Automatic Virtual Machine Clustering based on Bhattacharyya Distance for Multi-Cloud Systems

Claudia Canali, Riccardo Lancellotti
Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia
{claudia.canali, riccardo.lancellotti}@unimore.it

ABSTRACT

Size and complexity of modern data centers pose scalability issues for the resource monitoring system supporting management operations, such as server consolidation. When we pass from cloud to multi-cloud systems, scalability issues are exacerbated by the need to manage geographically distributed data centers and exchange monitored data across them. While existing solutions typically consider every Virtual Machine (VM) as a black box with independent characteristics, we claim that scalability issues in multi-cloud systems could be addressed by clustering together VMs that show similar behaviors in terms of resource usage. In this paper, we propose an automated methodology to cluster VMs starting from the usage of multiple resources, assuming no knowledge of the services executed on them. This innovative methodology exploits the Bhattacharyya distance to measure the similarity of the probability distributions of VM resources usage, and automatically selects the most relevant resources to consider for the clustering process. The methodology is evaluated through a set of experiments with data from a cloud provider. We show that our proposal achieves high and stable performance in terms of automatic VM clustering. Moreover, we estimate the reduction in the amount of data collected to support system management in the considered scenario, thus showing how the proposed methodology may reduce the monitoring requirements in multi-cloud systems.

Categories and Subject Descriptors

I.5.3 [**Pattern recognition**]: Clustering; D.4.8 [**Operating systems**]: Performance—*Modeling and prediction*; H.3.4 [**Information storage and retrieval**]: System and software—*Distributed systems*

Keywords

Cloud Computing, Virtual Machine clustering, Bhattacharyya Distance, Spectral Clustering

1. INTRODUCTION

In the few last years, the rapid growth in demand for modern applications combined with the shift to the cloud computing paradigm has led to the establishment of large-scale virtualized data centers. The advent of multi-cloud further extends this vision by considering multiple data centers (possibly owned by different entities) cooperating together. Multi-cloud data centers based on Infrastructure as a Service (IaaS) paradigm typically host several customer applications, where each application consists of different software components (e.g., the tiers of a multi-tier Web application). We assume that customer applications can be replicated over multiple cloud data centers, for example to place applications closer to the users or to improve availability. At the level of data centers, each physical server hosts multiple virtual machines (VMs) running different software components with complex and heterogeneous resource demand behavior.

Due to the increasing size and complexity of these infrastructures, the process of monitoring VMs resource usage to support server consolidation in multiple data centers has become extremely challenging. As VMs are considered as independent black boxes, server consolidation requires information about every VM in every data center. To cope with the scalability issues of monitoring multi-cloud data centers, a widely used approach is to limit the data collection to only a few resources (typically only CPU or memory) [5, 15, 20, 25]. However, this approach cannot fully capture the VM behavior and may hinder the effectiveness of the subsequent consolidation tasks.

We claim that the scalability of monitoring in multi-cloud systems may be improved by leveraging the similarity between VM behaviors. In this way, server consolidation considers the VMs not as single objects, but as members of classes of VMs running the same software component (e.g., Web server or DBMS of the same customer application). It is worth to note that we refer to a multi-cloud scenario characterized by *long-term commitments*, that is, we focus on cloud customers that outsource their data centers to multiple cloud provider purchasing VMs for extended periods of time (for example, integrating a private cloud with 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 [12]. Thus, we can assume that customer VMs do not change frequently the software component they are running (e.g., in the order of weeks or months).

The main contribution of this paper is the proposal of an automated methodology to cluster similar VMs in an IaaS multi-cloud system on the basis of their resource usage. Our approach is consistent with the IaaS vision 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. Clustering VMs into classes characterized by similar behavior allows the system to reduce the monitoring for server consolidation

MultiCloud'13, April 22, 2013, Prague, Czech Republic.

to a subset of VMs that are considered representative for the identified classes.

To determine the similarity among VM behavioral patterns, we exploit the Bhattacharyya distance [6], a statistical technique measuring the similarity of discrete probability distributions. Then, a spectral clustering technique is used to group similar VMs into classes. Our methodology takes into account multiple information for clustering, including network and I/O related resources, while current solutions for data center management mainly consider just CPU or memory. A qualifying point of our methodology is the capability to automatically select *which* information are most useful for the clustering process, to avoid considering data that do not carry any meaningful information and may degrade the clustering performance due to the presence of spiky or noisy behaviors. To the best of our knowledge, the automatic clustering of VMs with similar behavior is a new problem, only recently analyzed in [7, 8], where clustering was based on the correlation among resource usage. Our proposal outperforms the solutions in [7, 8] thanks to the use of Bhattacharyya distance and automatic selection of resources for VM clustering.

