Elastic Allocation and Automatic Migration Scheme for Virtual Machines

Hugo E. T. Carvalho and Otto C. M. B. Duarte
Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil
Email: {hugo, otto}@gta.ufrj.br

***Abstract*— Cloud computing offers on-demand access to computational resources. One of the major challenges in cloud environments is to enforce the elasticity of the processes that execute in the cloud, avoiding Service Level Agreements (SLAs) violations and reducing waste with idle resources. We propose an autonomic resource management system for cloud computing, called VOLTAIC (Volume Optimization Layer To AssIgn Cloud resources). The proposal analyzes usage profiles of physical and virtual elements and defines heuristics based on differential utilization level that guarantee an enhanced allocation of virtual elements. VOLTAIC introduces algorithms to determine proper parameters to allocate cloud elements and to automatically migrate those elements to avoid performance degradation due to server saturation. Results obtained through the implementation of the system in a small-scale environment show that the system efficiently assigns virtual elements and ensures proper resource allocation to virtual elements. We also developed a virtual network simulator for cloud environments to attest the high performance of VOLTAIC in broader scenarios. Results show improvements in up to 10% in the amount of offered cycles due to correct assignment of virtual elements.**

I. Introduction

Cloud computing introduces a new provisioning model for technology infrastructure. In this model, clients hire providers that dynamically offer processor, memory, disk, and network resources. This can be achieved through the use of the virtualization technology [1], which implements a hardware abstraction that enhances the flexibility of resource allocation. This flexibility confers elasticity in the cloud environment, defined as the capacity of providing resources on-demand and at the same time ensuring Quality of Service (QoS) of clients [2]. Service providers must develop efficient cloud systems to avoid the waste with idle resources, ensure QoS and fulfill dynamic workload demands. Armbrust *et al.* [3] claim that the main cloud challenges are the service availability and the elastic resource provision that scales with the demand and reduces costs without violating Service Level Agreements (SLA).

Virtualization allows on-demand remapping of virtual resources over physical resources and thus enables the adaptation of systems to dynamic workloads. This remapping primitive is defined as migration and allows workload transfers among different physical machines without interrupting their execution. Currently, migration is manually triggered by network managers to load balance data centers [10]. This reallocation scheme is inefficient because of its high reaction time, which is inadequate to dynamic workload environments like clouds. The autonomic migration is an even greater challenge, because there is a need to consider multiple parameters of the current machines and to estimate future resource demands of machines.

In this article, we propose VOLTAIC, which performs autonomic migrations to provide elasticity in the resource provisioning of a cloud environment, guaranteeing QoS for the clients and enhancing the usage of available resources. Using the profile analysis of virtual and physical elements, the system performs a dynamic allocation of elements. The utilization profiles of virtual elements are compared among themselves and among the profiles of resources offered by the physical machines. VOLTAIC searches the most adequate physical server to each virtual element by considering the likelihood between the profile of the virtual element and the profile offered by the physical server.

VOLTAIC was implemented and tested in a real environment and uses Libvirt API [4]. Thus, VOLTAIC is applicable to all virtualization platforms that supports Libvirt, such as Xen [5], VMWare [6], KVM [7], etc. In order to validate the proposed system in large scale environments, we developed a cloud environment simulator. The simulator receives utilization profiles of real machines and generates outputs that validate the modeling for the proposed scenario. The obtained results show that VOLTAIC is efficient for elasticity provision and enhances the availability of resources. The proposal reduces in up to 10% the denial rate of processor resources when compared to proposals of the literature.

The article is structured as follows. Section II presents the related work, which aims in management of cloud environments and migration of virtual elements. Section III shows the proposed architecture, its behavior and the proposed algorithms. Section IV and Section V shows the implementation of VOLTAIC, the development of the simulator and the results of the proposal. Finally, Section VI presents the conclusions and future directions of this work.

This paper is based on "VOLTAIC : Volume Optimization Layer To AssIgn Cloud resources," by Carvalho, H. E. T., and Duarte, O. C. M. B, which appeared in the International Conference on Information and Communication systems (ICICS12), Jordan, Irbid, April 2012.

doi:10.4304/jetwi.4.4.333-342

II. Related Work

The development of mechanisms that ensure elasticity in resource provision is a big challenge. There are many works that address the virtual element allocation in physical substrates, but most of the proposals focus on the admission control of virtual elements and ignore the resource consumption variability that requires dynamic reallocations. Fajjari *et al.* developed an admission system based on ant colony meta-heuristics to solve this kind of problem [8]. Alkmim *et al.* developed mapping algorithms that minimizes the resource utilization in virtual network environments [9]. The minimization goal is to optimize the resource utilization by avoiding the instantiation of unneeded nodes. For instance, if it's possible to fulfill all virtual network needs in a smaller number of physical resources, the virtual elements can be consolidated in a smaller number of physical elements thus minimizing the resource waste.

SandPiper [10] is a system that monitors virtual machines with the objective of detecting and fixing hotspots in physical servers. Hotspots are defined as the unavailability of resources in physical servers. This unavailability causes degradation in the performance of virtual machines which share resources in these physical servers. The results of the proposal demonstrate that singular hotspots can be detected and mitigated in less than 20 seconds and that the proposal can be extended to data center scenarios. The proposal is tested with some artificially generated workload, generated with Httperf. This artificial workload states that clients make requests to virtual machines and each small request demands lots of processing power. Besides, the proposal offers two monitoring approaches. The first approach is the black-box approach, where the monitoring happens independently of the operational systems and the applications which execute in virtual machines. The second approach is the gray-box approach, which explores processor behaviors from the executing virtual machines.

