Joint Minimization of the Energy Costs from Computing, Data Transmission, and Migrations in Cloud Data Centers

Claudia Canali*, Luca Chiaraviglio†‡, Riccardo Lancellotti*, Mohammad Shojafar‡

* Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia,

Email: {claudia.canali, riccardo.lancellotti}@unimore.it

† Department of Electronic Engineering, Univerisity of Rome Tor Vergata,

Email: luca.chiaraviglio@uniroma2.it

‡ Italian National Consortium for Telecommunications (CNIT),

Email: mshojafar@cnit.it

Abstract

We propose a novel model, called JCDME, for the allocation of Virtual Elements (VEs), with the goal of minimizing the energy consumption in a Software-Defined Cloud Data Center (SDDC). More in detail, we model the energy consumption by considering the computing costs of the VEs on the physical servers, the costs for migrating VEs across the servers, and the costs for transferring data between VEs. In addition, JCDME introduces a weight parameter to avoid an excessive number of VE migrations. Specifically, we propose three different strategies to solve the JCDME problem with an automatic and adaptive computation of the weight parameter for the VEs migration costs. We then evaluate the considered strategies over a set of scenarios, ranging from a small sized SDDC up to a medium sized SDDC composed of hundreds of VEs and hundreds of servers. Our results demonstrate that JCDME is able to save up to an additional 7% of energy w.r.t. previous algorithms, without a substantial increase in the solution complexity.

Index Terms

Cloud computing, Software-defined networks (SDNs), Software-defined cloud data centers (SDDCs), Energy consumption, Optimization models.

I. Introduction

Data center (DC) capacity is growing at a rapid clip to serve the vast needs of online businesses. In addition, according to different studies (e.g. [2]), DCs are consuming far more power than they need. As a result, the reduction of DC energy consumption is a goal of fundamental importance. Actually, most of the proposed solutions targeting

A preliminary version of this work appeared in [1].

energy consumption reduction in Infrastructure as a Service (IaaS) cloud DCs mainly focus on computing elements, and in particular on server consolidation [3], [4], [5], [6]. By adopting this approach, the number of turned on physical servers hosting the active Virtual Machines (VMs) is minimized. However, server consolidation typically does not consider the contribution of network traffic exchange to the overall energy consumption of the cloud infrastructure. In this context, solutions that are not network-aware are likely to cause sub-optimal VMs allocation. This is due to the fact that network elements tend to consume about 10%-20% of the energy of DC in normal usage, and may account for up to 50% energy during low loads [7]. In addition to this aspect, few models for VMs allocation consider the contribution of VMs migration to energy consumption, both in terms of computing and network costs. For example, the network-aware model for VMs allocation proposed in [8] re-computed from scratch VMs allocation for the whole DC every time the model is solved, without considering the costs of migrations. On the other hand, other models (e.g., [9]) take into account the number of migrations (and not their actual cost). More in depth, the authors of [9] propose an approach that involves the setting of different weights in order to trade between the minimization of the number of turned on physical servers and the VMs migrations. Although all these works aim to shed lights on the energy management problem in the DC, there is still a gap in precisely taking into account the costs derived from data transmission over the network and from VMs migrations.

In addition, the need of jointly managing computing and networking components will be even more evident in future architectures, which will extensively adopt virtualization techniques. To this aim, it is expected that future DCs will exploit different types of VMs, ranging from the ones devoted to classical computing tasks to elements running virtualized network functions and Software-Defined Networks (SDNs) nodes [10]. In such Software-Defined Cloud Data Centers (SDDCs), the capability of supporting a more flexible network reconfiguration allows the migration of both VMs and virtualized network devices, such as Virtual Routers (VRs) [11] (throughout the paper we will refer to VMs and VRs as to *Virtual Elements* – VEs). In this scenario, the exploitation of traditional VMs allocation policies taking into account only server consolidation will result in large energy inefficiencies.

In this context, several questions are arising, such as: How to model the contributions of computing and networking VEs to the SDDC energy consumption? How to minimize the energy consumption of an SDDC hosting VEs? The goal of this paper is to answer these questions. With respect to these questions, we provide multiple contributions. The first group of contributions concerns the proposal of an innovative problem model, namely *Joint Computing, Data Transmission and Migration Energy costs (JCDME)*, for the allocation of VEs with the goal of minimizing the total energy consumption of the SDDC. Unlike existing proposals, in our model a VE may be either a VM or a VR deployed on the physical servers of the cloud infrastructure. In addition, the proposed energy consumption model integrates: i) the costs for providing the computing functionalities of VEs on the physical servers, ii) the costs for transferring data across the VEs, and iii) the costs for migrating the VEs between the servers. We adopt a dynamic programming approach where the VEs placement at previous Time Slot (TS) is taken as the input to compute the allocation solution for the future TSs. Preliminary experiments show that it is important to take into account traffic variation over time among the VEs: in particular, if the traffic patterns across VEs are subject to not very frequent variations over time, the migration of a large number of VEs may prove convenient in the long period, even if the amount of energy consumed by the migration in the current TS is large. Hence, we introduce a

weight parameter to tune the benefit from optimal VE allocation against the overhead of migrations. This approach has been hinted at in literature [1], [9]. However, all these solutions assume a fixed value over time for the weight parameter. The second group of contributions goes two steps further from [1], [9] by: i) exploring the possibility of automatically adapting the weight parameter, and ii) proposing three different *adaptive* strategies to target this goal.

All the contributions of our paper are validated through experiments based on different scenarios and network traffic parameters. With respect to the model proposal, our results indicate that the overall energy-efficiency is improved when the allocation of VEs is performed by jointly considering the different contributions to energy consumption. Furthermore, we demonstrate that our adaptive method outperforms previous works in terms of energy efficiency by introducing a way to tune the weight parameter strategies. Finally, we validate our proposal in terms of scalability with respect to the number of VEs, demonstrating the viability of our approach for SDDC when hundred of VEs and hundreds of servers are considered.

The remainder of this paper is organized as follows. Sec. II overviews the considered SDDC architecture and its main components. A simple (yet representative) case study is analyzed in Sec. III to motivate our work. Sec. IV details the proposed optimal formulation, including our set of strategies for adaptively setting the weight parameter. Sec. V describes the considered scenario. The obtained results are detailed in Sec. VI. Sec. VII discusses the related work. Finally, Sec. VIII concludes the paper with some final remarks and outlines open research problems.

II. SDDC Architecture and Components

In this section, we first present the considered SDDC architecture. Then, we detail the main management components of this architecture, namely the network manager and the allocation manager. Finally, we discuss how the dynamic variation of traffic among the VEs may impact the allocation strategies in the considered SDDC architecture.

A. Software-Defined Data Center (SDDC) Architecture

Fig. 1 presents the general scheme of an SDDC. More in detail, we rely on an IaaS paradigm, where cloud customers can deploy single VMs or entire networks consisting of VMs and VRs on the cloud system. In this scenario, VEs can be dynamically added, moved and removed from the system: VMs and VRs can be deployed and destroyed at any time by cloud customers. In addition, active VMs and VRs may be migrated from one physical server to another one according to the DC management strategies. The physical servers hosting the VEs are grouped into *pods*, as shown in Fig. 1. Focusing then on the costs for transferring information among the VEs, the communication between VEs hosted on the same server does not have an impact on the energy consumption, while the data exchange between VEs located on different servers triggers the network energy cost. As for the DC network structure, we consider a fat-tree topology [12], which is very popular in modern DCs. However, we would like to stress that the specific topology does not affect the generality of the proposed model for VEs allocation that we propose in this work. In addition, Fig. 1 shows three levels of switches: i) the switches of the edge level connecting the servers of the same pod, ii) an upper layer of aggregation switches and iii) a top layer, named *data*

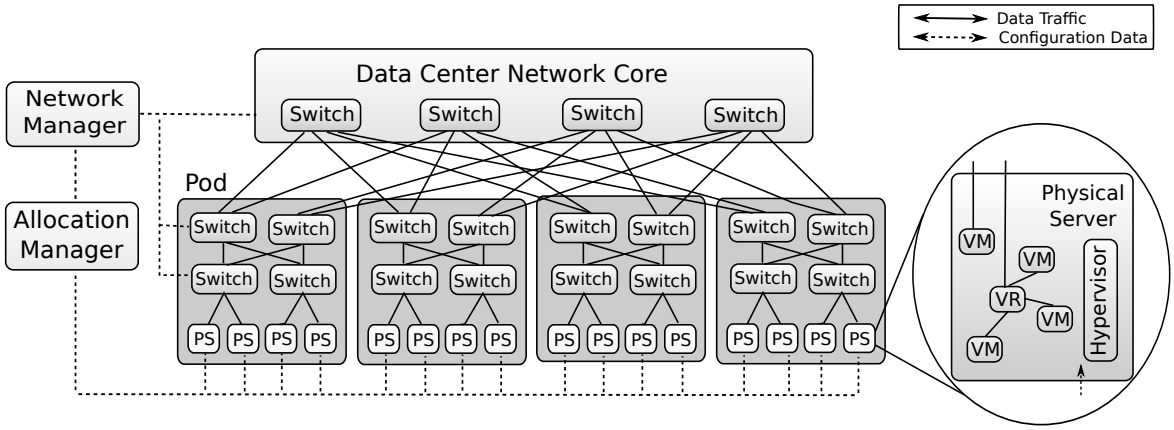


Fig. 1: Software-Defined Cloud Data Center (SDDC). PS:= Physical Server.

center network core, managing the communication among multiple pods. This structure implies different network costs for transferring data between: a) servers connected by an edge level switch; b) servers belonging to the same pod (i.e., passing through the aggregation level switches); and, c) servers belonging to different pods (i.e., passing through the DC core network).

B. Network Manager

In the considered SDDC architecture, the network manager component (which is shown on the top-left of Fig. 1) implements the network control logic (i.e., the control plane) for the whole infrastructure, by defining the rules for forwarding data across the infrastructure. Moreover, the network manager acts as an aggregator for the network utilization data, while the actual data collection is implemented at the level of the data plane. An example of data collection may be the exploitation of counters defined in the OpenFlow tables inserted in every network element [13]. Interestingly, the adoption of an SDDC architecture makes inherently scalable the operations of network monitoring, differently from what happens in traditional DCs.

C. Allocation Manager

The information collected by the network manager is given as input to the DC allocation manager, which is shown on the left part of Fig. 1. In addition to the network utilization information, the allocation manager also receives the monitoring information from the hypervisors located on each physical server. An example of collected data is the utilization of VEs resources, such as the consumed CPU and memory. In this scenario, the scalability of the data collection process may be improved if the system adopts a monitoring mechanism based on classes of VMs that have similar behavior in terms of resource utilization (that is, VMs running the same customer application). An example of a monitoring solution based on this approach is described in [14].

The allocation manager is then responsible for running the VEs allocation model, whose goal is to determine the optimal allocation of VEs on the physical servers to minimize the global energy consumption. After achieving a solution, the allocation manager notifies the servers about the VEs migrations that need to be applied. The communication between the allocation manager and the physical servers is realized by means of logical links, which are marked as dashed lines in Fig. 1.

In addition, it is worth to note that Traffic Engineering (TE) techniques are typically applied in SDDCs architectures [15]. These techniques allow performing the data transferring due to the VEs migration (that is, the actual transfer of the VE memory size from the source to the destination server) without affecting the performance of normal (i.e., application-related) network traffic. This property is one of the reasons supporting the integration of complex resource management policies in a software-defined architecture.

