Fluid Modeling and Control for Server System Performance and Availability

Luc Malrait
NeCS Control Theory Research Group
INRIA – Gipsa Lab
Grenoble, France
Luc.Malrait@inria.fr

Sara Bouchenak *
SARDES Distributed Systems Research Group
Univ. of Grenoble I – INRIA
Grenoble, France
Sara.Bouchenak@inria.fr

Nicolas Marchand
NeCS Control Theory Research Group
CNRS – Gipsa Lab
Grenoble, France
Nicolas.Marchand@inria.fr

Abstract

Although server technology provides a means 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. In this paper, we examine the impact of server configuration on the central tradeoff between service performance and availability. First, we present a server model as a nonlinear continuous-time model using fluid approximations. Second, we develop admission control of server systems for an optimal configuration. We provide two control laws for two different QoS objectives. $AM\text{-}\mathscr{C}$ *is an availability-maximizing admission control that achieves the highest service availability given a fixed performance constraint; and* $PM\text{-}\mathscr{C}$ *is a performance-maximizing admission control that meets a desired availability target with the highest performance. We evaluate our fluid model and control techniques on the TPC-C industry-standard benchmark. Our experiments show that the proposed techniques improve performance by up to 30 % while guaranteeing availability constraints.*

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

1. Introduction

A large variety of Internet services exists, ranging from web servers to e-mail servers [27], streaming media services [3], e-commerce servers [2], and database systems [23]. 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 [5, 8, 4]. 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 [12, 22].

A classical technique used to prevent servers from thrashing when the workload increases is admission control [11]. 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). Existing approaches to admission control either rely on ad-hoc tuning and heuristics without optimality guarantees [6, 21, 20], or apply linear control theory which does unfortunately not capture the intrinsic nonlinear behavior of server systems [25, 7], or follow a queuing theory approach where the system can be accurately modeled but at the expense of a hard model calibration process which makes it unwieldy to use [28, 30, 24]. We believe that modeling server systems is necessary to provide guarantees on the QoS. However, we argue that for the effective deployment of server modeling, the models must accurately capture the *dynamics* and the *nonlinear* behavior of server systems while being *simple* to deploy on existing systems.

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 main contribution of the paper is twofold:

- The design and implementation of a nonlinear continuous-time model of server systems that is simple to use since it involves very few external parameters, and which still accurately captures the dynamics of server systems as fluid flows.

- The design and implementation of nonlinear admission control for server systems. Two variants of control laws

*Contact. Phone: +33 (0)4 76 61 53 82, Fax: +33 (0)4 76 61 52 52
Submission category: Regular paper, Approximate word count: 7000 – The material has been cleared through the author affiliations.

are proposed: AM-$\mathscr{C}$ is an availability-maximizing optimal server admission control that achieves the highest service availability given a fixed performance constraint, and PM-$\mathscr{C}$ is a performance-maximizing optimal server admission control that meets a desired availability target with the highest performance.

The paper presents our experiments on the TPC-C application, an industry-standard benchmark, running on the PostgreSQL database server. The results of the experiments show that the proposed techniques provide significant benefits on the performance and the availability of the controlled system compared to a non-controlled system.

The remainder of the paper is organized as follows. Section 2 gives an overview of the background. Sections 3 and 4 present our contributions in terms of respectively a fluid model for server systems, and feedback admission control laws for servers. Section 5 describes the results of our experimental evaluation, and Section 6 presents the related work. Finally, Section 7 draws our conclusions.

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 the 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. Furthermore, server workload may vary over 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 [11]. 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 [20]. In the following, we consider in particular two metrics that reflect performance and availability from the user's perspective [20], namely *latency* and *abadon rate*.

Service performance – Latency. Client request latency is defined as 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 describes the impact of server admission control and MPL value on client request latency, when the workload amount varies [1] . Here, three values of MPL are considered, a low value (1), a medium value (25) and a high value (75). The low MPL is very restrictive regarding client concurrency on the server and thus, keeps the server unloaded and implies a low client request latency. In contrast, with a high MPL, when the server workload amount increases client request latency increases too.

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 2 describes the impact of MPL on client request abandon rate [1]. A low MPL is very restrictive regarding client concurrency on the server, and obviously implies a higher abandon rate compared to a high MPL which accepts more clients.

