Power Consumption-Aware Virtual Machine Placement in Cloud Data Center

Giuseppe Portaluri, Davide Adami, Andrea Gabbrielli, Stefano Giordano, and Michele Pagano

***Abstract*—In this paper we present a set of power-aware dynamic allocators for Virtual Machines (VMs) in Cloud Data Centers (DCs) taking advantage of the Software Defined Networking (SDN) paradigm. Each VM request is characterized by four parameters: CPU, RAM, disk and bandwidth. We design the allocators in order to accept as many VM requests as possible, taking into account the power consumption of the network devices. In this paper, we introduce 10 different allocation strategies, and compare them with a baseline that consists in using the first available server (First Fit). The allocators differ in terms of allocation policy (Best Fit/Worst Fit), allocation strategy (Single/Multi objective optimization), and joint/disjoint selection of IT and network resources. For both joint and disjoint approaches, we evaluate the behavior of all possible pairs policy-strategy, varying the load of the DC and the number of VMs to be allocated. Moreover, the experimental results highlight that joint approaches outperform disjoint ones.**

***Index Terms*—Cloud Computing, Data Center, Software Defined Networking, Virtual Machine, Fuzzy Logic, Multi-Objective Optimization, Resources Allocation, Best Fit, Worst Fit, Pareto Front.**

I. Introduction

In the last few decades, Data Centers (DCs) have rapidly evolved not only in terms of hardware resources and services, but also from the architectural point of view. Indeed, many and various services are made available to users, from online storage to a variety of "apps". All these services require resources that are located "somewhere" in the Cloud. Users do not care where such resources are, but only that they are available when needed and with the desired quality and security levels. In this evolutionary process a key role is played by the Software Defined Networking (SDN) paradigm that leads to virtualize DCs, enabling customers to deploy secure, dynamic and highly distributed cloud infrastructure [1].

To satisfy the growing requirements of the users, adopting a flexible and dynamic DC architecture has become a major challenge. In this framework, the most critical issue to be addressed is represented by the efficient use of network and IT (i.e., storage and computational) resources for the allocation of virtual machines (VMs) [2]. Initially, the allocation of IT and network resources were addressed focusing on IT resources and assuming no limitation at the network layer. Really, data have to be exchanged between end-users and VMs (and sometimes among VMs) and this can lead to a significant performance degradation in case the network links are congested (or not properly used in case simple minimum-hop paths are selected).

In this work, we propose a novel approach able to tackle this critical issue within an SDN framework, taking advantage of one of SDN key features, i.e., the clean separation between control and data planes [1]. The data plane forwards packets using the flow tables computed by the control plane, while the control logic is separated and implemented in a controller that sets up the forwarding tables. In this way, switches implement only the data plane logic, with a significant reduction in terms of cost and complexity. Furthermore, the SDN paradigm enables control plane centralization and programmability, giving the possibility of implementing relatively complex control features, which can make use of real-time state information exchanged through the OpenFlow communication protocol [3].

This paper is an extension of our previous work [4] with two new contributions. First of all, we added a comparison between two versions of our allocators:

- the *disjoint allocator*, which performs a two step allocation, first choosing the server where to place the VM and then looking for the proper network path;
- the *joint allocator* that considers at the same time both computational and network requirements.

Moreover, in [4] we evaluated the behavior of the joint allocators when we saturate our DC by allocating as many VMs as possible; now, instead, we consider the behavior of all our allocators during the transient phase, showing some "snapshots" of the DC state and comparing the DC performance at different loads.

The rest of the paper is organized as follows: Section II presents a thorough review of the literature, Section III describes the behavior of our allocators, detailing all the policies and allocation strategies that we adopt, Section IV outlines the power model and the proposed solution to minimize the consumption, Section V shows our experiments and their results, and Section VI concludes the paper, pointing out some future researches.

II. Related works

A variety of published works focuses on the problem of VM allocation in Cloud DC. We classified the literature in three main parts: works that do not consider power consumption; power-aware allocators for group of VMs (i.e., *static* allocators); and power-aware allocators that allocates VMs as soon a new request is received (i.e., *dynamic* allocators).

A. Power-unaware allocators

In [5], authors present an optimization technique designed for VM allocation minimizing the costs and guaranteeing some performance levels. The allocator presented in [6] combines both Genetic Algorithms (GA) and Mixed Integer Linear

Programming (MILP) to optimize the allocation of VMs while it meets four different QoS requirements. In [7] authors propose a resource allocator for distributed clouds that is based on five approximation algorithms able to reduce inter-cloud and inter-rack network traffic, but they do not consider any computational requirement for VMs only constraining the number of VMs that could be allocated on each server. Authors apply in [8] the Hungarian method to optimize the resource penalty during each VM allocation.

B. Power-aware static allocators

Authors in [9] implement a power-aware VM allocator for groups of VMs using a modified version of the Multiple Knapsack problem. An approach based on Fuzzy Logic is presented in [10]. It performs the VM placement combining Genetic Algorithms and Fuzzy Logic, but it has a high completion time. Authors present in [11] a static power-efficient allocator for groups of tasks based on Multi-Objective Genetic Algorithms and Simulated Annealing. In [12], authors introduce a power-aware VM allocation procedure for DCs, but they neglect both the path allocation and the power consumption of the network. Authors in [13] present two exact algorithms for energy efficient VM scheduling. In [14], authors suggest a heuristic optimization algorithm based on particle swarm, but they suppose that the number of VM requests is much lower than the number of physical servers, and they disregard the power consumption of network devices. In [15], two exact algorithms for energy-efficient scheduling of VMs are compared, but all VMs have the same requirements, and the network resources are not taken into account.

