Noname manuscript No.
(will be inserted by the editor)

Exploiting Ensemble Techniques for Automatic Clustering of Virtual Machine Clustering in Cloud Systems

Claudia Canali · Riccardo Lancellotti

the date of receipt and acceptance should be inserted later

Abstract Cloud computing has recently emerged as a new paradigm to provide computing services through large-size data centers where customers may run their applications in a virtualized environment. The advantages of cloud in terms of flexibility and economy led many enterprises to migrate from local data centers to cloud platforms, thus contributing to the success of such infrastructures. However, as size and complexity of cloud infrastructures grows, new scalability issues arises in the monitoring and management of resources. Available solutions to cope with such issues typically consider every Virtual Machine (VM) as a black box each with independent characteristics, and face scalability issues by reducing the number of monitored resources, considering in most cases only the CPU usage patterns. We claim that scalability issues can be addressed by leveraging the similarity between VMs in terms of resource usage patterns. In this paper, we propose an automated methodology to cluster similar VMs starting from their usage information about multiple resources, assuming no knowledge of the software executed on them. This is an innovative methodology that combines the Bhattacharyya distance and ensemble techniques to provide a stable evaluation of similarity between probability distributions of multiple VM resource usage, considering both system- and network-related data.

We evaluate the methodology through a set of experiments on data coming from an enterprise data center. We show that our proposal achieves high and stable performance in automatic VMs clustering, with a reduction in the amount of data collected of one order of magnitude which allows to lighten the monitoring requirements of a cloud data center.

Keywords clustering; clustering ensemble; bhattacharyya distance; cloud computing

Department of Engineering “Enzo Ferrari”
University of Modena and Reggio Emilia
E-mail: {claudia.canali, riccardo.lancellotti}@unimore.it

1 Introduction

In the few last years, the rapid growth in demand for modern applications combined with the shift to the Cloud computing paradigm have led to the establishment of large-scale virtualized data centers following the so-called *Infrastructure as a Service* (IaaS) paradigm. The growing popularity of the cloud is a catalyst for the migration from traditional data center, with traditional server consolidation strategies to IaaS cloud data centers. This migration process is testified by the abundance of cloud scenarios characterized by *long-term commitments*, where customers outsource their data centers to a cloud provider purchasing VMs for extended periods of time (for example, using the Amazon so-called *reserved instances*). This scenario is and is expected to be a significant part of the cloud ecosystem also in the future [13]. The final result is a scenario where virtualization-based data centers host several customers applications, where each application consists of different software components (e.g., the tiers of a multi-tier Web application). In a virtualized data center, each physical server hosts multiple virtual machines (VMs) running different software components with complex and heterogeneous resource demand behavior. As we consider long term commitments of cloud costumers, VMs tend to not change frequently the software component they are running and VMs are acquired or released with relatively low frequency.

Due to the rapid increase in size and complexity of IaaS infrastructures, the processes of monitoring and managing cloud data centers are becoming challenging tasks. The monitoring process presents scalability issues due to the amount of data to collect and store when a large number of VMs is considered, each with several resources (e.g., CPU, memory, network) which are monitored with high sampling frequency [2]. Management strategies assume that each VM is a stand-alone object, whose behavior is independent from the other VMs of the cloud infrastructure, and take decisions on the basis of information coming from the resource monitoring system, thus not scaling well due to the large amount of data to analyze [31].

A typical approach to address the scalability issues about monitoring and management of a cloud data center is to reduce the problem size. Available solutions tend to reduce the number of VM resources that are taken into account, typically considering only CPU-related information [4,16,31]. However, this approach is likely to suffer from important drawbacks, because limiting the monitoring to the CPU resource may not be sufficient to support VM consolidation when I/O bound or network bound applications are involved. This motivates the need of a scalable collection of data about multiple resources to support management in cloud data centers [24].

We argue that the scalability of monitoring and management in cloud infrastructures may be addressed by leveraging the similarity between VM behaviors, considering VMs not as stand-alone objects but as members of classes composed by objects which are running the same software component (e.g., Web servers or DBMS). In particular, we recall that we focus ona cloud scenario characterized by *long-term commitments*, where customer VMs change the software component they are running with a relatively low frequency, in the order of weeks or months. Once identified classes of similar VMs, we may improve monitoring scalability by focusing fine-grained observations only on a few representative VMs for each class.

The main contribution of this paper is the proposal of an automated methodology to cluster together similar VMs in a private IaaS cloud data center on the basis of their resource usage. Our approach is consistent with the IaaS vision [32, 37] as it does not require any direct knowledge of the application logic inside a software component, but it only relies on the information about OS-level resource usage of each VM. Furthermore, we aim to guarantee high performance in the VM clustering and, even more important, we aim to guarantee stable performance, that is we ensure that the clustering outcome is scarcely dependent on the parameters of the methodology.

The proposed methodology exploits the distance of Bhattacharyya [10], a statistical technique measuring the similarity of different discrete probability distributions, to define the similarity between VMs and determine which VMs are following the same behavioral patterns. A further qualifying point of our proposal is the use of clustering ensemble techniques to improve the stability of the methodology performance by integrating a quorum-based mechanism into the clustering process.

A main advantage of our methodology is that we take into account multiple resources, including network and memory related information, differently from most of the current solutions, which mainly consider CPU-related information. To the best of our knowledge, the proposal of techniques for automatically clustering VMs with similar behavior is a new problem, only recently analyzed in [7, 6, 8]. However, the solutions that model VM behavior using the correlation between the metrics provide poor performance in the presence of short time series and in the case where some VMs remain idle for periods of time [7, 6]. On the other hand, available solutions based on the Bhattacharyya distance have performance that depend heavily on the parameters used when computing such distance. Our proposal outperforms all these solutions and provides a major advantage in the performance stability with respect to the previous results. We apply the proposed methodology to a dataset coming from an enterprise private cloud environment with VMs running Web servers and DBMS software. We show that our methodology can achieve high performance in clustering VMs, with a reduction in the amount of collected data samples by an order of magnitude. Furthermore, we demonstrate that exploiting ensemble techniques to merge together multiple clustering solutions is of key importance to obtain stable performance in the clustering process.

The remainder of this paper is organized as follows. Section 2 presents the reference scenario and motivates our proposal, while Section 3 describes the proposed methodology for automatically clustering VMs in a cloud environment. Section 4 described the experimental testbed used to evaluate our methodology, while Section 5 presents the case study used to evaluate our methodology and describes the results of our experiments. Section 6 discusses the related work and Section 7 concludes the paper with some final remarks.

2 Motivation and reference scenario

We now provide a motivating example for our proposal. We consider a multi-tier Web application characterized by a high degree of horizontal replication. In our example we have an application for e-health deployed on 110 VM, with a nearly half of the VM devoted to the back-end tier and the other half supporting the front.end tier. The application is used by ***X*** users, with the typical usage patterns characterized by high resource utilization in the office hours and lower utilization during the night. We consider that this application is meant to be migrated from a traditional data center to a cloud platform. This scenario is a typical case where moving to a IaaS platform involves long term commitments, that is the VM are unlikely to change in number and in functions for ling periods of time. As the cloud provider has no knowledge on the characteristics of each VM, the most straightforward option to provide a clear picture of the application status is to monitor every VM at the deepest level of detail. Assuming that monitoring considers $\overline{K}$ metrics for each VM that are collected with a frequency of 1 sample every 5 minutes, we have to manage a volume of data $288 \cdot \overline{K}$ samples per day per VM. Considering our scenario with 110 VMs, the total amount of data is in the order of $3.2 \times 10^4 \cdot \overline{K}$ samples per day. Our proposal identifies two sets of VM with different behavior and monitors at the granularity of 5 minutes only a few representative VMs per class, while the remaining VMs can be monitored with a coarse-grained granularity, for example of 1 sample every few hours. Assuming to select 3 representatives for each of the 2 VM classes the amount of data to collect after clustering is reduced to $17.2 \times 10^2 \cdot \overline{K}$ samples per day for the class representatives; for the remaining 104 VMs, assuming to collect one sample of the $\overline{K}$ metrics every 6 hours for VM, the data collected is in the order of $4.2 \times 10^2 \cdot \overline{K}$ samples per day. Hence, we observe that our proposal may reduce the amount of data collected by nearly a factor of 15, from $3.2 \times 10^4 \cdot \overline{K}$ to $21.4 \times 10^2 \cdot \overline{K}$.

As a side note it is worth to observe that, while our experimental evaluation will focus on this case study, Web applications are not the only type of applications that can benefit from our approach: every information system characterized by highly replicated components can benefit from our proposal when is migrated from a traditional data center to a IaaS cloud platform.

