DiffServer: Application Level Differentiated Services for Web Servers

Gautam Rao
San Jose, CA

Byrav Ramamurthy
University of Nebraska-Lincoln, bramamurthy2@unl.edu

IEEE International Conference on Communications, 2001. ICC 2001.
Volume 5, Digital Object Identifier: 10.1109/ICC.2001.937196
Publication Year: 2001 , Page(s): 1633 - 1637 vol.5

DiffServer: Application Level Differentiated Services for Web Servers

Gautam Rao*
Cisco Systems, 125 West Tasman Dr.
San Jose, CA 95134

Byrav Ramamurthy
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588-0115
Email: grao@cisco.com, byrav@cse.unl.edu

ABSTRACT

Web content hosting, in which a Web server stores and provides Web access to documents for different customers, is becoming increasingly common. For example, a web server can host webpages for several different companies and individuals. Traditionally, Web Service Providers (WSPs) provide all customers with the same level of performance (best-effort service). Most service differentiation has been in the pricing structure (individual vs. business rates) or the connectivity type (dial-up access vs. leased line, etc.). This report presents *DiffServer*, a program that implements two simple, server-side, application-level mechanisms (server-centric and client-centric) to provide different levels of web service. The results of the experiments show that there is not much overhead due to the addition of this additional layer of abstraction between the client and the Apache web server under light load conditions. Also, the average waiting time for high priority requests decreases significantly after they are assigned priorities as compared to a FIFO approach.

1. Introduction

Due to the enormous growth of the World Wide Web and the ever-increasing resource demands on the servers, Web content hosting is an increasingly common practice. The continuous growth of the Internet and emerging multimedia applications place demands for higher bandwidth on the Web Service Providers (WSPs). Companies expect that the requests from their potential clients (users that access their web pages) are serviced with a quality proportional to the amount of money they pay for these hosting services. Also, the system administrators may choose to give preferential services to requests from certain hosts and domains. As a result, WSPs are finding it necessary to offer their customers alternative levels of service. Besides meeting new customer expectations, this allows the WSP's to improve their revenues through premium pricing and competitive differentiation of service offerings, which in turn can fund the necessary expansion of the network. There is a need to improve the web hosting technology in terms of performance, scalability and delivery of new functionalities. Quality of Service (QoS) has become a rallying cry for all forms of communications on the Internet.

Three basic levels of end-to-end QoS can be provided across a heterogeneous network:

- **Best-effort service**---Best-effort service is basic connectivity with no guarantees (lack of QoS).
- **Differentiated service (also called soft QoS)**---Some traffic is treated better than the rest (faster handling, more bandwidth on average, lower loss rate on average).
- **Guaranteed service (also called hard QoS)**---An absolute reservation of network resources for specific traffic.

Deciding which type of service is appropriate to deploy in the network depends on several factors such as-

1. The application or problem the customer is trying to solve.
2. The rate at which customers can realistically upgrade their infrastructures.
3. Last, but not the least, the cost involved in implementing these services.

In general, the cost of implementing and deploying guaranteed service is likely to be more than that for differentiated service.

There is a clear need for relatively simple and coarse methods of providing differentiated classes of service for Internet traffic, to support various types of applications, and specific business requirements. This paper presents, *DiffServer*, a program that implements two simple, server-side, application-level mechanisms (server-centric and client-centric) to provide different levels of web service. The rest of

* This work was performed while the author was a graduate student at the University of Nebraska-Lincoln

this paper is organized as follows: section 2 discusses background and previous work. Section 3 discusses our implementation, including the *DiffServer* model, priority determination and the program flow. Section 4 examines the performance and presents the results from our experiments of the *DiffServer*. Finally, section 5 presents our conclusions and future work.

2. Background and Previous Work

The behavior of HTTP servers is quite unpredictable in cases where there is a large burst of requests. Apart from the fact that under such situations the servers tend to drop requests indiscriminately, the requests to popular pages have a tendency to overwhelm the requests for others, possibly more important pages. Apache [1], one of the most used Web servers, handles incoming requests in a first-come, first-served (FCFS) manner. Furthermore, most implementations of HTTP servers do not perform any discrimination between requests. Therefore a site cannot enforce any kind of priority scheme it may wish to implement. Since the possibility of the backbone providing differentiated services is becoming all the more promising these days, it is imperative that the end-servers also provide differentiated services to circumvent some of the typical problems such as indiscriminate dropping of requests, overwhelming of certain requests [2] etc.

