Adaptive Prediction Models for Data Center Resources Utilization Estimation

Shuja-ur-Rehman Baig, Waheed Iqbal, Josep Lluis Berral, Abdelkarim Erradi, and David Carrera

***Abstract*—Accurate estimation of data center resource utilization is a challenging task due to multi-tenant co-hosted applications having dynamic and time-varying workloads. Accurate estimation of future resources utilization helps in better job scheduling, workload placement, capacity planning, proactive auto-scaling, and load balancing. The inaccurate estimation leads to either under or over-provisioning of data center resources. Most existing estimation methods are based on a single model that often does not appropriately estimate different workload scenarios. To address these problems, we propose a novel method to adaptively and automatically identify the most appropriate model to accurately estimate data center resources utilization. The proposed approach trains a classifier based on statistical features of historical resources usage to decide the appropriate prediction model to use for given resource utilization observations collected during a specific time interval. We evaluated our approach on real datasets and compared the results with multiple baseline methods. The experimental evaluation shows that the proposed approach outperforms the state-of-the-art approaches and delivers 6% to 27% improved resource utilization estimation accuracy compared to baseline methods.**

***Index Terms*—Data center, resource management, data classification, modeling and prediction, dynamic prediction model, feature extraction.**

I. Introduction

TECHNOLOGICAL advances in server virtualization and cloud computing allow cost-effective hosting of multiple applications in a secure, customizable, and isolated computing environment managed by modern data centers. This yields higher resources utilization with reduced costs.

Manuscript received December 2, 2018; revised March 28, 2019 and July 13, 2019; accepted July 27, 2019. Date of publication August 2, 2019; date of current version December 10, 2019. This work is partially supported by the European Research Council (ERC) under the EU Horizon 2020 programme (GA 639595), the Spanish Ministry of Economy, Industry and Competitiveness (TIN2015-65316-P and IJCI2016-27485), the Generalitat de Catalunya (2014-SGR-1051), and NPRP grant # NPRP9-224-1-049 from the Qatar National Research Fund (a member of Qatar Foundation) and University of the Punjab, Pakistan. The statements made herein are solely the responsibility of the authors. The associate editor coordinating the review of this article and approving it for publication was Y. Diao. *(Corresponding author: Shuja-ur-Rehman Baig.)*

S.-R. Baig, J. L. Berral, and D. Carrera are with the Barcelona Supercomputing Center, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain (e-mail: shuja.baig@bsc.es; josep.berral@bsc.es; david.carrera@bsc.es).

W. Iqbal is with the Punjab University College of Information Technology, University of the Punjab, Lahore 54590, Pakistan (e-mail: waheed.iqbal@pucit.edu.pk).

A. Erradi is with the Department of Computer Science and Engineering, College of Engineering, Qatar University, Doha, Qatar (e-mail: erradi@qu.edu.qa).

Digital Object Identifier 10.1109/TNSM.2019.2932840

Additionally, cloud consumers can acquire compute, storage and networking resources on-demand from Infrastructure-as-a-Service (IaaS) providers on pay-per-use basis. IaaS users can control the leased resources and scale them to optimize their usage accordingly to their needs. To ensure better Quality of Service, IaaS providers distribute data center resources across multiple geographical locations to enhance proximity to the application users. Also, virtualization and holistic data center management enable providers to maximize the data center utilization while minimizing their operational cost [1].

Efficient methods for estimating resource utilization in data centers can significantly ease self-management and usage optimization for both users and providers. Users can dynamically adjust the leased resources to minimize costs for hosting their applications while maintaining the desired performance and service quality [2]. Further, accurate estimates of resources utilization enable the providers to efficiently allocate virtual machines (VM) and other virtual resources to workloads, migrate VMs to consolidate or balance resource usage [3], [4], plan in advance resource capacities [5], [6], also take awareness of energy requirements in advance for expected workloads and users [7], [8].

Accurate estimation of future resources utilization for data centers is challenging due multi-tenant co-hosted applications having dynamic and time-varying workloads. While there are several estimation methods for cloud resource utilization using time-series learning or deep-learning networks [9], [10], [11], all use a single model that often does not accurately capture the workload dynamics. To address these problems, we propose a novel method to adaptively and automatically identify the most appropriate model to accurately estimate data center resources utilization. Our adaptive multi-methods approach considers different scenarios encountered in a production data center and enables selecting the the predictive method that learns best. Our approach focuses on training estimation models using different methods then selecting the one that will yield the best prediction given the current scenario and the previous batch of collected data.

To test and validate our selective multi-method approach, we have conducted experiments using Alibaba [12] and Bitbrains [13] data center utilization datasets. We compared the results of our experiments with exiting baseline approaches that use a single model such as Linear Regression (LR), Support Vector Machines (SVM), Gradient Boosting Tree (GBT) and Gaussian Process also known as *Krigin* (KR). Figure 1 shows the motivation to adaptively select an appropriate method to effectively estimate resource

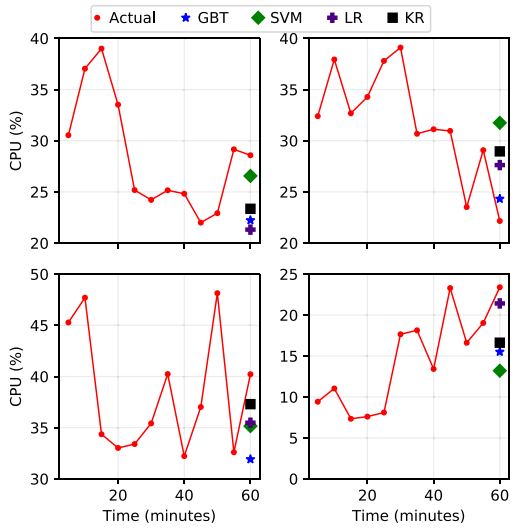

Fig. 1. CPU estimation using different methods and scenarios for Alibaba data set. Different predictors yield better estimation, each for different scenarios.

utilization for different scenarios. The figure shows CPU utilization estimations using different methods for four different machines from Alibaba data set. Each estimation method is trained using 55 minutes time interval data and estimated the utilization for the next 5 minutes. We observed that different predictors yield better resource utilization estimation for different scenarios. Therefore, it adds value to build a system to identify the best predictor for forecasting resource utilization at each time interval.

In this paper we focused on classical machine learning approach and did not use deep learning as the learning process is considered as black-box [14] and to understand the reasoning of the model's prediction behavior is not apparent. Deep learning works well with a large amount of high dimensional data, and it is also extremely computational expensive due to which, it requires a specialized type of hardware. The resource utilization of data centers is a low dimensional data, and traditional machine learning methods can be effectively used for estimations. Moreover, deep learning performs quite well once trained for a particular problem; however, model miserably fails when applying on a similar type of other problems and required to retrain. Due to these reasons, we selected the traditional machine learning approach and propose a novel *adaptive model selector* method, to dynamically identify the best prediction method for estimating resource utilization of data centers, from a bag of trained methods with different characteristics and accuracy over different data center behaviors. The data center telemetry contains burstiness behavior which represents sudden spikes and peaks of resource utilization. In general, it is challenging to predict the burstiness behavior, and we address this issue with the help of an adaptive selection of an appropriate prediction method at every estimation step. After some experiments and model selection, we chose Random Decision Forests (RDF) as the best mechanism for learning the expected accuracy for each candidate predictor.

Our proposed method trains on the statistical features of historical resource utilization and predictor correctness for sliding windows of a specific size, to identify which predictor will produce the best forecast given the current resource utilization. We evaluate our method by comparing its decision and forecasting capabilities with baseline methods, using datasets from Alibaba and Bitbrains monitored data centers. Results show that the proposed method outperforms the baseline methods for both of the datasets. Notice that in this work we focus on CPU resource consumption as the primary resource on high-performance computing data centers, but our solution is generic and can be used to predict utilization of all system resources. The main contributions of this work are:

- A novel method to dynamically select the best prediction model for estimating and forecasting cloud resource utilization for a given recent time window of observed resource utilization.
- Use RDF for choosing an appropriate prediction model to be used for estimating data center resource utilization.
- A comparison of different baseline models, currently used in the state-of-the-art, as candidate models for resource utilization estimation, aside of validation for the presented approach.
- Analyze the impact of different window sizes on the proposed resource estimation systems.

The rest of the paper is organized as follows. Related work is presented in Section II. Our proposed resource estimation system is explained in Section III. Prediction methods and Adaptive Model Selector (AMS) are explained in Section IV. Feature extraction and selection is discussed in Section V. We provide details about the experimental evaluation in Section VI. The experimental results are presented in Section VII. Finally, conclusion and future work are discussed in Section VIII.