C. Power-aware dynamic allocators

The approach, presented in [16], manages dynamic resource re-allocation in DCs using a multi-agent version of the fuzzy controller. Only CPU and RAM are taken into account, while disk and network requirements are neglected. Authors present a dynamic allocator in [17] that neglects the power consumption due to the networking devices. The allocator presented in [18] dynamically allocates VMs putting in sleep mode the underutilized servers. Finally, in [19], authors describe a dynamic VM allocator addressing the problem of energy-efficient task allocation in the system in the presence of a time-varying grid energy price and the unpredictability and time variation of provisioned power by renewable energy sources, but they neglect the path allocation phase.

Our proposal is novel with respect to the reviewed approaches, because it:

- takes into account both computational and network requirements;
- evaluates different allocation strategies either solving a single-objective or a multi-objective optimization;
- compares the disjoint and the joint allocation strategies;
- optimizes the power consumption of network devices.

III. Virtual Machine Allocation

The main goal of the IT Resource Allocators (ITRA) is to accept as many VM requests as possible, reducing at the same time the network power consumption. Each VM request is characterized by four parameters representing the peak utilization of CPU, RAM, disk and bandwidth. The server selection consists of the following steps:

1) Compute the candidate server list, i.e., the set of servers with enough IT resources to satisfy the request:
   a) if the list is empty, the request is rejected;
   b) otherwise, go to next step.
2) Select the policy between:
   a) *Multi Resource Best Fit* (BF) that strongly consolidates the system resource utilization choosing the server that has the least resources availability;
   b) *Multi Resource Worst Fit* (WF) that selects the server having the highest resources availability, so as to balance the load among all the available servers.
3) Select the *best* server according to one of the possible strategies:
   a) disjoint or joint Analytic ITRA (described in Section III-A);
   b) disjoint or joint Fuzzy ITRA (described in Section III-B);
   c) disjoint or joint Multi-Objective Dynamic Allocator (MODA) (described in Section III-C).

The joint allocation strategies consider at the same time both computational and network requirements to perform the allocation of VMs. Instead, disjoint strategies split the allocation procedure in two different steps:

1) choose the server where to allocate the VM evaluating only the computational requirements rejecting the request if no server is available;
2) taking into account the bandwidth requirement, find the minimum-cost path connecting the chosen server to the gateway rejecting the request if no path is available.

ITRA associates the minimum-cost network path with each available server when a new request comes, and it discards servers that do not have enough resource nor at least an available path. The cost of the path is computed as the amount of power that will be consumed by the new network flow. More clearly, ITRA allocates the network path minimizing the increment of power consumption of the network devices. We provide a more detailed view about the path allocation procedure in Section IV.

For the sake of clarity, we summarize the symbols used in Table I.

A. Analytic ITRA

Analytic ITRA (A-ITRA) computes for each candidate server the *A-ITRA Availability Index* (I_A) that takes into account the availability of IT resources. The joint version of the A-ITRA Availability Index is computed as follows:

$$I_A^s = \frac{1}{300}\left[\mathrm{CPU}_s + \mathrm{RAM}_s + \mathrm{DISK}_s\right] + \alpha\frac{\mathrm{PC}_s}{\mathrm{PCM}} \qquad (1)$$

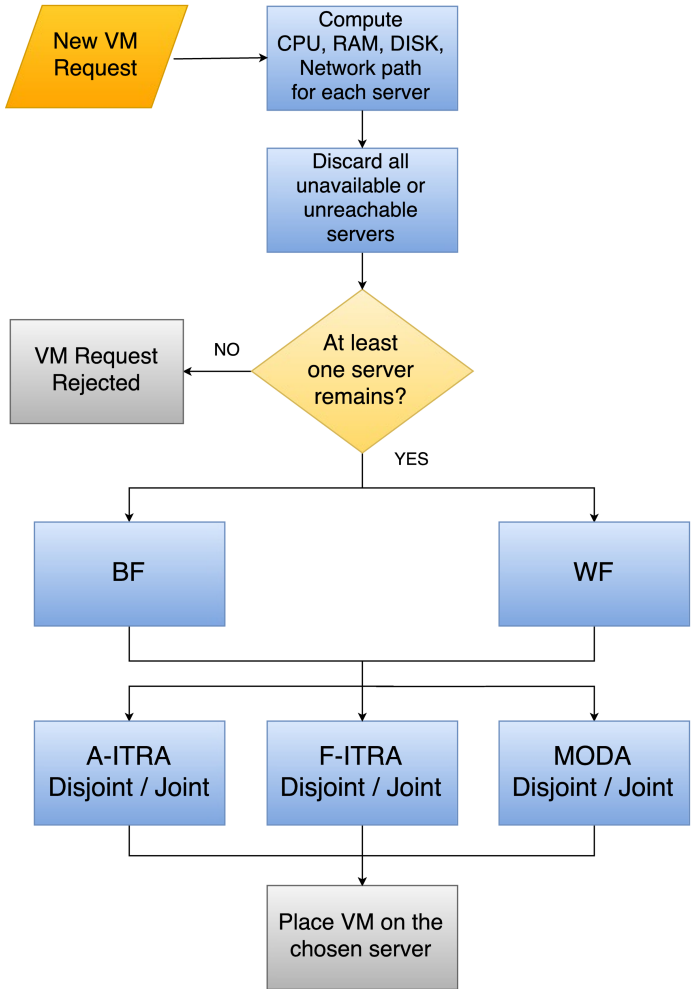

Fig. 1. IT Resource Allocator algorithm.

where

$$\alpha = \begin{cases} -1 & \text{if WF is adopted;} \\ 1 & \text{if BF is adopted.} \end{cases} \quad (2)$$

The first component of I_A represents the normalized availability of the system resources (by definition the three values range from 0 to 100) and the second one is the normalized path cost (i.e., $\frac{\text{PC}_s}{\text{PCM}}$). A-ITRA chooses the server minimizing I_A in case of BF, or the server maximizing I_A in case of WF. Note that different values of α are used since in both cases the path cost must be minimized.

