Balancing between cost and availability for CDNaaS resource placement

Louiza Yala*, Pantelis A. Frangoudis‡, Giorgio Lucarelli§, and Adlen Ksentini‡
*IRISA/University of Rennes 1, France
‡EURECOM, Sophia Antipolis, France
§Univ. Grenoble Alpes, CNRS, INRIA, LIG, F-38000 Grenoble France
Email: *louiza.yala@irisa.fr, ‡name.surname@eurecom.fr, §giorgio.lucarelli@imag.fr

***Abstract*—We focus on the problem of optimal compute resource allocation and placement for the provision of a virtualized Content Delivery Network (CDN) service over a telecom operator's Network Functions Virtualization (NFV) infrastructure. Starting from a Quality of Experience (QoE)-driven decision on the necessary amount of CPU resources to allocate to satisfy a virtual CDN deployment request with QoE guarantees, we address the problem of distributing these resources to virtual machines and placing the latter to physical hosts, optimizing for the conflicting objectives of management cost and service availability, while respecting physical capacity, availability and cost constraints. We present a multi-objective optimization problem formulation, and provide efficient algorithms to solve it by relaxing some of the original problem's assumptions. Numerical results demonstrate how our solutions address the trade-off between service availability and cost, and show the benefits of our approach compared with resource placement algorithms which do not take this trade-off into account.**

I. Introduction

Network Functions Virtualization (NFV) has emerged as a disruptive new paradigm for network service provision. NFV decouples network functions from specialized hardware implementations, allowing their execution as virtual instances on top of cloud infrastructures on general purpose hardware. This brings flexible service deployment, automatic scaling capabilities, and, in turn, dynamic and optimized resource management and better response to service demand dynamics. Therefore, cost efficiency and improved end user experience can be achieved.

Significant standardization efforts are in progress with respect to NFV. The ETSI has recently proposed an NFV Management and Orchestration framework (NFV-MANO) [1], which specifies a set of architectural components and interfaces to realize the NFV vision. Among the many use cases envisioned in such an environment is the provision of a virtualized Content Delivery Network (vCDN) service [2, Use Case #8].

Content delivery is a field which has received significant attention, in particular regarding the interactions between CDN providers and network operators [3]. These interactions give rise to various content delivery models which involve different degrees of cooperation between stakeholders [4]. One such model is the telco CDN, where a network operator installs and operates its own content delivery infrastructure at strategically selected points in its network, offering a complete CDN service to content providers. Proposing such a service over a virtualized infrastructure has specific benefits for the network operator and is facilitated by NFV. Instead of deploying dedicated hardware to host CDN components, such as content origin servers, caches, load balancers, and DNS resolvers, the operator can lease its telco cloud resources for launching virtual instances (Virtual Machines (VMs) or other containers) of the above components on demand, allocate resources intelligently at the optimal locations in its network, and scale them to match dynamic workloads.

In our prior work [5], we presented an architecture for CDN-as-a-Service (CDNaaS) provision over a network operator's cloud, with a design following the NFV-MANO spirit. Our architecture allows content providers to request the dynamic instantiation of a full vCDN using a REST API, expressing their target end-user demand per region, as well as Quality of Experience (QoE) and service availability requirements. Specific service orchestration components are then responsible for taking all necessary steps for the deployment and runtime management of the vCDN instance on the underlying NFV Infrastructure (NFVI) of the operator, abstracting internal details and offering the customer (content provider) only the necessary entry point to the vCDN, e.g., to infuse content or terminate the service.

Two important challenges arise. First, the operator has to appropriately dimension the service, i.e., decide on the necessary amount of computing (and other) resources to dedicate to satisfy the requirements of the customer. Second, it has to derive an appropriate placement of the virtual resources on its NFVI, aiming to minimize cost and potentially offer availability guarantees. Intuitively, these two objectives are conflicting; utilizing more virtual or physical resources to improve on service availability via redundancy and fault tolerance naturally increases management (e.g., energy) cost. It should be noted that the importance of resilience and availability for NFV has been acknowledged by the ETSI NFV Industry Specification Group, which has published specific

This work has been partially supported by the French FUI-18 DVD2C project. It has also been supported in part by the European Union's Horizon 2020 research and innovation programme under the 5G!Pagoda project with grant agreement no. 723172.

requirements and guidelines, identifying, among others, the cost-availability trade-off [6].

