On load balancing and resource allocation in cloud services

On Load Balancing and Resource Allocation in Cloud Services

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

***Abstract*—Service providers must guarantee high quality of service (QoS) for each web application in a data center and simultaneously achieve optimal utilization of their infrastructure. Meeting the Service Level Objectives (SLOs), such as response time in a dynamic environment with a dense load and varying capacity, and simultaneously minimizing the energy consumption of the data center is an open research problem. This paper presents a control framework that addresses both problems of load balancing and resource allocation of consolidated web services in cloud computing infrastructure. The proposed approach aims at succeeding the customer requirements described in a Service Level Agreement (SLA) while maximizing server utilization. A hierarchical two-layer controller is established. The local (lower) level controllers determine the capacity and admitted workload of Virtual Machines (VMs), which correspond to a set of feasible operating points with performance guarantee. The global (upper) level decides the number and topology of active VMs that serve the total service demand and activates only the minimum number of servers. The cooperation of the two control layers ensures the system stability against the fluctuations of incoming requests and the system constraints.**

I. Introduction

Power consumption and energy efficiency are becoming very important in the design and management of cloud computing data centers. Due to the large number of servers and the increasing complexity of the network infrastructure, energy costs are quickly rising in large-scale service centers. A sustainable and power-aware management system should provide a trade-off between service performance and energy consumption.

Virtualization provides a promising approach for consolidating multiple online services in fewer computing resources within a data center. This technology allows a single server to be shared among many performance-isolated platforms called virtual machines (VMs). Virtualization also provides the on-demand or utility computing – a just-in-time resource provisioning model in which computing resources such as CPU, memory, and disk space are made available to applications only as needed and not allocated statically based on the peak workload demand. By dynamically provisioning VMs, consolidating the workload, and turning on only the necessary servers, data center operators can maintain the desired Quality of Service (QoS) while achieving higher server utilization and energy efficacy.

The aim of this paper is to develop optimal resource allocation policies for web applications. The goal is to provide services which achieve QoS requirements, while minimizing the energy consumption of the computing infrastructure. In this context, QoS requirements, formally stipulated in SLA contracts, are difficult to satisfy due to the high variability of request workloads, which may vary by orders of magnitude. This leads to the problem of an efficient usage of the resources and the reduction of energy consumption. Modern cloud computing infrastructure hosts multi-tier applications in virtualized environments. Physical resources (e.g., CPU, disks, and network bandwidth) are partitioned into multiple virtual ones, creating isolated VMs which run at a fraction of the physical system capacity.

Autonomic self-managing techniques are implemented by network controllers which can establish the set of applications executed by each server (i.e., application placement problem), the request volumes at various servers (i.e., load balancing problem), and the capacity devoted to the execution of each application at each server (i.e., capacity allocation problem). In this paper we propose a two-layer controller that addresses cooperatively the problems of load balancing, capacity allocation while not violating SLA objectives and minimizing energy consumption (or equivalently making optimal use of resources).

In detail, in the local level, we apply fully distributed controllers for each VM [1]. Each controller determines a set of feasible operation points (e.g., for an agreed response time of 1 sec a CPU share of 20% is able to consolidate a workload of 50 requests/sec) together with stabilizing control strategies for each point. The local controller controls the variables of allocated capacity regulating response time.

The global (upper) level supervisory controller, considering the available set of VMs that correspond to specific operating points determined by the local controllers, determines the number and the size of VMs that are necessary to satisfy the total incoming request rate. The distribution of the workload among the activated VMs and the placement of them in the cluster of servers is made in such a way that the minimum number of servers is switched on.

It is worth observing that the control problem solved by the proposed two-stage framework cannot be addressed by approaches based on queuing theory or utility functions. The main contributions of the proposed solution are the following:

- Linear Parameter Varying (LPV) state space modeling which captures the dynamic behaviour of the underlying infrastructure.
- The determination of the set of feasible operating points relaxes the complexity of the global control optimization problem and improves scalability.
- The hierarchical two-layer controller guarantees system stability and satisfaction of the existing constraints.

The rest of the paper is structured as follows. Section II discusses related work. Section III contains an analytical description of both global and local level controllers. In section IV, the implementation and evaluation of the approach is presented. Conclusions are drawn in section V.

