IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 25, 2020, accepted December 2, 2020, date of publication December 4, 2020,
date of current version December 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3042716

Dynamic K-Means Clustering of Workload and Cloud Resource Configuration for Cloud Elastic Model

TARIQ DARADKEH[1], (Member, IEEE), ANJALI AGARWAL[1], (Senior Member, IEEE), MARZIA ZAMAN[2], AND NISHITH GOEL[2]

[1]Department of Electrical and Computer Engineering, Concordia University, Montreal, QC H3G 1M8, Canada
[2]Cistech Limited, Ottawa, ON K2E 7K3, Canada

Corresponding author: Tariq Daradkeh (t_darad@encs.concordia.ca)

This work was supported by the Natural Sciences and Engineering Research Council and Mitacs Accelerate Program in Collaboration with Cistech.

ABSTRACT Cloud elasticity involves timely provisioning and de-provisioning of computing resources and adjusting resources size to meet the dynamic workload demand. This requires fast, and accurate resource scaling methods at minimum cost (e.g. pay as you go) that match with workload demands. Two dynamic changing parameters must be defined in an elastic model, the workload resource demand classes, and the data center resource reconfiguration classes. These parameters are not labeled for cloud management system while data center logs are being captured. Building an advance elastic model is a critical task, which defines multiple classes under these two categories i.e. for workload and for provisioning. A dynamic method is therefore required to define (during configuration time window) the workload classes and resource provisioning classes. Unsupervised learning model such as K-Means has many challenges such as time complexity, selection of optimum number of clusters (representing the classes), and determining centroid values of the clusters. All clustering methods depend on minimizing mean square error between center of population in same class member. These methods are often enhanced using guidelines to find out the centroids, but they suffer from K-Means limitations. For the application of clustering cloud log traces, most of the reported work use K-Means clustering to label workload types. However, there is no work reported that label data center scaling classes. In this work, a novel method is proposed to analyze the characteristics of both workloads and datacenter configurations using clustering method, and is based on random variable model transformation (kernel density estimator) guide. This method enhances K-Means clustering by automatically determining optimum number of classes and finding the mean centroids for the clusters. In addition, it improves the accuracy and the time complexity of standard K-Means clustering model, by best correlating between clustering attributes using statistical correlation methods.

INDEX TERMS Elastic model, kernel density estimator, dynamic k-means clustering, workload, data center configuration, logs analysis.

I. INTRODUCTION

Cloud Elastic Model (i.e. rapid elasticity) is one of the basic cloud service characteristics as per NIST definition [1]. The configurable resources must be tuned dynamically with minimum management effort to meet client workload demands "such that at each point in time the available resources match the current demand as closely as possible" [2]. Cloud resource scaling has three main methods to rescale resources: i) horizontal scaling that increases or decreases number of virtual instances like application containers or virtual machines (VMs), ii) vertical scaling that increases or decreases the virtual instances like memory size, processor numbers or performance, storage, and iii) migration, by moving VMs or applications from one physical host server to another. All these parameters must consider the size of resource units that can match the workload demands need. On the other hand, workload

The associate editor coordinating the review of this manuscript and approving it for publication was Haruna Chiroma.

types are also important considerations in cloud elasticity, which must be investigated and characterized. Workload is of dynamic and disparate nature, which depends on user, and web-to-web activities, and events like the social media web 2.0 multi-tier applications. The workload is affected by inter-user activities, e.g. a famous person's tweet can cause a massive workload from other people, which may increase or decrease the resource demands unpredictably. As can be seen, the workload type and the resource configuration are two related factors that must be considered in cloud elastic actions. Good matching between workload demands and optimal resource provisioning will reduce the cost both for cloud service providers and the cloud users.

The challenges [3] in cloud elastic resource provisioning are as following: 1) accuracy in scaling resources to match workload demands, 2) management cost time and space complexity to find optimal configuration set, 3) configuration cost, cloud orchestration and initial service setup time (spin up time [4]), and 4) scaling dimensionality (scale in or out) i.e. resource scaling type (vertical or horizontal [5]) and unit of scale capacity resource units. Cloud resource capacities has been enhanced and developed by big cloud players who introduced cloud computing hardware platform to support customer's business requirements such as Amazon, Google, and Microsoft. The idea is to increase granularity of provisioned resources like servers' processes, storage and network that allow more control over performance and cost. Intel and Amazon tried to make processor aware about workload by increasing or decreasing processor speed, and the number of instructions per clock cycles (number of MIPS per second) [6]. Facebook also worked on cloud hardware platform by introducing an open source project called Open Compute Project (OCP) [7]. The goal of this project is to allow cloud services to choose most suitable hardware (server, storage, network) design for cloud data center [8]. Microsoft innovated the Olympus hardware project that is a ''next generation hyperscale cloud hardware design and a new model for open source hardware development with the (OCP) community'' [9]. Google cloud data center cluster is built by utilizing existing servers dynamically [10], which is a web-based subscription architecture model used to link resource units. The combinations of all these technologies increase the provisioned complexity and mapping tasks to workload demands. By labeling resources using unsupervised machine learning clustering method will allow data center cloud manager to accurately and efficiently configure the resources at reduced cost.

Workload applications are varied in cloud environments especially with changes in virtualization technologies and overlay networking setup. Software Defined Networking (SDN) [12] and Network Function Virtualization (NFV) [11] are impacting the resource provisioning based on network topology setup and services that need to be investigated and labeled as classes demands and released resources. Workload clustering in SDN and NFV environments is crucial, because it will reduce resources orchestration cost and time by defining architectures groups of computing resources that can interchange in provisioning and releasing the resource during service time. This allows cloud manager to pick one of these groups of data center resources that can match overlay network changes.

Clustering of cloud workload and configuration set is very important to make elastic decision too. By clustering workload and datacenter configuration, it can define a labeled data set such that the cloud management system can decide the best action based on the predefined demand classes to provision resource classes, using look up table. Many works had been done investigating cloud workload using real log traces. Google Cluster workload traces [13] have been clustered and analyzed in [14]–[23]. Also, Alibaba cluster traces are investigated in [24] to validate workload behavior in real cloud environment. MapReduce workload in Taobao e-commerce company has been studied in [25] to understand workload characterization in large scale cloud environment.

Reading cloud resource and workload activity logs in real time manner to be used in management decision action, is investigated in [26], [27]. Dimensionality reduction methods (wavelet transform and Principal Component Analysis PCA) are used to store and replay logs of Google Cluster workload traces to allow machine learning model utilizing decision time. Another method to reduce clustering time of cloud workload using Hyper-gamma distribution applying moments method is proposed in [28]. Defining a set of classes for workload types and cloud data center resource configuration set is a difficult task that needs to be addressed since provisioning decision will be based on these two parameters.

Our contributions in this work are to introduce an enhanced K-Means clustering approach for cloud workloads and datacenter configurations types, and a correlation matrix that evolves the relation between cloud resources and workloads. Our proposed method analyzes workloads and datacenter configuration traces based on kernel density estimator to find number of classes and classes center. Our method labels cloud workload demands and datacenter configuration capabilities to investigate workload types by related jobs submitted and data center capabilities, server types, capacity, and configuration setup. A dynamic mapping method based on demands and provisioned resources (expert system) is proposed to make the proper action during accepted configuration time. This will allow cloud service providers to respect service level agreements and minimize cost for customers and service providers.

This will allow cloud manager to find the best match between requested resources with provisioned resources considering all workload and data centers characteristics by using real data from cloud data center traces. The tasks carried out to achieve this goal are as follows: 1) analyze data center resource capabilities and configurations, 2) analyze workload demand types and behavior from different perspectives such as resource types needed (CPU, RAM, Network, Storage), demand speed, mass, format and messaging, protocols, and communications, 3) label workload and

data center configuration with appropriate classes, 4) develop proper mapping that achieves the best elastic match between demands and resource provisioning, and 5) apply methods to reduce resource provisioning complexity, time, and cost by using hash look up with a key-value pair of demand and provisioned resources, respectively.

This paper is organized as follows. Section II presents the related work, and methodology background is discussed in Section III. Our proposed Dynamic Clustering method is presented in Section IV. Experiments and results are shown in Section V and conclusion is presented in Section VI.

II. RELATED WORK

Workload clustering and classification have been studied in literature for many reasons such as for cloud modeling and evaluation, workload emulation and forecasting, and cloud resource reconfiguration. An adaptable model for generic large-scale workload is proposed in [29]. In this work, the authors formulate an accurate, realistic, and adapted workload model. They use Google Cluster Workload Trace schema to represent large scale workloads. The analysis of the workload is defined using four laws: 1) submission time, defined by the inter-arriving rate of task and modeled by Pareto distribution, 2) type, i.e. either task type or service type, 3) make span, defined by the task duration and modeled using long-tailed distributions like Pareto or log-normal, and 4) priority of the tasks order based on importance. K-Means is used to cluster submitted jobs with k equal to four classes for task and service types. The modeled workload characteristics are dynamic in nature with different parameters such as frequency, mass, and disparity. In [24] authors cluster Alibaba cloud cluster traces using K-Means method to analyze and identify job group characteristics and the relationship among different job groups which in turn will help in scheduling jobs at run time. Authors in [23] proposed simple set theory to enhance K-Means clustering methods for cloud workloads and data center configuration parameters. In [30] a Markov model with hidden layers is used to characterize workload stochastic behavior to cluster and classify workload patterns. K-Means is used in [20] to cluster google traces jobs and tasks as workload characterization to gain insight on the performance of workload demands. Hierarchical and conventional K-Means clustering is used to characterize workload by assigning common groups of jobs and common groups of machines in [21]. The goal is to schedule submitted jobs with appropriate data center resources while reducing power and batch job assignment latency.

An elastic scaling framework for cloud computing layers using combined AI methods with optimization are discussed in [31], [32] and [33]. Authors in [31] analyzed and clustered workload using metaheuristic-based method for elastic scaling. Their work depends on two AI methods: Genetic Algorithm (GA), and Fuzzy C-means. An elastic framework with hybrid workload clustering using two methods, K-Means, and Imperialist competition algorithm, is proposed in [32]. In [33] resource provisioning framework is proposed for elastic scaling in cloud PaaS layer using autonomic computing and reinforcement learning (RL).

VM consolidation considering workload characterization patterns in cloud data center is proposed in [18]. The authors propose a fully distributed and threshold free Dynamic Virtual Machine Consolidation (DVMC) algorithm called GLAP that combines Q-Learning with a gossip-based protocol. A VM consolidation method is proposed in [34] that uses optimization methods to reduce multi-tier workload latency. Workload modeling based on common pattern clustering of the workload is proposed in [22]. Selecting clustering method that can match workload and data centers characteristics is an issue. A new approach to select clustering method that relates VMs to tasks in large data centers is proposed in [35], [36]. The framework selects the best clustering algorithm from a set of clustering algorithms based on cluster validation methods. In [37], the author proposed a scaling dimension method which is determined using linear algorithm related to the workload type. Workload analysis is used in [38] for optimal cloud resource configuration. In this work, deep reinforcement learning is used to handle heterogeneous big data workload. The authors proposed a new model (named SARA), which does three tasks: 1) cluster workloads into groups using Bisecting K-Means, 2) search the optimal configuration of clustered workloads using deep reinforcement learning, and 3) continue to cluster new workload and find the best configuration set.

