Load Prediction for Data Centers Based on Database Service

Rui Cao[1], Zhaoyang Yu[1], Trent Marbach[1], Jing Li[1,2], Gang Wang[1*], Xiaoguang Liu[1*]
[1]Nankai-Baidu Joint Lab, College of Computer and Control Engineering, Nankai University, Tianjin, China
[2]College of Computer Science and Technology, Civil Aviation University of China, Tianjin, China
Email: {caorui, yzz, trent.marbach, lijing, wgzwp, liuxg}@nbjl.nankai.edu.cn

***Abstract*—In the era of cloud computing, the over-occupancy of data center resources (CPU, memory, disk) and subsequent machine failure have resulted in great loss to users and enterprises. So it makes sense to anticipate the server workload in advance. Previous research on server workloads has focused on trend analysis and time series fitting. We propose an approach to forecast the workloads of servers based on machine learning. And our data comes from a database-based data center that is real, large-scale, and enterprise-class. We use the servers' historical monitoring data for our models to predict future workloads and hence provide the ability to automatically warn overload and reallocate resources. We calculate the failure detection rate and false alarm rate of our overload detection models, as well as put forward an evaluation based on the overload processing cost. Experimental results show that machine learning methods especially Random Forest can better predict the server load than traditional time series analysis method. We use the forecast results to propose some scheduling strategies to prevent server overload, achieve intelligent operation and maintenance, and failure prediction. Compared with the traditional time series analysis method, our method uses less data and lower dimensions, and yields more accurate predictions.**

***Index Terms*—data center, load prediction, overload processing cost**

I. Introduction

As the core of data processing, data exchange, and data storage, the data center has become one of the most important infrastructures in the cloud computing era [1], [2]. The demand for data communication and information services is rapidly increasing, so data centers that provide database services are widely used. Also due to the rapid increase in demand, some issues faced by data centers are prevalent, such as managing and maintaining a wide spectrum of data. In addition, high utilization of workloads such as CPU, memory, and disk not only affects users' usage, but also may cause data center failures that result in huge losses to enterprises.

According to an Emerson Network Power-sponsored study conducted by the Ponemon Institute[1], the average cost of a single data center outage today is about $730,000 and the cost can rise significantly. As an example, the Delta Airlines data center outaged in August 2016, which grounded about 2,000 flights over the span of three days and lost the company $150 million [3].

A survey conducted by The Data Center Journal[2] analyzed the cost of a data center outage within different industries. They found the cost of a data center failure to the energy industry is $2.1878 million, to the telecommunications industry is $2.0662 million, and to the financial industry is $1.4951 million.

In 2012, the UPTIME Institute[3] surveyed 94 computer rooms around the world and created a report including 291 incidents and 8 failures in total [4]. Of the 291 incidents, 39% were caused by operational reasons such as overload. Among the 8 failures, 25% were caused by operations. Of the 213 incidents, 13% were compensated and rescued by operational intervention. This shows that in a data center it is important to not only physically maintain the equipment, but also that operational tasks are performed well.

To avoid server failure, a scheme is needed to notify operations engineers when a server is overloaded. At present, this is done by the threshold method using real-time workload monitoring in which a threshold is set for each monitoring item, and the operations engineers are notified to intervene when the load exceeds the threshold [5]. However, this method has some problems.

- When an emergency situation occurs, for example a quick rise in resource usage, the monitoring indicator may deteriorate rapidly. If the operations engineers can not handle it promptly, the performance of the system may be seriously affected.
- In addition, for a large cluster of servers, a great number of overloaded machines may need to be handled simultaneously, which would put a significant burden on operations engineers.

In a database service-based data center, the failure of a machine in one cluster may affect other machines related to this [6]. Therefore, the automatic and intelligent operation of the data centers is particularly important, which we hope to achieve through our forecasting work and improve the availability of data centers based database services.

We use workloads data from a data center over a period of time to make a prediction of future servers overload. With this, we are able to provide overload detection in advance, and take timely measures to deal with failures. The method

[1]https://www.ponemon.org/

[2]http://www.datacenterjournal.com/

[3]https://uptimeinstitute.com/

turns a 'passive response' after the incident is underway into an 'active prevention' that takes action before a failure occurs. This means the data and tasks on the database server can be migrated to ensure the normal and efficient working status of the server. In addition, the server may select to redistribute tasks, thereby reducing resource consumption while preventing failures. A pursuit of business is to maximize profits, which means that reducing unnecessary cost is crucial. This fail-safe approach not only improves the user experience, but also helps the company save significant amounts of money and effort. We use the overload processing cost to measure different models as a way of distinguishing their effectiveness.

We make the following contributions:

- For a large enterprise data center based on database services, we compare time series method and several machine learning models and analyze the characteristics of the workload according to the experimental results.
- We evaluate the six models which used in our experiment to predict the workload by a new evaluation standard that demonstrates the cost of the model in terms of overload processing cost.
- We propose several strategies for task scheduling based on the results of load forecasting, which can avoid overload or save resources.

The rest of the paper is organized as follows: Section II discusses related work, Section III presents an overview of the workload prediction framework, Section IV details the experimental process and analysis, and we present the conclusion in Section V.

II. Related Work

Time series analysis [7] is used to analyze dynamic timing data and is often handled with statistical approaches and machine learning methods. Popular statistical methods include ARIMA [8], SARIMA [9], Gaussian [10], and HMM [11]. Machine learning models such as SVM [12], RNN [13], and LSTM [14] have also been used for time series analysis.

In the past, most of the research within machine learning has focused on network traffic, malicious behavior detection, intrusion detection, registry anomalies, and so on [15]–[19]. There is little research on the workload prediction of data centers based on database services.

Stokely et al. [20], [21] studied disk allocation in a distributed storage system. They collected disk occupancy, I/O rates, and the duration of data storage from thousands of different engineering users and teams to forecast the feature usage. Their work is similar with ours but we focus on the workloads of data centers, which mainly depends on the server processing tasks, while Stokely's work is for users in distributed storage, which is affected by users' habits. Wamba et al. [22] proposed two workload forecasting models based on constrained programming and neural networks to predict the CPU usage of a cloud data center's physical servers to facilitate resource deployment in a cloud environment. Both works of Stokely et al. and Wamba et al. are about a single system attribute, but the performance of the data center is affected by several system attributes.

