Performance Evaluation of a Cache Server: A Service Differentiation study in a DiffServ Network

Iran C. Abrão[1], Marcos José Santana[2], Regina H. C. Santana[2], Júlio Cézar Estrella[2]
[1] Pontifícia Universidade Católica de Minas Gerais
Poços de Caldas, Minas Gerais, Brasil
iran@pucpcaldas.br
[2]Universidade de São Paulo – Instituto de Ciências Matemáticas e de Computação
São Carlos, São Paulo, Brasil
{mjs,rcs,jcesar}@icmc.usp.br

Abstract. This paper addressed the impact observed on performance when traditional cache servers are used in Web environments with differentiated services on the network level. It was introduced a cache server, supporting service differentiation, where several environment arrangements are tested allowing accounting the gains realized with the different configurations. The system is modeled and tested through an event-oriented simulation. The results show that the introduction of QoS is needed on different levels of the web structure, such as network and application levels, and that disconnected QoS solutions may not generate the expected results.

Keywords: Differentiation of Services, Quality of Service, Web Cache Server.

1. Introduction

The Internet best-effort [1] service model has presented signs of strangulation due to the Internet growth in recent years. According to the best effort model, the whole traffic is treated in a uniform manner, without any kind of differentiation or priority. This trend also reflects on critical web services projects, whose servers mostly deal with applications of customers according to the philosophy that, the first to come will be the first to be served.

The introduction of Quality of Service - QoS - on the Internet is a need and a requirement for companies offering services for the Web [2]. QoS is needed on the different levels of the Web structure, such as network and application levels. Disconnected solutions of QoS may not generate the expected effects, because all network elements must provide a differentiated control to avoid congestion points, which may degrade the whole system performance.

This paper evaluates the use of a modified cache server to support environments with service differentiation at the network level, named Cache with DiFferentiation of services (CDF). This cache server analyzes the type of service requested by clients and, if necessary, makes a request to the Web Server for the same type of service.

Three scenarios are analyzed: the first one with no cache, the second one with a traditional cache and the last one with the CDF.

In this paper, considers four classes of users that have different service priorities in order to evaluate the differentiation mechanisms during the execution of user requests. A discrete event simulation, built by using the OPNET Modeler environment [3] to measure the results obtained with the Web model proposed has been chosen.

The remaining parts of this paper are organized as follows. Section 2 provides a brief overview of quality of service concepts; section 3 discusses the use of web cache servers and in section 4 it is introduced the CDF server model. Section 5 presents the Web environment model proposed in this work. Section 6 shows the simulation results and finally, in section 7 the main conclusions of this study are presented.

2. Related Work

Quality of Service (QoS) can be defined as the ability to provide to a network element (application, client, server or router) some level of assurance that its traffic and service requirements will be fulfilled [4]. Providing QoS is not a trivial task in small and proprietary systems and it is still more difficult in a global system as Internet. To enable QoS, the cooperation of all network layers from top-to-bottom is required, as well as every network element from end-to-end. Any QoS assurances are only as good as the weakest link in the "chain" between sender and receiver [4]. It is important to emphasize that the use of QoS does not increase bandwidth, because the network cannot provide what it does not have. QoS only manages the existing bandwidth according to application demands and network management settings. A network able to provide QoS will continue supporting the best-effort traffic, however part of the bandwidth will be reserved for applications of higher priority.

In this work a computer networking architecture with Differentiation of Services - DiffServ [5] is used. The DiffServ approach is based on the idea of flow aggregation in a few classes of services. The DiffServ network provides local differentiation of service for large flows of traffic. The marking of packages as final hosts and edge router is done at network ingress points. In this way, one of the basic principles of Internet project that still holds is to put the complexity at the network border. A router without DiffServ capacity will dispatch the package maintaining the previous marking.

