Private cloud resource prediction based on combinatorial optimization algorithm

Mengxin Song*[a], Ruicai Shen[b], Hongping Miao[a], Zixuan Li[a]
[a]Research Institute of Petroleum Exploration & Development, CNPC, Beijing 100083, China;
[b]Beijing Yuexin Times Technology Co., Ltd., Beijing, China

ABSTRACT

In private cloud environments, resource usage is determined based on experience, which often leads to excessive resource allocation and low resource utilization. This paper proposes a private cloud resource consumption prediction algorithm based on combinatorial optimization algorithm. Based on the virtual machine performance data collected from the private cloud platform, the key features are selected by recursive feature elimination method (SVM-RFE), and then the model is trained by eXtrem Gradient Boosting (XGBoost) algorithm for model training to predict the resource usage. Compared with Random Forest, LSTM and other algorithms, the proposed algorithm has higher prediction accuracy and smaller prediction error. This paper adopts a data-driven approach to achieve intelligent prediction of resource usage in private cloud environments, which improves resource utilization and provides decision support for optimal allocation of resources.

Keywords: Private cloud, time-sequence forecasting, resource usage prediction, SVM-RFE, XGBoost algorithm, combinatorial optimization

1. INTRODUCTION

In recent years, the cloud platform has attracted widespread attention because of its cheap and extensible computing resource, stable and high capacity storage room, flexible access anytime and anywhere and many other advantages. To increase the utilization of resources in the cloud platform, the researchers have proposed many scheduling algorithms[1]. By building the prediction model based on the historical data, it could predict the resource consumption in future. However, with the increase demanding, the requirements for the accuracy prediction of future resource consumption are also becoming higher and higher. The bigger the forecast error, the heavier waste of the resources. How to reduce the waste of private cloud resource is becoming one of the most important problems faced today. Similarly, how to predict the resource consumption accurately is also becoming the key point for the research in private cloud field.

In order to predict the resource usage of cloud platform, many scholars have conducted a series of studies. Delimitrou et al.[2] proposed to analyze the workload offline in 2014. First, they let the task run for a while, and got the possible resource utilization through the recommended algorithm, and then rescheduled. In fact, offline analysis is not workable, because in the actual production procession, the workload input before the running of the tasks is usually unavailable. Cortez et al.[3] conducted a detailed analysis of the resource consumption of the cloud platform—Microsoft Azure in 2017, indicating the low resource consumption and the predictability of cloud platform, and used the random forest regression model to predict the resource consumption of virtual machines, but the performance is not good enough. Mehmood et al.[4] in 2018 classified the resource utilization of tasks in Google Cloud Platform into high, medium and low levels and used classification techniques to predict the resource utilization levels of tasks, but this classification is based on coarse granularity and does not effectively help the scheduler to allocate resources.

Most of the above methods are for public resources and use data analysis to study the public cloud resource usage or use other models to predict the resource usage. However, both the coarse and fine granularity of prediction and the accuracy of prediction results do not meet the demand for resource applications at this stage. Based on the characteristics of private cloud platform, this paper proposes a prediction model of private cloud resource usage based on the combinatorial optimization algorithm, which filters the existing data with features closely related to resource usage. The XGBoost algorithm-based resource usage prediction model is built for the filtered features. The strong generalization ability of the model is easily seen by using evaluation index.

* mxsong@petrochina.com.cn

Third International Conference on Computer Science and Communication Technology (ICCSCT 2022)
edited by Yingfa Lu, Changbo Cheng, Proc. of SPIE Vol. 12506, 125060M
© 2022 SPIE · 0277-786X · doi: 10.1117/12.2662480