When we adopt the disjoint A-ITRA, we calculate the Availability Index taking into account only the computational resources:

$$I_A^s = \frac{1}{300}\left[\text{CPU}_s + \text{RAM}_s + \text{DISK}_s\right]. \quad (3)$$

As for the joint version, disjoint A-ITRA first chooses the server minimizing or maximizing I_A^s when we respectively adopt BF or WF. Next, it computes the minimum-cost path towards the gateway, and associates it with the selected server.

B. Fuzzy IT Resource Allocator

Fuzzy Logic [20], [21] is a technique that deals with uncertain, imprecise, or qualitative information, as well as with precise information in systems which cannot be described by a formal and analytically tractable mathematical model. In Boolean logic, an element x can or cannot belong to a set $\mathcal{A}$ with a membership degree respectively equal to 1 or 0. Instead, in Fuzzy Logic, the membership degree of x to a fuzzy set $\mathcal{F}$ has a value in a continuous interval between 0 and 1. Fuzzy set theory allows an element to have partial membership degree in one or more fuzzy sets. This membership degree is obtained through membership functions that map elements into the interval [0, 1].

Fuzzy Logic can be used for the design of control systems, called Fuzzy Logic Controller (FLC). The core of the FLC [22], [23] is the inference engine, whose role is to apply the inference rules (IF-THEN rules) contained in the rule base. IF-THEN rules are made by premises and conclusions, and embody the system control strategies. Since fuzzy rules use fuzzy sets and their associated membership functions to describe system variables, two operations are necessary for translations between conventional and fuzzy values: fuzzification and defuzzification. The former maps input values into one or more fuzzy sets, the latter produces a single conventional value that best represents the inferred fuzzy values.

TABLE I
MODEL PARAMETERS

Parameter	Description
CPU_s	% of the s-th server free CPU after the placement of the VM.
RAM_s	% of the s-th server free space in RAM after the placement of the VM.
$DISK_s$	% of the s-th server free space in the storage after the placement of the VM.
PC_s	Cost value of the minimum-cost path from server s to the external gateway.
PCM	Sum of the costs of all the links in the network.
I_A^s	Availability index for the s-th server made by A-ITRA.
I_F^s	Availability index for the s-th server made by F-ITRA.
$f(\boldsymbol{x})$	Defuzzification method applied to the input vector $\boldsymbol{x}$.
N	Number of servers.

More specifically, our FLC includes the following modules (see [24] for more details):

- *Inputs fuzzification*: we defined four input fuzzy sets ("Not available", "Small", "Medium", and "Large") and their membership functions. Figure 2 shows an example: while RAM_s and $DISK_s$ are fuzzified in a single fuzzy set (i.e., "Medium" for RAM and "Large" for DISK), CPU belongs to two different sets with a certain probability (0.7 in "Small" and 0.3 in "Medium");
- *Inference engine*: we considered two different inference processes:
  1) Mandami (F-ITRA M), that uses the min operator;
  2) Sugeno [25] (F-ITRA S), that applies both linear or constant membership functions;

  and six output sets, "Empty", "Almost empty", "Medium", "Almost full", "Full" and "Not available", whose corresponding membership functions as depicted in Figure 3;
- *Defuzzification method*, denoted with f, that computes the center of gravity as defuzzification algorithm for the aggregated fuzzy subset.

Similarly to A-ITRA, we use the FLC to compute the *F-ITRA Availability Index (I_F)* for each server. We define the joint and disjoint F-ITRA Availability Index in (4) and (5) respectively assuming as values for α the ones reported in (2). Analogously to A-ITRA, both disjoint and joint F-ITRA choose the server maximizing or minimizing I_F^s each in order when WF or BF is adopted. Moreover, disjoint F-ITRA computes the minimum-cost path only when the server is selected.

$$I_F^s = f(\mathrm{CPU}_s, \mathrm{RAM}_s, \mathrm{DISK}_s) + \alpha \frac{\mathrm{PC}_s}{\mathrm{PCM}} \tag{4}$$

