Model Predictive Control for Energy-Efficient, Quality-Aware, and Secure Virtual Machine Placement

Mauro Gaggero and Luca Caviglione

***Abstract*—Modern datacenters rely on virtualization to deliver complex and scalable cloud services. To avoid inflating costs or reducing the perceived service level, suitable resource optimization techniques are needed. Placement can be used to prevent inefficient maps between virtual and physical machines. In this perspective, we propose a holistic placement framework considering conflicting performance metrics, such as the service level delivered by the cloud, the energetic footprint, hardware or software outages, and security policies. Unfortunately, computing the best placement strategies is nontrivial, as it requires the ability to trade among several goals, possibly in a real-time manner. Therefore, we approach the problem via model predictive control to devise optimal maps between virtual and physical machines. Results show the effectiveness of our technique in comparison with classical heuristics.**

Note to Practitioners: ***Abstract*—This paper is motivated by the success of cloud-based services. Specifically, we consider the case where the resources of datacenters are accessed through the Infrastructure-as-a-Service paradigm. To efficiently handle the load of requests and to tame operational costs, a proper optimization is needed. To this end, we focus on the selection of the best maps between virtual and physical machines. We propose a holistic placement framework for the deployment of user-requested virtual machines by pursuing different performance goals, such as counteract to hardware outages or reboots due to software aging issues, ensure proper security policies, maintain a suitable service level perceived by users, and reduce power requirements. The optimal strategies are computed with model predictive control, which allows to consider complex constraints and take advantage of future information. The proposed framework is tested in different scenarios characterized by a variety of workloads and traces gathered in a real cloud datacenter. Results indicate that our approach outperforms bin-packing techniques.**

***Index Terms*—Cloud computing, energy efficiency, model predictive control, quality-aware placement, security, virtual machine placement.**

I. Introduction

Virtualization enables to access on-demand computing and storage services in a flexible and cost-effective manner. This is the key component of the cloud paradigm, where computing, storage, and network resources of physical machines (PMs) available in a datacenter are partitioned into virtual machines (VMs) [1]. Thus, it is crucial to define how to map VMs over PMs, possibly to satisfy some performance criteria. To this end, two complementary approaches are available: *placement* and *consolidation*. Placement prevents inefficient allocations when VMs are firstly created by choosing the most suitable PMs on which to deploy VMs requested by users. However, fluctuations in the workload jointly with hardware/software outages may require to update how VMs are placed. To this end, consolidation periodically migrates VMs over PMs to maintain proper performance levels. Many works already investigated consolidation (see [2] for a survey), and all point out that live migrating a VM requires significant resources and poses several technical challenges [3]. Therefore, in this paper we concentrate on placement and propose a novel strategy based on model predictive control (MPC).

Designing new placement techniques is a nontrivial task, especially in Internet-scale datacenters, where hardware failures may lead to permanent or temporary unavailability of PMs. Besides, the intensive usage of virtualization frameworks could lead to software aging issues requiring proper rejuvenation strategies [4], [5]. Both situations may cause a reduction of the service level perceived by users since some VMs could be unavailable. Concerning issues due to aggressive packing strategies, running too many VMs over a single PM may create race conditions on shared resources or colocation interference, potentially causing performance losses [6], [7]. Security further complicates the selection of the PM where to deploy a VM, as some VMs should not be mixed to prevent data leakage or privacy breaches [8]. Lastly, the power required by the datacenter may be influenced by the placement policy and lead to high environmental footprints or heat dissipation problems [9].

Unfortunately, the conflicting nature of the aforementioned constraints increases the effort needed to guarantee desired performance levels. For instance, pursuing energy efficiency may require to reduce the number of active PMs by packing many VMs over the same server (see, e.g., [10] for the case of energy-aware consolidation). However, this may violate security policies or overload PMs, thus causing performance degradation. In general, such aspects influence the available pool of resources, which can be further reduced by the temporary unavailability of PMs [11]. As a consequence, without proper countermeasures, the "quality" experienced by users can degrade in a quick and severe manner.

In this perspective, along the lines of the preliminary work [12], we propose a novel mechanism to devise optimal placement strategies considering all the aforementioned performance aspects. In particular, the goal is to counteract hardware outages or reboots due to software aging issues,

M. Gaggero and L. Caviglione are with the Institute of Intelligent Systems for Automation, National Research Council of Italy, I-16149 Genoa, Italy (e-mails: mauro.gaggero@cnr.it; luca.caviglione@cnr.it).

prevent violations of security policies, reduce operational costs by taming power requirements, and maintain a suitable service level perceived by users. To this aim, we adopt MPC, especially owing to its ability to handle constraints while exploiting future information [13] as well as its proven capacity to face a variety of engineering problems [14], [15]. The best map between VMs and PMs is chosen by solving a finite-horizon optimal control problem at each time step using a suitable discrete-time dynamic model of the cloud infrastructure.

The contributions of this work are: (i) the development of a holistic approach for placement allowing a tradeoff among the service level delivered by the cloud and technological constraints characterizing large-scale scenarios; (ii) the definition of a new, dynamic model to capture the evolution of VMs created over PMs; (iii) the adoption of MPC to solve the VM placement problem, possibly by satisfying real-time constraints; (iv) the comparison with heuristics proposed in the reference literature.

The rest of this paper is organized as follows. Section II presents the related work, while Section III introduces the problem statement. Section IV deals with the discrete-time dynamic model of the datacenter. Section V describes the proposed MPC approach for placement, while Section VI contains the metrics used to quantify performances. Section VII showcases the numerical results. Finally, Section VIII concludes the paper.

II. Related Works

Placement allows to select the most suitable PM for hosting a VM. An ideal scheme should consider several performance goals, such as the reduction of the network traffic or the energy consumed by the datacenter, as well as prevent the need of future VM migrations or aggressive consolidation policies [16]. It is also fundamental to balance the load offered to the datacenter, implement fail-over disciplines, improve scalability, and avoid wastages of resources [17]. Several techniques have been proposed during the years, but the most recent trend exploits placement to pursue very specific goals [16], [17] or some form of energy-awareness [18].

As regards methods considering specific aspects of a datacenter, in [19] a structural-aware placement scheme is proposed to optimize how VMs are located within a server rack. A similar idea is used in [20], where a power consumption model is used to exploit the "natural" grouping of PMs, e.g., racks, computing rooms, or company-defined branches. Guaranteeing network performances is another goal usually enforced through placement mechanisms. For instance, [21] showcases a method to deploy VMs over PMs to avoid bottlenecks in the links connecting remote datacenters. Similar approaches are pursued in [22] and [23], which propose strategies to avoid wastages of resources by taking into account also the traffic among VMs. Reference [24] explicitly considers heterogeneous servers by computing relations among VMs to maximize the used resources, whereas [25] focuses on the peak usage of every VM. A relevant portion of works focuses on how placement can be used to provide some form of fault-tolerance to users. Specifically, [26] and [27] investigate approaches for placing VMs to counteract server failures. A similar idea based on redundancy is discussed in [28], but authors prefer to focus on legislation aspects and colocation issues. Maintaining a proper service level is another important topic, especially to maximize revenues [29] or support novel cloud-based environments (see, e.g., [30] for the case of MapReduce). In general, the "quality" of the service perceived by users of a cloud datacenter strictly depends on the efficiency of sharing physical resources among different VMs. To this aim, the placement strategy plays a major role, as aggressive packings could lead to performance degradation. In this vein, [31] investigates placement and consolidation strategies considering cache contention, while [32] and [33] define affinity metrics to avoid performance losses due to competing VMs. Security is another important goal that has to be considered when designing placement mechanisms for real-world datacenters [34]. In more detail, some VMs should be maintained separated to avoid information leakage [8] or co-residence attacks [35].

Energy-awareness is the other major research trend. In this case, placement aims at reducing the energetic cost of the datacenter, including additional consumption due to thermal dissipation [36]. The literature abounds of works mixing green computing aspects with placement (see, e.g., [37] for a recent survey on energy-aware placement and consolidation techniques). Concerning possible examples, [38] showcases how VMs can be efficiently deployed over PMs by means of suitable predictions of the used CPU. Reference [39] deals with an ant-colony optimization strategy to reduce the energy impact of VMs, while [40] investigates geo-distributed datacenters, possibly equipped with renewable power sources. Other works dealing with *ad-hoc* placement strategies for pursuing energy-awareness are [41] and [42].

In general, works mixing energetic aspects with specific performance goals are only a fraction [37]. Moreover, when energy is a performance indicator, the remaining constraints are usually limited or treated as less important (see, e.g., [21], [20]).

Concerning the use of MPC in the context of cloud computing applications, it has been already employed for consolidation [10], [43], but not for placement (apart our preliminary work [12]). Rather, it has been used to solve specific problems, such as to adjust the amount of resources allocated to each VM to match the demand while minimizing the energy cost [44], or to distribute computations in parallel computing systems [45].

Therefore, at the best of our knowledge, this work is the first one considering a holistic placement approach based on MPC allowing to trade power consumption and other performance criteria, such as the overall service level of the cloud application, hardware or software failures including rejuvenation issues, and security constraints.