We apply the proposed methodology to a dataset coming from a cloud provider hosting VMs running Web servers and DBMS. We show that our methodology can achieve high and stable performance in clustering VMs on the basis of the resource usage monitored over different time periods; in particular, the proposed clustering is effective even when the VM resource monitoring covers short periods of time (e.g., few days). Furthermore, we quantify the reduction in the amount of data collected in our scenario for management support, demonstrating the potential benefit for monitoring scalability. Finally, our results prove that blindly feeding every available information into the clustering process does not necessarily increase the clustering performance, demonstrating the advantage of automatically selecting relevant resources.

The remainder of this paper is organized as follows. Section 2 motivates our proposal and describes the reference scenario. Section 3 presents the proposed methodology for VM clustering. Section 4 describes the experimental testbed used to evaluate our methodology, while Section 5 presents the results of the experimental evaluation. Section 6 discusses the related work and Section 7 concludes the paper with some final remarks.

2. MOTIVATION AND REFERENCE SCENARIO

We recall that our reference scenario is a multi-cloud environment characterized by *long-term commitment* between cloud customers and providers. In particular, we can assume that the software components hosted on each VM do not change frequently (they typically remain the same for weeks or months). Another assumption is that the cloud providers of the data centers belonging to the multi-cloud system have an agreement on common monitoring and management strategies for the global infrastructure.

In such a complex scenario, resource management strategies are needed to guarantee an efficient use of the system resources while avoiding overload conditions on physical servers. We consider a management strategy for multi-cloud systems which consists of two mechanisms, as in [16]: (a) a reactive VM relocation that exploits live VM migration when overloaded servers are detected [27]; (b) a periodic global consolidation strategy that places customer VMs on as few physical servers as possible to reduce the infrastructure costs and avoid resource over provisioning [3, 25].

The global consolidation is carried out periodically and aims to produce an optimal (or nearly optimal) VM placement to reduce the number of shared physical servers. Existing consolidation strategies typically try to predict VM workload over a planning period of time (e.g., hours or days) based on resource usage patterns observed on past measurements, that are usually carried out with a fine-grained granularity (e.g., 5-minute intervals). Since consolidation strategies consider each VM as a stand-alone object with independent resource usage patterns, the amount of information about each VM that is collected and exchanged over a geographical network is likely to arise scalability issues for the monitoring system.

The proposed methodology aims to address the monitoring scalability issues by automatically clustering similar VMs. The main goal of the methodology is to cluster together VMs of the same customer application which are running the same software component (e.g., VMs belonging to the same tier of a Web application), and therefore show similar behaviors in terms of resource usage. For each identified classes of similar VMs, few representative VMs are selected. Then, only the representative VMs of each class are monitored with fine-grained granularity to collect information for the global consolidation task, while the resource usage of the other VMs of the same class is assumed to follow the representatives behavior. It is worth to note that more than two representatives (at least three) are 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 [10]. 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 us to reduce the amount of information that is periodically collected by the monitoring system to support global consolidation.

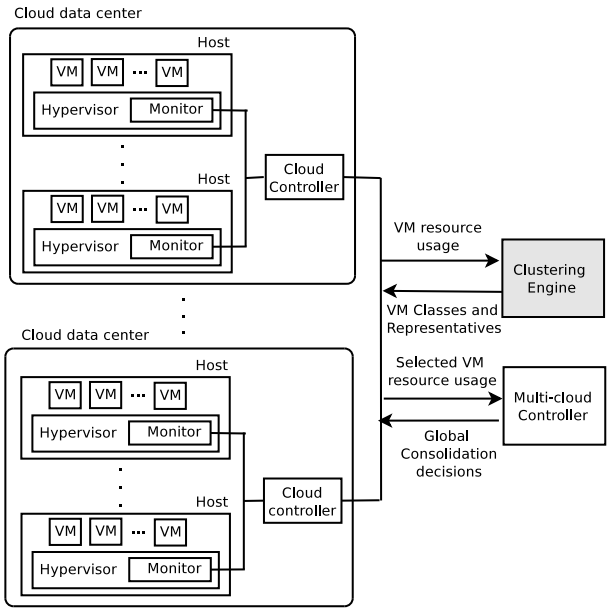

Figure 1: Multi-cloud system

The reference scenario is depicted in Figure 1. The scheme represents a multi-cloud system, where each cloud data center hosts physical servers, namely *hosts*, each running several VMs. A *monitor* process on each host periodically collects samples of the VM resources usage using the *hypervisor* APIs, and sends the collected data to the *cloud controller* of the data center. The cloud controllers

perform several tasks. First, they aggregate VM resource usage time series collected by the monitor processes. Second, they are responsible for taking decisions about live VM migration within the data center [27]. Finally, the cloud controllers communicate (and select) VM data to two centralized components: the *clustering engine*, which runs the proposed methodology to automatically cluster VMs, and the *multi-cloud controller*, which is responsible for global consolidation strategies. We represent the clustering engine and the multi-cloud controller as external to the cloud data centers, but they may be placed indifferently within a data center or in alternative external locations.