We recall that the allocation manager operates by planning VEs migrations across the infrastructure to accomplish the goal of minimizing the energy consumption of the cloud DC in terms of both computing and networking contributions. In this context, many solutions targeting the energy-aware VEs allocation are based on reactive behaviors, which rely on events to trigger the VE migrations [3], [9], [6]. These event-driven approaches are effective when the CPU utilization is considered. Specifically, it is feasible and easy to define events (which are typically based on thresholds) in order to trigger migrations from the analysis of the CPU utilization. On the other hand, this task is much more complex when network-related energy costs are taken into account. To go beyond this issue, in this work we have considered an approach based on time slots, where a control of the optimality of the VEs allocation on the physical servers is periodically performed by the allocation manager. In this way, it is less complex to define the conditions triggering the VE migrations.

III. Migration Costs vs. Data Transmission Costs: A Simple Case Study

Our work takes into account the combination of the computing costs for running the VEs on the physical servers, the costs for transferring data among the VEs and the costs for migrating the VEs across the physical servers. A natural question is then: What is the impact of the data transmission costs w.r.t. the migration ones? Or, in other words, is it better to migrate a VE to another physical server or to keep it on the current server? In order to shed light on these issues we report here a simple, yet representative, case study.

Fig. 2 details the considered SDDC case study, which is composed of two physical servers, denoted with "PS1" and "PS2", respectively. On these two servers, we place four VEs, namely two VMs (denoted with "VM1" and "VM2", respectively) and two VRs (denoted with "VR1" and "VR2", respectively). We represent the data transmission between VEs with dashed lines. For the sake of simplicity, we assume that only four data transmission patterns can occur, each of them involving a VM and a VR pair. The data transmission patterns are marked with the index of the involved VEs (e.g., VM1$\leftrightarrow$VR1: "1-1", VM1$\leftrightarrow$VR2: "1-2", and so on). As further assumptions, each data transmission pattern requires the same amount of data D and each physical server can run up to two VEs. In addition, each VE has the same footprint in terms of resource demands. As a consequence, the computing costs, denoted with $\mathcal{E}_C$, are the same for every VE allocation, and therefore do not have an impact on the allocation strategy.

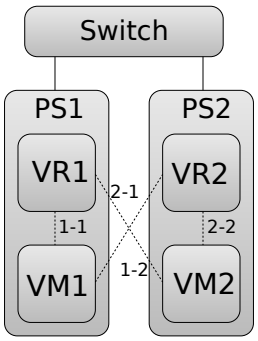


Fig. 2: Case study scenario.

TABLE I: Comparison of the MA and MUA strategies in the static network scenario.

	Algorithm	**Time Slot**			
		t	$t+1$	$t+2$	$t+3$
Data Transmission Patters between VEs	-	1-2 2-1	1-2 2-1	1-2 2-1	1-2 2-1
Migrations Performed	MA	n/a	n/a	n/a	n/a
	MUA	VR1$\rightarrow$ PS2 VR2$\rightarrow$ PS1	n/a	n/a	n/a
Energy Consumption per Time Slot	MA	$2\mathcal{E}_D$	$2\mathcal{E}_D$	$2\mathcal{E}_D$	$2\mathcal{E}_D$
	MUA	$2\mathcal{E}_M$	0	0	0
Cumulative Energy Consumption	MA	$2\mathcal{E}_D$	$4\mathcal{E}_D$	$6\mathcal{E}_D$	$8\mathcal{E}_D$
	MUA	$2\mathcal{E}_M$	$2\mathcal{E}_M$	$2\mathcal{E}_M$	$2\mathcal{E}_M$

In the following, we focus on the energy consumed by the migration and the data transmission processes. More formally, we denote with $\mathcal{E}_D$ the energy consumed by the data transferred between two VEs on different servers. We assume that the amount of data transferred among VEs running on the same server does not affect the total energy consumption. In addition, we denote with $\mathcal{E}_M$ the energy consumed by a migration of a VE among two physical servers. Finally, we assume that the energy cost due to a migration is slightly higher than the energy needed to transfer the data among VEs. More formally, it holds that $\mathcal{E}_D < \mathcal{E}_M < 2\mathcal{E}_D$.[1]

We now introduce the dynamic evolution of the data transferred between VEs. Let us assume that at time slot (TS) $(t-1)$ the VEs placement is as in Fig. 2, with VM1 and VR1 on server PS1 and VM2 and VR2 on server PS2, respectively. Furthermore, we consider two possible scenarios for the temporal variation of the data transferred between VEs, which we denote as *static network* and *dynamic network*, respectively. In the static case, the VEs pairs exchanging data are kept fixed across the set of TSs. Specifically, in our case study, VM1 will always exchange data with VR2. Similarly, also the pair VM2-VR1 will be kept unchanged. In the dynamic case, instead, the VEs pairs are varied in each TS. Note that, in both cases, the amount of data transferred between the VEs is taken as

[1]The exact setting of the input parameters to derive the values of $\mathcal{E}_D$ and $\mathcal{E}_M$ representative of current SDDCs will be detailed in Sec. VI.

TABLE II: Comparison of the MA and MUA strategies in the dynamic network scenario.

	Algorithm	**Time Slot**			
		t	$t+1$	$t+2$	$t+3$
Data Transmission Patterns between VEs	-	1-2 2-1	1-1 2-2	1-2 2-1	1-1 2-2
Migrations Performed	MA	n/a	n/a	n/a	n/a
	MUA	VR1→ PS2 VR2→ PS1	VR1→ PS1 VR2→ PS2	VR1→ PS2 VR2→ PS1	VR1→ PS1 VR2→ PS2
Energy Consumption per Time Slot	MA	$2\mathcal{E}_D$	0	$2\mathcal{E}_D$	0
	MUA	$2\mathcal{E}_M$	$2\mathcal{E}_M$	$2\mathcal{E}_M$	$2\mathcal{E}_M$
Cumulative Energy Consumption	MA	$2\mathcal{E}_D$	$2\mathcal{E}_D$	$4\mathcal{E}_D$	$4\mathcal{E}_D$
	MUA	$2\mathcal{E}_M$	$4\mathcal{E}_M$	$6\mathcal{E}_M$	$8\mathcal{E}_M$

an input by the allocation manager (i.e., the allocation manager does not influence the amount of data transferred between VEs).

We then introduce the allocation policy run by the allocation manager to map the VEs to the physical servers in each TS. To this aim, we consider two possible strategies, namely *Migration-UnAware (MUA)* and *Migration-Aware (MA)*, introduced in [1]. More in depth, MUA takes into account only $\mathcal{E}_C$ and $\mathcal{E}_D$, while not considering $\mathcal{E}_M$ at all. On the other hand, MA includes the impact of migration costs $\mathcal{E}_M$ in addition to the other terms. The results obtained by applying MUA and MA for each TS are reported in Tab. I and in Tab II for the static network and the dynamic one, respectively.

In the *static network* scenario (reported in Tab. I), the MA strategy performs no migrations because, within the single TS, the migration cost is higher than the data transmission cost, i.e., $\mathcal{E}_M > \mathcal{E}_D$. As a consequence, the energy is minimized by keeping the VEs allocation fixed across the TSs. On the contrary, the MUA strategy, in order to minimize the data transmission costs, will move VR1 on PS2 and VR2 on PS1 at TS t, resulting then in two migrations performed. Then, in the following TSs, MUA does not alter anymore the VE allocation, being able to keep zero data transmission costs. The final results, obtained at the end of TS $(t+3)$, indicate that MA has consumed a cumulative value of $8\mathcal{E}_D$ units of energy, while MUA only $2\mathcal{E}_M$ (we recall that, in our example, $\mathcal{E}_M < 2\mathcal{E}_D$).

Tab. II reports the results in the *dynamic network* scenario. In this case, the VEs pairs exchanging data are varied in each TS. Similarly to the *static network* scenario, the MA strategy does not perform any migration (as expected). On the other hand, the MUA strategy this time introduces plenty of migrations. In this case, MA has consumed only $4\mathcal{E}_D$ units of energy at TS $(t+3)$, while MUA $8\mathcal{E}_M$ units.

From these results, it is clear that the evolution over time of the data transferred between VEs has an impact on the results. Specifically, in the *static network* scenario the migrations performed by MUA at TS t have a benefit which spans across subsequent TSs, namely $(t+1), (t+2), (t+3)$, thanks to the fact that the VEs pairs transferring data are not varied. Therefore, the MUA strategy consumes less energy than MA in this case. On the other hand,

TABLE III: Main Notation.

	Symbol	Description
Input Parameters	$\mathcal{M}$	Set of on servers in the SDDC
	$\mathcal{N}$	Set of VEs
	$\mathcal{T}$	Set of TSs
	τ	TS Duration
	$x_{i,j}(t-1)$	Binary parameter: 1 if VE j is allocated to server i at TS $(t-1)$, 0 otherwise
	$c_j(t)$	CPU utilization of VE j at TS t
	$m_j(t)$	Memory requirement demand of VE j at TS t
	$d_{j_1,j_2}(t)$	Data transfer rate between VE j_1 and j_2 at TS t
	c_i^{MAX}	Maximum CPU utilization of server i
	d_i^{MAX}	Maximum amount of data manageable by the network interfaces of server i in each TS
	$\mathcal{E}_{d_{i_1,i_2}}$	Energy consumption for transferring one byte of data from i_1 to i_2
	m_i^{MAX}	Maximum memory utilization of server i
	P_i^{MAX}	Maximum power consumption of server i
	P_i^{IDL}	Idle power consumption of server i
	P_i^{NET}	Power consumption of the network interfaces in idle mode of server i
	K_{C_i}	Ratio between P_i^{IDL} and P_i^{MAX} of server i
	K_{M_i}	Computational overhead when server i is involved in a migration
	$\gamma(t)$	Weight for VE migration costs
Variables	$x_{i,j}(t)$	Binary variable: 1 if VE j is allocated to server j at TS t, 0 otherwise
	$O_i(t)$	Binary variable: 1 if server i is powered on, 0 if server i is powered off
	$g_{i,j}^{-}(t)$	Binary variable: 1 if VE j migrates *from* server i at TS t, 0 otherwise
	$g_{i,j}^{+}(t)$	Binary variable: 1 if VE j migrates *to* server i at time t, 0 otherwise
	$\mathcal{E}_{C_i}(t)$	Computing cost of server i at TS t
	$\mathcal{E}_D(t)$	Total data transmission costs at TS t
	$\mathcal{E}_{M_j}(t)$	Migration cost for VE j at TS t

in the *dynamic network* scenario, the benefit of migrations tends to be limited to the single TS since the VEs pairs transferring data are varied across the TS. As a result, the MA strategy achieves a better performance than MUA.

As a result, the presented case study reveals that neither MUA nor MA is the always-winning solution, as the dynamicity of data transferred over time heavily impacts the results. To solve this issue, we have investigated a new solution, which is detailed in the next section, in order to properly take into account the VEs migration costs.

IV. Problem modeling

We target the minimization of the total energy consumption in an SDDC, by jointly considering the computing costs, the data transmission costs, and the migration costs. The Joint Computing Data transmission and Migration Energy costs (JCDME) problem is run by the allocation manager at each TS. To pursue the goal of the energy minimization across multiple TSs, we exploit two intuitions: i) an optimization problem, including all the aforementioned energy costs, is solved at each TS, *and* ii) the migration costs in the objective function are weighed by a parameter, denoted with $\gamma(t)$, to reduce or to increase the impact of this term w.r.t. to the other terms. In particular, the first step aims to achieve a short-term objective, i.e., the minimization of the energy in the current TS, while

the second one introduces a long-term goal, i.e., the effect of the migrations costs on multiple TSs (as shown in the case study of Sec. III).