J. Almeida, M. Dabu, A. Manikutty and P. Cao [4] have explored priority-based request scheduling by providing differentiated levels of service at both the user and kernel levels. They found that simple strategies such as controlling the number of processes can improve the response time of high-priority requests notably while preserving the system throughput. They also found that the kernel-level approach tends to penalize low-priority requests less significantly than the user-level approach, while improving the performance of high-priority requests similarly.

R. Pandey, J.F. Barnes and R. Olsson [2] have presented a notion of a quality of service (QoS) model that enables a site to customize how an HTTP server should respond to external requests by setting priorities among page requests and allocating server resources. As part of the QoS model, they have devised a notation, which they call WebQoSL that supports specifications such as

i. Allocation of a specific and relative amount of server resources to specific page requests.
ii. Availability of groups of pages at all times.
iii. Time-based and link-relation-based allocation of resources.
iv. Specification of guarantees about byte transfer and page request rates.

X. Chen and P. Mohapatra [5] studied approaches and performance issues in providing differentiated services from an Internet server. In their experiments on the effectiveness of priority based scheduling they found that performance degradation of high priority tasks happened at a much higher utilization compared to the non-priority-based model. Also, high priority requests incurred low delay even when the system approached full utilization. Their studies proved that under near-saturation of web server utilization, differentiated services provide significantly better services to high priority tasks compared to a traditional web server.

Our *DiffServer* implementation runs in application space and hence is very easy to set up and run. It eliminates the need to deploy and provision expensive network devices like routers, which are very challenging tasks. Also, unlike the approaches in [2], [4], [5] no modifications have been made to either the client (Netscape Navigator, Internet Explorer) or the Apache Web Server source code. Section 3 deals in detail with the model and program flow of *DiffServer*.

3. Design and Implementation of DiffServer

We implemented a user-level program, *DiffServer* [17], which runs in application space and acts as a module around the web server. Our implementation included both a server-centric as well as a client-centric differentiated services scheme.

A priority scheme based on the hostname and domain of the client (client-centric) is enforced on the incoming burst of requests. For each set of incoming requests, a file name based priority (server-centric) scheme is also enforced. Both these priority orders are "hard coded" into a configuration file named `options.conf`. Essentially the *DiffServer* architecture involves three kinds of threads, a parent thread, Child threads and a Scheduler thread. The parent first creates a Scheduler thread, which later picks a Child Thread from the pool to handle an incoming request. The parent thread assigns priorities to the incoming requests according to the criteria listed in the `options.conf` file and then the requests are inserted into a queue in the ascending order of priorities. The requests are assigned priorities based on the document they are requesting as well the hostname and domain of the source of the requests. The scheduling algorithm is a non-preemptive scheme, since at

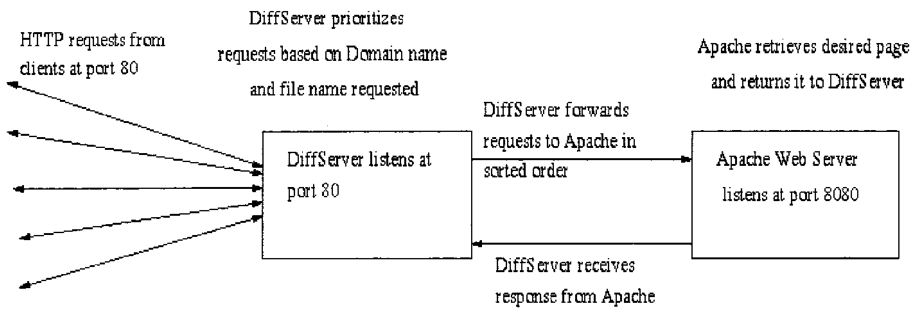
HTTP requests from clients at port 80
DiffServer prioritizes requests based on Domain name and file name requested
DiffServer listens at port 80
DiffServer forwards requests to Apache in sorted order
Apache retrieves desired page and returns it to DiffServer
Apache Web Server listens at port 8080
DiffServer receives response from Apache

Fig. 1 : Conceptual Model of the DiffServer

the user level we cannot interrupt a running process and block it in order to allow a new process to run.

