A Comparison of Techniques to Detect Similarities in Cloud Virtual Machines

Authors not disclosed

Abstract: Scalability in monitoring and management of cloud data centers may be improved through the clustering of virtual machines (VMs) exhibiting similar behavior. However, available solutions for automatic VM clustering present some important drawbacks that hinder their applicability to real cloud scenarios. For example, existing solutions show a clear trade-off between the accuracy of the VMs clustering and the computational cost of the automatic process; moreover, their performance shows a strong dependence on specific technique parameters. To overcome these issues, we propose a novel approach for VM clustering that uses Mixture of Gaussians (MoGs) together with the Kullback Leiber divergence to model similarity between VMs. Furthermore, we provide a thorough experimental evaluation of our proposal and of existing techniques to identify the most suitable solution for different workload scenarios.

Keywords: Cloud Computing; Clustering; Virtual Machines; Cloud Monitoring; Kullback-Leibler Divergence; Mixture of Gaussians.

1 Introduction

Cloud computing will play a major role in the future Internet of Services, enabling on-demand provisioning of applications and computing infrastructures. The capability of cloud computing systems to cope with the increasing resource demand in the next few years will be critical for the future development of the emerging digital society [Moreno-Vozmediano et al. (2013)].

Among the existing cloud service models, this paper focuses on Infrastructure as a Service (IaaS) systems, where different customer applications are hosted in virtualized environments: typically, a customer application is structured in multiple software components (e.g., the tiers of a multi-tier Web application, or Web Services to be composed) [Dai et al. (2014)], and each component runs on a separate virtual machine (VM). The growing popularity of IaaS cloud systems has led to constantly increasing size and complexity of the IaaS data centers, that are facing novel challenges for the scalability of resource monitoring and managing tasks. Moreover, it is worth to note that from the point of view of IaaS cloud providers the VMs are usually considered as black boxes with independent behaviors: this means that data need to be collected with fine granularity about each single VM of the data center, thus exacerbating the scalability issues concerning the monitoring task. A similar scalability problem occurs when the provider strives to place VMs on the infrastructure minimizing the number of physical servers, due to the computational cost of the underlying optimization problem.

Recent solutions proposed in literature [Canali and Lancellotti (2014d,c)] show that scalability issues related to resource monitoring in IaaS cloud systems can be addressed by automatically clustering VMs showing a similar behavior in terms of resource usage. The knowledge of similar VMs clusters allows the monitoring system to collect data with fine granularity only on a subset of representative VMs for each cluster, achieving a significant reduction in the amount of globally collected data. However, existing solutions for automatic VMs clustering present important drawbacks: indeed, they may be very accurate, at the price of high computational costs [Canali and Lancellotti (2014c)], or able to provide very fast results with limited accuracy [Canali and Lancellotti (2014d)]. Moreover, the performance of most solutions may change significantly depending on specific parameters or pre-processing steps that, if not correctly tuned, may result in poor performance.

The main contribution of this paper is twofold. First, we present a non-parametric clustering technique, namely *KL-based*, that exploits Mixture of Gaussians (MoGs) to model VM behavior and the Kullback-Leibler divergence to measure the similarity between VMs, with the goal to provide fast and accurate VM classification without relying on any specific parameter. Second, we compare several existing solutions for VM clustering, such as Ensemble-based [Canali and Lancellotti (2014c)], Correlation-based, and PCA-based [Canali and Lancellotti (2014d)] techniques, with the new KL-based proposal to provide an insight on the pros and cons of each technique.

A preliminary version of *KL-based* clustering was proposed in [Canali and Lancellotti (2014b)], but the present study extends the previous work in several ways. First, we apply the clustering to a wider range of workloads and to shorter time series (up to 6 hours) to achieve a deeper understanding of the clustering performance over different scenarios. Second, we analyze in detail the issues of clustering based on short time series in a dynamic scenario, that is when newly acquired VMs can enter the system any time. Third, we discuss the potential benefits for the monitoring system in terms of reduction of collected data. We carry out an experimental evaluation based on two different scenarios: a private cloud data center hosting a multi-tier Web application and a synthetic Web benchmark deployed on a commercial cloud infrastructure. Experimental results assess

the limitations of existing solutions and evaluate our proposal. We demonstrate that the KL-based technique achieves a clustering with an accuracy comparable with the best existing solutions, but with a computational requirements significantly lower and without need to tune any algorithm parameter.

The rest of this paper is organized as follows. Section 2 describes the reference scenario for the application of VM clustering. Section 3 presents the proposed KL-based technique and Section 4 describes the existing approaches. Section 5 provides the experimental evaluation of clustering techniques applied to different scenarios. Section 6 discusses the related work, while Section 7 concludes the paper with some final remarks.

2 Reference Scenario

Let us introduce the scenario of a IaaS cloud data center that exploits a VM clustering solution to improve its monitoring and management scalability [Canali and Lancellotti (2014c,d)]. This will be the reference scenario for our proposal.

Figure 1 represents a Iaas cloud system that rely on a two-level management strategy [Gong and Gu (2010)]. The first level focuses on *local management* and is carried out on each host node (that is the physical server where VMs run). The goal of local management is to detect in real-time overload conditions leveraging the resource measurements of the VMs hosted on the node and relying on live VM migration in case of overload [Wood et al. (2007)]. The second level, namely *global management*, is carried out on a management node. Global management excutes periodically a global consolidation to place VMs on as few host nodes as possible to reduce the infrastructure costs and avoid expensive resource over-provisioning [Ardagna et al. (2012)]. Since consolidation strategies in IaaS cloud infrastructures usually consider each VM as a stand-alone object with independent resource usage patterns, detailed information has to be collected with high sampling frequency (typically 1 sample every 5 minutes [Ardagna et al. (2012), Setzer and Stage (2010)] or even less) about each VM. This fine-grained sampling may determine scalability issues for the monitoring system. Furthermore, as the server consolidation must consider every VM of the data center, solving optimization problem for the global management task may pose scalability problems as well.