We have addressed the first challenge in our prior work [7] focusing on a video content delivery service; we experimentally quantified the relationship between video QoE and service workload, and used this information to optimally decide on the amount of CPU resources to allocate to satisfy customer-defined minimum QoE constraints while minimizing the cost for the operator.

In this work, we respond to the second challenge and make the following contributions: (i) We capture the conflicting objectives of service availability and deployment cost by proposing a multi-objective optimization formulation for the problem of joint compute resource allocation and VM placement (Section II). (ii) We provide efficient solutions to this problem by relaxing some of its assumptions (Section III). Our numerical results (Section IV) demonstrate quantitatively the trade-off between availability and cost and show our algorithms to outperform less sophisticated and cost/availability-unaware resource allocation and placement schemes.

II. A model for joint vCPU-to-VM allocation and VM placement

A. Preliminaries

We assume that the CDNaaS operator has data centers in a number of regions, on which vCDN instances for video distribution can be deployed.

The vCDN delivers video content over HTTP via a number of caches which operate as streaming servers. Caches are implemented as VNFs. An end-user request for a video item is redirected to the closest cache using DNS geolocation techniques (other options are also possible). Each cache VNF instance is composed of a set of VMs with identical functionality (HTTP servers streaming video), and user requests are load-balanced among them. Cache management strategies and content placement in caches are outside the scope of this work.

A customer (content provider) uses a northbound API to request the instantiation of a vCDN to cover a subset of the regions where the operator has presence. The request includes a demand specification (target maximum number of simultaneous video streams, duration of the service) and a quality specification (minimum acceptable video QoE and service availability levels) per region. We focus our discussion on a single region, since the target of the operator is to deliver content to each region's end users from the local data center, thus resources are allocated in a region-local manner. The procedures we present in this paper are to be executed per region included in the customer request.

For each vCDN deployment request, the operator first has to decide on the number p of virtual CPUs (vCPUs) needed to satisfy user demand (in terms of numbers of parallel video streams) respecting the minimum QoE constraint set by the customer. This is a problem we have treated in our prior work [7].

Second, the operator derives an assignment of the p vCPUs to a number of virtual machines (VMs) and the placement of the latter in (a subset of) the available m physical machines (PMs) of a regional data center, aiming to minimize the deployment cost and maximize service availability, while respecting PM vCPU capacity constraints. The outcome of this process is a matrix $X = (x_{ij})$ with $i \in [1, p]$ and $j \in [1, m]$, where x_{ij} denotes the number of vCPUs assigned to VM i hosted in PM j. The upper bound for i is the number of vCPUs to assign (since our unit of processing is a vCPU, we cannot have more VMs than the number of vCPUs to assign). The calculation of the optimal assignment should respect physical capacity, cost, and availability constraints.

B. Cost model

We consider that the deployment of a VM comes with a fixed management overhead which is not a function of its workload. For example, this cost can account for the energy consumed for booting the VM or for operating other system- or service-level components (e.g., operating system). We further assume that for each PM which hosts service instances, there is a fixed overhead which is not a function of the PM workload nor the number of VMs hosted by it (e.g., energy cost for keeping the physical machine in an operating state, overhead of various system-level components). We model the above costs as linear functions of the number of VMs and PMs utilized by a service deployment, which is in line with the experimental observations of Callau-Zori et al. [8].

In matrix X, the number of non-zero elements represents the number of VMs deployed. Therefore, the cost of an assignment X at the VM level is given by

$$C_V(X) = e_V \sum_{i=1}^{p} \sum_{j=1}^{m} \mathbb{1} x_{ij}, \tag{1}$$

where e_V is the fixed cost per deployed VM. In a similar spirit, the cost of an assignment X at the PM level is determined by the number of PMs that host at least one VM. This corresponds to the number of columns in X that contain at least one non-zero element. This cost is thus given by

$$C_P(X) = e_P \sum_{j=1}^{m} \mathbb{1}(\sum_{i=1}^{p} x_{ij} > 0), \tag{2}$$

where e_P is the fixed cost per used PM, and the overall cost of an assignment follows:

$$C(X) = C_V(X) + C_P(X). \tag{3}$$

C. Availability model

We define service availability as the ability of the system to offer at least a minimal service, i.e., to have, at any time, at least one VM accessible, which implies that at least one PM should be up to host the respective VM(s).

We make the following assumptions:

- A VM i can fail with probability $q_i^{(V)}$, independently of the other VMs and PMs, and irrespectively of the load imposed on the VM.