III. Problem Statement

The Infrastructure as a Service (IaaS) is one of the most popular paradigms at the basis of cloud applications [2], [10], [37]. In this paper, we focus on an IaaS model delivered

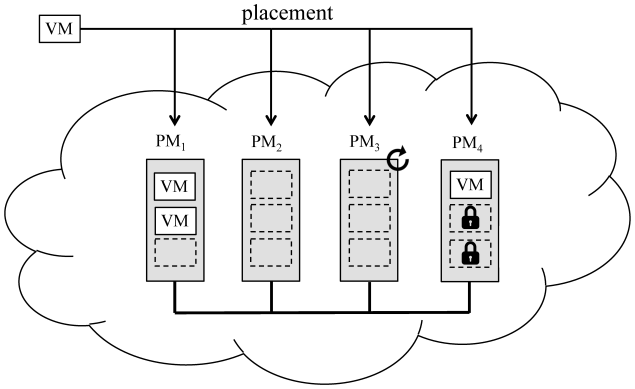

Fig. 1. Different choices for the placement of a new VM.

TABLE I
NOTATION FOR THE DYNAMIC MODEL.

state variables	y_{ij}	class of a VM
	x_{ij}^k	resource used by a VM
	a_{ij}	remaining lifetime of a VM
	s_{ij}	security requirement of a VM
control inputs	u_{ij}	request for placing a new VM
	v_{ij}	placing location of a new VM
input variables	$\hat{y}_l$	class of a new VM
	$\hat{x}_l^k$	resource used by a new VM
	$\hat{a}_l$	remaining lifetime of a new VM
	$\hat{s}_l$	security requirement of a new VM
	g_i	availability of a PM
	b_i^k	capacity of a PM

through a datacenter composed of M PMs, each one able to host up to N VMs. As it happens in Internet-scale scenarios, we assume that the datacenter can always satisfy all the incoming requests [46]. According to the IaaS paradigm, users request a desired number of VMs for a specific timeframe. Each VM is modeled as a bundle of three different resources, i.e., CPU, disk, and network [10], [47], [48]. If successfully created over a PM, the VM is said to be "deployed," whereas the term "undeployed" denotes that it has been removed. To capture a realistic infrastructure, we assume that users can select various classes of VMs reflecting specific service plans [49]. Each plan is characterized by different levels of utilization for the CPU, disk, and network [7]. Without loss of generality, we assume that users do not change their plan over time. In the case of upgrades, a VM is undeployed and a new, compliant one is created [50]. As extensively done in the literature (see, e.g., [7], [10], [50]), we consider S classes of VMs, each one characterized by different utilization levels for the CPU, disk, and network.

The map between PMs and VMs usually influences the performances. For example, a PM running two CPU-bound VMs performs better than one running two I/O-bound VMs [51]. Therefore, VMs aggressively competing on the same PM could reduce the amount of resources made available through the cloud infrastructure. Another issue concerns the fraction of resources needed by the virtualization layer to run VMs and manage the underlying hardware. In the following, we will take into account such aspects and refer to them as colocation interference and overhead, respectively. Furthermore, VMs could become temporarily unavailable due to hardware issues (e.g., a PM is unreachable due to network or local outages) or rejuvenation mechanisms (e.g., the virtualization layer needs a reboot to face software glitches) [1], [5]. Lastly, applications managing sensitive data or performing mission-critical tasks may require to run VMs in a secure environment [8], [34], which prevents a PM to host VMs with mixed security profiles.

As hinted, choosing the PM where to create VMs requested by users is a challenging task. In fact, a successful placement strategy should mitigate the impact of outages and interferences, ensure security, and reduce the energetic footprint of the datacenter by turning off PMs without running VMs. Figure 1 depicts a toy example of an IaaS framework running on a datacenter characterized by $M = 4$ and $N = 3$, together with the different alternatives that the placement strategy could implement. For instance, deploying the new VM over PM_1 favors power savings, as PM_2 and PM_3 could be put in sleep since they do not host any VM. Unfortunately, this could reduce the resources available to users, as the VM might experience colocation interference issues. To mitigate interferences, the new VM could be deployed over PM_2 or PM_3, but a the price of renouncing to pursue power efficiency. However, the creation of the VM over PM_3 could generate a service interruption since PM_3 requires a reboot for rejuvenating the hosting platform. Finally, placing the new VM over PM_4 could be possible or not depending on security credentials.

IV. Dynamic Model and Constraints

To compute the optimal placement strategy, a dynamic model describing the time evolution of VMs offered by the IaaS framework is needed. Hence, in this section we present proper dynamic equations and suitable constraints.

A. Dynamic Model

Let us consider discrete times $t = 0, 1, \ldots, T$, where T is a given horizon, and introduce proper state, control, and input variables, partially borrowed from [10], [12] and summarized in Table I.

State variables are needed to keep track of the map between VMs and PMs at each time instant t. The following quantities are defined for the j-th VM running on the i-th PM, where $i = 1, \ldots, M$, $j = 1, \ldots, N$, and $t = 0, 1, \ldots, T$:

- $y_{ij}(t) \in \{1, \ldots, S+1\}$ is the class of the VM. If the machine is not deployed, $y_{ij}(t) = S + 1$. As it will be detailed later on, $S + 1$ is a "dummy" class that avoids burdening the notation;
- $x_{ij}^k(t) \in [0, 1]$ is the fraction of CPU ($k = c$), disk ($k = d$), and network ($k = n$) used by the VM. For a non-deployed VM, $x_{ij}^k(t) = 0$;
- $a_{ij}(t) \in \mathbb{N}$ is the remaining lifetime of the VM. When it is equal to 0, the VM has completed its life cycle or it is not deployed;

- $s_{ij}(t) \in \{0,1\}$ indicates whether the VM has security requirements. In this case, $s_{ij}(t) = 1$.

For the sake of brevity, let us define $\underline{x}_t \triangleq \text{col}\left[y_{ij}(t), x_{ij}^k(t), a_{ij}(t), s_{ij}(t), i = 1,\dots,M, j = 1,\dots,N, k = c,d,n\right] \in \mathbb{R}^{6MN}$.

To account for the characteristics of new VMs requested by users, we introduce "hat" versions of the state variables. Specifically, at each time $t = 0,1,\dots,T-1$, users demand for a given number of VMs. We denote such a number by $L(t)$. To characterize the l-th new VM requested at time t, where $l = 1,\dots,L(t)$, we consider the variables $\hat{y}_l(t) \in \{1,\dots,S\}$, $\hat{x}_l^k(t)$, $\hat{a}_l(t)$, and $\hat{s}_l(t)$, grouped in the vector $\underline{r}_t \triangleq \text{col}\left[\hat{y}_l(t), \hat{x}_l^k(t), \hat{a}_l(t), \hat{s}_l(t), l = 1,\dots,L(t)\right] \in \mathbb{R}^{6L(t)}$.

To decide the mapping between the requested VMs and the available PMs, at each time $t = 0,1,\dots,T-1$ we define proper control inputs for $i = 1,\dots,M$ and $j = 1,\dots,N$:

- $u_{ij}(t) \in \{0,1\}$ is equal to 1 if the j-th VM on the i-th PM is created at time t. Otherwise, it is equal to 0;
- $v_{ij}(t) \in \{1,\dots,L(t)\}$ is the index of the new VM created as the j-th VM on the i-th PM at time t.

Let us collect all the control inputs in the vector $\underline{u}_t \triangleq \text{col}\left[u_{ij}(t), v_{ij}(t), i = 1,\dots,M, j = 1,\dots,N\right] \in \mathbb{R}^{2MN}$.

To model the reliability and the heterogeneity of PMs available in the datacenter, the following additional variables are introduced. For the i-th PM, $i = 1,\dots,M$, we define the following quantities at each time $t = 0,1,\dots,T-1$:

- $g_i(t) \in [0,1]$ is the probability of having the PM correctly running. To avoid burdening the notation, $g_i(t)$ considers different types of outages (e.g., network or server failures and software crashes), together with temporary unavailability due to reboots for software rejuvenation;
- $b_i^k \in (0,1]$ is the percentage of CPU ($k = c$), disk ($k = d$), and network ($k = n$) capacity with respect to the most powerful PM in the datacenter, for which $b_i^k = 1$.

Let us gather all such additional quantities in the vector $\underline{z}_t \triangleq \text{col}\left[g_i(t), b_i^k, i = 1,\dots,M, k = c,d,n\right] \in \mathbb{R}^{4M}$.

The evolution of the state variables is governed by the following discrete-time dynamic equations, where $t = 0,1,\dots,T-1$, $i = 1,\dots,M$, $j = 1,\dots,N$, and $k = c,d,n$:

$$\begin{aligned}
y_{ij}(t+1) &= \chi[a_{ij}(t)]\, y_{ij}(t) + u_{ij}(t)\, \hat{y}_{v_{ij}(t)}(t) \\
x_{ij}^k(t+1) &= \chi[a_{ij}(t)]\, x_{ij}^k(t) + u_{ij}(t)\, \hat{x}_{v_{ij}(t)}^k(t) \\
a_{ij}(t+1) &= \max\{0, a_{ij}(t) - 1 + u_{ij}(t)\, \hat{a}_{v_{ij}(t)}(t)\} \\
s_{ij}(t+1) &= \chi[a_{ij}(t)]\, s_{ij}(t) + u_{ij}(t)\, \hat{s}_{v_{ij}(t)}(t)
\end{aligned} \tag{1}$$

where the function $\chi(\cdot)$ is such that $\chi(z) = 1$ if $z \neq 0$ and $\chi(z) = 0$ otherwise.

