Scalable Correlation-aware Virtual Machine Consolidation Using Two-phase Clustering

Xi Li, Anthony Ventresque, Jesus Omana Iglesias, John Murphy
Lero and School of Computer Science and Informatics
University College Dublin, Dublin, Ireland
Email: {xi.li, jesus.omana-iglesias.1}@ucdconnect.ie; {anthony.ventresque, j.murphy}@ucd.ie

***Abstract*— Server consolidation is the most common and effective method to save energy and increase resource utilization in data centers, and virtual machine (VM) placement is the usual way of achieving server consolidation. VM placement is however challenging given the scale of IT infrastructures nowadays and the risk of resource contention among co-located VMs after consolidation. Therefore, the correlation among VMs to be co-located need to be considered. However, existing solutions do not address the scalability issue that arise once the number of VMs increases to an order of magnitude that makes it unrealistic to calculate the correlation between each pair of VMs. In this paper, we propose a correlation-aware VM consolidation solution *ScalCCon*[1], which uses a novel two-phase clustering scheme to address the aforementioned scalability problem. We propose and demonstrate the benefits of using the two-phase clustering scheme in comparison to solutions using one-phase clustering (up to 84% reduction of execution time when $17,446$ VMs are considered). Moreover, our solution manages to reduce the number of PMs required, as well as the number of performance violations, compared to existing correlation-based approaches.**

***Keywords*—Scalability; Correlation; Consolidation; Clustering; Performance degradation; VM placement;**

I. Introduction

The sustained increasing demand of computing resources has driven the growth of data center scale in recent years, while the underutilization of servers within these data centers has also received more and more attention as it leads to enormous resource and energy waste. With the adoption of virtualization technology, the number of servers in use can be reduced by consolidating multiple virtual machines (VMs) on one physical machine (PM) in order to improve the resource utilization efficiency and reduce power consumption. Virtualization is achieved by hypervisors or virtual machine monitors (e.g., Citrix XenServer, VMware ESX/ESXi, Microsoft Hyper-V), which is a layer between the operating system instances and hardware that partitions a PM's physical resources to provide performance isolation between co-located VMs [1]. Usually, the initial memory allocation and disk capacity are statically partitioned, and the sharing of CPU is taken care by the CPU scheduler. However, several resources, such as cache and I/O bandwidth, cannot be entirely isolated by hypervisors, and could cause contention among co-located VMs sharing these resources [2], [3]. Such contention can result in application performance degradation and service level agreement (SLA) violation. Contentions are common when a PM is hosting a large number of VMs. Therefore, there is a trade-off between the packing density of VMs and the VMs' performance.

A common strategy to tolerate the performance degradation due to VM co-location contention is to reserve a fixed proportion of a server's capacity as a buffer [4], [5]. This simple strategy has been proven to be efficient and easy to implement. However, the reserved resources can be utilized more efficiently with a deeper understanding of the VMs' resources utilization. A significant number of researchers [3], [6]–[16] have focused on analyzing and controlling the performance impact caused by VM contention at different levels and from various perspectives. A recent study, [2], summarizes the solutions for mitigating performance overhead (i.e., unpredictable performance) in a single server into two types: resource isolation among co-located VMs and optimization of VM assignment. The first type is usually realized by resource schedulers, and is complementary to the second type as it does not determine where to place the VMs. Our paper focuses on the second type of solutions and addresses the problem of *planning VM placement to both minimize the number of PMs required and alleviate the performance degradation caused by VM co-location contention*.

A common approach to address this problem is based on the model of performance interference among co-located VMs. This approach is cost-efficient when the model is accurate, which requires pre-running each VM with background benchmark workloads, while varying the background workloads' resource utilization in enviroments same as the target PMs, to train the model. This is obviously only applicable when there are a few varieties of workloads on homogeneous PMs. Another type of approach leverages the fact that VMs with strongly correlated resource utilization are more likely to have utilization peaks that coincide and to cause resource contention. This type of approach provides an estimation of the performance interference between two VMs instead of an accurate value, but it is considered sufficient in guiding an appropriate VM placement [2]. Moreover, this type of approach is more practical than constructing accurate interference models in large-scale data centers, because it requires only the VMs' historical utilization information or the predicted future utilization based on the historical utilization, but does not

[1] Scalable Correlation-aware Consolidation

require pre-running each VM in a particular environment for model training. However, one drawback of most of the existing correlation-based approaches is that the *Pearson correlation coefficient* for each pair of VMs needs to be computed. Thus, the size of the correlation matrix becomes extremely large as the number of VMs grows, which makes it not scalable and not applicable in large-scale environments. A possible solution for solving the scalability issue is to consider a hierarchy of VM clusters.

In this paper, we propose a scalable correlation-aware VM consolidation solution, *ScalCCon*, in which:

- We extended the Envelop [15], a method that transforms raw changeful resource usage time series to time series alternating between only two values, to be applicable for both time series with peaks and time series with valleys;
- We propose a novel two-phase clustering scheme to address the scalability issue, which first clusters VMs based on their utilization peaks, and then clusters VMs based on the correlation among them;
- We propose an VM placement algorithm based on the hierarchy of VM clusters obtained from the two-phase clustering.

