# Automated clustering of VMs for scalable cloud monitoring and management

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

***Abstract*—The size of modern datacenters supporting cloud computing represents a major challenge in terms of monitoring and management of system resources. Available solutions typically consider every Virtual Machine (VM) as a black box each with independent characteristics and face scalability issues by reducing the number of monitoring resource samples, considering in most cases only average CPU utilization of VMs sampled at a very coarse time granularity. We claim that better management without compromising scalability could be achieved by clustering together VMs that show similar behavior in terms of resource utilization. In this paper we propose an automated methodology to cluster VMs depending on the utilization of their resources, assuming no knowledge of the services executed on them. The methodology considers several VM resources, both system- and network-related, and exploits the correlation between the resource demand to cluster together similar VMs. We apply the proposed methodology to a case study with data coming from an enterprise datacenter to evaluate the accuracy of VMs clustering and to estimate the reduction in the amount of data collected. The automatic clustering achieved through our approach may simplify the monitoring requirements and help administrators to take decisions on the management of the resources in a cloud computing datacenter.**

## 1. Introduction

In the few last years, cloud datacenters running different applications have become popular in a variety of domains, such as Web hosting, enterprise systems, and e-commerce sites. Modern datacenters typically run thousands of different applications with complex and heterogeneous resource demand behavior. Virtualization has been increasingly used to run multiple applications in virtual machines (VMs), that are jointly hosted on physical servers and share server resources over time.

The processes of monitoring and managing such infrastructures, that are typically composed of thousands of nodes, each hosting tens or hundreds of VMs, are extremely complex. The monitoring of such infrastructures is likely to present scalability issues due to the amount of data to collect and store when a large number of VMs are considered each with several resources monitored with an high sampling frequency [1]. Also efficient management of VMs in a datacenter (for example, through periodic consolidation of VMs) do not scale well due to the large amount of data to analyze [16]. The scalability issue is particularly difficult to tackle because 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 virtual machine [17], [20]. Hence, most monitoring and management strategies in cloud datacenters assume that each VM is a single object whose behavior is independent from the other VMs of the cloud infrastructure. To reduce the complexity of VM monitoring and management problem, the typical approach in IaaS cloud is to reduce the problem size. To this aim, available solutions may rely on low sampling frequency of VM status or reduce the number of VM resources that are taken into account, typically considering only CPU-related information [2], [6], [16]. However, these approaches are likely to suffer important drawbacks: on one hand, reducing the sampling frequency leads to low reactivity to changes in demand; on the other hand, limiting the monitoring to the CPU resource may not be sufficient to capture changes in the behavior of VMs running I/O bound or network bound applications.

We argue that the scalability of monitoring and management tasks in cloud infrastructures may be improved by leveraging the similarity between VM behaviors, considering VMs not as single objects but as members of a class composed by objects with a similar behavior (e.g., Web servers or DBMS). Indeed, once we have identified classes of similar VMs, we may select a few representative VMs for each class and carry out monitoring at a detailed level only on these representatives. Other VMs can be monitored at a much coarser granularity level with the goal to discover if their behavior is changing with respect to the class they belong to. This coarse-grained approach can easily reduce the amount of data collected by one order of magnitude with respect to the fine-grained monitoring of every single VM, improving the scalability of cloud monitoring and management.

The main contribution of this paper is the proposal of an automated, non parametric methodology to cluster together similar VMs in an IaaS cloud datacenter on the basis of their behavior. The proposed methodology exploits the correlation between the usage of several VM resources to determine which VMs are following the same behavioral patterns. To the best of our knowledge, this is the first paper that proposes to automatically cluster together VMs with similar behavior starting from the usage information of several resources for management purposes. A main advantage of our methodology is that we take into account multiple resources, including network and memory related information, differently from most of the current solutions that mainly consider CPU-related information. We apply the proposed methodology to a dataset coming from a real cloud environment with VMs hosting Web servers and DBMS. We demonstrate that

our methodology can achieve an accuracy in clustering VMs that is between 85% and 100% for every considered scenario, with a reduction in the amount of collected data samples by a factor of 20.

The remainder of this paper is organized as follows. Section 2 describes the proposed methodology for clustering similar VMs in a cloud environment. Section 3 presents the case study used to evaluate our methodology and describes the results of our experiments. Section 4 discusses the related work and Section 5 concludes the paper with some final remarks.