$$I_F^s = f(\mathrm{CPU}_s, \mathrm{RAM}_s, \mathrm{DISK}_s) \tag{5}$$

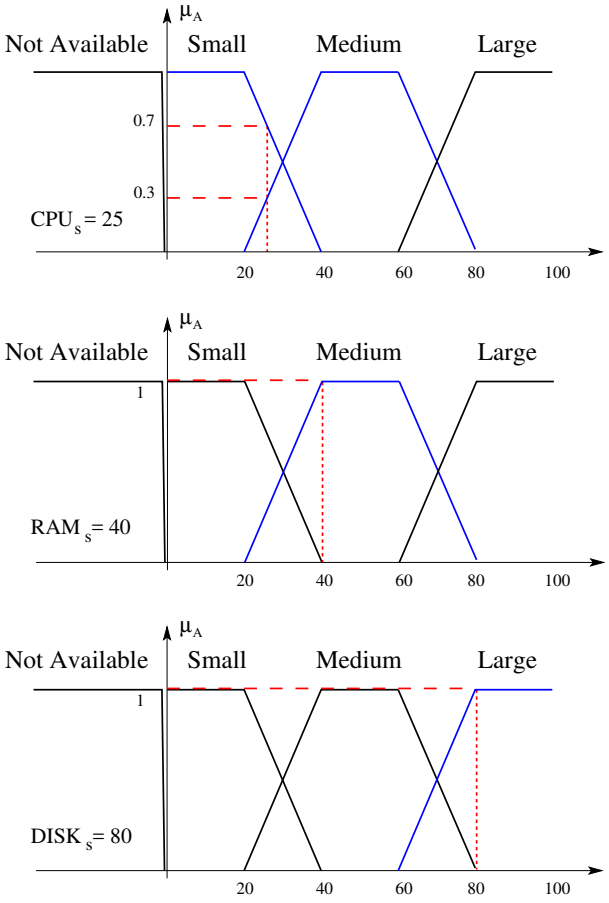

Fig. 2. Inputs membership functions

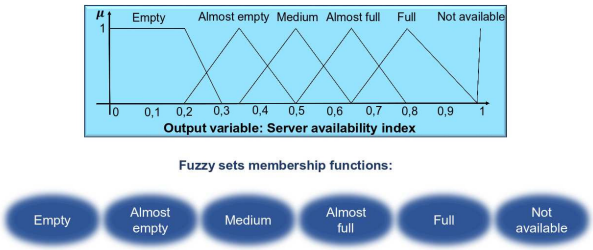

Fig. 3. Inference process outputs fuzzy sets and their membership functions.

C. Multi-Objective Dynamic Allocator (MODA)

MODA allocates VMs using a technique based on the multi-objective optimization: available resources are the objectives that should be optimized all together. Solutions are not comparable among each others in a multi-objective computation, so we use the concept of Pareto front [26] to deal with the incommensurability of vector solutions.

If we consider two solutions s and t for a minimization problem, and we associate a set of m objectives denoted with $f_1, f_2, \ldots, f_m$, we say that s dominates t if an integer value d exists such that:

$$\begin{cases} f_d(s) < f_d(t) & 1 \le d \le m; \\ f_l(s) \le f_l(t) & 1 \le l \le m \wedge l \ne d. \end{cases} \tag{6}$$

Solutions s and t are *non-dominated* if both s does not dominate t and viceversa, and the set including all the non-dominated solutions is called *Pareto front*. Within a Pareto front, the solution that optimizes all the objectives at the same time is called *ideal vector* [27].

MODA computes in multiple steps the allocation procedure; when a new VM should be allocated, MODA creates a list of servers able to fit the request. The difference between joint and disjoint MODA concerns only the number of objectives that should be optimized:

- disjoint MODA considers only CPU_s, RAM_s and DISK_s, and it computes the minimum-cost path after the phase of server selection;
- joint MODA considers CPU_s, RAM_s, DISK_s and PC_s, and it associates the minimum-cost path with each server during the server selection phase.

Then, MODA computes the Pareto front for the current allocation, and finally it chooses one of the possible allocations according to the applied policy (i.e., BF or WF) and one of the two possible strategies:

1) Choosing randomly one solution within the Pareto front (MODA-R);
2) Normalizing all the objectives and taking the solution that has the minimum distance from the ideal vector (MODA-D).

The joint MODA solves the minimization problem described in formula (7) when it adopts the BF policy:

$$\begin{array}{lll} \text{minimize} & \min\limits_{1\le s\le N}\{\text{CPU}_s,\ \text{RAM}_s,\ \text{DISK}_s,\ \text{PC}_s\} & \\ \text{subject to:} & 0 \le \text{CPU}_s \le 100 & 1 \le s \le N \\ & 0 \le \text{RAM}_s \le 100 & 1 \le s \le N \\ & 0 \le \text{Disk}_s \le 100 & 1 \le s \le N \\ & 0 \le \text{PC}_s \le \text{PCM} & 1 \le s \le N \end{array} \tag{7}$$

The allocator maximizes instead the computing resources in case we set the WF policy, but it still minimizes the path cost:

$$\begin{array}{lll} \text{minimize} & \min\limits_{1\le s\le N}\{\text{-CPU}_s,\ \text{-RAM}_s,\ \text{-DISK}_s,\ \text{PC}_s\} & \\ \text{subject to:} & 0 \le \text{CPU}_s \le 100 & 1 \le s \le N \\ & 0 \le \text{RAM}_s \le 100 & 1 \le s \le N \\ & 0 \le \text{Disk}_s \le 100 & 1 \le s \le N \\ & 0 \le \text{PC}_s \le \text{PCM} & 1 \le s \le N \end{array} \tag{8}$$

As described above, disjoint MODA solves the same optimization problems without considering *PC* and associating the minimum-cost path only with the selected server.

IV. Power Consumption and Network Path Computation

We adopt the power consumption model for switches presented in [28], whose parameters are summarized in Table II.

TABLE II
NOTATION OF THE PARAMETERS RELATED TO POWER CONSUMPTION

Parameter	Description
P_{switch}	the average power consumption of a single switch.
$P_{chassis}$	the constant power consumed by chassis.
n	the number of line cards in the switch.
$P_{linecard}$	the fixed amount of power consumed by line cards.
$P_{load}(i,r)$	the power consumed by the i-th port while transmitting at bit-rate r.
$P_{max}(i)$	$P_{load}(i,r)$ while transmitting at the maximum bit-rate.
P_{fix}	$P_{chassis} + n * P_{linecard}$.
$B(i)$	the bandwidth transmitted by the i-th line card.
$B_{max}(i)$	the maximum capacity of the i-th line card.
R	the bandwidth request of a new (i.e., not allocated) VM.
w	the link weight.

