Average Delay Guarantee in Server Systems Using Admission Control

Luc Malrait, Nicolas Marchand, Sara Bouchenak

Average Delay Guarantee in Server Systems Using Admission Control

Luc Malrait * **Nicolas Marchand** * **Sara Bouchenak** **

* *NeCS team, INRIA, GIPSA-lab, Univ. of Grenoble (e-mail: {Luc.Malrait},{Nicolas.Marchand}@inrialpes.fr)*
** *SARDES team, INRIA, LIG, Univ. of Grenoble (e-mail: Sara.Bouchenak@inrialpes.fr)*

Abstract: Although server technology provides a mean to support a wide range of online services and applications, their ad-hoc configuration poses significant challenges to the performance, availability and economical costs of applications. One of the main concerns is that under a heavy load, the delay introduced by a server in the transaction process may grow unbounded. This paper precisely addresses this issue. First, we present the design of a server model as a non-linear continuous-time model. Second, we develop an admission control algorithm that allow to ensure a maximum average delay on the server. Model and control algorithm were implemented and applied on the standard PostgreSQL database server running the TPC-C warehouse application. The experiments show that the proposed method provides significant benefits for database servers management.

Keywords: Modeling, Control, Server systems, Database systems, Admission control, QoS

1. INTRODUCTION

A large variety of Internet services exists, ranging from web servers to e-mail servers (Sendmail.org, 2007), streaming media services (Apple Inc., 2007), e-commerce servers (Amazon.com Inc, 2007), and database systems (PostgreSQL, 2008). These services are usually based on the classical client-server architecture, where multiple clients concurrently access an online service provided by a server (e.g. reading web pages, sending emails or buying the content of a shopping cart). Such server systems face varying workloads as shown in several studies (Arlitt and Williamson, 1996; Arlitt and Jin, 1999). For instance, an e-mail server is likely to face a heavier workload in the morning than in the rest of the day, since people usually consult their e-mails when arriving at work. In its extreme form, a heavy workload may induce server thrashing and service unavailability, with underlying economical costs. These costs are estimated at up to US$ 2.0 million/hour for Telecom and Financial companies (North American Systems International Inc., 2008).

A classical technique used to prevent servers from thrashing when the workload increases is admission control (Hyman et al., 1992; Hellerstein et al., 2004). It consists in limiting client concurrency on servers – also known as the multi-programming level (MPL) configuration parameter of servers. Obviously, servers' MPL configuration has a direct impact on server performance, availability and quality-of-service (QoS).

While the improvement of server performance and availability is usually achieved by system administrators using ad-hoc tuning (Brown, 2008; Microsoft, 2008), new approaches tend to appear to ease the management of such systems. In (Menascé et al., 2001) a heuristic for the management of the QoS of servers through the determination of the multi-programming level (MPL) of servers using the hill-climbing optimization technique is proposed. Althrough performing well in a variety of applications, hill-climbing does not guarantee optimality. In (Elnikety et al., 2004), a similar technique is applied; however the MPL is determined offline and thus, does not adapt to changing workloads. Other solutions to MPL identification were proposed specifically to some server technologies, such as transactional servers (Schroeder et al., 2006). Other approaches aim at modeling the system in order to characterize its capacity. In (Heiss and Wagner, 1991), a simulation-based study is conducted and an analytic model is proposed to adjust server MPL according to changing workloads. However, this model is restricted to performance functions with a parabola shape and thus, does not apply to criteria such as request latency and abandon rate that usually underly service level objectives (SLOs) as perceived by clients. Other works aiming at applying control theory to server systems appeared in the last decade. A first approach consists in applying well-known linear control theory on servers modeled as SISO (single-input single-output) or MIMO (multiple-inputs multiple-outputs) black-boxes (Parekh et al., 2002; Diao et al., 2002; Hellerstein et al., 2004). Nevertheless, due to the intrinsic non-linear behavior of these systems, linear control theory does not provide much success. Other approaches are based on non-linear models derived from queuing theory (Tipper and Sundareshan, 1990; Wang et al., 1996) with a theoretical proposal in (Kihl et al., 2003; Robertsson et al., 2004). The resulting models interestingly predict the performance of the system, but this is obtained at the expense of a hard calibration of model parameters in order to provide accurate results.

