Canali C, Lancellotti R. Improving scalability of cloud monitoring through PCA-based clustering of virtual machines. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 29(1): 38–52 Jan. 2014. DOI 10.1007/s11390-013-1410-9

Improving Scalability of Cloud Monitoring Through PCA-Based Clustering of Virtual Machines

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

Department of Information Engineering, University of Modena and Reggio Emilia, 41125 Modena, Italy

E-mail: {claudia.canali, riccardo.lancellotti}@unimore.it

Received February 12, 2013; revised June 13, 2013.

Abstract Cloud computing has recently emerged as a leading paradigm to allow customers to run their applications in virtualized large-scale data centers. Existing solutions for monitoring and management of these infrastructures consider virtual machines (VMs) as independent entities with their own characteristics. However, these approaches suffer from scalability issues due to the increasing number of VMs in modern cloud data centers. We claim that scalability issues can be addressed by leveraging the similarity among VMs behavior in terms of resource usage patterns. 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. The innovative contribution of the proposed methodology is the use of the statistical technique known as principal component analysis (PCA) to automatically select the most relevant information to cluster similar VMs. We apply the methodology to two case studies, a virtualized testbed and a real enterprise data center. In both case studies, the automatic data selection based on PCA allows us to achieve high performance, with a percentage of correctly clustered VMs between 80% and 100% even for short time series (1 day) of monitored data. Furthermore, we estimate the potential reduction in the amount of collected data to demonstrate how our proposal may address the scalability issues related to monitoring and management in cloud computing data centers.

Keywords cloud computing, resource monitoring, principal component analysis, k-means clustering

1 Introduction

In the last few years the widespread adoption of virtualization techniques and service-oriented architectures have led to the popularity of the cloud computing paradigm. In the Infrastructure as a Service (IaaS) perspective, cloud providers allow customers to run their applications in modern virtualized cloud data centers. Customer applications typically consist of different software components (e.g., the tiers of a multi-tier Web application) with complex and heterogeneous resource demand behaviors. In a virtualized cloud data center, multiple independent virtual machines (VMs) are jointly hosted on physical servers, and each VM runs a specific software component of a customer application.

Due to the rapid increase in size and complexity of cloud data centers, the process of monitoring these systems to support resource management (e.g., periodic VMs consolidation) is becoming a challenge from a scalability point of view. We should also consider that providers of IaaS cloud infrastructures do not have direct knowledge of the application logic inside a software component, and can only track OS-level resource utilization on each VM[1-2]. Hence, most monitoring and management strategies consider each VM of a cloud data center as a single object, whose behavior is independent of the other VMs. This approach exacerbates the scalability issues related to the monitoring of cloud data centers due to the amount of data to collect and store when a large number of VMs are considered, each with several resources monitored at high sampling frequency[3].

Existing monitoring solutions for IaaS cloud data centers tend to address scalability issues by reducing the number of VM resources that are taken into account, typically considering only CPU- or memory-related information[4-8]. However, these approaches are likely to suffer important drawbacks, because limiting the monitoring to CPU or memory resources may not be sufficient to efficiently support VMs consolidation strategies that cope with I/O bound or network bound applications.

We claim that the scalability of monitoring in cloud infrastructures may be improved by leveraging the similarity between VMs behavior, considering 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). In particu-

Regular Paper

lar, we refer to a cloud scenario characterized by *long-term commitments*, that is, we focus on customers that outsource their data centers to a cloud provider purchasing VMs for extended periods of time (for example, using the Amazon so-called *reserved instances*①). This scenario is expected and is a significant part of the cloud ecosystem[9]. Hence, we can assume that customer VMs change the software component they are running with a relatively low frequency, in the order of weeks or months. Once we have identified classes of similar VMs, we may select a few representative VMs for each class and carry out fine-grained monitoring only on these representatives, with a major benefit on cloud monitoring scalability.

In this paper we propose an innovative methodology, namely *PCA-based*, to automatically cluster together similar VMs on the basis of their resource usage. To the best of our knowledge, the proposal of techniques for automatic clustering of similar VMs is new, and is only recently analyzed in [10-11]. The main innovation of our proposal is the use of the principal component analysis (PCA) to automatically remove not relevant information from the VM behavior description, providing an improvement in terms of performance and computational costs.

We apply the proposed methodology to two case studies: a dataset coming from an enterprise cloud data center with VMs running Web servers and DBMS, and a dataset originated by a virtualized testbed running e-business applications with a synthetic workload. We demonstrate that our methodology can achieve high performance in automatic clustering even for short time series (1 day) of monitored VM resource usage. Experimental results show that the use of the PCA-based approach allows us to automatically select the most relevant information for the clustering process, thus achieving a twofold improvement with respect to previous studies[10-11]: first, we obtain better and more stable clustering performance, with a percentage of correctly classified VMs that remains between 100% and 80% for every considered scenario; second, we reduce the computational cost of the VM clustering phase. Finally, we quantify the reduction in the amount of data collected for management support in our enterprise cloud case study, demonstrating the potential benefit for monitoring scalability.

The remainder of this paper is organized as follows. Section 2 describes the reference scenario and motivates our proposal, while Section 3 describes the methodology for automatically clustering VMs with similar behavior. Section 4 presents the two case studies and Section 5 describes the results of the methodology evaluation. Finally, Section 6 discusses the related work and Section 7 concludes the paper with some final remarks.

2 Motivation and Reference Scenario

In a complex scenario such as an IaaS cloud system, resource management strategies are needed to guarantee an 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 [12]: 1) a reactive VM relocation that exploits live VM migration when overloaded servers are detected[2]; 2) 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[4,13].

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)[4,13]. Since consolidation strategies usually consider each VM as a stand-alone object with independent resource usage patterns, detailed information has to be collected with high sampling frequency about each VM, thus creating scalability issues for the monitoring system.

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, 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 us to significantly reduce the amount of information collected for periodic consolidation strategies.

The process of VM clustering is carried out periodically with a frequency sufficient to cope with changes in the VM classes. We recall that our reference scenario

① Amazon EC2 Reserved Instances, http://aws.amazon.com/ec2/reserved-instances/, June 2013.

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.

Fig.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 a *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 also have 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[2]. 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 aggregator of each host sends 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 noting 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[14].

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 tasks. 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 manager on each host to be executed.

