Int. J. Appl. Math. Comput. Sci., 2025, Vol. 35, No. 3, 535–545
DOI: 10.61822/amcs-2025-0038

CLUM: A CLUSTERING–CUM–MARKOV MODEL FOR RESOURCE PREDICTION IN A DATA CENTER

MADHUPRIYA GOVINDARAJAN [a,*], MERCY SHALINIE SELVARAJ [a], NAGARATHNA RAVI [a]

[a]Department of Computer Science and Engineering
Thiagarajar College of Engineering
Thirupparankundram, Madurai-625015, Tamil Nadu, India
e-mail: {gmadhupriya,shalinie@tce.edu},rathnaravi2013@gmail.com

High-end data centers are required to process the user requests and provide them with a better quality of service. The prominent issues in building a sustainable data center are reduced carbon footprint, dynamic capacity planning to reduce resource provisioning time and cost, minimized virtual machine migration to prevent higher downtime and enhanced return on investment and resource utilization. Realizing true elasticity will be a solution for these issues. Better elasticity can result if the data center is aware of the workload before its entry. Hence, the data center has to have a predictive model to forecast the resource requirements before the arrival of the workload. We propose a novel methodology called clustering-cum-Markov to predict the workload resource requirements proactively. It runs in the data center's controller and collects the statistics of the incoming workload. It characterizes the workload and predicts the necessary resources two-time slots ahead. We evaluate the modle in our data center and also with the benchmark Google Workload dataset. The results are compared with the state-of-the-art solutions based on various metrics, including the environment metrics. The proposed model achieves a 99.01% precision and exhibits optimal values with respect to the environmental metrics.

Keywords: data center, Markov model, randomized algorithm, resource prediction, workload characterization.

1. Introduction

As the applications and user demands are growing exponentially in the current digital era, the data centers are cropping up in many locations to maintain the promised quality of service (Liu *et al.*, 2020). Though the data centers aid in satisfying the service level agreements made with the users, there are several issues. They turn out to be a major contributor to the carbon footprint (Sun *et al.*, 2016). Efficient capacity planning is needed to curtail resource provisioning time, cost, and e-waste issues. In major cases, the utilization of data center resources is not at par with the planned capacity (Chen *et al.*, 2016). Over-provisioning of ready-state resources fails to achieve the expected return on investment and resource utilization index. Whereas, under-provisioning of ready-state resources leads to frequent virtual machine migrations, which may increase downtime. Hence, it is necessary to realize a true degree of elasticity. The rate of resources like memory, processor, disk, network bandwidth affects the energy consumption of the data center (Khan *et al.*, 2022). One naive approach is to get the workload, analyze the required resources to process it, and finally boot the required servers. But this will lead to a delay in processing the workloads with a stringent deadline. Hence, this approach will violate the service level agreement and lead to monetary loss.

Researchers and industrialists identified this problem of bringing about a sustainable and scalable data center while maintaining a thriving business several years back and chartered several plans. One such strategy is to characterize the workloads received in the data center in terms of their unique features like resources consumed, intensity, etc. Based on the workload characterization, the resources needed to satisfy the workload that will arrive in the subsequent time frames are forecasted. Hence, we can enable the necessary resources in the ready state for the future workload. This solution can effectively curtail the over-provisioning of resources and maintain the quality of service in an energy-efficient manner.

Some of the classical resource prediction models

*Corresponding author

include the grey forecasting model, chaotic theory, Fourier transform, exponentially weighted moving average model, and wavelet with Markov chains (Jheng *et al*., 2014; Qazi *et al*., 2014; Gong *et al*., 2010; Nguyen *et al*., 2013; Xiao *et al*., 2012). Later on, the network started to show higher levels of dynamism due to wide spread usage of technology. Hence, machine-learning-based prediction models are adopted to identify the patterns in the workload trace (Saxena *et al*., 2023). The state-of-the-art solutions use deep learning models, which include neural networks, autoencoders, Bayesian networks, LSTM, CNN and BiLSTM (Lu *et al.*, 2019; Chen *et al.*, 2019; 2022; Kim *et al.*, 2014). Though deep learning models improve prediction accuracy, they consume resources during prediction and frequent retraining of the entire model to address the dynamics in the workload characteristics. The limitation in the classical prediction models, the energy-efficiency issues in state-of-the-art solutions, and difficulty in modeling the workload dynamism and evolving a sustainable data center motivates us to investigate the problem of evolving an optimal workload characterization and resource prediction model.

We present a novel methodology called clustering-cum-Markov, abbreviated as CluM, to characterize the workload and predict the resources that will be required in the subsequent time frame. CluM is deployed in the data center's controller and monitoring system to facilitate the collection of the statistics needed for resource prediction. CluM has two major modules, namely, workload characterization and resource prediction. Initially, CluM analyzes the workload trace to infer its characteristics. Based on the observed characteristics of the workload, resource prediction is done. The key contributions of our work are as follows.

- While investigating the hardness of the problem, we find that it is an NP-hard problem. To account for the hardness, we propose a novel randomized machine learning model named repeated random sampling graph-based clustering (RRSG) for workload characterization. The model does not involve complex operations. Hence, it is lightweight and suitable for real-time operation. On mathematical analysis of the model's consensus, we find it has an optimal error probability.
- We propose a novel heuristic model named online adaptive multistep Markov (OAMM) for resource prediction. The OAMM model considers not only the present workload but also the historical trace. It also has a methodology to adapt the Markov state transition table to suit the dynamism. We do not predict the resource requirements for the immediate time frame but the second time frame. The resources need time to power on from sleep state. So, predicting two-time steps prior will be the suitable approach for the real-time scenario.
- Finally, we evaluate the CluM model in our fully functional data center to test its feasibility in a real-time scenario. We also test on the benchmark dataset and make a comparison with the state-of-the-art solutions.

