Cloud Data Center Resource Demand Prediction Model Development on Apache Spark

Moh Moh Than[1], Thandar Thein[2]

University of Computer Studies, Yangon[1], University of Computer Studies (Maubin)[2]

mohmohthan@ucsy.edu.mm[1], thandartheinn@gmail.com[2]

Abstract

Dynamic resource allocation in cloud data centers is a challenging problem. Resource prediction is a key feature for on-demand resource planning and efficient resource management of dynamic workload. This requires a highly accurate demand prediction. Hyper-parameter optimization can largely affect the performance of the prediction model. The process of identifying the optimal parameters for a machine learning (ML) algorithm involves the search for a broad range of value combinations of parameter sets. This paper presents a resource demand prediction model with the cloud computational frameworks Apache™ Hadoop® and Apache Spark™. The model is developed on the powerful ML technique, Decision Tree (DT) algorithm, and hyper-parameter optimization for DT algorithm is performed to achieve the prediction model with high accuracy. The evaluation of prediction model is conducted on real data center workload traces and the evaluation results show that hyper-parameter optimization can save the prediction error significantly.

Keywords**:** *Apache Hadoop, Apache Spark, Hyper-parameter Optimization, Machine Learning, Resource Prediction*

1. Introduction

Cloud computing allows contemporary business owners to rent and use resources or services needed to run their businesses in a pay-as-you-go manner [1]. Cloud data centers provide low-cost, flexible and powerful executing environment for users. The resource demand to data centers is often dynamic, changing as the overall data centers workload. The cloud provider must ensure they have enough resources to meet the resource amount needed for incoming requests. Accurate resource demand prediction is essential to allocate resources dynamically and effectively.

Machine learning is mostly used to predict the resource usage by different tasks under current workloads. It is a subfield of data mining area charge for modeling systems from real examples of their past behavior. These models can then be used to predict future behaviors. Modern supervised ML algorithms involve hyper-parameters which define the model architecture that have to be set before running them. An important task in ML is model selection to find the best parameters for the model of a given task. Hyper-parameter optimization can largely affect the predictive performance of ML algorithms [2].

In this paper, DT algorithm is applied to develop the CPU resource demand prediction model for the cloud data center. To find the model that can predict more accurate result, the performance of DT algorithm is enhanced by hyper-parameter optimization. Apache Spark is used as backend processing engine, as it is better suited for iterative applications, such as Data Mining and Machine Learning [3].

The rest of this paper is organized as follows. Section 2 describes literature review, section 3 presents preliminary, section 4 discusses details about resource demand prediction model development, section 5 presents experiments and results discussion, and finally section 6 provides concluding remarks.

2. Literature Review

Resource demand prediction is needed for efficient resource management of dynamic workload for cloud data centers. Several researches concerning cloud data center resource demand prediction with different ML algorithms are discussed in this section.

The authors [4] investigated three different methods: Linear Regression (LR), Adaptive Neuro-Fuzzy Inference Systems (ANFIS), and Nonlinear Autoregressive Network (NAR) to predict CPU-cores and memory consumption in the Google cluster trace. NAR predicts more accurately (MAPE less than

7.1%) than the others and uses short time for training and prediction.

The paper [5] proposed a resource demand prediction framework to predict the CPU demand in multi-tenant service cloud environments. The proposed framework employs data mining techniques, which extract high level characteristics from historical demand behavior and provision resources in advance. They applied Polynomial regression (PR), Auto-regressive with external input (ARX) and Autoregressive moving average with external input (ARMAX) prediction techniques. They observed that ARX gives the more precise result (MSE 0.075) and less error compared to other techniques.

A. Mozo et al. [6] presented the use of convolutional neural networks (CNNs) to forecast short-term changes in the amount of traffic crossing a data center network. They validated their approach with the experiments using a dataset collected at the core network of an Internet Service Provider. They ran a set of experiments to compare the accuracy of the CNN, artificial neural networks (ANN) and Autoregressive integrated moving average (ARIMA) models by computing mean absolute error. The obtained results show that CNN models outperform both ARIMA and ANNs in both mean squared error and mean absolute error.