- Each PM j can fail with probability $q_j^{(P)}$, independently of the other PMs or the load imposed on it. The above probabilities are assumed to be known by the operator as a result of measurement studies, prior experience, or other historical information. (The same applies to VM failure probabilities.)
- If a PM fails, all VMs deployed on it are assumed to fail because of that.

Therefore, a VM may become inaccessible either because it fails or because the PM that hosts it fails. VM failures can be correlated due to their dependence on the underlying PMs. Based on this, we define a *correlated group* of VMs as the VMs which are executed on the same PM. For a correlated group to be available, the following conditions should be met:

- The PM is up.
- At least one of the VMs deployed on the PM does not fail.

The probability that a correlated group deployed on PM j is available is thus given by

$$a_j = (1 - q_j^{(P)})(1 - \prod_{i \in [1,p] | x_{ij} > 0} q_i^{(V)}) \tag{4}$$

For a vCDN service deployment to be available, at least one correlated group should be available. Since correlated groups fail independently, the probability that a vCDN service deployment is available is given by

$$\begin{aligned} A(X) &= 1 - Pr\{\text{All correlated groups fail}\} \\ &= 1 - \prod_{j \in [1,m] | \Sigma_{i=0}^{p} x_{ij} > 0} (1 - a_j) \\ &= 1 - \prod_{j \in [1,m] | \Sigma_{i=0}^{p} x_{ij} > 0} \left[q_j^{(P)} + (1 - q_j^{(P)}) \prod_{i \in [1,p] | x_{ij} > 0} q_i^{(V)} \right] \end{aligned} \tag{5}$$

Since, by construction, any feasible solution includes at least one PM with at least one VM assigned to it, both product terms in (5) are over non-empty sets.

D. Problem formulation

The aim of the system operator is to derive an optimal assignment X^* which minimizes cost while maximizing availability. These two criteria are conflicting: the more the VMs deployed and the PMs used to host them, the less the risk of service unavailability, but, at the same time, the higher the cost of the deployment. Since it is not possible to optimize for both criteria simultaneously, we apply a scalarization approach to transform the problem to a single-objective one. The relative importance of the two criteria in deriving an optimal assignment is dictated by a specific *policy*, which is encoded in a pair of weights w_a and w_c (resp. availability and cost) such that $w_a + w_c = 1$. Given a specific policy, the system operator derives the optimal solution to the following problem:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & w_c C(X) - w_a A(X) && (6) \\ \text{subject to} \quad & C(X) < E && (7) \\ & A(X) > A && (8) \\ & \sum_{j=1}^{m} \mathbb{1}(x_{ij} > 0) \leq 1, \forall i \in [1,p] && (9) \\ & \sum_{i=1}^{p} \sum_{j=1}^{m} x_{ij} = p && (10) \\ & \sum_{i=1}^{p} x_{ij} \leq C_j, \forall j \in [1,m]\,. && (11) \end{aligned}$$

To deal with the potential difference in the magnitude of the two components of the objective function, the values of $C(X)$ and $A(X)$ are appropriately normalized in the $(0,1)$ interval using the upper-lower-bound approach [9]. This model supports specific maximum cost and minimum availability constraints (C and A, respectively; see (7) and (8)). Constraint (9) ensures that, for any VM, its vCPU resources are allocated on a single PM, since a VM cannot be split. Note that an assignment can result in this sum being zero for multiple values of i. This occurs when the number of VMs to deploy is less than the number of vCPUs to assign, a case typical in practice. Constraint (10) ensures that all vCPUs are allocated, and constraint (11) guarantees that the CPU capacity of each PM is not exceeded.

III. Solving a relaxed version of the problem

Deriving an exact optimal solution to this problem is prohibitive computationally,[1] and we resort to a heuristic algorithm instead. Specifically, instead of jointly deciding on optimally distributing the number of vCPUs to VMs and placing the latter in PMs, we treat the two subproblems independently in two phases: We first decide on the number of VMs to utilize, and then on their actual physical placement and CPU resource distribution. Both steps can be efficiently solved in polynomial time.

The overall procedure of deriving an appropriate resource allocation and placement following a customer's CDNaaS instance deployment request involves three subproblems summarized below:

1) Find an optimal number of vCPUs to allocate per region to serve customer demand. We solve this problem using an algorithm proposed in our prior work [7], which decides on the minimum amount of vCPU resources necessary to satisfy operator capacity and customer QoE constraints. This algorithm uses empirical models of QoE as a function of the service workload per processing unit (vCPU), which we derived after testbed experiments.