2. Related work

As the workload prediction in the data center is beneficial in various ways, there exists a wide array of research work in this area. We give a terse description of the state-of-the-art solutions in this section.

Bi *et al*. (2019) propose to use Savitzky–Golay filter to eliminate the outliers and smooth the incoming workload. For decomposing the workload into multiple components for analysis, the authors use wavelet decomposition. The processed workload is given to the stochastic configuration network to predict the expected workload in the ensuing timeframe.

Lu *et al*. (2019) predict the future workload arrival pattern using the K-RVLBPNN model, which is a combination of K-means variant and back-propagation neural network variant. The value of 'K' is determined by the types of workloads. The neural network variant introduces dynamism in the learning rate based on the training error. The authors also classify the workload based on the latency-sensitivity parameter, which improves the accuracy. The L-PAW prediction model is a combination top-sparse auto-encoder and gated recurrent unit (Chen *et al*., 2019). The top-sparse auto-encoder reduces the dimensionality of the data, which is given as input to the gated recurrent unit for prediction.

The online Sparse BLSTM model is used by Gupta *et al*. (2020) to forecast the future CPU usage in the data center. The authors use Levenberg–Marquardt and gradient descent for online learning of the model. The authors also try to reduce the number of parameters using the sparse variant of the BLSTM model. Hieu *et al*. (2017) propose a multiple linear regression model to forecast the resources required k steps ahead in time. The model feedbacks the predicted value for predicting the next step.

Singh *et al*. (2021) use the evolutionary quantum neural network-based model to predict the workload. The model encodes the workload into qubits and propagates them across the network. The self-balanced adaptive differential evolution procedure optimizes the weights. Bi *et al*. (2021) propose a prediction model, the BG-LSTM model, which is a combination of Bi-LSTM and Grid-LSTM. The authors use a series of pre-processing steps to reduce the standard deviation, noise interference, and outliers using logarithmic

operation and Savitzky–Golay filter. The processed data is given as input to the trained BG-LSTM model.

The article by Santos *et al*. (2023) proposes a scheduling policy for Kubernetes based containerized application environment. The scheduling decision is based on the latency in network, bandwidth, dependency between microservices of the application as metrics. Thus their scheduler provides an improved application deployment and responsiveness for containerized applications.

Xu *et al*. (2024) propose an artificial intelligence (AI) based scheduler to optimize the performance of Kubernetes clusters. It uses the deep learning technique to monitor the performance of the workloads in the large scale cloud systems. Based on the monitored characteristics the reinforcement learning strategy adjust the task scheduler for efficient resource utilization and task execution.

Sanjalawe *et al*. (2025) presented a detailed summary of the traditional scheduling methods used in cloud environments. It also discusses the AI based scheduling methods that can be used in large scale cloud systems to manage the dynamic, heavy load tasks requesting for heterogenous resources. It also states the benefits like scalability, reliability, efficiency etc. that can be obtained by AI techniques.

Różycki *et al*. (2016) proposed a energy aware real time scheduling method to speed up the processing of preemptable jobs using an mathematical model. According to the model the continuous allocation of resources over time to the independent jobs with the availability of resources being dynamic i.e varied over time can provide an optimal scheduling policy for real time jobs.

Vasiliu *et al*. (2017) propose a hybrid scheduler for dynamic environments with the high data and task heterogeneity. The task deadline is considered as on one of the important quality metric in scheduling policy for resource utilization. Thereby big data intensive cloud applications can make use of this scheduler for multi task computing challenges with balanced workload execution.

Next, we point out some eminent features that our solution (i.e., CluM) provides on the state-of-the-art methodologies.

- The solutions proposed by Bi *et al*. (2019), Lu *et al*. (2019), Chen *et al*. (2019), or Singh *et al*. (2021) do not dynamically train the model to suit the changing workload patterns in the data center. A small change in the workload pattern may lead to erroneous predictions in the subsequent periods. This may affect the availability of service as per the service level agreement. The CluM model has a module for online training based on the observed changes.
- The methodology of Hieu *et al*. (2017) includes the predicted values in the model without verifying the correctness of the prediction. This may propagate unnecessary erroneous data and garble the efficacy of the model. The CluM model updates the model with the observed data.
- The model by Bi *et al*. (2021) takes high-dimensional data as input. Hence, processing the data in a real-time environment will consume time. The CluM model takes up low-dimensional data as input. So, it is feasible to implement the model in a real-time environment.

3. CluM

In this section, we demonstrate our proposed working methodology, called the CluM, to characterize the workload and predict the resources required for the upcoming workload. Before moving on to the explanation of the methodology, we first analyze the hardness of the problem.

Theorem 1. *The problem of characterizing the workload and predicting the resources is an NP-hard problem.*

Proof. We prove the NP-hardness by reducing our problem to the classical *berth allocation problem*. It (Chen *et al*., 2024; Guan *et al*., 2002) involves allocating berth space for serving the incoming vessels as soon as possible, subject to constraints like optimizing the service time. The problem that we have taken in hand also involves allocating the resources to the incoming workload, subject to the constraints. We can verify the solution to the problem in polynomial time if the resources can serve the workload efficiently. Hence, our problem is polynomial-time reducible to the conventional berth allocation problem. So, our problem comes under the category of NP-hard. ■

Hence, we model CluM as a randomized procedure to account for our NP-hard problem. CluM comprises of two phases, namely, workload characterization and resource prediction. We pre-define a *period*. It is imperative to set an optimal *period*. We propose computing the maximum time taken to power up the resources from the sleep state in the data center. The maximum time is set as the *period*. This will allow time to predict and power up the necessary resources to the ready state to serve the incoming workload, thereby maintaining the service level agreements. Once in a *period*, we aggregate the incoming requests into jobs. Based on the latency sensitivity of the requests, the task scheduler schedules the requests (task scheduling is not under the scope of the article, as we focus on the prediction of the resources). At the end of the period, we assign the scheduled tasks to the servers. Immediately

Algorithm 1. Workload characterization.