II. Related Work

In this section, we recall some representative studies of interest that are close to the approach proposed here. Ardagna et al. [2] presented a distributed algorithm for capacity allocation and load balancing. They modelled the systems dynamic behaviour with a M/G/1 queue and solved the joint control problem as an optimization problem applying a decomposition technique for non-linear 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 [3], 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. [4], [5] 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. Also the scalability of the approach was not demonstrated. Urgaonkar et al. [6] presented a utility based solution, which maximizes the average application throughput and energy cost of the data center. Using queuing theory models and an optimization technique framework, they addressed the problems of admission control, server selection and resource allocation separately. In [7], the authors proposed a holistic approach for application placement, admission control, resource allocation. However an assumption of linearity between server performance and CPU frequency is made. They used queuing theory models and a greedy algorithm to solve the placement problem and consequently they split admission control and capacity allocation as being two decoupled subproblems. There, admission controller is designed separately from [8] ignoring the resource allocation solution. Wang et. al. [9] proposed a two level controller combined with queuing modeling. The global level determines whether the CPU share is efficient for the total incoming workload which activates a certain number of servers. The local controller solves the resource allocation problem in order to satisfy the predicted workload. However, it was 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. Giani et al. [10] 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 [11], 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 operation points. Since all the above solutions use queuing models they are valid only for steady state conditions and they do not examine whether the operational points are feasible and stable or not.

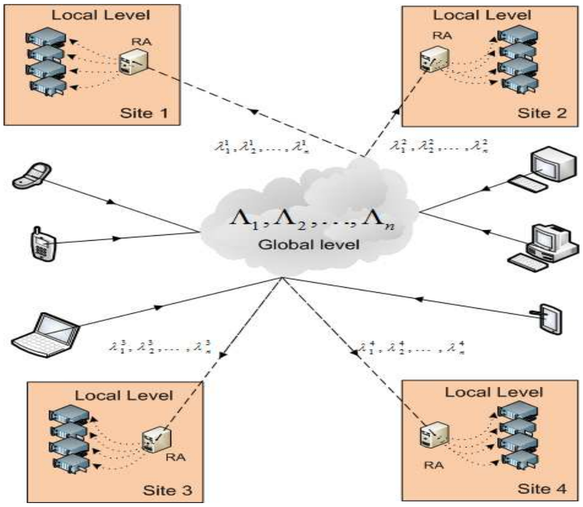

Local Level
RA
Site 1
Local Level
RA
Site 2
$\lambda_1^1, \lambda_2^1, \ldots, \lambda_n^1$
$\lambda_1^2, \lambda_2^2, \ldots, \lambda_n^2$
$\Lambda_1, \Lambda_2, \ldots, \Lambda_n$
Global level
$\lambda_1^3, \lambda_2^3, \ldots, \lambda_n^3$
$\lambda_1^4, \lambda_2^4, \ldots, \lambda_n^4$
Local Level
RA
Site 3
Local Level
RA
Site 4

Fig. 1: Cloud System Control Framework.

III. Problem Formulation

Figure 1 illustrates how the total workload Λ_i of each application is distributed to λ_i^j share to each site on a distributed cloud computing platform. It is readily apparent that centralized designs become quickly intractable for large systems. Fortunately, hierarchical or decentralized control, where multiple controllers interact with each other to satisfy system-wide QoS goals, can be used to reduce the complexity of the overall problem. In a hierarchical structure, a local controller is responsible for optimizing the behaviour of a subset of the components while satisfying the constraints imposed by a higher-level controller.

In what follows, we now briefly outline a hierarchical solution for controlling a large-scale computing system hosting

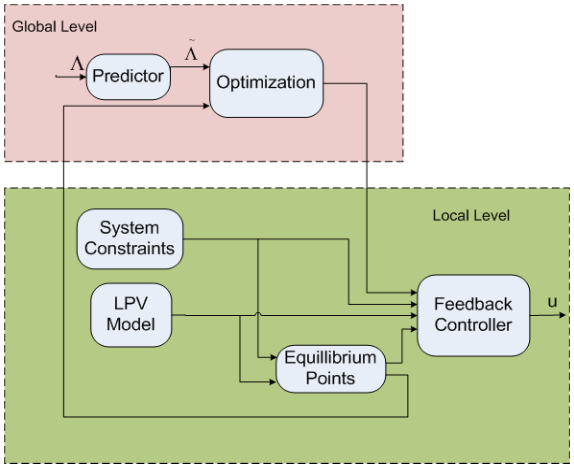


Fig. 2: Structure of the Hierarchical Control Framework.

