Workload forecasting and energy state estimation in cloud data centres: ML-centric approach

Tahseen Khan\textsuperscript{a,∗}, Wenhong Tian\textsuperscript{a}, Shashikant Ilager\textsuperscript{b}, Rajkumar Buyya\textsuperscript{b,\textsuperscript{a}}

\textsuperscript{a} School of Information and Software Engineering, University of Electronic Science and Technology of China, China
\textsuperscript{b} The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computer and Information Systems, The University of Melbourne, VIC 3010, Australia

\textbf{A R T I C L E I N F O}

Article history:
Received 14 August 2021
Accepted 16 October 2021
Available online 28 October 2021

Keywords:
Workload prediction
Energy state estimation
Resource management
Distributed data centre

\textbf{A B S T R A C T}

Resourcing management in data centres continues to be a critical problem due to increased infrastructure complexity and dynamic workload conditions. Workload and energy consumption prediction are crucial for efficient resource management decisions in cloud data centres. Existing solutions only consider forecasting the usage of virtual machine resources such as CPU and memory; they do not consider provisioned resources (CPU and memory) and disk, network transmission rates, which significantly affect the energy consumption of the host as well. VM-level energy consumption can be estimated for automated energy management decisions in modern data centres. However, it is not easy to measure energy for VM devices such as CPU, memory, and disk at the software level. In this way, we propose an ML-based model to predict load and energy to aid resource management decisions. For modelling workload predictions, we investigated several distinctive ML algorithms such as Linear Regression (LR), Ridge Regression (RR), ARD Regression (ARDR), ElasticNet (EN) and deep learning (DL) algorithm like Gated Recurrent Unit (GRU). The model’s predictions are measured using standard evaluation metrics like root mean square error (RMSE). We have discovered that GRU has performed very well by accomplishing the most negligible RMSE value for all the workload performances based on experimental results. For energy state estimation, we propose four diverse clustering algorithms, including, semi-supervised affinity propagation based on transfer learning (TSSAP), CLA based on transfer learning (TCLA), kmeans based on transfer learning (TKmeans), P-teda based on transfer learning (TP-teda) to discover similar groups of VMs dependent on features that may influence energy consumption as opposed to estimating it for each VM. The TSSAP has acquired promising clustering accuracy with 87.48% and 53.80% in identifying the VM classes which have been calculated using standard metric such as micro-precision for the chosen workload in comparison to affinity propagation (AP) and the average of other proposed clustering algorithms respectively.

© 2021 Elsevier B.V. All rights reserved.

1. Introduction

Cloud computing is an Internet-based computing paradigm that is capable of providing on-demand services to the end-users through virtualization of hardware resources in data centres [1]. Resource management is often a difficult task in a data centre due to multitenant users, changing workload conditions, and extremely complex infrastructures. The modern data centres comprise highly non-linear workloads. For instance, in an IBM survey, average CPU and memory usage of cloud workloads vary between 17.76% and 77.99% [2]. According to a study conducted by Google, the CPU and memory consumption of a cluster could not exceed 60% leaving a large resource inefficiency in Cloud data centres [3]. Consequently, the workload’s non-linearity usage patterns result in inconsistent performance, high energy consumption, and degraded quality of services (QoS). In addition, it increases operating costs and causes service providers to lose revenue. Since data centres are expensive to build and operate, it is necessary to optimize resource usage. An intelligent resource prediction approach can effectively resolve the issue to increase resource usage and reduce operating costs while ensuring the application’s Quality of Service (QoS).

A prediction mechanism produces insights into the future demand of a particular resource such as CPU, memory, disk and network based on rich historical workload. These predictions can be used to deal with non-linear resource utilization and energy consumption in the data centres and aid resource management decisions such as resource provisioning and VM consolidation etc. For instance, a resource provisioning mechanism based on these
future insights can deal with efficient resource allocation (i.e., allocating more and fewer resources to VMs based on their needs). Furthermore, decisions can be more proactive than current reactive approaches (e.g., provisioning required resources beforehand to improve QoS and avoid bottlenecks such as resource bootup time). In this regard, ML techniques can be used to make workload predictions [4]. ML-based predictions are ideal because they are derived from real features and capable of learning highly non-linear workload behaviour caused by multiple factors in data centre environments. Recent resource prediction works focuses on CPU, and memory usage and ignore provisioned (requested) resources such as CPU and memory [5,6]. When a new VM is instantiated on a host, these provided resources also make a significant contribution to energy consumption [7]. Furthermore, they ignore resource metrics like disk throughput, which has a direct impact on a host’s energy consumption [8]. Another essential metric to consider when consolidating virtual machines to save resources is network throughput [9]. Furthermore, many machine learning algorithms have been used to accomplish this task, but no single machine learning algorithm can address any non-linear workload well. As a consequence, using an ensemble learning method that involves several machine learning algorithms to predict both provisioned and used non-linear workloads with various metrics such as provisioned CPU, provisioned memory, CPU usage, memory usage, disk throughput, and network throughput would be advantageous.

Along with workload forecasting, energy estimation is crucial in data centre resource management. Energy consumption is a massive challenge in data centres, and data centre providers intend to minimize overall energy consumption through efficient resource management. The hosts in modern data centres have various sensors to monitor energy at the host level. Recent research [10,11] have focused on calculating energy consumption for each virtual machine (VM) using various power models. However, it is not easy to calculate the energy consumption of VMs at the software level. For instance, the energy consumption of memory is determined based on the events raised by each VM on each core’s last level cache (LLC). We need to collect these LLC metrics to determine energy consumption, which makes calculating the energy of each VM a difficult task [12]. Therefore, instead of calculating energy for each VM, we chose to look at patterns of similar VMs in different energy-consumption states. This is done by examining the available features related to energy consumption and using clustering analysis to identify VMs with similar patterns.

In this work, we use real work workload traces to build the prediction models. We mainly use Bitbrams data [13], which includes provisioned and used resource performance of several thousand VMs hosted in distributed Clouds. We propose prediction modelling for two tasks, i.e, workload prediction and energy state estimation of VMs, respectively. Our system model consists of two parts: Resource Management System (RMS) and the Prediction Module. We present an implementation of the Prediction Module in this paper. In this regard, we investigate various machine learning techniques for workload prediction, and the best models are chosen for further RMS actions. We use an ensemble learning approach to deal with energy state estimation and propose four different clustering methods to consider the best performing algorithm among semi-supervised affinity propagation based on transfer learning (TSSAP), CLA based on transfer learning (TCLA), kmeans based on transfer learning (TKmeans), and P-teda based on transfer learning (TP-teda). Based on our experiments, the TSSAP outperformed other methods by achieving the highest accuracy in clustering. Furthermore, we use the Univariate selection method with the ChiSquare ($\chi^2$) test to select the highly relevant features related to energy-consuming states in this method. Afterwards, we use t-Distributed stochastic neighbour embedding to cluster these features in the two-dimensional plane (t-SNE). Eventually, this clustered data is transferred to a different domain for further clustering analysis by using four clustering algorithms such as AP [14], CLA [15], kmeans [16] and P-teda [17].

