Identifying Communication Patterns between Virtual Machines in Software-Defined Data Centers

Claudia Canali
Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia
Via P. Vivarelli 10
41125, Modena, Italy
claudia.canali@unimore.it

Riccardo Lancellotti
Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia
Via P. Vivarelli 10
41125, Modena, Italy
riccardo.lancellotti@unimore.it

ABSTRACT

Modern cloud data centers typically exploit management strategies to reduce the overall energy consumption. While most of the solutions focus on the energy consumption due to computational elements, the advent of the Software-Defined Network paradigm opens the possibility for more complex strategies taking into account the network traffic exchange within the data center. However, a network-aware Virtual Machine (VM) allocation requires the knowledge of data communication patterns, so that VMs exchanging significant amount of data can be placed on the same physical host or on low cost communication paths. In Infrastructure as a Service data centers, the information about VMs traffic exchange is not easily available unless a specialized monitoring function is deployed over the data center infrastructure. The main contribution of this paper is a methodology to infer VMs communication patterns starting from input/output network traffic time series of each VM and without relaying on a special purpose monitoring. Our reference scenario is a software-defined data center hosting a multi-tier application deployed using horizontal replication. The proposed methodology has two main goals to support a network-aware VMs allocation: first, to identify couples of intensively communicating VMs through correlation-based analysis of the time series; second, to identify VMs belonging to the same vertical stack of a multi-tier application. We evaluate the methodology by comparing different correlation indexes, clustering algorithms and time granularities to monitor the network traffic. The experimental results demonstrate the capability of the proposed approach to identify interacting VMs, even in a challenging scenario where the traffic patterns are similar in every VM belonging to the same application tier.

1. INTRODUCTION

The need to reduce the energy consumption of large scale cloud data centers is attracting a lot of attention towards energy-efficient strategies for resource management and VMs allocation. Most research studies merely consider VMs allocation as a way to reduce the number of active servers [5, 8, 7, 15], while more recent proposals also take into account the energy consumption related to the data transfers among VMs, for example with the aim to co-locate on the same server VMs that intensively communicate among themselves [14, 18].

The recent advent of Software-Defined Data Centers (SDDCs), where all elements of the infrastructure – including networking – are virtualized as software components and the separation of the network control plane from the data plane allows flexible network management and reconfiguration, gave a further impulse to energy-efficient strategies taking into account data traffic exchanges within the data center. However, the adoption of a network-aware VMs allocation strategy requires the knowledge about the communication patterns between VMs. In this paper, we consider the point of view of the provider of an Infrastructure as a Service (IaaS) data center, where each VM is seen as a black box without any further knowledge of the software running on it. In this scenario, obtaining a data transfer matrix between the VMs is not possible unless a specialized monitoring infrastructure is developed and deployed over the data center, as in [16, 19]. The most popular monitoring services in industry[1] and literature [3], indeed, just provide the input/output traffic rate of each VM, without a per-source/per-destination breakdown. Hence, specific strategies are required to infer the VMs communication patterns starting from these low level information.

In order to address this issue, we propose a novel approach explicitly aiming to identify interacting VMs in the case multi-tier applications are horizontally replicated over a SDDC, as shown in Fig. 1. The tiers of each application are organized in vertical stacks that are replicated over the data center; each tier of an application is hosted on a separate VM, and we assume that each VM communicates only with other VMs within the same vertical stack, while no communication between VMs belonging to two different vertical stacks occurs. In this scenario, a request dispatcher typically distributes the incoming workload among the vertical stacks of the application. It is important to note that the presence of the load-balancing request dispatcher leads to very similar utilization and communication patterns of VMs belonging to the same tier but to different vertical stacks of the same application (e.g., VMs VM1 and VM2 in the figure). This similarity represents one of the most challenging aspects for the problem of identifying communicating VMs, because input/output traffic patterns may be correlated even if no communication occurs between them.

The main contribution of this paper is the proposal of a methodology that, starting from the time series of the aggregated network traffic of each VM, is able to identify which VMs intensively communicate each other and which ones belong to the same vertical stack of the applications deployed over a SDDC. Both these pieces of information are extremely valuable as input to VMs allocation strategies to reduce the energy consumption due to the network traffic exchange. Ideally, indeed, VMs exchanging large amount of data and/or belonging to the same vertical stack of an application should be placed on the same physical host or on hosts connected by low cost links to reduce energy consumption. The pro-

[1] https://aws.amazon.com/cloudwatch/

posed methodology exploits correlation techniques to identify co-occurring similarities in the communication patterns of different VMs, and clustering algorithms to identify VMs belonging to the same vertical stacks of an application. In this paper, we compare the use of different correlation indexes and clustering algorithms applied to VMs network utilization time series collected with varying time granularities in order to evaluate the capability of the proposed methodology of identifying the interacting VMs. Our experiments, based on the deployment of a benchmark over a cloud infrastructure, show that the analysis of the correlation among VMs input/output time series and the application of clustering algorithms represent a promising way to identify VMs that intensively communicate within a SDDC.