Let us now provide a motivating example for our proposal showing how the clustering of similar VMs may improve the scalability of the monitoring system. We consider a multi-tier Web application characterized by

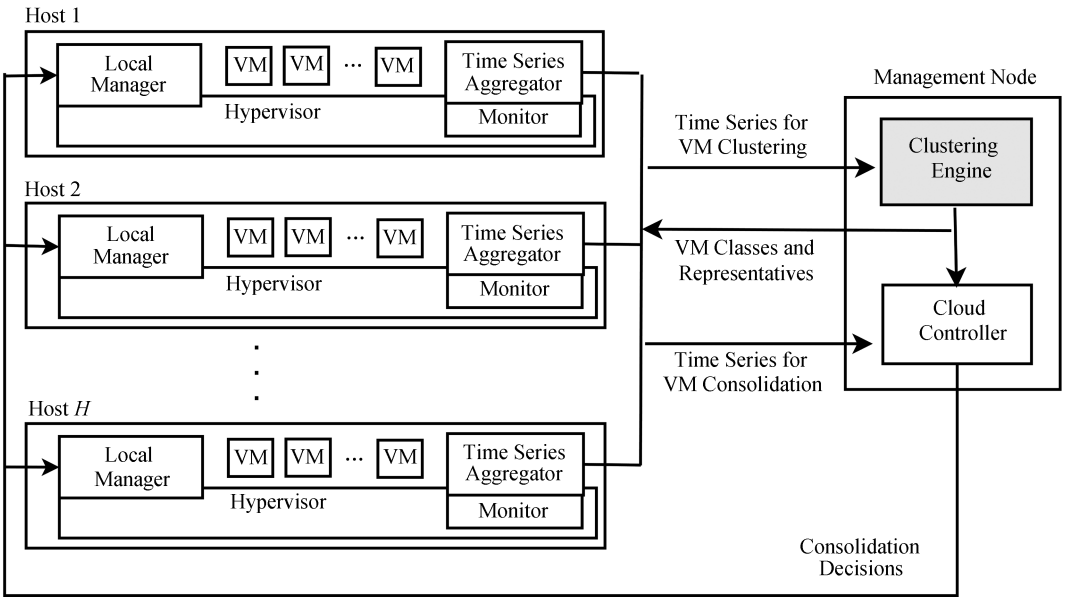

Fig.1. Cloud system.

a high degree of horizontal replication. The application is deployed on 110 VMs, divided in front-end Web servers and back-end DBMS servers. We consider that this application is going to be migrated from an enterprise data center to an IaaS cloud system. This scenario is a typical case where moving to an IaaS platform involves long term commitments, that is the VMs are unlikely to change frequently in typology. As the cloud provider has no knowledge on the software component running in each VM, it is necessary to monitor every VM at fine-grained granularity to accomplish periodic consolidation tasks. Assuming that monitoring considers $\overline{K}$ resources for each VM, which are collected with a frequency of one sample every five minutes, we have to manage a volume of data $288 \times \overline{K}$ samples per day per VM. Considering 110 VMs, the total amount of data is in the order of $32 \times 10^3 \times \overline{K}$ samples per day. The proposed methodology automatically identifies two sets of similar VMs and monitors at the granularity of five minutes only a few representative VMs per class, while the remaining VMs can be monitored with a coarse-grained granularity, for example of one sample every few hours. Assuming to select three representatives for each of the two VM classes the amount of data to collect after clustering is reduced to $1.7 \times 10^3 \times \overline{K}$ samples per day for the class representatives; for the remaining 104 VMs, assuming to collect one sample of the $\overline{K}$ metrics every six hours for VM, the data collected is in the order of $4.2 \times 10^2 \times \overline{K}$ samples per day. Hence, we observe that our proposal may reduce the amount of data collected for periodic consolidation by nearly a factor of 15, from $32 \times 10^3 \times \overline{K}$ to $2.1 \times 10^3 \times \overline{K}$.

3 Methodology for Automatic VMs Clustering

In this section we describe the PCA-based methodology to automatically cluster similar VMs on the basis of their resource usage. We first present the overview of the methodology, then we detail the main steps. Finally, we describe the previous approach to VMs clustering presented in [11], which will be used as a term of comparison in the methodology performance evaluation (Section 5).

3.1 Methodology Overview

To cluster together VMs running the same software component, the proposed methodology has to automatically capture the similarities in VMs behavior. To this aim, we exploit the correlation between the usage of different VM resources. The basic idea is that capturing the inter-dependencies among the usage of several resources, such as CPU utilization, network throughput or I/O rate, allows us to describe the VM behavior during the monitored period of time. For example, network usage in Web servers is typically related to the CPU utilization[15], while in the case of DBMS CPU utilization tends to change together with storage activity[16].

A first naïve approach to clustering VMs based on correlation of resource usage has been proposed in [10-11]. Considering only CPU and memory resources, as typically done by resource management strategies in cloud data centers, leads to poor performance in automatic VMs clustering[10]. However, the high-dimensional dataset consisting of multiple resources usage information may contain data which are not relevant for VM clustering and may have a twofold negative effect on methodology performance. First, clustering algorithms typically have a computational complexity that grows with the size of the input feature vector describing each VM. Second, not relevant information may be detrimental for the clustering performance, because it may introduce elements that hinder the capability of the clustering algorithm to correctly classify similar VMs. For this reason, we need an automatic mechanism to discriminate between relevant and less relevant information in the description of VMs behavior.

We consider variance as a main statistical property that quantifies the relevance of a resource in the overall system behavior, as in [17]. In literature, there are several algorithms to reduce the dimensionality of a high-dimensional and heterogeneous dataset, such as Independent Component Analysis (ICA)[18], Correspondence Analysis[19], Factor Analysis[20]. To automatically select the most relevant information in terms of variance, we exploit the statistical technique of principal component analysis (PCA)[21], because it is able to express the intrinsic structure of a dataset in terms of variance without requiring any prior knowledge about the statistical characteristics of the initial datasets. The use of PCA allows us to identify and retain just the most relevant information, thus reducing the problem dimensionality and improving the performance of the VM clustering. In the rest of this section we describe the main steps of the methodology in details.

3.2 Methodology Steps

Fig.2(a) outlines the main steps of the PCA-based methodology:

- computation of correlation matrices describing the VMs behavior;
- eigenvalue decomposition of the correlation matrices to discriminate between relevant and not relevant information;
- selection of principal components to eliminate scarcely relevant data from the VM description;

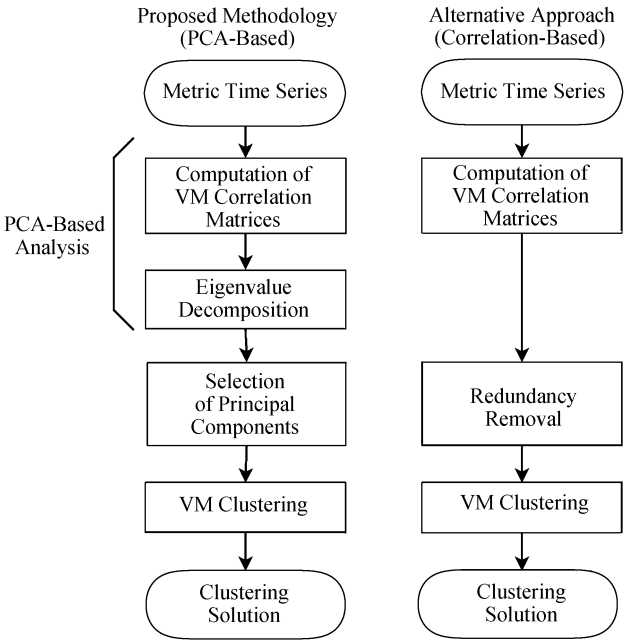

Fig.2. (a) Steps of PCA-based methodology. (b) Alternative correlation-based approach.

• clustering to identify classes of similar VMs.

It is worth noting that the first two steps of the methodology represent the core operations of a principal component analysis over the original time series.

Fig.2(b) describes the alternative approach, namely *correlation-based*, which has been presented in a previous study[11]. In this case, the main difference is that the final clustering step operates directly on the correlation values among the VM metrics, after a simple removal of redundancies in the VM correlation matrices. We now describe in detail each step of the proposed PCA-based methodology and of the alternative correlation-based approach.