3.1 Design of the DiffServer Architecture

Fig. 1 shows the conceptual model of the *DiffServer*. The *DiffServer* module listens on port 80 for requests coming in from the clients. When a request is received, it is assigned a priority based on its domain (client-centric approach) as well the filename requested (server-centric approach). The position of the request in the queue is determined by its priority. The Apache web server listens on port 8080 for the sorted requests that are forwarded by the *DiffServer* module. Apache returns the page requested to *DiffServer*, which in turn returns the response to the client.

3.2 Priority Determination

The system administrator assigns priorities to the domains and files in the `options.conf` file. The priorities are assigned on a scale of 1 (Lowest) to 10 (Highest). They may also specify any subset of the *hostname.domain* entry in the URL, for example, entries in the `options.conf` file could be of the form:

andes.unl.edu	10
unl.edu	8.5
edu	5.1

Hence, the administrators can provide differentiated services to a very fine granularity since they are not just restricted to a domain name when specifying priorities but can also specify a host or machine name within that domain. Priorities are also assigned to filenames and directories of the requested documents. For example,

/images/background.gif	9.5
/images/list.html	7
/images/	4
/index.html	8

In the above example, the file *background.gif* and *list.html* in the */images* directory are treated differently (assigned priorities 9.5 and 7 respectively) than any other files under */images* (assigned priority 4). If a priority value is not entered next to the domain or file entry, it is assigned the lowest priority of 1. If a match is not found in the `options.conf` file, the request is assigned the lowest priority of 1. The final priority of the request is determined by its *domain based priority* and *file based priority*. The administrator specifies a DOMAIN_NAME_FACTOR (DNF) and FILE_NAME_FACTOR (FNF), which are multiplied by the domain based priority, and the file based priority respectively to compute the *Overall Priority* of the request as given below. This *Overall priority* determines the request's position in the queue.

*Overall priority = DNF * domain priority + FNF * file priority*

4. Evaluation

4.1 Experimental Setup

To generate the WWW workload, we used *httperf* [10], a configurable load generator from Hewlett-Packard. It provides a flexible facility for generating various HTTP workloads and for measuring server performance. The development and experiments for this project was carried out in the Advanced Networking and Distributed Experimental Systems (ANDES) Lab in the Department of Computer Science and Engineering (CSE) at the University of Nebraska-Lincoln (UNL). The network used for our tests was a 100 Mb shared Ethernet with a network file system. Each of the client nodes that ran *httperf* was an Intel Pentium II 266 MHz or AMD 400 MHz with 128 MB of RAM running Red Hat Linux 5.2 with version 2.2.10 of the Linux kernel. The *DiffServer* and the Apache web server were run on a Pentium Celeron 333 MHz with 256 MB RAM. We initiated sessions of 1500 connections each with both HTML (static) as well as CGI (dynamic) content. The average HTML file sizes were 6401 bytes whereas the average CGI file sizes were 2204 bytes for the experiments. Testing with dynamic files is necessary since more and more dynamic content is appearing on the Web. In order to evaluate the overhead of the search and insertion of the request in the sorted linked list, we ran test cases for the *DiffServer*'s scheduling algorithm both as a FIFO list as well as a sorted priority list. A server access log

