Connection reservation algorithm in a Web server with service differentiation

Paulo S. F. Eustáquio, Ricardo Figueiredo, Sarita M. Bruschi, Regina H. C. Santana, Marcos J. Santana
Institute of Mathematics and Computer Science
University of São Paulo
São Carlos, Brazil
Email: {psfe, ricardo, sarita, rcs, mjs}@icmc.usp.br

Abstract—This paper presents an architecture prototype, named Web Server with Service Differentiation, able to provide QoS to different classes of services. With the implemented prototype, an admission control algorithm, named connection reservation algorithm is proposed and compared to the negotiation algorithm. The results of the performance evaluation have showed both algorithms met proportionally a higher number of high priority class (Class 1) requests in relation to low priority class (Class 2), although the connection reservation algorithm fitted all workload variance better. The connection reservation algorithm can be extended to the Web, where workload dynamic characteristics predominate.

Index Terms—Performance evaluation, Admission control, QoS, Web server.

I. INTRODUCTION

The Internet is currently the most efficient way to provide interaction between people and its popularization has made the Web the most used medium to change information among users.

Recently a new concept has been noticed on the Internet: the business-oriented paradigm. Companies have been moving their services to the Web [1], as they can have an accessible platform anywhere with a browser.

The amount of information on the Internet is so high that it causes a degradation in some parts of the net, affecting priority users in some Internet applications. In this context, this paper deals with the implementation, evaluation and analysis of the behavior of two admission control algorithms (negotiation [2] and connection reservation). Such algorithms aim to avoid the server overload and offer service differentiation between Classes. The performance evaluation of the algorithms is conducted by means of SWDS (in Portuguese: Servidor Web com Diferenciação de Serviços) model [3], which is a distributed web server able to provide service differentiation to different users priority.

The first contribution of this paper is the implementation of SWDS as a prototype, because in [3] this model was simulated. The second contribution is to evaluate the negotiation algorithm (proposed by [2]) and the connection reservation algorithm (proposed here) in the prototype.

The remainder of the paper is organized as follows: Section 2 presents the works related to this paper; Section 3 explains the SWDS architecture used in this paper; Section 4 presents the algorithms proposed and evaluated; Section 5 presents the environment and the design of experiments; Section 6 analyses the performance evaluation and the influence of factors; finally, Section 7 comprises a brief conclusive remark and discussions on future research directions.

II. RELATED WORKS

Plenty of researches can be found in Web servers when the concept of admission control is associated with QoS (Quality of Service) [4], [5], [6], [7], [8].

Bartolini et al. [9] proposed an admission control algorithm, named self overload control, which allows the self configuration of a dynamic limit of acceptance in order to respect service level agreements and maximize the utilization of resources at the same time. Requests are met until the limit of the SLA (Service Level Agreement) and server’s maximum limit. Experiments using the algorithm were conducted by simulation and characteristics that interfere in the final results were abstracted.

Semprebom et al. [10] proposes an adaptive admission control mechanism that implements two types of Web pages: a complete page (precise) and a partial one (imprecise). A server is designed and under an overload condition, it can meet a request with a partial one (imprecise). The model proposed by [10] does not use a distributed Web server. In our research the algorithms are evaluated in a distributed Web server architecture with service differentiation.

Poggi et al. [11] proposed a machine-learning technique that utilizes history navigation to estimate the probability of users’ buying in an e-commerce site. An admission control uses this technique to either prioritize sessions or drop them under an overload condition. It is possible to improve the sales, because clients that buy more have more priority than others. The system was implemented in an e-commerce site and it was proved that by utilizing a learning machine it is possible to meet 78% of buyers’ demands against 30% using a random strategy.

According to the related works, admission controls are important in Web servers because the computational capacity can be used correctly.

III. SWDS ARCHITECTURE

The SWDS architecture was simulated in a previous work [3]. In this paper a prototype of the SWDS model was implemented to evaluate the algorithms. The model is composed of
a Classifier, an Admission Control and a Web Server Cluster. The Classifier divides requests that arrive in the server by priority. The Admission control is responsible for meeting new requests in the system based on the information sent by the Web cluster machines. In an overload scenario, the request is dropped and a message error is sent to the user. If a request is admitted, it will be scheduled for a cluster Web node and processed, and the answer will be sent to the user.

