# A control-theoretic approach towards joint admission control and resource allocation of cloud computing services

Dechouniotis, D., Leontiou, N., Athanasopoulos, N., Christakidis, A., & Denazis, S. (2015). A control-theoretic approach towards joint admission control and resource allocation of cloud computing services. *International Journal of Network Management*, *25*(3), 159-180. https://doi.org/10.1002/nem.1889

**Published in:**
International Journal of Network Management

**Document Version:**
Peer reviewed version

# A Control-Theoretic Approach Towards Joint Admission Control and Resource Allocation of Cloud Computing Services

Dimitrios Dechouniotis[1], Nikolaos Leontiou[1], Nikolaos Athanasopoulos[2], Athanasios Christakidis[1] and Spyros Denazis[1]
[1]Electrical and Computer Engineering Department,University of Patras, Rion Patras 26500, Greece.
{ddexouni, nleontiou, schristakidis, sdena}@ece.upatras.gr
[2]Electrical Engineering Department, Eindhoven University of Technology, Eindhoven, 5600 MB,the Netherlands.
n.athanasopoulos@tue.nl

**Abstract**

Meeting the performance specifications of consolidated web services in a data center is a challenging research problem, since the control of the underlying cloud computing infrastructure must meet the Service Level Agreement (SLA) requirements and satisfy the system's constraints. In this article, we address the admission control and resource allocation problem jointly, by establishing a unified modeling and control framework. Convergence to a desired reference point and stability and feasibility of the control strategy are guaranteed, while achieving high performance of the co-hosted web applications. The efficacy of the proposed approach is illustrated in a real testbed.

## 1 Introduction

Virtualization technology allows the consolidation of many services on a server cluster. The co-hosted applications share the available resources according to their requirements. This implies that providers allot the computational and network resources to the competing services and customers pay only for the consumed resources according to a usage-based price model. The performance of a web service is described in a Service Level Agreement (SLA) between the service provider and the customer; the Service Level Objectives (SLOs) are the network metrics, which prescribe the desired Quality of Service (QoS) levels. From the customer's view, the goals are the satisfaction of some predefined nominal values of the SLOs and a guaranteed level of QoS with the minimum financial cost. These objectives usually conflict with the goals of the provider,

who aims for a management system that allocates the resources to each service optimally in a manner that achieves SLOs, ensures availability of resources under any workload circumstances and minimizes the operational cost, e.g., the energy consumption. In addition, the workload of these web services is generally unpredictable and highly variant, while the available resources are subject to constraints. Due to the above contradictory targets between the customer and provider, quantifying and managing the performance of such complex systems is a critical problem.

There are several emerging challenges that an autonomic management framework must address in order to achieve the above goals. Network controllers are implemented in two levels, namely, the local level and the global level. Global level controllers focus on load balancing, which shares the volume of requests among the replicas of an application. Additionally, they arrange the Virtual Machines (VM) placement, which determines the set of applications executed by each server. Local level controllers deal with application specific or server specific problems, i.e., the admission control (AC) that rejects requests under peak workload conditions, and the resource allocation (RA), which assigns to each VM the available resources (CPU, memory, network bandwidth). Admission control is implemented by either changing the capacity of the VM or by using the Dynamic Voltage and Frequency Scaling (DVFS) [26] mechanism, thus changing the server's service rate.

Contrary to most existing approaches, in this article we propose a local controller scheme that addresses admission control and resource allocation simultaneously in a unified framework, thus, making the cooperation with existing global level controllers easier.

In order to capture the dynamic behaviour of the response times of the hosted applications and their dependencies from the request rates, the allocated CPU capacities and the percentage of the accepted requests, we utilize a Linear Parameter Varying (LPV) systems [30] modeling framework. LPV models have a structure similar of a linear system, which however varies in time, depending on a set of parameters that relate to the behaviour of the system. Thus, we are able to achieve a tradeoff between the simplicity of the resulting model and the accurate description of the system dynamics. Moreover, using off-the shelf identification algorithms for LPV systems [9] we are able to create such models using a set of data, which is available in most cloud computing architectures.

Next, based on the established LPV state space model, we determine a set of feasible operating set-points by solving off-line optimization problems. In detail, we compute feasible operating points that satisfy predetermined desired QoS nominal values, i.e., sets of desired response times for each application, allocated CPU capacities and percentages of admittance of requests. Moreover, we compute state feedback control laws which guarantee constraint satisfaction and convergence of the closed-loop system to the desired operating point. The control strategies are computed using quadratic Lyapunov functions [19]. Loosely speaking, a Lyapunov function is a distance metric that describes how far the states of the system are from the desired operating point. These functions are utilized to design controllers which guarantee that the distance from

the operating point is strictly decreasing at each time instant. Since we obtain different controllers for each desired operating point, we propose a stabilizing switching scheme between such different operating points by computing and appropriately ordering their domains of attraction. The computational complexity of the controller implementation is small, since at every time instant a linear programming problem and a point location problem are required to be solved.

The remaining of the paper is structured as follows. Section 2 discusses related work. Section 3 contains an analytical description of the system identification, the determination of feasible equilibrium points and the controller synthesis and implementation. In section 4, the application and evaluation of the approach in a real testbed is presented, along with a comparison with existing methods. Conclusions are drawn in section 5. For clarity of exposition, some technical background and results regarding the controller synthesis and implementation are presented in the Appendix.

## 2 Related Work

We categorize the existing approaches that concern the modeling and control of web services according to the employed model and the control method.

The models of consolidated web services on a cloud computing infrastructure are derived mainly from queuing theory and discrete-time state space models. In specific, the models based on queuing theory rely on the relatively strong assumption that the system is at the steady state condition. On the other hand, linear time invariant (LTI) state-space models can efficiently describe the transient behavior of the system. However, the model is only accurate close to an operation point. The most promising category of state space models employed are LPV models, since they are allowed to vary according to a modeling parameter. This modeling approach is suitable for describing the dynamics of web services behavior, because it captures the effect of various parameters such as incoming request rate or service time. Table 1 summarizes the benefits and drawbacks of each modeling method that is used in the relevant studies.

The second categorization of the existing approaches concerns the control method used, namely the admission control and/or the resource allocation. In specific, resource allocation approaches can be distinguished between those that change dynamically the CPU capacity of service VMs and those that consider as control variable the CPU frequency of the servers with fixed CPU capacity. The above techniques can be combined with the addition or migration of VM using a pool of idle servers. Most of the resource allocation solutions assume that servers are protected from overloading. However, admission control and resource allocation are two problems that need to be solved jointly in order to ensure the performance of the web service under any workload variations. Table 2 summarizes the benefits and drawbacks of each control method used in the related studies.

Independent of the modeling and control approach, it appears that no existing method considers the explicit determination of a desired operating point.

| | Strengths | Weaknesses | Related Studies |
|---|---|---|---|
| Queuing Models | - Scalability. | - Valid only in steady state.<br>- Specific workload distributions. | [6]- [8], [17], [20], [21], [23], [31], [33], [34] |
| LTI Models | - Capture transient behaviour. | - Scalability.<br>- Accuracy. | [13], [22], [35] |
| LPV Models | - Capture transient behaviour.<br>- Embed system's parameters.<br>- Scalability.<br>- Accuracy. | - Identification Procedure. | [16], [26]- [28] |

Table 1: Comparison of existing modeling approaches.

The LTI or LPV models combined with feedback control infer an operation point from measurements or the system identification procedure. However, if the state variables are far away from that nominal operation point, the underlying servers are saturated and the model becomes non-linear. Similarly, queuing theory approaches combined with an optimization method suffer from the same limitation. In detail, queuing models assume specific distribution (e.g., M/G/1) for the incoming request and service rate and determine the operating point only for steady state conditions.

In the following paragraphs, we review some representative studies of interest that are close to the approach proposed here. Ardagna et al. [7] presented a distributed algorithm for capacity allocation and load balancing. The system's dynamic behavior was modeled by an M/G/1 queue, while the joint control problem was solved as as an optimization problem applying a decomposition technique for nonlinear programming. Although the resource allocation is simplified, since they assume single tier web applications, VMs have predefined fixed capacity and are homogeneous in terms of resources (CPU, RAM). In [6], a joint admission control and resource allocation framework utilized queuing theory to capture systems dynamics. The two controllers were separated in two different optimization subproblems that were solved sequentially in an iterative fashion. However, it was not examined whether the sequence of solutions provides any performance guarantee. Kusic et al. [20], [21] derived an analytical mathematical model from queuing theory and tackled resource provisioning as an optimization problem solved with a predictive control method. Some system parameters (e.g., service rate) were empirically computed while the implementation considered the VM placement and capacity allocation problems separately. Urgaonkar et al. [31] presented a utility based solution, which maximizes the average application throughput and energy cost of the data center. Using queu-

| | Strengths | Weaknesses | Related Studies |
|---|---|---|---|
| Admission Control (AC) | - Overloading Protection. | - Fixed capacity.<br>- Rejection of requests under heavy workload. | [16], [17], [22], [23] |
| Resource Allocation (RA) | - Dynamic capacity. | - No performance guarantee under heavy workload. | [7], [20], [21], [27]-[29], [33]- [35] |
| AC+RA | - Stability guarantee.<br>- Overloading protection.<br>- Dynamic capacity. | - Complexity. | [6], [8], [13], [26], [31] |

Table 2: Comparison of existing control approaches.

ing theory models and an optimization technique framework, they addressed the problems of admission control, server selection and resource allocation separately. In [8], the authors proposed a holistic approach for application placement, admission control, resource allocation. They used queuing theory models and a greedy algorithm to solve the placement problem and consequently split admission control and capacity allocation as two decoupled subproblems. There, admission controller is designed separately from [12], ignoring the resource allocation solution. Wang et al. [33] proposed a two level controller combined with queuing modeling. The global level determines if the CPU share is efficient for the total incoming workload and it activates a certain number of servers. The local controller solves the resource allocation problem in order to satisfy the predicted workload. However, it is assumed that the incoming load to each application replica is proportional to the CPU share and a linear relationship between CPU share and processing rate is proposed. Since all the above solutions use queuing models they are valid only for steady state conditions and they do not examine if the operational point is feasible or not.

