EAVIRA: Energy-Aware Virtual Infrastructure Reallocation Algorithm

Denivy B. Ruck, Maurício A. Pillon, Charles C. Miers, Guilherme P. Koslovski
Graduate Program in Applied Computing – Department of Computer Science – Santa Catarina State University
e-mail: denivy.ruck@labp2d.udesc.br, {mauricio.pillon, charles.miers, guilherme.koslovski}@udesc.br

***Abstract*—The elastic provisioning of Virtual Infrastructures (VIs) enables a dynamic management of cloud resources (computing and communication) in order to meet the hosted application's requirements. Thus, to perform elasticity requests, providers usually rely on reallocation mechanisms and policies. The concerns regarding the environment and the operational costs indicate energy consumption of the data centers as recurring topic in providers policies. Moreover, energy-aware provisioning is beneficial for tenants also. Recent cost models have introduced an implicit incentive to use computing and networking resources just when need to avoid high rent costs. However, reallocation and elasticity requests can unbalance the Data Center (DC) unnecessarily increasing the number of active servers. In this paper we propose EAVIRA algorithm, which takes into account the proportional sharing of CPU usage of DC servers to calculate individual usage costs to: disable idle equipments, and reallocate VIs. EAVIRA acts on online requests for elasticity configuration and performs an offline load balancing, triggered by the Infrastructure as a Service (IaaS) provider. Our experimental analysis indicates a reduction of energy consumption and an increasing on acceptance ratio of allocation requests.**

I. Introduction

The economical and administrative benefits bring out by IaaS clouds to tenants led to the wide dissemination of VIs. With the IaaS model, providers offer idle DC capacity as services, amortizing the operational costs and generating revenue with a reduced marginal cost. Each tenant composes a private and isolated VI in accordance with the hosted application's requirements specifying network and compute parameters. Moreover, IaaS tenants rent VIs for various purposes without concern for maintenance in physical resources [1]. A key enabler for cloud adoption is the elastic provisioning of virtual resources. The variation in the processing load of hosted applications requires mechanisms to expand, reduce and reorganize, in number and configuration, the virtual resources previously allocated [2]. Instead of provisioning resources for the worst case scenario, the elastic management guarantees the configuration required to accommodate the dynamic workload reducing rent costs.

Among the administrative tasks performed by a cloud provider, we highlight the allocation and reallocation of physical resources for hosting VIs. The former algorithm analyses the residual DC capacity to find suitable servers for hosting the VI, while the latter adapts the elastic virtual resources atop the DC. A request to create or reconfigure a VI is characterized as an online submission, as the tenant seeks an immediate response. On the other hand, a reallocation considering all previously allocated VIs can be triggered by a provider for load balancing purposes [3]. For performing a reallocation of VIs, the DC is usually reorganized (*i.e.,* virtual resources are migrated) to accommodate the demands respecting the Service Level Agreement (SLA). Both problems belong to NP-hard class challenging cloud providers [4].

Allocation and reallocation are guided by policies representing the IaaS provider goals. Currently, the concern about energy consumption of servers is a growing tendency as it directly implies administrative costs as well as environmental damage. For instance, studies accounted that in 2011 DCs were responsible for 3% of US energy consumption, with further growth predicted [5]. This figure is mainly associated to the cooling of servers representing 53% of the total number [6].

The unpredictably of arrival and the non-deterministic configuration of elastic requests are critical factors for energy reduction. In short, the definition of an energy-aware reallocation policy comprising the management of elastic requests is a challenging problem. A simple request can unbalance the entire DC, creating communication bottlenecks, or unnecessarily increasing the number of active servers. With regard to tenants, the commonly applied approaches to server consolidation often violate the SLA [7] [5].

We claim that providers can provide elastic VIs and simultaneously reduce the energy consumption of their DCs. In order to enforce this assertion, a recent study indicated that CPU energy consumption can be used to perform a fair share of the server provisioning costs among providers and tenants [8]. The individual processing cost (virtual CPUs and network traffic) of Virtual Machines (VMs) is combined with a proportional fraction of management costs (*e.g.,* hypervisor management operations) to compose a final cost. Indeed, energy-aware provisioning is beneficial to providers and tenants, decreasing management and rent financial investments.

The algorithm we propose in this work deals with elastic VI reallocation and considers the proportional sharing of host CPUs to calculate individual consumption, disable idle servers, and reorganize virtual resources atop the DC. Besides performing the traditional server consolidation, our algorithm (termed EAVIRA) innovates having as premise a proportional cost sharing model. In short, the increase in energy cost from the provider perspective is directly related not only to the number of active servers, but also to the processing load of VMs. In addition to meet the SLA requirements, the algorithm accounts the fair fraction of energy-based cost

to each tenant for performing the DC load balancing. Our experimental results shows a reduction on energy consumption and an increase on acceptance ratio of new requests. In short, we propose two contributions: *(i)* EAVIRA receives and processes online elastic requests for virtual resources reconfiguration, complementing a lack on existing allocation mechanisms, that usually ignore the adaptation of previously allocated resources [9] [10]; and *(ii)* EAVIRA performs a DC reorganization in order to reduce energy consumption guided by a proportional cost sharing. The paper is organized as follows: Section II motivates the problem and reviews related work. Section III presents EAVIRA. The experimental analysis is discussed on Section IV, and Section V states our considerations and future work.