By detecting the hotspots, SandPiper applies an interactive algorithm which orders servers as function of their volumes, defined as

$$Vol = \frac{1}{1-cpu} \cdot \frac{1}{1-memory} \cdot \frac{1}{1-network}, \quad (1)$$

which is a value that represents the volume of used resources as a function of processor, memory and network. After the ordering procedure, the algorithm classifies, within each machine, the virtual elements that use more resources. Then, the system iteratively reallocates the virtual elements that belong to higher volume machines into lower volume machines, until all hotspots are mitigated. The resource allocation of VOLTAIC is significantly different from SandPiper because it takes into consideration the compatibility of the usage similarity of physical and virtual machines. Besides, our proposal uses a more flexible volume metric that allows a better pondering in the importance of each resource in the accounting of the system load. The new parameters indicate the need to execute management algorithms before the environment reaches critical situations. VOLTAIC also allows the utilization of an adaptation of the punishment algorithm proposed by Carvalho *et al.* to enforce that VOLTAIC algorithms can achieve enough processing power to fulfill its objectives [11].

Violin [12] presents a proposal that is similar to SandPiper. It provides an environment capable of scaling dynamically and migrating elements. The proposal focuses on the utilization of the memory ballooning mechanism, which allows the dynamic memory allocation, and the processor scheduling of Xen platform to deliver resources to virtual elements. The proposal uses relocation policies that verify if a given policy of a virtual element can be fulfilled in the current physical node. If it is not possible, Violin migrates the virtual element to another physical node. The proposed environment is composed of virtual machines connected through a virtual network which allows the separation of the management of Violin from the management tasks of the physical infrastructure. The proposal is divided in two main components:

1) **Enabling Mechanisms:** The enabling mechanisms include the virtual environments from users and the resource monitoring processes of physical machines. These processes monitor the processor and memory consumption through hypervisor calls to detect the availability of resources;
2) **Adaption Manager:** The adaption manager communicates with monitoring process to generate a global view of resource availability. The global view deals with monitoring information of all Violin instances.

The proposal does not offer optimal resource allocations but instead uses relocation based policies to verify if a given virtual element can have its policies fulfilled in the current physical node. If the policies are not fulfilled, Violin migrates the virtual element to a different physical node capable of providing the needed policies.

Gong *et al.* propose Press (PRedictive Elastic ReSource Scaling for cloud systems) [13]. The system is focused on cloud environments where elasticity must minimize the operational costs of providers and at the same time enforce service level objective (SLOs), defined as key elements of the SLAs established between providers and clients. The SLOs provide ways to measure the performance of service providers in a manner that allows both sides to attest if the SLAs are violated. The greatest challenge of the elasticity is to decide where and when to allocate resources, which is a non-trivial problem because application demands can vary over time, hardening the characterization of its patterns. The objective of Press is to develop an efficient prediction schema based on models that avoid the complexity of profile analysis and model calibration.

In order to predict resource consumption, Press uses two complementary techniques. First, it uses signal processing techniques to identify repetitive patters, called signatures, which are used in prediction. If the techniques

are not able to identify signatures, Press uses a statistic approach to detect short-term patterns and applies Markov chains to predict the near future resource consumption. The proposal avoids sub-estimation of resources and tolerates overestimation to reduce SLO violations.

Press is compatible with Xen platform and validates its results through Google cluster data. The system presents good prediction results. One of the greatest contributions of the authors is the idea of using utilization signatures, which take into account the profile variation as a metric to know the reliability of the current profiles when they are used as near future predictors.

Hirofuchi *et al.* developed an efficient migration algorithm based on post-copy migration to optimize migration in cloud environments [14]. Authors say that current migration algorithms are based on iterative pre-copy migration which may require an unpredictable amount of time to finish migration process. In data center scenarios this time unpredictability harms the efficiency of migration algorithms and management strategies. By using the implementation of Hirofuchi, it is possible to use post-copy migration on KVM, which is based on the idea of migrating virtual machine states first and then copy memory pages. This migration enables an accurate prediction of migration time and allows faster migrations because it only depends on the time to transfer the entire memory and the time to transfer the virtual machine states. Given the post-copy algorithm implementation, the authors develop a machine consolidation mechanism that uses post-copy and an allocation scheme to ensure performance. They define dedicated and shared servers. All virtual machines execute in shared servers. When a given machine uses more than a defined threshold, the system wakes a dedicated server and migrates the virtual machine to it. The proposal cannot predict efficiently the resource demands and can lead the system to situations where there are no sufficient dedicated servers to fulfill the resource demands. In this case the proposed system may suffer performance problems. In this case, a solution like VOLTAIC can relocate virtual machines according to their consumption profiles to attenuate the problem.

Houidi *et al.* propose an adaptive mechanism to provide resources in virtual network environments [15]. Authors take into account network restrictions such as topology limitations and SLA constraints to enhance network performance and enhance tolerance to failures. They believe that in the same manner that virtual machines can be allocated dynamically in different physical machines, virtual networks can be dynamically allocated in different machines as well. This extension to virtual networks adds new management parameters such as the connectivity among virtual elements, delays in virtual links and virtual network topologies. The proposed system enables the dynamic relocation of virtual networks as a response of the creation of new networks. The proposed system is distributed and based on agents which monitor physical elements. Agents detect link failures and change virtual network allocation to maintain the constraints of each virtual network.

We have proposed SLAPv, an adaptive control mechanism for virtual network environments [11]. The proposal is based on adaptive punishments applied to virtual elements to enforce SLAs and provide a better utilization of idle resources. SLAPv verifies if virtual machines violate the contracted SLAs and the proposal is limited to the distribution of resources within a single physical node. Another contribution is the adoption of a fuzzy metric to verify the saturation level of physical resource on each physical node, defined as system charge. The system charge reflects parameters such as processor usage, memory and network. This metric was adapted into VOLTAIC as a criterion to execute dynamic allocation mechanisms. Besides, VOLTAIC allows the management of resources in a broader level, because it allows the utilization of the migration primitive to enhance the resource provision for virtual elements. In this manner, the resource control is not restricted to a single physical machine and allows the allocation of resources among different physical elements of a cloud environment. The profiles are not applied to explicitly verify the SLAs. Instead, they are used to understand the variation and the correlation of machine behavior and its time variation, because it is one of the criteria that is applied to select migration candidates in VOLTAIC.