The power model for switches is:

$$P_{switch} = P_{chassis} + n * P_{linecard} + \sum_{i=1}^{n} P_{load}(i,r). \tag{9}$$

We express an equivalent (and simplified) formulation of (9) by grouping all the fixed terms and taking into account that the load-dependent power component. The power consumption is linearly proportional to the transmission rate, as the authors showed in [29], and we assumed that the line cards of a link are active even when no packet is transmitted:

$$P_{switch} = P_{fix} + \sum_{i=1}^{n} \frac{B(i)}{B_{max}(i)} * P_{max}(i). \tag{10}$$

We might estimate the utilization values retrieving periodically the number of bytes transmitted by each link from the SDN controller; then, we could be able to compute the average bandwidth during the considered period [30]. After pruning the links without enough available bandwidth, i.e.,

$$B(i) + R \le B_{max}(i) \tag{11}$$

we used the Dijkstra's algorithm for minimizing power consumption; for each line card, we set as weight the increment of the power cost due to the transmission of the new flow.

$$w_i = \frac{R}{B_{max}(i)} * P_{max}(i). \tag{12}$$

In this analysis we considered only the power consumption of network devices and we neglected the server consumption because we supposed all the servers to be equal, and we applied them the same consumption profile.

V. Experiments and Results

We describe in this section two sets of experiments. In Section V-B, we focus on power consumption and path allocation algorithm comparing our power-aware path allocation with the classical equal cost routing. Then, we evaluate the power consumption of the networking devices. In the second experiment, we analyze the performance of our allocators in terms of accepted VMs; we describe this experiment in Section V-C. At the end, we show the difference between the joint and disjoint ITRAs.

TABLE III
VM REQUEST PARAMETERS

Parameter	Value
CPU percentage RAM percentage Disk percentage	Uniformly distributed in [10; 30]
Bandwidth	Uniformly distributed in [100; 300]Mbps

TABLE IV
POWER CONSUMPTION AND NETWORK PARAMETERS

Parameter	Value
Access switch P_{fix}	160W
Access switch P_{max}	200W
Aggregation switch P_{fix} Core switch P_{fix}	2000W
Aggregation switch P_{max} Core switch P_{max}	2500W
Server-Access link bw	1Gbps
Access-Aggregation link bw Aggregation-Core link bw	10Gbps

In our experiments we use as baseline the simplest possible allocation strategy named *First Fit* (FF) that allocates VMs one by one on the first available server, and we execute 30 independent runs for each experiment.

A. Simulation scenario

The simulation scenario consists of 16 servers interconnected through a three-tier fat tree network topology [31], which is one of the most widely adopted topologies within DCs [26]. Three-tier fat tree is structured in three different layers: *access*, *aggregation*, and *core*. Access switch provides connection to servers, and it is connected to a pair of aggregation switches. Access switches provide connection to servers and are connected to a pair of aggregation switches. Servers and access switches connected to the same couple of aggregation switches belong to the same *pod*. Core switches guarantee connectivity between all the pods and the external gateway. All the servers had the same hardware and they were empty at the beginning.

In our experiments, we supposed that VMs remained active until the end of the simulation. As already mentioned, VMs are characterized by CPU, RAM, disk and bandwidth. The values of such parameters are generated as described in Table III, while the parameters related to power consumption and network equipment are reported in Table IV.

B. Power-aware network path allocation

In this experiment, we evaluate the power efficiency of our path allocation strategy. The only differences with respect to the parameters reported in Table IV are the two power consumption profiles for aggregation and core switches. Within the same pod, the two aggregation switches have different power profiles and both of them are connected to the same number of less consuming and more consuming core switches. The two profiles for aggregation and core switches are shown in Table V. We generate 60 requests with the parameters described in Table III, and we allocate them using

TABLE V
AGGREGATION AND CORE SWITCH POWER PROFILES

Profile	P_{fix} [W]	P_{max} [W]
First profile	2000	2500
Second profile	2500	5000

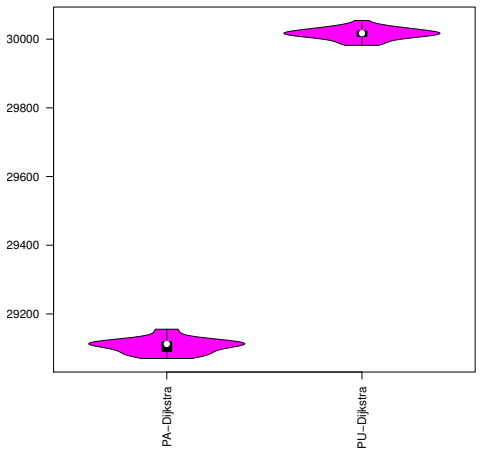

Fig. 4. Power consumption after 60VM requests

the FF policy. More specifically, we adopted FF since we were interested only in evaluating the behavior of the network path allocator regardless the server placement strategy.

We use violin plots [32] to show the main statistical information for our allocators. Violin plots represent the first and the third quartiles as the extreme points of the bold line and the median using a white dot, and they trace the probability density function symmetrically at both sides of the plot. We compare our power-aware network allocation strategy (PA-Dijkstra) with the classic power-unaware equal cost path (PU-Dijkstra) in Figure 4. The PA-Dijkstra saves on average around 1 kW with respect to the PU-Dijkstra. The amount of power saved corresponds to the 3% of the total power consumption due to all the switches.

C. Performance evaluation of the allocation strategies

We carried out two different experiments evaluating:

- the "steady state" of the system after 80 VMs, when on average all the allocators saturate the available resources, and new VM requests may not be allocated;
- the "transient state" of the system after 20, 40, and 60 VM requests.

In both the experiments we compared our joint with disjoint allocators evaluating them in terms of accepted/rejected requests. We used Matlab Simulink to perform our simulations, doing 30 independent runs for each experiment.