II. Motivation and Literature Review

A. Elastic Virtual Infrastructure Provisioning

An IaaS cloud provider periodically receives requests from tenants to allocate or reconfigure VIs. The former requires the initial configuration of virtual resources while the latter details the updates necessary to accommodate the current workload of hosted applications. An elastic operation is started by tenants to update the configuration of VMs (*e.g.*, memory, vCPU) and/or number of resources composing a VI.

Figure 1 exemplifies both, the arrival of allocation and reallocation requests (virtual capacity requests are given as weights on graphs). At discrete time intervals, a provider receives one or more allocation and reconfiguration requests that must be immediately analyzed. Initially, VI 1 and VI 2 are mapped atop the DC by an allocation mechanism. The SLA for VI 1 specifies that migration is not allowed for one of its VM (in black). Latter, replication, resizing (up and down) and release operations are requested for VI 1 and VI 2. Complementary, the DC administrator periodically runs a reallocation mechanism to rebalance the DC load (consolidation step).

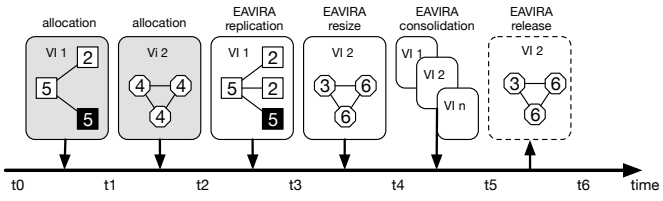


Fig. 1. Online allocation of VIs is performed by a third-part algorithm whereas the reallocation (*i.e.,* replicating, resizing, and releasing VI) and DC consolidation are performed by EAVIRA.

VIs can be hosted on DCs formed of thousands servers interconnected by a structured network topology. A single update executed on a virtual resource or the instantiation of a few VMs may unbalance the DC increasing energy consumption and operational costs. In this context, mechanisms for elastic provisioning are a key enabler for a DC efficient management.

1) Mechanisms for Elastic Provisioning: Both tenants and providers enjoy the benefits of elasticity. On tenants perspective, such technique is used to react on changes at hosted application workload and performance, while for providers the elasticity it is applied to consolidate VMs on servers, increasing the acceptance ratio of new requests (Section IV).

An elastic operation can be categorized in four types [11]: *(i)* release, representing the deactivation of a virtual resource and deallocation of computing, storage and networking capacities; *(ii)* resizing, in which virtual capacities are increased or decreased; *(iii)* replication, representing the activation of a new virtual resource based on an exiting one, commonly used in application load balancing [2]; and *(iv)* migration, in which all the content of a VM is transferred to a new host. Migration is mainly applicable in administrative tasks to reorganize the DC. Eventually, requests for resizing and replication can be converted to migration requests. This scenario is observed when the residual capacity of a host is not enough to attend the request. Finally, the resizing of a virtual resource is the less time costly technique [12].

2) Elastic Provisioning Requests: When requesting a VI, tenants establish an SLA with the IaaS provider. By defining the Service Level Indicators (SLIs), the SLA specifies characteristics that must be guaranteed by the provider, such as reliability, performance, security, geographical location, among others [13]. Complementary, the Service Level Objective (SLO) is used to measure and control the service performance according to the SLA, accounting metrics such as bandwidth, availability, and response time [11]. If execution peaks occur and are identified, then the SLO may be temporary violated. In this case, in order to keep the hosted service quality, elastic requests are triggered. In the present work, in addition to the traditional requirements for VM configuration (vCPU, memory, storage and bandwidth), the SLA comprises the definition of the maximum time allowed for the migration task. The option was included since it is known that VM migration time induces an overhead in the hosted application performance [14].

The tenant can request elastic adjustment based on proactive or reactive monitoring and control mechanisms [15] [16]. However, the algorithm we proposed in Section III is agnostic to metrics and monitoring tools.

B. Related Work

The specialized literature related with the present work comprises allocation and reallocation techniques, mechanisms for elastic provisioning, and policies guided by energy efficiency.

1) VI Allocation and Reallocation: As online allocation algorithms must provide answers on suitable execution time, they are usually based on heuristics to reduce the search space. Regardless of the Quality of Service (QoS) requirements allowed on SLA and the IaaS provider goals, an exhaustive search considering all resources on a DC is inadequate. Following this line, Oliveira *et al.* proposed an algorithm based on trees to speed up the allocation [17], termed VITreeM. The algorithm converts both, a DC graph and a VI request into trees.

Latter, the trees are compared to find a suitable match. An approach based on linear programming was introduced in D-ViNE and R-ViNE [18]. Both proposals sought the allocation with focus on ensuring the shortest route to host a virtual path. By reviewing the literature, we highlight that the algorithm proposed in the present paper (Section III) is agnostic to the VI allocation mechanism. That is, among the many existing solutions [9] [10] [3], the provider can select the appropriate mechanism based on its private objectives. VITreeM [17] and Modified Best Fit Decreasing (MBFD) [3] were selected to compose the experimental scenario (Section IV) but a future work can explore different solutions.

With regard to the reallocation of VIs, an algorithm for DC load balancing was proposed in [19] prioritizing requests with high revenue. In turn, Duan *et al.* developed an algorithm based on load prediction to distribute the VMs atop a DC [5]. An ant colony optimization was applied to reduce the energy cost following the traditional DC consolidation approaches. Beloglazov *et al.* developed an energy-aware heuristic for consolidation based on VM migration [3]. The proposal, termed Minimization of Migrations (MM), is used as baseline for our comparisons (Section IV).