In this paper, we propose VOLTAIC system that is an autonomous resource manager for cloud environments, which allocates virtual elements, enhances the QoS offered to clients and avoids the waste of computational resources. The system charge metric used in our SLAPv proposal was adapted into VOLTAIC as a criterion to execute dynamic allocation mechanisms. The proposal allows resource management in a broader level, because it allows the utilization of the migration primitive to enhance the resource provision for virtual elements. The system uses `libvirt` to interact with virtualization platforms and manage physical machines. Thus, it is compatible with any virtualization platform that supports `libvirt`, such as Xen, VMWare, KVM, etc., differently from existent proposals that work only with specific virtualization environments [10], [11], [13]. By adopting the utilization of a single interface to manage virtual platforms, we can augment the applicability of the proposed system, but it inherits some problems related to which information can be extracted by each platform due to `libvirt` limitations. Besides, results show that the absence of some platforms-specific information does not affect the performance of the proposal. We must also mention that the platforms must provide live migration mechanisms as well.

III. The Voltaic System

The name VOLTAIC is inspired in nature, where electrical charge differences induce charge exchange through voltaic arcs. In the same way real clouds balance electrical charges among each other, the proposed VOLTAIC system

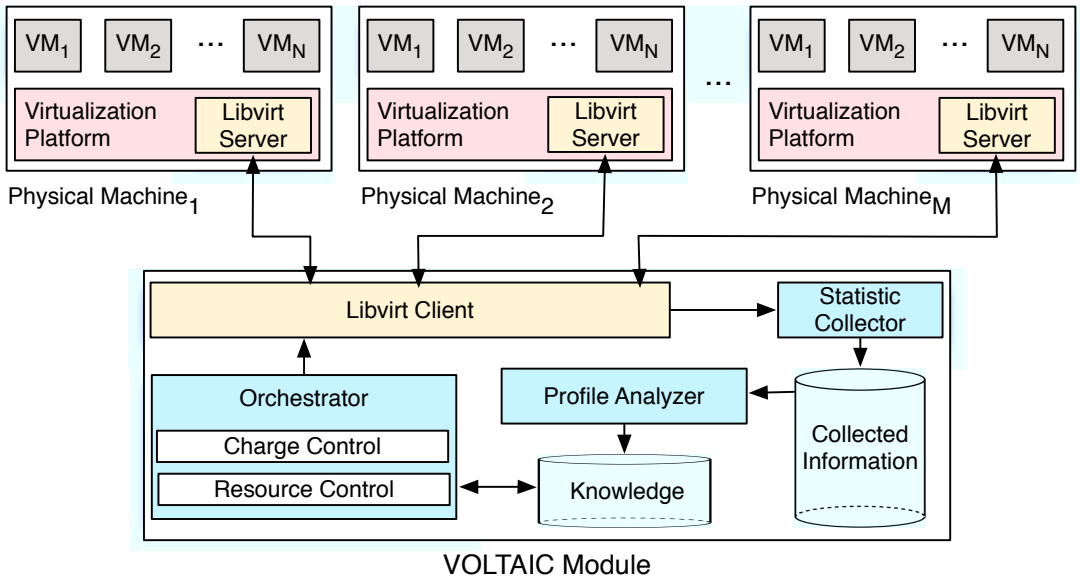

Figure 1. A VOLTAIC module managing a set of physical machines and orchestrating the resource allocation.

balances resource consumption in cloud computing environments. The autonomic management is characterized as a process of observation and decision making in the proposed environment, without human intervention. Therefore, the autonomic manager perceives the behavior of physical and virtual machines, analyzes and predict possible saturation situations, and makes decisions. These decisions can encompass the temporary resource pruning [11] and live migration of virtual elements [16].

The system is composed of three main modules. The first module is the Statistic Collector (SC), which interacts with `libvirt`, retrieves the monitoring statistics of each physical machine, and stores this information in a database. The second module is the Profile Analyzer (PA), which uses the information collected and stored by the Statistic Collector to extract knowledge from the virtualization platforms. This knowledge comprises utilization profiles of virtual elements, offered profiles (OPs) of physical machines, system charge, and the mapping of virtual elements in physical elements. The third module is the Orchestrator (OC) that uses the knowledge acquired by the Profile Analyzer to manage physical and virtual machines. It controls the amount of resources that are offered to each virtual machine and organizes virtual element migrations to balance the resource distribution.

A. VOLTAIC Architecture

The system uses a management model in which a physical machine (PM) configured with a VOLTAIC module manages a given set of physical machines, as seen in Fig. 1. VOLTAIC manages this set of machines, controls the offered resources, and dynamically migrates virtual elements, avoiding resource saturation. If a set of PMs is inside a single administrative domain, it is possible to enable interaction among them, with resource announcements and requisitions. Hence, we extend the system capability, offering resources to virtual elements under domain of other VOLTAIC machines, improving resource utilization and enhancing the provided services.

B. The Statistic Collector Module

The Statistic Collector (SC) module uses `libvirt` to interact with the physical machines and retrieves monitoring information. The SC retrieves processor utilization, allocated memory and network utilization of physical machines and virtual elements. We can adjust the sampling frequencies, avoiding that well defined events synchronizes with times of inactivity in SC retrieval process. The sampling frequency is correlated with the reaction time of the system.

