Balancing Accuracy and Execution Time for Similar Virtual Machines Identification in IaaS Cloud

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

***Abstract*—Identification of VMs exhibiting similar behavior can improve scalability in monitoring and management of cloud data centers. Existing solutions for automatic VM clustering may be either very accurate, at the price of a high computational cost, or able to provide fast results with limited accuracy. Furthermore, the performance of most solutions may change significantly depending on the specific values of technique parameters. In this paper, we propose a novel approach to model VM behavior using Mixture of Gaussians (MoGs) to approximate the probability density function of resources utilization. Moreover, we exploit the Kullback-Leibler divergence to measure the similarity between MoGs. The proposed technique is compared against the state of the art through a set of experiments with data coming from a private cloud data center. Our experiments show that the proposed technique can provide high accuracy with limited computational requirements. Furthermore, we show that the performance of our proposal, unlike the existing alternatives, does not depend on any parameter.**

***Keywords*—KL Divergence, VM Management, Clustering**

I. Introduction

Cloud computing is a fundamental resource for the development of novel services as it is a successful solution to cope with the growing and highly variable requests of processing power and storage capacity that characterize the evolution of the emerging digital society.

The most notable outcome of the cloud success is the ever-increasing size and complexity of cloud data centers hosting a constantly growing number of virtual machines (VMs), that are creating a challenge for the scalability of monitoring and managing tasks. Resource monitoring is particularly challenging in Infrastructure as a Service (IaaS) cloud systems, where several customer applications are hosted in virtualized environments. A customer application typically consists of multiple software components (e.g., the tiers of a multi-tier Web application), and each component runs on a separate VM. In these cloud systems, VMs are usually considered as black boxes with independent behaviors, hence information needs to be collected about each single VM of the data center, thus exacerbating the scalability issues of the monitoring task.

A solution recently proposed to address the scalability of monitoring in IaaS cloud system is based on clustering VMs that exhibit a similar behavior in terms of utilization of system resources [1]–[3]. In this case, fine-grained monitoring can be limited to a subset of representative VMs for each cluster, thus significantly reducing the amount of information collected by the monitoring system. Some existing solutions for automatic VM clustering are very accurate, but at the price of high computational costs [3]. Other solutions [1], [2] are much faster, but less accurate. Furthermore, all the techniques require some specific parameter or data filtering step that, if not correctly tuned, may result in poor performance.

In this paper we propose a novel approach, namely *KL-based* clustering, that exploits Mixture of Gaussians (MoGs) to model VM behavior and the Kullback-Leibler divergence to measure the similarity between VMs. To the best of our knowledge, this is the first technique that aims to provide fast and accurate VM classification without relying on any parameter or data filtering step.

We test our proposal against the existing solutions in a case study with data coming from a private cloud data center hosting a multi-tier e-health application, which is deployed on VMs running Web servers and DBMS. Our experiments achieve the double goal to assess the limitations of existing solutions and to evaluate our proposal. We demonstrate that the proposed 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 II describes the reference scenario for the application of VM clustering. Section III presents the proposed technique based on Kullback-Leibler divergence and Section IV describes the existing approaches. Section V provides the experimental comparison of our proposal with existing techniques for VM clustering, while Section VI concludes the paper with some final remarks.

II. Reference Scenario

We now describe the reference scenario for our proposal, where a IaaS cloud data center integrates a clustering technique [1]–[3] to improve the scalability of monitoring and management.

We assume that the IaaS cloud system adopts a two-level management strategy, as in [4]. The first level consists in a *local management*, that is performed on each physical server: it detects overload conditions in real-time making use of the resource measurements of the VMs hosted on the server, and exploits live VM migration whenever overloaded servers are detected [5]. The second level is a *global management*, which is controlled by a central node: it is responsible for periodically executing a consolidation technique to place VMs on as few physical servers as possible to reduce the infrastructure costs and avoid expensive resource over-provisioning [6], [7]. 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 [6], [7]) about each VM, thus creating scalability issues for the monitoring system.

A VM clustering technique may improve scalability of the monitoring process by automatically grouping together VMs showing similar behaviors in terms of resource usage [1]–[3]. The process of VM clustering is carried out periodically to identify classes of VMs that are running the same software component of the same customer application. Once the clustering is done, few representatives are selected for each identified class. 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 [8]. 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.

