Tensor-Based Resource Utilization Characterization in a Large-Scale Cloud Infrastructure

Waltenegus Dargie
waltenegus.dargie@tu-dresden.de
Technische Universität Dresden
Dresden, Saxony

ABSTRACT

The introduction of virtualization and cloud computing has enabled a large number of containers/virtual machines to share computing resources. Nevertheless, the number and size of data centres are still on the rise, partly on account of an ever increasing amount of generated data and workloads worldwide. On the other hand, independent studies indicate that a large number of servers in contemporary data centres are underutilised. One of the strategies currently adopted by the research community in order to deal with resource inefficiency is dynamic workload consolidation. The idea behind is dynamically balancing the supply of computing, communication, and storage resources with the demand for resources. This entails populating physical servers with an optimal number of complementary workloads. Most existing or proposed approaches employ multi-variate optimisation to achieve this goal but do not easily lend themselves to fast and intuitive solutions. In this paper, we investigate the scope and usefulness of dimensionality reduction techniques (tensor decomposition) to identify execution and resource utilisation patterns in hosted containers/virtual machines. Our analysis is based on two large-scale data centres, one of them hosts 1190 commercial virtual machines on 59 physical computing servers and 29 physical storage servers organised in 9 clusters and the other 44 373 containers on 3985 physical servers. Our analysis shows that "spatial" and "temporal" patters can be uncovered with tensor decomposition, based on which efficient clustering can be realised.

UCC'19, Dec 2019, Auckland, New Zeeland

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CCS CONCEPTS

• **Distributed Architecture** → **Cloud Computing**; • **Distributed systems organizing principles** → *Cloud Computing*; • **Computer systems organization** → Data Centers.

KEYWORDS

Cloud computing, consolidation, containers, data center workload, resource utilization, tensor decomposition, virtual machines

ACM Reference Format:
Waltenegus Dargie. 2019. Tensor-Based Resource Utilization Characterization in a Large-Scale Cloud Infrastructure. In *Proceedings of ACM Conference (UCC'19).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Contemporary data centres interconnect a large number of physical servers using data centre networks [15]. A high-level software architecture (virtualization) enables containers and virtual machines to share computing, storage, and/or communication resources without compromising on their privacy and security [10]. This approach has given rise to cloud computing and resulted in a number of advantages in terms of reduction of cost in infrastructure set-up and management, flexibility in dynamic adaptation of resource supply, and better security. It has also contributed in the reduction of the energy consumption of information and communication technologies worldwide [9]. Nevertheless, studies show that data centres often provide resources to their customers in excess in order to ensure that service-level agreements are respected [4, 14]. This practice inevitably leads to resource underutilisation and energy inefficiency.

Existing approaches attempting to address resource utilisation inefficiency aim at dynamic workload consolidation. Broadly speaking, this is carried out in three steps:

(1) The overall workload of the data centre is estimated to determine the number of physical servers needed.
(2) Containers/virtual machines are profiled according to their predominant resource demand.

(3) Physical servers are populated by containers/virtual machines exhibiting complementary resource utilisation characteristics.

The second step is critical to achieve high performance computing and to avoid resource overload or underutilisation. The assignment is challenging since the resource demand of hosted containers changes over time along multiple dimensions. In other words, containers which are complementary in one dimension may be contentious in another. Similarly, containers which are complementary at one time, may become contentious at another. In order to simplify the consolidation task, some researchers aim at optimising the utilisation of a handful of resources which consume a significantly large amount of energy (such as CPU cores) or which often become performance bottlenecks (such as the network and storage bandwidth).

In this paper, we propose the use of multi-way tensor decomposition and analysis in order to characterise the resource utilisation of hosted containers/virtual machines. This approach enables to achieve two opposing objectives at the same time. Firstly, it enables to efficiently process a large amount of statistical data pertaining to the resource utilisation of a large number of containers/virtual machines. Secondly, it enables to efficiently characterise both the temporal and the "spatial" aspect of resource utilisation. Originally arising in the fields of psychometrics and chemometrics [8], tensor decomposition has induced a great deal of interest in modelling and reasoning about complex relationships in a wide range of research areas. Its popularity lies in its capacity to:

- structure a large amount of data in a comprehensible way;
- exploit multi-dimensional correlations in order to significantly reduce the dimensions of the original data; and,
- extract latent features which can be examined from different vantage points.

Even though tensor decomposition and analysis appear to be the ideal strategies to deal with big data in cloud computing, to the best of our knowledge, ours is the first proposal to use these techniques for managing computing resources in data centres. Our analysis is based on statistics obtained from two independent data centres: One of them is the Enterprise Cloud Infrastructure (ECI) at the Centre for Information Services and High-Performance Computing, TU Dresden, Germany[1]. ECI consists of 59 physical computing servers and 29 physical storage servers organised into 9 clusters. At the time we took measurement, the data centre was hosting 1190 commercial virtual machines. The other belongs to one of Alibaba's Production Clusters (APC)[2] and consists of 3985 physical servers hosting 44 373 Linux containers (LXC).

The remaining part of the paper is organized as follows: In Section 3, we lay out the research goals this paper sets out to achieve. In Section 4, we introduce dimensionality reduction techniques, highlighting tensor decomposition and its relevance to characterise resource utilisation. In Section 5, we demonstrate how we apply tensor decomposition to analyse the resource utilisation characteristics of hosted virtual machines and containers in order to identify "spatial" and temporal characteristics. Finally, in Section 6, we provide concluding remarks and outline future work.

2 RELATED WORK

Existing or proposed approaches for managing data centre resources can be broadly classified as proactive or reactive. The former attempt to predict potential resource underutilisation or overload and consolidate workloads based on their complementary characteristics. The latter react to resource overload or "hotspots" by relocating the containers which give rise to it.