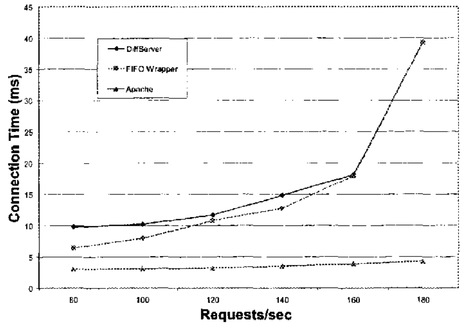

Fig. 2: Comparison of Connection Times for HTML Connections

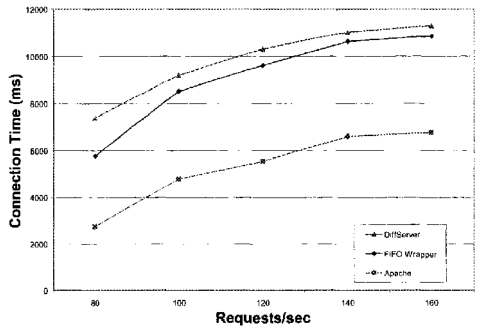

Fig. 3: Comparison of Connection times for cgi-bin connections

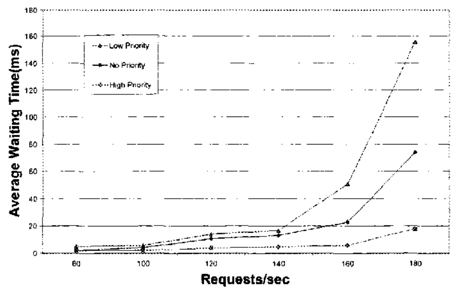

Fig. 4: Comparison of Average Waiting Time for File based Priority

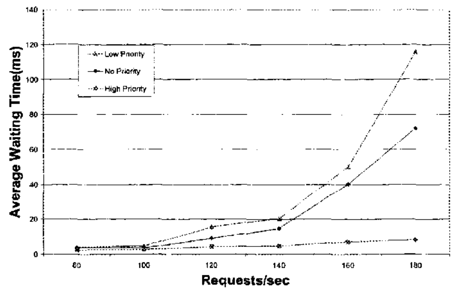

Fig. 5: Comparison of Average Waiting Time for Domain based Priority

file from the ANDES web server provided us with a trace that was used by *httperf* to perform our experiments for different rates of requests/sec.

The graphs below (see Figures 2, 3, 4 and 5) show the results obtained. Fig. 2 depicts the comparisons of the HTML connection times obtained for an Apache web server, a FIFO wrapper* and the *DiffServer*. Fig.3 depicts the same for cgi-bin connections. We notice that the Connection time is much higher in the CGI case, since a CGI program runs as a separate process in the server machine every time a CGI document is requested and therefore is very costly. For HTML connections for low values of requests/sec (<160) the FIFO wrapper has better connection and response times than the *DiffServer*. However, as the requests/sec increase, the two curves practically overlap which indicates that all the *Child* threads are busy servicing requests hence they have to wait until one becomes available. Figures 4 and 5 compare the average waiting times for file based and domain based priorities. We compared the average waiting time for a request in the queue if it is serviced in the traditional FIFO way (no priority assigned to requests) against the average waiting time after it is assigned a priority based on the domain name and filename. The average waiting time for a high priority request in the queue was significantly decreased (76% and 88% for file based and domain based priority respectively). The average waiting time for low priority requests increases by 115% (file based priority) and 60% (domain based priority) as compared to the case where no differentiated QoS policy is used. In both cases, as the load (requests/sec) increases the curve for low priority requests rises sharply showing an increase in waiting time whereas the waiting time for high priority requests is not severely affected.

5. Conclusions and Future Work

The *DiffServer* architecture presented in this report is a multi-threaded, application-space and very scalable module. *DiffServer* is very easy to set up and run and also has user configurable QoS parameters. This project implements an original approach to provide differentiated services. The results of the experiments show that there is not much overhead due to the addition of this additional layer of abstraction between the client and the Apache web server under light load conditions. Also, the average waiting time

* The FIFO wrapper is an application-level program like the *DiffServer*, but handles the incoming requests differently. The requests are forwarded to the Apache web server on a first-come first-served basis unlike the *DiffServer*, which assigns priorities to the incoming requests and then sorts them before forwarding them to Apache.

for high priority requests decreases significantly after they are assigned higher priorities as compared to a FIFO approach.