A. Model Formalization

To formalize the JCDME problem, consider a set of servers $\mathcal{M}$ and a set of VEs $\mathcal{N}$. Moreover, time is described as a discrete succession of TSs denoted as $\mathcal{T}$, where each TS has a fixed duration τ. We then consider the current TS t. We assume that at current t the VEs resource requirements for current TS and the VEs allocation at previous TS $(t-1)$ are known. At TS t, each server $i \in \mathcal{M}$ hosts multiple VEs (which may be VMs or VRs). Each VE $j \in \mathcal{N}$ has requirements in terms of CPU utilization $c_j(t)$, memory $m_j(t)$ and network bandwidth. In particular, we model the last term with the parameter $d_{j_1,j_2}(t)$, which denotes the amount of data transferred between VE j_1 and VE j_2 at TS t. It is worth to note that, in our model, we assume to have an estimation *before* the beginning of TS t of the requirements in terms of computational, memory and data transfer for each VE throughout the whole TS. This is a common assumption in the literature [1], [5], [9], as the requirements for the current TS can be obtained e.g., by adopting simple predicting techniques, possibly relying on periodic or cycle-stationary processes as suggested in [16], [17], [18].

We assume also that the respect of VE Service Level Agreement (SLA) is guaranteed *if and only if* the VE obtains its required resources in terms of CPU, memory, and network bandwidth (as in [3],[2][1]). The considered SLA is representative of an IaaS scenario. Finally, the decision variables of our problem are: i) an allocation binary matrix, whose elements $x_{i,j}(t) \in \{0,1\}$ describe the allocation of VE j on server i, and ii) a binary vector whose elements $O_i(t) \in \{0,1\}$ represent the status (powered off or powered on) of the physical server i.

B. Optimization goal

The JCDME problem targets the following objective function:

$$\mathcal{E}_{Obj}(t) = \sum_{i \in \mathcal{M}} \mathcal{E}_{C_i}(t) + \mathcal{E}_D(t) + \gamma(t) \sum_{j \in \mathcal{N}} \mathcal{E}_{M_j}(t) \quad \text{[J]} \tag{1}$$

where $\mathcal{E}_{C_i}(t)$ is the computing cost of server i, $\mathcal{E}_D(t)$ are the total data transmission costs, $\gamma(t)$ is the weight parameter for VEs migration costs, and $\mathcal{E}_{M_j}(t)$ is the migration cost of VE j. The $\gamma(t)$ parameter can be used to take into account migrations in different ways without the need to alter the model. For example, if we consider the case where $\gamma(t) = 0$ or $\gamma(t) = 1$, we obtain the approaches described in [1] as Migration UnAware (MUA) and Migration Aware (MA). More interestingly, the model allows investigating intermediate strategies ($0 < \gamma(t) < 1$), where we tune the short-term optimization problem to achieve a benefit for a long-term goal. As implied in Section III, we could ideally distribute the cost of migration over multiple TSs and may obtain an energy reduction not achievable with existing models. While for the short-term optimization problem we consider the objective function Eq. 1, we also introduce a *total* energy consumption that is significant for the long-term problem discussed

[2]Actually, in [3] only the CPU is considered for the SLA and only traditional VMs are taken into account.

later in this section (Eq. 2). Total energy consumption differs from the JCDME objective as it does not consider the $\gamma(t)$ parameter.

$$\mathcal{E}_{Tot}(t) = \sum_{i \in \mathcal{M}} \mathcal{E}_{C_i}(t) + \mathcal{E}_D(t) + \sum_{j \in \mathcal{N}} \mathcal{E}_{M_j}(t) \quad \text{[J]} \tag{2}$$

We now detail each term of $\mathcal{E}_{Obj}(t)$ and $\mathcal{E}_{Tot}(t)$.

1) VEs Computing Costs: The computing cost is defined as the energy consumption of the server, which is the sum of a static term, that has to be counted if the server is powered on, and a dynamic one, which instead scales linearly with the amount of CPU used by the VEs running on the server. The considered model is in line with the one presented by Beloglazov *et al.* in [3]. More formally, we have:

$$\mathcal{E}_{C_i}(t) = O_i(t) \cdot \tau \cdot P_i^{MAX} \left[K_{C_i} + (1 - K_{C_i}) \frac{\sum_{j \in \mathcal{N}} x_{i,j}(t) c_j(t)}{c_i^{MAX}} \right] \quad \text{[J]} \tag{3}$$

where τ is the TS duration [s], P_i^{MAX} [W] is the maximum power consumption of server i, K_{C_i} is defined as $K_{C_i} = \frac{P_i^{IDL}}{P_i^{MAX}}$, where P_i^{IDL} [W] is the power consumption of the server in idle mode (i.e., no VEs running on it, but powered on), and c_i^{MAX} is the maximum CPU utilization of server i.

2) VEs Data Transmission Costs: We consider the following contributions to compute the VEs data transmission costs: i) a fixed term, which includes the power consumption of the network interfaces of the server in idle mode, and ii) a linear term, which is proportional to the amount of transferred data between VEs. The considered model is consistent with previous works [3], [19], [16]. In addition, the linear term will be even more suitable for future SDDCs, which will largely exploit virtual network functions [11]. The data transmission costs are then expressed as:

$$\begin{aligned} \mathcal{E}_D(t) = & \sum_{i \in \mathcal{M}} O_i(t) \cdot \tau \cdot P_i^{NET} + \\ & + \sum_{j_1 \in \mathcal{N},\, j_2 \in \mathcal{N}} \sum_{i_1 \in \mathcal{M}} \sum_{i_2 \in \mathcal{M}} x_{i_1,j_1}(t) \cdot x_{i_2,j_2}(t) \cdot d_{j_1,j_2}(t) \cdot \tau \cdot \mathcal{E}_{d_{i_1,i_2}} \quad \text{[J]} \end{aligned} \tag{4}$$

where P_i^{NET} [W] is the power consumption of network interfaces in idle mode of server i, and $\mathcal{E}_{d_{i_1,i_2}}$ [J/B] is the cost for exchanging one byte of data between server i_1 and server i_2. More in detail, the matrix $\mathcal{E}_{d_{i_1,i_2}} \forall i_1, i_2 \in \mathcal{N}$ is able to reflect the characteristics of any topology of the DC network. We provide an example in Section IV-C to clarify how the network topology is mapped in $\mathcal{E}_{d_{i_1,i_2}} \forall i_1, i_2 \in \mathcal{N}$.

3) VEs Migration Costs: When a generic VE j migrates we observe two main effects. First, the whole amount of memory $m_j(t)$ consumed by VE j at TS t is transferred to the destination server.[3] Second, during the memory copy between two servers, we observe a performance degradation that we quantify using the parameter K_{M_i} for server i hosting VE j. According to the results in the literature (see e.g., [21]), this performance degradation is typically in the order of 10%, even though it is limited to a few tens of seconds (which is an amount of time that

[3]Actually, the amount of data transferred is slightly higher due to the need to retransmit dirty memory pages, but we neglect this effect due to the typically small size of the active page set w.r.t the global memory space of the VE [20].

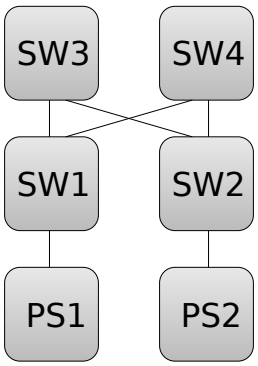


Fig. 3: Topology Example.

may be significantly lower compared to a typical TS duration τ). The energy cost for the migration of VE j is expressed as:

$$\mathcal{E}_{M_j}(t) = \sum_{i_1 \in \mathcal{M}} \sum_{i_2 \in \mathcal{M}} g^-_{i_1,j}(t) \cdot g^+_{i_2,j}(t) \cdot \Big[m_j(t) \cdot \mathcal{E}_{d_{i_1,i_2}} + \\ + (1 - K_{C_{i_1}}) \cdot P^{MAX}_{i_1} \cdot K_{M_{i_1}} \cdot \tau + (1 - K_{C_{i_2}}) \cdot P^{MAX}_{i_2} \cdot K_{M_{i_2}} \cdot \tau \Big] \quad \text{[J]} \tag{5}$$

where $g^-_{i_1,j}(t)$ is a binary variable taking the value 1 if VE j is migrated *from* server i_1 at time t (0 otherwise), $g^+_{i_2,j}(t)$ is another binary variable which takes value 1 if VE j is migrated *to* server i_2 at time t (0, otherwise),[4] $K_{M_{i1}}$ and $K_{M_{i2}}$ are the migration overheads for server i_1 and server i_2, respectively, which means that if the computational overhead for a migration on hypervisor i_1 is 1% over a TS of length τ, we will set $K_{M_{i1}} = 0.01$.

C. How to capture the SDDC physical network topology

We now discuss an important detail concerning the costs derived from the physical network elements in the considered model. Actually, the matrix $\mathcal{E}_{d_{i_1,i_2}} \forall i_1, i_2 \in \mathcal{N}$ implicitly embeds within its values the topology of the network. For example, when the physical network is realized in such a way that there is only one path connecting each pair of servers i_1 and i_2, then the data transmission cost $\mathcal{E}_{d_{i_1,i_2}}$ will be directly proportional to the number of switches in the path between i_1 and i_2. However, typical physical network topologies adopted for DCs (e.g., fat-tree ones) tend to guarantee multiple paths between each pair of servers, due to fault tolerance and load balancing reasons. In addition, different physical links may be turned off in order to save energy. In this context, the original network topology is reduced to a subgraph of physical links, still supporting the DC traffic.[5] In our work, we do not embed directly the topology (and the set of links powered on) in our model, in order to limit the increase in the algorithm complexity. In particular, we assume that a separate model is run to decide which physical links are powered on or powered off. Next, for each source-destination server pair, we consider the possible paths and we compute the overall cost. We use the example in Fig. 3 to show the paths between two possible servers

[4]The variables $g^-_{i_2,j}(t)$ and $g^+_{i_2,j}(t)$ are used to model migrations, as in [9].

[5]This issue has already been addressed in the literature, for example through heuristics as in [22].

PS1 and PS2, and how the cost $\mathcal{E}_{d_{PS1,PS2}}$ is computed. Both PS1 and PS2 are connected through two paths: the first one includes the links PS1-SW1, SW1-SW3, SW3-SW2, SW2-PS2, while the second one is composed of the links PS1-SW1, SW1-SW4, SW4-SW2, SW2-PS2. If we denote as $\mathcal{E}_{l_{PS1,SW1}}$ the energy to transfer one byte of data over the link between PS1 and SW1, we can write the energy for data transfer in the example as: $\mathcal{E}_{d_{PS1,PS2}} = \mathcal{E}_{l_{PS1-SW1}} + p_{SW3}(\mathcal{E}_{l_{SW1-SW3}} + \mathcal{E}_{l_{SW3-SW2}}) + p_{SW4}(\mathcal{E}_{l_{SW1-SW4}} + \mathcal{E}_{l_{SW4-SW2}}) + \mathcal{E}_{l_{SW2-PS2}}$, where p_{SW3} and p_{SW4} are the probabilities of using the paths that passes through SW3 and SW4, respectively. Finally, we assume that the probability of using a given path is inversely proportional to the link capacity.