The present work includes tenants expectations on problem formulation: although the main objective is to reduce the energy consumption (a provider decision), the algorithm processes elastic requests respecting SLA requirements. In addition to traditional SLA definitions, a tenant can define the maximum acceptable time for migrating a VM (Section III-C), limiting the consolidation level achieved by a provider.

2) Mechanisms for Elastic Provisioning: Having as main motivation the reduction of VM provisioning costs, Sharma *et al.* proposed an elasticity method based on linear programming [12]. When an elasticity request is received, the provider selects the VM instance flavor with the lowest provisioning cost. Latter, the mechanism applies migrations to consolidate the DC, as well as VM replications for load balancing. Darolt *et al.* adopted reactive elasticity approach to optimize multi-tiered applications and decrease the provisioning cost [16]. The present proposal is independent of internal application monitoring and controlling mechanisms. The definition of thresholds and specification of SLA are considered as tenant actions. Some existing models and techniques may help tenants on defining the allocation and reallocation requests [15] [20].

The virtual resource elastic provisioning focused on high performance computing was proposed in [2]. The mechanism provisions VMs whenever the application performance is lower than expected, innovating by anticipating the realization of elastic requests, reducing the waiting time. EAVIRA can be combined with this approach to decrease the energy consumption, even when executing CPU-intensive applications.

3) Energy Efficiency on IaaS Providers: Some factors related to the impact of elastic provisioning on cloud computing were investigated by Assunção *et al.* [21]. The simulations elucidated that traditional approaches to deactivate servers should be replaced by early reservation strategies, following the considerations of [2]. Based on this assertion, Section IV addresses the application of the proposed mechanism in scenarios based on different requirements for elasticity and migration. With regard to the calculation of the provisioning cost, a common practice for public providers is to dilute both the capital and operational costs on predefined pricing sheets. A recent work motivated the calculation performing a proportional sharing of energy consumption costs [8]. Thus, tenants are charged according to their actual CPU consumption, while providers are responsible for the adjacent costs. The energy-aware cost model (Section III-C1) was incorporated into the algorithm described in Section III.

Finally, it is important to emphasize that the differential of our proposal is the combined reconfiguration of virtual resources and energy-aware load balancing, guided by a proportional share of energy costs, as far as our knowledge a contribution not identified on the related work.

III. EAVIRA

The proposed mechanism, termed *Energy-Aware Virtual Infrastructure Reallocation Algorithm* (EAVIRA), aims to attend requests for VI reconfiguration simultaneously decreasing the DC energy consumption. However, this promising goal must be met respecting the SLA, which in this context defines the maximum acceptable time for the migration of a virtual resource. EAVIRA consists in three main steps: *(i)* definition of a baseline infrastructure; *(ii)* processing elasticity requests; and *(iii)* VIs reallocation atop a DC. The steps are discussed in the following sections.

A. Definition of a Baseline Infrastructure

The starting point of EAVIRA is the definition of a Baseline Infrastructure (BI). Since reallocation is an NP-hard problem, this infrastructure is obtained to represent only a fraction of the DC decreasing the number of servers analyzed at a given execution. In short, this pruning speedups the reallocation process as the number of servers and interconnecting paths is reduced. Thus, only the resources participant of a BI are candidates to eventually receive the migration of virtual resources. A BI is formally defined as the slice of a physical infrastructure (servers and links) that hosts the costly VI. The VI cost is given as follows: considering that a VI is composed of VMs and switches (R) interconnected by virtual links (E), the provisioning cost ($C(VI)$) is defined as $C(VI) = \sum_{i \in R} c(i) + \sum_{l \in E} c(l) \times len(p)$, where $c(.)$ denotes the virtual capacity provisioned for a resource [19]. A virtual link l is hosted by a DC physical path p. Summing up, the cost of a VI is given by the sum of the capacities provisioned for each virtual resource. The hypothesis for this approach is that the costly VI corresponds to the one with the greatest difficult for reallocation (and migration). For recognizing changes that have occurred in the DC, the BI definition is realized at each execution of EAVIRA.

B. Processing Elasticity Requests

After defining the BI, EAVIRA processes requests for virtual resources release, resizing, replication, and migration.

The realization follows an ascending order of execution cost (*e.g.*, resource consuming and processing time) [22]. Initially, by releasing a VI, the providers have more choices of reconfiguration and consolidation. Among the three elasticity mechanisms, resizing has the lowest computational cost, followed by replication and migration [12].

1) Releasing Virtual Infrastructures: This operation consists of releasing the capacity of servers and links previously reserved to host the VIs. The trigger of this operation is the end of a reservation, or when spontaneously invoked by a tenant.

2) Resizing Virtual Resources: EAVIRA considers the scaling up and down of VM configuration (*e.g.*, CPU, RAM) as an on-the-fly resizing operation. However, when the residual capacity of the hosting server is not enough to provision the new configuration, a migration must be performed (Section III-C). The goal behind the migration is to move the VM to a new server with enough residual capacity to satisfy the request (and probably a new one).

3) Virtual Machine Replication: The replication is commonly applied for load balancing of multitiered applications. This task consists on a dispatcher which is on charge to distribute the requests to a set of VMs. When all VMs are compromised processing and answering users requests, a new replica can be deployed to release the application load [11] [2] [16]. In short, the replication of a VM may consist on copying a disk image for a new server and latter activating the new VM. Afterwards, the operating system and all hosted applications are started, and the appliance enters on resource pool. The replication time ($t(i)$) of a VM i is given by $t(i) = \alpha + \frac{D(i)}{b} + \beta$, where $D(i)$ is the image size, b is the available bandwidth to move a new VM to the new host, and constants α and β represents the activation time of hibernating servers and the starting time on target host [12], respectively.