2. INTRODUCTION TO ALGORITHM MODEL

2.1 Recursive feature elimination method based on supporting vector machine

In the process of data mining, data processing is usually required, we call it the data pre-processing stage. Appropriate data processing is very important to improve the performance of the model. Generally speaking, data pre-processing consists of removing dirty data, sorting and transforming data, and data streamlining[5]. Besides of the commonly used data pre-processing methods, a cyclic method of removing features (SVM-RFE) is used to remove unnecessary features. Guyon et al.[6] proposed the SVM-RFE method in 2002. This algorithm is a wrapper feature screening method with high performance, which includes the strategy of removing backward features and SVM (Support Vector Machine). The core of the algorithm is to sort all the features according to the impact on the target. The smaller the impact, the sooner the features will be deleted, and then remodel the remaining features. In this way, the iterative cycle will continue until the number of features is empty. The SVM-RFE algorithm is shown in Table 1.

Table 1. Workflow of SVM-RFE algorithm.

Algorithm: SVM-RFE algorithm
Input: Training data D
Output: Best_Features subset
Process
Start
(1) The initialization of the algorithm is that the current feature subset current_features contain all Features and the optimal feature subset Best_Features is empty;
(2) A number is set which indicates the number of features k removed in each iteration;
(3) The following steps are run in a loop until the current features subset is empty;
(4) The SVM model is constructed according to the Current_Features and the accuracy *w* is obtained. Then the value of w is sorted in descending order. Finally, the last *k* features in the current features are removed;
(5) The best feature subset Best_Features returns.

2.2 XGBoost algorithm

XGBoost (eXtrem Gradient Boosting, also known as Extreme Gradient Lifting Algorithm)[7] is an improved method based on gradient lifting decision tree. In order to improve the accuracy of the algorithm, the algorithm is trained to generate a new tree to fit the residual of the previous tree. The algorithm has been widely used in various fields since it was proposed[8,9]. The novelty of the algorithm is to build a base learner and add a new learner based on this learner. The new learner fits the previous residuals, which ensures that the objective function value decreases gradually in each iteration. The change of the objective function when simulated to add a new learner as equation (1):

$$\begin{cases} \hat{y}_i^{(0)} = 0 \\ \hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\ \hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\ \hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i) \end{cases} \tag{1}$$

In this formula, $\hat{y}_i^{(t)}$represents the prediction result of the *t*th time, and its value is the sum of the previous result and the new function $f_t(x_i)$. If there are too many leaf nodes in the decision tree, the risk of the model falling into overfitting increases. Therefore, a penalty term should be set to limit the number of leaf nodes. The expression of the penalty term is shown in equation (2).

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \quad (2)$$

In this formula, γ is the punishment;λ stands for a parameter, ω represents the weight of each leaf node; T denotes the number of leaf nodes. The above equation shows that the higher the number of nodes, the higher the penalty. The objective function with penalty term is shown in equation (3).

$$Obj^{(t)} = \sum_{i=1}^{n} l((y_i, \hat{y}^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) = \sum_{i=1}^{n} l[y_i, \hat{y}_i^{(t-1)} + f_t(x_i)] + \Omega(f_i) + C \quad (3)$$

To approximate the objective function a Taylor expansion is used, which takes the form shown in equation (4):

$$\mathrm{f}(\mathrm{x} + \Delta\mathrm{x}) \approx \mathrm{f}(\mathrm{x}) + \acute{f}(x)\Delta\mathrm{x} + \frac{1}{2}\acute{\acute{f}}(\mathrm{x})\,\Delta\mathrm{x}^2 \quad (4)$$

The following representation is derived from the Taylor expansion form of equation (4) and the objective function of equation (3):

$$\begin{aligned} \Delta\mathrm{x} &= f_t(x_i) \\ \mathrm{f}(\mathrm{x}) &= l\left(y_i, \hat{y}_i^{(t-1)}\right) \\ g_i &= \sigma_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right) \\ h_i &= \sigma_{\hat{y}_i^{(t-1)}}^2 l\left(y_i, \hat{y}^{(t-1)}\right) \end{aligned} \quad (5)$$