Require: Incoming request.

Step 1. Aggregate the incoming requests into job.

Step 2. Schedule the job.

Step 3. Run background process and measure CPU usage, memory usage, bandwidth usage and makespan.

Step 4. Input the parameters (P) to the RRSG procedure.

RRSG (P):

1. Most likely cluster list $= \emptyset$.
2. Compute the $number\ of\ samples = log\frac{2}{\alpha}/2t^2$.
3. If the number of clusters is C, then pick randomly $number\ of\ samples/C$ samples from each cluster.
4. $Euclid$ = Euclidean distances between the samples and the test sample
5. Most likely cluster list = *most likely cluster list* $\cup\ Cluster\_ID(Min\{Euclid\})$.
6. Repeat from step 2 for a pre-defined number of runs each time picking a set of random samples.
7. Normalize the distances in *most likely cluster list* to a range of 0 to 1.
8. Use most frequent (MoF) consensus to deduce the final output.

Output: Cluster membership of the job.

after assigning, the workload characterization module gets triggered.

3.1. Workload characterization. Algorithm 1 gives an overview of the execution of *workload characterization*. We further explain in detail in the ensuing paragraphs. Workload characterization learns a representation of the nature of the workload, which in turn aids in the process of capacity planning. The accuracy of the resource prediction greatly depends on the representation. There are several ways to generate a representation, which include business characterization, functional characterization, and resource characterization. As we intend to predict the optimal resource requirements, we take up the resource characterization technique. Resource characterization-based representation interprets the workload with respect to its consumption of the server resources.

A background process runs in the data center to measure the CPU usage, bandwidth consumption, memory usage, and makespan of the job. These four parameters are passed on to the workload characterization module.

As the workload is dynamic, we use a machine learning (ML) model to identify the intricate patterns in the workload. ML models are broadly classified as supervised, unsupervised, and semi-supervised. Though supervised and semi-supervised models can model a system in a better way, we choose unsupervised models. As the environment is dynamic and differs across the zones, a labeled dataset-based training may not be a universal approach. The unsupervised clustering-based model can model the network dynamism, identify the pattern in the historical trace, detect outliers, and analyze new characteristics. We prefer a graph-based clustering algorithm to other clustering methods. A graph-based clustering models the correlation between the parameters by plotting them across various axes and groups the similar data. Graph-based models do not require fixing an initial number of clusters. The incoming data don't need to fit in the existing clusters. It can form new clusters, which is a necessary trait for dynamic data center. We propose a novel variant of graph-based clustering algorithm named RRSG. The above-mentioned four parameters are passed to the RRSG model.

3.1.1. RRSG. In this section, we describe the speculation behind the proposal of the RRSG model. We also outline its working methodology in this section.

The native model of graph clustering involves plotting the normalized training samples on the graph space and forming clusters based on the intragroup and intergroup variance. The formation of the clusters is based on the computation of the distance between the sample and the centroid of the clusters. The centroid-based representation of the clusters has the following issues:

- The jobs from the users are diverse, and hence, a single centroid will not be sufficient to test the cluster membership of the incoming data.
- The greedy solution to the above issue is to compare the new sample with all members in the graph space. But this will consume computation and time cycles and is deemed to be unsuitable in real-time conditions. Even the computation of the centroid each time has to input all the members of the cluster.

We give a randomized solution for the issues stated above.

In the RRSG model, the cold start problem is addressed through an initial graph-based clustering (Zhuang *et al*., 2025) approach. At the beginning, when no clusters exist, a graph is constructed by randomly selecting a set of initial samples and plotting them in the graph space. These initial samples serve as the first

nodes in the graph. As new incoming samples arrive, the model computes distances between these samples and the initial graph nodes (representing initial clusters). If an incoming sample's distance is sufficiently small to one of the existing nodes, it is added to that node's cluster. If not, a new node (cluster) is created in the graph.

The benchmark dataset considered is the Google cluster traces 2019 dataset. It is a dataset that represents the complex relation between the various jobs and the machine resources required for them. Here the nodes of the graph represent the jobs and the relationship, edges between the nodes represent the resources needed and dependent for them. Here resources represent the CPU, memory and the total completion time of the jobs are taken as significant parameters which are captured by the RRSG model. The initial clusters are set by selecting random samples from this dataset and then for every sample data from the dataset the Euclidean distance measure is calculated to find its position in the graph space as either in existing cluster or a new cluster. This way the analysis of the benchmark dataset is proceeded with that similar to the steps applied for the TCE (Thiagarajar College of Engineering) data center workload analysis.

The RRSG model picks up a few random samples from each cluster. To quantify the number of random samples, we adopt the Hoeffding lemma. The Hoeffding lemma is as follows:

$$number\ of\ samples \geq \frac{log\frac{2}{\alpha}}{2t^2},$$

where $1-\alpha$ is the confidence interval, t is the desired variation between the actual value and the expected value. Here t is inversely proportional to $number\ of\ samples$. Hence, to achieve a small t, we need a large $number\ of\ samples$. But this will lead to increase in computation time and complexity. Hence, it is advisable not to choose a small value for t. Repetition of runs will guarantee the reduction of error probability which we prove in the further discussion through Theorem 2. If the number of clusters is C, then pick randomly $number\ of\ samples/C$ samples from each cluster.