In this paper, we apply a nonlinear continuous-time control theory based on fluid approximations, in order to model and control the QoS of server systems. The QoS to control in this paper is the delay before a client is served. The total amount of clients, considered as an exogeneous input, directly impacts this delay. The main contribution of the paper is twofold:

- The design and implementation of a nonlinear continuous-time model of server systems for control purpose that accurately captures the main dynamics of server systems.
- The design and implementation of nonlinear admission control for server systems.

The paper presents our experiments on the TPC-C application, an industry-standard benchmark, running on the PostgreSQL database server.

2. BACKGROUND

2.1 Server systems

We consider server systems such as database servers and web servers that follow the client-server architecture where servers provide clients with some online service, such as online bookstore, or e-banking. Clients and servers are hosted on different computers connected through a communication network. Basically, a client remotely connects to the server, sends it a request, the server processes the request and builds a response that is returned to the client before the connection is closed. Multiple clients may concurrently access the same server.

Server workload is characterized, on one hand, by the number of clients that try to concurrently access a server (i.e. *workload amount*), and on the other hand, by the nature of requests made by clients (i.e. *workload mix*), e.g. read-only requests mix vs. read-write requests mix. Workload amount is denoted as N while workload mix is denoted as M. Server workload may vary over the time. This corresponds to different client behaviors at different times. For instance, an e-mail service usually faces a higher workload amount in the morning than in the rest of the day.

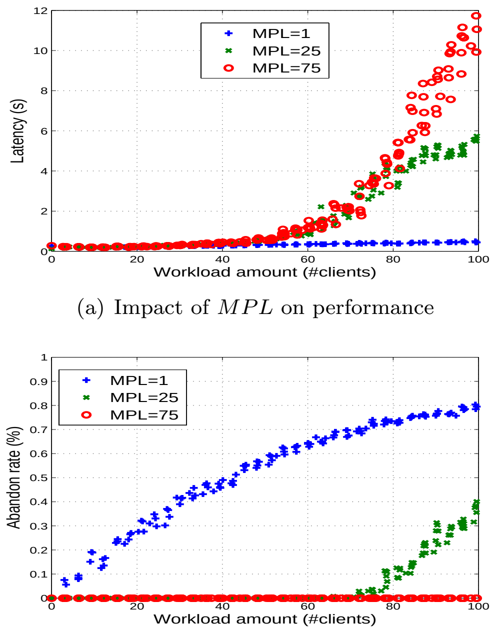
Server admission control is a classical technique to prevent a server from thrashing when the number of concurrent clients grows (Hyman et al., 1992). It consists in fixing a limit for the maximum number of clients allowed to concurrently access a server – the Multi-Programming Level (MPL) configuration parameter of a server. Above this limit, incoming client requests are rejected. Thus, a client request arriving at a server either terminates successfully with a response to the client, or is rejected because of the server's MPL limit. Therefore, due to the MPL limit, among the N clients that try to concurrently access a server, only N_e clients actually access the server, with $N_e \leq MPL$. Servers' MPL has a direct impact on the quality-of-service (QoS), performance and availability of servers as discussed below.

2.2 Service performance and availability

Several criteria may be considered to characterize service performance and availability (Menascé et al., 2001). In the following, we consider in particular two metrics that reflect performance and availability from the user's perspective (Menascé et al., 2001), namely *latency* and *abandon rate.*

Service performance – Latency. Client request latency is defined as the delay between the arrival of a request on a server and the service (ie the time needed by the server to process a request). The average client request latency is denoted as L. A low client request latency (or latency, for short) is a desirable behavior which reflects a reactive system. Figure 1(a) describes the impact of server admission control and MPL value on client request latency, when the workload amount varies Here, three values of MPL are considered, a low value (1), a medium value (25) and a high value (75). Low MPL is very restrictive regarding to client concurrency on the server and thus keeps the server unloaded which implies low client request latency. In contrast, with a high MPL, the server workload amount increases with client request latency.

Service availability – Abandon rate. Client request abandon rate is defined as the ratio between requests rejected due to admission control and the total number of requests received by a server. It is denoted as α. A low client request abandon rate (or abandon rate, for short) is a desirable behavior that reflects service availability. Figure 1(b) describes the impact of MPL on client request abandon rate. A low MPL is very restrictive regarding to client concurrency on the server. Obviously it implies a higher abandon rate compared to a high MPL.