[1]Single-objective variants of the VM placement problem where each VM has predefined resource specifications have been shown to be NP-hard by reduction to the bin packing problem [10].

2) Decide on the optimal number of VMs to launch under a specific policy, satisfying cost and availability constraints.
3) Place the decided number of VMs on an optimal number of physical hosts and distribute vCPUs to them, by implementing a suitable placement algorithm having as input the results of the previous steps (optimal number of VMs, number of vCPUs). Specific host capacity, cost, and availability constrains apply.

We detail our mechanisms to solve the second and third subproblems in sections III-A and III-B respectively. We treat each subproblem independently, providing for each one a problem formulation and an algorithm to optimally solve it. We should note that, for each subproblem, the same minimum availability constraint value (A) as the original problem is applied (although the availability functions may differ). Regarding management cost, there is a specific budget for each subproblem (E_V and E_P for the VM- and the PM-level problems respectively), such that $E_V + E_P = E$. It is up to the system operator to decide how the overall budget is split; we do not consider this issue in this work.

A. Deciding on the number of VMs to launch

In this section, we describe the second step of our problem, which aims to find the optimal number of VMs to launch. We define an objective function to minimize, which includes a management cost and an availability component.

1) Availability: Given a known VM failure probability q_V, assumed to be fixed and identical for all VMs of the same type included in a vCDN instance (VMs of a specific type, e.g. streaming servers, are considered identical and their failure probability does not depend on the resources allocated to them nor their workload), we define service availability at the VM level as

$$A_V(x) = 1 - q_V^x, \tag{12}$$

where x is the number of VMs to deploy. This expression for availability corresponds to the probability that at least one VM is available and is heuristic in the sense that it ignores PM failures and considers VM failures independent.

2) Management cost: The management cost at the VM level is given by (1). A simplified expression defined in terms of the number x of VMs to deploy is

$$C_V(x) = e_V x. \tag{13}$$

3) Problem formulation: Applying the specified policy expressed as the combination of weights (w_c, w_a), the function to minimize at the VM level becomes

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & w_c C_V(x) - w_a A_V(x) && (14)\\ \text{subject to} \quad & A_V(x) \geq A, && (15)\\ & C_V(x) \leq E_V, && (16)\\ & x \geq m_{min}, && (17)\\ & x \leq p, && (18) \end{aligned}$$

where A, E_V are the respective constraints of availability and cost, and m_{min} is the minimum number of PMs capable of accommodating the required number of vCPUs. The latter is given by

$$m_{min} = \min\{l \in [1, m] | \sum_{i=1}^{l} C_i \geq p\}, \tag{19}$$

where $C_1 \geq C_2 \geq C_3 \geq .... \geq C_m$ are the capacities of the m PMs of a given region in non-increasing order.

Minimizing (14) means finding an optimal number of x VMs, under reliability (15) and cost (16) constraints. Constraint (17) ensures that x is such that no single VM is assigned more vCPUs than the maximum single-host available capacity. (For instance, when x=1 (with p vCPUs), if there is no physical host with a capacity superior to p, such a solution is infeasible.) On the other hand, (18) ensures that the VMs to be launched will not be more than the number of vCPUs to allocate.

4) Selection of the optimal number of VMs: To solve (14) under a specific policy defined by w_a and w_c, we propose an algorithm whose input is (i) the total number, p, of vCPUs to be allocated across the region's hosts in order to serve the customer request, derived in step 1, (ii) the constraints per objective (A, E_V), and (iii) the combination of weights w_c, w_a that defines the adopted policy.

We first observe that the objective function is convex and the value that minimizes its continuous version with $x \in \mathbb{R}_{>0}$ is $x_0 = \log_{q_V} \frac{w_c e_V}{w_a \ln q_V^{-1}}$. Constraints (15) and (16) can be rewritten as $x \geq \log_{q_V}(1 - A)$ and $x \leq \frac{E_V}{e_V}$, respectively. Constraints (15) and (17) thus provide a lower bound to the optimal value of x. Which of the two constraints is more restrictive depends on the configuration of the problem. Similarly, an upper bound is provided by the cost constraint (16) and (18).

To minimize (14) we take the following steps: We calculate the value $x_0 \in \mathbb{R}_{>0}$ which minimizes the continuous version of (14). If x_0 lies within the feasible region defined by the above bounds, we select the closest (feasible) integer value to x_0 as the optimal. Otherwise, we select the value of the constraint that x_0 violates.