Clustering has the potential to improve the scalability of monitoring and management processes by automatically grouping together VMs showing similar behaviors in terms of resource usage [Canali and Lancellotti (2014c,d)]. The goal of clustering is the identification of VMs that host the same software component of the same customer application.

To improve monitoring scalability we select few representative VMs for each identified class as soon as the clustering is done. We choose to select at least three representatives due to the possibility that a selected representative unexpectedly changes its behavior with respect to its class: quorum-based techniques can be exploited to cope with byzantine failures of representative VMs [Castro and Liskov (1999)]. At this point, only the representative VMs of each class are monitored with high sampling frequency 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. Moreover, sudden changes leading to server overload are handled by the local management through live VM migration.

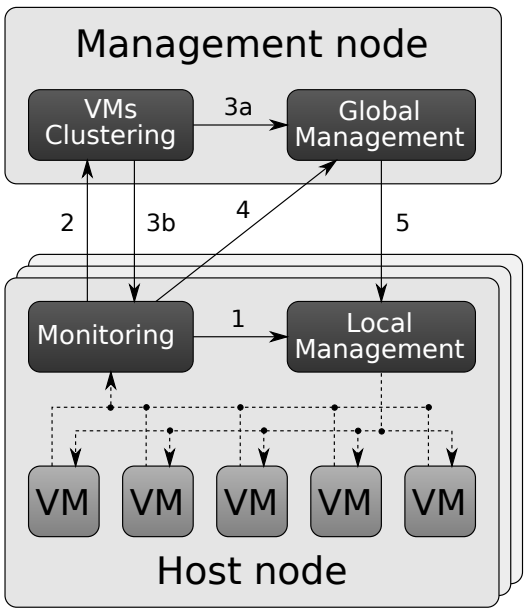


Figure 1: Cloud system using VM clustering

The clustering technique may also improve the scalability of the global management operations. The critical step of server consolidation is the solution of a multi-dimensional bin-packing problem, where each VM must be assigned to a host node without exceeding the node capacity in terms of available resources. This problem is typically solved using heuristics and simplifications (such as considering only one resource such as the CPU [Setzer and Stage (2010)]). We argue that, by knowing that VMs belong to classes, we can switch to a *cluster-based consolidation* that solves a much smaller optimization problem (with the possibility to use more complex and accurate algorithms). Then, the VM allocation solution may be replicated as a building block to create the server consolidation solution for the whole data center.

Figure 1 shows how the previously introduced components interacts. Collection of data about resource usage of the hosted VMs is performed by the *monitoring* system (hosted on each node). Such data are then sent to the *local management* system (arrow 1), which is responsible for triggering live VM migration in case of host overload [Wood et al. (2007)]. A second function of the monitoring system is to process and send data to the *VM clustering* system (2), which automatically builds a VM behavior model and groups similar VMs applying one of the techniques proposed in [Canali and Lancellotti (2014c,d)]. The identified

VM classes and the representatives of each class are then communicated to the the *global management* ($3a$) and to the *monitoring* system ($3b$). This result is used by the monitoring system to differentiate the sampling frequency between representative and non representative VMs. The data collected with different granularity are sent to the *global management* system (4) which is responsible for two tasks. First, the global management executes periodically the cluster-based consolidation strategy, exploiting the resource usage of the representative VMs to characterize the behavior of every VM of the same class and possibly taking also into account information about the partition of VMs into clusters. The local manager receives the global consolidation decisions (5) and applies them. Second, the global management system checks for behavioral drifts of non representative VMs, that must be re-classified.

It is worth to note that, unlike server consolidation, VM monitoring and clustering are not necessarily periodic operations. A new VM can enter the system at any time, and when it happens fine-grained monitoring starts for that VM to build a representation of its behavior. As soon as the data to build the VM behavior model is available, the clustering function is invoked. Clustering uses both already available VM behavior models (collected in previous periods for the existing VMs) and the fresh data for the new VM. In a similar way, VMs that change their behavior, as well as VMs causing overload of physical servers detected by *local management*, are marked as unclassified VMs and are monitored again with high sampling frequency to be re-clustered.

From the description of the reference scenario, we understand how the application of an automatic clustering technique may improve the scalability of cloud monitoring and management. However, the existing clustering solutions [Canali and Lancellotti (2014c,d)] present drawbacks related to the computational cost, the stability of the accuracy results and the dependence on specific parameter values. In the next section, we present a novel VM clustering technique that provides high accuracy with limited computational requirements, and does not depend on any parameter. This new technique will then be compared with the existing available solutions to understand which solution is the most appropriate for the different possible scenarios of cloud data centers.

3 KL-based clustering technique

Let us now describe in detail the KL-based technique for automatic VMs clustering. We provide an overview of the structure of the proposed technique by identifying three conceptual steps that are common to every solution for VM clustering. This approach allows us to provide a common background for comparing our proposal with the existing alternatives, that will be described in Section 4. The proposed KL-based technique, as well as the existing clustering alternatives, includes the following three main steps:

1. Extraction of a *quantitative model* for describing the VM behavior
2. Definition of a *distance* representing the similarities among the VMs
3. *Clustering* based on the proposed distance to identify classes of similar VMs

3.1 VM behavior quantitative model

The description of the VM behavior is obtained by modeling the usage of the VM resources through a linear combination of multiple Gaussian distributions, that we call a *Mixture of Gaussians* (MoG). To formalize our model, we consider a set of N VMs, and for each VM $n \in [1, N]$ a set of M metrics, where each metric $m \in [1, M]$ represents the usage of a VM resource.