The ensuing step is to compute the Euclidean distances between the samples and the test sample. The minimal distance and the respective cluster gets added to the *most likely cluster list*. The *most likely cluster list* contains the probable clusters of the test sample. The above procedure is repeated over several runs each time picking a set of random samples based on without replacement strategy. At the end of the runs, we have to choose one cluster for the sample from the *most likely cluster list*. Normalize the distances in *most likely cluster list* to a range of 0 to 1.

We use most frequent (MoF) consensus to deduce the final output. MoF outputs the most frequent cluster seen in the list as the output. There is also a possibility that the test sample may belong to a new cluster. We use the 50–50 rule to test the formation of a new cluster. If the normalized minimal distances of the most frequent cluster exceed 0.5, we create a new cluster and add the sample to it.

The time complexity of the RRSG model is $\mathcal{O}(\#(samples) \times \#(runs))$. Also, we substantiate the performance of the consensus by evaluating its error probability.

Theorem 2. *The error probability of the consensus is*

$$e^{-\frac{k(n-2r)^2}{2n^2}},$$

where k is the number of runs, r is the number of data points in the correct cluster and n is the total number of data points in all the clusters.

Proof. Let the total number of runs $= k$. Assume that the total number of data points across all the clusters $= n$. Assume that the number of data points in the correct cluster $= r$. The probability that the correct cluster is output $= \frac{r}{n}$. The probability that the incorrect cluster is output $= 1-\frac{r}{n}$.

Let I_j be an indicator random variable that states if the cluster output is incorrect in the j^{th} run, $j \in 1, dots, k$

$$I_j = \begin{cases} 1, & \text{if } incorrect\ cluster\ is\ the\ output, \\ 0, & \text{otherwise,} \end{cases}$$

$$I = \sum_{j=1}^{k} I_j.$$

As the the runs do not depend on each other, I is a binomial random variable and its parameters are $(k, 1-\frac{r}{n})$.

The output of one run is independent of the output of other runs, hence I_js are pairwise independent. By Chernoff bound, probability of returning an incorrect cluster equals

$$e^{-2k(1-\frac{r}{n}-\frac{1}{2})^2} = e^{-\frac{k(n-2r)^2}{2n^2}}.$$

■

A few examples that illustrate the error probability reduces with the increase of runs are as follows: $n = 10000$, $r = 600$, $k = 10$, then error probability $= 0.0208$; $n = 10000$, $r = 600$, $k = 20$, then error probability $= 0.0004$.

3.2. Resource prediction. We give an outline of the resource prediction phase in Algorithm 2. Further in the section, we first explain how we model our system and then give details of the OAMM model that predicts the resource.

Algorithm 2. Resource prediction.

Require: Cluster membership of historical job trace over n time frames and number of clusters C.

Step 1. Build the transition probability table to evolve the Markov model.

Step 2. Building the Markov model:

Step 2.1. Instantiate C^n states to include all the possible combinations of clusters in the historical job trace over n time frames.

Step 2.2. Compute the transition probabilities based on the observation during experimental period.

Step 3. Sparsification of the Markov model:

Step 3.1. Remove the states with nil transition probability.

Step 3.2. Use greedy methodology to curtail 80% of the states.

Step 3.3. Square the transition probability matrix for $(n + 2)$-th time frame.

Step 4. Dynamic update of the Markov model:

Step 4.1. If the misclassification of the cluster goes beyond 50% over a period, then recompute the transition probabilities.

Step 5. Search for the cluster sequence of historical job trace in the transition table.

Step 6. If (there is a match):

Output the cluster of the highest probability among the probabilities in the tuple.

Else:

Compute cosine similarity between the states and historical cluster sequence.

The state with highest similarity is the desired tuple.

Step 7. Output the cluster of the highest probability among the probabilities in the tuple.

Step 8. If (there is a single cluster with high probability):

Check the $resource\ usage$ table to find the required resources.

Step 9. If (multiple clusters have high probability):

Check the $resource\ usage$ table and fetch the statistics of the clusters.

Step 10. Output the required resources as $Max(statistics\ of\ the\ clusters)$.

Output:Resource prediction for $(n+2)$-th time frame.

In this section, we first explain how we model our system and the OAMM model that predicts the resource.

We model the system as a stochastic process as follows.

Let X_t be the random variable, where X denotes the resource usage at time 't'. The number of clusters $= C$, the count of previous time instances included in the prediction $= n$ (n is decided based on trial and error), the number of states $= C^n$. For instance, if $n = 3$ and $C = 4$, then the transition probability table having four clusters and considering three time frames, should have 64 possible states.

We compute the transition probabilities of our stochastic model from the observed changes in the characteristics of the workload over an experimental period. Even though the above methodology considers all the possible states, one issue is the count of states keeps increasing intractably with increasing clusters. To have the situation under control, we use a greedy methodology to minimize the number of states. Initially, we remove the states that have nil transition probability. Next, we compute the sum of the incoming transition probabilities of every state. The summed-up transition probabilities are sorted in decreasing order. We apply Pareto's 80–20 rule, i.e., 20% causes lead to 80% effect to curtail the states. We keep the top 20% of the sorted states and discard the rest of the states from the Markov model.

3.2.1. OAMM model. As we try to predict the resource requirements for the $(n + 2)$-th period, we square the transition probability matrix as per the Chapman–Kolmogorov equation (Miroshin, 2016).

The RRSG model outputs the cluster to which the current period workload belongs. OAMM model combines this output with the clusters of the previous n time frames. A search for the tuple of the combination in the transition table is done. The cluster of the highest probability among the probabilities in the output tuple is our desired value. There are high chances that the workload in the $(n + 2)$-th time frame will belong to this cluster. There is a possibility that no exact match for the historical cluster sequence will be available in the transition table because it may have been removed during the process of sparsification. In such cases, perform a cosine similarity between the historical cluster sequence with the states. The state with high similarity is the desired tuple.