Let us now detail how the proposed approach to monitoring ca be integrated in a IaaS cloud system. We recall that monitoring is aimed to support efficient use of the system resources, while avoiding overload conditions on the physical servers. We consider a management strategy for the cloud system which consists of two mechanisms, as in [17]: (a) a reactive VM relocation that exploits live VM migration when overloaded servers are detected [38]; (b) a periodic consolidation strategy that places customer VMs on as few physical servers as possible to reduce the infrastructure costs and avoid expensive resource over provisioning [3, 30].

The consolidation task is carried out periodically with the aim to produce an optimal (or nearly optimal) VM placement which reduces the number of shared hosts. Existing consolidation decision models typically try to predict VM workload over a planned period of time (e.g., few hours) based on resource usage patterns observed on past measurements, that are usually carried out with a fine-grained granularity (e.g., 5-minute intervals) [3, 30].

The proposed methodology aims to address cloud monitoring scalability issues by automatically clustering similar VMs. The main goal is to cluster together VMs running the same software component of the same customer application, and therefore showing similar behaviors in terms of resource usage. For each identified class,

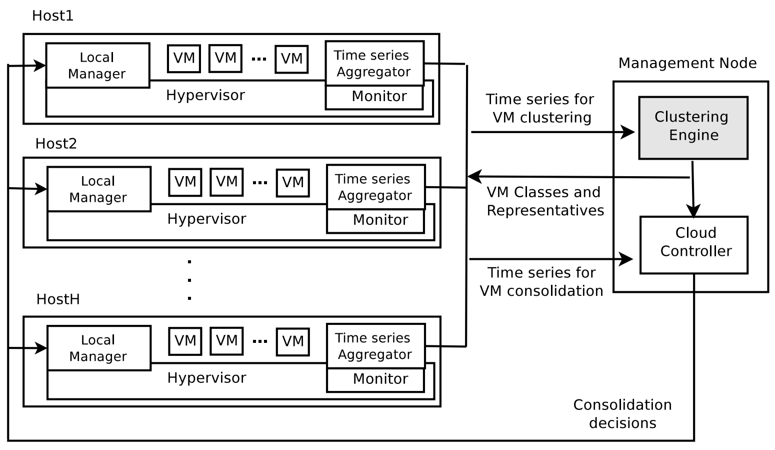


Fig. 1 Cloud system

only few representative VMs are monitored with fine-grained granularity to collect information for the periodic consolidation task, 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 coarse-grained granularity to identify behavioral drifts that could determine a change of class. At the same time, sudden changes leading to server overload are handled by the reactive VM relocation mechanism. This approach allows to significantly reduce the amount of information collected for periodic consolidation strategies.

The process of VM clustering is carried out periodically with a frequency that allows to cope with changes in the VM classes. We recall that our reference scenario is a cloud environment characterized by long-term commitment between cloud customers and providers, where we can assume that the software component hosted on each VM changes with a relatively low frequency in the order of weeks or months. Hence, clustering can be carried out with a low periodicity (e.g., once every one or few weeks). Furthermore, the clustering may be triggered when the number of exceptions in VMs behavior exceeds a given threshold, where for exception we mean newly added VMs or clustered VMs that changed their behavior with respect to the class they belong to. Anyway, a precise determination of the activation period or strategy of the clustering process is out of the scope of this paper.

Figure 1 depicts the reference scenario. The scheme represents a cloud data center where each physical server, namely *host*, runs several VMs. On each host we have an *hypervisor*, with a *monitor* process that periodically collects resources usage time series for each VM. The collected data are sent to the *time series aggregator* running on the host. The time series aggregator selects the data to be communicated (with different periodicity) to the *clustering engine*, which executes the proposed methodology to automatically cluster VMs, and to the *cloud controller*, which is responsible for running the consolidation strategy. On each host we have also a *local manager*, which performs two tasks. First, it is responsible for taking decisions about live VM migration to trigger in case of host overload [38]. Second, it executes the consolidation decisions periodically communicated by the cloud controller.

Let us now consider the dynamics occurring in the considered cloud system to support VM clustering and server consolidation. The process of VM clustering starts from the collection of time series describing the resources usage for each VM over a certain period of time. The monitor processes are responsible for this data collection. Then, the time series aggregators of each host send the data to the clustering engine, which executes the proposed methodology with the aim to cluster together VMs belonging to the same customer application and running the same software component. Once the clustering is complete, few representative VMs are selected for each class. It is worth to note that more than two representatives (at least three) should be selected for each class, 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 [9]. When only one VM is changing its behavior, we can still use the remaining two representatives to identify the rogue VM and to preserve a description of the cluster.

The information on VM classes and selected representatives are sent to the time series aggregators on each host and to the cloud controller for periodic consolidation task. The time series aggregators selectively collect the resource time series of the representative VMs of each class, then send the data to the cloud controller. This latter component carries out the consolidation task, 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 managers on each host to be executed.

3 Methodology

In this section we describe the methodology to automatically cluster VMs in a cloud data center on the basis of the usage information about multiple resources. For each customer application, we aim to group together VMs which are running the same software component (e.g., VMs belonging to the same tier of a Web application), and are therefore showing similar behaviors in term of resource usage.

We recall that our reference scenario is a private cloud where we assume that the software components hosted on each VM do not change for long periods of time (i.e., they remain the same for months), and VMs are seldom acquired or released. The process of clustering similar VMs and the related collection of data about VM resource usage is carried out periodically with a frequency that allows to cope with changes in the VM behavior, for example once every several weeks. Hence, the actual computational cost of the methodology is not considered as critical, due to its low invocation frequency.

The main benefit achievable through the proposed methodology is to improve the scalability of resource usage monitoring to support data center management and server consolidation by means of VMs clustering. Indeed, we may select some representative VMs for each class and perform fine-grained monitoring only on these representatives. The resource usage of the representatives is used to describe *every*

VM belonging to the same class, while the remaining VMs can be monitored according to a more coarse-grained approach, with a meaningful reduction in the amount collected data. The choice to consider more than one representative for each class is due to the possibility that a selected class representative unexpectedly changes its behavior with respect to the class it belongs to. When more than one representative is used, quorum-based techniques can be exploited to identify a misbehaving VM within the list of representatives.

To measure the behavior similarity between VMs, the proposed methodology exploits the Bhattacharyya distance. This statistical technique determines the similarity between two probability distributions: in our case, we consider the probability distributions of the usage samples of the considered resources, that we will call *metrics*. We should consider that the Bhattacharyya distance provides *per-metric* distances between VMs. Hence, the similarity of different VMs is represented through a multi-dimensional distance, leaving open the issue of how to merge such information for the clustering of similar VMs. We address this issue through a technique based on clustering ensemble, which allows us to merge information about multiple metrics and improve the stability of the methodology performance with respect to its design parameters. The clustering ensemble approach will be compared against an alternative approach, proposed by the authors in [8] and described later in this section, that reduces the dimensionality of the per-metric set by mapping them into an euclidean space.

The main steps of the proposed methodology are outlined in the main branch of Figure 2:

– Extraction of a *quantitative model* describing the VM behavior
– Generation of a set of *Bhattacharyya distance matrices* representing VMs similarities for the single metrics
– *Per-metric clustering* of VMs based on the Bhattacharyya distance matrices
– *Clustering ensemble* to merge the per-metric clustering solutions

The right branch in Figure 2 presents the alternative approach, namely *euclidean clustering*, where the per-metric distance matrices are merged into a single distance matrix using an euclidean space. The matrix is then fed into a final clustering algorithm.

Throughout the remaining of this section we describe in detail each step of the proposed methodology and of the considered alternative approach.

3.1 Extraction of quantitative model for VM behavior

We now formally define the quantitative model chosen to represent the behavior of each VM, and discuss a critical design choice involved in this step.

Given a set of N VMs, the first step of the methodology aims at representing the behavior of each VM $n, \forall n \in [1, N]$, taking into account for a set of M metrics, where each metric $m \in [1, M]$ represents the usage of a VM resource.

