A Decentralized Approach for Adaptive Workload Estimation in Virtualized Environments

Nisrine Ghadban
ICD - STMR - UMR 6281 CNRS
Autonomous Network
Environment Team
Troyes University of Technology
France
Email: nesrine.ghadban@gmail.com

Rémi Cogranne
ICD - STMR - UMR 6281 CNRS
Laboratory of Systems Modelling
and Dependability
Troyes University of Technology
France
Email: remi.cogranne@utt.fr

Guillaume Doyen
ICD - STMR - UMR 6281 CNRS
Autonomous Network
Environment Team
Troyes University of Technology
France
Email: guillaume.doyen@utt.fr

***Abstract*—Cloud computing is gaining an important role in providing high quality IT services. However, the heterogeneous and dynamic nature of the activities it hosts makes the related management operations, serving performance or security purposes, complexes. Leveraging the autonomic paradigm, represents a promising solution but it requires efficient grounded monitoring and analysis functions which can in turn implement advanced control algorithms. In this effort, this paper presents a robust and cost effective solution to monitor and estimate the workload in a virtualized environment. It consists in a decentralized algorithm leveraging an incremental Principal Component Analysis (PCA) featuring the system activity of multi-tenants execution environments. To evaluate the relevance of our proposal in terms of both performance and cost, we consider real execution traces of more than one thousand PlanetLab containers hosted on more than forty servers belonging to more than one hundred tenants.**

I. Introduction

For a decade, cloud computing, which refers to the Information Technology (IT) and network services and resources that can be deployed quickly and on a large scale through virtualization means, has emerged. Large companies such as Google, Amazon and Microsoft strive to provide more reliable, powerful and cost-efficient cloud platforms. Cloud providers deliver different service levels (Infrastructure, Platform or Software as a Service, all known as IaaS, PaaS and SaaS) to different consumers which can reduce both of their capital and operational expenditures by outsourcing part of their previous activity on mutualized platforms leveraging multi-tenancy. These services are made available as subscription-based services in a usage-based payment model, thus bringing high flexibility for their consumers.

From a cloud provider perspective, being able to accurately estimate the workload of its infrastructure is a key element toward the implementation of any subsequent management and control mechanism. According to traditional functional area of network and service management, the latter can serve for (1) performance purposes [1], for instance to deploy and balance Virtual Machines (VM) on appropriate servers, (2) accounting, for an accurate counting of resource usage, and (3) security for the detection of abnormal behaviors and threats [2], [3], within the operated infrastructure.

However, accurately monitoring and estimating the workload in a cloud environment is challenging for the following reasons. Firstly, the workload is highly heterogenous and dynamic. The hosted services have complex and individual composition, provisioning, configuration, and deployment features which also vary over the time according to the tenant activity and requirements. Secondly, a cloud provider cannot use intrusive probes to monitor the tenants' activities in a fine-grained manner. Indeed, in most clouds' infrastructures, and especially public IaaS solutions, the tenant-related space is private, since leased by the provider. Consequently, monitoring solutions can only operate in a black-box way, by leveraging the sole metrics available at the virtualization layer, to ensure the tenant's privacy. Finally, given the large scale of cloud solutions which can host hundreds of thousands of servers all hosting millions of virtual machines, the workload monitoring system must support a large volume and velocity of monitoring data while still providing a sufficient level of accuracy against a reasonable computation and communication costs.

In this paper, we propose a solution which aims at estimating the global workload of a cloud infrastructure in a decentralized manner. Such a solution aims at providing a basic ground that any subsequent management or control framework can leverage to provide advanced functionality. The system metrics we consider in our approach are the processing time, the memory consumption and the network input/outputs (I/O) which stands for the three main resources a cloud provider offers to its customers while being easily monitored at a virtualization level without breaking the tenant privacy. In order to feature accurately such a footprint, we consider a Principal Component Analysis (PCA) [4], [5]. The broad scope of the PCA makes it one of the most widely used methods used in machine learning [6], pattern recognition [7], multi-label classification [8], detection [9], [10] and quality control [11], [12], thus being a good candidate for any advanced management function. It is also a relevant solution within the scope of scalable workload estimation since PCA allows the characterization of behavior regardless the volume of the activity, thus motivating its choice as a ground system footprint abstraction. Since the computational

978-3-901882-89-0 @2017 IFIP

complexity of eigen-decomposition, at the core of the PCA computation, is cubic with the size of the dataset, we advocate in this paper an in-network processing to estimate the most relevant principal components without computing the sample covariance matrix and its eigen-decomposition. This approach, based on the CCIPCA proposal [32], greatly reduces the computational complexity, the memory storage capacity and the cost of communication. Finally, in order to make our solution scalable and compliant with autonomic management requirements, we consider a decentralized collaboration between local monitoring probes which can thus intrinsically scale with the system size, while enabling all of them to get the workload estimation at any time.

In order to validate our workload model algorithms, we considered a dataset collected over more than 40 PlanetLab servers hosting over a thousand LXC containers belonging to more than a hundred tenants, thus providing an amount of more than 2,5 million system traces. In this context, we demonstrate to what extent our approach allows the accurate estimating of workload from the original noisy dataset at reasonable cost.

The rest of paper is organized as follows. In Section II, we describe the related work. Section III presents our decentralized method of PCA. We evaluate our approach in Section IV. Conclusion and future works are presented in Section V.

II. Related Work

Our work lies at the intersection of three research areas which are (1) workload estimation in cloud environments (2) PCA in Big Data contexts, and (3) decentralized PCA. We survey all of them below and also provide some background on PCA which our contribution relies on.