As shown, the right-hand-side of each equation in (1) describes the behavior at time $t+1$ of a given variable and is composed of two terms. The first one is inherited from time t and is a snapshot of the IaaS service, i.e., it tracks the current map between VMs and PMs. In particular, for each VM, the remaining lifetime is decreased of one unit until it reaches the zero level. If the VM expires, we have $\chi(a_{ij}(t)) = 0$, and the contribution of time t is voided. Instead, the second term accounts for new requests to be placed. In this case, the "hat" variables update the corresponding portion of the state since $u_{ij}(t) = 1$.

Using the vectors $\underline{x}_t$, $\underline{u}_t$, and $\underline{r}_t$ defined above, (1) can be rewritten in compact form for $t = 0,1,\dots,T-1$, as follows:

$$\underline{x}_{t+1} = f_t(\underline{x}_t, \underline{u}_t, \underline{r}_t) \tag{2}$$

with the obvious definition of $f_t : \mathbb{R}^{6MN} \times \mathbb{R}^{2MN} \times \mathbb{R}^{6L(t)} \to \mathbb{R}^{6MN}$. The subscript t suggests the time-varying length of the vector $\underline{r}_t$.

B. Constraints

The model given by (1) is completed through suitable constraints. To guarantee the placement of all the requested VMs, we impose the following for $t = 0,1,\dots,T-1$:

$$\sum_{i=1}^{M}\sum_{j=1}^{N} u_{ij}(t) = L(t). \tag{3}$$

Clearly, each new VM must be deployed only once. Thus, for $l = 1,\dots,L(t)$ and $t = 0,1,\dots,T-1$, the following constraints hold:

$$\sum_{i=1}^{M}\sum_{j=1}^{N} u_{ij}(t)\,(v_{ij}(t) - l) = 1. \tag{4}$$

We point out that the number of constraints in (4) is time-variant and equal to $L(t)$. Then, the placement procedure must ensure that only one VM of index j exists on the i-th PM. This is taken into account through the following constraints for $i = 1,\dots,M$, $j = 1,\dots,N$, and $t = 0,1,\dots,T-1$:

$$\begin{aligned}
u_{ij}(t) &\geq -R(1 - \chi(a_{ij}(t))) \\
u_{ij}(t) &\leq R(1 - \chi(a_{ij}(t)))
\end{aligned} \tag{5}$$

where R is a large positive constant. Constraints (5) are of equality type if $\chi(a_{ij}(t)) = 1$, i.e., if another VM with the same pair i and j is deployed. Otherwise, they are trivially satisfied.

Lastly, the outcome of the placement should not exceed the amount of resources available for a given server. This limits the number of VMs that can be hosted on a PM. Hence, for $i = 1,\dots,M$, $k = c,d,n$, and $t = 0,1,\dots,T$, the following constraint is entrusted:

$$\sum_{j=1}^{N} x_{ij}^k(t) \leq b_i^k. \tag{6}$$

As done for the dynamic equations, it is also possible to write the constraints in a compact form using the vectors $\underline{x}_t$, $\underline{u}_t$, and $\underline{z}_t$. Specifically, (3) can be expressed as

$$\underline{d}'\underline{u}_t = L(t) \tag{7}$$

where $\underline{d} \in \mathbb{R}^{2MN}$ is a suitable 0-1 vector. Constraint (4) can be written by means of a proper function $h : \mathbb{R}^{2MN} \to \mathbb{R}^{L(t)}$ and a vector $\underline{1}_{L(t)} \in \mathbb{R}^{L(t)}$ with all the components equal to 1, i.e.,

$$h(\underline{u}_t) = \underline{1}_{L(t)}. \tag{8}$$

Then, (5) becomes

$$A\underline{u}_t = p(\underline{x}_t) \tag{9}$$

where $A \in \mathbb{R}^{2MN \times 2MN}$ is a 0-1 matrix and $p : \mathbb{R}^{6MN} \to \mathbb{R}^{2MN}$ is a suitably-defined function. Lastly, after introducing a proper 0-1 matrix $B \in \mathbb{R}^{4M \times 6MN}$, (6) becomes

$$B\underline{x}_t \leq \underline{z}_t. \tag{10}$$

V. Placement Using MPC

In this section, we present the placement mechanism based on MPC used to compute how the VMs requested by users are deployed on the PMs available in the datacenter. To this aim, an efficient VM-to-PM map should consider the following goals:

1) reduce the effects of churn;
2) mitigate colocation interference;
3) minimize power consumption;
4) enforce security requirements.

As it will be detailed later on, goals 1), 2) and 3) are pursued through proper penalization terms in a cost function to be maximized, whereas 4) is enforced via a hard constraint.

1) Reduce the effects of churn: as said, machine or network outages together with software rejuvenation may lead to churn, i.e., servers enter and leave the pool of available PMs. Therefore, the placement mechanism should prevent that a large number of VMs is deployed on "unreliable" PMs, i.e., those for which the probability $g_i(t)$ is low. In fact, when a PM becomes unavailable, all the hosted VMs become unavailable as well. In general, the recovery from such a situation is nontrivial and requires proper fault-tolerance mechanisms, such as checkpointing or replication (see, e.g., [52], [53], and the references therein). However, the investigation of these aspects is outside the scope of this paper. Clearly, the more VMs are deployed on the same PM, the higher is the impact of churn in terms of reduction of resources. For the i-th PM, this is taken into account by the quantity $\rho_i(t) \in [0,1]$ defined as follows for $i = 1, \ldots, M$ and $t = 0, 1, \ldots, T$:

$$\rho_i(t) \triangleq \prod_{j=1}^{N} \chi\left(a_{ij}(t)\right) g_i(t) + \left(1 - \chi\left(a_{ij}(t)\right)\right).$$

In particular, the larger is the number of VMs deployed over an unreliable PM, the lower is the value of $\rho_i(t)$. Instead, if the i-th PM is reliable (i.e., $g_i(t) = 1$) or it has no running VMs, we have $\rho_i(t) = 1$. To consider all the PMs composing the datacenter, let $\rho(t) \in [0,1]$ be the average of $\rho_i(t)$ over M for $t = 0, 1, \ldots, T$:

$$\rho(t) \triangleq \frac{1}{M} \sum_{i=1}^{M} \rho_i(t).$$

2) Mitigate colocation interference: the amount of resources effectively available to VMs depends on how they are placed on PMs. For instance, deploying too many VMs on the same server may lead to contentions and race conditions, typically defined as colocation interference. Hence, we introduce an interference matrix $\Theta \in [0,1]^{S+1 \times S+1}$, whose elements θ_{kl} measure the interference between the classes of the k-th and l-th VMs when they are deployed on the same PM. Specifically, $\theta_{kl} \simeq 0$ indicates that the classes of the k-th and l-th VMs severely interfere, whereas $\theta_{kl} \simeq 1$ states that they can efficiently coexist on the same PM. For the j-th VM deployed on the i-th PM, we introduce the variable $\zeta_{ij}(t) \in [0,1]$ quantifying the contention of resources among the VM and the other ones running on the same PM. Therefore, for $i = 1, \ldots, M$, $j = 1, \ldots, N$, and $t = 0, 1, \ldots, T$, we consider

$$\zeta_{ij}(t) \triangleq \prod_{r=1, r \neq j}^{N} \theta_{y_{ij}(t) y_{ir}(t)}. \tag{11}$$

Clearly, the lower is $\zeta_{ij}(t)$, the larger are the interferences. As hinted, the dummy class $S+1$ denoting a non-deployed VM has been introduced to write $\zeta_{ij}(t)$ as a product in (11), and it avoids burdening the notation. Then, we define a measure of efficiency $\eta_i(t) \in [0,1]$ for the i-th PM equal to the average of the interferences experienced by all the VMs deployed on it, where $i = 1, \ldots, M$ and $t = 0, 1, \ldots, T$:

$$\eta_i(t) \triangleq \frac{1}{N} \sum_{j=1}^{N} \zeta_{ij}(t).$$

In other words, $\eta_i(t)$ weights the amount of CPU, disk, and network resources of the i-th PM made available to users. For instance, a small value of $\eta_i(t)$ indicates that the placement of VMs on the i-th PM is causing severe contentions. As a consequence, the efficiency $\eta(t) \in [0,1]$ of the entire datacenter can be defined as follows for $t = 0, 1, \ldots, T$:

$$\eta(t) \triangleq \frac{1}{M} \sum_{i=1}^{M} \eta_i(t).$$

3) Minimize power consumption: in general, the power required by a datacenter is proportional to the number of active PMs, i.e., servers with at least one running VM. In fact, a PM not hosting any VMs can be put in a low-power mode. To this aim, the placement mechanism should maximize the number of PMs with no running VMs, denoted by $\omega(t)$ and defined as follows for $t = 0, 1, \ldots, T$:

$$\omega(t) \triangleq M - \sum_{i=1}^{M} \chi\left(\sum_{j=1}^{N} a_{ij}(t)\right).$$

Specifically, large values of $\omega(t)$ reflect in a reduced power consumption. Unfortunately, saving power by aggressively packing VMs on PMs may conflict with the mitigation of the effect of churn or exacerbate colocation interference, as they could require a more "sparse" allocation.