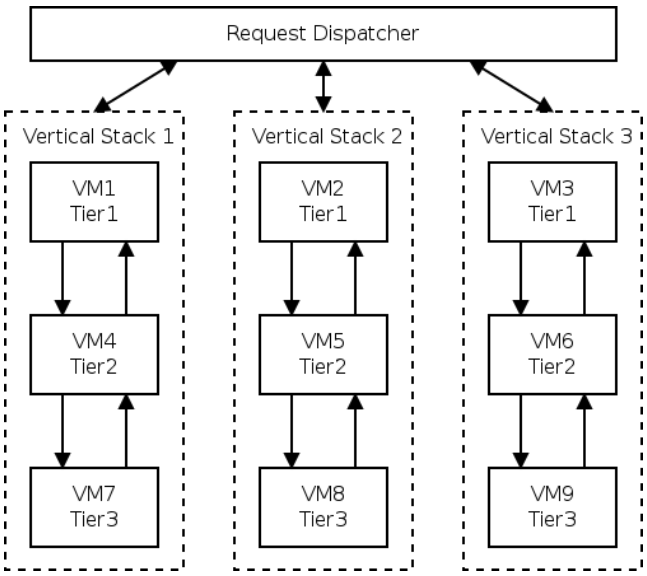


Figure 1: Multi-tier application deployed over a cloud data center

The remainder of this paper is organized as follows. Section 2 describes the architecture of a Software-Defined Data Center. Section 3 models the problem and outlines the proposed methodology. Section 4 describes the experimental results used to validate our proposal. Finally, Section 5 concludes the paper with some final remarks.

2. SOFTWARE-DEFINED DATA CENTER

Fig. 2 shows the structure of a Software-Defined Data Center (SDDC) where virtualized physical hosts run multiple VMs (that are logically organized into applications, tiers and vertical stacks as in Figure 1); the figure represents the physical infrastructure of the data center and the logical organization of the management processes.

The physical infrastructure is divided into manageable subareas, called PODs [4], representing building blocks including physical hosts connected to first level network switches; these switches are in turn connected to a Data Center Network Core, that may be contain more levels of switches. This network structure represents a typical fat-tree topology, which is one of the most popular used in both traditional and Software-Defined data centers because it can provide fault tolerance and scalability thanks to the ability to exploit multiple paths between end-nodes [1], as shown in Fig. 2.

Within each physical host, multiple VMs are running different tiers of the software applications. The VMs are connected by a virtual network to the physical infrastructure. On each host we have a monitoring subsystem, typically operating at the level of the virtualized hypervisor, that periodically collects resources usage time series for each VM (e.g., CPU, memory, input/output). The collected information usually includes also aggregated data about the input and output network traffic of each VM. In a IaaS data center, this is the only information available about the traffic exchange of a VM.

The dashed box on the left side of the figure describes the Data Center Management and the related components. This block receives two types of input. First, the VMs Monitor process receives the data collected by the hypervisors located on each physical host: these data include the usage time series for different resources (CPU, memory, input/output and network traffic). Second, the Network Monitor receives the information coming from the data plane of the data center network infrastructure, that is the set of network switches. All these collected data are sent to the core components of the Data Center Management of a SDDC, which are the VMs Manager and the Network Manager. The former is responsible for running the strategies for VMs allocation, that are communicated to the hypervisors local to each host; it is worth to note that, differently from VMs allocation based only on computational resources, network-aware allocation strategies require the VMs Manager to take into account also information about the network traffic exchange between VMs. The Network Manager includes the control plane, which is responsible for implementing strategies of network management and reconfiguration, that are communicated to the data plane; this separation between control and data plane introduces the benefits of a centralized approach to network configuration, that is programmable at the software level rather than manually reconfigurable through distributed low-level interfaces [10].

From an energy point of view, SDDCs realize a more seamless integration of the network within the data center IT processes, opening up to the possibility of novel energy-efficient resource strategies for the cloud infrastructures integrating complex and adaptive network management. Network-aware VMs allocation strategies to reduce energy consumption in this kind of data centers not only aim to minimize the number of physical host turned on, but also to place strongly interacting VMs on low cost links. The preferable solution from an energy point of view for couples of VMs characterized by an intense traffic exchange is to be placed on the same physical host, and secondarily on low cost links (e.g., within the same POD). Other strategies to save energy due to network traffic exchange are based on the possibility to take advantage of underused links to perform traffic consolidation in few active communication links and network devices [12, 20]. To implement these strategies, the SDDC management needs not only information about the input/output traffic of each VM but also to know which couples of VMs are intensively communicating with each other and/or which VMs are running the different tiers of the same vertical stack of an application. The proposed methodology to infer these information from the aggregated data on the input and output network traffic of each VM is presented in the next section.