Curino et al. [2] propose Hydra, a large-scale container manager in a federated data centre. The design focus is on the cluster organisation and coordination by cleanly separating the concerns of demand prediction, resource appropriation, and task scheduling. Hence, the prevailing idea is setting up loosely-connected sub-clusters which can be organised into large clusters when demand grows and turned off in unison when demand reduces. This decision is managed by a control plane that can push scheduling policies across tens of thousands of nodes within seconds. The proposed approach works well for long-running batch workloads, the statistics of which is stable.

Pahlevan et al. [12] and Canali et al. [1] explore the usefulness of machine learning in identifying contentious or complementary virtual machines based on their utilisation statistics. The former combine unsupervised clustering and integer linear programming and observe that whereas the machine learning strategy yields a fast and comprehensible result, the more accurate, however, is the latter. The proposed approach is quite extensive, but users can tune the model, based on their preference, so as to achieve the desire trade-off between computation accuracy and speed. Canali et al., on the other hand, apply Principal Component Analysis in order to cluster virtual machines based on their resource consumption characteristics. They used traces obtained from a virtual testbed running on Amazon EC2 platform and a proprietary data centre. However, their analysis is limited

[1] https://tu-dresden.de/zih (Last accessed on 14 May 2019, 10:00 CET).

[2] https://github.com/alibaba/clusterdata/tree/master/cluster-trace-v2018 (Last accessed on 4 September 2019, 15:15 CET).

to simple cases (2-dimensional) and the number of virtual machines they considered are relatively small.

Haehnel et al. [7] extend the Cutting Stock Problem [5] for consolidating containers with stochastic workloads. They employ the aggregate probability density function of co-located and simultaneously executing jobs to establish valid patterns, where a valid pattern is one yielding an overall resource utilisation below a set threshold. The validation of the proposed approach was based on a 16-core server with 29 different benchmarks. The workloads of these benchmarks have been generated based on the CPU utilisation traces of 100 real-world containers from a Google data centre. Altogether, the authors considered 540 different consolidation scenarios and investigated system overload probability, job completion time, and energy consumption. However, the consolidation assignment considered only the CPU utilisation of the consolidated services.

Similarly, Nguyen et al. [11] employ multiple linear regression (Original Least Squares) to predict (1) the future use of a named resource by a virtual machine and (2) the potential of overloading the same resource. Virtual machine consolidation is triggered if a resource bottleneck is predicted. The proposed algorithm assesses the state of all the servers in the data centres before a consolidation process begins, so that a global balance is achieved between the demand for and the supply of resources. The authors employed real Google Data Centre traces both to train and test their model. However, the proposed approach is elaborate and works well only for linear relationships.

Ferdaus et al. [3] formulate the problem of virtual machine consolidation in data centres as a discrete combinatorial optimisation problem with the objective of minimizing data centre resource wastage, power consumption, and overall migration overhead. As a solution to the problem, the authors propose a scalable Ant Colony Optimization (ACO) meta-heuristic and a hierarchical, decentralized dynamic consolidation framework which localises migration-related network traffic and reduce network cost. Similarly, Tang et al. [13] propose a hybrid genetic algorithm for dynamic virtual machine consolidation in a hierarchical data centre. The algorithm extends the genetic algorithm by incorporating an infeasible solution repairing procedure and a local optimisation procedure in order to enhance the exploitation capacity and the convergence of the original genetic algorithm. The proposed algorithm exploits the well-structured (hierarchical) topology of the data centre to yield a consolidation result in a polynomial time. The validation of the algorithm was carried out by employing self-generated virtual machines.

Yu et al. [16] define a probabilistic threshold of exceeding the capacity of a server in a cloud environment. If this threshold is crossed, the server is labelled as a hotspot. Subsequently, they order the hotspots by decreasing overload risk. Then, the authors go through each hotspot, starting with the server with the highest overload risk. The virtual machines on each server are sorted as well as grouped by a decreasing metric describing the potential of reducing the server's overload risk. Finally, the virtual machines from the group with the highest potential but lowest migration cost are migrated until the server is no longer a hotspot. However, the authors focus only on overloaded servers as a result of which the algorithm optimises only locally.

Most of the approaches we reviewed in this section employ multi-variate optimisation to utilise resources efficiently but do not easily lend themselves to fast and intuitive solutions. In this paper, we investigate the scope and usefulness of dimensionality reduction techniques to identify execution patterns and resource utilisation complementarity in hosted virtual machines/containers. Our analysis offers multiple and intuitive vantage points ("spatial" as well as temporal) from which resource utilisation can be examined.

3 RESEARCH GOALS

In this paper, we strive to address the following research questions:

(1) Given a large set of hosted containers (virtual machines) having stochastic workloads and a corresponding set of computing resource demands (CPU, memory bandwidth, memory capacity, network bandwidth, storage size, disk read/write bandwidth, etc.), is it possible to identify containers/virtual machines having complementary as well as contentious resource utilisation characteristics?
(2) Considering the large amount of hosted containers and the large amount of statistics pertaining to resource utilisation, is it possible to develop analytic strategies which (a) are efficient to compute, (b) yield tractable solutions, and (c) are intuitive to identify workloads exhibiting higher-level features?
(3) Using the same sets of analytic tools and sets of data, is it possible to simultaneously uncover hidden, non-overlapping features and predict the temporal evolution of the resource demand of hosted containers/virtual machines? This aspect is useful for performing dynamic workload consolidation.

4 DIMENSIONALITY REDUCTION

The utilisation of a particular resource, say a CPU, by hosted containers in a data centre can be represented by a matrix, assuming that the samples are taken synchronously. The columns of the matrix can represent the sampling intervals (*time*) and the rows, the containers, or vice versa. How the matrix should be structured depends on what one wishes to achieve.