We tested *ScalCCon* using data from the publicly available Google traces [17], and demonstrate that our proposed solution: 1) is scalable when the size of VM set increases; 2) requires fewer PMs than other state-of-the-art solutions; and 3) generates fewer resource capacity violations. We also show that the number of clusters in both of the two clustering phases in *ScalCCon* has negligible impact on the consolidation results.

The remainder of this paper is organized as follows. Section II presents some related work. The proposed solution, *ScalCCon*, is introduced in Section III. Section IV presents and discusses the evaluation, and Section V concludes the paper.

II. Related Work

Considerable efforts have been made in the VM placement problem in recent years, aiming at minimizing the amount of servers required or energy consumption [15], [18]–[20]. In order to guarantee the promised SLA, the interference among co-located VMs has also been studied, usually through analyzing low-level hardware usage metrics and workload characteristics [6]–[8], as well as consolidation approaches built on top of the interference models [3], [9], [10]. However, these approaches require pre-running each VMs to construct interference models as mentioned in Section I.

Alternatively, several studies plans the co-location of VMs by analyzing the correlation between each two VMs' resource utilization time series. Kuangyu et al. [11] made use of correlation analysis to jointly minimize the power consumption of both servers and data center networks. An algorithm that identifies VMs with complementary demand patterns by computing a matrix of correlation among VMs was also proposed [12]. This algorithm places VMs by pair without considering the correlation among the pair of VMs to be placed and the VMs that are already placed on a particular PM. Halder et al. [13] proposed a first fit decreasing (FFD) based correlation-aware VM placement algorithm that iteratively updates the correlation matrix for each placement decision, and places the VM having the minimal correlation with the placed VMs on a PM. A dynamic power management solution for scale-out applications in data centers was proposed by Kim et al. [14]. This solution employs both server consolidation and voltage and frequency (v/f) scaling. It defines a new metric to quantify the correlation between two VMs instead of using the Pearson correlation coefficient, and the matrix of correlation is computed using the new metric. One issue faced by such approaches that consolidate based on a correlation matrix is the scalability when the problem size becomes large.

The application of clustering in correlation-based VM consolidation improves the scalability to some extent. Verma et al. [15] proposed a Peak Clustering based Placement (PCP) algorithm, and a two-level Envelop scheme to transform VMs' utilization time series, based on the observation that correlation between peaks of VMs is more important than correlation across the complete time series. PCP clusters workloads that are strongly correlated in their transformed Envelops, and selects a set of workloads from each cluster proportionally. In contrast, pSciMapper [16] uses a distance measure, which focuses on the interference among workflow tasks' resource demand, to group scientific workflow tasks having dissimilar demand patterns together, and then map the clusters to a set of PMs. However, even though clustering methods are employed, the Pearson's correlation or other defined metrics between each two VMs still needs to be computed as the distance matrix in clustering methods such as hierarchical clustering.

The approach proposed in this paper solves the scalability issue by using a two-phase clustering scheme, which downsizes the scope of distance matrix computation, while also reducing the number of PMs and resource violation. *ScalCCon* considers the correlation not only among VMs to be placed, but also with placed VMs. Unlike the existing work, *ScalCCon* considers this by comparing the correlation between the aggregated resource usage of existing VMs and the representative utilization time series of a cluster, instead of individual VMs, thus further improving the solution's scalability.

III. Scalable Correlation-aware VM Consolidation

An essential policy to guarantee VMs' performance is to limit the instances when the server's resource utilization approaches 100%, which consequently cause performance degradation. In order to use the resources of a server as efficient as possible while minimizing the probability of violating the server's capacity limit, it is necessary to place VMs that are least likely to have simultaneous utilization peaks on the same PM, by analyzing the VMs' resources utilization time series. Given that the resource utilization is in the form of time series,

the possibility of two VMs having simultaneous utilization peaks can be measured by the *Pearson correlation coefficient*:

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}} \quad (1)$$

where x_i and y_i $(i = 1, 2, ..., n)$ are n measurements (i.e., average resource utilization for a sample period) of VM x and VM y respectively, over a period of time. However, it is not practical or scalable to analyze the correlation between each pair of VMs for the allocation of every single VM. Clustering methods have been studied recently [15], [16] to guide performance-aware VM placements. Even so, the correlation matrix or the distance matrix used in clustering methods, such as agglomerative hierarchical clustering, is computationally expensive, especially for large amount of VMs.

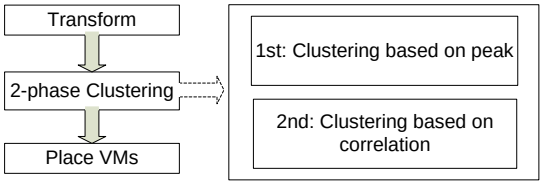
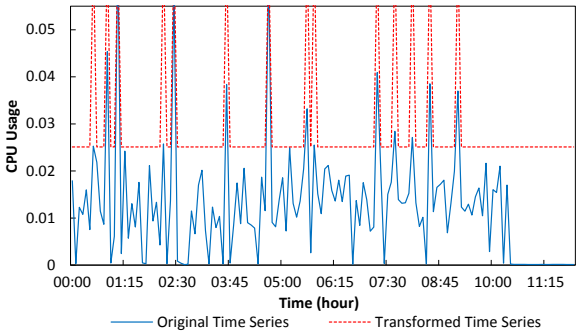


