and B frames. The larger the numbers the better is the compression, but worser the quality. A Q scale factor of 1 corresponds to the best quality and the Q scale factor of 31 corresponds to the worst worst quality or best compression. We use these Q scale values in order to generate MPEG files of varying quality.

Chapter 4

Web Server Load Monitoring

The performance of a Web server depends on its hardware and software. The main hardware components are the processor, memory, disk and the network. Server overload may occur due to saturation of the CPU or the disk at the server or due to saturation of the communication link connecting the server to the network. An overload on any of the resources of the Web server leads to a drop in its performance. Hence, it is critical to monitor server load and identify overload so that steps can be taken to alleviate load. The most relevant components to the servers' performance are the hardware, HTTP server and the operating system.

The hardware platform that we used was an Intel Pentium III, 500 MHz with a SCSI disk and 128 MB of main memory, with a standard 10 Mbps Ethernet card. The operating system was Linux version 2.2.14. Linux is a general purpose, pre-emptive, multi-threaded operating system. Linux has been widely used as an operating system for various types of servers such as database applications, network and Web servers.

The Web server software used is Apache [14] version 1.3.12. Apache is a secure, efficient and extensible server which provides HTTP services in synchronization with the current HTTP standards. Apache can run from the inetd system daemon or in standalone mode. When running from inetd, a new copy of the server is started from scratch for each connection made to the server, resulting in high overhead per connection. Hence, we run Apache in standalone mode which exhibits much lower overhead.

4.1 Methodolgy

4.1.1 Metrics

Load on a server is often indicated by its CPU, disk and network utilization.

CPU: The time the processor is busy is broken down into two components: time spent in user mode and time spent in system mode. These two measures allow one to understand how the system behavior is dependent on the operating system implementation. Dynamic Web page generation technologies like CGI require some processing on the part of the server application. Such operations may involve querying huge database files and performing certain operations on the data and generating a Web page on the fly. This overloads both the processor and the disk. With an increase in e-commerce there has been tremendous increase in dynamic Web page generation technologies, such as FastCGI, Servlets, Server API's and Active Server pages (ASP). These technologies have made server side scripting popular and their use widespread. However, these techniques have the drawback of being computationally expensive. We consider CGI which is one of the most widely used dynamic Web page generation technologies. Previous work in [36] shows the performance of most dynamic page generation technologies are similar. The imposed overheads in CGI are mainly because of the process creation, management and synchronization necessary to create a new process in order to handle each request. Unlike static pages which involve just fetching a page and sending it over the network, dynamic pages need to carry out some processing and access large files and generate a Web page on the fly before sending it. This represents a significant overhead on the system and a heavy load on the server.

Disk: A Web server continuously receives requests for access to many different files. This leads to disk accesses for different types of files. These disk accesses are significant for multimedia files which tend to be rather large, on the order of MB's. For example, a low quality lossy compressed 640x480 pixel can take 25kB while a high quality non-lossy compressed 1280x1024 pixel image takes as much as 2.5MB. For multimedia files the disk bandwidth required is usually much larger as shown in Table 4.1.

Specifications	**Space Requirements**
Voice Quality Audio	64 Kbps
CD Quality Audio	1.4 Mbps
NTSC Quality Video	8.7 MBps
HDTV Quality Video	351 MBps

Table 4.1: Multimedia Storage Requirements

Continuous playback of a media stream consists of a sequence of periodic tasks with deadlines, where tasks correspond to retrievals of media blocks from disk and deadlines correspond to the scheduled playback times. Dynamic page generation technologies too, need to access databases and retrieve vast amount of information on the fly. Thus the disk performance has become a critical factor due to the storage intensive nature of multimedia and dynamic web page generation applications

Network: The basic function of a Web server is to serve data from the storage subsystem over the network. Since this operation involves network activity, the analysis of the utilization of the network bandwidth is important for an accurate understanding of the server behavior. Although the communication link to the server is getting faster, the network utilization nevertheless is still one of the most important parameters to determine the server load.

4.1.2 Measurement

Linux provides a virtual file system known as the *proc*[1] file system that can be used to monitor the performance of any computer in a distributed environment. The *proc* file system acts as an interface to the internal data structures in the kernel. It can be used to obtain information about the system and to change certain kernel parameters at runtime. The *proc* file system contains information for every process, the processor utilization, the internal workings of the kernel, the number of bytes read and written and also the number of packets sent over the network interfaces.

The performance values obtained from the proc file system change transiently as the system load changes. To obtain average utilization values rather than instantaneous ones we use exponential averaging.

$$\text{util} = \alpha \times \text{newutil} \; + \; (1 \; - \; \alpha) \times \text{oldutil}$$

[1]The name proc is an abbreviation for process

where *newutil* is the latest value of the utilization parameter, *oldutil* is its value x seconds before. Here, α is a constant fractional value between 0.0 and 1.0. The larger the value of α, the more quickly the average utilization is adjusted to represent the current state of server utilization. With values of α too large, the utilization value will be more influenced by the instantaneous values. On the other hand, as the value of α becomes too small, average utilization measurements adjust to their actual value too slowly, resulting in an unresponsive server. For our measure, we take the value of α as 0.40 as it seemed to be effective in providing an appropriately responsive server in our pilot studies. Similarly we use 40 seconds as our timing parameter for all the three utilization measurements, as it best captured the variation, in the utilization values during our pilot studies.

1. **CPU Utilization**

   CPU Utilization is an important capacity and performance metric of a Web server. To measure CPU utilization we made use of the *top* system command. *Top* provides an ongoing look at processor activity in real time. It displays a listing of the most CPU-intensive tasks on the system, and can provide an interactive interface for manipulating processes. It shows the percentage of CPU time in user mode, system mode, niced tasks, and idle. We measure the processor utilization as the sum of the user and system processes every 40 seconds. Our measurement monitors the following performance metrics.

   % user: The percentage of time the system spent in execution at the user (or application) level.

   % sys: The percentage of time the system spent in execution at the system (or kernel) level.

   The above performance measures can directly be converted into total CPU utilization by adding the $\%user$ and $\%sys$values. We average the CPU utilization exponentially to account for the transient spikes in the processor utilization.

2. **Network Utilization**

   Network utilization is a measure of the available bandwidth to the server. To measure network utilization of the server, we use the statistics from the *proc* file system in Linux. We use the statistics from proc file system to determine network utilization. The */proc/dev* pseudo file contains statistics such as the number of packets received / transmitted. We calculate the network utilization in terms of bytes over the theoretical maximum bandwidth available (10 Mbps) on our server.

3. **Disk Utilization**

   Disk utilization is a measure of the number of accesses made to the disk. With an increase in file size and the frequency of accesses disk utilization increases. We measure disk utilization as a percentage of the maximum disk bandwidth. In order to measure disk utilization we make use of the statistics from the */proc/stat* file in the *proc* file system. It contains the data about the number of blocks read and written. We use these numbers to determine the number of bytes read over a period of time and then calculate utilization in terms of the bytes accessed over the theoretical maximum disk bandwidth of the disk (24Mbps in our server).