Let us now consider the dynamics occurring in the multi-cloud system to support VM clustering and server consolidation. The process of VM clustering starts from the collection of the time series of VM resources usage over a certain period of time. The monitor processes are responsible for this data collection. Then, the cloud controllers of each data center send to the clustering engine the collected VM resource time series. The clustering engine executes the proposed methodology to cluster together VMs belonging to the same customer application and running the same software component. Once the clustering is complete, some representative VMs are selected for each class. The information on VM classes and selected representatives are sent to the cloud controllers in each data center and to the multi-cloud controller for periodic consolidation purposes. The cloud controllers in the data centers selectively collect data about the resources of the representative VMs of each class, then send the data to the multi-cloud controller. This latter component carries out global consolidation strategies using 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 cloud controllers in each data center to be applied. It is worth to note that the process of VM clustering is carried out periodically with a frequency that allows to cope with changes in the VM classes (e.g. 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. However, a precise determination of the activation period or strategy of the clustering process is out of the scope of this paper.

3. METHODOLOGY

In this section, we describe the methodology to automatically cluster similar VMs in a multi-cloud system carried out by the clustering engine component in Figure 1. For each multi-cloud customer application, we aim to group in the same classes VMs which are running the same software components (e.g., VMs belonging to the same tier of a Web application), and consequently show similar resource usage patterns.

The proposed methodology relies on the Bhattacharyya distance to measure the behavior similarity between VMs. This statistical technique allows us to determine the similarity between the discrete probability distributions of the usage samples of the considered VM resources. Then, we exploit a clustering technique based on spectral analysis [17] to group similar VMs into classes. Specifically, the methodology consists of the following steps:

- Extraction of a *quantitative model* to describe VM behavior
- Definition of a *distance matrix* representing the similarities among the VMs
- *Clustering* based on the distance matrix to identify classes of similar VMs

The steps of the methodology are described in detail in the rest of this section.

3.1 VM behavior quantitative model

We now formally define the quantitative model chosen to represent the behavior of VMs, then we discuss some critical design choices 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 each VM 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 VM n. The probability density function $p(\mathbf{X}_m^n)$ of each time series can be considered as the description of the behavior of metric m on VM n. We represent the probability function of finite-length time series through *normalized histograms*. Each normalized histogram consists of a specified number of *bins*, where each bin is associated to an interval of values the metric can take and represents the sample frequency for the interval, that is the fraction of samples in the time series falling within the interval.

If B_m is the number of bins considered for metric m, the histogram of the time series $\mathbf{X}_m^n$ is the set $\mathbf{H}_m^n = \{h_{b,m}^n \forall b \in [1, B_m]\}$, where $h_{b,m}^n$ is the frequency 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 range $[X_m^l(b), X_m^U(b)]$ and $|\mathbf{X}_m^n|$ is the number of samples in the time series. The bin upper and lower bounds 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 a bin for metric m, that is $\Delta x_m = \frac{Xmin_m - Xmax_m}{B_m}$. Figure 2 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 is important for the following step of the methodology because we need to compare same-sized histograms of different VMs.

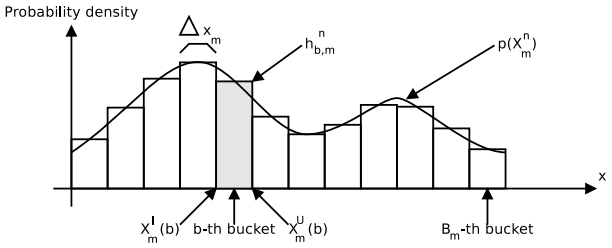

Figure 2: Histogram example

The extraction of the quantitative description of the VM behavior involves some critical design choices which may affect the final outcome of the clustering process. Specifically, we need to address the two following issues:

- Selecting which information is useful to capture the VMs behavior
- Determining the number of the bins used in the normalized histograms

Automatic selection of the metrics is a first critical task that may affect the VM clustering process. A naïve approach of just feeding into the clustering process as much information as possible may be counter-productive, because non-significant data may have an effect comparable to noise and adversely affect the performance of clustering. Furthermore, any human intervention in the information selection process should be avoided because it would hinder the applicability of the methodology to large-scale data centers. Hence, we need to define an automatic process to select relevant metrics. In the management of data centers, CPU and memory are typically considered as representative VM metrics [3, 26], but we claim that they are not sufficient for VM clustering. For metric selection, we rely on two statistical properties of metric time series: the *autocorrelation function* (ACF) and the *coefficient of variation* (CV). We perform a first selection based on the values of the ACF of each time series: a quick decrease of the ACF means that the observed metric exhibits low (or null) autocorrelation. This is the case of metrics characterized by random perturbations and/or spikes varying in time and intensity which may be detrimental for VM clustering purposes [4]. Hence, we choose to retain metrics showing a slow decrease of the ACF, which have a strong dependency among its values. However, a slow decreasing of ACF may be caused by two conditions: (a) the metric is characterized by trends or periodical patterns that are likely to be relevant to describe the VM behavior; (b) the metric values show very low variations during the observation period, thus resulting not relevant for capturing differences in VM behaviors. To eliminate metrics corresponding to the latter condition, we consider the CV of the metric time series. Specifically, a very low CV ($\ll 1$) indicates metrics whose values vary into very small ranges, and which are likely to provide a not meaningful informative contribution for VM clustering.