Figure 1. Overview of ScalCCon

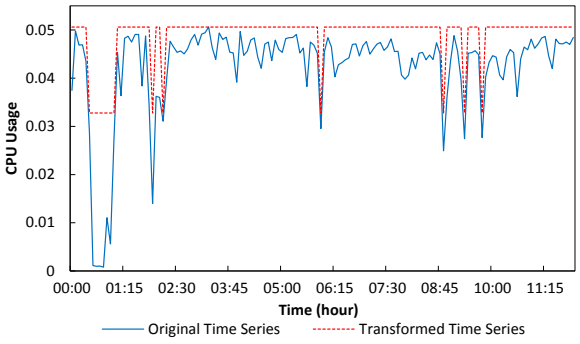
Our paper proposes *ScalCCon*, a scalable correlation-aware VM consolidation solution that uses a novel two-phase clustering scheme to consolidate large numbers of VMs while minimizing the servers' resource capacity violation. Figure 1 presents a high-level overview of the *ScalCCon*'s main components. Due to the frequent fluctuation in a VM's resource utilization over time, clustering based on the raw resource utilization time series generates a large amount of clusters and does not focus on reflecting the coinciding peaks. *ScalCCon* adopted and extended the two-level Envelop proposed in PCP [15] to first transform the raw time series into Envelops, which is presented in Section III-A. Following the transformation process is the two-phase clustering, which clusters VMs based first on peak and then on correlation (see Section III-B). Eventually, VMs are placed on the PMs based on the obtained cluster hierarchy using the algorithm presented in Section III-C.

A. Transformation

The purpose of transformation is to focus on the correlation between the peaks of two time series, instead of every single value. The prior two-level Envelop [15] is a time series that alternates between two values: a high percentile value (e.g., $90th$) C_B and the maximum value C_{max} of the original resource utilization time series. Any original utilization value below C_B is transformed to C_B and any value above is transformed to C_{max}. This Envelop is applicable to time series with peaks. We call it *Peak Envelop* in this paper. Figure 2a is an example showing the raw and transformed CPU utilization time series over 12 hours. The utilization time series in Figure 2 are extracted from the Google cluster trace. It is important to mention that in this type of environments it is also common to find utilization *valleys* instead of peaks. In such case, the existing Envelop will generate time series with very small peaks or even constant lines that neglect the valleys which can be complementary to other VMs' resource utilization peaks. Therefore, we extended the existing Envelop to be also applicable to time series with valleys, which we call the *Valley Envelop*. In this extended version, C_B is set to a low percentile value (e.g., $10th$), and the transformation rule remains. An example of time series with valleys and the transformed *Valley Envelop* is shown in Figure 2b.



(a) Peak Envelop



(b) Valley Envelop

Figure 2. Transform Time Series to Envelops

B. Two-phase clustering

A novel two-phase clustering scheme is proposed in this paper to address the scalability issue encountered in correlation-based VM consolidation solutions. As shown in Figure 3, VMs are first clustered based on their utilization peak (i.e., the difference between the maximum utilization and the high percentile value), generating a set of Peak Clusters. VMs in the same Peak Cluster have similar peak values. Following, in the second phase, within each Peak Cluster, VMs are clustered based on the correlation among their transformed resource utilization time series, producing a group of sub-clusters or CorClusters. VMs in the same CorCluster tend to have peaks that occur at the same or very similar time, thus, one of these time series is selected as the representative of this CorCluster (e.g., medoid: the object having the minimal average dissimilarity to all the objects in the cluster). In order to further improve the scalability, the time series representatives are used in the placement process to select CorClusters

having the minimal correlation with the aggregated utilization on the considered PM, instead of selecting individual VMs (see Section III-C).

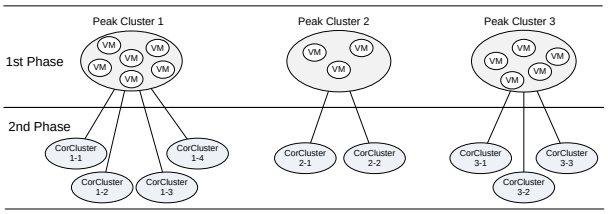


Figure 3. Two-phase Clustering Scheme

The reasons and advantages of using the proposed two-phase clustering scheme are as follows:

- Using a hierarchical structure improves the scalability of the solution, so does the policy of selecting CorClusters instead of individual VMs as mentioned above.
- In the case of VMs that have utilization traces containing peaks, the resource usage values are most of the time much lower than its maximum peak. If a peak buffer as big as the VM's maximum peak has been reserved, placing other VMs with similar peaks occurring at different times results in better utilization of the reserved buffer. The first peak-based clustering phase differentiates VMs with peaks in different ranges which the VM placement process can make use of.
- Both the peak size and the time when a peak occurs affect the correlation between two VMs. The peak-based clustering phase removes the impact of the dissimilarity between peak sizes by grouping VMs with similar peak sizes together, so that the correlation analyzed in the second phase is mainly determined by the time of the peak occurrence.