II. Related Work

Data center resource utilization and workload prediction is an active research area. Recently, there have been several attempts to use machine learning methods for predictions of data center resources. For example, recent work by Kim *et al.* [15] proposed an ensemble approach which uses multiple predictors together to produce an output. The proposed ensemble technique uses Linear Regression, SVM, ARMA, and ARIMA together to predict future workload for the data centers by dynamically determining the weight of each predictor using the regression method. Another recent work by Rahmanian *et al.* [16] also proposed an ensemble-based approach to predict CPU utilization of application usage of VMs. The proposed approach uses automata theory to adjust the weight for each predictor in the ensemble method to predict the CPU usage. Subirats and Guitart [17] proposed an ensemble-based prediction strategy which forecasts the infrastructure energy requirement by predicting the future CPU utilization of VMs. Their ensemble-based approach uses the moving average, exponential smoothing, linear regression, and double exponential smoothing methods. Chen *et al.* [18] propose an ensemble model based on the fuzzy neural network to predict the resource demand. They use the second moving average (SMA), exponential moving method (EMA), autoregression model (ARM), and trend seasonality

model (TSM) as base predictors. Cetinski and Juric [19] combine statistical and machine learning methods to predict application specific workload volume. Tseng *et al.* [20] used a multi-objective genetic algorithm to forecast resource utilization and energy consumption in data centers. Jiang *et al.* [21] proposed ensemble prediction mechanism to predict the cloud workloads for capacity planning in data centers. They used five prediction algorithms named as moving average, autoregression, artificial neural network, support vector machine, and gene expression programming to predict the future workload estimations.

There have been several efforts to use typical time series solutions to predict data center resource utilization. For example, Calheiros *et al.* [22] used autoregressive integrated moving average (ARIMA) method to predict the arrival rate for the applications hosted on the cloud. Liao *et al.* [23] use typical time series prediction methods namely autoregressive moving-average, moving average, and auto-regressive together as an ensemble approach to predict CPU usage of VMs. The proposed method combines the output of time series prediction techniques as input to another linear prediction model to predict CPU utilization of VMs. Vazquez *et al.* [24] used various time series prediction models to forecast the number of requests which helps in the dynamic scaling of cloud resources proactively. For this purpose, they evaluated the autoregressive model (AR), moving average model (MA), simple exponential smoothing, double exponential smoothing, automated ARIMA method, and neural network autoregression method. Dmytro *et al.* [25] use ARIMA to forecast load on the cluster which helps in scheduling the data center resources by migrating the VMs. Fang *et al.* [26] used ARIMA to predict the future CPU utilization and several requests for the applications hosted in the cloud.

There have been several efforts to employ deep learning methods for predicting data center resource utilization. For example, Zhang *et al.* [9] use autoencoders to predict the CPU utilization of VMs. The authors used tensor rank decomposition technique to reduce the training time by compressing the input parameters. Qiu *et al.* [27] used a deep belief network using multiple-layered restricted Boltzmann machines (RBMs) and a regression layer to predict the CPU usage of VMs. The RBMs are used to extract high-level features, and the regression layer is used to predict CPU utilization. Zhang *et al.* [11] also use RBMs to predict CPU and RAM utilization in data centers. They use backpropagation as global supervised learning to minimize the loss function. Mason [10] predict the CPU consumption of the host by using evolutionary Neural Networks (NN). To train the network weights of neural networks, they used Particle Swarm Optimization (PSO), Differential Evolution (DE), and Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES). Song *et al.* [28] use long short-term memory (LSTM) model to predict the host load. To train the recurrent networks, the authors used truncated back-propagation through time technique. Duggan *et al.* [29] predict host CPU utilization by using Recurrent Neural networks. They also use the back-propagation through time (BPTT) technique to train the network.

Recurrent Neural Networks are a hot topic on many modeling scenarios, including resource management for Data-Centers. However, RNNs imply a set of trade-offs to have into consideration on scenarios where data-streams must be constantly modeled or evaluate. Moreover, the selection of hyper-parameters for RNNs and their different methods (LSTMs, GRUs, etc.) imply extra decisions to be searched and tuned, while simpler methods can provide similar accuracy (or lower but good-enough accuracy) with less computational and human-tuning effort. One of the problems of time-series algorithms like RNNs, time-series related NNs like CRBMs, or filters like period-adaptive-Kalman, is that most rely on a delay or memory hyper-parameter on their design, on a scenario where behavior regimes and their length may not be known a-priori. Not to say that interpretability is a requirement on knowledge discovery, to help data-center architects to improve the DC infrastructure. In conclusion, it is true that advances have been done on RNNs towards DC management, but in this work, we advocate for more simpler (in terms of training and operability) and more readable models.

The work in this area most relevant to ours [30] adaptively picks either Regression (LR) or Support Vector Machine (SVM) predictors to estimate CPU utilization of VMs. The proposed method dynamically select LR for slow changing workloads and SVM for rapidly changing workloads. Moreover, most of the existing works use ensemble-based approaches in which multiple estimation methods are collectively used to produce the final output whereas in our proposed solution the final output is produced using only a single machine learning predictor which is dynamically identified using the recent resource utilization observations. Our approach uses four different estimators and dynamically identifies the estimator using a machine learning approach and time series features. To the best of our knowledge, no existing work which uses time series features to adaptively identify and use the best prediction method to minimize the estimated error of cloud resource utilization. Table I presents the comparison and explain how proposed solution is different from existing state of the art work.

III. Proposed System Overview

The overall proposed system is illustrated in Figure 2. Different steps are numbered and labeled to explain the working flow of the system. The system work in the following steps:

- Historical resource utilization logs of the data center are divided into sliding windows of a fixed size consists of the last k intervals. Then each sliding window data is used to fit different prediction models including Linear Regression (LR), Support Vector Machine (SVM), Kriging (KR), and Gradient Boosting Tree (GBT) to predict the next interval resource utilization. The system selects the prediction method yields a minimum prediction error for the given sliding window data.
- For each sliding window, the system identifies a specific set of features as explained in Section V.

TABLE I
COMPARISON OF RELATED WORK WITH PROPOSED SOLUTION

Reference	Observed Metric	Feature Based Classification	Adaptivly Predictor Selection	Ensemble	Methodology
Kim [15]	job arrival rate/time	No	No	Yes	Four predictors collectively determine the output of future workload by assigning weight to each predictor using multiclass regression.
Rahmanian [16]	cpu	No	No	Yes	Predicts the future by combining the prediction values of all constituent models and use automata theory to adjust the weight of each predictor.
Subirats [17]	cpu	No	No	Yes	Selects the forecast provided by the predictor with the smallest mean average absolute error using four predictors (MA, ES, LR, DES) which is calculated for every time interval.
Chen [18]	network traffic	No	No	Yes	Use fuzzy neural networks to predict the future value which takes input from multiple base predictors as well as raw input.
Tseng [20]	cpu,memory, energy	No	No	No	Use a multi-objective genetic algorithm to forecast resource utilization and energy consumption.
Jiang [21]	vm request	No	No	Yes	Use five predictions methods to predict future value. It uses relate error between actual and predicted value is used to update the weights of each base predictor.
Rodrigo [22]	arrival rate	No	No	No	Use ARIMA to predict the future data point.
Liao [23]	cpu	No	No	Yes	Use prediction of multiple time series predictors as an input to another linear prediction model to predict the final future value.
Vazquez [24]	number of requests	No	No	No	Use various time series prediction models independently to forecast the number of requests and compare the results.
Dmytro [25]	vm migration	No	No	No	Use ARIMA to predict the future data point.
Liu [30]	cpu	No	Yes	No	Use either LM or SVM according to slow or fast changing workload.
Zhang [9]	cpu	No	No	No	Use autoencoders to predict the future CPU utilization of VMs.
Alex [31]	-	Yes	No	No	Only provide the time series classification based on feature vector.
Fulcher [32]	-	Yes	No	No	Only provide time series classification based on feature vector.
Proposed	cpu	Yes	Yes	No	Forecast the future resource utilization using the best predictor obtained from the classifier trained on the time series features. The best predictor is identified adaptively at every estimation time step.

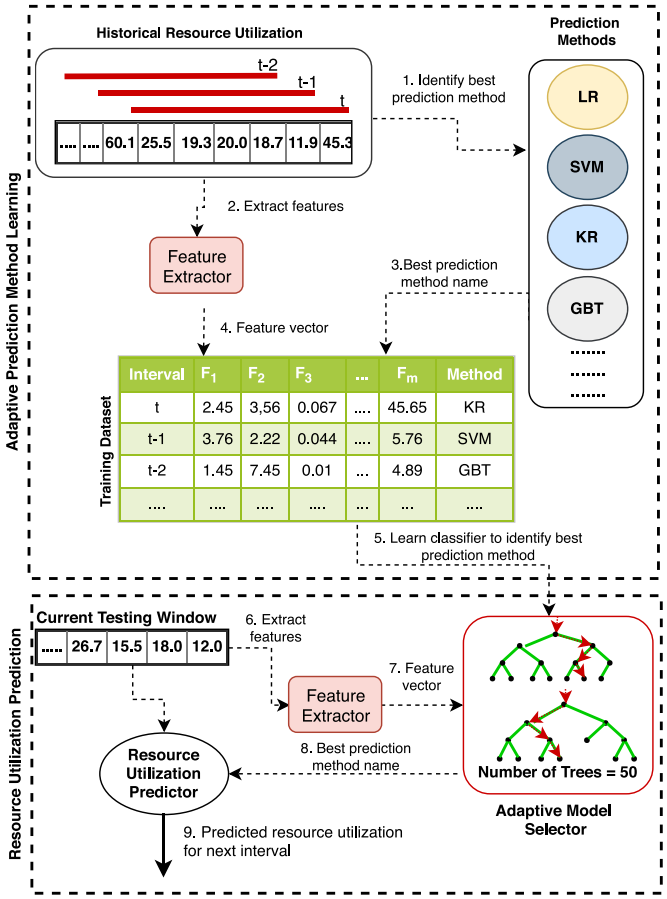

Fig. 2. Purposed system overview to learn adaptive model selector and using it to estimate the data center resource utilization.