Let $(\mathbf{X}_1^n, \mathbf{X}_2^n, \ldots, \mathbf{X}_M^n)$ be a set of time series, where $\mathbf{X}_m^n$ is the vector consisting of the samples of the resource usage represented by the metric m of the VM n. We

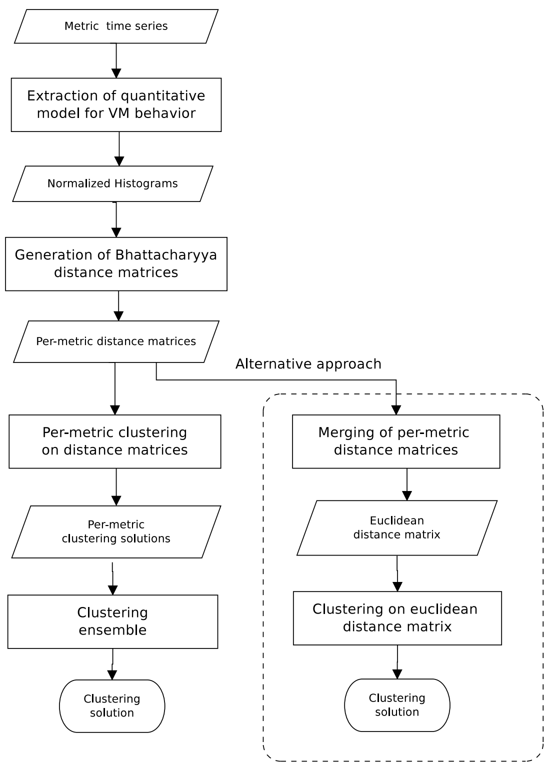

Metric time series
Extraction of quantitative model for VM behavior
Normalized Histograms
Generation of Bhattacharyya distance matrices
Per-metric distance matrices
Alternative approach
Per-metric clustering on distance matrices
Per-metric clustering solutions
Clustering ensemble
Clustering solution
Merging of per-metric distance matrices
Euclidean distance matrix
Clustering on euclidean distance matrix
Clustering solution

Fig. 2 Proposed methodology and alternative approach

choose to consider the probability density function $p(\mathbf{X}_m^n)$ of each time series as the description of the behavior of metric m on VM n. We represent the probability function of a finite-length time series through a *normalized histogram*. The histogram is composed by *bins*, each associated to an interval of values the metric can take. Each bin represents the frequency of samples for the considered interval, that is the fraction of samples from the time series that fall within the bin interval.

If B_m is the number of bins considered for metric m, the normalized histogram from the time series $\mathbf{X}_m^n$ is a set $\mathbf{H}_m^n = \{h_{b,m}^n \forall b \in [1, B_m]\}$, where $h_{b,m}^n$ is the value associated to the b-th histogram bin and defined as:

$$h_{b,m}^n = \frac{|\{x \in \mathbf{X}_m^n : x > X_m^l(b), x \leq X_m^U(b)\}|}{|\mathbf{X}_m^n|}$$

where $|\{x \in \mathbf{X}_m^n : x > X_m^l(b), x \leq X_m^U(b)\}|$ is the number of samples in the interval $[X_m^l(b), X_m^U(b)]$ and $|\mathbf{X}_m^n|$ is the number of samples in the time series. The upper and lower bounds of the bin b are defined as: $X_m^l(b) = Xmin_m + (b-1)\Delta x_m$ and $X_m^U(b) = Xmin_m + b\Delta x_m$, where $Xmin_m$ is the minimum value of metric m for every VM, $Xmax_m$ is the maximum value of metric m for every VM, and Δx_m is the width of each bin for metric m, that is $\Delta x_m = \frac{Xmin_m - Xmax_m}{B_m}$. Figure 3 provides a graphical example of the above defined histogram. This definition ensures that for each metric m the number of bins is the same for every VM. This latter property is fundamental because in the following steps of the methodology we need to compare same sized histograms referring to different VMs.

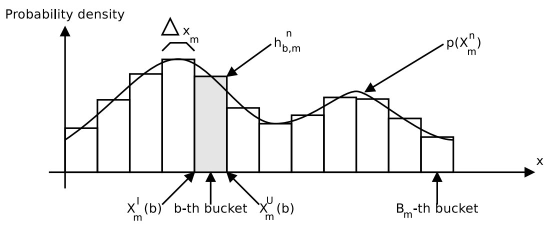


Fig. 3 Histogram example

The extraction of the quantitative model raises a critical design choice which may affect the final outcome of the clustering process, that is the determination of the number of bins in each histogram. The difficulty in estimating the number of bins is due to the heterogeneity of the metrics considered in the VM monitoring. As there is no a universally accepted way to define the "best" number of bins, we consider three widely adopted rules of thumb for the estimation of the number of bins in the histograms: the Freedman-Diaconis rule [15], the Scott rule [29] and the square root rule. The impact of the rule choice on the final outcome of the clustering will be investigated in the experimental evaluation.

The Freedman-Diaconis rule defines the number of bins as a function of the inter-quartile range of the data set, that is the difference between 1^{st} and 3^{rd} quartile of the samples. According to the FD rule $B_m = 2\frac{\mathrm{IQR}(\mathbf{X_m})}{\sqrt[3]{|\mathbf{X_m}|}}$, where $|\mathbf{X_m}|$ is the number of samples in the time series for metric m and $\mathrm{IQR}(\mathbf{X_m})$ is the inter-quartile range of the time series.

The Scott rule determines the number of bins as a function of the standard deviation of the samples. For this reason, the Scott rule is typically adopted for samples following a Gaussian distribution. According to the Scott rule, we define the number of bins as $B_m = \frac{3.5\sigma_m}{\sqrt[3]{|\mathbf{X}|}}$, where σ_m is the standard deviation of the samples in the time series for metric m.

The square root rule is adopted in some software for data management and analysis, including the popular Excel spreadsheet, and simply defines the number of bins as the square root of the number of samples, that is $B_m = \sqrt{|\mathbf{X_m}|}$.

3.2 Generation of Bhattacharyya distance matrices

The second step of the methodology consists in building a set of distance matrices to define similarities between VMs starting from the histogram-based representation of the VM behavior for each single metric.

To build this set of per-metric distance matrices we exploit the Bhattacharyya distance [5], which is used to measure the similarity between two data sets based on their probability distribution. The Bhattacharyya distance $D_m(n_1, n_2)$ computed according to metric m between two VMs n_1 and n_2 is defined as follows:

$$D_m(n_1, n_2) = -ln(\sum_{b=1}^{B} \sqrt{h_{b,m}^{n_1} \cdot h_{b,m}^{n_2}})$$

where $h_{b,m}^{n_1}$ and $h_{b,m}^{n_2}$ are the b-th bin values in the histograms of metric m for VM n_1 and VM n_2, respectively. Since the histograms are normalized, the Bhattacharyya distance may take values ranging from 0 (identical histograms) to ∞ (histograms where the product of every pair of bins is 0), as shown in Figure 4.

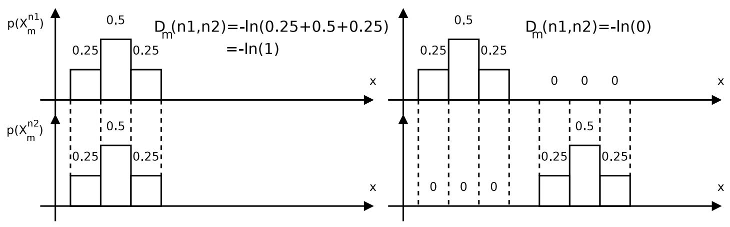


Fig. 4 Bhattacharyya distance example

The distances between each couple of VMs for any given metric m are organized in a set of matrices $\mathbf{D}_m$, $m \in [1, M]$. Due to the nature of the Bhattacharyya distance, each of these distance matrices is symmetrical and the elements on the main diagonal have zero value.

3.3 Per-metric clustering on distance matrices

This step of the methodology aims to obtain a clustering solution from each per-metric distance matrix $\mathbf{D}_m$, $m \in [1, M]$. To this aim, we need to transform each distance matrix $\mathbf{D}_m$ into a *similarity* matrix $\mathbf{S}_m$. This step is carried out by applying a Gaussian kernel operator, that is a common approach to translate distance into similarity [12]. Specifically, we define the similarity as $s_{i,j} = e^{\frac{-d_{i,j}^2}{\sigma^2}}$, where $d_{i,j}$ is an element of a distance matrix and σ is a blurring coefficient of the kernel function [20], typically set to the value of 0.5. Preliminary analyses on the impact of the σ coefficient on the clustering results suggest that the choice of this parameter is not critical for the performance of the clustering algorithms.

To cluster together elements starting from a similarity matrix, traditional algorithms based on coordinate systems (such as k-means or kernel k-means) are not viable options. On the other hand, spectral clustering is widely adopted in these circumstances because it is explicitly designed to manage as input a similarity matrix or a matrix-based representation of graphs [19, 28].

The spectral clustering algorithm computes the Laplacian operator from the input similarity matrix. The eigenvalues and eigenvectors of the Laplacian are then used to extract a new coordinate system that is fed into a k-means clustering phase [22]. About this last phase of the clustering process, we must recall that the k-means algorithm starts with a random set of centroids. To ensure that the k-means result is not affected by local minimums, we iterate the k-means multiple times and we compare the ratio between the sum of squares of the distances among elements belonging to different clusters (inter-cluster) and elements belonging to the same cluster (intra-cluster). Then, we finally select the best solution across multiple k-means runs, that is the solution maximizing inter-cluster distances and minimizing intra-cluster distance.