S ervice 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

[1]Details on the underlying experimental testbed are given in Section 5.1.

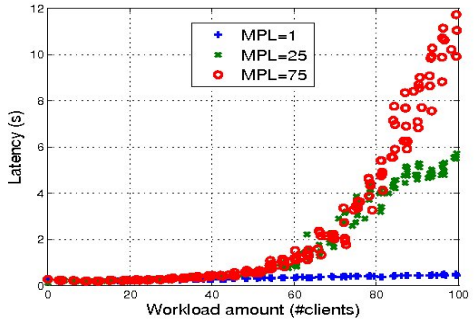


Figure 1. Impact of MPL on performance

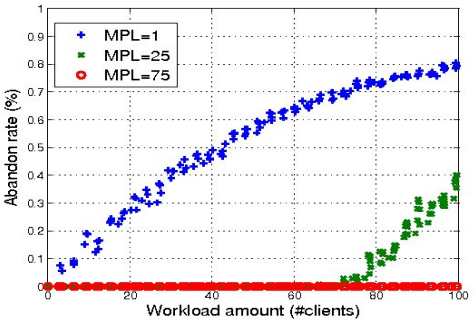


Figure 2. Impact of MPL on availability

state of servers in terms of performance and availability. Roughly speaking, fluid approximation consists in looking at all the state variables of the system - that are most integers - as real variables in $\mathbb{R}$. This enables to write the infinitesimal variation of characteristic state variables of the system with respect to time. Those variations can be seen as fluid flows, e.g. client request flows in the present case; and a request queue on the server is similar to a fluid tank [1]. The model is therefore built as a set of differential equations - as for most physical systems in mechanics, physics, electricity, etc. - that describe the time evolution of state variables. In the present case, we identify three state variables that describe and have an impact on server performance and availability, namely the current number of concurrent client requests in the server N_e, the server throughput T_o and the client request abandon rate α. State variables are usually influenced by themselves and by input variables. The inputs of the proposed model are: the server workload amount N and workload mix M exogenous inputs, and the server MPL tunable parameter that can be used to control the admission to the server. In addition to input and state variables, the model has output variables such as the average latency L to process a client request on the server. In the following, we describe the proposed fluid model through the formulas of its state and output variables.

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

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

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

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

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

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

Deriving from (1), (2) and (3), we have $\dot{N}_e$, the derivative of N_e

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

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 [4]. 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

$$N_o(t+dt) = No(t) + sr(t, t+dt)$$

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 [17].

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 $N_e = MPL$ (see Figure 3,

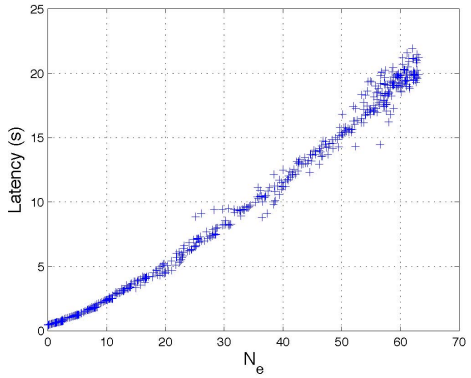


Figure 3. Accuracy of modeled abandon rate

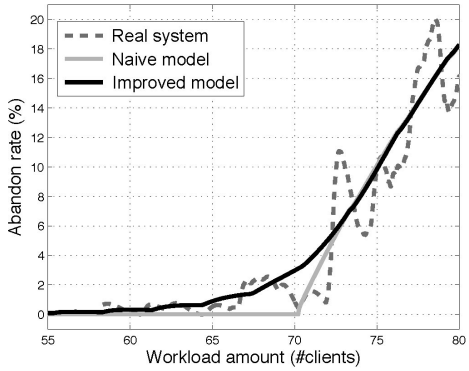


Figure 4. Latency as a function of N_e

naive model). 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. This is illustrated in Figure 3 which compares the actual measured abandon rate with the naive estimation of the abandon rate, showing a mismatch between the two [2] . 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. Figure 3 shows that this improved method provides a more accurate estimation of the abandon rate. Finally, it follows that

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

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

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 4 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:

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

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.

In summary, the proposed fluid model is given by equations (4) to (7) that reflect the dynamics of the state and outputs of server systems in terms of performance and availability. Section 4 then describes the proposed control techniques that build upon the fluid model in order to guaranty service performance and availability level objectives.