D. Optimization formal model

The JOINT COMPUTING DATA TRANSMISSION AND MIGRATIONS ENERGY COSTS (JCDME) problem is formalized as follows.

$$\min \left[\sum_{i \in \mathcal{M}} \mathcal{E}_{C_i}(t) + \mathcal{E}_D(t) + \gamma(t) \sum_{j \in \mathcal{N}} \mathcal{E}_{M_j}(t) \right] \tag{6}$$

subject to:

Eq. (3)-(5)

$$\sum_{j \in \mathcal{N}} x_{i,j}(t) \cdot c_j(t) \leq c_i^{MAX} \cdot O_i(t) \quad \forall i \in \mathcal{M}, \tag{7}$$

$$\sum_{j_1 \in \mathcal{N}} \sum_{j_2 \in \mathcal{N}} \left[x_{i,j_1}(t) + x_{i,j_2}(t) - 2x_{i,j_1}(t)x_{i,j_2}(t) \right] d_{j_1,j_2}(t)$$
$$\leq d_i^{MAX} \cdot O_i(t) \quad \forall i \in \mathcal{M}, \tag{8}$$

$$\sum_{j \in \mathcal{N}} x_{i,j}(t) \cdot m_j(t) \leq m_i^{MAX} \cdot O_i(t) \quad \forall i \in \mathcal{M}, \tag{9}$$

$$\sum_{i \in \mathcal{M}} x_{i,j}(t) = 1 \quad \forall j \in \mathcal{N}, \tag{10}$$

$$\sum_{i \in \mathcal{M}} g_{i,j}^{+}(t) = \sum_{i \in \mathcal{M}} g_{i,j}^{-}(t) \leq 1 \quad \forall j \in \mathcal{N}, \tag{11}$$

$$g_{i,j}^{-}(t) \leq x_{i,j}(t-1) \quad \forall j \in \mathcal{N}, i \in \mathcal{M}, \tag{12}$$

$$g_{i,j}^{+}(t) \leq x_{i,j}(t) \quad \forall j \in \mathcal{N}, i \in \mathcal{M}, \tag{13}$$

$$g_{i,j}^{-}(t) + g_{i,j}^{+}(t) \leq 1 \quad \forall j \in \mathcal{N}, i \in \mathcal{M}, \tag{14}$$

$$x_{i,j}(t) = x_{i,j}(t-1) - g_{i,j}^{-}(t) + g_{i,j}^{+}(t) \quad \forall j \in \mathcal{N}, i \in \mathcal{M}, \tag{15}$$

under decision variables: $x_{i,j}(t) \in \{0,1\}$, $O_i(t) \in \{0,1\}$

More in detail, Eq. (3)-(5) compute the different energy terms of the objective function (6). Constraint (7) imposes that the CPU used by the VEs allocated on each server has to be lower than the maximum server CPU utilization c_i^{MAX}. Constraint (8) imposes that the amount of traffic exchanged between all the VEs allocated on the server i and the other ones allocated on other servers has to be lower than the maximum amount of data manageable by

the server d_i^{MAX} in each TS. More in detail, we consider only the VEs pairs in which one VE is allocated on the server, and another one is allocated on another server. On the other hand, when both the VEs are allocated on the same server, they do not consume the networking resources. Such condition corresponds to the operation $x_{i,j_1}(t) \oplus x_{i,j_2}(t)$, which is then translated into $x_{i,j_1}(t) + x_{i,j_2}(t) - 2x_{i,j_1}(t) \cdot x_{i,j_2}(t)$ in (8). In addition, constraint (9) limits the memory utilization of the VEs on each server to be smaller than the maximum memory utilization m_i^{MAX}.

Focusing then on the VEs constraints, Eq. (10) ensures that each VE is assigned to *one and only one* server. The update of the migration variables $g_{i,j}^{+}(t)$ and $g_{i,j}^{-}(t)$ is done through constraints (11)-(15). Specifically, Eq. (11) imposes that: i) a VE may be involved in at most one migration (i.e., the inequality constraint), and ii) a VE involved in a migration has to appear in both the summations $\sum_{i \in \mathcal{M}} g_{i,j}^{-}(t)$ and $\sum_{i \in \mathcal{M}} g_{i,j}^{+}(t)$. Then, constraint (12) ensures that a VE may migrate only from a server where the VE was allocated in TS $(t-1)$. Moreover, constraint (13) imposes that a VE may migrate only to a server where the VE is allocated in TS t (this constraint is redundant because it is inherently satisfied by constraint (15), but we add it for the clarity of the model), while constraint (14) rules out migrations of the same VE from one server to the same server. In addition, constraint (15) imposes the allocation of a VE to a server at TS t, under the following conditions: i) the VE was allocated to the server at TS $(t-1)$, and it is allocated to the same server also for TS t, i.e., no migrations have occurred, or ii) the VE was allocated to another server at TS $(t-1)$, and then at TS t the VE has been allocated to the current server i, or iii) the VE was allocated to the current server i at TS $(t-1)$, and it is allocated to another server at TS t. Tab. (III) reports the main notation introduced in the paper.

E. Problem Complexity

We analyze the complexity of the optimization problem in terms of number of variables and number of constraints. Focusing on the VMs to servers assignment $x_{i,j}(t)$, these variables are stored in a matrix of size $|\mathcal{M}| \times |\mathcal{N}|$. Similarly, also $g_{i,j}^{-}(t)$ and $g_{i,j}^{+}(t)$ are stored in matrices of size $|\mathcal{M}| \times |\mathcal{N}|$. In addition, the power state of the server $O_i(t)$ is stored in an array of $|\mathcal{M}|$ elements. Focusing then on the variables used to update the consumed energy, $\mathcal{E}_{C_i}(t)$ and $\mathcal{E}_{M_j}(t)$ require $|\mathcal{M}|$ and $|\mathcal{N}|$ elements, respectively. Finally, $\mathcal{E}_D(t)$ is a single variable. Overall, the whole formulation adopt $3 \times |\mathcal{M}| \times |\mathcal{N}| + 2 \times |\mathcal{M}| + |\mathcal{N}| + 1$ variables.

Focusing then on the number of constraints, the computation of the energy consumption in Eq. 3,4,5,6 is performed in four constraints. In addition, the maximum CPU utilization of Eq. 7 requires $|\mathcal{M}|$ constraints. Similarly, the limitation of the maximum amount of data (Eq. 8) and the maximum memory utilization (Eq. 9) require $|\mathcal{M}|$ constraints. Moreover, $|\mathcal{N}|$ constraints are needed to ensure the assignment of VMs to servers (Eq. 10), as well as to limit the number of migrations for each VM (Eq. 11). Finally, Eq. 12,13,15,14 need to be ensured for each VM and each server, resulting in $|\mathcal{M}| \times |\mathcal{N}|$ constraints for each inequality. The total number of constraints of the whole formulation is then equal to $4 \times |\mathcal{M}| \times |\mathcal{N}| + 3 \times |\mathcal{M}| + 2 \times |\mathcal{N}| + 4$.

F. Adaptive estimation of $\gamma(t)$ parameter

Up to now, the weight parameter $\gamma(t)$ is kept as an input parameter to the JCDME problem. However, to provide a fully adaptive technique to identify the value of $\gamma(t)$ that can minimize the long-term total energy consumption

(as discussed in Sec. III) we present three different strategies for the automatic and adaptive estimation of $\gamma(t)$. Our idea is that $\gamma(t)$ is adaptively estimated at each TS, and then passed as an input parameter to the optimization problem JCDME. We now describe each strategy in more detail.

1) Adaptive-JCDME (A-JCDME): The A-JCDME strategy introduces a simple feedback based on the number of migrations occurred in the previous TSs. The intuition is the following: if a low number of migrations has been experienced, then the algorithm is likely to be too conservative with respect to the migration energy contribution. On the other hand, if a large number of migrations has been observed, then the algorithm should take into account the impact of energy migration significantly. As a result, we set $\gamma(t)$ to be directly proportional to the average number of migrations occurred at TS $(t-1)$. More formally, we have:

$$\gamma(t) = \frac{1}{N} \sum_{j \in \mathcal{N}, i \in \mathcal{M}} g^{+}_{i,j}(t-1), \tag{16}$$

Once $\gamma(t)$ is computed, it is passed as an input parameter to the JCDME problem, which is solved for the current TS t.

2) Adaptive-plus-Smoothing JCDME (AS-JCDME): The A-JCDME strategy is able to modify the value of $\gamma(t)$ based on the migrations performed at TS $(t-1)$. However, the computed value of $\gamma(t)$ may be subject to large oscillations between one TS and the following one. For example, if a large number of migrations is performed at TS $(t-1)$, $\gamma(t)$ will be increased at TS t, thus drastically reducing the number of migrations performed at TS t. On the contrary, a TS with a very small number of migrations may determine a low value of $\gamma(t)$ with an opposite effect in the subsequent TS. In both cases, the increase/decrease of $\gamma(t)$ may be too large, thus leading to a sub-optimal solution. To solve this issue, we introduce the Adaptive-plus-Smoothing JCDME (AS-JCDME) strategy by applying a EWMA smoothing filter [17] to $\gamma(t)$. In particular, we introduce a parameter, denoted with θ, acting as a weight for $\gamma(t-1)$. The value of $\gamma(t)$ is then computed as:

$$\gamma(t) = \theta \cdot \frac{1}{N} \sum_{j \in \mathcal{N}, i \in \mathcal{M}} g^{+}_{i,j}(t-1) + (1-\theta) \cdot \gamma(t-1) \tag{17}$$

In this way, the value of θ allows to trade between the value of γ computed at current TS with the one at TS $(t-1)$ (which, in turn, recursively includes the values of γ computed in previous TSs).

3) Adaptive Newton-based JCDME (AN-JCDME): The AN-JCDME strategy relies on the Newton method, a largely exploited solution to find the roots of the derivative of a function, and hence finding its minimum values. The total energy consumption in $\mathcal{E}_{Tot}(t)$ (in Eq. (2)) depends on the solution of the optimization problem. This solution depends on the $\gamma(t)$ parameter, hence we aim to find the minimum $\mathcal{E}_{Tot}(t)$ as a function of $\gamma(t)$. Ideally, using the Newton method, we approximate the curve $\mathcal{E}_{Tot}(t)$ *vs.* $\gamma(t)$ as a parabola and we find its minimum [23]. We anticipate that the experimental results in Fig. 6 confirms our intuition that this curve has a convex shape. More in detail, we start from the weight value $\gamma(t-1)$ at previous TS. We then solve the JCDME optimization problem at TS t with $\gamma(t-1)$, and we store the total energy consumption in a variable, denoted with $\mathcal{E}^{0}_{Tot}$. In the following,

we define two new candidate values of $\gamma(t)$, namely:

$$\gamma^{-}(t) = \gamma(t-1) - \delta \tag{18}$$

$$\gamma^{+}(t) = \gamma(t-1) + \delta \tag{19}$$

where δ is a fixed parameter. In the next step, we solve the JCDME optimization problems for the cases $\gamma^{-}(t)$ and $\gamma^{+}(t)$, and store the obtained energy values in the variables $\mathcal{E}^{-}_{Tot}(t)$ and $\mathcal{E}^{+}_{Tot}(t)$, respectively. Then, we numerically compute the first and second derivative of the total energy function in the following way:

$$\mathcal{E}'_{Tot}(t) = \frac{d\mathcal{E}_{Tot}(t)}{d\gamma(t)} = \frac{\mathcal{E}^{+}_{Tot}(t) - \mathcal{E}^{-}_{Tot}(t)}{2\delta}, \tag{20}$$

$$\mathcal{E}''_{Tot}(t) = \frac{d^2\mathcal{E}_{Tot}(t)}{d^2\gamma(t)} = \frac{\mathcal{E}^{+}_{Tot}(t) - 2\mathcal{E}^{0}_{Tot} + \mathcal{E}^{-}_{Tot}(t)}{\delta^2} \tag{21}$$

The Newton method allows to find the minimum of a function that is characterized by a zero value for the first derivative. Therefore, we set the new value of $\gamma(t)$, which as:

$$\gamma(t) = \gamma(t-1) - \frac{\mathcal{E}'_{Tot}(t)}{\mathcal{E}''_{Tot}(t)} = \gamma(t-1) - \frac{\delta}{2} \cdot \frac{\mathcal{E}^{+}_{Tot}(t) - \mathcal{E}^{-}_{Tot}}{\mathcal{E}^{+}_{Tot}(t) - 2\mathcal{E}^{0}_{Tot} + \mathcal{E}^{-}_{Tot}(t)} \tag{22}$$

The computed value of $\gamma(t)$ is then passed as an input for the next TS $(t+1)$.