A. Cloud Workload Estimation

The problem of workload characterization and prediction has been widely studied [13], [14], [15]. Among them, some research focus on creating mathematical and statistical models to characterize the workload in a cloud environment. The classification of tasks based on CPU and memory usage in [16] relies on the statistical identification of qualitative coordinates (i.e., small, medium, large). A similar approach is applied in [17] to study the CPU and memory usage of tasks and jobs and discover task shapes and durations. In [18], periodicities and patterns with homogeneous behaviors are identified with spectral and autocorrelation analyses. The work which is the closest to ours lies in a decentralized clustering approach presented in [19] for a dynamic mix of heterogeneous applications in cloud environments. The presented autonomic mechanisms handle VMs provisioning in order to improve resource utilization, which is achieved by reducing overprovisioning at two levels. In comparison, our work does not aim at classifying activities but rather provides the best average estimation of the global activity. Besides, our method does not rely on a priori knowledge or model but is fully data-driven for high adaptivity while being completely decentralized.

B. PCA for Large Datasets

1) Background: Principal component analysis (PCA) is a powerful technique for analyzing and identifying patterns in data. It finds the most important axis to express the scattering of data by determining the subspace which holds the largest variance. This subspace is spanned by the principal axes and the projection of data in this subspace constitutes the principal components, which reflects the approximate distribution of data.

PCA applied on the data matrix $\boldsymbol{X}$ of V observations (in column), also called individuals, composed of p variables, solves the eigenvectors decomposition problem:

$$\boldsymbol{C}\boldsymbol{w}_i = \lambda_i \boldsymbol{w}_i, \qquad i = 1, 2, \cdots, p, \tag{1}$$

where $\boldsymbol{C}$ is the covariance matrix of the matrix $\boldsymbol{X}$, calculated when the observations have zero means by $\boldsymbol{C} = \frac{1}{V}\boldsymbol{X}^\top \boldsymbol{X}$ with $\boldsymbol{X}^\top$ the transpose of $\boldsymbol{X}$. The values $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$ represent the eigenvalues sorted in descending order and $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_p]$ is the corresponding eigenvector matrix, where $\boldsymbol{w}_i$ is the i^{th}-axis direction. The mapping of the data to principal axis $\boldsymbol{w}_i$, which refers to as the i^{th} principal axis, whose variance is λ_i , is given by the projection onto w_i: $\boldsymbol{Y}_i = \boldsymbol{w}_i^\top \boldsymbol{X}$. In other words, $\boldsymbol{Y}_i$ represents the contribution of i-th axis $\boldsymbol{w}_i$.

The computational complexity of calculating the covariance matrix is $\mathcal{O}(Vp^2)$, and it requires $\mathcal{O}(p^2)$ memory storage units. The computational complexity of the eigen-decomposition problem is $\mathcal{O}(p^3)$ [20]. To these costs, one should also include the communication cost, which is $\mathcal{O}(Vp)$ over a distance $\mathcal{O}(1)$. In the following, we propose a strategy that avoids the computation of the sample covariance matrix and its eigen-decomposition, thus allowing to significantly reduce the computational complexity.

2) PCA in Big Data: PCA is widely used in Big Data context [21], [12] since it is usually leveraged as a first step in data mining. However, because of the complexity of the covariance matrix computation, PCA still appears as a challenging problem when dealing with large datasets, and hence an active research topic. In this area, the main strategies to make PCA scalable, consists in reducing the dataset in order to allow the computing of PCA. In [22] for instance, coresets are defined as small sets that provably approximate the original data. This method is based on merge-and-reduce that permits the solving of computational problems such as PCA in parallel. In [23], a model based on PCA is built on a small subset of a large network producing a large amount of data. The principal components of this smaller subgraph allow the out-of-sample extension property.

C. Distributed PCA Approaches

Several attempts have been made to alleviate the problem of scalability of PCA in networking contexts by mainly distributing the computation. This was proposed for instance

in [24], [25] where algorithms are investigated to compute the eigenspace, but still with high computational cost. Recently, in [26], collective-PCA technique was proposed in which a fusion center only receives the principal components, instead of the whole time series. In [27], [28], the most relevant principal axis is estimated by the power iteration method. However, this method requires the computation of the sample covariance matrix and since the latter can only be achieved by gathering all the dataset on one single node, it is inappropriate for large-scale data. In [28], the power iteration method is used but with sparse matrices in order to reduce computational complexity. The PCA-based distributed approach (PCADID) proposed in [29] operates through a "divide-to-conquer" scheme in order to reduce the computational complexity of the eigen-decomposition problem. In [30], the covariance matrix is first estimated by means of a consensus averaging algorithm, then each node performs a local eigenvector decomposition. The distribute adaptive covariance matrix eigenvector estimation (DACMEE) algorithm, presented in [31], recursively updates the eigenvector estimates without explicitly constructing the full covariance matrix that defines them. Nodes share only the first fused observation and compute compressed covariance matrices. However, they still require the eigen-decomposition of several matrices which does not fit the context of large-scale networks. Candid covariance-free fast incremental PCA (CCIPCA) [32] algorithm offers a very good compromise between statistical accuracy and computational speed. It also has the advantage of not having major dependence on tuning parameters. However, this incremental algorithm is centralized and not adapted for solutions built on decentralized approaches.

To conclude, to the best of our knowledge, there is currently no appropriate method exhibiting a low computation and communication cost and operating in a decentralized way for computing principal components incrementally in the context of large datasets while the latter is necessary to enable any autonomic management function to operate. Consequently, in this paper, we propose a novel framework based on CCIPCA for estimating the principal axes that stand for the footprint of VMs in a cloud environment, iteratively and in a distributed in-network scheme, without the need to estimate the covariance matrix.

III. Workload Estimation with Decentralized and Low Computational Cost PCA

We propose to estimate the workload of VMs by introducing a method for computing the principal components of system activities that is both simple and decentralized. The principal components represent the different possible behaviors, regardless of the amount of activity. Relying on this decentralized estimation of PCA, each novel observation of a VM activity is compared with the first principal axes in order to find the most relevant behaviors and only the contribution onto the closest axes are kept. The novel observations are also used in the decentralized computation of the PCA in order to update the previous estimations.