The researchers in [17] analyze data center resource to find Zombie servers that consume energy by running in idle state. Through the workload analysis, it was shown that significant reductions in power consumption and CO_2 emission can be achieved by optimum resource scaling. Workload forecasting for elastic resource management in edge clouds is proposed in [39]. The authors use a combined method between time series analysis called Auto Regressive Moving Average (ARMA) and Elman Neural Network (ENN) to forecast workload based on error correction. Authors in [19] used a combination between K-medoids clustering algorithm and multilayer perceptron neural network (MLP) model for workload prediction. The proposed work focuses on prediction workload pattern of new submitted tasks based on pool of historical tasks.

Cloud resources and capabilities have been investigated in [40] where the authors surveyed cloud hardware resource design for AI-enabled cloud computing. Deep machine learning workload framework is proposed in [41] to handle machine learning stages (storing data sets, training phase, evaluation, and production model) in cloud environments. This will allow users to deploy and test machine learning models in distributed computing models without considering resource limitation and configuration using standard Application Programming Interface (API). In [42], the authors used reinforcement learning method on GPU clusters for deep learning workload scheduling. To the best of our knowledge, there is no method to dynamically cluster cloud workloads and data center resource configuration set.

TABLE 1. Related work summary.

Clustering Method	Workload Type	Experiment Design	Performance Evaluation
K-Means [20], [21], [23], [24], [29], [35], [36], [38]	Google [17]–[23], [29]	Statistical model [17], [19], [24], [29], [30], [34], [39]	Execution and submission time [20], [23], [24], [29], [32]–[35], [39], [41]
Hierarchical [21], [36]	Alibaba [24]	Correlation vector [20]–[24], [36], [41]	Memory, CPU [17]–[22], [33], [37]
Hidden Markov [30]	Synthesized workload [30], [33]–[37]	Analytical model [18], [30], [31], [37], [39], [42]	Workload semantic [21]–[23], [30], [35], [36]
Hybrid model (Genetic algorithm, Fuzzy C-Means) [33]	FIFA, NASA [31], [32]	Framework model [31]–[35], [39], [41]	Cost, elasticity [33], [36], [38], [39]
Hybrid model (Imperialist Competition Algorithm (ICA), K-Means) [32]	ClarkNet [31]		Power [17], [18]
Reinforcement Learning [31], [42]	HiBench [38]		Scalability [37]
Bisecting K-Means [38], Q-Learning [18], [38]			
Dynamic Time Warp [19], [22]			

Specifying number of classes and initial group characterization for cloud workload and resource configuration is the first step in dynamically clustering datacentre workload. All workload aspects such as needed resources (RAM, CPU, Disk, Network), duration, volume, speed, location, and tasks must be considered for the analysis. In addition, data center resources in cloud environment continues to change in terms of capabilities, computation power and configuration technology. Servers, storage, and network infrastructure must be characterized such that they can be adapted for new resource types, workload types and provisioning methods. This work focuses on determining number of classes dynamically adapting to the changes of workload demands and data center resources using hybrid method that uses probabilistic statistical theory and unsupervised learning with optimization for selecting best solution.

Table 1 summarizes the related work based on four criteria: clustering method, workload type, experiment design, and performance evaluation.

III. METHODOLOGIES REVIEW

Methodologies used in this work are based on statistical, optimization theory and machine learning model. Using statistical analysis methods to find cloud data center traces logs mean and variance, then formulate mapping function of random variable model, allow to model workload and datacenter configuration characteristics. Unsupervised machine learning customized K-Means method is then applied to minimize population set inertia (mean square error between same group set) guided by density function to find out number of centers (means) and centers initial values.

A. KERNEL DENSITY ESTIMATE

Kernel Density Estimate (KDE) [43] is a method that can describe sampled statistics population probability distribution using random variable definition. The probability density function (pdf) for continuous model or probability mass function (pmf) for discrete model [44], are transformation functions for the events in the sample space which associate probability values to the outcome of random experiment. The estimator can expose some characteristics of statistical data such as skewness and multi-modality. A common approximation method for estimator is histogram, which is a frequency distribution for the population events. It is found by dividing the range of data into intervals named bins and then counting the number of data points in the intervals, which represents height of the chart. Define h as bin width, x_0 origin or start of sample range of population X set, m number of bins and $i \in 1, 2, \ldots, m$ an integer number to represent bin index, then bin interval is defined as $[x_0 + ih, x_0 + (i+1)h]$. The histogram $\hat{f}(x)$ can be defined as Equation 1, which shows the number of elements per bin divided by the product of bin width and the total number of data points, N. The term N is used in the equation for normalization. A more generalization of histogram method is obtained by making bin size variable in context to data set distribution as shown in Equation 2.

$$\hat{f}(x) = \frac{count(x_i)\ per\ bin}{Nh}, \tag{1}$$

$$\hat{f}(x) = \frac{count(x_i)\ per\ bin}{N\ (h\ of\ x_i\ bin)}. \tag{2}$$

From histogram, the probability of each bin can be defined by dividing number of elements in the same bin to the total number of elements. Using this method, a probability mass function ($f(x)$) can be formulated as a random variable to describe the behavior of the sample set. The main two statistical parameters that are considered in any random variable are mean (μ), which is the expected value defined as $E(x) = \sum xf(x)$, and variance that represents square of standard deviation (σ^2) $Var(x) = \sum x^2 f(x) - \mu^2$. Mean

point to the most common value of population and variance show how far population deviate from mean. These values can be obtained from data set using sampled mean and variance. However, these values are not enough to characterize the data fully and build statistical prediction model.

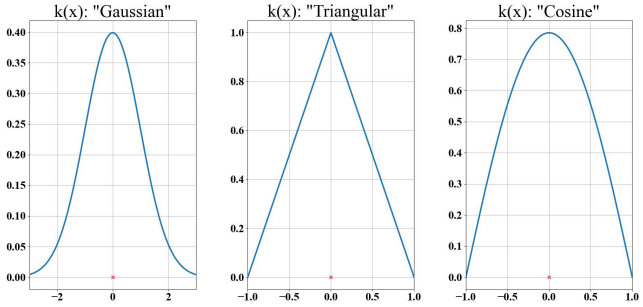


FIGURE 1. Kernel types.

A solution for this limitation is to use random variable model for population, which can provide all information about any population including mean and variance. Histogram is a simple clustering method that can help in finding population classes by grouping elements in bins (a naive description for random variable). In machine learning models [45], unsupervised learning method called descriptive learning aims to find the system interested output pattern that belongs to system output. Histogram clusters sample data input D without any information about the expected output values, where input data is defined as $D = \{x_i\}_i^N$ where N is the number of data points. Formulating probability mass function for sample data using histogram distribution as a random variable is used in clustering to indicate number of classes and the centroids initial values. However, the limitation in histogram method in clustering and nonparametric analysis is caused by the discontinuity of probability density function. A solution to this problem is to introduce a function that represents a weight for each population point and by finding the total overlapped weight. Using this function called kernel $k(x)$, the density estimator can be defined as Equation 3. Kernel function must satisfy three conditions: 1) it must be a positive function $k(x) \geq 0$, 2) it must be symmetric $k(x) = k(-x)$, and 3) it must be continuous and decreasing for $x > 0, k'(x) \leq 0$. There are many types of kernel function that can be used, but the most common used in machine learning Python libraries [46]–[49] are Gaussian, Triangular and Cosine as depicted in Figure 1.

$$\hat{d}(x) = \frac{1}{N}\sum_{i=1}^{N} k(x - x_i). \tag{3}$$

Introducing width h window attribute for the kernel function that can be used as a smoother for kernel estimator function, as Equation 4 depicts. The bandwidth of the estimator will show the overlapping in density in sampled space. This can show the points neighbor relation as likelihood estimators.

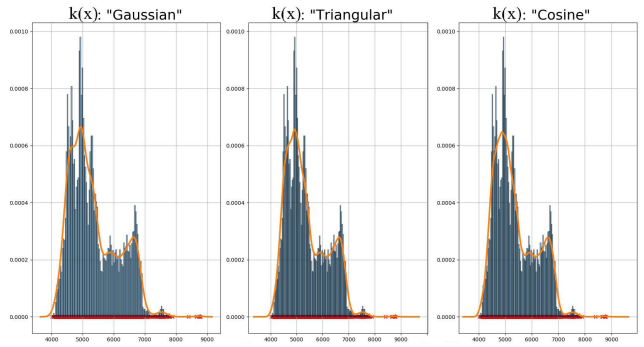


FIGURE 2. Kernel fitting.

$$\hat{d}(x) = \frac{1}{Nh}\sum_{i=1}^{N} k(\frac{x - x_i}{h}). \tag{4}$$

Generalizing kernel estimator using weight will make a correction for the estimator for high related point used as validation for the function fitting, as shown in Equation 5. The weight with the kernel function will smooth over all density function estimator. This function can be a tradeoff between over fitting and under fitting problems.

$$\hat{d}(x) = \frac{1}{h}\sum_{i=1}^{N} w_i k(\frac{x - x_i}{h}), \quad where \sum_{i=1}^{N} w_i = 1. \tag{5}$$

Choosing kernel function when there is a high number of sampled points does not have a big weight because the fitting shape will be the same for all types of kernel functions, as shown in Figure 2 for a RAM request workload. Here the number of bins is selected experimentally and it is 150. The important factor is to choose bandwidth h for the kernel function. It will impact the overall density function shape because it will control the density function smoothness, where wider bandwidth will make function more smooth and resilient against spikes shapes (non deterministic events), as shown in Figure 3.

A complex statistical method is developed by statisticians Ramsay and Silverman for choosing the bandwidth, h. The theory of this method is outside the scope of this paper, and provided in [50]. In our work, a toolbox in Python programming language library is used. The method uses sampled mean $\bar{x} = \sum_{i=1}^{N} \frac{x_i}{N}$ and sampled variance $s^2 = \sum_{i=1}^{N} \frac{(x_i - \bar{x})^2}{N-1}$ of the population. In Python library there are three types of bandwidth estimator that can find out the bandwidth based on statistical population. The default method with Gaussian kernel, called ''normal_reference'', can be shown by Equation 6, where $A = min(s^2(X), IQR(X))$ is inter quartile range, N number of data points, and C is a constant matrix of Hansen values based on Silverman rule of thumb [50].

$$h = \frac{A \times C}{\sqrt{N}}. \tag{6}$$

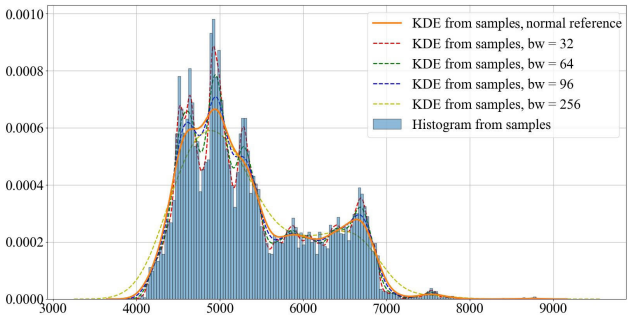


FIGURE 3. Kernel bandwidth fitting.