The output of the clustering is one vector $\mathbf{C}_m$ for each metric $m \in [1, M]$, where the n-th element c_m^n is the identifier of the cluster to which the VM n is assigned.

3.4 Clustering ensemble

The final step of clustering ensemble combines the set of M clustering solutions $\mathbf{C}_m$, $m \in [1, M]$, into a *co-occurrence* matrix $\mathbf{A}$. The matrix $\mathbf{A}$ stores for each pair of VMs n_1 and n_2 the number of clustering solutions where n_1 and n_2 occurs in the same cluster, divided by the total number of clustering solutions M.

The co-occurrence matrix represents a measure of the affinity between the VMs and is used as a similarity matrix for a subsequent clustering step [33]. Again, we use the spectral clustering to create the final clustering solution $\mathbf{C}_s$, that groups together similarly behaving VMs. The use of clustering ensemble aims to improve the stability of the clustering solution with respect to the parameters of the methodology, such as the number of bins or the length of the time series used. Indeed, merging multiple clustering solutions in such way implements a sort of quorum-based decision system, which is likely to improve the robustness of the clustering performance [18, 19].

3.5 Alternative approach

The *euclidean clustering* is an alternative approach to the above presented methodology which has been previously proposed in [8]. The euclidean clustering, depicted in the right branch of Figure 2, aims to combine the M Bhattacharyya distance matrices $\mathbf{D}_m$ to build a single distance matrix expressing the distances among all VMs. To this aim, we consider that the set of Bhattacharyya distance matrices represent a multi-dimensional distance between the VMs. To reduce the dimensionality of the problem from M to 1, we combine the multiple dimensions as if we were in an euclidean space. Hence, we define a multimetric-based distance between VMs as the sum of squares of the corresponding Bhattacharyya distances for each metric, that is:

$$D_e(n_1, n_2) = \sqrt{\sum_{m=1}^{M} D_m(n_1, n_2)^2}$$

where $D_m(n_1, n_2)$ is Bhattacharyya distance between n_1 and n_2 according to metrics m.

The new euclidean distance matrix $\mathbf{D}_e$ is then fed into the spectral clustering algorithm to obtaining the final solution $\mathbf{C}_e$ of the euclidean clustering approach.

4 Experimental testbed

*** DA SPOSTARE **** Let us now describe the application of the proposed methodology to the considered case study. The methodology has been implemented using popular technologies for data management and analysis. Specifically, we use the R language [1] for the statistical analysis functions, Python [2] for the task of reading and writing data, and as a wrapper for the R core. Finally, we use Bourne shell [3] to invoke the main steps of the methodology. These choices ensure that our proposal can be easily deployed directly in currently available cloud infrastructures.

To evaluate the performance of the proposed methodology, we consider a dataset coming from an enterprise data center supporting one customer Web-based application deployed according to a multi-tier architecture. The data center is composed of 10 nodes on a Blade-based system and exploits virtualization to support the Web application. The nodes host 110 VMs that are divided between two classes: Web servers and back-end servers (that are DBMS).

We collect detailed data about the resource usage of every VM for different periods of time, ranging from 1 to 180 days. The samples are collected with a frequency of 5 minutes. 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 1 along with a short description.

Table 1 Virtual machine metrics

Metric		Description
X_1	SysCallRate	Rate of system calls [req/sec]
X_2	CPU	CPU utilization [%]
X_3	DiskAvl	Available disk space [%]
X_4	CacheMiss	Cache miss [%]
X_5	Memory	Physical memory utilization [%]
X_6	PgOutRate	Rate of memory pages swap-out [pages/sec]
X_7	InPktRate	Rate of network incoming packets [pkts/sec]
X_8	OutPktRate	Rate of network outgoing packets [pkts/sec]
X_9	AliveProc	Number of alive processes
X_{10}	ActiveProc	Number of active processes

To the above metrics, we apply the methodology described in Section 3. For each metric m of VM n we compute the normalized histogram $\mathbf{H}_m^n$, expressing the behavior of the VM according to considered metric. Then, the generated histograms are used to compute the per-metric Bhattacharyya distance matrix $\mathbf{D}_m$, representing the distances between each pair of VMs according to metric m. The next step of the proposed ensemble approach applies the spectral clustering to each distance matrix $\mathbf{D}_m$. We run 10^3 times the internal k-means clustering and we select the best solution (as described in Section 3.3) which represents the per-metric clustering solution $\mathbf{C}_m$. Finally, we carry out the ensemble of the per-metric clustering solutions, creating the

[1] R project home page: http://www.r-project.org/
[2] Python home page: http://www.python.org/
[3] Bourne shell home page: http://www.gnu.org/software/bash/

co-occurrence matrix and applying on it the final spectral clustering step to produce the final vector solution $\mathbf{C}_s$.

We also implement the alternative approach based on euclidean clustering. In this case, we take the Bhattacharyya distance matrices $\mathbf{D}_m$ produced as output of the second methodology step, and compute the multimetric-based distance matrix $\mathbf{D}_e$, which represent the distance between each pair of VMs as if we were in an euclidean space. A final step of spectral clustering is applied to the matrix $\mathbf{D}_e$ and produces the final vector solution $\mathbf{C}_e$.

To evaluate the performance of the proposed clustering methodology, we consider as the main metric the clustering *purity* [1]. Purity, that is one of the most popular metrics for cluster evaluation, considers the fraction of correctly identified VMs as the measure of the clustering performance. Specifically, purity is defined by comparing the generic final solution $\mathbf{C}$ of the VM clustering with the ground truth vector $\mathbf{C}^*$, which represents the correct classification of Web servers and DBMS servers into two clusters. Purity is thus defined as:

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

where c^n is the cluster to which VM n is assigned in the considered solution, c^{n*} is the *correct* classification of VM n, and N is the number of clustered VMs.

**** Questo andrebbe spostato nella sezione 2 **** To understand the benefits for the monitoring system achievable through the proposed methodology, it is interesting to determine the potential reduction in the amount of data collected to support consolidation in the described scenario. Assuming that the global consolidation strategy considers $\overline{K}$ metrics for each VM that are collected with a frequency of 1 sample every 5 minutes, we have to manage a volume of data $288 \cdot \overline{K}$ samples per day per VM. Considering our scenario with 110 VMs, the total amount of data is in the order of $3.2 \times 10^4 \cdot \overline{K}$ samples per day. After the clustering, we need to continue monitoring every 5 minutes only a few representative VMs per class, while the remaining VMs can be monitored with a coarse-grained granularity, for example of 1 sample every few hours. Assuming to select 3 representatives for each of the 2 VM classes, as described in Section 2, the amount of data to collect after clustering is reduced to $17.2 \times 10^2 \cdot \overline{K}$ samples per day for the class representatives; for the remaining 104 VMs, assuming to collect one sample of the $\overline{K}$ metrics every 6 hours for VM, the data collected is in the order of $4.2 \times 10^2 \cdot \overline{K}$ samples per day. Hence, we observe that our proposal may reduce the amount of data collected by nearly a factor of 15, from $3.2 \times 10^4 \cdot \overline{K}$ to $21.4 \times 10^2 \cdot \overline{K}$. ****

5 Performance evaluation

In this section, we apply the proposed methodology to the described testbed to evaluate the performance of automatically clustering similar VMs based on their resource usage. The main goals of this experimental evaluation are:

- To evaluate the performance of the proposed methodology, based on clustering ensemble, and compare it with the alternative approach, based on the euclidean distance between VMs

– To investigate the sensitivity of the methodology performance with respect to the number of histogram bins used to generate the quantitative description of VM behavior
– To evaluate the impact on performance of reducing the set of metrics considered for VM clustering
– To perform a sensitivity analysis of methodology performance and clustering time with respect to the number of VMs considered for clustering

All experiments evaluate the performance of the methodology considering time series of VM metric samples of different lengths, ranging from 1 to 180 days. Except when differently stated, the Friedman-Diaconis rule is used to compute the number of bins in the histograms of the VM metrics.

5.1 Methodology evaluation

Table 2 shows in the last column (*Ensemble*) the purity achieved by the proposed methodology for different time series lengths. The previous columns (from 2 to 11) of the table report the purity values obtained by clustering VMs on the basis of per-metric Bhattacharyya distance. The last line of the table presents the average purity values computed over the different time series lengths. For each line, the highest value of clustering purity is emphasized in bold font.