In this formula, g_i represents the first-order derivative form of the Taylor expansion; h_i represents the second-order derivative form of the Taylor expansion, which can be regarded as a constant because the value of $l\left(y_i, \hat{y}^{(t-1)}\right)$ is fixed. Therefore, the objective function can be expressed by equation (6):

$$Obj^{(t)} \approx \sum_{i=1}^{t} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \quad (6)$$

3. PRIVATE CLOUD RESOURCE PREDICTION BASED ON COMBINATORIAL OPTIMIZATION ALGORITHM

3.1 Data collection and pre-processing

The experimental data is from the monitored unit notes and the index data related to the memory consumption. By installing Node-Exporter collector in different types of targeted servers, it can regularly collect the target data to the influxdb time series database, group the nodes and filter the data on Prometheus, and finally presented in the form of charts through Grafana. The experimental data covers 47 pieces of relevant indicators from 2022/2/21 10:51:52 to 2022/4/24 23:59:47, with a total of 354408 pieces of data. The relevant indicator data is shown as Table 2.

Due to the inconsistent format of the collected data, anomalies and redundancy of sample features, the data needs to be processed as necessary to ensure that the model can recognize the collected data. The main processing steps include file parsing, missing values and abnormal value processing, feature selecting and data merging.

(1) Data analysis

This part mainly parses the collected raw data into a recognizable fixed format, and the specific parsing steps include conversion of timestamp and time, separation of data, filtering of redundant information and data rearrangement, etc.

(2) Missing value and abnormal value processing

This part is mainly to process the existing special data which mainly includes abnormal data and missing data. The specific processing method is to get the average value of the 4 data point s around the abnormal data, then fill in the obtained value. The reason to do so is because that the time interval of data collection is 15 seconds. Having analyzed the data pattern, we found that the data will not change strongly within 1 minute.

(3) Feature selection

The collected memory data is as high as 47 pieces, to reduce numbers of features input into the model, the paper adopts SVM-RFE method which is introduced in Section 2.1 to filter the feature. The target data memory consumption could be expressed by equation (7).

$$Y = [1 - \mathrm{memory}_{\mathrm{MemAvailable}_{\mathrm{bytes}}}(memory_{MemTotal_{bytes}})^{-1}] \times 100 \tag{7}$$

In this formula, *y* represents memory occupation, memory-MemAvailable-bytes represents the available bytes of the memory, memory-MemTotal-bytes indicates total amount of the memory bytes.

Table 2. Index data related with memory nodes.

memory_Active_anon_bytes	memory_Active_bytes	memory_Active_file_bytes
memory_AnonHugePages_bytes	memory_AnonPages_bytes	memory_Bounce_bytes
memory_Buffers_bytes	memory_Cached_bytes	memory_CmaFree_bytes
memory_CmaTotal_bytes	memory_CommitLimit_bytes	memory_Committed_AS_bytes
memory_DirectMap1G_bytes	memory_DirectMap2M_bytes	memory_DirectMap4k_bytes
memory_Dirty_bytes	memory_HardwareCorrupted_bytes	memory_HugePages_Free
memory_HugePages_Rsvd	memory_HugePages_Surp	memory_HugePages_Total
memory_Hugepagesize_bytes	memory_Inactive_anon_bytes	memory_Inactive_bytes
memory_Inactive_file_bytes	memory_KernelStack_bytes	memory_Mapped_bytes
memory_MemAvailable_bytes	memory_MemFree_bytes	memory_MemTotal_bytes
memory_Mlocked_bytes	memory_NFS_Unstable_bytes	memory_PageTables_bytes
memory_Percpu_bytes	memory_Shmem_bytes	memory_Slab_bytes
memory_SReclaimable_bytes	memory_SUnreclaim_bytes	memory_SwapCached_bytes
memory_SwapFree_bytes	memory_SwapTotal_bytes	memory_Unevictable_bytes
memory_VmallocChunk_bytes	memory_VmallocTotal_bytes	memory_VmallocUsed_bytes
memory_Writeback_bytes	memory_WritebackTmp_bytes	

Eventually, we derived 9 features which have high correlation with the target features. The detailed features are expressed as Table 3.