B. K-MEANS

K-Means is an unsupervised learning model that works to cluster data set into specific number of cluster groups defined by random or selected centroid values $\mu_1, \ldots, \mu_k$, where μ_k represents centroid of cluster k. $\mu_k = \frac{1}{p}\sum_{i \in c_k} x_i$, where $p = |c_k|$ represents number of data group points in cluster k, and c_k a subset of C represents clustered grouped points. It works by iteratively finding the minimum distance between centers and all points x_i in the same cluster group set c_k, until reaching distance threshold. The objective function J is defined as the Mean Square Error (MSE) of the distance between centroid and data point member in same group, as shown in Equation 7. This objective function needs to be minimized at each iteration.

$$J(c_1, \ldots, c_p, \mu_1, \ldots, \mu_k) = \frac{1}{p}\sum_{i=1}^{p} ||x_i - \mu_{c(i)}||^2. \quad (7)$$

K-Means works in two nested loops that requires a time complexity of the product of number of group members and number of centroids, i.e. $O(k \times p)$. There are many clustering methods, which are characterized based on following perspectives [51]: 1) architecture, 2) robustness, 3) restrictions on latent features, and 4) reconstruction loss. K-Means is the simplest and most widely used method with three limitations: 1) setting number of clusters k and initial centroids points value μ_{c_p}, 2) finding out number of iterations per minimization cycle (minimize cost function $J(C, P)$) by defining the inertia distance threshold value, and 3) validation for the clustering and the accuracy for the class members in context to group points logical relation. There are many enhanced versions of K-Means method that overcome some limitations to be used in cloud workloads clustering [38] like Bisection K-Means [52], which solves the local minimum problem in objective function optimization. It works by setting k value to 2 for the data set for initial centroids to find two groups. It is close to hierarchical clustering method, top down (divisive), and evaluates each group using sum of the square error and repeatedly dividing until number of groups is smaller than k. All clustering algorithms work using unguided search on the data points set, which can produce non logical result. Using informative methods about population to help clustering method will significantly influence the result towards the best consistence and accurate groups. In this work a novel way is used to enhance K-Means method of clustering using kernel density estimator to cluster cloud workloads and data center configuration options dynamically.

IV. DYNAMIC CLUSTERING METHOD

In this paper, a combined method to cluster cloud workloads and datacenter configuration is proposed using histogram, kernel density estimator and K-Means. This guided clustering method will dynamically determine number of classes based on statistical population characteristics of cloud workloads and data center configuration. Figure 4 depicts the flow chart of the proposed dynamic clustering method. It is built by integrating five phases: 1) Silverman to find bin width and number of bins, 2) Histogram to describe the logs trace distribution for attribute extraction and correlation, 3) KDE to find random variable for the log type that generates PDF and CDF, 4) K-Means to cluster correlated logs, and 5) Validation of the clustering process to check accuracy of clustering. The proposed method starts by data cleaning and ends by validating the clustering result as depicted in Algorithms 1, 2, 3, 4, and 5, which are discussed in details later.

The flow chart in Figure 4 begins in Algorithm 1 by gathering and storing the logs and replaying in a timely manner to be processed. Data cleaning and extraction is done using Algorithm 2 to evaluate logs value, more specifically to check whether it is empty or of a wrong data type or format. It is followed by extracting log attributes as vectors, which are processed individually. The attribute vectors are statistically analyzed to find population mean, variance and Hessian constant matrix. These values are used to calculate bin bandwidth using Silverman Equation 6, in order to find number of bins over all the population range. After that histogram is applied, as defined in Equation 2, and the sampled mean and variance are obtained for population relation evaluation for all points of each group, to create CDF using Equation 5 in Algorithm 2. A new vector $M = \bar{x}_1, \ldots, \bar{x}_k$ represents sampled mean of all bin point members of each group that will be used as K-Means initial centroids. The empty bins must be removed, and the number of bins must be updated as the k value, the number of clusters. Information that must be obtained for clustering are by correlating the two attributes, which are number of means (vector size) and the initialization of each bin center. Two vectors with the minimum number of classes are joined, and vector with higher number of classes are merged to same as number of vector with lower number of classes (Algorithm 3). Algorithm 4 applies K-Means clustering using Equation 7, and evaluates the clustering inertial using kernel density estimate (Equation 3) distributions. A full information about the data set attribute vectors will be formulated using KDE that generates the probability distribution function, and probability density function. These two functions can inform the bin group probability volume, using a convenient ratio factor as the event probability weight. Bin boundaries can be defined based on bandwidth of kernel function by segmenting the distribution into the most representing

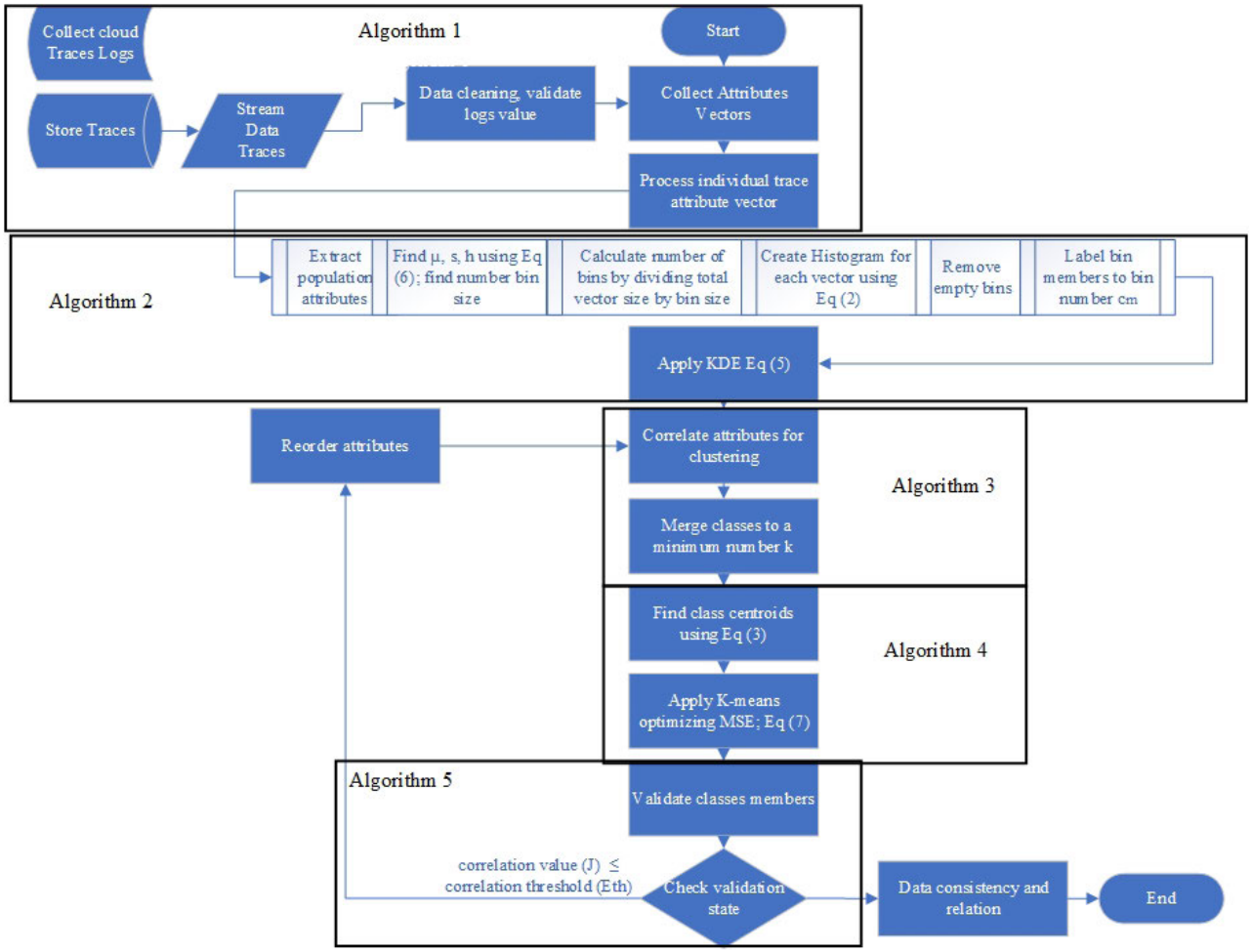


FIGURE 4. Algorithm flow chart.

bins, that is the bins that have a higher probability value. The result of this stage will be used to validate the group points relation. Finally correlation and validation is applied for clustering, as implemented in Algorithm 5.

Table 2 summarizes all symbols used in the equations and algorithms.

The five phases of the proposed dynamic clustering process are presented in detail in the following algorithms:

1) Algorithm 1, Logs Cleaning and Normalization phase: In this part, the algorithm starts by reading the data center and workload demand trace attributes in a real time manner. These trace attributes are processed individually. Cloud data center and workload log traces are raw data that must be processed before being clustered. The processing involves extracting valid and meaningful log traces, by checking log values to see if there is any missing or an invalid value. The algorithm will approximate the missing value by averaging the neighbors of the current point as shown in Line 2 of Algorithm 1. Log values must be normalized to the maximum value for all log types as rational value to resource units, as shown in Line 6.

TABLE 2. Symbols summary.

Symbols	Definitions	Symbols	Definitions
h	bin width	x_0	origin point of data set
X	data set	m	number of bins
i	integer index	N	total number of data points
k	number of clusters	p	number of points that belong to the same group
$\hat{f}(x)$	histogram	μ	mean
σ^2	variance	D	data logs
x	log point value	$k(x)$	kernel function
$\bar{x}$	sampled mean	s^2	sampled variance
$Attno$	attributes number	$Binc$	bin count
H	histogram array of bins	$\bar{M}$	sample mean array of m points
Bb	bin boundary	Bpl	bin points member label
MSE	mean square error	KDE	kernel density estimator
PDF	probability density function values	CDF	cumulative density function values
$A1, A2$	attribute vectors	Eth	threshold error
E	Euclidean distance vector	V	Silhouette values
SE	standard error		

Tables 3 and 4 show the lists of workload and data center configuration attributes. Table 3 summarizes the extracted workload trace attributes where each attribute is denoted as WL_i. Table 4 summarizes the datacentre

TABLE 3. Workload extracted attributes.

Workload Attributes Name	
Number of submitted jobs per time window (WL1)	Number of submitted tasks per job (WL2)
CPU demands as resource units (WL3)	RAM demands as resource units (WL4)
Task duration (WL5)	Task volume, total number of submitted tasks (WL6)
Task speed, number of tasks finished in time (WL7)	Number of failed tasks (WL8)
Overall demanded accumulative CPU resource, until current time (WL9)	Overall demanded accumulative RAM resource, until current time (WL10)
Task resources usage (WL11)	Number of successful tasks (WL12)

TABLE 4. Datacenter configuration extracted attributes.

Datacenter Configuration Attributes Name	
Number participated machine (DC1)	Machine list (DC2)
CPU provisioned as resource units (DC3)	RAM provisioned as resource units (DC4)
Machine list activity during time configuration window (DC5)	Total CPU provisioned, scale in or out (DC6)
Total RAM provisioned, scale in or out (DC7)	Scaling factor, scaling accuracy during time window (DC8)
Datacenter Capacity, CPU available resource (DC9)	Datacenter Capacity, RAM available resource (DC10)
Datacenter Capacity, CPU used resource (DC11)	Datacenter Capacity, RAM used resource (DC12)

attribute vector which is denoted as DC_i. Both attribute types WL_i and DC_i are part of the normalized data set array D stored in the Tables as colon. This notation is used to build a relation table for presentation simplicity.