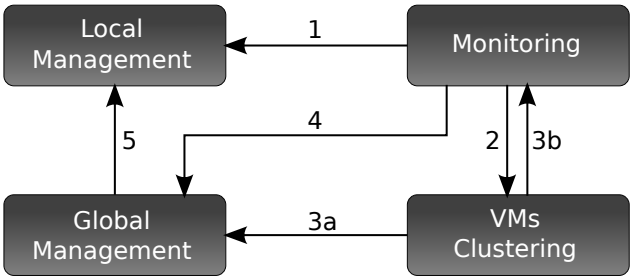

Fig. 1: Cloud system using VM clustering

Figure 1 depicts the interactions between the main components of the reference scenario. The *monitoring* system on each physical server collects data about resource usage of the hosted VMs and sends them to the *local management* system (arrow 1), which is responsible for triggering live VM migration in case of host overload [5]. Moreover, the *monitoring* system processes periodically send data to the *VM clustering* system (2), which automatically groups similar VMs applying one of the techniques proposed in [1]–[3]. The clustering results (identified VM classes and representatives) are sent to the *global management* (3a) and to the *monitoring* system (3b). The *monitoring* system exploits this information to differentiate the sampling frequency between representative and non representative VMs. The data collected with different granularity are sent to the *global management* system (4) which is responsible for two tasks. First, it periodically executes the cluster-based consolidation strategy, exploiting the resource usage of the representative VMs to characterize the behavior of every VM of the same class: the consolidation decisions are finally communicated to the *local management* systems (5) on each server to be executed. Second, the *global management* system checks for behavioral drifts of non representative VMs. It is worth to note that 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.

Thanks to differentiated sampling frequencies based on the knowledge of VM clusters, this approach may significantly reduce the amount of data collected for the global management of the cloud data center, as discussed in [1]–[3]. However, the solutions proposed in these studies present drawbacks related to the computational cost, the stability of the accuracy results or the dependence on specific parameter values. In the next section, we present a novel VM clustering technique that aims to provide high accuracy with limited computational requirements, and does not depend on any parameter.

III. KL-BASED CLUSTERING TECHNIQUE

Throughout this section we present the newly proposed KL-based technique for VM clustering. Before delving into the details of this technique, we introduce the main steps that are common to every solution for VM clustering. This approach has the twofold goal to (1) provide an overview of the structure of the proposed technique, and (2) provide a common background to compare our proposal with the existing alternatives described in Section IV. Specifically, we can divide any VM clustering technique in the following steps:

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

The steps of the KL-based technique are described in detail in the remaining of this section.

A. VM behavior quantitative model

Our proposal to describe the VM behavior is to model the usage of VM resources using 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 $\pi^n_{m,i}$, $\mu^n_{m,i}$, $\sigma^n_{m,i}$ 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 [9]. 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.

B. 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 [10]. 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, g_2 [11].

For MoGs, we can use an approximation, namely variational divergence [11], that extends Equation 1. The KL divergence for two Mixtures of Gaussian distributions mog_1 and $mog2$ 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:

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

C. Clustering

The last step of the technique aims to obtain 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 [12] 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 [13]. 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-grained granularity. To this purpose, it is worth to note that the output of the K-means internal phase of spectral clustering provides as additional output the coordinates of the centroids for each identified class. In this case, the representative VMs can be selected as the VMs closest to the centroids.

IV. Alternative clustering techniques

We now discuss briefly a few alternative approaches, discussing the different choices used to implement the three main steps previously outlined for a VM clustering technique.

A. Ensemble-based

The *Ensemble-based* technique, described in [3], exploits the probability density of the utilization of VM resources to model the VM behavior, as done in the KL-based approach. But, differently from the proposed technique, the behavioral model of a VM is described through a set of normalized histograms, one for each considered metric of the VM.

Then, the Ensemble-based approach compute a per-metric distance between every couple of VMs by exploiting the Bhattacharyya distance [14], which determines the distance between two different histograms. The use of this distance for VM clustering has been proposed for the first time in a preliminary paper [15].

