AGATE: Adaptive Gray Area-based TEchnique to Cluster Virtual Machines with Similar Behavior

Claudia Canali, *Member, IEEE,* and Riccardo Lancellotti, *Member, IEEE*

Abstract—As cloud computing data centers grow in size and complexity to accommodate an increasing number of virtual machines, the scalability of monitoring and management processes becomes a major challenge. Recent research studies show that automatically clustering virtual machines that are similar in terms of resource usage may address the scalability issues of IaaS clouds. Existing solutions provides high clustering accuracy at the cost of very long observation periods, that are not compatible with dynamic cloud scenarios where VMs may frequently join and leave. We propose a novel technique, namely AGATE (Adaptive Gray Area-based TEchnique), that provides accurate clustering results for a subset of VMs after a very short time. This result is achieved by introducing elements of fuzzy logic into the clustering process to identify the VMs with undecided clustering assignment (the so-called gray area), that should be monitored for longer periods. To evaluate the performance of the proposed solution, we apply the technique to multiple case studies with real and synthetic workloads. We demonstrate that our solution can correctly identify the behavior of a high percentage of VMs after few hours of observations, and significantly reduce the data required for monitoring with respect to state-of-the-art solutions.

Index Terms—Cloud Computing, Clustering, Resource Management, Metrics/Measurement.

1 Introduction

Cloud computing has rapidly become a widely adopted paradigm for delivering complex services over the Internet. Both the number of cloud-based services and the complexity of the infrastructures behind them have quickly increased in the last years. The capability of cloud computing infrastructures to cope with the increasing resource demand in the next few years will be critical for the future development of the emerging digital society. Resource monitoring and management are particularly critical tasks in Infrastructure as a Service (IaaS) cloud systems, where a huge and ever-growing amount of data is collected for the management of the virtualized environment hosting customer applications [1]. In these cloud systems, data center administrators typically apply a black-box approach where each virtual machine (VM) is considered as independent from the others, with negative consequences on the scalability of monitoring and management tasks.

Recent studies in literature [2], [3] show that the scalability issues in IaaS cloud systems may be improved by automatically clustering VMs with similar behaviors in terms of resource usage. For example, the automatic determination of classes of similar VMs allows the system to identify for each class few representative VMs, whose behavior is followed by the other members of the same class: this knowledge has been exploited to improve the scalability of monitoring strategies and has recently been applied to a case of VM management that is the server consolidation in IaaS data centers [4]. However, existing clustering techniques [2], [3] show a clear trade-off between the capability to correctly cluster VMs and the length of the resource usage time series needed to determine the VMs behavior: specifically, long time series (up to days of collected measurements) are necessary to achieve an accurate clustering for every VM. The resulting mechanism is poorly reactive to changes in VMs configuration, and may be suitable for quite static scenarios characterized by long-term commitments [5], where cloud customers purchase VMs for extended periods of time (for example, using the Amazon so-called *reserved instances*). On the other hand, the emerging cloud scenario requires solutions that support a dynamic behavior where VMs frequently join and leave the system.

In this paper we present an adaptive technique, namely AGATE (Adaptive Gray Area-based TEchnique), that dynamically selects the length of the resource usage time series used to model the VMs behavior depending on the degree of uncertainty resulting from the clustering process. The proposed technique exploits elements of fuzzy logic for the dynamic determination of the required time series length, leaving uncertainly clustered VMs in a *gray area* of not-yet-clustered items. This solution allows the system to take decisions on the VMs behavior without the need to wait for long monitoring periods, thus leading to a reactive mechanism able to cope with a dynamic environment characterized by deployments and removals of customer VMs in a cloud data center.

A preliminary version of the gray area-based technique was proposed in [6], but this study improves our previous work in several ways. First, the original mechanism was based on a fixed threshold parameter, while in this proposal we adaptively compute the threshold depending on the results of the clustering operation: this greatly reduces the amount of data collected and the time required to determine the VMs behavior. Second, we apply the clustering to short time series (up to 1 hour) and we discuss the novel

• *The authors are with the Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, Modena, Italy.*
E-mail: claudia.canali@unimore.it,riccardo.lancellotti@unimore.it.

issues arisen by the reduced series length. Third, we apply the technique to diverse scenarios with varying demands instead of limiting the evaluation to a single dataset.

The proposed technique is applied to datasets obtained from different scenarios with both synthetic and real workloads. The experiments show that the application of our adaptive technique allows the system to rapidly cluster the majority of the VMs with an accuracy higher than 97% for every considered scenario and without requiring any parameter tuning. We also measure the potential reduction in terms of data collected by the monitoring system with respect to existing solutions. Finally, we consider the specific case of clustering based on short resource usage time series: we give insights on the best clustering algorithm to be integrated in the proposed technique, showing the benefits of an approach based on the correlation between usage measurements.

The remainder of this paper is organized as follows. Section 2 describes the reference scenario for the application of VM clustering and motivates the need for an adaptive technique. Section 3 presents the proposed technique, while Section 4 details the issues related to cluster VMs based on short time series. Section 5 describes the experimental results. Section 6 discusses the related work and Section 7 concludes the paper with some final remarks.

2 VMs Clustering in cloud systems

The reference system for the proposed technique is an IaaS cloud data center where customer applications are hosted in a virtualized environment. We anticipate that the reference scenario described in this section has been implemented in the data center of a small cloud provider. In such systems, each customer application typically consists of multiple software components (e.g., the tiers of a multi-tier Web application), and each component runs on a separate VM: automatic clustering aims to identify VMs running the same software component of the same customer application. Throughout this section, we discuss the potential benefits of applying VMs automatic clustering to an IaaS cloud system. Then, we describe how VMs clustering can be integrated in an IaaS cloud system to achieve scalable monitoring and management. Finally, we motivate the need for an adaptive technique that overcomes the issues of the existing clustering techniques.

2.1 Benefits of cluster-based approach

Providers and administrators of cloud IaaS systems typically consider each VM as a black-box that needs to be monitored and managed independently from other VMs, thus exacerbating the scalability issues of these tasks. The capability of automatically clustering similar VMs may improve monitoring and management tasks of a cloud computing system in several ways. Specifically, potential benefits can be achieved in terms of:

- scalability of the monitoring system,
- scalability of the server consolidation process,
- efficiency of system resource demand estimation.

As regards monitoring scalability, exploiting the knowledge of VMs clusters allows the monitoring system to significantly reduce the amount of data collected by introducing a mechanism of differentiated sampling frequencies for the VMs, as discussed in [2], [3]. Basically, few representative VMs are selected for each identified class as soon as the clustering is done. Only the representative VMs of each class are monitored with high sampling frequency to collect information for resource management purposes, while the resource usage of the other VMs of the same class is assumed to follow the representatives behavior. On the other hand, the non representative VMs of each class are monitored with coarser granularity to identify behavioral drifts that could determine a change of class. This cluster-based approach and its integration in an IaaS cloud system are described in detail in Section 2.2.

The application of an automatic clustering technique may also improve operations related to the server consolidation process used for cloud resource management. Server consolidation is a widely used approach [7], [8] to minimize the number of servers that an infrastructure requires, thus avoiding waste of resources. VMs clustering can be applied to improve the scalability of a critical step of server consolidation, that is the solution of a multi-dimensional bin-packing problem where each VM must be assigned to one server without exceeding its capacity in terms of available resources [7], [9]. This problem is typically solved using heuristics and simplifications (for example, considering only one resource such as the CPU [10]). We argue that, by knowing classes of similar VMs, a cluster-based consolidation can be applied to solve a much smaller optimization problem (with the possibility to use global optimizations rather than heuristics [11]); then, the solution can be replicated as a building block to obtain the global placement solution for the consolidation of the whole data center [4].

Another critical aspect of server consolidation is the need to correctly estimate the VMs resource demand to allocate them on physical servers, minimizing the total number of servers and avoiding overload conditions at the same time. This estimation must consider current VMs demands and predict their evolution in the near future. Recent studies in literature propose solutions to predict the resource usage patterns based on historical data [12] and to aggregate VM-level information by means of a hierarchical management architecture [13]. However, these operations are hindered by the black-box vision of IaaS providers, that are usually not aware of the application running on each VM. On the other hand, knowing which VMs are running the same software component allows the system to perform an application profiling; moreover, the analysis of each application can be limited to few selected cluster representatives. For these reasons, a cluster-based approach may improve both accuracy and scalability of VMs demand estimation.

2.2 System model

We now detail the scenario of an IaaS system that exploits VMs automatic clustering to achieve a scalable approach for cloud monitoring and management. To this aim, we refer to the prototype implementation deployed on the data center of a small cloud provider. The considered system adopts a

two-level management strategy, which is a widely adopted solution in IaaS clouds [14], [15]. The first level consists in a *local management*, that is performed on each physical server of the data center: it detects overload conditions in real-time making use of the VMs resource measurements hosted on the server, and exploits live VMs migration whenever overloaded servers are detected as in [16]. The second level is a *global management*, which is hosted on a management node: it is responsible for periodically executing a consolidation technique to place VMs on as few servers as possible to reduce infrastructure costs and avoid expensive resource over-provisioning, as described in [4].

