Black-box Performance Models for Virtualized Web Service Applications *

Danilo Ardagna
Politecnico di Milano
Dipartimento di Elettronica e Informazione
32, Piazza L. da Vinci, 20133
Milano, Italy
ardagna@elet.polimi.it

Mara Tanelli
Politecnico di Milano
Dipartimento di Elettronica e Informazione
32, Piazza L. da Vinci, 20133
Milano, Italy
tanelli@elet.polimi.it

Marco Lovera
Politecnico di Milano
Dipartimento di Elettronica e Informazione
32, Piazza L. da Vinci, 20133
Milano, Italy
lovera@elet.polimi.it

Li Zhang
IBM Research
T.J. Watson Research Center
Yorktown Heights, NY 10598
zhangli@us.ibm.com

ABSTRACT

In order to reduce the operating costs of IT systems, nowadays service applications are executed in virtualized infrastructures and a time varying fraction of the physical servers' capacity is shared among running applications. The performance modelling of a virtualized server is very challenging as the impact of the choice of the Virtual Machine Monitor (VMM) scheduler, its parameters and I/O management overhead is still only partially understood. In this paper, black-box models based on the Linear Parameter Varying (LPV) framework are proposed for the run-time modelling and performance control of Web services in virtualized hosting environments. As the behavior of the application response time is highly time varying and the workload conditions substantially change within the same business day, LPV models seem very promising for predicting the performance of such systems. Specifically, the suitability of subspace LPV identification methods for multi-variable systems is investigated and their performance assessed on experimental data gathered on Xen environments.

*The work of Danilo Ardagna and Mara Tanelli has been partially supported by the GAME-IT project funded by Politecnico di Milano. The work of Danilo Ardagna has been partially supported also by the EU Commission with the Q-ImPrESS research project and by the programme IDEAS-ERC, project SMScom. The work of Mara Tanelli and Marco Lovera has been partially supported by the MIUR project "New methods for Identification and Adaptive Control for Industrial Systems."

WOSP/SIPEW'10, January 28–30, 2010, San Jose, California, USA.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Performance attributes

General Terms

Experimentation, Measurement, Performance

1. INTRODUCTION

Virtualization of physical resources was proposed in the early '70s to allow resource sharing in large expensive mainframes among multiple application environments. With the development of distributed systems, as hardware costs went down, the need for virtualization faded away. Nowadays virtualization is driving again the interest of research both from industry and academia as a way to improve system security and reliability, reduce costs, and provide flexibility of the physical infrastructure [21].

In a virtualized environment, physical resources (e.g., CPU, disks, communication network) are partitioned into multiple virtual ones, creating isolated virtual machines each running at a fraction of the physical system capacity. A virtual machine monitor (VMM), like Xen [48] or VMWare [44], is installed on each physical server and manages physical resources, providing performance differentiation and performance isolation to competing running VMs. Usually, each VM is dedicated to the execution of a single application, thus increasing system security and reliability, and has access to a time varying fraction of the physical server capacity defined by the *VMM resource allocation parameter* (i.e., VMWare *shares* or Xen *weights*, [48, 44]).

VMM resource allocation parameters can be updated by accessing the VMM API and the change takes effect in few milliseconds without introducing any system overhead. Hence, the VMM resource allocation parameters can be adopted effectively to control and adapt the application performance to incoming workload variations on very fine-grained time scales.

This VMM feature is particularly important since Internet workload can vary by orders of magnitude within the same business day [6]. In order to guarantee application Quality of Service (QoS) and implement a cost-effective environment [8, 11], infrastructure

providers can dynamically allocate resources among different VMs according to a prediction of system performance [2, 37] under the current workload conditions and configuration. This prediction is based on the continuous monitoring of the system which, according to the autonomic computing paradigm [15], can trigger dynamic reconfiguration if the performance predicted for the running applications lead to QoS violation.

On the other hand, virtualization introduces new problems related to resource multiplexing and scheduling of multiple VMs. The performance modelling of virtualized environments is challenging, as the impact of the choice of the VMM scheduler, its parameters and I/O management overhead is still only partially understood (see [7]). An overview of VMM scheduling problems, communication issues and I/O overhead management in virtualized hosting platforms can be found in [12, 47].

Furthermore, traditional queueing network models are inadequate for the run-time modelling of virtualized systems performance at a very fine-grained time scale. Indeed, under strict time constraints, only bounding techniques or mean value analysis can be considered. Both approaches are based on the assumption that the system is in a steady state and therefore cannot accurately model system transients. Performance models able to predict the system behaviour at very fine-grained time resolution (seconds) and in transient conditions are hence needed.

The aim of this work is to derive – via system identification techniques – Linear Parameter Varying (LPV) models for the run-time modelling of service based systems hosted in a virtualized environment. A LPV model is linear in the parameters and a vector of scheduling variables enters the system matrices in an affine or linear fractional way [18, 3, 23]. In particular black-box models are derived from experimental data, which are capable of describing the execution of service applications with an accuracy suitable for run-time control. Note that, due to the presence of multiple VMs, the identification problem arising in this context is genuinely multi-variable. To the best of the authors' knowledge, this is the first attempt to describe the behavior of such systems by means of Many-Input-Many-Output (MIMO) dynamic models. LPV models performance will be assessed on experimental data gathered on Xen environments. Results show that LPV models allows predicting VMs performance with an average percentage error lower than 20% at a 10s time resolution.

This paper is organized as follows. Section 2 introduces the adopted notation and the identification problem setting. Section 3 illustrates the LPV state space models employed for identification and the related identification algorithms. Section 4 is devoted to present the experimental set-up, the experiments performed, and to discuss the obtained results. Section 5 provides a review of the literature. Conclusions are finally drawn in Section 6.

2. PROBLEM STATEMENT

In this paper, CPU bounded Web service applications which share a virtualized computing platform will be considered. The VMM is configured to support work-conserving mode [12] and, for the sake of simplicity, each VM hosts a single application. In the following the LPV-model identification problem will be formulated for applications running on a single core but can be easily extended to multiple-cores system implemented in moderns CPUs. Indeed, an independent LPV model for each core can be derived and managed at run-time without introducing any significant overhead in the system.

Our aim is to derive a dynamic model capable of capturing the virtualized system behaviour at a very fine-grained time resolution (seconds), with an accuracy suitable for control purposes. This identification process provides a control-oriented dynamical description of the server behavior and it is the first step to be achieved in order to design a closed-loop controller for the autonomic management of a virtualized system. The design of the closed-loop controller is the focus of our ongoing work. In the remainder of the paper the following notation will be adopted:

- Δt denotes the sampling interval;
- k is the discrete time index over the interval $[k\Delta t,\ (k+1)\Delta t]$;
- R_k^i is the average response time, i.e., the overall time a request stays in the system, for application i in the k-th time interval.
- ϕ_k^i represents the VMM resource allocation parameter, i.e., the fraction of capacity devoted for executing the VM which hosts application i in the k-th time interval;
- n denotes the number of running VMs.

The VMM allocates CPU to competing VMs proportionally to the weights ϕ_k^i that each VM has been assigned in the k-th time interval. The service discipline implemented by work-conserving mode can be modeled as a first approximation by the Generalized Processor Sharing (GPS) scheduling. Under GPS, the server capacity devoted to the i-th VM at time t is:

$$\frac{\phi_k^i}{\sum_{i' \in \mathcal{I}(t)} \phi_k^{i'}}$$

where $\mathcal{I}(t)$ is the set of applications with waiting requests at the server at time t. For example, if two VMs have been assigned the same fraction of capacity, then they are guaranteed to receive 50% of the CPU each. Under work-conserving mode [7] if one of the two aforementioned VMs is blocked for an I/O operation (or simply because no requests are running in the system), then the other one can consume the entire CPU.