Table 2 Purity for ensemble vs. per-metric clustering

Time series	**Metric**										
length [days]	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	**Ensemble**
180	0.70	**1.00**	**1.00**	0.60	0.95	0.75	0.75	0.95	0.95	0.88	**1.00**
120	0.82	0.90	0.64	0.65	**1.00**	0.63	**1.00**	0.75	0.60	0.60	**1.00**
60	0.82	**1.00**	0.88	0.66	**1.00**	0.56	0.94	0.93	0.68	0.65	**1.00**
40	0.80	0.75	0.80	0.65	0.90	0.60	0.90	0.85	0.65	0.65	**0.95**
30	0.76	0.80	0.82	0.63	0.85	0.62	0.68	0.88	0.63	0.65	**0.91**
20	0.76	0.81	0.79	0.62	0.79	0.58	0.87	0.87	0.62	0.69	**0.89**
15	0.79	0.82	0.79	0.62	0.74	0.62	0.85	0.84	0.62	0.61	**0.87**
10	0.78	0.81	0.78	0.63	0.71	0.58	0.83	0.84	0.58	0.58	**0.86**
5	0.78	0.82	0.77	0.65	0.71	0.57	0.80	0.81	0.56	0.58	**0.86**
4	0.78	0.82	0.76	0.63	0.70	0.57	0.79	0.81	0.56	0.56	**0.86**
3	0.77	0.81	0.72	0.62	0.68	0.56	0.79	0.80	0.54	0.55	**0.85**
2	0.76	0.81	0.68	0.58	0.68	0.55	0.78	0.80	0.53	0.54	**0.85**
1	0.75	0.80	0.68	0.56	0.67	0.55	0.77	0.80	0.52	0.54	**0.84**
Mean	0.77	0.84	0.78	0.62	0.80	0.59	0.83	0.84	0.63	0.63	**0.91**

We note that the performance of the ensemble approach always exceeds or (in few cases) equals the purity achievable by considering per-metric clustering, obtaining a purity that ranges from 1 to 0.84 as the time series length decreases from 180 to 1 days, with a mean purity value of 0.91. It is worth to note that the purity remain rather high even for very short time series of only one day, while other proposals in literature [7] present a performance degradation when the amount of data to describe the VM behavior is reduced. Some of the single metrics achieve quite poor results, such as X_4 and X_6, while other metrics perform better, such as X_2 and X_8. However, none of the metrics reaches a mean purity value close to 0.9, while the ensemble clustering exceeds this value. The reason for the best performance in terms of mean purity value can be found in the stability of the results of the ensemble clustering: the purity of the proposed methodology shows a monotonically decreasing behavior for decreasing time series length, while no other metric presents the same trend. Even the best-performing metrics show negative peaks of purity in correspondence to certain time series lengths, for example 120 and 40 days. To investigate the reason of such negative peaks, we carried out some statistical analysis and we found the presence of multiple local maxima (modes) in the probability distributions of almost every metric; the multi-modal nature of the distributions is likely to hinder the performance of the VM clustering process. The variable behavior showed by the per-metric clustering is undesirable, because it means that per-metric distances are too sensitive to the length of the time series and may not give stable results in describing VMs similarities for clustering purposes. On the other hand, the monotonic decrease of purity achieved by the ensemble clustering is an expected and desirable behavior, which is due to the increasing difficulty to correctly associate VMs to the belonging class when shorter sequences of characterizing measurements are available.

It is worth to analyze in details the results related to the metrics X_2 and X_5 (representing CPU and memory, respectively), which typically are the only resources considered in the state-of-the-art for monitoring and management tasks [4, 35, 16, 31]. From Table 2 we see that the use of these single metrics is not sufficient to successfully capture the VMs behavior for clustering purposes. For these metrics, the clustering results not only are worse than the ensemble results for almost every length of the time series, but they also suffer of excessive sensitivity to this parameter. These results confirm the need of considering together multiple resources to characterize VMs behavior for automatic clustering purposes.

5.2 Ensemble vs. euclidean clustering

Let us now compare the results of the proposed ensemble clustering methodology with the alternative approach based on euclidean clustering. Figure 5 shows the purity of VM clustering as a function of time series length for the ensemble and euclidean approaches.

The euclidean approach shows worse and more variable results with respect to the monotonically decreasing curve of the ensemble clustering, with negative peaks for 120 and 40 days. It is interesting to note that the variable behavior of the euclidean-based curve reflects the instability of the per-metric contributions previously analyzed. For example, we can see from Table 2 that basically all the metrics, included the best-performing ones, present negative peaks corresponding exactly to the time series lengths of 120 and 40 days. On the other hand, the ensemble clustering, which implements a quorum-based mechanism, seems to overcome this issue and stabilize the results over all the time series length.

To better understand why the VM clustering benefits from a quorum-based mechanism, we investigate the relationships between the per-metric clustering solutions.

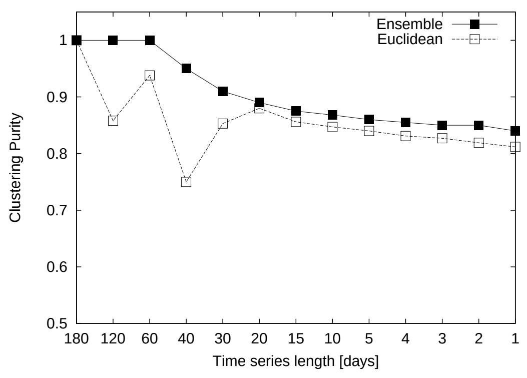


Fig. 5 Purity of Ensemble vs. Euclidean clustering

Specifically, we want to evaluate the capability of clustering based on different single metrics to identify sets of correctly clustered VMs that are not completely overlapped. To this aim, we exploit the *Jaccard Index*, also known as the Jaccard similarity coefficient: this index measures the ratio between the size of the intersection and the size of the union of two sample sets, and it is typically used to compare the sets similarity. For each metric $m \in [1, M]$, we consider the per-metric clustering solution $\mathbf{C}_m$. From $\mathbf{C}_m$ we extract the sub-vector $\mathbf{C}_m^{'}$ including only the elements that represent VMs assigned to the correct cluster identifier according to the ground truth vector $\mathbf{C}^*$. We consider the Jaccard index $J(m_1, m_2)$ between two metrics m_1 and m_2 as:

$$J(m_1, m_2) = \frac{|\mathbf{C}_{m_1}^{'} \cap \mathbf{C}_{m_2}^{'}|}{|\mathbf{C}_{m_1}^{'} \cup \mathbf{C}_{m_2}^{'}|}$$

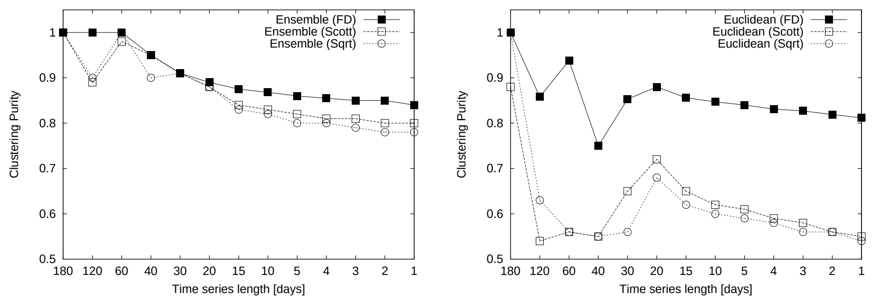
The Jaccard index may take values in the range $[0, 1]$: a value of $J(m_1, m_2)$ equal to 0 means that the clustering solutions for metrics m_1 and m_2 have no correct VM assignments in common, while a value of 1 means that the two per-metric clustering solutions correctly cluster exactly the same set of VMs. Table 3 shows some statistical properties of the Jaccard Index computed over all the possible pairs of the M metrics for time series lengths ranging from 1 to 180 days.

Table 3 Similarity of clustering solutions for per-metric distance

Jaccard index	**Time series length [days]**						
	180	**60**	**30**	**15**	**5**	**3**	**1**
Mean	0.68	0.62	0.62	0.63	0.55	0.58	0.58
Std. dev.	0.19	0.20	0.16	0.19	0.20	0.18	0.15

The mean value of the Jaccard index is between 0.55 and 0.68 for every time series length, meaning that each pair of metrics has, on average, a common intersection of correctly clustered VMs that is between 55% and 68% of the correct solutions identified by each single metric. Furthermore, we see that the standard deviation never exceeds 0.20 for every time series length. These results show that the per-metric clustering correct solutions present a common intersection, but this intersection is almost never complete, being limited to 60% on average. This means that each metric is able to capture similarities among VMs behavior that are different from the similarities detected by other metrics. In other words, each metric gives its peculiar contribution to the quorum-based mechanism which is at the base of the ensemble clustering, thus contributing to obtain stable performance not achievable by relying on per-metric clustering or summing up single metric distances, as in the euclidean approach.