As described above, we generated 80 VM requests in the first experiment. Our purpose is to evaluate the steady state of the system reducing the availability of many resources as possible. In Figures 5 and 6 we show the results respectively for joint and disjoint allocators using as metric the number of rejected requests and the FF policy as comparison term. Furthermore, we show in Table VI all the numerical values related to each policy and allocation strategy. Joint

TABLE VI
VMs ALLOCATED BY DISJOINT ALLOCATORS AFTER 80 REQUESTS

	Joint		Disjoint	
	Mean	Std Dev	Mean	Std Dev
A-ITRA BF	66.38	1.76	63.10	11.56
A-ITRA WF	66.00	1.46	65.38	1.78
F-ITRA M BF	66.34	1.70	63.03	11.61
F-ITRA M WF	66.07	1.25	65.48	1.30
F-ITRA S BF	66.34	1.70	61.66	12.58
F-ITRA S WF	65.90	1.26	65.76	1.48
MODA-D BF	66.55	1.62	63.10	11.56
MODA-D WF	66.34	1.37	65.38	1.82
MODA-R BF	66.34	1.86	62.28	12.46
MODA-R WF	65.93	1.60	65.38	1.52
FF	66.41	1.64	61.72	12.70

TABLE VII
VMs ALLOCATED BY DISJOINT ALLOCATORS AFTER 60 REQUESTS

	Joint		Disjoint	
	Mean	Std Dev	Mean	Std Dev
A-ITRA BF	60	0	52.62	10.38
A-ITRA WF	59.86	0.35	58.93	1.56
F-ITRA M BF	60	0	52.66	10.34
F-ITRA M WF	59.86	0.35	58.79	1.42
F-ITRA S BF	60	0	51.45	11.28
F-ITRA S WF	59.93	0.26	58.83	1.39
MODA-D BF	60	0	52.69	10.34
MODA-D WF	59.97	0.19	58.83	1.47
MODA-R BF	60	0	52.10	11.15
MODA-R WF	59.97	0.19	59.14	1.03
FF	60	0	51.62	11.23

allocators reach on average best results compared to the disjoint approaches, and they also guarantee a lower standard deviation. We underlined the two best possible mean and standard variation values among all possible combinations. All joint allocators are able to satisfy on average 66 requests regardless the adopted policy or strategy. On the contrary, disjoint allocators decrease their effectiveness when BF policy is adopted because the consolidation procedure quickly saturates the bandwidth availability producing congestion at the higher layers of the topology. Moreover, disjoint BF allocators have a very high standard deviation because sometimes the network path computation (performed only when the server is already chosen) may fail, and the request may be blocked even if there is enough available computational resources. Lastly, all the BF allocators have a higher standard variation compared to the corresponding WF ones, so WF allocators are usually more stable. In the second experiment, we focused on the transient phase inspecting the evolution of the system after steps of 20 allocation requests. We show the number of rejection after 20 and 40 VM requests respectively in Figures 7 and 8 for the disjoint allocators. It is worth noting that joint allocators do not reject any VM until the 40th request. After the 60th request, BF joint allocators are still able to allocate all the requests unlike disjoint and WF joint allocators. We show the violin plots in Figures 9 and 10 and the numerical value in Table VII.

Summarizing all the results, joint allocators perform always equally to or better than the corresponding disjoint approach. In the low load phase (i.e., when the DC is not overloaded), BF joint allocators allocates all the requests, while WF ones

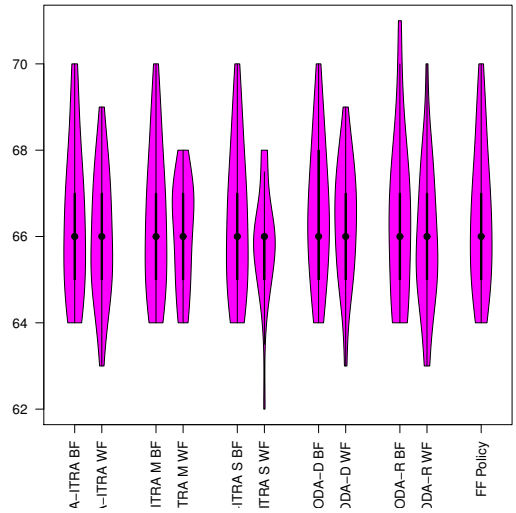

Fig. 5. VMs allocated by joint allocators

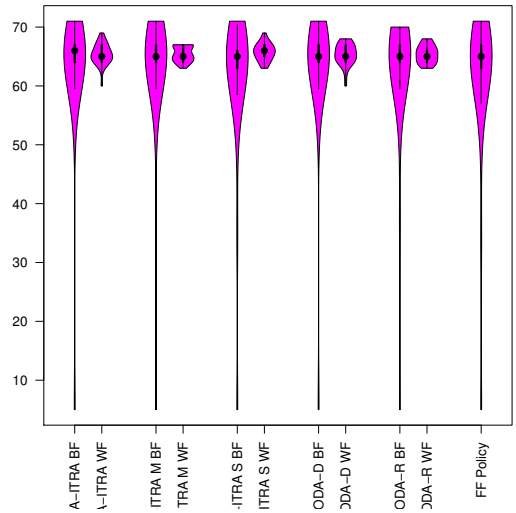

Fig. 6. VMs allocated by disjoint allocators

perform better during the high load phase. For what concerns only disjoint allocators, WF policies reach results similar to the joint ones, so we suggest to adopt WF policy in this case choosing the allocation strategy according to the parameter that preferably should be optimized (i.e., mean or standard deviation of the allocated requests). In our scenario, all the joint allocation strategies using the same policy (i.e., BF or WF) obtained similar results with respect to the mean and the standard deviation. The A-ITRA approach is the least computational expensive, while the others increase their complexity obtaining little improvements. However, all the joint allocation strategies are valid alternatives to be adopted in a real system. For what concerns the two policies, BF groups all the VMs in the least number of servers, so it may reduce the number of active computing devices; on the other hand, BF quickly saturates the access network links. The WF policy uses all the servers in the transient phase since it is