In [22], a feedback admission controller using an LTI model was proposed, which also takes into account CPU utilization to regulate the admitted workload. ACRA [13] is an integrated framework that solves jointly the admission control and resource allocation problems using a group of LTI models, which is not scalable as the number of consolidated services increases. Furthermore, it was shown that the determined equilibrium point is always feasible. Yaksha [17] presented an admission controller, which is based on an initial queuing model and an LTI model that is derived from the linearisation of the queuing model, while a proportional-integral (PI) feedback controller is designed. However, the obtained linearised residual error model may be large. Liu et al. [23] addressed the above limitation by using an adaptive controller to design the feedback loop

working together with the queuing model through on-line tuning of model parameters from measurements. This self-tuning approach lowers the need for system characterization experiments in designing the controller and makes the system more robust. However the range of admitting probability adjustment is limited because of the linearisation model near the operation point. In [35], the authors demonstrated a resource allocation framework that regulates CPU frequency and VM capacity to minimize the power consumption of server clusters. They used decoupled linear models and PI and MPC (Model Predictive Control) controllers. Qin et. al. [27], used an LPV model to design a robust controller on frequency domain. However, the convergence of the identification algorithm is very sensitive to some parameters. Similarly in [28], the authors de-

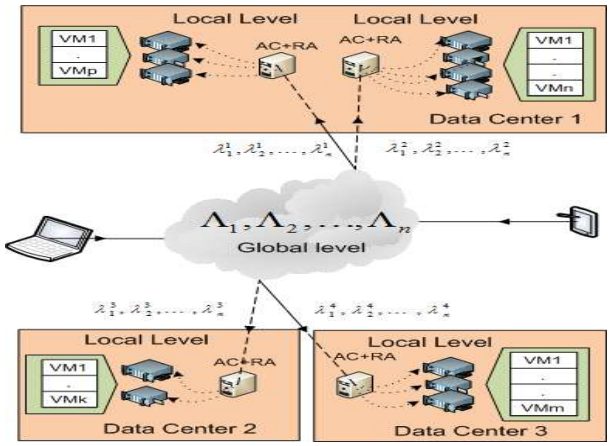



Figure 1: Cloud System Control Framework.

rived an LPV model from the linearisation of a queuing model and they designed a robust resource allocator in frequency domain using DVFS (Dynamic Voltage and Frequency Scaling) technology. In [26], the authors combined the admission control and resource allocation using LPV system identification and Model Predictive Control (MPC), which calculates the admission probability and CPU frequency using DVFS technology. Although their controller offers a trade-off between power consumption and SLOs achievement, it does not guarantee the performance among different operating points. Giani et al. [16] presented an LPV modeling combined with a PI admission control, without addressing resource allocation. It is one of the few studies that provided a stability analysis of their controller. In [34] the authors presented a two level control framework. The global level computes the number of active servers, while the local level optimizes the CPU share of VMs in each server using an MPC scheme. The

model is derived from analytical equations where the operation point was chosen arbitrarily. Finally DynaQos [29] is a model-free self tuning fuzzy controller that on the global level computes the necessary aggregated capacity, while the local level computes the capacity share such that the SLA requirements are met. Apart from [13], all the aforementioned papers do not examine if their solution guarantees the feasibility of steady-state operation, the stability and the consolidated applications' performance simultaneously.

# 3 Autonomous Framework

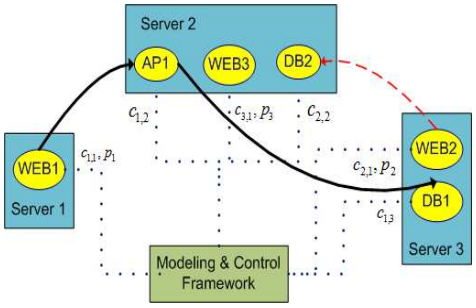



Figure 2: Local Level Architecture.

Figure 1 illustrates the general control architecture for the consolidation of web applications on a distributed cloud computing platform. An Infrastructure as a Service (IaaS) or Software as a Service (SaaS) provider has server clusters in different data centers, which are heterogeneous in terms of resource capacity. The VMs of each application are consolidated in these servers. The global level controller must split the total incoming workload, $\Lambda_i, i = 1, \ldots, n$ of n applications, into partial fractions of load ($\lambda_i^j$) among the VMs of the application running at different locations. The objective of the global level controller can be load balancing or Distributed Rate Limiting (DRL), aiming to equalize the performance in each of the underlying data centres according to a performance indicator.

The local level controllers aim at satisfying the SLOs reference value, the physical constraints and at minimizing the power consumption. Figure 2 depicts the system's architecture for the consolidation of two multi-tier applications and one single-tier application. Typically, web applications consist of many components such as web servers (WEB), application servers (AP) and databases (DB), shown in Figure 2. Black solid and red dashed lines on Figure 2 show the communication between the components of each service in order to complete a

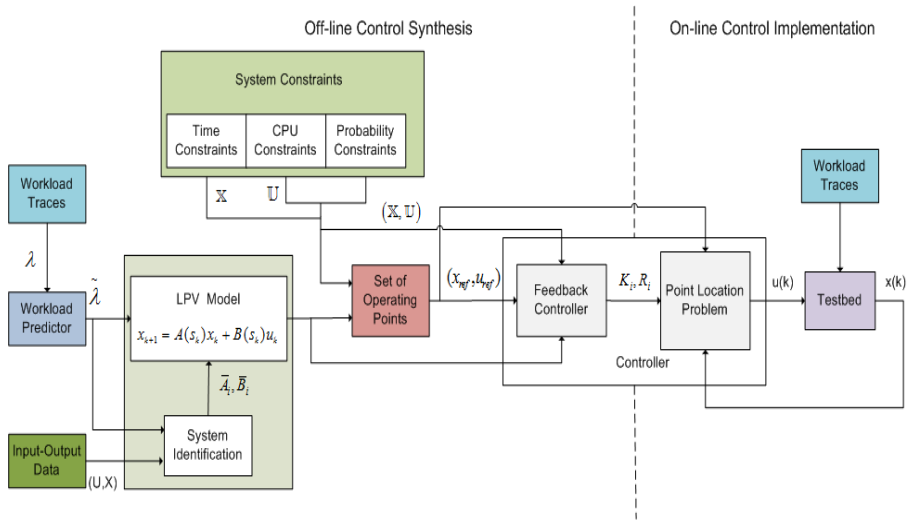



Figure 3: Structural diagram of the proposed modeling and control framework.

task. The components of each application are assigned to servers according to a application placement algorithm. In this article, we assume that the topology of VMs in the server cluster is predefined. The local controller must solve admission control and resource allocation as a joint decision problem. For example, in the setting shown in Figure 2 the goal is to determine at each time instant the admittance probability $(p_1, p_2, p_3)$ for every service, as well as the capacity (CPU) share for their VMs $(c_{1,1}, ..., c_{3,1})$.