Three major categories of clustering methods for static data, namely, partitioning methods, hierarchical methods, and model based methods, have been utilized directly or modified for time series clustering [21]. Hierarchical clustering is one of the most widely used approaches, and is a good fit for the second phase of our proposed scheme, since the correlation among VMs can be used as the distance matrix required by hierarchical clustering. However, it lacks scalability due to its quadratic computational complexity [22]. K-means, belonging to the partitioning methods category, is the most commonly used clustering method. It is faster than hierarchical clustering, referring to [22], since it does not require calculating all the distances between each observation and every other observation. Therefore, we utilize k-means in the first phase to quickly cluster a large number of VMs, in order to reduce the sizes of data sets analyzed separately in the second phase.

Studies show that the contention that degrades application performance significantly lies in shared caches and memory bandwidth [9]. Furthermore, the burstiness in memory demand is much lower than the burstiness in CPU demand [23]. Therefore, *ScalCCon* focuses on the correlation among VMs' CPU utilization. *ScalCCon* takes VMs' utilization time series, such as monitored historical usage traces or predicted usage traces as input. Using accurately predicted usage time series as input can reduce the probability of having unexpected resource violation after the actual VM deployment. However, workload estimation and forecasting is orthogonal to this paper's scope.

C. VM Placement

We implemented two versions of *ScalCCon*:

- *ScalCCon*$_{cpu}$: considers only CPU in the VM placement. The idea is to focus on evaluating the solution's performance without the constraint of other resources.
- *ScalCCon*$_{multi}$: considers both, CPU and RAM, in the VM placement by integrating RBP, which is our prior VM placement algorithm using a fixed buffer to handle utilization peaks [18]. The idea is to demonstrate 1) the application of *ScalCCon* in multiple dimensional scenarios; 2) through the comparison with RBP that the size of the fixed reserved buffer can be reduced, so that resources can be utilized more efficiently by leveraging correlation analysis.

These two versions are only different in the way of sorting VMs and selecting a VM to be placed from a sub-cluster.

The VM placement process is executed based on the hierarchy of VM clusters. A Peak Cluster becomes the $selectedPCluster$ as long as at least one of its VM has been placed on the considered PM. Afterwards, a CorCluster from the $selectedPCluster$ is selected, from which a VM is selected to be placed. During this process, *ScalCCon* analyzes both the correlation among VMs to be placed and the correlation with existing workloads on a PM. To be precise, the correlation between the representative time series of each CorCluster inside the $selectedPCluster$ and the considered PM's aggregated utilization is analyzed. Following this analysis, the CorCluster with the weakest correlation is selected, from which a VM is then selected to be placed, using different polices in *ScalCCon*$_{cpu}$ and *ScalCCon*$_{multi}$. It is noteworthy that the computation overhead is significantly reduced, since our apporach only considers the correlation between CorClusters and a PM's existing workload, instead of the correlation between each pair of VMs.

The main logic of this process is presented in Algorithm 1. PMs and Peak Clusters are firstly sorted by available CPU capacity and by the maximum peak of each Peak Clusters respectively (line $1, 2$). For each PM in the sorted list, the algorithm firstly selects the first VM to be placed. The first VM that can fit in the currently considered PM is selected from the first Peak Cluster (i.e., the Peak Cluster with the largest peak) (line 6). VMs in the studied Peak Cluster are sorted by CPU in *ScalCCon*$_{cpu}$, or by cosine similarity as used in our previously proposed VM placement algorithm RBP [18] (i.e., by how similar the VM's shape is with that of the current

Algorithm 1 *ScalCCon* VM Placement Algorithm

1: $pmList \leftarrow sortPMs()$//by CPU in descending order
2: $peakClusterList \leftarrow sortPeakClusters()$ //by maximum peak of clusters in descending order
3: **for all** pm in $pmList$ **do**
4: //Pick the first VM to be placed
5: *sort VMs in the first Peak Cluster* $pCluster$ //by CPU or cosine similarity in descending order
6: *place the first* vm *that can fit in* pm *or the* vm *having the most similar shape with* pm
7: **if** $pCluster$ has only one sub-cluster **then**
8: $selectedPCluster \leftarrow pCluster + 1$ //move to next peak Cluster
9: $peakBuffer \leftarrow vm's$ peak
10: **else**
11: $selectedPCluster \leftarrow pCluster$
12: $peakBuffer \leftarrow$ maximum peak of $pCluster$
13: **end if**
14: //Fill pm with VMs that are least correlated
15: **while** pm is not full **and** $vmList.size() > 0$ **do**
16: *select the sub-cluster least correlated with* pm
17: *sort VMs in the selected sub-cluster*
18: *select the* vm *having the biggest CPU or the most similar shape with* pm
19: **if** $selectedPCluster$ has only one sub-cluster **then**
20: $selectedPCluster \leftarrow selectedPCluster + 1$ //move to next Peak Cluster
21: **end if**
22: **end while**
23: **if** $vmList == NULL$ **then**
24: break
25: **end if**
26: **end for**

PM's residual capacity) in *ScalCCon*$_{multi}$ (line 5). The size of a VM in terms of CPU is estimated as the $90th$ percentile of the VM's CPU usage over the time series. If the current Peak Cluster, $pCluster$, has only one sub-cluster, then the algorithm moves to the next Peak Cluster, since one VM from this sub-cluster has been placed and the other VMs tend to peak at the same or similar time. In addition, the placed VM's peak is set equal to the reserved buffer on the current PM. Otherwise, $pCluster$ is set equal to the $selectedPCluster$, and the maximum peak of the $pCluster$ is set as the reserved buffer on the current PM (line 7-13). The second part of the algorithm is to fill up the PM. Inside the $selectedPCluster$, the algorithm first selects the sub-cluster whose representative time series is least correlated with the aggregated utilization of existing VMs on the PM, and then sorts the VMs in the selected sub-cluster by CPU usage (*ScalCCon*$_{cpu}$) or cosine similarity (*ScalCCon*$_{multi}$) (line 16, 17). Finally, the algorithm places the first VM that can fit from the sorted list (line 18). Similarly, the algorithm moves to the next Peak Cluster if the $selectedPCluster$ has only one sub-cluster, otherwise it continues the placement until the PM is full (line 19-21).