In summary, the key major contributions of this work are as follows:

- We propose an intelligent prediction modelling based on machine learning for two tasks: workload prediction and energy state estimation.
- We explore different ML algorithms for workload prediction in nonlinear conditions using features comprising provisioned and utilized resources from a cloud hosting distributed data centre. The features include performance metrics, such as provisioned CPU, provisioned memory, CPU utilization, and memory utilization, disk throughput and network throughput.
- We present a novel approach to VM-level energy state estimation using an ensemble learning approach that includes four different proposed clustering methods for identifying similar groups of VMs based on VM-level features that may affect energy consumption.
- In our workload prediction models, GRU provides the least RMSE values for all features.
- In our energy state estimation models, TSSAP obtains a significant accuracy of 53.80% to identify VMs’ classes in comparison to other clustering models.

The rest of the paper is organized as follows: Section 2 discusses the relevant literature for this project. Section 3 explains the motivations for this work as well as the implications of resource management in the cloud. A resource management model is proposed in Section 4. The used cloud workload traces are described in Section 5. Section 8 is where performance and results are analysed. Finally, Section 9 concludes the paper and provides the future directions.

2. Related work

Machine learning-based prediction has been extended to a wide range of applications. Workload prediction and E-state prediction are two tasks performed by our model. The important work associated with both tasks is mentioned below. Tables 1 and 2 show a comparison of our research with related work.

First, we discuss related work for the proposed model’s first task. In the proposed system Resource Central, Cortez et al. [5] used the Random Forest method to predict CPU utilization in released traces of Microsoft Azure VM workload (RC). This system obtains VM features and learns these behaviours offline using machine learning before providing online prediction to various resource managers via a client-side library. Islam et al. [18] proposed an evolutionary approach to build an effective prediction model for CPU utilization for adaptive resource provisioning in the cloud, based on the machine learning algorithm Linear Regression (LR). It can help e-commerce applications with dynamic and proactive resource management scheduling and capacity planning. They used a dataset generated by the TPC-W benchmark. Barati and Sharifian [19] used a tuned support vector regression method in Google cloud workload traces to predict CPU and memory utilization for the purpose of proactively resource provisioning to keep resource utilization and service level agreements (SLAs) at an acceptable level. Farahnakian et al. [20] used Linear Regression to predict short-term future CPU utilization based on each host’s historical data. This process was used to determine whether a host was overloaded or underloaded based
A comparison on resource parameters and algorithms with related work.

<table>
<thead>
<tr>
<th>Study</th>
<th>Provisioned resources</th>
<th>Utilized resources</th>
<th>Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Provisioned CPU</td>
<td>Provisioned memory</td>
<td>CPU utilization</td>
</tr>
<tr>
<td>Cortez et al. [5]</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Islam et al. [16]</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Barati and Sharifian [19]</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Farahnakian et al. [20]</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Farahnakian et al. [21]</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Abdelsamea et al. [22]</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Farahnakian et al. [6]</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Proposed model</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 2

A comparison on power model and algorithms with related work.

<table>
<thead>
<tr>
<th>Study</th>
<th>Power Model</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kansal et al. [23]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Chen et al. [24]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Wen et al. [25]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Krishnan et al. [26]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Quesnel et al. [27]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Aldossary and Djemame [10]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Gu et al. [11]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Proposed model</td>
<td>×</td>
<td>✓</td>
</tr>
</tbody>
</table>

In the TANGO project, Aldossary and Djemame [10] presented an energy-based cost model that takes energy consumption as the main parameter in relation to the actual resource usage of VM. Gu et al. [11] presented a tree-regression-based method for estimating the power consumption of VMs on the same host. For each VM, Jiang et al. [28] presented a two-dimensional lookup table. CPU utilization, last level cache (LLC) miss rate, and the power value computed from CPU utilization and LLC miss rate are all included in the table.

3. Motivation: Intricacies in cloud data centre’s resource management

Resource management is a critical component in a distributed cloud data centre operations. The presence of multi-tenant users and their heterogeneous workloads makes estimating the workload load and energy consumption. In cloud data centres, the hosts have varying numbers of virtual machines over time. As a result, the host experiences variable workloads and energy consumption. It is crucial to analyse the non-linearity of VM workloads, examine host characteristics such as whether they are over-utilized and under-utilized, and take resource management decisions accordingly (e.g., resource provisioning and VM consolidation). The data-driven methods based on machine learning are being researched to save energy and optimize resource usage. The parameters like CPU, memory, disk and idle power all contribute to a host’s total energy [8]. To accurately estimate the energy consumption of the host, all the necessary contributing elements should be considered.

The CPU has a significant impact on the host’s energy consumption, particularly when running CPU-intensive applications. The authors in [23] ran a series of tests and discovered that in mixed workloads, the CPU consumes 58% of the total host’s energy. According to other studies [29] and [23], memory accounts between 20% to 30% of the total energy by a host due to memory accessing and page swapping at the host level. However, measuring the power of each VM at the VM level is difficult due to the need to collect LLC (last level cache) events raised by each VM on each core [12]. In the case of disks, however, energy is generated by spinning platters and disk head movement. [23] also presented a linear energy model based on disk read and write throughput in their research. Resource provisioning and VM consolidation are the two main ways to deal with energy efficiency. VM consolidation aims to improve resource utilization and energy efficiency by consolidating VMs to fewer hosts via VM migration while ensuring SLAs [30]. Intelligent predictive VM consolidation is being used these days, which is considered to be more efficient. Network throughput, on the other hand, is an important metric that can help with VM consolidation to save energy indirectly by reducing resources [9]. According to a study, by 2020, 51,774GB/sec of internet traffic will be generated as a result of computing as a service via cloud computing, which will
have an impact on cloud networks [31]. As a result, in the case of dynamic VM placement, this factor will affect VM migration time and violate SLA [32]. However, some researchers considered predicting CPU utilization only in the case of VM consolidation to save energy [1]. Therefore, the above facts indicate that elements such as memory and disk throughput and network throughput should also be considered for prediction in VM consolidation to save energy.