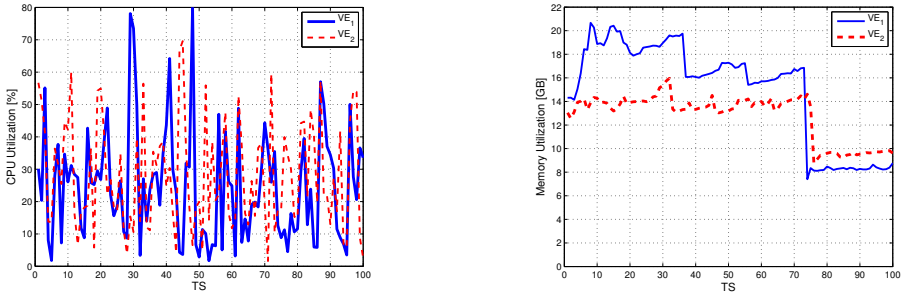
Compared to the classical Newton method [23], which requires several iterations, we stop at the first one, in order to limit the required computation time. Our aim is in fact to dynamically (and automatically) tune $\gamma(t)$ to an admissible value, which is not strictly required to be the best one. As a result, the above method requires the solution of the allocation problem three times for each TS. To solve this issue, we may assume that $x_{i,j}(t), g^{+}_{i,j}(t), g^{-}_{i,j}(t)$ are continuous variables in the interval $[0, 1]$ (and not binary ones), for the computation of $\mathcal{E}^{+}_{Tot}(t)$ and $\mathcal{E}^{-}_{Tot}(t)$. In this way, we expect to reduce the algorithm complexity. The evaluation of this improvement is left as future work.

G. *Application to Large Data Centers*

As highlighted in Sec. IV-E, the complexity of the JCDME problem grows with the number of VMs $|\mathcal{N}|$ and the number of servers $|\mathcal{M}|$. As a result, it may be challenging to solve the whole formulation in a scenario in which hundreds of servers and thousands of VMs are taken into account. Nevertheless, we believe that the presented formulation is still meaningful when a *divide et impera* approach is pursued. In particular, a common intuition in server consolidation problems is that a VM normally does not exchange data with all the other VMs, but it tends to communicate with a small subset of VMs [5], [24]. As a result, VMs can be clustered into groups, in order to analyze their behavior and to understand which VMs belong to the same application [14]. Similarly, a VR realizing a connection between the VMs of the same application will be independent of the other VRs connecting other applications. In this scenario, therefore, it is possible to cluster the VEs into independent groups and to preallocate each group to a subset of servers. Then, as a second step, the energy consumption is optimized for each group of VMs and each subset of servers, by adopting one of the strategies proposed in this work.

TABLE IV: Parameters Setting.

Parameter	Value/Explanation	Reference	Appears in Eq.
$\mathcal{M}$	Set of considered servers in the SDDC, with $\lvert\mathcal{M}\rvert = 5 - 250$	[1]	-
$\mathcal{N}$	Set of considered VEs in the SDDC, with $\lvert\mathcal{N}\rvert = 12 - 600$ for the VR=20% case and $\lvert\mathcal{N}\rvert = 13 - 650$ for the VR=30% case	[1]	-
$\mathcal{T}$	Set of considered TSs in the SDDC, with $\lvert\mathcal{T}\rvert = 100$	[1]	-
$x_{i,j}(t-1)$	Input parameter depending on the configuration at previous TS	-	(12),(15)
τ	15 [m]	[1]	(3),(4),(5)
m_i^{MAX}	128 [GB]	[25]	(9)
c_i^{MAX}	Maximum utilization of the CPU of the server: 300% (corresponding to 3 VEs, each of them requiring 100% of the CPU)	[25]	(3),(7)
d_i^{MAX}	15 [m] · 1 [Gb/s] = 125 [GB]	[1]	(8)
$c_j(t)$	CPU utilization of VE j at TS t, depending on the available trace	[1]	(3),(7)
$m_j(t)$	Memory requirement demand of VE j at TS t, depending on the available trace	[1]	(9)
$d_{j_1,j_2}(t)$	Data transfer rate between VE j_1 and j_2 at TS t, see Sec. V for explanations.	[1]	(4),(8)
P_i^{MAX}	328.2 [W]	[25]	(3),(5)
P_i^{IDL}	197.6 [W]	[25]	Used to compute K_{C_i} in (3),(5)
K_{C_i}	0.6	[25]	(3),(5)
P_i^{NET}	42.7 [W]	[26], [27]	(4)
$\mathcal{E}_{d_{i_1,i_2}}$	Topology dependent value: 3 [W/B] in the case only the Top-of-Rack switch is traversed from i_1 to i_2, 6 [W/B] if also the inner part of the DC network is traversed from i_1 to i_2.	[26], [27]	(4),(5)
K_{M_i}	0.01	[1]	(5)
$\gamma(t)$	Input parameter depending on the considered strategy	-	(1),(6)



(a) Utilization of a single core CPU.

(b) Memory utilization.

Fig. 4: CPU and memory utilization vs. TS for two exemplary VEs.

V. Description of the Scenarios

We initially describe a basic scenarios, and then we detail how we have generated larger scenarios, i.e. up to hundreds of servers and hundreds of VEs.

A. *Basic Scenario*

Tab. IV reports the settings of the input parameters. More in detail, we consider a set of 5 servers and 100 TSs. Each server is a Dell R410 with a 2×6 cores Xeon X5670 2.93MHz and 128 GB of RAM. The maximum number of VEs running on each server is limited by the maximum server CPU utilization c_i^{MAX}. In the worst case (i.e., all VMs requiring 100% of CPU resources), it is possible to host no more than three VEs on each server. Focusing then on the number of VMs, we consider a total of 10 machines. In addition, we consider two settings for the number of VRs, namely: i) VR=20% of the number of VMs, corresponding to two VRs, and ii) VR=30% of the number of VMs, corresponding to three VRs. In addition, the DC network is based on a fat-tree topology, consisting of a three-level hierarchy of switches, with the upper layer of networking managing the communication among multiple pods of physical servers (we refer the reader to Fig. 1 for a high-level scheme of the DC elements and their interconnections). Moreover, the variation of CPU and memory over time for each VE is taken from a real DC deployed over a private cloud infrastructure and hosting an e-Health application [1]. To this aim, Fig. 4 reports two exemplary traces of a single core CPU utilization and memory utilization vs. the TS index. From the figure, we can clearly see that both the CPU and the memory requirements vary w.r.t. the TS index, as well as across different VEs.

Focusing then on the amount of data exchanged between each pair of VEs $d_{j_1,j_2}(t)$, the total amount of data from/to each VE to the other ones is known, i.e., $\sum_{j2\in\mathcal{N}} d_{j,j_2}(t)$ and $\sum_{j1\in\mathcal{N}} d_{j_1,j}(t)$ are available for each server j. As a result, we had to reconstruct the original matrix $d_{j_1,j_2}(t)$ in a synthetic way. More in detail, we exploit a random distribution of the incoming traffic so that the total summation is equal to the available data. To do so, we built a symmetric matrix with [(VR+VM)×(VR+VM)] elements, where $VR = VE \cdot \kappa$ (i.e., VR is defined as the number of the engaged VRs to serve VMs) with $\kappa \in [0, 100\%]$ in such a way that, i) the total traffic per row and per column for each VE j is equal to $\sum_{j2\in\mathcal{N}} d_{j,j_2}(t)$ and $\sum_{j1\in\mathcal{N}} d_{j_1,j}(t)$, respectively, and ii) the network traffic exchange of the diagonal elements is set to zero (i.e., each VE does not exchange traffic with itself).

In the following, we consider the input parameters related to power. More in detail, the server idle power consumption P_i^{IDL} and the maximum power consumption P_i^{MAX} are set equal to 197.6 [W], and 328.2 [W], respectively (in accordance to the publicly available datasheet of [25]). Consequently, it holds that $K_{C_i} = 0.6$. In the following, we focus on the setting of $\mathcal{E}_{d_{i_1,i_2}}$, which we recall is the energy needed to send one byte of data from one server to another one. In this context, the energy consumption values are derived from the basic consumption values for the switching infrastructure of a DC [26], [27]. From these sources, we set $\mathcal{E}_{d_{i_1,i_2}}$ equal to 3 [J/B] in the case we are passing only through the ToR switch and 6 [J/B] if the inner part of the network (i.e., the DC network core) is involved, respectively. Finally, we set the overhead K_{M_i} equal to 0.01, in accordance to [1].

B. *Generation of Scenarios with Different Size*

In the following, we generate a set of scenarios of different sizes, i.e., from a small sized DC (composed of few servers and VEs) up to a medium sized one (composed of hundreds of servers and hundreds of VEs). More in depth, we proceed as follows. First, we take into account the input data coming from the available real dataset trace of [1] with $|\mathcal{M}| = 5$ servers, $|\mathcal{N}| = 12$ and $|\mathcal{N}| = 13$ VEs for the VR=20% and VR=30% cases (respectively), and

TABLE V: Energy Breakdown of JCDME for different values of γ and percentage of VRs (numerical values)

Fixed weight parameter	E^{AVG-TS}	E_C^{AVG-TS}	E_D^{AVG-TS}	E_M^{AVG-TS}
VR=20%				
$\gamma = 0$	13320.11 [kJ]	7849.05 [kJ]	1331.88 [kJ]	4139.18 [kJ]
$\gamma = 1$	9942.67 [kJ]	7849.05 [kJ]	1957.52 [kJ]	136.10 [kJ]
VR=30%				
$\gamma = 0$	12905.40 [kJ]	7953.16 [kJ]	1497.64 [kJ]	3454.60 [kJ]
$\gamma = 1$	10489.59 [kJ]	7972.92 [kJ]	2354.90 [kJ]	161.77 [kJ]

$|\mathcal{T}| = 100$ Time Slots (TSs). We recall that the input data include the CPU and memory requirements for each VE, as well as the amount of data exchanged by the VMs. This information is available for each TS. In the following, we generate a set of 50 traces by performing for each trace a random permutation of the TSs from the original data. We then group the generated data into 50 scenarios, with the following rule: given the index $s \in (1, 50)$ of the scenario, the cardinality of the scenario is then defined as $|\mathcal{M}| = 5 \cdot s$, $|\mathcal{N}| = 12 \cdot s$ and $|\mathcal{M}| = 5 \cdot s$, $|\mathcal{N}| = 13 \cdot s$ when the percentage of VRs is equal 20% and 30%, respectively. Finally, the remaining parameters are kept the same as in the basic scenario.

VI. Performance Evaluation

We initially run the considered strategies over the basic scenario. We adopt the KNITRO 9.0 [28] software to solve the optimization problems. We then select different evaluation metrics to assess the performance of our solutions. More in detail, we compute the energy consumption over the entire set of TSs for the different energy components:

$$E_C^{TOT} = \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{M}} \mathcal{E}_{C_i}(t) \tag{23}$$

$$E_D^{TOT} = \sum_{t \in \mathcal{T}} \mathcal{E}_D(t) \tag{24}$$

$$E_M^{TOT} = \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{N}} \mathcal{E}_{M_j}(t) \tag{25}$$

We then define the total energy consumption over the whole set of TSs as:

$$E^{TOT} = \sum_{t \in \mathcal{T}} \mathcal{E}_{Tot}(t) = E_C^{TOT} + E_D^{TOT} + E_M^{TOT} \tag{26}$$

In addition, we also compute the average energy consumptions per TS as:

$$E_C^{AVG-TS} = \frac{E_C^{TOT}}{|\mathcal{T}|} \tag{27}$$

$$E_D^{AVG-TS} = \frac{E_D^{TOT}}{|\mathcal{T}|} \tag{28}$$

$$E_M^{AVG-TS} = \frac{E_M^{TOT}}{|\mathcal{T}|} \tag{29}$$

$$E^{AVG-TS} = E_C^{AVG-TS} + E_D^{AVG-TS} + E_M^{AVG-TS} \tag{30}$$

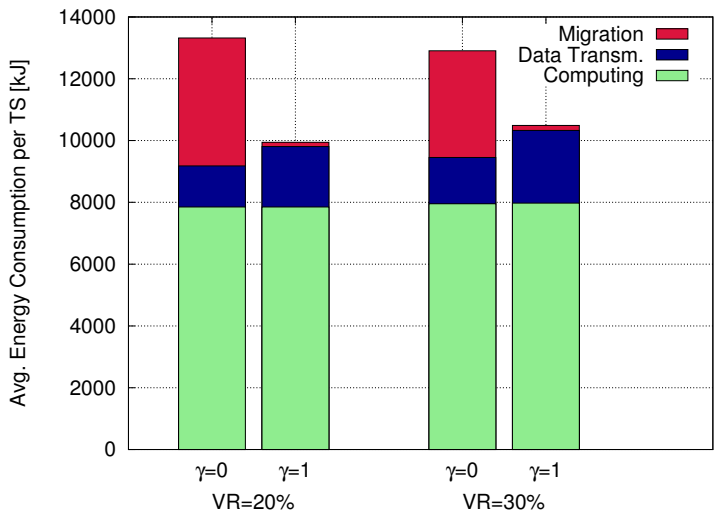


Fig. 5: Energy Breakdown of JCDME for different values of γ and percentage of VRs.