(a) Impact of MPL on performance

(b) Impact of MPL on availability

Fig. 1. Impact of MPL

Service performance and service availability are part of the SLA (Service Level Agreement). The SLA specifies the service level objectives (SLOs) such as the maximum latency L_{max} and the maximum abandon rate α_{max} to be guaranteed by the server.

3. FLUID MODEL FOR SERVER SYSTEMS

We propose a fluid model which renders the dynamics of server systems and captures characteristics that reflect the state of servers in terms of performance and availability.

The choice of the control inputs, the system outputs and the state variable outputs is crucial for the design of a model. As the MPL is a tunable parameter of a server and has a meaningful effect on its performance, it is natural to take it as control input of the system. We assume that the state of the system can be described by three variables: the current number of concurrent requests in the server N_e, the throughput T_o and the rejection rate α. The server workload amount N can be seen as an exogenous input. Obviously, the outputs will depend on the chosen SLOs.

Among the N concurrent clients that try to connect to a server, admission control authorizes N_e concurrent clients to actually enter the server, with $0 \leq N_e \leq N$ and $0 \leq N_e \leq MPL$.

Let $cr(t, t+dt)$ be the number of client connections created on the server between t and $t + dt$, and $cl(t, t + dt)$ be the number of client connections closed on the server between t and $t + dt$. Thus, a balance on N_e between t and $t + dt$ gives

$$N_e(t + dt) = N_e(t) + cr(t, t + dt) - cl(t, t + dt) \qquad (1)$$

Let T_i be the incoming throughput of the server, measured as the number of client connection demands per second.

It comes that the number of connections created between t and $t + dt$ is

$$cr(t, t + dt) = (1 - \alpha(t + dt)) \cdot T_i(t + dt) \cdot dt \qquad (2)$$

where α is the abandon rate of the server.

Similarly, let T_o be the outgoing throughput of the server, measured as the number of client requests a server is able to handle per second. Thus, the number of connections closed between t and $t + dt$ is

$$cl(t, t + dt) = T_o(t + dt) \cdot dt \qquad (3)$$

Deriving from (1), (2) and (3) the derivative of N_e

$$\dot{N}_e(t) = (1 - \alpha(t)) \cdot T_i(t) - T_o(t) \qquad (4)$$

Moreover, we assume that the system reaches a steady state in a reasonably short period of time Δ; this is particularly reflected in state variables outgoing throughput T_o and abandon rate α. During this short period of time, the workload is relatively stable, which is consistent with studies such as (Arlitt and Jin, 1999). Thus, the dynamics of T_o and α can be approximated by first order systems through their derivatives as follows

$$\dot{T}_o(t) = -\frac{1}{\Delta}\left(T_o(t) - \bar{T}_o\right)$$
$$\dot{\alpha}(t) = -\frac{1}{\Delta}\left(\alpha(t) - \bar{\alpha}\right)$$

where $\bar{T}_o$ and $\bar{\alpha}$ are the steady state values of respectively the outgoing throughput and the abandon rate of the server. The next step naturally consists in finding the expression of $\bar{T}_o$ and $\bar{\alpha}$. A balance on the number of served client requests (or outgoing requests) N_o gives

$$N_o(t + dt) = N_o(t) + sr(t, t + dt)$$

where $sr(t, t + dt)$ is the number of served request between t and $t + dt$. Since there are N_e concurrent clients on the server and the average client request latency is L, the number of served requests during dt will be $sr(t, t + dt) = \frac{dt}{L} N_e$. Thus, we get $\dot{N}_o = \frac{N_e}{L}$, that is $\bar{T}_o = \frac{N_e}{L}$ which is an expression of Little's law (Little, 1961).

By definition, $\bar{\alpha}$ is equal to zero if N_e is smaller than MPL, and $\bar{\alpha}$ is equal to $1 - \frac{T_o}{T_i}$ if $N_e = MPL$. However, the stochastic nature of the client request arrival may lead to situations where the measured average N_e is smaller than MPL but where punctually, the number of clients that try to access the server is actually higher than MPL, and thus, some clients are rejected.

In order to take this behavior into account, we choose to write $\bar{\alpha} = \frac{N_e}{MPL} \cdot \left(1 - \frac{T_o}{T_i}\right)$. This renders that the probability to reject a client connection is higher when the average N_e is close to MPL. Finally, it follows that

$$\dot{T}_o(t) = -\frac{1}{\Delta}\left(T_o(t) - \frac{N_e(t)}{L(t)}\right) \qquad (5)$$

$$\dot{\alpha}(t) = -\frac{1}{\Delta}\left(\alpha(t) - \frac{N_e(t)}{MPL(t)} \cdot \left(1 - \frac{T_o(t)}{T_i(t)}\right)\right) \qquad (6)$$

Now that we have defined the model state variables, the last step consists in expressing the model output variable latency L. Latency obviously depends on the global load of the server, i.e. the workload mix M and the number of concurrent clients on the server N_e.