3.2.1 Computation of Correlation Matrices

Given a set of N VMs, we consider each VM n, $\forall n \in [1, N]$, as described by a set of M metrics, where each metric $m \in [1, M]$ represents the usage of a VM resource. Let $(\boldsymbol{X}^n_1, \boldsymbol{X}^n_2, \ldots, \boldsymbol{X}^n_M)$ be a set of time series, where $\boldsymbol{X}^n_m$ is the vector consisting of the resource usage samples represented by the metric m of VM n.

Before computing the correlation between metric time series, we should consider the presence of periods where the VMs are in an idle state: during idle periods, the correlation between the metrics is not meaningful to describe the VM behavior and, consequently, may lead to a wrong final clustering, as we will demonstrate in the experimental evaluation (Subsection 5.1.2). Moreover, the presence of idle periods is likely to have the worst effects for short time series, where they may represent a significant portion of the monitored data. To avoid this issue, it could be necessary to apply a filtering technique to eliminate the idle periods from the VM metric time series. Specifically, we consider that a VM is in an idle period when its CPU usage is below a given threshold for a specific time window.

After data filtering, we compute the correlation to capture the inter-dependencies between VM metrics. For each VM n we compute a correlation matrix $\boldsymbol{S}^n$, where $s^n_{m_1,m_2} = corr(\boldsymbol{X}^n_{m_1}, \boldsymbol{X}^n_{m_2})$ is the correlation coefficient of the filtered time series $\boldsymbol{X}^n_{m_1}$ and $\boldsymbol{X}^n_{m_2}$ of metrics m_1 and m_2, respectively. We choose the Pearson product-moment correlation coefficient (PPMCC) to measure the correlation between pairs of time series defined as:

$$s^n_{m_1,m_2} = \frac{\sum_{i=1}^{X}(x^n_{m_1}(i) - \bar{x}^n_{m_1})(x^n_{m_2}(i) - \bar{x}^n_{m_2})}{\sqrt{\sum_{i=1}^{X}(x^n_{m_1}(i) - \bar{x}^n_{m_1})^2}\sqrt{\sum_{i=1}^{X}(x^n_{m_2}(i) - \bar{x}^n_{m_2})^2}},$$

where X is the length of the metric time series ($X = |\boldsymbol{X}^n_m|$, $\forall m \in [1, M]$, $\forall n \in [1, N]$), while $x^n_m(i)$ and $\bar{x}^n_m$ are the i-th element and the average value of the time series $\boldsymbol{X}^n_m$, respectively.

Finally, the correlation matrices are given as input to the second step of the methodology.

3.2.2 Eigenvalue Decomposition

This step applies an eigenvalue decomposition to the VM correlation matrices. This operation results in a PCA coordinate transformation that maps the initial dataset, that is the M time series, on a new coordinate system of M axes, which are called principal components. For each correlation matrix $\boldsymbol{S}^n$ of a VM n, we compute the matrix $\boldsymbol{E}^n$ of eigenvectors as:

$$(\boldsymbol{E}^n)^{-1}\boldsymbol{S}^n\boldsymbol{E}^n = \boldsymbol{D}^n,$$

where $\boldsymbol{D}^n$ is the diagonal matrix of the eigenvalues of $\boldsymbol{S}^n$. Matrix $\boldsymbol{E}^n$ has dimension M x M and contains M column vectors, each of length M, representing the M eigenvectors of the correlation matrix $\boldsymbol{S}^n$. Each eigenvector is associated with a principal component and with an eigenvalue, which represents the magnitude of variance along the corresponding principal component.

3.2.3 Selection of Principal Components

By convention, eigenvalues are sorted from large to small according to their contribution to the overall variance. We exploit these values to discriminate between relevant information, represented by the principal com-

ponents associated with large eigenvalues, from less relevant information, corresponding to the least important components in terms of variance. Several methods exist[21] to choose the number H of principal components to retain, ranging to the Kaiser criterion, that takes into account the eigenvalue related to the components, to graphical approaches as the scree plot, which is based on the percentage of variance expressed by each component. We exploit the scree plot method, which is widely used for its reliability and easy interpretation. It is worth noting that typically we have $H \ll M$, thus allowing us to significantly reduce the dimensionality of the problem.

After selecting the H principal components to retain, we build for each VM n the corresponding feature vector $\boldsymbol{V}^n$, which is given as input to the final clustering step. The feature vector $\boldsymbol{V}^n$ consists of the first H eigenvectors of the matrix $\boldsymbol{E}^n$. Fig.3 provides an example of creation of the feature vector $\boldsymbol{V}^n$ from the eigenvector matrix $\boldsymbol{E}^n$ in the case $H = 2$.

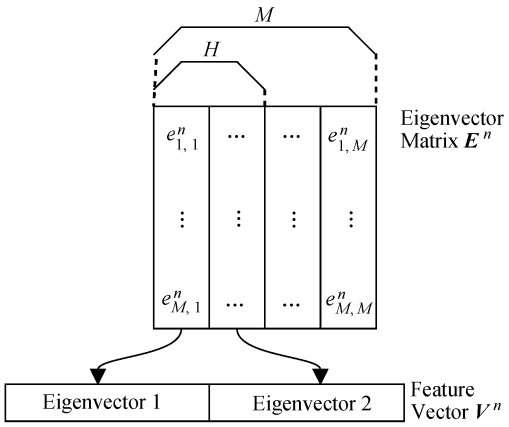

Fig.3. Creation of VM feature vector for the PCA-based approach.

3.2.4 Clustering of Virtual Machines

The feature vector $\boldsymbol{V}^n$ is used by the clustering algorithm as the coordinate of VM n in the feature space. We define $\boldsymbol{C}$ as the vector resulting from the clustering operation. The n-th element of vector $\boldsymbol{C}$, c^n, is the number of the cluster to which VM n is assigned. Many algorithms exist for clustering, starting from the widespread k-means to more complex kernel-based solutions, up to clustering based on spectral analysis[22-23]. We choose to adopt one of the most popular solutions, that is the k-means clustering algorithm[22]. It is worth noting that the execution of the k-means algorithm starts from the selection of a random set of centroids. To ensure that the k-means clustering solution is not affected by local minimums, we iterate the k-means multiple times. Finally, we select as clustering output $\boldsymbol{C}$ the best clustering solution across multiple k-means runs, which is the solution that maximizes inter-cluster distances and minimizes intra-cluster distances[22].

Once the clustering is complete, we select few representative VMs for each class for the purpose of simplifying the monitoring task. Clustering algorithms such as k-means provide as additional output the coordinates of the centroids for each identified class. In our scenario, the representative VMs can be selected as the VMs closest to the centroids.

3.3 Alternative Correlation-Based Approach

Fig.2(b) represents the correlation-based approach for VM clustering proposed in [11]. In this naive approach, the problem dimensionality is reduced simply by removing the redundancy implicit in the correlation matrices. Specifically, the redundancy is caused by the symmetric nature of the correlation matrices, which have the main diagonal consisting of "1" values. We build the feature vector $\boldsymbol{V}^n$ using the elements of the lower triangular sub-matrix; the feature vector is defined as:

$$\boldsymbol{V}^n = (s^n_{2,1}, s^n_{3,1}, s^n_{3,2}, \ldots, s^n_{M,1}, \ldots, s^n_{M,M-1}).$$

Fig.4 provides an example of creation of the feature vector from the correlation matrix in the case $M = 4$.