Let $(\boldsymbol{X}_1^n, \boldsymbol{X}_2^n, \ldots, \boldsymbol{X}_M^n)$ be a set of time series, where $\boldsymbol{X}_m^n$ is the vector consisting of the samples of the resource usage represented by the metric m of VM n. The probability density function $p(\boldsymbol{X}_m^n)$ of each time series can be considered as the description of the behavior of metric m on VM n. We approximate the probability density using the previously introduced MoG:

$$p(\boldsymbol{X}_m^n) \approx mog_m^n = \sum_{i=1}^{G_m^n} \pi_{m,i}^n \cdot g(\mu_{m,i}^n, \sigma_{m,i}^n)$$

where G_m^n is the number of Gaussian distributions used to model the probability distribution of samples for metric m on VM n, and $\pi_{m,i}^n$, $\mu_{m,i}^n$, $\sigma_{m,i}^n$ are the weight, mean value and standard deviation of the specific component of the MoG.

The approximation of the probability distribution through a MoG is carried out using the *mclust* package provided by the *R* statistical analysis software [Fraley et al. (2013)]. The package performs a clustering of the data samples to automatically identify the number of modes in the probability density function and then iteratively adjusts the parameters of each Gaussian component in order to obtain a close fitting of the probability density function with the MoG. It is worth to note that no parameters are involved in this process.

3.2 Definition of a distance

The second step of the technique consists in introducing a distance to define similarities among VMs starting from the representation of their behavior. To define the VM distance we exploit the Kullback-Leibler (KL) divergence, which measures the similarity between two probability distributions, possibly modeled as MoG [Kullback (1997)]. The KL divergence between two MoG mog_1, mog_2 is defined as:

$$KL(mog_1, mog_2) = \int_{x=0}^{\infty} mog_1(x) \ln\left(\frac{mog_1(x)}{mog_2(x)}\right) dx$$

However, an analytical solution in a closed form of such equation is not always possible, and numeric approximation of the integrals is computationally expensive. For two Gaussian distributions g_1 and g_2, the KL divergence can be defined with the following closed analytical form:

$$KL(g_1, g_2) = \log(\frac{\sigma_2}{\sigma_1}) + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \qquad (1)$$

where μ_1, μ_2, σ_1, and σ_2 are the mean values and the standard deviation of the two distributions g_1 and g_2 [Hershey and Olsen (2007)].

For MoGs, we can use an approximation, namely variational divergence [Hershey and Olsen (2007)], that extends Equation 1. The KL divergence for two Mixtures of Gaussian distributions mog_1 and mog_2 is thus defined as:

$$KL_{VD}(mog_1, mog_2) = \sum_{i=1}^{n1} \pi_{1,i} \cdot \ln\left(\frac{\sum_{j=1}^{n1} \pi_{1,j} e^{-KL(g_{1,i}, g_{1,j})}}{\sum_{k=1}^{n2} \pi_{2,k} e^{-KL(g_{1,i}, g_{2,k})}}\right)$$

Finally, the distance between two VMs n_1 and n_2 is the sum of squares of the KL divergence between the MoGs representing the VM behavior for each metric:

$$D(n_1, n_2) = \sum_{m=1}^{M} \left(KL_{VD}(mog_m^{n_1}, mog_m^{n_2})\right)^2$$

3.3 Clustering

The last step of the technique returns the final clustering solution. The distance previously defined is computed for each couple of VMs, obtaining a distance matrix $\boldsymbol{D}$. Then, the distance matrix is fed into a clustering algorithm to obtain the final solutions.

It is worth to note that to cluster together elements of a set starting from a distance matrix, traditional algorithms such as K-means or Kernel K-means [Jain (2010)] are not viable options because they expect as input a set of coordinates for each element to cluster. For this reason, we exploit the widely adopted spectral clustering algorithm, which is explicitly designed to manage as input a similarity matrix or a matrix-based representation of graphs [Ng et al. (2001)]. The output of the clustering step is a vector $\boldsymbol{C}$, 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 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 Existing approaches

In this section we present three existing solutions to implement the steps for a VM clustering technique outlined in Section 3.

4.1 Ensemble-based

The *Ensemble-based* technique, presented in [Canali and Lancellotti (2014c)], uses the probability density of the utilization of VM resources to model the VM behavior, like the KL-based approach. But, differently from the proposed technique, the behavioral model of a VM consists in a set of normalized histograms, one for each considered metric of the VM.

Next, the Ensemble-based approach uses the Bhattacharrya distance [Bhattacharyya (1943)], that provides the distance between two different histograms, to compute a per-metric distance between every couple of VMs. The use of this distance for VM clustering has been proposed for the first time in a preliminary paper [Canali and Lancellotti (2013)].

Then a set of per-metric clustering solutions are obtained from the distance matrices obtained applying the per-metric distance definition. To this aim, we rely on the same spectral clustering algorithm already described for the proposed KL-based clustering technique. To merge the separate clustering solutions, we exploit a further clustering step where the input is a co-occurrence matrix determined through ensemble techniques. Specifically, the co-occurrence matrix contains, for each couple of VMs $(n_1, n2)$, the number of times that the VMs are clustered together throughout the whole set of per-metric clustering solutions. The last clustering step exploits again the spectral algorithm to obtain the final clustering solution.

4.2 Correlation-based

The *correlation-based* technique [Canali and Lancellotti (2014d)] describes the behavior of a VM using the correlation between the time series of different metrics of the same VM. From the correlation between each couple of metrics (m_1, m_2) we obtain a vector of length $L = \frac{M \cdot (M-1)}{2}$, that is the feature vector characterizing the VM.

The output of the first step of the technique is a set of N feature vectors (one for each VM) that define a L-dimensional vector space. The vector space uses an Euclidean distance to compute the distance between two feature vectors.

Finally, the set of N feature vectors describing the VMs is used as an input for the clustering algorithm. Specifically, we use the K-means algorithm [Jain (2010)] for clustering.