The OAMM model maintains one more table structure, named *resource usage* whose dimension is $C \times 4$. There is one row for every cluster. The first column states the ID of the cluster. The subsequent columns have statistical values of the members of the cluster. The statistical values include standard deviation, mean, lower bound of the resource confidence interval, and upper bound of the confidence interval. The data center has to be ready with the resource capacity given in the upper bound column. For instance, if $cluster_1$ has 4 data points and their resource consumption in terms of CPU and memory are {(18.5, 56.6), (20.6, 56.9), (19.5, 58.1), (18.3, 60.3)}, then the upper bound resource consumption with 95% confidence interval is (20.1, 59.3). Hence, the data center has to get ready resources that can provide 20.1% CPU and 59.3% memory. If the tuple

from the transition table has multiple entries with equally high probability, then the upper bound of the resource confidence interval of all the equally probable clusters is fetched from the $resource\ usage$ table. The maximum values among them are chosen as the required resources. For instance, $cluster_1$ and $cluster_4$ have 0.4 and 0.4 transition probabilities respectively. Their upper bound resource consumption entries are (30.2, 63.3) and (60.2, 45.5). The maximum of CPU and memory usages is (60.2, 63.3). Hence, the data center has to get resources ready that can provide 60.2% CPU and 63.3% memory. Here we take the maximum rather than average to not give a chance of not meeting the service level agreement and losing business.

To handle the dynamism, the OAMM model has procedure to update the transition table. Initialize a value of time frames, F. Initialize a counter with the value 0, and keep track of the number of times the prediction of the cluster is incorrect over F. If the percentage of misclassification goes beyond 50%, then the transitions over the period F, are included in the computation of transition probabilities. The transition probabilities are recomputed. To fix a value for F, use the committed availability percentage in the service level agreement because misclassification may lead to downtime. Use the following formula to compute F:

$$F \times period \approx 1\% \, of \, acceptable \, downtime \, per \, week.$$

As is evident from the above formula, we set the value of F to ensure that the downtime is minimal per week.

Statistical analysis metrics of the workloads observed in the data center are used as data in the RRSG and OAMM model. The workload characterization module (RRSG) used to capture the dynamic characteristics of the workloads handled in the data center performs Predictive analysis using graph-based clustering ML method to predict the pattern of resources utilized in the historical traces of the workloads. It also detects outliers and analyse new pattern characteristics.

The resource usage table used in the OAMM model consists of the mean, standard deviation and confidence level of data points in the cluster obtained by the RRSG graph based clustering method.

Since this OAMM model predicts the resource requirements for the next future request, the statistical analysis techniques of descriptive, inferential and predictive analysis are performed over the workload data's collected from the data center. Mean and standard deviation of descriptive analysis and confidence interval of inferential analysis were computed over the data points in the cluster and used as resource requirement prediction data metrics.

4. Experimental setup and performance evaluation

We discuss the experiments done to assess the efficacy of the CluM model in this section. We execute the experiments in our data center. Besides this, to verify the working of the CluM model in the large-scale data center, we train the model using the benchmark Google Workload dataset.

The Thiagarajar College of Engineering (TCE) data center uses the high computing blade servers with Xeon processors mounted on the racks. There are five high computing blade servers that run the cloud services, namely, SaaS, PaaS, and IaaS. Every server has the system architecture with the components of Xeon processor, 300 GB hard disk and 98 GB RAM. The resource of the data center hosts the PaaS, SaaS and IaaS services by launching Virtual Machines (VMs) on top of the physical resources of the blade server using Xen hypervisor.

Xen has a Para Virtualization based architecture thereby having a knowledge of the resources allocated to the VM. PVM based virtualization architecture is used to create VMs to render SaaS, PaaS, and IaaS services. Xen hypervisor is a Type 1 hypervisor that has the capability to access and tune the system resources CPU, memory and bandwidth requirement of the client service launched in the VM.

The proposed RRSG model is deployed on the hypervisor to capture the patterns of the workload traces. The workload that executed in the TCE data center for the duration of one week and its resource consumption is captured by the RRSG model using the Xen hypervisor. The work load is observed in the data center for 24x7 every day, as it hosts all the services for the entire day. When the network bandwidth reaches nearly 1 GB of service request and response traffic, at that time the CPU and memory usage is almost 97% and 94% respectively. This is the peak load time and others are normal load. All these load values are captured as samples by the RRSG model. The number of samples collected are approximately 550 samples per hour of workload characteristics representing the VMs CPU, memory and total time taken by the process are captured for analysis.

The benchmark Google Cluster traces version 3 has the workload requests handled by the Google clusters in May 2019 (Wilkes, 2019). The workload represents jobs and their resource requests. Each job is a collection of tasks represented with Linux processes that can execute on a single machine or a collection. The resource represents the CPU and memory usage of the job. We process the data to evaluate the makespan, which is nothing but the representation of the time elapsed between the commencement and ending execution of the job.

Table 1. Accuracy metrics of workload characterization.

Model	Precision(%)	Recall(%)	F-measure (%)
K-means	91.57	85.92	88.67
CURE	95.31	93.22	94.25
Graph-based	92.67	89.56	91.08
Decision tree	88.41	95.11	91.64
K-NN	84.7	66.18	73.84
RRSG	98.71	97.52	98.11

Table 2. Time-based metrics of workload characterization.