Figure 3 illustrates the modeling and control framework presented in this paper. On the left part of this figure all the necessary off-line steps are presented. Initially, the LPV model, $x_{k+1} = A(s_k)x_k + B(s_k)u_k$, is identified $(\overline{A}_i, \overline{B}_i)$ using a set of available input, output $(U, X)$ data and the prediction $(\tilde{\lambda})$ of the incoming request rate $(\lambda)$ from real traces. Next, we quantify the existing physical system's constraints on the control inputs $(\mathbb{U})$ and states $(\mathbb{X})$ of the system. In specific, the CPU share of each VM and the admission probability are the two control inputs of the system (the vector $u(k)$ on Figure 3). The CPU capacity of each VM varies between two limit values, while the sum of the VMs capacities in each server cannot exceed the total capacity of the server. The admission probability ranges between a minimum and a maximum value. In addition, the state vector $(u(k))$ having as elements the response times of each application $(x(k))$ on Figure 3) varies between two extreme values that can be extracted precisely by experiments that lead servers near to their saturation point. The produced LPV model and the system constraints are taken into account by the corresponding element of Figure 3 that determines the set of feasible operating points $(u_{ref}, x_{ref})$. The determination of these points is addressed as an

optimal decision problem. Finally, a stabilizing state-feedback control law $u$ is designed off-line and implemented at each time step $u(k)$. The offline control strategy takes into account the SLOs target values, the system's constraints and the determined set of equilibrium points in order to calculate the gain matrix $K_i$ and the domain of attraction $R_i$, the area of the state-space where it is guaranteed that the system trajectory will be driven on the equilibrium point, for each operating point. Then at every time interval a point location problem determines the current operation point and the final input vector $u(k)$ is applied to the real testbed. The details of the above components on Figure 3 will be analytically described in the forthcoming subsections. Table 3 summarizes the notation used throughout the article.

| **LPV Model** | |
|---|---|
| $n$ | Number of consolidated applications |
| $m$ | Number of active servers |
| $\overline{A}_i$ | LPV state matrices, $i = 0, ..., n$ |
| $\overline{B}_i$ | LPV input matrices, $i = 0, ..., n$ |
| $s_k$ | LPV model parameter |
| $\lambda_{i_k}$ | Incoming request rate of $i^{th}$ application in the $k^{th}$ time interval |
| $\tilde{\lambda}_{i_k}$ | Prediction of incoming request rate of $i^{th}$ application in the $k^{th}$ time interval |
| **States (vector)** | |
| $x_k$ | $90^{th}$ percentile of the response times at the $k^{th}$ time interval |
| **Inputs (vector)** | |
| $c_{i,j_k}$ | CPU capacity of $VM_j^i$ that belongs to $i^{th}$ application on $j^{th}$ server at the $k^{th}$ time interval |
| $p_{i_k}$ | admittance probability of $i^{th}$ application at the $k^{th}$ time interval |
| $u_k$ | concatenated input vector, contains $c_{i,j_k}$ and $p_{i_k}$ in the $k^{th}$ time interval |
| **Constraints** | |
| $\mathbb{X}$ | State constraint set |
| $\mathbb{U}$ | Input constraint set |
| **Reference Values** | |
| $x_{\text{ref}}$ | Equilibrium response time vector |
| $u_{\text{ref}}$ | Equilibrium point input vector |
| $T_{ref}$ | SLA highest acceptable response time vector |

Table 3: The parameters, states and input signals, constraint sets and reference values for the LPV system (2).

## 3.1 Modeling and Identification

Modeling the dynamic behaviour of consolidated web services on a cluster of servers is challenging, since there are no analytical equations that capture the system dynamics. In a previous work [13], a group of LTI models that covers the range of the incoming workload was used. While the approach is more precise than a single LTI model, it is not scalable with respect to the workload range and the number of co-hosted applications. In order to overcome this issue, a Linear Parameter Varying (LPV) model [30] is adopted, which is tuned according to a system parameter. Thus, in this context, the LPV model replaces the group of LTI models in [13], making it scalable with respect to the range of incoming workload. Additionally, it is proven to be more accurate. The advantages of LPV models against the other proposed modeling methods in literature are shown in Table 1.

System identification is a mature field in systems theory [24] . Generally, a mathematical model of a system, is chosen to describe analytically the dynamic operation of a process. The system model is a function of the input vector $u$, which contains all the control signals, and the state vector $x$, which contains the dominant variables of the system that have memory. For the discrete-time case, a difference equation associates the states at the next time instant with the states and control inputs at the current time instant, i.e., $x_{k+1} = f(x_k, u_k)$. The vector-valued mapping $f(\cdot, \cdot)$ can be linear or nonlinear. Having chosen the structure of the model, i.e., a family of functions or a set of parameters, the identification procedure determines an explicit model, using data sets of input signals and states.

We consider $n$ consolidated applications on $m$ servers and each application has one VM on every server. We select as state variables $x_i \in \mathbb{R}$, $i = 1, .., n$, the $90^{th}$ percentile of response time. The state vector is denoted by $x \in \mathbb{R}^n$, i.e.,

$$x := [x_1 \quad \dots \quad x_n]^\top .$$

We consider as inputs the CPU capacities $c_{i,j}$ of each VM, where $i = 1, \dots, n$ denotes the application and $j = 1, \dots, m$ the server, and the admittance probabilities $p_i$, $i = 1, ..., n$, for each application. Consequently, the input vector $u \in \mathbb{R}^{(m+1)n}$ is defined as follows

$$u := [c_{1,1} \quad ... \quad c_{1,m} \quad c_{2,1} \quad ... \quad c_{2,m} \quad ... \quad c_{n,1} \quad ... \quad c_{n,m} \quad p_1 \quad ... \quad p_n]^\top . \tag{1}$$

The LPV model is of the form

$$x_{k+1} = A(s_k)x_k + B(s_k)u_k, \tag{2}$$

where $A(s_k) \in \mathbb{R}^{n\times n}, B(s_k) \in \mathbb{R}^{n\times(m+1)n}$ are the system matrices which depend on the time varying LPV parameter $s_k$ and $k$ is the time variable. In specific, the parameter vector $s \in \mathbb{R}^n$ contains the prediction values $\tilde{\lambda}_i$, $i = 1, .., n$, of the incoming request rate $\lambda_i$ of each application i.e.,

$$s = [\tilde{\lambda}_1 \quad \tilde{\lambda}_2 \quad \dots \quad \tilde{\lambda}_n]^\top.$$

The parameter varying matrices $A(s_k), B(s_k)$ are chosen to have a linear dependence on vector $s_k, k \in \mathbb{N}$, i.e.,

$$A(s_k) = \overline{A}_0 + \sum_{i=1}^{n} s_i \overline{A}_i, \quad B(s_k) = \overline{B}_0 + \sum_{i=1}^{n} s_i \overline{B}_i \tag{3}$$

where $\overline{A}_i \in \mathbb{R}^{n \times n}, i = 0, \ldots, n$ and $\overline{B}_i \in \mathbb{R}^{n \times (m+1)n}, i = 0, \ldots, n$ are linear invariant matrices, whose elements are identified below.

The system (2) is identified using the algorithm of [9]. In specific, the following LPV input-output (LPV-IO) representation method is used to derive the unknown coefficients from past measurements

$$x_k = a_1(s_k)x_{k-1} + \cdots + a_{n_a}(s_k)x_{k-n_a} + b_1(s_k)u_{k-1} + \cdots + b_{n_b}(s_k)u_{k-n_b}, \tag{4}$$

where $n_a, n_b$ represent the number of past values of the state and the input vector respectively. The coefficients $a_i(s_k), i = 1, ..., n_a$ and $b_i(s_k), i = 1, ..., n_b$ are used to construct the matrices $\overline{A}_i, \overline{B}_i$ of (3).

For example, if there are two applications ($n = 2$) consolidated in one server ($m = 1$), then the corresponding LPV-IO represantation (4) is,

$$x_k = a_1(s_k)x_{k-1} + b_1(s_k)u_{k-1},$$

and the LPV model has the following form,

$$\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \end{bmatrix} = \begin{bmatrix} a_{11}(s_k) & a_{12}(s_k) \\ a_{21}(s_k) & a_{22}(s_k) \end{bmatrix} \begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix} + \begin{bmatrix} b_{11}(s_k) & b_{12}(s_k) & b_{13}(s_k) & b_{14}(s_k) \\ b_{21}(s_k) & b_{22}(s_k) & b_{23}(s_k) & b_{24}(s_k) \end{bmatrix} \begin{bmatrix} c_{1,1,k} \\ c_{2,1,k} \\ p_{1,k} \\ p_{2,k} \end{bmatrix}. \tag{5}$$

The timestamps of real traces from [5] were used to produce data sets of the incoming request rate $\lambda_k$ and the control variables of the CPU capacity $c_{i,j}$ of each VM and the admittance probabilities $p_i$ of each application follow uniform distribution respectively. Usually a data set with 2000 samples is sufficient to achieve highly accurate models.

In order to evaluate the produced models, the standard Best Fit Rate (BFR) [26] score was used. The second column of Table 4 illustrates the accuracy of model (5) that corresponds to $n_a = 1$, $n_b = 1$. This model succeeds in 25% higher score than the corresponding LTI model, which is produced by the classical RLS algorithm [36] for LTI modeling. In a typical testbed described analytically in Section 4, the identification experiments show that taking into account past values of states and inputs increases the accuracy of the model, as shown in Table 4. However, for $n_a = 3$, $n_b = 3$ the improvement is negligible.

## 3.2 Request Rate Predictor

In order to predict the incoming request rate of each application, Holt's linear exponential smoothing (LES) filter [37] is used, which can capture the linear

| | $n_a = 1, n_b = 1$ | $n_a = 2, n_b = 2$ | $n_a = 3, n_b = 3$ |
|---|---|---|---|
| LPV-RLS | $76.72\% - 82.61\%$ | $79.37\% - 86.79\%$ | $80.29\% - 87.17\%$ |
| RLS | $43.45\% - 57.04\%$ | $45.08\% - 59.23\%$ | $45.54\% - 59.64\%$ |

Table 4: Comparison of the BFR score of the LPV-RLS and RLS identification algorithms.

trend in the time series. For example, during time step k, the estimated value $\tilde{\lambda}_k$ of incoming request rate $\lambda_k$ for a one-step prediction horizon is obtained as follows,

$$\begin{aligned} \tilde{\lambda}_k &= \hat{\lambda}_k + b_k, \\ \hat{\lambda}_k &= \alpha\lambda_k + (1-\alpha)(\hat{\lambda}_{k-1} + b_{k-1}), \\ b_k &= \beta(\hat{\lambda}_k - \hat{\lambda}_{k-1}) + (1-\beta)b_{k-1}, \end{aligned} \tag{6}$$

where $\alpha, \beta$ are smoothing constants, $\hat{\lambda}_k$ denotes the smoothed value for time step $k$ and $b_k$ represents the linear trend in the measurement series. For initialization, we use a random value for $\hat{\lambda}_0$ inside the range of incoming request rate and $b_0 = 0.5$.