## 2. METHODOLOGY

In this section we describe the proposed methodology to automatically cluster similar VMs in a cloud datacenter. The methodology consists in the following steps:

- Extraction of a quantitative model for describing the VM behavior.
- Clustering based on the VM behavior model to identify classes of similar VMs

Given a set of $n$ VMs, the first step of the methodology aims at representing the behavior of each VM $i, \forall i \in [1, n]$. We consider that capturing the inter-dependencies among the usage of different resources, such as CPU utilization, network throughput or I/O rate, can describe the VM behavior during a period of time. For example, in Web servers network usage is typically is related to the CPU utilization [3], while for DBMS CPU utilization tends to change together with storage activity [9]. We consider each VM as described by a set of $m$ metrics, where each metric $j \in [1, m]$ represents a resource of the VM.

Let $(X_1^i, X_2^i, \ldots, X_n^i)$ be a set of time series, where $X_j^i$ is the vector of the samples of the time series describing the metric $j$ of VM $i$. The inter-dependencies between VM metrics are measured using correlation values, that can be represented as elements of the correlation matrix $S^i$, where $s_{j,k}^i = \mathrm{cor}(X_j^i, X_k^i)$ is the correlation coefficient between the two time series $X_j^i$ and $X_k^i$.

The correlation matrix $S^i$ describing the behavior of the VM $i$ is given as input to the second step of the methodology, that aims to group similar VMs into classes. Starting from the matrix $S^i$, we build a *feature vector* $V^i$ that is fed into a clustering algorithm. Clustering algorithms typically have a computational complexity that grows with the size of the feature vector, hence the performance of the clustering task can be reduced by avoiding redundancies in the $V^i$ vector. To this aim, we exploit the symmetric nature of the matrix $S^i$ and the fact that the main diagonal is composed of "1" to reduce the length of $V^i$. We define $V^i$ as $V^i = (s_{2,1}^i, s_{3,1}^i, s_{3,2}^i, \ldots, s_{m,1}^i, \ldots, s_{m,m-1}^i)$.

The feature vector $V^i$ is used by the clustering algorithm as the coordinate of VM $i$ in the feature space. We define $C$ as the vector resulting form the clustering operation. The $i$-th element of vector $C$, $c^i$, is the number of the cluster to which VM $i$ is assigned. Many algorithms are available for clustering, starting from the simple and widespread *k-means* to more complex kernel-based solutions, up to clustering based on spectral analysis [8]. In this first proposal of a methodology to automatically cluster similar VMs, we adopt one of the most popular solutions for clustering, that is the k-means algorithm, leaving the comparison of different clustering algorithms as a future work. However, we are aware that a problem left unaddressed by the k-means algorithm is the automatic identification of the number of cluster to be used. More advanced techniques, such as techniques based on the Spectral Analysis, can be used to this aim [14].

Once the clustering is complete, we can select some representative VMs for each class to 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 this case, the representative VMs can be selected as the VMs closest to the centroids. The choice to consider more than one representative for each class is due to the possibility that a selected class representative changes its behavior with respect to the class it belongs to. When more than one representative is used, quorum-based techniques can be exploited to identify a misbehaving VM within the list of representatives.

## 3. CASE STUDY

To evaluate the results of the proposed methodology, we consider a case study based on a dataset coming from an enterprise datacenter supporting Web applications deployed according to a multi-tier architecture. The datacenter is composed of 10 nodes on a Blade-based system and exploits virtualization to support Web applications. The nodes host 110 VMs that are divided between Web servers and back-end servers (that are DBMS).

We collect detailed data about the resource usage of every VM for a short period of time. The samples are collected with a frequency of 5 minutes. For each VM we consider 11 metrics describing the usage of different resources (such as CPU, memory, disk, and network). The complete list of the metrics is provided in Table 1 along with a short description.

Table 1
VIRTUAL MACHINE METRICS

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

It is worth to note that, to collect data about 11 VM metrics with a frequency of 1 sample every 5 minutes, we need to manage a volume of data in the order of $16 \times 10^3$ samples per day per VM. Considering a datacenter hosting 110 VMs, the total amount of data is in the order of $1.7 \times 10^6$ samples per day. After the clustering, we need to continue monitoring every 5 minutes only a few representatives per class, while the remaining VMs can be monitored with a coarse time granularity, for example of 1 sample every few hours. Assuming to select 3 representatives for each of the 2 classes, the amount of data to collect after clustering is reduced to $95 \times 10^3$ samples per day for the class representatives; for the remaining 104 VMs, assuming to collect 1 sample every 6 hours for VM, the data collected are reduced to $4.5 \times 10^3$ samples per day. From this example we observe that our proposal may reduce the amount of data collected by nearly a factor of 20, from $1.7 \times 10^6$ to $99 \times 10^3$.