3. METHODOLOGY DESCRIPTION

In our reference scenario, the SDDC hosts multi-tier applications. The tiers of each application are organized in vertical stacks that are horizontally replicated and each tier of an application is hosted on a separate VM, as shown in Figure 1. We recall that each VM communicates only with other VMs within the same vertical stack. However, the presence of the request dispatcher leads to similar communication patterns for VMs belonging to the same tier but to different vertical stacks of the same application, thus complicating the identification of communicating VMs based only on the

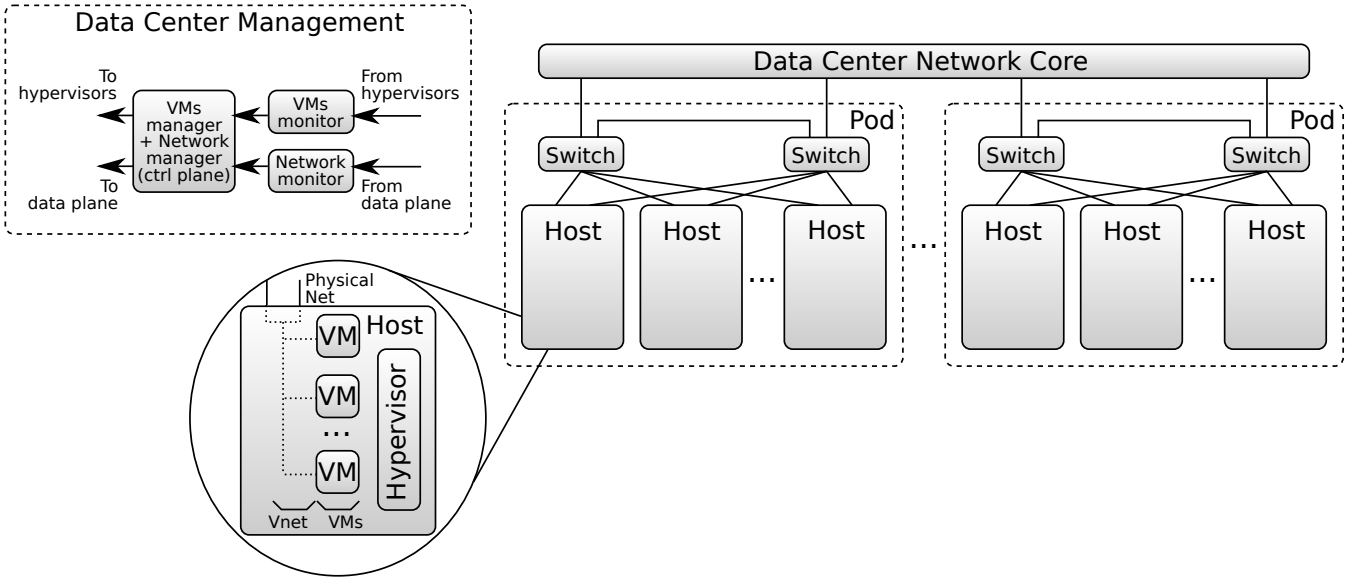


Figure 2: Software-Defined Data Center

input/output packet rate information of each VM.

To support a network-aware VMs allocation in this scenario, the proposed methodology has two main goals. A first goal is to identify couples of VMs exchanging data; this information may be directly used in the VMs allocation problem, for example by adding in the objective function of the underlying optimization problem a term that takes into account the cost of data exchange [6]. A second goal is to achieve a more detailed knowledge about how the deployment of the applications over the data center in order to provide useful input information to higher level VMs allocation approaches; for example, higher availability and survivability of the application can be achieved simply by locating non-interacting vertical stacks of the same application on different pods of the data center.

We now introduce a more formal definition of the basic principles previously sketched. Let us consider a set $\mathcal{N}$ of VMs. We define $P_{j_1}^{out}$ as the time series describing the output packet rate of a generic VM $j_1 \in \mathcal{N}$, and $P_{j_2}^{in}$ as the time series of input packet rate for a VM j_2. We denote as τ the time interval between the samples of the time series. Starting from these time series describing the network activity of the VMs in the data center, we aim to infer which VMs communicate among themselves by considering the correlation in their input and output traffic. We assume that the vertical stacks remain the same (that is, they are not re-organized) during the time series data collection. The proposed methodology to identify communicating VMs is based on the following steps:

1. Interpolation and synchronization of the time series;
2. Computation of a correlation matrix between input and output traffic of couples of VMs;
3. Identification of interacting VMs through correlation-based analysis and clustering.

3.1 Traces interpolation and synchronization

The cloud monitor that collects information about the VMs network utilization is based on the prototype described in [3]. The data collector produces a sequence of tuples in the form `<timestamp, pkt_in, pkt_out>`, where the last two values contain the number of packets received and transmitted since the last data sampling. However, the agents monitoring each VM are not explicitly synchronized. As a result, traces collected from different VMs may be not synchronized.

Preliminary tests with artificially generated traces demonstrate the detrimental effect of not synchronizing traces before applying the correlation analysis of network traffic patterns belonging to different VMs. For this reason, we introduce in our system a trace synchronization step: to achieve this result we rely on data interpolation. Our approach can be summarized as follows:

- we remove samples at the beginning from each time series such that each time series starts in the time interval $[t_0, t_0+\tau]$ and we make sure that each time series contains T samples;
- we compute a synchronization time t_0^* that is the average of the starting times of each trace; from this time we have the interpolated time series timestamps that we define as $t_0^*, t_0^* + \tau, \ldots, t_0^* + i\tau, \ldots, t_0^* + T\tau$;
- for each time series $P_{j_1}^{out}$, we define a synchronized time series $P_{j_1}^{*out}$ using cubic interpolation. In our prototype the implementation of the cubic interpolation is provided by the python *Pandas*[2] framework. A similar procedure is carried out also for the time series of inbound packets $P_{j_2}^{in}$.