A VMs clustering technique [2], [3] is integrated in this system to automatically group together similar VMs, that are VMs running the same software component of the same customer application. For scalability reasons, clustering is applied to the VMs of the same customer. After the clustering, few representatives are selected for each identified class. In our prototype, we choose to select at least three representatives due to the possibility that a selected representative unexpectedly changes its behavior with respect to its class: quorum-based techniques can be exploited to cope with byzantine failures of representative VMs [17]. At this point, the representative VMs of each class are monitored with high sampling frequency (that is, 1 sample every minute) to collect information for the periodic consolidation task. On the contrary, the non representative VMs are assumed to follow the behavior of the representatives of the same class, and are monitored with coarser granularity (in our system, one sample every 30 minutes) to identify behavioral drifts that could determine a change of class. It is worth to note that the fine-grained collection of data for clustering purposes, and the subsequent VMs clustering, are repeated after a given period of time; the frequency of the VMs clustering is not correlated with the frequency of the consolidation task (in our prototype we run clustering every hour, while the consolidation task runs every two hours). Moreover, the clustering process can be triggered whenever new unclustered VMs are detected in the system: this situation may occur either when newly deployed VMs enter the system or when previously clustered VMs are marked as "unclassified". A VM is unclassified if it changed its behavior with respect to the class it belongs to (for example, due to software faults), or caused overload of the physical server. In these cases, the VMs of the system are monitored again with high sampling frequency to perform re-clustering.

Let us now detail the interactions among the main components of the system (shown in Figure 1). The *monitoring* process on each server collects data about resource usage of the hosted VMs and sends them to the *local management* system (arrow 1). The *local management* is responsible for triggering live VMs migration in case of server overload as in [16]: this task is accomplished with the *local management* system issuing a query to the *global management* system, although other options, such as relying on a local exchange of load information among neighbors, may be adopted.

Then, the *monitoring* process sends data to the *VMs clustering* system in the management node (2), which automatically groups similar VMs applying one of the clustering techniques proposed in [2], [3]. The clustering results (VM classes and representatives) are sent to the *global management* ($3a$) and to the *monitoring* system ($3b$). The *monitoring* system exploits this information to differentiate the sampling frequency between representative and non representative VMs. The data collected with different granularity are sent to the *global management* system (4) which is responsible for two tasks. First, it periodically executes the cluster-based consolidation strategy (as described in [4]), exploiting the resource usage of the representative VMs to characterize the behavior of every VM of the same class: the consolidation decisions are finally communicated to the *local management* system (5) to be executed. Second, the *global management* system checks for behavioral drifts of non representative VMs with respect to the class they belong to.

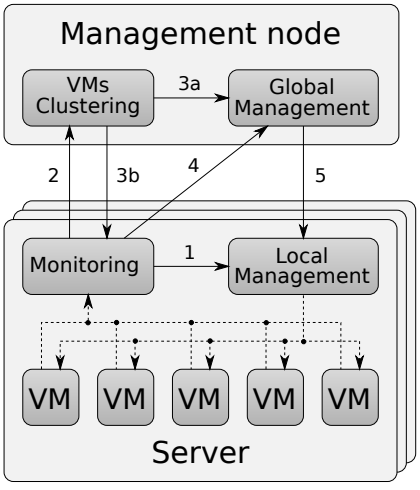


Fig. 1: Cloud system using VMs clustering

2.3 Background on clustering

Throughout our analysis we focus on two different approaches, namely *Bhattacharyya-based* [2] and *PCA-based* [3] clustering techniques, that have been identified in previous works by the authors as the most promising approaches for VMs clustering. Both techniques share the common trait of taking as input the usage time series of multiple resources for every VM. From these time series a behavior description for each VM is computed and used for a final clustering step, that groups together VMs with similar behavior. The two clustering techniques differ for the algorithms they use to model the VMs behavior and to cluster similar VMs.

The Bhattacharyya-based clustering technique exploits histograms to model the VM behavior. One histogram is generated from the usage time series of each resource of a VM to approximate its probability density. The VMs behavior is described based on a subset of significant resources, that are identified using the Autocorrelation function (ACF) and the Coefficient of Variation (CV) of their time series [2]. Next, we use the Bhattacharyya distance [18] to measure the similarity between two histograms representing the same resource in different VMs: the Bhattacharyya distance is 0 for identical histograms and ∞ for histograms that are completely non-overlapping. The result of this step is a set of distance matrices, one for each considered resource. Then,

the per-resource distance matrices are combined (using the euclidean distance) in a global distance matrix, which is fed into a spectral clustering algorithm [19] to obtain the final clustering solution.

The PCA-based clustering technique [3] represents a completely different approach where the representation of the VM behavior is based on the correlation between the usage time series of different resources of the same VM. Basically, we create a correlation matrix for each VM; then, the matrix is processed to extract the most significant information for clustering purposes. Following a process based on the Principal Component Analysis, we extract the first eigenvector of the correlation matrix (that is, the eigenvector related to the highest eigenvalue). This eigenvector is finally used as the feature vector representing the VM behavior for the subsequent clustering, which is based on the K-means algorithm [20].

For both clustering algorithms, we repeat the clustering several times with multiple random starting points. For each iteration we compare the sum of squares of distances of VMs within the same cluster ($WCSS$) and the sum of square of distances between VMs belonging to different clusters ($BCSS$). We consider as the final clustering solution the iteration minimizing the ratio $WCSS/BCSS$, to avoid the risk of being stuck in a local minimum during the clustering operations.

We point out that the output of the clustering step is also exploited to identify the representative VMs for each identified class: although other solutions are possible, we consider that the VMs closer to the cluster centroids are the most straightforward choice as representatives. It is worth to note that, whenever the clustering operation is repeated, the cluster representatives may change according to the centroids position.

2.4 Need for adaptive clustering

The existing clustering techniques previously described require long resource usage time series to achieve a high clustering accuracy [2], [3]. Indeed, the accuracy tends to decrease with the time series length, requiring a 24-hours monitoring period to correctly cluster the 80% of the VMs in the case studies analyzed in [3]: it is evident that the misclassification of almost one fifth of the VMs is likely to represent a major issue for cloud systems exploiting cluster-based monitoring and management. On the other hand, having a clustering accuracy close to 100% requires longer time series of collected data (in the order of tens of days) [2], [3]. The need to wait so long periods of time to have clustering information causes cluster-based strategies to be slow and scarcely reactive to changes in VMs configuration. This delay is not suitable in the emerging dynamic cloud scenario: to cope with these requirements we need an *adaptive* mechanism that can automatically determine the length of the time series to provide fast and accurate clustering of the VMs. The proposal of such adaptive technique is described in the next section.

3 Adaptive Gray Area-based TEchnique

To provide highly accurate clustering within a time frame that is compatible with cloud dynamic environments, we propose the novel Adaptive Gray Area-based TEchnique (AGATE) that can be easily integrated within the IaaS cloud system described in the previous section. The AGATE proposal, which is run by the *VMs clustering* component on the management node in Figure 1, extends previously proposed clustering techniques [2], [3] to add adaptivity principles to the cluster-based approach. It is worth to note that the AGATE technique does not depend on any specific algorithm for VMs clustering, but can easily integrate different existing solutions. In the rest of this section we present the proposed technique and describe the concept of gray area and the mechanism to adaptively determine its size.

3.1 Technique Overview

The AGATE technique proposes two main novel ideas. The first innovative contribution of the proposed technique is to introduce concepts derived from fuzzy logic in the belonging relationship between VMs and clusters. Such approach is better suited to an adaptive technique than the standard boolean logic used in existing solutions [2], [3]. Specifically, we take into account a *degree of membership* of a VM to each possible cluster. The additional insight provided by the fuzzy logic allows the clustering process to discern between VMs that more likely belongs to a cluster and VMs whose cluster attribution is still uncertain. To this aim, we introduce the concepts of *gray* and *white* areas at the level of clustering data space: a VM in the gray area does not clearly belong to a specific cluster and additional information is required to take a decision; on the other hand, a VM within the white area is definitely assigned to one and only one cluster. From the data center point of view, cluster-based monitoring and management of a VM can start as soon as the VM enters the white area, while every VMs in the gray area must be finely-grained monitored for an additional period of time.

Another qualifying point of our proposal is the use of resource usage time series with different lengths for clustering purposes. We recall that clustering occurs on the basis of a VM behavior model based on the time series collected by the monitor system [2], [3]: longer time series determine higher clustering accuracy as a result of a more accurate description of the VM behavior [3]. We combine this observation with the previously introduced concept of white/gray areas as follows. After a clustering attempt, VMs within the white area are described by a behavior model that is considered sufficiently accurate to identify their membership. On the other hand, for VMs in the gray area, we need to improve the VM behavior description exploiting longer time series collected during an additional monitoring period. This leads to an adaptive selection of the time series length used to describe the VM behavior that depends on the clustering results.