### 3.3 State and Input Constraints

The computational resources of a server cluster, which hosts a group of applications, are inherently limited. Moreover, there are network constraints that bound the range of response time and incoming request rate. The $90^{th}$ percentile of response time $x_i$ of the application $i$ in any time interval varies from a small positive value $\varepsilon$ (e.g., $0.01ms$) to the value when the system is saturated or the requests have expired due to network constraints. Thus, states are bounded in the constraint set $\mathbb{X}$, where

$$\mathbb{X} = \{x \in \mathbb{R}^n : \ \varepsilon \leq x_i \leq x_{\max}, \ i = 1, \ldots, n\}. \tag{7}$$

On the other hand, input constraints consider the restrictions on VM capacity and admittance probability. For each VM of application $i$ on server $j$, the CPU capacity $c_{i,j}$ varies from a minimum value $c_{min}$ to a maximum $c_{max}$. Additionally, the sum of VMs capacity should not exceed the total capacity of the server $CT_j$. The admittance probability ranges from a minimum value $p_{min}$ to $p_{max} = 1$. All the above constraints form the input constraint set $\mathbb{U}$,

$$\begin{aligned} \mathbb{U} = \{u \in \mathbb{R}^{(m+1)n} : \quad & c_{\min} \leq c_{i,j} \leq c_{\max}, i = 1, \ldots, n, \ j = 1, \ldots, m, \\ & \textstyle\sum_{i=1}^{n} c_{i,j} \leq CT_j, j = 1, \ldots, m, \\ & p_{\min} \leq p_i \leq p_{\max}, i = 1, \ldots, n\}. \end{aligned} \tag{8}$$

We remind the reader that the input vector is defined by (1).

### 3.4 Determination of the Operating Point

The modeling framework in section 3.1 allows for identifying a number of desired feasible operating points. From a set of candidate equilibrium points we choose the ones satisfy the SLA requirements and the state and input constraints (7) and (8) respectively. The determination of this point is a decision problem that includes several competing objectives, such as small response times and low power consumption. Thus, depending on the priority of the goals set and selecting accordingly the corresponding cost function, an optimization problem can be formulated, whose solution is the desired equilibrium point. In particular, we desire to satisfy the response time reference value of each service and maximize the admittance probability without violating the system constraints.

To this end, given a desired response time vector $x_{\text{ref}}$, an input vector $u_{\text{ref}}$ is computed by the solution of the following linear optimization problem,

$$\min_{c_{i,j_{\text{ref}}}, p_{i_{\text{ref}}}, d_i} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} c_{i,j_{\text{ref}}} + \sum_{i=1}^{n} w_{d_i} d_i \right\} \tag{9a}$$

subject to

$$x_{\text{ref}} = \hat{A}_l x_{\text{ref}} + \hat{B}_l u_{\text{ref}},\ l = 1, \ldots, 2^n, \tag{9b}$$

$$\varepsilon \leq x_{\text{ref}_i} \leq x_{max},\ i = 1, \ldots, n, \tag{9c}$$

$$c_{\min} \leq c_{i,j_{\text{ref}}} \leq c_{\max}, i = 1, \ldots, n,\ j = 1, \ldots, m, \tag{9d}$$

$$\sum_{i=1}^{n} c_{i,j_{\text{ref}}} \leq CT_j, j = 1, \ldots, m, \tag{9e}$$

$$p_{\max} - p_{i_{\text{ref}}} \leq d_i,\ i = 1, \ldots, n, \tag{9f}$$

$$d_i \geq 0,\ i = 1, \ldots, n. \tag{9g}$$

The matrices $\hat{A}_l$, $\hat{B}_l$, $l = 1, ..., 2^n$ represent the values of the matrices $\bar{A}(s), \bar{B}(s)$, for all extreme values of the LPV parameter $s$. A detailed exposition of this transformation is in the Appendix. Consequently, Eq.(9b) ensures that $(x_{\text{ref}}, u_{\text{ref}})$ is the equilibrium point for all possible parameter variations. Eq.(9c)-(9e) ensure state and input constraint satisfaction. The positive numbers $w_{ij}, i = 1, \ldots, n, j = 1, \ldots, m$, and $w_{d_i, i=1,\ldots,n,}$ are weights of the optimization cost function and their values correspond to the trade-off among the competitive objectives. The auxiliary variables $d_i$ in the cost function and the last two equations are used to maximize the admittance probability. In accordance with the notation used in (1), the reference input vector $u_{\text{ref}}$ is

$$u_{ref} = \begin{bmatrix} c_{1,1_{ref}} & \cdots & c_{1,m_{ref}} & \cdots & c_{n,1_{ref}} & ... & c_{n,m_{ref}} & p_{1_{ref}} & \cdots & p_{n_{ref}} \end{bmatrix}^\top .$$

### 3.5 Controller Design

In this subsection, we construct state-feedback control laws. At each time instant, a state-feedback controller takes as input the current measurements of

the states of the system (2). After processing this information, the values of the control inputs (1), which in this case are the CPU capacities of each VM and the percentage of the admitted requests, are calculated and fed to the system. In specific, our goal is to compute a control law that drives the states of the system, i.e., the response times of each application $i$, $i = 1, ..., n$, to a desired pre-specified equilibrium point, while the physical constraints (7), (8) are satisfied at all times. We tackle the problem of constraint satisfaction and of convergence to the desired equilibrium point simultanesouly, by utilizing quadratic Lyapunov functions for the system (2). The benefit of the proposed method is that it leads to the computation of a stabilizing control law, while at the same time it characterizes a set of initial conditions (i.e., initial response times) that can be transferred to the equilibrium point (desired response times) without violating the state and input constraints. These sets are called regions of attraction (RoA), associated to the chosen control law and the equilibrium point [19].

The problem to be investigated can be formally formulated as follows: Given the system (2), a set of desired operating points $x_{\mathrm{ref},i}$, $i = 1, .., M$ and the state and input constraints (7), (8), compute state-feedback control strategies $u := g_i(x)$, $i = 1, \ldots, M$ and a domain of attraction (DoA) $R_i \subset \mathbb{R}^n$ for each operating point, such that the closed-loop system is locally asymptotically stable with respect to $x_{\mathrm{ref},i}$.

Since the incoming workload is unpredictable and sudden changes occur, it is important to have different operating points in order to have more flexibility. For example, if the incoming request rate dramatically increases, then changing to another operating point with higher response time but also higher throughput may be a profit-full strategy. Consequently, a second problem to be investigated is to compute a control strategy such that the closed-loop system is guaranteed to change efficiently between different operating points.

We consider the system (2), retrieved by the identification procedure described in Subsection 3.1. The LPV parameter $s_k$ is calculated at each time instant by the request rate predictor described in Subsection (3.2), the state and input constraints have been formulated in Subsection 3.3, while the set of feasible equilibrium points has been computed in (3.4). The proposed controller synthesis is described by the following off-line steps,

Step 1 For each chosen equilibrium point $x_{\mathrm{ref},i}$, we apply a coordinate transformation in order to obtain the translated system with its equilibrium point at the origin.

$$\begin{aligned} z_k &= x_k - x_{\mathrm{ref},i}, \\ v_k &= u_k - u_{\mathrm{ref},i}, \end{aligned} \tag{10}$$

$$z_{k+1} = A(s_k)z_k + B(s_k)v_k. \tag{11}$$

The state and input constraints are transformed accordingly to $z \in \overline{\mathbb{X}}_i, v \in \overline{\mathbb{U}}_i$.

Step 2 For each translated system (11), we consider a quadratic candidate Lyapunov function [10, chapter 2], [18, chapter 4] $V_i(\cdot), i = 1, \ldots, N$, i.e.,

$$V_i(z) = z^\top P_i z. \tag{12}$$

Moreover, we consider a linear state-feedback control law $g_i(\cdot)$ for each system (24) $i,\ i = 1, \ldots, N$ (11), of the form $g_i(z) = K_i z,\ i = 1, \ldots, N$. Thus, the closed-loop system that relates to the equilibrium point $i$ is

$$\overline{M}_i : z_{k+1} = (A(s_k) + B(s_k)K_i)z_k, \quad i = 1, .., N, \tag{13}$$

For each closed-loop system (24), we compute simultaneously the matrix $P_i \succ 0$ that satisfies the one-step decrease along the trajectories of the system (24) requirement of the Lyapunov function, and as the feedback gain $K_i$. Computing the pairs $(P_i, K_i)$, $i = 1, ..., N$ is equivalent to solving a convex optimization problem [32].