Since the beginning of web, information traffic is growing every day and the Internet overload can be noticed by the high amount of time spent during the information recovery, as well as in the degradation of the system performance due to redundant traffic. To improve the system performance, different techniques have been used, such as the adoption of cache servers. The main objective of the caches in the Web is the storing of frequently used documents for future use, so that data recovery from the original server is not necessary when such documents are requested. The cache server reduces data access time, leaving data as close as possible to customers. The use of web cache servers is one of the most popular methods to try to achieve good environment performance. However, in general, the hit rate of a simple cache server is not greater than 40% [6].

There are many benefits that justify the use of cache servers in the Web, namely, traffic reduction, server load reduction and latency decrease. Despite the fact that cache server has many advantages, it introduces a new set of problems, which are often difficult to be solved, such as:

Document is not consistent: documents in cache may not be updated and it is not easy to predict the validity time for a document.

Required document is not found in cache: when there are many failures, the document recovery time increases due to the additional time to verify if it is in the cache. To avoid this problem, the hit rate must be maximized and the failure cost must be minimized during the project of cache systems.

Dynamic content storage: Traditional cache servers store only static content. To solve this problem, some researchers have been conducted to define new methodologies to allow dynamic content storage in cache servers [7].

The web cache can be implemented in three ways: cache client, cache server and cache proxy. In this work, a transparent proxy cache server is used. Such an implementation uses network components to redirect the server requests, normally from layer 4 and 5 switches [6]. This technique is sad to be transparent, because clients (web browsers) do not need to define explicit configurations from the access point to the cache server. Layer 5 switches (L5) allow the client requests redirection by using information in TCP packets and/or in HTTP requests headlines [6].

3. Proposed Model

Traditional cache servers dispatch all client requests in the same way, disregarding existing requests with higher priority. This fact may influence the inclusion of mechanisms in the system to improve performance, such as the use of services differentiation in the network level. Some research efforts have been developed to provide better cache servers integration in environments that support service differentiation [8] [9] [12] [13]. All web server requests that have best-effort service type (standard class) are treated in the same way, without considering if the initial request has higher priority. To solve this problem, a cache server named CDF (with differentiation support) was implemented. Every requests from the CDF cache server to a Web Server is performed with the same priority that was originally requested by the client.

3.1. Cache with Differentiation of Service (CDF)

The CDF cache was implemented by means of the OPNET Modeler, based on a traditional cache server with some modifications in the programming of the Finite State Machines (FSM) for the application element of a cache server. The FSM includes a classifier to check the client requests priority and to send the request to a web server keeping this priority. The classifier only runs if the user request is not stored in the CDF cache. To evaluate the CDF cache server performance with service

differentiation in Web environments, it was considered three scenarios and the following characteristics described in Table 1.

Table 1. - Scenarios with service differentiation

Scenarios	
CS	With no cache server
CC	With traditional cache server
CDF	With cache server modified to support service differentiation

A new general model was specified (base scenario) to be used as a starting point to all other models used in the simulation. The base scenario is composed as described in Table 2.

Table 2. – Base Scenario

LAN	
N. Clients	400 (100 clients p/ category – Standard, Bronze, Silver and Gold)
Client-Side switch (SC) (Layer 5)	Responsible for interconnecting the LAN, the cache server and the client-side router (RC);
Client-Side switch (SC) (Layer 5)	Responsible for interconnecting the LAN, the cache server and the client-side router (RC);
Client-Side router (RC)	Responsible for interconnecting the client-side switch (SC) with the server-side router (RS);
Server-Side router (RS)	Responsible for interconnecting the server-side switch (SS) with the client-side router (RC);
Server-Side switch (SS)	Responsible for interconnecting the Web server with the RS router;
Web Server	Mono-processed

The connection between the client components (LAN, SC and RC) and the connection of the server components (SS and Web server) is made by 100BaseT links. The connection between the routers RS and RC is made through PPP DS1 links in two channels of 1024 Kbps. The scenarios use the priority queuing (PQ) policies which have the following priority order: Gold Class User > Silver Class User > Bronze Class User > Standard Class User.