Let us now perform a further comparison between ensemble and euclidean clustering approaches: we evaluate the sensitivity of the two approaches to the rule used to define the number of bins in the generation of metric histograms. For the previous experiments, we used the Freedman-Diaconis (FD) rule to generate the histograms of the metric distributions. In this experiment, we compare the purity of ensemble and euclidean clustering approaches when three different rules are used for the histogram generation: Freedman-Diaconis (FD), Scott and Square Root. Figure 6 shows the results as a function of the time series length.



(a) Ensemble clustering

(b) Euclidean clustering

Fig. 6 Impact of different rules to compute histogram bins number

It appears very clearly from the figure that the rule choice has a completely different impact on the ensemble and euclidean clusterings. The proposed ensemble approach (Figure 6(a)) achieves very similar results for every rule: the use of the FD rule leads to slightly better purity, but the curves are all very close for almost every time series length. On the contrary, the euclidean clustering (Figure 6(b)) appears to be extremely sensitive to the choice of the rule to compute histograms. We should consider that Scott and Square Root follows very different criteria to compute the histogram characteristics: Square Root determines the number of histogram bins only on the basis of the number of measurements in the time series, while Scott exploits the characteristics of the metric distribution. However, in both cases we can see that the achieved purity decreases significantly whit respect to the FD rule, dropping below 0.6 for several time series lengths.

These results show that the performance of the euclidean clustering is very unstable not only, as previously shown, with respect to the time series length, but also to the choice of the rule for generating histograms. On the other hand, the stability of the ensemble clustering represents an important result for the applicability of the proposed methodology for VM clustering in cloud computing environments. Since there is no a generally accepted “best” rule to determine the histogram bin number for a given distribution, the stability with respect to the rule choice represents a fundamental feature for any automated approach for VM clustering.

5.3 Sensitivity to histogram bin number

In this last experiment, we aim to provide a sensitivity analysis evaluating the stability of the ensemble clustering performance with respect to the histogram characteristics. Specifically, we evaluate the clustering purity for different values of histogram bin number.

We take as a reference points Freedman-Diaconis and Scott rules, because they share a desirable feature for the automated generation of metric histograms: the number of bins is computed based on the characteristics of metric distribution, but it is independent of the time series length, oppositely to the Square Root approach. We also consider that the number of bins generated according to the Freedman-Diaconis rule tends to be significantly higher (1 or 2 orders of magnitude) with respect to the Scott case. For these reasons, we choose to consider an interval of bin numbers which includes the gap between the two rules, and also explores lower and higher numbers of bins.

Since the bin numbers generated by Freedman-Diaconis and Scott rules may differ of orders of magnitude depending on the specific metric, we use a logarithmic scale to fix the number of bins to consider for the sensitivity analysis. For each metric $m \in [1, M]$, B_m^{FD} and B_m^{Scott} are the number of bins according to Freedman-Diaconis and Scott rules, respectively. We choose to evaluate two intermediate points between the two rules; hence, we use the cube root of the ratio between B_m^{FD} and B_m^{Scott} as the step Π_m of the sequence of bin numbers to evaluate:

$$\Pi_m = \left(\frac{B_m^{FD}}{B_m^{Scott}}\right)^{\frac{1}{3}}, \forall m \in [1, M]$$

We consider the set of vectors $\mathbf{B}^i$, $i \in [l, U]$, where each vector $\mathbf{B}^i$ represents the sequence of bin numbers to be used for each metric and is defined as follows:

$$\mathbf{B}^i = \{B_m^i : B_m^{Scott} \times (\Pi_m)^i, m \in [1, M]\} \forall i \in [l, U]$$

The values of l and U define the lower and upper bounds for the interval of bin numbers to evaluate. Since we want to exceed the gap between Freedman-Diaconis and Scott rules, we consider $l = -2$ and $U = 4$. It is worth to note that $B_m^0 = B_m^{Scott}$ and $B_m^3 = B_m^{FD}$ for every metric m.

Figure 7 shows the clustering purity as a function of different time series lengths (x axis) and histogram bin numbers (y axis). We observe that for every bin number higher that $\mathbf{B}^0$ (corresponding to Scott rule) the achieved purity presents similar behavior, reaching values equal to 1 for the longest time series and decreasing monotonically as the time series length goes down to 1 day, with a clustering purity which always remains above 0.82 even for the shorter time series. On the other hand, the clustering performance significantly decreases for low number of bins ($\mathbf{B}^{-1}$ and $\mathbf{B}^{-2}$). Specifically, it is evident that for bin numbers lower than $\mathbf{B}^{0}$, the performance decreases with the number of bins, ranging from 0.67 to 0.97 for $\mathbf{B}^{-1}$ and from 0.54 to 0.95 for $\mathbf{B}^{-2}$. The reason for these worse results is related to a too low number of bins leading to coarse-grained histograms, which are not able to exhaustively capture the behavior of the VMs.

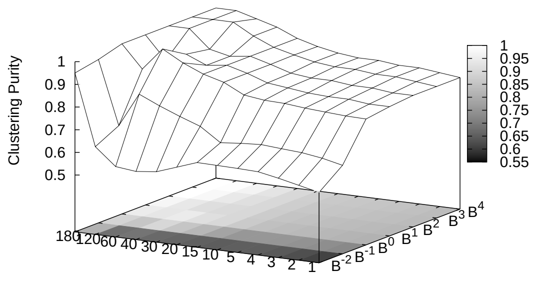


Fig. 7 Clustering purity for different time series lengths and histogram bin numbers

This results strengthens the characteristic of stability of the ensemble clustering with respect to both the length of the metric time series and the histogram characteristics. Furthermore, an interesting insight on the suitable way to determine the number of bins of the histograms can be deduced from the observed results. This experiment suggests that the Freedman-Diaconis rule is the better choice for the generation of the quantitative VM description. Indeed, the use of the Freedman-Diaconis rule leads to the highest and most stable clustering purity for every time series length, as shown in Figure 7. Moreover, using higher number of bins is not desirable because it could easily increase the computational costs of the Bhattacharyya distances without adding any gain in the clustering performance. On the other hand, significantly decreasing the number of bins may cause worse results: in this sense, we can consider the bin numbers determined by the Scott rule as a sort of low boundary under which the clustering performance is likely to significantly decrease.

5.4 Sensitivity to metric selection

In this experiment we aim to evaluate how the methodology performance varies when the number of metrics considered for clustering decreases. To this aim, we consider the purity obtained by clustering VMs on the basis of per-metric Bhattacharyya distance, which are reported in Table 2, Section 5.1. In Table 4 we sort the metrics in decreasing order of per-metric clustering purity, averaged over all the time series lengths (1 to 180 days). We note that the four best performing metrics (in bold in the table) are related to CPU, memory and I/O resources, which are the main resources typically considered in the state-of-the-art to characterize VM behavior [4, 35, 16, 31].

Table 4 VM metrics sorted by decreasing clustering purity

Metric		Mean purity
X_2	**CPU**	**0.84**
X_8	**OutPktRate**	**0.84**
X_7	**InPktRate**	**0.83**
X_5	**Memory**	**0.80**
X_3	DiskAvl	0.78
X_1	SysCallRate	0.77
X_9	AliveProc	0.63
X_{10}	ActiveProc	0.63
X_4	CacheMiss	0.62
X_6	PgOutRate	0.59

To investigate the stability of the methodology performance for different sets of metrics, we first apply the clustering ensemble methodology considering the basic set composed of these four metrics (X_2, X_8, X_7, X_5); then, we carry out the clustering ensemble by adding one-by one the remaining metrics as listed in Table 4, that is in decreasing order of achieved per-metric clustering purity. The results of this experiment carried out for short time series lengths (from 1 to 4 days) are reported in Fig. 8, where boxes and error bars respectively represent mean and standard deviation of the clustering purity achieved for different sets of metrics.

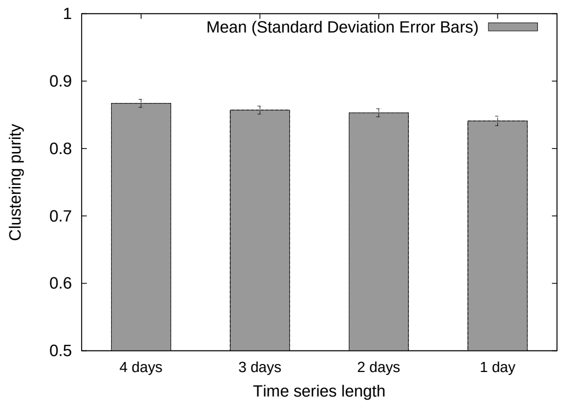


Fig. 8 Sensitivity to metric selections

From Fig. 8 we observe that the clustering purity remains surprisingly stable for different sets of metrics for all the considered time series lengths. In particular, the variation of the clustering purity always remains below 1.5%. For deeper understand-ing, we show in Table 5 the disaggregate results for short time series (from 4 to 1 days): the clustering purity is reported for different set of metrics, starting from the basic set of 4 metrics and then adding one metric at the time, as previously described.