Xue et al. [23] presented a neural network-based framework called PRACTISE to predict the future workload of the CPU, memory, disk, and network bandwidth, and showed the model had reliable accuracy, robustness, and flexibility. They developed a bagging module to reduce the randomness of training and testing sets, used online-updating models, and trained a neural network. Unfortunately, the bagging module they developed is not computationally efficient in an online context. Karakurt et al. [24] used machine learning methods include LR, SVM, and RF to predict the failure of database-related management systems and showed that supervised learning algorithms, especially RF, have satisfactory recall rate and accuracy. However, we find no model has an absolute advantage, so we put forward a new evaluation system based on the failure detection rate and the false alarm rate, which takes into account the cost of migrating and storing data.

Predicting workload is only part of the job, and more importantly, forecasting result should be used to optimize the performance of servers in data centers. Cortez et al. [25] used Resource Center (RC) to collect VM telemetry, learn behaviors offline, and provide predictions to various resource managers online through client libraries. They modified the VM schedulers, which was used to demonstrate that predictive notifications improve utilization and prevent physical resources from running out. Hu et al. [26] put forward three models to predict workload. They also proposed a new trigger strategy for the cloud computing elasticity mechanism. Their works give us the inspiration for our scheduling strategies.

III. Overview

In this section, we describe the work flow of the workload prediction in a real data center. Figure 1 demonstrates the overview of the workload predicting system. We first collect historical workload data from the monitoring system of the data center, including CPU, memory, and disk data. Then we clean the data, performing tasks such as filling-in the missing values, deleting invalid values, and standardizing the values to ensure we have high-quality and accurate data. After feature selection, we train the models using the cleaned data, and then predict the workload after certain periods of time. Finally, the forecast results are fed back to the system and the operations engineers for scheduling policies and other troubleshooting actions.

A. Data Collection

In a real data center, daily overload prediction is necessary, so we need to take the CPU, memory and disk data from the server monitoring system every morning. After the data is acquired, we organize it into a fixed data format that contains the values of the monitored items (represented as CPU, memory, and disk utilization) and sampling time points.

In total we take the previous 5 days worth of data from the system as a training sample. This data includes 2 days considered as feature samples and 3 days of data to extract

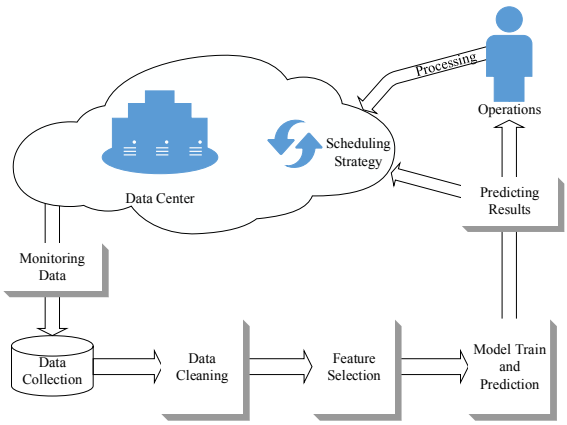


Fig. 1: Overview of the work flow for workload prediction

the target values. In the last 3 days, we label the target values by whether overload occurs within 1 hour, 6 hours, 12 hours, 1 day, 2 days, and 3 days. For the testing samples, we use the previous 2 days data as the input of trained models and predict the load for the next 3 days. Since the number of overloaded servers is much smaller than the number of servers that are not overloaded, in order to guarantee the quality of the prediction model, we need to use additional overload data, which is older and held in a repository of all the servers' recent overloads.

B. Data Cleaning

The collected data is drawn from multiple business systems. So such data may have varying levels of accuracy, and may even conflict with each other. Inaccurate data such as this is not useful to our models, so we have to clean the collected data. Data cleaning is the process of re-examining and verifying data for the purpose of removing duplicate information, supplementing missing information, and providing data consistency.

In the system framework of this paper, the process of data cleaning is completed by the computer program automatically and includes two parts: lost and duplicated data processing and data normalization.

- Various reasons (such as failure of the monitoring system) can cause some single data points to be lost or duplicated. In this paper, we use the imputation method for missing values, that is, taking the average of the two adjacent time points as a supplement value of the missing one. In the case that larger amounts of data is missed from some servers during the sampling period because of server outages or replacements, we abort the servers from the original sampling set. For duplicated data, if the data values are consistent, we make up the sample, otherwise delete the sample.
- Due to the various numeric ranges of the values that the data could take, it is beneficial to normalize the data, and doing so can improve the classification results [27]. In this paper, we use Min-Max standardization, shown as Formula (1). It applies a linear transform to the original data so that each of the resulting transformed data lies on the $[0,1]$ interval.

$$\text{Value}_{\text{new}} = \frac{\text{Value}_{\text{origin}} - \text{Value}_{\text{min}}}{\text{Value}_{\text{max}} - \text{Value}_{\text{min}}} \tag{1}$$

In above formula, $\text{Value}_{\text{new}}$ is the standardized data and $\text{Value}_{\text{origin}}$ is the original data, both of which correspond to certain time points. The symbols $\text{Value}_{\text{max}}$ and $\text{Value}_{\text{min}}$ represent the maximum and minimum values of the corresponding features over the entire sampling period.

C. Feature Selection

Feature selection refers to selecting some features and ignore others [28], as using all of the sampled data as a training set not only fails to improve the predictive performance of the model, but also lengthens the time to train it. In real-time online prediction systems, it is unrealistic to analyze the importance of each feature without automation. Therefore, this paper applies feature selection based on tree models which can achieve it based on the weights of each feature.