[2]Details on the underlying experimental testbed are given in Section 5.1.

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 concurrent clients admitted to the server with respect to this tradeoff. In particular, we provide two variants of control laws, namely AM-$\mathscr{C}$ and PM-$\mathscr{C}$. AM-$\mathscr{C}$ is an availability-maximizing optimal server admission control that achieves the highest service availability given a fixed performance constraint. Symmetrically, PM-$\mathscr{C}$ is a performance-maximizing optimal server admission control that meets a desired availability target with the highest performance. In the present case, service availability is measured as the client request acceptation rate (i.e. 1 - α), and service performance is measured as the average client request latency (i.e. L).

4.1 AM-$\mathscr{C}$ availability-maximizing control

AM-$\mathscr{C}$ aims at guarantying 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 $L = L_{\max}$. Although 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 time.

We propose another approach which avoids this online identification of model's parameters. It is obtained via a simple input-output linearization technique in which the considered output is latency L [14]. Roughly speaking, the approach aims at determining how to control the MPL value in such a way that

$$\dot{L} = -\gamma_L (L - L_{\max}) \tag{8}$$

As soon as $\gamma_L > 0$, this will ensure the convergence of L to its maximum $L_{\max}$. From Eq. (7), we have $\dot{L} = (2aN_e + b)\dot{N}_e$. And since T_o and α reach a steady state in a reasonably short period of time, $T_o(t) = \bar{T}_o$ and $\alpha(t) = \bar{\alpha}$ Therefore, with Eq. (4) we have

$$\dot{L} = (2aN_e + b)\left(1 - \frac{N_e}{MPL}\right)(T_i - \bar{T}_o) \tag{9}$$

As a result from Eq. (8) and (9), MPL should be controlled as follows

$$MPL = \frac{N_e}{1 + \frac{\gamma_L}{(2aN_e+b)(T_i-\bar{T}_o)}(L - L_{\max})}$$

To free ourselves from a and b, we choose to use $\gamma'_L = \frac{\gamma_L}{(2aN_e+b)(T_i-\bar{T}_o)}$, which produces

$$MPL = \frac{N_e}{1 + \gamma'_L(L - L_{\max})} \tag{10}$$

where $\gamma'_L > 0$ is a tuning parameter. It follows that with Eq. (8) and control described in (10), the dynamic evolution of L is given by:

$$\dot{L} = -\left(\gamma'_L (2aN_e + b)(T_i - \bar{T}_o)\right)(L - L_{\max})$$

Here again, L will converge to $L_{\max}$.

In summary, it is interesting to notice that the feedback control law given in (10) will reflect one of the following situations. If the current latency L is higher than L_{max}, property (P1) is not guaranteed and the control law will produce an MPL as a decreased value of the current number of admitted concurrent clients N_e (since $(1+\gamma'_L(L-L_{\max})) > 1$), which aims at meeting (P1). Symmetrically, if L is lower than L_{max}, property (P1) holds but property (P2) may not hold, and the control law will produce an MPL as an increased value of N_e (since $(1 + \gamma'_L(L - L_{\max})) < 1$), which aims at meeting (P2). Finally, if L is equal to L_{max}, both properties (P1) and (P2) hold.

4.2 PM-$\mathscr{C}$ performance-maximizing control

Similarly, PM-$\mathscr{C}$ aims at guarantying the following trade-off between server performance and availability where:

(P3) the client request abandon rate does not exceed a given maximum abandon rate $\alpha_{\max}$,

(P4) with the lowest average client request latency..

In this context, (P4) will be ensured given (P3) iff the MPL converges to the smallest value that guarantees $\alpha \leq \alpha_{max}$. Once again, we use an input-output linearization approach, taking α as the output, to solve the problem

$$\dot{\alpha} = -\gamma_\alpha(\alpha - \alpha_{\max}) \tag{11}$$

with $\gamma_\alpha > 0$. Furthermore, since the workload remains relatively stable during a short period of time, as stated previously, $\dot{N}_e = 0$. Then, from Eq. (4) and (6), we get

$$\alpha = 1 - \frac{T_o}{T_i}$$

$$\dot{\alpha}(t) = -\frac{1}{\Delta}\alpha(t)(1 - \frac{N_e(t)}{MPL(t)}) \tag{12}$$

Thus, from Eq. (11) and (12) and with the following control applied to MPL, α will converge to α_{max}

$$MPL = \frac{\alpha N_e}{\alpha + \gamma'_\alpha(\alpha - \alpha_{\max})} \tag{13}$$

where $\gamma'_\alpha = \gamma_\alpha \Delta$.