4.2 Tools

This section gives a brief description of the various tools which we used for our measurements.

4.2.1 Httperf

In order to generate a high request rate we used httperf [40] a tool to measure Web server performance. *httperf* uses both the HTTP protocols, HTTP/1.0 and HTTP/1.1, and offers a variety of workload generators. The most basic operation of httperf is to generate a fixed number of HTTP GET requests and to measure how many replies (responses) came back from the server and at what rate the responses arrived. httperf prints out the overall results, including results pertaining to the TCP connections, results for the requests that were sent, results for the replies that were received, CPU and network utilization figures, as well as a summary of the errors that occurred (timeout errors are common when the server is overloaded). Appendix A explains *httperf* in detail.

4.2.2 Netperf

Netperf [41] is a benchmark that can be used to measure various aspects of networking performance. Its primary focus is on bulk data transfer and request/response performance using either TCP or UDP. We used Netperf primarily to generate background load alongwith multiple ftp and scp sessions.

4.2.3 Bonnie

Bonnie [42] performs a series of tests on a file of known size. If the size is not specified, Bonnie uses 100 MB. On a big modern server the file size should be larger than the available RAM to avoid serving the file only from the memory. Bonnie works with 64-bit pointers. For each test, Bonnie reports the bytes processed per elapsed second, per CPU second, and the % CPU usage (user and system). We use bonnie to write and read from the disk in order to vary the disk utilization.

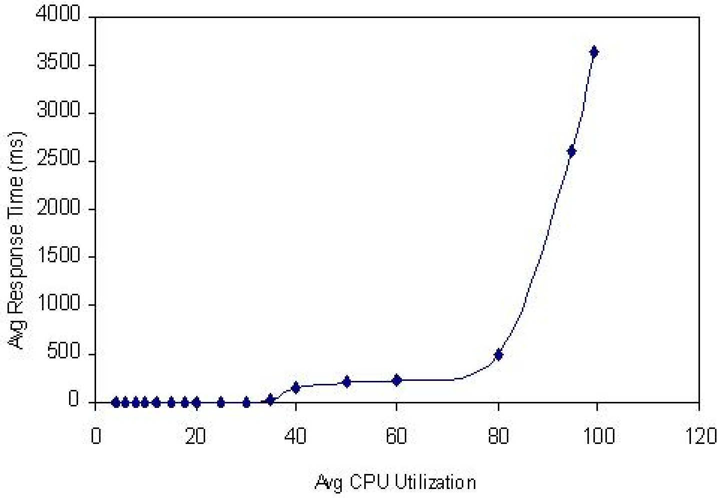

Figure 4.1: Average Response Time vs Average CPU Utilization

4.3 Results and Analysis

This section discusses the results of profiling the Web server. We measure the response time of the server by independently varying utilization of the three resources, CPU, disk and the network. We measured the response time for different values of server utilization. We use httperf [40] in order to generate requests. The file size used for these measurements is 64KB, which is the average file size on the Web [4].

In order to increase the processor utilization we load the CPU with computationally intensive jobs. These jobs were scripts which performed processor intensive work like counting to infinity to load the processor. We generated the HTTP requests and measured the mean response time with httperf. Figure 4.1 shows the relationship between the response time and CPU utilization. The horizontal axis shows the average CPU utilization for the duration of the test and the vertical axis corresponds to the response time in milliseconds. As CPU utilization increases, the

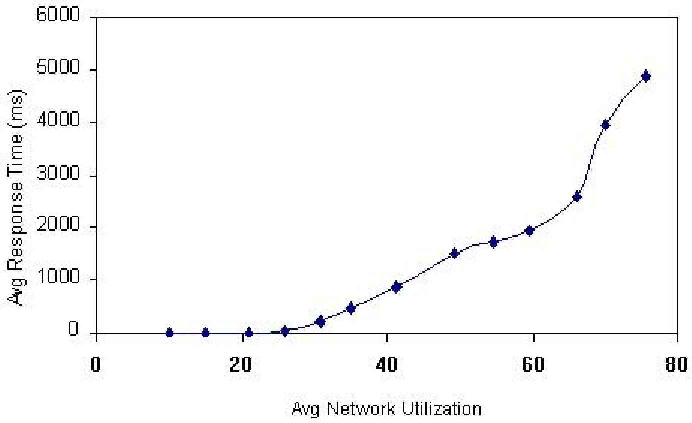

Figure 4.2: Average Response Time vs Average Network Utilization

response time also increases, and it increases exponentially as the utilization approaches 80%.

To measure the behavior of the response time with network utilization we increased the network traffic gradually. To generate network traffic, we used Netperf primarily to generate background load along with multiple FTP and SCP sessions. Figure 4.2 shows the graph for the average network utilization versus the response time. The response time increases as the network starts to become a bottleneck. At around 40% utilization the response time begins to increase exponentially.

To measure the effect of disk utilization on response time, we varied the disk utilization and measured the corresponding response time. In order to load the disk we used a program that wrote a huge file to the disk as fast as it can. We also used bonnie [42], a file system benchmark to read and write from the disk. Figure

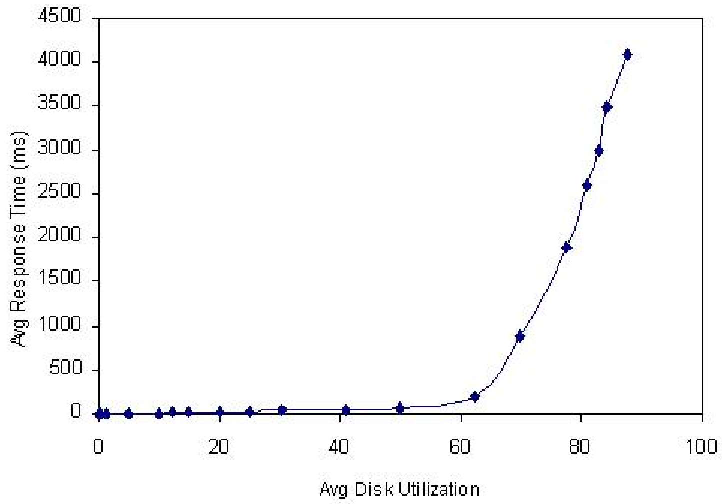

Figure 4.3: Average Response Time vs Average Disk Utilization

4.3 shows the relation between the average disk utilization and the response time. The graph is similar to the CPU utilization one, where the response time increases exponentially on reaching a threshold around 80%.