GPS analysis is notoriously difficult [49]. No closed formula exists even for open models and accurate results can be obtained only by resorting to simulation which, requiring a significant computation time, cannot be adopted to predict system performance at run-time with very fine-grained time resolutions.

In the following we assume that the system is configured such that $\sum_{i=1}^{n} \phi_k^i = 1,\ \forall k$, i.e., the entire system capacity is devoted to application execution (this is usual in this context, see e.g. [2]).

3. LPV STATE SPACE MODELS AND IDENTIFICATION ALGORITHMS

The problem of model identification can be formulated as the one of deriving a mathematical representation for the behaviour of a physical system on the basis of input-output data collected in dedicated experiments. As far as linear models are concerned, the main *ingredients* of an identification problem are essentially: 1) a definition of the class of models to be considered and 2) a suitable algorithm for the estimation of the model parameters on the basis of the available data. Classical model identification problems for Single-Input Single-Output (SISO) time-invariant systems are formulated using models in input-output form (i.e., difference equations relating the measured input and output variables in a direct way), the parameters of which are estimated using least squares techniques. Whenever Multiple-Input Multiple-Output and/or time-varying systems must be dealt with, however, state-space representations turn out to be a more flexible and reliable model class.

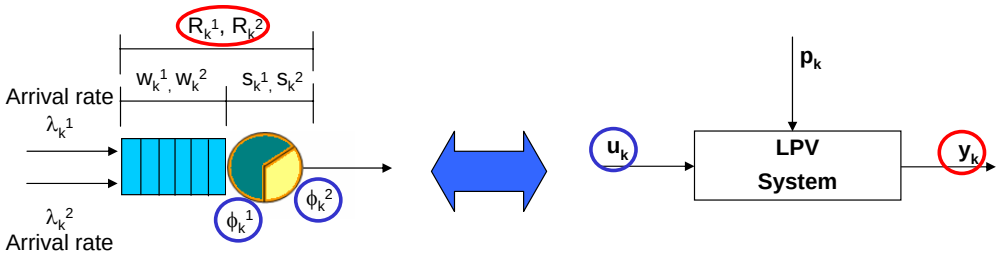


Figure 1: Relation between the GPS queue model of the reference virtualized system and the control oriented LPV models.

Flexibility is due to the fact that the same parameter estimation algorithms can be used regardless of the input and output dimension; reliability stems from the numerical stability of the available parameter estimation algorithms, particularly as far as the so-called subspace model identification (SMI) class of methods (see, e.g., [14, 42] and the references therein) is concerned.

In this paper, discrete-time linear state-space models will be considered, described by the following state space representation

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k \\ y_k &= C_k x_k + D_k u_k, \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$ is the vector of control inputs and $y \in \mathbb{R}^l$ is the vector of measured outputs. More precisely, with reference to the specific modelling problem considered in this study, $u_k = [\phi_k^i]\ i = 1, \ldots, n-1$ and $y_k = [R_k^i]\ i = 1, \ldots, n$, i.e., the goal of the model identification problem considered in this paper is to derive a state-space model describing the dynamic relationship between the VMM resource allocation parameters and the server response time. Note that, under the condition $\sum_{i=1}^{n} \phi_k^i = 1, \forall k$, if n VMs are running concurrently, the available control variables are $n-1$, since there are only $n-1$ VMM resource allocation parameters which are linearly independent. The correspondence between the queue model of the reference virtualized system and the control oriented LPV model which will be derived by black-box identification techniques is illustrated in Figure 1.

In (1) generically time-varying state space matrices $\{A_k, B_k, C_k, D_k\}$ have been considered. In what follows, however, we will introduce additional assumptions on the time-variability of the model dynamics, suitably tailored for the model identification problems associated with the management of virtualized systems.

More precisely, the generic time-variability will be restricted to the LPV class. LPV systems are linear time-varying plants whose state space matrices are fixed functions of some vector of varying parameters. LPV model identification algorithms are available in the literature both for input-output and state-space representations of parametrically-varying dynamics [18, 3, 23, 40, 10]. In particular, in the recent works [25, 24] an input-output modelling approach was adopted. If, however, the aim of the identification procedure is to eventually work out LPV models in state-space form for control design purposes, one should keep in mind that the usual equivalence notions applicable to Linear Time Invariant (LTI) systems cannot be directly used in converting LPV models from input-output to state space form, as the *time-variability* of LPV systems ought to be taken into account (see, e.g., the discussion in [36]). Bearing this in mind, we focus in this work on state-space LPV models

$$\begin{aligned} x_{k+1} &= A(p_k) x_k + B(p_k) u_k \\ y_k &= C(p_k) x_k + D(p_k) u_k, \end{aligned} \tag{2}$$

where $p_k \in \mathbb{R}^s$ is the parameter vector. Note that in order to keep the technical issues in the presentation of LPV models and the associated parameter estimation methods to a minimum, in the following we will deliberately focus on purely deterministic models, i.e., as is already apparent comparing (1) and (2), we will ignore the presence of the noise terms in the model representation. It is important to point out, however, that the theory underlying the parameter estimation algorithms used in this work can effectively deal with the presence of process and measurement noise; for a detailed treatment of such aspects we refer the interested reader to, e.g., [40].

It is often necessary to introduce additional assumptions regarding the way in which p_k enters the system matrices. The most common are the following:

1. Affine parameter dependence (LPV-A):

$$A(\delta_k) = A_0 + A_1 p_{1,k} + \ldots + A_s p_{s,k} \tag{3}$$

and similarly for B, C and D, and where by $p_{i,k}, i = 1, \ldots, s$ we denote the i-th component of vector p_k. This form can be immediately generalized to polynomial parameter dependence.

2. Input-affine parameter dependence (LPV-IA): this is a particular case of the LPV-A parameter dependence in which only the B and D matrices are considered as parametrically-varying, while A and C are assumed to be constant: $A = A_0$, $C = C_0$.

Identifying LPV models in general state space form is a difficult task. It is usually convenient to consider first the simplest form, i.e., the LPV-IA one, as its parameters can be retrieved by using Subspace Model Identification (SMI) algorithms for LTI systems by suitably extending the input vector. In this work the MOESP class of SMI algorithms (see [41]) has been considered. LPV-IA models also provide a useful initial guess for iterative methods which can be used for the identification of fully parameterized models in LPV-A form, along the lines of [18, 39].

The classical way to perform linear system identification is by minimizing the error between the real output and the predicted output of the model. A similar approach can be used for LPV state-space systems of the form (2). Letting the system matrices of (2) be completely described by a set of parameters θ, identification can be carried out by minimizing the cost function

$$V_N(\theta) := \sum_{k=1}^{N} ||y_k - \widehat{y}_k(\theta)||_2^2 = E_N^T(\theta) E_N(\theta),$$

with respect to θ, where

$$E_N^T(\theta) = \left[\left(y_1 - \hat{y}_1(\theta) \right)^T \quad \cdots \quad \left(y_N - \hat{y}_N(\theta) \right)^T \right],$$