Furthermore, resource provisioning is the allocation of physical resources based on an estimation to improve resource utilization and energy efficiency. This estimation based on the prediction of future resource behaviour can better deal with efficient resource provisioning. This estimation, based on future resource behaviour predictions, can help with resource provisioning more efficiently. For a prediction-based estimate in resource provisioning, the majority of researchers focused solely on the use of physical resources like CPU, memory, storage and network bandwidth [33]. However, the current study does not take into account both provisioned and utilized resources when making predictions. In [7], the power models show that when a host instantiates a new VM, provisioned CPU and memory have a linear relationship with energy consumption. As a result, resource provisioning based on estimating the combined provisioned and utilized resources can provide a better perspective for IaaS service providers to save energy.

To understand the intricacies of power consumption of a host, we did a case study on workload traces. We obtained CPU usage (%) of two different VMs sampled at 5-minute intervals in a fastStorage trace obtained from Bitbrain’s dataset as shown in Fig. 1. If a host’s peak CPU utilization exceeds a fixed threshold (e.g., 80%) [34], it is considered over-utilized, and if it falls below a selective threshold (e.g., 30%) [1], it is considered under-utilized. We looked at 1250 VMs’ fastStorage data over a month. For example, in Figs. 1a and 1b, peak CPU utilization is between 80% and 100% and 3.5% to 4% for two different VMs, respectively. The CPU capacity for both of these VMs in the same trace is the same. As a result, it is clear that during a given month, CPU utilization for VM-1 reached up to 97.87%, while CPU utilization for VM-2 could not exceed 3.8%, indicating that a host is either over-utilized or under-utilized in this long run. According to [8], the CPU has a direct linear relationship with the total energy consumption of the host. It means that if a host’s VM’s CPU is over-utilized, it consumes a lot of energy, and if it is under-utilized, its processing power is wasted, yet spending a large amount of energy in terms of idle power.

In terms of energy efficiency, both cases such as workload forecasting and energy state estimation are critical for a data centre and must be addressed. As a result, observing the energy of each VM for the total energy of a host would be appropriate. Each component of a host, such as the CPU, memory, and disk, contributes to the total energy of the host [8]. Thus, monitoring energy of hosts can benefit from the visibility of energy consumption at the VM level, but measuring energy consumption of VM devices at the software level is extremely difficult. LLC (last-level-cache) events raised by each VM on each core must be collected at the VM level, making it more difficult to measure [12]. As a result, rather than measuring the energy of each VM, we decided to analyse the patterns of similar VMs that are suffering from over-utilization and under-utilization. Clustering analysis can be used to look for VMs that have similar patterns. The research is going towards automation. Thus, to learn these states by the machine automatically, we use a machine learning approach such as clustering, which automatically finds similarities between features and divides data into similar and dissimilar categories. Based on the factors discussed above, we consider the four different cases of peak CPU utilization as low, medium, high, and critical, respectively, 0%–40%, 40%–70%, 70%–95% and above 95%. The cases low and (high, critical) correspond to under-utilized and over-utilized i.e., low and high, critical energy-consuming states denoted by “E-state” (see Table 5). This type of analysis is carried out by observing which VMs are correctly divided using four proposed clustering methods. Our approach is not limited to these ranges; it can be seen by experimenting with different ranges based on the workload’s observations.

4. System model

A cloud platform is made up of several physical machines that provide end-users with on-demand services, and applications are deployed on these physical machines using virtualisation techniques. Fig. 2 depicts an overview of our system model. We chose a data-driven, machine-learning approach that uses historical application workload to learn from the past and predict the future workload level and energy states of VM. ML algorithms learn from historical data and help to make decisions in data-driven approaches.

Our work focuses on two types of tasks: (1) workload prediction, to which we investigate various machine learning methods and select a model with the lowest RMSE value. (2) E-state prediction, i.e., determining which virtual machines are in low and high energy-consuming states, we propose four different clustering methods for categorizing virtual machines based on features related to energy consumption. To deal with these two tasks, we use real workload traces that include various features.
including provisioned (CPU, memory) and used resources (CPU, memory, disk, and network throughput). The proposed model’s main component is the Prediction Module. The Resource Management System (RMS) can take decisions for different resource management tasks in cloud data centres; also, it makes energy management decisions with the help of the Cluster Management System from the Prediction Module. We only present the implementation of its Prediction Module in this paper. Our future work will include the performance of RMS for resource provisioning, VM consolidation, and other management functions based on the output of the Prediction Module. Therefore, the following subsections discuss the critical components of its Prediction Module.

5. Workload traces

The ML-based prediction system is as good as the data used to train and training data can include application and physical level features to train the model in the data centre domain [35]. Physical resource includes host-level resource usage such as CPU, Memory, IO, and so on, and application features include CPU cycles, cache metrics, and so on. We use two traces representatives collected from a distributed cloud hosting data centre and released by [13] that contains business-critical workload. A business critical workload is obtained from a service provider that specializes in managed hosting and business computation for enterprises. The details of this business-critical workload are shown in Table 3 and the definition of each feature is shown in Table 4. The vCloud Operation tools record seven performances per VM in these traces, which are sampled every five minutes.

These two traces collect data for 1750 virtual machines (VMs) across over 5000 cores and 20 TB of memory, accumulating over 5 million CPU hours in four months of operation, making them long-term and large-scale time series. The first trace, fastStorage, contains 1250 virtual machines (VMs) connected to storage area network (SAN) storage devices, and its performance was tracked for a month. The second trace, Rnd, contains 500 virtual machines (VMs) connected to much slower Network Attached Storage (NAS), and the performance of these traces has been monitored for three months. The dataset is smoothed by taking the average of each performance recorded for each VM [36]. We compute 1250 entries as the average of each feature for each VM in fastStorage trace for one month, and 500 entries for Rnd for three months. As a result, we have a total of 1500 entries for Rnd.