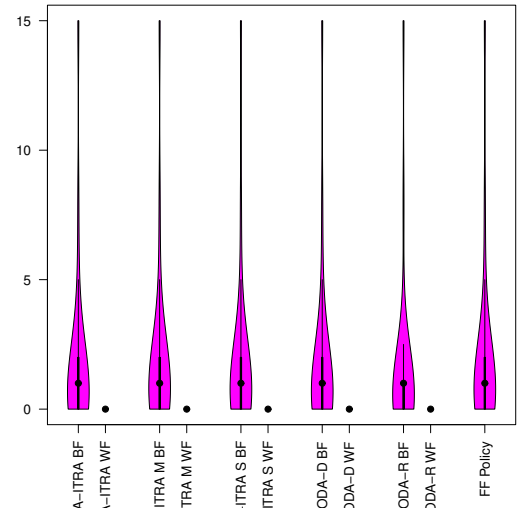

Fig. 7. VMs rejected by the network after the 20th request

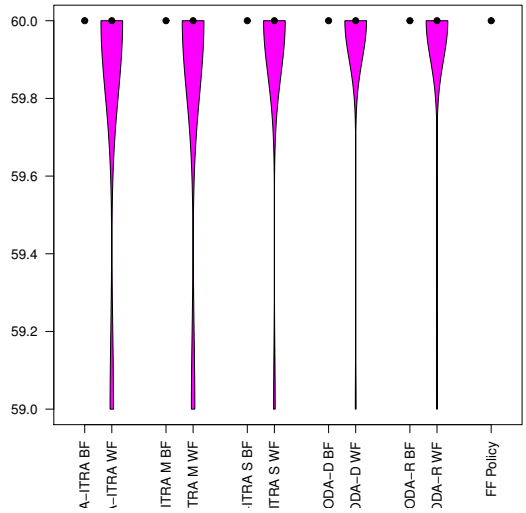

Fig. 9. Snapshot of the joint allocators after 60 requests

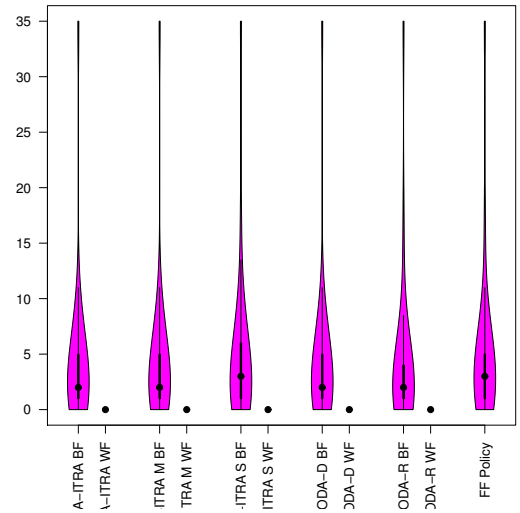

Fig. 8. VMs rejected by the network after the 40th request

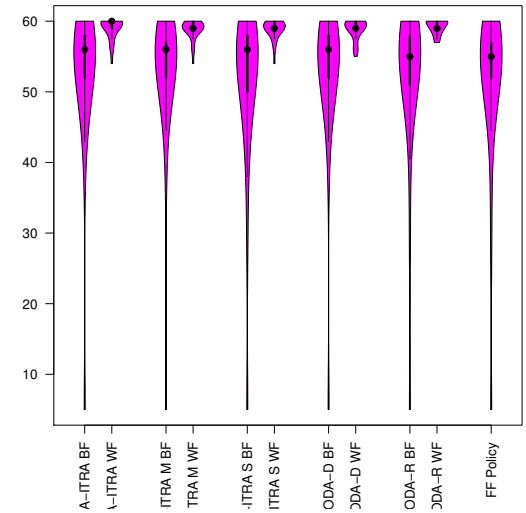

Fig. 10. Snapshot of the disjoint allocators after 60 requests

a load balancing policy and it less congests the network. We suggest to adopt BF policy with a high oversubscription rate between the network layers or when servers could be put in sleep mode for power saving reason. On the contrary, we strongly suggest to use the WF policy if the DC provider prefers to leave all computing resources active and running, or the oversubscription rate is low.

VI. Conclusion

In this paper we presented a set of power-aware dynamic allocators for VMs that take into account CPU, RAM, disk and bandwidth. We allocated all the network flows on the the most power-efficient paths using a modified version of the Dijkstra algorithm. We implemented two policies (i.e BF and WF) and 10 allocation strategies, and we evaluated the performance of our joint and disjoint allocators in terms of number of accepted requests. Simulation results highlighted that the modified version of Dijkstra allocated the network flows reduced the consumption of 1kW (about 3.3% of the total previous consumption) with respect to the classical equal cost path allocation. For what regards the performance experiments, joint allocators performed better than the disjoint ones reaching a higher average acceptance rate and a lower standard deviation. Results showed also a different behavior between BF and WF allocators especially when we adopted disjoint allocators: disjoint BF allocators reached higher values of standard deviation because of the two-step allocation procedure that may reject VM requests during the second phase due to the lack of network resources. Concerning the allocation strategies, the joint version of BF MODA allocates more VMs on average while the joint WF F-ITRA achieves the lowest standard deviation. Lastly, A-ITRA represents the best choice when the less CPU-intensive strategy is needed.