Figure 3 provides an example of the above described process. We start with two time series, represented with squares and circles in the upper part of the figure. For both time series we have an interval of τ between two consecutive samples. In the middle part of the figure we show a spline interpolation of the samples that creates a continuous space of values for each possible sampling instant. Finally, the bottom part of the figure shows the re-sampling of the time series. The output is new couple of time series (again represented with squares and circles) where the sampling instants are synchronized.

As a simplified notation, in the following of the paper we will use $P_{j_1}^{*out}(i)$ and $P_{j_2}^{*in}(i)$ for the i-th sample of a synchronized time series.

[2] http://pandas.pydata.org/

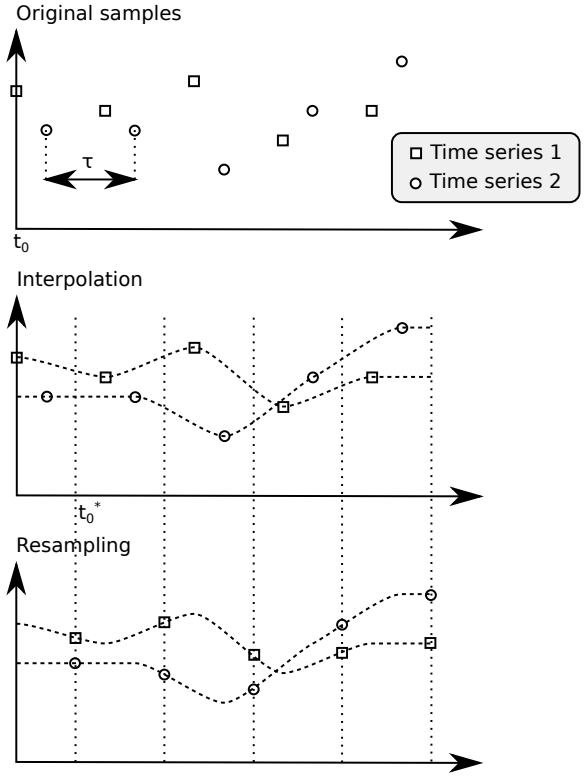


Figure 3: Interpolation and synchronization

3.2 Correlation matrix

Starting from the synchronized time series we can compute the correlation matrix $\mathbf{C}$ between input and output packet rates of each pair of VMs. Our choice is to focus on the correlation between output packet of a VM and input packet of another, to identify data exchange flows from one VM to another. This approach is consistent with other approaches that take into account data exchange to optimize VM communication such as [19, 20] where not just the amount of data exchanged but also the direction of this data exchange is taken into account. Specifically, we define the generic element c_{j_i,j_2} of the matrix as:

$$c_{j_i,j_2} = \mathrm{Cor}(P_{j_1}^{*out}, P_{j_2}^{*in}) \tag{1}$$

where j_1 and j_2 are two generic VMs and $\mathrm{Cor}(\cdot)$ is a function that computes the correlation among two different time series. In literature multiple functions have been proposed to measure the correlation between data series. It is worth to note that this approach leads to the creation of a correlation matrix that may not be symmetric because we may have a strong correlation of data exchange from VM i to VM j (e.g., if VM i sends large amount of data to VM j) but low correlation in the opposite direction (in the previous example the main data flow from VM j to VM i could consist of just ACK messages). In our analysis we consider two different alternatives, that are the *Pearson* and the *Spearman* correlation functions [17]. The first is the most widely used solution to compare time series, while the second emerged in preliminary results as one of the most interesting alternatives in terms of ability to discriminate groups of time series characterized by similar long-term trends, as is the case of our experiments.

We recall that the Pearson correlation index ρ is defined as:

$$\rho(P_{j_1}^{*out}, P_{j_2}^{*in}) = \frac{E[(P_{j_1}^{*out} - \mu(P_{j_1}^{*out}))(P_{j_2}^{in} - \mu(P_{j_2}^{*in}))]}{\sigma(P_{j_1}^{*out})\sigma(P_{j_2}^{*in})} \tag{2}$$

where $\mu(\cdot)$ is the mean value of a time series, $\sigma(\cdot)$ is the standard deviation, and $E[\cdot]$ is a shorthand for the average function (in equation 2 the average is used to compute the covariance of the two time series).

The Spearman correlation index ρ_s corresponds to the correlation between two time series where the original values have been substituted by their ranks in the time series:

$$\rho_s(P_{j_1}^{*out}, P_{j_2}^{*in}) = 1 - \frac{6\sum_{i=0}^{T} r(P_{j_1}^{*out}(i)) - r(P_{j_2}^{*in}(i))}{T(T^2-1)} \tag{3}$$

where T is the number of samples in the time series and $r(\cdot)$ is the rank of a sample among the other samples of the same time series. Switching from the values to the ranks in a time series increases significantly the capacity of this function to discriminate the small fluctuations in the packet rate that may place apart two time series with the same long-term behavior. This is a promising solution to cope with the characteristics of our scenario, where we assume to have a similar workload on every element of the replicated vertical stacks of the considered application.