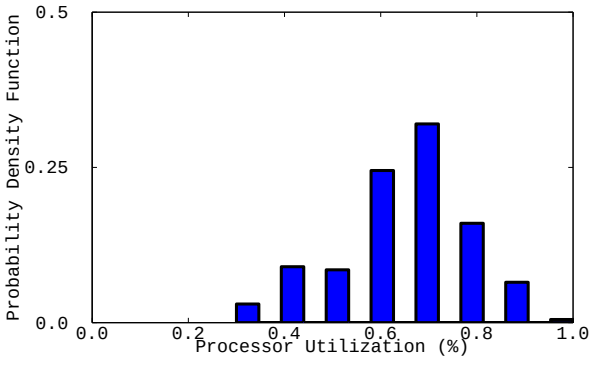

Figure 2. Probability density function (PDF) for a virtual machine executing RIPv2 protocol.

C. The Profile Analyzer Module

The Profile Analyzer (PA) module processes information acquired by the Statistic Collector. The analysis involves the collection of information regarding the

mapping of virtual elements into each physical element and the generation of time series, which reflects the relationship between resource consumption and time. VOLTAIC generates profiles based on cumulative distribution functions (CDFs) and probability density functions (PDFs). These profiles represent the consumption pattern of each virtual machine. We observe in Fig. 2an example extracted from virtual machines executing RIPv2 routing protocol. The profiles represent how the virtual machines use processor resources in time.

These functions allow the estimation of future resource demands of each machine and allows the estimation of a given element be served in a given physical machine. For instance, given a probability distribution of the past activities of the virtual machines, its possible to estimate the amount of resources that will be used in near future.This kind of analysis is based on Sandpiper [10]. Besides the utilization profile, physical machines also possess an offered profile (OP), which represents the profile of availability of resources. This profile is then processed when there is a need to select the proper physical machine to serve a given virtual machine that needs to be migrated.

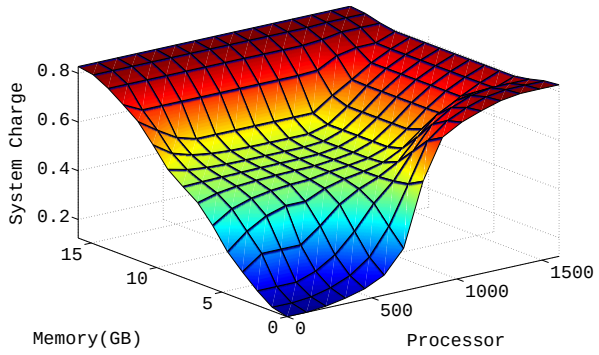

Figure 3. System charge associated with processor and memory utilization.

Based on the profiles and time series, the Profile Analyzer generates a metric defined as system charge. This metric is generated according to Carvalho *et al.* and represents a nebulous conjunction among system variables, such as processor utilization, memory and network utilization [11]. We can see in Fig. 3 a graphical representation of how system charge value varies for different memory and processor utilization values.

Therefore, system managers can model which parameters are the most important in the system charge and how the variation of parameters influences the decision making scheme. Through this metric and its time variation, the system detects physical machines that are close to saturation. By detecting this risk, VOLTAIC pro-actively eliminates the problem through dynamic reallocation of virtual elements.

D. The Orchestrator Module

The Orchestrator is the main module of VOLTAIC. It is responsible for decision making and for machine management. The decision is based on the execution of charge and resource control algorithms. The charge control algorithm examines the variability of the charge of physical systems and detects bottlenecks in resource offering that generates losses. If the average load of the last samples extrapolates a given security threshold, the load control algorithm begins the reallocation procedures. The resource control algorithm allows the system to estimate future resource consumptions and decides the best allocation for each virtual element.

1) Resource Allocation Algorithms: The load control algorithm monitors the load of physical systems and detect if a given physical machine sustained an average load that exceeds a security threshold for a predefined time period. If this happens, migration algorithms are triggered. In the implementation and in the simulation, this period comprises the last five system charge measurements. The security limit was defined as 0.80. This design choice was made because the value of 0.80 minimizes the number of migrations whilst at the same time the system is sensible enough to react to system load variations. Values higher than 0.8 makes the system less reactive to resource utilization variation and thus migrations occur too late and the amount of wasted resources increases. Values lower than the defined value keeps the system working but makes the system hard to converge because it induces more migrations than needed to assure good performance. This limit can assume values in interval $[0, 1]$, which is the range of values that can be assumed by the system load. The critical machine selection algorithm sorts physical machines as a function of their system loads and the amount of critical virtual machines on each physical machine.

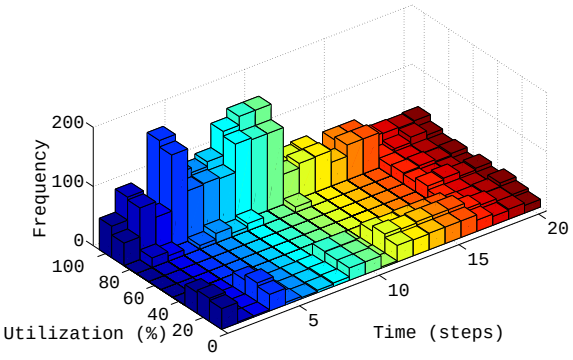

Figure 4. Evolution of time profiles for a workload with low correlation.

Critical virtual machines are those allocated on saturated physical systems and that are responsible for an amount of the system load that is higher than a predetermined limit and which profile variations in time demonstrate low correlation. The profile variation in time is a reflex of the probability that the virtual machine possesses a predictable behavior. If a virtual machine shows high correlation among consecutive profiles, this

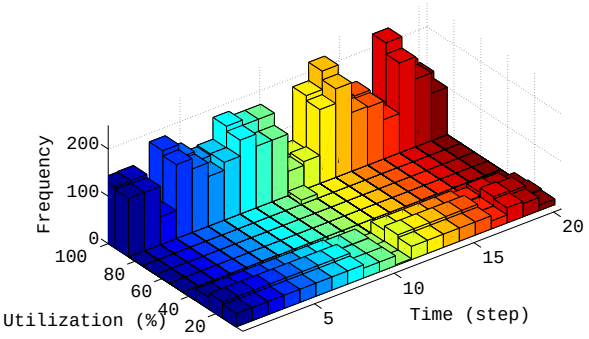

Figure 5. Evolution of time profiles for a workload with high correlation.