In summary, the proposed admission control techniques require a unique external parameter, that is γ. This parameter has an impact on both the convergence time of the control (i.e. the number of iterations to converge to the optimal MPL) and the stability of the system. Indeed, with a low value of γ, the convergence time toward the optimal MPL would be long. Whereas if γ is too high, this could induce system oscillations. A difficult part resides in choosing the right value of this parameter, which partly depends on the time necessary for the considered system QoS criteria (e.g. the abandon rate in case of PM-$\mathscr{C}$) to reach its steady state.

5 Evaluation

This section first describes the environment that underlies our experiments, before presenting the results of the evaluation of the proposed fluid model and feedback controllers.

5.1 Experimental setup

Testbed application. The evaluation of the proposed fluid model and feedback controllers has been conducted using

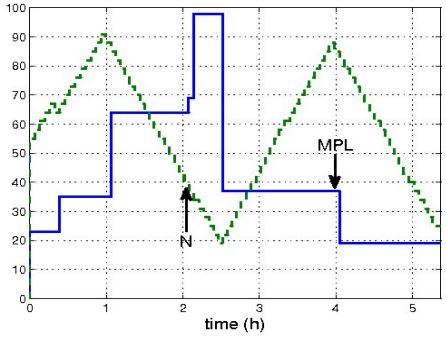
the TPC-C benchmark [29]. TPC-C is an industry standard benchmark from the Transaction Processing Council that models a realistic database server application as a warehouse system where clients request transactions on warehouses stored on a database server. TPC-C comes with a client emulator which emulates a set of concurrent clients that remotely send requests to the database server. The TPC-C client emulator allows to specify the number of concurrent clients to launch (i.e. the workload amount N). It also specifies the client think time, that is the interarrival time between two consecutive client requests. We extended the client emulator in order to be able, on the one hand, to vary the workload amount N over time, and on the other hand, to vary the workload mix M over time. For the latter extension, we considered two mixes of workload, one consisting of read-only requests, and another consisting of read-write requests.

Software and hardware environment. Our 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 [23]. The proposed model and controllers were deployed as follows. An online monitoring of the system allows to maintain the state of the model and well-known Kalman filtering techniques were applied [13]. A a proxy-based approach was followed to implement the AM-$\mathscr{C}$ and PM-$\mathscr{C}$ controllers where a proxy stands in front of the database server to implement online feedback admission control. In the following experiments, AM-$\mathscr{C}$ and PM-$\mathscr{C}$ were initialized with respectively $\gamma = 0.1$ and $\gamma = 0.3$. 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.

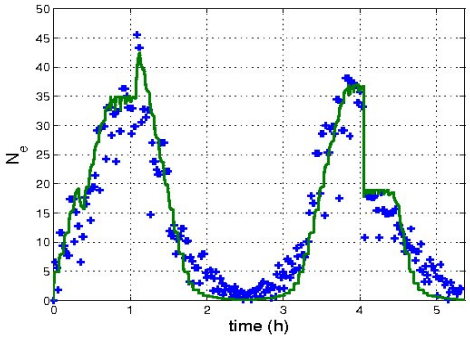
5.2 Model validation

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. The variation of the state of the system is described by the state variables N_e for the number of concurrent clients admitted in the server, T_o for the outgoing throughput of the server, and α for the client request abandon rate. Thus, for the same set of input variables, the state reified by the model is compared with the actual state of the real system.