Throughout our experiments, a feature selection method based on the Gradient Boosting Decision Tree (GBDT) [29] works best. In this method, all the features are used to establish the training model, and certain features whose weight is larger than the given threshold in the training model are chosen for prediction. In addition, we have six prediction targets (1 hour, 6 hours, 12 hours, 1 day, 2 days, 3 days), each of which has the corresponding selected features. We consider some feature valid when there are four or more targets have chosen it. Such feature selection reduces the data dimension and makes classification more accurate and effective.

D. Model Training and Prediction

In this subsection, we use the selected features and the corresponding algorithms to build a predictive model, and then enter the test sample into the model and conclude whether a server will overload for the next three days. The following describes some of the models used in this article.

1) DT: Decision Tree (DT) is a tree structure, in which each non-leaf node represents the judgment of the characteristic attributes and each leaf node represents a category [30]. The decision-making process is to proceed from the root node, judge the corresponding attributes of the samples layer by layer and select the corresponding branch until a leaf node is reached. The leaf node's category is the sample's category.

In this paper, we apply the Classification And Regression Tree (CART) [31] to train the DT. The generation of a CART is a recursive process of building a binary decision tree that can be used both for classification as well as for regression. For classification trees, CART uses the Gini coefficient minimization criteria for feature selection and generates a binary tree. For a given node t, its impurity is a measure of the proportion of records in this node that belong to differing category. The Gini coefficient shown as Formula (2) is used to evaluate the nodes impurity:

Algorithm 1 The CART Algorithm

Input: Data set $D = \{(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_n)\}$; Feature set F=$\{f_1, f_2, \ldots, f_k\}$; Cut Point set C=$\{C_1$=$\{f_{1,1}, f_{1,2}, \ldots, f_{1,m_1}\}, C_2$=$\{f_{2,1}, f_{2,2}, \ldots, f_{2,m_2}\}, \ldots, C_k$=$\{f_{k,1}, f_{k,2}, \ldots, f_{k,m_k}\}\}$; Stop Condition
Output: A CART Decision Tree T

```
T ← create a tree node based on data set D
if Stop Condition then
    return T
end if
for every feature f_i in F do
    for every f_{i,j} in C_i do
        G_{i,j} ← Gini coefficient of f_{i,j}
    end for
end for
the minimum G_{i,j} which belongs to feature f_i
D_l ← subset of D if sample point satisfy f_i < f_{i,j}
D_r ← subset of D if sample point satisfy f_i ≥ f_{i,j}
T_l ← Call the CART with (D_l, F, C, Stop Condition)
T_r ← Call the CART with (D_r, F, C, Stop Condition)
Return T
```

$$\mathrm{Gini}(t) = 1 - \sum_j [\mathrm{p}(j \mid t)]^2, \tag{2}$$

where $\mathrm{p}(j \mid t)$ is the relative frequency of class j at node t (that is the proportion of records belonging to class j in a given node t).

The gain Δ shown in Formula (3) is a criterion that can be used to determine the effect of division under a property condition test which compares the impurity of the node before the division and the sub-nodes' impurity after the division. The bigger the difference is, the better the branch conditions it choose.

$$\Delta = I(\mathrm{parent}) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j) \tag{3}$$

In the above formula, $I(.)$ is the metric of impurity for a given node (for us, the Gini coefficient), N is the total number of records on a parent node, k is the number of attribute values, and $N(v_j)$ is the number of records associated with the child node v_j. Because $I(\mathrm{parent})$ is a constant value for all test conditions, the maximized gain is equivalent to the minimized weighted average of the impurity of the child node.

The process of CART building is shown in Algorithm 1. In this algorithm, the data set D is what we use to train a DT, the feature set F is the features of each sample in D and the Cut Point set C is all the values of each feature. The Stop Condition is that all instances belong to one class, the Gini coefficient is smaller than some threshold, the number of samples is smaller than some threshold, or F is empty.

2) RF: Random Forest (RF) [32] is a classifier that uses multiple DTs to make a prediction, selecting the output category determined by a voting mechanism of all the individual decision trees' predictions.

Algorithm 2 The Random Forest Algorithm

Input: Data set D=$\{(X_1,y_1),(X_2,y_2),\ldots,(X_n,y_n)\}$; Feature subset size K; the number of estimators m.
Output: Majority vote of estimators $\{\mathrm{T}_1, \mathrm{T}_2, \ldots, \mathrm{T}_m\}$

```
for m_i ∈ {1, 2, ..., m} do
    D_{m_i} ← bootstrap sample from D for n' times
    F_{m_i} ← select K features from F randomly;
    Construct classifier T_{m_i} based on D_{m_i} for feature set F_{m_i} through Algorithm 1
end for
```

The DT model is easy to overfit, however the RF model can improve upon this shortcomings. The good performance is benefits from two aspects: Bagging algorithm and Random Subspace Method (RSM). Bagging algorithm ensures the random selection of samples. It constructs sub-datasets by bootstrap sampling to build sub-decision trees and generate a strong classifier by combining these weak sub-decision tree classifiers. RSM guarantees the random selection of features. Instead of using all the features, each sub-tree in the RF selects some of the features randomly, reducing the correlation between each sub-tree and making them highly varied. It improves the diversity of the system and so enhances the classification performance.

RF is described as Algorithm 2 with the RSM and the Bagging strategy.

The Bagging strategy steps are as follows:

a) Choose n' samples from the sample set containing n samples by bootstrap sampling;
b) Establish classifiers for all attributes based on the n samples;
c) Repeat the above two steps m times to obtain m classifiers;
d) Predict with the m classifiers and finally decide which category the sample belongs to based on the voting results of the m classifiers.

E. Scheduling Strategies

The predicting results are fed back to the system and operations engineers, then we can apply some scheduling strategies to migrate some of the tasks in the server which is going to overload to other free servers to ensure the normal operation of the current server.

In addition, we also propose some other scheduling strategies to save resources according to the predicting results in subsection IV-D, but have not yet verified their benefit through experiments. Furthermore, the use of experienced operations engineers to deal with proposals will help achieve automatic migration of the system.