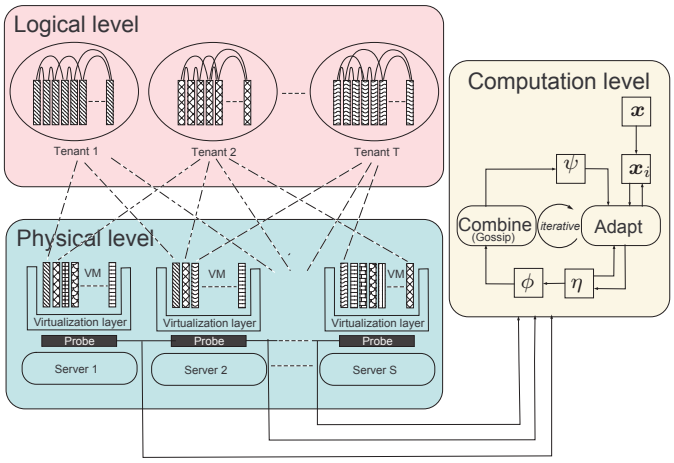


Fig. 1. Experimental environment

This approach offers the advantages of (1) allowing a good adaptivity with respect to dynamic variations of activities in time, (2) separating the heterogenous behavior within different similar activities and (3) representing the activities with axes gathering similar activities regardless of the volume of those activities. For clarity purposes, all notations used in the following are provided in Table I.

A. Decentralized CCIPCA

We introduce a decentralized version of candid covariance-free fast incremental PCA (CCIPCA) algorithm [32], to compute principal components incrementally without estimating the covariance matrix. This method is divided at each step into two phases: the *combine* phase and the *adapt* phase. The combine-then-adapt strategy has been investigated in linear adaptive filtering literature in [33], and recently in unsupervised learning in [6]. In the context of cloud computing, and with respect to Fig. 1, this strategy is based on the separation between the physical deployment of VMs among servers and the logical level of tenants to whom those VMs belong. The probes are implemented on the servers at the level of virtualization layer. First, a *combine* step averages the estimates, denoted ϕ, of all the servers that host the same tenant, resulting in an intermediate estimate denoted ψ. Then, each probe gathers the estimates ψ of the VMs it hosts, thus gathering the estimation from several tenants. This adapts step lead to update estimations of PCA ϕ.

Fig. 1 describes an illustrative environment where a set of physical servers represent the physical infrastructure of a cloud operator. Each physical server hosts different VMs belonging to different tenants. In order to understand the communication between probes, the scheme shows the logical level where a tenant is able to see only its own VMs. Communication is held between these VM using the gossip [34] protocol. This type of communication needs to define the neighborhood between nodes and in our work, we define the neighborhood randomly with a maximum number of neighbors. Note that from a practical point of view, the communications between VMs belonging to the same tenant are insured by the probes.

Symbol	Denotation
T	Number of tenants
S	Number of servers
V	Number of VMs
$\boldsymbol{C}$	Covariance matrix
$\boldsymbol{w}$	Principal axis
λ	Eigen value
η	Estimate of $\lambda\boldsymbol{w}$
ϕ	Intermediate estimate of $\boldsymbol{w}$ at servers level
$\psi^\star$	Ideal estimation of $\boldsymbol{w}$ at tenants level
ψ	Approximation of $\psi^\star$; intermediate estimate of $\boldsymbol{w}$ at tenant level
t	Iteration index
ϵ	Convergence error threshold
l	Amnesic parameter
θ	Time of data extraction
Θ	Angle between the i^{th} estimated and real principal axis

TABLE I
NOTATIONS USED TO DESCRIBE OUR ALGORITHM

B. A Method for Principal Components Estimation

Working in a decentralized manner, the probes update their estimates using cooperation between each other. We recall (see Equation (1)), that the principal components are given as the eigen vectors of the covariance matrix $\boldsymbol{C} = \frac{1}{V}\boldsymbol{X}\boldsymbol{X}^\top = \frac{1}{V}\sum_{v=1}^{V}\boldsymbol{x}_v\boldsymbol{x}_v^\top$. In the proposed cooperative method for each VM, with index v, each probe replaces the covariance matrix $\boldsymbol{C}$ by a local estimate $\boldsymbol{x}_v\boldsymbol{x}_v^\top$. Thus, at iteration step t, the VM v is associated with the estimation of principal components denoted $\eta_v(t)$ with, at initialization $\eta_v(0) = \boldsymbol{x}_v$.

The first step consists in combining the estimation of principal components $\eta_v(t)$ from all VMs. Because this is not possible at a large scale, we start by gathering data estimated at the infrastructure level, by a simple averaging of the estimates from all the VMs it hosts:

$$\phi_s(t) = \frac{1}{n_v}\sum_{v\in s}\eta_v(t). \tag{2}$$

In Equation (2), s represents the index of a server, $v \in s$ represents the index of VMs hosted on server s and n_v denotes the number of VMs on server s.

The second step for combining all the estimations $\eta_v(t)$ is carried out by leveraging the fact the VMs that belong to one tenant t are distributed among several physical servers, as depicted in Fig.1. However, once again, in a large-scale context the tenant may have a high number of VMs making the sharing of their estimations between all of them unrealistic because of the prohibitive communication cost. Ideally, we would thus like to approximate the average of all the VMs belonging to the tenant T:

$$\psi_T^\star(t) = \frac{1}{n_s}\sum_{s\in s_T}\phi_s(t) \tag{3}$$

$$= \frac{1}{n_s \times n_v}\sum_{s\in s_T}\sum_{v\in s}\eta_v(t). \tag{4}$$

with n_s the number of servers in the set s_T.

While in Equation (2), $\phi_s(t)$ represents the estimates at the probe of server level, the estimate $\psi_T(t)$ in Equation (4) represents the estimates at the tenant level which is in fact the average of the estimate from all the VMs deployed over all the server in the set s_T.