6. Workload estimation using prediction algorithms

We choose regression-based methods for our workload prediction because we want to estimate a numerical output variable like CPU utilization, which have also been used in earlier work on non-linear workloads such as (Linear Regression (LR), Ridge Regression (RR)) [33], ARD Regression (ARDR) [37], ElasticNet (EN) [38]. We also choose a deep learning method, recurrent neural networks (RNNs) with gated units, commonly known as gated recurrent units (GRUs) [39], as it outperforms traditional RNNs with other units [40]. For fastStorage and Rnd traces, we always consider the average of each VM resource over one month and three months of data and make predictions for these VMs based on this. The reason for selecting the average value has been discussed in Section 2. Furthermore, the peak CPU utilization in red rectangular boxes is rapidly decreasing after a short time interval, as shown in Fig. 3(a), so it would be feasible and efficient to use the average of each VM to train the ML model and forecast the average prediction value of each VM provisioned and used resources based on this learning. To implement all of the ML methods, we use the sci-kit learn [41] and the Keras [42] package to implement the deep learning method GRU. In this implementation, the parameters for each of the ML methods are set to their default values. The parameters for RR are set to $\alpha = 0.2$ and normalize = true. To predict the target variable, all ML-regression methods are trained with multiple features. For example, if the target variable is set to average CPU utilization, the remaining features are chosen from the traces to train the ML regression methods. Furthermore, we use the Root Mean Square Error (RMSE) metric to assess the goodness of fit of various methods, which is a standard evaluation metric in regression-based problems [43].

The RMSE can be defined as the dissimilarity forecasted value of a network and the actual value. It can be stated mathematically as follows:

$$\text{RMSE} = \sqrt{\frac{1}{T} \sum_{t=1}^{T} (e_t)^2}$$

**Table 3**

<table>
<thead>
<tr>
<th>Trace</th>
<th>VMs</th>
<th>Collection period</th>
<th>Memory</th>
<th>CPU cores</th>
<th>Collection interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>fastStorage</td>
<td>1250</td>
<td>30 days</td>
<td>17.729 GB</td>
<td>4057</td>
<td>5 Min</td>
</tr>
<tr>
<td>Rnd</td>
<td>500</td>
<td>90 days</td>
<td>5485 GB</td>
<td>1444</td>
<td>5 Min</td>
</tr>
</tbody>
</table>

**Table 4**

<table>
<thead>
<tr>
<th>Features</th>
<th>Definition (Average)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{CPU}$</td>
<td>Provisioned CPU capacity [MHZ]</td>
</tr>
<tr>
<td>$U_{CPU}$</td>
<td>CPU usage [MHZ]</td>
</tr>
<tr>
<td>$R_{memory}$</td>
<td>Provisioned memory capacity [kB]</td>
</tr>
<tr>
<td>$U_{memory}$</td>
<td>Memory usage [kB]</td>
</tr>
<tr>
<td>$D_{thrust}^r$</td>
<td>Disk read throughput [kB/S]</td>
</tr>
<tr>
<td>$D_{write}^r$</td>
<td>Disk write throughput [kB/S]</td>
</tr>
<tr>
<td>$N_{thrust}^r$</td>
<td>Network received throughput [kB/S]</td>
</tr>
<tr>
<td>$N_{trans}^r$</td>
<td>Network transmitted throughput [kB/S]</td>
</tr>
</tbody>
</table>
In Eq. (1), $P_i$ is actual value and $\hat{P}_i$ is the forecasted output variable by prediction network, and $T$ is the total number of predictions. As a result, the model will be more accurate if the RMSE values are lower. In addition, the model is examined to be more precise if its RMSE value is adjacent to 0.

Tables 6 and 7 show the performance of various ML-regression methods and deep learning method. These results represent the RMSE value for different features (see Table 4) of the selected traces. We can see from these tables that the deep learning method GRU has very low RMSE values, implying that residuals or prediction errors are lower and predictions are more accurate. Furthermore, different regression techniques have produced similar results. Because GRU results are more promising and have the lowest RMSE value. Therefore, we concentrate more on this algorithm to explore it further and explain it in Section 6.1 below.

6.1. Learning with Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) is introduced by [39] is the gated recurrent unit (GRU). In this mechanism, we train the model with 80% of the target variable, such as CPU utilization, and the trained model predicts with 20% of the target variable. The equations that define the GRU architecture are as follows:

\[
\begin{align*}
    u_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r), \\
    r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r), \\
    \tilde{h}_t &= \tanh(W_h x_t + U_h r_{t-1} r_t + b_h), \\
    h_t &= u_t h_{t-1} + (1 - u_t)\tilde{h}_t
\end{align*}
\]  

(2)

The vectors $u_t$ and $r_t$, for example in Eq. (2), correspond to the update and reset gates, respectively. The state of the vector at time $t$ is represented by $h_t$. The activation function of both gates are sigmoid function which is represented by $\sigma$. This function is in charge of limiting the range of values for $u_t$ and $r_t$ from 0 to 1. Furthermore, a hyperbolic tangent tangent function evaluates the candidate state $h_t$. The GRU network is fed with input $x_t$ (in our case, a vector of CPU usage $U_{CPU}$ values) and the feed-forward connections $W_r$, $W_h$, and $W_u$ as well as the recurrent weights $U_r$, $U_h$, and $U_u$. Before non-linearities in the network, the trainable bias vectors $b_r$, $b_h$, and $b_u$ are included.

In addition, the Pandas and Numpy [44] libraries are used to load workload traces as a pandas data frame and convert integer values to floating-point values that are more suitable for working with a neural network. The data is re-scaled from 0 to 1 using the MinMaxScaler. The dataset is then converted to a different shape using the original dataset and the look-back parameter assigned to 1, which denotes the number of previous time steps to be used as input variables to predict the next period [33]. Besides, this model consists of one input layer, one hidden layer, and one output layer with one input, five neurons, and one output forecast as optimized results are obtained with five neurons. In addition, the model can be trained with more neurons to achieve better-optimized performance. Eventually, the model is compiled using mean square error as a loss function and Adam optimizer [45] and the network is trained for epochs = 100 and batch_size = 64. We have optimized performance for these hyperparameters as discussed in Section 8.2. We also use the validation_data parameter in the training phase, which is the data on which the loss and any model metrics are evaluated for validation at the end of each epoch, but the network is not trained on this data. After the model has been fitted, the RMSE is used to evaluate the model’s performance on test data.