We should note that in other problem settings where the objective function is not convex, an optimal solution can be found in pseudo-polynomial time in p by evaluating the objective function and the constraints for each $x \in [m_{min}, p]$. The case $x = m_{min}$ corresponds to a deployment which minimizes the number of deployed VMs, while $x = p$ refers to a deployment that performs a one-to-one vCPU-to-VM assignment. This algorithm runs efficiently for realistic values of p.

B. VM placement and vCPU distribution

Having decided on a number of VMs to launch, we decide on their placement on the underlying PMs and the allocation of a number of vCPUs to each one of them. The output of

this step is an assignment $X = (x_{ij})$, with $i \in [1, v]$ and $j \in [1, m]$, which represents a heuristic solution to (6).

Similar to how we treated the sub-problem of selecting an optimal number of VMs, we define appropriate availability and cost functions at the PM level as follows.

1) Availability: To measure a solution's availability, we use (5), assuming identical PMs with a known failure probability fixed to $q_P^{(j)} = q_P, \forall j \in [1, m]$.

2) Management cost: Assuming identical PMs, and in turn a fixed management overhead for each PM on which we are placing the customer's VMs, irrespective of their number and the service workload imposed, we model PM-level cost as a linear function of the number of PMs eventually used for hosting at least one VM. This cost is given by (2), substituting p for v in the upper limit of the second summation, since the number of VMs in this case is already known, as calculated at the second step of our scheme.

3) Problem formulation: We propose the following multi-objective formulation for the problem of placing v VMs to a subset of the available m PMs and distributing p vCPUs to them under capacity, availability and cost constraints and given policy (w_c, w_a):

$$\underset{X}{\text{minimize}} \quad w_c C_P(X) - w_a A(X) \tag{20}$$

$$\text{subject to} \quad A(X) \geq A, \tag{21}$$

$$C_P(X) \leq E_P, \tag{22}$$

$$\sum_{j=1}^{m} \mathbb{1}(x_{ij} > 0) = 1, \forall i \in [1, v] \tag{23}$$

$$\sum_{i=1}^{v} x_{ij} \leq C_j, \forall j \in [1, m] \tag{24}$$

where A, E_P are the respective availability and cost constraints.

Constraint (23) ensures that a VM is allocated to only one PM, and the set of constraints (24) guarantees that the available capacity per PM is not exceeded.

4) An algorithm for VM placement and vCPU allocation: We propose the following algorithm to derive an optimal solution to (20) in polynomial time, whose input is (i) the number, p, of vCPUs to be allocated, (ii) the policy and constraints per objective, (iii) the sorted PMs in non-increasing order of capacities, i.e, $C_1 \geq C_2 \geq C_3 \geq .... \geq C_m$, (iv) m_{min}, and (v) the number, v, of VMs; the latter three are obtained in step 2.

This algorithm consists in creating and evaluating a candidate assignment of VMs to m' PMs, for each $m' \in [m_{min}, \min(m, v)]$. More than v PMs are not necessary, since that would directly mean that at least one PM would not host any VM. We observe that the value of the objective function only depends on the number of PMs used and the number of VMs assigned to each PM. Our algorithm starts by carrying out the following steps for each m':

1) Distribute the v VMs among the m' first PMs proportionally to their capacities.
2) Evaluate the objective function and the constraints for the derived assignment.

Eventually, our algorithm returns the assignment of VMs to PMs which minimizes the objective function for the given policy, or responds that there is no feasible solution under the given constraints.

This procedure has $\mathcal{O}(m \times \min(m, v))$ complexity: The two steps have to be repeated at most $\min(m, v)$ times. Step 1 takes time linear on the number of PMs. (Using the *largest remainder* apportionment method for proportional distribution is $\mathcal{O}(m)$.) Evaluating the availability function could also be implemented to take $\mathcal{O}(m)$ time, if we separately keep track of the number of VMs assigned per PM in a candidate solution.

If a feasible solution using m^* PMs is found, the algorithm ends by:

1) Distributing the p vCPUs among the first m^* PMs proportionally to their capacities.
2) For each of the first m^* PMs, distributing the number of the allocated vCPUs evenly to the VMs assigned to it.