- The selected features and identified prediction methods are logged as training data. For each historical sliding window, the training data set contains the corresponding feature vector and the best prediction method.
- Once training data is prepared, the system builds a classifier using Random Decision Forest (RDF) to predict the best model for a given sliding window data. We call this classifier "Adaptive Model Selector". We explain this in Section IV-B.
- Once the Adaptive Model Selector is trained than the system predicts the data center resource utilization in real time. For the current time interval t, system select last k observation to extract features and then use the Adaptive Model Selector to identify the best prediction method to predict the resource utilization for the $t+1$ time interval.
- The selected prediction method is used to train a regression model using the last k interval's observed resource usage data to estimate the resource utilization for the $t+1$ future time interval.

IV. MACHINE LEARNING METHODS

In this work we use Machine Learning (ML) techniques for two main purposes: first, predict future workload behaviors and traces; second, from a set of ML methods and a context, choose one that predicts the workload better. The ensemble presented here focuses on different algorithms for regression used to predict the workload, while a trained decision maker selects at each time a regression model that is expected to produce the most accurate prediction.

In this section, we introduce the different algorithms used for the prediction and decision-making processes.

A. Workload Prediction Methods

To predict workload, we explore a diversity of Machine Learning techniques commonly used in the literature, ones more complex than others with different properties each. The learned regression models are to predict our target variable which is next data point in time series from known input

features [33] such as skewness, standard deviation, kurtosis, autocorrelation for different lags, absolute sum of changes, etc. As the data we are dealing is in the form of a time series, evaluation of prediction must be based not just on accuracy but also on the significance of results which is often a difficult problem on regression analysis.

Our presented methodology shows a multi-model approach, where different models are trained, each one with a different set of strong and weak properties. The models are applied to a dynamic window to predict future interval workloads. The studied models for workload prediction are: Linear Regression, Support Vector Machines for regression, Gradient Boosting, and Gaussian Process Regression.

Linear Regression: Linear Regression (LR) is one of the simplest but effective approaches in machine learning modeling and prediction specifically when a linear relation exists among variables. LR assumes there is a linear relation between output variable Y and input variables $X = \{x_1 \dots x_n\}$, and attempts to find a vector $W^T = \{w_1 \dots w_n\}$ and a scalar b where $\tilde{Y} = X \cdot W + b$ while minimizing the error $\epsilon = |Y - \tilde{Y}|$. Minimization is usually performed using the Least Squares Error approach, although other approaches using the deviation or specific cost function exist. LR variants include Polynomial and Multinomial Regression, where variable relations are assumed more complex, thus learning algorithms also become more complex.

Support Vector Regression: Support Vector Machine (SVM) methods are common for classification although they can be used for regression as Support Vector Regression machines (SVR) [34]. The advantage of SVMs is that non-linear functions can be learned as linear ones thanks to a transformation of data known as the *kernel trick*.

SVMs allow learning non-linear functions by mapping them into a higher dimensional feature space, using a defined kernel function. Input X are mapped into an h-dimensional feature space using a predefined non-linear kernel function to produce a linear model. Similar to LR, we can express SVMs as $\tilde{Y} = k(X) \cdot W + b$, where k is the function making the space for X linear. SMVs error minimization consists on building two margin functions (support vectors) $X \cdot W + b \pm \epsilon$, where final error ξ is computed for those elements outside the margins. As a disadvantage, margin ϵ can become an hyper-parameter.

Gradient Boosting: Gradient Boosting is the combination of the Gradient Descent optimization and Boosting techniques [35], [36]. As any other boosting technique, the learned model is the composition of weaker models focusing on subsets of data, forming a stronger model when combined. Usually, decision and regression trees are used on Gradient Boosting techniques, but any other modeling technique can be used for boosting.

On Gradient Boosting, a model is fitted as $\tilde{Y} = f(X)$ minimizing $\epsilon = |Y - \tilde{Y}|$. Then function f can be fine-grain tuned using another function h fitted to ϵ, learning and correcting the errors on the first function, and so on recursively. This recursion can continue until we rest satisfied with the resulting aggregation of models.

Gaussian Process Regression: Gaussian Process Regression (also known as *Kriging*) [37] is a non-parametric regression method, where the modeled function is trained after a Gaussian process using the covariances of previous examples. This process is used mainly for interpolation which requires some example observation points. Kriging method predicts by computing the weighted average of the values for neighbors from the known examples. Kriging models can model non-linear as well as linear behavior. Typical regression methods are extended by statistical models based on stochastic processes. However, Kriging also estimates the associated statistical variations using the distribution and correlation of observed data. Recently, Kriging is used for self-adaptive provisioning of resources in cloud-hosted applications [38].

B. Adaptive Model Selector (AMS)

On multi-model methodologies, different regression models produce predictions altogether, and a trained expert system decides which prediction is followed, or how they are aggregated into a final prediction. Such a trained expert can be a machine learning model, like in Boosting methods. In our proposed solution, before producing workload predictions, we use a trained decision maker to choose the best predictor to be used. The decision maker will classify each scenario into the best-expected predictor for it.

Our decision maker input will be features [33] such as skewness, standard deviation, kurtosis, autocorrelation for different lags,the absolute sum of changes, etc., and it will output the regression method which is expected to be the best. At each time step, the decision maker predicts the best regression model and then produces the workload prediction using the predicted regression model. Here we present the different classification models studied in this work.

K-Nearest Neighbors: The k-Nearest Neighbors (k-NN) algorithm allows to memorize a set of characteristic examples, and classify new data instances by finding the *k nearest neighbors*, and returning the class of the majority (or the probabilities per class on those k examples). The nearest neighbors are those examples with minimum distance, often euclidean, Hamming or Manhattan distances. Here we select k-NN as one of the tentative classifiers, as it is one of the easier models to train (it memorizes the training set), in exchange of the not-so-easy search process when predicting a new data instance.

Naïve Bayes: The Naïve Bayes algorithm is a classifier based on computing the likelihood of a feature given each class, then use the Bayes theorem to compute the conditional probability of a class given that feature. The method extracts from data the probabilities of each feature value $P(Feature = X)$, each class $P(Class = C)$, and each likelihood of features per class $P(Feature = X \mid Class = C)$. This method assumes independence among features, in contrast to Bayesian Networks. The probabilities per class are the product of their probabilities per feature, and the algorithm returns the class with a higher probability (or the rank of classes per probability).

We selected Naïve Bayes as one of the classifiers for its low complexity, as training implies keeping the count of element

occurrences, then probabilities can be computed on demand at prediction time.

Multilayer Perceptron: Multilayer Perceptron (MLP) is a kind of Artificial Neural Network (ANN) used for both classification and regression problems for non-linear systems. The most commonly used ANN for classification problem is "one-hidden layer" Feed-Forward ANN, where the ANN is composed of a single layer of perceptrons (neuron units) and an output layer.

Data passes through the hidden layer to the output producing a value for each class, then the class with a higher value is chosen. Neurons aggregate input data, usually through a linear function $X_o = X_i \cdot W + bias$, then passes outputs X_o to the next layer (here the output layer). Output neurons also pass their produced aggregation through sigmoid functions to approximate their outputs to 0 or 1 $Y = sigm(X_o)$. Fitting those functions is done by passing data repeatedly and comparing the network output with the real output, then updating neurons weights *W* and *bias* using Gradient Descent techniques.

Neural networks can be complicated to fine-tune, as their architecture must be treated as a hyper-parameter, deciding how many neuron units are in the hidden layer, how many times data must be passed for training, etc. We used Keras [39] sequential model to implement MLP for classification. We evaluated MLP with different number of hidden layers and found that with three hidden layers, it yields the best results.

Since we are using four machine learning predictors for evaluation purposes, the output layer contains 4 neurons. We use "relu" as the activation function for hidden layers and "softmax" for output layer. We use "adam" as optimizer and "categorical crossentropy" as loss function. The total epochs used are 1000 with batch size equal to 2000.

Random Decision Forest: Random Decision Forests (usually referred as Random Forests) are an ensemble method for classification and regression, based on the aggregation of specialized decision trees [40]. The ensemble builds a set of decision trees, trained from different data subsets, then predicted data is classified as the most voted class from all decision trees (the trend). The main reason to use random forests is to prevent over-fitting single decision tree models and get a more accurate and stable prediction.

Random Forests are known to produce decent results for classification and regression problems, without the need for much tuning or hyper-parameters. For our experiment, we tune the number of trees and set it to 50.

Gradient Boosting: Gradient Boosting can also be used for both regression and classification problems. For methods where the boosted algorithm is already a classification problem, the most voted class from all partial models is selected. For regression boosted algorithms, we can turn outputs into binary values using similar approaches like in SVMs, considering each value as a class and its value between -1 and 1 as its scoring.