Table 3. The features after the feature selection.

Name of features
memory_Buffers_bytes
memory_Inactive_anon_bytes
memory_Inactive_bytes
memory_Inactive_file_bytes
memory_MemAvailable_bytes
memory_Shmem_bytes
memory_Slab_bytes
memory_SReclaimable_bytes
memory_SUnreclaim_bytes

(4) Data merging

In this stage, it is mainly to merge the filtered data. The experimental data is monitored every 15 seconds. In actual use, such fine-grained data changes will be neglected. Thus, the data will be merged in the experiment for 10 minutes, that is to say the data which is obtained within 10 minutes will be merged into a new data.

3.2 Combinatorial optimization model building & application

The device of this experiment is NVIDIA Quadro P4000 GPU, Python3.7.3, sklearn, and other kits. The resource consumption prediction model is built based on XGBoost algorithm. The filtered features and the memory consumption are taken as the already known data to predict the memory occupation in the next week. Seventy percent of the processed data will be identified as the training set and the remaining as the test set. The experimental chart is shown in Figure 1.

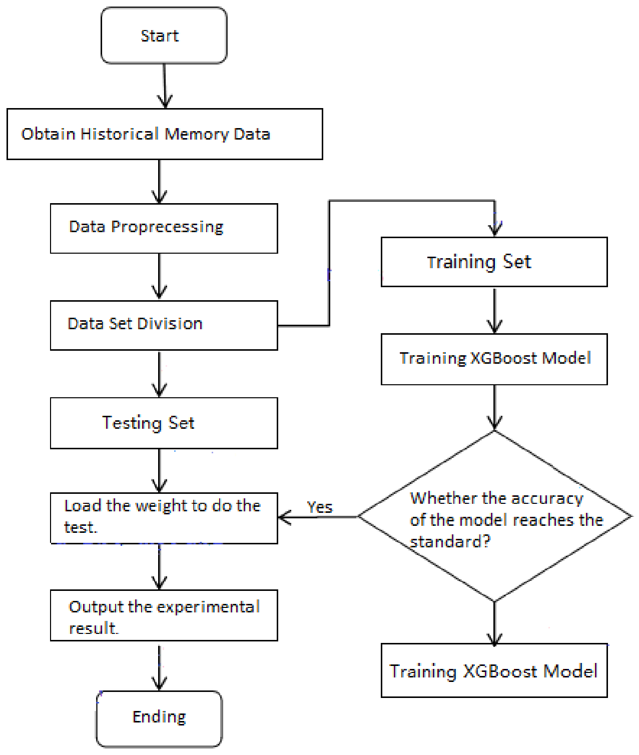

Start
Obtain Historical Memory Data
Data Proprecessing
Data Set Division
Testing Set
Load the weight to do the test.
Output the experimental result.
Ending
Training Set
Training XGBoost Model
Whether the accuracy of the model reaches the standard?
Yes
Training XGBoost Model

Figure 1. Workflow of the proposed algorithm.

The experiment involves a large number of parameters, and the selection of different parameters has different effects on the model. To prevent the model from falling into underfitting or overfitting caused by unreasonable parameters, the experiment adopts grid search to screen the optimal parameter combination. The parameters after grid search are learningRate=0.1, gamma =1, maxDepth=10, nEstimators=100, minChildWeight=6.

3.3. Experimental Result

The model is trained by loading the optimal combination of parameters from the grid search into the XGBoost-based memory usage prediction model, whose training results are shown in Figure 2, where the blue dashed line represents the model prediction results, the green solid line represents the real data, and the horizontal coordinate represents the time unit in hours; the ordinate represents the memory occupancy (unit is %, in which 36.0 represents the memory occupancy is 3.0%). Through the figure, it is found that the prediction result value of the memory occupation prediction model based on XGBoost is close to the real value, and the prediction effect is good.