EAVIRA chooses to replicate the VM on the same host since the system image is already available locally to minimize $t(i)$. When the server does not have enough residual capacity to host a new VM, the mechanism selects a host from BI to receive the new replica. If the BI has no candidates apt to host the replica, another server from the DC is selected and incorporated to the BI. The chosen candidate must leave the smallest gaps in residual capacity (best-fit approach) to induce server consolidation [23].

C. Migrating Virtual Infrastructures

After executing the release, resize, and replication requests, the successive migrations of VMs are started. This step is essential for consolidating the DC.

1) Energy-Aware Cost Model: As depicted by Fig. 2, the energy consumption of a virtualized server can be proportionally shared among hosted VMs [8].

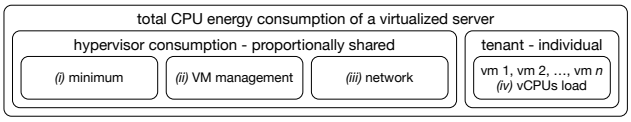


Fig. 2. Proportional sharing of energy consumption [8].

In short, the total cost is composed of: *(i)* minimum energy consumption representing the hypervisor energy consumption without any active VMs; *(ii)* management energy consumption referent to hypervisor processing to schedule, allocate and multiplex resources; *(iii)* networking energy consumption of hypervisor processing to handle VM data transfers; and *(iv)* VM energy consumption originated by application workload.

Both itens *(iii)* and *(iv)* depends on VM workload. However, itens *(i)* and *(ii)* are proportionally shared between tenants and provider. In this sense, the VM consolidation atop a single server is beneficial for a provider as all fractions are diluted between hosted tenants. However, scheduling a VM migration to meet a request for elasticity represents a critical point due to the potential activation of a new server.

2) Predicting VM Migration Time: Implementing VM live migration [24] is a challenging task for cloud providers, moreover, the prediction of total VM migration time based on hosted application workload is a trick responsibility. The prediction must consider VM memory usage, network bandwidth, and restart overhead (pre- and pos-actions) [14], as summarized by Algorithm 1.

Algorithm 1: Predicting VM migration time [14].

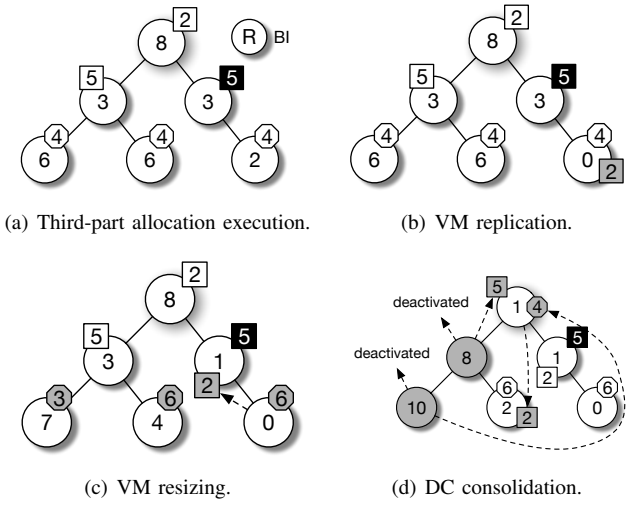
input : $m_{th}, v_{th}, p_{th}, v_{mem}, b, d, l$
output: VM migration time
1 let $v_0 = v_{mem}, v_{mig} = 0, t_{mig} = 0, t_{down} = 0$;
2 **for** $i = 0; i < m_{th}$ **do**
3 $\quad t_i = v_i/b$;
4 $\quad v_{i+1} = t_i * d * l$;
5 $\quad t_{mig} = t_{mig} + t_i$;
6 $\quad v_{mig} = v_{mig} + v_i$;
7 $\quad$ **if** $v_{mig} > v_{th} \vee \frac{v_{i+1}}{l} > p_{th}$ **then**
8 $\quad\quad break$;
9 $t_{down} = v_{i+1}/b$;
10 $t_{mig} = \alpha + t_{mig} + t_{down}$;
11 return t_{mig};

The algorithm is executed in rounds and only transfers the memory content assuming that storage is synchronized (using network file systems or replication tools). At each round, the amount of memory pages changed since the previous round is transferred to destination host (lines 3 to 6). A set of parameters and variables are used to predict the migration time. Initially, representing the system configuration, v_{mem} denotes the total VM memory, l the page size, and b the available bandwidth to transfer VM migration data. The maximum number of iterations that can be performed is given by m_{th}. For given the algorithm parameterization, two thresholds are defined: v_{th} indicates the maximum amount of memory that can be transferred per round, and p_{th} represents the maximum number of dirty pages (changed pages) per iteration. At each round, d is updated with the number of pages that must be transferred (a constant, for prediction), and both the migration volume (v_{mig}) and time (t_{mig}) are accounted. When a threshold is extrapolated, the algorithm performs a stop-and-copy phase (time is accounted in line 9). It is worthwhile to mention that a destination server may be deactivated before

receiving the VM migration. In this sense, α (line 10) is used to represent the activation time when the server is not running. As EAVIRA uses VMs consolidation to disable idle servers, VMs with SLA migration restriction (the allowed migration time is zero) prevent the shutdown of servers that host them, even when they are the unique VMs provisioned on the server.