Step 3 Transforming back to the initial state space, the domain of attraction for each closed-loop system (2) relating to the equilibrium point $x_{\text{ref},i}$ is

$$R_i = \{x \in \mathbb{R}^n : (x - x_{\text{ref},i})^T P_i (x - x_{\text{ref},i}) \leq 1\}. \tag{14}$$

Thus, given the desired equilibrium point, $x_{\text{ref},i}$, the control law $u_k$ for the system (2) is a function of the state vector $x_k$, measured at the time instant $k$. If $x_k$ is in the RoA $R_i$, the control action is of the form.

$$u_k = K_i(x_k - x_{\text{ref},i}) + u_{\text{ref},i}. \tag{15}$$

The procedure is explained in more detail in the Appendix, while it is presented in an algorithmic manner in Algorithm 1.

**Algorithm 1** - Offline Controller Synthesis

1: **for all** $x_{\text{ref},i},\ i = 1, \ldots, N$ **do**
2: compute $(x_{\text{ref},i}, u_{\text{ref},i}),\ i = 1, \ldots, N$ by solving (9a)
3: compute $\overline{\mathbb{X}}_i, \overline{\mathbb{U}}_i$
4: compute $R_i, K_i, P_i,\ \forall (x_{\text{ref},i}, u_{\text{ref},i}),\ i = 1, \ldots, N$
5: **end for**

Figure 4 shows a graphical representation of the domains of attraction in the state space, for each operating point for the system (5). For example the DoA of $x_{\text{ref}} = [1 \quad 1]^\top$ is highlighted. If any point inside the blue ellipsoid is the initial state then an admissible control law exists that transfers the system trajectory to the equilibrium point $x_{\text{ref}} = [1 \quad 1]^\top$.

Due to the unpredictable nature of incoming workload, it is essential to design a control mechanism that ensures the performance of the system whenever an important change occurs. From the set of desired operating points $x_{\text{ref},i},\ i = 1, .., N$, there is one operating point that is closest to the SLA requirements. According to the SLA, for each consolidated application there is a highest acceptable response time $T^\star_{ref,i},\ i = 1, \ldots, n$. Thus, an accepted operating area is implicitly shaped on the state space where the state variables should remain for all time intervals. From the set of feasible operating points, determined in subsection (3.4), we select the operation point, which its corresponding

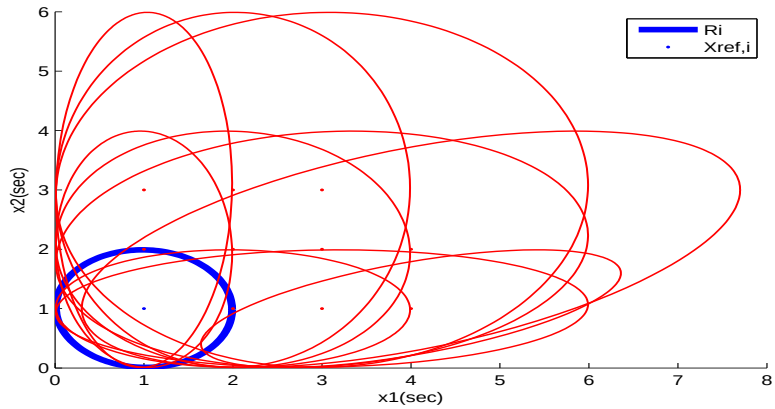



Figure 4: DoAs $R_i$ of each operating point $x_{\text{ref},i},\ i = 1, .., N$.

DoA covers the largest part of the accepted operating area. This operating point is denoted as $(x_{\text{ref},i^\star}, u_{\text{ref},i^\star})$, its corresponding DoA $R_{i^\star}$ and feedback gain matrix $K_{i^\star}$. Every time step $k$ the response time vector $x_k$ is measured, if this value is not in $R_{i^\star}$, then we have to choose the closest operating point that contains it according to the following Euclidean distance metric,

$$\min_{x_{\text{ref},i}} \ \|x_{\text{ref},i} - x_{\text{ref},i^\star}\| \tag{16a}$$

subject to

$$x_k \in R_i,\ i = 1, .., N \tag{16b}$$

which is a standard point location problem.

Because of the overlap of the DoA of the operating points in Figure 4 , the system trajectory will move from one DoA to another, closest to $R_{i^\star}$, until it converges to the desired $R_{i^\star}$. The online implementation of the control strategy is described in an algorithmic fashion in Algorithm 2.

**Algorithm 2** - Online Controller Implementation

1: **if** $x_k \in R_{i^\star}$ **then**
2: $\quad u_k = K_{i^\star}(x_k - x_{\text{ref},i^\star}) + u_{\text{ref},i^\star}$
3: **else**
4: $\quad$ Find $R_i$ by solving (16a)
5: $\quad u_k = K_i(x_k - x_{\text{ref},i}) + u_{\text{ref},i}$
6: **end if**
7: $x_k \leftarrow x_{k+1} = A(s_k)x_k + B(s_k)u_k$
8: Go to line 1

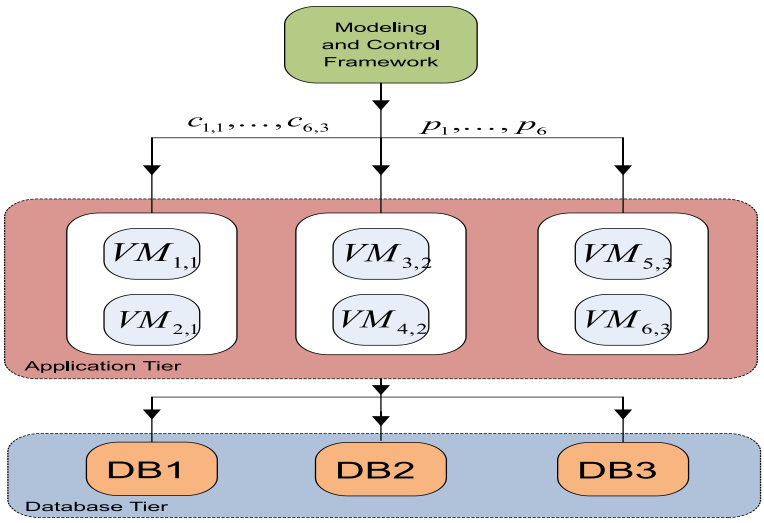



Figure 5: Tested Diagram - $VM_j^i$ of $i^{th}$ application on $j^{th}$ server.

# 4 Evaluation

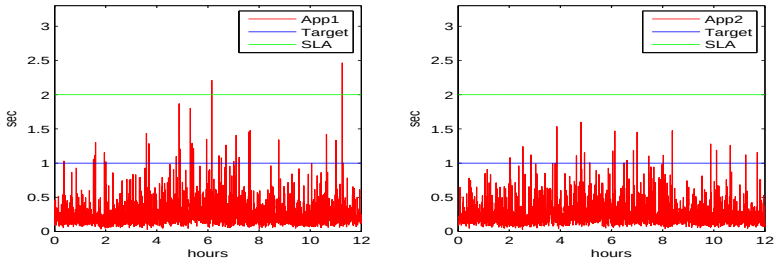
We have built a real testbed in order to evaluate the performance of our system modeling and control framework. A server with 8-core Intel(R) Xeon(R) CPU E5405, 8GB memory and Debian squeeze, two servers with 4-core Intel(R) Xeon(R) CPU E31220, 8GB memory and Ubuntu 11.10 are used. Also XEN HV:xen-hypervisor-4.0-i386 (4.0.1-4) is used to set up the VMs. Six services were deployed, which consist of an application and a database tier. Each tier is hosted on a VM and the topology of VMs is shown in figure 5. The CPU capacity of database tier is fixed and only the capacity of the application tier VMs is assumed as bottleneck and changes dynamically, which is a typical scenario in cloud computing. We used real, highly variant and non-stationary traces from [5] to create the incoming workload of every application. As it is mentioned earlier, the timestamps of the incoming requests are used to build the incoming workload. Also the data that are used for the evaluation of the controller are different from the traces that were used for the identification of the LPV model. These traces are still used by recent studies [38], [15] since they are more realistic than benchmarks such as TPC-W [4] and RUBis [2] that have stationary request generators. The workload predictor and the controller are implemented using CVXPY [14], which a Python toolbox for convex optimization problems. We demonstrate an experiment where the upper desired reference $90^{th}$ percentile value of response time, defined by SLA, is $T_{\text{ref}} = 2sec$.

According to the concept described in Section 3.5, the following target value of the equilibrium point arises $X_{\text{ref}} = 1sec$. We assume the prediction of the incoming request rate $\tilde{\lambda}_i(k)$ as the LPV model parameter $s_k$. The control values are updated every 30 seconds. As is shown on Figure 5, every server hosts the VMs of two applications. This implies that there are three decoupled systems with $n = 2$ and $(m = 1)$, which are described by system (5).