Algorithm 1 Logs Cleaning and Normalization

Require: X /* X is input logs matrix. */
1: **for** $i \leftarrow 0$ to $Sizeof(X)$ **do**
2: **if** $nan(x_i)$ **then**
/* nan, a function to validate vector elements. */
3: $Approximate(x_i)$
/* Approximate missed value using average of neighbor points.*/
4: **end if**
5: $Max_i \leftarrow FindMax(X_i)$
6: $D_i \leftarrow Normalize(x_i, Max_i)$
/* normalize each attribute vector value to the maximum resource or log value. */
7: **end for**

2) Algorithm 2, Preparation and Initialization phase: This part involves applying statistical analysis on cleaned trace attributes individually. It starts by finding population attributes for each trace vector and then finding sampled mean and variance for the whole population. By using Equation 6 (applying the Silverman rule) the population segment bandwidth is calculated in Line 4. Bin numbers, $Binc_i$ can be found in Line 5 by dividing vector size by step bandwidth. These values are used in Line 6 to find attribute vector segments (class or bin boundary) Bb_i, class point members label Bpl_i, and points count for each bin group H_i. Lines 8, 9 and 10 calculate sample mean, variance and mean square error between the bin points. Then, using kernel density estimator equation (Line 11), the probability density function and cumulative density function are obtained as shown in Lines 12 and 13 respectively. Finding CDF has high time cost because of integration process.
3) Algorithm 3, Merge phase: In this part a relation between attribute vectors are calculated. First,

Algorithm 2 Preparation and Initialization

Require: $D = \{x_i\}_i^N$ /* Initialize attribute vector x_i. */
1: $X \leftarrow D$ /* Initialize array X by data set matrix. */
2: $Attno \leftarrow AttributesNumber$ /* Column number of X, x_i is one single attribute vector. */
3: **for** $i \leftarrow 0$ to $Attno$ **do**
4: $h_i \leftarrow BinBandwidth$ /* Find Bin Bandwidth Array using normal reference set h bin size, using equation 6 Silverman rule. */
5: $Binc_i \leftarrow \frac{sizeof(x_i)}{h_i}$ /* Find bin count $Binc_i$ for each attribute by dividing attribute size by bandwidth size. */
6: $H_i, Bb_i, Bpl_i \leftarrow \hat{f}(x_i)$ /* Apply Equation 2 to find histogram array H_i and its bin data points count, bin boundary Bb_i, and bin points member label Bpl_i. */
7: $H, Bb, Bpl \leftarrow MergeToOneVector(H_i, Bb_i, Bpl_i)$
8: $\bar{M}_i \leftarrow \sum_{j=1}^{m} \frac{x_{ij}}{m}$ /* Sample mean array of m number of points in that vector. */
9: $S_i^2 \leftarrow \sum_{j=1}^{m} \frac{(x_{ij}-\bar{x}_i)^2}{m-1}$ /* Sample variance array. */
10: $MSE_i \leftarrow J(x_i, \mu_i)$ /* MSE array for data point in each bin Equation 7, this will validate the bins point data consistency for later use in validation phase. */
11: $KDE_i \leftarrow \hat{d}(x_i)$ /* Apply Equation 4 to find KDE for fast processing in online mode. */
12: $PDF_i \leftarrow \hat{d}(x_i)$ /* Apply Equation 5 to find PDF array for validation in offline mode. */
13: $CDF_i \leftarrow \sum_{j=1}^{m} PDF_{ij}$ /* Calculate CDF array from PDF_i, offline mode for validation.*/
14: **end for**

the histogram empty bins are identified and removed, as shown in Line 4 of Algorithm 3. Next, two attribute vectors are combined based on minimum vector bin numbers. The merging process works on converting the longer vector to the same size of shorter vector attribute bin number, by finding the merge step size, as shown in Line 7. After the merge step is obtained, this value is used to find the bin boundaries and new bin labels,

as shown in Lines 11 and 12, respectively. In Line 13, the new point count for each bin is calculated. Finally, through Lines 14 to 17, the new mean and variance are calculated for the merged bins.

4) Algorithm 4, Correlation and Clustering phase: This part works by applying K-Means using the number of bins as class number k and the bins mean of each group members as centroids, as shown in Lines 1 and 2 of Algorithm 4. Clustering accuracy error is evaluated using the maximum *inertia* value as shown in Line 4. This is an iterative process until the error value becomes lower than the threshold value. Here *inertia* is the Mean Square Error matrix of clustering classes for each bin member point. Next, the classes' centroid values are updated in Line 3 using standard deviation as a step by step guide projection. The *inertia* error matrix is compared with bin's variance vector between group members as shown in Line 7, and overall mean square error is calculated in Line 8.
5) Algorithm 5, Validation phase: This part focuses on validating the accuracy of the dynamic clustering algorithm. The algorithm starts by initializing an evaluation relation matrix *MM* in Line 1 to a default value of zero (means no relation). This matrix is used as a test for attributes' relation and clustering accuracy. The *MM* matrix represents the relation as strong with a value of 1 (Line 12), good with a value of 0.75 (Line 14), normal with a value of 0.50 (Line 16), and weak with a value of 0.25 (Line 18). Testing the relation conditions are based on statistical threshold values. The first test is done by comparing the maximum Euclidean distance *Eth* with the standard error, as shown in Line 8. In the second test Silhouette value clustering index, which indicates a relation consistency between classes' point members, is checked against a threshold value of 0.6. If the Silhouette value is lower than 0.6, it indicates a good clustering K-Means relation, as shown in Line 13. In the third test the slope value per bin is compared against a threshold value, as shown in Line 15. The threshold value is selected as 0.7 of the maximum slope since it was statistically found (our previous work [23]) that 0.7 of maximum slope shows a normal relation between the attributes.

A. ALGORITHMS COMPLEXITY

Our proposed approach works based on capturing logs during time window which is of a fixed size ($sizeof(X)$), which means in general the whole algorithm complexity is a constant. Since the buffer size is also fixed, the time complexity is also a constant. Algorithm complexity analyses are different for individual algorithms. For Algorithm 1 Logs Cleaning and Normalization, and for Algorithm 2 Preparation and Initialization the complexity is $O(1)$. For Algorithm 3 Merge, time complexity depends on the minimum value of Silverman bins number (Min) and logs dimensions ($Attno = sizeof(X)$). There are nested dependent loops that

Algorithm 3 Merge Phase

Require: $X, Binc, H, Bb, Bpl, M, S, MSE$ /* *All input arguments are obtained from preparation phase as arrays of attributes.* */

1: **for** $j \leftarrow 0$ to $sizeof(X)$ **do**
2: **for** $c \leftarrow sizeof(X)$ to $j+1$ **do**
3: $A1, A2 \leftarrow x_j, x_c$ /* *extract two attribute vectors.* */
4: $Binc_j \leftarrow cleanEmptybins(A1)$ /* *cleanEmptybins() function removes any histogram empty bin count, then updates original bins count.* */
5: $Binc_c \leftarrow cleanEmptybins(A2)$ /* *this will reduce number of cluster centroids that might be considered by dividing the population range using fixed bandwidth size.* */
6: $Min, Max \leftarrow MinMax(Binc_j, Binc_c)$ /* *This function will find minimum value and maximum value of two numbers, for merge algorithm will consider minimum number of bins as reference, which will represent number of cluster in K-Means.* */
7: $MergeStep \leftarrow Floor(Max/Min)$ /* *Find overlap step to convert larger number of bins to be same number as the minimum.* */
8: **for** $k \leftarrow 0$ to Min **do**
9: **if** $(k + MergeStep) < Min)$ && $(k + 2 \times MergeStep) < Min)$ **then** /* *validate loop boundary not to exceed the array limit.* */
10: $Bba1 \leftarrow Bb_{cj}[k : k + MergeStep]$ /* *update attributes bins boundary.* */
11: $Bpla1 \leftarrow JoinLabel(Bpl_j[k : k + MergeStep])$ /* *update attributes new bin label.* */
12: $Ha1, Ha2 \leftarrow Sum(H_j[k : k + MergeStep]), H_c$ /* *update attributes bins points counts.* */
13: $\mu_1 \leftarrow M_j$ /* *update means for each combined bins. find new mean for each merge bins.* */
14: $\mu_2 \leftarrow M_c$ /* *get mean values for lower bins number list.* */
15: $s1 \leftarrow S_j$ /* *get variance values for lower bins number list. find new variance for each merge bins.* */
16: $s2 \leftarrow S_c$ /* *get variance values for lower bins number list.* */
17: **else**
18: $Arraysindex \leftarrow (Min - k)$ /* *for special case index last bin.* */
19: **end if**
20: $k \leftarrow k + MergeStep$ /* *update index by increment by merge step.* */
21: **end for**
22: $CallCorrelationClustering(Bba1, Bpla1, Ha1, Ha2, \mu_1, \mu_2, s1, s2)$
23: **end for**
24: **end for**

TABLE 5. Data set attributes relation.

	WL1	WL2	WL3	WL4	WL5	WL6	WL7	WL8	WL9	WL10	WL11	WL12	DC1	DC2	DC3	DC4	DC5	DC6	DC7	DC8	DC9	DC10	DC11	DC12
WL1	00000	73865	72558	76679	78548	69145	70557	67660	78548	74819	75463	71017	72883	76410	73631	74738	72916	78083	74610	78647	74898	70909	72313	74899
WL2	3	00000	33222	48124	22198	43037	58513	42145	52558	36050	60731	34351	37067	61554	38849	52837	61718	53809	75576	50690	42433	68065	60807	73014
WL3	3	74	00000	59225	42295	68120	45163	31041	57388	72741	52041	36896	50768	40098	72140	42674	42159	40321	54125	78970	43827	54724	67443	76896
WL4	3	105	74	00000	59903	44793	548043	52332	3629	38981	45216	47875	63234	36269	39800	42220	51645	47934	65163	53889	57528	66520	69558	66794
WL5	3	122	74	105	00000	29140	23843	40778	53124	73869	46027	69767	48090	65638	51378	58597	51101	33272	53570	53032	98017	54725	54024	65966
WL6	3	45	45	45	45	00000	38927	34698	36106	34511	24911	46447	61470	63857	38573	47326	46162	67408	31817	63013	30136	68568	61549	70936
WL7	3	13	13	13	13	13	00000	26585	45045	41455	32957	31533	37096	46869	34441	32390	52177	30977	62619	52283	62831	55399	62318	75400
WL8	3	95	95	95	95	45	13	00000	42892	42099	31980	42218	30957	47932	63013	50745	51104	47910	41257	47242	40321	69982	69680	66896
WL9	3	8	8	8	8	8	8	8	00000	40961	48516	57443	37214	48043	24558	55516	68927	69096	59898	29897	51257	61288	50936	68920
WL10	3	39	39	39	39	39	13	39	8	00000	42285	40214	25366	59941	24665	67730	61973	57928	63758	50323	59825	35067	61923	72707
WL11	3	65	65	65	65	45	13	65	8	39	00000	38882	39367	36137	42973	75086	58120	52199	63103	56922	51329	53442	58110	76672
WL12	3	40	40	40	40	40	13	40	8	39	40	00000	35752	36134	33705	52067	40957	49421	59756	43684	49850	43939	70403	72026
DC1	3	22	22	22	22	22	13	22	8	22	22	22	00000	68286	43993	41832	69467	60271	43012	43223	43458	42305	65382	76645
DC2	3	34	34	34	34	34	13	34	8	34	34	34	22	00000	64191	60186	47768	47031	52100	47380	5994	60065	58421	77296
DC3	3	38	38	38	38	38	13	38	8	38	38	38	22	34	00000	64016	42985	68228	48413	47344	69417	42702	61861	78736
DC4	3	34	34	34	34	34	13	34	8	34	34	34	22	34	34	00000	35217	64418	39467	67831	45125	55030	49983	70164
DC5	3	9	9	9	9	9	9	9	8	9	9	9	9	9	9	9	00000	58839	50890	59329	60893	61893	61137	65442
DC6	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	00000	47236	41909	50885	42781	58919	59428
DC7	3	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	4	00000	69320	61110	65895	52281	66267
DC8	3	19	19	19	19	19	13	19	8	19	19	19	19	19	19	19	9	4	6	00000	51667	59980	52844	65429
DC9	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	00000	53751	61504	52884
DC10	3	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	4	6	7	4	00000	63859	73827
DC11	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	00000	62152
DC12	3	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5	5	4	5	4	00000

