Machine Learning Based Workload Prediction in Cloud Computing

Jiechao Gao, Haoyu Wang and Haiying Shen
Department of Computer Science
University of Virginia
Charlottesville, VA, USA
{jg5ycn, hw8c, hs6ms}@virginia.edu

***Abstract*—As a widely used IT service, more and more companies shift their services to cloud datacenters. It is important for cloud service providers (CSPs) to provide cloud service resources with high elasticity and cost-effectiveness and then achieve good quality of service (QoS) for their clients. However, meeting QoS with cost-effective resource is a challenging problem for CSPs because the workloads of Virtual Machines (VMs) experience variation over time. It is highly necessary to provide an accurate VMs workload prediction method for resource provisioning to efficiently manage cloud resources. In this paper, we first compare the performance of representative state-of-the-art workload prediction methods. We suggest a method to conduct the prediction a certain time before the predicted time point in order to allow sufficient time for task scheduling based on predicted workload. To further improve the prediction accuracy, we introduce a clustering based workload prediction method, which first clusters all the tasks into several categories and then trains a prediction model for each category respectively. The trace-driven experiments based on Google cluster trace demonstrates that our clustering based workload prediction methods outperform other comparison methods and improve the prediction accuracy to around 90% both in CPU and memory.**

***Index Terms*—Cloud Computing, Machine learning, Workload Prediction**

I. Introduction

Cloud computing is a widely used IT service, which provides various services under one roof. Multiple types of services such as storage, computing and web hosting now can be provided by one cloud service provider. Many businesses move their services to clouds due to their flexible service model such as pay-as-you-go business model [1]. Such elasticity of the service model brings about cost saving for most businesses by eliminating the need of developing, maintaining and scaling a large private infrastructure [2].

Cloud computing plays an important role nowadays which allows clients to use cloud resources in a pay-as-you-go fashion. It can satisfy the cloud resource requirements of the clients so that the clients need not to concern about the overprovisioning of a service whose resource utilization does not meet the predictions, and then wasting costly resources, or underprovisioning of a service which turns into popular in the future, and then missing potential revenue [3].

Using hardware virtualization, cloud service providers let a physical machine (PM) run multiple virtual machines (VMs) (i.e., tasks) with different resource allocations. A cloud hosts multiple applications on the VMs. Since the load of each VM on a PM varies over time, a PM may become overloaded, i.e., the resource demand from its VMs is beyond its possessed resource. Such load imbalance in a PM adversely affects the performance of all the VMs (hence the applications) running on the PM. Insufficient resources provision to customer applications also violates the Service Level Agreement (SLA) [4]. An SLA is an agreement between a cloud customer and the cloud service provider that guarantees the application performance of the customer. In order to uphold the SLA, a cloud service provider must prevent PM overload and ensure VMs receive their demanded resources. As shown in Figure 1, the prediction model can get the historical data from resource manager and send back the predicted workload for each task to resource manager. The resource manager then can arrange each VM on PMs according to the predicted results respectively.

As cloud data centers are often oversubscribed, resources such as CPU and bandwidth are stretched thin as they are shared across many tenants. In particular, when VMs with intense resource requirements are located on the same PM, they compete for scarce resources, which may lead to poor performance of applications. Much resource effort has been devoted to developing strategies for resource provisioning in the initial VM allocation and VM migration phases. Recently, some methods [5]–[11] have been proposed to predict VM resource demand in a short time for sufficient resources provisioning or load balancing. In the proactive load balancing, a PM predicts whether it will be overloaded by predicting its VMs' resource demands and moves out VMs when necessary. In the previous research, statistical approaches [12]–[15], machine learning (ML) approaches [16]–[20] and deep learning approaches [21]–[27] are used for the resource demand prediction. There has been no effort that conducts the comparison study on these prediction approaches.

To better understand these approaches, we compare the representative methods in these approaches using the Google cluster trace [28]. In the previous prediction methods, there is 0 time gap between the input workload data points and the predicted workload data point (we call *0-gap prediction*). Then, it may leave little time for task scheduling based on predicted workload. We propose *m-gap prediction* that keeps a gap of m time points between the input data points and the predicted data point in order to leave enough time for the task scheduling. Our experimental results show that *m-gap prediction* does not compromise the prediction accuracy performance of *0-gap prediction*. Also, previous prediction methods build one prediction model for all the tasks, which may not catch the patterns of all the heterogeneous tasks for more accurate prediction. In order to improve the accuracy performance of the previous prediction methods, we propose a clustering-based prediction method. It clusters tasks with similar workload patterns into a group for training to create a model, and uses the corresponding model of a task to predict its workload.