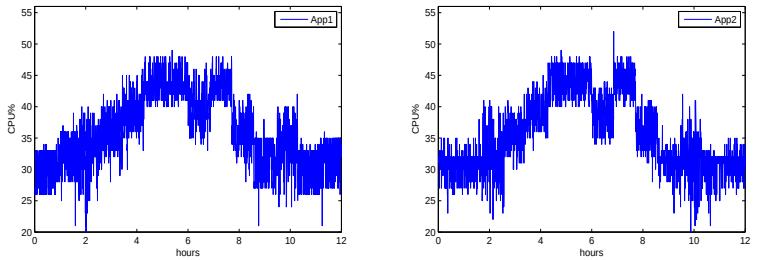
Figures 6a-6d illustrate the performance of the proposed framework for a 12-hour experiment (1440 samples). Indicatively we present the performance of services App1 and App2 whose VMs share the first server in the application tier of Figure 5. Figure 6a shows the $90^{th}$ percentile value of response time for both applications. The percentage of time intervals that the response time is less than the target value $X_{\text{ref}}$ (blue line), is 98.78%. Also there are very few time intervals whose response time violates the reference value $T_{\text{ref}}$ (green line). Figures 6b and 6d indicate that it is essential to tackle resource allocation and admission control as a common decision problem. When the incoming workload is low the VM capacity is also low and it increases as the incoming load increases. On the other hand, admission probability is close to one when the workload is light whereas it decreases if more requests arrive. Figure 6c illustrates that although the incoming request rate (blue dotted line with circle) is dynamic and unpredictable, the existed predictor efficiently estimates the future values of workload (red dashed line).

Figure 7 illustrates a short example of the control strategy presented in this article. The red dots depict different operating points $x_{ref,i}, u_{ref,i}$, the red ellipsoids the corresponding DoAs $R_i$ and the blue dot and line correspond to the target $x_{\text{ref},i^\star}, u_{\text{ref},i^\star}, Ri^\star$. The black solid line corresponds to the system trajectory which begins from $x_0 = [4.3 \quad 3.9]^\top$ far away from the desired operating point, $x_{\text{ref},i^\star} = [1 \quad 1]^\top$. Applying Algorithm 2, the appropriate control law is computed and by solving the point location problem (16a) the response time vector enters the acceptable region (blue line) on state-space, $x(k) \leq T_{\text{ref}}$, after a few steps.

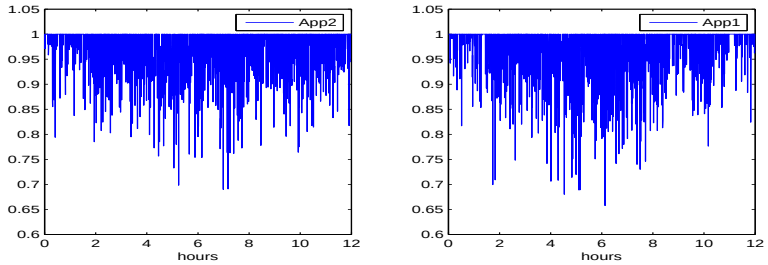
We compare the proposed solution with study [26], which also uses LPV modeling and MPC control without guaranteeing the feasibility of the operating point. We select this particular study because is closest to our control perspective and it uses a popular and well-established control method. This study also provides a trade-off between response time restrictions and energy consumption adjusting the parameter $\alpha$ to the cost function $J = \alpha||x(k)|| + (1-\alpha)\sum_{k=K}^{K+H-1} ||u(k)||$. We used the same incoming workload in order to compare the two approaches and Figures 8a-8c show their performance for a 5-hour experiment (600 samples). It can be seen that our method outperforms the MPC controller. In particular, the MPC scheme results in a large percentage of response violations, especially for App2, and it consumes more capacity resources and rejects a significant portion of the incoming requests. Table 5 summarizes the performance of the two control solutions. The MPC controller does not perform very well because of the lack of a feasible equilibrium point that will assure system stability. Also it is very sensitive to the values of the parameter



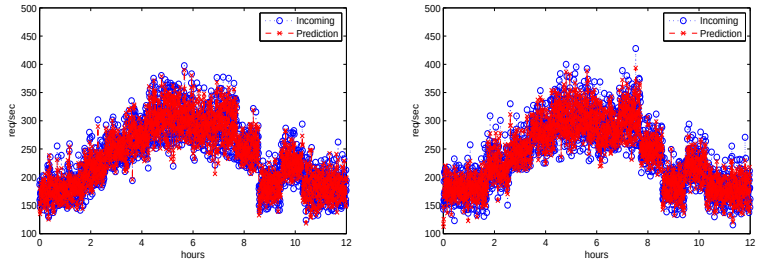

(a) Response Time.




(b) VMs Capacity.




(c) Incoming Request Rate.




(d) Admittance Probability.

Figure 6: Overall Performance of Consolidated Applications.

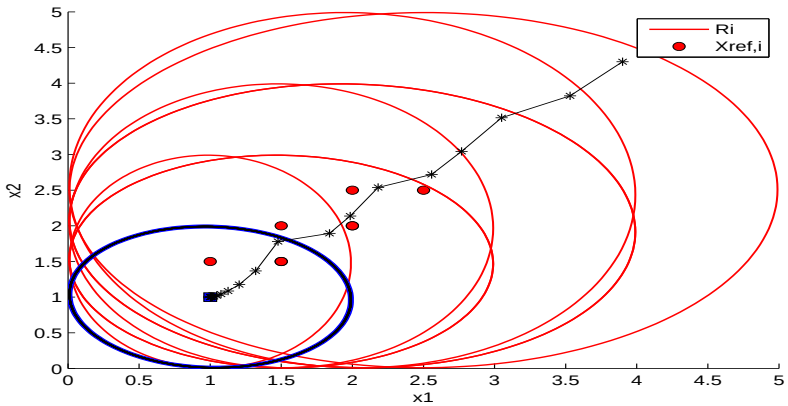



Figure 7: DoA of operating points (red lines and circles), DoA of the desired operating point (blue line and square) and System Trajectory (black solid line).

$\alpha$ of the cost function. This results in the input oscillations in Figure 8b and Figure 8c.

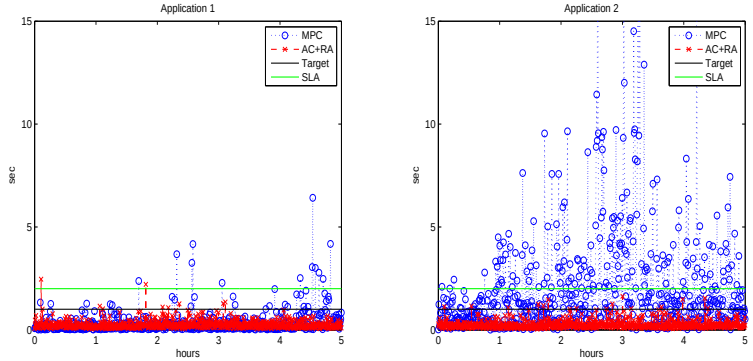
| | (%) Accepted RT | Capacity | Probability |
|---|---|---|---|
| AC+RA | 98.55% | 36.08% | 96.69% |
| MPC | 69.92% | 45% | 60.32% |

Table 5: Performance of AC+RA vs MPC.

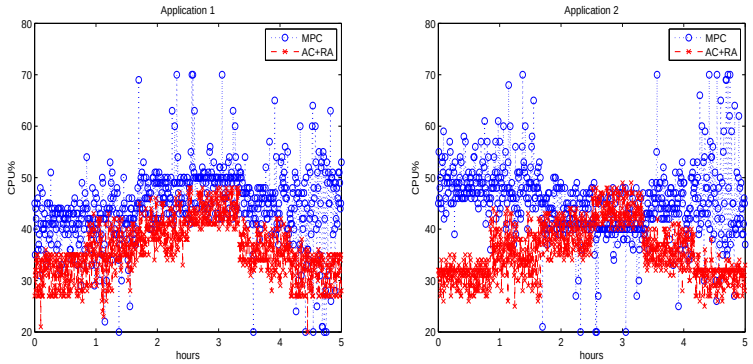
# 5 Conclusion and Future Work

A unified local level modeling and control framework for consolidated web services in a server cluster was presented, which can be a vital element of a holistic distributed control platform. Admission control and resource allocation were addressed as a common decision problem. Stability and constraint satisfaction was guaranteed. A real testbed was built and from a range of examples, in different operating conditions, we can conclude that both the identification scheme and controller provide a high level of QoS. A novel component of this approach is the determination of a set of feasible operating (equilibrium) points, which allows the selection of the appropriate equilibrium point, depending only on what our objectives are, such as maximizing throughput, minimizing consumption or maximizing profit. Evaluation shows that our approach has high performance compared to well-known solutions, such as queuing models and measurement approach of equilibrium points.

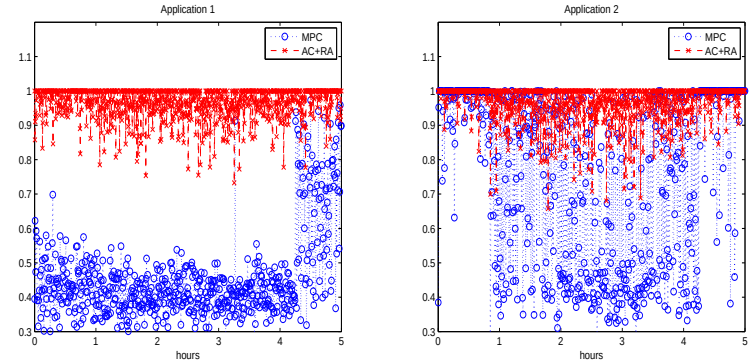
For future work we intend to extend our work using this study as an element




(a) Response Time.