Although they used different ML algorithms for resource demand prediction, they didn't consider enhancing the prediction accuracy of algorithms. The process of identifying optimal hyper-parameters that enhance the performance of DT algorithm is studied in [7] to find the model that can predict more accurate result.

3. Preliminary

This section describes the experiment workload traces, DT algorithm and prediction measurement metric used for this work.

3.1. Experiment Workload Traces

There are enormous collection of workload traces (datasets) of a variety of High Performance Computing especially for Cluster and Grid Computing. Three real-life workload traces chosen for the experimentation are the DAS2 dataset (Distributed ASCI in Netherlands Supercomputer-2) with 225711 instances, the RICC dataset (RIKEN Integrated Cluster of Clusters) with 447794 instances, the MetaCentrum dataset (Czech National Grid Infrastructure MetaCentrum) with 103656 instances. All the datasets are publicly available at Parallel Workload Archive [8]. They contain trace of several thousands of jobs with continuous variables (numerical data).

3.2. Decision Tree Algorithm

Decision Tree algorithms are a popular choice, since they are robust and efficient to construct. Moreover, they have the advantage of producing comprehensible models and satisfactory accuracy levels in several application domains. Like most of the Machine Leaning algorithms, these algorithms have some hyper-parameters whose values directly affect the performance of the induced models.

DT works reasonably well with the default values of the hyper-parameters specified in software packages. Nevertheless, tuning the hyper-parameters can improve the performance of DT.

3.3. Prediction Measurement Metric

To measure the prediction accuracy, the most popular evaluation metric Mean Absolute Error (MAE) as in (1) is used. The accuracy of a predictor is estimated by computing an error based on the difference between the predicted value y_i' and the actual known value y_i for each of test instances, d is the number of instances.

$$MAE = \frac{\sum_{i=1}^{d} |y_i - y_i'|}{d} \qquad (1)$$

4. Resource Demand Prediction Model Development

The system for constructing cloud data center resource demand prediction model is illustrated in Figure 1. Resource demand prediction model predicts the amount of CPU to allocate for each request in a data center.

The proposed system to develop the CPU resource demand prediction model is implemented by using Hadoop Distributed File System (HDFS) [9], Spark processing engine and Spark Machine Learning Library (Spark MLib) [10]. It consists of three layers and the function of each layer is described as follow:

Storage Layer: HDFS is used to provide scalable and reliable data storage.

Processing Layer: Yarn Cluster Manager and Spark executor are used to process data in-parallel on

clusters of commodity hardware in a reliable and fault-tolerant manner.

Analytic Layer: To generate the resource demand prediction model, data pre-processing, hyper-parameter optimization, models development and evaluation, and model selection are performed. All of the processes from Analytic Layer are executed in parallel manner by using Spark engine. The procedures of resource demand prediction are implemented by using Spark MLib.

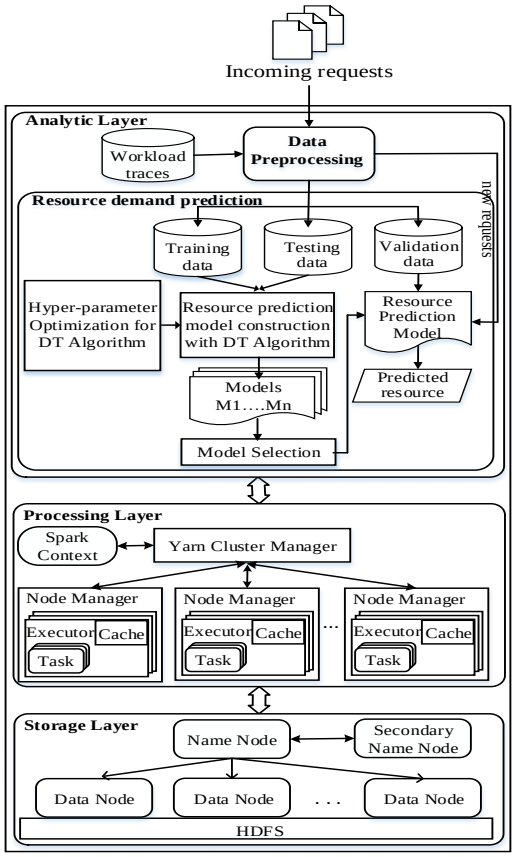

Figure 1. Proposed system architecture

4.1. Data Pre-processing

This involves pre-processing historical workload traces by filtering out unnecessary information from raw data. It removes or reduces the noise data and the treatment of missing values by replacing a missing value with the most commonly occurring value for that attribute. It identifies or removes the outliers, and also resolves inconsistencies. The pre-processed workload traces are divided into three disjoint parts: training data set needed to build (or fit) the models, testing data set used to estimate the test error for model selection and validation data set used for validating the selected model respectively.