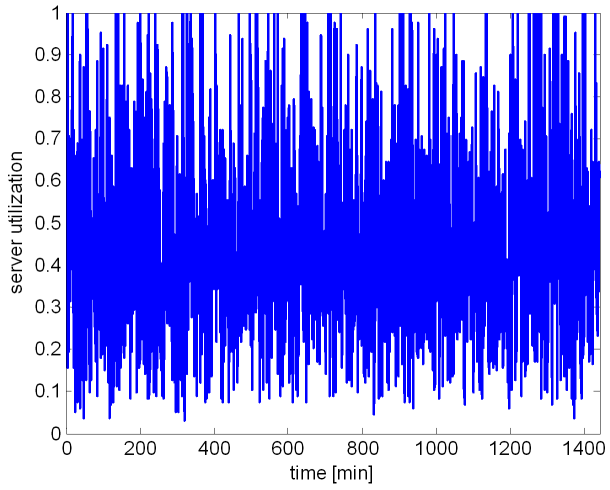


Figure 2: Time history of the physical server CPU utilization used in identification experiments.

y_k denotes the measured output and $\widehat{y}_k(\theta)$ denotes the output of the LPV model to be identified. In general, the minimization of $V_N(\theta)$ is a non-linear, non-convex optimization problem. Different algorithms exist to numerically search for a solution to such an optimization problem. One popular choice is a gradient search method known as the Levenberg-Marquardt algorithm (see [22]). This is an iterative procedure that updates the system parameters θ with the following rule

$$\theta^{(i+1)} = \theta^{(i)} - \alpha^{(i)} \left(\beta^{(i)} I + \Psi_N^T(\theta^{(i)})\Psi_N(\theta^{(i)})\right)^{-1} \times \Psi_N^T(\theta^{(i)}) E_N(\theta^{(i)}), \quad (4)$$

where $\alpha^{(i)}$ is the step size, $\beta^{(i)}$ is the Levenberg-Marquardt regularization parameter and $\Psi_N(\theta) := \frac{\partial E_N(\theta)}{\partial \theta^T}$.
The step size $\alpha^{(i)}$ can be determined by performing a line search, for example with mixed quadratic and polynomial interpolation as in the *Matlab Optimization Toolbox*.

4. EXPERIMENTAL RESULTS

In this Section the experimental setup implemented for collecting the data both for identification and validation purposes will be presented. Furthermore, the effectiveness of the LPV models to capture Web service applications behaviour will be evaluated by means of quantitative performance metrics.

Two different scenarios have been analyzed. In the first class of experiments, reported in Section 4.1, a micro benchmarking Web service application has been considered. The applications have been instrumented to accurately determine the service time of each request and to provide the best conditions for the application of LPV models.

In the second scenario, reported in Section 4.2, this latter assumption has been relaxed and the SPECweb2005 industrial benchmark suite [30] has been adopted. The data required for the parametrization of the LPV models (mainly the VMs' utilization) has been gathered from the hypervisor without instrumenting the code.

Experiments with up to 4 VMs per core have been considered. Note that, the number of VMs which can be assigned to each core in x86 environment with the *classic VMM monitor* like VMWare ESX or Xen is limited (larger installations run up to 170 VM on 32 physical cores [45], but in order to limit performance degradation most users run less than 10 VMs per server [4]). The experimental evaluation of *operating system based VMM* like OpenVZ or Virtuozzo which are intrinsically more scalable and allows supporting a larger number of VMs per core [7] is left as part of future work.

To quantitatively evaluate the models two metrics based on the results obtained by evaluating virtualized systems performance through the identified model will be considered: the percentage Variance Accounted For (VAF), defined as

$$VAF = 100\left(1 - \frac{Var[y_k - \widehat{y}_k]}{Var[y(k)]}\right), \quad (5)$$

where y_k is the measured signal and $\widehat{y}_k$ is the output obtained by the identified LPV model, and the percentage average error e_{avg}, computed as

$$e_{avg} = 100\left|\frac{E_t[y_k - \widehat{y}_k]}{E_t[y_k]}\right|. \quad (6)$$

This choice allows to assess quantitatively the model both in a 1 and in a 2-norm sense. In fact, even if the VAF metric is certainly the most common in the identification community [20], in the queueing theory field 1-norm metrics (such as the percentage average error) are widely used [17].

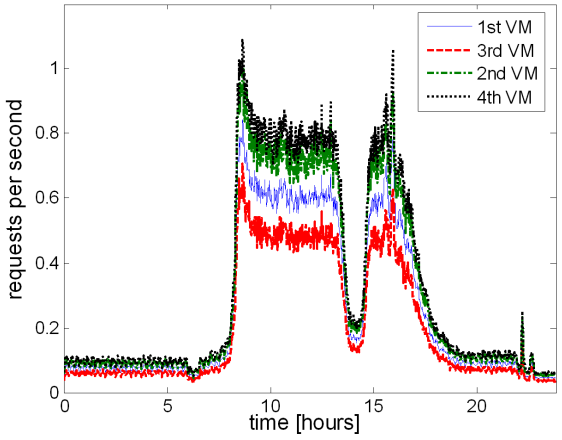


Figure 3: Time history of the request rates applied during a validation test with four VMs.

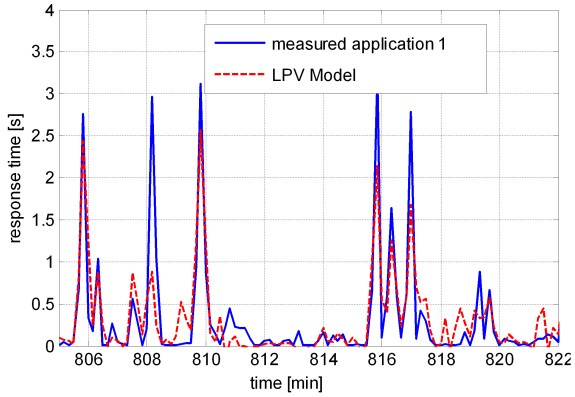
4.1 Micro-benchmarking Web Service Application Experiments

In the experimental framework, the micro-benchmarking Web service application is hosted within the Apache *Tomcat* 6.0 application server. The Web service is a Java servlet designed to consume a fixed CPU time. The workload is generated by a custom extension of the Apache *JMeter* 2.3.1 workload injector, which allows to generate workload according to an open model (see, e.g., [28]) with a Poisson arrival process. Several analyses of actual e-commerce site traces, see for example [43], have shown that the Internet workload follows a Poisson distribution as a first approximation. The analysis of burstiness behaviour and long range dependent phenomena is left as part of future work.

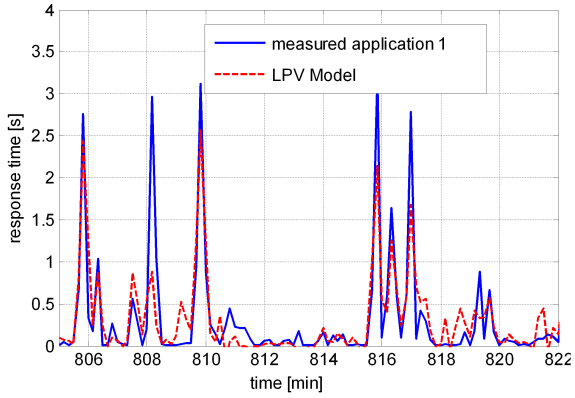
The virtualization environment is based on Xen 3.0 Fedora 8.0 distribution, which runs on a AMD Opteron 2347HE Barcelona dual socket quad-core system. Up to four instances of the micro-benchmarking Web service applications have been considered hosted in Linux Fedora 8 VMs. The VMM monitor is assigned to

a dedicated core, while the VMs are assigned to the same core by setting CPU affinity. Both identification and validation experiments have been carried out for two, three and four VMs, respectively.

The application service time s_k^i have been randomly generated in the range 0.06 s 1.1 s following a log-normal distribution (as observed for several real applications, [38]) with a coefficient of variation equal to 4 (i.e., the standard deviation is four times larger than the average service time).