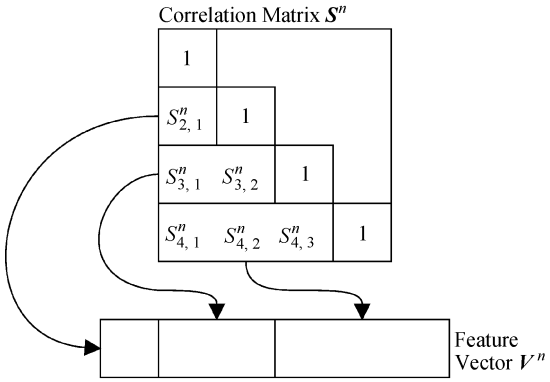

Fig.4. Creation of VM feature vector for the correlation-based approach.

Then, the feature vector $\boldsymbol{V}^n$ is fed into the final clustering step. Also in this case, we use the k-means algorithm for the clustering step, which produces the final solution $\boldsymbol{C}$ of the alternative approach.

4 Case Studies

To evaluate the performance of the proposed methodology we consider two case studies: 1) a dataset coming from a virtualized testbed hosting typical e-business applications with synthetic workload; 2) a real

dataset coming from an enterprise data center hosting Web-based applications. Let us describe the two case studies in details.

4.1 EC2 Amazon Case Study

The first case study, namely EC2 Amazon, is based on a dataset coming from a virtualized testbed running a Web-based e-commerce application. The considered application, based on the TPC-W benchmark[②], is specifically built to evaluate the PCA-based methodology and is deployed over the Amazon Elastic Computing infrastructure. The application uses a java-based application server, a DBMS and a set of emulated browsers, issuing both HTTP and HTTPS requests. The benchmark is hosted on a set of 36 VMs (we use the micro instances of VM provided by Amazon EC2), with 12 VMs dedicated to emulated browsers, 12 to Web servers and 12 to DBMS. The monitoring system periodically collects samples about the VM resource usage. Each sample provides an average value computed over the period between subsequent samplings. In this scenario, the virtualized infrastructure is monitored through a framework explicitly designed for cloud platforms[24]. The complete list of the metrics collected by the monitoring system is provided in Table 1 along with a short description.

Table 1. VM Metrics for EC2 Amazon Case Study

Metric		Description
X_1	BlockOut	Rate of blocks written to storage (Blk/s)
X_2	CtxSwitch	Rate of context switches (Cs/s)
X_3	CPUIdle	CPU idle time (%)
X_4	CPUSystem	CPU utilization (system mode) (%)
X_5	CPUUser	CPU utilization (user mode) (%)
X_6	CPWait	CPU waiting time (%)
X_7	Interrupts	Rate of interrupts (Int/s)
X_8	MemBuff	Size of filesystem in memory (Read/Write access) (MB)
X_9	MemCache	Size of filesystem in memory (Read only access) (MB)
X_{10}	MemFree	Size of free memory (MB)
X_{11}	NetRxBs	Rate of network incoming bytes (B/s)
X_{12}	NetRxPkts	Rate of network incoming packets (pkts/s)
X_{13}	NetTxBs	Rate of network outgoing bytes (B/s)
X_{14}	NetTxPkts	Rate of network outgoing packets (pkts/s)
X_{15}	ProcRun	Number of running processes

As the considered application is supporting a synthetic workload, the patterns of client requests are stable over time without the typical daily patterns that characterize Web traffic. For this reason, we collect samples only for one day: longer time series would not provide additional information from a statistical point of view in this steady state scenario. On the other hand, having a complete control on the monitoring infrastructure allows us to change the sampling frequency for the metrics of each VM. Specifically, we consider sampling frequencies ranging from 1 to 5 minutes.

4.2 Enterprise Data Center Case Study

The second case study, namely Enterprise data center, is based on a dataset coming from an enterprise data center hosting one customer Web-based application for e-health deployed according to a multi-tier architecture. The application is composed of a front-end tier, that hosts the J2EE application implementing the presentation and business logic, and a back-end, that is a set of partitioned and replicated databases on an Oracle DBMS. The application is accessed by a few thousands of users, both private citizens and hospital operators, with the typical daily usage patterns characterized by high resource utilization in the office hours and lower utilization during the night. The data center is composed of 10 nodes on a blade-based system (ProLiant BL460c blades). Each blade is equipped with two 3 GHz quad-core CPUs, and each blade hosts 64 GB of RAM. The data center exploits virtualization to support the Web application. The blades host 110 VMs that are divided between Web servers and back-end servers (that are DBMS).

Data about the resource usage of every VM are collected for different periods of time, ranging from 1 to 120 days. The samples are collected with a frequency of 5 minutes, considering average values over the sampling period. For each VM we consider 11 metrics describing the usage of several resources including CPU, memory, disk, and network. The complete list of the metrics is provided in Table 2 along with a short description.

Table 2. VM Metrics for Enterprise Data Center Case

Metric		Description
X_1	SysCallRate	Rate of system calls (req/s)
X_2	CPU	CPU utilization (%)
X_3	DiskAvl	Available disk space (%)
X_4	CacheMiss	Cache miss (%)
X_5	Memory	Global memory utilization (%)
X_6	UserMem	User-space memory utilization (%)
X_7	SysMem	System-space memory utilization (%)
X_8	PgOutRate	Rate of memory pages swap-out (pages/s)
X_9	InPktRate	Rate of network incoming packets (pkts/s)
X_{10}	OutPktRate	Rate of network outgoing packets (pkts/s)
X_{11}	ActiveProc	Number of active processes

[②]TPC-W, http://www.tpc.org/tpcw/, June 2013.

4.3 Methodology Application

Let us now describe the application of the proposed methodology to the considered case studies. For the EC2 Amazon case study, the final goal of our methodology is to cluster the VMs in three classes: Web servers, DBMS and emulated browsers. We also consider the emulated browsers to increase the number of VMs classes that are taken into account. For the Enterprise data center case study, we aim to cluster the VMs in two classes: Web servers and DBMS servers.

It is worth noting that for the EC2 Amazon case study we do not need to apply a filter to the monitored data, because we have short time series of a single day not containing idle periods. On the other hand, in the Enterprise data center case study we have to apply a preliminary step of data filtering because we have idle periods in the monitored time series. For this scenario, we remove from the original data the sequences of samples where the CPU utilization is below 10%[25-26] for periods of at least four consecutive hours. Then, the methodology computes a correlation matrix for each VM. An eigenvalue decomposition is carried out on the correlation matrices to identify the principal components of the dataset and discriminate between relevant and less relevant information. To reduce the dimensionality of the problem, we select the principal components associated with the highest eigenvalues. To this aim, we exploit the visual method of the scree plot, which is based on the components percentage of variance. We compute and sort the percentages of variance in decreasing order to separate the most important components, characterized by high variance, from the least important ones, characterized by low values. The analysis carried out on both case studies shows that we can consider just the first principal component ($H = 1$), while discarding the other ones. The scree plot for one VM belonging to the first case study is shown in Fig.5 as an example of the variance behavior

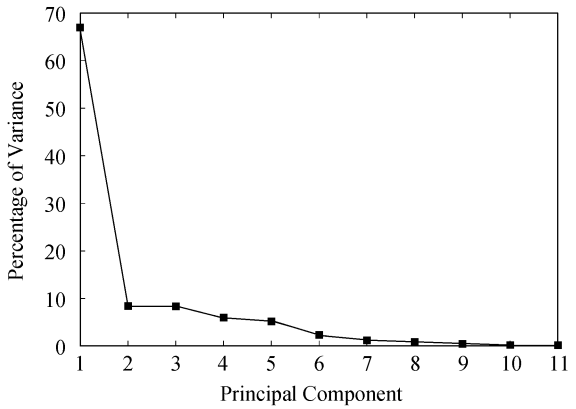

Fig.5. Scree plot.

in our datasets: we see that a sharp "elbow" is clearly visible in the graph. This allows us to select the most relevant information included in the first principal component, which contributes to almost the 70% of variance, while discarding the other irrelevant components, which contribute to the overall variance of the dataset for less than 10% each.