Figure 1 shows the implemented model. The proxy server (Frontend) receives a request (1) and sends it to the mod_rewrite (2). mod_rewrite receives the request, sends it to an external program and classifies the request as Class 1 (high priority) or Class 2 (low priority) with HTTP header information. The external program (3) must perform 3 other tasks: receive status information from the Backend, decide on the acceptance of the request and (if the request is accepted) dispatch it in accordance with the implemented algorithm (4). The communication between mod_rewrite and the external program is made by stdin/stdout file manipulators. mod_status sends the number of requests processed by the Backend to the Frontend through sockets. After the decision-making process, the request is sent to the mod_proxy (5, 6), which sends it to the server specified by the schedule policy (7, 8, 9). Finally, the selected server meets the request and the Frontend receives the response, which is returned to the user (10, 11, 12).

IV. ADMISSION CONTROL ALGORITHMS

A. Negotiation algorithm

The main objective of an admission control is to reduce overload and the waste of system resources under overload conditions.

The negotiation algorithm [2] allows requests that were not met in their class to be met in a low priority class (QoS requirements lowered).

Requests dropped from the high priority class (Class 1) return to the classifier, and from an analysis in the idle simultaneous connections proportion of the low priority class (Class 2), the algorithm decides on their acceptance.

The SWDS architecture has two service classes. From a study [12], it was decided that half of the computational resources would belong to Class 1, and the other half to Class 2. When Class 1 resources are overloaded, the requests that will be dropped from Class 1 are negotiated to Class 2, ie, Class 1 requests can return to the system once again, and can be met as Class 2 requests.

Figure 2 shows how the negotiation algorithm works in the SWDS architecture. If a Class 1 request is dropped, the algorithm sends it to the classifier and marks it as a Class 2 request. Therefore Class 1 requests have two chances to be met in the system.

The negotiation algorithm was evaluated in a simulation of SWDS [2] and our motivation is to evaluate this algorithm in a prototype of the SWDS.

B. Connection reservation algorithm

The connection reservation algorithm proposed in this paper allocates exclusive connections to Class 1 dynamically. Therefore, the algorithm does not allocate an entire resource (in this paper, a machine), but only the necessary connections to Class 1 in a period of time, making a better cluster utilization.

The algorithm uses two Classes of service (similarly to the negotiation algorithm). After each request has been added, the percentage of Class 1 and Class 2 requests that are in the system is calculated. From this calculation, a thread is started to reserve the connections in each second. Class 1 requests use the system until the maximum connection limit and Class 2 limit is modified every second according to Equation 1:

\[ LC2 = LIMIT \times (1 - PC1 \times (PR1/0,5)) \]  \hspace{1cm} (1)

Where:
- \( LC2 \): number of connections that Class 2 can use concurrently with Class 1;
- \( LIMIT \): maximum limit of simultaneous connections available in the system;
- \( PC1 \): exclusive percentage of Class 1 connections (from 0 to 1) in an initial scenario of 50% of Class 1 workload and 50% of Class 2 workload.
- \( PR1 \): Percentage of Class 1 requests that arrive in the system (from 0 to 1);

Based on [12], the PC1 variable was set to 0.625 for all experiments. This decision maintains a higher number of exclusive connections to Class 1 independently of the workload.

After LC2 has been calculated, it is verified if the workload of the system is not higher than LC2. If not, Class 2 request is met; otherwise it is dropped.

V. ENVIRONMENT AND DESIGN OF EXPERIMENTS

Nine machines compose the SWDS prototype. They are allocated as follows:

- **Clients**: two machines connected through a gigabit switch (a machine for each class) with the frontend;
- **Frontend**: machine responsible for classifying, admitting, and/or dropping requests;
- **Backend**: six machines connected in the frontend through a gigabit switch. These machines process requests.
Figure 3 depicts the implemented prototype and Table I shows the hardware and software specifications.

HTTPPerf was used to generate the workload [13]. This tool has been used in several projects related to Web cluster, such as [14], [1], and [7]. A dynamic Web page developed in PHP with CPU and IO/Bound characteristics was used to overload the system in the experiments.

The experiments were conducted based on Table II. The Algorithm Factor corresponds to the algorithms showed in this paper, the Limit Factor corresponds to the maximum limit of simultaneous connections available in each backend, and the Workload Factor corresponds to the workload percentage submitted to each Class.

The workload corresponds to 20000 requests submitted to the SWDS server in a rate of 1000 requests per second, ie, when 50% of workload have been submitted to each Class, 10000 requests will be submitted to each Class in a rate of 500 requests per second. Other levels of workload follow the proportional distribution cited in Table II.

Full factorial was the methodology chosen to conduct the experiments [15], ie, all levels of all factors were combined.