input : $SysChargeList[\,], VLimit$
output: $PmMigrationCandidates$
$PmMigrationCandidates = [\,]$;
for $PM \in SysChargeList$ **do**
  $VirtualCriticalNumber = 0$;
  for $VM \in PM$ **do**
    $critic = criticality(VM, VLimit)$;
    if $critic[0] == True$ **then**
      $VirtualCriticalNumber+ = 1$;
    end
  end
  $Info = (MF, VirtualCriticalNumber)$ $result =$
  $PmMigrationCandidates.append(Info)$;
end
$sortByCriticalVMs(result)$;

Figure 6. Selection of critical physical machines.

indicates that this virtual machine has higher chances of behaving in the same way in near future. Otherwise, the machine shows unstable behavior and the proposed algorithm aims to allocate it in another physical machine to avoid the disturbance of well-behaved machines that share the same physical resources. The algorithm can be seen in Fig. 6. In the algorithm, the `Vlimit` input means the established threshold for virtual machines. There, the `criticality()` function evaluates the contribution of the virtual machine in the total load of the physical machine which hosts it and the profile variability of the virtual machine. Criticality is defined as

$$criticidade = \alpha \cdot [\ v.charge\ ] + (1 - \alpha) \cdot [1 - \ abs(\rho)\ ] \quad (2)$$

and establishes a relation of the impact of the virtual machine in the physical machine and the correlation of adjacent profiles of the same machine (represented as ρ). The machine profile is stored as a sliding window of fixed length. In our experiments, α was set as 0.5, which means that the equal weight is given to both charge and correlation. After the evaluation of criticality, the function returns a tuple with the criticality value and a boolean value which indicates if this value violates the established threshold. In Fig. 4, we observe an example of how a virtual machine profile varies in time. In this example, the correlation among profiles is low. In Fig. 5 we observe a virtual machine in which the correlation of subsequent profiles is very high.

The `sortByCriticalVMs()` function groups physical machines with similar charge values and sort this groups in function of the number of critical virtual machines in each of them. Therefore, the first returned elements represent machines with higher charge and more critical virtual machines.

After this procedure, VOLTAIC executes the virtual machine migration selection algorithm. This algorithm, seen in Fig. 7, iterates over physical machine candidates. For each candidate, the algorithm observes if the recent consumption profile and its correlation between the profile and the offered profile (OP) of physical machines, sorted from lower to higher charge. The adopted correlation method is the Pearson correlation, which is obtained by dividing the covariance by the product of standard deviation of two variables. If the profile is correlated and the average consumption of the virtual machine is smaller than the one offered by the physical machine, the virtual machine is migrated. Otherwise, the next virtual machine is analyzed.

input : $PmMigrationCandidates[\,], Plimit, Vlimit$
output: $VmMigrationCandidates[\,]$
$VmMigrationCandidates = [\,]$;
for $PM \in PmsMigrationCandidates$ **do**
  if $PM.charge \geq Plimit$ **then**
    for $VM \in PM$ **do**
      $distribution = getDist(VM)$;
      if $criticality(VM, Vlimit)[0] == True$
      then
        $info = (distribution, VM, PM)$;
        $VmMigrationCandidates.append(info)$;
      end
    end
  end
end
$VmMigrationCandidates.sort(reverse = True)$;

Figure 7. Selection of migration candidates.

IV. Implementation and Simulation

VOLTAIC is implemented in Python and uses *python-libvirt* for virtualization platform communication. The implementation is based on multi-thread programming, where threads are responsible for monitoring each machine. The monitoring stores information in a database. The Profile Analyzer uses this information to generate usage profiles and the system charge.

In order to test a broad range of parameters and perform migration tests in larger scale, we also developed a discrete event simulator for virtual environments. The

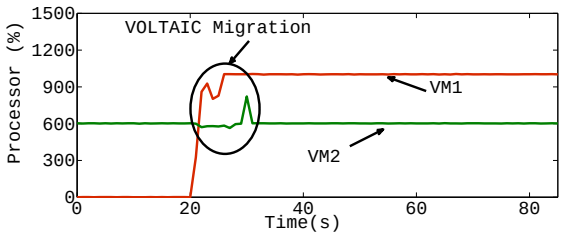

Figure 8. Processing variation as function of time.

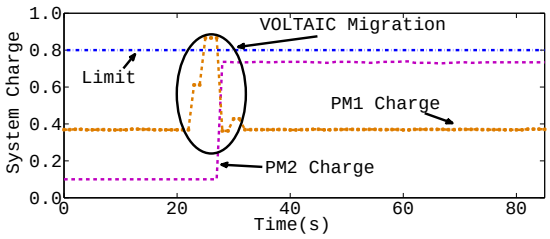

Figure 10. System charge variation.

simulator is developed in python and allows the configuration of physical and virtual machines. The simulator [1] offers creation, destruction, migration, and event scheduling primitives. Each execution step represents a unit of simulation time. The main classes of the simulator are detailed below.

The simulator was developed according to object oriented modeling paradigm. The simplified class diagram cam be seen in Fig. 9. In the diagram, we see the main entities of the model. The simulator is the main entity. It is responsible for storing processing simulation steps. We can configure the number of steps, log the information of each step, configure each physical and virtual machine, and so on. Besides, we can program event triggers to execute methods related with the operation of the simulator. The simulator stores algorithm objects and physical machine objects. Algorithm objects represent all the algorithms that can be used to manage virtualized environments. Physical machine objects represent real physical machines and store virtual machine objects. Finally, the virtual machine entity represents a virtual machine which executes in the system and its resource consumption pattern.