D. EAVIRA Algorithm and Examples

In order to exemplify EAVIRA execution, Fig. 3 depicts the steps to attend VI elasticity requests and DC consolidation (based on Fig. 2).



(a) Third-part allocation execution. (b) VM replication.

(c) VM resizing. (d) DC consolidation.

Fig. 3. Execution of EAVIRA to attend VI elasticity requests and DC consolidation from Fig. 1.

A partial graph representing the DC servers hosting the baseline infrastructure (Section III-A) is presented. Each node has a residual capacity representing the amount of available resources to allocate new demands. VMs from VI 1 and VI 2 are mapped to DC graph. Initially, the allocations of VI 1 and VI 2 are performed by a third-part mechanism beyond the scope of the present work (a possible solution is presented on Fig. 3(a)). Afterwards, EAVIRA executes the replication, resizing, and consolidation steps as:

1) Processing Elastic VM Requests: The elasticity requests are on-the-fly demands that must be promptly processed. For instance, at time $t3$ from Fig. 1 a replication is required for VI 1. The VM replica is allocated following a best-fit of residual capacity (grey square) as depicted by Fig. 3(b). Latter, at $t3$ (Fig. 1), three operations for resizing VMs from VI 2 are requested: decrease capacity from 4 to 3, and increase two VMs from 4 to 6. The resulting scenario after resizing the VMs is shown in Fig. 3(c). Two requests are promptly performed, however, the resizing of a VM to increase capacity from 4 to 6 requires the migration of a VM (represented by the dashed line). The VM with lowest impact (in terms of migration time – Section III-C2) is selected to be migrated for accommodating the resizing request. We want to highlight that EAVIRA performs all requests aiming VM consolidation (following a best-fit allocation based on residual capacity) in accordance with the cost model discussed in Section III-C1.

2) Reallocating Virtual Infrastructures: A DC stakeholder can periodically invokes EAVIRA to perform a DC consolidation through VI migration. This scenario is exemplified at $t4$ on Fig. 1. EAVIRA migrates VMs to consolidate servers and decreases the energy cost fraction attributed to a provider. Moreover, the VMs consolidation respect the SLA: when allowed, a migration must be performed in accordance to the maximum specified time. For instance, for achieving the consolidation scenario depicted by Fig. 3(d), EAVIRA kept the anchor VM from VI 1 on its original host. However, other VMs were moved around releasing two DC servers (in grey). Latter, both servers are deactivated.

The Algorithm 2 represents the mechanism in the pseudocode form.

Algorithm 2: EAVIRA reallocation algorithm for energy-aware DC consolidation.

```
input : BI
output: Energy consumption
1  resources = get_resources(BI);
2  nodes = {∀neighb; neighb ∈ adj(node) ∧ node ∈ resources};
3  visited = {};
4  for ∀node; node ∈ nodes do
5      nodes = nodes − {node};
6      visited = visited ∪ {node};
7      resources = get_resources(BI);
8      nodes = nodes ∪ {∀neighb; neighb ∈ adj(node) ∧ neighb ∉
       resources ∧ neighb ∉ nodes ∧ neighb ∉ visited};
9      rollback_list = {};
10     for ∀vnode ∈ get_virtual_resources(node) do
11         resources = get_resources(BI, vnode.type);
12         if ¬migrate(vnode, resources) then
13             failed = True;
14             break;
15         else
16             rollback_list = rollback_list ∪ {vnode};
17     if failed then
18         rollback(rollback_list);
19         rollback_list = {};
20 return get_dc_energy_consumption();
```

Initially, all servers hosting the baseline infrastructure are identified (line 1). Thus, because all servers from a BI represent possible migration anchors, EAVIRA aims their consolidation. In this sense, all neighbors of BI are candidates to be deactivated (line 2) after migrating their hosted VMs. For each neighbor (line 4), EAVIRA marks it as visited (line 6) and tries to migrate all hosted VMs to the BI servers (lines 10 to 15). When a migration is not possible (SLA infringement), EAVIRA discards the target host as candidate (line 16) and all process is rollbacked. It is worthwhile to highlight that the consolidation algorithm is based on VM migration time prediction for performing the analysis, without moving VMs around. If a better scenario is found based on DC consumption, then the migration plan is deployed and VMs are migrated.

At the end of EAVIRA consolidation algorithm, the active resources fraction on a DC will be less energy consuming than or equal to the previous scenario. Indeed, in the worst case the DC load will not change, and in the best case, the number of active servers will be reduced guided by the proportional

sharing of energy costs. If there are consolidations, then the same amount of virtual resources will remain allocated.

In short, for performing a VM migration, EAVIRA requires: *(i)* the maximum migration time constraint defined in SLA is respected; *(ii)* there is a destination host with enough available resources; and *(iii)* all VMs connected to the migrated resource have their links reconnected to the destination host. Otherwise, the migration is not performed.