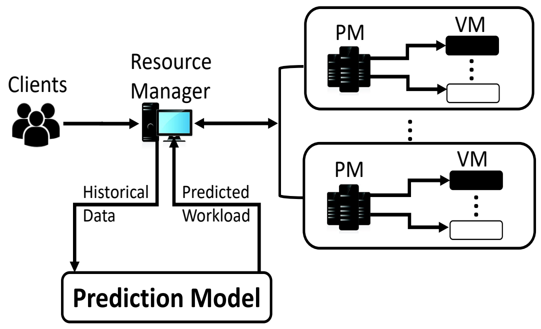


Fig. 1. Overview of workload prediction procedure.

Our contributions in this paper are as follows:

(1) We conduct experimental comparison on the prediction accuracy performance of state-of-the-art prediction methods from statistical approaches, machine learning approaches and deep learning approaches respectively.
(2) We propose *m-gap prediction* that keeps a gap of m time points between the input data points and the predicted data point in order to leave enough time for the task scheduling based on predicted workload.
(3) We also propose a clustering-based prediction method for higher prediction accuracy and use two clustering algorithms here. The method first clusters all the tasks into several categories and then generates a model for each task category. Since each model can capture the different features in each task category, the accuracy prediction performance is much better than the prediction methods that have only one model for all the tasks.
(4) We implement our proposed methods and construct extensive experiments. The experimental results show that *m-gap prediction* does not compromise the accuracy performance of *0-gap prediction* used in previous prediction methods. Also, our clustering-based prediction method achieves much better accuracy performance than all the previous methods.

The rest of the paper is organized as follows. Section II presents the related work. Section III presents the Google cluster trace preparation and the measurement results for state-of-the-art prediction methods. Sections IV presents our clustering-based prediction methods and the performance evaluation of our methods. Section V concludes the paper with remarks on our future work.

II. Related Work

We classify all the previous resource demand (i.e., workload) prediction works into three parts: statistical approaches, machine learning approaches and deep learning approaches.

Statistical approaches. The statistical approaches are popular ways in predicting workload. Khan *et al.* [12] discovered the repeatable workload patterns of VMs, and then introduced an approach based on Hidden Markov Modeling to characterize and predict workload patterns. Jiang *et al.* [13] presented an online temporal data mining system called ASAP, which is used to model and predict the cloud VM demand by using Moving Average (MA) model. Morais *et al.* [14] proposed a framework for the implementation of auto-scaling services that are based on several CPU utilization prediction methods including Auto Correlation (AC), linear regression (LR), auto regression (AR), Auto Regression Integrated Moving Average (ARIMA) and so on. Gong *et al.* [15] developed PRESS that uses a pattern matching and state-driven approach to predict workloads. It first employs signal processing techniques to check if the CPU utilization in a VM exhibits repeating patterns. If yes, the repeating patterns are used to predict future workloads; otherwise, PRESS employs a statistical state-driven approach, and uses a discrete-time Markov chain to predict the demand for the near future. However, many datasets are unstable (i.e., the variance between each two neighboring is large or some points missed), but these time series prediction approaches employ the linear prediction structure, which is more suitable for stable dataset.

Machine learning approaches. Machine learning approaches are widely used for VM workload prediction as well. Imam *et al.* [29] presented time delay neural network (NN) and regression methods to predict the workload of each VM. Farahnakian *et al.* [30] developed resource measurement and provisioning strategies using NN and linear regression to predict upcoming VMs' demands. Bankole *et al.* [17] developed a cloud client prediction model to predict the resource demand of each VM using three machine learning models: support vector regression, NN and linear regression. Islam *et al.* [16] proposed an approach using NN and Linear Regression algorithms to predict the future CPU load of a VM and they have concluded that NN surpasses Linear Regression in terms of accuracy. In addition, they have shown that the accuracy of both algorithms depends on the input window size. Nikravesh *et al.* [19], [20] have evaluated the Support Vector Machine (SVM), NN and Linear Regression machine learning prediction methods. They found that, if the resource utilization of each VM changes periodically, SVM has better prediction accuracy compared to the other methods. However, when the size of the dataset is large (such as the Google cluster trace, SVM cannot achieve high prediction accuracy as indicated in [31]. Gopal *et al.* [32] proposed Bayesian model for resource prediction of each VM and compared with linear regression method and support vector regression. They observed that by using Bayesian based model, the workloads of approximately 75% of the servers in datacenter could be predicted with accuracies over 80%.