multiple enterprise applications. The incoming application workload (Λ_i) is dispatched to the appropriate VMs (λ_i^j) hosted within the servers. The main components of our network hierarchical framework are shown in Figure 2. They consist of a monitor, a workload predictor and the two layer controllers. The monitor measures the workload and performance metrics of each application (e.g average response time in a time interval). The workload predictor forecasts future system load conditions based on historical data. The controllers within the hierarchy have the following responsibilities:

- The supervisory controller makes high-level switching decisions based on the system state and the estimation ($\tilde{\Lambda}$) of the incoming workload intensity (Λ) that dictates which physical servers are turned on or off. Moreover it decides the VM placement and chooses from a set of feasible operating points that satisfy its optimization criterion.
- Local controllers on each server initially determine feasible equilibrium points via identification. In addition, they dynamically optimize the CPU share,cap_i^j provided to VMs (Resource Allocation - RA) and the workload under their control, i.e., λ_i^j (Admission control - AC) in order to regulate the system at the chosen operating point.

A. *Workload Predictor*

In order to predict the incoming request rate of each application, we use Holt's linear exponential smoothing (LES) filter [12], which can capture the linear trend in the time series. For example, during time step k, the estimated value $\tilde{\Lambda}_k$ of incoming request rate Λ_k for a one-step prediction horizon is obtained as follows,

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

where α, β 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 $b_0 = 0.5$. Table I summarizes the notation used throughout the paper.

B. *Local Level Controller*

The local level controller is designed by utilizing the results in [1]. This controller addresses admission control and resource allocation simultaneously in a unified framework, thus making the cooperation with existing global level controllers easier. In specific, Linear Parameter Varying (LPV) state space modelling is adopted to capture the dynamic behaviour of the underlying infrastructure. The operating conditions are determined according to an optimization criterion. A set of feasible operating points $\mathbb{X}_{ref}$ is calculated to satisfy the desired QoS nominal values. Each equilibrium point is described by the triplet $(T_{ref}, C_{ref}, L_{ref})$, where T_{ref} is the desired average response time (SLA highest acceptable response time), C_{ref} is the allocated CPU capacity of VM and L_{ref} is the request rate directed and served by VM.

$$\mathbb{X}_{ref} := \{VM_i = (T_{ref,i}, C_{ref,i}, L_{ref,i}),\ i = 1, \ldots, N\}. \tag{2}$$

The resulting stabilizing state feedback control law is an affine control law which is computed using quadratic Lyapunov functions. The computational complexity of the controller implementation is small, since at every time interval only a linear program and a point location problem are solved. Furthermore, convergence to the feasible operating point and satisfaction of the system's constraints are guaranteed for a number of desired operating points of interest. Alternatively, one can design the controllers based on [13]. It is worth noting that the choice of the local controller also depends on the modeling/identification method. Thus, for linear parameter varying modeling one can use [1] while for a set of linear systems the approach in [13] is more suitable.

C. *Global Level Controller*

The global level controller decides the number of servers activated, the number and the sizes of VMs allocated to the active servers and balances the workload between the activated servers. This is done by solving a mixed integer programming optimization problem [14] whose objective function minimizes the number of active servers.

We assume that there are M available identical servers in the cluster, A is the variable that represents the number of the activated servers and $\tilde{\Lambda}_k$ is the prediction of the total incoming workload. C_j represents the total capacity of j^{th} server. $VM_{i,j}$ is member of $\mathbb{X}_{ref}$ and is placed to the j^{th} server. In order to minimize power consumption

M	Total Number of Servers
A	Number of Active Servers
Λ_k	Total incoming request rate in the k^{th} time interval
$\tilde{\Lambda}_k$	Prediction of total incoming request rate in the k^{th} time interval
Λ_{th}	Workload Threshold
$VM_{i,j}$	VM of i^{th} Equilibrium point on j^{th} server
L^i_{ref}	Target Request Rate of i^{th} Equilibrium point
C^i_{ref}	Target VM Capacity of i^{th} Equilibrium point
T^i_{ref}	Target Response Time of i^{th} Equilibrium point
$\mathbb{X}_{ref}$	Set of feasible operating points

TABLE I: Notation of System Variables.