Algorithm 4 Correlation and Clustering Phase

Require: $Bba1, Bpla1, Ha1, Ha2, \mu_1, \mu_2, s1, s2$

1: $Threshold \leftarrow 0.0001$ /* *initialize tolerance threshold value.* */
2: $centroids \leftarrow [\mu_1, \mu_2]$ /* *K-Means centers are initialized by pair of the two attributes means.* */
3: $classno \leftarrow sizeof(Ha2)$ /* *both attributes arrays are same size (number of bins).* */
4: **while** $Max(inertia) \geq Threshold$ **do**
5: $inertia \leftarrow Kmeans(centroids, classno,)$ /* *call K-Means function with initialization of centroids and number of classes, return Error of clustering classes.* */
6: $centroids \leftarrow [\mu_1, \mu_2] \times [s1, s2]$ /* *K-Means centers are updated by multiplying standard deviation with initialized mean centered, instead of using random initialization.* */
7: $eval \leftarrow compare(inertia, [s1, s2])$ /* *evaluate relation between classes points relation to the attributes bins itself relation using standard deviation.* */
8: $Jeval \leftarrow J(eval)$ /* *Store two attributes population as attributes relation matrix. Find lower error code value (chosen as table 5 shows), lower error code and deviation values indicate higher relation.* */
9: **end while**

Algorithm 5 Validation Phase

Require: $inertia, CDF, PDF, MSE, Attno, classno, Jeval, s1, s2$ /* *using standard K-Means methods of validations. Attno is the set of correlated attributes in table 5* */

1: $MM \leftarrow [0]$ /* *relation matrix index for attributes statistical correlation.* */
2: $E \leftarrow Euclidean(inertia)$ /* *evaluate distance between the points in each class point members with MSE.* */
3: $V \leftarrow Silhouette(inertia)$ /* *evaluate threshold accepted value.* */
4: $S \leftarrow Slope(CDF)$ /* *evaluate slope for each bins range.* */
5: $SE1, SE2 = \frac{s1}{\sqrt{m_{s1}}}, \frac{s2}{\sqrt{m_{s2}}}$ /* *find standard error SE, where m_s number of points in class.* */
6: $SE = Min(SE1, SE2)$
7: $Sth = 0.7\ Max(S)$ /* *find slope relation as 0.7 of max slope.* */
8: $Eth = SE$ /* *find attributes relation as ratio of maximum distance.* */
9: **for** $i \leftarrow 0$ to $Attno$ **do** /* *loop for all attributes.* */
10: **for** $j \leftarrow 0$ to $classno$ **do** /* *loop for clustering minimum number of class Min.* */
11: **if** $E_{ij} \leq Eth$ && $V_{ij} \leq 0.6$ && $S_{ij} \geq Sth$ **then**
12: $MM_{i,j} \leftarrow 1$ /* *one means a high correlation and clustering.* */
13: **if** $V_{ij} \leq 0.6$ && $S_{ij} \geq Sth$ **then**
14: $MM_{i,j} \leftarrow 0.75$
15: **if** $S_{ij} \geq Sth$ **then**
16: $MM_{i,j} \leftarrow 0.50$
17: **else**
18: $MM_{i,j} \leftarrow 0.25$
19: **end if**
20: **end if**
21: **end if**
22: **end for**
23: **end for**

can be solved using series, with time complexity $O(Min \times log(Attno))$. Algorithm 4 Clustering and Correlation has a constant time complexity of $O(1)$, where K-Means class numbers are set and center initialization is updated using projection multiplication process. This is because the updated process is limited by a constant iteration number that does not exceed the standard deviation. Algorithm 5 Validation Phase cost is $O(Min \times Attno)$ for two nested independent loops. The overall complexity of our proposed approach is therefore $O(Min \times Attno)$.

V. EXPERIMENTS

A. IMPLEMENTATION LIBRARIES AND TOOLS

Python programing language is used for development and implementation of the proposed methodology. For machine learning and statistical analysis, the following python libraries are used: (i) scikit-learn 0.22.2 [47], [48] for K-Means clustering and (ii) python library developed by Vanderplas [49] for kernel density estimation.

B. RESULTS

In this section, a full analysis of the proposed dynamic clustering method will be presented. For attributes correlation and clustering mapping, the relation between all attributes is shown in Table 5. In this table, the upper right triangle, separated by blue cells, represents the total of MSE (J) as a rational normalized value to the maximum value of 100000, that can represent clustering accuracy. The bottom left triangle of the table represents the number of classes k used to make the clustering. There is an inverse relation between the number of classes and the total MSE error value, such that with a lower number of classes the error value is higher. But in some cases, such as between datacenter attributes DC5 and DC4, the value of this error is very small because the distribution of the statistical points is close (e.g. machine list and CPU resource units are correlated factors).

The algorithm tests the relation between all attribute types to find attribute relation magnitude. The relation between the attributes is described using normalized MSE, as shown in Equation 7. A good relation has a lower index value. Clustering works to define number of classes and its member points that belong to each class. This creates a labeled data. The matrix in Table 5 shows the results of all the correlation experiments between all trace attributes that can indicate the matching relation between the attributes. There are two matching aspects; first, logical matching where correlation vectors considered are in the same characteristics set. For example, in Table 3 workload clustering is defined based on correlation between all attributes to each other, such as defining job classes according to number of submitted tasks. Second, statistical analysis of log trace correlation that can measure the nature of the trace's relation.

In Figure 5, job WL1 has initially 7 bins that are distributed among jobs range, which becomes three bins after histogram cleaning. As well, tasks WL2 has 403 histogram bins distribution and after cleaning it becomes 164 active sets. The merging set that shows jobs and tasks correlation will use the minimum number of classes, because population relation density (the PDF) of jobs are concentrated in a narrow range, as Figure 6 depicts. In Figure 6, CDF guides the probability of tasks and jobs occurring during small ranges. This can give us a small number of job classes based on the number of task classes relation, which is a good logical attribute for the workload characteristics, as Figure7 indicates the labeled points.

Repeating the correlation process for all attributes of workloads and data center configurations, we notice some relations that are not directly connected and correlated, such as correlation between submitted jobs $WL1$ and CPU demands $WL3$. As Figure 8 depicts, job (7 bins) distribution in histogram model is not related at all and there is no doubt that classifying jobs based on CPU demands (75 bins) will not be accurate. In Figure 9, the PDF indicates two totally non consistent distributions as well as it shows the job CDF slope is close to one, which indicates jobs are arriving in batches grouped with fast growing rate. On the other hand, CPU demand PDF

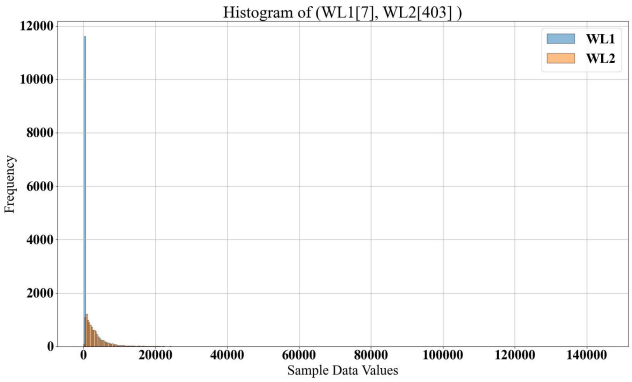


FIGURE 5. Histogram of jobs and tasks.

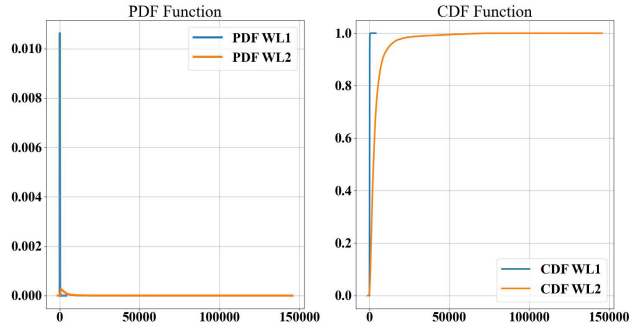


FIGURE 6. PDF and CDF of jobs and tasks.

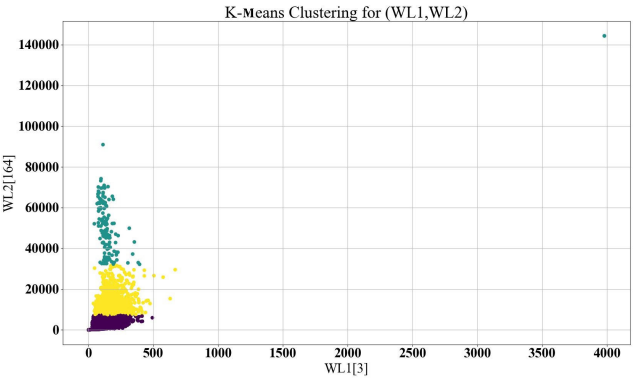


FIGURE 7. K-Means clustering of jobs and tasks.

is more distributed and the CDF slope is lower in magnitude, which also has a higher variance.

Figure 10 shows a view of jobs and CPU demands cluster labels. From Figure 10 we can conclude that most of CPU demand bins are segregated of jobs bin ranges except a limited number of job classes, due to which many of CPU demands classes' granular details may be lost. On the other hand, logical attribute for the workload characteristics, as shown in Figure 13, indicates a high correlation between tasks and CPU demands. From Figure 11, it is observed that there is a high overlap in the population histogram between number of tasks and CPU demands. There are a large number

of classes in both WL2 and WL3 attributes. For WL2 it is 403 and for WL3 it is 75. In the PDF and CDF shown in Figure 12 the clustering result of the CPU demands per submitted tasks is depicted. It shows that CPU demands with respect to tasks distribution is faster in growth and more concentrated in a fixed narrow range with high slope value. However, the submitted tasks distribution is wider in range and has a lower slope value in CDF. After merging and bin cleaning phase of Algorithm 3, the CPU demand bin number becomes 74 and the task bin number becomes 164. Figure 13 shows the clustering results from CPU demand and tasks for 74 classes, which is the minimum between CPU demand and task bin numbers.

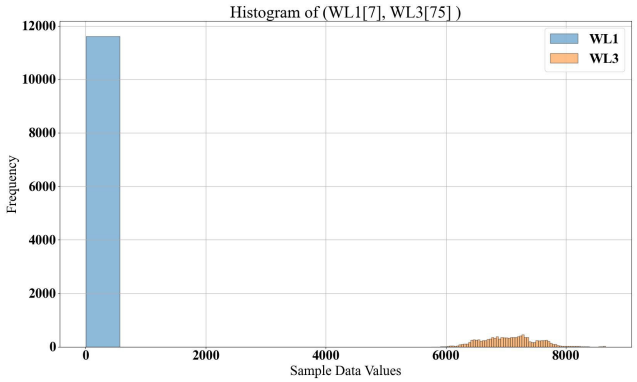


FIGURE 8. Histogram of jobs and requested CPU demand.