We initially solve the JCDME problem by first selecting two fixed values over time of the weight parameter γ, namely $\gamma = 0$ and $\gamma = 1$. In addition, we consider also the variation of the percentage of VRs w.r.t to the total number of VEs. Fig. 5 reports E^{AVG-TS}, by detailing also the breakdown between computing, data transmission and migration energy costs. The corresponding numerical values for this experiment are reported in Tab. V. We recall that the solution of the JCDME problem with $\gamma = 0$ corresponds to the Migration UnAware (MUA) strategy of [1]. On the other hand, the solution with $\gamma = 1$ matches the Migration-Aware (MA) algorithm of [1]. In both cases, we observe similar values of the computing costs E_C^{AVG-TS}, as the server consolidation is always pursued both settings of γ. As expected, the setting of $\gamma = 1$ allows reducing the migration energy E_M^{AVG-TS}. More in detail, E_M^{AVG-TS} passes from 4138.18 [kJ] when $\gamma = 0$ to only 136.1 [kJ] when $\gamma = 1$ in the VR=20% case. A similar reduction is also experienced for the VR=30% case. However, an increase of the data transmission energy E_D^{AVG-TS} is observed compared to the case $\gamma = 0$. The resulting effect, however, is a reduction E^{AVG-TS} when $\gamma = 1$. Finally, we observe that E^{AVG-TS} is increased when the percentage of VRs passes from 20% to 30%, for the $\gamma = 1$ case. Thus, the impact of data transmission when migrations are considered tends to increase when the percentage of VRs is increased.

By observing Fig. 5, a natural question arises: Why is the energy consumption of computing similar to the different bars? Focusing on the same VR setting (e.g., the VR=20% case), the total number of VEs is kept constant. Hence, the same computing cost is experienced when $\gamma = 0$ or $\gamma = 1$, since the energy due to computing is always much higher than the one due to migrations and data transmission. Focusing then on different VR setting (e.g., the first bar compared to the third one, and the second bar compared to the fourth one), the variation of VRs from 20% to 30% involves the adding of one VR, which tends to slightly increase the total computation costs. Nevertheless, we point out that the computation costs are mostly impacted when the servers are powered off and powered on. In our case, the adding of one VR does not introduce a substantial change in the powering off and powering on

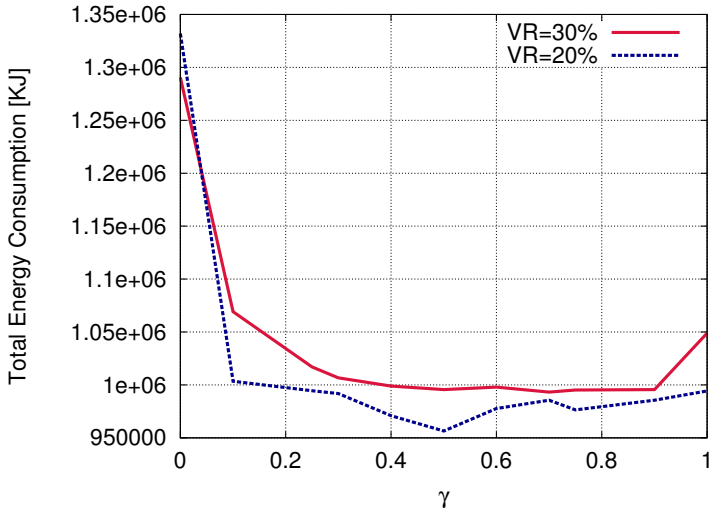


Fig. 6: Total energy consumption of JCDME for different values of γ

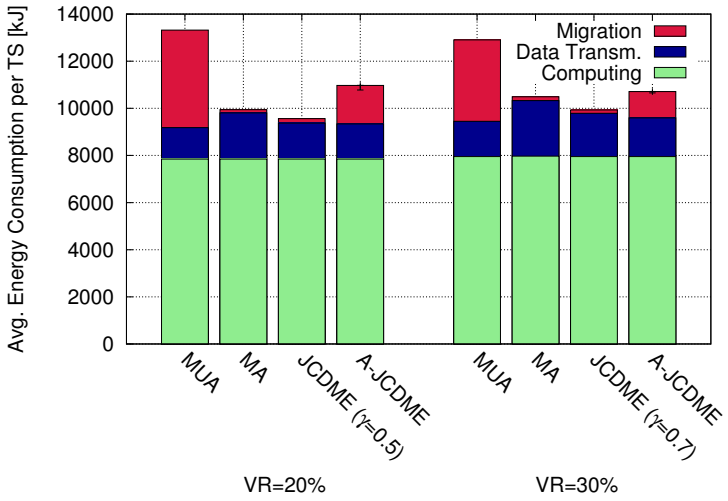
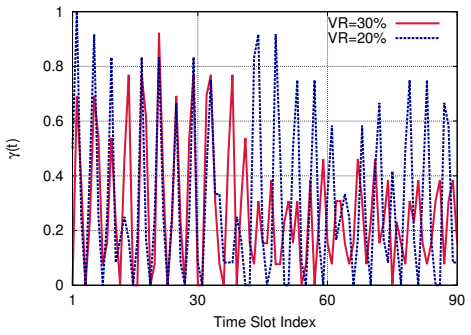


Fig. 7: Comparison of MUA, MA, JCDME and A-JCDME strategies in terms of E^{AVG-TS} and its components.

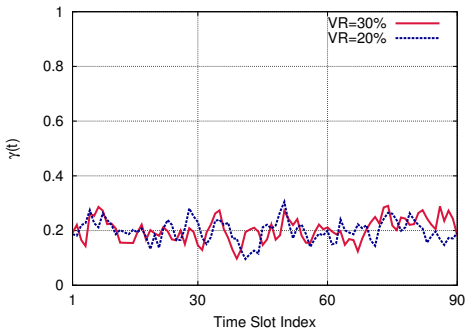
patterns of the servers, yielding to a small increase in the computation costs.

In the following, we run the JCDME problem by considering a variation of the γ parameter in the interval $[0, 1]$. Fig. 6 reports the total energy E^{TOT} consumed in the DC vs. the variation of γ and the percentage of VRs. As expected, the value of γ minimizing the total energy consumption is not the one at the extremes 0 and 1, but it is located in an intermediate zone. In particular, E^{TOT} is minimized when $\gamma = 0.5$ and $\gamma = 0.7$ when the percentage of VRs is equal to 20% and 30%, respectively.

In the next part, we consider the impact of adding an adaptive estimation of the weight parameter $\gamma(t)$. We first focus on the A-JCDME strategy. Fig. 7 reports E^{AVG-TS} by comparing A-JCDME against the MUA and MA strategies of [1], and the JCDME with the best gamma found in the previous step (i.e., the values of γ minimizing



(a) A-JCDME: Evolution of $\gamma(t)$ vs. TS index.



(b) AS-JCDME: Evolution of $\gamma(t)$ vs TS index.

Fig. 8: Evolution of $\gamma(t)$ vs TS index for A-JCDME and AS-JCDME.

TABLE VI: E^{AVG-TS} with A-JCDME, different values of VRs, different values of $\gamma(0)$.

$\gamma(0)$	VR=20%	VR=30%
0	10972.88 [kJ]	10903.01 [kJ]
0.25	10653.70 [kJ]	10845.46 [kJ]
0.5	10653.70 [kJ]	10715.32 [kJ]
0.75	10655.12 [kJ]	10714.01[kJ]
1	10655.12 [kJ]	10714.01 [kJ]

the total energy consumption in Fig. 6). Interestingly, A-JCDME is able to reduce the energy compared to MUA. However, the other strategies are able to save more energy than A-JCDME. To better investigate this issue, we report the evolution of $\gamma(t)$ vs. the TS index in Fig. 8a. We recall that the A-JCDMA strategy computes the value of $\gamma(t)$ as a value proportional to the number of migrations in TS $(t-1)$. Therefore, $\gamma(t)$ is subject to strong oscillations between 0 and 1 values, as Fig. 8a clearly shows. Moreover, we have also verified if the initial value of $\gamma(0)$, which has to be set for the first TS, has an impact on the results. Tab. VI details the obtained results. Not surprisingly, only minor variations in E^{AVG-TS} are experienced when $\gamma(0)$ is varied. Therefore, the performance of A-JCDME is mainly impacted by the oscillations of $\gamma(t)$.

We then evaluate the performance of the adaptive plus smoothing strategy AS-JCDME. Fig. 8b reports again the variation of $\gamma(t)$ over time. As expected, the AS-JCDME strategy is able to stabilize the oscillations of $\gamma(t)$ w.r.t. A-JCDME (see Fig. 8a). This is beneficial also for the energy consumption of AS-JCDME, which is clearly lower than A-JCDME and close to the solution JCDME with fixed γ, as reported in Fig. 9-10 (for different percentages of VRs). Moreover, we recall that the JCDME strategy with fixed γ requires finding the optimum by solving the JCDME problem for different values of γ, while with A-JCDME and AS-JCDME $\gamma(t)$ is computed through an unsupervised (and computationally light) procedure. The only parameter required to be given as input to AS-JCDME is θ, which acts as a weight for the number of migrations at TS $(t-1)$ and the value $\gamma(t-1)$ (see Eq. (17)). However, as reported in Fig. 9-10, the variation of θ between 0.25 and 0.75 has a pretty limited impact on the algorithm performance. Finally, Tab. VII reports E^{TOT} vs. the variation of $\gamma(0)$ and θ for AS-JCDME (with 20%

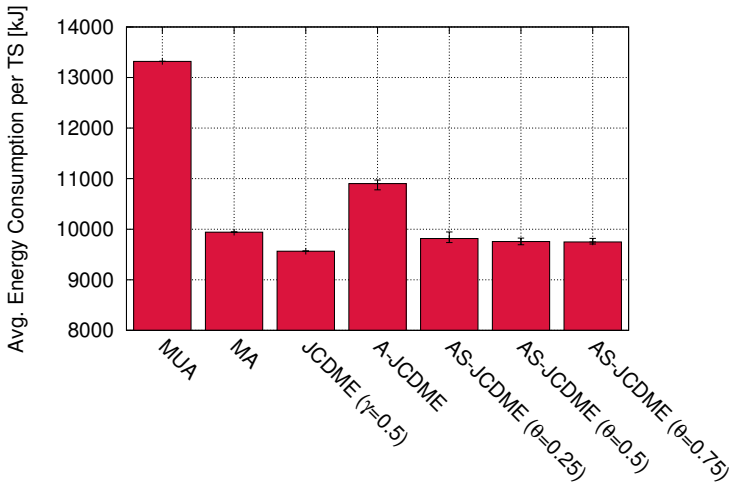


Fig. 9: Comparison of MUA, MA, A-JCDME and AS-JCDME strategies with VR=20%.