A. Physical Machines

The physical machines entities are similar to real machines. We define the maximum processor capacity, memory, and network resources that are offered on each simulation step and also we associate virtual machines to physical machines. The implementation also simulates a generic virtualization platform, which allows the utilization of virtualization primitives (creation, destruction, and migration), and also the implementation of different resource schedulers. The simulator user adds costs to perform the primitives and schedule tasks.

The implemented processor scheduler makes a fair resource distribution among virtual machines (proportional division). For instance, if the physical machine only provides 100 processing units per step and two machines try to use 100 units each, each machine receives only 50 processing units and the simulator stores that virtual machines lost resources in this interaction. In this case, each machine achieved a loss of 50 processing units.

[1]The simulator is available for download and can be found in http://www.gta.ufrj.br/~hugo/virtsim/.

B. Virtual Machines

Virtual machine entities inherit characteristics from physical machine entities, but there is no implementation of the virtualization platform. We define processor, memory, and network thresholds. Besides, the virtual machine entity allows the utilization of customized profiles for each resource, enabling the injection of real resource consumption patterns in the simulator to see the VOLTAIC reaction to it. We can also opt for distribution functions to generate machine profiles. In this manner, given that the developer possesses knowledge of the types of virtual machines that will be deployed in real scenarios, it is possible to simulate access patterns even without the existence of real profiles.

C. VOLTAIC and the Simulation Manager

In simulation, VOLTAIC possesses its own entity, capable of interacting with physical and virtual machines. VOLTAIC allows the execution of the monitoring tasks, migration algorithms, and migration candidate selection algorithms. This interaction is accomplished through the simulation manager. The simulation manager coordinates all simulation activities. We can define the number of simulation rounds and the number of steps for each round. The simulation manager also helps the system to perform migrations, by invoking sending and reception methods on each physical machine. At the end of the execution, we can generate a log of all operations and procedures that were taken during the simulation.

V. Results

The implementation and simulation results were obtained in two identical physical machines (PM1 and PM2), connected by Intel 82599EB 10 Gbps cards through an optic fiber. The servers have two Intel Xeon X5570 processors, with a total of 16 physical cores and 24 GB of DDR3 1066 MHz memory. The virtualization platform that was used was KVM and the `libvirt` version was 0.9.6.

The test methodology follows the evolution of VOLTAIC system, with the intention to demonstrate the validity of each step of the development of the proposal. Initially, we perform tests that demonstrate the proper working of the proposal when it is implemented in a small real environment. The parameters generated in this small real environment is then used to calibrate the simulator. We capture real utilization profiles from this scenario. The

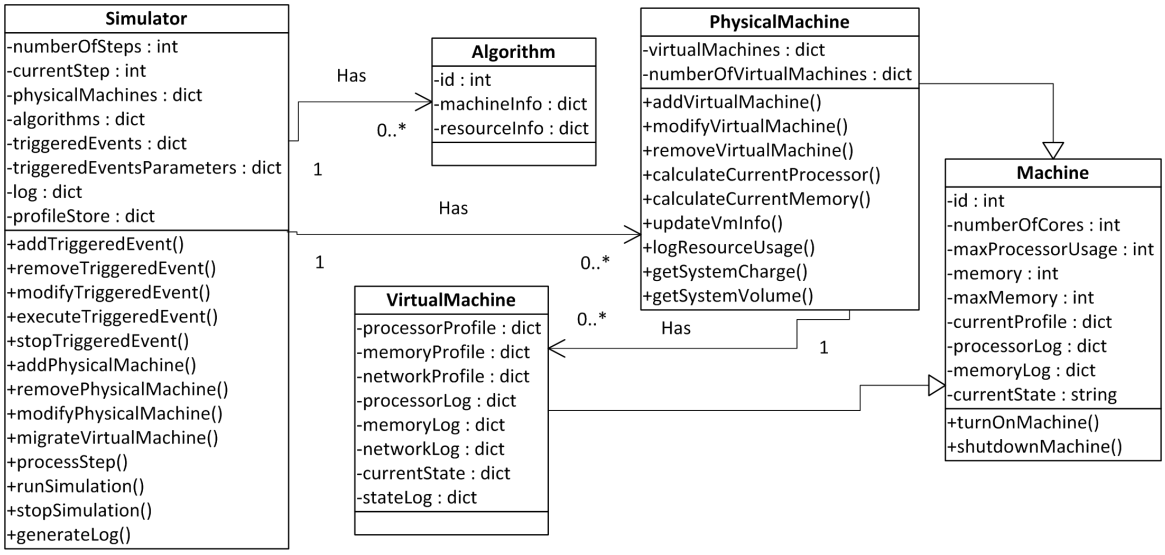

Figure 9. Simplified class diagram of the proposed simulator.

simulator is then fed with the real scenario information, such as the estimated migration time and the resource utilization profiles. Next, we test the behavior of the simulator when it is subjected to real profiles. We observe that the simulator shows a behavior that is equal to the real system implementation. After that, we simulate environments with higher number of virtual and physical machines, to evaluate the performance of the system and the resource allocation schemes in broader scenarios.

In the first implementation test, we created two virtual machines. Each virtual machine receives processing tasks and uses up to 16 of the available cores. If the physical machine cannot provide enough processing, virtual machines suffer from utilization capability reduction until there are enough resources available.

In Fig. 8, we observe the processor utilization of two virtual machines (VM1 and VM2) during time. The processing scale represents the total utilization of processor resources, which varies from zero to 1.600%, which represents the full utilization of all the 16 available cores. Initially the two virtual machines are allocated in physical machine 1. We can see that in the instant *t=20* seconds VM1 receives an intensive processing task, which elevates its processor utilization from 0% to 1.000%. In this situation, the physical machine is totally saturated and even a small positive variation of processor usage can generate performance losses. VOLTAIC detects the augment in the processor demand and opts for automatically migrate VM1, which is the virtual machine that shows higher probability of saturating PM1 to PM2, to fairly distribute the charge among physical machines, as seen in Fig. 10. Therefore, VOLTAIC relocates virtual machines according to the physical machines that can provide the resource profile which is the most compatible with each virtual machine.