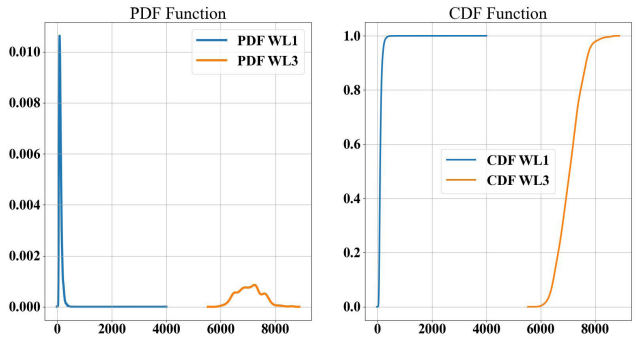


FIGURE 9. PDF and CDF of jobs and requested CPU demand.

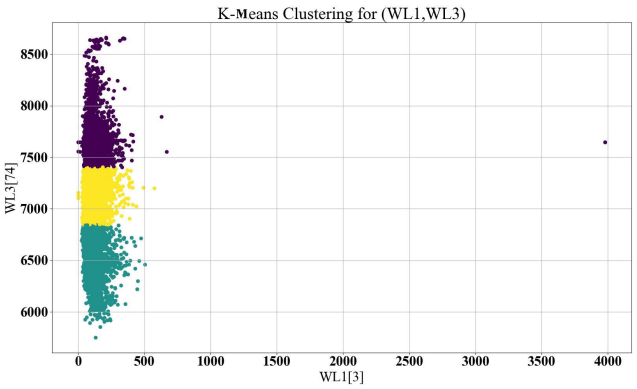


FIGURE 10. K-Means clustering of jobs and requested CPU demand.

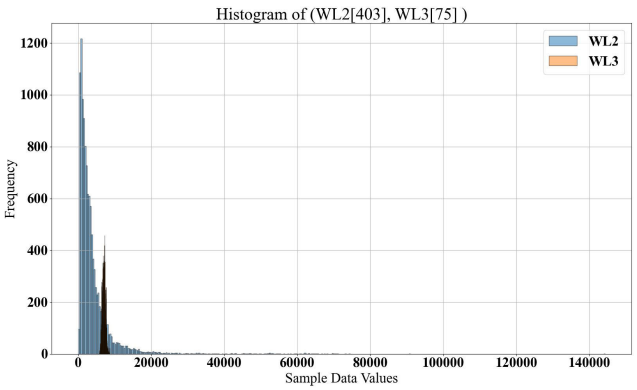


FIGURE 11. Histogram of tasks and requested CPU demand.

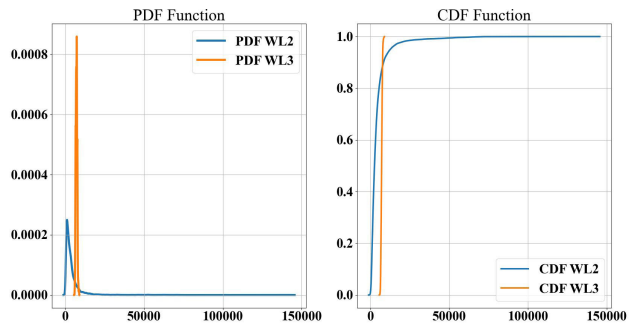


FIGURE 12. PDF and CDF of tasks and requested CPU demand.

For the datacenter configuration attributes set, the mapping between all of DC configuration attributes are listed in Table 4. All the attributes are correlated, but the most correlated attributes for elastic model are as follows: machine list (DC2) correlated with datacenter capacity of both available CPU and RAM resources (DC9, DC10) and provisioned CPU and RAM resources (DC3, DC4). The correlation result for total MSE is depicted in Table 5, and some examples are discussed in the following text. Not all of them are mentioned due to paper size limitation and similarity in result analysis. Similarity in DC attributes come from datacenter configuration method, which is based on just adding or removing physical hosts with minor differences in machine resource units.

Figure 14 shows mapping of two histograms: Provisioned RAM (DC4) with 34 bins and Datacenter RAM capacity (DC10) with 78 bins. The distribution of PDF provision RAM is more compact compared to datacenter configuration capacity (number of machines), which is continuously increasing. This means upgrading of the DC machines is expanded (scale out) as Figure 15 depicts. There are some lags in hardware scaling compared to provisioning resources, this is due to the capacity resource scaling as Figure 17 depicts. The full available resources are provisioned as they become available for both CPU and RAM. This reflects existing machine lists that are scaled out or the new added machines are involved

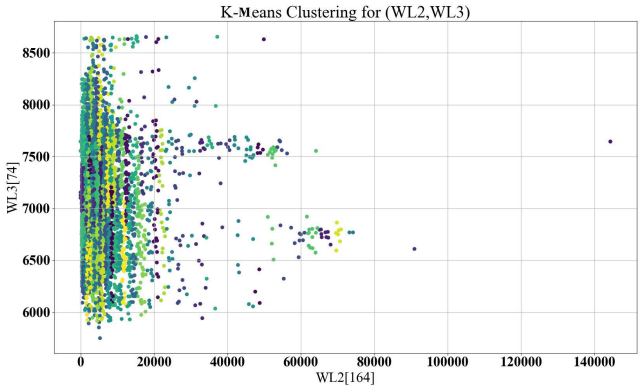


FIGURE 13. K-Means clustering of tasks and requested CPU demand.

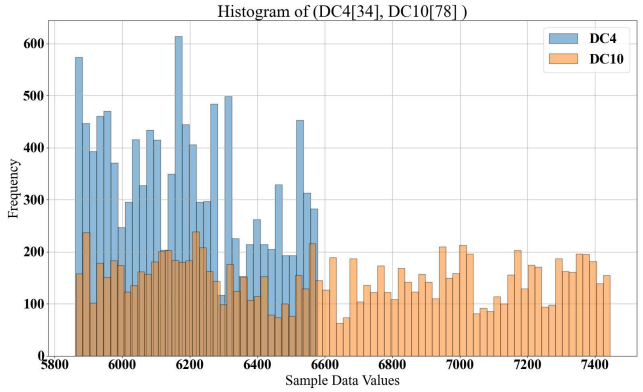


FIGURE 14. Histogram of provisioned RAM and datacenter RAM capacity.

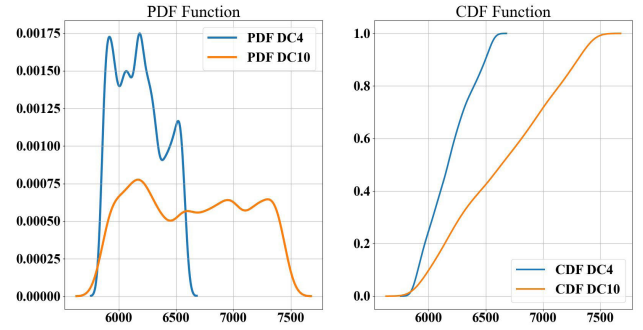


FIGURE 15. PDF and CDF of provisioned RAM and datacenter RAM capacity.

in DC production resources. The clustering of datacenter RAM capacity with respect to RAM provisioning is depicted in Figure 16. The classes here are very clear with scaling units (almost equal scale RAM resources units) that can participate in each configuration time window.

Another example that should be considered in clustering elastic attributes is DC and WL attributes correlation, which is non logical for elastic model correlation (because it reflects the datacenter actions not elastic scaling condition). Figure 18 depicts the RAM demands (WL4) with 125 bins to provisioned RAM (DC4) with 34 bins. The PDF and CDF distributions in Figure 19 show the demanded resources are for a longer duration, however the provisioning is done faster in a shorter period as CDF slope indicates. This causes a violation of the elastic feature (same scale of RAM resources units) that participate in each configuration time window. Labeling these kinds of relation (provisioned to demand) directly without elastic conditions check, will cause violation in elastic scaling and SLA as shown in Figure 20.

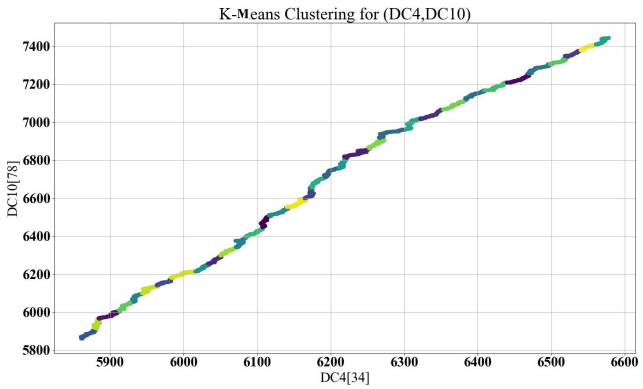


FIGURE 16. K-Means clustering of provisioned RAM and datacenter RAM capacity.

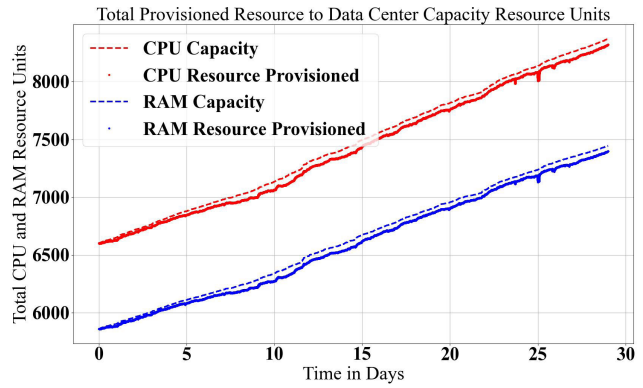


FIGURE 17. Datacenter CPU and RAM capacity, and CPU and RAM provisioned.

Elastic model works on matching the provisioned class to the demand class using simple (look-up) method while connecting the labeled demand class to the provisioned data center configuration. Mapping resources to demands in cloud resource orchestrator depends on appropriate linking between the demand workload class and the data center provisioned resource class. This is supposed to be independent from the pre-existing old actions (the map between demands and provisioned resources). The solution is to cluster scaling resources unit with respect to provisioned resources, then map them to demands. This will ensure a more effective cloud elastic manager's action. Results show the overlap between correlated attributes that indicates the relation between workload demand class types to its equivalent best matching class with the provisioned class.

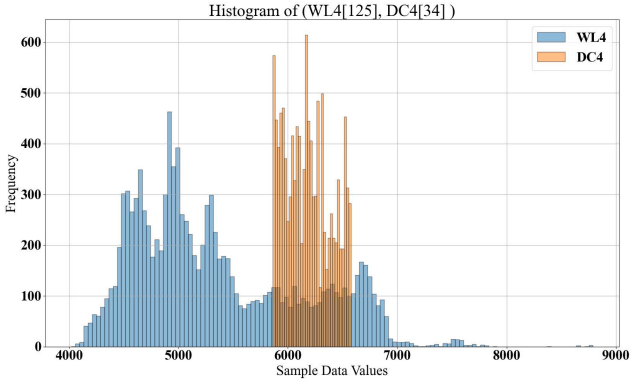


FIGURE 18. Histogram of demanded RAM and provisioned RAM.

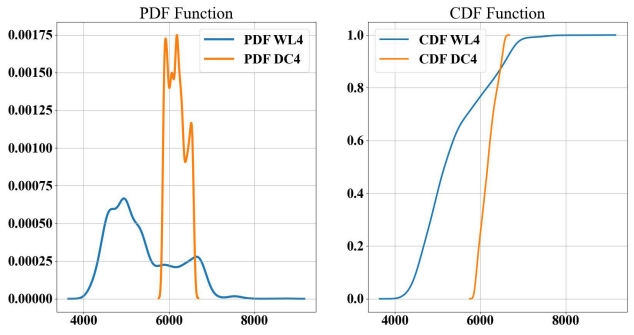


FIGURE 19. PDF and CDF of demanded RAM and provisioned RAM.