Both traditional cache server and the cache supporting service differentiation have an average hit rate of 30% [6]. All scenarios were simulated with four different configurations for the link used between RS and RC routers, aiming at studying the environment performance in network overload situations. Other traffic scenarios were created in the network in order to overload the communication link, by adding traffic of 616Kb/s (40%) and 1232Kb/s (80%).

The equipments features considered in the several scenarios defined in this work are very close to the reality and the models used in the OPNET were evaluated and validated by the companies that produce those equipments.

3.2. Workload Generation

There are several studies of workload characterization in the web. In this study were considered static and dynamic workloads. The workloads used are based on the analysis of log files from the CISC (The Computation Center of São Carlos) Apache Server, carried out by Silva [10]. The static workload is described in Table 3 and the dynamic one in Table 4.

Table 3**.** – Static requests characteristics by class of user

User	Class of Service	HTTP Request Configuration (Objects per page)
Standard	BE	
Bronze	AF1x, AF2x	1 HTML object size. bytes = lognormal (8.55,1.42)
Silver	AF3x, AF4x	2 IMAGE objects size bytes= lognormal (8.25,1.62)
Gold	EF	

Table 4**.** – Dynamic requests characteristics by class of user

User	Class of Service	HTTP Request Configuration (Objects per page)
Standard	BE	
Bronze	AF1x, AF2x	1 DYNAMIC object size. bytes = lognormal (8.55,1.42)
Silver	AF3x, AF4x	
Gold	EF	2 IMAGE object size bytes= lognormal (8.25,1.62)

As shown in Tables 3 and 4, the difference between static and dynamic requests lies in the fact that there is one object HTML in the static context and one DYNAMIC object in the dynamic context. The other objects are identical in both contexts. In addition, the DYNAMIC and HTML objects have the same size and the difference is that the DYNAMIC object is the result of a query performed on the server. This query requires internal processing and is executed by using calls to a database. The average response time for executing and returning the dynamic object is described by a uniform distribution of 0.75 to 0.81 seconds. The processing time for the dynamic object is based on studies conducted by IBM to obtain the processing time for a part of an e-commerce transaction using DB2 database [11].

4. Results

To evaluate the results obtained from the proposed model, a discrete event simulation was built by using the OPNET Modeler environment [4]. This environment is used to model and simulate computer systems, designing communication networks, devices, protocols and applications with flexibility and scalability. For each scenario fifteen simulations were performed. Each simulation took 600 seconds and collected data corresponding to a period from 90 to 600 seconds. The obtained outcomes show the average of the simulation results, with a

confidence interval of 95%. All simulations were conducted with the static workload and then with the dynamic workload. After each simulation, the average HTTP response time was examined, as well as the link usage between the routers and the total number of downloads and canceled requests. The results for static and dynamic workload are presented in the next two sections.

4.1. Results for Static Requests

About 20.400 requests were executed and 99,90% obtained success and only 0,10% of the requests were canceled. This value had small variations in different scenarios, however, all variations occurred within the estipulated confidence interval. To study the average of HTTP response time the average time of general response to all requests and the average time for each class of user (standard, bronze, silver and gold) were analyzed. Figure 1 shows the average time for general HTTP responses.

As illustrated in Figure 1, the scenarios using cache server CC and CDF have a lower general HTTP response time than in the CS scenario. In scenarios CC and CDF it is possible to notice that the use of cache server reduces HTTP response time, with respect to CS scenario, in approximately 24% in all congestion levels.