Figure 4.4 shows the relation between the response time and file sizes. The plot shows that the response time for small file sizes is almost independent of file size. For smaller file sizes, the response time is dominated by the connection setup cost. For larger file sizes the response time increases approximately linearly. Larger files reduce system performance and increase response time because serving such files uses more available bandwidth and more server processing capacity.

Our main thrust in profiling the server was to determine the thresholds at which we need to switch content. Based on the results obtained, we set two threshold values for each utilization parameter, a high threshold value and a low threshold value, to

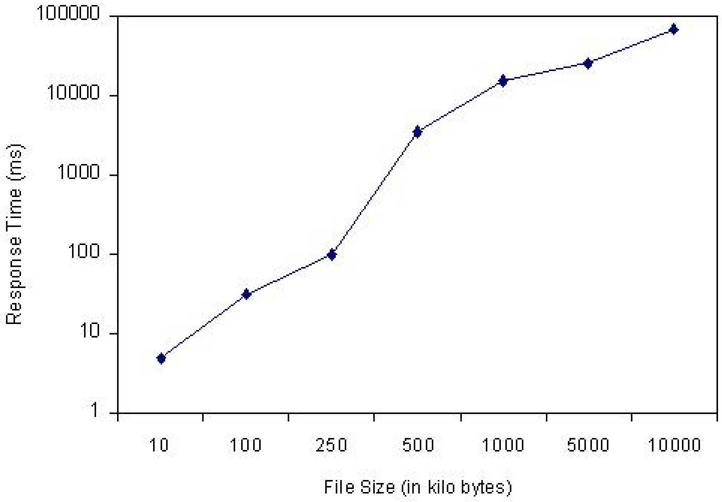

Figure 4.4: Response Time (ms) vs File Size (KB)

prevent the system from thrashing from high quality to low quality frequently. As shown from the graphs we selected the following threshold values. The response time for CPU utilization increases exponentially as the utilization of the processor approaches 80%. We chose 75 and 60 as the high and low water marks. For network utilization, we use 55 and 35 as the high and low thresholds while for disk utilization we used 75 and 60 as the threshold values. We select two threshold values; high and low to prevent the system from thrashing from one quality content to another.

Chapter 5

Web Server Performance

This chapter discusses the performance of the Web server and we compare the performance of our adaptive server with a non-adaptive one.

Web server performance depends upon several factors: hardware platform, operating system, server software, network bandwidth and workload. There are various well-known methodologies for performance evaluation of computer systems, as pointed out in [43]. For instance, a Web user's perception of performance has to do with response time and connections refused. On the other hand, a Webmaster's perception of performance is oriented towards high connection throughput and high availability [44]. We evaluate the performance for our system based on the response time, throughput, error rates and the number of concurrent multimedia and dynamic clients.

5.1 Methodology

This section describes the methodology followed in order to evaluate the performance of our adaptive server as compared to a non-adaptive server. The load monitoring module consists of four separate programs monitoring the CPU, Network and Disk utilization and a *http_ping* program which we developed for measuring the response time of the server representing the actual request load. The *http_ping* periodically sends http requests and measures the corresponding response time. The measured response time is proportional to the server's input request queue. Thus, the *http_ping* program gives a good indication of the load on the server and the length of the server queue. When the server queue is short the response time is less as compared to when the server queue is large. The experiments of server profiling described in Section 4.4 give an indication of the average response time. We take the average response time as the response time of a file of size 64KB, which is the average size of a page on the Web [45].

The adaptation module then decides the need for adaptation from the data obtained from the load monitor. The adaptation module makes a decision on the need for content switching depending on the utilization parameters and the observed request rate. Content is adapted only if the utilization values are above their respective thresholds and the observed response time is greater than the acceptable threshold set for it. There is also a need of a mechanism to switch back to higher quality content once the server utilization drops. We switch back to the full quality content if the utilization values fall below their respective thresholds.

The content selector module performs the actual content switching as indicated by the adaptation module. The content switching is a transparent operation and involves switching the symbolic links and redirecting them to point to the appropriate quality of content.

5.1.1 Metrics

Throughput at the server and *response time* are the two important metrics to analyze server performance. The rate at which http requests are serviced is known as the throughput. We measure the throughput in terms of the responses/sec from the server. The response time of a server is the time it takes from the sending of a request until the arrival of the last byte of the response. We also measure the number of errors. An error is any failure in attempting an interaction with the server. Increased errors are an indication of the degradation of server performance. Thus, an overflow on the connection queue at the server constitutes an error. This means that an attempt by the client to connect to the server will be ignored. The client tries connecting until there is available space in the queue or a times out after a predefined period.

Our test results are based on three client machines simulating multiple clients issuing uniform sized HTTP GET requests. The clients were connected to the network with a T1 link. The server was an Intel Pentium III, 500 MHz running Linux version 2.2.14 with a SCSI disk, 128 MB of main memory and a standard 10 Mbps Ethernet card

5.2 Results and Analysis

5.2.1 Server Performance

This section discusses results comparing the adaptive server with the non-adaptive server. We compare the performance for a HTTP workload, a multimedia workload and a dynamic workload.

Figure 5.1 shows the dynamic content switching operation of the system over time. This graph shows the working of the system is terms of its content switching.

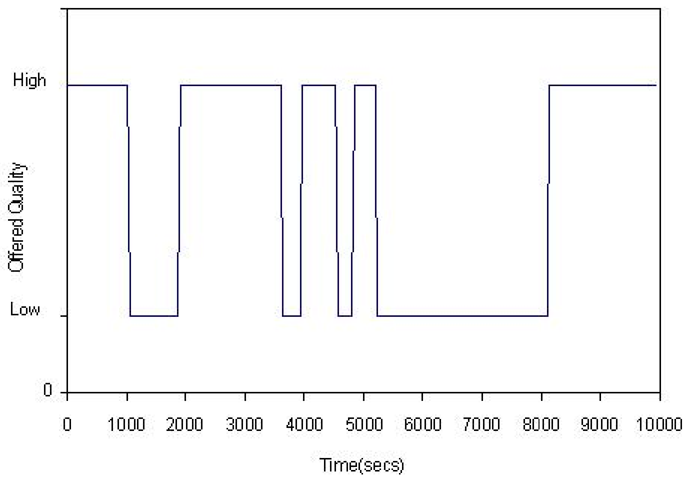

Figure 5.1: Snapshot of the Adaptive Content Delivery System

The graph in Figure 5.2 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 the system gets overloaded much later and hence acceptable response times are obtained for larger requests/secs

Figure 5.3 shows the number of responses/sec against the number of requests/sec. It shows that the requests/sec obtained from the server increases linearly with offered load until the server starts to become saturated at around 80 requests/sec. As the