**TABLE I**

<table>
<thead>
<tr>
<th>Function</th>
<th>Hardware configuration</th>
<th>Software configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clients</td>
<td>Core 2 Quad <a href="mailto:Q9400@2.66GHz">Q9400@2.66GHz</a> 4GB RAM Ubuntu 9.04</td>
<td>HTTPerf</td>
</tr>
<tr>
<td>Frontend</td>
<td>Core 2 Quad <a href="mailto:Q6600@2.40GHz">Q6600@2.40GHz</a> 8GB RAM</td>
<td>Ubuntu Server 9.04 Apache HTTPD v2.2.2</td>
</tr>
<tr>
<td>Backend</td>
<td>Core 2 Quad <a href="mailto:Q6600@2.40GHz">Q6600@2.40GHz</a> 2GB RAM</td>
<td>Ubuntu Server 9.04 Apache HTTPD v2.2.2</td>
</tr>
</tbody>
</table>

Fig. 1. SWDS model

Fig. 3. SWDS prototype.
TABLE II
FACTORS AND LEVELS OF THE DESIGN OF THE EXPERIMENTS

<table>
<thead>
<tr>
<th>Factors</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>negotiation algorithm</td>
</tr>
<tr>
<td></td>
<td>connection reservation algorithm</td>
</tr>
<tr>
<td>Limit</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td>800</td>
</tr>
<tr>
<td></td>
<td>1200</td>
</tr>
<tr>
<td>Workload</td>
<td>50% reqs. Class 1 and 50% reqs. Class 2</td>
</tr>
<tr>
<td></td>
<td>75% reqs. Class 1 and 25% reqs. Class 2</td>
</tr>
<tr>
<td></td>
<td>25% reqs. Class 1 and 75% reqs. Class 2</td>
</tr>
</tbody>
</table>

Each combination was run ten times for a confidence interval of 95% of the output variables. The mean response time and percentage of requests met are the output variables analyzed in this paper.

VI. PERFORMANCE EVALUATION

This section presents the results of the design of the experiments.

A. Negotiation Algorithm

The two first columns of Figures 4 and 5 correspond to 25% of workload for Class 1 and 75% of workload for Class 2. The middle columns correspond to 50% of workload for Class 1 and 50% of workload for Class 2. The two last columns correspond to 75% of workload for Class 1 and 25% of workload for Class 2. This standard of figure will be maintained in all bar graphs of both algorithms.

![Fig. 4. Mean response time - negotiation algorithm - lim. 400](image)

![Fig. 5. Percentage of requests met - negotiation algorithm - lim. 400](image)

In the third group of columns of Figure 7, when the simultaneous connections limit is high and Class 2 workload is low, the mean response time of Class 2 is lower than that of Class 1 because Class 2 requests do not overload the servers, ie, Class 2 requests are met only in underload conditions (see Figure 8).

![Fig. 6. Mean response time - Connection reservation algorithm - lim. 400](image)

B. Connection reservation algorithm

According to Figure 6, Class 1 has a low mean response time in relation to Class 2, because the limit of simultaneous connections is low and does not overload the system.

In the third group of columns of Figure 7, when the simultaneous connections limit is high and Class 2 workload is low, the mean response time of Class 2 is lower than that of Class 1 because Class 2 requests do not overload the servers, ie, Class 2 requests are met only in underload conditions (see Figure 8).

![Fig. 8. Percentage of requests met - connection reservation algorithm - lim. 400](image)

This section presents a comparison between the algorithms. Three tables were elaborated grouping all output variables.

C. Comparison of algorithms

This section presents a comparison between the algorithms. Three tables were elaborated grouping all output variables.

During the analysis of the results, the algorithms showed different behaviors. The Negotiation algorithm has three exclusive machines for Class 1; when these machines are overloaded and many Class 1 requests are negotiated to Class 2.

In all workload configurations Class 1 met more requests in relation to Class 2, because Class 1 requests have two chances of being met in the system.

B. Connection reservation algorithm

According to Figure 6, Class 1 has a low mean response time in relation to Class 2, because the limit of simultaneous connections is low and does not overload the system.

In the third group of columns of Figure 7, when the simultaneous connections limit is high and Class 2 workload is low, the mean response time of Class 2 is lower than that of Class 1 because Class 2 requests do not overload the servers, ie, Class 2 requests are met only in underload conditions (see Figure 8).

Figure 8 shows the percentage of requests met in both Classes. When Class 1 workload grows, fewer Class 2 connections are met because more connections are allocated exclusively to Class 1.

C. Comparison of algorithms

This section presents a comparison between the algorithms. Three tables were elaborated grouping all output variables.