4.3 PCA-based

The *PCA-based* technique [Canali and Lancellotti (2014d)] is an evolution of the previously described Correlation-based solution aiming to improve the quality of the clustering results.

This technique exploits the correlation values between each couple of time series referring to the M considered metrics of the same VM like the previous Correlation-based solution. The correlation values are assembled into a $M \times M$ square matrix; then, we compute the eigenvectors of this correlation matrix. In other words, we apply a Principal Component Analysis (PCA) over the time series of each VM. Using the well known rule of the scree plot visual analysis [Abdi and Williams (2010)], we identify the number P of components that are significant to reconstruct the VM behavior (in our experiments $P = 1$, as in [Canali and Lancellotti (2014d)]). As each component is associated to an eigenvector of the correlation matrix, we build a feature vector to describe the VMs behavior using only the P eigenvectors associated to the highest eigenvalues.

As for the Correlation-based approach, the feature vector of length $P \cdot M$ defines a feature vector space with an Euclidean distance. Clustering is then carried out with the K-means algorithm.

5 Experimental evaluation

In this section we evaluate the proposed methodology and compare it with existing approaches for VMs clustering. Specifically, we aim to point out pros and cons of each technique, with particular attention to critical elements such as stability of the results, parameter dependence and computational cost. To this purpose, we apply the clustering techniques to two case studies that are described below. The experimental evaluation initially discusses the parameters that may affect the performance of each solution. We provide an experimental comparison of the clustering accuracy of the different techniques, then we pass to analyze the issues of clustering on short time series (less than 24 hours) in dynamic scenarios. Next, we analyze the sensitivity of the performance to the number of considered metrics. Finally, we compare the execution times of the different clustering approaches and discuss the potential benefits in terms of reduction of monitored data.

5.1 Case studies

To compare the performance of different clustering techniques we consider two case studies: the *Enterprise* dataset coming from a real data center, and the *EC2 Amazon* case study based on a synthetic dataset.

The *Enterprise* case study is based on a Web e-health application for the automated management of lab exams, which is hosted on a private enterprise data center. The application is deployed on 110 VMs according to a multi-tier architecture. The VMs are divided between the two software components of the Web application: Web servers and back-end servers (that are DBMS). This case study represents a real Web application hosted on a distributed data center, with resource usages showings the typical daily patterns that characterize Web traffic. We collect data about the resource usage of every VM for different periods of time, ranging from 5 days to 6 hours, with a sampling frequency of 5 minute.

The *EC2 Amazon* case study is based on a virtualized testbed running an e-commerce application: the application is built on the RUBiS benchmark and deployed over the Amazon Elastic Computing infrastructure (*micro* instances of VMs). The benchmark is hosted on a set of 36 VMs, with 12 VMs dedicated to Web servers, 12 to DBMS and 12 VMs running a set of emulated browsers. As the considered application is supporting a synthetic workload, the patterns of client requests are stable over time. For this reason, we collect samples with a frequency of 5 minute for 12 hours: longer time series would not provide additional information from a statistical point of view in this steady state scenario.

In both the case studies, 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 resources

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

The final goal of VM clustering applied to these case studies is to correctly classify the VMs running different software components: Web server and DBMS for the Enterprise scenario, and Web servers, BDMS and emulated browsers for the EC2 Amazon case study. As a performance indicator to evaluate the performance of VM clustering, we consider a widely used measure, namely *purity* [Amigó et al. (2009)], that expresses the fraction of correctly classified VMs. 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.

5.2 Comparison of Clustering Approaches

We now compare the different clustering techniques applied to our case studies. In particular, we consider the proposed KL-based methodology and three existing approaches: Ensemble-based [Canali and Lancellotti (2014c)], PCA-based and Correlation-based [Canali and Lancellotti (2014d)].

A first comparison is related to the parameters affecting the performance of the different techniques. The Ensemble-based solution relies on normalized histograms to represent VM behavior. A parameter involved in this approach is the number of bins used to compute the histograms. Multiple rules are available to determine this number, and results in literature show that the selected rule can affect the quality of the clustering performance [Canali and Lancellotti (2014c)]. On the other hand, the PCA-based solution is characterized by the number of principal components to feed into the clustering step, as mentioned in Section 4. Again, results in literature demonstrate that the number of considered components may affect the clustering performance [Canali

and Lancellotti (2014d)]. Finally, the Correlation-based solution is not dependent on any parameter, like the proposed KL-based technique [Canali and Lancellotti (2014d)].

The second comparison is based on a quantitative evaluation of the achieved clustering performance. The clustering purity achieved by the different clustering techniques in the Enterprise scenario is shown in Figure 2 as a function of the time series length, which ranges from 5 days to 6 hours. We see from Figure 2 that the performance of the Correlation-based approach significantly decreases for time series shorter than 3 days. On the other hand, the results of the other three techniques remains quite stable for every time series length, with a slight decrease of achieved purity for time series shorter than 1 day. The best performance is achieved by the Ensemble-based approach. The proposed KL-based technique shows a good stability for different time series lengths, while achieving performance only slightly worse with respect to the Ensemble-based approach, with differences in clustering purity ranging from 3% to 3.5%.

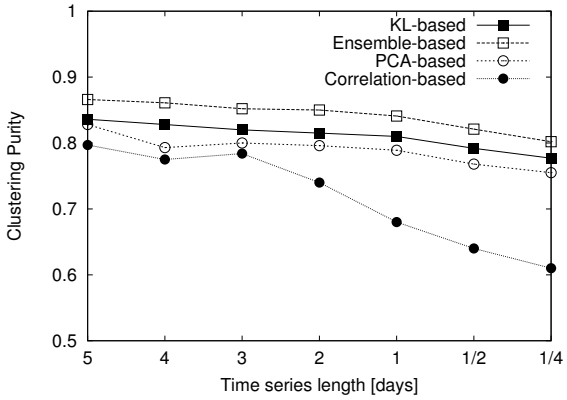


Figure 2: Clustering purity in the Enterprise scenario

We now pass to evaluate the performance of the clustering techniques when applied to the EC2 Amazon case study. The results are shown in Table 2.