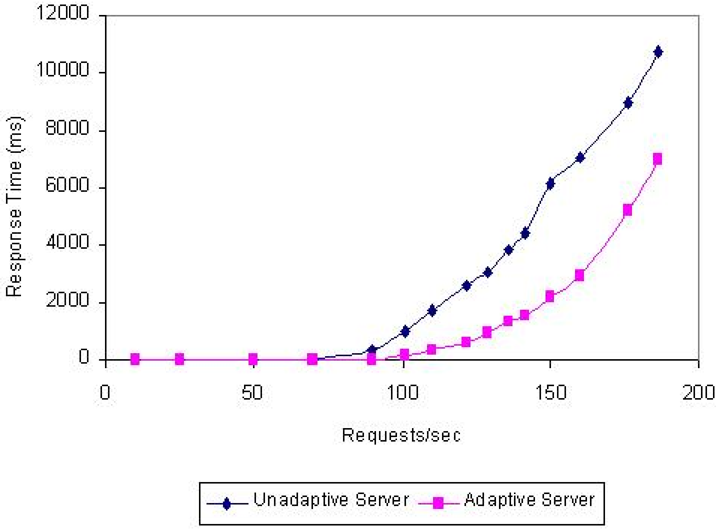

Figure 5.2: Response Time (ms) vs Requests/sec

load increases the responses/sec essentially remains constant as all other requests which the server is not able to service are rejected. The adaptive server is able to serve up to 25% more requests than the non adaptive one, because of the savings in serving the less resource intensive contents. The response rate of the adaptive server gradually falls closer to that of the non-adaptive server because of the saturation of the operating system resources.

Figure 5.4 is a graph of the number of requests that failed. As the request rate increases, the server gets saturated and the number of requests that are rejected increases. This explains the response rate dropping in Figure 5.3 as the time spent by the server in the kernel to handle calls that fail.

Figure 5.5 shows the overhead for the content adaptation system. The utilization values shown are the average values for 1000 seconds. The graph shows the CPU

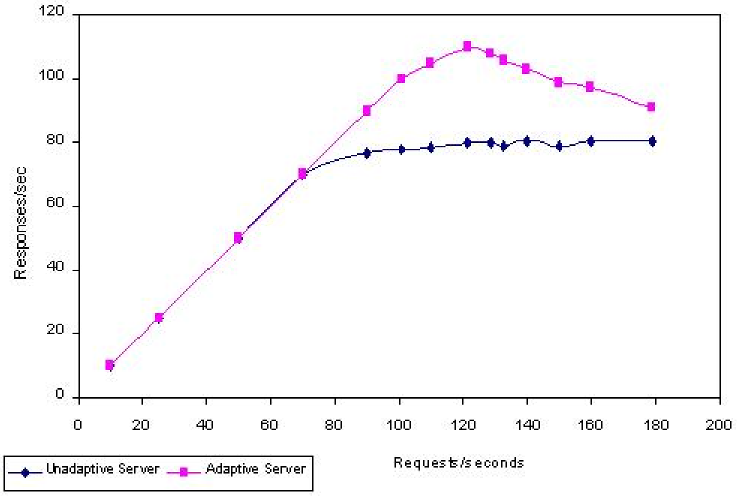

Figure 5.3: Responses/sec vs Requests/sec

utilization for the both the adaptive and non-adaptive servers, and the network and disk utilization values for the adaptive server. The values for the CPU utilization for an non-adaptive server show the minimal overhead for our system.

5.2.2 Dynamic Requests

Figure 5.6 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 the 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 a smaller, less resource intensive file.

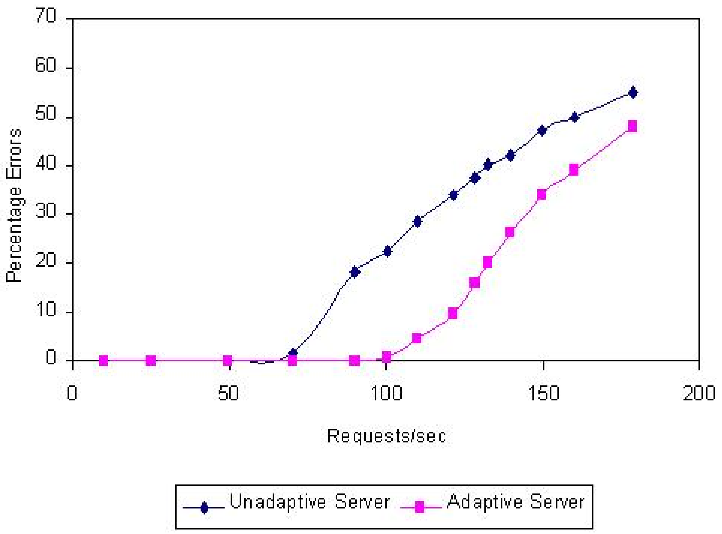

Figure 5.4: Errors vs Requests/sec

Figure 5.7 shows a graph for the failure rate vs the number of CGI clients. 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.

5.2.3 Multimedia

This section evaluates the benefits of the adaptive content delivery system for multimedia requests. We used the techniques mentioned in Section 3.2 in order to generate the varying quality content for multimedia files.

Figure 5.8 shows the graph for the frames per second sent to the client against the number of clients requesting the file. We modified a streaming MPEG server used in a previous study [46] to serve MPEG files. We built a MPEG client which

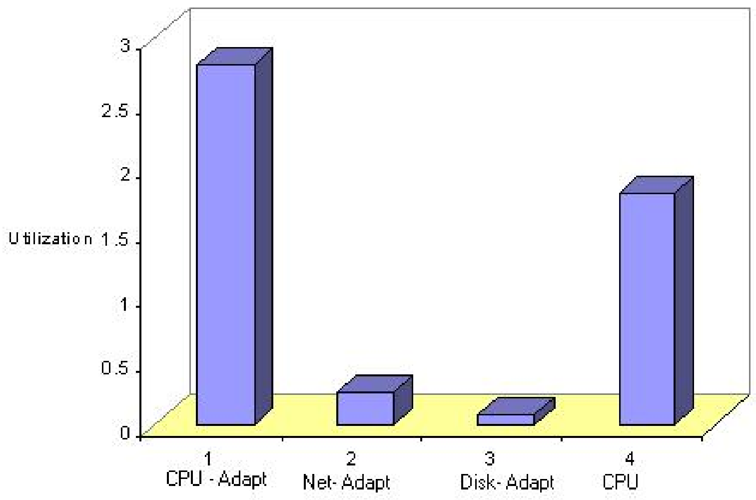

Figure 5.5: Overhead for the Adaptive Content Delivery System

connected to the streaming server. We varied the number of clients and noted the corresponding frames per second sent by the server. 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 from after the third client itself and it falls below 15 frames per second at 6 concurrent clients. On the other hand, the adaptive server can handle 6 concurrent multimedia clients and it falls below 15 frame per second only at 11 concurrent connections. The adaptation leads to the server serving a smaller video file which significantly reduces the load on the server resulting in a better frame rate for the server.