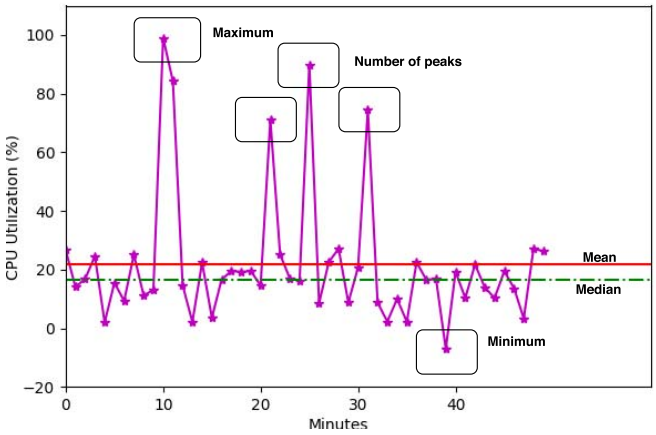

Fig. 3. Example of time series features that are extracted from TSFRESH [41] library. These features consist of statistical and time series features such as minimum, maximum, variance, standard deviation, number of peaks, autocorrelation at different lag intervals, entropy, kurtosis, skewness, fourier transformation, mexican hat wavelet transformation, and etc.

V. Feature Extraction and Selection

Appropriate features can play an important role to improve the prediction accuracy of machine learning models. In our data set, the resource utilization of data centers is available as a time series data. We explore multiple ways to extract time series features from the given data set which includes manual extraction, automatically extraction by the help of open source libraries such as Cesium [42], TSFRESH [41]. However we selected TSFRESH as it provides us most useful and a comprehensive set of time series features which is not available in any other library. Time Series Feature extraction based on scalable hypothesis tests (TSFRESH) [41], [43] is an open-source Python library available to extract features for a given time series data. In our proposed system, we used TSFRESH to extract features for data center resource utilization data available as time series. TSFRESH automatically calculates a large number of time series characteristics based on scalable hypothesis tests.

Figure 3 shows some of the features that TSFRESH extracts for the given time series data. It provides hundreds of statistical and time series features including minimum, maximum, variance, mean, standard deviation, sum of values, autocorrelation of the specified lags, measure of non linearity in the time series, Mexican hat wavelet, first and last location of minimum and maximum, number of peaks, quantile, and sample entropy etc. However all of these features are not necessary, and appropriate features should be identified to improve the performance of machine learning methods [44], [45].

The proposed system filters the features obtained from TSFRESH using another open-source library available for feature selection [46]. We selected this library because it includes a comprehensive set of functions to filter the features by using different approaches for identifying the most appropriate features for time series classification. The library provides five different methods to filter features for `missing values`, `single unique`

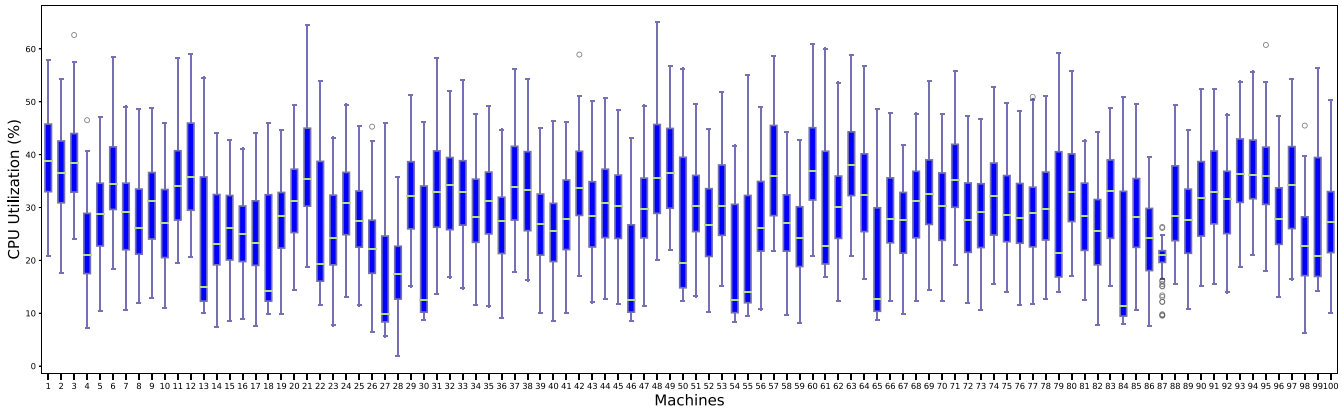

Fig. 4. Box plot of CPU utilization for randomly selected 100 machines from Alibaba data set.

`values`, `collinear features`, `zero importance features`, and `low importance features`. However, in our proposed system, we only used three methods to filter the features obtained through TSFRESH. First, we apply three methods to filter the feature. First, we apply `single unique value` method which remove the features with identical unique values. Second, we apply `identify collinear` which remove the features which are highly correlated with one another. We used 98% correlated threshold in this method to ensure only remove the features which correlated 98%. Finally, we apply `zero importance features` which uses Gradient Boosting Machine (GBM) learning model to identify the features which have zero importance for the given set of features. After applying these methods, we obtain one hundred and six features in total which include standard deviation, kurtosis, skewness, absolute some of the changes, auto-correlation at different lags, partial auto-correlation at different lags, the first location of minimum, linear least-squares regression [47], and many others.

VI. Experimental Evaluation

In this section, we explain the datasets used to evaluate our proposed method, the details about the experiments used to validate it, the baseline methods used for comparison, and the used evaluation metrics.

A. Datasets

1) Alibaba Data Set: The first data set we use is the *Alibaba cluster logs* [12], publicly available, containing performance traces of 1,313 machines for 12 hours duration. The Alibaba monitored cluster provides interactive services and batch processing workloads. The metrics represented are CPU, memory and disk utilization for all machines, aggregated on 5-minute averages. For simplification purposes, we are focusing and experimenting with CPU time series. The average CPU utilization in the Alibaba data set is 26.46%, with a standard deviation of 10.66% CPU. Figure 4 shows a CPU utilization sample for 100 randomly selected machines from the data set.

2) BitBrains Data Set: The second data set we use is the *Bitbrains data set* [13], publicly accessible, containing performance logs of 1,750 VMs for 30 days of data. The Bitbrain monitored cluster provides interactive services and batch processing workloads. The metrics represented are CPU, memory, network and disk utilization for all the virtual machines, aggregated on 5-minute averages. From this data set, we randomly selected 20 VMs with average CPU utilization greater than 30%, as most of the VMs with low usage do not show critical metric patterns or utilization tends to be constant on the lowest part of the spectrum demand. Figure 5 shows the box-plot for one-day data of the average CPU utilization for the selected machines.

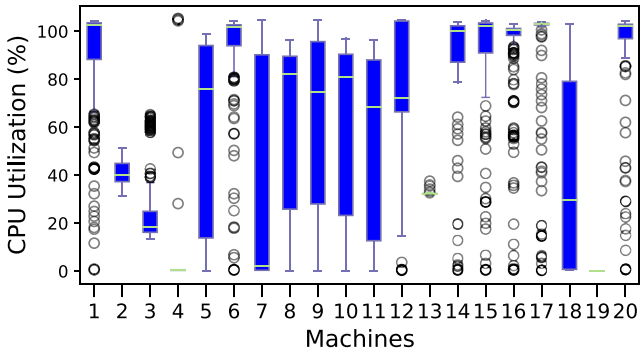

Fig. 5. Box plot for Bitbrain data set of 20 randomly selected VMs for one-day data.

3) Google Data Set: The Google cluster traces [48] are the publicly available traces published by Google. To create the CPU and the Memory utilization, the tasks of each job were aggregated by summing their CPU and Memory consumption every five minutes in a period of 24 hours. The dataset was extracted over the first ten days period by filtering the utilization of CPU and memory from 5 to 90 percent, resulting in a total of 1,600 VMs [49]. We randomly selected 500 VMs from this data set for the experiments and the average CPU utilization in the selected data set is 21.89%, with a standard deviation of 3.63% CPU.

B. Methodology

For the current experiments, we are using the Alibaba data set to show a comprehensive evaluation of the proposed solution. Whereas, the BitBrains data set is used for testing to

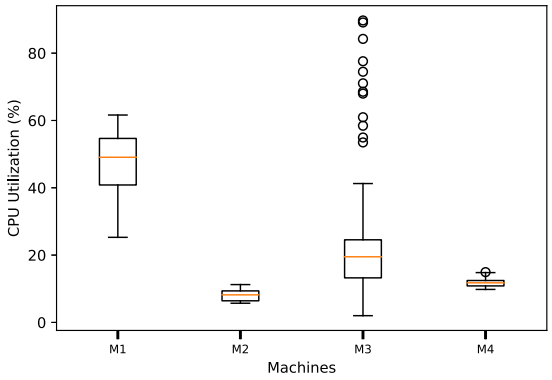

Fig. 6. Box plot for CPU utilization of selected four machines with different characteristics from the Alibaba data set. M1 = high load, M2 = low load, M3 = high variation, and M4 = low variation.

show that the proposed methodology does not over-fit to the main data set (Alibaba).