Figure 2 describes the evolution of latency L as a function of N_e, for a given workload mix [2]. One can see that a second degree polynomial in N_e is a good approximation of the latency L. Thus:

$$L(N_e, M, t) = a(M, t)N_e^2 + b(M, t)N_e + c(M, t) \qquad (7)$$

The parameter c is positive as it represents the zero-load latency. a and b are also positive since they model the processing time of requests. Equation (7) represents the delay of the system. This delay is clearly impacted by the the workload mix M and by the number of concurrent clients on the server N_e - which is a state variable of the system that dynamically varies according to equation (1).

In summary, the proposed fluid model is given by equations (4) to (7) that reflect the dynamical behavior of states and outputs of server systems in terms of performance and availability.

4. CONTROL OF SERVER SYSTEMS

In the following, we study the tradeoff between the performance and the availability of server systems, and derive the optimal admission control of server systems based on the proposed fluid model, that is the optimal number of

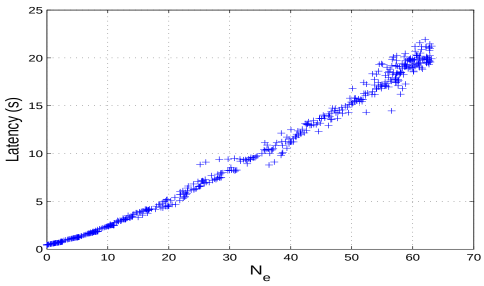


Fig. 2. Latency as a function of N_e

concurrent clients admitted to the server with respect to this tradeoff.

We want to guarantee a tradeoff between server performance and availability with the following properties:

(P1) the average client request latency does not exceed a maximum latency $L_{\max}$, and
(P2) the abandon rate α is made as small as possible.

To that end, a feedback control law is proposed to automatically adjust the MPL server admission control parameter in order to satisfy this tradeoff. The basic idea behind this law is to admit clients in such a way that the average client request latency L is close (equal) to $L_{\max}$. By construction, this maximizes the number of admitted clients N_e, which induces a minimized abandon rate α.

A first approach could consist in solving Eq. (7) in such a way that $L = L_{\max}$. Althrough accurately reflecting the system, such an approach is unwieldy since it requires the knowledge of accurate values of parameter a, b and c in equation 7, through an online identification of these parameters since the workload may change over the time. We propose another approach which avoids this online identification of model's parameters. It is a Lyapunov approach. First, let us assume that the dynamic of the load's variations is much smaller than the dynamic of T_o and α. It follows:

$$\begin{aligned} L(N_e) &= aN_e^2 + bN_e + c \\ \alpha(t) &= \bar{\alpha} \\ T_o(t) &= \bar{T}_o \end{aligned}$$

We choose $V(N_e) = \frac{1}{2}(L - L_{max})^2$ as a Lyapunov function candidate. Then with (4) we get

$$\dot{V}(N_e) = (L - L_{max})\,(2aN_e + b)\left(1 - \frac{N_e}{MPL}\right)(T_i - \bar{T}_o)$$

Taking $MPL = \frac{N_e}{1+\gamma_L(L-L_{\max})}$ will give:
$\dot{V}(N_e) = -\gamma_L(L - L_{max})^2(T_i - \bar{T}_o)(2an + b)$.
In case of overload $(T_i - \bar{T}_o > 0)$ since $2aN_e + b > 0$, we have $\dot{V}(N_e) \leq 0$ for every $\gamma_L > 0$. Invoking Lasalle's Invariance principle, L globally asymptotically converges to L_{max}.

The proposed admission control technique requires a unique external parameter, that is γ_L. This parameter has an impact on both convergence time of the control and stability of the system.

5. EVALUATION

5.1 Experimental setup

The evaluation of the proposed fluid model and feedback controllers has been conducted using the TPC-C benchmark (TPC-C, 2008). TPC-C comes with a client emulator which emulates a set of concurrent clients that remotely send requests to the database server. We extended the client emulator in order to be able, on the one hand, to vary the workload amount N over the time, and on the other hand, to vary the workload mix M over the time. For the latter extension, we considered two mixes of workload, one consisting of read-only requests, and another of read-write requests.