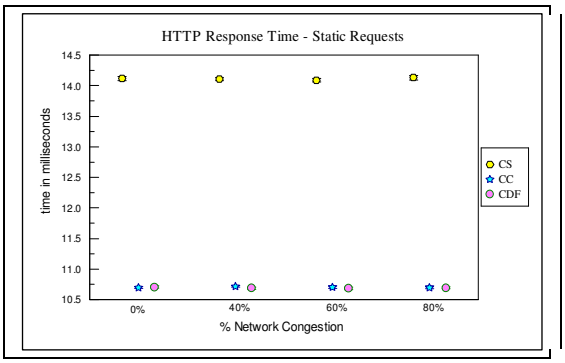

Fig. 1.. - Average HTTP response time

Figures 2 and 3 illustrate, respectively, the HTTP response time by class of users in environments with 40% and 80% of congestion. Figures 3 and 4 show that CC environment presents the same response time for all classes of users. This is because every requisition transmitted from the traditional cache server to the web server is of the best-effort type (standard). Comparing the performance of scenarios with cache server CDF and CC, it can be verified that gold and silver user classes are, on the average, respectively 4.12% and 1.95% faster than the CDF scenario. However, the bronze and standard classes are 1.15% and 4.72% slower

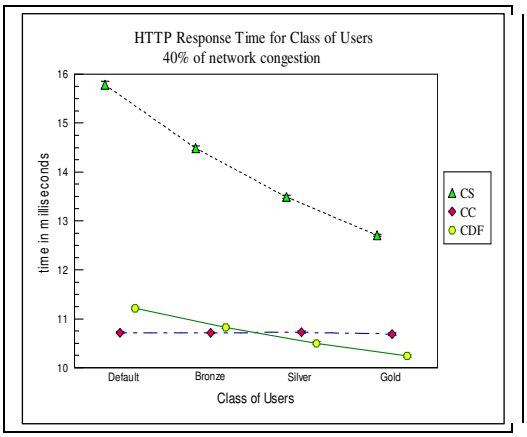

Fig. 2. - HTTP Response time by users classes in environments with 40% of congestion

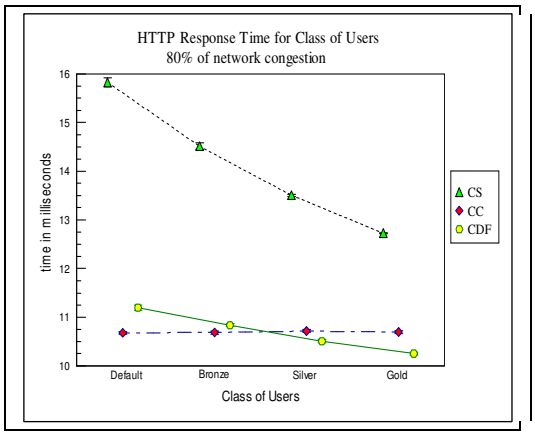

Fig. 3. - HTTP Response time by users classes in environments with 80% of congestion

4.2. Results for Dynamic Requests

The second experiment was conducted with dynamic workload in the same scenarios from the previous experiment. On the average, 20,400 requests were performed, in which 92.22% of them have succeeded and only 7.78% of the requests were canceled. To obtain the average HTTP response time for requests, the average time to all requests and also the average time for each class of user (standard, bronze, silver and gold) was examined. Figure 4 shows the average HTTP general response time.

Figure 4 shows that scenarios using cache server *CC* and *CDF* present an HTTP general response time slightly lower than *CS* scenario. The difference between the scenarios is small and this situation is explained because the dynamic request component has the same response time for any user, since there is no differentiation of service in the request execution on the server. Figures 5 and 6 illustrate,

respectively, the HTTP response time in environments with 40% and 80% of network overload