As discussed in Section 3, the first eigenvector corresponding to the selected principal component is used to build the feature vector describing VMs behavior. Then, the feature vectors are given as input to the last step of the methodology, which uses the k-means algorithm to cluster similar VMs. As the k-means algorithm starts each run with a set of randomly-generated cluster centroids, we run the final clustering 1000 times, then we select the best clustering solution. Finally, we compare the output of the clustering step with the ground truth, represented by the correct classification of VMs, to evaluate the clustering results. To evaluate the performance of the methodology, we aim to measure the fraction of VMs which are correctly identified by the clustering. To this purpose, we consider the clustering purity[27], that is one of the most popular measures for clustering evaluation. The clustering purity is obtained by comparing the clustering solution $\boldsymbol{C}$ with the vector $\boldsymbol{C}^*$, which represents the ground truth. Purity is thus defined as:

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

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

5 Experimental Results

In this section we present the results of a set of experiments to evaluate the proposed PCA-based methodology in different scenarios. We first evaluate the performance of the PCA-based methodology applied to the case studies described in Section 4, and compare the results with those obtained through the correlation-based approach. Then, we analyze the computational cost of the methodology for varying number of VMs and considered metrics. Finally, we perform a sensitivity analysis to evaluate how the clustering purity is influenced by the number of selected principal components and by the number of VMs to cluster.

5.1 Performance Evaluation

We now compare the PCA-based and the correlation-based approaches for the two considered case studies.

5.1.1 EC2 Amazon Case Study

The EC2 Amazon case study represents a dataset obtained in a controlled and reproducible environment with a limited number of VMs that are monitored for 24 hours at different sampling frequencies (from 1 to 5 minutes). We exploit this case study for a twofold purpose: to have a first comparison of PCA-based and correlation-based approaches, and to investigate how the frequency of metric sampling may affect the performance of VM clustering. As for the latter goal, we should consider the potential trade-off related to the choice of the metric sampling frequency: a fine-grained monitoring of the considered metric is likely to produce a detailed and precise representation of the VM behavior, which could improve the performance of the clustering; on the other hand, a coarse-grained monitoring reduces the amount of data collected for metric sampling, thus reducing the scalability issues of the monitoring system. Hence, it is interesting to investigate how a different granularity of the resource monitoring affects the performance of the proposed methodology.

Fig.6 shows the clustering purity of PCA-based and correlation-based approaches for different sampling frequencies ranging from 1 to 5 minutes. We observe that the proposed PCA-based methodology correctly identifies a high percentage of VMs (ranging from 88% to 94%), and outperforms the correlation-based approach for every sampling frequency, thus giving us a first confirmation that the automatic selection of the information to feed into the clustering algorithm has a positive effect on the methodology performance.

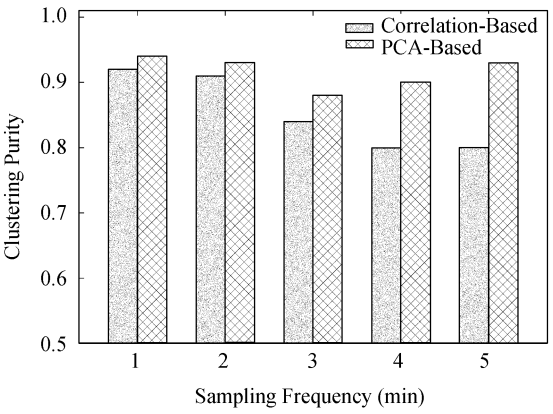

Fig.6. Clustering purity for different sampling frequencies.

If we focus on the impact of the sampling frequency, we note that the difference in the clustering purity grows as the frequency decreases, with a gap ranging from 0.02 up to 0.13 for the lowest considered frequency (5 minutes). We can deduce that the correlation-based approach is significantly more sensible to the granularity of the monitoring sampling, while the performance of the proposed methodology is stable for different sampling frequencies. This represents an important result for the applicability of our methodology to large-scale cloud data centers for a twofold reason: first, the stability of the results allows to avoid high sampling frequencies that would increase the amount of the collected data and worsen the scalability issues of the monitoring systems; second, the methodology is compatible with existing monitoring systems of cloud data centers, which usually exploit a sampling frequency of 5 minutes to monitor VM resources.

5.1.2 Enterprise Data Center Case Study

Now we consider the Enterprise data center case study, which represents a real dataset where VMs are monitored at a sampling frequency of 5 minutes. In this case, we aim to evaluate the clustering performance of the PCA-based and correlation-based approaches as a function of the length of the metric time series. Specifically, we consider time series ranging from 1 to 120 days. First we apply the two clustering approaches without carrying out the preliminary filtering of idle periods on the metric time series.

The histogram in Fig.7 presents the clustering purity for this experiment. We observe that for every time series length the proposed PCA-based methodology equals or exceeds the purity achieved by the correlation-based approach. This result confirms, also in a real scenario, the superiority of the proposed methodology with respect to a naive approach, which directly exploits the correlation values between pairs of metrics for VMs clustering.

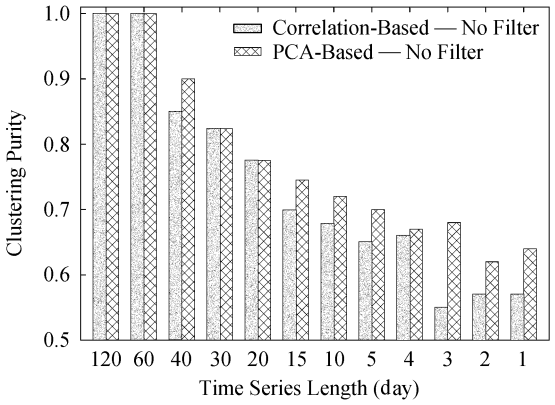

Fig.7. Clustering purity for different time series lengths.

Considering the clustering purity as a function of time series length, we observe that for very long time series the clustering is perfect, that is every Web server and every DBMS are correctly identified. On the other hand, the purity significantly decreases as we reduce

the monitoring period. In particular, when the length of the time series is below 20 days, the purity is below 0.7, reaching 0.65 for a time series of only 5 days. The significant purity reduction for short time series is due to the presence of idle periods in the data. Indeed, we find that some VMs present a bimodal behavior, with periods of time where the VM is mostly idle (CPU utilization below 10%) mixed with periods where the VM is heavily utilized. In particular, for very short time series (1∼5 days) some VMs are characterized almost exclusively by idle periods. During the idle periods, the correlation between the metrics describing the VM is altered, thus leading to a wrong clustering which explains the poor performance of both approaches.