IV. Experiment

In this section, we describe the experiment in detail. First, the experiment setup is introduced including data set generation, feature selection, model training and prediction. Next,

we give the failure detection rate (FDR) and false alarm rate (FAR) after different periods of time for the six models. The word 'failure' here means overload rather than system failure. Then, we analyze the experiment performance based on the prediction results. In order to evaluate the quality of the models, we propose an overload processing cost from the perspective of practical application of enterprises. The methods can prevent failures caused by overload, and relocate tasks by several strategies which can take up less load and save resources.

A. Experiment Setup

1) Data Set: Our data comes from Baidu Inc., an Internet company in China. It is a real-world, large-scale data set that represents an enterprise-wide system and therefore can support our experiments. Due to commercial security, our data is not accessible to the public. The monitoring values of each server's workload (CPU, memory, and disk) are sampled every 10 minutes. We collect data from 5,731 servers over 67 days between December 19, 2016 and February 23, 2017, yielding a total of 55,311,984 records. The data contains the workload utilization rate and corresponding time point. Due to the severe loss of record information on some servers, these data were removed. After data cleaning, there are 5,466 servers and 52,735,968 sample records in total.

Figure 2 plots some examples of the load records of CPU, memory, and disk. We find that the workload can be divided into periodic and aperiodic. In our experiment, we did not distinguish between periodicity to train different models. And in a real data center, such as the one we surveyed, the server's CPU load usually fluctuates drastically, and memory and disk usage usually change more smoothly.

2) Compared Methods: To compare the performance of multiple predictive models, we use the following methods:

- ARIMA: Autoregressive Integrated Moving Average Model (ARIMA) [33] is often used to analyze stationary stochastic processes and has important applications in modeling and predicting time series. The data sequence formed by the prediction object over time is considered as a random sequence that can be fitted by a class of constant coefficient difference equations. Once the mathematical model has been established by the past time series, future values can be predicted from the fitted equations.
- NB: Based on the Bayesian theorem, Naive Bayes (NB) model [34] classifies some item as coming from one of several classes, depending on the items characteristics. For the given item to be classified, the model considers the probability of occurrence of each category under the conditions that appear. It classifies the item into the category that has the largest probability. This model performs well for small-scale data and can handle multi-category tasks for incremental training.
- LR: Logistic Regression (LR) [35] is a widely used classification machine learning algorithm. For problems where the samples' features can be continuous values and the range of the values is unbounded, logistic regression fits the data to the function with a binary dependent variable denoting our prediction.
- SVM: In machine learning, Support Vector Machine (SVM) [36] is a supervised learning model commonly used in pattern recognition, classification, and regression analysis. For linearly separable cases, its goal is to find a hyperplane that maximizes the separation of categories. For linearly non-separable, one way is to map the sample space into a high-dimensional space and transform the problem into a linearly separable one. And another way is to use a straight line but without guaranteeing complete separability, and add a penalty function that indicates the distance of this point from its correct category position.
- In addition, we use DT and RF to train the prediction model introduced in subsection III-D

The methods of time series analysis emphasize the periodicity and trend of sequence changes. In order to find the periodicity, it is usually necessary to use continuous time series. It should be noted, however, that we have found that longer time series do not improve the accuracy of predictions when predicting servers' load, and even reduce the accuracy for most servers. This is because the workloads of servers change frequently and recent data is more likely to be useful to predict future load. The methods of machine learning can be independent of continuous time, and they build models using important features obtained through feature selection rather than all of the data.

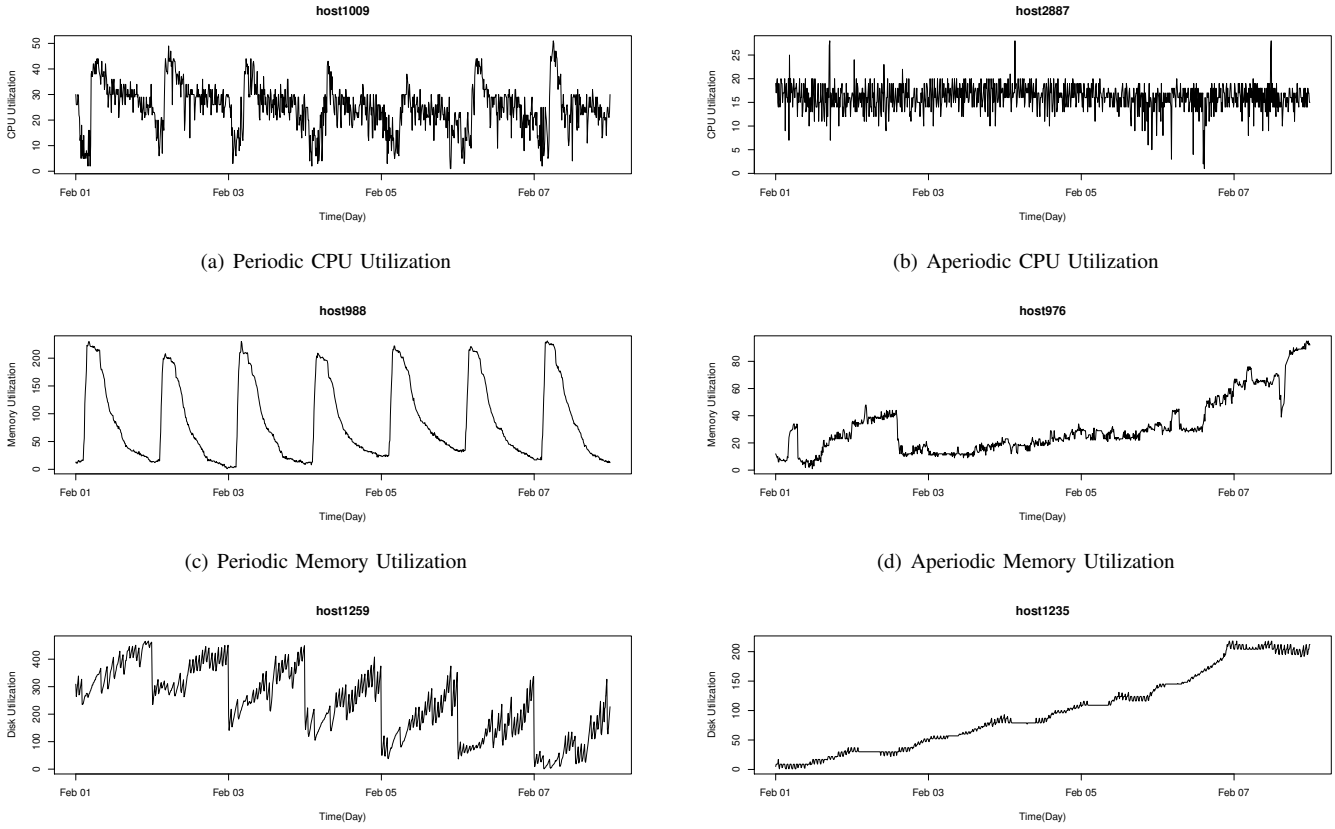
3) Model Training: As we are predicting server overload, rather than failure, we define an overload threshold T of CPU to be 80%, memory to be 80%, and disk to be 75%. These threshold choices and the different weight of positive and negative samples in models ensure the overloaded and normal samples reach equilibrium. We divide the training set and testing set into $6:4$. We determine the model parameters by ten-fold cross-validation (10-fold CV) in the training set for four models (LR, SVM, DT, and RF). After determining the model parameters, ten experiments are conducted separately, using the average as a result. We use *auto.arima*[4] in R to implement an ARIMA model and the rest of the machine learning methods are implemented from the scikit-learn package[5] in Python. The five machine learning models predict whether an overload will occur (binary) within 1 hour, 6 hours, 12 hours, 1 day, 2 days, or 3 days, while the ARIMA model predicts the value of the load at these time. In order to standardize this, we only predict an overload for the ARIMA model when it returns a predicted load above the threshold T, and otherwise predict that no overload will occur.