4.2. Resource Demand Prediction

The resource demand prediction models are generated setting all possible combinations of hyper-parameter values. The prediction results of the generated models are analyzed to select the resource demand prediction model with the lowest error. After the best model selection, it is validated whether to check it can predict the correct number of the CPU demand. The procedure of resource demand prediction is presented in Figure 2.

Procedure: Resource_Demand_Prediction

Input: pre-processed workload traces
Output: predicted resource demand
1. Begin
2. Develop the prediction model by using default hyper-parameter values of DT
-Evaluate model accuracy (minMAE←MAE)
3. for(MaxDepth=0,MaxDepth<=30,MaxDepth ++)
4. for(MinInfoGain=0,MinInfoGain<=1, MinInfoGain +=0.1)
5. Develop the prediction model by using MaxDepth and MinInfoGain for DT
-Evaluate model accuracy (print MAE)
6. if (minMAE>MAE) minMAE←MAE
7. endif
8. endfor// MinInfoGain
9. endfor// MaxDepth
10. Predict resource demand by using selected model with minMAE
11. End

Figure 2. Procedure of resource demand prediction

4.2.1. Hyper-parameter Optimization

Hyper-parameter optimization for Decision Tree (DT) algorithm is operated over the training dataset intended to effective model development. This explores the optimal parameter values for the DT.

In the approach of hyper-parameter optimization, it needs to determine the parameter setting for the learning algorithm that will produce models with low error rate. One of the parameters chosen for optimization is the Maximum tree depth (MaxDepth) - the tree can grow to, as it is shown to affect DT performance [11]. The next parameter is Minimum Information Gain (MinInfoGain) - no split

candidate leads to an information gain greater than MinInfoGain [7]. The following range and step sizes for each of the selected hyper-parameter values are set for the corresponding datasets:

Maximum tree depth: A decision tree with depths can be ranged from 0 to 30. In this study, MaxDepth is set 31 different values from 0 to 30 with step size 1.

Minimum information gain: For a node to be split further, the split must improve at least this much (in terms of information gain). MinInfoGain is set 11 different values from 0 to 1, step of 0.1 for all three datasets.

These settings result in a uniform point cover of 341 (31 MaxDepth values x 11 MinInforGain values) different combination parameter sets for each dataset.

4.2.2. Prediction Models Construction

The prediction models are generated by using DT algorithm based on all possible combinations of hyper-parameters pairs and all features provided in training data. In this study, 341 models are developed by using all possible combinations of 31 MaxDepth values and 11 MinInfoGain values for each dataset with same input and different parameters. The accuracy of the generated models is evaluated on testing data.

4.2.3. Model Selection

The resource demand prediction model with the lowest error rate is selected for each dataset by analyzing the accuracy of the models generated with all possible combinations of hyper-parameter values. The best model is feed to the validation dataset whether to check it can predict the correct number of the CPU demand. The CPU demand of incoming requests will be predicted by using the finally chosen model.

5. Experiments and Results Discussion

The required system specification, the experiment results comparison and analysis are presented in this section.

5.1. Experiment Specification

In this experiment, Spark data processing engine is developed using a virtual machine. The specifications of devices and necessary software component are presented in Table 1.

Table 1. System Specification

OS	Ubuntu 16.04 LTS
Host Specification	Intel ® Core i7-7500U CPU @ 2.90GHz, 8GB Memory, 1TB Hard Disk
VM Specification	4GB RAM, 100 GB Hard Disk
Software Component	- Apache Hadoop 2.6.0 - Apache Spark 1.5.2 - Scala version 2.10.4

5.2. Comparison and Analysis

The accuracy of resource demand prediction varies depending on each set of parameter values and the characteristics of the workload dataset. Each set of all different combination parameter sets for each dataset were input to the DT algorithm to generate the prediction models based on a set of training data. The prediction performance of the considered models is evaluated on test set and MAE obtained for each model was recorded together with the set of parameter values that generated it.