In order to carry out the computation of the estimate $\psi_T^\star(t)$ at a reasonable communication cost, we propose to replace it with the approximation obtained using a symmetric gossip described [37]. In detail, at iteration t, the probe associated with VM v randomly selects one of its neighbors (belonging to the same tenant), say u, and they both exchange estimations, obtained at the server level, to perform an update as follows:

$$\psi_v(t) = (1-\nu)\times\phi_v(t) + \nu\times\phi_u(t), \tag{5}$$

$$\psi_u(t) = \nu\times\phi_v(t) + (1-\nu)\times\phi_u(t), \tag{6}$$

with $\nu \in [0,1]$ a mixing parameter that defines the rate of update. It is important to note that the estimations at tenant level may differ from VMs to VMs due to the random selection of a neighbor. Hence we adopted the notation $\psi_v(t)$ instead of ψ_T. However, because the principal axes update described in Equations (5)-(6) is performed by all the probes, at the end of this gossip process, the VMs of each tenant should share similar estimations ψ that approximate $\psi_T^\star$ (4).

At the end of the *combine* step, the very last step consists in adapting the estimations at the VMs level, by taking into account the combined estimations obtained at the tenant level. To this end, we propose not to use the original CCIPCA method [35]. The reason is that CCIPCA has been investigated in the context of computing a PCA only once in a centralized manner with a dataset that does not change. Here, we must take into account the fact that the activity is dynamic by incorporating an amnesic factor [32] to forget the oldest estimates. Adapting the CCIPCA to the case of dynamic lead us to redefine the update as follows:

$$\eta_v(t) = \frac{t-1-l}{t}\psi_v(t-1) + \frac{1+l}{t}\boldsymbol{x}_v^\top\boldsymbol{x}_v\frac{\psi_v(t-1)}{\|\psi_v(t-1)\|}, \tag{7}$$

where the positive parameter l is referred to as the amnesic parameter. Note that the two modified weights still sum to 1. With the presence of l, larger weight is given to new samples and the effect of old samples will fade out gradually. Typically, l ranges from 2 to 4.

The second improvement we proposed is simple and obvious but greatly speed up the computation of the principal components estimations. When new data are received and the process has not started over, instead of initializing the method with $\eta_v(0) = \boldsymbol{x}_v$, we allow the use of estimations obtained with the previous data. The idea is that though activity is

dynamic, it is not likely to change abruptly very often.

Finally, it is important to note that the estimation of other principal components, associated with smaller eigenvalues, can be computed using the fact that the eigenvectors are orthogonal. It is thus possible, by using the Gram-Schmidt process, to subtract from the data contribution from the first axis $\eta_{1,v}(t)$ with the following projection:

$$\boldsymbol{x}_{2,v}(t) = \boldsymbol{x}_{1,v}(t) - \eta_{1,v}(t)\frac{\boldsymbol{x}_{1,v}(t)^\top \eta_{1,v}(t)}{\|\eta_{1,v}(t)\|^2}, \tag{8}$$

where $\boldsymbol{x}_{1,v}(t) = \boldsymbol{x}_v$. The obtained residual, $\boldsymbol{x}_{2,v}(t)$, which is in the complementary space of $\eta_{1,v}(t)$, serves as the input data to the iteration step. In this way, the first r dominant eigenvectors are obtained sequentially. One can also note that through this iterative of computing principal components, the computational complexity and the communication costs are both linear with respect to the number of components it is aimed at computing.

C. Using the Estimated Principal Components for Workload Estimation

After several iterations, the method described in the previous subsection allows all the VM to share the same principal components. In other words, we have $\eta_{i,v}(t) \approx \lambda_i \boldsymbol{w}_i$ for all VM v and for all eigenvector $\boldsymbol{w}_i$.

Once those estimates are shared by all the VM, the workload estimation is obtained by simply keeping only the part of activity measurements that lies within the subspace spanned by the principal components that contribute the most to the activity. From a practical perspective, the idea is that though tenants may have very different usage, they can likely be classified within a few main global behaviors (e.g. computing, storage, etc.). Thus, the principal components should reveal those global behavior and by selecting, for each VM data $\boldsymbol{x}_v$ the few components that are the most significant, we are likely to select what characterizes its type of behavior.

In practice, this is implemented as follows. Since the principal components are orthonormal, it is thus straightforward to get an estimation of $\boldsymbol{w}_i$ from $\eta_{i,v}(t)$ via the normalization:

$$\widehat{\boldsymbol{w}}_{i,v} = \frac{\eta_{i,v}}{\|\eta_{i,v}\|_2}. \tag{9}$$

The contribution due to each axis is simply computed as the projection of the activity measurements $\boldsymbol{x}_v$ onto each axis :

$$y_{i,v} = \widehat{\boldsymbol{w}}_{i,v}^\top \boldsymbol{x}_v. \tag{10}$$

The final estimation of the activity is obtained by keeping the r greatest contributions among $y_{1,v}, \ldots, y_{i,v}$. For the sake of clarity, let us denote $\boldsymbol{W}_v$ the matrix that contains only those most relevant axes to characterize activity measurements $\boldsymbol{x}_v$. The estimation of the workload can be written as:

$$\widehat{\boldsymbol{x}}_v = \boldsymbol{W}_i \boldsymbol{W}_i^\top \boldsymbol{x}_v. \tag{11}$$

To conclude, it is worth noting that the proposed method allows the dynamic adaptation of the workload estimation to the change that may happen as the principal components estimation are recomputed prior to all workload estimations. Second, the proposed method leverages the computation of principal components over all the VMs to cluster the main activity profiles and adapt this analysis to each VM by using only the most relevant.

The proposed method thus allows the online and decentralized estimating of all containers workload. However, we must also acknowledge that the proposed method estimated the workload globally, over all the containers. Though this estimation has a global overall good accuracy, it may performs worse for individual container or tenants with very small number of containers.

D. Algorithm

In the following, we describe the pseudo-code of our approach. For clarity purpose, we show a global algorithm which considers all servers, tenants and VMs. Basic operations of a probe are given in Algorithm 2, while Algorithm 1 shows the decentralized CCIPCA algorithm. In this algorithm, line 2 to line 8 represent the initialization phase. Then, the *Combine* phase is performed between line 10 and line 12. In this phase, estimations of the servers on which the same tenant is deployed are averaged using the gossip protocol described in Equations (5)-(6). Finally, the *Adapt* phase is highlighted between line 13 and line 18, where each probe adapts its estimation according to Equations (2)-(7)-(8). Finally, line 21 to line 23 normalize η_i to attain the eigenvectors and eigenvalues.