4) Enforce security requirements: to avoid information leakage or prevent attacks, VMs may need to run in a secured or isolated environment. This prevents certain allocations, as VMs with security requirements cannot be mixed on the same PM. Since security is usually a non-negotiable parameter, we introduce a hard constraint for $i = 1, \ldots, M$ and $t = 0, 1, \ldots, T$, as follows:

$$\sum_{j=1}^{N} s_{ij}(t) \leq 1. \tag{12}$$

This constraint guarantees that only one VM with strict security requirements (i.e., such that $s_{ij}(t) = 1$) can be placed

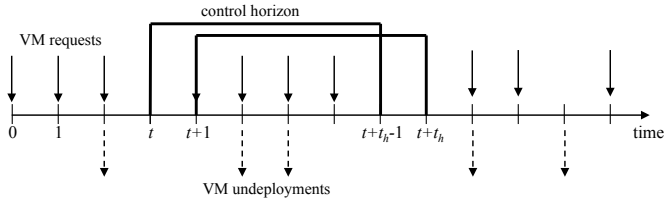

Fig. 2. Rolling horizon of the MPC for VM placement.

over a given PM. As done for the constraints introduced in Section IV-B, also (12) can be rewritten in compact form after introducing a proper 0-1 matrix $C \in \mathbb{R}^{M \times 6MN}$ and a vector $\underline{1}_M \in \mathbb{R}^M$ whose components are all equal to 1, i.e.,

$$C\underline{x}_t \leq \underline{1}_M. \tag{13}$$

A. Definition of the Optimal Control Problem

As hinted, the previously-discussed goals 1)-4) are conflicting. Specifically, a user may desire that the owner of the datacenter increases the "quality" characterizing the delivered service. This is achieved through goals 1), 2), and 4). Typically, this is also desirable from the perspective of the owner of the datacenter. However, he/she may also want to minimize operational costs by pursuing 3). Summarizing, 1)-4) concur to define the global performances and need a proper tradeoff. Recalling that 4) is a hard constraint enforced by (13), we search for the tradeoff by defining a suitable cost function to be maximized accounting for 1), 2), and 3).

According to the rolling horizon paradigm of MPC, the cost function to be maximized for all $t = 0, 1, \ldots, T-1$ is the following:

$$J_t \triangleq \sum_{\tau=t}^{t+t_h-1} e^{t-\tau}\Big(\underbrace{c_1\,\rho(\tau)}_{\text{churn}} + \underbrace{c_2\,\eta(\tau)}_{\text{interference}} + \underbrace{c_3\,\omega(\tau)}_{\text{power savings}} \Big) \tag{14}$$

where $c_1 > 0$, $c_2 > 0$, and $c_3 > 0$ are weight constants, and t_h is a given time horizon. The exponential term is a discount factor penalizing instants that are far from the current time t.

The cost function (14) is composed of three different terms, each one considering the goals 1), 2), and 3) described above. In more detail, the first term accounts for the reduction of the effect of churn by avoiding that VMs are deployed on unreliable PMs. The second term mitigates the performance losses due to colocation interferences caused by aggressive packings. Lastly, the third term accounts for power savings since it maximizes the number of PMs that can be put in sleep. We point out that the cost function J_t spans over a time window of length t_h starting from the current time t. Within the time interval $[t, t+t_h-1]$, we suppose to have suitable predictions of the workload offered to the datacenter in terms of new VM requests. To this aim, several methods have been proposed to forecast workloads, such as [54] and [55]. However, the detailed investigation of workload prediction is outside the scope of this work.

Notice that, in real-world scenarios, reducing the effects of churn and colocation interference could be more important than power savings. In fact, they may heavily influence the service level agreement subscribed by users. To avoid too many violations causing reputation or revenue losses to the service provider, a certain minimum "quality" should be guaranteed. This can be done in a "soft" manner by suitable choices of the coefficients c_1, c_2, and c_3 in (14). However, this may be insufficient, thus we introduce another "hard" constraint to be considered along with (7)–(10) and (13) for $t = 0, 1, \ldots, T$:

$$\rho(t)\eta(t) \geq \Xi \tag{15}$$

where $\Xi \in [0, 1]$ is the considered minimum quality level.

According to the MPC paradigm, at each time $t = 0, 1, \ldots, T$ we have to solve the following finite-horizon optimal control problem:

$$\max_{\underline{u}_{t,t+t_h-1}} J_t \tag{16}$$

subject to the constraints (2), (7), (8), (9), (10), (13), and (15) from time t to $t+t_h-1$. The unknowns are the components of the vector $\underline{u}_{t,t+t_h-1} \triangleq \mathrm{col}(\underline{u}_t, \ldots, \underline{u}_{t+t_h-1})$. Once their optimal value $\underline{u}^*_{t,t+t_h-1}$ has been computed, only the first action $\underline{u}^*_t$ is applied. The procedure is iterated for the other time steps, with a one-step-forward shift of the time window $[t, t+t_h-1]$. Figure 2 depicts a sketch of the MPC approach, where the horizons of the optimal control problems at times t and $t+1$ are reported.

Problem (16) is of nonlinear integer programming type with non-smooth functions. The number of unknowns is given by $2MNt_h$, i.e., it is proportional to the dimension of the datacenter and the length of the control horizon. Since finding an exact solution is difficult, we exploit an approximate solution method based on random sampling [56, chap. 3]. The pseudo-code of the adopted optimization procedure is reported in Algorithm 1. In particular, at each time t, Z different placement strategies are evaluated, and the corresponding values of the cost J_t are computed. The best placement is the one maximizing the cost. The z-th placement strategy, $z = 1, \ldots, Z$, is computed as follows. For each new VM to be deployed, different placement configurations are randomly generated. The process is iterated until constraints are satisfied. To avoid the generation of too many tentative placements, if constraints are not yet satisfied after K trials, a "fallback" placement is performed by using suitable heuristics that allows to obtain a placement satisfying constraints without further random trials. Clearly, the results provided by the optimization procedure reported in Algorithm 1 may be rough, especially for small values of Z and K. However, it enables to compute optimal placement strategies in a reduced amount of time. This allows to satisfy real-time requirements also for large-scale datacenters and to implement our holistic placement approach by using commodity hardware.

VI. Performance Metrics

In this section, we define the indexes used to evaluate the effectiveness of the proposed placement mechanism. First, we need to quantify how the churn of PMs and the colocation interference caused by competing VMs impact on the quality perceived by users. Unfortunately, quantifying the expectations

Algorithm 1 Solution of the MPC placement problem

// Inputs:
$t \leftarrow$ current time instant
$t_h \leftarrow$ control horizon
$L(\tau) \leftarrow$ number of new VMs for $\tau = t, \ldots, t+t_h-1$
$Z \leftarrow$ number of cost evaluations
$K \leftarrow$ number of placement configurations
// Main loops:
for z **from** 1 **to** Z **do**
  // Loop over the time horizon:
  for τ **from** t **to** $t + t_h - 1$ **do**
    // Loop over the VMs to be placed:
    for l **from** 1 **to** $L(\tau)$ **do**
      $p \leftarrow 0$
      $k \leftarrow 0$
      while $p = 0$ **and** $k \leq K$ **do**
        $k \leftarrow k + 1$
        perform a random placement
        if constraints are satisfied **then**
          $p \leftarrow 1$
        end if
        if $k = K$ **and** $p = 0$ **then**
          perform a fallback placement
        end if
      end while
    end for
  end for
  compute the z-th cost $J^{(z)}$
end for
$\underline{u}^{(z)*} \leftarrow \arg\min J^{(z)}$
$\left(\underline{u}_t^*, \ldots, \underline{u}_{t+t_h-1}^*\right) \leftarrow \underline{u}^{(z)*}$
// Outputs:
return $\underline{u}_t^*$

of users strongly depends on the specific application. As an example, a cloud offering multimedia streaming services should be able to sustain real-time data delivery, whereas for computation-intensive applications it should guarantee a steady processing flow. For the case of the IaaS services considered in this paper, defining specific Quality of Experience (QoE) metrics is still an open research problem [57]. Among others, an interesting approach is based on the adherence to the service level agreement [58]. Hence, we define for $t = 0, 1, \ldots, T$ a measure of QoE as follows:

$$\mathrm{QoE}(t) \triangleq \frac{\sum_{i=1}^{M} \left[\sum_{j=1}^{N} \sum_{k=c,d,n} x_{ij}^{k}(t)\zeta_{ij}(t)\right] \rho_i(t)}{\sum_{i=1}^{M} \sum_{j=1}^{N} \sum_{k=c,d,n} x_{ij}^{k}(t)}. \quad (17)$$

In essence, the QoE is defined as the ratio between the amount of resources made available to users and the overall number of resources that are requested. In fact, the numerator in (17) represents the reduction of resources due to churn and colocation interference. This is taken into account by the terms $\rho_i(t)$ and $\zeta_{ij}(t)$, respectively.

To condense the temporal evolution of the datacenter, we consider the average over time of the QoE in (17), i.e.,

$$\overline{\mathrm{QoE}} = \frac{1}{T} \sum_{t=0}^{T} \mathrm{QoE}(t)$$

which indicates that the expectation of the user is maximized if the datacenter provides the requested amount of resources for the longest timeframe.