(a)



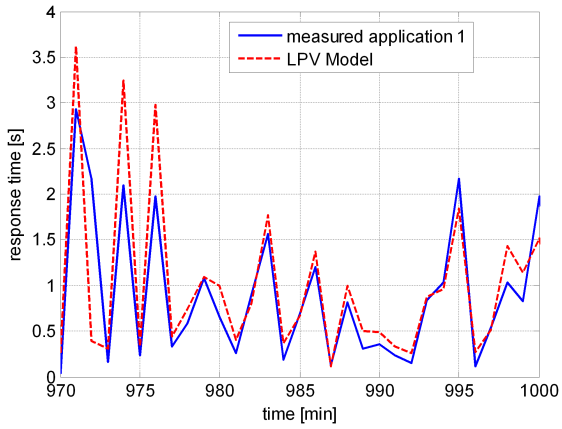
(b)

Figure 4: Detail of the measured (solid line) and response time obtained with an LPV-IA model (dashed line) for the first (a) and for the second (b) application of the VM on identification data with $\Delta t = 10$ s.

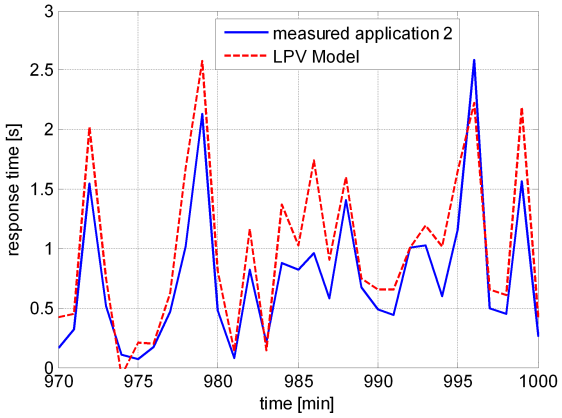
For system identification purposes, incoming request rates λ_i $i = 1, \ldots, n$ accessing each of the n VMs have been randomly generated and vary stepwise every 1 minute, with values between 0.1 req/s and 1.2 req/s according to a Poisson arrival process [46].

Overall, 1,440 intervals (24 hours) have been considered and the average utilization of the physical server (given by $\rho = \sum_{i=1}^{n} \lambda_k^i s_k^i$) has been varied in the range $[0.05, 1]$ in order to analyze the behavior of the physical server under both light load and saturation conditions. Figure 2 shows a plot of the server utilization ρ used in the identification tests. Finally, the VMM resource allocation parameters ϕ_k^i have been selected as a realization of a uniform random variable with values between 0.1 and 0.9. Note that the n-th VM is then forced to have access to a capacity of $\phi_k^n = 1 - \sum_{i=1}^{n-1} \phi_k^i$.

For model validation, a synthetic workload inspired by a real-world usage has been employed. The incoming workload reproduces a 24 hour trace obtained from an on-line banking application. The workload injector is configured to follow a Poisson process with request rate changing every minute according to the trace. Figure 3 reports, as an example, the request rates applied to the four VMs case during a validation test. The request rate follows a bimodal distribution with two peaks around 11.00 and 16.00 (see Figure 3). Resource allocation parameters have been randomly generated as in identification.



(a)



(b)

Figure 5: Detail of the measured (solid line) and response time obtained with an LPV-IA model (dashed line) with $\Delta t = 1$ min for the first (a) and for the second (b) application of the VM on validation data at high load.

Note that, the average response time R_k^i (i.e., the system output) is given by:

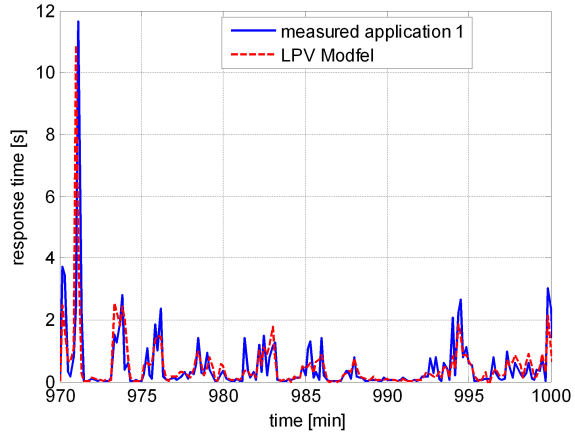
$$R_k^i = s_k^i + w_k^i, \tag{7}$$

where w_k^i denotes requests average waiting time for application i, i.e., the average time requests spend in the queue. Equation (7) must be regarded as an additional output equation of an LPV system of the form (1) with state x_k^i and output w_k^i. Therefore, R_k^i can be computed at run-time at the same time instant at which the service time s_k^i is measured and w_k^i is provided as output by the LPV model without the need to recur to any prediction technique.

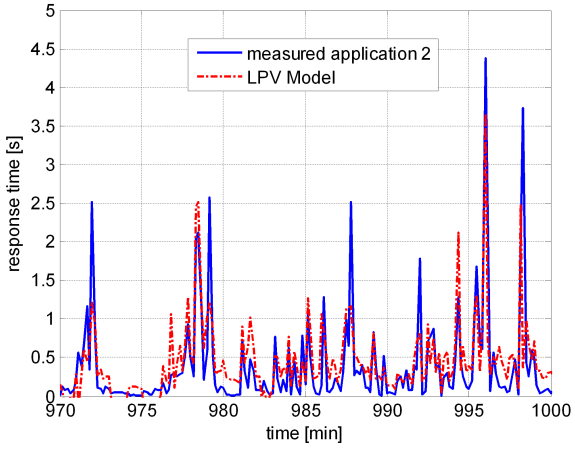
Since requests service time can be obtained by the instrumentation, the real dynamics of the system have to be found in the variable w_k^i. In this Section, MIMO LPV models will be identified for the dynamics of w_k^i only, and the final response times for each application will then be computed by equation (7). Specifically, we consider as input the VMM parameters, i.e., $u_k = [\phi_k^i]$,

$i = 1, \ldots, n-1$ and use the service times s_k^i and each application utilization $\rho_k^i = s_k^i \lambda_k^i$ as scheduling parameters, that is $p_k = [s_k^i \quad \rho_k^i]$, $i = 1, \cdots, n$. The system outputs are the service response times obtained for each application, that is $R_k^i, i = 1, \cdots, n$. The model order - after some comparative analysis - was set to 2 for the case of two VMs and to 4 for the cases of three and four VMs.

According to the results discussed in [32, 33], the LPV-IA model class was selected and the sampling time set to $\Delta t = 1$ min and $\Delta t = 10$ s, for comparison purposes. A plot of the response times obtained with such models are compared to the measured ones in Figure 4(a)-4(b) for the case of two VMs considering identification data.



(a)



(b)

Figure 6: Detail of the measured (solid line) and response time obtained with an LPV-IA model (dashed line) with $\Delta t = 10$ s for the first (a) and for the second (b) application of the VM on validation data at high load.

It is interesting to analyze the performance of such models on validation data: the details of the results are reported in Table 2 and Table 3 with a sampling interval $\Delta t = 1$ min and $\Delta t = 10$ s, respectively. There exists a trade-off between the performance (and the optimal choice of the sampling interval) in light and heavy load conditions. For the considered validation test (see also Figure 3), the light load data are those in the time interval $t \in [0,\ 8.30) \cup (13.30,\ 14.45) \cup (17.30,\ 24]$ h, while the heavy load data in the time interval $t \in [8.30,\ 13.30] \cup [14.45,\ 17.30]$ h. Considering the value of VAF, it is apparent that better performance can be achieved with a sampling interval $\Delta t = 10s$ in the heavy load, while in the light load part of the data a smaller sampling interval has to be selected (see also Figure 5(a)-5(b) and Figure 6(a)-6(b)). Indeed, at a 1 minute time scale resolution, the models cannot fully capture the heavy workload intensity, which results in a significant underestimation of the response time in heavy load conditions (see Figure 5(a)-5(b)). These data show that the identified models are extremely effective at light load but lose generalization capabilities when tested at heavy load.