3.3 Identification of interacting VMs

The final step of our proposal is the definition of couples/groups of interacting VMs based on the correlation matrix obtained in the previous step.

This step can either focus on identifying single couples of VMs that intensively exchange network traffic or focus on providing a broader view of the system grouping together VMs that belong to the same vertical stack of an application (as in Figure 1).

If we consider the first task, that is identifying couples of interacting VMs, the most straightforward approach is to apply a threshold on the correlation matrix to distinguish between *correlated* and *uncorrelated* time series of network resource utilization. Basically, we consider as correlated every couple of time series with a correlation value higher or equal than a defined threshold, while if the correlation value is lower than the threshold, we assume the corresponding time series to be uncorrelated.

On the other hand, if our goal is to identify the vertical clusters of VMs running tiers of the same application, we need a more complex approach aiming to cluster VMs that show a similar behavior in terms of network traffic patterns. The correlation matrix is considered as an *affinity matrix* between VMs and a clustering algorithm is applied to group together the VMs. Supporting an input in the form of an affinity/distance matrix (instead of a feature vector) is a critical point in the choice of the clustering algorithm. Specifically, we take into account three possible solutions: *spectral clustering* [13], *affinity propagation* [11], and *agglomerative clustering* [9].

The spectral clustering algorithm computes the Laplacian operator from the input similarity matrix. The eigenvalues and eigenvectors of the Laplacian are then used to extract a new coordinate system that is fed into a k-means clustering phase [13]. Affinity propagation is a fast clustering approach that aims to identify cluster representatives by simulating the exchange of messages among data points. Once the cluster representatives are found, clustering is carried out by assigning to each representative the most affine other elements. Finally, the agglomerative clustering is a hierarchical clustering algorithm that starts with each VM in its own cluster

and, then, at each iteration merges the two most affine clusters (we use the average distance between elements of the two clusters as a representation of the cluster affinity, but preliminary experiments show that other metrics such as the minimum of the maximum distance provides similar results). The resulting *dendogram* is then cut depending on the number of clusters we want to obtain.

It is worth to note that all the considered algorithms can automatically determine the number of clusters to use, either directly (as for the affinity propagation clustering that includes the estimation of the number of clusters in the algorithm) or indirectly (for example, by performing a spectral gap analysis for the spectral clustering algorithm by adding constraints on the cluster size to determine when the agglomerative clustering stops). For all the three algorithms, we rely on their implementation in the *SciKit-Learn* library[3].

4. EXPERIMENTAL RESULTS

4.1 Experimental setup

We test our infrastructure using an experimental setup where multiple copies of the TPC-W benchmark[4] are executed in parallel over a cloud data center hosted by the Amazon EC2 infrastructure[5]. The monitoring is based on the architecture described in [3]. Our traces refer to a TPC-W run with 12 VMs divided into 4 vertical stacks for a period of 12 hours, with samples collected by default every 30 seconds (but we present also experiments with traces where data collection has a granularity of 1 and 2 minutes).

As metrics for the comparison of the different correlation functions, we distinguish between the two goals of identifying the interacting couples of VMs and clustering together VMs to identify the vertical stacks.

For the first goal we consider the classic measures of classification problems, that are *precision*, *recall*, and *F-measure*. *Precision* is defined as the fraction of identified couples of VMs that are actually communicating, while *recall* is the fraction of communicating couples of VMs that are correctly identified. In terms of true/false positives/negatives, we can define the precision $P = \frac{TP}{TP+FP}$, and the recall $R = \frac{TP}{TP+FN}$. F-measure is the harmonic mean of precision and recall, that is: $F = 2\frac{P \cdot R}{P+R}$.

For the clustering, we consider the clustering *purity* [2] as the main metric to compare different solutions. Purity, that is one of the most popular metrics for cluster evaluation, considers the fraction of correctly identified VMs as the measure of the clustering performance. Specifically, purity is defined by comparing the generic final solution $\mathbf{S}$ of the VM clustering with the ground truth vector $\mathbf{S}^*$, which represents the correct classification of the VMs into the vertical stacks. Purity is thus defined as:

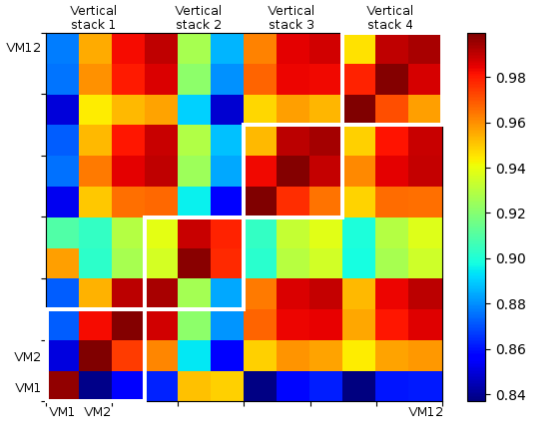
$$purity = \frac{|\{s^j : s^j = s^{j*}, \forall j \in \mathcal{N}\}|}{|\mathcal{N}|}$$

where s^j is the cluster to which VM j is assigned in the considered solution, s^{j*} is the *correct* classification of VM j, and $\mathcal{N}$ is the set of clustered VMs.