Table 2 Clustering purity for EC2 Amazon case study

Clustering technique			
KL	**Ensemble**	**PCA**	**Correlation**
0.917	0.926	0.909	0.815

The clustering purity basically confirms the results of the Enterprise scenario. The Correlation-based technique achieves the lowest purity, while the other three alternatives show very similar results, with the Ensemble-based technique achieving a slightly better performance. We also observe that the EC2 Amazon scenario provides a clustering purity that is generally higher if compared to the Enterprise case study, even if the overall data collection time is limited to 12 hours. This result is motivated by the use of a synthetic workload for the EC2 Amazon case study, which is characterized by regular access patterns that increase the accuracy of the clustering algorithms.

5.3 Short time series in dynamic scenario

Using short time series (below 24 hours) to cluster similar VMs may present some issues if we consider a dynamic scenario where newly acquired VMs may frequently enter the system. A typical example is represented in Figure 3a. The graph shows the CPU utilization of a Web server (Enterprise case study) over a period of about 78 hours: the resource usage is characterized by the daily patterns that are typical of a Web application. Let us assume that during the first 6 hours (period $M1$) the monitoring system collects the resource usage time series for each VM in the system with fine granularity, that is one sample every 5 minutes. We also assume that at time $t = 36h$ new VMs enter the system, and fine-grained monitoring starts to collect their resource usage for the next 6 hours (period $M2$). At time $t = 42h$ the clustering function is invoked and operates on VM behavior models built on time series collected during $M1$ for the VMs that were initially in the system and during $M2$ for the newly entered VMs.

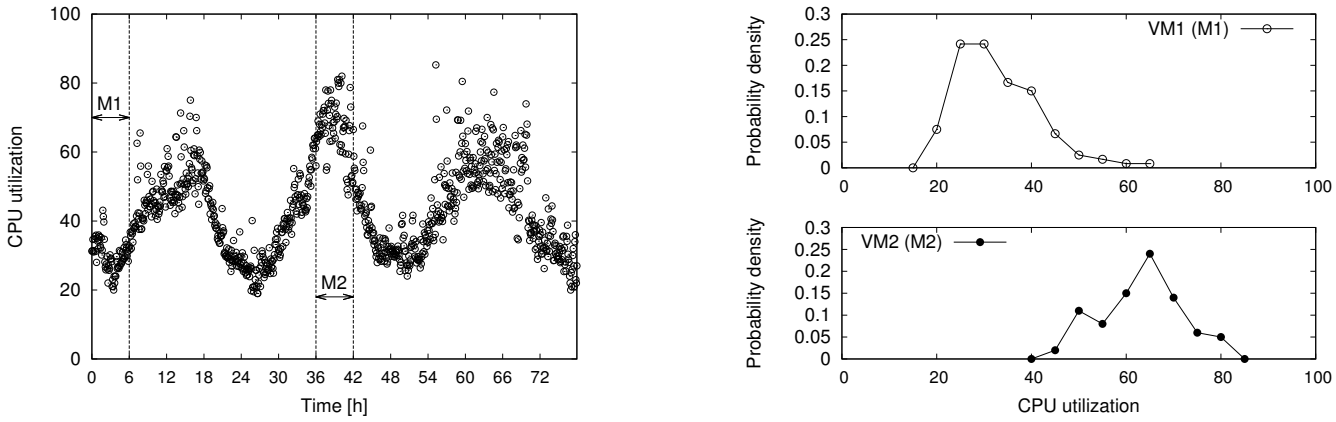
The problem that may be caused by working on short time series collected in different periods of time is evidenced in Figure 3b, that shows the probability density functions (that is used to model the VM behavior in the KL-based technique) for two VMs $VM1$ and $VM2$ monitored during periods $M1$ and $M2$, respectively. We observe that the two behavior descriptions are completely different, with a mode in the probability distribution close to 25% for $VM1$ and to 65% for $VM2$, hence the resulting Kullback-Leibler divergence of the two distributions is likely to be high. It is important to note that the differences in the behavior of the VMs are not related to a difference in the software component run on them (both VMs are Web servers) but to the different monitoring periods. In this experiment we aim to evaluate whether this difference may hinder the capability of the clustering technique to identify similar VMs.

To emulate the described dynamic scenario, we consider the 110 VMs of the Enterprise case study, and we apply the clustering techniques to resource usage time series collected during the period $M1$ for 85 VMs, and during period $M2$ for the remaining 25 VMs. Table 3 shows the achieved purity for the dynamic scenario, and compares it with the results obtained for a static scenario where all the 110 VMs are monitored during period $M1$.

Table 3 Clustering purity in static and dynamic scenarios

Clustering	**Scenario**	
Technique	**Static**	**Dynamic**
KL-based	0.778	0.721
Ensemble-based	0.802	0.743
PCA-based	0.755	0.751
Correlation-based	0.615	0.609

We observer that the clustering purity tends to decrease for all the techniques passing from the static to the dynamic scenario, but with a significant difference: the performance deterioration is more evident for KL- and Ensemble-based techniques (up to 6%), while the PCA- and Correlation-based approaches achieve more stable results, with a purity decrease



(a) Metric collection periods

(b) Probability density function

Figure 3: Clustering issue for short time series in dynamic scenario

below 1%. The motivation can be found in the different mechanism used to model the VM behavior. The approaches using the probability density function of the resource usage, like KL- and Ensemble-based techniques, suffer from the issue shown in Figure 3b and tend to be less effective in identifying VM similarities. On the other hand, the PCA- and Correlation-based techniques model the VM behavior based on the correlation between the time series of different metrics of the same VM: such correlation remains quite stable even in presence of workload daily fluctuation, thus explaining the smaller decrease of clustering purity in the dynamic scenario. It is worth to recall that this effect only applies when short time series (below 24 hours) are used to model the VM behavior, otherwise the presence of complete daily patterns does not affect the clustering performance.