REFERENCES

[1] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, 2013. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/6658648/

[2] M. Gharbaoui, B. Martini, D. Adami, G. Antichi, S. Giordano, and P. Castoldi, "On virtualization-aware traffic engineering in OpenFlow Data Centers networks," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE, 2014, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/6838318/

[3] A. Lara, A. Kolasani, and B. Ramamurthy, "Network Innovation using OpenFlow: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 493–512, 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6587999/

[4] G. Portaluri, D. Adami, A. Gabbrielli, S. Giordano, and M. Pagano, "Power consumption-aware virtual machine allocation in cloud data center," in *2016 IEEE Globecom Workshops (GC Wkshps)*, Dec 2016, pp. 1–6.

[5] J. Díaz, J. Entrialgo, M. García, J. García, and D. García, "Optimal allocation of virtual machines in multi-cloud environments with reserved and on-demand pricing," *Future Generation Computer Systems*, vol. 71, pp. 129–144, 2017.

[6] T. Guérout, Y. Gaoua, C. Artigues, G. Da Costa, P. Lopez, and T. Monteil, "Mixed integer linear programming for quality of service optimization in clouds," *Future Generation Computer Systems*, vol. 71, pp. 1–17, 2017.

[7] M. Alicherry and T. V. Lakshman, "Network aware resource allocation in distributed clouds," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 963–971.

[8] A. Vichare, Z. P. Gomes, N. Fernandes, and F. Cardoza, "Cloud computing using OCRP and virtual machines for dynamic allocation of resources," in *Technologies for sustainable development (ICTSD), 2015 International Conference on*. IEEE, 2015, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7095898/

[9] S. R. M. Amarante, F. M. Roberto, A. R. Cardoso, and J. Celestino, "Using the Multiple Knapsack Problem to Model the Problem of Virtual Machine Allocation in Cloud Computing." IEEE, Dec. 2013, pp. 476–483. [Online]. Available: http://ieeexplore.ieee.org/document/6755257/

[10] J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, Dec 2010, pp. 179–188.

[11] G. Portaluri and S. Giordano, "Power efficient resource allocation in cloud computing data centers using multi-objective genetic algorithms and simulated annealing," 2015, pp. 319–321.

[12] J. V. Wang, C. T. Cheng, and C. K. Tse, "A power and thermal-aware virtual machine allocation mechanism for cloud data centers," in *2015 IEEE International Conference on Communication Workshop (ICCW)*, June 2015, pp. 2850–2855.

[13] C. Ghribi, M. Hadji, and D. Zeghlache, "Energy efficient vm scheduling for cloud data centers: Exact allocation and migration algorithms," in *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, May 2013, pp. 671–678.

[14] P. Aruna and S. Vasantha, "A particle swarm optimization algorithm for power-aware virtual machine allocation," in *2015 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, July 2015, pp. 1–6.

[15] C. Ghribi, M. Hadji, and D. Zeghlache, "Energy efficient vm scheduling for cloud data centers: Exact allocation and migration algorithms," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, May 2013, pp. 671–678.

[16] D. Minarolli and B. Freisleben, "Virtual machine resource allocation in cloud computing via multi-agent fuzzy control," in *Cloud and Green Computing (CGC), 2013 Third International Conference on*, Sept 2013, pp. 188–194.

[17] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, June 2013.

[18] C.-T. Yang, K.-C. Wang, H.-Y. Cheng, C.-T. Kuo, and W. C. C. Chu, "Green Power Management with Dynamic Resource Allocation for Cloud Virtual Machines." IEEE, Sep. 2011, pp. 726–733. [Online]. Available: http://ieeexplore.ieee.org/document/6063066/

[19] D. Hatzopoulos, I. Koutsopoulos, G. Koutitas, and W. Van Heddeghem, "Dynamic virtual machine allocation in cloud server facility systems with renewable energy sources," in *Communications (ICC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4217–4221. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/6655225/

[20] R. Fuller, *Introduction to Neuro-Fuzzy Systems*, ser. Advances in Soft Computing Series. Physica-Verlag Heidelberg, 2000.

[21] L. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338 – 353, 1965. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S001999586590241X

[22] C. C. Lee, "Fuzzy logic in control systems: fuzzy logic controller. i," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 404–418, Mar 1990.

[23] ——, "Fuzzy logic in control systems: fuzzy logic controller. ii," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 419–435, Mar 1990.

[24] D. Adami, A. Gabbrielli, S. Giordano, M. Pagano, and G. Portaluri, "A fuzzy logic approach for resources allocation in cloud data center," in *2015 IEEE Globecom Workshops (GC Wkshps)*, Dec 2015, pp. 1–6.

[25] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 1, pp. 116–132, Jan 1985.

[26] G. Portaluri, S. Giordano, D. Kliazovich, and B. Dorronsoro, "A power efficient genetic algorithm for resource allocation in cloud computing data centers," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, Oct 2014, pp. 58–63.

[27] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[28] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. U. Khan, "Greencloud: A packet-level simulator of energy-aware cloud computing data centers," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, Dec 2010, pp. 1–5.

[29] P. Ruiu, A. Bianco, C. Fiandrino, P. Giaccone, and D. Kliazovich, "Power comparison of cloud data center architectures," in *IEEE ICC 2016 - Communication QoS, Reliability and Modeling Symposium*, 2016.

[30] D. Adami, S. Giordano, M. Pagano, and G. Portaluri, "A novel sdn controller for traffic recovery and load balancing in data centers," in *2016 IEEE 21st International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*, Oct 2016, pp. 77–82.

[31] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, ser. SIGCOMM '08. New York, NY, USA: ACM, 2008, pp. 63–74. [Online]. Available: http://doi.acm.org/10.1145/1402958.1402967

[32] J. L. Hintze and R. D. Nelson, "Violin plots: A box plot-density trace synergism," *The American Statistician*, vol. 52, no. 2, pp. 181–184, May 1998.