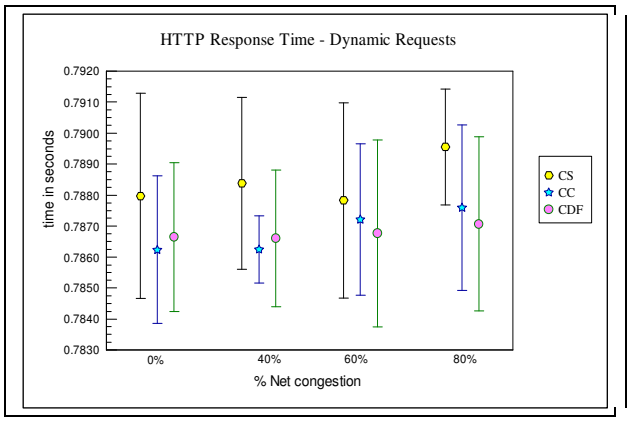

Fig. 4. - Average response time HTTP

As illustrated in Figures 5 and 6, the CC environment also shows the same response time for all classes of users, as in the previous experiment. The CS scenario, which has no cache server, has an HTTP response time very similar to scenarios with cache server. This fact is due to the presence of a dynamic component, that is not stored in the cache and always has to be recovered in the web server. Still, the scenarios with cache server have, on the average, a small improvement in performance, because in an HTTP request there are still two other static objects that can be recovered faster than the cache server.

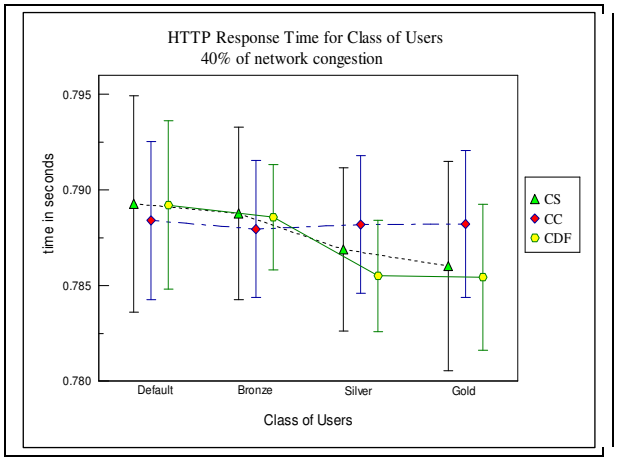

Fig. 5. - HTTP Response time by class of users in environments 40% of congestion

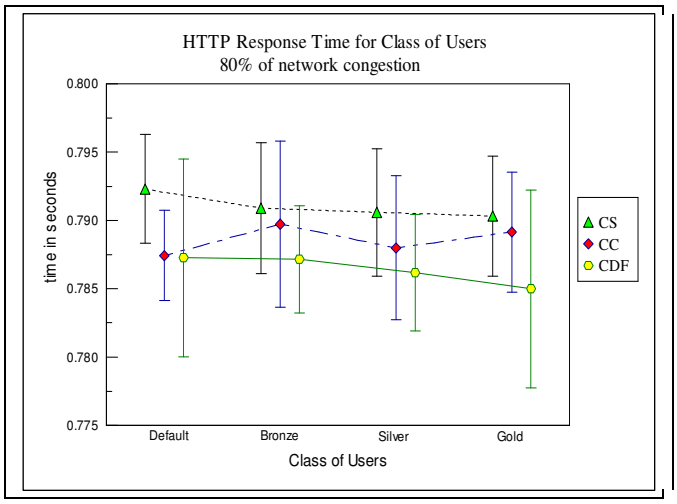

Fig. 6.. – HTTP Response time by class of users in environments 80% of congestion

5. Conclusion

The introduction of QoS on the Internet is a necessity for many applications and services offered by the Web. QoS is required at different levels of the web structure, and disconnected solutions cannot generate the desired effect, since all network elements must allow a differentiated control, where no bottlenecks could obstruct the system. It has been proved that the use of cache servers without service differentiation support in an environment with service differentiation, as CC scenario, leads to unsatisfactory results, eliminating any effort to prioritize classes of users, because all requests from the cache server to the Web server are made in the same class of standard user. The use of cache servers decreased in approximately 24% the HTTP response time in the experiment with static workload. In the experiment with dynamic workload, the cache server did not have significant influence on the general HTTP response time, because the dynamic object is always executed in the web server and it takes a longer time to be finished. The time that is saved with the static requests that are in the cache is too low compared with the time lost in the web server.