If one expects strong correlation between the samples along both dimensions (temporal correlation as well as synchronised execution amongst the containers), it is possible to decompose the matrix into more compact and basic matrices which reveal underlying execution features (patterns). One of the most widely used techniques is the Singular Value Decomposition (SVD) [6]. Thus, for a given utilisation matrix $\mathbf{R}$ having $M \times N$ dimensions, its SVD yields the following:

$$\mathbf{R} = \mathbf{U}\Sigma\mathbf{V}^{\mathsf{T}} \tag{1}$$

$\mathbf{U} \in \mathbb{R}^{M\times P}$ and $\mathbf{V} \in \mathbb{R}^{N\times P}$ are said to be orthogonal (uncorrelated) matrices and $\Sigma \in \mathbb{R}^{P\times P}$ is a diagonal matrix having entries which are naturally arranged according to their magnitude (i.e., $\sigma_{11} \geq \sigma_{22} \geq \ldots \sigma_{pp}$). In order to explain the significance of SVD, we shall give a simple example. Suppose we have the following utilisation matrix – assuming the rows represent the containers and the columns, the time intervals:

$$\mathbf{R} = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \end{array} \begin{array}{c} \begin{array}{ccccc} S_1 & S_2 & S_3 & S_4 & S_5 \end{array} \\ \begin{bmatrix} 1 & 5 & 9 & 13 & 17 \\ 2 & 6 & 10 & 14 & 18 \\ 3 & 7 & 11 & 15 & 19 \\ 4 & 8 & 12 & 16 & 20 \end{bmatrix} \end{array} \tag{2}$$

We chose the matrix deliberately to demonstrate the existence of hidden features in it. As can be seen, if we concatenate the columns of the matrix, they result in a sequence of whole numbers ranging from 1 to 20. So, one of the hidden features of this matrix is that it consists of a sequence of whole numbers. The other hidden feature is that the sequence is divided into five columns. Applying SVD on this matrix yields the following basic matrices:

$$\mathbf{U} = \begin{bmatrix} -0.44 & -0.71 \\ -0.48 & -0.26 \\ -0.52 & 0.18 \\ -0.55 & 0.63 \end{bmatrix} \tag{3}$$

$$\Sigma = \begin{bmatrix} 54 & 0 \\ 0 & 2 \end{bmatrix} \tag{4}$$

$$\mathbf{V} = \begin{bmatrix} -0.10 & 0.77 \\ -0.25 & 0.49 \\ -0.39 & 0.21 \\ -0.54 & -0.07 \\ -0.69 & -0.35 \end{bmatrix} \tag{5}$$

Notice the entries of Σ. It is not by accident that there are only two diagonal entries, as the original matrix contains only two hidden features. Moreover, notice that the two diagonal entries of Σ do not have equal significance. The first is more significant than the second. What will happen if we carry out the following matrix operation?

$$\hat{\mathbf{R}} = \sigma_{11}\left(\mathbf{u}_1 \circ \mathbf{v}_1^{\mathsf{T}}\right) \tag{6}$$

where σ_{11} is the first element in Σ and $\circ$ refers to an outer product between the first column of $\mathbf{U}$ and the transpose of the first column of $\mathbf{V}$:

$$\hat{\mathbf{R}} = 54 \times \begin{bmatrix} -0.44 \\ -0.48 \\ -0.52 \\ -0.55 \end{bmatrix} \begin{bmatrix} -0.10 & -0.25 & -0.39 & -0.54 & -0.69 \end{bmatrix} \tag{7}$$

The result is the following:

$$\hat{\mathbf{R}} = \begin{bmatrix} 2.29 & 5.82 & 9.35 & 12.89 & 16.42 \\ 2.48 & 6.31 & 10.13 & 13.96 & 17.78 \\ 2.67 & 6.79 & 10.91 & 15.03 & 19.15 \\ 2.86 & 7.27 & 11.69 & 16.10 & 20.51 \end{bmatrix} \tag{8}$$

Clearly, there is a strong similarity between $\hat{\mathbf{R}}$ and $\mathbf{R}$. If we compute the difference between the two matrices, the result is the following:

$$\mathbf{e} = \mathbf{R} - \hat{\mathbf{R}} = \begin{bmatrix} -1.29 & -0.82 & -0.35 & 0.11 & 0.58 \\ -0.48 & -0.31 & -0.13 & 0.04 & 0.22 \\ 0.33 & 0.21 & 0.09 & -0.03 & -0.15 \\ 1.14 & 0.73 & 0.31 & -0.10 & -0.51 \end{bmatrix} \tag{9}$$

So, apart from uncovering hidden features in $\mathbf{R}$, we can also reconstruct the original matrix by just considering a single column in $\mathbf{U}$ and a single column in $\mathbf{V}$. For a large raw matrix, the second aspect implies that SVD considerably reduces our storage demand to save $\mathbf{R}$. To summarise some of the advantages of SVD:

(1) Firstly, one does not need to make any assumption as regards the hidden features. Their number and significance is dynamically revealed by the diagonal matrix Σ.
(2) Secondly, one can express the original utilisation matrix $\mathbf{R}$ as the outer products of the columns of $\mathbf{U}$ and $\mathbf{V}$ (refer also to Fig. 1):

$$\mathbf{R} = \sum_{p=1}^{P} \sigma_{pp}\mathbf{u}_p \circ \mathbf{v}_p \tag{10}$$

where $\mathbf{u}_r$ and $\mathbf{v}_r$ refer to the r-th column of the matrices $\mathbf{U}$ and $\mathbf{V}$, respectively. We refer each matrix on

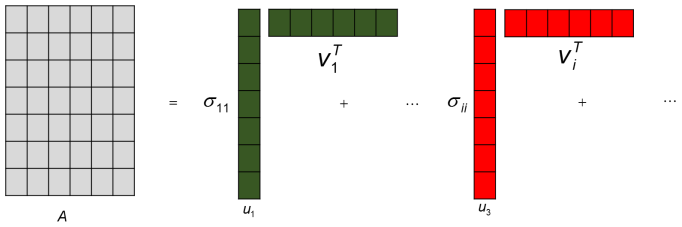

Figure 1: Expressing the utilisation matrix as the summation of SVD components.

the right side as a component. Note that the relevance of each component is associated with the relevance of σ_{ii}.