Table 5 Clustering purity for different metric selections

Time series	**Number of Metrics**						
length [days]	**4**	**5**	**6**	**7**	**8**	**9**	**10**
4	0.875	0.870	0.868	0.865	0.865	0.862	0.861
3	0.868	0.861	0.859	0.856	0.853	0.852	0.852
2	0.864	0.860	0.858	0.855	0.853	0.851	0.850
1	0.856	0.854	0.851	0.848	0.845	0.842	0.841

As expected, clustering purity monotonically decreases as we pass from the basic set of the four best performing metrics to all the ten metrics initially considered, showing however a decrease of only XXX%. This result is important for the applicability of the proposed methodology because it shows how the clustering performance is not highly dependent on the choice of the metrics, unlike the approach previously presented in [6]: in the case of ensemble clustering, the inclusion of the main resources related to CPU, memory and network I/O in the set of considered metrics should guarantee good performance in VM clustering. At the same time, a broader choice of metrics does not negatively affect the achieved results.

5.5 Sensitivity to VM number

In this last experiment we analyze the sensitivity of the methodology performance and execution time with respect to the number of VMs to cluster. To this aim, for each of the 110 VMs we consider multiple time series with the length of one-day (24 hours) of the four metrics (X_2, X_8, X_7, X_5) which constitute the minimum set considered in the previous example. In this way, we emulate the presence of an increasing number of VMs to cluster, ranging from 110 to 1100. Fig. 9 shows the achieved purity and the clustering time as a function of the number of VMs to cluster.

We observe that the clustering purity remains quite stable for increasing number of VMs, showing a slight decrease from 0.843 to 0.816 as as we increase the VM number by a factor of 10. As regards the execution time for the clustering phase, we should consider that the spectral clustering algorithm is executed multiple times in the proposed ensemble approach: a step of clustering is carried out on each per-metric distance matrix, then a final clustering is performed on the co-occurrence matrix generated by the ensemble. Hence, the spectral clustering is executed $M+1$ times, where M is the number of metrics considered for clustering ($M = 4$ in this experiment). From Fig. 9 we note that the total clustering time linearly increases with the number of VMs, which is consistent with the known computational cost of the spectral clustering algorithm [23]**** and remains acceptable even for large setups in the order of ******. Furthermore, the low frequency of invocation of the clustering ensemble

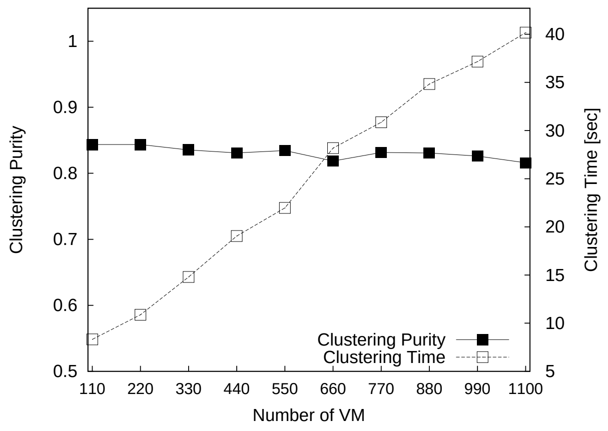


Fig. 9 Clustering purity and execution time for increasing number of VMs

methodology (e.g., once every one or few weeks), confirms that the execution time does not represent an issue for the applicability of the methodology to large cloud data centers.

5.6 Summary of results

The experimental results presented in this section can be summarized as follows:

- The proposed methodology based on ensemble clustering provides high performance, with purity ranging from 0.84 to 1 for every time series length. Furthermore, the performance are stable, with a monotone decreasing pattern with respect to the time series length.
- The ensemble clustering proves to be stable with respect to the parameters used to compute the histograms of the metric distributions, which represent the basic quantitative description of each VM behavior. ****The performance is stable for a wide range of bin numbers and decreases only when the number of bin is too low to effectively describe the VM behavior.****
- The achieved purity is independent of the set of metrics considered for clustering, provided that the set includes the main resources considered in the state of the art, that are CPU, memory and I/O packets.
- The sensitivity analysis with respect to the number of VMs to cluster shows that the achieved purity remains stable even when the VM number is increased by a factor of 10. At the same time, the clustering time has a linear dependence on the VM number, consistently with the computational cost of the spectral clustering algorithm.

6 Related Work

The research activities related to the scalability issues in cloud data centers concern two main topics that are strictly correlated: resource management and infrastructure monitoring.

Many existing studies propose resource management strategies based on the usage of one or few resources compared against thresholds. For example, the studies in [4] and [16] propose solutions for consolidation of virtual machines based on adaptive thresholds regarding the CPU utilization values. Wood *et al.* [38] propose a reactive, rule-based approach for virtual machine migration that defines threshold levels regarding the usage of few specific physical server resources, such as CPU-demand, memory allocation, and network bandwidth usage. Kusic *et al.* [21] address the issue of virtual machine consolidation through a sequential optimization approach; the drawback is that the proposed model requires simulation-based learning and the execution time grows very fast even with a limited number of nodes. All these studies perform a per-node analysis based on the usage of one or few resources; however, these approaches are likely to suffer from scalability issues in large scale distributed systems, such as IaaS cloud computing data centers.

Few recent studies aim to reduce the dimensionality of the resource management problem, such as [31], [30], [34]. The studies in [31], [30] exploit a statistical analysis based on Singular Value Decomposition (SVD) to predict the workload demand aggregated on different virtual machines to anticipate overload conditions on physical servers and trigger virtual machine migrations. Tan *et al.* [34] apply Principal Component Analysis (PCA) to evaluate resource usage patterns across different nodes. The proposal consists in placing on the same physical server virtual machines with negatively correlated resource patterns to reduce the usage variability on the servers. However, all these studies have a different goal with respect to our paper, because they address the specific problem of virtual machine consolidation in cloud data centers. Moreover, all their solutions consider only one resource, that is the CPU utilization of virtual machines, while we aim to support management strategies that consider multiple resources, from CPU to network and disks.

***JCOMSS -¿ stabilità dei risultati rispetto a scelta metriche *** In a preliminary work [7,6], we exploit the correlation between the resource usage of virtual machines to cluster them depending on their resource demand behavior. The methodology presented in [7,6] suffers from some drawbacks: the clustering performance decrease rapidly for short time series of resource usage as well as in presence of periods of time, even short, where the virtual machines are idle. On the other hand, the approach proposed in this paper allows us to overcome these issues thanks to the use of spectral clustering technique [14], [27] and of the distance of Bhattacharyya [10] as the metric to determine the similarity between virtual machines. The distance of Bhattacharyya has been widely used in the context of image processing [25], and has been proposed in the context of characterization of virtual machine behavior in [8]. However, this paper represents a clear step ahead with respect to the previous proposal as we introduce the clustering ensemble, which provides a clear gain for the stability of the custering performance.

As regards the issue of monitoring large data centers, current solutions typically exploit frameworks for periodic collection of system status indicators. Solutions such as Cacti[4] and Munin[5] are more oriented towards the periodic collection of data. Cacti is an aggregator of data transferred through the SNMP protocol, while Munin is a monitoring system based on a proprietary local agent interacting with a central data collector. Both these solutions are typically oriented to medium to small data centers because of their centralized architecture that limits the overall scalability of the data collection process. A more scalable monitoring solution is provided by Ganglia[6], which supports a hierarchical architecture of data aggregators that can improve the scalability of data collection and monitoring process. As a result, Ganglia is widely used to monitor large data centers [11], [26], even in cloud infrastructures [36], by storing the behavior of nodes and virtual machines by organizing the data in time series. Another solution for scalable monitoring is proposed in [2], where data analysis based on the map-reduce paradigm is distributed over the levels of a hierarchical architecture to allow only the most significant information to be processed at the root nodes. However, all these solutions share the same limitation of considering each monitored object (being it a VM or a host) independent from the others. This approach fails to take advantage from the similarities of objects sharing the same behavior. On the other hand, a class-based monitoring system may perform a fine-grained monitoring for only a subset of objects that are representative of a class, while other members of the same class can be monitored at a much more coarse-grained level. We believe that integrating our solution into existing hierarchical models for monitoring can significantly improve the scalability of monitoring operations.

7 Conclusions

Modern data centers supporting IaaS cloud computing represent a major challenge for the monitoring and management of resources, mainly due to scalability issues affecting such large-scale cloud infrastructures.