### Table 5

<table>
<thead>
<tr>
<th>E-states with CPU utilization.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak CPU utilization (%)</td>
</tr>
<tr>
<td>--------------------------</td>
</tr>
<tr>
<td>0–40</td>
</tr>
<tr>
<td>40–70</td>
</tr>
<tr>
<td>70–95</td>
</tr>
<tr>
<td>Above 95</td>
</tr>
</tbody>
</table>

### Table 6

<table>
<thead>
<tr>
<th>Features</th>
<th>GRU</th>
<th>LR</th>
<th>RR</th>
<th>ARDR</th>
<th>EN</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{CPU}$</td>
<td>3.46</td>
<td>417.66</td>
<td>1899.17</td>
<td>418.29</td>
<td>605.21</td>
</tr>
<tr>
<td>$U_{CPU}$</td>
<td>0.44</td>
<td>911.95</td>
<td>1002.97</td>
<td>1886.30</td>
<td>923.67</td>
</tr>
<tr>
<td>$R_{memory}$</td>
<td>9.29</td>
<td>9484.32</td>
<td>7930.79</td>
<td>929.67</td>
<td>605.21</td>
</tr>
<tr>
<td>$U_{memory}$</td>
<td>372.42</td>
<td>3679.61</td>
<td>384.48</td>
<td>368.17</td>
<td>366.03</td>
</tr>
<tr>
<td>$D_0$</td>
<td>0.37</td>
<td>3176.39</td>
<td>3192.39</td>
<td>3245.52</td>
<td>3183.62</td>
</tr>
<tr>
<td>$D_0^2$</td>
<td>0.06</td>
<td>325.19</td>
<td>323.77</td>
<td>338.36</td>
<td>324.26</td>
</tr>
<tr>
<td>$N_0^2$</td>
<td>0.33</td>
<td>53.17</td>
<td>54.18</td>
<td>68.22</td>
<td>53.37</td>
</tr>
<tr>
<td>$N_0^4$</td>
<td>0.23</td>
<td>93.83</td>
<td>93.22</td>
<td>101.90</td>
<td>94.21</td>
</tr>
</tbody>
</table>

### Table 7

<table>
<thead>
<tr>
<th>E-state with CPU utilization.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak CPU utilization (%)</td>
</tr>
<tr>
<td>--------------------------</td>
</tr>
<tr>
<td>0–40</td>
</tr>
<tr>
<td>40–70</td>
</tr>
<tr>
<td>70–95</td>
</tr>
<tr>
<td>Above 95</td>
</tr>
</tbody>
</table>

7. VM energy state estimation using clustering algorithms

We propose four clustering methods to form similar groups of VMs, we predict the energy state of a VM such as low, high and critical. These models would help various resource management decisions to increase the resource efficiency.

These techniques are based on the four clustering algorithms listed below:

- AP [14]: This is an exemplar-based algorithm which is used to propose TSSAP.
- CLA [15]: Every data point in this algorithm is given a mass and is linked to a special force called the local resultant force (LRF) generated by its neighbours.
- Kmeans [16]: This algorithm aims to group n data points into K classes, with each data point being a neighbour of the cluster centre closest to it.
- P-teda [17]: This algorithm is designed to handle high-frequency data. This method incorporates the TEDA theory concept and inherits all of its benefits.

In all of these methods, we use the [46] transfer learning approach to learn robust clusters for the target domain using knowledge from a source domain. Therefore, we provide the same source domain knowledge to all methods, as discussed in Section 7.1.1 for TSSAP. Apart from transfer learning, we restrict the affinity propagation (AP) algorithm to produce several clusters equal to the actual number of clusters and perform some additional additions such as semi-supervised learning using pairwise [47] and non-matrix factorization [48]. Thus, semi-supervised affinity propagation based on transfer learning (TSSAP), CLA based on transfer learning (TCLA), Kmeans based on transfer learning (TKmeans), and P-teda based on transfer learning are the names given to the proposed methods (TP-teda).
Because TSSAP has produced promising clustering results, we will primarily focus on this method in the following subsections to explain it in detail.

TSSAP is a semi-supervised clustering method that can use a small amount of supervised data in the form of partial labels to provide some supervision to unsupervised data in order to form more accurate similar clusters. Since the results are promising, this information about similar clusters is sent to the CMS. It separates the clusters of VMs into Low, High, and Critical energy-consuming states before submitting them to the Broker, which then sends these three clusters to the RMS for further analysis.

Fig. 3 depicts the proposed model. This method’s operation has been described in detail below.

7.1. Transfer learning-based semi-supervised AP

7.1.0.1. Transfer learning. Transfer learning is a type of learning that focuses on learning robust classifiers for a target domain using knowledge from a source domain [49]. We use the univariate feature selection method, in which the best features are chosen using univariate statistical tests like the chi-square test [50]. It is used in the context of feature selection on a labelled dataset to see if the class label is independent of a given feature.

**Definition 1.** If a feature has m different values and k classes, the chi-square score $\chi^2$ is calculated as follows:

$$\chi^2 = \sum_{i=1}^{m} \sum_{j=1}^{k} \frac{(b_{ij} - v_{ij})^2}{v_{ij}}$$

(3)

where $v_{ij}$ is the number of samples with the $i_{th}$ value and $b_{ij} = n_i n_j / n$

The number of samples that take the $i_{th}$ value of a feature is $b_i$ in this case. The number of samples in the $j_{th}$ class is $n_j$, and the number of samples in the input data is $n$.

We use t-Distributed stochastic neighbour embedding (t-SNE) [49] to reduce high-dimensional features to two-dimensional features via a matrix of pair-wise similarities after obtaining the best features that are most related to class labels. It effectively divides data into clusters, and we further cluster these divisions using modified AP method that improves clustering accuracy.

**Definition 2.** Given a set of $n$ high-dimensional data points $(y_1, y_2, \ldots, y_n)$, the conditional probability $P_{ji}$ that corresponds to similarities between two data points $y_i$ and $y_j$, for $i \neq j$,

$$P_{ji} = \frac{\exp(-||y_i - y_j||^2 / 2\nu_i^2)}{\sum_{k \neq i} \exp(-||y_i - y_k||^2 / 2\nu_i^2)}$$

(5)

The Gaussian variance for data point $y_i$ is $\nu_i$. This algorithm then uses the gradient descent algorithm to minimize the sum of Kullback–Leibler divergence. Furthermore, the variance of the t-Distribution is chosen as the parameters centred on the data point $y_i$ in high-dimensional space. Since the density of data changes, it is impossible to find a single optimal value for all data points. t-SNE produces a user-defined perplexity value with $\nu_i$ such that,

$$perp(\nu_i) = 2^h(\nu_i)$$

(6)

where $h(\nu_i)$ is the Shannon entropy of $\nu_i$ and defined as,

$$h(\nu_i) = -\sum_j L \log L$$

(7)

where $L = P_{ji}$.