(3) Thirdly, if the samples of the utilisation matrix exhibit strong correlations, then, $\mathbf{R}$ can be approximated by taking the first K components only:

$$\mathbf{R} \approx \sum_{p=1}^{K} \sigma_{pp} \mathbf{u}_p \circ \mathbf{v}_p \tag{11}$$

for $K < P$. If the difference in magnitude between the successive σ_{rr} entries is significantly large, then a strong correlation is identified in the original utilisation matrix and, hence, the error resulting from our approximation will be significantly small.

4.1 Utilisation Tensor

One of the limitations of working with SVD is that it is two dimensional[3]. This forces us to analyse the utilisation of a single resource at a time. If we wish to analyse the utilisation of multiple resources using SVD, we have to analyse the average utilisation. But the average utilisation disregards the temporal variations of resource utilisation and leads to a considerable resource overload or underutilisation should the containers be consolidated without the knowledge of this aspect.

The most plausible alternative is to model resource utilisation using a three-way tensor, as shown in Figure 2. As can be seen, the tensor is a three-dimensional array consisting of elements intersecting three orthogonal axes. Hence, in the same way every element of a matrix can be referred to by two indices (the row index i and the column index j), every element of the tensor can be referred to by three indices. So, for example, r_{ijk} refers to the utilisation of the j-th element by the i-th VM in the k-th time slot. Now we have three dimensions along which we can search for hidden features (and, hence, three degrees-of-freedom to cluster the containers).

In the same way a matrix can be decomposed (factorised) into basic constituting elements, a tensor can be decomposed into basic constituting elements. However, unlike decomposing a matrix, decomposing a tensor is not straightforward. To start with, an assumption has to be made about the number of hidden factors embedded in the original tensor, whereas this is done automatically with SVD. Secondly, different decomposition strategies employ different statistical estimation techniques, which may result in different outcomes.

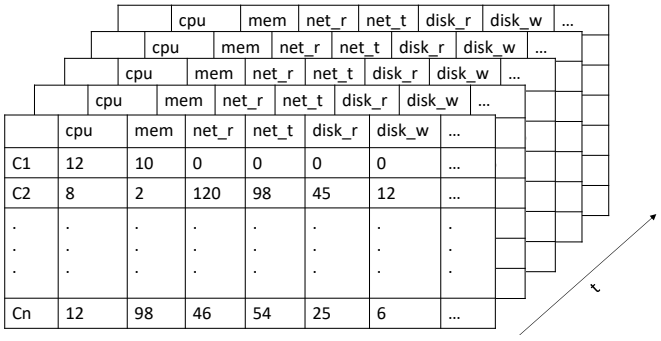

Figure 2: A three-way tensor representing the resource utilisation statistics of hosted containers.

A closer look into the utilisation tensor reveals that it provides three orthogonal views which can serve different purposes. For example, the front view (borrowing an expression from architecture) provides a matrix describing the utilisation of all resources by all hosted containers at the k-th sampling interval – i.e, (container versus resources)$_k$. This view is called the front slice. Likewise, the top view provides a matrix describing the utilisation of all resources by the i-th virtual machine over a period of time – i.e., (resources vs. time)$_i$. This is called the horizontal slice. Finally, the side view provides a matrix describing the utilisation of the j-th resource by all containers over a period of time – i.e., (container vs. time)$_j$. This is called the lateral slice. It is this flexibility, among others, which makes a tensor desirable.

4.2 Tensor Decomposition

The chief task of a tensor decomposition is to identify multi-dimensional features in terms of which the containers can be explained (categorized). Compared to the size of the tensor, the basic features should be significantly small in size, so that the clustering process is computationally tractable. A tensor analysis begins by unfolding (flattening) the tensor into a matrix. The unfolding can take place in different ways, but whichever way is chosen, the entries along each dimension form a column vector. For example, let the first three slices (i.e., the samples of the first three time instances) of the tensor $\mathcal{R} \in \mathbb{R}^{I \times J \times K}$ – where I is the number of containers, J the number of computing resources, and K, the number of

[3]One important aspect to notice is that in carrying out SVD the row dimension of the original matrix $\mathbf{R}$ is preserved in the $\mathbf{U}$ matrix and the column dimension in the $\mathbf{V}$ matrix. In other words, the $\mathbf{U}$ matrix encodes the relationship of the hidden features with the row variables (the containers) and the $\mathbf{V}$ matrix encodes the relationship of the hidden features with the column variables (the samples intervals).

sample points – are represented as follows:

$$\mathbf{R}_1 = \begin{bmatrix} 0.76 & -0.56 & 1.44 & \cdots \\ 1.66 & 3.24 & -0.78 & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$
$$\mathbf{R}_2 = \begin{bmatrix} 1.45 & 1.65 & -1.24 & \cdots \\ 0.12 & 1.23 & -0.86 & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad (12)$$
$$\mathbf{R}_3 = \begin{bmatrix} 0.26 & 1.48 & 0.01 & \cdots \\ -0.50 & 1.18 & 0.18 & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

The mode-1 unfolding of the above tensor takes each column vector as they are and put them together side-by-side:

$$\mathbf{R}_{(1)} = \begin{bmatrix} 0.76 & \cdots & 1.45 & \cdots & 0.26 & \cdots \\ 1.66 & \cdots & 0.12 & \cdots & -0.50 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad (13)$$

The mode-2 unfolding takes each raw entry of the matrices and places them as column vectors in a single matrix:

$$\mathbf{R}_{(2)} = \begin{bmatrix} 0.76 & \ldots & 1.45 & \cdots & 0.26 & \cdots \\ -0.56 & \ldots & 1.65 & \cdots & 1.48 & \cdots \\ 1.44 & \ldots & -1.24 & \cdots & 0.01 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad (14)$$