Table 2. MAE results of generated models using different combinations of hyper-parameters (DAS2)

Max Depth	MinInfoGain				
	0.0	0.1	0.2	0.3	0.4
0	13.16	13.16	13.16	13.16	13.16
1	5.96	5.96	5.96	5.96	5.96
2	2.37	2.37	2.37	2.37	2.37
3	1.93	1.93	1.93	1.93	1.93
4	1.49	1.49	1.49	1.49	1.49
5	1.59	1.59	1.52	1.52	1.51
6	1.34	1.34	1.46	1.46	1.46
7	1.26	1.28	1.46	1.46	1.46
8	1.27	1.18	1.45	1.46	1.46
9	1.80	**1.17**	1.44	1.44	1.44
10	1.92	1.20	1.46	1.47	1.47
11	1.60	1.18	1.45	1.45	1.45
12	1.66	1.22	1.48	1.46	1.46
13	1.55	1.24	1.50	1.47	1.48
14	1.58	1.23	1.48	1.46	1.46
15	1.52	1.23	1.48	1.45	1.46
16	1.53	1.23	1.48	1.45	1.46
17	1.53	1.23	1.48	1.45	1.46
18	1.52	1.23	1.48	1.46	1.46
19	1.52	1.23	1.48	1.46	1.46
20	1.52	1.23	1.48	1.46	1.46

Sample results for MAE values of generated models against combinations of hyper-parameter set pairs (0 to 20 MaxDepth values and 0.0 to 0.4 MinInforGain values) for DAS2 dataset are shown in Table 2. After analyzing the prediction results of 341 generated models for DAS2 dataset having maximum MAE = 13.16 and minimum MAE = 1.17, the model which has minimum MAE is observed in MinInfoGain = 0.1. The error rate does not vary much with changes in higher MaxDepth values (above 16) for all MinInfoGain values as shown in Table 2.

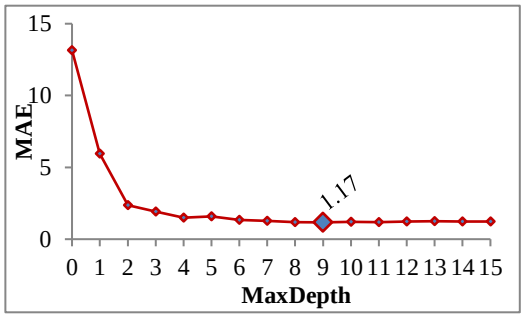

Figure 3. MAE results of generated models using different MaxDepth values with MinInfoGain = 0.1 (DAS2)

MAE results of generated models against 15 different MaxDepth values with the optimal MinInfoGain = 0.1 for DAS2 dataset are shown in Figure 3. The error rate is generally lower for MaxDepth = 8 to 11. From comparing the prediction results of the generated models, the model with hyper-parameter pair (MaxDepth = 9 and MinInfoGain = 0.1) gives better result MAE = 1.17. It is selected to predict the future CPU demand, as it is the model with minMAE.

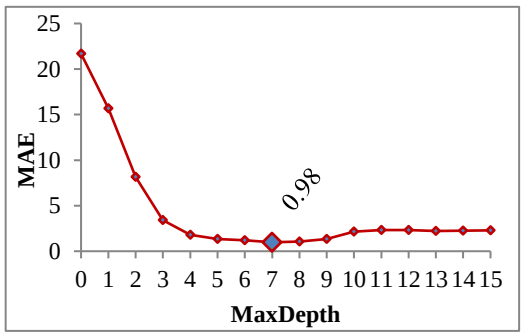

Figure 4. MAE results of generated models using different MaxDepth values with MinInfoGain = 0 (RICC)

After analyzing the prediction results of generated models for RICC dataset having maximum MAE = 21.69 and minimum MAE = 0.98, the model which has minimum MAE is observed in MinInfoGain = 0. The error rate does not vary significantly with changes in higher MaxDepth values for all MinInfoGain values. MAE results of the generated models against 15 different MaxDepth values with the optimal MinInfoGain = 0 for RICC dataset are clearly shown in Figure 4. The error rate is generally lower for MaxDepth = 5 to 9. From comparing the prediction results of the generated models, the model with hyper-parameter pair (MaxDepth = 7 and MinInfoGain = 0) gives better result MAE = 0.98. It is chosen to predict the future CPU demand, as it has the lowest error rate.