Model	Training time (s)	Testing time (s)	Total time (s)
K-means	27.9	3.7	31.6
CURE	35.6	4.5	40.1
Graph-based	71.4	4.0	75.4
Decision tree	67.3	6.4	73.7
K-NN	59.4	5.9	65.3
RRSG	10.1	1.3	11.4

Table 3. CPU based accuracy metric of resource prediction models for data center.

Model	Precision(%)	Recall(%)	F-measure(%)
K-RVLBPNN	81.23	83.45	82.33
L-PAW	88.76	86.19	87.46
Sparse BLSTM	91.36	92.56	91.96
BG-LSTM	95.01	94.16	94.58
CluM	99.23	98.01	98.61

Table 4. Memory based accuracy metric of resource prediction models for data center.

Model	Precision(%)	Recall(%)	F-measure(%)
K-RVLBPNN	82.59	83.94	83.26
L-PAW	86.29	89.13	87.69
Sparse BLSTM	90.23	92.26	91.23
BG-LSTM	94.76	92.61	93.67
CluM	98.56	98.31	98.43

4.1. Accuracy: Workload characterization. We examine the RRSG model to verify the accuracy of characterizing the workload. Tables 1 and 2 highlights the values of various accuracy and time metrics of the RRSG model and other machine learning models that we observe in our data center.

As seen in accuracy and time-based metrics from Tables 1 and 2, the RRSG model exhibits better performance. The performance of the K-Means clustering algorithm depends on the pre-determined value of 'K', whereas the RRSG model has a module to test if the data belongs to a new cluster. Clustering Using REpresentatives (CURE) algorithm involves identifying the outliers using random samples. Even though CURE's performance with respect to accuracy is better when compared to other models, its time complexity is $\mathcal{O}(n^2 \log n)$, which leads to higher time than other models. The k-NN classifier also investigates the test data with the data points in its space. The `RRSG` model's time complexity is dependent on the count of runs and the samples taken in each run, which are less than the total number of data points in the clustering space. Hence, it shows better values in the total time metric. Graph-based clustering involves comparing the test data with the centroid or an average of the points in the clusters. The centroid is not sufficient to model the characteristics of all the data in the clusters and introduces a convex bias into the model. As the data center workload is dynamic and slight variations in their characteristics create a significant difference, the decision tree tends to have many branches, which leads to the overfitting issue.

4.2. Accuracy: Resource prediction. We investigate the accuracy of predicting the CPU and memory resource requirements using the Google Workload dataset and real-time workload in the data center (Tables 3–4). The CPU-Precision, CPU-Recall, CPU-F-measure, Memory-Precision, Memory-Recall, Memory-F-measure of the proposed model with the datacenter is obtained as 99.23%, 98.01%, 98.61%, 98.56%, 98.31% and 98.43%, respectively. These values show the proposed CluM model performs better than the conventional methods.

4.3. Response time. Response time is the average time taken to allocate the resources for the incoming workload. Figure 1 shows the response time that we observe in the data center over a period. We measure response time under three experimental conditions. We want to check if the sparsification of the Markov model has a notable effect or not. Hence, we include it as one of the experiments. As seen in Fig. 1, there is not much difference in response time in CluM and no CluM experimental conditions because CluM predicts the necessary resources. Small hikes in response time are due to incorrect predictions at those instances. CluM without sparsification increases the response time as it involves an exhaustive search across all the possible Markov states.

4.4. Availability. Availability is the ratio between the server uptime and total experimental time. Figure 2 shows the availability percentage that we observe in the data center over a period. As seen in Fig. 2, there is not much difference in availability index in both the experimental conditions because the CluM model predicts the necessary resources proactively with better accuracy.

4.5. Environmental metrics. We observe some environment-related metrics to verify if the CluM model is

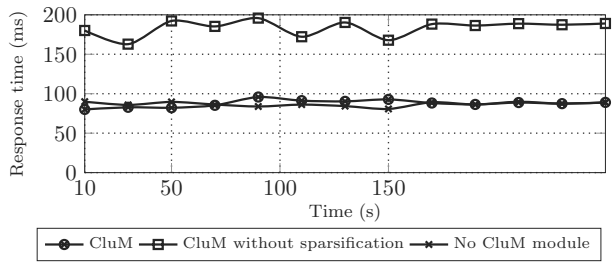

Fig. 1. Response time.

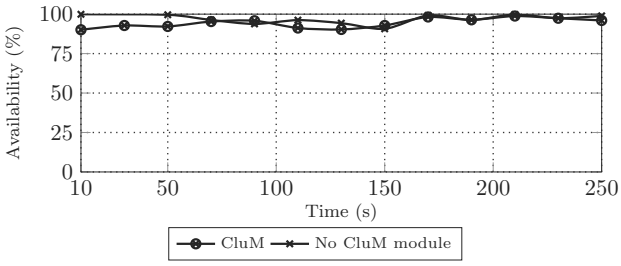

Fig. 2. Availability.

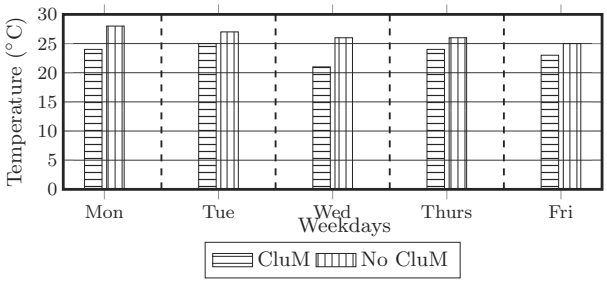

Fig. 3. Data center temperature.