Likewise, the mode-3 unfolding takes the n-th entry of each slice (along the time dimension) and puts them together as column matrix:

$$\mathbf{R}_{(3)} = \begin{bmatrix} 0.76 & 1.66 & -0.56 & \cdots & -0.78 & \cdots \\ 1.45 & 1.12 & 1.45 & \cdots & -0.86 & \cdots \\ 0.26 & 1.18 & 1.48 & \cdots & 0.18 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad (15)$$

There are different tensor decomposition strategies; which of them is suitable for a particular decomposition depends on what we wish to achieve. We chose to employ the Canonical Decomposition/Parameter Factorisation (referred in the literature as CANDECOM/PARAFAC, or, in short, CP) [8] because it is intuitive to interpret. It decomposes a tensor into three matrices:

$$\mathcal{R} = \mathbf{ABC} \quad (16)$$

or

$$\mathcal{R} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \quad (17)$$

where $\mathbf{a}_r$, $\mathbf{b}_r$, and $\mathbf{c}_r$, are the r-th columns of the matrices **A B**, and **C**, respectively. In the existence of a strong correlation in the utilised resources, the utilisation tensor can be approximated only by the outer product of the first K column vectors of the matrices **A**, **B**, and **C**, respectively.

The three basic matrices have the following significance: The matrix **A** characterises the hosted containers in terms of the unique features. The matrix **B** associates the unique features with the resources utilised and the matrix **C** reveals the temporal characteristics of the containers without explicitly referring to the particular resources they utilise. For the consolidation task, the most relevant matrices are **A** and **C**, because the former reveals the "spatial" characteristics whereas the latter reveals the "temporal" characteristics of the containers. However, the matrix **B** is vital to understand or interpret the unique (hidden) features, as we shall demonstrate in the next section.

5 EVALUATION

The VMware managing ECI provides a large number of key performance indicators (KPI) to monitor resource utilisation. We selected 14 of these (listed in Table 1) to build our utilisation tensor. APC, on the other hand, provides 9 KPI, but two of these – cycle per instruction and Missed Predictions per 1000 (=Kilo) Instructions – are applicable only to monitoring the physical machines and one of them – machine ID – is invariant. The other – timestamps – does not refer to a specific resource. Therefore, we selected the five KPIs – CPU, mem, net in, net out, and disk IO – to construct our utilisation tensor. The ECI tensor contains a 24-hour period utilisation trace sampled every 5 minutes (288 samples per virtual machine per resource) whereas the APC tensor contains 8 days of measurements, also sampled every 5 minutes (960 samples per container per resource). In order to make our analysis comprehensible, we randomly selected 45 virtual machines from ECI and 1000 LXC from APC. So, for the first, our utilisation tensor has a dimension of $45 \times 14 \times 288$, and for the second, $1000 \times 5 \times 960$.

The ECI utilisation metrics have different units and scales. This tends to produce bias in favour of metrics having large numbers during the computation of correlations (underlying the tensor decomposition). To remove the effect of this bias, we normalised the samples, so that they all vary in the same range (i.e., between 0 and 1) using `min-max` feature scaling [17]. The APC KPIs had already been normalised when we obtained the measurements.

Table 1: A summary of the utilisation metrics used to construct the ECI utilisation tensor.

ID	Metric	ID	Metric
1	Average CPU usage (in MHz)	2	Average CPU workload
3	MEM usage	4	NET transmit average
5	NET received average	6	NET broadcast TX summation
7	Storage total read latency	8	Storage total write latency
9	Data store read average	10	Data store write average
11	Virtual disk read average	12	Virtual disk write average
13	Disk read average	14	Disk write average

5.1 Hidden Features

The type and magnitude of resources a virtual machine or a LXC utilises at any given time change over time. Likewise, the type of resources for which containers contend and the level of contention arising as a result change as well. Furthermore, some resources are not utilised in isolation. For example, the utilisation of a memory bandwidth involves the CPU and the memory; the utilisation of a network bandwidth involves the utilisation of the memory and, to some extent, the CPU. Similarly, the utilisation of a virtual storage involves the memory. It is this dependency the tensor decomposition exploits in order to uncover distinct utilisation features.

Before a tensor decomposition takes place, one has to estimate the number of unique features (factors) hidden in the original tensor. In order to ensure that we uncover all the hidden features relevant to characterise the hosted containers, we examined the coefficient of determination (R^2) for different number of factors[4]. For the ECI tensor, the CANDECOMP/PARAFAC decomposition yielded a reconstruction accuracy of 99 % ($R^2 = 0.996$) when the number of factors were set just to 2 but for the APC, the same accuracy could be achieved when the number of factors were set to three. Thus, we decided to decompose the utilisation tensors by assuming that only two hidden factors in the first tensor and three in the second tensor were embedded in the original tensors. This resulted in a 45×2 **A** matrix (VMs vs hidden features), a 14×2 **B** matrix (resources vs hidden features) and 288×2 **C** matrix (samples vs hidden features) as a consequence of the decomposition of the decomposition of the first tensor and 1000×3 **A** matrix (containers vs hidden features), a 5×3 **B** matrix (resources vs hidden features) and 960×3 **C** matrix (samples vs hidden features) as a consequence of the decomposition of the second tensor. Hence, besides its usefulness in uncovering hidden execution patterns, the tensor decomposition significantly reduced the sample size we deal with, from $45 \times 14 \times 288 = 181,440$ to $(45 \times 2) + (14 \times 2) + (288 \times 2) = 694$ for the first tensor and from $1000 \times 5 \times 960 = 4,800,000$ to $(1000 \times 3) + (5 \times 3) + (960 \times 3) = 5895$ for the second tensor.