(1) Evaluation index of prediction result

In order to evaluate the model result and better estimate the discrepancy between the real result and the prediction result, the following performance indicators are adopted to measure the model. Absolute mean error (MAE) and determination (R^2) and Root mean square error (RMSE). The closer the R^2 to 1, the model performance is better. The definition for each index formula is shown as equations (8)-(10).

$$\text{RMSE} = \left(\frac{1}{m}\sum_{i=1}^{m}(y_i - \acute{y}_i)^2\right)^{\frac{1}{2}} \tag{8}$$

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}|y_i - \acute{y}_i| \tag{9}$$

$$R^2 = 1 - \sum_{i=1}^{m}(y_i - \acute{y}_i)^2 \left(\sum_{i=1}^{m}(y_i - \bar{y})^2\right)^{-1} \tag{10}$$

In this formula, y_i represents true value; $\acute{y}_i$ represents predicted value; $\bar{y}$ represents the average; m is the number of the testing samples.

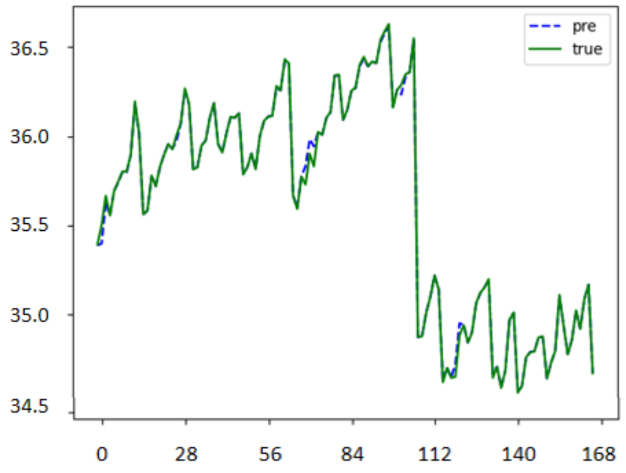
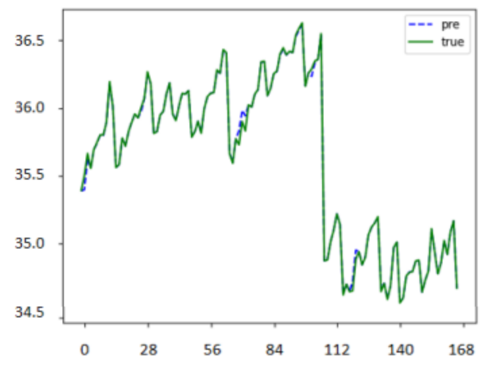


Figure 2. Prediction model result diagram of memory occupation based on XGBoost algorithm.

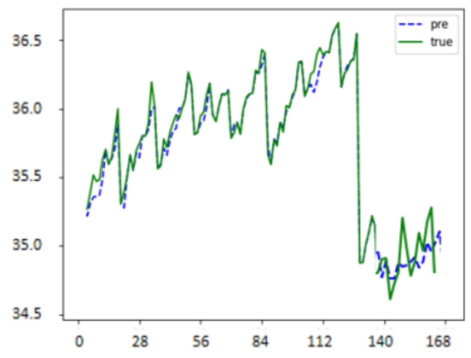
(2) Experimental analysis

To compare the performance of the model used in this paper with that of the co-directional model, the comparative experiments of Long Short-Term Memory (LSTM) algorithm, and Random Forest (RF) method[10] were carried out. The result is shown in Figure 3. Similarly, in each figure, x-axis represents the time unit as hour; vertical-axis represents the memory consumption data (unit is %). The blue dashed line represents the model prediction results, the solid green line indicates the real data; As can be seen from the figure, the prediction results of the model proposed in this paper are better than the other two models. Because of the integrated learning idea in RF, the data prediction result is relatively small. Due to the mechanism of LSTM model, when the model is far from the target value, the data memory is poor, thus the error is larger. Therefore, the prediction result of the model in such a long time in this paper is not good.