Figure 2 shows how the basic principles discussed above are combined into the main steps of the proposed adaptive technique. In the flow chart, operations are marked with a small gear in the upper right corner of the boxes, while intermediate results are represented with the white icon of a data matrix. The first step (upper left side of the flow chart) represents the beginning of the technique, that starts with a group of VMs with unknown behavior: new running

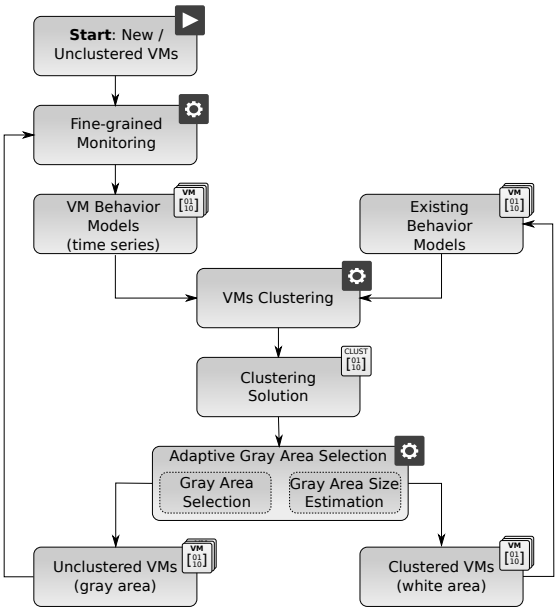


Fig. 2: AGATE technique steps

VMs that just entered the system or VMs that are marked as unclustered because they changed their behavior or caused a server overload. Through fine-grained monitoring, we obtain a set of time series that are used to define the VMs behavior models, which represent the input for the subsequent clustering operation. Then, the clustering solution is processed according to the proposed adaptive approach to separate the VMs into white and gray areas. The two internal components of the *Adaptive Gray Area Selection* step, namely *Gray Area Selection* and *Gray Area Size Estimation*, are described in the rest of this section. Finally, VMs in the white area are assigned to a cluster and no longer monitored with fine granularity: their existing behavior model is stored and re-used for subsequent clustering operations. On the other hand, for VMs in the gray area we collect additional data to determine a behavior model based on longer time series. Then, the process is re-iterated.

3.2 Gray Area Selection

We now detail the *Gray Area Selection* step of the proposed technique, that separates the VMs into gray and white areas. We recall that the clustering step maps each VM within a multi-dimensional space, where clusters and their centroids are located. This mapping is a typical mechanism for the majority of the clustering algorithms, including the k-means and the spectral algorithms (used in [3] and [2], respectively). Assuming that the space supports an euclidean distance operator, we can define for each VM n a vector $D^n = \{d_1^n, \ldots, d_C^n\}$ containing the distances of VM n from the centroids of the C identified clusters. The existing clustering techniques assign each VM to the cluster with the closest centroid. However, if a VM is *nearly equidistant* from two or more centroids, there is a high risk that the clustering algorithm mis-classifies that VM. Hence, the basic idea is to identify a gray area including VMs that are nearly equidistant from at least two centroids, and classify them as unclustered.

The vector D^n of distances from the centroids is used to define the criteria for the gray area selection. Let us now consider a VM n: let c_i be its closest centroid and d_i^n the corresponding distance; d_j^n is the distance between the VM n and each other centroid c_j, with $j \in [1, C], i \neq j$. We express the condition of *near equidistance* by means of a threshold parameter $\epsilon_{i,j}$ (with $i, j \in [1, C], i \neq j$): the VM n is in the gray area if and only if $\exists(i, j)$ such that $1 - \epsilon_{i,j} < \frac{d_i^n}{d_j^n}$. We recall that $\frac{d_i^n}{d_j^n} < 1$, since c_i is the closest centroid to VM n.

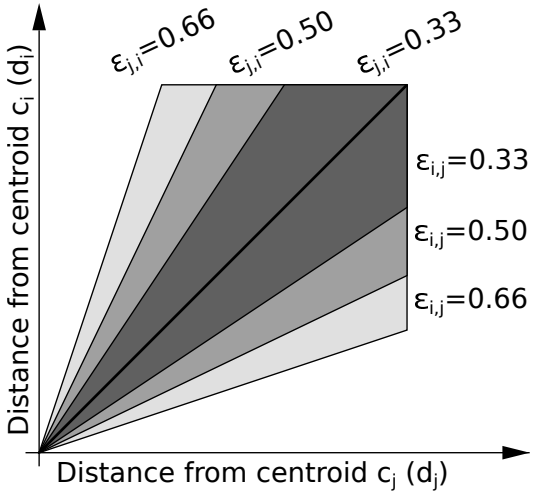


Fig. 3: Gray area representation

To clarify the process of gray area selection, we consider the representation of this area on a bi-dimensional space (Figure 3), where the distances of a VM from centroids c_i and c_j can be visualized on a plane. The bisecting line represents the points where a VM is equidistant from the centroid of the two clusters. Any point below the bisecting line represents a VM assigned to cluster C_i ($d_i < d_j$), while VMs assigned to C_j are mapped in the space above the line. The gray area aims to include the VMs whose classification is more uncertain, that are located in the proximity of the bisecting line. Figure 3 shows different possible extensions of the gray area for different values of the threshold parameters $\epsilon_{j,i}$ and $\epsilon_{i,j}$: we see that the gray area increases in size as the parameters grow from 0 to 1. It is worth to note that we consider two different thresholds: $\epsilon_{j,i}$ is used to include in the gray area the VMs that actually belongs to cluster C_i but are mis-classified and assigned to cluster C_j, while $\epsilon_{i,j}$ is for VMs of cluster C_j that are wrongly assigned to C_i.

The proposed approach considers the distance between a VM and a centroid as a measure of the degree of membership of the VM to the cluster. The alternative would be to directly adopt a *fuzzy clustering* algorithm, such as the *fuzzy c-means* [21], that returns directly a degree of membership of the VM to each cluster. The degree of membership can replace the distance from the centroids in the gray area selection.

3.3 Gray Area Size Estimation

The effectiveness of the Gray Area Selection depends on the correct dynamic estimation of the ϵ parameters that

determine the gray area size: if the adaptive thresholds are too low (small gray area), we may reduce the accuracy of the clustered VMs in the white area to an unacceptable level. On the other hand, too high values tend to overestimate the number of VMs that are in the gray area, thus reducing the benefits of the adaptive clustering. The estimation of the gray area size is the second fundamental component of the automatic gray area selection step of the technique described in Figure 2.

A preliminary task to determine the value of the ϵ parameters is the analysis of the data distribution of the distances between VMs and centroids of the identified clusters. To this aim, we consider the actual belonging of VMs to clusters (thus removing clustering errors). It is worth to note that this simplification does not hinder the applicability of our approach, because we can extend our results to the real case (that is, with clustering errors), as detailed in Appendix A.

We consider the VMs belonging to each identified cluster, and fit the data about their distances from each cluster centroid to the closest theoretical distribution using the Anderson-Darling (AD) Goodness of Fit test. We found that the centroid distances can be approximated by a normal distribution, that is the theoretical distribution with the lowest AD-value. Figure 4 presents the Probability Density Function (PDF) as an example of the similarity between the empirical data on the VMs distance from a cluster centroid and the theoretical distribution obtained as a result of the fitting process.

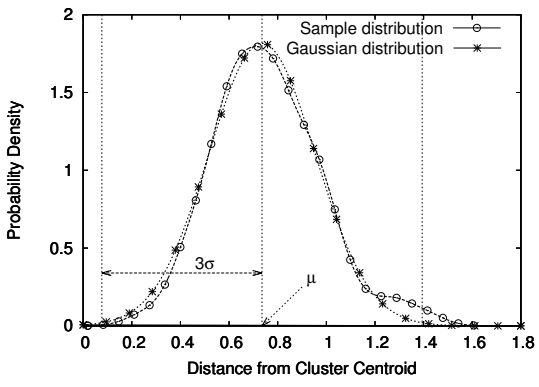


Fig. 4: PDF of distance from cluster centroid

In the space of the centroid distances used to define the gray area, the generic VM n is a point with coordinates $(d_1^n, \dots d_C^n)$. For the definition of the gray area, we consider a two dimensional view of this space, where each VM can be represented as a point (d_j^n, d_i^n) as in Figure 3. Let us now consider the VMs belonging to cluster C_i: since their distances from centroids c_i and c_j may be approximated by normal distributions with means μ_i, μ_j, and standard deviations σ_i, σ_j, the scatterplot of the points representing the VMs assumes the shape of an ellipse with the following properties. The ellipse, shown in Figure 5, is centered in the point of coordinates (μ_j, μ_i) with $\mu_i < \mu_j$ for VMs belonging to cluster C_i, and has semi-axes $k\sigma_j$ and $k\sigma_i$, where k is a multiplier. The choice of a value for k determines the probability to include all the VMs of cluster C_i in the ellipse; this probability can be expressed as a function of k as $\text{erf}\left(\frac{k}{\sqrt{2}}\right)$, where $\text{erf}(\cdot)$ is the Gauss error function. Figure 4 shows the area considered if we truncate the Gaussian curve to the interval $[\mu - k\sigma, \mu + k\sigma]$: in our experiments we use the well known rule of thumb for Gaussian distributions and we impose $k = 3$. A generic VM will fall within the ellipse with 99.7% probability, thus reducing the error probability to 0.3%.

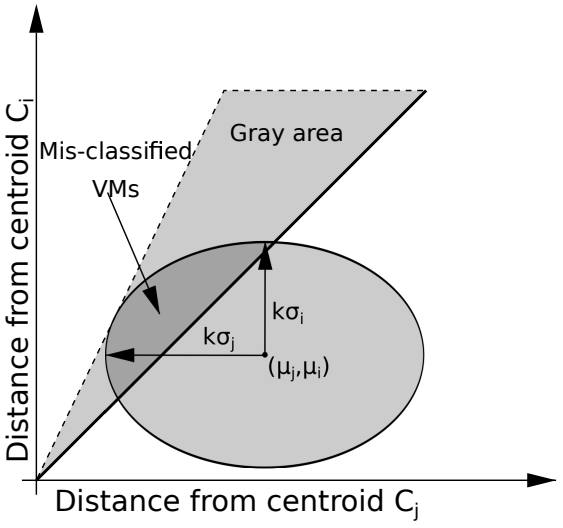


Fig. 5: Gray area definition

The elliptic area containing the VMs belonging to cluster C_i can thus be defined according to the following equation:

$$\left(\frac{d_j^n - \mu_j}{k\sigma_j}\right)^2 + \left(\frac{d_i^n - \mu_i}{k\sigma_i}\right)^2 \leq 1$$

In Figure 5, the part of the ellipse above the bisecting line contains the VMs that would be mis-classified. To address this problem, this area must be included within the gray area. From this observation, we can derive the value of $\epsilon_{j,i}$ as follows.