Let us now describe the application of how methodology to the considered case study. For each considered VM, we compute the correlation between couples of measured metrics. As discussed in Section 2, the resulting correlation matrix is used to build a feature vector describing the VM behavior, that is used for the subsequent VM clustering step. Then, we compare the results of the clustering step with the actual clusters (we consider that Web servers and DBMS servers are divided into two different clusters) and we evaluate the percentage of correctly identified VMs. As the used k-means clustering algorithm starts each run with a set of randomly-generated cluster centroids, we carry out the clustering step $10^4$ times to have a statistically significant set of data and we consider the average fraction of correctly identified VMs over the different iterations as a measure of the clustering *accuracy*. Specifically, we define accuracy by comparing the output of the clustering algorithm $C$ with the vector $C^*$ representing the correct clustering solution. Accuracy is thus defined as:

$$accuracy = \frac{|\{c^i : c^i = c^{i*}, \forall i \in [1, n]\}|}{|C|}$$

In our experiments, we evaluate the accuracy and the scalability of the clustering as a function of: (1) the length of the time series expressing the metric sampling; (2) the number of VMs. In the first analysis, we consider time series length ranging from 5 to 50 days. The time required for the clustering on an Intel Xeon 2GHz node is in the order of less than 2 minutes for every considered dataset. The results in our experiments show that the actual analysis of data for clustering VMs into classes poses no performance issue for the datacenter of our case study.

The histogram in Figure 1 presents the clustering accuracy as a function of the time series length. We show that, given a very long time series, the clustering is perfect, that is every Web server and every DBMS is correctly identified. On the other hand, the accuracy significantly decreases as we reduce the amount of data used to create the correlation matrix. In particular, when the time series is below 20 days, the accuracy is below 0.7, reaching 0.65 for a time series of only 5 days.

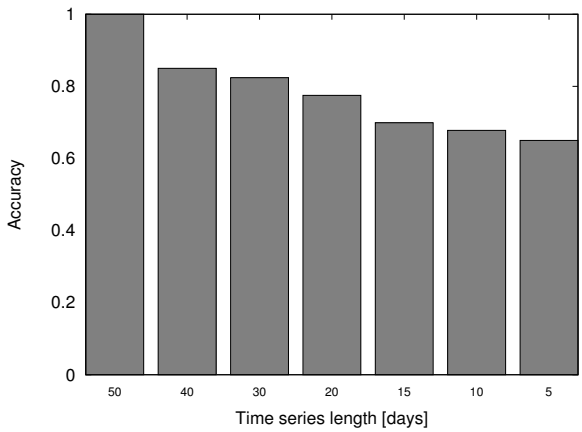


Figure 1. Accuracy for different time series length

Analyzing the reasons for the poor performance obtained with time series shorter than 20 days, we observe that some VMs present a bimodal behavior, with periods where the VM is mostly idle mixed with periods where the VM is heavily utilized. When we consider short time series (e.g., 5 days), we notice some time series composed almost exclusively by idle periods. This is the reason for the poor performance of the methodology: during the idle periods, the correlation between the metrics describing the VM is significantly reduced, thus leading to wrong clustering. To avoid this effect, we consider a different approach, where we filter the time series in order to extract a sequence of samples with no idle periods in between. In Figure 2, we compare the results of our filtered data against the time series of the same length used previously.