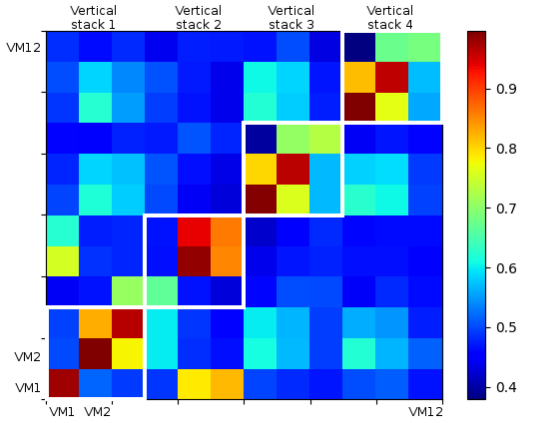
4.2 Comparison of correlation indexes

We first present the correlation matrices computed over every couple of time series using both Pearson and Spearman correlation indexes.

[3]http://scikit-learn.org/
[4]www.tpc.org/tpcw/
[5]https://aws.amazon.com/ec2/



(a) Pearson correlation index



(b) Spearman correlation index

Figure 4: Heatmap of the correlation matrices

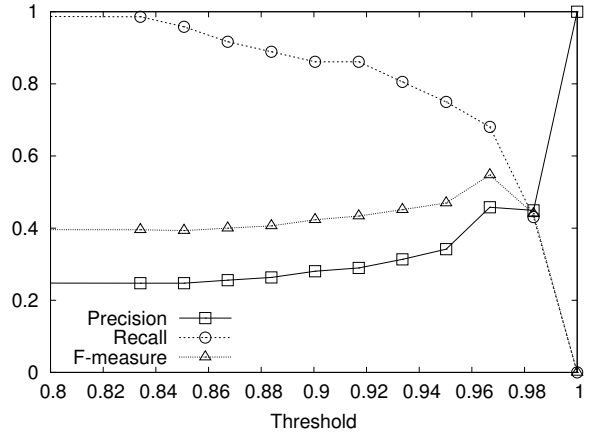
Figure 4 represents the two correlation matrices represented as heat maps. For each VM (numbered 1 to 12 in the figure) we consider the two time series of input and output packets. Each picture presents also an outline of the vertical stacks (shown as four white square frames on the main diagonal of the matrix). Ideally the correlation should present high values within the four squares delimiting each vertical stack and low values outside.

A qualitative analysis of Figures 4a and 4b provides some interesting insights: we observe clearly that the Pearson correlation index (Figure 4a) tends to return higher values (as testified by the general predominance of reddish colors). Particularly critical is the presence of red-orange shades far from the main diagonal (and outside from the vertical stack frames), that suggests high correlation also between VMs belonging to different vertical stacks. On the other hand, the Spearman correlation index (Figure 4b) provides a clearer distinction between groups of VMs in the same vertical stack (the red areas close to the diagonal) and VMs belonging to different stacks, suggesting a better capacity to distinguish communicating from not interacting VMs.

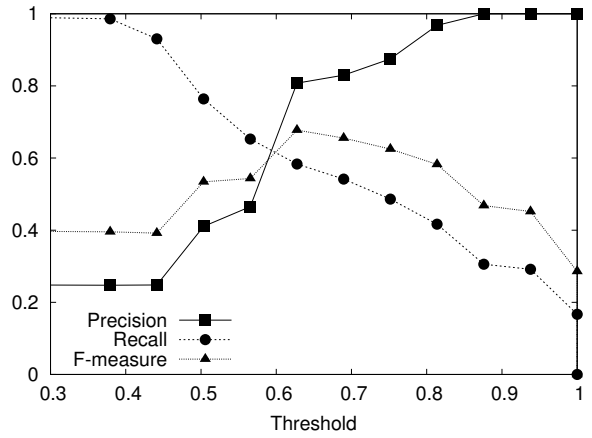
4.3 Identification of interacting VMs

We now consider how the two considered correlation indexes

can be used in a threshold-based identification of the interacting VMs couples. In particular, we evaluate sensitivity of the VMs classification with respect to the threshold value.



(a) Pearson correlation index



(b) Spearman correlation index

Figure 5: Precision, Recall, and F-measure

Figure 6 summarizes this comparison showing the accuracy for both correlation indexes as a function of the threshold value. It is clear that the Spearman correlation index provides a twofold advantage. First, it achieves better performance than the Pearson index, with significantly higher maximum F-measure values. Second, it provides more stable performance with a range of threshold values returning an accuracy higher than 0.5 that spans from 0.5 to 0.8, while the best values for the Pearson function are limited in a single peak around 0.96.