B. Experimental results

Our prediction is binary, classifying a sample as either positive or negative. In this paper, positive refers to the overloaded workload and negative represents the normal workload. As such, our prediction can result in one of four cases. If

[4]https://cran.r-project.org/web/packages/forecast/

[5]http://scikit-learn.org/



(a) Periodic CPU Utilization

(b) Aperiodic CPU Utilization

(c) Periodic Memory Utilization

(d) Aperiodic Memory Utilization

(e) Periodic Disk Utilization

(f) Aperiodic Disk Utilization

Fig. 2: Workload Utilization Records

an instance is positive and is also predicted as positive, it is called True Positive (TP), and if an instance is negative but is predicted as positive, it is called False Positive (FP). Correspondingly, if an instance is negative and is predicted as negative, it is called True Negative (TN), and if an instance is positive but is predicted as negative we name it False Negative (FN). Table I tabulates the category of each predicted sample.

TABLE I: Prediction cases

		Actual class	
		$\geq T$	$< T$
Predicted class	$\geq T$	TP($Type1$)	FP($Type1$)
	$< T$	FN($Type2$)	TN
total		P	N

We will list the FDR (see Formula (4)) and FAR (see Formula (5)) for each model, as well as workload types and time periods. Given a threshold of T, the FDR depicts the proportion of the positive instances correctly identified by the classifier as positive instances. The FAR calculates the proportion of negative instances that the classifier incorrectly predicts as positive.

$$\text{FDR} = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{\text{TP}}{\text{P}} \tag{4}$$

$$\text{FAR} = \frac{\text{FP}}{\text{FP}+\text{TN}} = \frac{\text{FP}}{\text{N}} \tag{5}$$

The predicted FAR and FDR results are shown in Table II.

C. Experimental analysis

In this subsection, we first analyze the experimental results of each model to predict the CPU, memory, and disk usage, and then describe the new evaluation system.

1) Result Analysis:

- We can find from the Table II, for CPU and disk, almost all of the models show a downward trend in FDR and an upward trend in FAR as predicted time increases. It is easy to understand that as a prediction time point far from the data of the training set, the correlation between the data of the training set will decrease, the probability of being influenced by other factors during the period will increase, and the weight of the available training features of the data will be weakened, which result in a decrease in FDR and an increase in FAR.

TABLE II: Experiments on the workload (CPU, memory, and disk) using six models, showing the FDR and FAR for each case

Workload	Model	one hour(%)		six hours(%)		twelve hours(%)		one day(%)		two days(%)		three days(%)	
		FDR	FAR	FDR	FAR	FDR	FAR	FDR	FAR	FDR	FAR	FDR	FAR
CPU	ARIMA	78.57	0.27	73.33	0.37	72.37	0.38	27.64	2.64	22.09	3.56	20.45	3.92
	NB	93.71	2.11	92.11	2.48	92.30	2.46	78.27	3.62	73.44	3.89	73.13	3.96
	LR	92.73	0.52	90.12	1.36	88.34	1.36	75.14	3.76	75.36	4.58	77.27	4.69
	SVM	89.81	0.25	85.15	0.98	84.52	0.91	77.32	2.44	76.61	3.06	77.80	2.97
	DT	88.41	0.73	84.08	1.10	82.49	1.11	72.44	26.64	77.61	28.32	79.37	29.30
	RF	90.52	0.19	86.60	0.20	84.22	0.20	64.07	1.06	67.77	1.23	68.43	1.37
Memory	ARIMA	100.0	0.00	100.0	0.00	100.0	0.00	100.0	0.00	99.62	0.02	99.24	0.05
	NB	97.20	14.43	97.20	14.43	97.20	14.43	97.20	14.43	97.20	14.46	97.21	14.47
	LR	100.0	1.20	100.0	1.20	100.0	1.20	100.0	1.20	100.0	1.28	100.0	1.35
	SVM	100.0	0.66	100.0	0.66	100.0	0.66	100.0	0.66	100.0	0.70	100.0	0.77
	DT	100.0	0.01	100.0	0.01	100.0	0.01	100.0	0.01	99.89	0.16	99.63	0.26
	RF	100.0	0.00	100.0	0.00	100.0	0.00	100.0	0.00	99.89	0.02	99.63	0.07
Disk	ARIMA	99.25	0.15	98.33	0.22	97.61	0.35	97.30	0.38	96.31	0.64	95.67	0.90
	NB	99.85	3.56	99.72	3.66	99.82	3.64	99.72	3.82	99.42	5.09	98.69	5.58
	LR	99.85	1.78	99.67	1.87	99.39	1.92	98.21	2.02	98.83	2.52	98.30	2.70
	SVM	99.85	1.61	99.77	1.60	99.53	1.60	99.26	1.61	98.61	2.06	97.90	2.08
	DT	99.76	0.29	99.68	0.51	99.48	0.68	99.26	0.83	98.69	1.32	98.02	1.32
	RF	99.38	0.25	99.09	0.38	99.05	0.44	98.98	0.36	97.79	0.40	96.81	0.38