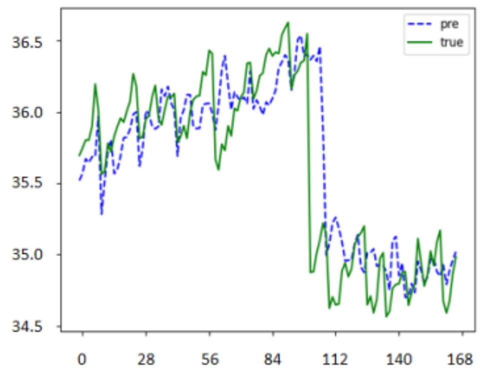
As shown in Table 4, the root mean square error, the absolute mean error, and the R^2 evaluation index of both RF-based and LSTM-based prediction models are inferior to the XGBoost model used in this paper for memory usage prediction.



(a)



(b)



(c)

Figure 3. Comparison of the prediction result of different models. (a): XGBoost model result; (b): RF model result; (c): LSMT model result.

Table 4. Comparison of The Different Models Under Different Indexes

Model	RMSE	MAE	R^2
XGBoost	3.81	0.61	0.976
RF	5.32	1.31	0.813
LSTM	7.17	3.1	0.636

4. SUMMARY AND PROSPECT

By collecting and processing related data of private cloud resources, this paper proposes a resource consumption prediction algorithm based on combinatorial optimization algorithm, which solves the problems of coarse granularity and low prediction accuracy of resource consumption in previous studies. Through comparative analysis, the proposed combinatorial optimization algorithm has more accurate prediction effect and lower prediction error compared with RF algorithm and LSTM algorithm. The combined optimization algorithm improves the application efficiency of resources in private cloud environments and lays the foundation for intelligent recommendation of resource usage.

REFERENCES

[1]Bi, J., Yuan, H. T., Tan, W., et al., "Application-aware dynamic fine-grained resource provisioning in a virtualized cloud data center," IEEE Transactions on Automation Science and Engineering 14(2), 1172-1184 (2017).
[2]Delimitrou, C. and Kozyrakis, C., "Quasar: Resource-efficient and QoS-aware cluster management," Proc. of Inter. Conf. on

Architectural Support for Programming Languages and Operating Systems, 127-144 (2014).

[3]Cortez, E., Bonde, A., Muzio, A., et al., “Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms,” Proc. of the 26th Symp. on Operating Systems Principles, 153-167 (2017).

[4]Mehmood, T., Latif, S. and Malik, S., “Prediction of cloud computing resource utilization,” Proc. of the 15th Inter. Conf. on Smart Cities: Improving Quality of Life Using ICT & IoT (HONET-ICT), 38-42 ((2018).

[5]Mundra, P. A. and Rajapakse, J. C., “SVM-REF with MRMR filter for gene selection,” IEEE Transactions on Nanobioscience 9(1), 31-37 (2010).

[6]Guyon, I., Weston, J., Barnhill, S., et al., “Gene selection for cancer classification using support vector machines,” Machine Learning 46(1-3), 389-422 (2002).

[7]Chen, T. and Guestrin, C., “XGBoost: A scalable tree boosting system,” Proc. of the 22nd ACM SIGKDD Inter. Conf. on Knowledge Discovery and Data Mining, 785-794 (2016).

[8]Seyfioğlu, M. and Demirezen, M., “A hierarchical approach for sentiment analysis and categorization of Turkish written customer relationship management data,” Proc. of the Federated Conf. on Computer Science and Information Systems, 361-365 (2017).

[9]Athanasiou, V. and Maragoudakis, M., “A novel, gradient boosting framework for sentiment analysis in languages where NLP resources are not plentiful: A case study for modern Greek,” Algorithms 10(1), 34 (2017).

[10] Breiman, L., “Random forests,” Machine Learning 45(1), 5-32 (2001).