To find the boundary of the gray area, we need to identify the lines that are tangent to the ellipse and pass through the axis origin. Their angular coefficient can be obtained through the following formula.

$$a_{1,2} = \frac{-\mu_j\mu_i \pm \sqrt{\mu_j^2\mu_i^2 - (k\sigma_j^2 - \mu_j^2)(k\sigma_i^2 - \mu_i^2)}}{k\sigma_j^2 - \mu_j^2}$$

We are interested in the higher value of the two angular coefficients (in our case a_2, because $k\sigma_j < \mu_j$ and $k\sigma_i < \mu_i$), and we can obtain $\epsilon_{j,i}$ as follows:

$$\epsilon_{j,i} = 1 - \frac{k\sigma_j^2 - \mu_j^2}{-\mu_i\mu_j - \sqrt{\mu_j^2\mu_i^2 - (k\sigma_j^2 - \mu_j^2)(k\sigma_i^2 - \mu_i^2)}}$$

The same process is applied to every other couple of clusters C_i and C_j, to determine the thresholds for the gray area. We recall that the parameters of the ellipse (μ_i, μ_j, σ_i, σ_j) are derived from the set of VMs that actually belong to cluster C_i, not including clustering errors due to mis-classification of VMs. When the knowledge of the ground truth is not available, as in a real application of the proposed technique, we can still estimate the ellipse parameters using the process detailed in Appendix A.

4 CLUSTERING SHORT TIME SERIES

The proposal of an adaptive approach to VMs clustering descends from the need to identify VMs similarities in short periods of time to support a reactive resource management within the cloud system. A basic requirement for a reactive mechanism is to apply clustering techniques on short resource usage time series (length of few hours), differently from previous studies where VMs were monitored for long periods (one or more days). Considering short time series poses new challenges for the accuracy of clustering techniques: specifically, the presence of daily patterns in the cloud workload may significantly hinder the capability of previously proposed techniques [2], [3] to cluster VMs whose behavior is monitored at different time of the day.

To understand this problem, we consider the example shown in Figure 6a, representing the CPU utilization for a Web server during a 24-hours period. In the graph we clearly see a daily pattern, with a peak of resource utilization during the daily hours and a low utilization in the night. We recall that, in a dynamic cloud environment, newly acquired VMs can enter the system at any time: as explained in Section 2.2, in a system adopting a VMs clustering technique for monitoring and management purposes, new VMs are monitored with fine-granularity and the collected time series are given as input for a new clustering step along with the information previously collected about the other VMs. The asynchronous monitoring is not a problem when the time series length is a multiple of the day, because whole daily patterns are taken into account, but the situation changes if we consider shorter time series. Let us assume a monitoring period of 12 hours in a scenario where new VMs enter the system at time t = 20:00 and t = 8:00. Figure 6b shows the CPU time series with length of 12 hours for two VMs: VM2 (entered at t = 20:00) is monitored during the night and VM1 (entered at t = 8:00) is monitored during the day. It is important to note that the differences in the behavior of the VMs is not related to a different software component (both VMs are Web servers), but to the different monitoring periods.

Depending on the approach adopted by the clustering technique to model the VM behavior (Bhattacharyya-based [2] or PCA-based [3]), monitoring VMs in different time periods may have side effects.

The Bhattacharyya-based clustering technique [2] models the similarity of two histograms using the Bhattacharyya coefficient, that is the basis for the Bhattacharyya distance: the coefficient is 1 for identical histograms and 0 for histograms that are completely non-overlapping. Let us consider again the 12-hours CPU time series of VM1 and VM2: the resulting histograms are shown in Figure 6c. We observe that the two behavior descriptions are completely different, with a mode in the probability distribution close to 60% for VM1 and to 20% for VM2, and a limited overlap between the histograms. The resulting similarity of the two histograms is rather low: in our example, the Bhattacharyya coefficient is 0.0334, that is a very low value typically identifying resource histograms of VMs belonging to different clusters.

On the other hand, the PCA-based clustering technique [3] is based on the correlation between the usage time series of different resources on the same VM. It is worth to note that, differently from histogram-based distance, the correlation among the resource usage remains quite stable over different periods of the day even in presence of workload fluctuation. To verify this claim, we consider the previous example of two Web servers monitored in different periods of the day and we compute the correlation between the resource usage of the VMs. Specifically, we consider a set of six metrics: CPU utilization, input and output packet rates, number of alive processes, frequency of system calls and memory utilization.

TABLE 1: Correlation for short time series

Resources	VM1 (Day)	VM2 (Night)	Correlation Δ
CPU/InputPkts	0.77	0.75	0.02
CPU/OutputPkts	0.80	0.79	0.01
CPU/AlivePr	0.18	0.21	0.03
SysCall/Memory	0.17	0.14	0.03

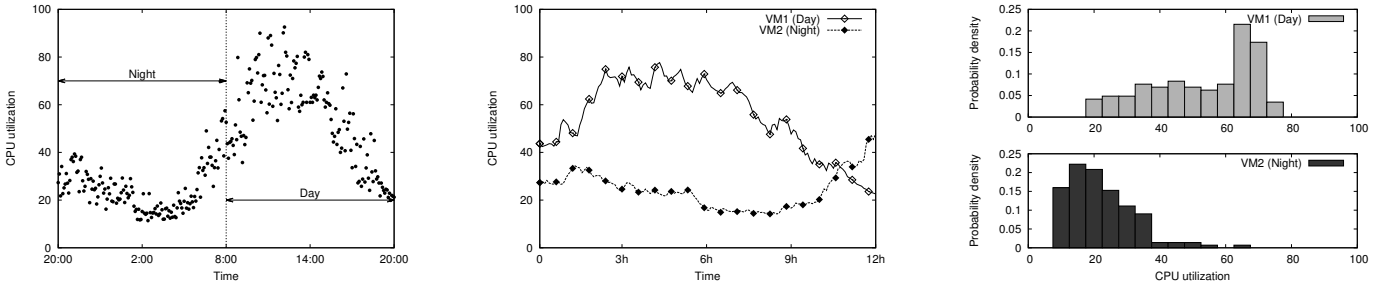
Table 1 provides the correlation values between the metric time series in the two considered VMs. The first two rows of the table provide examples of highly correlated time series, such as the CPU and the input/output packet rates (that in a Web server are highly correlated), while the latter rows present examples of uncorrelated time series. The most interesting result is provided by the last column of the table: for all the time series, the difference between the correlation of VM1 and VM2 is quite low. As the correlation matrices are quite similar for VMs observed during nightly and daily periods, we can conclude that a behavior model based on correlation is scarcely sensitive to the period of the day during which the model is built. Hence, a clustering technique exploiting this behavior model may provide a better alternative with respect to histogram-based techniques when dealing with time series of length below 24 hours.

5 EXPERIMENTAL RESULTS

In this section we evaluate the applicability and the effectiveness of the proposed technique by applying it to three different case studies. After describing the case studies, we carry out multiple experiments showing that: (1) for short time series the PCA-based clustering technique outperforms the Bhattacharyya-based approach; (2) the dynamic determination of the parameters $\epsilon_{i,j}$ is effective to identify the potentially mis-classified VMs; (3) the application of the AGATE proposal allows the system to rapidly identify a set of correctly clustered VMs that grows at every iteration of the technique; (4) the amount of data collected by the monitoring system can be significantly reduced with respect to existing solutions.

Throughout the experimental evaluation, we measure the clustering *accuracy*, that we define as follows. Let $S = \{s^n, n \in [1, N]\}$ be the clustering solution that assigns each VM n to a cluster (with N being the number of VMs). We define accuracy by comparing the clustering solution S with the vector S^* representing the correct clustering based on the knowledge of the actual software components running on the VMs. Accuracy is thus defined as:

$$accuracy = \frac{|\{s^n : s^n = s^{n*}, \forall n \in [1, N]\}|}{N}$$



(a) Daily pattern (b) Sample VMs (c) Probability histogram

Fig. 6: Bhattacharyya distance problem for short time series

Basically, the accuracy represents the fraction of the VMs which are correctly identified by the clustering algorithm.

5.1 Case studies

The first case study, namely *EC2-Amazon*, is based on a data set coming from a virtualized data center hosting a Web-based e-commerce application with a synthetic workload. The application, based on the TPC-W benchmark, is deployed over the Amazon Elastic Computing infrastructure. The application uses a Java-based application server, a DBMS and a set of emulated browser (EBs), issuing both HTTP and HTTPS requests. The benchmark is hosted on a set of 36 VMs (we use the *M3 xlarge* VM instances provided by Amazon EC2), with 12 VMs dedicated to Web servers, 12 to DBMS and 12 VMs running the EBs. We consider two workloads for this scenario: a *Constant* workload using 400 EBs on each VM, and a *Step* workload where the number of EBs changes from 200 to 600. In both workloads, each EB issues requests at a constant rate. Data collection is based on a prototype cloud monitoring framework described in [22]. In our experiments every type of VM (EB, Web servers, and DBMS servers) is monitored and considered in the clustering process. Figure 7 represents the two workloads used in this scenario: the dashed line is the Constant workload, where the number of EBs does not change over time, while the Step workload is characterized by a varying number of EBs, with a peak period and an off-peak period. For this case study, we assume that the monitoring period, which determines the length of the time series to consider for VMs clustering, is equal to T. We also assume that 50% of the VMs enter the system at time $t = 0$ (monitored until $t = T$), while the other 50% of the VMs enter at time $t = T$ (monitored until $t = 2T$).