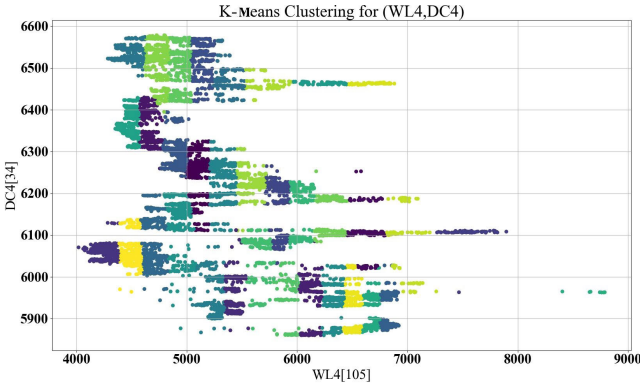


FIGURE 20. K-Means clustering of demanded RAM and provisioned RAM.

C. VALIDATION

We adopted three ways to validate clustering accuracy and consistency:

First, clustering accuracy is determined by running K-Means with changing k values. A statistical measurement has been applied that shows k values close to the extracted number of bins will produce the maximum Euclidean distance between cluster member nodes, with the minimum rate per bin. This can indicate a good relationship between bin cluster point members.

The second method is to test the cluster member consistency using the Silhouette method. In this validation method, K-Means clustering is evaluated in the range between $[-1, 1]$, where a high value means the member objects are matched in a good way. Figure 21 depicts the relationship between cluster members, which is not negative and is more than 0.6. This indicates a well-matched object to its own cluster. The third validation method uses Cumulative Distributed Function (CDF) slope. CDF $\hat{F}(X) = \sum_{i=0}^{M} \hat{inf}(x)$, which is defined as accumulating bin groups probability, can describe the behavior of each cloud log attribute population as a random variable. Using CDF slope, we can describe the attribute bin probability growth within the vector range. Close slope values between two attribute vectors can indicate a high relation between these two attributes. The slope of the CDF curve can indicate the speed of probability density distribution events. With large slope values, the probability intensity of the events becomes high, but with smaller slope values, the probability of the events, in that range, is rare and the probability intensity is lower. Using this method, we can indicate the highest range in the two attributes population that can be correlated and grouped by comparing with the clustering methods over population ranges. This shows an instantaneous view for specific events to check if there is a continuous bins relation. If continuous bins slope is equal, then a high relation between these two bins point members is found by CDF segmentation.

Applied on probability volume based on area integration using histogram bins as increment step, the points members in the same bin have very high relation. Also the points in neighbors' bins are related. Updating class boundaries of bin centroid groups relies on K-Means minimum distance inertia value between same bin group members and the total mean square error as shown in Equation 7. By combining related centroids (merging similar bin groups) will reduce M attribute vectors, and K-Means iteration cost by finding new sample mean and variance on new merged bins in the attributes vector instead of using random substitution. This way centroid update will be reduced because the mean and variance of bin distribution have high correlation based on KDE probability distribution.

The validation of this work is based on the aforementioned three methods, which can be applied on all attributes clustering combinations regardless if they are logically connected or not (for elastic condition). The first method calculates the maximum Euclidean Distance (ED) for all clustering group classes elements and the distance is compared with standard deviation error (SE). In all cases, we found $ED < SE$, which indicates good clusters are formed using the proposed methodology. In the second method, the Silhouette method is applied on all clustering cases achieving a reasonable correlation data consistency index (Silhouette score) of 0.4 value, as Figure 21 depicts. All Silhouette coefficients (for each class value) of clustering classes are passing the Silhouette score value indicating good cluster formations. Finally, the last method involves checking the normalized *MSE* as calculated using Equation 7. A value less than 0.70 of the maximum error ratio indicates a good

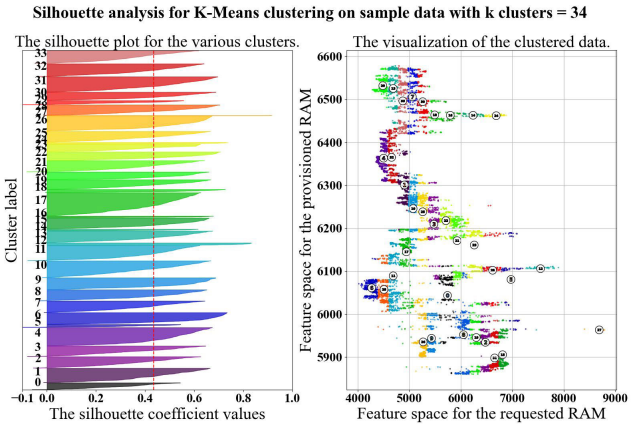


FIGURE 21. K-Means clustering evaluation using silhouette plot of demanded RAM and provisioned RAM.

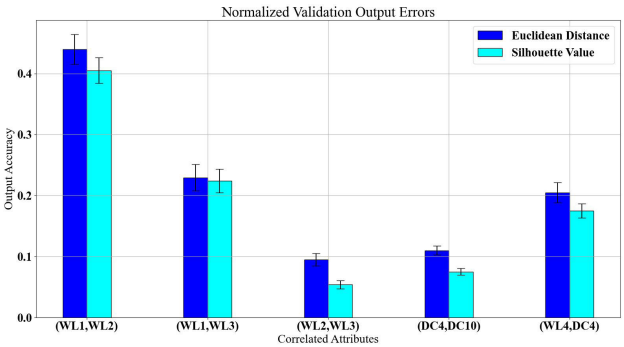


FIGURE 22. Confidence interval validation

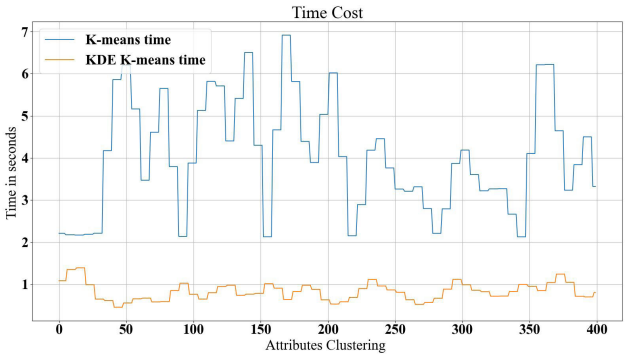


FIGURE 23. Time cost.

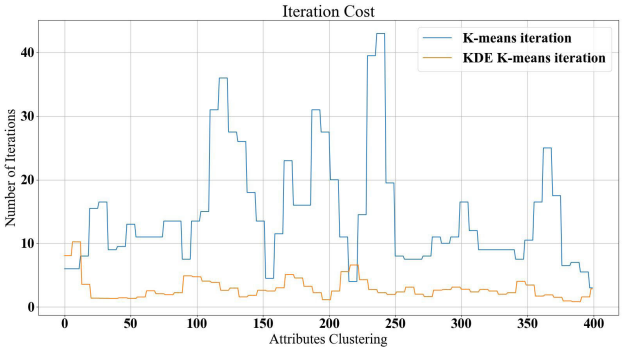


FIGURE 24. Iteration cost.

cluster. With these constraints we achieved a reliable dynamic clustering method, as Table 5 depicts. But in some cases, there are violations for absolute mean square error between some attributes. We found 24 such cases out of total 288 cases, which is about 8.34%. The accuracy of the cluster therefore can be considered as 91.66%, which is an acceptable value.

To test the behavior of the algorithms and their reliability, we run the experiment 10 times for each of the two attributes. Figure 22 depicts the variation (Confidence Interval) in Silhouette and Euclidean distance for each experiment. The outputs are normalized to the maximum value to make the figure more readable. Confidence range is represented in the graph as black middle bar in the output value. From the figure we can conclude the error range for each experiment does not exceed 0.093 in WL1 and WL3 cases, which is an acceptable value and shows that the algorithms work properly.

D. PERFORMANCE EVALUATION

The experiments have been applied with K-Means by selecting the number of clusters and centers dynamically, the number of maximum running iterations fixed to 300 as the default value. The time complexity achieved with the proposed method is reduced to be $O(1)$ rather than $O(k \times p)$. This is achieved because of the initialization of K-Means classes centroids and the way of updating them using variance guide, which reduces number of K-Means iterations. The execution time is significantly reduced, as shown in Figure 23. Using the proposed enhanced method by selecting the number of classes and centroids initialization using attributed statistical measurement instead of initializing them with random values, the number of iterations is reduced to a maximum of 10 iterations, as Figure 24 depicts. This is a significant reduction from standard K-Means method, where the maximum number of iterations is found to be 42. The validation methods for the proposed approach depends on bin number as time complexity, which is acceptable. The only challenge in validation is to calculate CDF, which uses integration method. However, it is not a major issue since the validation process can be done in offline mode.

VI. CONCLUSION

In this work, we have introduced a customized guided K-Means clustering method for cloud elastic model that depends on KDE and the Silverman method to find the initial centers and number of classes. The proposed approach can reduce K-means time complexity and enhance accuracy of the clustering. From our detailed analysis, a reduction of about 75% on average is obtained in execution time from regular K-Means algorithm and a 87.5% reduction is obtained in K-Means iterations. With this method, datacenter configuration and workload demands are mapped dynamically with adaptation to their characterization changes, which allows

the cloud management system to accommodate any type of workload and datacenter hardware configuration types. Our next work will be about classification of workload and DC configuration for elastic scaling module in cloud management system.

REFERENCES

[1] M. Peter and G. Tim, "The NIST definition of cloud computing," Comput. Secur. Division, Nat. Inf. Technol. Lab., Gurgaon, Haryana, Tech. Rep. MD 20899-8930, 2011.

[2] N. R. Herbst, S. Kounev, and R. Reussner, "Elasticity in cloud computing: What it is, and what it is not," in *Proc. 10th Int. Conf. Autonomic Comput. (ICAC)*, San Jose, CA, USA, 2013, pp. 23–27.

[3] G. Galante and L. C. E. D. Bona, "A survey on cloud computing elasticity," in *Proc. IEEE 5th Int. Conf. Utility Cloud Comput.*, Nov. 2012, pp. 263–270.

[4] P. C. Brebner, "Is your cloud elastic enough?: Performance modelling the elasticity of infrastructure as a service (IaaS) cloud applications," in *Proc. 3rd Joint WOSP/SIPEW Int. Conf. Perform. Eng.*, 2012, p. 263.

[5] C. Ferreira, C. Sousa, F. Rubens, R. Leal, G. Danielo, and J. Neuman, "Elasticity in cloud computing: A survey," in *Proc. Telecommun.*, 2015, pp. 289–309.

[6] *State of the Union: Amazon Compute Services*. AWS Compute Blog, Seattle, DC, USA, 2014. [Online]. Available: https://aws.amazon.com/blogs/compute/reinvent2014/

[7] (2009). *Open Compute Project*. [Online]. Available: https://www.opencompute.org/

[8] O. Bärring, M. Guerri, E. Bonfillou, L. Valsan, A. Grigore, V. Dore, A. Gentit, B. Clement, and A. Grossir, "Experience of public procurement of open compute servers," *J. Phys., Conf. Ser.*, vol. 664, no. 5, Dec. 2015, Art. no. 052004.

[9] K. Vaid. (2017). *Microsoft's Project Olympus Delivers Cloud Hardware Innovation at Scale*. [Online]. Available: https://azure.microsoft.com/en-ca/blog/microsofts-project-olympus-delivers-cloud-hardware-innovation-at-scale/

[10] L. A. Barroso, J. Dean, and U. Holzle, "Web search for a planet: The Google cluster architecture," *IEEE Micro*, vol. 23, no. 2, pp. 22–28, Dec. 2003.