(b) VMs Capacity.




(c) Admittance Probability.

Figure 8: Comparison with MPC Controller.

of a general control framework, which also includes a global level controller that distributes the incoming workload according to an optimization problem, either minimizing the number of active severs or equalizing the amount of load on all underlying servers. Additionally in this global level, we are able to construct node controllers that co-operate with the others and determine the number of VMs per server with availability guarantees.

# References


[1] Matlab robust control toolbox. `http://www.mathworks.com/products/robust/`.

[2] RUBiS: Rice university bidding system. `http://rubis.ow2.org/`.

[3] SDPT 3 - a matlab software package for semidefinite-quadratic-linear programming. `http://www.math.cmu.edu/~reha/sdpt3.html`.

[4] TPC-W: a transactional web e-commerce benchmark. `www.tpc.org/tpcw/`.

[5] World football cup 1998 network traces. `http://ita.ee.lbl.gov/html/contrib/WorldCup.html`.

[6] J. Almeida, V. Almeida, D. Ardagna, I. Cuhna, and C. Francalanci. Joint admission control and resource allocation in virtualized servers. *Elsevier Journal of Parallel and Distributed Computing*, 70(4):344–362, 2010.

[7] D. Ardagna, C. Ghezzi, B. Panicucci, and M. Trubian. Service provisioning on the cloud: Distributed algorithms for joint capacity allocation and admission control. In *Proc. of the 3rd European Conference ServiceWave*, pages 1 – 12, 2010.

[8] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang. Energy-aware autonomic reource allocation in multitier virtualized enviroments. *IEEE Transactions on Services Computing*, 5(1):2–19, 2012.

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

[10] F. Blanchini and S. Miani. *Set-theoretic methods in control*. Springer, 2008.

[11] S. Boyd. *Linear matrix inequalities in system and control theory*, volume 15. Siam, 1994.

[12] L. Cherkasova and P. Phaal. Session-based admission control: A mechanism for peak load management of commercial web sites. *IEEE Transactions on Computers*, 51(6):669–685, 2002.

[13] D. Dechouniotis, N. Leontiou, N. Athanasopoulos, G. Bitsoris, and S. Denazis. ACRA: A unified admission control and resource allocation framework for virtualized environments. In *Proc. of the 8th IEEE International Conference on Network and Service Management (CNSM)*, pages 145–149, 2012.

[14] S. Diamond, E. Chu, and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization, version 0.2. `http://cvxpy.org/`.

[15] A. Gandhi, Y. Chen, D. Gmach, M. Arlitt, and M. Marwah. Hybrid resource provisioning for minimizing data center sla violations and power consumption. *Elsevier Sustainable Computing: Informatics and Systems*, 2(2):91–104, 2012.

[16] P. Giani, M. Tanelli, and M. Lovera. Controller design and close-loop stability analysis for admission control in web service systems. In *Proc. of the 18th IFAC World Congress*, pages 6709 – 6714, 2011.

[17] A. Kamra, V. Misra, and E. Nahum. Yaksha: A self-tuning controller for managing the performance of 3-tiered web sites. In *Proc. of the 12th IEEE International Workshop on Quality of Service (IWQOS)*, pages 47–56, 2004.

[18] H. K. Khalil. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, 2002.

[19] H. K. Khalil and JW. Grizzle. *Nonlinear systems*, volume 3. Prentice Hall Upper Saddle River, 2002.

[20] D. Kusic and N. Kandasamy. Risk-aware limited lookahead control for dynamic resource provisioning in enterprise computing systems. In *Proc. of the IEEE International Conference on Autonomic Computing (ICAC)*, pages 74–83, 2006.

[21] D. Kusic, J.O. Kephart, J.E. Hanson, N. Kandasamy, and G. Jiang. Power and performance management of virtualized computing enviroments via lookahead control. *Springer Journal on Cluster Computing*, 12(1):1–15, 2009.

[22] N. Leontiou, D. Dechouniotis, and S. Denazis. Adaptive admission control of distributed cloud services. In *Proc. of the IEEE International Conference Network and Service Management (CNSM)*, pages 318 – 321, 2010.

[23] X. Liu, J. Heo, L. Sha, and X. Zhu. Queueing-model-based adaptive control of multi-tiered web applications. *IEEE Transactions on Network and Service Management*, 5(3):157–167, 2008.

[24] L. Ljung. *System identification*. Springer, 1998.

[25] B. Plugmers. *Robust Model Based Predictive Control - An Invariant Set Approach*. PhD thesis, Katholieke Universiteit Lauven, 2006.

[26] C. Poussot-Vassal, M. Tanelli, and M. Lovera. A control-theoretic approach for the combined management of the quality-of-service and energy is service centers. In *Run-time Models for Self-managing Systems and Applications*, pages 73–96. Springer, 2010.

[27] W. Qin and Q. Wang. Feedback performance control for computer systems: an lpv approach. In *Proceedings of the American Control Conference (ACC)*, pages 4760–4765, 2005.

[28] 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.

[29] J. Rao, Y. Wei, J. Gong, and C. Xu. Dynaqos: Model-free self-tuning fuzzy control of virtualized resources for qos provisioning. In *Proc. of the 19th IEEE International Workshop on Quality of Service (IWQOS)*, pages 1–9, 2011.

[30] R. Tóth. *Modeling and identification of linear parameter-varying systems*, volume 214. Springer, 2010.

[31] R. Urgaonkar, U. Kozat, K. Igarashi, and M.J. Neely. Dynamic resource allocation and power management in virtualized data centers. In *Proc. of the IEEE Network Operations and Management Symposium (NOMS)*, pages 479 – 486, 2010.

[32] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.

[33] R. Wang and N. Kandasamy. On the design of decentralized control architectures for workload consolidation in large-scale server clusters. In *Proceedings of the 9th IEEE International Conference on Autonomic Computing (ICAC)*, pages 115–124, 2012.

[34] R. Wang, D. Kusic, and N. Kandasamy. A distributed control framework for performance management of virtualized computing environments. In *Proc. of the 7th IEEE/ACM International Conference on Autonomic Computing (ICAC)*, pages 89–98, 2010.

[35] X. Wang and Y. Wang. Coordinating power and performance management for virtualized server clusters. *IEEE Transactions on Parallel and Distributed Systems*, 22(2):245 – 259, 2011.

[36] P.E. E. Wellstead, E., and M. B. Zarrop. *Self-Tuning Systems: Control and Signal Processing*. John Wiley & Sons, Inc., 1991.

[37] S.C. Wheelwright and R. J. Hyndman. *Forecasting: methods and applications.* John Wiley & Sons Inc, 1998.

[38] T. Xie and Y. Sun. Understanding the relationship between energy conservation and reliability in parallel disk arrays. *Elsevier Journal of Parallel and Distributed Computing*, 71(2):198–210, 2011.

# Appendix

In order to obtain a representation of the model that allows the development of the subsequent results, the matrices $A(s_k), B(s_k)$ of the system (2) can be equivalently expressed as the convex combination of extreme subsystems defined by the matrix pairs $(\hat{A}_i, \hat{B}_i), i = 1, ..., 2^n$, which correspond to the extreme values of $s_k$, i.e.,

$$A(s) = \sum_{i=1}^{2^n} m_i(s)\hat{A}_i,\ i = 1, \ldots, 2^n, \tag{17}$$

$$B(s) = \sum_{i=1}^{2^n} m_i(s)\hat{B}_i,\ i = 1, \ldots, 2^n, \tag{18}$$

$$\tag{19}$$

where $m_i(s) \geq 0\ i = 1, \ldots, 2^n$ and $\sum_{i=1}^{2^n} m_i(s) = 1$. This standard equivalent representation corresponds to linear discrete-time systems with polytopic uncertainties [10]. Since $n$ is the number of consolidated applications and their incoming request rate varies between its minimum and maximum value, the total number of the matrix pairs $(\hat{A}_i, \hat{B}_i)$ is $2^n$.