Figure 5.9 shows the variation of the CPU utilization with the number of clients. The processor utilization increases almost linearly as the number of clients increase. Since the adaptive server serves a degraded quality content, the processor utilization

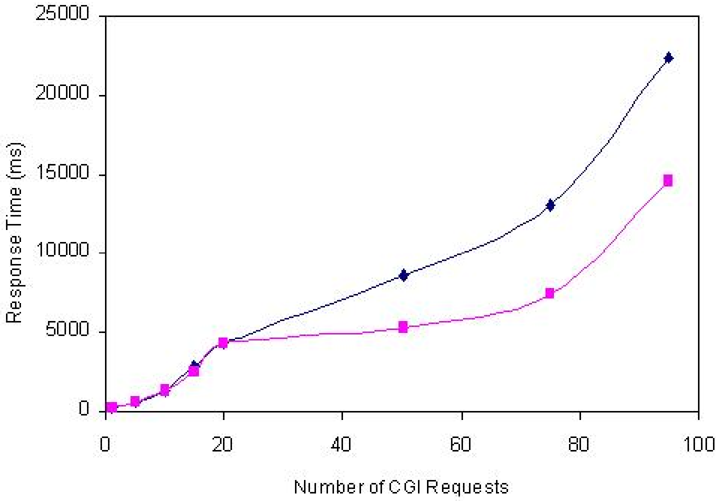

Figure 5.6: Response Time vs Number of CGI Clients

does not increase as fast as in the non-adaptive case.

Figure 5.10 shows the graph for network utilization against the number of clients. The horizontal axis represents the number of concurrent clients, while the vertical axis gives the network utilization. As shown in the figure the network utilization increases until it reaches its peak value. Once it reaches a peak value, the network utilization falls, since there is an increase in the number of packets being dropped at the network interface. Also, with an increase in the number of clients the CPU is overloaded and hence it spends most of its time switching between processes. On the other hand, the adaptive server reduces the amount of data it is sending and hence its network utilization does not reach its peak value as early as in the first case.

The graph for the disk utilization and the number of clients is shown in Figure

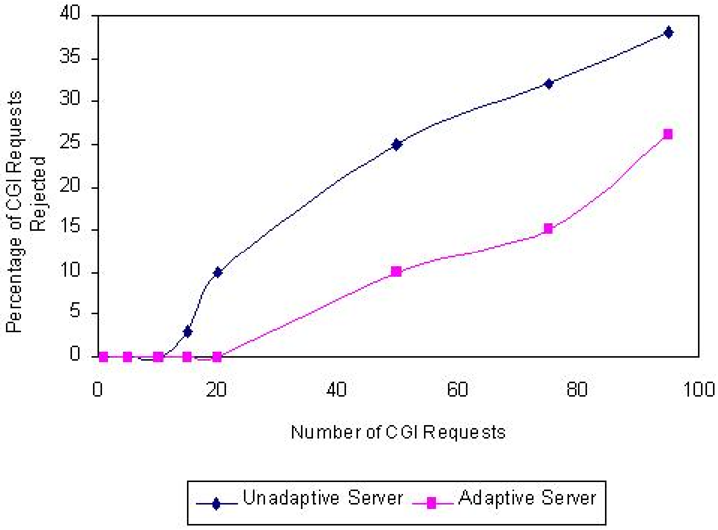

Figure 5.7: Failure Rate vs Number of CGI Clients

5.11. The disk utilization also increases almost linearly with the number of clients. As the number of clients increase, the number of accesses to the disk also increases resulting in high disk utilization. For the adaptive server, the disk utilization increases almost linearly with the number of concurrent multimedia clients. The disk utilization for the adaptive server does not follow the utilization graph of the non-adaptive server, as by the time the disk has reached 30% utilization the system has already switched content due to the high network utilization. Hence, the disk utilization of the adaptive server varies slowly as compared to the non-adaptive server.

5.2.4 Client Side Measurements on the Web

In order to measure the effect of file size variations and the benefits of our system to the client, when done over a WAN instead of just a LAN, we carried out some tests on the Web. We selected a set of 25 of the most popular sites from [3] to retrieve

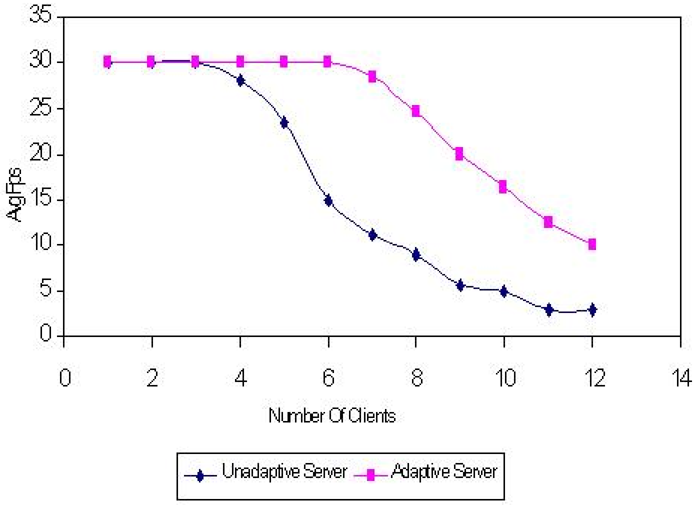

Figure 5.8: Frame Rate (fps) vs Number of Multimedia Clients

web pages from. The sites included Microsoft, CNN, Yahoo, Ebay, Aol etc.

We accessed files of varying sizes from these servers. To measure their response time, the files were retrieved using *httperf* and their corresponding response times were noted. We split the response time as the connection set up time and the transfer time. The connection set up time is the constant time it takes to set up a TCP connection, while the transfer time is the time it takes to transfer the actual data from the server to the client.