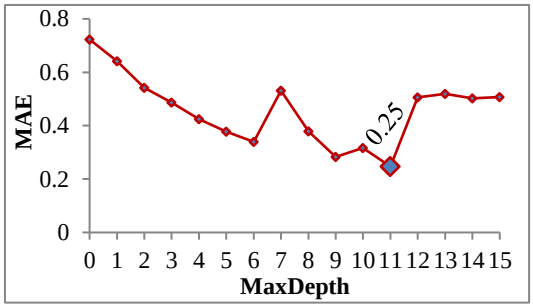

Figure 5. MAE results of generated models using different MaxDepth values with MinInfoGain = 0 (MetaCentrum)

After analyzing the prediction results of generated models for MetaCentrum dataset having maximum MAE = 0.72 and minimum MAE = 0.25, the model which has minimum MAE is observed in MinInfoGain = 0. The error rate does not vary much with changes in higher MaxDepth values for all MinInfoGain values. MAE results of the generated models against 15 different MaxDepth values with the optimal MinInfoGain = 0 for MetaCentrum dataset are presented in Figure 5. The error rate is generally lower at MaxDepth = 6, 9, 10 and 11. From comparing the prediction results of the generated models, the model with hyper-parameter pair (MaxDepth = 11 and MinInfoGain = 0) gives better result MAE = 0.25. It is chosen to predict the future CPU demand because it has the lowest error rate.

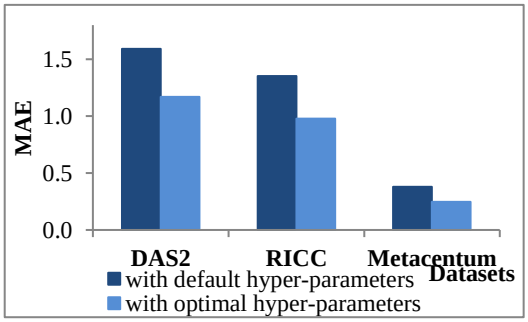

Figure 6. MAE comparisons of prediction models with default and optimal hyper-parameters

Figure 6 presents MAE comparisons of prediction models generated by setting default and optimal hyper-parameters for all three datasets. For DAS2, MAE value of the generated model by using default hyper-parameters is 1.59 and that by using optimal hyper-parameter pair is 1.17, saving MAE by 26%. For RICC, MAE value of the generated model by using default hyper-parameters is 1.35 and that by using optimal hyper-parameter pair is 0.98, saving MAE by 28%. For Metacentum, MAE value of the generated model by using default hyper-parameters is 0.38 and that by using optimal parameter pair is 0.25, saving MAE by 35%. Therefore, setting the optimal hyper-parameters can significantly affect the resulting model's performance. The results show that identifying the optimal hyper-parameter saves MAE 26%, 28% and 35% for the corresponding datasets respectively.

The actual and predicted CPU demand for the incoming requests of the validation dataset for DAS2 is presented in Figure 7.

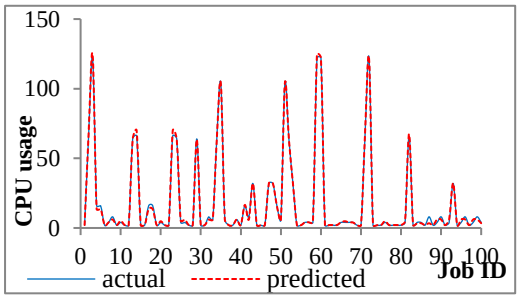

Figure 7. Actual and predicted CPU demand (DAS2)

Figure 8 illustrates the actual and predicted CPU demand for the incoming requests of the validation dataset for RICC.