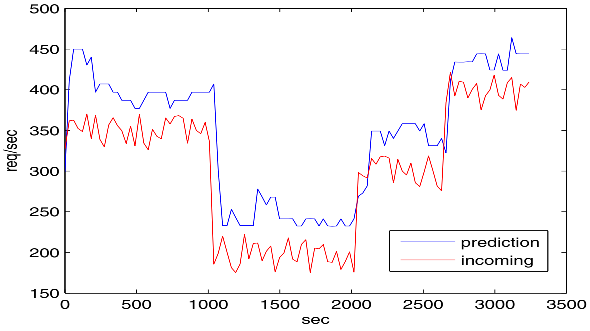


Fig. 3: Total Request Rate.

and optimize resource utilization, we formulate the following mixed integer programming optimization problem,

$$\min_{C_{ref_{i,j}}, L_{ref_{i,j}}, VM_{i,j}, A} \{A\} \tag{3a}$$

subject to

$$\sum_{j \in M} \sum_{i \in N} L_{ref_{i,j}} \geq \tilde{\Lambda}_k, \tag{3b}$$

$$\sum_{i \in N} C_{ref_{i,j}} \leq C_j, \tag{3c}$$

$$VM_{i,j} \in \mathbb{X}_{ref}, \tag{3d}$$

$$1 \leq A \leq M. \tag{3e}$$

The cost function A of the above optimization problem(3a) is the number of active servers. In specific, relation (3b) ensures that the solution of the problem consolidates the total incoming workload. Relation (3c) corresponds to the capacity constraints on each server. Relation (3d) assigns to each $VM_{i,j}$ a feasible predetermined operating point. Last, relation (3e) ensures that the number of active servers is less than the number of available servers. Variable A is an integer, thus the formulated problem (3a)-(3e) is a mixed integer linear programming problem.

Additionally, in order to reduce overhead caused by switching on and off servers and VMs at every time interval, the optimization problem is solved if a significant change of the incoming request rate is observed. Thus Λ_{th} is the minimum workload change that activates the execution of the optimization problem. Lower workload fluctuations are easily handled by local level controllers. They are able to regulate system performance near the desired operating point. When the output of the global level controller dictates the shut down of a VM, then the solution of the optimization problem maintains it active for the next control time interval in order to serve the already directed requests to it. If there is available capacity on the server, VM is allocated with the capacity of the previous time interval otherwise the rest of the available capacity is allocated.