The power used by the datacenter is another important metric. Along the lines of [10], [43], [59], and [60], we assume that the power required by a PM is given by the superposition of a constant contribution and a term that is proportional to the number of deployed VMs. As a consequence, the power required by the i-th PM running $n_i(t) \triangleq \sum_{i=1}^{M} \chi\left(a_{ij}(t)\right)$ VMs at time t, $i = 1, \ldots, M$, $t = 0, 1, \ldots, T$, is given by:

$$p_i(t) \triangleq P_{\max}\left(0.7 + \frac{0.3}{N}\, n_i(t)\right)$$

where $P_{\max}$ is the power when the server is loaded at its full capacity, i.e., N VMs are deployed. Without loss of generality, we also assume that a server in sleep consumes a negligible amount of power, i.e., $p_i(t) = 0$ [10], [43], [59], [60]. Therefore, the overall consumption of the datacenter at time t is:

$$P(t) \triangleq \sum_{i=1}^{M} p_i(t). \quad (18)$$

Also in this case, to condense the temporal evolution of (18), we consider the average over time:

$$\overline{P} \triangleq \frac{1}{T} \sum_{t=0}^{T} P(t)$$

which represents the mean power used by the datacenter to provide the IaaS cloud service.

As said, the MPC approach should also be able to provide a proper tradeoff between the delivered QoE and the power consumption. This is an indicator of the QoE-cost relation useful for engineering successful cloud applications and datacenters [61]. To this aim, a possible metric to be considered is how a variation of the QoE impacts on the power required by the datacenter and viceversa (see [62] for a use case addressing wireless networks). Hence, the metrics defined in (17) and (18) can be merged in a unique indicator:

$$D(t) \triangleq \frac{\mathrm{QoE}(t)}{P(t)}.$$

In more detail, $D(t)$ measures the "density" of quality in terms of required power, and it allows to estimate the tradeoff between the conflicting objectives of users and owners of the datacenter. To assess the impact on the overall time horizon, we consider the average over time as follows:

$$\overline{D} \triangleq \frac{\overline{\mathrm{QoE}}}{\overline{P}}.$$

Lastly, let us introduce a Key Performance Index (KPI), which assess how the proposed MPC placement technique addresses the performance goals 1), 2), and 3) as a whole. To this aim, the KPI considers each component of the cost

function (14) from $t = 0$ to $t = T$, i.e.,

$$\text{KPI} \triangleq \sum_{t=0}^{T} c_1\rho(t) + c_2\eta(t) + c_3\omega(t) \tag{19}$$

where c_1, c_2, and c_3 are the same coefficients of (14).

VII. Simulation Results

In this section, we evaluate the performances of our holistic placement technique by means of numerical simulations, performed in Matlab[1] on a 2.5 GHz Intel Xeon PC with 16 GB of RAM.

We focused on the same setting considered in [10], composed of $M = 500$ PMs, each one able to host up to $N = 5$ VMs. Considering 500 servers does not represent a limitation since large infrastructures are usually partitioned into smaller subsets. This limits the burden needed to manage the datacenter and simplifies the computation of the VM-to-PM map. Besides, such design choice enforces fault tolerance, scalability, and locality of network traffic [7], [10], [49]. Therefore, our setting can be considered either a small computing infrastructure or a portion of a larger datacenter.

As usually done in the literature, the different classes of VMs offered by the cloud infrastructure were modeled by considering the following aspects [10], [43], [48], [50], [59]: (i) VMs primarily use only one type of resource among CPU, disk, and network; (ii) VMs are often available according to two different usage plans, i.e., small and large. Therefore, we considered an IaaS cloud with 7 classes of VMs, i.e., $S = 6$. We recall that the dummy class 7 has been introduced to model non-deployed VMs, thus the percentages of resource utilization of this class are equal to 0. Table II reports the usages of resources for each class of VMs considered in our simulations.

To account for colocation interference, we used the interference matrix Θ defined in Section V. In particular, by using measurements provided by [6], we modeled the following aspects characterizing virtualization technologies: (i) the degradation of the performances of the CPU are mitigated by efficient scheduling disciplines, (ii) aggressive contentions on the I/O (e.g., to access disk or network resources) cause major performance degradations, (iii) the more a given resource is used by different VMs, the larger is the interference. To this end, Table II also reports the classes of VMs with the components of the considered interference matrix Θ. Clearly, the dummy class 7 does not account for any interference (i.e., $\theta_{k7} = 1$ for all k). We point out that the quantities provided in Table II also consider the impact of the virtualization on the available resources in terms of overhead. Besides, instead of quantizing the CPU in execution cores we used a "continuous" representation to avoid burdening the model [10], [43], [50].

The simulation horizon T was chosen equal to 168 steps, equivalent to 1 week with a sampling time of 1 hour. As a consequence, we considered as negligible all the behaviors characterizing IaaS applications evolving in the range between few seconds to minutes. For instance, this is the case of the time needed by a VM to start or a PM to reboot [10], [63].

[1]The code used for the simulations is available online at `www.diptem.unige.it/gaggero/software.html`.

TABLE II
Interference matrix Θ and resource usages of the VM classes.

class	1	2	3	4	5	6	7	CPU	disk	net
1	0.9	0.8	1.0	1.0	1.0	1.0	1.0	0.23	0.05	0.05
2	0.8	0.7	1.0	1.0	1.0	1.0	1.0	0.46	0.05	0.05
3	1.0	1.0	0.7	0.6	1.0	1.0	1.0	0.05	0.20	0.05
4	1.0	1.0	0.6	0.5	1.0	1.0	1.0	0.05	0.40	0.05
5	1.0	1.0	1.0	1.0	0.7	0.6	1.0	0.05	0.05	0.17
6	1.0	1.0	1.0	1.0	0.6	0.5	1.0	0.05	0.05	0.34
7	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0	0	0

The power $P_{\max}$ consumed by a PM when loaded at full capacity was chosen equal to 250 W [10], [59], [60]. As regards the probability $g_i(t)$ of having a PM correctly running, to the best of our knowledge, the literature lacks of a common, "one-fits-all" model to holistically consider the different technological and physical causes of outages (e.g., RAM issues, disk malfunctions, overheats, human or configuration errors) as well as rejuvenation and other software issues [64]. Therefore, without loss of generality, we considered servers having a $g_i(t)$ drawn from a uniform distribution in the range [0.8, 1].

Concerning the workload, we considered three different scenarios, each one capturing a specific behavior of IaaS frameworks. In more detail, Scenarios A and B use synthetic workloads built to reflect realistic use cases, whereas Scenario C uses traces captured in a real datacenter. Along the lines of [10], we considered:

1) Scenario A: it represents a datacenter receiving a "steady" flow of VM requests with a nearly constant mean over time. At each time t, the number of requested VMs for a given class was taken from uniform distributions. Specifically, we considered the ranges [0,12] for type-1 and type-2 VMs, [0, 20] for type-3 and type-4 VMs, and [0, 25] for type-5 and type-6 VMs. The portion of VMs characterized by strict security requirements was taken equal to 15%.

2) Scenario B: it considers a peak of VM requests triggered by certain events (e.g., the release of a software update). At each time t, the number of requested VMs for a given class was taken from uniform distributions in the ranges [0, 19] for type-1 and type-2 VMs, [0, 22] for type-3 and type-4 VMs, and [0, 24] for type-5 and type-6 VMs, with a positive peak from $t = 40$ to $t = 120$. The portion of VMs characterized by strict security requirements was taken equal to 15%.

3) Scenario C: it is based on the workload of a real datacenter properly scaled to match the size of our simulated environment. It spans over a week of usage to capture oscillations occurring within the single day (e.g., due to work shifts). The requests of the various types of VMs as well as those with specific security criteria were obtained by using several slices of the available traces.

For each VM, we considered a lifespan (i.e., the time for which it is requested) in the range [5, 20] hours. Moreover, all the random numbers characterizing the workload were

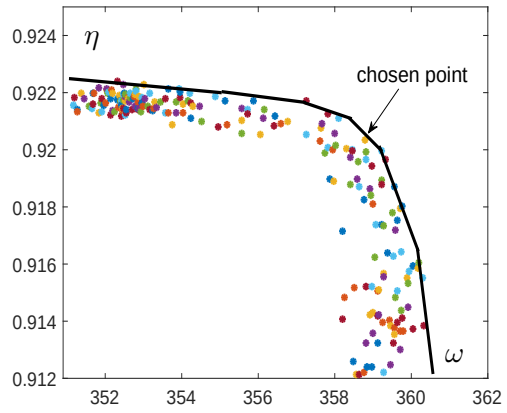

Fig. 3. Pareto frontier for the terms η and ω of the cost obtained in Scenario A with $t_h = 3$.

rounded to the nearest integer value. Figure 4 depicts the VM workloads at the various time instants in the three scenarios.

To assess a variety of realistic use cases, we considered datacenters having different hardware equipment. Specifically, Scenario A is composed of identical PMs, i.e., $b_i^k = 1.0$ for all $i = 1, \ldots, M$ and $k = c, d, n$. Instead, Scenarios B and C account for a heterogeneous deployment characterized by PMs with different computing capabilities, but similar storage and networking capacities, as also done in [10]. In more detail, we fixed $b_i^c = 0.8$ for $i = 1, \ldots, 166$, $b_i^c = 0.9$ for $i = 167, \ldots, 333$, $b_i^c = 1.0$ for $i = 334, \ldots, 500$, $b_i^d = 1.0$ for $i = 1, \ldots, 500$, and $b_i^n = 1.0$ for $i = 1, \ldots, 500$.