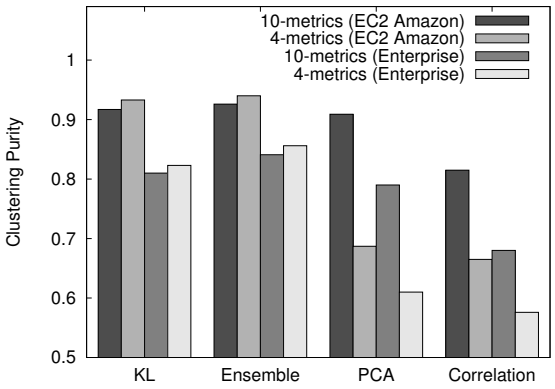


Figure 4: Clustering purity for different sets of metrics

5.4 Sensitivity to number of metrics

Another element that may affect the performance of the VM clustering is the number of metrics used to determine the VM behavior model. Using a high number of metrics may be counter-productive because non-significant data are likely to be fed into the final clustering step, with effects comparable to noise that degrades the clustering performance. In this experiment, we evaluate the sensitivity to the number of metrics on the clustering techniques by considering a reduced set of metrics, which is limited to 4 metrics mostly used in data center management strategies [Ardagna et al. (2012), Hu et al. (2012), Gong and Gu (2010)]: CPU and memory utilization, input and output packet rate. It is worth to note that an automatic mechanism to select metrics for VM clustering purposes has been proposed in a preliminary study by the authors [Canali and Lancellotti (2013)]: the selection, which is based on the analysis of autocorrelation and coefficients of variation of the time series, confirms the presence of the above mentioned metrics in the selected set.

Figure 4 shows the purity of the clustering approaches in the Enterprise and EC2 Amazon scenarios (time series of 1 day and 12 hours, respectively) for the entire set of 10 metrics and for the reduced set of 4 selected metrics. We observe that the number of metrics has very different impacts on the performance of the considered approaches, and the effect is similar in both the case studies. The KL-based and Ensemble-based techniques achieve quite stable results over the different set of metrics, with a purity slightly improved in the case of few representative metrics. On the other hand, the performance of the PCA-based and Correlation-based approaches drastically decreases in case of few metrics. For the latter approaches, indeed, the reduction of metrics causes an excessive decrease in the dimension of the feature vector space used to describe the VMs behavior. Differently, both KL-based and Ensemble-based techniques exploit a distance matrix that do not change dimension with the metric reduction, maintaining the capability to achieve accurate clustering. This result demonstrates that the KL-based and Ensemble-based techniques may achieve good clustering performance which is not affected by the presence of metric selection, thus outperforming their PCA- and Correlation based counterparts.

5.5 Execution time of clustering techniques

The global execution time required for VM clustering consists of three different contributions, corresponding to the

main steps of the methodology defined in Section 3: first, the time to extract the quantitative model of VMs behavior; second, the time to compute the VMs distance; third, the time to perform the clustering step. In this experiment, we evaluate the execution times of each step of the methodology on a machine equipped with an Intel Xeon, 2GHz CPU. Table 4 shows the execution times of the three contributions for the considered clustering techniques.

Table 4 Times [s] for clustering techniques

Clustering Technique	Step		
	Model	**Distance**	**Clustering**
KL-based	5.82	390.27	11.61
Ensemble-based	1.32	1417.42	69.72
PCA-based	0.11	n/a	5.21
Correlation-based	0.06	n/a	8.21

It is worth to note that the extraction of the quantitative behavioral model has to be performed separately for every considered VM, and can be parallelized on distributed nodes. On the other hand, the second and third contributions represent a centralized task that can not be parallelized. Hence, for a fair comparison we measure the time for extracting the behavioral model of a single VM (second column of the table), while the other contributions (third and fourth columns) are measured by considering the corresponding step computed on all the 110 VMs of the dataset.

We observe that the PCA-based and the Correlation-based approaches show lower execution times than the other techniques for every considered step. With regard to the Ensemble-based approach, we note that the execution times are particularly high in the case of the second and third contributions, due to the expensive computation of the Bhattacharyya distance and to multiple clustering steps. On the other hand, the proposed KL-based technique requires longer time than the Ensemble-based approach for the VM model computation, but is much faster on the other steps. Since the model computation corresponds to the only step of the methodology that can be parallelized on multiple nodes, we believe that the KL-based technique is a preferable choice to apply clustering in large data centers, while the Ensemble-based approach is better suited for smaller-sized infrastructures due to the high costs of the centralized steps.

5.6 Benefits for monitoring scalability

We now evaluate the reduction in terms of collected data that is achievable by integrating a clustering technique in a cloud monitoring system. For this experiment, we consider the Enterprise case study where the multi-tier Web application is deployed on 110 VMs, divided between Web servers and DBMS. If no clustering technique is applied, it is necessary to monitor every VM at fine-grained granularity to accomplish periodic consolidation tasks. Assuming that the monitoring system considers $\overline{K}$ resources for each VM, which 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 110 VMs, the total amount of data is in the order of $3.2 \times 10^4 \cdot \overline{K}$ samples per day. If we apply a clustering technique in the same scenario, the proposed methodology automatically identifies two sets of similar VMs (Web servers and DBMS) and monitors at the granularity of 5 minutes only a few representative VMs per class, while the remaining VMs can be monitored with a coarse-grained granularity, for example of 1 sample every few hours. Assuming to select 3 representatives for each of the 2 VM classes the amount of data to collect after clustering is reduced to $17.2 \times 10^2 \cdot \overline{K}$ samples per day for the class representatives; for the remaining 104 VMs, assuming to collect one sample of the $\overline{K}$ metrics every 6 hours for VM, the data collected is in the order of $4.2 \times 10^2 \cdot \overline{K}$ samples per day. Hence, we observe that our proposal may reduce the amount of data collected for periodic consolidation by nearly a factor of 15, from $3.2 \times 10^4 \cdot \overline{K}$ to $21.4 \times 10^2 \cdot \overline{K}$.