IV. Simulations and Analysis

EAVIRA is agnostic to online allocation mechanism applied to find an initial solution. It focus resides on performing elasticity updates and DC consolidation based on proportional sharing of energy costs. Based on the literature review, the VITreeM [17] and MBFD [3] algorithms were selected for composing the simulation scenario. With regard to reallocation mechanisms, EAVIRA can be compared to MM mechanism [3]. In summary, the simulation scenario comprehends the execution of VITreeM and MBFD for allocation; EAVIRA and MM for reallocation. Considering Fig. 1, VITreeM and MBFD are assigned of the allocations at $t0$ and $t1$, while EAVIRA and MM are invoked for replication, resizing, consolidation, and release. In addition to allocation and reallocation mechanisms, a discrete event simulator (for controlling and distributing VI requests) was implemented[1]. VITreeM, MM and MBFD mechanisms are not detailed on this paper. Further information is available on [17] [3]. Finally, for performing a fair analysis, MM and MBFD were updated for allocating and reconfiguring virtual links (a shortest-path is calculated considering the bandwidth requirement), and for respecting the migration time SLA.

A. Metrics

Five metrics were selected for analysis and comparison between the mechanisms: *(i)* Acceptance ratio of allocation requests, identifying the number of requests which were allocated. One can expects an increasing on this ratio as the reallocation mechanisms consolidate DC load. *(ii)* Energy consumption guided by the proportional sharing model [8]. This metric quantifies the beneficial impact introduced by EAVIRA DC consolidation. *(iii)* DC load in terms of CPU, storage, memory, and network bandwidth. *(iv)* SLA violations representing the number of events violating the vCPU allocation due to servers consolidation. *(v)* the number of VMs migrations performed for load balancing and consolidation. The last two metrics are proportionally calculated by the number of accepted VIs.

B. Parameterization

1) Cloud DC: The DC topology adopted on out experiments represents the Grid'5000 private cloud [25], specifically, the Grenoble DC. On our simulations, servers were homogeneously defined, 107 in total, interconnected by 5 switches. Each server was equipped with two processors (12 cores each), 256 GB RAM, 1 TB storage, and 1 Gbps networking interface. Switches were interconnected by 3 Gbps links. The amount of energy consumption, required for calculating the cost, is based on [8], and is used for all mechanisms (118.11 watts and 202.43 watts for minimum and maximum consumption, respectively).

2) VI Requests: Concerning the VI requests, Amazon EC2 and Google Computing Engine introduced the dynamic provision of Virtual Private Clouds (VPCs), composed of a subset of access point rules and a set of VMs attached to it, composing private local networks that are managed by the tenant. Each VPC requires 1 virtual switch and 5 VMs. The capacity for composing VI requests are based on commonly used $m3.large$ VM instances from Amazon EC2 [26] requiring 2 vCPUs, 8 GB RAM, and 32 GB storage. The bandwidth for communicating VMs is selected between 5, 10, and 15 Mbps. A total of 300 VIs are requested and each VI remains hosted between 5 and 10 discrete intervals. For simulating elastic reconfiguration requests, each VM can be upgraded/downgraded between 10% and 40% of its running capacity. The selection of VI configurations follows a uniform distribution for each item.

3) Algorithms and Configurations: We compared MBFD and VITreeM performing allocation. VITreeM was configured using best-fit ordering of residual capacity for servers selection for improving DC consolidation [17], while MM usage thresholds were configured as 20% and 80% for under- and over-utilization, respectively. Both algorithms perform the allocation considering the total vCPUs utilization of VMs (100%). Afterwards, each VM simulates an application with variable workload, which is modeled to generate the utilization of CPU (between 50% and 100%) according to a uniformly distributed random variable. The migration anchor was defined as 10% of VMs, while the maximum allowed SLA migration time was uniformly defined between 60 and 600 seconds. In order to predict the VM migration time, as summarized by Alg. 1, VM operating system dirty-page rate is chosen between 100 and 1000, page size is 4096 bytes, and the migration threshold is 30 rounds. Finally, the simulation runs by 1000 discrete internal and allocation, replication and delete requests also follow a uniform arrival.

C. Simulation Results

The set of results comes from crossing allocation and reallocation mechanisms. Energy consumption (Fig. 4) is summarized as average and standard deviation with a 95% confidence interval, while acceptance ratio gives the percentage of allocated VIs (Fig. 5) for the simulation.

Analyzing the energy consumption, Fig. 4 shows the total energy consumption and, based on proportional sharing model [8], the provider's fraction. When a physical resource is completely idle, without host any virtual switch, machine or link, it is considered deactivated and the energy consumption is not accounted. Even allocating more VI requests, EAVIRA reduced the DC energy consumption and the fraction attributed to provider. The reduction is also observed when few VIs

[1]Source code available at: https://bitbucket.org/denivyruck/eavira

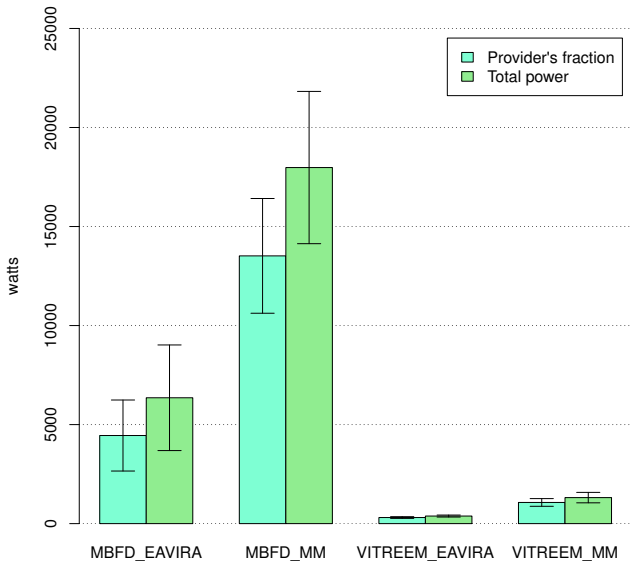


Fig. 4. DC energy consumption.

are allocated (using the VITreeM). In short, a provider using EAVIRA reduces the total energy consumption almost 3 times while the provider's fraction referent to energy cost is decreased by 3.1 times.