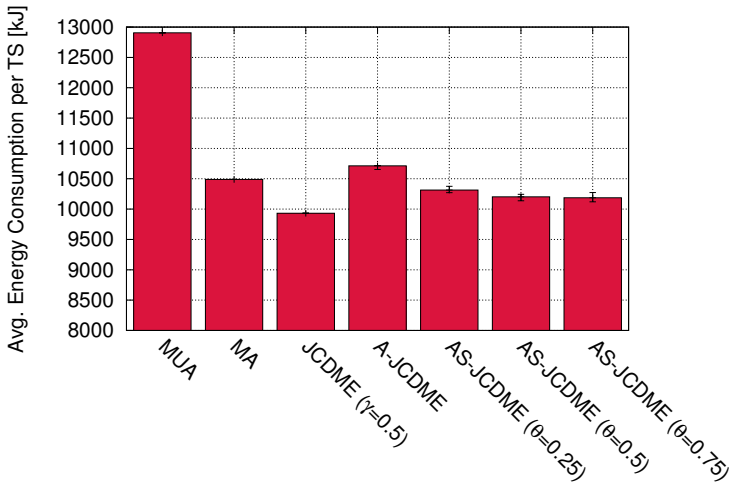


Fig. 10: Comparison of the MUA, MA, A-JCDME and AS-JCDME strategies with VR=30%.

of VRs). As expected, minor variations in the total energy are experienced (the results with 30% of VRs - omitted here for clarity - reveal similar trends).

In the following, we evaluate the performance of the Newton-based adaptive algorithm AN-JCDME. Fig. 11 reports the average energy consumption per TS by comparing MA, JCDME (with fixed γ), AS-JCDME and AN-JCDME, again with different percentages of VRs. In all cases, AN-JCDME is able to outperform all the other strategies, including JCDME with fixed γ. In particular, AN-JCDME is able to introduce up to an additional 7% of saving compared to MA. This confirms our intuition that a solution adaptively adjusting the weight parameter is the best one in the long-term (i.e., when different TSs are considered). In addition, AN-JCDME does not require any input parameter, apart from the value of δ to select the candidates $\gamma^{-}(t)$ and $\gamma^{-}(t)$ in Eq. (18)-(19). In our case,

TABLE VII: Total Energy E^{TOT} with AS-JCDME for different values of $\gamma(0)$ and θ (VR=20%)

$\gamma(0)$	$\theta = 0.25$	$\theta = 0.5$	$\theta = 0.75$
0	978637.36 [kJ]	974521.84 [kJ]	973776.54 [kJ]
0.25	994563.35 [kJ]	971292.87 [kJ]	974975.74 [kJ]
0.5	984677.36 [kJ]	969330.04 [kJ]	973156.28 [kJ]
0.75	973680.84 [kJ]	980307.82 [kJ]	970777.82 [kJ]
1	976064.12 [kJ]	982386.86 [kJ]	981741.21 [kJ]

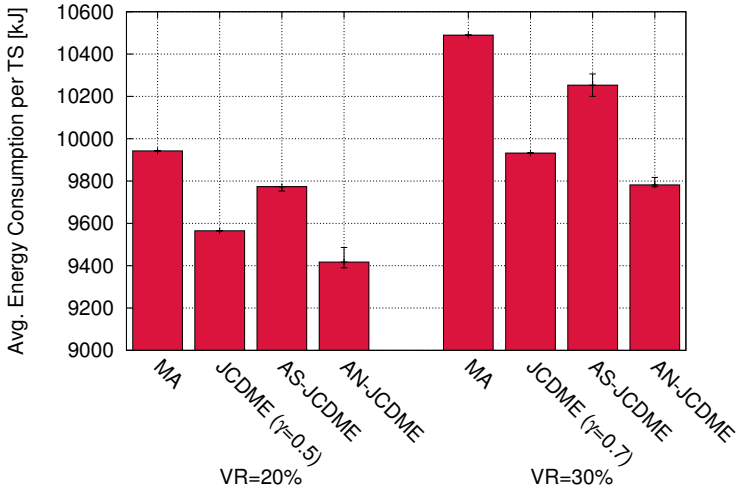
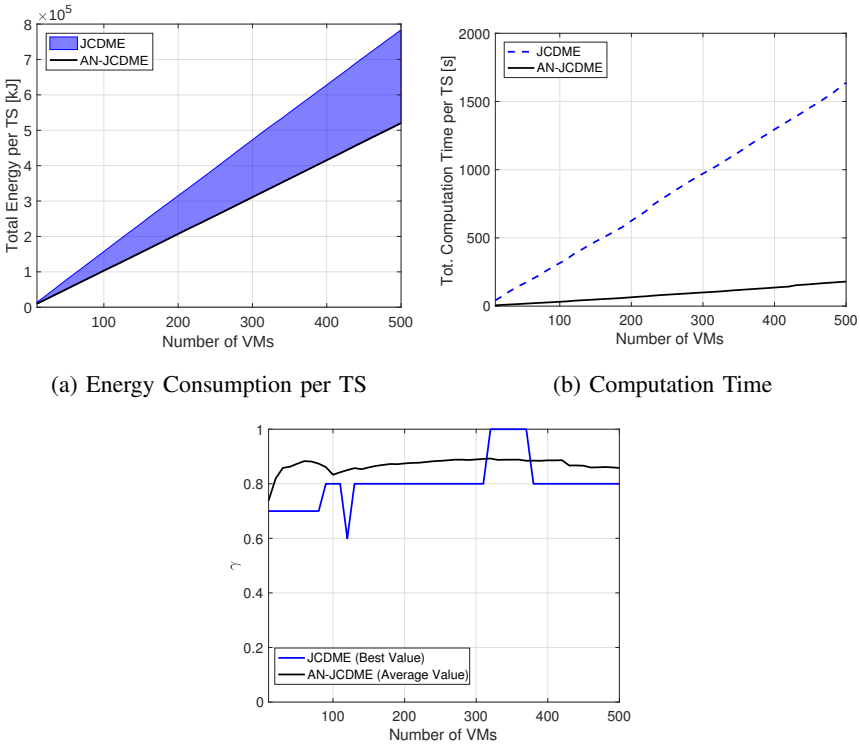


Fig. 11: E^{AVG-TS} with the MA, JCDME, AS-JCDME and AN-JCDME strategies.

we set $\delta = 0.05$. Moreover, additional experiments with $\delta \in [0.02, 0.2]$ (omitted here) suggest that AN-JCDME algorithm is not highly sensitive to δ.

In the last part of our work, we evaluate the AN-JCDME and JCDME strategies over the variation of the size of the scenario. We recall that each scenario is generated in accordance to Sec. V-B. We then adopt the proposed *divide et impera* approach, by considering groups of 5 servers. In addition, the number of VEs in each group is equal to 12 and 13 for the VR=20% and VR=30% cases, respectively. For each group, we run the JCDME/AN-JCDME strategies and collect the considered metrics. This procedure is repeated across all the generated scenarios. Fig. 12 reports the results for the VR=20% case, by showing: i) the energy consumed per TS (Fig. 12a), ii) the computation time per TS (Fig. 12b), and iii) the comparison of the γ parameter minimizing the total energy (Fig. 12c). Each subfigure reports the considered metrics vs. the number of VMs, which are scaling with the considered scenario. We omit the figures for the case VR=30% for space reasons.

Focusing first on the energy consumption (Fig. 12a), the blue area of JCDME corresponds to the energy consumed by varying the γ parameter $\in [0, 1]$. Clearly, the highest consumption occurs when γ is set to 0, i.e., when the energy cost of migrations is not considered at all in the objective function. On the contrary, the energy tends to be minimized for values of γ typically lower than one (reported in Fig. 12c). Interestingly, we can note that AN-



(a) Energy Consumption per TS

(b) Computation Time

(c) Best Value of γ

Fig. 12: Comparison of JCDME strategy with $\gamma \in [0, 1]$ and AN-JCDME vs. the variation of the number of VMs (with 20% of VRs).

JCDME always performs close to the minimum value of energy reached by JCDME. As expected, the energy grows with the number of VMs (and consequently with the size of the scenario). However, we point out that JCDME is more computationally intensive than AN-JCDME, due to the fact that the former requires a brute force search of the value of γ minimizing the energy, while the latter is able to automatically tune the γ parameter. To this extent, Fig. 12b reports the computation times required by the two algorithms. Not surprisingly, the time needed to retrieve a solution is proportional to the size of the scenario. However, we can note that AN-JCDME is able to consistently reduce the computation time compared to JCDME.[6] Finally, Fig. 12c reports the values of γ for the two algorithms. In particular, the figure reports the best value of γ minimizing the total energy consumption for JCDME and the average value of γ computed by AN-JCDME. Three considerations hold in this case. First, the value of γ is lower than one for most scenarios (as expected). Second, the γ value is not fixed, but it tends to change when the size of the scenario is varied. Third, the estimation of the γ parameter performed by AN-JCDME is always close to the JCDME strategy.

[6]We point out that the computation times of JCDME can be potentially decreased by running in parallel the optimization problems, each of them with a given value of γ. However, even in this case, JCDME requires more computation resources compared to AN-JCDME.

VII. Related work

The topic of virtual resources allocation in a cloud scenario has gained growing interest over the last few years. The available literature ranges from the joint problem of distributing the requests and content over public and private cloud infrastructure [29] to the problem of VMs allocation in traditional cloud systems. One of the most significant works in this latter filed, which is closer to the area of interest of the present paper is [3], in which authors define the allocation problem and propose different heuristics for its solution. Generally, VMs allocation considers the placement of new VMs and the migration of existing VMs when the utilization of the physical servers exceeds a specific threshold; in other words, the threshold condition occurs when the risk of SLA violations is high. In addition, in [6] authors propose a self-organizing and distributed approach for the consolidation of VMs, which is based on two resources (namely CPU and RAM). A recent study [30] proposes an original approach to VMs allocation where physical machines are self-organized in a hypercube overlay network to improve energy-efficiency and scalability of the solution, while in [31] authors present two greedy approximation algorithms for energy-aware VMs placement in a multi-tenant cloud scenario. However, the VMs allocation process in these studies [3], [6], [30], [31] does not take into account the network data exchanges between VMs, focusing mainly on computational requirements with the aim of (efficiently) minimizing the number of turned on physical servers.

A VMs allocation model taking into account the network-related costs is proposed in [8]. However, this study does not model the cost of VMs migration and re-computes the whole allocation from scratch every time the model is solved, thus resulting in a large number of VMs migrations. In a similar way, the main goal of [5], [24] is a long-term VMs allocation that does not explicitly consider the cost of migration. However, it is worth to note that [5] proposes a modular approach to the problem of VMs allocation that is adapted to our problem to improve the scalability of the VM allocation (as shown at the end of Section VI).

Other studies are more focused on providing a detailed model for describing the impact of live VMs migration, including the management of dirty pages in memory [32]. This feature could be easily added to our model. However, in this paper, we preferred to drop the issue of dirty pages as the datasets in our possession contain little information about the rate of a page writes on the considered VMs. Other interesting studies dealing with VMs migration are [9], [33], which propose an online algorithm considering both computational demands and migration costs. More in detail, the authors of [9] follow a dynamic programming approach where the previous VMs allocation is taken into account as the basis for the future allocation solution. However, the objective function of the proposed model is quite straightforward from an energy point of view, as the model simply weighs the number of servers turned on against the number of migrations, without providing a detailed model for energy consumption. Similarly, in [33] a genetic algorithm is used to optimize a *consolidation score* which is basically a linear combination of a number of migrations and energy consumption. Hence, we can conclude that [9], [33] rely on externally-controlled weights to merge the sub-objectives of reducing energy consumption and limiting migrations, while we propose a more robust approach that does not need any tuning. Moreover, both studies focus on a traditional cloud DC, where the VMs are the main focus of the migrations and the network infrastructure has no programmable capability. Our study aims to tackle the more modern view of an SDDC-based architecture, where the network elements are programmable

and/or virtualized.