5.7 Summary of clustering comparison

Table 5 summarizes the characteristics of the clustering approaches. For each technique, we evidence with bold font the elements that represent potential drawbacks for the applicability and the performance achievable in a real cloud environment.

Table 5 Comparison of clustering approaches

Clustering Approach	Parameters	N. of Metrics	Execution Time
KL-based	None	Low	Medium
Ensemble-based	**# bins**	Low	**High**
PCA-based	**# components**	**High**	Low
Correlation-based	None	**High**	Low

From the table we note that the KL-based technique is not sensitive to any parameter. Moreover, its stable performance with respect to the number of considered metrics allows us to reduce the amount of monitored resources to describe the VM behavior. For these reasons and for its computational cost, this approach may be preferable with respect to the other alternatives. On the other hand, the Ensemble-based approach achieves slightly better performance than the proposed technique, but it is sensitive to the choice of the bin number for histograms computation and requires multiple clustering steps, that cause higher execution times. Hence, we can conclude that the KL-based approach is applicable to a wide range of scenarios, while the Ensemble-based technique may be a preferable solution for cases where the number of VMs is limited and the workload is stable to allow a tuning of the metric histogram bin numbers.

Our analysis also points out an unexpected result concerning VM clustering based on short monitoring periods of few hours in a dynamic scenario. As discussed in Section 5.3, in this case the performance of KL- and Ensemble-based techniques may be deteriorated if the workload presents daily patterns or fluctuations. On the other hand, PCA-based clustering achieves more stable results, but this technique is sensitive to the number of principal components and requires a high number of metrics. A recent effort to investigate the problem of clustering VMs in

dynamic scenarios relying on short monitoring periods has been done in [Canali and Lancellotti (2014a)], but it remains on open issue that needs further investigation.

6 Related Work

Scalability issues concerning resource monitoring and management in cloud systems have received a lot of attention by academic and commercial communities in the last few years, but only recent studies have explored solutions based on automatic clustering that take advantage from similarities between VMs sharing a common behavior.

The study in [Zhang et al. (2011)] proposes a method to automatically identify similarities between VMs in cloud systems that is based only on storage resources, with the goal to perform storage consolidation strategies. The study in [Jayaram et al. (2011)] investigates similarities in VM images used in public cloud environments; focusing on the static images of cloud VMs to provide insights for de-duplication and image-level cache management. These studies apply clustering to a very limited set of resources for specific purposes, while our approach considers several resources to model the general VMs behavior, and leverage similarities to improve scalability of cloud monitoring and management.

Recent studies [Canali and Lancellotti (2014c,d)] propose clustering techniques that group together cloud VMs with similar behavior on the basis of their resource usage. The technique proposed in [Canali and Lancellotti (2014c)] achieves very accurate VM clustering, but at the price of high computational costs. On the hand, in [Canali and Lancellotti (2014d)] much faster solutions were presented, but less accurate. Furthermore, all the techniques require some specific parameter that, if not correctly tuned, may result in poor performance. The proposed KL-based clustering overcomes the limitations of the previous approaches, being a non-parametric technique that does not need any specific tuning. A preliminary version of the KL-based clustering was originally presented in [Canali and Lancellotti (2014b)]. This study clearly extends the scope of the previous work for several reasons. We consider two different workloads with varying demands and shorter time series (up to 6 hours) to model the VMs behavior: this analysis allows us to give useful insights with respect to the performance of different clustering techniques in presence of specific conditions and workloads patterns. Moreover, we evaluate the reduction in terms of monitored data that is achievable by integrating the clustering technique into the cloud monitoring system.

The benefits of the proposed VM clustering techniques have been specifically evaluated in the context of cloud monitoring, with the aim to improve the scalability of the data collection process. A survey on monitoring solutions and available platforms may be found in [Aceto et al. (2013)]. In literature current approaches typically address monitoring scalability issues by propagating the collected data to the management process only after aggregation and filtering, in order to reduce their volume. Most of the proposed solutions adopt a subsystem to propagate data or rely on agents, which are responsible for performing data collection, filtering and aggregation [Mehrotra et al. (2011), Shao and Wang (2011), Azmandian et al. (2011), Kertesz et al. (2013), Andreolini et al. (2011)]. Different aggregation strategies have been proposed: extraction of high-level performance metrics by mean of machine learning algorithms [Shao and Wang (2011)]; extraction of predicted parameters by combining metrics from different layers (hardware, OS, application and user) and by applying Kalman filters [Mehrotra et al. (2011)]; linear combination of OS-layer metrics [Azmandian et al. (2011)]; extraction of high-level statistics from OS and application layers [Kertesz et al. (2013), Andreolini et al. (2011)].

Several open source platforms have also been proposed to monitor cloud systems. Among the most popular solutions, some examples deserving a mention are: Nagios, a well-known open source monitoring platform that has been extended to support the monitoring of cloud infrastructures both in terms of virtual instances and storage services [Katsaros et al. (2011)]; DARGOS, a distributed monitoring architecture using a push/pull approach to disseminate information [Povedano-Molina et al. (2013)]. All these solutions share a common limitation, that is considering each monitored object (being it a VM or a host) independent from the others, thus failing to take advantage from VMs similarities. Furthermore, it is worth to note that the proposed clustering technique may be integrated in any existing cloud monitoring solution as an additional step that selects the granularity of the data collection for different sets of VMs.