To train and validate the machine learning models in the AMS classifier, we are using a random split of 80% data for training, and the remaining 20% for validating the models. The test data also includes the four machines which are discussed in the following paragraph.

As applications running on data-centers can have different profiles, we selected four machines from the Alibaba data set with very distinct CPU demands, to test the resource estimation on different demand behaviors. Figure 6 shows the boxplot of CPU utilization of the four selected machines: Machine M1 serves a workload demanding high CPU resources; machine M2 serves a workload requiring low CPU resources; machine M3 serves a workload requiring CPU resources with highly fluctuating demand; and finally machine M4 serves a workload requiring CPU resources with low fluctuating demand.

Our proposed solution is then compared with the aforementioned baseline methods, proposed by Liu *et al.* [30], using Linear Regression (LR) and Support Vector Machines (SVM) methods to estimate adaptively CPU utilization of VMs. The combination of the two methods, LR for the slow-changing workloads and SVM for the fast-changing ones, are here labeled as "Liu" method. In addition, we also add the methods namely LR, SVM, Kriging (KR) and Gradient Boosting Tree (GBT) to consider for comparison with ours.

C. Experimental Details

1) Adaptive Model Selector Evaluation: The Adaptive Model Selector (AMS) is in charge to estimate which of the available ML algorithms will provide better modeling for the current data being monitored. We performed a set of experiments to evaluate different methods to make such estimation by comparing different classifiers namely Random Decision Forest (RDF), Gradient Boosting Tree (GBT), Multi-layer Perceptron (MLP), K-Nearest Neighbors(k-NN), Gaussian Naive Bayes (NB), and Support Vector Machine (SVM) with linear kernel. These classifiers are trained and validated using the Alibaba data set. We trained all classifiers on 80% of the entire Alibaba data set and then tested on the remaining 20% data to compare and identify the best classifier to used in AMS.

Training and validation data will be structured in time windows, as explained in Section VI-C3. The classifiers are evaluated through True/False Positive Rates (TPR and FPR), accuracy, recall, f-measure, and precision. We also consider the performance of the AMS by measuring the training time, prediction time, and the size of the model on disk.

2) Resource Estimation Evaluation: Finally, when integrating the different techniques of model selection and resource modeling, we perform a set of experiments to evaluate the resource estimation using our proposed adaptive ensemble. The final goal is to identify, on-line, the best regression method that will build a prediction model for estimating the resource utilization of the next future interval, given the current monitored data.

As this problem is a regression one, we evaluate the complete mechanism using the Root-Mean Square Error (RMSE) as shown on Equation 1 to show how our method deviates from the truth, also the Mean Absolute Error (MAE) as shown on Equation 2 to show the absolute magnitude of the produced error. Here a_t is the true CPU utilization and p_t is the estimated CPU utilization at time interval *t*, and *n* is the number of performed estimations.

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{n} (a_t - p_t)^2}{n}} \tag{1}$$

$$\text{MAE} = \frac{1}{n} \sum_{t=1}^{n} |a_t - p_t| \tag{2}$$

Again, training and validation data will be structured in time windows (Section VI-C3), and then for each sliding window, we use the AMS to identify the best regression method to estimate the resources for the following intervals.

3) Window Size Sensitivity: A specific observation window size is required to train the AMS. In this experiment, we evaluate the effect of different window sizes on the proposed solution by quantifying the estimation error using Alibaba dataset. We tested window sizes of 20, 40, 60, 80 and 90 minutes of data to train and validate the proposed solution for resource estimation. To segment the data set into training/validation sets, we performed a random split 80%/20%. We organized the training data into windows of the aforementioned sizes, and evaluate the models and ensemble, using the RMSE and MAE metrics to quantify the effect of each different window size.

VII. Experimental Results

A. AMS Evaluation

The Adaptive Model Selection method is evaluated through the aforementioned quality metrics for different classifiers, and check not only accuracy but also the performance requirements for each, like time for training and predicting, and size of the resulting model.

Table II shows the evaluation results of the AMS using the selected features of the raw data set to identify the best prediction method for CPU resource utilization estimation

TABLE II
AMS EVALUATION RESULTS USING DIFFERENT CLASSIFIERS FOR ALIBABA DATA SET

Classifier	TPR	FPR	TNR	FNR	Precision	Recall	F-measure	Accuracy
KNN	0.62	0.11	0.88	0.37	0.65	0.65	0.65	0.65
MLP	0.64	0.11	0.88	0.35	0.66	0.67	0.66	0.67
NB	0.33	0.22	0.77	0.66	0.38	0.31	0.29	0.31
RDF	**0.65**	**0.10**	**0.89**	**0.34**	**0.68**	**0.68**	**0.68**	**0.68**
GBT	0.48	0.16	0.83	0.51	0.55	0.53	0.51	0.53

TABLE III
TIME AND SPACE EFFICIENCY OF AMS USING DIFFERENT CLASSIFIERS FOR ALIBABA DATA SET

Classifier	Training Time (sec)	Prediction Time (sec)	Prediction Time per Request (ms)	Size (KB)
KNN	3.23	593.61	17.017	255283.2
Multi-layer Perceptron	728.13	0.34	0.010	180.7
Naive Bayes (Guassian)	0.59	0.13	0.004	7.5
RDF	**57.43**	**0.51**	**0.015**	**201523.2**
GBT	186.45	0.28	0.008	140.9

using Alibaba data set. The table shows true positive rate (TPR), false positive rate (FPR), true negative rate (TNR), false negative rate (FNR), precision, recall, f-measure, and accuracy for using kNN, Multi-layer Perceptron, Naive Bayes, RDF, and GBT as classification methods in AMS to identify the prediction method which can be used to estimate the CPU resources with high accuracy. The RDF outperforms all other classifiers. We observed that KNN, as second best classification method in AMS also provides comparable and closest results to RDF.

To profile the time and space efficiency of different classifiers for AMS using Alibaba data set, we profile training time, testing time, and the size of the trained model on the disk. Table III shows the time and space efficiency of AMS using different classification methods. We observed Naive Bayes classifier is efficient by consuming the least time to train and test the AMS. Whereas, the classification performance of Naive Bayes is significantly lower than RDF specifically for precision, recall, f-measure, and accuracy.

Although kNN classification performance is comparable to RDF, however, training, testing, and disk size of AMS using kNN is worst comparing to other classification methods. The RDF training and test time are reasonably good, and it outperforms other classification methods for all evaluation metrics. Therefore, we chose RDF classifier to use in our proposed AMS.

Figure 7 shows the Receiver Operator Characteristics (ROC) curve using RDF with AMS for different classes. ROC curves for all the classes are better than the random classifier. We observed that the proposed AMS with RDF efficiently classifies the test data for all the classes. The area under the ROC curves is 0.84, 0.89, 0.90, and 0.90 for SVM, LR, GBT, and KR labels respectively.

Overall, we observed that using RDF in AMS performs excellently to identify appropriate prediction method to use adaptively for the given data for resource estimations.

B. Resource Utilization Estimation

Table IV shows RMSE and MAE for CPU utilization estimations on test data of Alibaba data set for the proposed

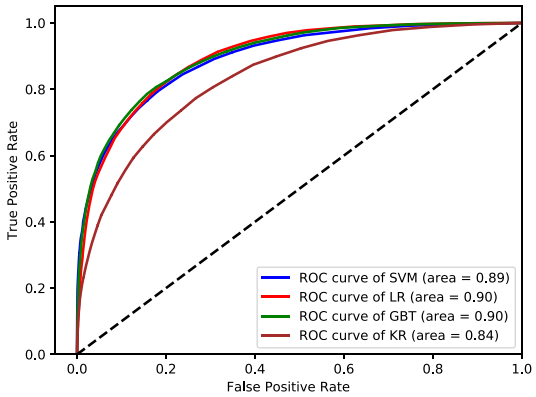

Fig. 7. ROC curves using RDF with AMS for different classes.

TABLE IV
RMSE AND MAE FOR RESOURCE ESTIMATION USING THE PURPOSED SYSTEM FOR ALIBABA DATA SET

Method	RMSE	MAE
GBT	4.57	3.43
LR	5.12	3.87
SVM	5.63	4.23
Kriging	5.26	3.99
Liu [30]	5.34	3.94
Proposed	**3.32**	**2.29**

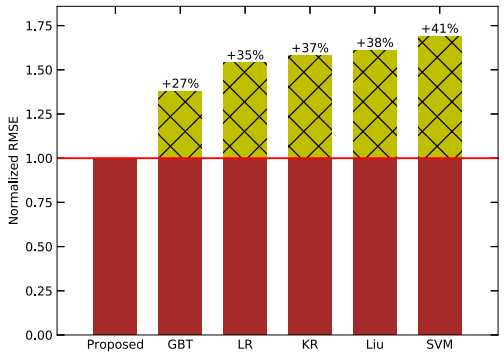

Fig. 8. Comparison of normalized RMSE for baseline methods with the proposed method using Alibaba data set.