The second case study, namely *Interactive-Web*, is based on a dataset coming from a real private cloud data center supporting a customer Web-based application. Specifically, the data center hosts Customer Relationship Manager (CRM), that is deployed on 150 VMs according to a multi-tier architecture. The VMs are divided between Web servers and back-end servers (that are DBMS), with 100 VMs hosting a Web Server and 50 VMs hosting a DBMS. Load sharing strategies are used so that the computation is evenly distributed over the VMs hosting the same software component. As the data refers to a real data center, the workload intensity changes over time and exhibits daily patters as in Figure 6a.

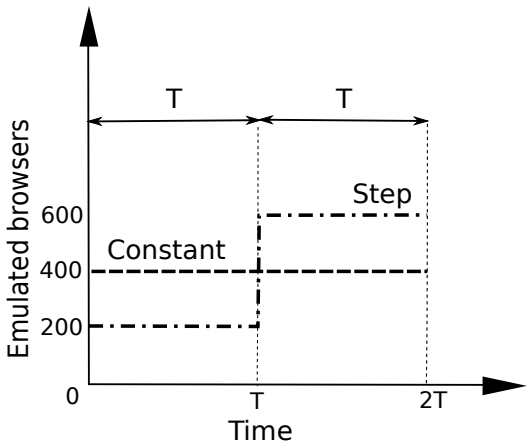


Fig. 7: Constant and Step Workloads (EC2-Amazon)

The third case study, namely *Mixed-Web*, considers another real scenario where a Web-based application (with Web servers and DBMS) is joined with a batch processing facility based on the map-reduce paradigm. The case study refers to 200 VMs, divided into 80 Web servers, 50 back-end servers and 70 map-reduce nodes. As for the other case study, the Web application exhibit daily patters, while the map-reduce nodes present a load that is evenly distributed throughout the 24 hours.

The goal of the VM clustering is to correctly separate the VMs running the different software components for each considered customer application. For example, in the Mixed-Web case study the goal is to identify three different clusters: Web servers, DBMS and batch components. It is worth to note that the target list of clusters does not change for different workloads, but only depends on the software components run by the VMs. For the map-reduce application, special purpose nodes are present (e.g., hadoop masters and HDFS name nodes). Although these nodes represent one-of-a-kind objects with a behavior that differ from the typical computing node, we consider also these nodes in the clustering process, as it would happen in a real world application of the proposed technique. Our finding is that these nodes are classified as members of the map-reduce cluster. However, their different behavior places them far from the cluster centroid and these nodes remains in the gray area for the whole duration of the experiments.

For each case study, we collect data about the resource

usage of every VM for different periods of time, ranging from 1 to 48 hours with a sampling frequency of 1 minute. For each VM we consider 10 metrics describing the usage of different resources related to CPU, memory, disk, and network. The complete list of the metrics is provided in Table 2 along with a short description.

TABLE 2: Virtual machine resources

Metric		Description
X_1	SysCallRate	Rate of system calls [req/sec]
X_2	CPUSys	System CPU utilization [%]
X_3	CPUUser	CPU utilization (user mode) [%]
X_4	CtxSwitch	Rate of context switches [CS/s]
X_5	Memory	Physical memory utilization [%]
X_6	BlockOut	Rate of blocks written to storage [Blk/s]
X_7	PgOutRate	Rate of memory pages swap-out [pages/sec]
X_8	OutPktRate	Rate of network outgoing packets [pkts/sec]
X_9	InPktRate	Rate of network incoming packets [pkts/sec]
X_{10}	AliveProc	Number of alive processes

5.2 Comparing clustering for short time series

The first experiment aims to evaluate the accuracy of different clustering techniques applied to short resource usage time series: we consider the Bhattacharyya- and PCA-based techniques that emerged as best performing clustering solutions in previous studies [2], [3]. For this evaluation we exploit two case studies with opposite characteristics. The first case study is the EC2-Amazon scenario with the two synthetic workloads previously described (*Constant* and *Step*). For each workload we perform multiple experiments: in each experiment the period T assumes a different value between 1 and 48 hours. Then, we consider the real Interactive-Web scenario, where the workload is subject to the typical daily patterns characterizing Web traffic represented in Figure 6a. We carry out multiple experiments also for this scenario: we assume that the entrance of the VMs in the system is uniformly distributed in a time frame of 24 hours, and each VM is monitored starting from its entrance for a period of time that ranges from 1 and 48 hours depending on the experiment. This means that, for short time series, VMs belonging to the same cluster are monitored in different parts of the day, as described in Section 4.

Figure 8 shows the accuracy achieved by the Bhattacharyya- and PCA-based clustering techniques for the considered scenarios and workloads. We first observe that the accuracy for the EC2-Amazon scenario is generally higher with respect to the Interactive-Web case: this result is expected because EC2-Amazon exploits a synthetic workload, where the regular access patterns tend to increase the effectiveness of the clustering algorithms. We also observe that the accuracy generally tends to worsen as the time series length decreases, because the identification of similarities between VMs behavior becomes more challenging when based on a limited number of observations (short monitoring periods); these results are consistent with [2], [3]. However, the magnitude of the accuracy reduction is significantly different depending on scenarios and clustering techniques. We can clearly see that the Bhattacharyya- and PCA-based approaches have similar performance for the Constant EC2-Amazon case study, with an accuracy that

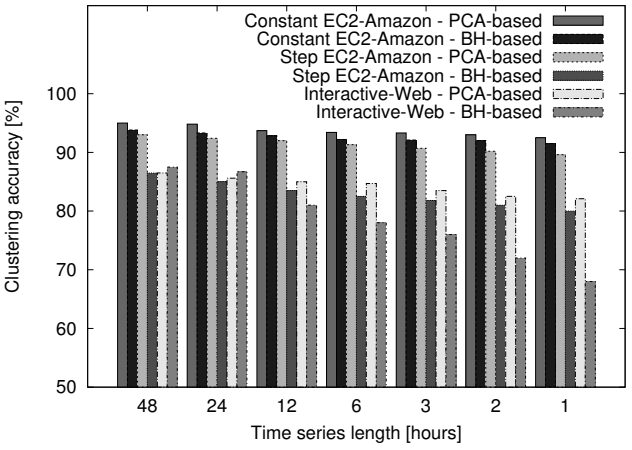


Fig. 8: Clustering accuracy for short time series

remains quite stable for varying time series length. This result is expected since the workload, and consequently the VM resources usage, is constant over time. On the other hand, the clustering techniques perform differently in the other two scenarios. In the Step EC2-Amazon case, the Bhattacharyya-based technique achieves worse accuracy than the PCA-based for every time series length, decreasing rapidly for short monitoring periods. A similar effect can be observed for the Interactive-Web scenario, but with a significant difference: the accuracy of the clustering techniques is similar for longer time series (48 and 24 hours). In presence of daily patterns, indeed, the Bhattacharyya-based clustering is effective when the time series length is a multiple of 24 hours because entire daily patterns are taken into account; on the other hand, for time series shorter than 24 hours the accuracy rapidly decreases, dropping to 68% for 1 hour time series. In the same scenario, the accuracy of the PCA-based clustering is more stable and remains over 82% even for time series of 1 hour: the best performance is due to the VM behavior model based on the correlation between resource usages, as discussed in Section 4. These results confirm that the PCA-based technique is a better option for clustering short time series in a dynamic cloud environment; for this reason, in the rest of the experimental evaluation we consider the PCA-based approach as the clustering technique used in the AGATE proposal.

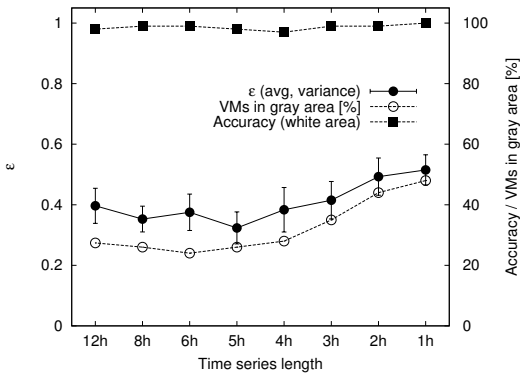
5.3 Fuzzy gray area analysis

This experiment aims to evaluate the impact of the fuzzy selection based on the gray area on the VMs clustering output. To this aim, we apply just one iteration of the AGATE technique to the entire set of VMs, considering behavior models described by time series of different lengths. We carry out our evaluation on the Interactive-Web and Mixed-Web scenarios, that are characterized by the presence of two and three clusters of VMs, respectively; analyses on the EC2-Amazon scenario confirm our main findings and are omitted for space reasons. To provide an insight on the proposed technique, we evaluate the value of the $\epsilon_{i,j}$ parameters, the fraction of VMs in the gray area and the clustering accuracy of the VMs in the white area, as shown in Figure 9.