The second critical issue involved in this step of the methodology is determining the number of histogram bins. The choice is difficult due to the heterogeneity of the metrics considered in monitoring. A number of statistical techniques are popularly used to automatically estimate the number of bins of an histogram, such as Scott and Sturges rules [24]. However, these techniques are optimized for normally distributed data, while we can not assume a normal distribution of samples for metrics describing VM resource usage in cloud data centers. Hence, our choice falls on the Freedman-Diaconis [14] rule, which is a popular technique to determine the number of histogram bins making use of the inter-quartile range of the data to cope with samples without the assumption of a normal distribution.

3.2 Distance matrix

The second step of the methodology consists in building a distance matrix to define similarities among VMs starting from the histogram-based representation of the VM behavior. To build the distance matrix we exploit the Bhattacharyya distance [6], which measures the similarity between two datasets based on their probability distributions. The Bhattacharyya distance $D_m(n_1, n_2)$ computed according to metric m between VMs n_1 and n_2 is defined as:

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

where $h_{b,m}^{n_1}$ is the b-th bin value in the histogram $\mathbf{H}_m^{n_1}$ of metric m for VM n_1, while $h_{b,m}^{n_2}$ refers to VM n_2. 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 3. However, for clustering purposes a single metric is not sufficient, so we need to combine together more metrics. To this aim, we define the multimetric-based distance as the sum of squares of the distances for each metric, that is:

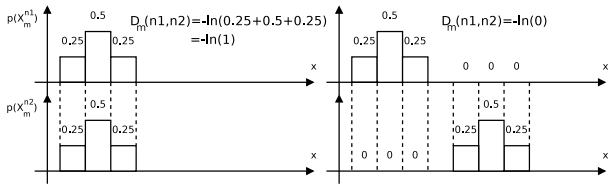

Figure 3: Bhattacharyya distance example

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

where $D_m(n_1, n_2)$ is Bhattacharyya distance between VMs n_1 and n_2 according to metrics m; the boolean variable a_m have value 1 or 0 depending on whether metric m is considered or not.

Finally, we build the distance matrix $\mathbf{D}$ using the distances between every pair of VMs. Due to the nature of the Bhattacharyya distance, the distance matrix $\mathbf{D}$ is positive, symmetric and elements on the main diagonal have value 0.

3.3 Clustering

The final step of the methodology aims to obtain a clustering solution from the distance matrix $\mathbf{D}$. To this aim, we must transform $\mathbf{D}$ into a *similarity* matrix $\mathbf{S}$. This step is carried out by applying a Gaussian kernel operator, that is a common approach to translate distance into similarity [11]: 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 the distance matrix $\mathbf{D}$ and σ is a blurring coefficient of the kernel function. Preliminary analyses on the impact of the σ coefficient on the clustering results suggest that the choice of the parameter is not critical for the performance of the clustering algorithm. We choose $\sigma = 0.6$ that is a default value for Guassian kernels in statistical software [18].

To cluster together elements of a set starting from a similarity matrix, traditional algorithms such as k-means or kernel k-means are not viable options because they expect as input a set of coordinates for each element to cluster. On the other hand, spectral clustering is a widely adopted solution which is explicitly designed to manage as input a similarity matrix or matrix-based representation of graphs [17].

The spectral clustering algorithm computes the Laplacian operator from the input similarity matrix $\mathbf{S}$. 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 [21]. 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, then we compare the ratio between inter-cluster distances (sum of squares of distances between elements belonging to different clusters) and intra-cluster distances (sum of squares of distances between elements of the same cluster). Finally, we select the best solution across multiple k-means runs as the solution that maximize inter-cluster distances and minimize intra-cluster distances. The output of the clustering is a vector $\mathbf{C}$, where the n-th element c^n is the ID of the cluster to which VM n is assigned. Once the clustering is complete, we need to select for each class some representative VMs that will be monitored with fine-grained granularity. To this purpose, it is worth to note that the output of the k-means internal phase of spectral clustering provides

as additional output the coordinates of the centroids for each identified class. In this case, the representative VMs can be selected as the VMs closest to the centroids.