This per-metric distance is used to create a set of distance matrices, that are used to compute separate per-metric clustering solutions. For this step we exploit 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.

B. Correlation-based

The *correlation-based* technique [1] describes the behavior of a VM using the correlation between the time series of different metrics of the same VM [1]. The correlation values for each couple of metrics (m_1, m_2) are then assembled into a single vector of length $\frac{M \cdot (M-1)}{2}$ that is the feature vector characterizing the VM. It is worth to note that, to improve the quality of the VM behavior representation, some pre-processing of the data may be necessary. In particular, previous experiments demonstrate that filtering the time series to remove

periods when the VMs are idle may significantly improve the clustering performance, especially when short time series are involved [1].

The output of the first step of the technique is a set of N feature vectors (one for each VM) that define a vector space of $\frac{M \cdot (M-1)}{2}$ dimensions. By defining the vector space we inherently define also a distance metric, that is the Euclidean distance between two feature vectors.

The set of N feature vectors describing the VMs is then fed into the clustering algorithm. For the clustering step we exploit a K-means algorithm [12].

C. PCA-based

The *PCA-based* technique [2] is an evolution of the previously described Correlation-based solution aiming to improve the quality of the clustering results.

Like the Correlation-based solution, this technique exploits the correlation values between each couple of time series referring to the M considered metrics of the same VM, and shares with the previous technique the need to filter the input time series to achieve good performance. 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 [16], we identify the number P of components that are significant to reconstruct the VM behavior (in our experiments only one component is sufficient to describe the VM behavior [2], so $P = 1$). 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 vector space with an Euclidean distance. Clustering is carried out with the K-means algorithm.

V. 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 a case study based on a dataset coming from an enterprise cloud data center. We start the technique evaluation with the discussion of the parameters that may affect the performance of each solution. Next, we provide an experimental comparison on the clustering results of the different approaches discussing the impact of data filtering, and we pass to analyze the sensitivity of the performance to the number of considered metrics. Finally, we compare the execution times of the different clustering approaches.

A. Case study

Our case study is based on a Web e-health application for the automated management of lab exams, which is hosted on a private data center and deployed on 110 VMs according to a multi-tier architecture. The 110 VMs are divided between the two software components of the Web application: Web servers and back-end servers (that are DBMS). We collect data about the resource usage of every VM for different periods of time, ranging from 1 to 5 days with a sampling frequency of 5 minutes. For each VM we consider 10 metrics describing the usage of several resources, including CPU, memories, disk, network, and number of active processes.

The final goal of VM clustering applied to this case study is to correctly classify the VMs in two groups: Web server and DBMS. To evaluate the performance of VM clustering we consider a widely used measure, namely *purity* [17], 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.

B. Comparison of Clustering Approaches

We now compare the different clustering techniques applied to our case study. In particular, we consider the proposed KL-based methodology and three existing approaches: Ensemble-based [3], PCA-based [2] and Correlation-based [1].

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 [3]. 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 IV. Again, results in literature demonstrate that the number of considered components may affect the clustering performance [2]. Finally, the Correlation-based solution is not dependent on any parameter, like the proposed KL-based technique [1].

The second comparison is based on a quantitative evaluation of the achieved clustering performance. Figure 2 shows the clustering purity as a function of the time series length for the different clustering techniques. In the legend we indicate an "(F)" next to PCA-based and Correlation-based approaches to recall that these techniques require a pre-filtering of data to eliminate idle periods from the time series, as described in Section IV. Otherwise, the performance of these approaches drastically decreases, as shown in Table I, where we present the clustering purity achieved without applying any data filtering.