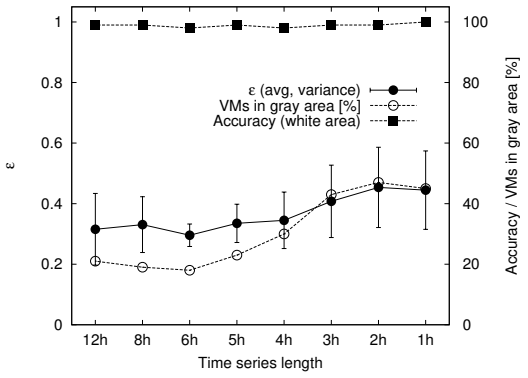
Figure 9a refers to the Interactive-Web scenario. The first significant result is about the clustering accuracy computed on the VMs in the white area (line with black squares in the graph), which is very close to 100% for every time series length. The high accuracy means that the selection of the gray area size is effective in identifying the potentially mis-classified VMs, while leaving in the white area the correctly clustered VMs. A second important observation is about the behavior of the $\epsilon_{i,j}$ parameters and their effect on the number of VMs in the gray area. The line with black circles in the graph shows the average values of the $\epsilon_{i,j}$ parameters, while the error bars represent their variance. We observe that, as the time series length increases, the average value of $\epsilon_{i,j}$ tends to decrease. The oscillations in the average value of $\epsilon_{i,j}$ are explained by the inherent variability of the workload that shows significant changes over time: the workload variability determines small fluctuations in the correlation matrix used for the clustering, that causes the small oscillations observed for the black circles curve. However, it is worth to note that the changes in the average $\epsilon_{i,j}$ values are smaller than the variance of the $\epsilon_{i,j}$ parameters for the different clusters, thus confirming the robustness of the proposed approach. Similarly to the $\epsilon_{i,j}$ parameters values, the percentage of VMs in the gray area (white circles) decreases for longer time series. These results confirm that the AGATE technique is effective in dynamically computing the $\epsilon_{i,j}$ parameters to adjust the gray area size without affecting the clustering accuracy of the VMs in the white area.

Figure 9b mainly confirms our findings for the Mixed-Web scenario. An interesting difference is about the $\epsilon_{i,j}$ parameters, that show a slightly lower average value but a higher variance than in the previous scenario. This is motivated by the group of VMs that process map-reduce jobs: as these VMs show a very different behavior with respect to VMs belonging to other clusters, their cluster has a more dense scatterplot (with the exception of a few outliers that are the map-reduce manager nodes). The $\epsilon_{i,j}$ parameters for the map-reduce clusters are characterized by lower values that reduce the average and increase the variance of the global set of $\epsilon_{i,j}$ parameters. We also note that the percentage of VMs in the gray area shows a clear reduction as the time series length grows. This effect can be explained if we consider that clusters tend to shrink in size, with VMs moving closer to the cluster centroid as the time length increases. To better evaluate this effect, let's consider the example in Figure 10.

Figure 10 refers to the Interactive-Web workload: the graphs show the scatter plots of the distances of all the Web servers VMs from the two cluster centroids. The three graphs are referred to time series of 6, 12 and 72 hours respectively and aims to provide an insight on the evolution of the clustering outcome as a function of the time series length. The Web servers VMs are represented as dots, with white dots representing the correctly identified VMS, and black dots the WMs potentially mis-classified that should be enclosed in the gray area. We recall that the parameters of the ellipse that should contain all the dots are approximated by the analysis of the distances between the VMs assigned to cluster C_1 and the centroids using the formulas of Section 3.3 and Appendix A; the tangent line to the ellipse



(a) Interactive-Web scenario



(b) Mixed-Web scenario

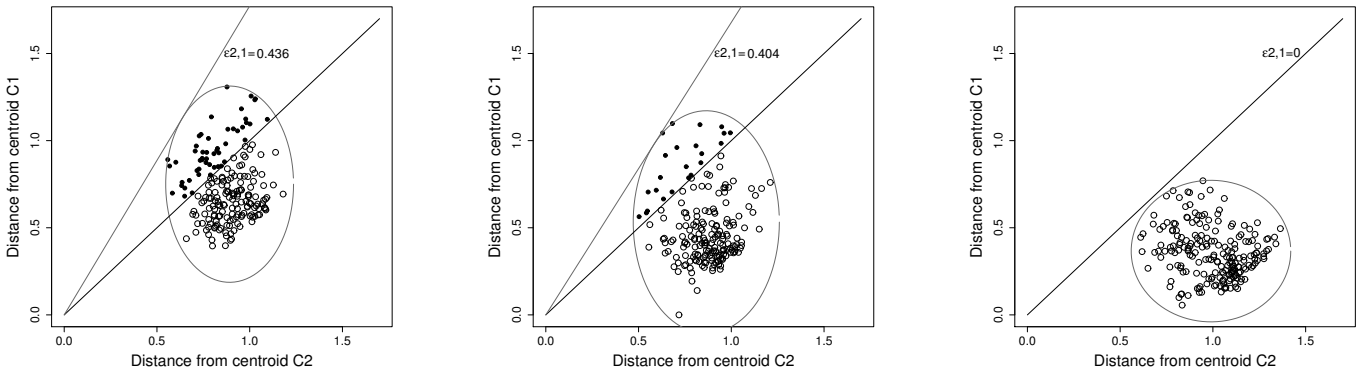
Fig. 9: Gray area analysis

is used to determine the value of $\epsilon_{2,1}$. From Figures 10a and 10b we observe that the estimated ellipse successfully encloses all the VMs of the cluster (including the black dots): hence, the identified gray area includes all the potentially mis-classified VMs for this cluster. This result explains the clustering accuracy close to 100% for VMs in the white area. We also note that the distance of the Web servers VMs from their centroid (C_1) decreases as the time series length grows (ellipse moving towards the lower side of the figures), with the subsequent reduction of $\epsilon_{2,1}$ and of the number of VMs in the gray area. Finally, for a 72 hours time series, every point in the scatter plot is on the right side of the bisecting line, and no gray area is required for the Web servers cluster ($\epsilon_{2,1} = 0$).

5.4 AGATE technique evaluation

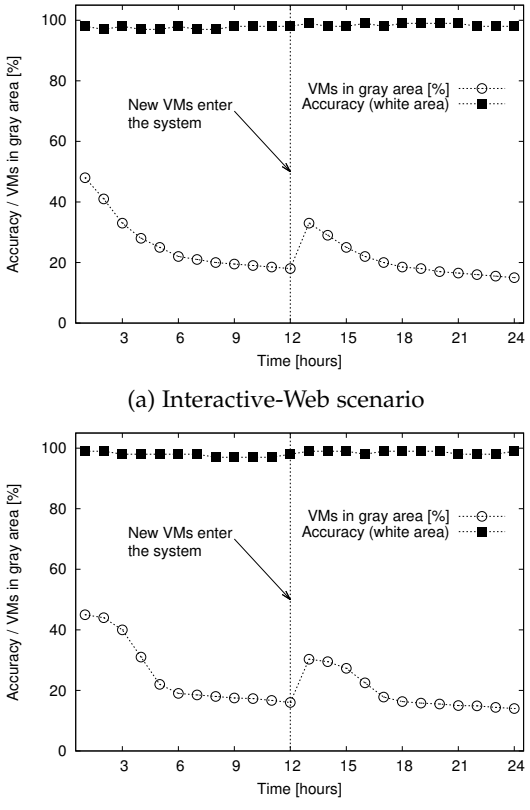
Having demonstrated the effectiveness of a single iteration of the proposed technique, we now focus on the complete process for adaptive VMs clustering considering multiple consecutive iterations. To this aim, we focus on the Interactive-Web and Mixed-Web scenarios considering a 24 hours-long experiment. For each scenario, we assume that 50% of the VMs enter the system at time $t = 0$, while the remaining 50% of the VMs at time $t = 12$. We execute an iteration of the AGATE technique every hour.

Figures 11a and 11b show the results of the experiment for the Interactive-Web and Mixed-Web scenarios, respectively. Each graph shows the evolution over time of the gray area size (the white circles represent the percentage of VMs



(a) Time series = 6h (b) Time series = 12h (c) Time series = 72h

Fig. 10: Scatter plot of distances from cluster centroids



(a) Interactive-Web scenario

(b) Mixed-Web scenario

Fig. 11: Application of AGATE technique

belonging to the gray area), together with the clustering accuracy of the VMs in the white area (black squares).

From Figure 11a we have a first confirmation that, starting from the first iterations, our solution is able to correctly cluster a significant set of VMs: at the end of the second and third iterations, we have in the white area the 59% and the 67% of the VMs, respectively; moreover, we note that the clustering accuracy for the VMs in the white area is close to 100%. This means that a cluster-based management can be applied to the VMs in the white area, which represent a large percentage of the total VMs, without having to wait longer monitoring times or cope with misclassification errors as in existing clustering solutions. The same effect can be observed when new VMs enter the system at time $t = 12h$. Another interesting result is that the size of the gray area monotonically decreases in the periods [1-12h] and [13-24h], thus leading to a number of correctly clustered VMs that grows at each step: the percentage of VMs in the gray area is reduced from 48% to 18% in the first 12 hours and from 33% to 15% in the second period. The discontinuity in the graphs that occurs in the interval [12-13h] is due to the new VMs entering the system.

The Mixed-Web scenario is analyzed in Figure 11b and basically confirms the results observed for the previous scenario. It is worth to note that the number of VMs in the gray area tends to be lower with respect to Figure 11a: this can be explained considering that the VMs belonging to the map-reduce cluster are more easily identified and less likely to fall within the gray area compared to Web server and DBMS clusters, as already pointed out in Section 5.3.