4.4 Clustering of VMs

A further analysis concerns the case where the correlation indexes can be used to create the input affinity matrix for the different matrices

Table 1: Clustering purity

Clustering algorithm	Spearman	Pearson
Spectral clustering	0.75	0.66
Affinity propagation	0.50	0.42
Agglomerative clustering	0.38	0.38

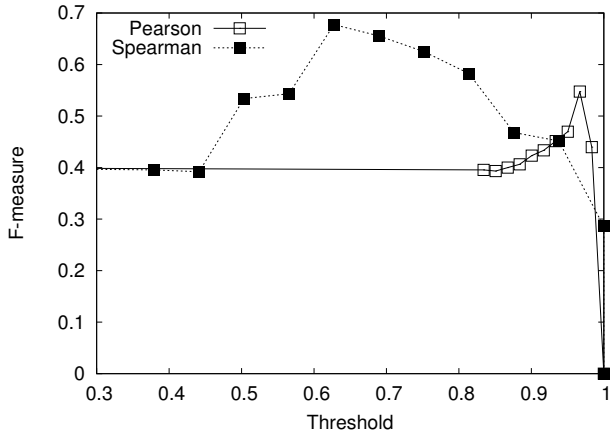


Figure 6: F-measure comparison

Table 1 provides an insight on the clustering purity of the three considered clustering algorithms for each of the two correlation indexes taken into account. The message from the table is rather straightforward, concerning both the clustering algorithms and the correlation indexes. If we observe the different algorithms with affinity input matrices based on the two correlation indexes, it is evident that the Spearman index leads to significantly higher clustering purity with respect to the alternative. Furthermore, if we compare the three considered clustering algorithms, we observe that the spectral clustering algorithm clearly outperforms the other two alternatives.

If we consider the reasons for the performance of the different clustering algorithms, we may refer to the example in Figure 7. It shows an example of clustering output (on the top – the example refers to the spectral clustering algorithm) against the correct clustering (bottom). We observe two types of errors: VMs 1 and 4 are swapped (each is assigned to the cluster where the other belongs), while VM 12 is simply misplaced. For the affinity propagation algorithm the poor clustering performance are reduced by the presence of an additional problem, that is a wrong estimation in the number of clusters. Finally, for the agglomerative clustering, we observe a large number of cluster swaps between VMs (as in the case of VMs 1 and 4 in the figure).

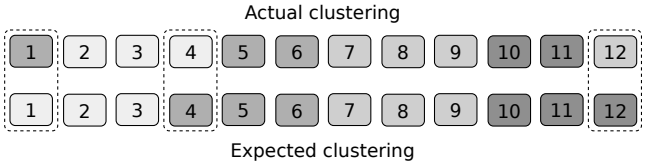


Figure 7: Clustering example

4.5 Comparison of time sampling intervals

As a final analysis, we evaluate how the data sampling granularity affects the quality of the classification of correlated and non-correlated time series and of the VMs clustering. We start our analysis focusing on the classification problem. To this aim, Figure 8 compares the F-measure achieved by the classification based on the Spearman correlation index as the sampling period τ of the network utilization ranges from 30 seconds to 2 minutes. We observe that, even with this small change in the sampling granularity, the impact on the classification F-measure is very significant. Specifically, we

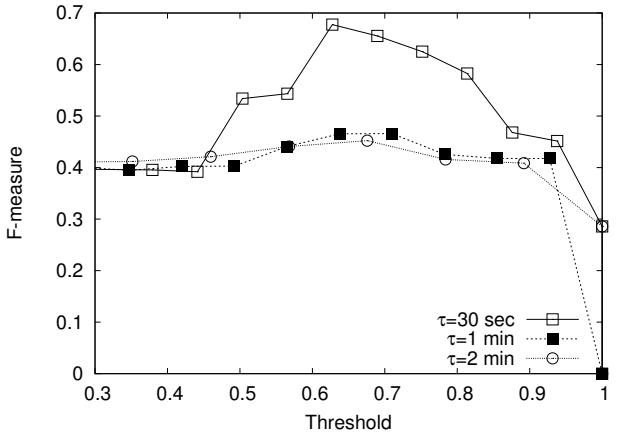


Figure 8: Sensitivity to sampling period τ

observe two main effects of a coarse-grained data collection. First, the sensitivity of the Spearman correlation to the threshold value is further reduced if we consider a large sampling period. This higher robustness is an effect of the smoothed time series: basically, we work with a time series where only the most evident fluctuations are taken into account. This means that the low-frequency and highly evident fluctuations, that are the only way to distinguish two vertical stacks, are easier to identify even over a wider range of threshold values, thus explaining the observed stability with respect to different threshold values. Second, if we consider the threshold range related to the higher F-measure ([0.6-0.8]), we observe that the accuracy tends to decrease dramatically as the sampling interval increases. This effect is rather intuitive, because by increasing the sampling period, we obtain a smoother time series of network resource utilization. This effect still captures the main effects of correlation, but the high frequency fluctuations that allow us to distinguish two vertical stacks are lost, with a consequent reduction of the classification performance.