[11] L. Cao, P. Sharma, S. Fahmy, and V. Saxena, "ENVI: Elastic resource flexing for Network function Virtualization," in *Proc. USENIX Workshop Hot Topics Cloud Comput.*, 2017, pp. 1–5.

[12] W. C. Arnold, D. J. Arroyo, W. Segmuller, M. Spreitzer, M. Steinder, and A. N. Tantawi, "Workload orchestration and optimization for software defined environments," *IBM J. Res. Develop.*, vol. 58, nos. 2–3, pp. 1–12, 2005.

[13] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces:format + schema, version 2.1," Google, Menlo Park, CA, USA, White Paper, 2011, pp. 1–14.

[14] T. Daradkeh, A. Agarwal, N. Goel, and J. Kozlowski, "Google traces analysis for deep machine learning cloud elastic model," in *Proc. Int. Conf. Smart Appl., Commun. Netw. (SmartNets)*, Dec. 2019, pp. 1–6.

[15] M. Alam, K. A. Shakil, and S. Sethi, "Analysis and clustering of workload in Google cluster trace based on resource usage," in *Proc. IEEE Intl Conf. Comput. Sci. Eng. (CSE)*, Aug. 2016, pp. 740–747.

[16] P. Minet, E. Renault, I. Khoufi, and S. Boumerdassi, "Analyzing traces from a Google data center," in *Proc. 14th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2018, p. 1167.

[17] J. Panneerselvam, L. Liu, J. Hardy, and N. Antonopoulos, "Analysis, modelling and characterisation of zombie servers in large-scale cloud datacentres," *IEEE Access*, vol. 5, pp. 15040–15054, 2011.

[18] M. Khelghatdoust, V. Gramoli, and D. Sun, "GLAP: Distributed dynamic workload consolidation through gossip-based learning," in *Proc. IEEE Int. Conf. Cluster Comput.*, Sep. 2016, pp. 80–89.

[19] Y. Yu, V. Jindal, I.-L. Yen, and F. Bastani, "Integrating clustering and learning for improved workload prediction in the cloud," in *Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2016, pp. 876–879.

[20] M. Rasheduzzaman, M. A. Islam, T. Islam, T. Hossain, and R. M. Rahman, "Task shape classification and workload characterization of Google cluster trace," in *Proc. IEEE Int. Advance Comput. Conf. (IACC)*, Feb. 2014, pp. 893–898.

[21] A. Jivrajani, D. Raghu, A. Kh, H. L. Phalachandra, and D. Sitaram, "Workload characterization and green scheduling on heterogeneous clusters," in *Proc. 22nd Annu. Int. Conf. Adv. Comput. Commun. (ADCOM)*, Sep. 2016, pp. 3–8.

[22] J. Patel, V. Jindal, I.-L. Yen, F. Bastani, J. Xu, and P. Garraghan, "Workload estimation for improving resource management decisions in the cloud," in *Proc. IEEE 12th Int. Symp. Auto. Decentralized Syst.*, Mar. 2015, pp. 25–32.

[23] T. Daradkeh, A. Agarwal, and Y. Alahmad, "Multiple attributes K-means clustering for elastic cloud model," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, Montreal, QC, Canada, Jun. 2020, pp. 100–105.

[24] W. Chen, K. Ye, Y. Wang, G. Xu, and C.-Z. Xu, "How does the workload look like in production cloud? Analysis and clustering of workloads on alibaba cluster trace," in *Proc. IEEE 24th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2018, pp. 102–109.

[25] Z. Ren, X. Xu, J. Wan, W. Shi, and M. Zhou, "Workload characterization on a production Hadoop cluster: A case study on taobao," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Nov. 2012, pp. 3–13.

[26] T. Daradkeh, A. Agarwal, N. Goel, and A. J. Kozlowski, "Elastic cloud logs traces, storing and replaying for deep machine learning," in *Proc. 3rd Int. Conf. Comput. Netw. Commun.* Trivandrum, India, 2019, pp. 585–590.

[27] H. Nguyen, Z. Shen, X. Gu, S. Subbiah, and J. Wilkes, "AGILE: Elastic distributed resource scaling for infrastructure-as-a-service," *Proc. 10th Int. Conf. Autonomic Comput.* Osaka, Japan, 2013, p. 69–82.

[28] W. M. A. Ahmed, S. A. Fomenkov, and S. V. Gaevoy, "Reducing approximation time of cluster workload by using simplified hypergamma distribution," in *Proc. Int. Conf. Ind. Eng., Appl. Manuf. (ICIEAM)*, 2018, pp. 1–5.cs.

[29] G. Da Costa, L. Grange, and I. Courchelle, "Modeling, classifying and generating large-scale Google-like workload," in *Proc. Sustain. Comput., Informat. Syst.*, 2018, pp. 2210–5379.

[30] N. Li and S.-Z. Yu, "Periodic hidden Markov model-based workload clustering and characterization," in *Proc. 8th IEEE Int. Conf. Comput. Inf. Technol.*, Jul. 2008, pp. 378–383.

[31] M. Ghobaei-Arani, S. Jabbehdari, and M. Pourmina, "An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach," *Future Gener. Comput. Syst.*, vol. 78, no. 1, pp. 191–210, 2018.

[32] A. Shahidinejad, M. Ghobaei-Arani, and M. Masdari, "Resource provisioning using workload clustering in cloud computing environment: A hybrid approach," *Cluster Comput.*, vol. 15, pp. 1–24, Apr. 2020.

[33] M. Ghobaei-Arani and A. Shahidinejad, "An efficient resource provisioning approach for analyzing cloud workloads: A Metaheuristic-based clustering approach," *J. Supercomput.*, vol. 23, pp. 1–40, Apr. 2020.

[34] K. Ye and C.-Z. Xu, "Reducing tail latency of interactive multi-tier workloads in the cloud," in *Proc. IEEE Int. Conf. Cluster Comput.*, Sep. 2018, pp. 162–163.

[35] S. Ismaeel and A. Miri, "A systematic cloud workload clustering technique in large scale data centers," in *Proc. IEEE World Congr. Services*, Jul. 2019, pp. 362–363.

[36] S. Ismaeel, A. Al-Khazraji, and A. Miri, "An efficient workload clustering framework for large-scale data centers," in *Proc. 8th Int. Conf. Modeling Simulation Appl. Optim. (ICMSAO)*, Apr. 2019, pp. 1–5.

[37] S. M. Aaqib, "An efficient cluster-based approach for evaluating vertical and horizontal scalability of Web servers using linear and non-linear workloads," in *Proc. 3rd Int. Conf. Trends Electron. Informat. (ICOEI)*, Apr. 2019, pp. 287–291.

[38] A. Xiao, Z. Lu, J. Li, J. Wu, and P. C. K. Hung, "SARA: Stably and quickly find optimal cloud configurations for heterogeneous big data workloads," *Appl. Soft Comput.*, vol. 85, Dec. 2019, Art. no. 105759.

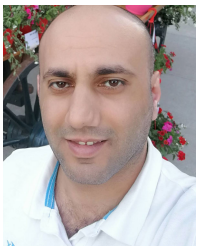
[39] B. Liu, J. Guo, C. Li, and Y. Luo, "Workload forecasting based elastic resource management in edge cloud," *Comput. Ind. Eng.*, vol. 139, Jan. 2020, Art. no. 106136.

[40] D. He, Z. Wang, and J. Liu, "A survey to predict the trend of AI-able server evolution in the cloud," *IEEE Access*, vol. 6, pp. 10591–10602, 2018.

[41] A. L. Garcia, "A cloud-based framework for machine learning workloads and applications," *IEEE Access*, vol. 8, pp. 18681–18692, 2020.

[42] Z. Chen, L. Luo, W. Quan, M. Wen, and C. Zhang, "Poster abstract: Deep learning workloads scheduling with reinforcement learning on GPU clusters," in *Proc. IEEE INFOCOM - IEEE Conf. Comput. Commun. Workshops*, Apr. 2019, pp. 1023–1024.

[43] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Atlanta, GA, USA: Chapman Hall, 1986.
[44] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*. Hoboken, NJ, USA: Wiley, 2003.
[45] E. Alpaydin, *Introduction to Machine Learning*, 2nd ed. Cambridge, MA, USA: MIT Press, 2010.
[46] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with Python," in *Proc. 9th Python Sci. Conf.*, 2010, pp. 1–5.
[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 7, pp. 2825–2830, Aug. 2011.
[48] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and E. Ga Varoquaux, "IAPI design for machine learning software: Experiences from the scikit-learn project," in *Proc. ECML PKDD Workshop, Lang. Data Mining Mach. Learn.*, 2013, pp. 108–122.
[49] J. VanderPlas. (2013). *Kernel Density Estimation in Python*. [Online]. Available: https://jakevdp.github.io/blog/2013/12/01/kernel-density-estimation/
[50] Z. I. Botev, J. F. Grotowski, and D. P. Kroese, "Kernel density estimation via diffusion," *Ann. Statist.*, vol. 38, no. 5, pp. 2916–2957, Oct. 2010, doi: 10.1214/10-AOS799.
[51] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, pp. 39501–39514, 2018.
[52] C. Robert, *Machine Learning, a Probabilistic Perspective*. New Delhi, India: Taylor & Francis, 2014.



TARIQ DARADKEH (Member, IEEE) received the bachelor's degree in computer engineering from Yarmouk University, Jordan, in 2006, and the master's degree in electrical and computer engineering in 2015. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Concordia University, Canada. His research interests include cloud computing networks and provisioning.

ANJALI AGARWAL (Senior Member, IEEE) received the B.E. degree in electronics and communication engineering from the Delhi College of Engineering, India, in 1983, the M.Sc. degree in electrical engineering from the University of Calgary, Alberta, in 1986, and the Ph.D. degree in electrical engineering from Concordia University, Montreal, in 1996. She is currently a Professor with the Department of Electrical and Computer Engineering, Concordia University. Prior to joining faculty in Concordia, she worked as a Lecturer with IIT Roorkee, and as a Protocol Design Engineer and a Software Engineer in industry. Her current research interests include cloud networks, heterogeneous networks, and wireless networks, including security and virtualization of cognitive radio networks.

MARZIA ZAMAN received the M.Sc. and Ph.D. degrees in electrical and computer engineering from the Memorial University of Newfoundland, Canada, in 1993 and 1996, respectively. She started her career at the Software Engineering Analysis Laboratory, Nortel Networks, Ottawa, ON, Canada, in 1996, where she later joined the OPTera Packet Core Project as a Software Developer. She has many years of industry experience as a researcher and a software designer at Accelight Networks, Excelocity, Sanstream Technology, and Cistel Technology Inc. Since 2009, she has been with the Center for Energy and Power Electronics Research, Queen's University, Canada, and one of its industry collaborators, Cistel Technology Inc., where she is working on multiple research and development projects. Her research interests include renewable energy, wireless communication and networks, machine learning, and software engineering.

NISHITH GOEL received the degree in Jodhpur, India, and the M.A.Sc. degree in electrical engineering and the Ph.D. degree in systems design engineering from the University of Waterloo. He began his professional career at Bell-Northern Research, Ottawa, in 1984, and then he moved on to Northern Telecom, in 1988. He is currently the CEO of Cistel Technology Inc., an information technology company he founded in 1995, which has operations in Canada, and the USA. He is also a Co-Founder of CHiL Semiconductor, iPine Networks, and Sparq Systems. He is also the Chair of the Queen's Center for Energy and Power Electronics Research, Queen's University. He serves on various corporate boards of directors. His research interests include information technology especially in the area of wireless networks, cloud computing, and network security.