We propose a methodology for automatically clustering VMs into classes that share similar behavior in order to improve the scalability of monitoring and management tasks. The methodology exploits the Bhattacharyya distance to measure the similarity between the probability distributions of the resources usage and determine the distance between different VMs. Furthermore, we exploit clustering ensemble techniques to merge information about multiple VM metrics and improve the stability of the clustering performance.

The application of the proposed methodology to a real data center hosting multi-tier Web applications shows that the accuracy of VMs clustering ranges between 100% and 84% for every considered scenario and can reduce the amount of data collected by a factor of 15 with respect to a traditional monitoring approach. Furthermore, we demonstrate that the clustering ensemble provides performance that is almost insensitive to the choice of the number of bins in the histograms used to

[4] http://www.cacti.net

[5] http://munin-monitoring.org/

[6] http://ganglia.sourceforge.net/

compute the Bhattacharyya distance, while alternative clustering approaches based on such distance may be extremely sensitive to this parameter.

References

1. Amigó, E., Gonzalo, J., Artiles, J., Verdejo, F.: A Comparison of Extrinsic Clustering Evaluation Metrics Based on Formal Constraints. Journal of Information Retrieval **12**(4), 461–486 (2009)
2. Andreolini, M., Colajanni, M., Tosi, S.: A software architecture for the analysis of large sets of data streams in cloud infrastructures. In: Proc. of the 11th IEEE International Conference on Computer and Information Technology (IEEE CIT 2011). Cyprus (2011)
3. Ardagna, D., Panicucci, B., Trubian, M., Zhang, L.: Energy-Aware Autonomic Resource Allocation in Multitier Virtualized Environments. IEEE Transactions on Services Computing **5**(1), 2 –19 (2012)
4. Beloglazov, A., Buyya, R.: Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centers. In: Proc. of (MGC'10). Bangalore, India (2010)
5. Bhattacharyya, A.: On a measure of divergence between two statistical populations defined by their probability distributions. Bulletin of the Calcutta Mathematical Society **35**, 99–109 (1943)
6. Canali, C., Lancellotti, R.: Automated Clustering of Virtual Machines based on Correlation of Resource Usage. Communications Software and Systems **8**(4) (2012)
7. Canali, C., Lancellotti, R.: Automated Clustering of VMs for Scalable Cloud Monitoring and Management. In: Proc. of 20th International Conference on Software, Telecommunications and Computer Networks (SOFTCOM'12). Split, Croatia (2012)
8. Canali, C., Lancellotti, R.: Automatic clustering of VM based on Bhattacharyya distance. Tech. Rep. DIEF-10.3-2012, Department of Engineering "Enzo Ferrari" – University of Modena and Reggio Emilia (2012). Submitted for publication – http://weblab.ing.unimo.it/papers/TR-DIEF-10.3-2012.pdf
9. Castro, M., Liskov, B.: Practical Byzantine Fault Tolerance. In: OSDI, pp. 173–186 (1999)
10. Choi, E., Lee, C.: Feature extraction based on the Bhattacharyya distance. Pattern Recognition **36**(8), 1703 – 1709 (2003)
11. Chung, W.C., Chang, R.S.: A new mechanism for resource monitoring in Grid computing. Future Generation Computer Systems **25**(1), 1 – 7 (2009)
12. Dhillon, I.S., Guan, Y., Kulis, B.: Kernel k-means: spectral clustering and normalized cuts. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '04, pp. 551–556. ACM, New York, NY, USA (2004). DOI 10.1145/1014052.1014118. URL http://doi.acm.org/10.1145/1014052.1014118
13. Durkee, D.: Why cloud computing will never be free. Queue **8**(4), 20:20–20:29 (2010)
14. Filippone, M., Camastra, F., Masulli, F., Rovetta, S.: A survey of kernel and spectral methods for clustering. Pattern Recognition **41**(1), 176 – 190 (2008)
15. Freedman, D., Diaconis, P.: On the histogram as a density estimator:L2 theory. Probability Theory and Related Fields **57**(4), 453–476 (1981)
16. Gmach, D., Rolia, J., Cherkasova, L., Kemper, A.: Resource pool management: Reactive versus proactive or let's be friends. Computer Networks **53**(17) (2009)
17. Gong, Z., Gu, X.: PAC: Pattern-driven Application Consolidation for Efficient Cloud Computing. In: Proc. of IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS'10). Miami Beach, Florida (2010)
18. Gullo, F., Tagarelli, A., Greco, S.: Diversity-based Weighting Schemes for Clustering Ensembles. In: Proc. of the 9th SIAM International Conference on Data Mining (SDM'09). Sparks, Nevada, USA (2009)
19. Jain, A.K.: Data clustering: 50 years beyond K-means. Pattern Recognition Letters **31**(8), 651 – 666 (2010)
20. Karatzoglou, A., Smola, A., Hornik, K., Zeileis, A.: kernlab - An S4 package for kernel methods in R. Tech. Rep. 9, WU Vienna University of Economics and Business (2004)
21. Kusic, D., Kephart, J.O., Hanson, J.E., Kandasamy, N., Jiang, G.: Power and Performance Management of Virtualized Computing Environment via Lookahead. Cluster Computing **12**(1), 1–15 (2009)
22. Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing **17**(4), 395–416 (2007). DOI 10.1007/s11222-007-9033-z. URL http://dx.doi.org/10.1007/s11222-007-9033-z

23. Manning, C.D., Raghavan, P., Schtze, H.: Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA (2008)
24. Meng, X., Pappas, V., Zhang, L.: Improving the scalability of data center networks with traffic-aware virtual machine placement. In: Proceedings of the 29th Conference on Information Communications, INFOCOM'10. San Diego, California, USA (2010)
25. Michailovich, O., Rathi, Y., Tannenbaum, A.: Image Segmentation Using Active Contours Driven by the Bhattacharyya Gradient Flow. IEEE Transactions on Image Processing **16**(11), 2787–2801 (2007)
26. Naeem, A.N., Ramadass, S., Yong, C.: Controlling Scale Sensor Networks Data Quality in the Ganglia Grid Monitoring Tool. Communication and Computer **7**(11), 18–26 (2010)
27. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, pp. 849–856. MIT Press (2001)
28. Sanguinetti, G., Laidler, J., Lawrence, N.: Automatic determination of the number of clusters using spectral algorithms. In: Machine Learning for Signal Processing, 2005 IEEE Workshop on, pp. 55 –60 (2005). DOI 10.1109/MLSP.2005.1532874
29. Scott, D.W.: On Optimal and Data-Based Histograms. Biometrika **66**(3), 605–610 (1979)
30. Setzer, T., Stage, A.: Decision support for virtual machine reassignments in enterprise data centers. In: Proc. of IEEE/IFIP Network Operations and Management Symposium Workshops (NOMS'10). Osaka, Japan (2010)
31. Setzer, T., Stage, A.: Filtering multivariate workload non-conformance in shared IT-infrastructures. In: Proc. of IFIP/IEEE International Symposium on Integrated Network Management (IM'11). Dublin, Ireland (2011)
32. Singh, R., Shenoy, P.J., Natu, M., Sadaphal, V.P., Vin, H.M.: Predico: A System for What-if Analysis in Complex Data Center Applications. In: Proc. of 12th International Middleware Conference. Lisbon, Portugal (2011)
33. Strehl, A., Ghosh, J.: Cluster ensembles — a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research **3**, 583–617 (2003)
34. Tan, J., Dube, P., Meng, X., Zhang, L.: Exploiting Resource Usage Patterns for Better Utilization Prediction. In: Proc. of the 31st International Conference on Distributed Computing Systems Workshops (ICDCSW'11). Minneapolis, USA (2011)
35. Tang, C., Steinder, M., Spreitzer, M., Pacifici, G.: A scalable application placement controller for enterprise data centers. In: Proceedings of the 16th international conference on World Wide Web, WWW'07. Banff, Alberta, Canada (2007)
36. Tu, C.Y., Kuo, W.C., Teng, W.H., Wang, Y.T., Shiau, S.: A Power-Aware Cloud Architecture with Smart Metering. In: Proc. of 39th International Conference on Parallel Processing Workshops (ICPPW'10). San Diego, CA (2010)
37. Wood, T., Shenoy, P., Venkataramani, A., Yousif, M.: Black-box and gray-box strategies for virtual machine migration. In: Proceedings of the 4th USENIX conference on Networked systems design and implementation, NSDI'07. Cambridge, MA (2007)
38. Wood, T., Shenoy, P., Venkataramani, A., Yousif, M.: Black-box and gray-box strategies for virtual machine migration. In: Proc. of the 4th USENIX Conference on Networked systems design and implementation, NSDI'07. Cambridge, MA (2007)