Table 2: Clustering purity

τ	**Spectral clustering**	**Affinity propagation**	**Agglomerative clustering**
$\tau = 30$ sec	0.75	0.50	0.38
$\tau = 1$ min	0.50	0.27	0.38
$\tau = 2$ min	0.50	0.27	0.33

The second analysis concerns the impact of sampling frequency on the clustering purity. To this aim, Table 2 compares the clustering purity for the three considered clustering algorithms when the Spearman correlation index is used. The table provides two clear confirmations. On one hand, it confirms the better performance of the spectral clustering over the other algorithms, with a purity that is 97% to 31% higher compared with the alternatives. On the other hand, we observe that, even for the spectral clustering algorithm, the increase of the sampling period τ has a major detrimental effect, with the purity reduced by 33% as τ grows from 30 seconds to 1 minute. This purity reduction is caused by the same reasons that determine a reduction of the F-measure for the identification of interacting VMs, that is the smoother time series makes more difficult to distinguish between different vertical stacks.

5. CONCLUSIONS

In this paper we analyzed the problem of identifying groups of VMs in a cloud infrastructure that exchange information without having access to a detailed model of the data transfer among them. In particular, we focus on the case of horizontally replicated vertical stacks, where each VM in a vertical stack corresponds to a tier of an application. We pointed out that this case is very common and challenging in a cloud scenario where scalability is achieved though replication.

We described a methodology to discover communicating VMs that is based on the analysis of correlation between the time series of network traffic of each VMs and on clustering algorithms for identifying the vertical stacks of VMs. We compare different correlation indexes to identify the most suitable alternative: our experiments show that the use of ranking-based techniques (as in the Spearman correlation index) clearly outperforms traditional correlation metrics, such as the Pearson index. Furthermore, we compare three clustering algorithms to identify the most suitable alternative for detecting the vertical stack: our experiments demonstrate that spectral clustering far outperforms the alternatives. Finally, we perform a sensitivity analysis demonstrating that fine-grained network data collection is essential to achieve high accuracy in the identification of communicating VMs.

Acknowledgement

The authors acknowledge the support of the University of Modena and Reggio Emilia through the project *S²C: Secure Software-defined Cloud*. The authors also thank Davide Ferrari for his help in conducting the experiments.

6. REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, pages 63–74, New York, NY, USA, 2008. ACM.

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

[3] M. Andreolini, M. Colajanni, and M. Pietri. A scalable architecture for real-time monitoring of large information systems. In *Proc. IEEE Symposium on Network Cloud Computing and Applications*, London, UK, Dec. 2012.

[4] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards predictable datacenter networks. *ACM SIGCOMM Computer Communication Review*, 41(4):242–253, 2011.

[5] A. Beloglazov, J. Abawajy, and R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768, 2012.

[6] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Y. Zomaya. Energy-efficient data replication in cloud computing datacenters. *Cluster Computing*, 18(1):385–402, 2015.

[7] C. Canali and R. Lancellotti. Exploiting ensemble techniques for automatic virtual machine clustering in cloud systems. *Automated Software Engineering*, 21(3):319–344, Sept 2014.

[8] C. Canali and R. Lancellotti. Exploiting Classes of Virtual Machines for Scalable IaaS Cloud Management. In *Proc. of the 4th Symposium on Network Cloud Computing and Applications (NCCA)*, Jun. 2015.

[9] W. H. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24, 1984.

[10] D. Drutskoy, E. Keller, and J. Rexford. Scalable network virtualization in software-defined networks. *IEEE Internet Computing*, 17(2):20–27, 2013.

[11] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[12] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. Elastictree: Saving energy in data center networks. In *Proc. of 7th USENIX Conference on Networked Systems Design and Implementation (NSDI)*, San Jose, California, 2010.

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

[14] A. Marotta and S. Avallone. A Simulated Annealing Based Approach for Power Efficient Virtual Machines Consolidation. In *Proc. of 8th International Conference on Cloud Computing (CLOUD)*. IEEE, 2015.

[15] C. Mastroianni, M. Meo, and G. Papuzzo. Probabilistic Consolidation of Virtual Machines in Self-Organizing Cloud Data Centers. *IEEE Transactions on Cloud Computing*, 1(2):215–228, 2013.

[16] X. Meng, V. Pappas, and L. Zhang. Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement. In *Proc. of the 29th Conference on Information Communications (INFOCOM)*, San Diego, California, USA, Mar. 2010.

[17] L. Myers and M. J. Sirois. *Spearman Correlation Coefficients, Differences between*. John Wiley & Sons, Ltd, 2014.

[18] M. Shojafar, C. Canali, R. Lancellotti, and E. Baccarelli. Minimizing computing-plus-communication energy consumptions in virtualized networked data centers. In *Proc. of 21st IEEE Symposium on Computers and Communications (ISCC)*, Messina, Italy, Jun. 2016.

[19] J. Sonnek, J. Greensky, R. Reutiman, and A. Chandra. Starling: Minimizing Communication Overhead in Virtualized Computing Platforms Using Decentralized Affinity-Aware Migration. In *Proc. of 39th International Conference on Parallel Processing (ICPP)*, San Diego, CA, Sept 2010.

[20] Y. Zhang and N. Ansari. Hero: Hierarchical energy optimization for data center networks. *IEEE Systems Journal*, 9(2):406–415, 2013.