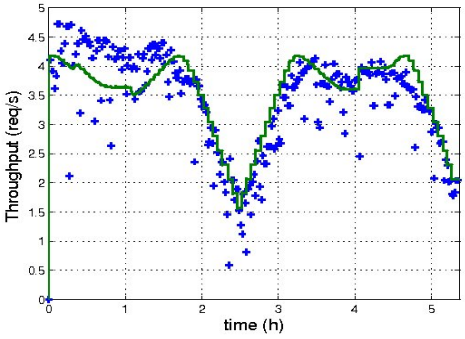
Figure 5 illustrates the case of a dynamic open loop system where both the workload amount N and the server MPL vary over time (see Figure 5(a)). Figures 5(b), 5(c) and 5(d) present the evolution over time of respectively the number N_e of concurrent clients admitted in the server, the outgoing throughput T_o and the abandon rate α, for both the real system



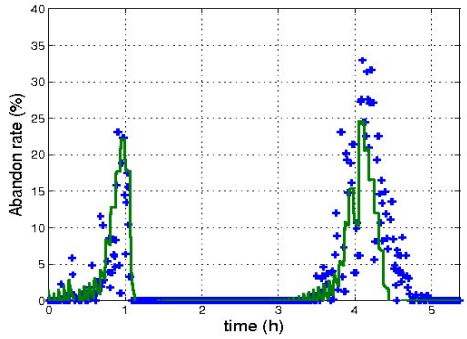
(a) Varying MPL and workload amount



(b) Admitted concurrent clients



(c) Throughput



(d) Abandon rate

Figure 5. System behavior with varying MPL and workload amount – Real system (+) vs. modeled system (solid line)

(+) 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

This section presents the results of the evaluation of the implemented feedback controllers presented in Section 4 when applied to the PostgreSQL database server that hosts the TPC-C database. The results of the experiments conducted with the AM-$\mathscr{C}$ availability-maximizing controller are first presented in Section 5.3.1, and the results of the PM-$\mathscr{C}$ performance-maximizing controller are then described in Section 5.3.2.

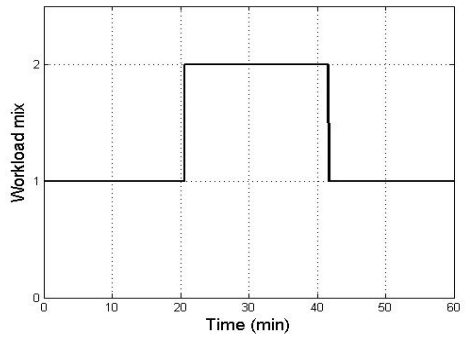
5.3.1 AM-$\mathscr{C}$ availability-maximizing control

In this section, we evaluate the proposed AM-$\mathscr{C}$ availability-maximizing feedback controller presented in Section 4.1. Here, we consider a performance constraint limiting the maximum average client request latency to 8 s. The role of AM-$\mathscr{C}$ is thus to guaranty that performance constraint while maximizing service availability, through online feedback control of the server MPL. We use two scenarii to evaluate this controller, each one illustrating a variation of one of the two exogeneous input variables of the system, i.e. the first scenario considers a changing workload mix, and the second scenario handles a varying workload amount N.

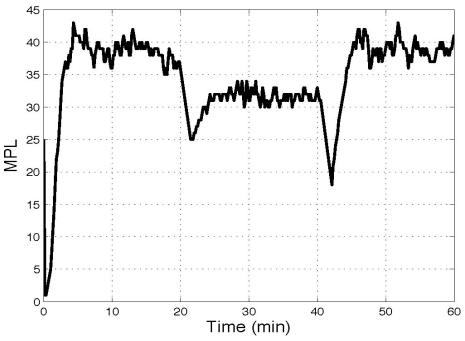
Figure 6 describes the first scenario where the workload mix varies from $M1$ to $M2$ and then back to $M1$ (c.f. Figure 6(a)), while the workload amount N is of 80 clients. The workload mix $M1$ consists of read-write requests while the workload mix $M2$ generates read-only requests. Figures 6(b), 6(c) and 6(d) present the variation over time of respectively the server MPL, the average client request latency and the client request abandon rate, comparing the non-controlled base system with a closed loop-based controlled system. Notice that the sudden change of MPL after the 20th and 40th minutes correspond to workload mix changes, which also has an impact on the latency and abandon rate.

Results demonstrate that the AM-$\mathscr{C}$ controller is able to dynamically adjust MPL in order to guarantee the latency performance constraint while keeping the service availability to its maximum, with an abandon rate minimized to 0% with $M1$ and to 9% in average with $M2$. Whereas, in the base system where concurrency on the server is not controlled and potentially unlimited, the QoS is not guaranteed and performance gets worse with a latency overhead of up to 30 %.