5.5 Monitoring overhead reduction

We now evaluate the reduction of the monitoring overhead in an IaaS cloud system achieved by the adoption of the proposed AGATE technique. For this experiment we apply a cluster-based monitoring to the VMs of the Interactive-Web and Mixed-Web scenarios for a period of time of 12 hours, with the aim to measure the reduction of the data collected by the monitoring system when the adaptive technique is applied instead of the existing PCA-based clustering solution [3]. For a fair comparison, we made the following assumption. In the case of PCA-based solution, we apply the VMs clustering to different time series lengths (ranging from 1 to 12 hours), while for the AGATE technique we carry out the VMs clustering and the gray area selection once every hour. In both cases, we assume that all the VMs enter the system at the beginning of the experiment. Tables 3 and 4 show the results of the experiment for the Interactive-Web and Mixed-Web scenarios, respectively.

For both tables, second and third columns refer to the existing PCA-based solution. For each time series length (first column of the table), the second column shows the amount of data collected by the monitoring system: in this

case, every VM is monitored with fine granularity (sampling frequency equal to 1 minute) for all the monitoring period (we indicate with $\overline{K}$ the amount of data collected to monitor with fine granularity a single VM for 1 hour). The third column indicates the percentage of misplaced VMs for the corresponding clustering step.

The last three columns of the tables refer to the adaptive AGATE technique, showing for each interaction the amount of collected data, the percentage of VMs in the gray area and the percentage of misplaced VMs within the white area. For the AGATE proposal, we recall that the collected data depend on the results of the gray area selection at the previous iteration. Let us consider, as an example, the first two rows of Table 3. The first iteration of the adaptive technique occurs at the end of 1 hour of data collection (first row of the table), during which all the 150 VMs are monitored with fine granularity ($150 \times \overline{K}$ collected data); as a result of the gray area selection, we have 47.8% of VMs (72 VMs) in the gray area. During the second hour of monitoring (second row of the table), we collect data with fine granularity only for the VMs of the gray area ($72 \times \overline{K}$ data), plus 3 representative VMs for each cluster ($6 \times \overline{K}$, having 2 clusters), for a total amount of data collected over the two hours equal to $228 \times \overline{K}$. In a similar way we compute the data collected in the next iterations. We omit the contribution of coarse-grained samples on non representative cluster VMs because the corresponding amount of data collected is not significant, being typically at least one order of magnitude less with respect to fine-grained sampling.

TABLE 3: Data reduction in Interactive-Web scenario

Time [h]	Existing Solution		AGATE		
	Data ($\times\overline{K}$)	Misplaced VMs [%]	Data ($\times\overline{K}$)	Gray Area VMs [%]	Misplaced VMs [%]
1	150	17.9	150	47.8	1.8
2	300	17.5	228	41.1	2.8
4	600	16.1	296	27.8	2.3
6	900	15.4	344	22.3	1.9
8	1200	15.2	383	19.9	1.7
10	1500	15.1	419	18.7	2.4
12	1800	14.8	453	17.8	2.1

TABLE 4: Data reduction in Mixed-Web scenario

Time [h]	Existing Solution		AGATE		
	Data ($\times\overline{K}$)	Misplaced VMs [%]	Data ($\times\overline{K}$)	Gray Area VMs [%]	Misplaced VMs [%]
1	200	14.6	200	44.8	1.2
2	400	14.1	299	43.4	1.5
4	800	13.3	395	30.9	1.8
6	1200	12.6	466	19.6	1.6
8	1600	11.9	514	18.2	2.3
10	2000	11.3	559	17.2	2.1
12	2400	10.8	603	16.1	1.8

For both scenarios we observe that the amount of VMs assigned to the gray area by the adaptive proposal is always higher than the percentage of misplaced VMs for the existing solution for the same period of time. However, it's important to emphasize that with the existing solution there is no way to know which VMs are correctly classified by the clustering, hence the management of the data center has to cope with a not negligible percentage of misclassified VMs (after 12 hours, 14.8% and 10.8% for the Interactive-Web and Mixed-Web scenario, respectively). On the other hand, as soon as the first iteration of the AGATE technique is completed, a large percentage of VMs is assigned to the white area. These VMs are classified with high accuracy (the percentage of misplaced VMs remains below 2.8% and 2.3% in the Interactive-Web and Mixed-Web scenario, respectively) and the cluster-based management can be applied to them without having to wait any longer. Moreover, the fine-grained monitoring is limited to the VMs remaining in the gray area, thus significantly reducing the amount of data collected with respect to the existing solution: at the end of the second hour of monitoring we have a reduction of collected data equal to 24% and to 25% for Interactive-Web and Mixed-Web scenario, respectively; at the end of the twelfth hour, the amount of data collected is reduced by almost 75% in both cases.

6 Related Work

Monitoring and management of large cloud data centers are critical tasks that have been receiving a lot of attention from academic and commercial communities in the last few years [23], [24].

Current approaches typically address monitoring scalability issues by aggregation and filtering the collected data before sending them to the management process in order to reduce their volume. Most of the proposed solutions require a specialized software layer, such as a library, or rely on agents, which are responsible for data collection, filtering and aggregation [25]–[29]. Different aggregation strategies have been proposed: extraction of high-level performance metrics by mean of machine learning algorithms [26]; extraction of predicted parameters by combining metrics from different layers (hardware, OS, application and user) and by applying Kalman filters [25]; linear combination of OS-layer metrics [27]; extraction of high-level statistics from OS and application layers [28], [29]. Several commercial and open source platforms have also been proposed to monitor cloud systems. Among the most popular platforms, some examples deserve a mention. Amazon CloudWatch [30] is a Web service for monitoring Amazon Web Services (AWS) cloud resources; LogicMonitor [31] is a commercial tool characterized by low complexity and ease of use to monitor physical and virtual infrastructures; DARGOS is a distributed monitoring architecture using a push/pull approach to distribute information [32]. However, all these solutions share a common limitation, that is considering each monitored object (being it a VM or a host) as independent from the others: such approach fails to take advantage from the similarities of objects which are sharing a common behavior.

Multiple proposals have been proposed in literature for the management of cloud systems, starting from the first studies about autonomic concepts applied to cloud computing [33]. Among the notable examples, Bobtail [34] is a library allowing VMs of a same application to identify placement problems that may cause long latencies in inter-VMs communication. The solution in [35] provides a mechanism to allocate VMs when multiple frameworks (e.g., Hadoop and MPI) compete for the same resources of a data-center.

However, these techniques require the intervention of the cloud consumers that must install specific software and libraries on their VMs, while in this paper we adopt the point of view of a IaaS cloud provider having no access to customer VMs. An approach based on distributed and dynamic VMs allocation is proposed in [36]: this local-based migration solution could be integrated with our AGATE technique. A study on how VMs clustering can be used to improve the scalability of data center management has been recently published by the authors [4].

The identification of similarities between VMs in cloud systems represents the core of our proposal. Relevant works are worth to be cited in this field. A method for VMs clustering in cloud systems is presented in [37], but the similarity detection is limited to storage resources to apply storage consolidation strategies. Another study [38] investigates similarities of VMs static images used in public cloud environments to provide insights for de-duplication and image-level cache management. While these studies apply clustering to a very limited set of resources for specific purposes, our approach considers several resources to model the general VMs behavior and leverage similarities that could improve the overall scalability of cloud monitoring.

Recent studies of the authors [2], [3] propose clustering approaches that group together VMs with similar behavior in terms of resource usage: the PCA-based technique [3] exploits resource correlation and Principal Component Analysis [39] to model VMs behavior, and a k-means algorithm for clustering; the Bhattacharyya-based technique [2] relies on a histogram-based representation to model VMs behavior, Bhattacharyya distance [18] to measure VMs similarity and a spectral algorithm for clustering. These techniques are applied to time series longer than 24 hours, and even in these conditions they share the common limit of a non-negligible amount of misclassified VMs that hinders the applicability and the effectiveness of these solutions for large cloud systems. The adaptive AGATE technique addresses this issue by separating the VMs clearly belonging to one cluster (in the white area) from the VMs that are undecided (in the gray area) and need further monitoring to be carried out; the clustering accuracy close to 100% for the VMs in the white area enables effective management for these VMs and scalable monitoring for the cloud system. As our proposal provides the first clustering results after just a few hours of observation, it enables the application of cluster-based monitoring and management to a much larger set of scenarios with respect to previous solutions.

A preliminary version of the gray area-based technique was proposed in [6], but this previous work has many limitations. First, the size of the gray area was determined on the basis of a fixed value instead of being adaptively computed according to the results of clustering, resulting in a significant increase in the amount of collected data and in the time needed to take decisions on VMs behavior. Second, the technique was applied to a single case study, which limited the evaluation of its applicability as it could potentially omit unexpected behaviors with different kinds of workloads. Third, metric time series shorter than 24 hours were not considered for clustering, thus preventing us to discover the performance issue of the histogram-based clustering applied to short time series, which is analyzed and discussed in this paper.

7 Conclusions

The success of cloud computing as a key enabling technology for the emerging digital society has determined new scalability issues for IaaS cloud infrastructures in terms of monitoring and management. We focus on techniques aiming to address scalability issues by clustering together VMs with similar resource usage behaviors. Existing solutions for VMs clustering require long resource monitoring periods to provide accurate classification, and are not viable for modern dynamic cloud systems. This observation motivates our proposal of a novel approach that, exploiting principles of fuzzy logic, adaptively selects the length of the resource time series used for clustering each VM. Such approach allows the cloud system to rapidly determine the VMs behavior, thus coping with the need for fast deploying and disposing of new VMs in a IaaS cloud datacenter.