To avoid this effect, we apply the preliminary filtering of the metric time series to extract a sequence of samples not containing idle periods, as described in Section 4. Fig.8 compares the results of PCA-based and correlation-based approaches applied to short time series (from 1 to 15 days) of filtered and not filtered data.

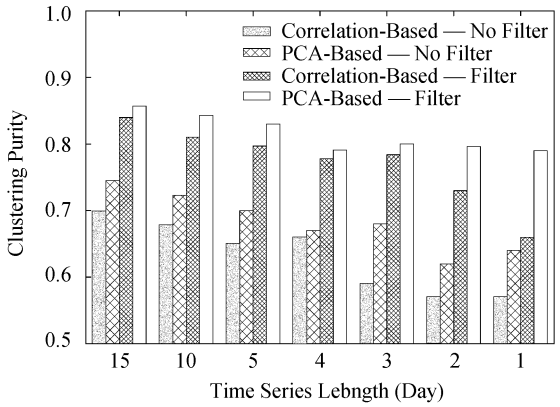

Fig.8. Impact of idle data filtering for short time series.

As expected, filtering idle periods from the monitored data significantly improves the performance of the VM clustering for both approaches. However, the PCA-based methodology is confirmed to outperform the correlation-based approach. The gain achieved by the proposed methodology ranges from 3% to 19% with respect to the purity obtained by the correlation-based approach, reaching higher gains for shorter time series (1 and 2 days). Moreover, the purity never drops below 0.8 even for the shortest considered time series. It is worth noting that the capability of the proposed methodology to achieve good performance for short time series allows us to reduce the period of time during which we need to monitor VM resources before applying clustering.

Fig.9 explains why the PCA-based approach outperforms the alternative. The graph shows the scatter plots of the distances separating each VM from the centroids of the two clusters (Web servers and DBMS) in the case of time series of one day. Web servers are represented by circles, while DBMSes correspond to crosses. The x-axis measures the distance from the centroid of the DBMS cluster (represented by the cross on the y-axis), while on the y-axis we have the distances from the centroid of Web servers cluster (represented by the circle on the x-axis). The distance is computed using the multi-dimensional feature vectors describing the VMs. Fig.9(a) refers to the PCA-based methodology, while Fig.9(b) represents the correlation-based approach. In both graphs we draw the bisector line that identifies points that are equidistant from the two centroids. All the points above the bisector line are classified as belonging to the DBMS cluster, while every point below the line is associated with the Web servers cluster.

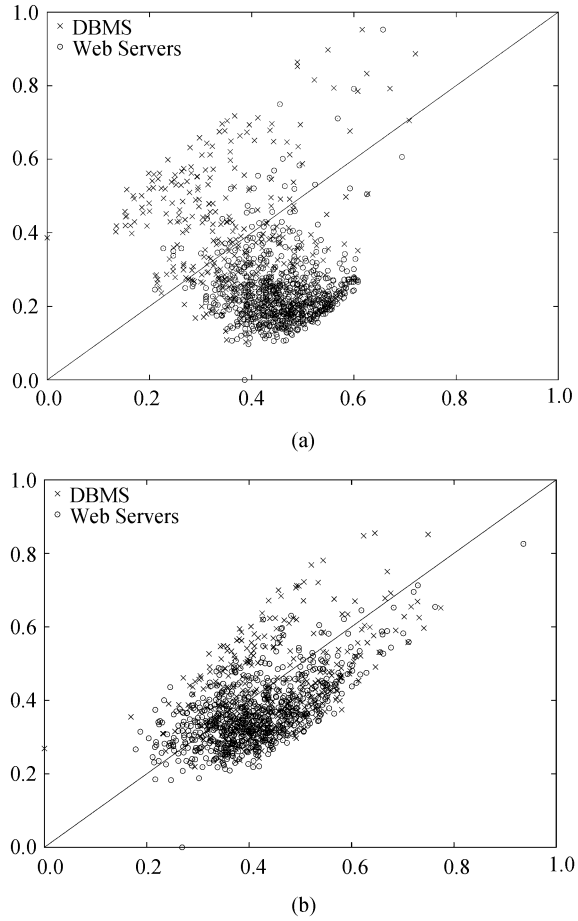

Fig.9. VMs distances from cluster centroids. (a) PCA-based methodology. (b) Correlation-based approach.

Fig.9 clearly shows that the PCA-based approach is more effective in discriminating between the two classes of VMs. In Fig.9(a) the points are spread on a wider area, while in Fig.9(b) they are located closer

to the bisector line. This means that the feature vectors computed with the PCA-based methodology can better capture and express the differences in behavior between VMs belonging to different clusters, thus facilitating the clustering task and improving the overall performance. It is worth noting that such advantage of the PCA-based approach is more evident with very short time series, as in the considered case where the time series length is equal to one day.

5.2 Methodology Computational Cost

Even if clustering is carried out with a low frequency, compared with other monitoring and management operations, it is worth investigating the computational costs of VM clustering. In particular, we evaluate to which extent the PCA-based methodology may reduce the execution time of the clustering step with respect to the correlation-based approach. We focus on the final clustering phase because every previous step of the methodology can be distributed over different nodes of the infrastructure. For example, we can consider to generate the feature vectors of each VM directly at the level of the time series aggregator on each host. On the other hand, clustering is more difficult to parallelize and its computational cost may affect the scalability of the methodology. The computational cost of the clustering step depends on two main elements: the number of considered metrics, which determines the length of the feature vectors given in input to the clustering algorithm, and the number of VMs to cluster. As pointed out in Section 3, the feature vectors generated by the proposed PCA-based methodology to describe VMs behavior are shorter than those computed through the correlation-based approach for the same number of considered metrics. To evaluate the reduction of the computational cost, we exploit the Enterprise data center case study; in particular, we consider a number of VMs equal to 50 and 110, and metric time series of 15 days. Fig.10 shows the execution time of the clustering step as a function of the number of metrics, ranging from 2 to 11, for PCA-based and correlation-based approaches.

The graph clearly shows the different trends of the clustering execution time for the two approaches. In the case of PCA-based methodology we observe a linear growth of the time required for clustering as the number of metrics increases, while for the correlation-based alternative the growth of clustering time is super-linear. This result is consistent with the theoretic computational complexity of the k-means algorithm[28], which is in the order $\mathcal{O}(IKNV)$, where I is the number of iteration of the k-means clustering, K is the number of clusters, N is the number of VMs, and V is the length

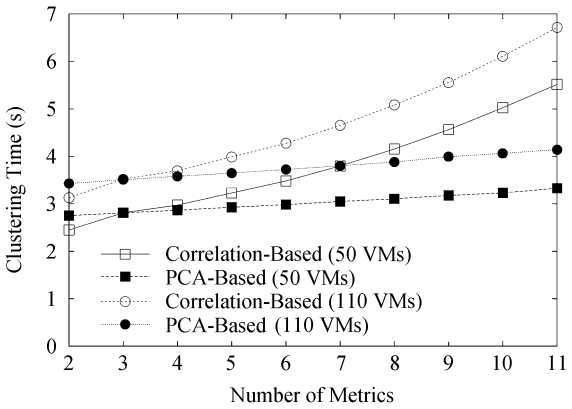

Fig.10. Clustering time for varying number of metrics and VMs.

of the feature vector used to describe each VM. In the PCA-based case, $V = M \times H$, with $H = 1$ because we consider for the feature vector only the principal component associated with the first eigenvector of the VM correlation matrix. For this reason, the dependency between clustering time and number of metrics is linear. On the other hand, in the case of correlation-based approach the length of the feature vector V corresponds to the size of the lower triangular sub-matrix of the VM correlation matrix ($V = \frac{M^2-M}{2}$); hence, V has a quadratic dependence on the number of metrics M, thus explaining the super-linear growth of the clustering time with respect to M.