7.1.0.2. Modified AP. In AP, the input parameters are similarities $S(i,j)$ between data points and preference $p$, which is the median or minimum of calculated similarities. As a result, two-dimensional features derived from the t-SNE operation were used to calculate input similarities using Euclidean distances like $S(i,j) = -||x_i - x_j||$ and preference, $p = min(S)$. The number of classes in data is not used as a parameter in AP, which results in a random number of exemplars. It could have an impact on AP’s clustering performance. Therefore, in AP’s input, we pass this supervised information, such as the number of classes K, along with $S(i,j)$ and $p$. The real-valued messages $a(i,k)$ and $r(i,k)$ are then computed by AP. We use non-matrix factorization (NMF) [48] to update similarities at this point.

**Definition 3.** Assume we have an $X$ matrix with m features and n samples. NMF decomposes matrix $X$ into two matrix $A(m \times q)$ and $B(q \times n)$ such that,

$$X \approx AB$$

(8)

In detail, it can be expressed as,

$$X = AB + e$$

(9)

The matrix norm of $X - AB$ is computed as $e$. $X$ is made up of similarities between data points calculated using the $S(i,j)$ Euclidean distance. As a result, the NMF method is applied to $S(i,j)$, which is decomposed into $A$ and $B$. These elements are upgraded repeatedly in order to reduce the estimation error $X \approx AB$. Various operations, such as Euclidean distance, can be used to calculate the distance between $AB$ and $X$.

$$d_{eu}(A, B) = \frac{1}{2}||X - AB||^2$$

(10)

and similarities $S(i,j)$ is updated with matrix $A$, such that

$$S(i,j) = A(i,j)$$

(11)

To provide more supervision to the updated similarities from NMF, we use semi-supervised learning. Semi-supervised learning
bridges the gap between unsupervised and supervised learning by incorporating both labelled and unlabelled components. When unlabelled data is combined with supervised data, the learning rate increases. Semi-supervised clustering has recently gained popularity, in which little supervision is provided by using various side information methods such as instance-level constraints, partial labels, and relative distance comparisons to increase precision in unlabelled data partitions. This method makes use of the instance-level constraints introduced by [47] to improve the accuracy of the results. These constraints indicate that two data points must link if they are in the same cluster, but cannot link if they are in different clusters.

**Definition 4.** Assume that a data set \( Y = \{y_1, y_2, \ldots, y_n\} \) exists, and that the cluster information is represented by a set \( \gamma \subset Y \times Y \), where \( \gamma = m_i \cup c_i \), and that for \( (i, j) \in (1, 2, \ldots, n) \),

\[
m_i = \{(y_i, y_j) \in Y \times Y : y_i \text{ and } y_j \text{ in same cluster}\}
\]

\[
c_i = \{(y_i, y_j) \in Y \times Y : y_i \text{ and } y_j \text{ in different clusters}\}
\]

The constraints for each pair of data points were determined using 30% of the actual labels, and the similarities obtained from NMF were updated again using these constraints. Similarities \( \chi^2 \) using 30% of the actual labels, and the similarities obtained from NMF were updated again using these constraints. Similarities \( \chi^2 \) for two data points \( (y_i, y_j) \) are updated with 1 or 0 if they are in the same cluster or not, respectively, for \( (i, j) \in (1, \ldots, n) \), such that,

\[
(y_i, y_j) \in m_i \Rightarrow \chi^2(i, j) = 1 \quad \text{and} \quad (y_i, y_j) \notin m_i \Rightarrow \chi^2(i, j) = 0
\]

Finally, using this similarity matrix as shown in Eq. (13), a fine set of potential exemplars is obtained. We also limited AP to producing a random number of exemplars by using a small amount of supervised data \( k \) that is passed into AP’s input. The accuracy of clustering improved as a result of this. The TSSAP pseudo code is shown in Algorithm 1:

**Algorithm 1:** Pseudo code of the proposed clustering approach TSSAP

```plaintext
Input: Features, labels, No. of clusters K
Output: E-state

1. \( R = \{\}, temp = \{\}, X = \{\}, f_1 = \{\}, f_2 = \{\}; \)
2. \( f_1 \leftarrow \chi^2(Feature, labels); \)
3. Select highest 4 in \( \chi^2 \) score features from \( f_1 \)
4. \( X \leftarrow tsne(f_2, euclidean); \)
5. \( S(i, j) \leftarrow Euclidean(x_i, x_j); \)
6. \( p \leftarrow \min(S); \)
7. Pass \( S(i, j) \) and \( p \) in AP’s input
   1. \( /\ast \) Execute AP, \( iter = 10 \) times
   2. Compute \( A(i, k) \) and \( R(k, i) \)
   3. \( (a, b) \leftarrow nmmf(S(i, j)); \)
   4. \( S(i, j) \leftarrow a(i, j); \)
   5. \( s \leftarrow \#(labels); \)
   6. \( \text{for } i = 1 \text{ to length(s)} \) do
     7. \( \text{for } j = i + 1 \text{ to length(s)} \) do
       8. if \( (x_i, x_j) \in c_i \) then
         9. \( S(i, j) \leftarrow 0; \)
       10. else
         11. \( S(i, j) \leftarrow 1; \)
     12. \( \text{/* where } C \text{ denotes cannot-link constraints } \)
   13. return idx
   14. E-state \( \leftarrow \) idx
```

7.1.1. Learning with semi-supervised affinity propagation based on transfer learning (TSSAP)

We chose one month of fastStorage trace data for the proposed methodology, which includes the performance of 1250 VMs running in a distributed data centre. As discussed in Section 2, the data from these VMs was analysed based on peak CPU utilization and labelled into different ranges. These ranges correspond to various energy-consuming states, as peak CPU utilization largely determines under- and over-utilization, i.e., low and high, critical energy consumption in the case of VMs assigned to hosts. As a result, the features of the 1250 VMs are used as the proposed method’s input. As a result, it would be practical and appropriate to analyse similar patterns based on these performances and observe the results with these ranges. The first step in the proposed method is to analyse the features using the univariate selection method. This method’s SelectKBest uses a \( \chi^2 \) test with \( k = 4 \) to select the best 4 features with a \( \chi^2 \) score. We use t-Distributed Stochastic Neighbour Embedding (t-SNE) to cluster the selected features such as \( R_{CPU}, U_{CPU}, R_{memory}, U_{memory} \) in a 2-dimensional plane after capturing the best features related to defined E-state. The data is then transferred to the modified AP model’s input, which more precisely clusters the data into different energy-consuming states.