- For CPU, ARIMA performs poorly compared to the other machine learning methods. But it performs well for disk and memory. As mentioned in IV-A2, this is because the predictions of the ARIMA model depending on time periodicity and trend, where as CPU usage changes significantly outside of periodic changes. So it is difficult to characterize its periodicity and hence the FDR is relatively low. In this paper, we do not distinguish between periodic and non-periodic samples, that is, training models that are not specific to a particular category.
- For memory, no matter what kind of models, the FDR is close to 1 and FAR is close to 0. The reason is that our experiment is aimed at the workload of a database service-based data center, and the applications running on the servers occupy certain amount of memory, so the usage changes little and makes it easier to train a model with good predictions.
- We notice that the CPU's FDR drop drastically in ARIMA and RF, and the CPU's FAR of DT model suddenly rise. This appears to be caused by the nature of the CPU usage, as the CPU's daily usage cycle is strongly dependent on the tasks running on the server, which varies from day to day. In addition, RF improves the shortcoming of easily overfitting of DT. Although the FAR of RF is lower than other models, but the FDR of RF is also affected.

We count the number of overloads in the samples of the workload for the six predicted time point targets, see Table III. In order to more intuitively show the trend of changes in the actual overload data, we collect data detailing the total number of servers that experience an overload and depict as a line chart in Figure 3. In this figure, we can see that the total number of servers with CPU overload increases over time. This is because CPU workload changes rapidly, and the number of servers that are overloaded in one day increases. But after one day, the increase is slower. This means that the servers that have CPU overload in the future will be different to the servers presently experiencing CPU overload. For memory and disk, the number of overloads increases slowly. This is because the changes are typically more gradual compared with the CPU changes. For example, the applications running on the server will often use a consistent amount of memory. The disk may have large amounts of data written to it at one time that leads

TABLE III: Positive (P) and Negative (N) samples of a test dataset

Target	Class	Workload		
		CPU	Memory	Disk
one hour	P	23	105	206
	N	2160	2078	1977
six hours	P	26	105	208
	N	2157	2078	1975
twelve hours	P	27	105	213
	N	2156	2078	1970
one day	P	84	105	218
	N	2099	2078	1965
two days	P	103	105	224
	N	2080	2078	1959
three days	P	108	106	228
	N	2075	2077	1955

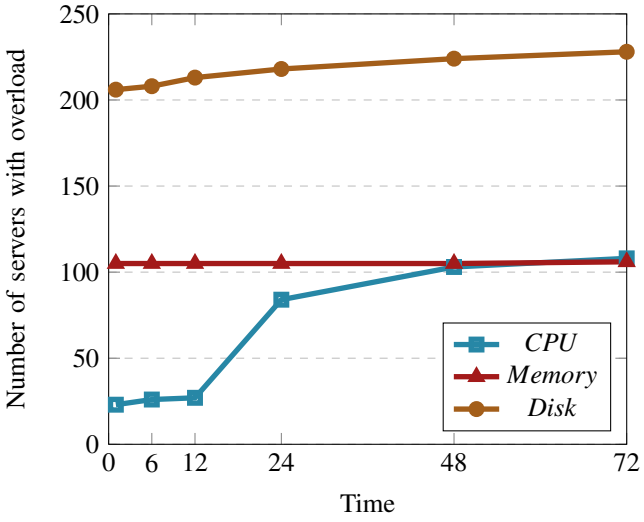


Fig. 3: Overloaded Quantity

to overload, and hence is slightly less stable than the memory.

2) evaluation system: For enterprises, workload forecasting is not focused primarily on FDR and FAR, as they are more concerned about the cost of forecasting which is diverse due to the difference of types and handling times of errors in practical business. After the workload prediction, data migration, data storage, and task migration may occur. Migration consumes bandwidth while storage takes up storage space. For Type1 predictions as in Table I where an overload is predicted, the system handles migration and storage work intelligently in a planned manner (such as at night or when the workload is low enough), while as for Type2 predictions where an overload is not predicted but actually occurs, the processing is unplanned and so data recovery may be performed under a heavy workload.

There are four kinds of model results. No treatment is taken for TN. The migration and storage are efficient and non-destructive for TP but invalid and wasteful for FP. For FN, their migration and storage efforts are detrimental to the system and likely to prove costly.

Although different servers process data differently, in order to compare the model results, we assume that the size of data which needs to be processed during an overload is the same as that of the actual server, and that we can complete the troubleshooting before an actual failure, that is, the migration of data and tasks as well as the storage of data. We consider several representations:

1) Type1 storage cost: α_1;
2) Type1 migration cost: β_1;
3) Type2 storage cost: α_2;
4) Type2 migration cost: β_2.