Despite the application of data filtering, we see from Figure 2 that the performance of the Correlation-based approach significantly decreases for shorter time series (1-2 days). On the other hand, the PCA-based technique remains quite stable

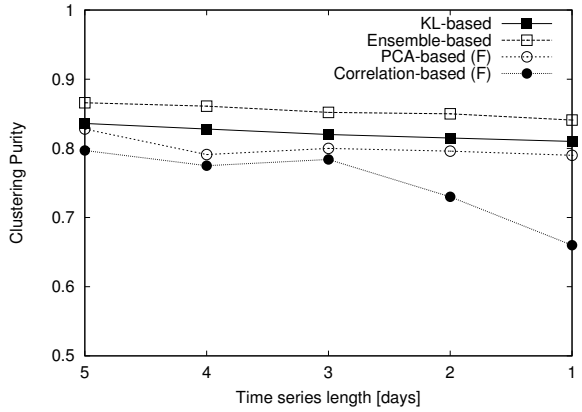

Fig. 2: Clustering purity for alternative approaches

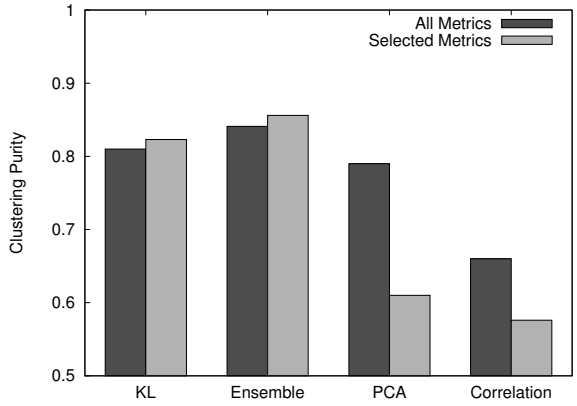

Fig. 3: Clustering purity for different sets of metrics

TABLE I: Clustering purity without data filtering

Clustering Approach	Time series length [days]				
	5	4	3	2	1
PCA-based (No F)	0.708	0.678	0.680	0.629	0.641
Correlation-based (No F)	0.670	0.662	0.598	0.575	0.571

for every time series length, while 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%.

C. Sensitivity to number of metrics

The number of metrics used to determine the VM behavior model may affect the performance of the VM clustering. 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 four metrics mostly used in data center management strategies [4], [6], [18]: 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 [15]: 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 3 shows the purity of the clustering approaches for the entire set of 10 metrics and for the reduced set of four selected metrics. We observe that the number of metrics has very different impacts on the performance of the considered approaches. 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.

D. 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 III: 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 II shows the execution times of the three contributions for the considered clustering techniques.

TABLE II: Time for clustering approaches

Clustering Approach	Time [s]		
	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.

E. Summary of results

Table III summarize the characteristics of the alternative 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 III: Comparison of clustering approaches

Clustering Approach	Parameters	Data Filtering	N. of Metrics	Execution Time
KL-based	None	Not Required	Low	Medium
Ensemble-based	**# bins**	Not Required	Low	**High**
PCA-based	**# components**	**Required**	**High**	Low
Correlation-based	None	**Required**	**High**	Low

From the table we note that the KL-based technique is not sensitive to any parameter, and does not require data preprocessing. 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. It is worth to recall that the Ensemble-based approach achieves slightly better performance than the proposed technique, but it is sensitive to the choice of the number of bins for histograms computation and requires multiple clustering steps that cause higher execution times. 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.

VI. Conclusions

Previous studies of the authors 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 a real dataset coming from an enterprise data center 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

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

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

[3] ——, "Exploiting ensemble techniques for automatic virtual machine clustering in cloud systems," *Automated Software Engineering*, pp. 1–26, 2013, Available online.

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

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

[6] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang, "Energy-Aware Autonomic Resource Allocation in Multitier Virtualized Environments," *IEEE Trans. on Services Computing*, vol. 5, no. 1, pp. 2 –19, Jan. 2012.

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

[8] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in *OSDI*, M. I. Seltzer and P. J. Leach, Eds. USENIX Association, 1999, pp. 173–186.

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

[10] S. Kullback, *Information Theory and Statistics*, ser. Dover Books on Mathematics. Dover Publications, 1997.

[11] J. Hershey and P. Olsen, "Approximating the kullback leibler divergence between gaussian mixture models," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, April 2007, pp. IV–317–IV–320.

[12] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651 – 666, 2010.

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

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

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

[16] H. Abdi and L. Williams, "Principal component analysis," *Computational Statistics*, 2010, in press.

[17] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, "A Comparison of Extrinsic Clustering Evaluation Metrics Based on Formal Constraints," *Journal of Information Retrieval*, vol. 12, no. 4, pp. 461–486, Aug. 2009.

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