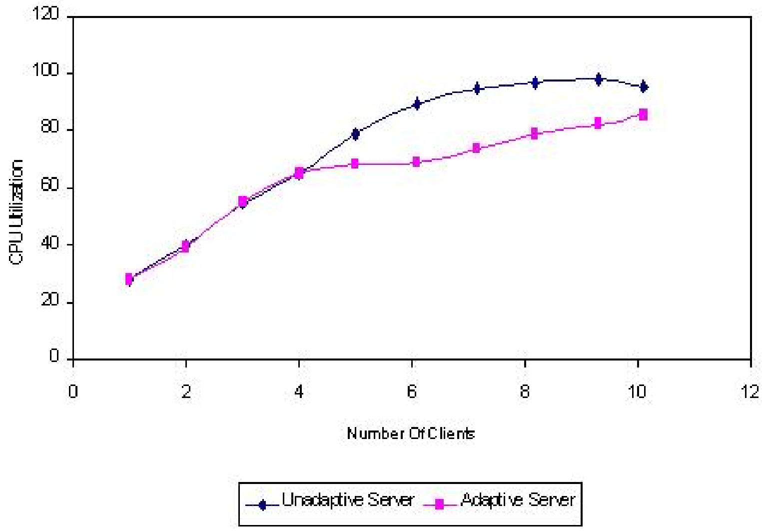

Figure 5.9: CPU Utilization vs Number of Multimedia Clients

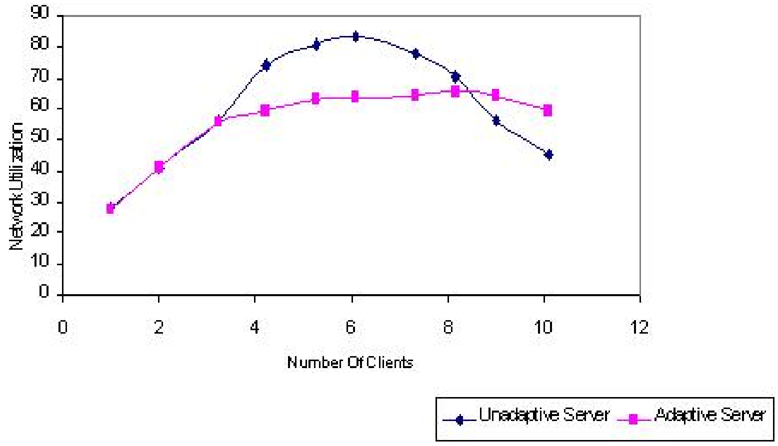

Figure 5.10: Network Utilization vs Number of Multimedia Clients

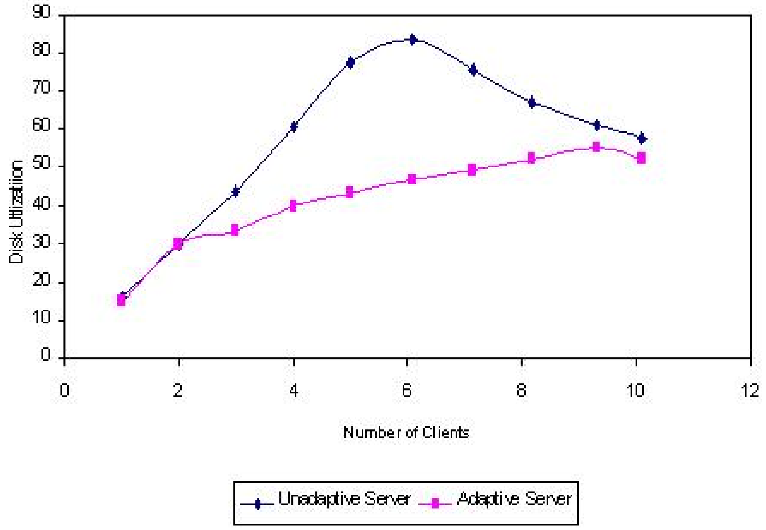

Figure 5.11: Disk Utilization vs Number of Multimedia Clients

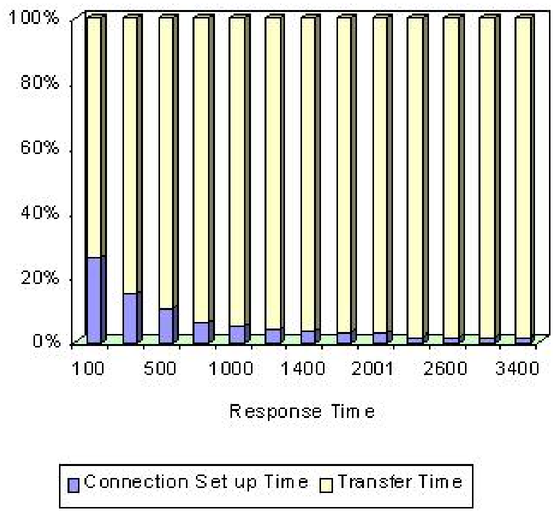

Figure 5.12: Connection Set up Time and Transfer Time as a Percentage of Response Time

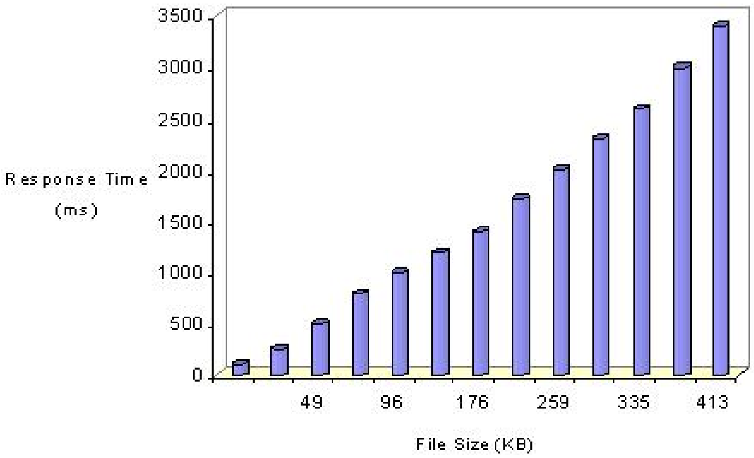

Figure 5.13: Response Time (ms) vs File Size (KB)

Chapter 6

Future Work

In this thesis, we designed a system for adapting Web content to alleviate server load. The main thrust of this thesis was adapting for multimedia and dynamic Web page generation technologies. There are still many research opportunities in the area of content adaptation.

Our content adaptation technique switches content between two levels of quality. In practice, it may be better if the quality variation was more finely grained. Work needs to be done to generate content with varying levels of quality for different overload scenarios.

There have been numerous approaches to providing QoS on the Internet. Diffserv, Intserv, MPLS talk about a guaranteed delivery between two end systems. These technologies are basically a network solution to the problem of QoS. However, network QoS with its associated packet priorities and bandwidth guarantees is ineffective when the server itself drops packets. Thus, QoS is essentially an end-to-end solution and hence there is a need to add QoS features to a Web server. One of the major advantages of using response time as a parameter to determine server load so it can be used to predict service times. Such a function could be the basis of an admission control technique used to admit only those clients which can be served within a pre-defined time limit. This gives a notion of QoS at the server. This can be of great advantage to ISP's and video servers who have clients who require guaranteed response times.

The content adaptation approach can also be used along with client resource determination techniques to deliver the appropriate quality content to the client. Our transparent content switching technique can be used to serve different quality content to heterogeneous clients like desktops, mobile phones or PDA's. Client browsers could explicitly request various quality levels from the server based on client side resources or personal preferences.

Our system has been tested only for HTTP 1.0. It would be interesting to test the impact of persistent connections using HTTP 1.1. The connection set up time is a major portion of the response time for small file sizes, so small objects can be bundled together to provide a better response time to the client.