Another aspect that may emerge is the problem of fragmentation of resources during the allocation process. This problem is not strictly specific to our proposal, but it is rather inherently related to any multi-dimensional bin-packing problem. We experienced this effect first-hand in [5]. However, in the considered experimental scenario, we found that the CPU utilization is the dominant resource, to the point that the relaxation of constraints such as Eq. 8 for network utilization and Eq. 9 for memory leads to almost identical VEs placement in most time slots (although there are a few time slots where also these constraints affect the solution). Hence, in our experiments, simplifying the multi-dimensional bin-packing into a mono-dimensional problem has a limited impact. Even in the mono-dimensional problem, we may have fragmentation, but, once again, this is a well-known effect already discussed in previous works (e.g., [5]). We decided not to delve into the details of this effect in the paper exactly for this reason: it is a well-known effect from a theoretical point of view [34], but in most data centers it is not so evident [35].

The idea of virtualizing network functionalities has been proposed in the literature by different works, such as [36], [37]. More in details, the authors of [37] propose a framework for virtualizing network functions. This approach may be also applied to the SDDC architecture considered in this paper. Another interesting solution is the exploitation of *virtual router (VRs)*, which can be either a VMs hosting an IP routing software [38], [39] or a specific snippet of firmware code for a virtualization-enabled router. Some studies for VR migrations are focused on a geographical scenario (see e.g. [16], [18]), proposing a network model of cycle-stationary changes in the network patterns. We adopt the vision of cycle-stationarity in traffic models, but, as suggested in [38], we focus on its application to a scenario where VRs can be moved within the DC rather than between different DCs to better support the dynamic environment of cloud computing. The combination of migrating seamlessly any type of virtual element (i.e., VM or VR) throughout the data center is a qualifying original point of our study.

Focusing on the energy-model for VMs in cloud DCs, the linear correlation between computational load on the physical servers and power consumption (which is the model actually used in this paper) is the most common approach [3], [40], [41]. Furthermore, research focusing on the reduction of network-related energy consumption has been mainly focused on traditional network architectures. Proposals such as [9] represent the first example of network-aware VMs allocation exploiting the idea of placing on the same physical server VMs that have a significant data exchange. Another group of works focused on energy consumption exploit models to capture network-related costs. For example, [19] introduces a linear power model for the link energy consumption, while [42] presents a detailed energy model for the router power consumption. Finally, in [43] Yi *et al.* present a solution to consolidate traffic into few switches in order to minimize energy consumption in DCs based on a fat-tree network topology. However, the proposed solution is dependent on the specific network architecture of the cloud DC.

VIII. Conclusions and Future Work

We have faced the problem of jointly considering the computing energy, the energy due to data transmission, and the energy due to migrations in an SDDC composed of VEs and physical servers. After optimally formulating the JCDME problem of minimizing the total energy consumed for each TS, we have considered the impact of VEs

migrations over multiple TSs. Since a migration is affecting the energy consumed by the servers over multiple TSs, it is of mandatory importance to properly take this term into account by introducing a weight parameter $\gamma(t)$. We have then proposed a set of strategies, namely A-JCDME, AS-JCDME and AN-JCDME, to adaptively and automatically estimate $\gamma(t)$ over time. Results, obtained over different sets of representative scenarios, show that A-JCDME performs better than MUA (i.e., a migration un-aware strategy previously proposed in the literature). In addition, we have shown that AS-JCDME performs close to a solution in which the weight parameter is obtained from a brute-force analysis (i.e., JCDME with fixed γ). Furthermore, we have shown that AN-JCDME is able to efficiently manage the energy consumption, by dynamically (and automatically) tuning the γ parameter. This allows AN-JCDME to save an additional 7% of energy in the basic scenario. Finally, we evaluated the scalability of the adaptive strategies with respect to the size of the scenario, showing that AN-JCDME is able to save an amount of energy comparable to the best setting of JCDME, but with a substantial decrease in the required computation time.

As future work, we plan to extend our analysis to consider the possibility of managing the VEs across a set of geographically distributed DCs. In addition, we plan to study the impact of further decreasing the computation times of the AN-JCDME strategy, by considering a relaxed version of the problem.

ACKNOWLEDGMENT

The authors acknowledge the support of the University of Modena and Reggio Emilia through the project *S^2C: Secure and Scalable Cloud*.

REFERENCES

[1] C. Canali, R. Lancellotti, and M. Shojafar, "A Computation- and Network-Aware Energy Optimization Model for Virtual Machines Allocation," in *Proc. of 7th International Conference on Cloud Computing and Services Science (CLOSER'17), Porto, Portugal*, Apr. 2017.

[2] *Americas Data Centers Are Wasting Huge Amounts of Energy*. NRDC Brief, Available at https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IP.pdf, last accessed on 22th May 2017.

[3] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.

[4] A. Beloglazov and R. Buyya, "Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[5] C. Canali and R. Lancellotti, "Exploiting Classes of Virtual Machines for Scalable IaaS Cloud Management," in *Proc. of the 4th Symposium on Network Cloud Computing and Applications (NCCA), Munich, Germany*, Jun. 2015.

[6] C. Mastroianni, M. Meo, and G. Papuzzo, "Probabilistic Consolidation of Virtual Machines in Self-Organizing Cloud Data Centers," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 215–228, Jun. 2013.

[7] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, Dec. 2008.

[8] D. Huang, D. Yang, H. Zhang, and L. Wu, "Energy-aware virtual machine placement in data centers," in *Proc. of IEEE Global Communications Conference (GLOBECOM), Anaheim, USA*, Dec. 2012.

[9] A. Marotta and S. Avallone, "A Simulated Annealing Based Approach for Power Efficient Virtual Machines Consolidation," in *Proc. of 8th International Conference on Cloud Computing (CLOUD), New York, USA*, Jun. 2015.

[10] "Research and Markets. Software-Defined Data Center (SDDC) Market Global Forecast to 2021," June 2016. www.researchandmarkets.com/research/grj2gz/softwaredefined last accessed on 5th June 2017.

[11] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *IEEE Internet Computing*, vol. 17, no. 2, pp. 20–27, 2013.

[12] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, 2008.

[13] *OpenFlow Archive*. Available at http://archive.openflow.org/wp/learnmore/, last accessed on 22th May 2017.

[14] C. Canali and R. Lancellotti, "Automated Clustering of Virtual Machines based on Correlation of Resource Usage," *Communications Software and Systems*, vol. 8, no. 4, pp. 102 – 110, Dec. 2012.

[15] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "Research Challenges for Traffic Engineering in Software Defined Networks," *IEEE Network*, vol. 30, no. 3, pp. 52–58, May 2016.

[16] V. Eramo, E. Miucci, and M. Ammar, "Study of reconfiguration cost and energy aware vne policies in cycle-stationary traffic scenarios," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1281–1297, 2016.

[17] A. Sang and S. qi Li, "A predictability analysis of network traffic," *Computer Networks*, vol. 39, no. 4, pp. 329 – 345, 2002.

[18] V. Eramo, M. Ammar, and F. G. Lavacca, "Migration energy aware reconfigurations of virtual network function instances in nfv architectures," *IEEE Access*, vol. 5, pp. 4927–4938, 2017.

[19] L. Chiaraviglio, D. Ciullo, M. Mellia, and M. Meo, "Modeling sleep mode gains in energy-aware networks," *Computer Networks*, vol. 57, no. 15, pp. 3051–3066, 2013.

[20] T. Wood, P. J. Shenoy, A. Venkataramani, M. S. Yousif, *et al.*, "Black-box and gray-box strategies for virtual machine migration.," in *Proc. of 4th Usenix Symposium on Networked System Design and Implementation, Cambridge, USA*, pp. 17–17, April 2007.

[21] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proc. of 2nd USENIX Symposium on Networked Systems Design and Implementation, Anaheim, USA*, April 2005.

[22] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," *Computer Networks*, vol. 57, no. 1, pp. 179 – 196, 2013.

[23] T. J. Ypma, "Historical development of the newton–raphson method," *SIAM review*, vol. 37, no. 4, pp. 531–551, 1995.

[24] C. Canali and R. Lancellotti, "Scalable and automatic virtual machines placement based on behavioral similarities," *Computing*, vol. 6, no. 99, pp. 575–595, June 2017.

[25] *EnergyStar*. Available at https://www.energystar.gov/index.cfm?c=archives.en\-ter\-prise_servers, last accessed on 5th June 2017.

[26] *Cisco Data Sheet Switching Infrastructure*. Available at http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-2960-x-series-switches/data_sheet_c78-728232.html, last accessed on 5th June 2017.

[27] *Cisco Power Data Sheet*. Available at http://blogs.cisco.com/enterprise/reduce-switch-power-consumption-by-up-to-80, last accessed on 5th June 2017.

[28] *Knitro Website*. Available at https://www.artelys.com/en/optimization-tools/knitro, last accessed on 5th June 2017.

[29] X. Qiu, H. Li, C. Wu, Z. Li, and F. C. M. Lau, "Cost-minimizing dynamic migration of content distribution services into hybrid clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, pp. 3330–3345, Dec 2015.

[30] M. Pantazoglou, G. Tzortzakis, and A. Delis, "Decentralized and Energy-Efficient Workload Management in Enterprise Clouds," *IEEE Transactions on Cloud Computing*, vol. 4, no. 2, pp. 196–209, 2016.

[31] X. Dai, J. M. Wang, and B. Bensaou, "Energy-Efficient Virtual Machines Scheduling in Multi-Tenant Data Centers," *IEEE Transactions on Cloud Computing*, vol. 4, no. 2, pp. 210–221, 2016.

[32] H. Liu, H. Jin, C.-Z. Xu, and X. Liao, "Performance and energy modeling for live migration of virtual machines," *Cluster Computing*, vol. 16, pp. 249–264, Jun 2013.

[33] Q. Wu, F. Ishikawa, Q. Zhu, and Y. Xia, "Energy and migration cost-aware dynamic virtual machine consolidation in heterogeneous cloud datacenters," *IEEE Transactions on Services Computing*, vol. Pre Print, 2017.

[34] E. G. Coffman, M. R. Garey, and D. S. Johnson, *Approximation Algorithms for Bin-Packing — An Updated Survey*, pp. 49–106. Vienna: Springer Vienna, 1984.

[35] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella, "Multi-resource packing for cluster schedulers," *SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 455–466, Aug 2014.

[36] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[37] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *IEEE Internet Computing*, vol. 17, pp. 20–27, Mar. 2013.

[38] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, "Virtual routers on the move: Live router migration as a network-management primitive," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 231–242, Aug. 2008.

[39] D. M. F. Mattos and O. C. M. B. Duarte, "Xenflow: Seamless migration primitive and quality of service for virtual networks," in *Proc. of IEEE Global Communications Conference 2014 (Globecom), Austin, USA*, pp. 2326–2331, Dec. 2014.

[40] A. Verma, P. Ahuja, and A. Neogi, "pMapper: power and migration cost aware application placement in virtualized systems," in *Proc. of 9th International Middleware Conference (Middleware 2008), Leuven, Belgium*, pp. 243–264, Dec. 2008.

[41] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, vol. 60, no. 2, pp. 268–280, 2012.

[42] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power awareness in network design and routing," in *Proc. of 27th IEEE Conference on Computer Communications (INFOCOM), Phoenix, USA*, IEEE, 2008.

[43] Q. Yi and S. Singh, "Minimizing energy consumption of fat-tree data center networks," *SIGMETRICS Performance Evaluation Review*, vol. 42, pp. 67–72, Dec. 2014.