As Type2 work must be done under unplanned conditions or even a high workload, we can assume $\alpha_2 > \alpha_1$, $\beta_2 > \beta_1$.

$$\begin{aligned} \text{Cost} &= \alpha_1 \times \text{TP} + \beta_1 \times \text{TP} + \alpha_1 \times \text{FP} + \beta_1 \times \text{FP} \\ &\quad + \alpha_2 \times \text{FN} + \beta_2 \times \text{FN} \\ &= (\alpha_1 + \beta_1) \times \text{P} \times \text{FDR} + (\alpha_1 + \beta_1) \times \text{N} \times \text{FAR} \\ &\quad + (\alpha_2 + \beta_2) \times \text{P} \times (1 - \text{FDR}) \end{aligned} \tag{6}$$

We apply the above evaluation system shown in Formula (6) in our experimental results. Due to the special circumstances of memory, that is the memory changes infrequently, we do not evaluate it quantitatively. The positive and negative samples for CPU and memory are shown in Table III. Since the cost of missing hits (FN) is much higher than the cost of the normal predicting, and the cost of storage will probably cost more than migration, we set the storage cost ratio to be 5 to 1, and the migration cost ratio to be 3 to 1, that is $\alpha_2 = 5 \times \alpha_1$ and $\beta_2 = 3 \times \beta_1$. According to [37] and [38], we set the cost of storage α_1 to be 100 USD and the cost of migration β_1 to be 80 USD respectively. Table IV and Table V tabulate the overload processing cost per TB.

TABLE IV: CPU Overload Processing Cost (K USD)

models	one hour	six hours	twelve hours	one day	two days	three days
ARIMA	7.950	10.00	10.51	59.13	76.81	82.21
NB	13.14	15.46	15.58	39.00	48.43	50.50
LR	7.114	11.40	11.90	41.03	49.92	50.72
SVM	6.4096	10.63	10.95	35.00	43.49	43.96
DT	8.465	11.27	11.83	128.7	137.5	141.4
RF	6.100	7.388	8.003	36.02	41.75	43.64

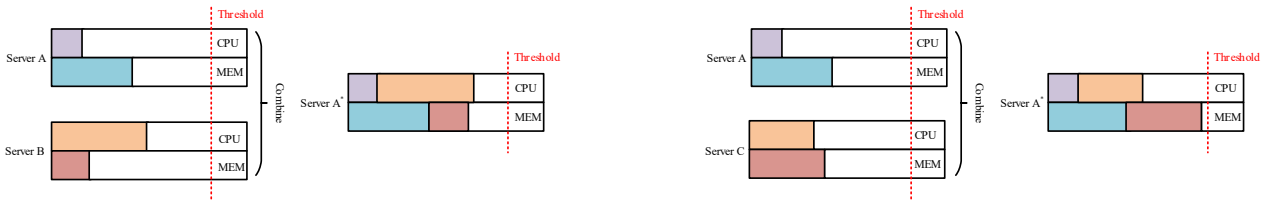
TABLE V: Disk Overload Processing Cost (K USD)

models	one hour	six hours	twelve hours	one day	two days	three days
ARIMA	38.38	40.17	42.43	43.88	47.21	59.74
NB	49.93	50.78	51.52	53.08	59.02	62.33
LR	43.59	44.47	45.89	47.33	50.68	52.69
SVM	42.98	43.40	44.57	45.85	49.34	51.04
DT	38.39	39.61	41.37	43.08	46.60	48.20
RF	38.69	39.83	41.03	41.76	44.55	46.45

From the cost calculation results, we found that for CPU, the performance of RF is significantly better than ARIMA's, and for disk, the gap of advantage narrows, but it is still significantly better than ARIMA except the result of one hour. As CPU has large fluctuations while disk is relatively stable, ARIMA is more suitable for periodic and trend-obvious time series while the machine learning method RF is more suitable for the fluctuating time series.

D. Discussion

The enterprises want to achieve a higher utilization rate to improve the performance of data center and reduce cost. Therefore, in addition to feeding back predictions to systems



(a) Scheduling on Different Types of Load

(b) Scheduling on Similar Types of Load

Fig. 4: Examples of Scheduling Strategies

and operations engineers for scheduling and performing necessary troubleshooting, we propose some server task scheduling strategies based on the database for these data centers in two application scenarios as shown in Figure 4 shown to achieve maximum utilization of limited resources.

1) Merge server tasks based on different types of workload utilization. Consider the case that two or more servers emphasize different types of resources shown in Figure 4 (a). Server A is the main memory consumer and uses less CPU, while server B is mainly responsible for CPU computing and data processing but uses less memory, we can combine these tasks to one server, thereby sparing another server to run other applications and reducing resource wastage.
2) Merge server tasks based on similar workload utilization. If two or more servers mainly consume the similar resources such as server A and server C shown in Figure 4 (b), the tasks on these two servers can be consolidated to one server, assuming that the total resource utilization does not exceed the threshold. So the servers can operate stably and efficiently, reducing resource consumption.

In addition, the applications running on data centers based on database services are mostly fixed, and have higher memory utilization as well as lower CPU utilization. It would be ideal to find other data centers based on other kinds of services and primarily consuming CPU resources, so that we could combine tasks to increase the utilization of data center resources and reduce the cost of enterprises correspondingly.

V. Conclusion and Future Work

In this paper, after the data was cleaned up, we collect 67 days of data from 5466 servers in order to analyze the data of data centers that are based on database services from a real, large-scale enterprise. Six different models are applied to predict the workload, which provide intuitive and accurate results in advance about whether the workload is going to be overloaded or not, and help the operations engineers handle them in a timely manner. In order to better meet the enterprises' requirement of minimizing the cost of prediction, we put forward an evaluation system based on the cost of overload processing. Experimental results show that machine learning methods, especially RF, have good predicting performance for CPU and disk. In the discussion section, we propose several scheduling strategies for data centers and suggest that enterprises should jointly utilize different service based servers according to the different emphasis of workload to reduce resource wastage, and achieve intelligent operation and maintenance in data centers.

We do not carry out actual and relevant experimental analysis for the proposed scheduling strategies. In future work, we consider experiments to verify the impact of scheduling strategies in real data centers.

VI. Acknowledgments

This work is partially supported by NSF of China (grant numbers: 61602266, 11550110491), Science and Technology Development Plan of Tianjin (17JCYBJC15300, 16JCYBJC41900) and the Fundamental Research Funds for the Central Universities (Grant number: 65141020).