For multimedia, we used quality scaling as our scaling technique. A comparison of other media scaling techniques can be made to find out the least resource intensive scaling technique. Lastly, all the tests for the dynamic workload we performed were with CGI. It would be interesting to measure the performance of other dynamic technologies like FastCGI, ASP's and Servlets.

Chapter 7

Conclusions

The phenomenal growth in the Internet traffic has placed a heavy load on the Web servers. Servers are now at the center of an evolving E-commerce oriented, multimedia rich communication medium. 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. These technologies have a bounded delay associated with them. Hence, serving such kind of files in a timely manner albeit at a lower quality is of primary concern.

We proposed an adaptive content delivery system to serve varying quality of content depending on the server load. We designed a system capable of quantifying the load on the server and transparently adapting the delivered content dependent the server load. We measured the load on the server in terms of its CPU, disk and network utilization. We profiled the server to determine the thresholds of each of the utilization values beyond which to switch content. We monitored the server periodically, to determine the instantaneous load and then switched content dynamically according to the utilization measures. The content switching is done transparently to both the client and the server. Our load monitoring and transparent content switching are lightweight standalone processes. We designed a streaming MPEG server and client which can react to the server load by scaling the quality of frames transmitted.

We evaluated the performance of our system for static, dynamic and multimedia workloads. 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. This increase in the throughput is obtained by serving a less resource intensive version of the data. 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. Our client-side experiments performed on the Internet show that the response time savings from our system are quite significant.

The main benefits of our approach include

- transparent content switching for content adaptation,
- alleviating server load by a graceful degradation of server performance and
- no requirement of modification to existing server software, browser or the HTTP protocol.

Appendix A

Tools Used

A.1 *httperf*

Benchmarking has been regarded as a useful approach for analyzing and predicting performance of computer systems. Several benchmarks have been proposed for measuring hardware and software speed, including compilers and operating systems. httperf is a tool to measure web server performance.

This section gives an introduction to the sample execution of httperf and an interpretation of its results.

1. **Sample execution of httperf :**

   A sample execution of httperf illustrates how to measure the request throughput of a Web server. The simplest way to achieve this is to send requests to the server at a fixed rate and to measure the rate at which replies arrive. Running the test several times and with monotonically increasing request rates, one would expect to see the reply rate level off when the server becomes satu-

rated, i.e., when it is operating at its full capacity. To execute such a test, it is necessary to invoke httperf on the client machines. Ideally, the tool should be invoked simultaneously on all clients, but as long as the test runs for several minutes, startup differences in the range of seconds do not cause significant errors in the end result.
A sample command line is shown below:

httperf –server hostname –port 80 –uri /test.html –rate 150 –num-conn 2000 –num-call 1 –timeout 5 httperf –server nile.wpi.edu –port 8080 –uri /wwwroot/filex.html –num-conns 1000 -rate 150

This command causes httperf to use the Web server on the host with IP name *nile.wpi.edu*, running at port 8080. The Web page being retrieved is *filex.html* and, in this simple test, the same page is retrieved repeatedly. The rate at which requests are issued is 150 per second. The test involves initiating a total of 1,000 TCP connections and on each connection one HTTP call is performed (a call consists of sending a request and receiving a reply). The timeout option selects the number of seconds that the client is willing to wait for a reply from the server. If this timeout expires, the tool considers the corresponding call to have failed. Note that with a total of 1,000 connections and a rate of 150 per second, the total test duration will be approximately 120 seconds, independent of what load the server can actually sustain.

2. **Results Of the Test:**

httperf –client=0/1 –server=nile.wpi.edu –port=8080 –uri=/wwwroot/file115k.html –send-buffer=4096 –recv-buffer=16384 –num-conns=1000 –num-calls=1
Maximum connects burst length: 1

Total: connections 1000 requests 1000 replies 1000 test-duration 119.179 s

Connection rate: 8.4 conn/s (119.2 ms/conn, ¡=1 concurrent connections)

Connection time [ms]: min 109.1 avg 119.2 max 246.9 median 117.5 stddev 8.1

Connection time [ms]: connect 1.1

Connection length [replies/conn]: 1.000

Request rate: 8.4 req/s (119.2 ms/req)

Request size [B]: 84.0

Reply rate [replies/s]: min 8.2 avg 8.4 max 8.6 stddev 0.1 (23 samples)

Reply time [ms]: response 2.8 transfer 115.3

Reply size [B]: header 233.0 content 117760.0 footer 0.0 (total 117993.0)

Reply status: 1xx=0 2xx=1000 3xx=0 4xx=0 5xx=0

CPU time [s]: user 44.76 system 70.53 (user 37.6% system 59.2% total 96.7%)

Net I/O: 967.5 KB/s (7.9*10$\hat{6}$ bps)

Errors: total 0 client-timo 0 socket-timo 0 connrefused 0 connreset 0

Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0

Bibliography

[1] "Scaling the Internet Web Servers."
http://www.cisco.com/warp/public/cc/pd/cxsr/400/tech/scale_wp.htm.

[2] "The Internet, Technology 1999, Analysis and Forecast, IEEE Spectrum."

[3] "Hot 100 Sites. http://www.100hot.com."

[4] "All Things Web, http://www.pantos.org/atw/35654.html."

[5] A. Ortega, F. Caringnano, S. Ayer, M. Vetterli, "Soft Caching : Web Cache Management Techniques For Images," in *Proceeding of IEEE Signal Processing Society 1997 Workshop on Multimedia Signal Processing*, June 1997.

[6] J. Mogul , K.K. Ramakrishnan , "Eliminating Receive Likelock in an Interrupt-Driven Kernel ," in *Proceedings of ACM Transcations on Computer Systems* , pp. 217–252, August 1997.

[7] T. Abdelzaher, N. Bhatti, "Web Content Adaptation to Improve Server overload Behavior," in *Proceedings of The International World Wide Web Conference*, May 1999.

[8] N. Bhatti, A. Bouch, A. Kuchinsky, "Integrating User Perceived Quality Into Web Server Design," tech. rep., January 2000.

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

[10] D. Andersen, T. McCune, "Towards a Heirarchial Scheduling System for Distributed WWW Server Cluster," in *Proceedings of The Seventh International Symposium on High Performance Distributed Computing*, July 1998.

[11] T. Abdelzaher, N. Bhatti, "Adaptive Content Delivery For Web Server QoS," in *International Workshop On Quality of Service*, June 1999.

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

[13] M. Stemm, S. Seshan, R. Katz, "Benefits of Transparent Content Negotiation in HTTP," in *Proceedings of Global Internet Mini Conference, Globecom*, November 1998.

[14] "The Apache Web Server Project. http://www.apache.org ."

[15] M . Colajanni, P. Yu, V. Cardellini , "Scheduling Algorithms For Distributed Web Server," in *Proceedings of 17th International Conference on Distributed Computing Systems*, May 1997.