This is due to the fact that, using as sampling interval $\Delta t = 1$ min, the request rates used as inputs in the identification phase are stepwise constant, and cannot account for the non-uniformity of the single arrival times within the interval itself. By analyzing the effective arrival times at which the single requests are issued in the system, the high correlation between such data and the peaks in the response time is apparent.

To capture this variability, the sampling interval must be reduced. By adopting $\Delta t = 10$ s, the effectiveness of having reduced the sampling interval is apparent: now the models are capable of providing a response time which correctly follows the peaks of the measured one (see Figure 5(a)-5(b)). Results are quantitatively reported in Table 3. Note that, at 10s time resolution in each test the average error is also more uniform among VMs.

	$\Delta t = 1$ min	$\Delta t = 10$ s
2 VM	0.129 s	1.109 s
3 VM	0.690 s	4.696 s
4 VM	1.899 s	13.691 s

Table 1: CPU time needed for LPV model identification.

The CPU time needed to perform the identification of the LPV-IA models for the case of two, three and four VMs are reported in Table 1. These data have been obtained on a PC Intel Core-Duo at 3GHz with 4 GB Ram using Matlab 7.04.

Note that, as the model will be finally employed to design a control law, at run-time only the control law itself will be computed. The control law complexity can vary according to the chosen control strategy. In the case where an LPV controller is designed, it will result in a dynamic LPV model of the same complexity as that of the identified ones. As such, the update of the control law requires a matrix-vector product which can be computed in few CPU milliseconds.

In general, these results confirm the suitability of the proposed models for control design purposes and run-time performance assessment. Since, as the incoming traffic can be regarded as a measurable disturbance and as the computational complexity and available memory pose no particular constraints in this type of applications, it is possible to have different models (and corresponding controllers) on board of the server (the one identified on a 10s and on 1 min basis), which can be selected based on workload conditions. Further observations concerning the fact that the use of a short sampling interval causes the models to privilege the explanation of fast transients phenomena, i.e., those pertaining to heavy load conditions, can be found in [32, 33]. As a final consideration, increasing the number of VMs requires to increase also the order of the model. This could be expected, since the behaviour of the GPS scheduling implemented by the VMM depends on the queue length of each single VM which is captured as a part of the state of the LPV model.

		VAF on 24h	VAF light-load	VAF heavy-load	e_{avg} on 24h	e_{avg} light-load	e_{avg} heavy-load
2 VMs	VM1	82.21%	82.8%	82.48%	5.46%	1.43%	13.4%
	VM2	73.41%	72.49%	82.22%	2.63%	6.42%	13.18%
3 VMs	VM1	81.59%	78.33%	88.26%	4.61%	4.13%	5.38%
	VM2	81.1%	79.45%	86.17%	1.16%	1.59%	6.08%
	VM3	72.97%	73.48%	61.62%	11.07%	12.84%	7.63%
4 VMs	VM1	78.66%	72.45%	90.09%	8.12%	8.73%	7.18%
	VM2	73.12%	70.69%	86%	11.36%	14.98%	4.79%
	VM3	73.57%	77.67%	67.88%	7.74%	7.53%	8.02%
	VM4	56.27%	52.52%	85.67%	15.06%	19.04%	7.15%

Table 2: Performance of the identified models with $\Delta t = 1$min on validation data.

		VAF on 24h	VAF light-load	VAF heavy-load	e_{avg} on 24h	e_{avg} light-load	e_{avg} heavy-load
2 VMs	VM1	74.85%	74.8%	86.15%	6.5%	8.3%	4.07%
	VM2	70.56%	67.9%	83.94%	3.4%	2.5%	4.17%
3 VMs	VM1	78.5%	75.19%	83.05%	6.9%	1.5%	12.76%
	VM2	78.46%	75.51%	88.78%	6.5%	3.27%	19.58%
	VM3	70.75%	69.14%	83.3%	2.76%	16.55%	18.21%
4 VMs	VM1	75.97%	66.37%	88.97%	6.91%	10.35%	3.16%
	VM2	72.72%	68.1%	87.23%	9.14%	12.23%	5.24%
	VM3	74.85%	69.22%	82.94%	5.84%	7.29%	4.38%
	VM4	66.80%	62.38%	89.06%	11.99%	16.92%	4.77%

Table 3: Performance of the identified models with $\Delta t = 10$s on validation data.

4.2 SPECweb2005 Experiments

SPECweb2005 is the industry standard benchmark for the performance assessment of Web servers. The benchmark includes three different loads, *banking*, *e-commerce* and *support* which simulate respectively the access to an on-line banking, an on-line trading and an enterprise products catalogue Web site. In the following a two VMs test case running the banking and e-commerce loads will be discussed, the support load analysis is left as part of future work.

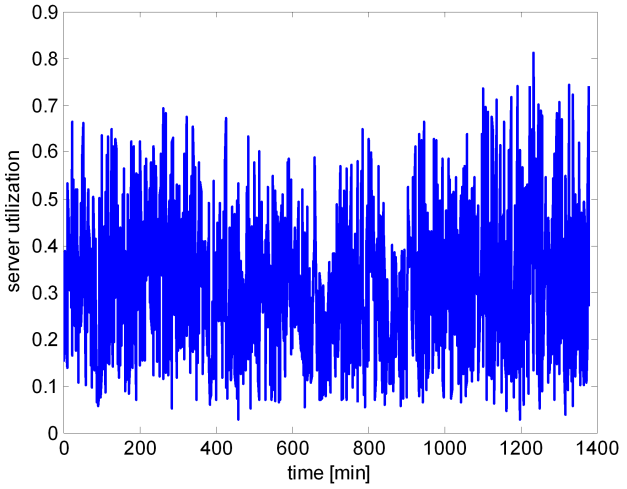


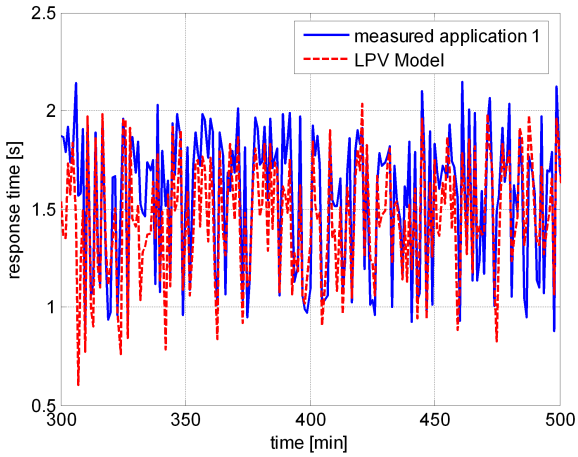
Figure 7: Time history of the physical server CPU utilization used in identification experiments in scenario 1.

The benchmark suite includes four components: the load generators, the client coordinator, the Web server, and the back-end simulator. The SPECweb2005 load generators inject workload to the system according to a closed model. Users sessions are started according to a given number of users who continuously send requests for dynamic Web pages, wait for an average 10s think time, and then access another page or leave the system according to a pre-defined session profile. The client coordinator initializes all the other systems, monitors the test, and collects the results. The Web server is the component target of the performance assessment, while the back-end simulator emulates the database and application parts of the benchmark and is used to determine the dynamic content of the Web pages.