For each equilibrium point $(x_{\text{ref},i}, u_{\text{ref},i})$, $i = 1, \ldots, N$, computed in subsection 3.4, we define the translated system

$$z_{k+1} = A(s_k)z_k + B(s_k)v_k,$$

where $z_k = x_k - x_{\text{ref},i}$, $v_k = u_k - u_{\text{ref},i}$. Also for the $i$th system (11), the state and input constraints are transformed accordingly to $z \in \overline{\mathbb{X}}_i, v \in \overline{\mathbb{U}}_i$, where

$$\overline{\mathbb{X}}_i = \{z \in \mathbb{R}^n :\ \varepsilon - x_{\text{ref},i} \leq z \leq x_{max} - x_{\text{ref},i}\}, \tag{20}$$

$$\begin{aligned} \overline{\mathbb{U}}_i = \{v \in \mathbb{R}^{(m+1)n} : c_{min} - c_{i,j_{ref}} \leq tc_{i,j} \leq c_{max} - c_{i,j_{ref}}, \\ i = 1, \cdots, n, j = 1, \cdots, m, \\ \textstyle\sum_{i=1}^{n} tc_{i,j} \leq CT_j - \sum_{i=1}^{n} c_{i,j_{ref}},\ i = 1, \cdots, n, j = 1, \cdots, m, \\ p_{min} - p_{i_{ref}} \leq tp_i \leq p_{max} - p_{i_{ref}}\}. \end{aligned} \tag{21}$$

where $tc_{i,j} = c_{i,j} - c_{i,j_{ref}}$ are the transformed VMs capacity inputs and $tp_i = p_i - p_{i_{ref}}$ are the transformed admission probability inputs. The above constraints can be rewritten in the following form of linear inequalities,

$$z \in \overline{\mathbb{X}}_i = \{z \in \mathbb{R}^n :\ C_i(j)z \leq 1,\ i = 1, \ldots, N,\ j = 1, \ldots, p\} \tag{22}$$

$$v \in \overline{\mathbb{U}}_i = \{v \in \mathbb{R}^{(m+1)n} :\ D_i(j)v \leq 1,\ i = 1, \ldots, N,\ j = 1, .., q\} \tag{23}$$

where $C_i(j) \in \mathbb{R}^{1\times n}$ and $D_i(j) \in \mathbb{R}^{1\times(m+1)n}$ are the rows of matrices $C_i, D_i$ respectively which contain all the constraint inequalities for each system (11). We consider a linear state-feedback control law $g_i(\cdot)$ for each system $i,\ i = 1,\dots,N$ (11), of the form $g_i(z) = K_i z,\ i = 1,\dots,N$,

$$\overline{M}_i : z_{k+1} = (A(s_k) + B(s_k)K_j)z_k, \quad i = 1,..,N, \tag{24}$$

For choose quadratic candidate Lyapunov functions $V_i(\cdot), i = 1,\dots,N$ of the form

$$V_i(z) = z^\top P_i z. \tag{25}$$

For each closed-loop system (24), we have to determine a positive definite matrix $P_i \in \mathbb{R}^{n\times n}$ that satisfies the one-step decrease along the trajectories of the system (24), i.e., MHTSO BALE TA Ai Bi ME KAPELA PANTOU

$$(\hat{A}_i + \hat{B}_i K_i)^\top P_i(\hat{A}_i + \hat{B}_i K_i) - P_i \prec 0. \tag{26}$$

To eliminate the non-linearities in (26), we set $Q_i = P_i^{-1}$ and $Y_i = K_i Q_i$ and by pre and post multiplying eq. (26) with $Q_i$ we get

$$(\hat{A}_i Q + \hat{B}_i Y_i)^\top Q^{-1}(\hat{A}_i Q + \hat{B}_i Y_i) - P_i \prec 0. \tag{27}$$

Applying the Schur complement [11] we get the following equivalent form

$$\begin{bmatrix} Q_i & \hat{A}_i Q_i + \hat{B}_i Y_i \\ (\hat{B}_i Q_i + \hat{B}_i Y_i)^\top & Q_i \end{bmatrix} \succ 0, \tag{28}$$

which is a linear matrix inequality with respect to $Q_i, Y_i$. Then we can compute the gain matrix $K_i$ and $Q_i, P_i$ for each system (24) $\overline{M}_i, i = 1,...,N$, by solving $N$ convex optimization problems

$$\min_{Q_i, Y_i, \epsilon} \ \mathrm{trace}(Q_i) \tag{29a}$$

subject to

$$\begin{bmatrix} \epsilon Q_i & \hat{A}_i Q_i + \hat{B}_i Y_i \\ (\hat{A}_i Q_i + \hat{B}_i Y_i)^\top & Q_i \end{bmatrix} \succ 0, \tag{29b}$$

$$\begin{bmatrix} 1 & C_i(j)Q_i \\ (C_i(j)Q_i)^\top & Q_i \end{bmatrix} \succeq 0,\ j = 1,\dots,p, \tag{29c}$$

$$\begin{bmatrix} 1 & D_i(j)Y_i \\ (D_i(j)Y_i)^\top & Q_i \end{bmatrix} \succeq 0,\ j = 1,\dots,q, \tag{29d}$$

$$Q_i \succ 0, \tag{29e}$$

$$0 \le \epsilon < 1. \tag{29f}$$

In the above problem, inequalities (29b) and (29e) ensure exponential stability for the closed–loop system. Inequalities (29c) and (29d) guarantee state and input constraint satisfaction. Parameter $\epsilon$ is a measure of the speed of convergence of the closed loop system to the equilibrium point. The problem is convex and

can be solved using off-the-self software, e.g the Matlab Robust Control Toolbox [1] or the SDPT 3.0 software toolbox [3]. The interested reader is referred to, e.g., [11], [25, Appendix A], for further details.

The Lyapunov matrix and the gain matrix for each operating point are finally computed as $P_i = Q_i^{-1}$ and $K_i = Y_i Q^{-1}$ respectively. Together with the controller, the set

$$S_{iz} = \{z \in \mathbb{R}^n : z^\top P_i z \leq 1\}$$

is an admissible DoA for the closed loop system , i.e. it is the set which contains all initial conditions that can be transferred asymptotically to each equilibrium point without violating the constraints. Transforming back to the initial state space, the domain of attraction for this system is $R_i = \{x \in \mathbb{R}^n : (x - x_{\text{ref},i})^T P_i (x - x_{\text{ref},i}) \leq 1\}$. while the control law for the initial system (2) is given by $u_k = K_i(x_k - x_{\text{ref},i}) + u_{\text{ref},i}$. Since (25) is a Lyapunov function, $R_i$ is an $\epsilon-$contractive for the closed-loop system (24) [10]. This property of the set $R_i$ ensures that for any $x(0) \in R_i$ the system will be driven to the desired operating point $x_{\text{ref},i}$ without violating the input and state constraints.

**Remark 1** *A modification to the offline controller synthesis and the online controller implementation must be made for the case where no feasible input vector $u_{ref,i,j}$ exists for a desired $x_{ref,i}$, i.e., the problem (8) has no solution.*

*Firstly for each extreme system $(\hat{A}_j, \hat{B}_j),\ j = 1, \ldots, 2^n$, problem (8) is solved in order to compute feasible input vectors $u_{ref,i,j},\ j = 1, \ldots, 2^n$ for the desired equilibrium point $x_{ref,i}$.*

*Next, for all extreme realizations $(\hat{A}_j, \hat{B}_j)$, the translated input constraints $\overline{\mathbb{U}}_{i,j},\ j = 1, \ldots, 2^n$ are formulated as (23). We consider as the common input constraint set $\overline{\mathbb{U}}_{i,c}$ in the translated space to be the intersection of each input constraint set, i.e., $Ui = \cap_{j=1}^{2^n} U_i, j$. Then for each operating point Lyapunov matrix $P_i$, the feedback gain matrices $K_i$ and the DoA $R_i$ are computed solving a similar problem to (20).*

*Regarding the online controller implementation, a feasible input vector $u_{ref,i(s_k)}$ has to be computed at each step for the desired operating point $x_{ref,i(s_k)} = x_{ref,i}$. In detail, the following linear programming problem is solved at each instant k.*

$$\min_{c_{i,j_k}, p_{i_k}, d_i, u_{ref,i(s_k)}} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} c_{i,j} + \sum_{i=1}^{n} w_{d_i} d_i \right\} \tag{30a}$$

*subject to*

$$x_{ref,i(s_k)} = A(s_k) x_{ref,i(s_k)} + B(s_k) u_{ref,i(s_k)}, \tag{30b}$$

$$c_{\min} \leq c_{i,j_k} \leq c_{\max}, i = 1, \ldots, n,\ j = 1, \ldots, m, \tag{30c}$$

$$\sum_{i=1}^{n} c_{i,j_k} \leq CT_j, j = 1, \ldots, m, \tag{30d}$$

$$p_{\max} - p_{i_k} \leq d_i,\ i = 1, \ldots, n, \tag{30e}$$

$$d_i \geq 0,\ i = 1, \ldots, n, \tag{30f}$$

$$u_k \in \mathbb{U}, u_k = K_i(x_k - x_{ref,i(s_k)}) + u_{ref,i(s_k)}. \tag{30g}$$

*The positive numbers* $w_{ij}, i = 1, \ldots, n, j = 1, \ldots, m,$ *and* $w_{d_i, i=1,\ldots,n,}$ *are weights of the optimization cost function. Equation (30b) ensures that* $(x_{ref,i(s_k)}, u_{ref,i(s_k)})$ *is the equilibrium point. Equations (30c)-(30e) ensure state and input constraint satisfaction. Equations (30d) and (30e) are used to maximize the admittance probability. Finally equation (30g) guarantees that the control law* $u_k$ *satisfies the input constraints at every time interval. The above procedure allows to compute a stabilization control law and a DoA that leads the system trajectory on the operating point. For this case, the input constraint satisfaction is not guaranteed a priori. However, if the optimization problem (24) is feasible, the input constraints are satisfied during the online implementation.*