IV. Experimental Evaluation

This section presents: 1) experiments comparing *ScalCCon* with existing solutions and other baselines to show that *ScalCCon* is effective in consolidating VMs (requires fewer PMs), while also minimizing capacity violations; 2) experiments comparing the execution time of *ScalCCon* and PCP when increasing the number of VMs, to test the scalability of *ScalCCon*; 3) experiments varying the number of clusters in each of *ScalCCon*'s two clustering phases, to investigate the impact the number of clusters has on the consolidation results.

A. Experimental Setup

1) Dataset: We use the utilization trace of a Google cluster [17], in particular the first 48 hours from which we extracted 5,482 "tasks" (i.e., VMs for our study) that lasted more than 12 hours. This allow us to have at least 144 utilization values for each VM to construct the time series. We consider two resources for our study, CPU and RAM, since it was only for these two resources that there was available information for all machines and tasks. The attributes that we considered for CPU and RAM are: the CPU rate, which indicates the average CPU utilization for a sample period of 5 minutes, and the canonical RAM usage, which represents the average RAM consumption for the same sampling period. The utilization values have been normalized between 0 and 1, according to the maximum capacity of a resource from the entire set of machines.

The entire trace is composed of 12,583 (heterogeneous) servers grouped in seven server configurations (presented as (CPU, RAM)): $(0.25, 0.2498)$, $(0.5, 0.9678)$, $(0.5, 0.749)$, $(0.5, 0.4995)$, $(0.5, 0.2493)$, $(0.5, 0.1241)$, $(1, 1)$. Since only a small set of PMs is needed in our experiments, we use 200 PMs with these seven configurations evenly distributed. Figure 2 demonstrates the CPU utilization of two VMs (tasks) from the dataset. In addition to this set up, that we call *heterogeneous scenario* in the following, we also define a *homogeneous scenario* which uses only the biggest type of PM's configuration (i.e., $(1, 1)$).

2) Baselines, metrics and parameters: ScalCCon is compared to the following solutions and baselines: 1) PCP$_{\text{H}}$: the approach uses a one phase clustering, and subsequently selects a set of VMs from each cluster proportionally, which are to be placed on a particular PM. As the clustering method employed in the original PCP [15] is not revealed in the paper where it was proposed, we implemented PCP$_{\text{H}}$ applying the same hierarchical clustering method as used in the second phase of *ScalCCon*. The goal is to perform a fair comparison, and to focus mostly on the advantages of a two-phase clustering over one phase clustering, in terms of scalability, instead of the comparison between different types of clustering methods; 2) RBP: a multi-dimensional VM consolidation algorithm, which balances a server's resource usage among different dimensions, and uses 25% of a server's total capacity as a buffer to handle utilization peaks and alleviate performance

degradation; 3) $\text{FFD}_{\text{Buffer}}$: a baseline which reserves a server's 25% capacity as buffer and applies FFD to place VMs; 4) FFD: a baseline that does not reserve any buffer and places VMs using FFD.

In order to evaluate *ScalCCon*'s effectiveness in consolidating VMs while minimizing the occurrence of resource capacity violation and *ScalCCon*'s scalability, the following metrics are considered in the experiments:

- Number of PMs required;
- Average buffer reserved on each PM (percentage of each PM's capacity).
- Average 90th percentile usage of each PM's aggregated CPU utilization after placement (in percentage).
- Resource capacity violation ratio computed as:

$$V_{ratio} = \frac{v_I}{N} \tag{2}$$

  where v_I is the number of instances when a PM's aggregated utilization exceed the PM's capacity, and N is the total number of observations of a time series.
- Execution time.

The reserved buffer is averaged, as *ScalCCon* reserves different sizes of buffer on each PM depending on the maximum peak of the $selectedPCluster$.

For the k-means clustering method employed in the first phase, we set its parameters as follows: the number of clusters $k = 10$ and $seed = 5$. The $seed$ is needed to guarantee that the same clustering result is obtained in different runs. As the number of VMs in each Peak Cluster (generated in the first phase) varies, we set the number of CorClusters (in each Peak Cluster) k' as the quotient of the total number of VMs in a Peak Cluster divided by a given number (e.g., 80), in order to control the granularity of CorClusters.