The use of Cache with Differentiation of services (CDF) decreased the HTTP response time to user classes with higher priority. Comparing the performance of scenarios with cache server, CDF, and scenario with traditional cache, CC, it can be verified that gold and silver user classes are, on the average, faster in the CDF scenario. In contrast, the bronze and standard classes are slower. The scenario with traditional cache (CC), in the experiment with static workload and hit rate cache at 30%, presents an HTTP response time lower than the one in the scenario without cache (CS). However, experiments have shown that if the cache hit rate is lower than 15%, then HTTP response time for classes with higher priority (silver and gold) are greater in scenario with traditional cache (CC), in comparison with the scenario without cache (CS). However, in experiments with a static workload, the use of

service differentiation gives good results when there is a need to treat classes of users differently. The HTTP response time for users with higher priority is lower, but the standard client service execution is disturbed. To summarize, considering the experiments with dynamic workload, there are not significant difference between the classes of users. This fact is justified since the differentiation of service is only used at network level and in the cache server. The system bottleneck is the Web server that handles the requests from clients according to the philosophy that the first to arrive will be the first to be processed, without using any classification for the users priority.

Acknowledgments. The authors would like to thank the financial support that the agencies CAPES, CNPq and FAPESP have provided to the projects of the Laboratory of Distributed Systems and Concurrent Programming (LaSDPC) from the ICMC-USP. Authors are also grateful to PUC Minas for the support from the Lecturers Enhancement Permanent Program (PPCD-APCH).

6. References

1 Gevros, P. Crowcroft, J. Kirstein, P. Bhatti, S, Congestion Control Mechanisms and the Best Effort Service Model. IEEE Network, Vol: 15, May 2001.
2 El-Gendy, M.A.; Bose, A.; Shin, K.G.; Evolution of the Internet QoS and support for soft real-time applications; Proceedings of the IEEE Vol 91, July 2003
3 OPNET Modeler – Accelerating Network R&D- from HTTP://www.opnet.com, May 2009.
4 Stardust; White Paper: The need for QoS.; 1999. doi=10.1.1.41.3374
5 Blake, S., Black, D. L., Carlson, M., Davies, E., Wang, Z. And Weiss, W., An Architecture for Differentiated Services. Internet RFC 2475; december 1998.
6 Zou, Q.; Martin, P. and Hassanein, H. 2003. Transparent distributed Web caching with minimum expected response time. In Proc. of the IEEE International Performance, Computing, and Communications, 2003.
7 Chen, W.; Martin P.; Hassanein, H. Differentiated Caching of Dynamic Content using Effective Page Classification. IEEE International Conference on Performance, Computing, and Communications, 2004
8 Wu-Hsiao Hsu, Ming-Chih Tung, Li-Yuan Wu, An integrated end-to-end QoS anycast routing on DiffServ networks, Computer Communications. Vol 30, Issue 6, 26 March 2007, Pages 1406-1418.
9 Zhou, J; Martin, P; Hassanein, H. QoS differentiation in switching-based Web caching. In IEEE Int. Conf. on Performance, Computing, and Communications, 2004
10 Silva, L. H. C.; Caracterização de carga de trabalho para testes de modelos de servidores Web; Master Thesis – ICMC – University of São Paulo – USP; September 2006.
11 P Martin, W Powley, HY Li, K Romanufa; Managing database server performance to meet QoS requirements in electronic commerce systems. International Journal on Digital Libraries Springer Berlin / Heidelberg, Vol 3, number 4. May 2002.
12 Wang, Q., Chen, J., Zhang, W., Yang, M., and Zang, B. 2007. Optimizing software cache performance of packet processing applications. SIGPLAN Not. 42, 7 (Jul. 2007), 227-236.
13 Ying L.; Abdelzaher T.F.; Saxena A. "Design, Implementation, and Evaluation of Differentiated Caching Services," IEEE Transactions on Parallel and Distributed Systems, vol. 15, no. 5, pp. 440-452, May, 2004.