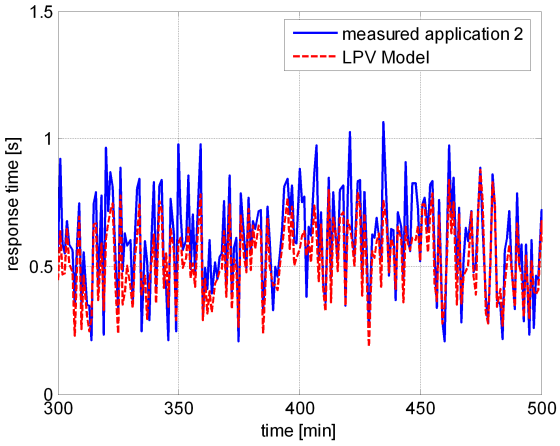
Our experimental setup includes an AMD machine configured as in the previous Section which runs into independendent VMs two load generators, two client coordinators and two backend simulators. The load generators are configured to access two different VMs hosting the Web components which provide the pages of the e-commerce and banking loads. These latter VMs run on a Xen 3.3 Ubuntu Server 8.10 environment, supported by an Intel Nehalem dual socket quad-core system. Also in this case, the Xen hypervisor is assigned to a dedicated core, while the two VMs under analysis are assigned to the same core by setting CPU affinity.

To assess the suitability of LPV models two different scenarios have been considered. In the first scenario the workload profiles and resource allocation parameters have been randomly generated, while in the second one they are inspired to real application settings. In both cases the model order - after some comparative analysis - was set equal to 4.

In the first scenario, two data sets have been generated (one for identification and one for validation purposes), where the number of users N_k^i accessing each of the two VMs varied stepwise every 1 minute, with values between 10 and 220. Also in this case, 1,440 intervals (24 hours) have been considered and the average utilization of the physical server varied in the range $[0.02, 0.8]$. Figure 7 shows a plot of the server utilization ρ obtained in the identification tests.



(a)



(b)

Figure 8: Detail of the measured (solid line) and response time obtained with an LPV-IA model (dashed line) with $\Delta t = 1$ min for the first (a) and for the second (b) application on validation data measured with SPECweb2005 in the first scenario.

As in the previous Section, the VMM resource allocation parameters ϕ_k^i have been selected as a realization of a uniform random variable with values between 0.1 and 0.9.

In this experimental setting, MIMO LPV models will be identified for the dynamics of R_k^i, as only the the VMs' utilization is gathered from the hypervisor, while the service time s_k^i is not directly measured, as the code was not instrumented (see also Section 4). Response time data have been obtained by the client coordinators, while the utilization of the VMs has been obtained by *xentop* [48] with a 5s time resolution.

Similarly to the case discussed in Section 4.1, we consider as input the VMM parameters, i.e., $u_k = [\phi_k^i], i = 1, \ldots, n-1$ but now we use the number of users N_k^i and the overall server utilization ρ_k as scheduling parameters, that is $p_k = [N_k^i \quad \rho_k]$, $i = 1, \cdots, n$. The system outputs are the service response times obtained for each application, that is $R_k^i, i = 1, \cdots, n$.

A plot of the response times obtained with the identified models are compared to the measured ones in Figure 8(a)-8(b) obtained on validation data in the first scenario. The quantitative measures of the obtained performance both on identification and validation data are reported in Table 4 and confirm the effectiveness of the proposed models also on a real benchmark application.

		VAF on 24h	e_{avg} on 24h
Identification data	VM1	60.74%	4.05%
	VM2	77.75%	4.15%
Validation data	VM1	56.32%	6.48%
	VM2	77.16%	10.39%

Table 4: Performance of the identified models with $\Delta t = 1$ min on identification and validation data in scenario 1.

In the second scenario, both in identification and validation, a synthetic workload similar to the one reported in Figure 3 obtained from an on-line banking application has been employed. In this scenario the e-commerce and banking loads have been shifted by 12 hours in order to obtain a more homogeneous overall incoming workload with lower peaks (see Figure 9). In this way, the workloads have more heterogeneous profiles, with peaks in different time periods, thus offering more opportunities for resource sharing. Note that, this is representative for a service provider with customers geographically distributed across different time zones and identification data can be obtained by inspecting system logs.

Both ϕ_k^i and N_k^i varied stepwise every 1 minute and ϕ_k^i were set equal to:

$$\phi_k^i = \max\left(0.1, \frac{N_k^i}{N_k^1 + N_k^2}\right)$$

i.e., the capacity of the competing VMs is heuristically assigned proportional to the incoming workload but a 10% reservation is dedicated to the lighted loaded application in order to avoid starvation. Note that this proportional allocation scheme is a natural way to assign the server capacity. It is provably the best resource allocation scheme in terms of stability regions and it is used as a benchmark in the autonomic computing literature (see e.g., [19]).

The average utilization of the physical server varied in the range $[0.05, 0.65]$. Figure 10 shows a plot of the server utilization ρ_k obtained in the identification tests.

The input/output configuration and the parameter vector are the same as in the first scenario. A plot of the response times obtained with the identified models are compared to the measured ones in Figure Figure 11(a)-11(b) obtained on validation data in the second scenario, while detailed quantitative results are reported in Table 5. Inspecting this latter Table it is apparent that, also in this challenging test performed on a standard benchmark and considering realistic workload and resource allocation policies, the proposed identification approach remains reliable and effective.

		VAF on 24h	e_{avg} on 24h
Identification data	VM1	59.85%	6.85%
	VM2	87.20%	6.97%
Validation data	VM1	64.51%	7.34%
	VM2	77.60%	10.63%

Table 5: Performance of the identified models with $\Delta t = 1$ min on identification and validation data in scenario 2.

LPV models have demonstrated to be able to capture the behavior of a virtualized server at very fine grained time scale with an average percentage error lower than 20% in the worst case.

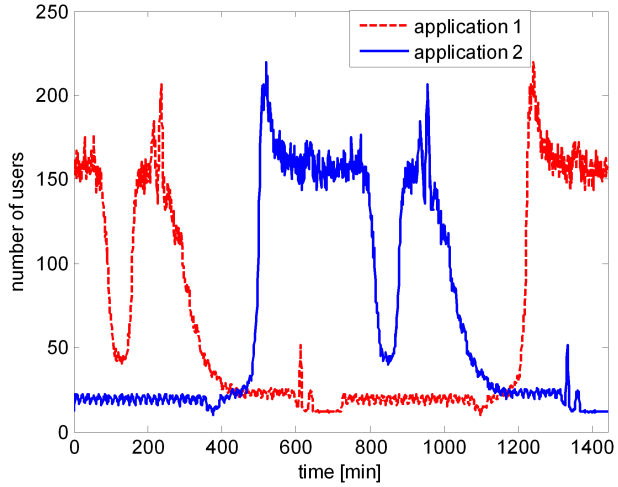


Figure 9: Plot of the number of users employed in scenario 2.