Note that the estimation of every VM v converge to the same eigenvector $\boldsymbol{w}_i$, in other words, $\boldsymbol{w}_i(v) \to \boldsymbol{w}_i \quad \forall\, v$.

IV. Evaluation of the Proposed Approach

In order to evaluate the performance and cost of our approach, we have considered real execution traces of Linux containers provided by the dataset presented in [39]. We have then implemented our algorithm in Matlab to simulate the case of a decentralized management system aiming at computing a global workload estimation and make it available to any upper management function. In this section we first present our dataset, then we present some intermediate results required to parameter our algorithm and finally we demonstrate to what extend (1) our approach is able to accurately follow the evolution of the workload while (2) exhibiting a reasonable computation cost, expressed as the number of algorithm iterations required to reach the convergence of the estimated axes, according to the number of tenants, servers and containers.

A. Dataset and Parameters

The dataset we consider to validate our approach consists in all the legitimate activity captured in the measurement campaign presented in [39]. The latter consists in real container execution traces monitored in the PlanetLab platform [40] which leverages LXC [41] as a lightweight virtualization technology. More precisely, we consider here 14 sets of 3-hour traces of system activities captured each second, from

Algorithm 1 Decentralized CCIPCA Compute the first r dominant eigenvectors

Require: $\boldsymbol{X} = [\boldsymbol{x}(1), \boldsymbol{x}(2), \cdots, \boldsymbol{x}(V)]$
Ensure: $\boldsymbol{w}_i$, λ_i

repeat
  $\boldsymbol{X}_{1,t} \leftarrow \boldsymbol{X}$
  for $i = 1 : \min\{r, t\}$ **do**
    if $i = t$ **then**
      INITIALISE $\eta_{i,1}(t), \eta_{i,2}(t), \cdots, \eta_{i,V}(t)$
      for $s = 1, 2, \cdots, S$ **do**
        $\phi_{i,s}(t) \leftarrow$ MEAN $(\eta_{i,v}(t))$
      end for
    else
      for $v = 1 : V$ **do**
        $\psi_{i,v}(t-1) \leftarrow$ GOSSIP_AVERAGE$(\phi_{i,s}(t-1))$
      end for
      for $s = 1 : S$ **do**
        for $v \in s$ **do**
          $[\eta_{i,v}(t), \boldsymbol{x}_{i+1,v}(t)] \leftarrow$ ADAPT$(\boldsymbol{x}_{i,v}(t), \psi_{i,v}(t-1))$
        end for
        $\phi_{i,s}(t) \leftarrow$ MEAN $(\eta_{i,v}(t))$
      end for
    end if
  end for
  for $i = 1 : \min\{r, t\}$ **do**
    $\boldsymbol{w}_{i,v} \leftarrow \eta_{i,v}(t) / \|\eta_{i,v}(t)\|, \qquad \lambda_{i,v} \leftarrow \|\eta_{i,v}(t)\|$
  end for
until $\|\eta_{i,v}(t) - \eta_{i,v}(t-1)\| < \epsilon$

Algorithm 2 Basic operations of a probe

real[] : INITIALISE
real[] : GOSSIP_AVERAGE(real[][])
real[], real[]: ADAPT(real[],real[])
real[] : MEAN(real[][])

Name	Denotation
SLICE	Slice name
ID	Slice context id
CPU	CPU consumption (%)
MEM%	Physical memory consumption (%)
MEM	Physical memory consumption (Ko)
MEM_V	Virtual memory consumption (Ko)
PRCS	Number of processes
TX1	Average sending bandwidth (Kbps) for last 1 min
TX5	Average sending bandwidth (Kbps) for last 5 min
TX15	Average sending bandwidth (Kbps) for last 15 min
RX1	Average receiving bandwidth (Kbps) for last 1 min
RX5	Average receiving bandwidth (Kbps for last 5 min
RX15	Average receiving bandwidth (Kbps) for last 15 min
IP	Local IP address of this node
PRCS_A	Number of active processes - that is, processes using the CPU cycle at the moment

TABLE II
RAW INFORMATION COLLECTED FOR EACH CONTAINER

2014, April to June, and measured with the Slicestat[1] server monitoring tool. In total, an amount of 48 servers traces have been considered, thus leading to the capture of more than 2,5 million of individual traces featuring 1 625 containers belonging to 128 tenants. TABLE II shows the set of raw information provided by Slicestat. In the following, we consider the sole CPU, memory and I/O related information, thus leading to 10 metrics to feature the overall system activity. Finally, one can note that for all subsequent results, simulations have been performed 240 times, and all depicted results are averaged and bounded with 95% confidence intervals.

The workload estimation algorithm, at time θ, is performed every 15 second on data averaged to remove noise and outliers from raw Slicestat measurements. In order to evaluate the convergence time of our approach, we consider the number of steps required to reach a difference lower than $\epsilon = 0.001$ between two consecutive estimations and we limit this number of steps to 100 in order to keep the convergence time lower than two consecutive measurements.