**Predefined Class**

$$1250 \text{ VMs} \rightarrow \text{Cluster}(R_{CPU}, U_{CPU}, R_{memory}, U_{memory})$$

In detail, this information is used to compute a similarity matrix using pairwise euclidean distance with \( n \) number of data points, resulting in a \( n \times n \) similarity matrix \( S \). Following that, the preference parameter \( p \) is set to \( p = \min(S) / \text{iter} \times 0.3 \), with \( \text{iter} \) denoting the iteration number, which is \( \text{iter} = 10 \). To improve accuracy, the preference parameter \( p \) can be tweaked with different input values. In our case, \( p = \min(S) / \text{iter} \times 0.3 \) provides optimal performance. The parameters \( S \) and \( p \), as well as the number of classes \( K \) and labels \( labels \), are passed into AP’s input for further calculations in order to evaluate the final clusters. Furthermore, TSSAP provides predicted labels for VM partitions, i.e., each VM’s predicted energy consumption states out of 1250 VMs. These predicted labels are compared to actual labels using standard evaluation criteria such as micro-precision. This metric is chosen because it compares the actual labels to the predicted labels to assess the accuracy of clustering approaches [51]. As a result, it also applies to our situation, because we define different ranges based on peak CPU utilization.

If a data set has \( K \) classes and \( n \) data points, the micro-precision \( M_p \) is defined as in Eq. (15):

$$M_p = \frac{1}{n} \sum_{i=1}^{K} c_i$$

where \( n \) is the number of data points and \( c_i \) is the number of data points assigned to the corresponding class in cluster \( i \). The value of \( MP \) is in the range \( 0 \leq MP \leq 1 \), with 1 representing the best possible clustering result with actual class labels. As a result, if the \( MP \) value of the clustering model is closer to 1, it is thought to be more accurate.

8. Performance evaluation

In this section, we assess the performance of the proposed model in conjunction with the prediction module, as well as compare the results.
The tests are performed on a machine with an Intel(R) Core(TM) i3-4030U CPU running at 1.90 GHz and 4 GB of main memory. The proposed model’s Prediction Module accomplishes two tasks: (1) implements various prediction algorithms using PyCharm Community 2020.2, and (2) predicts E-state using the proposed clustering methods implemented using PyCharm Community 2020.2 and Matlab R 2019a.

We use the sci-kit learn [41] package to implement all ML-based regression techniques. Furthermore, we use the Keras [42] deep learning library to implement GRU. We use a real-world dataset from Bitbrain [13], which we discuss in Section 4. This dataset was chosen because it contains both provisioned and used resources that meet the requirements of our first targeted task, and most importantly, it contains real-world cloud infrastructure usage patterns. Since the model requires the most promising predictions among all the prediction algorithms implemented, all of the prediction algorithms are evaluated using RMSE to observe the least residual errors of the predictions with actual data. For the second task, we use PyCharm 2020.2 to extract knowledge from one-month data of the fastStorage trace using Pycharm’s chiSquare test and Matlab’s t-sne. The data is then fed into four different clustering algorithms in Matlab’s programming tool, including modified AP, CLA, Kmeans, and P-tedal. To identify the most promising results, all of the proposed clustering methods are evaluated using micro-precision. We primarily focus on TSSAP results in Section 8.2 because it has produced the most promising clustering results in comparison to other proposed methods in our case.

### Table 8

<table>
<thead>
<tr>
<th>Epochs</th>
<th>batch_size</th>
<th>Train score</th>
<th>Test score</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>32</td>
<td>2.09</td>
<td>0.81</td>
</tr>
<tr>
<td>40</td>
<td>32</td>
<td>1.77</td>
<td>0.39</td>
</tr>
<tr>
<td>100</td>
<td>32</td>
<td>1.76</td>
<td>0.46</td>
</tr>
<tr>
<td>10</td>
<td>64</td>
<td>2.66</td>
<td>1.26</td>
</tr>
<tr>
<td>40</td>
<td>64</td>
<td>1.79</td>
<td>0.52</td>
</tr>
<tr>
<td>100</td>
<td>64</td>
<td>1.76</td>
<td>0.44</td>
</tr>
</tbody>
</table>

#### 8.1. Experimental setup

#### 8.2. Analysis of results

The prediction Module is used in the proposed model to handle two tasks: workload prediction and E-state prediction. First, we will go over the workload prediction results. The prediction module is capable of experimenting with various machine learning methods in order to provide workload predictions for various workload types. We investigate different machine learning (ML) methods and a deep learning method, such as LR, RR, ARDR, EN, and GRU, for predictions on different types of workload, including provisioned \((R_{CPU}, R_{memory})\) and utilized \((U_{CPU}, U_{memory}, D_{th}, D_{th}, N_{th}, N_{th})\). The lower the RMSE, the more accurate the forecast. The parameters for each ML method and GRU are detailed in Sections Section 6, while the outcomes of predicted cases in terms of performance measure are shown in Tables 6 and 7 for fastStorage and Rnd traces, respectively. These tables show the RMSE values achieved using the various methods. Tables 6 and 7 show that none of the ML algorithms, LR, RR, ARDR, and EN, fit the dataset well and produce consistent predictions. By observing results, it is concluded that when workload feature has a small digit value, the RMSE value is very less. For example, \(R_{CPU}, U_{CPU}, D_{th}, D_{th}, N_{th}, N_{th}\) have smaller digit values than \(R_{memory}\) and \(U_{memory}\) and so have lower RMSE values. It may be deduced that none of these models are capable of making accurate and reliable predictions.

However, in comparison to other ML approaches that have very big RMSE values indicating poor performance, the deep learning method GRU acquires very few RMSE values for all features. GRU, in particular, is better than ML regression algorithms at modelling workload time series. One of the most important features of GRU is the presence of two vectors that determine what information should be sent to the output. They are unique in that they can be taught to retain knowledge from the past without washing it away over time or removing information that is unrelated to the forecast.

As a result, GRUs perform better because they can keep track of context-specific temporal dependencies between workload features for a longer time while making future predictions. The results also show that when the dataset is large, GRU provides superior accuracy. With more data, the model can extract more patterns and change the layer weights more precisely, but with traditional regression approaches, the smaller the data, the higher the accuracy. As the tables show, a large dataset reduces the accuracy of traditional regression methods.

Furthermore, by using a better infrastructure such as a GPU cluster, we can expect to obtain lower residual errors in prediction provided by GRU with a larger training dataset and more hyper parameter tuning. We are not interested in using hyper parameter tuning to get the best model possible; instead, we want to provide a generic model that can be applied to other models. However, we used different hyper parameter values, such as epochs and batch size, to train the GRU model. The iterations over which the input data is provided are referred to as the epochs. The batch size parameter specifies the number of samples to be updated per gradient update; it is set to 32 by default. Table 8 shows how the model is trained using various hyper parameters.