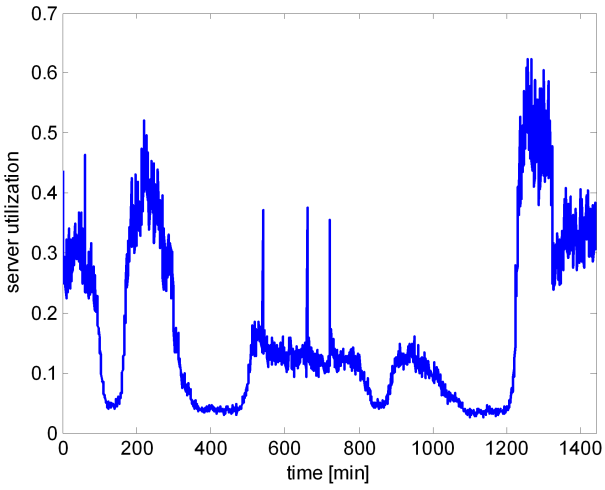
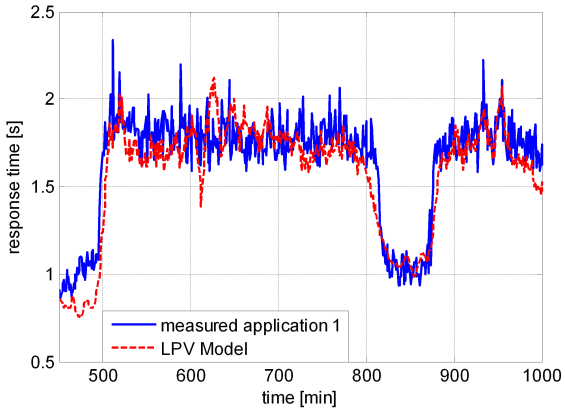
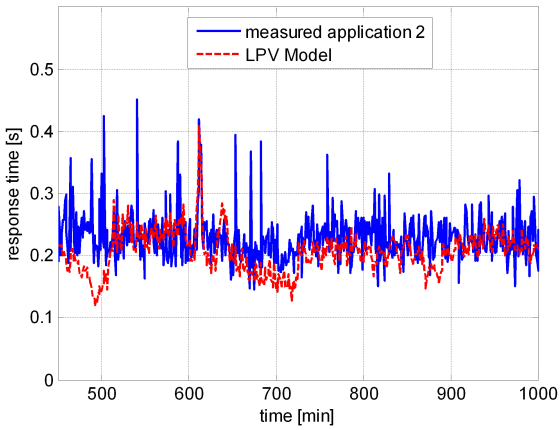


Figure 10: Time history of the physical server CPU utilization used in identification experiments in scenario 2.



(a)



(b)

Figure 11: Detail of the measured (solid line) and response time obtained with an LPV-IA model (dashed line) with $\Delta t = 1$ min for the first (a) and for the second (b) application on validation data measured with SPECweb2005 in scenario 2.

5. RELATED WORK

The performance evaluation of a virtualized systems under work-conserving mode is very challenging. The GPS scheduling discipline implemented by VMMs belongs to the *non-product form* class. In the queueing network literature, a number of solutions have been proposed for the analysis of GPS systems.

Authors in [9] have analyzed the steady state behaviour of a class of two-dimensional birth-and-death processes, which can approximate the execution of two running VMs. In particular a two-class M/M/c system with "mixed priorities' is analyzed. Of the c servers, k are devoted to class 1 jobs execution, and the other $c-k$ are designated class 2. Class 1 jobs have preemptive priority over class 2 jobs on class 1 servers, while class 2 jobs have preemptive priority on class 2 servers. Thus the serving capacity available to class 1 jobs is k when there are at least $c-k$ class 2 jobs in the system, but can rise (up to c) when that number drops below $c-k$. Similarly for the class 2 jobs. The analysis provided a solution method by means of the generating functions for the steady-state distribution and can approximate the GPS scheduling when c is large. In [49], bounding techniques are presented which are able to determine the percentile distribution of response time of multiple-classes. Anyway, the accuracy of the bounds has not been evaluated. A lot of work has been

performed also in the IP network literature since the GPS scheduling can be implemented by advanced routers in order to provide service differentiation to voice and video packets. In [13] bounding techniques have been proposed for evaluation of multiple-class systems characterized by long range dependent traffic.

The above mentioned techniques are able to predict performance of a system when it is in the steady state and therefore cannot accurately model system transients. Vice versa, LPV models and genuine feedback control techniques borrowed from the system identification and control theory research, relying on accurate dynamical models, can predict systems performance at very fine grained time scales and adjust the system configuration within the time frame of a transient. The first control-oriented contributions applied to the management of autonomic systems are reported in [1, 27, 31], and use feedback control to limit the utilization of bottleneck resources by means of admission control and resource allocation. More recent works [16] implemented a limited look-ahead controller for the autonomic management of a service center.

The first works which adopt the LPV framework to model and control server systems have been presented in [25, 24], where an input-output LPV representation has been identified to serve as a basis for the design of a gain-scheduling controller able to provide performance guarantees by means of frequency scaling and admission control, respectively. More recently, see [32, 33, 35], the authors of the present paper have proposed state-space subspace-based identification methods to model single class Web service systems where the control variable is the frequency of operation of the CPU. The adoption of the state space representation allows a seamless extension to Multiple-Input-Multiple-Output systems, needed to address multiple request classes, and a more sound model structure selection for LPV control design purposes (see [36]). The preliminary results we have achieved in virtualized infrastructures limited to a two VMs case are reported in [34]. This paper extends this latter previous work providing a more exhaustive experimentation, considering a larger number of VMs and validating the approach by adopting the SPECweb2005 industrial benchmark.

In the queueing theory literature, some recent proposals addressed the problem of modelling queue network transient behaviour by means of Markov models in order to study burstiness and long range dependency in system workloads. Authors in [5] studied a class of closed queueing networks where service times are represented by Markovian arrival processes which take into account time dependent features such as burstiness in service times. In [5], very accurate upper and lower bounds are also provided to determine arbitrary performance indexes, but the analysis is limited to single class models. Authors in [26] introduced matrix-analytic analysis to study an open queueing system with Markovian Arrival Process as input and Phase Type distribution as service time. Other studies, [29], focused on the analyses of non-renewal workloads but, due to the analysis complexity, only small size models based on one or two queues can be dealt with so far.

The main limitation of Markovian models is their complexity, which makes, even for very simple systems, e.g., a first come first served single server queue, the number of parameters to be determined quite large. These models suffer for high computation overhead and, hence, are presently not suitable for run-time modelling of virtualized systems.

6. CONCLUDING REMARKS AND OUTLOOK

This paper presented the identification of multi-variable LPV models for the performance control of virtualized environments. Specifically, the suitability of subspace LPV identification methods has been checked against experimental data measured on a custom implementation of a a micro-benchmarking Web service application and on the SPECweb2005 industry benchmark endowed with virtualization capabilities. A detailed analysis of the peculiarities of the considered applications has been provided, together with a quantitative assessment of the performance of the LPV models on validation data. Future work will be devoted to further validate our approach on real applications and to employ the identified model for control purposes.

Acknowledgements

The authors gratefully acknowledge the contribution of Luca Destito, Paolo Sala, Giovanni Pirotta, Gianluca Pisati, and Mauro Speroni for their development and test activities. Thanks are also due to Dr. Raffaela Mirandola for many fruitful discussions, to Dr. Giuseppe Amato of AMD Italy and to Dr. Massimo Leoni of IBM Italy for providing the physical infrastructure supporting the experiments and real applications trace data.

7. REFERENCES

[1] T. Abdelzaher, K. G. Shin, and N. Bhatti. Performance Guarantees for Web Server End-Systems: A Control-Theoretical Approach. *IEEE Transactions on Parallel and Distributed Systems*, 15(2), 2002.

[2] D. Ardagna, M. Trubian, and L. Zhang. SLA based resource allocation policies in autonomic environments. *Journal of Parallel and Distributed Computing*, 67(3):259–270, 2007.

[3] B. Bamieh and L. Giarré. Identification of linear parameter varying models. *International Journal of Robust and Nonlinear Control*, 12(9):841–853, 2002.

[4] B. Botelho. Virtual machines per server: A viable metric for hardware selection? http://itknowledgeexchange.techtarget.com/server-farm/virtual-machines-per-server-a-viable-metric-for-hardware-selection/.