The proposed AGATE technique, based on the introduction of a *gray area* (a set of VMs not yet assigned to any cluster), can automatically separate VMs that are assigned to clusters from VMs with a still undetermined behavior. Another qualifying point of our study is the effort to identify clustering techniques able to cope with short resource time series (e.g., few hours) in presence of daily workload patterns. We extensively test our proposal using both real case studies and synthetic benchmarks. Our results demonstrate that our solution is fully adaptive with respect to the workload characteristics and, without need of parameter tuning, can correctly identify most of the VMs after just few hours with an accuracy of 97% or higher. Furthermore, we evaluate the potential benefit of our proposal in terms of monitoring scalability with respect to existing approaches: in our experiments the data reduction reaches 25% after just two hours of monitoring and grows to 75% in twelve hours.

References

[1] J. Whitney and P. Delforge, "Data center efficiency assessmenty – scaling up energy efficiency across the data center industry: Evaluating key drivers and barriers," NRDC, Anthesis, Tech. Rep., 2014, – http://www.nrdc.org/energy/files/data-center-efficiency-assessment-IP.pdf.

[2] C. Canali and R. Lancellotti, "Automatic virtual machine clustering based on Bhattacharyya distance for multi-cloud systems," in *Proc. of International Workshop on Multi-cloud Applications and Federated Clouds*, Prague, Czech Republic, Apr. 2013, pp. 45–52.

[3] ——, "Improving Scalability of Cloud Monitoring Through PCA-Based Clustering of Virtual Machines," *Journal of Computer Science and Technology*, vol. 29, no. 1, pp. 38–52, 2014.

[4] ——, "Exploiting Classes of Virtual Machines for Scalable IaaS Cloud Management," in *Proc. of IEEE Symposium on Network Cloud Computing and Applications (NCCA)*, Munich, Germany, Jun. 2015.

[5] D. Durkee, "Why cloud computing will never be free," *Queue*, vol. 8, no. 4, pp. 20:20–20:29, Apr. 2010.

[6] C. Canali and R. Lancellotti, "An Adaptive Technique to Model Virtual Machine Behavior for Scalable Cloud Monitoring," in *Proc. of IEEE Symposium on Computers and Communications (ISCC)*, Madeira, Portugal, Jun. 2014.

[7] B. Speitkamp and M. Bichler, "A Mathematical Programming Approach for Server Consolidation Problems in Virtualized Data Centers," *Services Computing, IEEE Transactions on*, vol. 3, no. 4, pp. 266–278, Oct 2010.

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

[9] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proc. of IEEE INFOCOM*, Shanghai, China, April 2011.
[10] T. Setzer and A. Stage, "Decision support for virtual machine reassignments in enterprise data centers," in *Proc. of Network Operations and Management Symposium*, Osaka, Japan, Apr. 2010.
[11] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A Scalable Application Placement Controller for Enterprise Data Centers," in *Proc. of 16th World Wide Web Conference (WWW'07)*, Banff, Canada, May 2007.
[12] I. Moreno and J. Xu, "Neural network-based overallocation for improved energy-efficiency in real-time cloud environments," in *Proc. IEEE Int. Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, April 2012, pp. 119–126.
[13] M. Kesavan, I. Ahmad, O. Krieger, R. Soundararajan, A. Gavrilovska, and K. Schwan, "Practical Compute Capacity Management for Virtualized Datacenters," *Cloud Computing, IEEE Transactions on*, vol. 1, no. 1, pp. 1–1, Jan 2013.
[14] B. Addis, D. Ardagna, B. Panicucci, M. Squillante, and L. Zhang, "A Hierarchical Approach for the Resource Management of Very Large Cloud Platforms," *Dependable and Secure Computing, IEEE Transactions on*, vol. 10, no. 5, pp. 253–272, Sept 2013.
[15] J. Cao, K. Hwang, K. Li, and A. Y. Zomaya, "Optimal Multiserver Configuration for Profit Maximization in Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1087–1096, 2013.
[16] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proc. of Conference on Networked systems design and implementation (NSDI)*, Cambridge, Apr. 2007.
[17] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in *OSDI*, M. I. Seltzer and P. J. Leach, Eds. USENIX Association, 1999, pp. 173–186.
[18] E. Choi and C. Lee, "Feature extraction based on the Bhattacharyya distance," *Pattern Recognition*, vol. 36, no. 8, pp. 1703–1709, Aug. 2003.
[19] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, "A survey of kernel and spectral methods for clustering," *Pattern Recognition*, vol. 41, no. 1, pp. 176–190, Jan. 2008.
[20] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.
[21] N. Pal and J. Bezdek, "On cluster validity for the fuzzy c-means model," *Fuzzy Systems, IEEE Transactions on*, vol. 3, no. 3, pp. 370–379, Aug 1995.
[22] M. Andreolini, M. Colajanni, and M. Pietri, "A scalable architecture for real-time monitoring of large information systems," in *Proc. IEEE Symposium on Network Cloud Computing and Applications*, London, UK, Dec. 2012.
[23] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud Monitoring: A Survey," *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, Jun. 2013.
[24] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," in *Advances in Computers, Volume 82*, M. Zelkowitz, Ed. Academic Pres, 2011.
[25] R. Mehrotra, A. Dubey, S. Abdelwahed, and W. Monceaux, "Large scale monitoring and online analysis in a distributed virtualized environment," in *Proc. of 8th IEEE International Conference and Workshops on Engineering of Autonomic and Autonomous Systems*, Las Vegas, USA, April 2011, pp. 1–9.
[26] J. Shao and Q. Wang, "A Performance Guarantee Approach for Cloud Applications Based on Monitoring," in *Proc. of IEEE 35th Annual Computer Software and Applications Conference Workshops*, Munich, Germany, July 2011, pp. 25–30.
[27] F. Azmandian, M. Moffie, J. Dy, J. Aslam, and D. Kaeli, "Workload characterization at the virtualization layer," in *Proc. IEEE Int. Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*, Singapore, July 2011.
[28] A. Kertesz, G. Kecskemeti, M. Oriol, P. Kotcauer, S. Acs, M. Rodrguez, O. Merc, A. Marosi, J. Marco, and X. Franch, "Enhancing Federated Cloud Management with an Integrated Service Monitoring Approach," *Journal of Grid Computing*, vol. 11, no. 4, pp. 699–720, 2013.
[29] M. Andreolini, M. Colajanni, and S. Tosi, "A software architecture for the analysis of large sets of data streams in cloud infrastructures," in *Proc. of 11th IEEE Conference on Computer and Information Technology (IEEE CIT 2011)*, Cyprus, Aug.-Sept. 2011.
[30] "Cloudwatch - Amazon Web services," 2014, – http://aws.amazon.com/cloudwatch/.
[31] "Logicmonitor," 2014, – http://www.logicmonitor.com/.
[32] J. Povedano-Molina, J. M. Lopez-Vega, J. M. Lopez-Soler, A. Corradi, and L. Foschini, "Dargos: A highly adaptable and scalable monitoring architecture for multi-tenant clouds," *Future Generation Computer Systems*, vol. 29, no. 8, pp. 2041–2056, 2013.
[33] R. Buyya, R. N. Calheiros, and X. Li, "Autonomic Cloud computing: Open challenges and architectural elements," in *Proc. of 3rd International Conference on Emerging Applications of Information Technology, EAIT 2012*, 2012, pp. 3–10.
[34] Y. Xu, Z. Musgrave, B. Noble, and M. Bailey, "Bobtail: Avoiding long tails in the cloud," in *Proc. of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI)*, Lombard, IL, Apr. 2013.
[35] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," in *Proc. of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI)*, Boston, MA, Mar. 2011.
[36] C. Mastroianni, M. Meo, and G. Papuzzo, "Probabilistic consolidation of virtual machines in self-organizing cloud data centers," *Cloud Computing, IEEE Transactions on*, vol. 1, no. 2, pp. 215–228, July 2013.
[37] R. Zhang, R. Routray, D. M. Eyers *et al.*, "IO Tetris: Deep storage consolidation for the cloud via fine-grained workload analysis," in *IEEE Int. Conf. on Cloud Computing*, Washington, DC USA, July 2011.
[38] K. R. Jayaram, C. Peng, Z. Zhang, M. Kim, H. Chen, and H. Lei, "An empirical analysis of similarity in virtual machine images," in *Proc. of the Middleware 2011 Industry Track Workshop*, ser. Middleware'11. Lisbon, Portugal: ACM, 2011, pp. 6:1–6:6.
[39] A. Sharma and K. K. Paliwal, "Fast principal component analysis using fixed-point algorithm," *Pattern Recognition Letters*, vol. 28, no. 10, pp. 1151–1155, July 2007.

Claudia Canali Claudia Canali received Laurea degree summa cum laude in computer engineering from the University of Modena and Reggio Emilia in 2002, and Ph.D. in Information Technologies from the University of Parma in 2006. She is a researcher at the Department of Engineering of the University of Modena and Reggio Emilia since 2008. Her research interests include cloud computing, social networks analysis, and wireless systems for mobile Web access. On these topics, Claudia Canali published more than fifty papers on international journals and conferences. She is a member of IEEE Computer Society. For additional information: http://weblab.ing.unimo.it/people/canali

Riccardo Lancellotti Riccardo Lancellotti received the Laurea Degree in computer engineering summa cum laude from the University of Modena and Reggio Emilia in 2000 and the Ph.D. in computer engineering from the University of Roma "Tor Vergata" in 2003. He is a researcher at the University of Modena and Reggio Emilia since 2005. His research interests include geographically distributed systems, cloud computing, social networks and peer-to-peer systems. On these topics he published almost sixty papers on international journals and conferences. He is a member of the IEEE Computer Society and ACM. For additional information: http://weblab.ing.unimo.it/people/lancellotti