B. Effectiveness Evaluation

Tables I shows that *ScalCCon*$_{cpu}$ requires fewer PMs and leads to much less violation in both scenarios. FFD reveals that even though the 90th percentile resource usage is a good estimate of VM's resource demand, the chance of causing resource contention without a buffer is very high. The 25% buffer reduces the violation significantly as seen in $\text{FFD}_{\text{Buffer}}$, however, the violation ratio is still high in the heterogeneous scenario. The violation ratio is much lower in the homogeneous scenario because the PMs in this scenario have the biggest configuration, thereby the 25% buffer leaves larger space for utilization peaks than on other smaller PMs in the heterogeneous scenario. PCP_{H} performs better in the homogeneous scenario than in the heterogeneous scenario, reducing the number of PMs and buffer size, as well as the violation ratio compared to $\text{FFD}_{\text{Buffer}}$. However, as it selects a set of VMs from each cluster, there are co-located VMs that may be strongly correlated so that their peaks may coincide and exceed the capacity. *ScalCCon*$_{cpu}$ further reduces the violation ratio and buffer size in both scenarios compared to PCP_{H}. It is interesting to see from the results that a lower average buffer size results in a smaller number of PMs required.

TABLE I
COMPARISON OF $\text{SCALCCON}_{\text{CPU}}$, PCP_{H} AND BASELINES IN BOTH HETEROGENEOUS AND HOMOGENEOUS SCENARIOS CONSIDERING CPU

	Num of PMs	Average Buffer	Average 90th Percentile CPU Usage	Violation
Heterogeneous				
$\text{ScalCCon}_{\text{cpu}}$	115	13.69%	67.22%	6.25%
PCP_{H}	143	31.09%	54.10%	26.29%
$\text{FFD}_{\text{Buffer}}$	134	25%	60.76%	40.28%
FFD	93	0%	80.52%	110.42%
Homogeneous				
$\text{ScalCCon}_{\text{cpu}}$	72	11.32%	65.90%	6.25%
PCP_{H}	74	17.71%	63.51%	20.83%
$\text{FFD}_{\text{Buffer}}$	81	25%	60.98%	16.67%
FFD	61	0%	80.31%	73.61%

Table II presents the evaluation of the algorithms in the multi-dimensional scenarios. PCP_{H} is not evaluated here as it considers only CPU. Due to the constraint of RAM, every solution requires more PMs, while $\text{FFD}_{\text{Buffer}}$ can not place all the $5,482$ VMs on the 200 PMs. Interestingly, the number of PMs required by *ScalCCon*$_{multi}$ is more than that of RBP: the selection of VMs having similar shape with that of the PM's residual capacity is restricted inside a CorCluster, while RBP can select from all VMs. Consequently, RBP can find VMs that better balance the PMs' resource usage among different dimensions. *ScalCCon*$_{multi}$ sacrifices some flexibility in selecting VM candidates for a smaller reserved buffer, a lower violation ratio and more scalability in correlation analysis (compared to PCP).

TABLE II
COMPARISON OF $\text{SCALCCON}_{\text{MULTI}}$, RBP AND BASELINES IN BOTH HETEROGENEOUS AND HOMOGENEOUS SCENARIOS CONSIDERING BOTH CPU AND RAM

	Num of PMs	Average Buffer	Average 90th Percentile CPU Usage	Violation
Heterogeneous				
$\text{ScalCCon}_{\text{multi}}$	157	11.09%	50.82%	3.47%
RBP	135	25%	61.09%	25%
$\text{FFD}_{\text{Buffer}}$	200 (5258)	25%	49.93%	34.03%
FFD	163	0%	58.71%	169.44%
Homogeneous				
$\text{ScalCCon}_{\text{multi}}$	98	9.49%	49.12%	3.47%
RBP	81	25%	60.02%	25%
$\text{FFD}_{\text{Buffer}}$	113	25%	44.74%	18.06%
FFD	96	0%	52.22%	94.44%

C. Scalability Evaluation

We compare the scalability of *ScalCCon* and PCP_{H}, which uses one phase clustering, by measuring their execution time

when the size of the VM set is scaled up by duplicating the original data set. As shown in Figure 4, using the proposed two-phase clustering scheme makes *ScalCCon* more scalable. Its execution time grows nearly linearly, while PCP_H's execution time increments dramatically after the number of VMs reaches $14,964$.

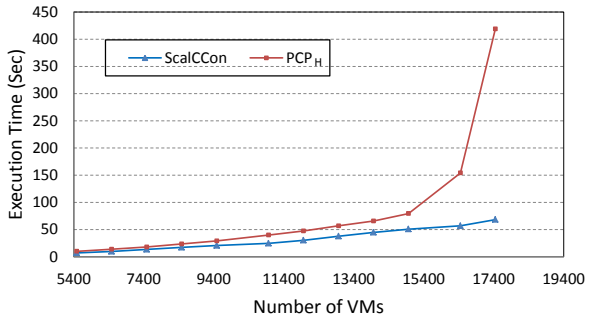


Figure 4. Scalability of ScalCCon and PCP_H

ScalCCon's execution can be further accelerated by adjusting the number of clusters (k) in the first phase as the number of VMs becomes large. As the most computational expensive part in *ScalCCon* is the correlation analysis and hierarchical clustering in the second phase, a bigger k narrows down the sizes of Peak Clusters which are analyzed disjointly in the second phase, and makes the correlation analysis inside each Peak Cluster faster. Figure 5 shows *ScalCCon*'s execution time versus scaling-up VM set with four different k values. As k increments, the execution time decreases, especially when k increases from 10 to 20. This impact is more significant when the number of VMs is large. As it is shown in the figure, when k increments from 30 to 40, the execution times for two k values are almost identical when the number of VMs is below $10,964$. Because the size of each individual Peak Cluster is not that big to cause distinct difference in computation time. However, the reduction in execution time becomes more and more notable when the number of VMs continues increasing.