We would like to thank the Baidu Online Network Technology Co. Ltd for providing the real workload datasets from the monitoring system. In particular, we thank Wang Long, Wang Jianxun and Yu Jie for providing valuable information in data preparation, and thanks to Rebecca J. Stones and Liu Dongshi's suggestions for this paper.

References

[1] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," *Eprint Arxiv*, vol. 12, no. 4, pp. 6–17, 2010.

[2] A. M. Ferreira and B. Pernici, "Managing the complex data center environment: an integrated energy-aware framework," *Computing*, vol. 98, no. 7, pp. 709–749, 2016.

[3] Y. Sverdlik, "Delta: Data center outage cost us $150m," http://www.datacenterknowledge.com/archives/2016/09/08/delta-data-center-outage-cost-us-150m, 2016.

[4] J. Weckworth, "The lack of transparency can be seen as a root cause of outages and incidents," https://journal.uptimeinstitute.com/data-center-outages-incidents-industry-transparency/, 2013.

[5] J. Moore, J. Chase, K. Farkas, and P. Ranganathan, "Data center workload monitoring, analysis, and emulation," in *Eighth Workshop on Computer Architecture Evaluation using Commercial Workloads*, 2005, pp. 1–8.

[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[7] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[8] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using arima model and its impact on cloud applications qos," *IEEE Transactions on Cloud Computing*, vol. 3, no. 4, pp. 449–458, 2015.

[9] V. Debusschere, S. Bacha *et al.*, "Hourly server workload forecasting up to 168 hours ahead using seasonal arima model," in *IEEE International Conference on Industrial Technology*, 2012, pp. 1127–1131.

[10] G. Dhiman, K. Mihic, and T. Rosing, "A system for online power prediction in virtualized environments using gaussian mixture models," in *Design Automation Conference, 47th ACM/IEEE*, 2010, pp. 807–812.

[11] Y. Lu, J. Panneerselvam, L. Liu, and Y. Wu, "Rvlbpnn: A workload forecasting model for smart cloud computing," *Scientific Programming*, vol. 2016, 2016.

[12] B. Bican and Y. Yaslan, "A hybrid method for time series prediction using emd and svr," in *Communications, Control and Signal Processing*, 2014, pp. 566–569.

[13] T. Guo, Z. Xu, X. Yao, H. Chen, K. Aberer, and K. Funaya, "Robust online time series prediction with recurrent neural networks," in *Data Science and Advanced Analytics*, 2016, pp. 816–825.

[14] S. Aditham, N. Ranganathan, and S. Katkoori, "LSTM-based memory profiling for predicting data attacks in distributed big data systems," in *Parallel and Distributed Processing Symposium Workshops*, 2017, pp. 1259–1267.

[15] M. M. Fernández, Y. Yue, and R. Weber, "Telemetry anomaly detection system using machine learning to streamline mission operations," in *Space Mission Challenges for Information Technology*, 2017, pp. 70–75.

[16] Y. Imamverdiyev and L. Sukhostat, "Anomaly detection in network traffic using extreme learning machine," in *Application of Information and Communication Technologies*, 2016, pp. 1–4.

[17] T. Salman, D. Bhamare, A. Erbad, R. Jain, and M. Samaka, "Machine learning for anomaly detection and categorization in multi-cloud environments," in *Cyber Security and Cloud Computing*, 2017, pp. 97–103.

[18] A. Abubakar and B. Pranggono, "Machine learning based intrusion detection system for software defined networks," in *Emerging Security Technologies*, 2017, pp. 138–143.

[19] L. Wei, W. Luo, J. Weng, Y. Zhong, X. Zhang, and Z. Yan, "Machine learning-based malicious application detection of android," *IEEE Access*, vol. 5, pp. 25 591–25 601, 2017.

[20] M. Stokely, A. Mehrabian, C. Albrecht, F. Labelle, and A. Merchant, "Projecting disk usage based on historical trends in a cloud environment," in *Scientific Cloud Computing*, 2012, pp. 63–70.

[21] M. Stokely and A. Merchant, "Storage provisioning and allocation in a large cloud environment," in *Management of big data systems*, 2012, pp. 35–36.

[22] G. M. Wamba, Y. Li, A.-C. Orgerie, N. Beldiceanu, and J.-M. Menaud, "Cloud workload prediction and generation models," in *Computer Architecture and High Performance Computing*, 2017, pp. 89–96.

[23] J. Xue, F. Yan, R. Birke, L. Y. Chen, T. Scherer, and E. Smirni, "Practise: Robust prediction of data center time series," in *Network and Service Management*, 2015, pp. 126–134.

[24] İ. Karakurt, S. Özer, T. Ulusinan, and M. C. Ganiz, "A machine learning approach to database failure prediction," in *Computer Science and Engineering*, 2017, pp. 1030–1035.

[25] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Operating Systems Principles*, 2017, pp. 153–167.

[26] Y. Hu, B. Deng, F. Peng, and D. Wang, "Workload prediction for cloud computing elasticity mechanism," in *Cloud Computing and Big Data Analysis*, 2016, pp. 244–249.

[27] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[28] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, pp. 1157–1182, 2003.

[29] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[30] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, pp. 660–674, 1991.

[31] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[32] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE transactions on pattern analysis and machine intelligence*, pp. 832–844, 1998.

[33] I. Ojo, "Autoregressive integrated moving average," *Asian Journal of Mathematics and Statistics*, pp. 225–236, 2010.

[34] K. P. Murphy, "Naive bayes classifiers," *University of British Columbia*, 2006.

[35] A. Agresti, *Logistic regression*. Wiley Online Library, 2002.

[36] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, pp. 293–300, 1999.

[37] I. Manousakis, S. Sankar, G. McKnight, T. D. Nguyen, and R. Bianchini, "m," in *FAST*, 2016, pp. 53–65.

[38] "Bandwidth pricing details of microsoft azure," https://azure.microsoft.com/en-us/pricing/details/bandwidth/, 2017.