During the analysis of the results, the algorithms showed different behaviors. The Negotiation algorithm has three exclusive machines for Class 1; when these machines are overloaded, other requests that will arrive are tagged as Class 2 and run for Class 2 machines. This algorithm is not good when Class 1 workload is small, because the allocated resources of this Class are underused.

The connection reservation algorithm does not reserve machines for a specific Class, but it allocates dynamically only the connections necessary to the Classes. The connections are reserved according to the workload system. If Class 1 workload is high, more exclusive connections are allocated to
Class 1. On the other hand, if Class 1 workload is low, fewer exclusive connections are allocated to Class 1. Therefore, the higher the Class 1 workload, the fewer the Class 2 requests met in the system.

Tables III, IV and V show a numerical comparison of the output variables. Table 3 presents the results of 25% of Class 1 workload and 75% of Class 2 workload. Table 4 presents the results of 50% of Class 1 workload and 50% of Class 2 workload and table 5 presents the results of 75% of Class 1 workload and 25% of Class 2 workload. Acronym TMRT presented in the tables stands for Total Mean Response Time.

Although the connection reservation algorithm met a low percentage of requests in most experiments, it obtained a low mean response time and prioritized Class 1 requests more than the negotiation algorithm. On the other hand, the negotiation algorithm met more requests in a longer response time and, in some workload configurations Class 2 met almost the same percentage of Class 1 requests.

### D. Influence of factors

An analysis combining the levels at 2 in 2 was performed to determine the influence of each factor on the mean response time and percentage of met requests using regression model [15]. The factors are represented by A (admission control algorithm), B (maximum limit of simultaneous connections) and C (workload imposed on the server). The number of levels of factor B was reduced to two, because it is possible to observe the influence of B using only two extreme levels (400 and 1200). Table VI represents the combinations of Factor C.

**Table VI Combination of the levels of Factor C**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>9, 10, 11, 12(a)</td>
<td>50% of Class 1 workload - 50% of Class 2 workload</td>
</tr>
<tr>
<td>75% of Class 1 workload - 25% of Class 2 workload</td>
<td></td>
</tr>
<tr>
<td>9, 10, 11, 12(b)</td>
<td>50% of Class 1 workload - 50% of Class 2 workload</td>
</tr>
<tr>
<td>50% of Class 1 workload - 75% of Class 2 workload</td>
<td></td>
</tr>
<tr>
<td>9, 10, 11, 12(c)</td>
<td>75% of Class 1 workload - 25% of Class 2 workload</td>
</tr>
<tr>
<td>25% of Class 1 workload - 75% of Class 2 workload</td>
<td></td>
</tr>
</tbody>
</table>

From Table VI, it is observed that the combination 9, 10, 11, 12 is the most influential combination, because the percentage of Class 1 met requests is improved proportionally, which makes it the best combination to achieve a higher response time.

For the evaluation of the combination, the factor with the highest influence on the mean response time is the admission control algorithm, which is represented by A. The results show that the negotiation algorithm met more requests in a longer response time and, in some workload configurations Class 2 met almost the same percentage of Class 1 requests.
the Class 2 workload, the lower the percentage of requests met when the connection reservation algorithm is used. On the other hand, the negotiation algorithm keeps the proportion of the requests met when Class 2 workload decreases.

VII. CONCLUSIONS AND FUTURE WORK

This paper has presented two important contributions: the implementation of SWDS prototype and the introduction and evaluation of two admission control algorithms coupled to SWDS prototype.

Negotiation algorithm is good to servers whose workload is similar, ie, everyday workload characteristics are almost the same (predictable). Therefore, it is possible to allocate an ideal number of machines to Class 1, however it is hard to predict the workload characteristics of Web.

The connection reservation algorithm allocates a portion of available resources to Class 1 using Equation 1. This equation allocates exclusive connections to Class 1 dynamically, with no resources waste as in the negotiation algorithm (when Class 1 workload is low).

Class 1 dominates the system when the workload of this type of class increases. However, from a limit of simultaneous
connections, a fixed number of connections is allocated to Class 2, and not all Class 2 requests are dropped.

The results allowed concluding the connection reservation algorithm can be extended to the Web, where dynamic workload characteristics predominate, i.e., the algorithm adapts to all workload conditions.

As future work, we are planning to incorporate a new priority class to the SWDS in order to verify the influence of a new Class. We also intend to set the PC1 variable of the connection reservation algorithm with other values to compare new mean response time and percentage of met requests.

ACKNOWLEDGMENT

The authors would like to acknowledge FAPESP for the financial support provided to this research.

REFERENCES


Fig. 12. Influence of factors on the Class 2 met requests.