4. EXPERIMENTAL TESTBED

To evaluate the performance of the proposed methodology, we consider a dataset coming from a real cloud data center hosting customer applications. Specifically, we consider 110 VMs belonging to one customer Web application which is hosted on the cloud data center and is deployed according to a multi-tier architecture. Specifically, the 110 VMs are divided in 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 5 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

As a first step of the methodology, we select a reduced subset of metrics based on the statistical properties of the metric time series. We first consider the autocorrelation function (ACF) computed for different values of the time-lag, and we evaluate how fast the ACF decreases as the time-lag increases: we discard metrics having a percentage decrease of ACF greater than 50% for time-lag equal to 1 [4]. Among the remaining metrics, we operate a further selection based on the coefficient of variation (CV): specifically, we discard metrics having a value of CV $\ll 1$, as discussed in Section 3. Then, we compute for each selected metric m of each VM n the normalized histogram $\mathbf{H}_m^n$, that is built on the time series of metric m. We compute the metric histograms according to the Freedman-Diaconis [14] rule (default) and to the Scott rule [24], which will be used as a term of comparison. In the second step of the methodology, the histograms of the selected metrics are used to compute the distance of Bhattacharyya between pairs of VMs, as described in Section 5.1. Then, the distance matrix $\mathbf{D}$ is computed starting from the Bhattacharyya distances, and $\mathbf{D}$ is given as input to the third and last step of the methodology. The last step applies the spectral clustering to the distance matrix $\mathbf{D}$. Since the internal clustering phase of k-means starts each run with a set of randomly-generated cluster centroids, we run the final clustering 10^3 times, then we select the best solution $\mathbf{C}$.

To evaluate the performance of the proposed clustering methodology, we need a measure indicating how many VMs are correctly identified as Web servers and DBMS. To this aim, we consider the clustering *purity*, which is one of the most popular measures for clustering evaluation [1] and represents the fraction of correctly identified VMs. Purity is determined by comparing the output $\mathbf{C}$ of the clustering algorithm with the ground truth vector $\mathbf{C}^*$, which represents the correct classification of VMs into two clusters consisting of Web servers and DBMS servers. Purity is defined as:

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

where $|\{c^n : c^n = c^{*n}, \forall n \in [1, N]\}|$ is the number of VMs correctly clustered and N is the total number of VMs.

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, 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

We now evaluate the performance of the proposed methodology by applying it to the described testbed. In particular, the experimental evaluation aims to investigate the critical design choices involved in the first step of the methodology, which concerns the extraction of the quantitative model to describe the VM behavior. In the rest of this section, we present two experiments, each investigating the impact on clustering performance of a specific aspect of the methodology first step, as described in Section 3: (a) the automated selection of the metrics to exclude useless or noisy information; (b) the choice of the statistical rule for determining the number of the bins of the metric histograms. All experiments evaluate the purity of the VM clustering for different lengths of metric time series, ranging from 5 to 180 days.

5.1 Impact of metric selection

In this first experiment, we aim to evaluate the impact on clustering performance of the automated selection of relevant VM metrics. The automated selection is based on the values of the Autocorrelation Function (ACF) and of the variation coefficient (CV) of each metric, that are reported in Table 2. In particular, third and fourth columns of the table report the percentage decrease of the values of ACF computed with time-lag equal to 1 and 5, respectively. We evidence with colored background the rows corresponding to discarded metrics, emphasizing with bold font the ACF and CV values that determine the elimination from the set of relevant metrics.

We first discard metrics X_4 and X_6 because they show a quick decrease of ACF even for short time-lag (68% and 85% for time-lag equal to 1, respectively), meaning that the metrics are likely to be characterized by random perturbations and/or spikes varying in time and intensity [4]. Then, we pass to consider the CV of metric time series. It is worth to note that metric X_6 has a CV value $\gg 1$, thus confirming that it is characterized by spiky and noisy behavior [9]. On the other hand, metric X_4 is characterized by random

Table 2: Statistical properties of VM metrics

Metric		ACF decrease		CV
		Lag=1	Lag=5	
X_1	SysCallRate	11%	20%	0.87
X_2	CPU	14%	23%	1.09
X_3	DiskAvl	1%	3%	**0.17**
X_4	CacheMiss	**68%**	**81%**	0.60
X_5	Memory	5%	9%	0.54
X_6	PgOutRate	**85%**	**93%**	23.13
X_7	InPktRate	12%	19%	1.29
X_8	OutPktRate	13%	21%	1.22
X_9	AliveProc	1%	3%	**0.07**
X_{10}	ActiveProc	17%	22%	0.67

perturbations that determines its lower variance (CV < 1). As regards the remaining metrics, we discard X_3 and X_9 because they have a CV $\ll 1$, indicating that these metrics show very low variations during the observation period, thus providing a not meaningful informative contribution to differentiate the behavior of VMs belonging to separated classes.

To demonstrate that the selected metrics bring a relevant contribution to VM description, we evaluate the clustering results achievable by computing the Bhattacharyya distance based just on single metrics; specifically, we consider two discarded metrics, X_6 and X_9, and the selected metric X_8. Figure 4 compares the clustering purity achieved for the three considered metrics. From Figure 4 we observe that both metrics X_6 and X_9 lead to poor clustering performance, with a purity that is always below 0.75 and 0.7, respectively. On the other hand, the best results are obtained using the metric X_8, that achieves a purity higher than 0.75 for every time series length. This result confirms our claim that metrics characterized by spiky behavior or by too low variance do not effectively contribute in capturing the VM behavior.