Experiments have been conducted on a set of two computers connected via a 100 Mb/s Ethernet LAN, one computer dedicated to the database server and another to the client emulator. The database server is PostgreSQL 8.2.6 (PostgreSQL, 2008). A proxy-based approach was followed to implement the controller where a proxy stands in front of the database server to implement online feedback admission control. Both client and server machines run Linux Fedora 7. The server machine is a 3 GHz processor with 2GB RAM, while the clients' computer is a 2 GHz processor with 512MB RAM.

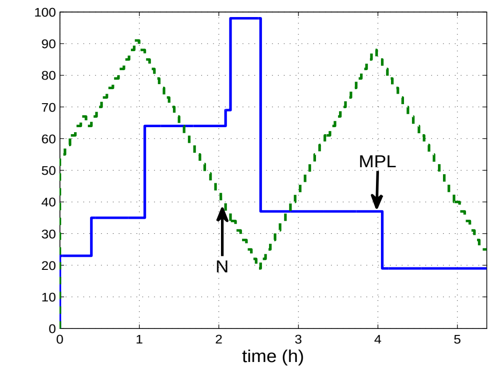
5.2 Model validation

We perform measurements to validate the accuracy of the proposed fluid model and its ability to render the dynamics of the system. In particular, we evaluate the ability of the model to reflect the variation of the state of the system when input variables such as the server MPL and the workload amount N vary. For the same set of input variables, the state reified by the model (which parameters have been off-line identified on another scenario) is compared with the actual state of the real system.

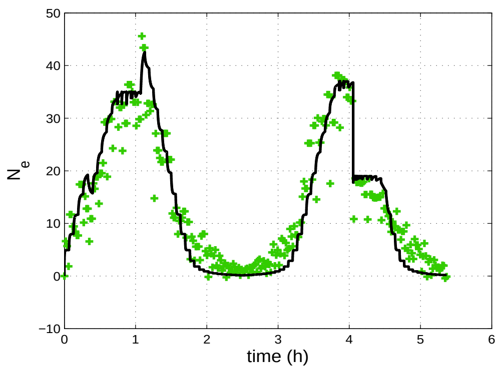
Figure 3 illustrates the case of an open loop system where both workload amount N and server MPL vary over the time (see Figure 3(a)). Figures 3(b), 3(c) and 3(d) present the evolution over the time of respectively the number N_e of concurrent clients admitted in the server, the outgoing throughput T_o and the abandon rate α, for both the real system (+) and the modeled system (solid line). Results show that the model is able to render the behavior of the real system.

5.3 Control evaluation

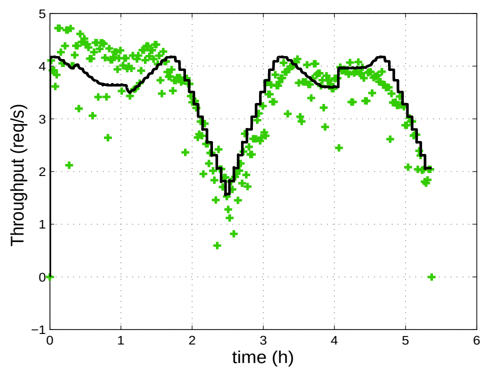
Figure 4 presents the system behavior under workload amount variation (c.f. Figure 4(a)) when the workload mix remains unchanged. Figures 4(b), 4(c) and 4(d) present the variation over the time of respectively the server MPL, the average client request latency and the client request abandon rate, comparing the non-controlled base system with the controlled system. Notice that, due to TPC-C client think time, the number of active clients at any given time may be different from (lower than) the actual load