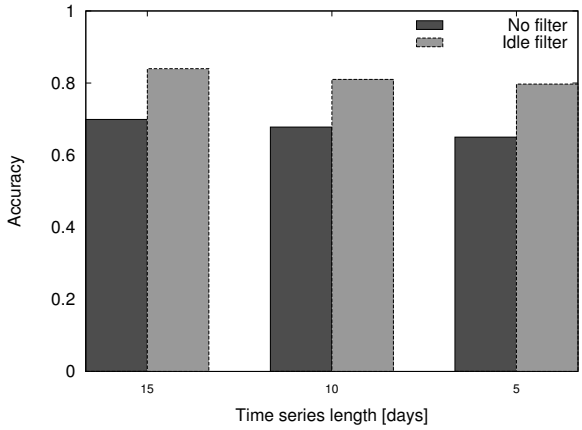


Figure 2. Impact of idle data filtering on clustering accuracy

Finally, we evaluate whether the accuracy of the clustering process may vary depending on the number of considered VMs. Table 2 shows how accuracy changes as the number of VMs considered in the clustering grows from 10 to 110, considering time series of 15 days. We observe that, with the notable exception of the simplest case of only 10 VMs, the accuracy of the clustering is mostly unaffected by the number of elements to cluster.

This result is important, because it shows the viability of the proposed methodology even in the case of large datacenters. Furthermore, the rightmost column of Table 2 presents the time for the clustering as a function of of the number of VMs considered. The results show a linear growth in the time required for clustering that proves the scalability of our solution for IaaS cloud datacenters.

Table 2
IMPACT ON ACCURACY OF VM NUMBER

| Number of VMs | Accuracy | Clustering time [s] |
|---|---|---|
| 10 | 1 | 49.7 |
| 30 | 0.86 | 59.5 |
| 50 | 0.84 | 68.6 |
| 70 | 0.84 | 78.0 |
| 90 | 0.83 | 88.3 |
| 110 | 0.84 | 95.3 |

## 4. RELATED WORK

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

Many existing studies propose resource management strategies based on the usage of one or few resources compared against thresholds. For example, the studies in [2] and [6] propose solutions for consolidation of virtual machines based on adaptive thresholds regarding the CPU utilization values. Wood *et al.* [21] propose a reactive, rule-based approach for virtual machine migration that defines threshold levels regarding the usage of few specific physical server resources, such as CPU-demand, memory allocation, and network bandwidth usage. Kusic *et al.* [10] address the issue of virtual machine consolidation through a sequential optimization approach; the drawback is that the proposed model requires simulation-based learning and the execution time grows very fast even with a limited number of nodes. All these studies perform a per-node analysis based on the usage of one or few resources; however, these approaches are likely to suffer from scalability issues in large scale distributed systems, such as IaaS cloud computing datacenters.

Few recent studies aim to reduce the dimensionality of the resource management problem, such as [16], [15], [18]. The studies in [16], [15] exploit a statistical analysis based on Singular Value Decomposition (SVD) to predict the workload demand aggregated on different virtual machines to anticipate overload conditions on physical servers and trigger virtual machine migrations. Tan *et al.* [18] apply Principal Component Analysis (PCA) to evaluate resource usage patterns across different nodes. The proposal consists in placing on the same physical server virtual machines with negatively correlated resource patterns to reduce the usage variability on the servers. All these studies have a different goal with respect to our paper, because they address the specific problem of virtual machine consolidation in cloud datacenters. Moreover, all their solutions consider only one resource, that is the CPU utilization of virtual machines, while we aim to support management strategies that consider multiple resources, from CPU to network and disks.

As regards the issue of monitoring large datacenters, current solutions can be divided into log aggregators and frameworks for periodic collection of system status indicators. Among log aggregators, often called also log collectors, the most widespread solution is the Syslog daemon, with its recent extension [7] that was explicitly designed to be used by cloud entities or applications to log and trace activities occurring in the cloud. Solutions such as Cacti and Munin [11], [12] are more oriented towards the periodic collection of data. Cacti is an aggregator of data transferred through the SNMP protocol, while Munin is a monitoring system based on a proprietary local agent interacting with a central data collector. Both these solutions are typically oriented to medium to small datacenters because of their centralized architecture that limits the overall scalability of the data collection process.

Ganglia [5] 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 datacenters [4], [13], even in cloud infrastructures [19], by storing the behavior of nodes and virtual machines by organizing the data in time series. Another solution for scalable monitoring is proposed in [1], where data analysis based on the map-reduce paradigm is distributed over the levels of a hierarchical architecture to allow only the most significant information to be processed at the root nodes. However, all these solutions share the same limitation of considering each monitored object (being it a VM or a host) independent from the others. This approach fails to take advantage from the similarities of objects sharing the same behavior. On the other hand, a class-based monitoring system may perform a fine-grained monitoring for only a subset of objects that are representative of a class, while other members of the same class can be monitored at a much more coarse-grained level. We believe that integrating our solution into existing hierarchical models for monitoring can significantly improve the scalability of monitoring operations.