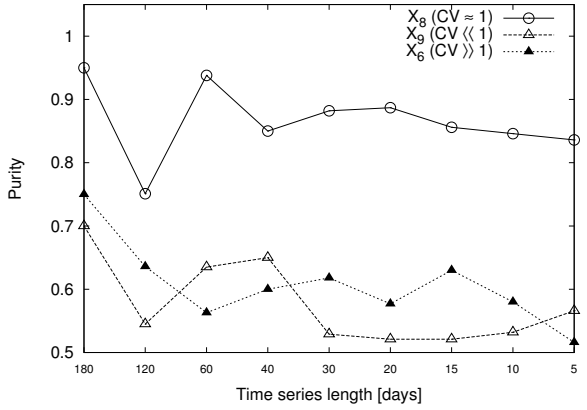

Figure 4: Clustering purity for single metrics

We now consider the set of most relevant metrics determined by the automated selection based on ACF and CV values, namely *Best Set*: SysCallRate (X_1), CPU (X_2), memory (X_5), InPktRate (X_7), OutPktRate (X_8) and ActiveProc (X_10). Figure 5 shows the clustering purity as a function of time series length for different metric combinations: the entire set of 10 metrics, namely *All Met*, the *Best Set* and the selected metric X_8 for comparison term.

We first observe that the use of the *Best Set* leads to two important achievements with respect to the entire set of metrics: better

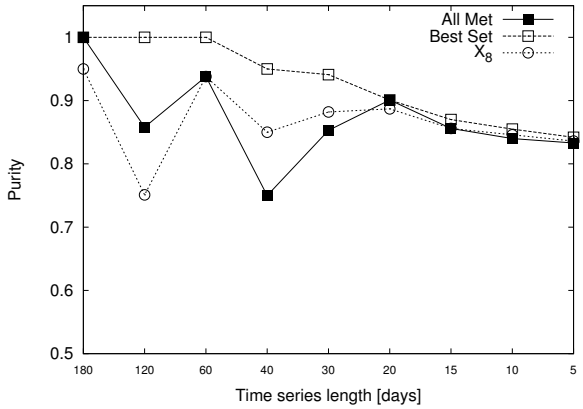

Figure 5: Clustering purity considering different metrics

performance and stability of the clustering results for different time series length. Discarding the metrics based on their ACF and CV values allows us to avoid the negative effect that adds variability to the performance of the *All Met* curve. Such negative impact is evident if we compare the curves referred to metric X_8 and *All Met*: for some time series length (e.g., 40 and 30 days) the single metric achieves better results than the *All Met* curve, whose performance is decreased by the negative effect of metrics which are discarded in the *Best Set*. It is also worth to note that the clustering purity for *All Met* is particularly variable for long time series (left part of the figure). This result is apparently counter-intuitive, because it should be easier for the clustering process to correctly associate VMs to the belonging class when longer sequences of characterizing measurements are available. However, the reason of this behavior can be found in the presence of multiple local maxima (modes) in the distributions of long metric time series. The multi-modal nature of these distributions, that was evidenced by preliminary statistical analysis carried out on metric time series, tends to hinder the performance of the clustering process. However, an appropriate selection of metrics significantly reduces the sensibility of the clustering results to the length of the time series.

Table 3: Clustering purity of Best Set vs. single metrics

Time Series length [d]	Single Metric						Best Set
	X_1	X_2	X_5	X_7	X_8	X_{10}	
180	0.70	**1.00**	0.95	0.75	0.95	**1.00**	**1.00**
120	0.82	0.90	**1.00**	**1.00**	0.75	0.81	**1.00**
60	0.82	**1.00**	**1.00**	0.94	0.93	0.93	**1.00**
40	0.80	0.75	0.90	0.90	0.85	0.90	**0.95**
30	0.76	0.80	0.85	0.68	0.88	0.88	**0.94**
20	0.76	0.81	0.79	0.87	0.88	0.83	**0.90**
15	0.79	0.82	0.74	0.85	0.85	0.77	**0.87**
10	0.78	0.81	0.71	0.83	0.84	0.71	**0.85**
5	0.78	0.82	0.68	0.80	0.83	0.70	**0.84**

Another interesting observation is that the clustering purity of the *Best Set* outperforms the results of each single metric selected as relevant, as clearly shown in Table 3, where the highest value of purity for every time series length is emphasized in bold font. Furthermore, it is worth to note that the purity of the *Best Set* outperforms (by up to 14%) the results of the clustering approach presented in our previous studies [7, 8], where clustering was based on correlation among VM metrics: for example, for 20-days time

series the *Best Set* achieves a purity higher than 0.9, against a value close to 0.8 of the correlation-based approach.

A final observation comes from the comparison between the results of the *Best Set* and those reported in second and third columns of Table 3, representing X_2 (CPU) and X_5 (Memory), respectively: the use of single resources like CPU and memory, that typically are the only metrics considered in cloud data center management, is not sufficient to capture the VMs behavior for clustering purposes. Hence, this experiment confirms the need to consider multiple resources to correctly characterize VM behavior, but also demonstrates the importance of selecting the information to avoid adding useless data which may significantly decrease the clustering performance.