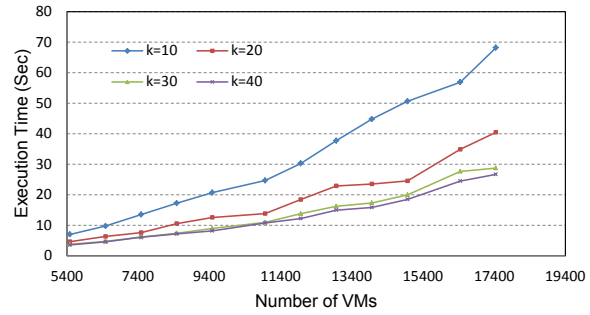


Figure 5. Scalability of ScalCCon with different k values

D. Discussion

The two clustering algorithms we use in *ScalCCon* force us to wonder what would be the best k (for the k-means) or k' value (for the cut of the hierarchical cluster tree). These questions are challenging, as finding the best cluster count depends not only on the distribution of the data points but also the cluster granularity the user wants, and consequently still remains an active research problem. A simple rule of thumb [24] for k-means is often mentioned:

$$k \approx \sqrt{n/2} \tag{3}$$

where n is the number of data points. But there exist also a number of clustering validity indices [2] [25] and a recent R package NbClust(), which provides 30 indices for determining the number of clusters, and can propose the number suggested by most of the indices [26]. However, many indices are computationally intensive, and the selection of indices is affected by a range of factors such as the structure and type of the dataset, and the clustering purpose. Since here clustering is employed only as a pre-process before consolidation, the general problem of finding the best number of clusters is not relevant for us and we turned it into the problem of *whether the selection of cluster count has a significant impact on the placement results*.

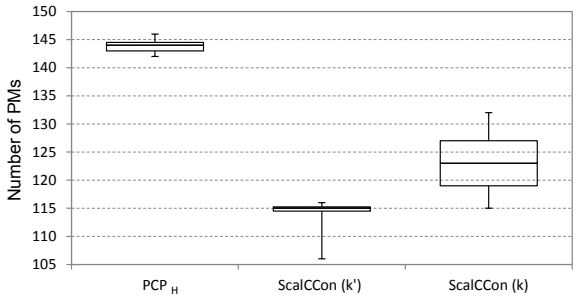


Figure 6. Number of PMs required in experiments given different number of clusters

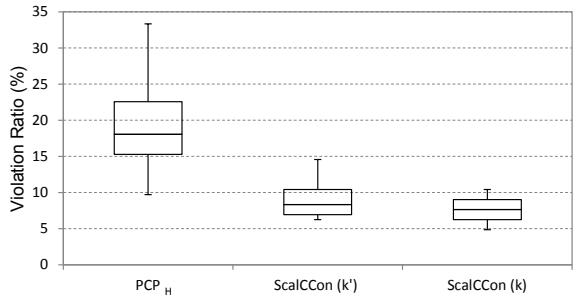


Figure 7. Violation ratio in experiments given different number of clusters

We performed the cluster count variation experiments for PCP_H and $\textit{ScalCCon}_{cpu}$. The numbers of clusters experimented for PCP_H are $\{5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 120, 150, 200\}$. The k for $\textit{ScalCCon}_{cpu}$ has a smaller range as there are two phases (i.e., $\{5, 10, 20, 30, 40, 50, 60\}$). For k' in the second phase, because the number of VMs in each Peak Cluster varies, instead of setting a fixed

number for k' directly, we set a number (d) to divide the number of VMs (M) in each Peak Cluster presented as $k' = \frac{M}{d}$. The list of divisors (d) experimented is as follows: $\{200, 150, 100, 90, 80, 70, 60, 50, 40, 30, 20, 10\}$. The distributions of the number of PMs required and the violation ratio in experiments given different number of clusters are presented as box plots in Figure 6 and Figure 7 respectively. We empirically show that the selection of the number of clusters in our case does not affect the consolidation results very much. *ScalCCon* is effective given different number of clusters in both phases. However, the total number of VMs to be placed is worth being considered when determining the number of cluster in *ScalCCon*. When the number of VMs is not very large, a smaller k can be chosen as the correlation analysis is not very heavy. A bigger k can speed up the execution when the number of VMs is huge.

V. Conclusion

This paper addresses the VM placement problem with the following objectives: 1) minimizing the number of PMs required; 2) alleviating the performance degradation (i.e., minimizing resource violations); 3) improving scalability (i.e., reducing execution time). We propose *ScalCCon*, a scalable VM consolidation solution that alleviates performance degradation via correlation analysis among co-located VMs on the same PM. In order to tackle the scalability issue, *ScalCCon* uses a novel two-phase clustering scheme, which first clusters VMs by their peak values, and then clusters VMs in each Peak Cluster obtained from the first phase, by their correlation among each other. Not only the correlation among VMs to be placed, but also the correlation with existing workloads on PM are considered during placement.