[5] G. Casale, N. Mi, and E. Smirni. Bound Analysis of Closed Queueing Networks with Workload Burstiness. In *Proc. of SIGMETRICS*, 2008.

[6] J. S. Chase and D. C. Anderson. Managing Energy and Server Resources in Hosting Centers. In *ACM Symposium on Operating Systems principles*, 2001.

[7] L. Cherkasova, D. Gupta, and A. Vahdat. Comparison of the three CPU schedulers in Xen. *SIGMETRICS Performance Evaluation Review*, 25(2):42–51, 2007.

[8] I. Cunha, J. Almeida, V. Almeida, and M. Santos. Self-adaptive capacity management for multi-tier virtualized environments. In *Integrated Network Management*, pages 129–138, 2007.

[9] G. Fayolle, P. King, and I. Mitrani. The Solution of Certain Two-Dimensional Markov Models. In *PERFORMANCE80 Proc.*, 1980.

[10] F. Felici, J. van Wingerden, and M. Verhaegen. Subspace identification of MIMO LPV systems using a periodic scheduling sequence. *Automatica*, 43(10):1684–1697, 2007.

[11] S. Govindan, A. R. Nath, A. Das, B. Urgaonkar, and A. Sivasubramaniam. Xen and co.: communication-aware cpu scheduling for consolidated xen-based hosting platforms. In *VEE*, pages 126–136. ACM, 2007.

[12] D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat. Enforcing performance isolation across virtual machines in xen. In *Middleware*, 2006.

[13] X. Jin and G. Min. Analytical modelling and evaluation of generalized processor sharing systems with heterogeneous traffic. *International Journal of Commununication Systems*, 21:571–586, 2008.

[14] T. Katayama. *Subspace Methods for System Identification*. Springer, 2005.

[15] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[16] D. Kusic, J. O. Kephart, N. Kandasamy, and G. Jiang. Power and Performance Management of Virtualized Computing Environments Via Lookahead Control. In *ICAC 2008 Proc.*, 2008.

[17] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative System Performance: Computer System Analysis Using Queueig Network Models*. Prentice-Hall, 1984.

[18] L. Lee and K. Poolla. Identification of linear parameter-varying systems using nonlinear programming. *ASME Journal of Dynamic Systems, Measurement and Control*, 121(1):71–78, 1999.

[19] Z. Liu, M. Squillante, and J. L. Wolf. On Maximizing Service-Level-Agreement Profits. In *Proc. of ACM Eletronic Commerce Conference*, October 2001.

[20] M. Lovera, M. Verhaegen, and C. T. Chou. State space identification of MIMO linear parameter varying models. In *Proceedings of the International Symposium on the Mathematical Theory of Networks and Systems*, pages 839–842, Padua, Italy, 1998.

[21] D. A. Menascé. Virtualization: Concepts, applications, and performance modeling. In *Int. CMG Conference*, 2005.

[22] J. J. Moré. The Levenberg-Marquardt algorithm: Implementation and theory. In G. A. Watson, editor, *Numerical Analysis*, volume 630 of *Lecture Notes in Mathematics*, pages 106–116. Springer Verlag, 1978.

[23] F. Previdi and M. Lovera. Identification of a class of nonlinear parametrically varying models. *International Journal on Adaptive Control and Signal Processing*, 17(1):33–50, 2003.

[24] W. Qin and Q. Wang. An LPV approximation for admission control of an internet web server: identification and control. *Control Engineering Practice*, 15(12):1457–1467, 2007.

[25] W. Qin and Q. Wang. Modeling and control design for performance management of web servers via an LPV approach. *IEEE Transactions on Control Systems Technology*, 15(2):259–275, 2007.

[26] A. Riska, M. Squillante, S. Z. Yu, Z. Liu, and L. Zhang. *Matrix-Analytic Analysis of a MAP/PH/1 Queue Fitted to Web Server Data. In Matrix-Analytic Methods: Theory and Applications, G. Latouche and P. Taylor Ed.*, pages 335–356. World Scientific, 2002.

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

[28] B. Schroeder, A. Wierman, and M. Harchol-Balter. Open vs closed: a cautionary tale. In *NSDI 2006 Proc.*, 2006.

[29] A. Shwartz and A. Weiss. Multiple time scales in markovian ATM models i. formal calculations, 1999.

[30] Standard Performance Evaluation Corporation. SPECweb2005.

[31] J. S. T. Abdelzaher and, C. Lu, R. Zhang, and Y. Lu. Feedback Performance Control in Software Services. *IEEE Control Systems Magazine*, 23(3):21–32, 2003.

[32] M. Tanelli, D. Ardagna, and M. Lovera. LPV model identification for power management of web service systems. In *2008 IEEE Multi-conference on Systems and Control, San Antonio, USA.*, 2008.

[33] M. Tanelli, D. Ardagna, and M. Lovera. On- and off-line model identification for power management of web service systems. In *2008 IEEE Multi-conference on Decision and Control, Cancun Mexico.*, 2008.

[34] M. Tanelli, D. Ardagna, and M. Lovera. LPV model identification in virtualized service center environments. In *15th IFAC Symposium on System Identification (SYSID 2009), Saint-Malo, France.*, 2009.

[35] M. Tanelli, D. Ardagna, M. Lovera, and L. Zhang. Model Identification for Energy-Aware Management of Web Service Systems. In *ICSOC08 Proc.*, 2008.

[36] R. Toth, F. Felici, P. Heuberger, and P. V. den Hof. Discrete time LPV I/O and state space representations, differences of behavior and pitfalls of interpolation. In *Proceedings of the 2007 European Control Conference, Kos, Greece*, 2007.

[37] B. Urgaonkar, G. Pacifici, P. J. Shenoy, M. Spreitzer, and A. N. Tantawi. Analytic modeling of multitier Internet applications. *ACM Transaction on Web*, 1(1), January 2007.

[38] E. Veloso, V. Almeida, W. M. Jr., A. Bestavros, and S. Jin. A Hierarchical Characterization of a Live Streaming Media Workload. *IEEE/ACM Transactions on Networking*, 14(1), 2006.

[39] V. Verdult. *Nonlinear System Identification: A State-Space Approach*. PhD thesis, University of Twente, Faculty of Applied Physics, Enschede, The Netherlands, 2002.

[40] V. Verdult, M. Lovera, and M. Verhaegen. Identification of linear parameter-varying state space models with application to helicopter rotor dynamics. *International Journal of Control*, 77(13):1149–1159, 2004.

[41] M. Verhaegen. Identification of the deterministic part of MIMO state space models given in innovations form from input output data. *Automatica*, 30:61–74, 1994.

[42] M. Verhaegen and V. Verdult. *Filtering and System Identification: A Least Squares Approach*. Cambridge University Press, 2007.

[43] D. Villela, P. Pradhan, and D. Rubenstein. Provisioning Servers in the Application tier for e-commerce Systems. In *IWQOS 2004 Proc.*, 2004.

[44] VMWare. Business Solution Datacenter.

[45] VMWare. Configuration Maximums. http://www.vmware.com/pdf/.

[46] A. Williams, M. Arlitt, C. Williamson, and K. Barker. Web Workload Characterization: Ten Years Later, 2006.

[47] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif. Black-box and Gray-box Strategies for Virtual Machine Migration. In *USENIX*, 2007.

[48] Xen. The Xen Hypervisor.

[49] Z. L. Zhang, D. Towsley, and J. Kurose. Statistical analysis of the generalized processor sharing scheduling discipline. In *IEEE Journal on Selected Areas in Comm.*, 2003.