5.2 Impact of histogram characteristics

To support the choice of the Freedman-Diaconis (FD) rule [14] to determine the number of histogram bins, we now evaluate the use of another statistical rule, namely Scott [24]. The Scott rule tends to minimize the integrated mean squared error of the density estimate, so it is widely used for random samples of normally distributed data. Figure 6 shows the values of the clustering purity as a function of time series length for histograms generated through FD and Scott rules. In this experiment we use the selected metrics belonging to the *Best Set* defined in the previous experiment to compute the Bhattacharyya distance.

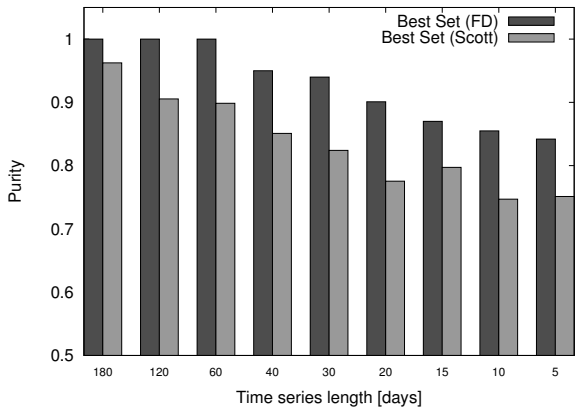

Figure 6: Purity for different time series length and histogram number of bins

As shown in Figure 6, the use of the FD rule allows us to achieve for every time series length a clustering purity that is significantly higher with respect to the case of the Scott rule, with a difference in the achieved purity up to 0.17. We should consider that the Scott rule produces histograms with a much lower number of bins with respect to the FD rule (1 or 2 orders of magnitude, depending on the metric). This explains the worse results of the Scott case, because few and coarse bins are not accurate enough to describe metric distributions and, consequently, VMs behavior.

6. RELATED WORK

Multi-cloud systems have been proposed in the last few years to address the shortcomings of traditional cloud infrastructures by providing large cloud services for customers. Research in this field ranges from the proposal of multi-cloud frameworks and federations of cloud providers [23] up to proposals of development models [2], and solutions for monitoring and management of cloud resources.

The Reservoir project [23] is one of the most significant efforts in the field of frameworks for multi-cloud. The proposed architecture aims to support federation of cloud providers offering a uniform interface explicitly designed to support the deployment of business services. Other frameworks have been proposes and analyzed in the scope of multi-cloud. For example, several frameworks such as Eucalyptus [1], Open Nebula [2], Nimbus [3] support different virtualization technologies, that represent a key feature for a multi-cloud environment.

As regards the issue of monitoring large data centers, current solutions typically exploit frameworks for periodic collection of system status indicators. Most frameworks for multi-cloud monitoring rely on standard building blocks such as Ganglia[4], Cacti [5] or Munin [6]. However, monitoring a large number of VMs in a geographically distributed environment remains a challenge unless some technique is used to reduce of the amount of data to collect and store. Our proposal explicitly aims to address this problem.

Similar scalability issues can be observed for multi-cloud management. Most management solutions are inherited from traditional cloud literature, and can be divided in two categories: reactive on-demand solutions that can be used to avoid and mitigate server overload conditions, and periodic solutions that aim to consolidate VMs exploiting optimization algorithms. The two approaches can be combined together [16]. Examples of reactive solutions are [5] and [15], that propose a mechanism based on adaptive thresholds regarding CPU utilization values. A similar approach is described also in Wood *et al.* [27] with a rule-based approach for live VM migration that defines threshold levels about the usage of few specific physical server resources, such as CPU-demand, memory allocation, and network bandwidth usage. We believe that this type of solution can be integrated in our proposal at the level of cloud controllers. An example of periodic VM consolidation solutions is proposed by Kusic *et al.* in [19], where VM consolidation is achieved through a sequential optimization approach. Similar solutions are proposed in [3, 26]. However, these approaches are likely to suffer from scalability issues in large scale distributed systems due to the amount of information needed by the optimization problem. The scalability issue is exacerbated in the context of multi-cloud infrastructures due to the geographic scale. Solutions like our proposal, aiming to reduce the amount of data to collect and consider for the management of cloud data centers, may play a major role for the applicability of consolidation strategies to multi-cloud systems.

In preliminary studies [7, 8], we exploit the correlation between resource usage to cluster VMs depending on their behavior. However, the methodology presented in [7, 8] suffers from some drawbacks: the clustering performance decreases rapidly for short metric time series as well as in presence of time periods, even short, during which VMs are idle. On the other hand, the approach proposed in this paper overcomes these issues thanks to the use of spectral clustering technique [13, 22] and Bhattacharyya distance [6] to determine the similarity between VMs. Zhang *et al.* [28] propose a method for VM clustering in cloud systems; however, they only consider storage resources in order to perform storage consolidation strategies. On the other hand, our proposal focuses on the

[1] http://www.eucalyptus.com
[2] http://www.opennebula.org
[3] http://www.nimbusproject.org/
[4] http://ganglia.sourceforge.net/
[5] http://www.cacti.net
[6] http://munin-monitoring.org/

more general problem of global server consolidation and takes into account multiple resources to describe VMs behavior.