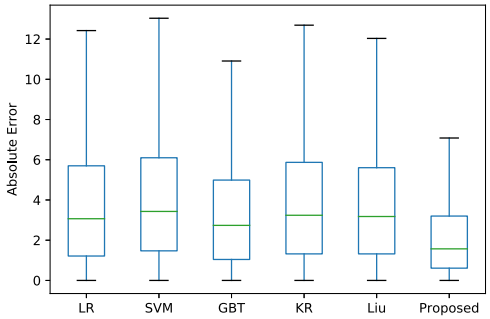

Fig. 9. Box plot of absolute error computed for each estimation using baseline and proposed methods for Alibaba data set.

and baseline methods. The proposed method outperforms all baseline methods by yielding minimum RMSE and MAE.

To compare the proposed method with baseline methods we normalized the RMSE with relative to the proposed solution, as shown in Figure 8. We observed 27%, 35%, 37%, 38%,

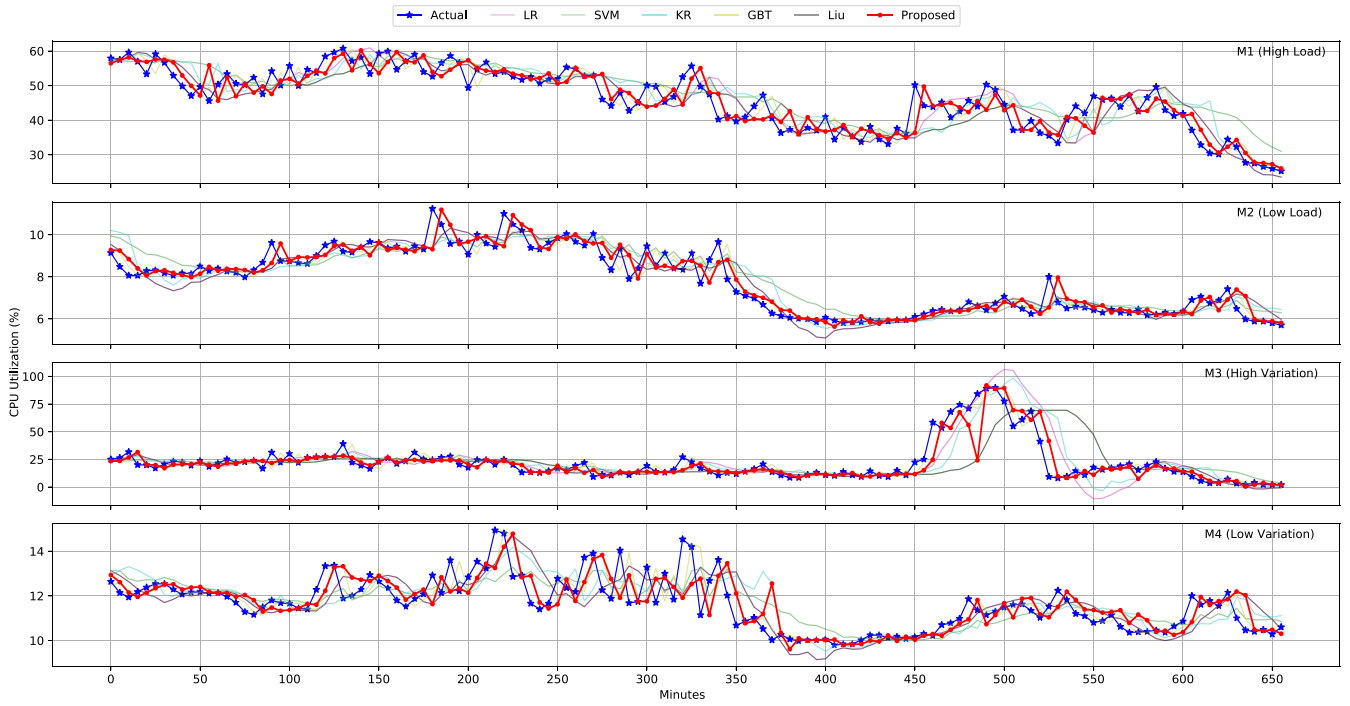

Fig. 10. Actual vs proposed method CPU prediction for Alibaba data set for four selected machines. M1 = Heavy workload, M2 = Low workload, M3 = High variation, M4 = Low variation. The window size used to train the prediction model is 60 minutes.

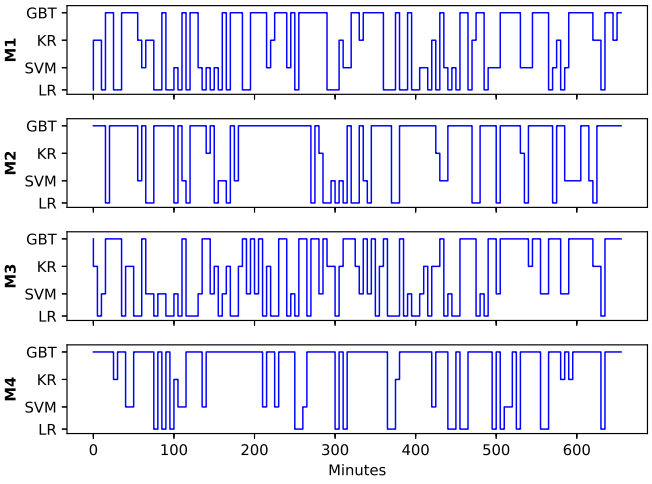

Fig. 11. Model selection of Adaptive Model Selector (AMS) for Alibaba data set for four selected machines. M1 = Heavy workload, M2 = Low workload, M3 = High variation, M4 = Low variation.

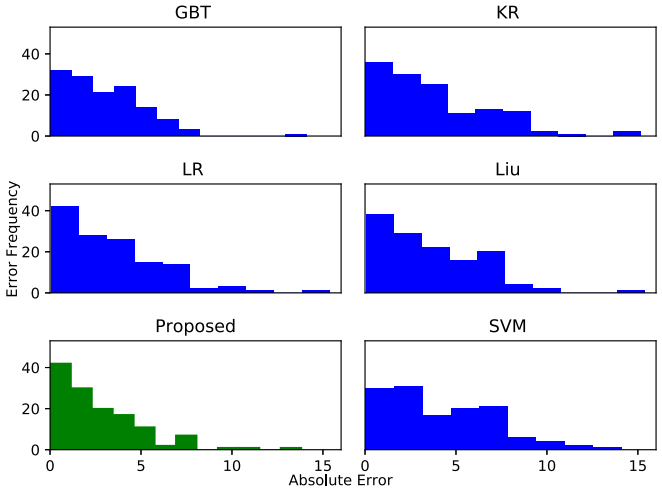

Fig. 12. Absolute error frequency of CPU utilization estimation for machine M1 (High Load).

and 41% less estimation error comparing to GBT, LR, KR, Liu, and SVM baseline methods.

Figure 9 shows the box plot for absolute error computed for each estimated CPU utilization using Alibaba data set for the proposed and baseline methods. We observed the proposed method outperforms the baseline methods to minimize the absolute error.

Figure 11 shows the recommendations proposed by AMS as a function of time for the selected four machines. The proposed method dynamically selects the most appropriate prediction model based on time series features of recent window. Figure 10 shows the comparison of actual and estimated CPU resources using baseline methods and with the proposed system for the four selected machines. The proposed method to estimate the CPU utilization shows significantly closer to the actual resource utilization for all of the machines serving a significantly different type of workloads. Moreover, it is hard to forecast in the presence of burst. For example, Figure 10 for M3 we observe a burst between 460 to 530 seconds, and the proposed solution tries to minimize the estimation error using different estimator as reflected in Figure 11. The proposed solution dynamically switches between different estimators to yield prediction with better accuracy.

To quantify and visualize the error for each estimation, we show absolute error frequency computed for machines M1

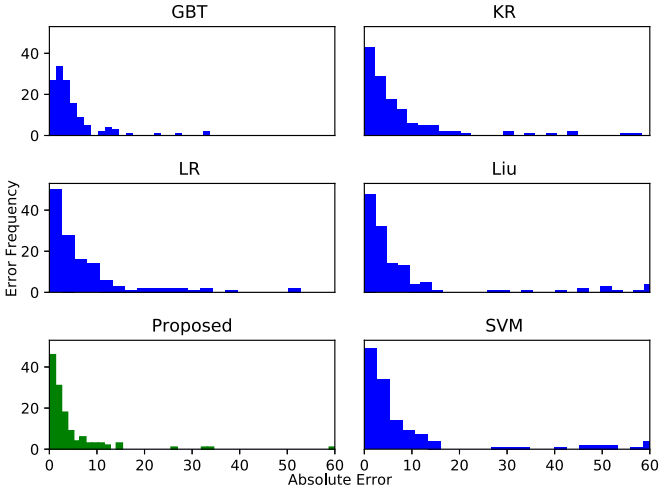

Fig. 13. Absolute error frequency of CPU utilization estimation for machine M3 (High Variation).

and M3 using baseline and proposed methods in Figure 12 and Figure 13 respectively. Where M1 serves a workload demanding high CPU resources consistently, and M3 served a workload requiring CPU resources with fluctuating demand. We observed that the proposed method always yield minimum error to estimate the CPU resource utilization for a different type of workloads. We also observed that the proposed method yield minimum absolute error for each estimation comparing to the baseline methods for both M1 and M3 machines.