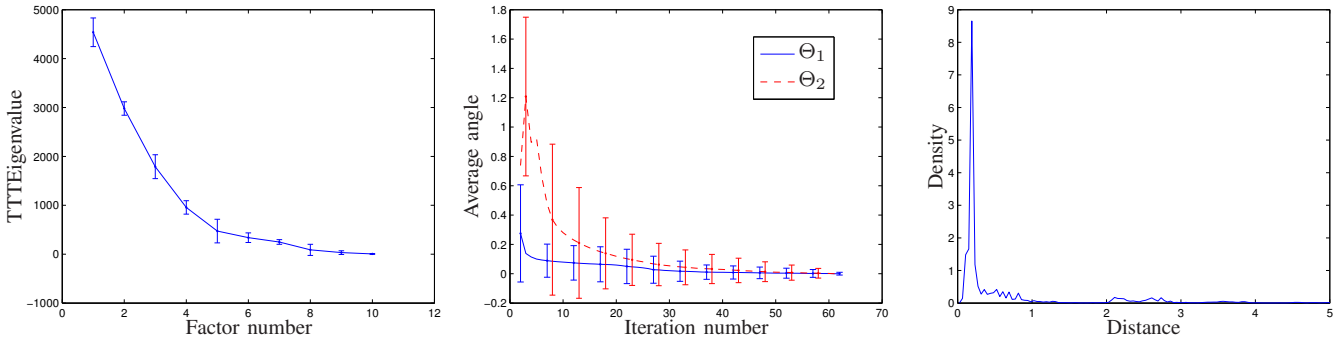
[1]http://codeen.cs.princeton.edu/slicestat/

B. Estimation Performance

In order to highlight the estimation performance of our approach, we consider in the following the results obtained for all the container activities at a given time θ. One can note that similar results have been obtained for any other θ.

The first estimation parameter we consider to setup the accuracy of our approach is the number of principal components required to feature the system workload in a satisfying way while maintaining a reasonable computation cost, as referred to Equation (8). The latter is depicted by the scree plot in Fig. 2.a which stands for the relationship between the relative magnitude of the eigenvalues and the number of factors. The number of principal components is traditionally determined by the abscissa value where the line stops descending precipitously and levels out. In our case, despite the fact that the curve exhibits a smooth decrease, it appears numerically that 2 principal components are enough since they restore about two thirds of the total variance of the original data while the 8 remaining only restore the last third. Consequently, we limit in the following of our study the workload estimation to the two first principal components.

In order to feature to what extent the decentralized CCIPCA approach we propose is able to accurately feature the two first



(a) (b) (c)

Fig. 2. Overall estimation performance: (a) Relationship between the relative magnitude of the eigenvalues and the number of factors; (b) Convergence of the proposed strategy, measured by the average angle Θ_1 and Θ_2; (c) Density of the distances between real data and its estimate

components as compared to a complete PCA computation, performed in a centralized way and considering the eigen-decomposition of the covariance matrix, we consider the estimation error as the angle between (1) the i^{th} principal axis $\boldsymbol{w}_i^*$, obtained from the complete PCA and (2) the i^{th} estimate $\boldsymbol{w}_{i,l}$ at probe l, namely

$$\Theta_i = \arccos\left(\frac{\boldsymbol{w}_{i,l}^\top \boldsymbol{w}_i^*}{\|\boldsymbol{w}_{i,l}\|\|\boldsymbol{w}_i^*\|}\right). \tag{12}$$

The results, depicted in Fig. 2.b, are averaged over all the probes hosted on monitored servers and they show the evolution of this estimation error according to the number of iteration of our algorithm with respect to Algorithm 1. For both components, after a few iterations where the error is large, one can see that they clearly converge toward a quasi-null error. From 28 iterations, the error is below 0.01 and 65 iterations are required to reach the $\epsilon = 0.001$ threshold, which stands for a satisfying performance result. One can remark that the confidence intervals of the second axis during the first iterations are larger than those of the first. The reason lies in the need for an accurate first axis to compute the second one in our algorithm, as stated in Equation (8).

Finally, in order to evaluate the overall estimation accuracy of our approach to feature the system workload, we consider the residual distance between the monitoring data projected into the two-dimensional axis space and the real data. Such a distance can be computed as follows:

$$\boldsymbol{d} = \|\boldsymbol{W}(\boldsymbol{W}^\top \boldsymbol{W})^{-1}\boldsymbol{Y} - \boldsymbol{X}\|, \tag{13}$$

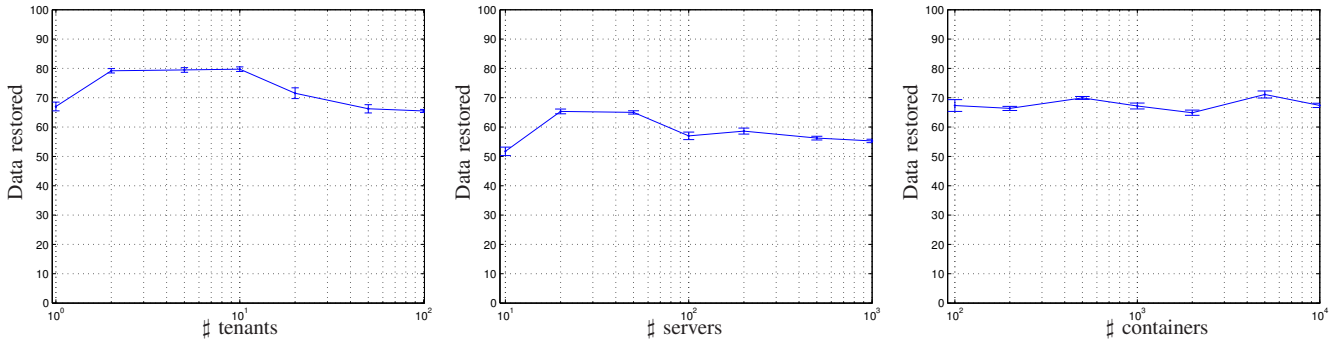
where $\boldsymbol{Y}$ is the matrix of the principal components. Fig. 2.c shows the density of the obtained results. It demonstrates the relevance of our method in representing the data, since 80% of all errors are clearly located below 1 and greater values only contain a insignificant part of the errors.