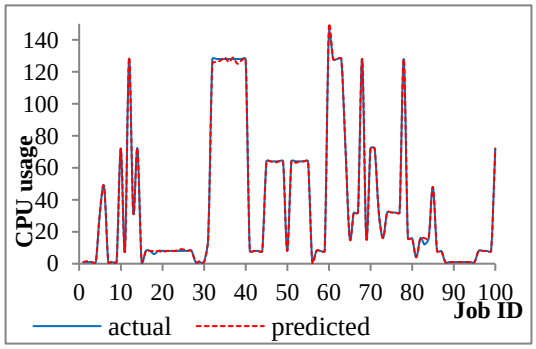

Figure 8. Actual and predicted CPU demand (RICC)

In Figure 9, the comparison of the actual and predicted CPU demand for the incoming requests of the validation dataset for MetaCentrum is shown.

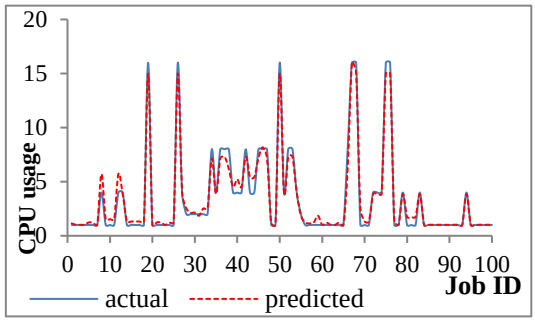

Figure 9. Actual and predicted CPU demand (MetaCentrum)

The results show that the actual and the predicted CPU demand by the proposed models for all three workload traces are nearly the same.

6. Conclusion

Resource demand prediction is a key feature for efficient resource management of dynamic workload for cloud data centers. In this paper, CPU resource demand prediction model implemented on Apache Spark is presented. The resource demand prediction model with low error rate is developed via hyper-parameter optimization. The prediction system is evaluated on real world workload data in cloud computing and high-performance computing paradigms. Experiment results show that the prediction model generated by setting the optimal hyper-parameters can significantly affect the resulting model's performance saving MAE round about 30%. It can be observed that the predicted CPU demand by the proposed models are as same as the actual CPU demand.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Journal of Future Generation Computer Systems, Vol. 25 Issue. 6, June, 2009, pp. 599–616.

[2] M. Feurer, T. Springenberg, and F. Hutter, "Initializing bayesian hyperparameter optimization via meta-learning," in Proceedings

of the Twenty- Ninth AAAI Conference on Artificial Intelligence, Jan. 2015, pp. 1128-1135.

[3] J. Fu, J. Sun, K. Wang, "SPARK—A Big Data Processing Platform for Machine Learning", in Proceedings of IEEE International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration, 2016, pp. 48-51.

[4] T.L. Anh, J. Tordsson, *Workload prediction for resource management in data centers*, Computational Science and Engineering, 2016.

[5] M. Verma, G. R. Gangadharan, R. Vadlamani, N. C. Narendra, "Resource demand prediction in multi-tenant service clouds". IEEE International Conference on Cloud Computing for Emerging Markets (CCEM), 2013.

[6] A. Mozo, B. Ordozgoiti, S. Canaval, "Forecasting short-term data center network traffic load with convolutional neural networks", PLoS ONE 13(2): e0191939, February 6, 2018.

[7] M. Camilleri, F. Neri, "Parameter Optimization in Decision Tree Learning by using Simple Genetic Algorithms", WSEAS TRANSACTIONS on COMPUTERS, Vol. 13, 2014, pp. 582–591.

[8] "Parallel Workloads Archive", Available at "http://www.cs.huji.ac.il/labs/parallel/workload/" [Online; accessed 5-September-2018]

[9] "Welcom to Apache Hadoop!", Available at "http://hadoop.apache.org/." [Online; accessed 2-September-2018].

[10] "Spark MLib", Available at "https:// spark.apache.org/mllib/." [Online; accessed 2-September-2018].

[11] "Decision Trees - RDD-based API", Available at "https://spark.apache.org/docs/2.2.0/mllib-decision-tree.html." [Online; accessed 5-September-2018].