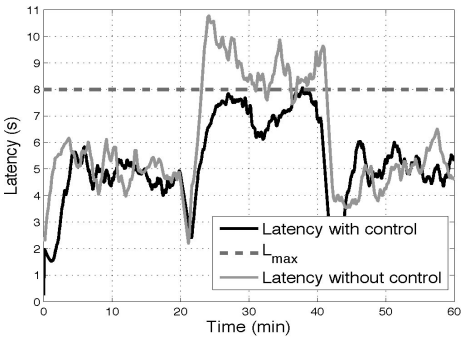
Figure 7 presents another dynamics of the system, that is the variation of the server workload amount over time (c.f. Figure 7(a)) when the workload mix remains at $M1$. Figures 7(b), 7(c) and 7(d) present the variation over time of respectively the server MPL, the average client request latency and 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



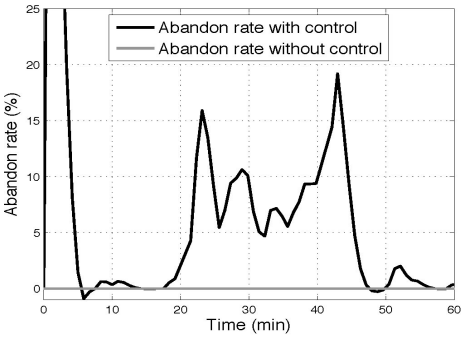
(a) Workload mix



(b) MPL of controlled system

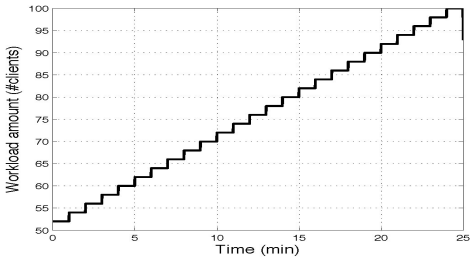


(c) Latency

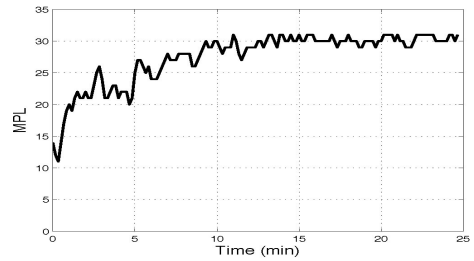


(d) Abandon rate

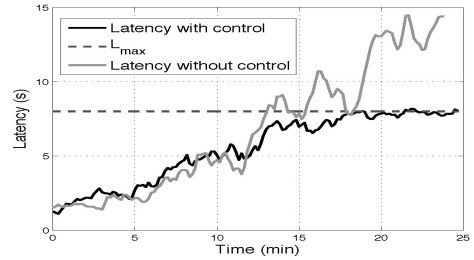
Figure 6. System behavior upon workload mix variation – AM-$\mathscr{C}$-based controlled system vs. non-controlled system



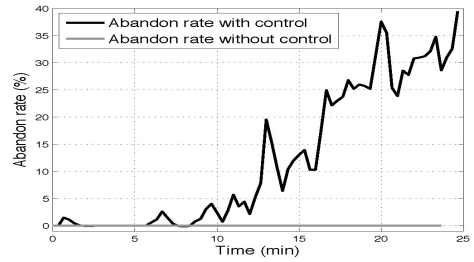
(a) Workload amount



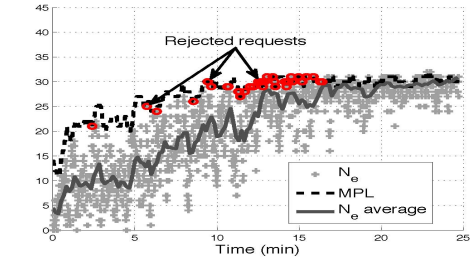
(b) MPL of controlled system



(c) Latency



(d) Abandon rate



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

Figure 7. System behavior upon workload amount variation – AM-$\mathscr{C}$-based controlled system vs. non-controlled system

than) the actual load 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 mainained 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 7(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 7(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 that is higher than MPL and thus, a non-null abandon rate is observed at that time (as depicted by circles in Figure 7(e)). To face this issue, an improved AM-$\mathscr{C}$ availability-maximizing control law consists in taking into account system underload situations an thus, minimize abanon rate further (i.e. increase MPL) as long as the latency constraint is guaranteed. Due to space limitation, we are not able to present results for this scenario.