Recalling that the QoE is important to qualify a datacenter in terms of user satisfaction, the minimum quality level Ξ in constraint (15) was chosen equal to 0.5. This has not to be regarded as an absolute value. Rather, it indicates how the datacenter is near to a "production quality" threshold, e.g., the five nines. The coefficients c_1, c_2, and c_3 of the cost function (14) were chosen equal to 10, 750, and 0.5, respectively. Figure 3 depicts the results of the Pareto analysis used to select the coefficients c_2 and c_3 in Scenario A. Similar plots could be displayed for the other combinations of weights and scenarios, but they are omitted for the sake of brevity.

To evaluate the performances of the MPC approach, we considered different values for the control horizon t_h, i.e., we chose 1, 3, and 5. To have a vis-à-vis comparison, we considered the *first fit*, *dotproduct*, and *norm2* placement policies based on bin packing, widely discussed in the literature and used to manage real-world datacenters [65] and suitably adapted to our purposes. The first fit method places VMs in the first available PMs. This can happen if and only if the PM has sufficient free resources to satisfy the new requests. Owing to its simplicity and computational efficiency, the first fit was chosen also to implement the fallback placement procedure for the MPC approach (see line 22 of Algorithm 1). The first fit allows to pack VMs in a reduced set of PMs, but at the price of possible wastage of resources due to colocation interference. For this reason, the dotproduct and norm2 can be considered a sort of refinements, as they provide more "resource-aware" placement mechanisms. Indeed, in both cases the packing of VMs is reduced since the number of VMs of the same class placed over a PM is limited to mitigate colocation interference.

TABLE III
AVERAGE NUMBER DISCARDED PLACEMENT CONFIGURATIONS PER NEW VM TO BE DEPLOYED.

	t_h=1	t_h=3	t_h=5
Scenario A	0.017	0.052	0.090
Scenario B	0.017	0.052	0.089
Scenario C	0.016	0.051	0.088

TABLE IV
AVERAGE NUMBER OF FALLBACK PLACEMENTS PER NEW VM TO BE DEPLOYED.

	t_h=1	t_h=3	t_h=5
Scenario A	0.000082	0.00025	0.00044
Scenario B	0.000084	0.00025	0.00044
Scenario C	0.000078	0.00024	0.00042

The optimization method described in Algorithm 1 to solve the MPC placement problem (16) was applied with $Z = 100$ and $K = 10$. These values were selected through a trial and error procedure, with the goal of reducing the number of placement attempts and the need of resorting to the fallback procedure. To this end, Table III reports the average number of "discarded" placement configurations per new VM to be deployed. In other words, it contains how many configurations did not satisfy the constraints. This corresponds to the number of iterations (apart from the first one) of the `while` loop at line 15 of Algorithm 1. Table IV showcases the average number of fallback placements per new VM to be deployed. It turns out that the numbers of "discarded" configurations and fallback placements are minimal and almost insensitive to the scenario. Their slight increase with the control horizon t_h is due to the need of satisfying a larger number of constraints as t_h grows, which requires more attempts on the average.

A. Numerical Results

Table V reports the numerical results for the performance indexes considering average values as described in Section VI. Specifically, it contains the values provided by the MPC, first fit, dotproduct, and norm2 approaches for all the considered scenarios. For the MPC, the results corresponding to the three chosen control horizons are showcased. To quantify the computational requirements of the various placement strategies, Table V also contains the index $\overline{T}$, which is the mean time required to compute a single VM-to-PM map. Figure 5 sketches the temporal evolutions of QoE, power consumption, and their relation in terms of density. The considered horizon for the MPC is $t_h = 5$.

As shown, the proposed MPC approach always outperforms the first fit, dotproduct, and norm2 heuristics in the considered scenarios for all the performance indexes. Concerning the QoE, the extreme packings enforced by the first fit do not correspond to better performances, as it ignores how the VMs are mixed on the single PM. Instead, the dotproduct, norm2,

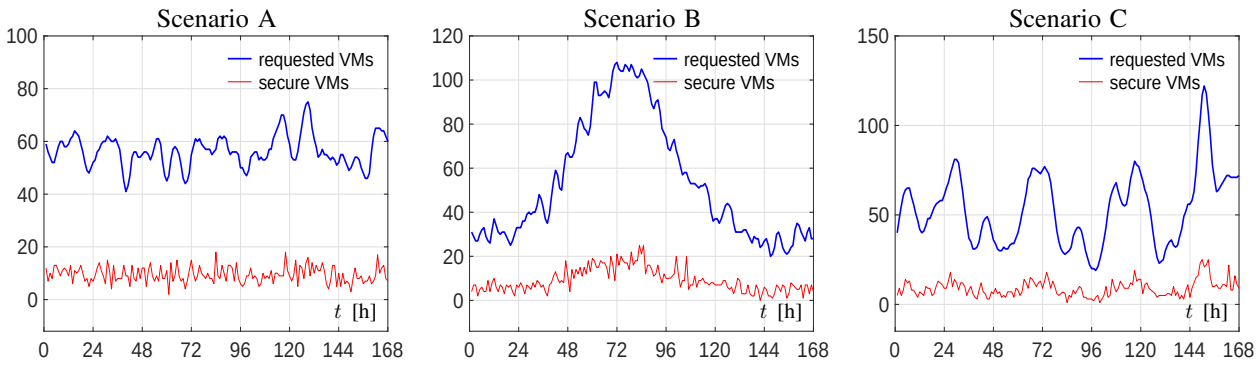

Fig. 4. Number of new requested VMs: the blue line is the overall workload, while the red one is the fraction of VMs with security requirements.

TABLE V
SUMMARY OF THE NUMERICAL RESULTS.

		MPC			first fit	dotproduct	norm2
		t_h=1	t_h=3	t_h=5			
Scenario A	$\overline{\text{QoE}}$	0.67	0.67	0.67	0.58	0.65	0.66
	$\overline{P}$ [kW]	34.6	34.6	34.3	39.2	34.7	34.6
	$\overline{D}$ [kW^{-1}]	194.1	194.4	195.4	149.7	189.2	191.1
	KPI $(\times 10^2)$	1471.3	1472.1	1473.5	1418.1	1456.4	1458.2
	$\overline{T}$ [s]	0.75	0.86	0.96	0.01	0.06	0.06
Scenario B	$\overline{\text{QoE}}$	0.69	0.69	0.69	0.63	0.67	0.69
	$\overline{P}$ [kW]	32.5	32.5	32.2	37.2	35.0	35.1
	$\overline{D}$ [kW^{-1}]	213.9	214.0	215.4	171.0	192.9	192.4
	KPI $(\times 10^2)$	1497.0	1497.3	1498.1	1450.0	1469.8	1469.6
	$\overline{T}$ [s]	0.71	0.86	0.99	0.01	0.06	0.06
Scenario C	$\overline{\text{QoE}}$	0.67	0.67	0.67	0.61	0.64	0.64
	$\overline{P}$ [kW]	32.5	32.3	32.2	36.4	34.4	34.5
	$\overline{D}$ [kW^{-1}]	208.7	209.5	210.2	168.3	187.6	187.5
	KPI $(\times 10^2)$	1491.0	1492.0	1492.5	1444.7	1463.5	1463.2
	$\overline{T}$ [s]	0.72	0.86	0.97	0.01	0.06	0.06

and MPC exploit a "multidimensional" information, i.e., they account for the interference among the different VM classes, thus providing higher values for the QoE. As regards the overall power consumed by the datacenter, it is worth noting that the MPC allows greater savings if compared with the other approaches. However, the energetic footprint should not be considered decoupled from the quality delivered to users. To this end, the behavior of the "density" of quality with respect to the required power confirms the superiority of the MPC to provide the best tradeoff. This is also evident from the values of the KPI. In fact, the behaviors of the KPI for the MPC with all the considered prediction horizons t_h are better than those obtained by using the first fit, dotproduct, and norm2. In general, the performance indicators slightly improve with the growth of t_h, as a greater amount of future information is taken into account.

Concerning the computational burden, the MPC is more demanding than the considered bin packing heuristics since it requires the solution of an optimization problem. However, the values of $\overline{T}$ required by the MPC for computing the optimal placement strategies are always less than 1 second (i.e., up to 0.99 s for Scenario B and $t_h = 5$, as shown in Table V). Such a timeframe is negligible compared to the dynamic of an IaaS application, which often evolves in hours or days. As a consequence, it turns out that the MPC approach is well-suited to being used as an effective placement mechanism for frameworks operating on line.

VIII. CONCLUSIONS

In this paper, we have presented a holistic approach for the placement of VMs in cloud datacenters. In more detail, we have proposed a framework to choose how to deploy VMs on PMs by pursuing conflicting performance goals, such as counteract to hardware outages or software aging issues, ensure proper security policies, maintain a suitable service level, and reduce power requirements. To compute the optimal strategies, we have developed a method based on MPC, which has allowed to take into account constraints while exploiting future information. Results have indicated that our method outperforms classic heuristics used in real-world datacenters in all the considered scenarios.