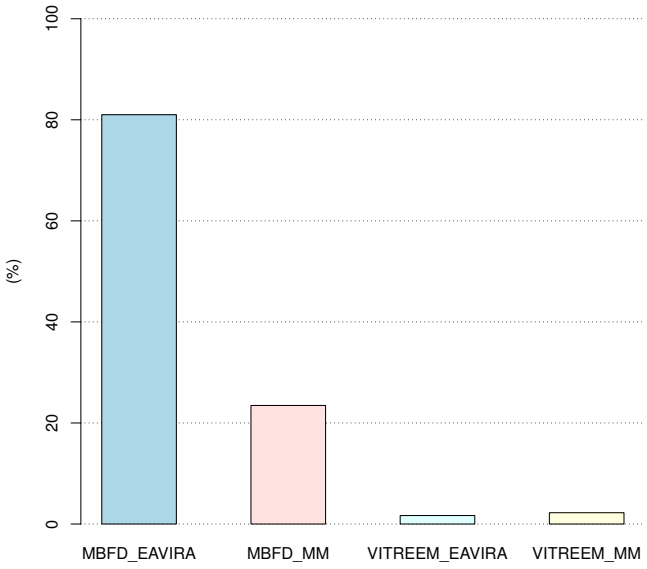


Fig. 5. Acceptance ratio of VI requests.

With regard to the acceptance ratio (Fig. 5), EAVIRA combined with MBFD allocated almost 4 times more VI requests faced to MM reallocation mechanism. Regardless of the reallocation mechanism (EAVIRA or MM), VITreeM resulted on lowest allocation ratio, indicating a barrier when applied for this DC and VI topologies. The DC load, accounted at each discrete event during the simulation, is summarized by Table I. The first quarter (Q1), median (Q2), average ($\overline{x}$), third quarter (Q3), and maximum values are presented for CPU, memory, storage and network bandwidth. The load balancing and consolidation performed by EAVIRA increase the allocation ratio and consequently the DC load. In all cases, the DC load distribution is asymmetric. MBFD_EAVIRA is more stable (without outliers) than other three. On the other hand, MBFD_MM has the most amount of outliers.

TABLE I
DC LOAD: FIRST QUARTER (Q1), MEDIAN (Q2), AVERAGE ($\overline{x}$), THIRD QUARTER (Q3), AND MAXIMUM VALUES.

		MBFD_MM	MBFD_EAVIRA	VITreeM_MM	VITreeM_EAVIRA
vCPU (#)	Q1	226	325.8	27	19
	Q2	236 (9%)	529.5 (20%)	29 (<1%)	20 (<1%)
	$\overline{x}$	217.1	580.9	28.5	19.9
	Q3	244	876.2	30	22
	Max	257	1077	34	23
RAM (GB)	Q1	840	1202	106	73
	Q2	887 (3%)	1959 (7%)	114 (<1%)	81 (<1%)
	$\overline{x}$	820.6	2152.9	113.9	80.6
	Q3	934	3246	122	89
	Max	966	3979	130	89
Storage (GB)	Q1	3288	4706	418	289
	Q2	3479 (3%)	7671 (7%)	450 (<1%)	321 (<1%)
	$\overline{x}$	3220	8427.9	450.2	319.4
	Q3	3670	12702	482	353
	Max	3798	15571	514	353
Net. (Mbps)	Q1	2070	5030	170	50
	Q2	2480 (2%)	8460 (7%)	210 (<1%)	50 (<1%)
	$\overline{x}$	2520.3	9242.7	201.0	53.5
	Q3	2980	14045	250	70
	Max	3530	18070	290	180

TABLE II
SLA VIOLATIONS AND VMS MIGRATIONS ACCOUNTED BY HOSTED VIS.

#	MBFD_EAVIRA	VITreeM_EAVIRA	MBFD_MM	VITreeM_MM
Mig.	10	0	34	287
SLA	562	97	0	9

However, a drawback from DC consolidation is the increase on SLA violations, as summarized by Table II. The application of EAVIRA increases the number of events in which virtual resources lack on vCPU availability for an application workload between 50% and 100% when compared to MM. MM softens the situation by reactively migrating VMs around the DC. However, even avoiding SLA violations, an elevated number of VM migrations can harm the performance of hosted applications. Surprisingly, with few VMs allocated atop a DC (using VITreeM), MM exaggerates the number of migrations, constantly moving the VIs.

V. Considerations & Future work

Regardless of the number of servers composing a DC, private and public cloud providers share the savings of energy consumption as a common long-term objective. It is not new the hardness of such a task. Moreover, the elastic provisioning introduced by cloud computing increased the problem dimensionality as a simple reconfiguration request can unbalance the DC load and increase energy consumption. In this context, the present work introduced EAVIRA, an energy-aware virtual infrastructure reallocation algorithm. EAVIRA attends reconfiguration and replication requests, complementary executing VMs migrations when need. The DC consolidation is guided by a proportional sharing of CPU usage and related energy consumption. In short, the cost model enables a decrease on provider's fraction of total cost by proportionally sharing among them the CPU energy consumption of VMs.