7 Conclusions

Previous studies in literature show that the automatic VMs clustering may improve the scalability of the monitoring process in large data centers. However, existing solutions are affected by some trade-offs regarding the computational costs, the accuracy of the results and the dependence on specific technique parameters. We propose a novel approach that exploits Mixture of Gaussians (MoGs) and Kullback-Leibler divergence to measure the similarity between VMs. The proposed KL-based approach is applied to two case studies and compared with the existing techniques. A wide range of experiments shows that the KL-based technique may guarantee results that are comparable with the best performing alternative and are stable thanks to its non-parametric approach. Moreover, the limited computational cost makes the proposed approach the preferable alternative in case of large cloud data centers.

References

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

Aceto, G., Botta, A., De Donato, W. and Pescapè, A.: 2013, Cloud Monitoring: A Survey, *Computer Networks* **57**(9), 2093–2115.

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

Andreolini, M., Colajanni, M. and Tosi, S.: 2011, A software architecture for the analysis of large sets of data streams in cloud infrastructures, *Proc. of the 11th IEEE International Conference on Computer and Information Technology (IEEE CIT 2011)*, Cyprus.

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

Azmandian, F., Moffie, M., Dy, J., Aslam, J. and Kaeli, D.: 2011, Workload characterization at the virtualization layer, *Proc. of IEEE 19th International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*, Singapore, pp. 63–72.

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

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

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

Canali, C. and Lancellotti, R.: 2014b, Balancing Accuracy and Execution Time for Similar Virtual Machines Identification in IaaS Cloud,, *Proc. of IEEE Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, Parma, Italy.

Canali, C. and Lancellotti, R.: 2014c, Exploiting Ensemble Techniques for Automatic Virtual Machine Clustering in Cloud Systems, *Automated Software Engineering* **21**(3), 319–344.

Canali, C. and Lancellotti, R.: 2014d, Improving Scalability of Cloud Monitoring Through PCA-Based Clustering of Virtual Machines, *Journal of Computer Science and Technology* **29**(1), 38–52.

Castro, M. and Liskov, B.: 1999, Practical Byzantine Fault Tolerance, *in* M. I. Seltzer and P. J. Leach (eds), *OSDI*, USENIX Association, pp. 173–186.

Dai, W., Liu, J. J. and Korthaus, A.: 2014, Dynamic on-demand solution delivery based on a context-aware services management framework, *Int'l Journal of Grid and Utility Computing (IJGUC)* **5**(1), 33–49.

Fraley, C., Raftery, A. and Scrucca, L.: 2013, *package mclust: Normal Mixture Modeling for Model-Based Clustering,Classification, and Density Estimation*.

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

Hershey, J. and Olsen, P.: 2007, Approximating the kullback leibler divergence between gaussian mixture models, *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, Vol. 4, pp. IV–317–IV–320.

Hu, L., Schwan, K., Gulati, A., Zhang, J. and Wang, C.: 2012, Net-cohort: detecting and managing VM ensembles in virtualized data centers, *Proc. of the 9th international conference on Autonomic computing (ICAC '12)*, ICAC '12, ACM, San Jose, California, USA, pp. 3–12.

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

Jayaram, K. R., Peng, C., Zhang, Z., Kim, M., Chen, H. and Lei, H.: 2011, An empirical analysis of similarity in virtual machine images, *Proc. of the Middleware 2011 Industry Track Workshop*, Middleware'11, ACM, Lisbon, Portugal, pp. 6:1–6:6.

Katsaros, G., Kubert, R. and Gallizo, G.: 2011, Building a Service-Oriented Monitoring Framework with REST and Nagios, *Proc. of 2011 IEEE International Conference on Services Computing (SCC)*, Washington DC, USA, pp. 426–431.

Kertesz, A., Kecskemeti, G., Oriol, M., Kotcauer, P., Acs, S., Rodriguez, M., Merce, O., Marosi, A., Marco, J. and Franch, X.: 2013, Enhancing Federated Cloud Management with an Integrated Service Monitoring Approach, *Journal of Grid Computing* **11**(4), 699–720.

Kullback, S.: 1997, *Information Theory and Statistics*, Dover Books on Mathematics, Dover Publications.

Mehrotra, R., Dubey, A., Abdelwahed, S. and Monceaux, W.: 2011, Large scale monitoring and online analysis in a distributed virtualized environment, *Proc. of 8th IEEE International Conference and Workshops on Engineering of Autonomic and Autonomous Systems*, Las Vegas, USA, pp. 1–9.

Moreno-Vozmediano, R., Montero, R. S. and Llorente, I. M.: 2013, Key Challenges in Cloud Computing: Enabling the Future Internet of Services, *IEEE Internet Computing* **17**(4), 18–25.

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

Povedano-Molina, J., Lopez-Vega, J. M., Lopez-Soler, J. M., Corradi, A. and Foschini, L.: 2013, Dargos: A highly adaptable and scalable monitoring architecture for multi-tenant clouds, *Future Generation Computer Systems* **29**(8), 2041 – 2056.

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

Shao, J. and Wang, Q.: 2011, A Performance Guarantee Approach for Cloud Applications Based on Monitoring, *Proc. of IEEE 35th Annual Computer Software and Applications Conference Workshops*, Munich, Germany, pp. 25–30.

Wood, T., Shenoy, P., Venkataramani, A. and Yousif, M.: 2007, Black-box and gray-box strategies for virtual machine migration, *Proc. of Conference on Networked systems design and implementation (NSDI)*, Cambridge.

Zhang, R., Routray, R., Eyers, D. M. et al.: 2011, IO Tetris: Deep storage consolidation for the cloud via fine-grained workload analysis, *IEEE Int'l Conference on Cloud Computing*, Washington, DC USA.