The complexity of the algorithm is dominated by the first two steps. Finally, since only the number of PMs and VMs and the placement of the latter influence the objective function value, any method for sharing vCPUs to PMs/VMs is equivalent.

IV. Numerical results

We present numerical results from our python implementation of the proposed scheme. We begin with an experiment where we run our algorithms for the same configuration and under different combinations of weights (w_c,w_a), each corresponding to a different policy. In particular, we use (i) the same number of vCPUs to allocate, (ii) the same number of available hosts and their capacities, and (ii) the same cost and availability constraints. We set the availability constraint to "five nines" (99,999%), a popular availability target for carrier-grade NFV. We arbitrarily fix the overall cost constraint to $E = 160$ and assume that the costs for a singe VM and PM are $e_V = 1$ and $e_P = 1$ respectively. The failure probabilities for VMs and PMs are set to $q_V = q_P = 0.001$. We consider a system with 50 PMs, each with a capacity between 2 and 15 vCPUs.

The obtained results of our heuristic scheme are shown in Fig. 1 and Fig. 2. Fig. 1 presents the solution space, where the x-axis and y-axis represent the absolute values of the cost and availability functions respectively. Each filled square point corresponds to the outcome of our algorithm (a vCPU-VM-PM assignment identified as the optimal) according to a specified policy, while empty squares are feasible but suboptimal solutions. The vertical and horizontal lines are the respective cost and availability constraints and limit the space of feasible solutions. (Note that in our tests it can happen that for two different policies the same optimal solution is derived. These points are superimposed in the figure.)

The selected solutions represent what our algorithm identifies as the *Pareto frontier*, which, in our context, is the set

of solutions for which it is not possible to improve on cost without sacrificing on availability. Each such solution represents a different availability-cost trade-off. A cost-centered policy, e.g. $(w_c, w_a) = (0.8, 0.2)$, guides our algorithm to select as the optimal a low-cost solution, which, according to the cost function, uses a small number of VMs/PMs. In turn, this corresponds to a lower service availability (the algorithm guarantees that it is a feasible solution, not violating the 0.99999 availability constraint). Availability is increasingly higher (≈ 1) when more hosts are used. Such solutions correspond to increasing w_a values, which our mechanism takes into account to drive the calculation to an appropriate VNF placement with more virtual instances and PMs.

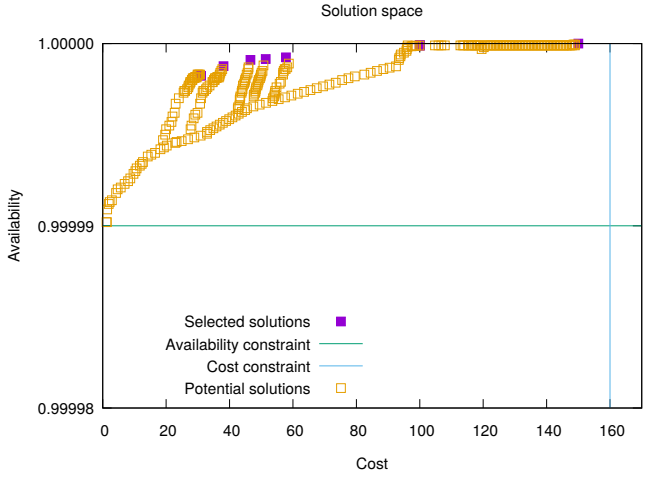

Fig. 1. Solution space. The straight lines represent cost and availability constraints. The filled boxes are the assignments selected as optimal by our algorithm under different policies, while the empty boxes are feasible but suboptimal solutions.

This is more evident in Fig. 2, where we detail the evolution of the value of each objective (cost and availability) in each optimal solution as a function of a given policy. For reasons of clarity, the values presented are normalized by mapping the lowest and highest value per objective to 0 and 1 respectively. The figure indicates that as the weight of the cost objective w_c increases, both functions are decreasing. This is positive from a cost but not from an availability perspective, and is due to less VMs/PMs being used as w_c approaches 1 and w_a approaches 0. (The inverse holds for availability.)

In our second experiment, we evaluate our heuristic solution by comparing it with two candidate algorithms, *random placement* and *first-fit*.