This *DiffServer* implementation has opened vistas for further enhancements. One area of research would be to explore the possibility of a weighted priority queue to ensure that all requests are handled and no starvation occurs. In this paper, we limit our investigation to Web server systems only, without addressing the different services issues in the internetworking infrastructure. In other words, we assume that the order in which the request packets arrive at the server and in which the packets are transmitted over the Internet are completely out of the control of the Web server. Obviously, a complete solution for this type of networking quality of service would require a combination of networking differentiated services [7] [8] and differentiated services supported by the *DiffServer*. Web servers now support several means to create dynamic content (e.g. CGI, FastCGI, vendor-dependent Web server APIs and Java servlets). These methods involve complex computation on the Web server and hence slow down its Connection time and Response time. Hence, in cases like this where Web server processing per request becomes more time-consuming, our differentiated services approach can be complemented with a Web Clustering approach [11] [12] [13], which could help alleviate the problem.

References:

[1] The Apache Server Foundation. URL:http://www.apache.org

[2] R. Pandey, J.F. Barnes and R. Olsson, "Supporting Quality of Service in HTTP servers", *Proc., Seventeenth Annual SIGACT-SIGOPS Symposium on Principles of Distributed Computing,* Puerto Vallarta, Mexico, June 1998.

[3] T.S. Sankaravadivelu, "Quibs: Incorporating QoS into Web Surfing", *A Project Report* for the Advanced Computer Networks course, Dept. of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, December 1999.

[4] J. Almeida, M. Dabu, A. Manikutty and P. Cao, "Providing differentiated levels of service in web content hosting", *1998 Workshop on Internet Server Performance*, Madison, Wisconsin, June 1998.

[5] X. Chen and P. Mohapatra, "Providing Differentiated Services from an Internet Server", *1998 Workshop on Internet Server Performance,* Madison, Wisconsin, June 1998.

[6] Differentiated Services Working Group (diffserv) URL:http://www.ietf.org/html.charters/ diffserv-charter.html

[7] Y. Bernet, J. Binder, S. Blake, M. Carlson, B. Carpenter, S. Keshav, E. Davies, B. Ohlman, D. Verma, Z. Wang and W. Weiss, "A Framework for Differentiated Services", IETF Internet Draft, February 1999. URL:http://search.ietf.org/internet-drafts/ draft-ietf-diffserv-model-02.txt

[8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentiated Services", *RFC 2475*, December 1998. URL:http://www.hypermail.org/rfcs/ rfc2475.html

[9] B. Lewis and D. Berg, *Multithreaded Programming with Pthreads*, New Jersey, Prentice Hall, December 1997.

[10] D. Mosberger and T. Jin, "httperf - A tool for measuring web server performance". URL:http://www.hpl.hp.com/personal/David_Mosberger /httperf.html

[11] X. Gan, "A Prototype of a Web Server Clustering System", *M.S. Project Report*, Dept. of Computer Science and Engineering, University of Nebraska - Lincoln, April 1999.

[12] X. Gan, T. Schroeder, S. Goddard and B. Ramamurthy, "LSMAC vs. LSNAT: Scalable Cluster-based Web Servers", *Cluster Computing: The Journal of Networks, Software Tools and Applications,* 2000.

[13] T. Schroeder, S. Goddard, B. Ramamurthy, "Scalable Web Server Clustering Technologies", *IEEE Network Magazine (Special issue on Web Performance),* vol. 14, May 2000.

[14] X. Gan and B. Ramamurthy, "LSMAC: An Improved Load Sharing Network Service Dispatcher", *The World Wide Web Journal*, vol. 3, 2000.

[15] L Eggert and J. Heidemann, "Application-Level Differentiated Services for Web Servers", *World Wide Web Journal*, vol. 3, 1999.

[16] R. Stevens, *Unix Network Programming*, vols. I & II, New Jersey, Prentice Hall, December 1998.

[17] G. Rao, "DiffServer : Application Level Differentiated Services for Web Servers", *M.S. Project Report*, Dept. of Computer Science and Engineering, University of Nebraska - Lincoln, May 2000.

[18] Beej's Guide to Network Programming. URL:http://www.ecst.csuchico.edu/~beej/ guide/net/

[19] A.D. Marshall, "Programming in C, UNIX System Calls and Subroutines using C", March 1999. URL:http://www.cm.cf.ac.uk/Dave/C/ CE.html

[20] D. Butenhof, *Programming With Posix Threads*, Addison-Wesley Professional Computing Series, May 1997.