To enhance the scale of tests, we developed a virtual environment simulator. To validate the simulator, the same

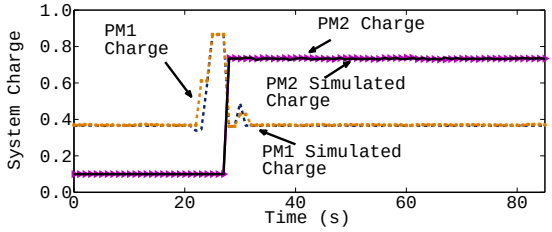

Figure 11. Equivalence between simulation and implementation.

utilization profile from Fig. 8 and the physical machines configurations were reproduced in the simulator and we observed the system charge variation. Results show that for the same real charge, the simulator shows the same behavior in processor variation and in system charge variation, as seen in Fig. 11. The simulation steps were associated to the time in seconds of the real system execution.

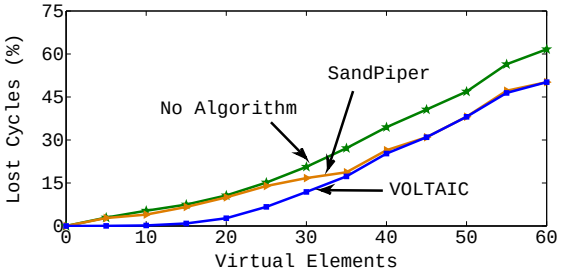

Figure 12. Lost cycles in function of the number of virtual machines.

To demonstrate VOLTAIC, the simulator was loaded with virtual machines with processor utilization profiles that follows a normalized distribution centered in 200% of processor utilization. Each virtual machine is configured with 256 MB of RAM memory and this amount is fixed for all machines. In the simulations, we do not consider the utilization of network interfaces. In the beginning of

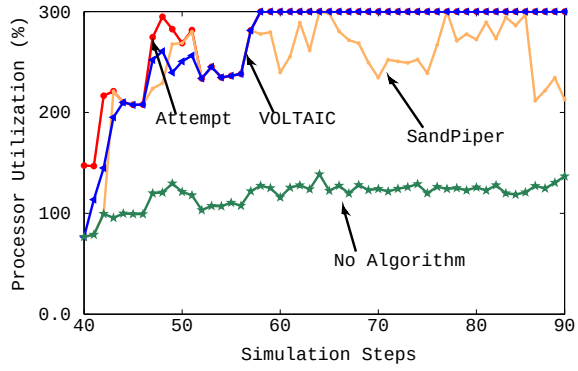

Figure 13. CPU utilization in details. VOLTAIC Outperforms SandPiper and the execution with no algorithms in the CPU provision.

each simulation round the initial allocation of each virtual machine is modified and we verify the amount of lost processing cycles in the system. The lost cycles are related to simulation steps in which resource demand is higher than the current configuration of the system can provide. The lost cycles can be defined as a function of the amount of cycles required by virtual machines and the amount of cycles that each physical machine can provide. The total required cycles of the virtual machines is defined as

$$C_{totalreq} = \sum_{step=1}^{n} \sum_{vm=1}^{m} proc_{vm}^{step} \quad (3)$$

where $proc_{vm}^{step}$ represents the amount of processor cycles required by the virtual machine vm in step $step$. We sum all the required cycles of each virtual machine in each simulation round and then calculate $C_{totalreq}$. In order to define the amount of lost cycles on each simulation step, we define Δ_{pm}^{step} that represents the difference between the resource demand and the resource capability of physical machine pm in step $step$. The formal definition of Δ can be seen in Eq. 4. If the number of required cycles $R_{req,pm}^{step}$ is greater than the available resources $R_{offer,pm}^{step}$, Δ is positive. Positive values of delta mean that there were lost cycles. If Δ is equal to zero, then all the required resources fit exactly on the physical machine capability. If Δ is negative, the physical machine was able to fulfill the cycle demand.

$$\Delta_{pm}^{step} = R_{req,pm}^{step} - R_{offer,pm}^{step} \quad (4)$$

We also define C_{lost}^{step} as the number of cycles that were lost in step $step$, as seen in Eq. 5. In this equation, if there were lost cycles, Δ_{pm}^{step} is positive and then $\frac{\Delta_{pm}^{step}+|\Delta_{pm}^{step}|}{2}$ is equal to the amount of lost cycles. If Δ_{pm}^{step} is less than or equal to zero, then $\frac{\Delta_{pm}^{step}+|\Delta_{pm}^{step}|}{2}$ is equal to 0 and there is no contribution to the number of lost cycles.

$$C_{lost}^{step} = \sum_{pm=1}^{m} \frac{\Delta_{pm}^{step} + |\Delta_{pm}^{step}|}{2} \quad (5)$$

Finally, we define $C_{totallost}$ as the total number of fulfilled cycles divided by the total number of cycles that were requested, as seen in Eq. 6.

$$C_{totallost} = \frac{100}{C_{totalreq}} * \sum_{step=1}^{n} C_{lost}^{step} \quad (6)$$

After the simulation, we develop a statistical analysis of data and compare the amount of lost resources in VOLTAIC, SandPiper, and without autonomic migration algorithms.

The results seen in Fig. 12 show the amount of processing cycles that were lost in function of the number of virtual machines allocated in the physical machines. The value of lost cycles is derived from the application of Eq. 6 in all tested conditions. These results were obtained in the execution of five rounds of 100 simulation steps and we simulated 10 physical machines. The results show that VOLTAIC reduces the amount of lost cycles in more than 10%. We can observe that until the number of virtual machines reaches 35, the proposal presents better results than SandPiper. This occurs mainly because the selection criterion of critical machines takes into account the correlation of the virtual machine profiles and the criticality of virtual machines.