5.3.2 PM-$\mathscr{C}$ performance-maximizing control

In this section, we evaluate the proposed PM-$\mathscr{C}$ performance-maximizing feedback controller presented in Section 4.2. Here, we consider an availability constraint limiting the maximum client request abandon rate to 10%. The role of PM-$\mathscr{C}$ is thus to guaranty this availability constraint while maximizing service performance, through online feedback control of the server MPL.

Figure 8 presents the variation of system behavior and dynamic control when the exogeneous input variable of workload mix M changes [1]. In Figure 8(a), the workload mix varies from $M1$ to $M2$ and then back to $M1$ when the workload amount N is of 80 clients. The workload mix $M1$ consists of read-write requests while the workload mix $M2$ generates read-only requests. Figures 8(b), 8(c) and 8(d) present the variation over time of respectively the server MPL, the client request abandon rate and the average client request latency, comparing the non-controlled base system with a controlled system. Here again, we notice a sudden change in the MPL when the workload mix suddenly changes, with an impact on the latency and abandon rate.

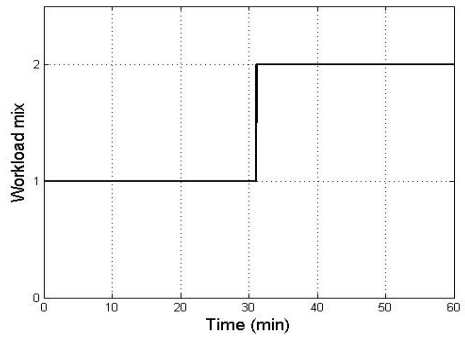
[1] Due to space limitation, results with a varying amout of workload N are not presented.

Results demonstrate that the PM-$\mathscr{C}$ controller is able to dynamically adjust MPL in order to guaranty the abandon rate constraint while keeping service performance to its maximum, with an average latency minimized to 4 s with $M1$ and to 8 s with $M2$. Compared to a non-controlled system, this improves system latency by up to 20 %.

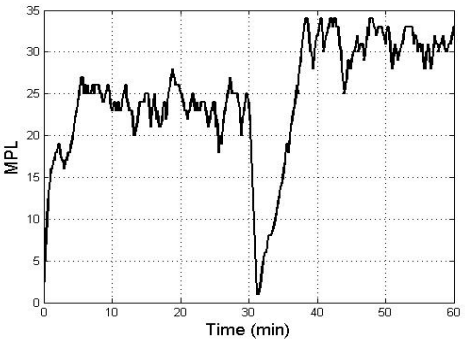
6 Related Work

System configuration is a crucial issue for the performance and availability of server systems [18, 19]. Much related work has been done in the area of system QoS management, investigating techniques such as admission control, service differentiation, service degradation and request scheduling. Due to space limitation, we briefly overview the work related to admission control for server system management. While the improvement of server performance and availability is usually achieved by system administrators using ad-hoc tuning [6, 21], new approaches tend to appear to ease the management of such systems. Menascé et. al. propose 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 [20]. Althrough performing well in a variety of applications, hill-climbing does not guarantee optimality. In [9], 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 [26]. Other approaches aim at modeling the system in order to characterize its capacity. In [10], 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 [25, 7]. 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 [28, 30] with a theoretical proposal in [15, 16, 24]. 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.

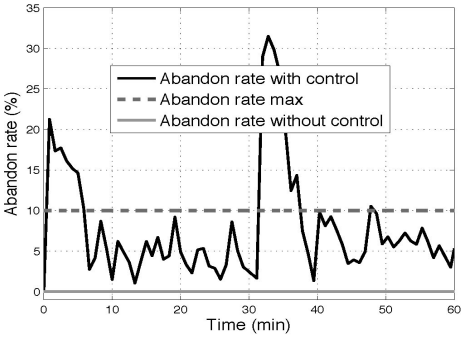
The proposed AM/PM-$\mathscr{C}$ system differs from the previous works in many respects. It applies control theory based on fluid approximation, which results in a simpler non-linear model with very few external parameters. Fluid approximation is successfully used to model and control various systems in other areas such as car flow control and population models.