C. Window Size Sensitivity Analysis

Figure 14 shows the RMSE and MAE for different windows sizes with the proposed system to estimate CPU resource estimation. We observe that increasing window size reduce the estimation error untill window size 60; however, after that, the error starts rising. The 20 minutes window size only contains four observations to fit the prediction models for an estimation which yields a maximum error. This experiment identifies that 60 minutes window size is optimal to use with the proposed system to minimize the estimation error. Therefore, in all of our experiments, we used 60 minutes window size with the proposed and baseline methods.

D. Evaluation Using BitBrains Data set

Table V shows RMSE and MAE for estimating CPU utilization on test data using Bitbrains data set for the proposed and baseline methods. The proposed method outperforms all baseline methods by yielding minimum RMSE and MAE.

To show the comparison of the proposed method with baseline methods, we normalized the RMSE with relative to the proposed solution. Figure 15 shows the comparison of baseline methods with the proposed solution by calculating normalized RMSE for the Bitbrains data set. We observed 6%, 39%, 42%, 54%, and 54% less estimation error comparing to GBT, LR, KR, SVM, and Liu baseline methods, respectively.

Figure 16 shows the box plot for absolute error computed for each estimated CPU utilization using Bitbrains data set for baseline and proposed methods. We observed the proposed

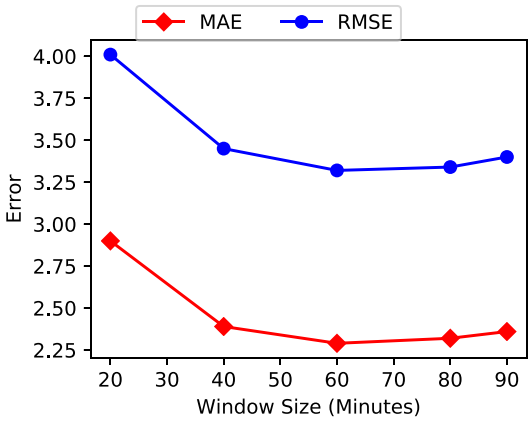

Fig. 14. RMSE and MAE using different window sizes with the proposed system for resource utilization estimation.

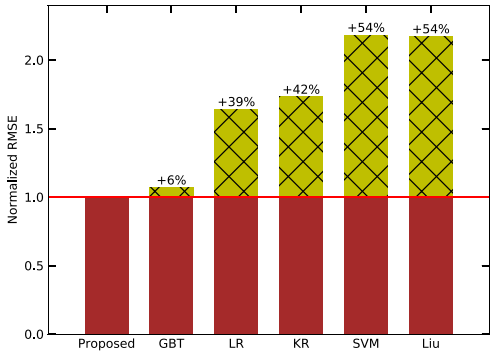

Fig. 15. Comparison of normalized RMSE for baseline methods with the proposed method using Bitbrains data set.

TABLE V
RMSE AND MAE FOR RESOURCE ESTIMATION USING THE PURPOSED SYSTEM FOR BITBRAINS DATA SET

Method	**RMSE**	**MAE**
GBT	9.74	2.85
LR	15.01	6.03
SVM	19.94	7.19
Kriging	15.80	6.05
Liu [30]	19.80	7.09
Proposed	**9.13**	**2.57**

method produces less absolute error compared to the baseline methods.

E. Evaluation Using Google Data set

After performing additional experiments using the Google dataset, the same used by Liu [30], we realized that while such dataset presents a behavior with less variance than Bitbrains (more than 80% of the machines report standard deviations below 4 in a range of 0 to 100), and all methods behave with similar good accuracy, also both methods Liu's and ours are better than the individual machine learning algorithms. But then, for the Bitbrains and Alibaba datasets with higher variance and more extreme behavior, and while Liu's method does not adapt that well, our method still does and improve the individual algorithms. Table VI shows RMSE and MAE for estimating CPU utilization on test data using Google data set

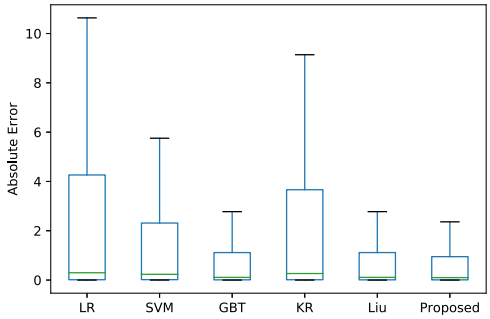

Fig. 16. Box plot of absolute error computed for CPU utilization estimation using baseline and proposed methods for Bitbrain data set.

TABLE VI
MSE AND MAE FOR RESOURCE ESTIMATION USING THE PURPOSED SYSTEM FOR GOOGLE CLUSTER DATA SET

Method	RMSE	MAE
GBT	2.31	1.24
LR	2.40	1.32
SVM	2.35	1.28
Krining	2.28	1.24
Liu	2.26	1.24
proposed	**2.22**	**1.14**

for the proposed and baseline methods. The proposed method outperforms all baseline methods by yielding minimum RMSE and MAE.

VIII. CONCLUSION

Building new methods for estimating resource utilization in data centers is an active and challenging problem, as most of the state-of-art techniques are based on specific machine learning methods, able to adjust to particular scenarios, but ineffective on extremely diverse environments. Therefore, we present a novel approach to adaptively and automatically identify the most appropriate machine learning method to be used for predicting future resource utilization, given recent observations of such resources.

In our proposed methodology, we use Random Decision Forest classifiers to determine, from a set of available machine learning techniques, which one is most appropriate for predicting resources on a next time interval, having monitored the previous one. The RDF is trained on the statistical features extracted from historical observations and samples of the best method identified for each time window. Our selected available methods include several techniques used in the current state of the art, as regression methods, neural networks, statistical learning, and bayesian approaches.

The proposed method is evaluated on real traces collected from Alibaba and Bitbrains data-center monitoring datasets, and our proposed approach can improve prediction accuracy from 6% to 27% over current methodologies. We also focused on the importance of monitoring time window sizes when modeling and predicting and evaluated different sizes. We found that 60 minutes of historical resource utilization observation can effectively be used to build the prediction model to estimate the future resource utilization.

We conclude that our methodology can help to identify the appropriate machine learning methods for each specific scenario over time, and future work will focus on investigating adaptive window size for modeling and predicting data center resource utilization. We also plan to extend the proposed system for online retraining automatically to adapt for changing characteristics. Moreover, we also intend to investigate the prediction for $t + n$ intervals.

REFERENCES

[1] A. Erradi, W. Iqbal, A. Mahmood, and A. Bouguettaya, "Web application resource requirements estimation based on the workload latent features," *IEEE Trans. Services Comput.*, to be published.
[2] W. Iqbal, M. N. Dailey, D. Carrera, and P. Janecek, "Adaptive resource provisioning for read intensive multi-tier applications in the cloud," *Future Gener. Comput. Syst.*, vol. 27, no. 6, pp. 871–879, Jun. 2011.
[3] Y. Xia, M. Tsugawa, J. A. B. Fortes, and S. Chen, "Large-scale VM placement with disk anti-colocation constraints using hierarchical decomposition and mixed integer programming," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 5, pp. 1361–1374, May 2017.
[4] T. H. Duong-Ba, T. Nguyen, B. Bose, and T. T. Tran, "A dynamic virtual machine placement and migration scheme for data centers," *IEEE Trans. Services Comput.*, to be published.
[5] L. Tang and H. Chen, "Joint pricing and capacity planning in the IaaS cloud market," *IEEE Trans. Cloud Comput.*, vol. 5, no. 1, pp. 57–70, Jan./Mar. 2017.
[6] M. Carvalho, D. A. Menascé, and F. V. Brasileiro, "Capacity planning for IaaS cloud providers offering multiple service classes," *Future Gener. Comput. Syst.*, vol. 77, pp. 97–111, Dec. 2017.
[7] A. Paya and D. C. Marinescu, "Energy-aware load balancing and application scaling for the cloud ecosystem," *IEEE Trans. Cloud Comput.*, vol. 5, no. 1, pp. 15–27, Jan./Mar. 2017.
[8] E. K. Lee, H. Viswanathan, and D. Pompili, "Proactive thermal-aware resource management in virtualized HPC cloud datacenters," *IEEE Trans. Cloud Comput.*, vol. 5, no. 2, pp. 234–248, Apr./Jun. 2017.
[9] Q. Zhang, L. T. Yang, Z. Yan, Z. Chen, and P. Li, "An efficient deep learning model to predict cloud workload for industry informatics," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3170–3178, Jul. 2018.
[10] K. Mason, M. Duggan, E. Barrett, J. Duggan, and E. Howley, "Predicting host CPU utilization in the cloud using evolutionary neural networks," *Future Gener. Comput. Syst.*, vol. 86, pp. 162–173, Sep. 2018.
[11] W. Zhang *et al.*, "Resource requests prediction in the cloud computing environment with a deep belief network," *Softw. Pract. Exp.*, vol. 47, no. 3, pp. 473–488, 2017.
[12] *Alibaba Cluster Log*. Accessed: Jul. 5, 2018. [Online]. Available: https://github.com/alibaba/clusterdata
[13] *Bitbrains Cluster Log*. Accessed: Jul. 12, 2018. [Online]. Available: http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains
[14] D. Erhan, A. Courville, and Y. Bengio, "Understanding representations learned in deep architectures," Comput. Sci. Dept. Oper. Res., Univ. Montreal, Montreal, QC, Canada, Rep. 1355, 2010.
[15] I. K. Kim, W. Wang, Y. Qi, and M. Humphrey, "Cloudinsight: Utilizing a council of experts to predict future cloud application workloads," in *Proc. IEEE Int. Conf. Cloud Comput.*, 2018, pp. 41–48.
[16] A. A. Rahmanian, M. Ghobaei-Arani, and S. Tofighy, "A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment," *Future Gener. Comput. Syst.*, vol. 79, pp. 54–71, Feb. 2018.
[17] J. Subirats and J. Guitart, "Assessing and forecasting energy efficiency on cloud computing platforms," *Future Gener. Comput. Syst.*, vol. 45, pp. 70–94, Apr. 2015.
[18] Z. Chen, Y. Zhu, Y. Di, and S. Feng, "Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network," *Comput. Intell. Neurosci.*, vol. 2015, p. 17, Jan. 2015. [Online]. Available: https://dl.acm.org/citation.cfm?id=2810647. doi: 10.1155/2015/919805.
[19] K. Cetinski and M. B. Juric, "AME-WPC: Advanced model for efficient workload prediction in the cloud," *J. Netw. Comput. Appl.*, vol. 55, pp. 191–201, Sep. 2015.
[20] F.-H. Tseng, X. Wang, L.-D. Chou, H.-C. Chao, and V. C. M. Leung, "Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1688–1699, Jun. 2018.