As the number of virtual machines increases, the physical machines became more saturated and eventually all of them are classified as critical. If this happens, the priority heuristic that chooses less stable machines and searches for adequate reception profiles became an algorithm that takes into account only the system charge, because all physical machines present saturation symptoms and critical machines. Even in these conditions, the system reduces the lost cycles in 10% when compared to the results that use no algorithms.

In Fig. 13, we verify the analysis of a random virtual machine in one of the execution rounds of the tests. We can verify the processor consumption attempt, and the processor consumption offered by VOLTAIC, SandPiper, and in the absence of reallocation algorithms. Our proposal, as time evolves, learns the behavior of virtual machines and selects a better placement for it. After 58 simulation steps, VOLTAIC found out the physical machine that better suits the virtual machine and ensure proper resource allocation for it.

VI. Conclusion

Quality of Service and elasticity provision are great challenges for cloud computing. Efficient resource allocation is fundamental for scalability of this computation model. We propose VOLTAIC that is an efficient system to dynamically reallocate virtual elements in physical machines. VOLTAIC analyzes utilization profiles and, based on usage correlations, reduces the amount of wasted processor cycles during normal and saturated scenarios. This reduction can be achieved through automated virtual machine migration.

The proposed heuristics predict saturation situations and trigger the migration algorithms. The algorithms

detect and select the physical machines that are closer to saturation and find the best candidates for machine migration. Results show that the proposed migration algorithms reduce in up to 10% the amount of wasted cycles in the offering of processor resources. Therefore, the results show that the proposal performs well for the analyzed data center scenarios, maximizing the amount of instantiated virtual machines, ensuring the fulfillment of services, avoiding resource waste, and enhancing the profit of providers.

Besides the dynamic allocation proposal, this article brings as a contribution an implementation of a virtual environment simulator. This simulator allows the instantiation of virtual elements and virtualization platforms. The profile of the virtual elements can be loaded from real traces. Finally, the simulator is extensible and allows the development of new schedulers and virtualization proposals. As a future work, we intend to extend the simulator and to implement new heuristics to dynamically allocate resources in cloud computing environments.

REFERENCES

[1] N. Fernandes, M. Moreira, Moraes, *et al.*, "Virtual networks: Isolation, performance, and trends," *Annals of Telecomm.*, pp. 1–17, 2010.

[2] S. Dustdar, Y. Guo, B. Satzger, and H. Truong, "Principles of elastic processes," *Internet Computing, IEEE*, vol. 15, no. 5, pp. 66–71, 2011.

[3] M. Armbrust, A. Fox, R. Griffith, *et al.*, "Above the clouds: A berkeley view of cloud computing," Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Tech. Rep., 2009.

[4] M. Bolte, M. Sievers, Birkenheuer, *et al.*, "Non-intrusive virtualization management using libvirt," in *Proc. of the CDATE*. EDAA, 2010, pp. 574–579.

[5] M. Bourguiba, K. Haddadou, and G. Pujolle, "Evaluating Xen-based virtual routers performance," *International Journal of Communication Networks and Distributed Systems*, vol. 6, no. 3, pp. 268–282, 2011.

[6] E. VMWare, "Server," *GSX Server, product documentation*, 2005.

[7] A. Kivity, Y. Kamay, D. Laor, *et al.*, "KVM: the linux virtual machine monitor," in *Proc. of the Linux Symposium*, vol. 1, 2007, pp. 225–230.

[8] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "VNE-AC: Virtual network embedding algorithm based on ant colony metaheuristic," in *Communications (ICC), 2011*. IEEE, 2011, pp. 1–6.

[9] G. P. Alkmin, D. M. Batista, and N. L. S. Fonseca, "Optimal mapping of virtual networks," *GLOBECOM 2011*, 2011.

[10] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Computer Networks*, vol. 53, no. 17, pp. 2923–2938, 2009.

[11] H. E. T. Carvalho, N. C. F. Fernandes, O. C. M. B. Duarte, and others., "SLAPv: a service level agreement enforcer for virtual networks," in *ICNC'12 - ISA*, Maui, Hawaii, USA, Jan. 2012, pp. 713–717.

[12] P. Ruth, J. Rhee, D. Xu, Kennell, *et al.*, "Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure," in *ICAC'06. IEEE International Conference on*. IEEE, 2006, pp. 5–14.

[13] Z. Gong, X. Gu, and J. Wilkes, "PRESS: Predictive elastic resource scaling for cloud systems," in *Network and Service Management (CNSM), 2010 International Conference on*. IEEE, 2010, pp. 9–16.

[14] T. Hirofuchi, H. Nakada, S. Itoh, and S. Sekiguchi, "Enabling instantaneous relocation of virtual machines with a lightweight vmm extension," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*. IEEE, 2010, pp. 73–83.

[15] I. Houidi, W. Louati, Zeghlache, *et al.*, "Adaptive virtual network provisioning," in *Proc. of the 2nd ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*. ACM, 2010, pp. 41–48.

[16] C. Clark, K. Fraser, S. Hand, Hansen, *et al.*, "Live migration of virtual machines," in *in Proceedings of the 2nd conference on NSDI*. USENIX Association, 2005, pp. 273–286.

Hugo E. T. Carvalho is currently a M.Sc. student at the Electric Engineering Program (PEE) from COPPE and Universidade Federal do Rio de Janeiro (UFRJ), Brazil. He received his Computer and Information Engineer degree from UFRJ, Brazil in 2011. His research interests include network security, server consolidation, machine learning, cloud computing and network management.

Otto C. M. B. Duarte received the Electronic Engineer degree and the M.Sc. degree in electrical engineering from Universidade Federal do Rio de Janeiro, Brazil, in 1976 and 1981, respectively, and the Dr. Ing. degree from ENST/Paris, France, in 1985. Since 1978, he has been a Professor with UFRJ. His major research interests are in QoS guarantees, security and big data.