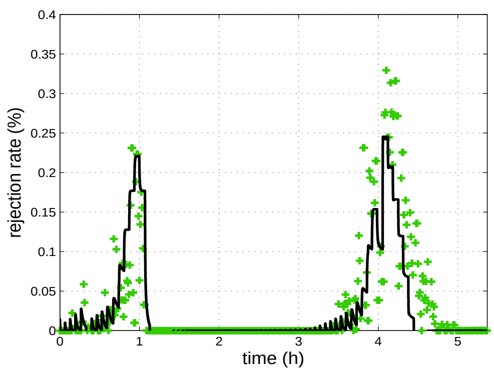
(a) Varying MPL and workload amount



(b) Admitted concurrent clients

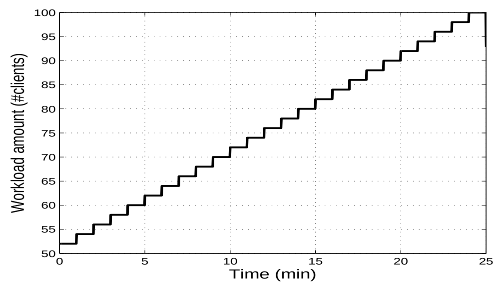


(c) Throughput

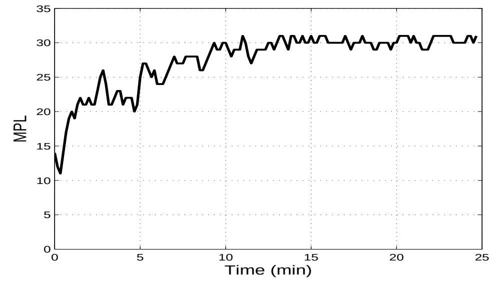


(d) Abandon rate

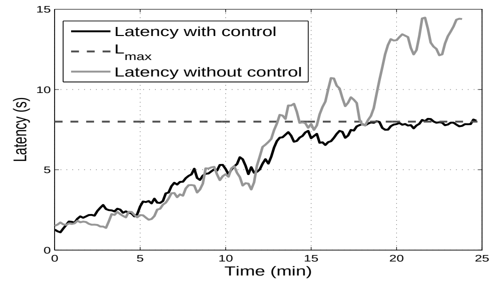
Fig. 3. System behavior with varying MPL and workload amount – Real system (+) vs. modeled system (solid line)



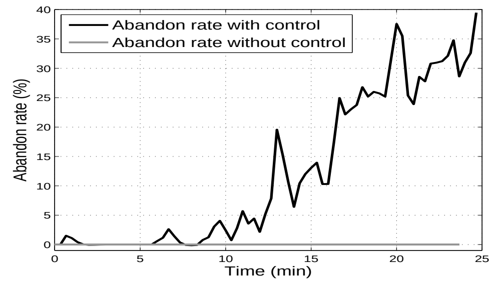
(a) Workload amount



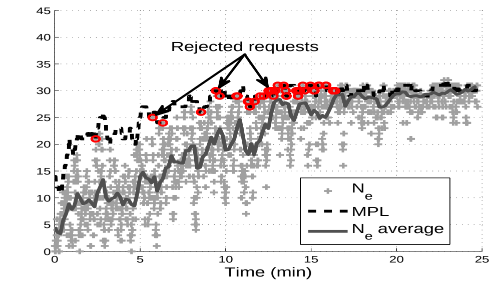
(b) MPL of controlled system



(c) Latency



(d) Abandon rate



(e) Admitted concurrent clients (N_e) vs. MPL

Fig. 4. System behavior upon workload amount variation – controlled system vs. non-controlled system

generated by TPC-C client emulator at that time. Results show that the controlled MPL is able to adjust its value to the optimal value so that the performance constraint is guaranteed. Whereas in the case of the non-controlled system, the latency grows up to 14.4 s, with an overhead of up to 80 % compared to the controlled system.

In the controlled system, the abandon rate is maintained below 5% with up to 75 clients (i.e. during the first half of the experiment). Then, the abandon rate increases with the increase of concurrent clients in the system, to attain its highest value when the number of clients is maximum, in order to keep latency below the target maximum latency. Notice that at the end of the experiment (between the 18th and 25th minutes), it seems justifiable to have a high abandon rate since latency attains its maximum authorized value (c.f. Figure 4(b)) and client request rejection is necessary at that time to guaranty the latency constraint. However, during the first part of the experiment where latency is lower than the authorized maximum latency, having an abandon rate which is higher that 0% is questionable. This is explained in Figure 4(e) by the stochastic nature of the workload where MPL is always higher than the average N_e but where punctually, there may be a client amount higher than the MPL and thus, a non-null abandon rate is observed (as depicted by circles in Figure 4(e)). To face this, an improved control law consists in taking into account system underload situations an thus, minimize abandon rate further (i.e. increase MPL) as long as the latency constraint is guaranteed. Due to space limitation, this scenario is not presented.