C. Scalability Support

In order to evaluate to what extend our approach supports scalability, we have considered two criteria which are the amount of variance restored by our estimation over the real data and the number of steps required to reach the algorithm convergence. The two latter respectively stand for the overall performance and cost indicators. As scalability factors, and with respect to realistic multi-tenant virtualized environments, we have selected subset of the whole dataset in order to change the number of tenants, varying from 1 to 100, the number of servers, from 10 to 1000 and finally the number of containers, varying from 100 to 10000, thus bringing two orders of magnitude for all the factors. The collected results are shown respectively in Fig. 3 and Fig. 4 for all the above mentioned indicators and factors. The overall result clearly shows that whatever the scale of the overall system, the estimation accuracy is almost constant with in average 70% of the data variance restored, thus leading us to the independence of our approach from the system size. Then, concerning the cost, it appears that the two decades highlight a linear cost of the approach according to the different scaling factors, thus here also demonstrating the acceptable cost of our approach under different scales. A logarithmic shape tends to appear but further results would be necessary to confirm that result. One can note that in case of small systems, our approach still needs a large number of iterations. This phenomenon can be explained by the difficulty of the method to compute its principal axes when the set of input data is too small making it useless for such environment.

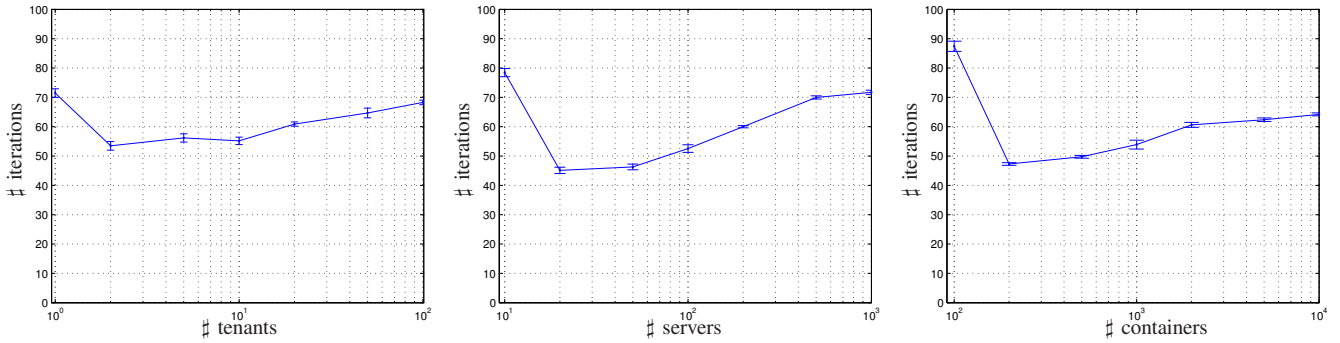
V. Conclusion

Being able to estimate the workload of a virtualized infrastructure in a cost-effective and accurate way is a mandatory step toward the design and implementation of advanced management and control functions related to performance or security. However, the large scale of these infrastructures which may count millions of virtual machines, the heterogenous nature of the activities they host and their dynamics in terms of operations and migrations, makes such an estimation hard to perform. In this paper, we have presented a novel estimation approach which proposes to incrementally compute the principal components of a workload in a decentralized



(a) (b) (c)

Fig. 3. Workload estimation accuracy featured by the percentage of data that lies within restored data according to the number of (a) tenants, (b) servers and (c) containers



(a) Impact of tenants number on PCA convergence (b) Impact of server number (c) Impact of container number

Fig. 4. Evolution of the number of iterations required to achieve the convergence in the proposed principal components estimation when increasing the number of (a) tenants, (b) servers and (c) containers

way, this way enabling any management element to get this information at any time. We have especially extended the CCIPCA [32] approach to allow it to (1) integrate the multi-tenancy of a virtualized infrastructure and (2) operate in a fully decentralized way. In order to validate our work, we have considered a large-scale dataset of Linux container activities. By implementing our approach in a simulation environment, we have demonstrated that (1) considering two principal components is enough to feature around two thirds of the workload variance of a virtualized infrastructure while maintaining a reasonable cost, given by the number of iterations required to reach the algorithm convergence; (2) the estimation accuracy is independent from the system scale in terms of number of tenants, servers and containers; (3) around 70 iterations are enough at worst to compute a precise estimation of a workload at the scale of 100 tenants, 1000 servers and 10000 containers and such a cost tends to grow in a linear way with the system scale, thus highlighting a constant cost for an individual element.

From the results we obtained in this work, the research perspectives are numerous. Firstly, our ongoing work consists in an in depth study of the computing cost of each iteration of our algorithm and the associated communication cost induced by gossip exchanges. We are also considering to what extent our workload estimation solution can automatically detect outliers activities related to malware execution in virtual machines. The first case we address concerns Botnets with highly diluted attacks mixed into a legitimate activity. Secondly, we plan to address other types of malicious activities such as data leakage. Lastly, in a long term perspective, we plan to compare the performance of such a decentralized approach to that of standard solution for big data computing such as Hadoop/Mahout.

ACKNOWLEDGEMENT

This work is co-funded by (1) the French Investment for Future (Développement de l'Economie Numérique), Request (REcursive QUEry and Scalable Technologies) project, started in 01/02/2014 and (2) the CRCA and FEDER CyberSec Platform $< D201304601 >$.

REFERENCES

[1] S. Krompass, A. Scholz, M. Albutiu, H. A. Kuno, J. L. Wiener, U. Dayal, and A. Kemper, "Quality of service-enabled management of database workloads," *IEEE Data Eng. Bull.*, vol. 31, no. 1, pp. 20–27, 2008.

[2] V. R. Kebande and H. S. Venter, "A cognitive approach for botnet detection using artificial immune system in the cloud," in *Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2014 Third Intl' Conf. on*, April 2014, pp. 52–57.

[3] M. R. Memarian, M. Conti, and V. Leppanen, "Eyecloud: A botcloud detection system," in *Trustcom/BigDataSE/ISPA, 2015 IEEE*, vol. 1, Aug 2015, pp. 1067–1072.

[4] I. T. Jolliffe, *Principal Component Analysis*. Springer Verlag, 1986.

[5] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[6] N. Ghadban, P. Honeine, F. Mourad-Chehade, C. Francis, and J. Farah, "In-network principal component analysis with diffusion strategies," *Intl' Journal of Wireless Information Networks*, vol. 23, no. 2, pp. 97–111, 2016.

[7] S. Wold, "Pattern recognition by means of disjoint principal components models," *Pattern recognition*, vol. 8, no. 3, pp. 127–139, 1976.