From Fig.10, we also observe that the difference in clustering execution time grows as the number of metrics increases, with a reduction for the PCA-based approach of 39% and 41% with respect to the correlation-based approach for 50 and 110 VMs, respectively. The graph shows just one exception for the case of two metrics, where the correlation-based clustering is faster than the PCA-based. This can be explained by considering that in this case the feature vector in the correlation-based approach consists of just one element (the correlation coefficient between the two considered metrics).

5.3 Sensitivity Analysis

Now we investigate the sensitivity of the PCA-based methodology to 1) the number of principal components selected to build the feature vectors and 2) to the number of VMs to cluster. The sensitivity analyses are based on the Enterprise data center case study.

5.3.1 Impact of Principal Components Selection

The computational cost of the methodology is proved to be significantly reduced with respect to the correlation-based approach thanks to the choice of selecting just the first principal component to build the

VM feature vectors. Hence, it is important to evaluate how the performance is affected by the number of principal components considered. We select a number of principal components ranging from 1 to 10 to build the VM feature vectors. Fig.11 shows the results of the PCA-based methodology for short time series lengths of 1, 3, 5 and 10 days.

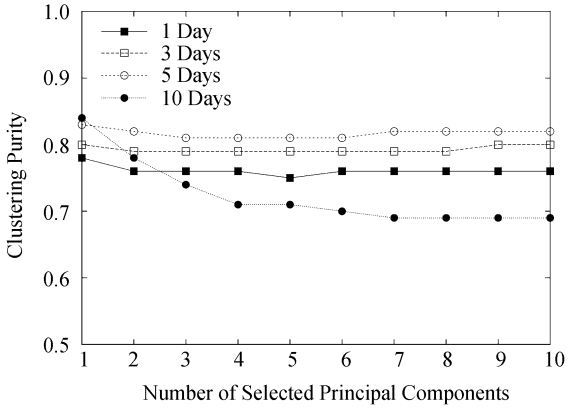

Fig.11. Clustering purity for increasing number of selected principal components.

We observe that the clustering purity does not increase as the number of principal components grows, but remains rather stable for almost all the considered time series lengths, with a more marked decrease for time series of 10 days. Fig.11 confirms that the first principal component, which is the most relevant component in terms of contribution to the overall variance of the dataset, is sufficient to characterize the VM behavior for our purpose of clustering (as indicated by the scree plot in Section 4). Adding more principal components does not add any relevant information: the data introduced into the feature vector by the additional components represent just background noise. In most cases this noise contribution just scatters evenly among the VMs over the newly added dimensions and does not affect the outcome of the clustering process. We present also a case (for the time series of 10 days) where the noise component determines a different clustering outcome, thus hindering the quality of the methodology results.

5.3.2 Impact of Virtual Machine Number

The last sensitivity analysis concerns the evaluation of the methodology performance for different numbers of VMs to cluster. This analysis is critical to demonstrate that the proposed approach can provide stable clustering performance even for large-scale cloud data centers where each customer may acquire large sets of VMs to run his/her applications.

We evaluate the clustering purity for different time series lengths and numbers of VMs. Table 3 shows how the clustering purity changes as the number of VMs grows from 20 to 110, considering time series with length of 1, 5, and 10 days.

Table 3. Impact on Purity of VM Number

Number of VMs	Time Series Length		
	10 Days	5 Days	1 Day
20	0.86	0.85	0.82
30	0.84	0.86	0.81
40	0.85	0.86	0.82
50	0.84	0.86	0.80
60	0.85	0.84	0.80
70	0.84	0.85	0.81
80	0.85	0.85	0.80
90	0.84	0.86	0.80
100	0.85	0.85	0.81
110	0.84	0.83	0.80

We observe that the clustering purity is mostly unaffected by the numbers of elements to cluster. We consider the stability of the clustering performance very important because it means that the proposed methodology is a viable option in the case of large data centers.

5.4 Summary of Results

The experimental results can be summarized as follows:

- The proposed PCA-based methodology outperforms the correlation-based approach in both case studies, achieving good clustering purity even for very short time series of only one day (94% for the EC2 Amazon and 80% for the Enterprise data center case studies), as shown in Subsections 5.1.1 and 5.1.2.
- The performance of the PCA-based methodology remains stable for sampling frequencies ranging from 1 to 5 minutes. Good performance is achieved for 5 minutes sampling frequency, that is commonly used in data centers monitoring systems (Subsection 5.1.1).
- The computational cost of VM clustering shows a linear dependency on the number of metrics against the quadratic trend of the correlation-based approach. Hence, the clustering of the PCA-based methodology provides good scalability especially when a large number of metrics is used (Subsection 5.2).
- The automatic selection of principal components can reduce the problem dimensionality by identifying the most relevant information; in our case studies, considering more than one principal component does not improve the clustering performance (Subsection 5.3.1).
- The performance of the automatic VM clustering is stable with respect to varying number of VMs to

cluster, thus confirming the viability of the proposed methodology even for large data centers (Section 5.3.2).

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.

Resource management strategies in large virtualized data centers 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 through optimization algorithms. The two approaches can be combined together[12]. Examples of reactive solutions are [5] and [6], that propose a mechanism based on adaptive thresholds regarding CPU utilization values. A similar approach was described also by Wood *et al.* in [2] 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 local manager level. On the other hand, an example of periodic VM consolidation solutions was proposed by Kusic *et al.* in [29], where VM consolidation is achieved through a sequential optimization approach. Similar solutions were proposed in [4, 8]. 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. 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 to improve the scalability of consolidation strategies for cloud systems. The first proposal to address scalability issues in monitoring through VM clustering was proposed in [10-11]. These studies exploit the correlation of VMs resources usage to identify similar behavior among VMs. However, these solutions suffer 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. This paper represents a clear step ahead with respect to the previous studies: the proposed methodology based on principal component analysis can achieve better and more stable results with respect to the correlation-based approach in different scenarios. Moreover, improvements of the computational cost are obtained thanks to the automatic selection and reduction of the information fed into the clustering algorithm.

As for the issue of monitoring large data centers, current solutions typically exploit frameworks for periodic collection of system status indicators, such as Cacti③, Munin④ and Ganglia⑤. Cacti is an aggregator of data transferred through 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. Ganglia provides a significant advantage over the previous solutions as it 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[30-31], even in cloud infrastructures[32], by storing resource usage time series describing the behavior of nodes and virtual machines. Another solution for scalable monitoring was proposed in [3], where data analysis based on the MapReduce 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 node independent of the others. This approach fails to take advantage from the similarities of objects sharing the same behavior. On the other hand, a class-based management allows the system to perform a fine-grained monitoring for only a subset of nodes that are representative of a class, while other members of the same class can be monitored at a much more coarse-grained level. Our proposal explicitly aims to address this problem.