Our experiments show that the two-phase clustering scheme is effective in improving scalability, and that *ScalCCon* reduces the number of PMs required while also generating less resource violations compared to the existing approaches. This paper also discussed the impact that the number of clusters has on the placement results. The experiments show that *ScalCCon* is not vulnerable to the variation of the number of clusters in either or both phases. However, the size of the VM set to be placed is the main factor to be considered when determining the number of clusters, and the k in the first phase is tunable to accelerate the solution execution when the VM set is large.

Acknowledgment

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie) and by Enterprise Ireland Innovation Partnership in cooperation with IBM and University College Dublin under grant IP/2010/0061.

References

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proc. SOSP*, 2003, pp. 164–177.

[2] F. Xu, F. Liu, H. Jin, and A. Vasilakos, "Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions," *Proceedings of the IEEE*, vol. 102, no. 1, pp. 11–31, Jan 2014.

[3] D. Novakovic, N. Vasic, S. Novakovic, D. Kostic, and R. Bianchini, "Deepdive: Transparently identifying and managing performance interference in virtualized environments," Tech. Rep., 2013.

[4] S. Lee, R. Panigrahy, V. Prabhakaran, V. Ramasubramanian, K. Talwar, L. Uyeda, and U. Wieder, "Validating Heuristics for Virtual Machines Consolidation," Microsoft Research, Tech. Rep., 2011.

[5] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Black-box and gray-box strategies for virtual machine migration." in *Proc. NSDI*, vol. 7, 2007, pp. 17–17.

[6] S. Govindan, J. Liu, A. Kansal, and A. Sivasubramaniam, "Cuanta: Quantifying effects of shared on-chip resource interference for consolidated virtual machines," in *Proc. SOCC*, 2011, pp. 22:1–22:14.

[7] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu, "An analysis of performance interference effects in virtual environments," in *Proc. ISPASS*, 2007, pp. 200–209.

[8] Q. Zhu and T. Tung, "A performance interference model for managing consolidated workloads in qos-aware clouds," in *Proc. CLOUD*, 2012, pp. 170–179.

[9] A. Roytman, A. Kansal, S. Govindan, J. Liu, and S. Nath, "Pacman: Performance aware virtual machine consolidation." in *Proc. ICAC*, 2013, pp. 83–94.

[10] I. Paul, S. Yalamanchili, and L. John, "Performance impact of virtual machine placement in a datacenter," in *Proc. IPCCC*, 2012, pp. 424–431.

[11] K. Zheng, X. Wang, L. Li, and X. Wang, "Joint power optimization of data center network and servers with correlation analysis," in *Proc. INFOCOM*, 2014, pp. 2598–2606.

[12] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via vm multiplexing," in *Proc. ICAC*, 2010, pp. 11–20.

[13] K. Halder, U. Bellur, and P. Kulkarni, "Risk aware provisioning and resource aggregation based consolidation of virtual machines," in *Proc. CLOUD*, 2012, pp. 598–605.

[14] J. Kim, M. Ruggiero, D. Atienza, and M. Lederberger, "Correlation-aware virtual machine allocation for energy-efficient datacenters," in *Proc. DATE*, 2013, pp. 1345–1350.

[15] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, "Server workload analysis for power minimization using consolidation," in *Proc. USENIX ATC*, 2009, pp. 28–28.

[16] Q. Zhu, J. Zhu, and G. Agrawal, "Power-aware consolidation of scientific workflows in virtualized environments," in *Proc. SC*, 2010, pp. 1–12.

[17] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format + schema," Google Inc., Mountain View, CA, USA, Technical Report, Nov. 2011, revised 2012.03.20. Posted at URL http://code.google.com/p/googleclusterdata/wiki/TraceVersion2.

[18] X. Li, A. Ventresque, J. Murphy, and J. Thorburn, "A fair comparison of vm placement heuristics and a more effective solution," in *Proc. ISPDC*, 2014, pp. 35–42.

[19] N. Tziritas, C.-Z. Xu, T. Loukopoulos, S. Khan, and Z. Yu, "Application-aware workload consolidation to minimize both energy consumption and network load in cloud environments," in *Proc. ICPP*, 2013, pp. 449–457.

[20] A. Sansottera, D. Zoni, P. Cremonesi, and W. Fornaciari, "Consolidation of multi-tier workloads with performance and reliability constraints," in *Proc. HPCS*, 2012, pp. 74–83.

[21] T. W. Liao, "Clustering of time series data – a survey," *Pattern Recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[22] X. Wang, K. Smith, and R. Hyndman, "Characteristic-based clustering for time series data," *Data Mining and Knowledge Discovery*, vol. 13, no. 3, pp. 335–364, 2006.

[23] A. Verma, J. Bagrodia, and V. Jaiswal, "Virtual machine consolidation in the wild," in *Proc. Middleware*, 2014, pp. 313–324.

[24] K. V. Mardia, J. T. Kent, and J. M. Bibby, *Multivariate analysis*. Academic press, 1979.

[25] J. Bezdek and N. Pal, "Some new indexes of cluster validity," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 28, no. 3, pp. 301–315, Jun 1998.

[26] M. Charrad, N. Ghazzali, V. Boiteau, and A. Niknafs, "NbClust: An R package for determining the relevant number of clusters in a data set," *Journal of Statistical Software*, vol. 61, no. 6, pp. 1–36, 2014.