6. CONCLUSION AND PERSPECTIVES

This paper presents the design, implementation and evaluation of a nonlinear continuous-time model, upon which admission control of servers is derived for optimal configuration of servers. The proposed control law allows to guaranty the maximum average delay introduced by the server while the availability of its service is maximized. Our experiments show its efficiency.

In a future work, we will consider optimizing the MPL of a server so that several QoS constraints can be respected. Moreover, the application of server control on more complex systems distributed Internet services will be studied.

REFERENCES

Amazon.com Inc (2007). Http://www.amazon.com/.

Apple Inc. (2007). QuickTime Streaming Server. Http://www.apple.com/quicktime/streamingserver/.

Arlitt, M. and Jin, T. (1999). Workload Characterization of the 1998 World Cup Web Site. Technical Report HPL-1999-35(R.1), HP Laboratories Palo Alto.

Arlitt, M. and Williamson, C.L. (1996). Web server workload characterization: the search for invariants. *SIGMETRICS Perform. Eval. Rev.*, 24(1), 126–137.

Brown, M. (2008). Optimizing Apache Server Performance. Http://www.serverwatch.com/ tutorials/article.php/3436911.

Diao, Y., Gandhi, N., Hellerstein, J., Parekh, S., and Tilbury, D. (2002). Using MIMO feedback control to enforce policies for interrelated metrics with application to the Apache Web server. *Network Operations and Management Symposium.*

Elnikety, S., Nahum, E., Tracey, J., and Zwaenepoel, W. (2004). A Method for Transparent Admission Control and Request Scheduling in E-Commerce Web Sites. In *13th international conference on World Wide Web*. New York, NY.

Heiss, H.U. and Wagner, R. (1991). Adaptive Load Control in Transaction Processing Systems. In *17th International Conference on Very Large Data Bases.* San Francisco, CA.

Hellerstein, J.L., Diao, Y., Parekh, S., and Tilbury, D.M. (2004). *Feedback Control of Computing Systems.* John Wiley & Sons.

Hyman, J., Lazar, A.A., and Pacifici, G. (1992). Joint Scheduling and Admission Control for ATS-based Switching Nodes. In *ACM SIGCOMM).* Baltimore, MA.

Kihl, M., Robertsson, A., and Wittenmark, B. (2003). Analysis of admission control mechanisms using nonlinear control theory. *8th IEEE International Symposium on Computers and Communication*, 1306–1311 vol.2. doi:10.1109/ISCC.2003.1214294.

Little, J.D.C. (1961). A proof for the queueing formula L = λ W. *Operation Research*, 9, 383–387.

Menascé, D.A., Barbara, D., and Dodge, R. (2001). Preserving QoS of E-Commerce Sites Through Self-Tuning: A Performance Model Approach. In *ACM Conference on Electronic Commerce*. Tampa, FL.

Microsoft (2008). Optimizing Database Performance. Http://msdn.microsoft.com/en-us/library/aa273605(SQL.80).aspx.

North American Systems International Inc. (2008). The True Cost of Downtime. Http://www.nasi.com/downtime_cost.php.

Parekh, S., Gandhi, N., Hellerstein, J., Tilbury, D., Jayram, T., and Bigus, J. (2002). Using Control Theory to Achieve Service Level Objectives In Performance Management. *Real-Time Syst.*, 23(1-2), 127–141.

PostgreSQL (2008). Http://www.postgresql.org/.

Robertsson, A., Wittenmark, B., Kihl, M., and Andersson, M. (2004). Admission control for web server systems - design and experimental evaluation. *43rd IEEE Conference on Decision and Control.*

Schroeder, B., Harchol-Balter, M., Iyengar, A., Nahum, E., and Wierman, A. (2006). How to determine a good multi-programming level for external scheduling. In *22nd International Conference on Data Engineering.* Atlanta, GA.

Sendmail.org (2007). Http://www.sendmail.org/.

Tipper, D. and Sundareshan, M. (1990). Numerical methods for modeling computer networks under nonstationary conditions. *IEEE Journal on Selected Areas in Communications*, 8(9), 1682–1695. doi:10.1109/49.62855.

TPC-C (2008). TPC Transaction Processing Performance Council. Http://www.tpc.org/tpcc/.

Wang, W.P., Tipper, D., and Banerjee, S. (1996). A simple approximation for modeling nonstationary queues. *IEEE INFOCOM.* doi:10.1109/INFCOM.1996.497901.