## 5. CONCLUSIONS AND FUTURE WORK

Modern datacenters supporting IaaS cloud represent a major challenge for the monitoring and management of resource utilization, mainly due to scalability issues. In this paper we propose a methodology for automatically clustering VMs into classes that share similar behavior aiming to improve the scalability of monitoring and management. This approach exploits information about the correlation among usage of multiple resources ranging from CPU to storage and network. The application of the proposed methodology to a real datacenter hosting multi-tier Web applications shows that the accuracy of VMs clustering ranges between 100% and 85% for every considered scenario and can reduce the amount of data

collected by a factor of 20.

This study is just a first step towards the definition of a general methodology for the automated classification of VMs in cloud datacenters. Future work includes a comparison of multiple clustering algorithms, the proposal of new metrics to identify the similarity of VMs, and solutions to improve the scalability of the clustering process in case of high-dimensionality feature vectors.

## REFERENCES


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

[2] A. Beloglazov and R. Buyya. Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centers. In *Proc. of (MGC'10)*, Bangalore, India, Dec. 2010.

[3] E. Cecchet, A. Chanda, S. Elnikety, J. Marguerite, and W. Zwaenepoel. Performance Comparison of Middleware Architectures for Generating Dynamic Web Content. In *Proc. of the 4th International Middleware Conference*, Jun. 2003.

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

[5] Ganglia Monitoring System, 2012. – http://ganglia.sourceforge.net/.

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

[7] G. Golovinsky, S. Johnston, and D. Birk. Syslog Extension for Cloud Using Syslog Structured Data. Internet-Draft, March 2011.

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

[9] S. Kavalanekar, D. Narayanan, S. Sankar, E. Thereska, K. Vaid, and B. Worthington. Measuring database performance in online services: A trace-based approach. In R. Nambiar and M. Poess, editors, *Performance Evaluation and Benchmarking*, pages 132–145. Springer-Verlag, Berlin, Heidelberg, 2009.

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

[11] S. M. I. Lavlu and D. Kundu. *Cacti 0.8 Network Monitoring*. Packt Publishing, 2009.

[12] The Munin Monitoring Homepage, 2012. – http://munin-monitoring.org/.

[13] A. N. Naeem, S. Ramadass, and C. Yong. Controlling Scale Sensor Networks Data Quality in the Ganglia Grid Monitoring Tool. *Communication and Computer*, 7(11):18–26, Nov. 2010.

[14] G. Sanguinetti, J. Laidler, and N. Lawrence. Automatic determination of the number of clusters using spectral algorithms. In *Machine Learning for Signal Processing, 2005 IEEE Workshop on*, pages 55 –60, sept. 2005.

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

[16] T. Setzer and A. Stage. Filtering multivariate workload non-conformance in shared IT-infrastructures. In *Proc. of IFIP/IEEE International Symposium on Integrated Network Management (IM'11)*, Dublin, Ireland, may 2011.

[17] R. Singh, P. J. Shenoy, M. Natu, V. P. Sadaphal, and H. M. Vin. Predico: A System for What-if Analysis in Complex Data Center Applications. In *Proc. of 12th International Middleware Conference*, Lisbon, Portugal, Dec. 2011.

[18] J. Tan, P. Dube, X. Meng, and L. Zhang. Exploiting Resource Usage Patterns for Better Utilization Prediction. In *Proc. of the 31st International Conference on Distributed Computing Systems Workshops (ICDCSW'11)*, Minneapolis, USA, Jun. 2011.

[19] C.-Y. Tu, W.-C. Kuo, W.-H. Teng, Y.-T. Wang, and S. Shiau. A Power-Aware Cloud Architecture with Smart Metering. In *Proc. of 39th International Conference on Parallel Processing Workshops (ICPPW'10)*, San Diego, CA, Sept. 2010.

[20] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif. Black-box and gray-box strategies for virtual machine migration. In *Proceedings of the 4th USENIX conference on Networked systems design and implementation*, NSDI'07, Cambridge, MA, Apr. 2007.

[21] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif. Black-box and gray-box strategies for virtual machine migration. In *Proc. of the 4th USENIX Conference on Networked systems design and implementation*, NSDI'07, Cambridge, MA, Apr. 2007.