If the model is well trained on the data, it is thought to provide better performance. Table 8 clearly shows that the model trained at epochs = 100 has the least trained RMSE score in both batch sizes, 32 and 64. Per gradient update, the number of samples is specified as 32 or 64. The case where batch size is 64 will obviously train the model faster than the case where batch size is 32. For these reasons, we choose the epochs = 100 and batch size = 64 tuning case for all features in order to train the model with the best performance. We chose to represent GRU results visually because the results have been found to be promising. Figs. 4 and 5 show visual representations of GRU results for fastStorage and Rnd. For fastStorage and Rnd, the total samples are 1250 and 1500. The model is trained with 80% of the data and tested with 20%. As a result, 250 and 300 samples for testing results for both traces can be clearly seen. Both figures clearly show the actual (blue) and predicted (red) data. The training and validation loss graphs for each feature of both traces are shown in Figs. 6 and 7 during epochs = 100. We can see from the loss plots that the model performs similarly on both training and validation data. If these two loss plots begin to move consistently, the learning should be stopped. At epoch = 100, all subfigures in Figs. 6 and 7 have a consistent movement, indicating that the model has learned very well. The model can be trained more efficiently by fine-tuning hyperparameters.

Now we will talk about the results of the E-state estimation. We propose four different clustering algorithms to cluster similar types of VMs based on their energy-consuming state, i.e. E-state, and compare the forecasting results obtained by the proposed methods. We select one-month data from fastStorage traces, which includes 1250 VMs with various features such as \(R_{CPU}, R_{memory}, U_{CPU}, U_{memory}, D_{th}, D_{th}, N_{th}, N_{th}\). As discussed in Section 2, we also define different energy-consuming states. We use the univariate selection method on these features, along with the \(x^2\) test, to find the best four features to use on these range labels, and to ensure that they are independent of other features. During this test, the variables \(R_{CPU}, U_{CPU}, R_{memory}\) and

---

memory appear with the highest $\chi^2$ score of $3.234e^{+5}$, $3.644e^{+5}$, $1.374e^{+9}$, and $1.032e^{+8}$, respectively. We can see that provisioned resources like $R_{CPU}$ and $R_{memory}$ also have an impact on the energy consumption of a host. As a result, the proposed clustering algorithms use these selected features to find similar groups of VMs based on these features, which have provided good precision and accuracy. As shown in Fig. 8, the proposed clustering algorithms, TCLA, TKmeans, and TP-teda have achieved 12.48%, 50.88%, 51.20% and 66.48% accuracy on the selected dataset. Furthermore, among all of them, TSSAP has the greatest clustering accuracy of 66.48%. TSSAP is proposed by using the AP clustering technique, which has an accuracy of 8.32%. As a result, TSSAP outperforms AP by 87.48% and outperforms the average of other proposed approaches by 53.80% since it incorporates two types of learning, transfer learning and semi-supervised learning, into its functionality. The accuracy of transfer learning improves as we learn the optimal features that have the largest impact on energy consumption. Furthermore, we use little side information such as the number of classes to limit the AP technique to produce the actual number of VM clusters rather than a random number of VM clusters. The semi-supervised technique pairwise constraints has also contributed to an increase in accuracy. The 66.48% accuracy indicates that nearly 831 of the 1250 virtual machines are correctly identified in energy-consuming states. Although, because CPU and memory are the largest energy consumers in a host [7], we chose only the best four features by performing a $\chi^2$ test. However, disk throughputs also contribute to a host’s energy consumption; therefore, if more features are taken into account, the precision of finding similar VMs can be increased. Instead of using the $\chi^2$ test and the t-SNE, we can increase accuracy by using several state-of-the-art clustering works and various types of ML techniques to analyse the features. We are mostly interested in putting our new idea for dealing with energy consumption into practise, which is to find similar VMs based on features that affect energy consumption primarily at the VM level.

9. Conclusions and future work

In this work, we studied workload and energy state estimation in cloud data centres. Because of the high non-linearity of data
centre workload, predicting workload in advance has become
difficult and existing ML-based workload prediction solutions,
primarily consider the utilization metrics CPU and memory ignoring
other important parameters. When a new VM is instantiated
on a host, along with actual usage level, provisioned resources
like CPU and memory are also responsible for energy consumption.
In addition, host’s energy consumption is also influenced by
disk and network throughput. Energy consumption visibility is a
 crucial component of data centre energy management. The host
in modern data centres has several built-in sensors to monitor
energy consumption, but the virtualized platform does not.
Moreover, measuring the energy consumption of VM resources such as
CPU, memory, and disk at the software level is difficult. However,
the current study proposes energy models that use VM resource
performance of CPU, memory, and disk to measure energy at
the VM level. To calculate memory energy consumption, however, we
must collect the last level cache (LLC) events raised by each VM
on each core, which is extremely difficult to obtain, making the
measurement even more difficult.

In this regard, we proposed a machine learning-based model
with a Prediction Module that deal with the above two tasks.
We explored different ML algorithms such as LR, RR, ARDR, EN,
and a deep learning method GRU. Based on best performing
model, its predictions helps RMS making efficient decisions. In
the second task, instead of measuring the energy consumption
of each VM, we came up with a novel idea of grouping similar
VMs into different groups based on features that affect energy
consumption. We chose clustering analysis as the method of
choice for this task because it is a powerful tool for analysing
data similarities. To that end, we proposed TSSAP, TCLA, TKmeans,
and TP-teda as four different clustering algorithms for identifying
similar groups of different energy-consuming states (E-state). Our
model’s main benefits include the following: (1) It is evaluated
using real workload traces that include both provisioned and util-
ized resources, and all metrics performances such as provisioned
CPU, provisioned memory, CPU utilization, memory utilization,
disk throughput, and network throughput, (2) It is efficient and
adaptable because it can select the best results from a variety
of machine learning methods and (3) It makes use of semi-supervised and transfer learning techniques to help group similar VMs more accurately.

In future, we intend to implement the RMS component of our model for resource provisioning and VM consolidation based on the best performing results of different ML methods proposed in this work. We will also investigate more sophisticated models in order to improve workload prediction accuracy and performance across all metrics. Specifically, we will investigate different clustering and learning methods, such as kernel learning instead of pairwise constraints.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is partially supported by NSFC with ID (61672136 and 61828202) and Melbourne-Chindia Cloud Computing (MC3) Research Network.

References