5.2 Interpretation

Since we wish to characterise the VMs according to the resources they utilise, we first examine the **B** matrix in order to determine what the hidden factors actually represent. This is because the **B** matrix describes the relationship of the hidden features with the resources utilised. In Fig. 3, the two hidden factors are plotted against the resource types. It is

[4]The reconstruction of the original tensor from the decomposed matrices entails some error. One way of measuring the difference between the reconstructed tensor and the original tensor is by using the coefficient of determination.

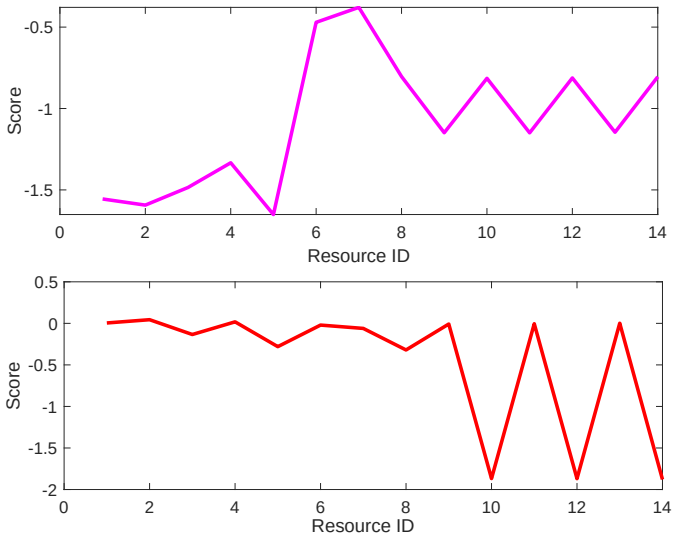

Figure 3: The B matrix explaining the contribution of the key performance indicators in uncovering the hidden features in the ECI utilisation tensor. Top: The First Factor (signifying dominant *write* operations). Bottom: The Second Factor (signifying dominant *read* operations).

worth to remark that we intentionally organised the *read* and *write* operations in Table 1 to make the interpretation of the graphs intuitive; the *read* operations are listed in the left column (odd numbers) and the *write* operations, in the right (even numbers).

The score distribution in the top plot (produced from the first column of the **B** matrix) can be categorised into three groups. In the first group, we find the utilisation metrics with IDs: $\{1, 2, 3, 4, 5\}$. These metrics , corresponding to compute and network activities, scored relatively low. In the second group, we have utilisation metrics with IDs: $\{9, 11, 13\}$ receiving modestly high scores (these are *read* operations). In the third group, we have utilisation metrics with IDs: $\{6, 7, 8, 10, 12, 14\}$ which received the highest scores. Referring to Tab. 1, the metrics which scored the highest refer essentially to *write* operations. So we can conclude that the hidden feature refers predominantly to *write-intensive* operations having almost no compute operations. Likewise, the second feature (produced from the second column of the **B** matrix) describes predominantly *read-* and *compute-intensive* operations containing almost no *write* operations.

Once we determined what utilisation aspects the two hidden factors uncovered, the next step is determining which of the VMs are classified as *write-intensive* and which of them as *read-intensive*. This can be determined by examining the two columns of the **A** matrix, since they encode the relationship between the VMs and the hidden factors. Fig. 4 displays the

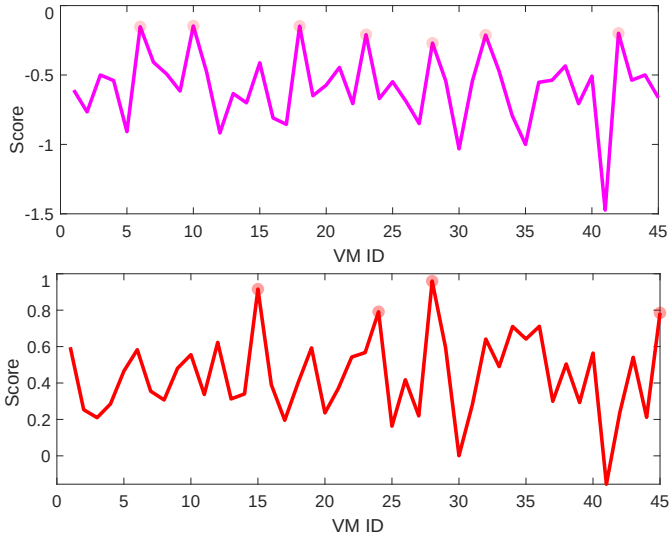

Figure 4: The scores in the A matrix explaining the utilisation characteristics of the virtual machines hosted by ECI in terms of the hidden features. Top: Predominantly *write-intensive* VMs. Bottom: Predominantly *read-intensive* VMs.

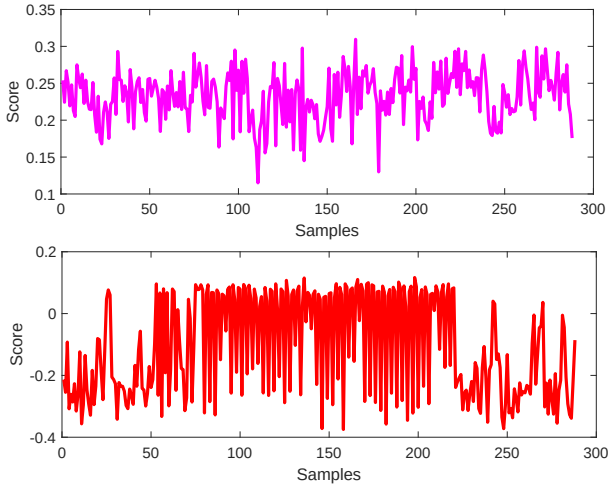

Figure 5: The scores in the C matrix explaining the temporal aspects of the write-intensive (top) and the read-intensive (bottom) operations.

contents of the **A** matrix. Accordingly, the first columns of the **A** matrix describes how *write-intensive* the hosted VMs are. The VMs which attained the highest scores are with IDs: {6, 10, 18, 23, 28, 32, 42}. These VMs are undoubtedly *write-intensive* VMs. Similarly, the bottom figure displays the scores the VMs attained in the second feature. Thus, we can conclude that VMs with IDs: {15, 24, 28, 45} are predominantly *read-intensive* VMs. Our analysis is consistent with the annual report of the data centre which states that in 2018, its approximate average annual resource utilisation was: 40 % CPU, 55 % MEM and 91 % disk.