7. CONCLUSIONS AND FUTURE WORK

We proposed a methodology for automatically clustering VMs into classes sharing similar behavior to improve the scalability of monitoring process supporting management operations in multi-cloud cloud data centers. The methodology exploits the Bhattacharyya distance to determine similarities among different VMs starting from their resource usage. The methodology considers multiple VM metrics and automatically selects which of them actually bring useful information for VMs clustering. The application of the proposed methodology to the data center of a cloud provider 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.

As a future work we plan to validate our methodology with multiple complex scenarios where the data centers host several different applications. Furthermore, we aim to evaluate if the VM clustering can be exploited to improve the scalability not only of the monitoring but also of the server consolidation process.

8. REFERENCES

[1] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo. A Comparison of Extrinsic Clustering Evaluation Metrics Based on Formal Constraints. *Journal of Information Retrieval*, 12(4):461–486, Aug. 2009.

[2] D. Ardagna, E. di Nitto, P. Mohagheghi, et al. MODAClouds: A model-driven approach for the design and execution of applications on multiple clouds. In *Proc. of Workshop on Modeling in Software Engineering (MISE)*, June 2012.

[3] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang. Energy-Aware Autonomic Resource Allocation in Multitier Virtualized Environments. *IEEE Trans. on Services Computing*, 5(1):2 –19, Jan. 2012.

[4] Y. Baryshnikov, E. Coffman, G. Pierre, D. Rubenstein, M. Squillante, and T. Yimwadsana. Predictability of Web-Server Traffic Congestion. In *Proc. of IEEE Workshop on Web Content Caching and Distribution (WCW)*, Sophia Antipolis, France, Sept. 2005.

[5] A. Beloglazov and R. Buyya. Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centers. In *Proc. of MGC Workshop*, Bangalore, India, Dec. 2010.

[6] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35:99–109, 1943.

[7] C. Canali and R. Lancellotti. Automated Clustering of Virtual Machines based on Correlation of Resource Usage. *Communications Software and Systems*, 8(4), Dec. 2012.

[8] C. Canali and R. Lancellotti. Automated Clustering of VMs for Scalable Cloud Monitoring and Management. In *Proc. of Conference on Software, Telecommunications and Computer Networks (SOFTCOM)*, Split, Croatia, Sept. 2012.

[9] S. Casolari, S. Tosi, and F. Lo Presti. An adaptive model for online detection of state changes in Internet-based systems. *Performance Evaluation*, 69(5):206–226, May 2012.

[10] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance. In M. I. Seltzer and P. J. Leach, editors, *OSDI*, pages 173–186. USENIX Association, 1999.

[11] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proc. of International Conference on Knowledge Discovery and Data Mining*, Seattle, USA, Aug. 2004.

[12] D. Durkee. Why cloud computing will never be free. *Queue*, 8(4):20:20–20:29, Apr. 2010.

[13] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176 – 190, Jan. 2008.

[14] D. Freedman and P. Diaconis. On the histogram as a density estimator:L2 theory. *Probability Theory and Related Fields*, 57(4):453–476, Dec. 1981.

[15] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. Resource pool management: Reactive versus proactive or let's be friends. *Computer Networks*, 53(17), Dec. 2009.

[16] Z. Gong and X. Gu. PAC: Pattern-driven Application Consolidation for Efficient Cloud Computing. In *Proc. of Symposium on Modeling, Analysis, Simulation of Computer and Telecommunication Systems*, Miami Beach, Aug. 2010.

[17] A. K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651 – 666, 2010.

[18] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. kernlab - An S4 package for kernel methods in R. Technical Report 9, WU Vienna University of Economics and Business, Aug 2004.

[19] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang. Power and Performance Management of Virtualized Computing Environment via Lookahead. *Cluster Computing*, 12(1):1–15, Mar. 2009.

[20] R. Lancellotti, M. Andreolini, C. Canali, and M. Colajanni. Dynamic Request Management Algorithms for Web-Based Services in Cloud Computing. In *Proc. of IEEE Computer Software and Applications Conference (COMPSAC)*, Munich, Germany, Jul. 2011.

[21] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, Dec. 2007.

[22] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856, 2001.

[23] B. Rochwerger, D. Breitgand, E. Levy, et al. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):4:1–4:11, July 2009.

[24] D. W. Scott. On Optimal and Data-Based Histograms. *Biometrika*, 66(3):605–610, 1979.

[25] T. Setzer and A. Stage. Decision support for virtual machine reassignments in enterprise data centers. In *Proc. of Network Operations and Management Symposium (NOMS'10)*, Osaka, Japan, Apr. 2010.

[26] 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.

[27] 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.

[28] 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'l Conference on Cloud Computing*, Washington, DC USA, July 2011.