The random placement algorithm assigns a random number of physical hosts where to run the virtual instances (we used the number of VMs derived in step 2). The first-fit algorithm places VMs directly in the first available PMs that have enough capacity to host them. Fig. 3 presents a comparison between our algorithm (purple curve, cross points) and the results of the random (green curve, "x" points) and first-fit (blue curve, "*" points) algorithms. Each point is the mean of 1000 iterations,

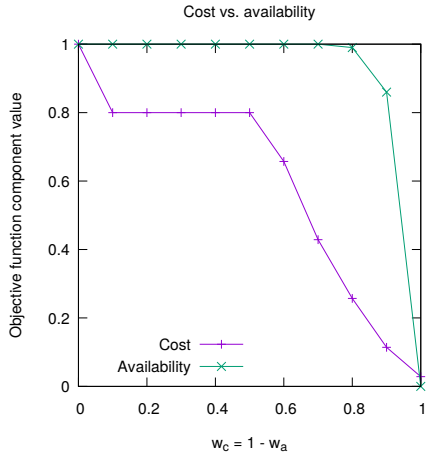

Fig. 2. Availability and cost as a function of the selected policy.

presented with 95% confidence intervals.

Our solution minimizes the multi-objective function value, compared with both the random and first-fit placement algorithms, which take into account neither the constraints nor the weightings selected. As can be observed in Fig. 3, the objective function value for our algorithm remains the minimum among the three candidate mechanisms, but after some point is approximated or coincides with the first-fit one. This happens for more heavily cost-centered policies (for $w_c \geq 0.7$, in this experiment configuration). The first-fit algorithm always uses the minimum number of PMs (since it always packs a VM in the first available PM) which however negatively impacts availability. This is why the benefits of our scheme which aims to balance between availability and cost are more pronounced when availability matters.

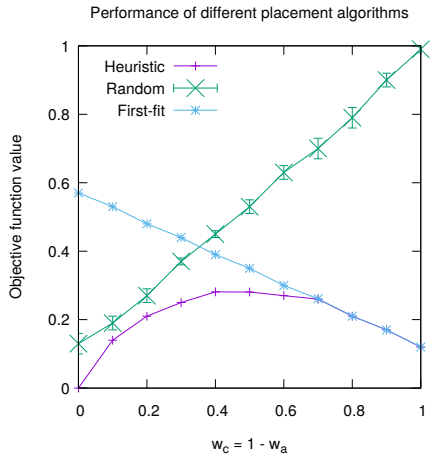

Fig. 3. Comparison of our heuristics with random and first-fit placement algorithms.

V. Related work

The placement of virtual instances to physical resources is central in NFV, and cloud computing in general. The objective is to find a suitable set of PMs with enough capacity to host VMs with specific resources allocated. Mann [10] reviews different VM placement models and algorithms, citing, among different formulations, the problem of packing the VMs into a minimal number of PMs, considering PM capacities and the load of VMs. This reduces to the bin packing problem, and various heuristics to solve it in this context exist, such as first-fit, where the items (VMs) are placed in the first suitable bin (PM) [11], [12].

The choice of a suitable placement is often driven by maximizing resource utilization or performance while minimizing cost in the forms of energy consumption, network traffic or penalties associated with SLA violations under various constraints [13], [14], [15]. The common ground in the way various flavors of the VM placement problem are tackled is the need to provide heuristics to problems with typically high computational complexity. Minimizing the number of physical hosts to place VMs for cost and performance optimizations, a procedure known as VM consolidation, may have an adverse effect on service resilience and availability in the face of PM failures. Contrary to the aforementioned works, service availability is our special focus. Yang et al. [16] focus on reliable VM placement, and, in a similar spirit to our case, model service availability as the probability that a subset of the requested VMs is operational. However, they address a different VM placement problem, which, among others, does not consider the distribution of the resources to be committed (in our case, vCPUs; in their case, storage), assuming fixed VM resource specifications, and does not take into account VM-level failures.

Qu et al. [17] focus on the problem of VNF chaining and aim to minimize the network-wide communication bandwidth for the operation of VNF chains under service reliability constraints. They do not consider failures at the VM level in their reliability functions, and their focus is rather on chain specific issues such as routing and VNF ordering, which constrain VNF-to-host placement. Furthermore, their model does not include CPU capacity constraints.

A relevant aspect, but outside the scope of this paper, is the cost of fault recovery to maintain high cloud service availability. Israel and Raz [18] present approximate algorithms with performance guarantees and heuristics to tackle it.