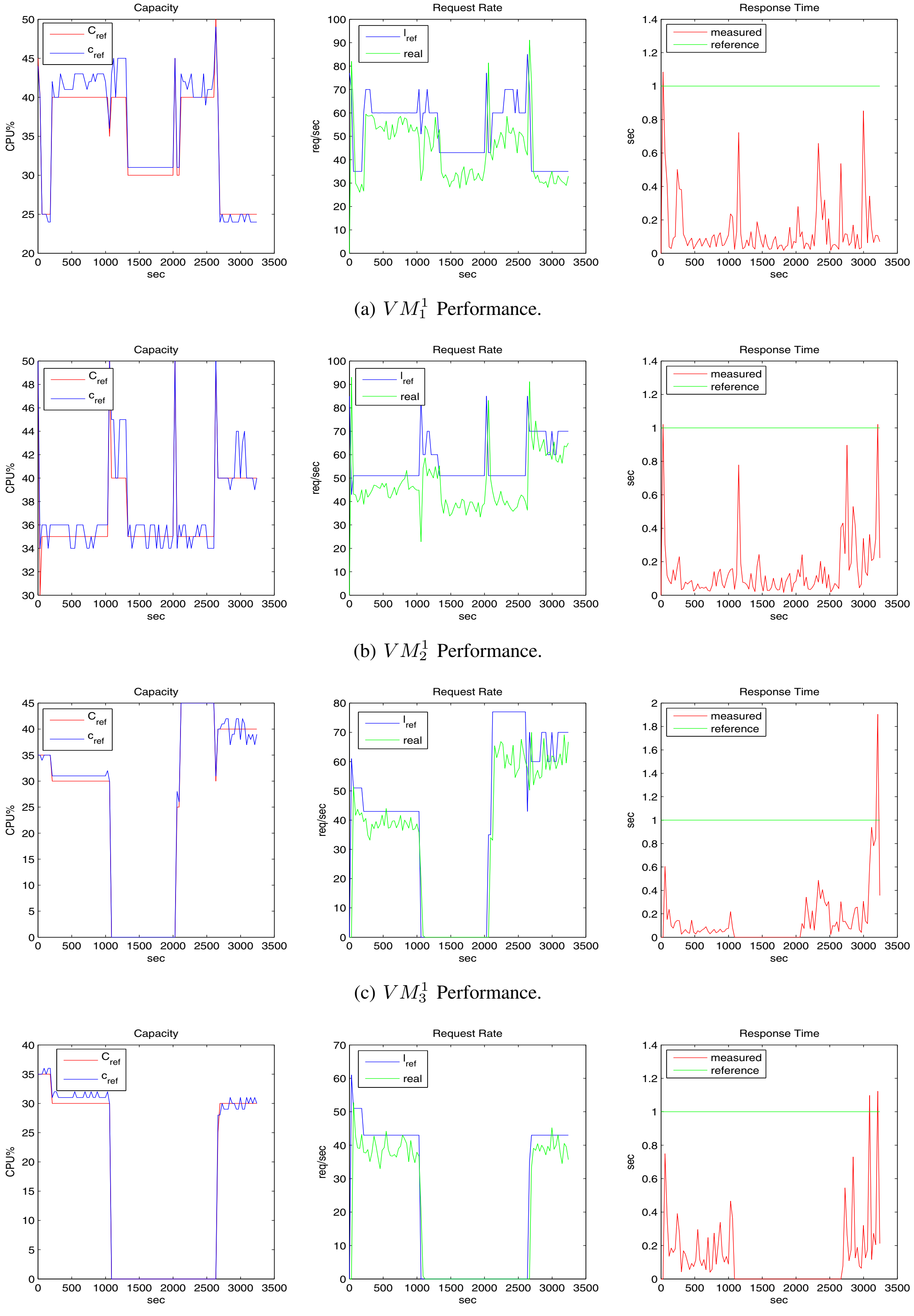
IV. Evaluation

A simple testbed is built to evaluate the performance of the two-layer control framework. The server cluster contains three identical servers ($M = 3$). The local controllers provide the global level controller with the necessary set $\mathbb{X}_{ref}$. The operating points are calculated in a grid defined by $T_{ref,i} = 0.5 : 0.5 : 3$ and $C_{ref,i} = 25 : 5 : 50$ according to [1, section 3.4]. Indicatively for $T_{ref} = 1sec$ the following operating points are calculated,

$$(T_{ref}, C_{ref}, L_{ref}) = \{(1, 20, 44), (1, 25, 50), (1, 30, 60), (1, 35, 68), (1, 40, 77), (1, 45, 86), (1, 50, 93)\}.$$

The value of Λ_{th} is selected empirically to 60. We make a 10% overestimation of the incoming workload which is a common technique [9] to eliminate the prediction error. The local level controllers compute the final control law for each VM according to section 3.5 of [1] taking into context the current response condition in order to guarantee the system stability and the satisfaction of the underlying constraints. The output of the local level controllers is updated every $30sec$. The desired reference value of response time for all VMs is set to $1sec$. After the determination of the final control law (actual CPU share and actual workload) from the local controllers, a weighted Round Robin algorithm is used to direct the assigned portion of the total incoming workload to the corresponding VM. In the following figures, we illustrate the performance of the two-layer control framework. Figure 3 illustrates the total incoming request rate and its prediction. Initially the average workload is around $350req/sec$ (heavy load) and after $1000sec$ drops to $200req/sec$ (light load). At $2000sec$ the workload increases to $300req/sec$ (medium load) and at the last part of the experiment is almost $400req/sec$ (heavy load).

Figure 4 shows the performance of four VMs on different servers. The rest of VMs are omitted due to lack of space. VM^i_j is the i^{th} active VM on j^{th} server. The left column of Figure 4 depicts the capacity (C_{ref} - red line) of operating point and the locally assigned capacities of VMS (c_{ref} - blue line) by the local controllers. Initially the incoming request rate is high and all servers are switched on. On the second part the workload decreases and the third server is turned off. After $2000sec$, the workload gradually increases, thus VM^1_3 is activated initially. When the request rate increases to $400req/sec$, VM^2_3 is also turned on. This shows that when a sudden change happens in the prediction of the workload, the global level controller is triggered and the



(a) VM_1^1 Performance.

(b) VM_2^1 Performance.