[16] A. Mourad and Huiqun -Liu, "Scalable Web Server Architectures," in *Proceedings of The Second IEEE Symposium on computer and Communications*, July 1997.

[17] K. Bharat, A. Broder, "Mirror, Mirror on the Web : A Study of Host Pairs with Replicated Content," in *Proceedings of The 8th International World Wide Web Conference*, May 1999.

[18] A. Iyengar, E. MacNair and T. Nguyen, "An Analysis of Web Server Performance," in *Globecom*, vol. 3, nov 1997.

[19] P. M. K.Kant, "Scalable Internet Servers : Issues and Challenges," in *Proceedings of International Network Workshop 2001 Workshop*, March 2001.

[20] V. Almeroth, M. Ammar, and Z. Fei, "Scalable Delivery of Web Pages Using Cyclic Best-Effort (UDP) Multicast," in *Proceedings of IEEE INFOCOM*, March 1998.

[21] M. S. Schechter, M. Krishnan, "Using Path Profiles to Predict http Requests," in *Proceedings of Seventh International World Wide Web Conference*, April 1998.

[22] A. Fox, S. Gribble, Y. Chawathe and E. Brewer, "Adapting to Network and Client variation using active proxies:lessons and perspectives," in *IEEE Personal Communications*, pp. 8 – 17, Dec 1998.

[23] Wei-Ying Ma, I. Bedner, G. Chang, A. Kuchinsky, H.Zhang , "A Framework for Adaptive Content Delivery in Heterogeneous Network Environments ," in *Hewlett Packard Labs*, May 2000.

[24] "World Wide Web Consortium, Web Content Accessibility Guidelines 1.0." http://www.w3.org/TR/WAI-WEBCONTENT/.

[25] "JPEG : Joint Pictures Expert Group. http://www.jpeg.org ."

[26] "GIF : Graphics Interchange Format . http://www.gifworks.com ."

[27] "MPEG : Moving Pictures Expert Group. http://www.cselt.it/mpeg."

[28] S. McCanne, M. Vetterli, V. Jacobson, "Low Complexity video Coding For Receiver Driven Layered Multicast," in *IEEE JSAC*, August 1997.

[29] " HP OpenPix. http ://image.hp.com."

[30] L. Bouthilliere, "Synchronized Multimedia on the Web ," Web Techniques Magazine.

[31] "W3C Synchronized Multimedia Integration Language." http://www.w3.org/TR/REC-smil/.

[32] "W3C Extensible Markup Language (XML). http://www.w3.org/TR/REC-xml ."

[33] "IETF Working Group on Content Negotiation." http://www.ietf.org/html.chaters/conneg-charter.html.

[34] "W3C Composite Capability/ Preference Profiles (CC/PP)." http://www.w3.org/TR/NOTE-CCPP/.

[35] "W3C Resource Description Framework(RDF) Schema Specification." http://www.w3.org/TR/PR-rdf-schema.

[36] M. Claypool, B. Kothari , "Performance Analysis of Dynamic Web Page Generation Technologies," in *Proceedings of The International Network Conference*, July 2000.

[37] M. Claypool, Y. Liu, "Using Redundancy To Repair Video Damaged By Network Data Loss ," in *Proceedings of ACM Multimedia Computing and Networking* , January 2000.

[38] "The Berkeley MPEG Player."
http://bmrc.berkeley.edu/frame/research/mpeg/mpeg_play.html.

[39] "The Parallel Berkeley Encoder."
http://bmrc.berkeley.edu/frame/research/mpeg/mpeg_encode.html.

[40] D. Mosberger, T. Jin, "httperf - A Tool for Measuring Web Server Performance," in *First Workshop on Internet Server Performance*, pp. 59—67, ACM, June 1998.

[41] Information Networks Division,HP, "Netperf - A Network Performance Benchmark." http://www.netperf.org/netperf/NetperfPage.html.

[42] T. Bray, "Bonnie - A File Server Benchmark."
http://www.textuality.com/bonnie/index.html.

[43] D. Menasce, V. Almeida, L. Dowdy, *Capacity Planning and Performance Modelling.* Englewood Cliffs, N.J : Prentice Hall , 1994.

[44] J. Almeida, V. Almeida, D.Yates, "WebMonitor : Tool for Measuring World Wide Web Performance." http://www.firstmonday.dk/issues/issue2_7/almeida/.

[45] J. Neilsen, *Usability Engineering.* San Fancisco, Ca: Morgan Kaufmann, 1994.

[46] J. Brzozoski and R. McDonald, "MPEG Jitter," Tech. Rep. Major Qualifying Project MQP-MLC-MJ98, Worcester Polytechnic Institute, May 1999. Advisor Mark Claypool.

[47] J. Almeida, V. Almeida, D. Yates, "Measuring The Behavior of a World Wide Web Server," in *Proceedings of Seventh Conference on High Performance Networking (HPN)*, pp. 57–72, April 1997.

[48] V. Almeida, A. Bestravos, M.Crovella, A. Oliveria , "Characterizing Reference Locality in the WWW," in *Proceedings of IEEE-ACM PDIS'96*, December 1996.

[49] M. Arlitt, C. Williamson, "Web Server Workload Characterization," in *Proceedings of SIGMETRICS Conference on Measurement and Modelling of Computer Systems*, May 1996.

[50] M. Crovella, A. Bestravos, "Self Similarity in World Wide Web," in *Proceedings of SIGMETRICS Conference on Measurement and Modelling of Computer Systems*, May 1996.

[51] J. Mogul, "Network Behavior of A Busy Web Server and its Clients ," tech. rep., October 1995.

[52] J. Mogul, "Operating System Support for busy Internet Servers," in *Proceedings of the Fifth Workshop on Hot Topics in Operating Systems*, May 1995.

[53] T. Abdelzaher, K. Shin, "QoS Provisioning with qContracts in Web and Multimedia Servers," in *Proceedings of IEEE Real Time Systems Symposium*, December 1999.

[54] N. Bhatti ,R. Friedrich, "Web Server Support for Tiered Services ," in *IEEE Network*, Oct 1999.

[55] "Internet Content Adaptation Protocol. http://www.icap.com."

[56] Binzhang Liu, "Characterizing Web Response Time," Master's thesis, April 1998. Virginia Polytechnic & State University.

[57] "Nortel Networks. http://www.nortelnetworks.com."

[58] "Cisco Systems. http://www.cisco.com."

[59] "Quickweb : Intel Corporation. http://www.intel.com/quickweb."

[60] "Inktomi Corporation. http://www.inktomi.com/products/network/."

[61] "Akamai Techologies. http://www.akamai.com."

[62] M. Ammar, K. Almeroth, R. Clark, and Z. Fei, "Multicast Delivery of Web Pages," in *Proceedings of the Workshop on Internet Server Performance*, June 1998.