[21] Y. Jiang, C.-S. Perng, T. Li, and R. N. Chang, "Cloud analytics for capacity planning and instant VM provisioning," *IEEE Trans. Netw. Service Manag.*, vol. 10, no. 3, pp. 312–325, Sep. 2013.
[22] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449–458, Oct./Dec. 2015.
[23] S. Liao, H. Zhang, G. Shu, and J. Li, "Adaptive resource prediction in the cloud using linear stacking model," in *Proc. 5th Int. Conf. Adv. Cloud Big Data (CBD)*, 2017, pp. 33–38.
[24] C. Vazquez, R. Krishnan, and E. John, "Time series forecasting of cloud data center workloads for dynamic resource provisioning," *J. Wireless Mobile Netw. Ubiquitous Comput. Depend. Appl.*, vol. 6, no. 3, pp. 87–110, 2015.
[25] K. Dmytro, T. Sergii, and P. Andiy, "Arima forecast models for scheduling usage of resources in it-infrastructure," in *Proc. 12th Int. Sci. Techn. Conf. Computer. Sci. Inf. Technol. (CSIT)*, vol. 1, 2017, pp. 356–360.
[26] W. Fang, Z. Lu, J. Wu, and Z. Cao, "Rpps: A novel resource prediction and provisioning scheme in cloud data center," in *Proc. IEEE 9th Int. Conf. Services Comput. (SCC)*, 2012, pp. 609–616.
[27] F. Qiu, B. Zhang, and J. Guo, "A deep learning approach for vm workload prediction in the cloud," in *Proc. 17th IEEE/ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel Distrib. Comput. (SNPD)*, 2016, pp. 319–324.
[28] B. Song, Y. Yu, Y. Zhou, Z. Wang, and S. Du, "Host load prediction with long short-term memory in cloud computing," *J. Supercomput.*, vol. 74, no. 12, pp. 6554–6568, 2018.
[29] M. Duggan, K. Mason, J. Duggan, E. Howley, and E. Barrett, "Predicting host CPU utilization in cloud computing using recurrent neural networks," in *Proc. 12th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, 2017, pp. 67–72.
[30] C. Liu, C. Liu, Y. Shang, S. Chen, B. Cheng, and J. Chen, "An adaptive prediction approach based on workload pattern discrimination in the cloud," *J. Netw. Comput. Appl.*, vol. 80, pp. 35–44, Feb. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516303198
[31] A. Nanopoulos, R. Alcock, and Y. Manolopoulos, "Feature-based classification of time-series data," *Int. J. Comput. Res.*, vol. 10, no. 3, pp. 49–61, 2001.
[32] B. D. Fulcher and N. S. Jones, "Highly comparative feature-based time-series classification," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 12, pp. 3026–3037, Dec. 2014.
[33] *Tsfresh Features*. Accessed: Jul. 5, 2018. [Online]. Available: https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html
[34] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Proc. NIPS*, 1996, pp. 155–161.
[35] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent," in *Proc. 12th Adv. Neural Inf. Process. Syst.*, 2000, pp. 512–518.
[36] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, 2001.
[37] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *J. Glob. Optim.*, vol. 13, no. 4, pp. 455–492, 1998.
[38] A. Gambi, M. Pezzé, and G. Toffetti, "Kriging-based self-adaptive cloud controllers," *IEEE Trans. Services Comput.*, vol. 9, no. 3, pp. 368–381, May/Jun. 2016.
[39] F. Chollet, *Deep Learning With Python*. New York, NY, USA: Manning, 2017.
[40] T. K. Ho, "Random decision forests," in *Proc. 3rd Int. Conf. Docu. Anal. Recognit. (ICDAR)*, vol. 1, Montreal, QC, Canada, 1995, p. 278 [Online]. Available: http://dl.acm.org/citation.cfm?id=844379.844681
[41] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh—A python package)," *Neurocomputing*, vol. 307, pp. 72–77, Sep. 2018.
[42] (2018). *Cesium*. [Online]. Available: http://cesium-ml.org/
[43] L. Ge and L.-J. Ge, "Feature extraction of time series classification based on multi-method integration," *Optik Int. J. Light Electron Optics*, vol. 127, no. 23, pp. 11070–11074, 2016.
[44] B. Gregorutti, B. Michel, and P. Saint-Pierre, "Correlation and variable importance in random forests," *Stat. Comput.*, vol. 27, no. 3, pp. 659–678, 2017.
[45] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Electric. Eng.*, vol. 40, no. 1, pp. 16–28, 2014.
[46] *Feature Selector*. Accessed: Jul. 5, 2018. [Online]. Available: https://github.com/WillKoehrsen/feature-selector
[47] *Linear Trend Feature*. Accessed: Jul. 5, 2018. [Online]. Available: https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_extraction.html#tsfresh.feature_extraction.feature_calculators.agg_linear_trend
[48] *Google Cluster Log*. Accessed: Jul. 10, 2019. [Online]. Available: https://github.com/google/cluster-data
[49] T. H. Nguyen, M. Di Francesco, and A. Yla-Jaaski, "Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data centers," *IEEE Trans. Services Comput.*, to be published.

Shuja-ur-Rehman Baig received the M.Sc. degree in computer science from the Lahore University of Management Sciences, Lahore, Pakistan. He is currently pursuing the Ph.D. degree with the Computer Architecture Department, Universitat Politècnica de Catalunya. He is also a Collaborator with the Data-Centric Computing Group, Barcelona Supercomputing Center. He is also a Lecturer with the Punjab University College of Information Technology, University of the Punjab, Lahore.

Waheed Iqbal received the Ph.D. degree from the Asian Institute of Technology, Thailand. He is an Assistant Professor with the Punjab University College of Information Technology, University of the Punjab, Lahore, Pakistan. He was a Post-Doctoral Researcher with the Department of Computer Science and Engineering, Qatar University from 2017 to 2018. His research interests in cloud computing, distributed systems, machine learning, and large-scale system performance evaluation.

Josep Lluis Berral received the degree in informatics, and the M.Sc. and Ph.D. degrees in computer architecture from BarcelonaTech-Universitat Politècnica de Catalunya (UPC) in 2007, 2008, and 2013, respectively. He is a Data Scientist with Barcelona Supercomputing Center, where he is working in applications of data mining and machine learning on data-centric computing scenarios. He was with the High Performance Computing Group, Computer Architecture Department, UPC, and the Relational Algorithms, Complexity and Learning Group, Computer Science Department, UPC.

Abdelkarim Erradi received the Ph.D. degree in computer science from the University of New South Wales, Sydney, NSW, Australia. He is an Associate Professor with the Computer Science and Engineering Department, Qatar University. He has authored several scientific papers in international conferences and journals. His research and development activities and interests focus on autonomic computing, self-managing systems, and cybersecurity. He leads several funded research projects in the above areas.

David Carrera received the M.S. and Ph.D. degrees from BarcelonaTech-Universitat Politècnica de Catalunya (UPC) in 2002 and 2008, respectively, where he is an Associate Professor with the Computer Architecture Department. He is an Associate Researcher with the Data-Centric Computing, Barcelona Supercomputing Center. His research interests are focused on the performance management of data center workloads. He has been involved in several EU and industrial research projects. He was a recipient of the IBM Faculty Award in 2010 and the ERC Starting Grant for the Project HiEST in 2015.