The experimental analysis investigated EAVIRA combined with two online allocation algorithms, and compared the performance faced to a VM migration-based mechanism. EAVIRA increased the acceptance ratio of VI requests and consolidated the DC resources. The promising results opened opportunities for future work. Initially, an analysis considering the tenant's perspective can help on understanding the tread-off between VM migrations and SLA violations. Potentially, this decision should be guided by hosted application impact. Complementary, the dynamism of virtual network load can be used to guide the elastic provisioning of virtual links. The update on virtual communication pattern can force the migration of VMs to respect the SLA.

As future work we are working on: *(i)* A study comprising other allocation mechanisms as well as experiments with different DC topologies; and *(ii)* An analysis of the tread-off between number of VMs migrations and SLA violations quantifying the tenant's perspective.

Acknowledgment

Experiments were carried out on Grid'5000, while development occurred in LabP2D (UDESC/FAPESC).

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, NIST SP 800-145, 2011.

[2] R. Righi, V. Rodrigues, A. da Costa, G. Galante, L. Bona, and T. Ferreto, "Autoelastic: Automatic resource elasticity for high performance applications in the cloud," *Cloud Computing, IEEE*, no. 99, 2015.

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

[4] E. Amaldi, S. Coniglio, A. M. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Electronic Notes in Discrete Mathematics*, vol. 52, pp. 213–220, 2016.

[5] H. Duan, C. Chen, G. Min, and Y. Wu, "Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems," *Future Generation Computer Systems*, 2016.

[6] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, no. 1, pp. 7–18, 2010.

[7] R. Buyya, A. Beloglazov, and J. H. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," in *Proc. of Int. Conf. on Parallel and Dist. Processing Techniques and Applications*, July 2010.

[8] M. Hinz, C. C. Miers, M. A. Pillon, and G. P. Koslovski, "Um modelo de custo para nuvens IaaS baseado no consumo de energia de máquinas virtuais," in *Simpósio Brasileiro de Sistemas de Informação*, May 2016.

[9] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, Fourth 2013.

[10] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 1–9.

[11] R. Righi, "Elasticidade em cloud computing: conceito, estado da arte e novos desafios," *Revista Brasileira de Computação Aplicada*, vol. 5, no. 2, pp. 2–17, 2013.

[12] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud," in *31st Int. Conf. on Distributed Computing Systems*, June 2011, pp. 559–570.

[13] J. Sauvé, F. Marques, A. Moura, M. Sampaio, J. Jornada, and E. Radziuk, "Sla design from a business perspective," in *Int. Workshop on Distributed Systems: Operations and Management*, 2005.

[14] S. Akoush, R. Sohan, A. Rice, A. W. Moore, and A. Hopper, "Predicting the Performance of Virtual Machine Migration," in *MASCOTS, 2010 IEEE Int. Symp. on*, Aug 2010, pp. 37–46.

[15] R. Pfitscher, M. Pillon, and R. Obelheiro, "Diagnóstico do Provisionamento de Recursos para Máquinas Virtuais em Nuvens IaaS," in *Simp. Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2013.

[16] D. Darolt, F. Souza, and G. Koslovski, "Explorando a elasticidade de nuvens IaaS para reconfigurar dinamicamente aplicações n-camadas," *Rev. Brasileira de Computação Aplicada*, vol. 8, no. 2, pp. 2–15, 2016.

[17] R. de Oliveira and G. P. Koslovski, "A tree-based algorithm for virtual infrastructure allocation with joint virtual machine and network requirements," *Int. Journal of Network Management*, vol. 27, no. 1, 2017.

[18] N. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 783–791.

[19] Y. Zhu and M. H. Ammar, "Algorithms for assigning substrate network resources to virtual network components." in *INFOCOM*, 2006.

[20] R. J. Pfitscher, M. A. Pillon, and R. R. Obelheiro, "Customer-oriented diagnosis of memory provisioning for IaaS clouds," *SIGOPS Oper. Syst. Rev.*, vol. 48, no. 1, pp. 2–10, May 2014.

[21] M. D. Assuncao, L. Lefevre, and F. Rossigneux, "On the impact of advance reservations for energy-aware provisioning of bare-metal cloud resources," in *Int. Cont. on Net. and Service Management*, October 2016.

[22] G. Galante and L. de Bona, "A survey on cloud computing elasticity," in *Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on*, Nov 2012, pp. 263–270.

[23] D. B. Ruck, R. Oliveira, and G. P. Koslovski, "Comparação de algoritmos para alocação de Infraestruturas Virtuais," *Revista Brasileira de Computação Aplicada*, vol. 6, no. 2, pp. 98–112, Oct 2014.

[24] 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 the Symp. on Networked Systems Design & Implementation*, 2005.

[25] D. Balouek, A. Carpen Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Pérez, F. Quesnel, C. Rohr, and L. Sarzyniec, "Adding virtualization capabilities to the Grid'5000 testbed," in *Cloud Computing and Services Science*, ser. Communications in Computer and Information Science, I. Ivanov, M. Sinderen, F. Leymann, and T. Shan, Eds. Springer International Publishing, 2013, vol. 367, pp. 3–20.

[26] V. Persico, P. Marchetta, A. Botta, and A. Pescape, "Measuring network throughput in the cloud: The case of amazon EC2," *Computer Networks*, vol. 93, Part 3, pp. 408 – 422, 2015.