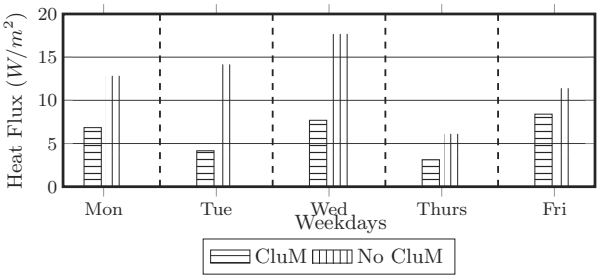

Fig. 4. Heat flux.

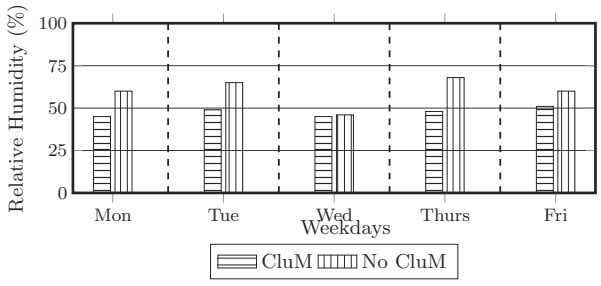

Fig. 5. Relative humidity.

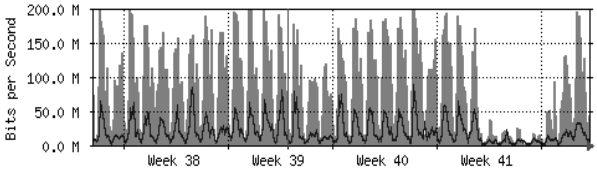

Fig. 6. Load during the experiment.

suitable to realize sustainable data center. Our data center has sensors to measure the temperature, relative humidity, and heat flux. It is necessary to maintain an optimal temperature and relative humidity throughout the day. The heat flux should be minimal in a sustainable data center. Figures 3, 4 and 5 show the average environment-metric values that we observe during two weeks. We operate the data center with the CluM model for a week and without the CluM model next week. Both the weeks were normal working days in our institution. To further ascertain if the data center receives a similar amount of workload over the two experimental weeks, we set a tracker to track the load. Figure 6 depicts the load handled by the data center over the two weeks (Week 38 and 39 in the figure). As seen in the figure, there is no significant difference in the load. As seen in Figs. 3, 4 and 5, the CluM model improves the data center environmental conditions.

The proposed CluM model uses RRSG based on graph based clustering, to determine the characteristics of the workload in a simple and efficient way. The Markov model of resource prediction for the future and near future is obtained from the past and present set of workloads. As the hidden Markov model extracts the time dependent temporal patterns of the workload, it is most suitable for the dynamic behavior based request of the data center. This is one of the significant benefit of the proposed CluM model when compared to its conventional methods. Furthermore this model performance can be evaluated against the deep learning models to see their performance in capturing the dynamic behaviour of workload prediction.

5. Conclusion

We propose the CluM methodology to predict the resources required to serve the workload in the future time frame. The proposed methodology contains two modules, namely, workload characterization and resource prediction. CluM fetches the statistics and gives them as input to the RRSG model. The trained RRSG model characterizes the workload to identify its nature. Based on the current workload characteristics and the characteristics of the historical trace, the OAMM model

predicts the nature of the workload that is to arrive two-time steps ahead. Based on the characteristic prediction, we approximate the resources requirements.

As future work, we anticipate working on improving the model with respect to the resource prediction accuracy. We also plan to probe into other statistical metrics and identify the metrics that will aid to enhance the prediction efficiency. Through CluM, we predict the CPU and memory requirements. We further plan to extend the model to predict if the workload can be served in an interleaved or parallel fashion without violating the latency requirement of the workload. This will further reduce the resource requirements and create significant improvements towards sustainable data centers.

References

Bi, J., Li, S., Yuan, H. and Zhou, M. (2021). Integrated deep learning method for workload and resource prediction in cloud systems, *Neurocomputing* **424**: 35–48.

Bi, J., Yuan, H., Zhou, M. and Liu, Q. (2019). Time-dependent cloud workload forecasting via multi-task learning, *IEEE Robotics and Automation Letters* **4**(3): 2401–2406.

Chen, C., Wang, F., Pan, J., Xu, L. and Gao, H. (2024). Algorithm design for an online Berth allocation problem, *Journal of Marine Science and Engineering* **12**(10): 1722.

Chen, L., Zhang, W. and Ye, H. (2022). Accurate workload prediction for edge data centers: Savitzky–Golay filter, CNN and BILSTM with attention mechanism, *Applied Intelligence* **52**(11): 13027–13042.

Chen, T., Zhang, Y., Wang, X. and Giannakis, G.B. (2016). Robust workload and energy management for sustainable data centers, *IEEE Journal on Selected Areas in Communications* **34**(3): 651–664.

Chen, Z., Hu, J., Min, G., Zomaya, A.Y. and El-Ghazawi, T. (2019). Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning, *IEEE Transactions on Parallel and Distributed Systems* **31**(4): 923–934.

Gong, Z., Gu, X. and Wilkes, J. (2010). PRESS: Predictive elastic resource scaling for cloud systems, *2010 International Conference on Network and Service Management, Niagara Falls, Canada*, pp. 9–16.

Guan, Y., Xiao, W.-Q., Cheung, R.K. and Li, C.-L. (2002). A multiprocessor task scheduling model for Berth allocation: Heuristic and worst-case analysis, *Operations Research Letters* **30**(5): 343–350.

Gupta, S., Dileep, A.D. and Gonsalves, T.A. (2020). Online sparse BLSTM models for resource usage prediction in cloud datacentres, *IEEE Transactions on Network and Service Management* **17**(4): 2335–2349.

Hieu, N. T., Di Francesco, M. and Ylä-Jääski, A. (2017). Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data centers, *IEEE Transactions on Services Computing* **13**(1): 186–199.