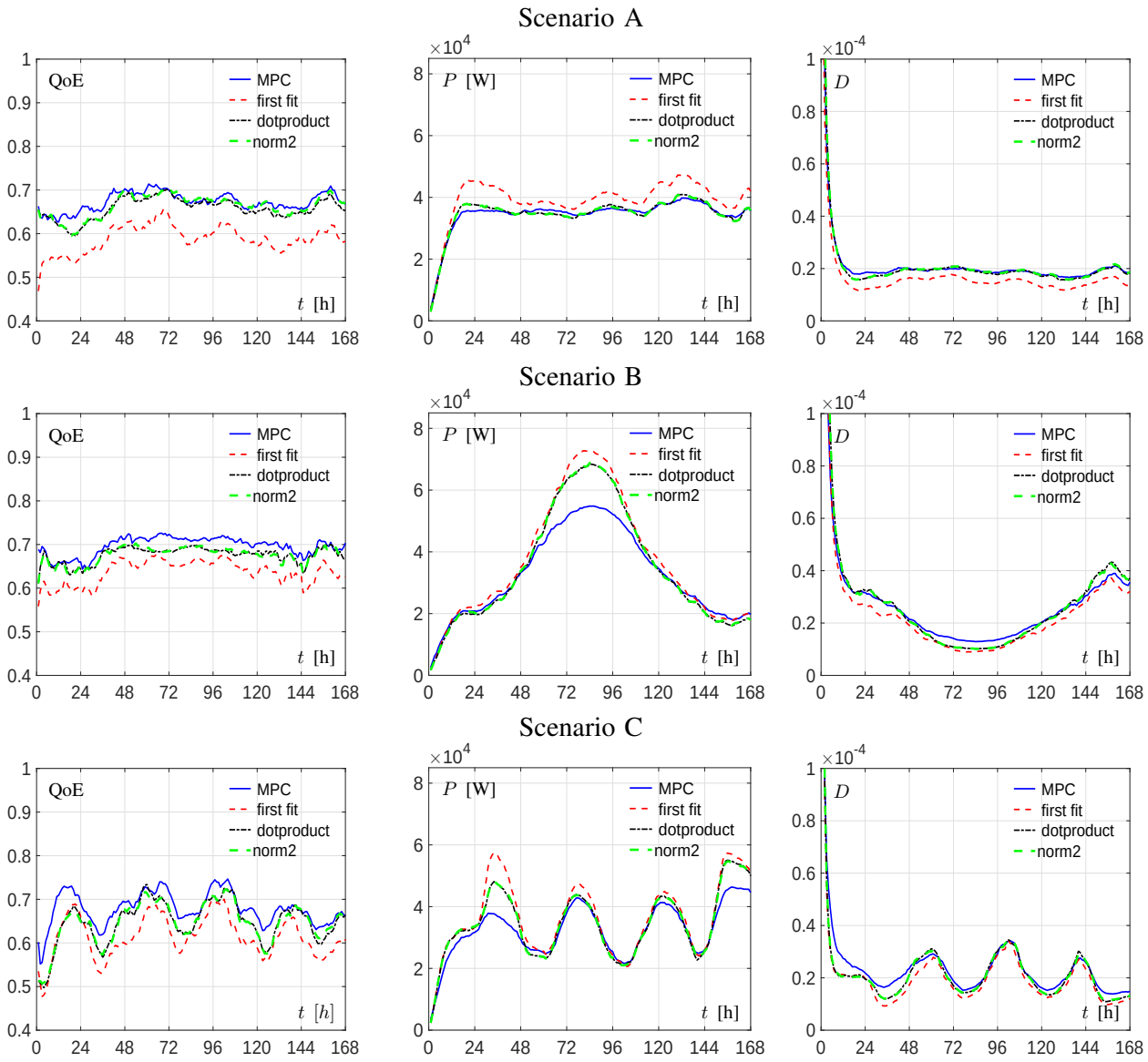

Fig. 5. QoE, power consumption, and QoE-power density provided by the first fit, dotproduct, norm2, and MPC with $t_h = 5$.

Part of ongoing research aims at merging the MPC placement framework with other tools for managing resources in cloud datacenters. Specifically, we are working towards its integration with consolidation techniques. Future works will focus on the evaluation of alternative, faster optimization methods for solving the MPC placement problem. Other subjects of future investigations are the creation of more fine-grained models, for instance to consider the RAM used by VMs, as well as how the MPC approach can be "ported" on emerging scenarios considering user mobility or geographically-sparse infrastructures.

ACKNOWLEDGMENTS

This work has been partially supported by the research project "GESTEC - Tecnologie orientate ai servizi per lo sviluppo e per l'integrazione di piattaforme ICT", funded by the Italian Ministry of University and Research.

Many thanks also to Netalia (http://www.netalia.it) for the access to its cloud for the generation of the workload used in the simulations.

REFERENCES

[1] M. Pearce, S. Zeadally, and R. Hunt, "Virtualization: Issues, security threats, and solutions," *ACM Comput. Surveys*, vol. 45, no. 2, pp. 1–39, 2013.

[2] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia, "A survey on virtual machine migration and server consolidation frameworks for cloud data centers," *J. Network and Computer Applications*, vol. 52, pp. 11–25, 2015.

[3] H. Liu, H. Jin, C. Xu, and X. Liao, "Performance and energy modeling for live migration of virtual machines," *Cluster Computing*, vol. 16, no. 2, pp. 249–264, 2013.

[4] D. Cotroneo, R. Natella, R. Pietrantuono, and S. Russo, "A survey of software aging and rejuvenation studies," *ACM J. Emerging Technol. in Comput. Systems*, vol. 10, no. 1, p. 8, 2014.

[5] F. Machida, J. Xiang, K. Tadano, and Y. Maeno, "Combined server rejuvenation in a virtualized data center," in *Int. Conf. Ubiquitous Intell. & Comput. and Int. Conf. Autonomic & Trusted Comput.*, 2012, pp. 486–493.

[6] F. Xu, F. Liu, L. Liu, H. Jin, B. Li, and B. Li, "iAware: Making live migration of virtual machines interference-aware in the cloud," *IEEE Trans. Comput.*, vol. 63, no. 12, pp. 3012–3025, 2014.

[7] J. Sugerman, G. Venkitachalam, and B. Lim, "Virtualizing I/O devices on VMware workstation's hosted virtual machine monitor," in *Proc. USENIX Annual Technical Conf.*, 2001, pp. 1–14.

[8] E. Caron and J. R. Cornabas, "Improving users' isolation in IaaS: Virtual machine placement with security constraints," in *Int. Conf. Cloud Comput.*, 2014, pp. 64–71.

[9] Z. Qi, C. Lu, and B. Raouf, "Cloud computing: state-of-the-art and research challenges," *J. Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[10] M. Gaggero and L. Caviglione, "Predictive control for energy-aware consolidation in cloud datacenters," *IEEE Trans. Contr. Syst. Technol.*, vol. 24, no. 2, pp. 461–474, 2016.

[11] Y. Xia, M. Zhou, X. Luo, Q. Zhu, J. Li, and Y. Huang, "Stochastic modeling and quality evaluation of infrastructure-as-a-service clouds," *IEEE Trans. Autom. Science Eng.*, vol. 12, no. 1, pp. 162–170, 2015.

[12] M. Gaggero and L. Caviglione, "Model predictive control for the placement of virtual machines in cloud computing applications," in *Proc. American Control Conf.*, 2016, pp. 1987–1992.

[13] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[14] P. Cortes, J. Rodriguez, P. Antoniewicz, and M. Kazmierkowski, "Direct power control of an AFE using predictive control," *IEEE Trans. Power Electronics*, vol. 23, no. 5, pp. 2516–2523, 2008.

[15] A. Alessandri, M. Gaggero, and F. Tonelli, "Min-max and predictive control for the management of distribution in supply chains," *IEEE Trans. Contr. Syst. Technol.*, vol. 19, no. 5, pp. 1075–1089, 2011.

[16] M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," *J. Network and Computer Applications*, vol. 66, pp. 106–127, 2016.

[17] M. Xu, W. Tian, and R. Buyya, "A survey on load balancing algorithms for virtual machines placement in cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 12, 2017.

[18] T. Kaur and I. Chana, "Energy efficiency techniques in cloud computing: A survey and taxonomy," *ACM Comput. Surveys*, vol. 48, no. 2, p. 22, 2015.

[19] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whally, and E. Snible, "Improving performance and availability of services hosted on iaas clouds with structural constraint-aware virtual machine placement," in *Int. Conf. Services Comput.*, 2011, pp. 72–79.

[20] Y. Wang and Y. Xia, "Energy optimal vm placement in the cloud," in *Int. Conf. Cloud Comput.*, 2016, pp. 84–91.

[21] L. A. Rocha and E. Cardozo, "A hybrid optimization model for green cloud computing," in *Int. Conf. Utility and Cloud Comput.*, 2014, pp. 11–20.

[22] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *J. Computer and System Sciences*, vol. 79, no. 8, pp. 1230–1242, 2013.

[23] J.-k. Dong, H.-b. Wang, Y.-y. Li, and S.-d. Cheng, "Virtual machine placement optimizing to improve network performance in cloud data centers," *The J. China Universities of Posts and Telecommunications*, vol. 21, no. 3, pp. 62–70, 2014.

[24] K. Su, L. Xu, C. Chen, W. Chen, and Z. Wang, "Affinity and conflict-aware placement of virtual machines in heterogeneous data centers," in *Int. Symp. Autonomous Decentralized Systems*, 2015, pp. 289–294.