[8] C. Ding and X. He, "K-means clustering via principal component analysis," in *Proc. of the twenty-first international conference on Machine learning*. ACM, 2004, p. 29.

[9] K. Tout, R. Cogranne, and F. Retraint, "Fully automatic detection of anomalies on wheels surface using an adaptive accurate model and hypothesis testing theory," in *Signal Processing Conf. (EUSIPCO)*. IEEE, 2016, pp. 508–512.

[10] K. Tout, R. Cogranne, F. Retraint, "Fully automatic detection of anomalies using an adaptive statistical model and testing theory: Application to wheel surface inspection." *(submitted)*, 2017.

[11] B. M. Wise, N. L. Ricker, and D. J. Veltkamp, "Upset and sensor failure detection in multivariable processes," AlChE Meeting, San Francisco, 1989.

[12] J. MacGregor, "Multivariate statistical methods for monitoring large data sets from chemical processes," AlChE Meeting, San Francisco, 1989.

[13] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," *SIGMETRICS Perform. Eval. Rev.*, vol. 26, no. 1, pp. 151–160, Jun. 1998.

[14] A. B. Downey and D. G. Feitelson, "The elusive goal of workload characterization," *SIGMETRICS Perform. Eval. Rev.*, vol. 26, no. 4, pp. 14–29, Mar. 1999.

[15] M. C. Calzarossa, L. Massari, and D. Tessera, "Workload characterization: A survey revisited," *ACM Comput. Surv.*, vol. 48, no. 3, pp. 48:1–48:43, Feb. 2016.

[16] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das, "Towards characterizing cloud backend workloads: Insights from google compute clusters," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 4, pp. 34–41, Mar. 2010.

[17] Y. Chen, A. S. Ganapathi, R. Griffith, and R. H. Katz, "Analysis and lessons from a publicly available google cluster trace," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-95, Jun 2010.

[18] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *Proc. of the 2007 IEEE 10th Intl' Symposium on Workload Characterization*, ser. IISWC '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 171–180.

[19] A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam, and N. Sharma, "Towards autonomic workload provisioning for enterprise grids and clouds," in *2009 10th IEEE/ACM Intl' Conf. on Grid Computing*, Oct 2009, pp. 50–57.

[20] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd Ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[21] V. Cevher, S. Becker, and M. Schmidt, "Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 32–43, Sept 2014.

[22] D. Feldman, M. Schmidt, and C. Sohler, "Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering," in *Proc. of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '13. Philadelphia, PA, USA: SIAM, 2013, pp. 1434–1453.

[23] C. Alzate and J. A. K. Suykens, "Multiway spectral clustering with out-of-sample extensions through weighted kernel pca." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 2, pp. 335–347, 2010.

[24] J. R. Bunch and C. P. Nielsen, "Updating the singular value decomposition," *Numerische Mathematik*, vol. 31, pp. 111–129, 1978.

[25] P. M. Hall, A. D. Marshall, and R. R. Martin, "Merging and splitting eigenspace models." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 9, pp. 1042–1049, 2000.

[26] H. Kargupta, W. Huang, K. Sivakumar, B. Park, and S. Wang, "Collective principal component analysis from distributed, heterogeneous data," in *Proc. of the 4th European Conf. on Principles of Data Mining and Knowledge Discovery*. London, UK, UK: Springer-Verlag, 2000, pp. 452–457.

[27] Y. Le Borgne, S. Raybaud, and G. Bontempi, "Distributed principal component analysis for wireless sensor networks," *Sensors*, vol. 8, no. 8, pp. 4821–4850, 2008.

[28] S. B. Korada, A. Montanari, and S. Oh, "Gossip PCA," in *Proc. of the ACM SIGMETRICS Joint Intl' Conf. on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '11. New York, NY, USA: ACM, 2011, pp. 209–220.

[29] M. Ahmadi Livani and M. Abadi, "A pca-based distributed approach for intrusion detection in wireless sensor networks," in *Computer Networks and Distributed Systems (CNDS), Intl' Symposium on*, Feb 2011, pp. 55–60.

[30] S. V. Macua, P. Belanovic, and S. Zazo, "Consensus-based distributed principal component analysis in wireless sensor networks," in *IEEE Intl' Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, June 2010, pp. 1–5.

[31] A. Bertrand and M. Moonen, "Distributed adaptive estimation of covariance matrix eigenvectors in wireless sensor networks with application to distributed {PCA}," *Signal Processing*, vol. 104, pp. 120 – 135, 2014.

[32] W.-S. Hwang, J. Weng, and Y. Zhang, "Candid covariance-free incremental principal component analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 1034–1040, 2003.

[33] A. H. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z. J. Towfic, "Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior," *Signal Processing Magazine, IEEE*, vol. 30, no. 3, pp. 155–171, May 2013.

[34] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Trans. Comput. Syst.*, vol. 23, no. 3, pp. 219–252, Aug 2005. [Online]. Available: http://doi.acm.org/10.1145/1082469.1082470

[35] Y. Zhang and J. Weng, "Convergence analysis of complementary candid incremental principal component analysis," Comput. Sci. Eng., Michigan State Univ., East, Tech. Rep., 2001.

[36] F. Fagnani and S. Zampieri, "Asymmetric randomized gossip algorithms for consensus," in *IFAC World Conf.*, 2008, pp. 9052–9056.

[37] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2508–2530, June 2006.

[38] F. Fagnani and S. Zampieri, "Randomized consensus algorithms over large scale networks," *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 4, pp. 634–649, May 2008.

[39] H. Badis, G. Doyen, and R. Khatoun, "Understanding botclouds from a system perspective: A principal component analysis," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–9.

[40] O. Babaoglu, M. Marzolla, and M. Tamburini, "Design and implementation of a p2p cloud system," in *Proc. of the 27th Annual ACM Symposium on Applied Computing*, ser. SAC '12. New York, NY, USA: ACM, 2012, pp. 412–417.

[41] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: An overlay testbed for broad-coverage services," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, Jul. 2003.