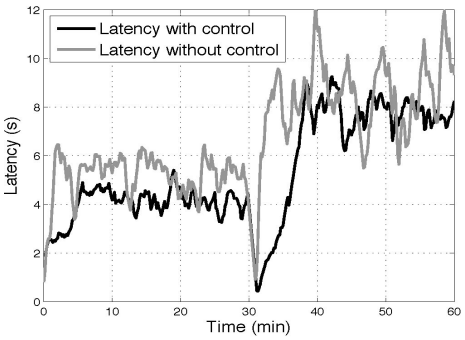
(a) Workload mix



(b) MPL of controlled system



(c) Abandon rate



(d) Latency

Figure 8. System behavior upon workload mix variation – PM-$\mathscr{C}$-based controlled system vs. non-controlled system

In the present work, we apply it to model and control server systems, and show how this allows to provide combined guarantees on service performance and service availability.

7 Conclusion

This paper presents the design, implementation and evaluation of a nonlinear continuous-time model based on the fluid flow control theory, upon which admission control of servers is derived for optimal configuration of servers. Two variants of control are proposed for two different QoS objectives. AM-$\mathscr{C}$ is an availability-maximizing optimal server admission control that achieves the highest service availability given a fixed performance constraint. PM-$\mathscr{C}$ is a performance-maximizing optimal server admission control that meets a desired availability target with the highest performance. Our experiments show that the proposed techniques improve performance by up to 30 % while guaranteeing availability constraints.

While this paper concentrates on QoS metrics such as client request latency and abandon rate, we believe that both the proposed modeling and control techniques may apply to other metrics, such as server throughput. Although the proposed modeling and admission control laws were applied to a database server, we believe that they could be easily applied to any sever system where admission control holds (e.g. web servers, application servers, etc.). Furthermore, we are interested in how these modeling and control techniques can be applied to distributed systems.

References

[1] T. Abdelzaher, Y. Lu, R. Zhang, and D. Henriksson. Practical application of control theory to Web services. *American Control Conference*, June 2004.

[2] Amazon.com Inc, 2007. http://www.amazon.com/.

[3] Apple Inc. QuickTime Streaming Server, 2007. http://www.apple.com/quicktime/streamingserver/.

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

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

[6] M. Brown. Optimizing Apache Server Performance, Feb. 2008. http://www.serverwatch.com/tutorials/article.php/3436911.

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

[8] J. A. Dilley. Web Server Workload Characterization . Technical Report HPL-96-160, HP Laboratories, Dec. 1996.

[9] S. Elnikety, E. Nahum, J. Tracey, and W. Zwaenepoel. 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, May 2004.

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

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

[12] Iron Mountain. The Business Case for Disaster Recovery Planning: Calculating the Cost of Downtime, 2001. http://www.ironmountain.com/dataprotection/resources/CostOfDowntimeIrnMtn.pdf.

[13] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[14] H. Khalil. *Nonlinear Systems*. Prentice Hall, 2002.

[15] M. Kihl, A. Robertsson, and B. Wittenmark. Analysis of admission control mechanisms using non-linear control theory. *8th IEEE International Symposium on Computers and Communication*, pages 1306–1311 vol.2, July 2003.

[16] M. Kihl, A. Robertsson, and B. Wittenmark. Performance modelling and control of server systems using non-linear control theory. In *18th International Teletraffic Congress*, Berlin, Germany, Sept. 2003.

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

[18] C. Loosley, F. Douglas, and A. Mimo. *High-Performance Client/Server*. John Wiley & Sons, Nov. 1997.

[19] E. Marcus and H. Stern. *Blueprints for High Availability*. Wiley, Sept. 2003.

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

[21] Microsoft. Optimizing Database Performance. http://msdn.microsoft.com/en-us/library/aa273605(SQL.80).aspx.

[22] North American Systems International Inc. The True Cost of Downtime, 2008. http://www.nasi.com/downtime_cost.php.

[23] PostgreSQL, 2008. http://www.postgresql.org/.

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

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

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

[27] Sendmail.org, 2007. http://www.sendmail.org/.

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

[29] TPC-C. Tpc transaction processing performance council, 2008. http://www.tpc.org/tpcc/.

[30] W.-P. Wang, D. Tipper, and S. Banerjee. A simple approximation for modeling nonstationary queues. *IEEE INFOCOM*, Mar. 1996.