Our characterisation of the virtual machines cannot be complete without closely examining their temporal execution characteristics, which is encoded in the **C** matrix (displayed in Fig. 5). As can be seen, there was a *write* operation (top, produced from the first column of the *C* matrix) present throughout the day and the night, albeit with modest fluctuation in intensity. By contrast, the *read* operations (produced from the second column of the **C** matrix) took place conspicuously during the day time with a markedly elevated activity between 6:30 AM and 8:00 PM, even though there were sporadic activities in the night as well.

Figure 6 displays a complete description of the decomposed utilisation tensor for APC. Each colour depicts one hidden feature. On the top are displayed the contributions of the utilised resources in describing the three hidden features (components of the **B** matrix). The middle three plots describe the hosted containers in terms of the hidden features (components of the **A** matrix); and, finally, the bottom three plots describe the temporal characteristics of the three hidden features (components of the **C** matrix). Thus, the hosted containers can be categorised into three clusters: The first are predominantly *compute-* and *IO-intensive* (disk). The second are predominantly *IO-intensive* (NET), and the third are *MEM-intensive*. A careful examination of the relationship of the hidden features with the samples (temporal aspect) reveals an interesting aspect. Whereas the *mem-intensive* containers persist throughout the weak (bottom left, green plot), the io-intensive (NET) containers have a periodic pattern (bottom middle, red plot). Likewise, the activities of the *compute-intensive* containers gradually increased in the beginning and made a steep increment afterwards, remaining intensive for the rest of the week (bottom right, pink plot).

5.3 Consolidation

As we have seen, the result of the tensor decomposition provides a complete and comprehensive view of the "spatial" and temporal utilisation characteristics of the hosted containers (virtual machines). This enables us to determine which of them are contentious and which complementary. For example, for the ECI, we can deduce that VMs {6, 10, 18, 23, 28, 32, 42} are contentious, since of all them scored high as *write-intensive* jobs. As a result, co-locating these VMs will result in high disk access latency. On the other hand, these VMs have very low CPU, MEM, and NET demand whereas VMs {15, 24, 28, 45} are *compute-* and *read-intensive* jobs. Hence, combining the VMs of the first set with the VMs of the second set will result in complementary resource utilisation. Furthermore, by

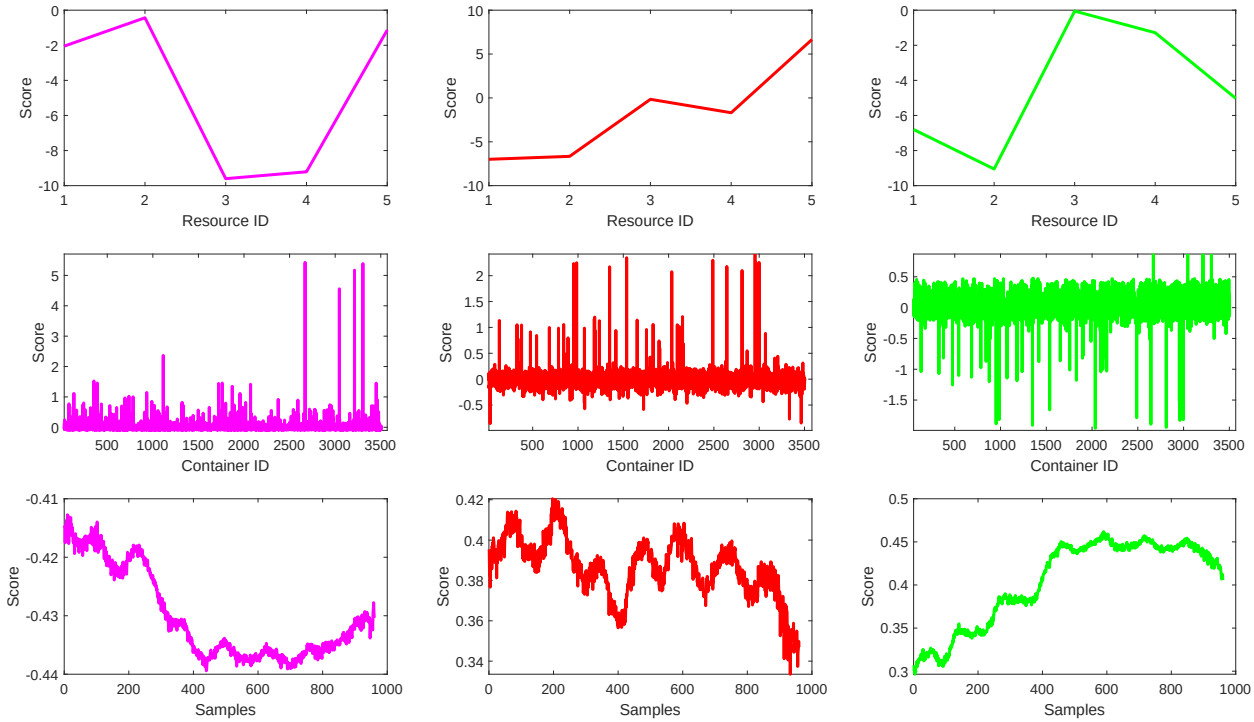

Figure 6: The results of the tensor decomposition applied on APC's utilisation tensor. Top: components of the B matrix. Middle: components of the A matrix. Bottom: components of the C matrix.

jointly examining the **A** and **C** matrices, it is possible to gain additional insights along the "spatial" and temporal dimensions. Similarly, we can cluster the APC LXCs, into three disjoint groups. In the first we have, for example, LXCs with IDs: {134, 353, 391, 694, 783, 933, 1114, 1321, 1721, 1871, 1956, 2072, 2375, 2591, 2670, 3212, 3304, 3463, 3450, 3596}. In the second, we have: {985, 1349, 1537, 2035, 2488, 2954, 2811, 2643}. In the third,we have: {2670, 3046, 3212, 3304}. Hosting each cluster on the same machines leads to a significant contention, whereas mixing containers from the three clusters leads to complementary resource utilisation which not only improves the performance of the containers but also ensures that all the resources are utilised with comparable efficiency, so that no resource utilises power without accomplishing any work.