Jheng, J.-J., Tseng, F.-H., Chao, H.-C. and Chou, L.-D. (2014). A novel VM workload prediction using grey forecasting model in cloud data center, *International Conference on Information Networking (ICOIN2014), Phuket, Thailand*, pp. 40–45.

Khan, T., Tian, W., Ilager, S. and Buyya, R. (2022). Workload forecasting and energy state estimation in cloud data centres: ML-centric approach, *Future Generation Computer Systems* **128**: 320–332.

Kim, S.-R., Prasad, A.K., El-Askary, H., Lee, W.-K., Kwak, D.-A., Lee, S.-H. and Kafatos, M. (2014). Application of the Savitzky–Golay filter to land cover classification using temporal modis vegetation indices, *Photogrammetric Engineering & Remote Sensing* **80**(7): 675–685.

Liu, R., Sun, W. and Hu, W. (2020). Workload based geo-distributed data center planning in fast developing economies, *IEEE Access* **8**: 224269–224282.

Lu, Y., Liu, L., Panneerselvam, J., Zhai, X., Sun, X. and Antonopoulos, N. (2019). Latency-based analytic approach to forecast cloud workload trend for sustainable datacenters, *IEEE Transactions on Sustainable Computing* **5**(3): 308–318.

Miroshin, R. (2016). Special solutions of the Chapman–Kolmogorov equation for multidimensional-state Markov processes with continuous time, *Vestnik St. Petersburg University: Mathematics* **49**: 122–129.

Nguyen, H., Shen, Z., Gu, X., Subbiah, S. and Wilkes, J. (2013). AGILE: Elastic distributed resource scaling for infrastructure-as-a-service, *10th International Conference on Autonomic Computing (ICAC'13), San Jose, USA*, pp. 69–82.

Qazi, K., Li, Y. and Sohn, A. (2014). Workload prediction of virtual machines for harnessing data center resources, *2014 IEEE 7th International Conference on Cloud Computing, Anchorage, USA*, pp. 522–529.

Różycki, R., Waligóra, G. and Węglarz, J. (2016). Scheduling preemptable jobs on identical processors under varying availability of an additional continuous resource, *International Journal of Applied Mathematics and Computer Science* **26**(3): 693–706, DOI: 10.1515/amcs-2016-0048.

Sanjalawe, Y., Al-E'mari, S., Fraihat, S. and Makhadmeh, S. (2025). AI-driven job scheduling in cloud computing: a comprehensive review, *Artificial Intelligence Review* **58**(7): 197.

Santos, J., Wang, C., Wauters, T. and De Turck, F. (2023). DIKTYO: Network-aware scheduling in container-based clouds, *IEEE Transactions on Network and Service Management* **20**(4): 4461–4477.

Saxena, D., Kumar, J., Singh, A.K. and Schmid, S. (2023). Performance analysis of machine learning centered workload prediction models for cloud, *IEEE Transactions on Parallel and Distributed Systems* **34**(4): 1313–1330.

Singh, A. K., Saxena, D., Kumar, J. and Gupta, V. (2021). A quantum approach towards the adaptive prediction of cloud workloads, *IEEE Transactions on Parallel and Distributed Systems* **32**(12): 2893–2905.

Sun, X., Ansari, N. and Wang, R. (2016). Optimizing resource utilization of a data center, *IEEE Communications Surveys & Tutorials* **18**(4): 2822–2846.

Vasiliu, L., Pop, F., Negru, C., Mocanu, M., Cristea, V. and Kolodziej, J. (2017). A hybrid scheduler for many task computing in big data systems, *International Journal of Applied Mathematics and Computer Science* **27**(2): 385–399, DOI: 10.1515/amcs-2017-0027.

Wilkes, J. (2019). Clusterdata 2019 traces, `https://github.com/google/cluster-data/blob/master/ClusterData2019.md`.

Xiao, Z., Song, W. and Chen, Q. (2012). Dynamic resource allocation using virtual machines for cloud computing environment, *IEEE Transactions on Parallel and Distributed Systems* **24**(6): 1107–1117.

Xu, Z., Gong, Y., Zhou, Y., Bao, Q. and Qian, W. (2024). Enhancing kubernetes automated scheduling with deep learning and reinforcement techniques for large-scale cloud computing optimization, *9th International Symposium on Advances in Electrical, Electronics, and Computer Engineering (ISAEECE 2024), Changchun, China*, pp. 1595–1600.

Zhuang, X., Wang, Y., Hao, S. and Wang, X. (2025). Enhancing graph topology and clustering quality: A modularity-guided approach, *International Conference on Pattern Recognition, Kolkata, India*, pp. 131–142.

Madhupriya Govindarajan holds a PhD degree from Anna University, Chennai. She is an associate professor in the Department of CSE, Thiagarajar College of Engineering, Madurai. She has published around 20 papers. Her research interests includes virtualization, scheduling, migration, and automation using AI and security.

Mercy Shalinie Selvaraj (IEEE Senior Member) holds a PhD degree from Madurai Kamaraj University, Madurai. She is a professor and the head of the CSE Department, Thiagarajar College of Engineering, Madurai. She had been a postdoctoral fellow with the University of California at Irvine, USA. She has published over 200 research papers. Her areas of research interests include machine learning and security systems.

Nagarathna Ravi (IEEE Member) holds a BTech degree in information technology from the Madras Institute of Technology, Anna University, Chennai, India, and an ME degree in computer science engineering from the Thiagarajar College of Engineering, Madurai, India. She has completed her doctorate at Anna University, Chennai. She has published around 15 papers. Her research interests are software-defined networks and Internet of Things security.

Received: 19 July 2024
Revised: 28 February 2025
Re-revised: 5 & 9 June 2025
Accepted: 10 June 2025