(c) VM_3^1 Performance.

(d) VM_3^2 Performance.

Fig. 4: Resource allocation, admission of requests and reponse time for VM's.

	$0 - 200s$	$200 - 1000s$	$1000 - 1200s$	$1200 - 2000s$	$2000 - 2600s$	$2600 - 3300s$
VM_1^1	25	40	40	30	40	25
VM_1^2	25	30	30	25	30	25
VM_1^3	25	30	0	25	30	25
VM_1^4	25	0	0	0	0	25
Server 1	**100**	**100**	**70**	**80**	**100**	**100**
VM_2^1	30	35	45	35	35	40
VM_2^2	30	35	40	35	35	30
VM_2^3	30	30	0	30	30	25
Server 2	**90**	**100**	**85**	**100**	**100**	**95**
VM_3^1	35	30	0	0	45	40
VM_3^2	35	30	0	0	0	30
VM_3^3	30	0	0	0	0	30
Server 3	**100**	**60**	**0**	**0**	**40**	**100**

TABLE II: VMs and Servers Capacities.

optimization algorithm redistributes the capacity of VMs in order to satisfy the varying demand with the minimum number of active servers. Table II shows the capacity of each VM and the total capacity of server when the optimization algorithm is executed by a workload change.

The second column of Figure 4 shows the reference value (l_{ref} - blue line) of the workload computed by the local controller and the real admitted request rate of each VM (green line) and the right column of Figure (4) illustrates the response time of each VM. It is obvious that the proposed control framework achieves high level of QoS and succeeds in maintaining the response time of each VM under the desired reference value in average and has few violations when changes in workload occurred. This happens due to the overhead caused by the switching on and shutting down VMs and servers and the delay of the workload prediction to adapt to the new values.

V. Conclusion and Future Work

An autonomous control framework of managing QoS in cloud computing is proposed. A two-layer hierarchical controller is implemented for this purpose. The main contribution of the approach is the dynamic determination of feasible operating points which allows to confront the high non linearity of the system contrary to the steady state solutions provided by queueing theory. Secondly this allows to reduce the complexity of the supervisor control (optimization problem) and improves scalability. Stability and constraints satisfaction are guaranteed. Different scenarios were tested in order to evaluate its performance. Experimental results in a real setup indicate that the proposed solution can achieve high level of QoS.

To improve the existing work we intend to implement different techniques for the two-level controller. Firstly for improving load balancing between servers we intend to insert load balancing constraints in the optimization problem (3a)-(3e). Secondly to reduce extensive identification procedure we will exploit the aspect of using Takagi Sugeno (T-S) adaptive fuzzy modelsfor identification and control of the local systems. For the global level controller in order to minimize overhead we will focus on constructing fuzzy decision maker for load balancing and application placement.

VI. Acknowledgements

Nikolaos Athanasopoulos is supported by the People Programme (Marie Curie Actions) of the European Unions Seventh Framework Programme (FP7/2007-2013) under REA grant agreement 302345.

References

[1] D. Dechouniotis, N. Leontiou, N. Athanasopoulos, A. Christakidis, and S. Denazis, "An autonomous admission control and resource allocation framework for consolidated web applications," University of Patras, http://nam.ece.upatras.gr/sites/default/files/report.pdf, Tech. Rep., 2013.

[2] 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*, 2010, pp. 1 – 12.

[3] 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*, vol. 70, no. 4, pp. 344–362, 2010.

[4] 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)*, 2006, pp. 74–83.

[5] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing enviroments via lookahead control," *Springer Journal on Cluster Computing*, vol. 12, no. 1, pp. 1–15, 2009.

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

[7] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang, "Energy-aware autonomic reource allocation in multitier virtualized enviroments," *IEEE Transactions on Services Computing*, vol. 5, no. 1, pp. 2–19, 2012.

[8] L. Cherkasova and P. Phaal, "Session-based admission control: A mechanism for peak load management of commercial web sites," *IEEE Transactions on Computers*, vol. 51, no. 6, pp. 669–685, 2002.

[9] 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)*, 2012, pp. 115–124.

[10] 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*, 2011, pp. 6709 – 6714.

[11] 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*. Springer, 2010, pp. 73–96.

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

[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)*, 2012, pp. 145–149.

[14] A. Schrijver, *Theory of linear and integer programming*. John Wiley and Sons, 1988.