6 CONCLUSION

In this paper we proposed tensor decomposition to investigate the existence of complementary and contentious characteristics in resource utilisation amongst hosted containers and virtual machines in large-scale data centres. We demonstrated that tensors can offer up to three degrees-of-freedom in analysing "spatial" and temporal aspects. A tensor decomposition dissolves a raw utilisation tensor into basic, constituting matrices in which each dimension of the original tensor is explained by underlying (hidden) factors. Hence, for our case, we constructed three-way tensors in which statistics pertaining to containers and utilised resources are recorded. The decomposition of these tensors using the CANDECOMP/PARAFAC decomposition technique yielded three constituting matrices, namely, the **A** matrix explaining the relationship of the containers with the hidden factors (i.e., *read-intensive*, *compute-intensive*, *write-intensive*, etc.); the **B** matrix explaining the relationship of the computing resources with the hidden factors; and the **C** matrix explaining the temporal characteristics of the hidden factors (*day-time* vs. *night-time* activities, *daily-* vs. *weakly-pattern*, etc.).

We demonstrated the usefulness of tensor decomposition by constructing two utilisation tensors based on traces we obtained from two independent data centres. The next step will be to automate the selection of containers from mutually exclusive clusters and consolidate them. Furthermore, the gains of this approach have to quantified in terms of performance, resource utilisation efficiency and energy consumption. This will be our future work.

REFERENCES

[1] Claudia Canali and Riccardo Lancellotti. 2014. Improving scalability of cloud monitoring through PCA-based clustering of virtual machines. *Journal of Computer Science and Technology* 29, 1 (2014), 38–52.

[2] Carlo Curino, Subru Krishnan, Konstantinos Karanasos, Sriram Rao, Giovanni Matteo Fumarola, Botong Huang, Kishore Chaliparambil, Arun Suresh, Young Chen, Solom Heddaya, et al. 2019. Hydra: a federated resource manager for data-center scale analytics.. In *NSDI*. 177–192.

[3] Md Hasanul Ferdaus, Manzur Murshed, Rodrigo N Calheiros, and Rajkumar Buyya. 2014. Virtual machine consolidation in cloud data centers using ACO metaheuristic. In *European Conference on Parallel Processing*. Springer, 306–317.

[4] Mostafa Ghobaei-Arani, Sam Jabbehdari, and Mohammad Ali Pourmina. 2018. An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach. *Future Generation Computer Systems* 78 (2018), 191–210.

[5] Paul C Gilmore and Ralph E Gomory. 1961. A linear programming approach to the cutting-stock problem. *Operations research* 9, 6 (1961), 849–859.

[6] Gene H Golub and Christian Reinsch. 1971. Singular value decomposition and least squares solutions. In *Linear Algebra*. Springer, 134–151.

[7] Markus Haehnel, John Martinovic, Guntram Scheithauer, Andreas Fischer, Alexander Schill, and Waltenegus Dargie. 2018. Extending the Cutting Stock Problem for Consolidating Services with Stochastic Workloads. *IEEE Transactions on Parallel and Distributed Systems* (2018).

[8] Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review* 51, 3 (2009), 455–500.

[9] Christoph Möbius, Waltenegus Dargie, and Alexander Schill. 2013. Power consumption estimation models for processors, virtual machines, and servers. *IEEE Transactions on Parallel and Distributed Systems* 25, 6 (2013), 1600–1614.

[10] Roberto Morabito, Vittorio Cozzolino, Aaron Yi Ding, Nicklas Beijar, and Jorg Ott. 2018. Consolidate IoT edge computing with lightweight virtualization. *IEEE Network* 32, 1 (2018), 102–111.

[11] Trung Hieu Nguyen, Mario Di Francesco, and Antti Yla-Jaaski. 2017. Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data centers. *IEEE Transactions on Services Computing* (2017).

[12] Ali Pahlevan, Xiaoyu Qu, Marina Zapater, and David Atienza. 2017. Integrating Heuristic and Machine-Learning Methods for Efficient Virtual Machine Allocation in Data Centers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2017).

[13] Maolin Tang and Shenchen Pan. 2015. A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers. *Neural Processing Letters* 41, 2 (2015), 211–221.

[14] Adel Nadjaran Toosi, Richard O Sinnott, and Rajkumar Buyya. 2018. Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using Aneka. *Future Generation Computer Systems* 79 (2018), 765–775.

[15] Thang X Vu, Symeon Chatzinotas, and Bjorn Ottersten. 2018. Edge-caching wireless networks: Performance analysis and optimization. *IEEE Transactions on Wireless Communications* 17, 4 (2018), 2827–2839.

[16] Lei Yu, Liuhua Chen, Zhipeng Cai, Haiying Shen, Yi Liang, and Yi Pan. 2016. Stochastic Load Balancing for Virtual Resource Management in Datacenters. *IEEE Transactions on Cloud Computing* PP, 99 (2016), 1–1. https://doi.org/10.1109/TCC.2016.2525984

[17] Alice Zheng and Amanda Casari. 2018. *Feature engineering for machine learning: principles and techniques for data scientists.* " O'Reilly Media, Inc.".