VI. Discussion and conclusion

We studied the problem of jointly allocating CPU resources to virtual instances and their placement on a cloud infrastructure for the provision of CDN-as-a-Service. Our special focus was on addressing the trade-off between service availability and management cost. To this end, we proposed a multi-objective optimization formulation of the problem, as well as efficient heuristic algorithms to solve it. Numerical results demonstrate how different policies as to the relative importance of availability and cost affect the selection of an appropriate solution to the problem, and verify the performance advantages of our scheme compared to less sophisticated, policy-unaware mechanisms.

Our approach captures various important aspects, but it has some specific limitations which are the subject of our ongoing work. First, our model does not consider heterogeneous machines in terms of reliability and performance. Second, our cost functions do not consider whether PMs are already active hosting virtual instances; it could be argued that favoring such PMs in the placement process would lead to more significant energy savings. Although our model can be directly extended to cover this case, our heuristic algorithms would not apply. Finally, taking into account failure recovery costs is another direction for future research.

References

[1] *Network Functions Virtualisation (NFV); Management and Orchestration*, ETSI Group Specification NFV-MAN 001, Dec. 2014.

[2] *Network Functions Virtualisation (NFV); Use Cases*, ETSI Group Specification NFV 001, Oct. 2013.

[3] B. Frank, I. Poese, G. Smaragdakis, A. Feldmann, B. Maggs, S. Uhlig, V. Aggarwal, and F. Schneider, "Collaboration Opportunities for Content Delivery and Network Infrastructures," *ACM SIGCOMM ebook on Recent Advances in Networking*, vol. 1, August 2013.

[4] N. Herbaut, D. Négru, Y. Chen, P. A. Frangoudis, and A. Ksentini, "Content delivery networks as a virtual network function: A win-win ISP-CDN collaboration," in *Proc. IEEE Globecom*, 2016.

[5] P. A. Frangoudis, L. Yala, and A. Ksentini, "CDN-as-a-Service provision over a telecom operator's cloud," *IEEE Trans. Netw. Service Manag.*, 2017, doi:10.1109/TNSM.2017.2710300.

[6] *Network Functions Virtualisation (NFV); Resiliency Requirements*, ETSI Group Specification NFV-REL 001, Jan. 2015.

[7] L. Yala, P. A. Frangoudis, and A. Ksentini, "QoE-aware computing resource allocation for CDN-as-a-Service provision," in *Proc. IEEE Globecom*, 2016.

[8] M. Callau-Zori, L. Samoila, A.-C. Orgerie, and G. Pierre, "An experiment-driven energy consumption model for virtual machine management systems," IRISA; Université de Rennes 1; CNRS, Research Report RR-8844, Jan. 2016.

[9] R. T. Marler and J. S. Arora, "Function-transformation methods for multi-objective optimization," *Engineering Optimization*, vol. 37, no. 6, pp. 551–570, 2005.

[10] Z. A. Mann, "Allocation of virtual machines in cloud data centers–a survey of problem models and optimization algorithms," *ACM Comput. Surv.*, vol. 48, no. 1, p. 11, 2015.

[11] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing SLA violations," in *Proc. IFIP/IEEE IM*, 2007.

[12] W. Song, Z. Xiao, Q. Chen, and H. Luo, "Adaptive resource provisioning for the cloud using online bin packing," *IEEE Trans. Comput.*, vol. 63, no. 11, pp. 2647–2660, 2014.

[13] J. Dong, X. Jin, H. Wang, Y. Li, P. Zhang, and S. Cheng, "Energy-saving virtual machine placement in cloud data centers," in *Proc. IEEE/ACM CCGrid*, 2013.

[14] N. Quang-Hung, N. T. Son, and N. Thoai, "Energy-saving virtual machine scheduling in cloud computing with fixed interval constraints," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXI*. Springer, 2017, pp. 124–145.

[15] D. Gmach, J. Rolia, and L. Cherkasova, "Resource and virtualization costs up in the cloud: Models and design choices," in *Proc. IEEE/IFIP DSN*, 2011.

[16] S. Yang, P. Wieder, and R. Yahyapour, "Reliable virtual machine placement in distributed clouds," in *Proc. 8th Int'l Workshop on Resilient Networks Design and Modeling (RNDM)*, 2016.

[17] L. Qu, C. Assi, K. Shaban, and M. Khabbaz, "Reliability-aware service provisioning in NFV-enabled enterprise datacenter networks," in *Proc. 12th International Conference on Network and Service Management (CNSM)*, 2016.

[18] A. Israel and D. Raz, "Cost aware fault recovery in clouds," in *Proc. IFIP/IEEE IM*, 2013.