[25] M. Mishra and U. Bellur, "Whither tightness of packing? the case for stable vm placement," *IEEE Trans. Cloud Comput.*, vol. 4, no. 4, pp. 481–494, 2016.

[26] F. Machida, M. Kawato, and Y. Maeno, "Redundant virtual machine placement for fault-tolerant consolidated server clusters," in *Network Operations and Management Symposium*, 2010, pp. 32–39.

[27] A. Zhou, S. Wang, B. Cheng, Z. Zheng, F. Yang, R. Chang, M. Lyu, and R. Buyya, "Cloud service reliability enhancement via virtual machine placement optimization," *IEEE Trans. Services Comput.*, 2016.

[28] D. Espling, L. Larsson, W. Li, J. Tordsson, and E. Elmroth, "Modeling and placement of cloud services with internal structure," *IEEE Trans. Cloud Comput.*, vol. 4, no. 4, pp. 429–439, 2016.

[29] L. Zhao, L. Lu, Z. Jin, and C. Yu, "Online virtual machine placement for increasing cloud provider?s revenue," *IEEE Trans. Services Comput.*, vol. 10, no. 2, pp. 273–285, 2017.

[30] X. Xu and M. Tang, "A new approach to the cloud-based heterogeneous mapreduce placement problem," *IEEE Trans. Services Comput.*, vol. 9, no. 6, pp. 862–871, 2016.

[31] L. Chen, H. Shen, and S. Platt, "Cache contention aware virtual machine placement and migration in cloud datacenters," in *Int. Conf. Network Protocols*, 2016, pp. 1–10.

[32] J. Chen, Q. He, D. Ye, W. Chen, Y. Xiang, K. Chiew, and L. Zhu, "Joint affinity aware grouping and virtual machine placement," *Microprocessors and Microsystems*, vol. 52, pp. 365–380, 2017.

[33] M. Rahman and P. Graham, "Compatibility-based static vm placement minimizing interference," *J. Network and Computer Applications*, vol. 84, pp. 68–81, 2017.

[34] R. Jhawar, V. Piuri, and P. Samarati, "Supporting security requirements for resource management in cloud computing," in *Int. Conf. Computational Science and Engineering*, 2012, pp. 170–177.

[35] Y. Han, J. Chan, T. Alpcan, and C. Leckie, "Using virtual machine allocation policies to defend against co-resident attacks in cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 1, pp. 95–108, 2017.

[36] J. Xu and J. A. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Proc. Int. Conf. Green Comput. and Communications & Int. Conf. Cyber, Physical and Social Comput.*, 2010, pp. 179–188.

[37] Z. Usmani and S. Singh, "A survey of virtual machine placement techniques in a cloud data center," *Procedia Computer Science*, vol. 78, pp. 491–498, 2016.

[38] F. Caglar, S. Shekhar, and A. Gokhale, "iplace: An intelligent and tunable power-and performance-aware virtual machine placement technique for cloud-based real-time applications," in *Int. Symp. Object/Component/Service-Oriented Real-Time Distributed Comput.* IEEE, 2014, pp. 48–55.

[39] C. Gao, H. Wang, L. Zhai, Y. Gao, and S. Yi, "An energy-aware ant colony algorithm for network-aware virtual machine placement in cloud computing," in *Int. Conf. Parallel and Distributed Systems*, 2016, pp. 669–676.

[40] A. Pahlevan, P. G. Del Valle, and D. Atienza, "Exploiting cpu-load and data correlations in multi-objective vm placement for geo-distributed data centers," in *Design, Automation & Test in Europe Conf. & Exhibition*, 2016, pp. 1333–1338.

[41] F. Alharbi, Y. C. Tain, M. Tang, and T. K. Sarker, "Profile-based static virtual machine placement for energy-efficient data center," in *Int. Conf. High Performance Comput. and Communications; Int. Conf. Smart City; Int. Conf. Data Science and Systems*, 2016, pp. 1045–1052.

[42] X. Li, Z. Qian, S. Lu, and J. Wu, "Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center," *Mathematical and Computer Modelling*, vol. 58, no. 5, pp. 1222–1235, 2013.

[43] M. Gaggero and L. Caviglione, "A predictive control approach for energy-aware consolidation of virtual machines in cloud computing," in *Proc. Conf. Decision and Control*, 2014, pp. 5308–5313.

[44] Q. Zhang, Q. Zhu, and R. Boutaba, "Dynamic resource allocation for spot markets in cloud computing environments," in *Int. Conf. Utility and Cloud Comput.*, 2011, pp. 178–185.

[45] G. Mencagli, "Adaptive model predictive control of autonomic distributed parallel computations with variable horizons and switching costs," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 7, pp. 2187–2212, 2016.

[46] J. Lango, "Toward software-defined SLAs," *Communications of the ACM*, vol. 57, no. 1, pp. 54–60, 2014.

[47] F. Ma, F. Liu, and Z. Liu, "Multi-objective optimization for initial virtual machine placement in cloud data center," *J. Information and Computational Science*, vol. 9, no. 16, 2012.

[48] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing SLA violations," in *Int. Symp. Integrated Network Management*, 2007, pp. 119–128.

[49] K. Mills, J. Filliben, and C. Dabrowski, "Comparing VM-placement algorithms for on-demand clouds," in *Proc. Int. Conf. Cloud Comput. Technol. and Science*, 2011, pp. 91–98.

[50] A. V. Papadopoulos and M. Maggio, "Virtual machine migration in cloud infrastructures: Problem formalization and policies proposal," in *Proc. Conf. Decision and Control*, 2015, pp. 6698–6705.

[51] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Proc. Conf. Power Aware Comput. and Systems*, 2008.

[52] B. Egger, Y. Cho, C. Jo, E. Park, and J. Lee, "Efficient checkpointing of live virtual machines," *IEEE Trans Computers*, vol. 65, no. 10, pp. 3041–3054, 2016.

[53] Y. Dong, W. Ye, Y. Jiang, I. Pratt, S. Ma, J. Li, , and H. Guan, "COLO: Coarse-grained lock-stepping virtual machines for non-stop service," in *Proc. Symposium Cloud Comput.*, 2013, pp. 1–16.

[54] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *Proc. Workshop Cloud Management*, 2012, pp. 1287–1294.
[55] I. Moreno, P. Garraghan, P. Townend, and J. Xu, "Analysis, modeling and simulation of workload patterns in a large-scale utility cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 208–221, 2014.
[56] K. T. Fang and Y. Wang, *Number-theoretic Methods in Statistics*. London: Chapman & Hall, 1994.
[57] T. Hobfeld, R. Schatz, M. Varela, and C. Timmerer, "Challenges of QoE management for cloud applications," *IEEE Comm. Mag.*, vol. 50, no. 4, 2012.
[58] S. Al-Shammari and A. Al-Yasiri, "Defining a metric for measuring QoE of SaaS cloud computing," *Proceedings of PGNET*, pp. 251–256, 2014.
[59] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster Comput.*, vol. 12, no. 1, pp. 1–15, 2009.
[60] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *Proc. Int. Workshop on Middleware for Grids, Clouds and e-Science*, 2010, pp. 1–4.
[61] J. He, Y. Wen, J. Huang, and D. Wu, "On the cost-QoE tradeoff for cloud-based video streaming under amazon EC2's pricing models," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 24, no. 4, pp. 669–680, 2014.
[62] X. Zhang, J. Zhang, Y. Huang, and W. Wang, "On the study of fundamental trade-offs between QoE and energy efficiency in wireless networks," *Trans. Emerging Telecommunications Technol.*, vol. 24, no. 3, pp. 259–265, 2013.
[63] A. V. Papadopoulos, A. Ali-Eldin, K.-E. Arzen, J. Tordsson, and E. Elmroth, "PEAS: A performance evaluation framework for auto-scaling strategies in cloud applications," *ACM Trans. Model. Perform. Eval. Comput. Syst*, vol. 1, no. 4, pp. 1–31, 2016.
[64] L. A. Barroso, J. Clidaras, and U. Hölzle, *The datacenter as a computer: An introduction to the design of warehouse-scale machines*. Morgan & Claypool Publishers, 2013.
[65] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder, "Heuristics for vector bin packing." Microsoft Research, 2011. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=147927

Mauro Gaggero received the M.Sc. degree in electronics engineering and the Ph.D. degree in mathematical engineering from the University of Genoa, Genoa, Italy, in 2005 and 2010, respectively. Since 2011, he has been a Research Scientist with the National Research Council of Italy, Genoa. His current research interests include control and optimization of nonlinear systems, distributed parameter systems, neural networks, and learning from data. Dr. Gaggero is an Associate Editor of the European Control Association Conference Editorial Board and of the IEEE Control Systems Society Conference Editorial Board.

Luca Caviglione received the Ph.D. degree in electronics and computer engineering from the University of Genoa, Genoa, Italy. He is currently a Research Scientist with the National Research Council of Italy, Genoa. He has been involved in research projects funded by ESA, EU, and MIUR. He is a work group leader of the Italian IPv6 Task Force, a contract professor, and a professional engineer. His current research interests include P2P systems, wireless communications, cloud architectures, and network security. Dr. Caviglione is involved in the technical program committee of international conferences and serves as reviewer for international journals.