7 Conclusions

The rapid increase in size and complexity of modern cloud data centers poses major challenges in terms of scalability for the monitoring and management of the system resources. In this paper we proposed to address scalability issues by clustering VMs into classes that share similar behaviors, starting from their resource usage. To cluster similar VMs, the proposed methodology considers multiple resources, ranging from CPU to storage and network, and exploits the statistical technique principal component analysis to automatically remove not relevant information from the VM description. The application of the proposed methodology to two case studies, a virtualized testbed and a real enterprise data

③Cacti, http://www.cacti.net/, June 2013.

④Munin, http://munin-monitoring.org/, June 2013.

⑤Ganglia Monitoring System, http://ganglia.sourceforge.net/, June 2013.

center hosting multi-tier Web applications, showed that the accuracy of VMs clustering ranges between 100% and 80% for every considered scenario, including the case of very short time series of the monitored resources with length of only one day. It is worth noting that the automatic selection of relevant data obtained thanks to the PCA technique leads to a twofold advantage, both in terms of performance and computational cost of VM clustering, with respect to the naive correlation-based approach. Furthermore, we demonstrated that the proposed methodology can reduce the amount of collected data, thus effectively contributing to address the scalability issues of the monitoring system.

References

[1] 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. the 12th International Middleware Conference*, Dec. 2011, pp.123-142.

[2] Wood T, Shenoy P, Venkataramani A, Yousif M. Black-box and gray-box strategies for virtual machine migration. In *Proc. the 4th USENIX Conference on Networked Systems Design and Implementation*, Apr. 2007, pp.229-242.

[3] Andreolini M, Colajanni M, Tosi S. A software architecture for the analysis of large sets of data streams in cloud infrastructures. In *Proc. the 11th IEEE International Conference on Computer and Information Technology* (*IEEE CIT 2011*), Aug. 31-Sept. 2, 2011, pp.389-394.

[4] Ardagna D, Panicucci B, Trubian M, Zhang L. Energy-aware autonomic resource allocation in multitier virtualized environments. *IEEE Transactions on Services Computing*, 2012, 5(1): 2-19.

[5] Beloglazov A, Buyya R. Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In *Proc. the 8th Int. Workshop on Middlewave for Grids, Clouds and e-Science*, Dec. 2010, Article No.4.

[6] Gmach D, Rolia J, Cherkasova L, Kemper A. Resource pool management: Reactive versus proactive or let's be friends. *Computer Networks*, 2009, 53(17): 2905-2922.

[7] Lancellotti R, Andreolini M, Canali C, Colajanni M. Dynamic request management algorithms for Web-based services in cloud computing. In *Proc. the 35th IEEE Computer Software and Applications Conference*, Jul. 2011, pp.401-406.

[8] Tang C, Steinder M, Spreitzer M, Pacifici G. A scalable application placement controller for enterprise data centers. In *Proc. the 16th International Conference on World Wide Web*, May 2007, pp.331-340.

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

[10] Canali C, Lancellotti R. Automated clustering of virtual machines based on correlation of resource usage. *Communications Software and Systems*, 2012, 8(4): 102-109.

[11] Canali C, Lancellotti R. Automated clustering of VMs for scalable cloud monitoring and management. In *Proc. the 20th International Conference on Software, Telecommunications and Computer Networks*, Sept. 2012, pp.1-5.

[12] Gong Z, Gu X. PAC: Pattern-driven application consolidation for efficient cloud computing. In *Proc. the IEEE Int. Symp. Modeling, Analysis & Simulation of Computer and Telecommunication Systems*, Aug. 2010, pp.24-33.

[13] Setzer T, Stage A. Decision support for virtual machine reassignments in enterprise data centers. In *Proc. the IEEE/IFIP Network Operations and Management Symposium Workshops* (*NOMS*), Apr. 2010, pp.88-94.

[14] Castro M, Liskov B. Practical Byzantine fault tolerance. In *Proc. the 3rd Symposium on Operating Systems Design and Implementation*, Feb. 1999, pp.173-186.

[15] Cecchet E, Chanda A, Elnikety S, Marguerite J, Zwaenepoel W. Performance comparison of middleware architectures for generating dynamic Web content. In *Proc. the 4th International Middleware Conference*, Jun. 2003, pp.242-261.

[16] Kavalanekar S, Narayanan D, Sankar S, Thereska E, Vaid K, Worthington B. Measuring database performance in online services: A trace-based approach. In *Lecture Notes in Computer Science 5895*, Nambiar R, Poess M (eds.), Berlin, Heidelberg: Springer-Verlag, 2009, pp.132-145.

[17] de Menezes M A, Barabási A L. Separating internal and external dynamics of complex systems. *Physical Review Letters*, 2004, 93(6).

[18] Hyvärinen A, Oja E. Independent component analysis: Algorithms and applications. *Neural Networks*, 2000, 13(4/5): 411-430.

[19] Greenacre M. Correspondence Analysis in Practice. Chapman and Hall/CRC, 2007.

[20] Mardia K V, Kent J T, Bibby J M. Multivariate Analysis (Probability and Mathematical Statistics). Academic Press, 1995.

[21] Abdi H, Williams L J. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2010, 2(4): 433-459.

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

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

[24] Andreolini M, Colajanni M, Pietri M. A scalable architecture for real-time monitoring of large information systems. In *Proc. the 2nd IEEE Symposium on Network Cloud Computing and Applications*, Dec. 2012, pp.143-150.

[25] Dinda P A, O'Hallaron D R. Host load prediction using linear models. *Cluster Computing*, 2000, 3(4): 265-280.

[26] Vogels W. Beyond server consolidation. *ACM Queue*, 2008, 6(1): 20-26.

[27] Amigó E, Gonzalo J, Artiles J, Verdejo F. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Journal of Information Retrieval*, 2009, 12(4): 461-486.

[28] Manning C D, Raghavan P, Schtze H. Introduction to Information Retrieval. New York, NY, USA: Cambridge University Press, 2008.

[29] Kusic D, Kephart J O, Hanson J E, Kandasamy N, Jiang G. Power and performance management of virtualized computing environment via lookahead. *Cluster Computing*, 2009, 12(1): 1-15.

[30] Chung W C, Chang R S. A new mechanism for resource monitoring in Grid computing. *Future Generation Computer Systems*, 2009, 25(1): 1-7.

[31] Naeem A N, Ramadass S, Yong C. Controlling scale sensor networks data quality in the Ganglia grid monitoring tool. *Communication and Computer*, 2010, 7(11): 18-26.

[32] Tu C Y, Kuo W C, Teng W H, Wang Y T, Shiau S. A power-aware cloud architecture with smart metering. In *Proc. the 39th International Conference on Parallel Processing Workshops*, Sept. 2010, pp.497-503.

Claudia Canali is a researcher at the University of Modena and Reggio Emilia since 2008. She received Laurea degree summa cum laude in computer engineering from the same university in 2002, and Ph.D. degree in information technologies from the University of Parma in 2006. Her research interests include cloud computing, social networks, and wireless systems for mobile Web access. On these topics, Claudia Canali published about forty articles in international journals and conferences. She is member of IEEE Computer Society.

Riccardo Lancellotti is a researcher at the University of Modena and Reggio Emilia since 2005. He received the Laurea degree summa cum laude in computer engineering from the University of Modena and Reggio Emilia in 2001 and the Ph.D. degree in computer engineering from the University of Roma "Tor Vergata" in 2003. His research interests include geographically distributed systems, peer-to-peer systems, social networks and cloud computing. On these topics he published more than fifty papers in international journals and conferences. He is a member of the IEEE Computer Society and ACM.