Deep learning approaches. Deep learning approaches are also applied for workload prediction in recent years. Qiu *et al.* [21] presented a deep learning approach (that consists of a Deep belief network (DBN) and a regression layer) for the VM workload prediction in the cloud system. Zhang *et al.* [24] presented an efficient deep learning model based on the canonical polyatomic decomposition to predict the workload of each VM. Their proposed model can achieve a high training speedup since it utilizes the canonical polyatomic decomposition to compress the parameters significantly with a low classification accuracy drop. Zhang *et al.* [23] proposed a DBN-based approach for cloud resource request prediction of each task that can be used for long-term and short-term prediction with improved accuracy compared with existing methods. Kumar *et al.* [25] developed prediction models based on Long Short Term Memory (LSTM) networks [33]. The proposed model is tested on three benchmark datasets of web server logs, and HTTP traces of NASA server, Calgary server, and Saskatchewan server. Song *et al.* [26] applied a model based on LSTM to predict the mean load over consecutive future time intervals and actual load multi-step-ahead using Google cluster trace, which achieves high prediction accuracy in a traditional distributed system.

III. Trace Analysis and Measurement

A. Google Cluster Trace

The Google cluster trace starts at 19:00 EDT on Sunday May 1, 2011, and it records 29 days' resource utilization of CPU and memory usage of each task on the Google cluster of about 12.5k machines. A job is comprised of one or more tasks, each of which is accompanied by a set of resource requirements. The trace contains 672,075 jobs and more than 48 million tasks in the 29 days. This trace is a randomly-picked 1 second sample of CPU/memory usage from within the associated 5-minute usage-reporting period for each task. We use the entire trace to conduct the trace-driven experiments and conduct measurement among the comparison methods.

We notice that in the Google cluster trace, the CPU usage and memory usage of some tasks are zero for a period of time. It doesn't necessarily mean that the task is paused as indicated in [28]. There could be many reasons for it. [28] indicates that when the measurements occur while the monitoring system or machine hosting the system is overloaded, memory and CPU for a task may not be collected and then set 0. In some cases, a task has no process for an extended period of time. Also, the measurement records may be missing, thus generating pathological data. These zero values make the data non-linear so that it may be difficult to predict the resource usage via statistical model. In this case, it is important to find the prediction algorithms that can better deal with pathological data.

B. Statistical and Machine Learning Methods

We implement the whole experiment on a local machine based on Tensorflow. Recall that the workload prediction approaches can be classified to three groups: statistical approaches, machine learning approaches and deep learning approaches. According to the performance, we choose ARIMA [34] to represent the statistical approaches since ARIMA can achieve an average prediction accuracy over 70% compared with other statistical methods in [35]. We choose Support Vector Regression (SVR) [36], [37] and Bayesian Ridge Regression [32], [38] to represent the machine learning approaches since these two methods are announced as the most effective prediction algorithms for cloud system in [19], [20]. We choose LSTM [26], [33] to represent the deep learning approaches since LSTM can achieve higher accuracy than Autoregressive method, artificial neural network, ARIMA in host load prediction in [26].

*1) **Auto Regression Integrated Moving Average (ARIMA):*** ARIMA is widely used in time series analysis. It is a generalization of an autoregressive moving average (ARMA) model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series. ARIMA model can be applied to the unstable datasets via one or more differencing steps. The differencing step forms data transformation which can be applied on the time-series data to make the data more stable.

*2) **Bayesian Ridge Regression (BRR):*** Bayesian Ridge Regression (BRR) has better performance when dealing with pathological data. The BRR has a probabilistic model for regression problems. When the pathological data occurs, the prediction variances are large so they may be far from the actual value. In BRR, by adding a degree of bias to the regression, it can reduce the standard errors and then achieve better prediction accuracy.

*3) **Support Vector Regression (SVR):*** SVR is based on the computation of a linear regression function in a multiple variables feature space where the input data can be used via a non-linear regression function [39]. The model produced by SVR depends only on a subset of the training data, because the cost function of building the model doesn't take into account of any training data which is closer to the prediction results. In another words, during the training of SVR model, it puts more weights on the data points further to the previous predicted value so that the model can consider more on further points to capture more possible patterns in a dataset.

*4) **Long Short Term Memory (LSTM):*** LSTM is a neural network model which is widely used in the field of deep learning. It can be applied to fit the time-series data. For the pathological data problem, it can void those pathological data via forget gate and input gate and then achieve high prediction accuracy.

In each method we mentioned above, we choose the same experiment settings as the previous paper indicates. For ARIMA, we implement the method with the same experiment settings mentioned in [35]. For SVR, we implement the method with the same experiment settings mentioned in [37]. For BRR, we implement the method with the same experiment settings mentioned in [32]. For LSTM, we implement the method with the same experiment settings mentioned in [26].

Then, we compare these methods under the circumstances of 0-gap prediction and m-gap prediction as follow.

C. Prediction Methods

*1) **0-gap Prediction**:* This method is used in the previous prediction methods. That is, the w data points before the n^{th} time point are used as inputs to predict the value at the n^{th} time point, denoted by V_n. w here means window size. For each task, we use the first 80 percent time of the trace as the training set and use the remaining 20 percent of the trace as the testing set. We call it 0-gap prediction because there is no time gap between the time of inputs and the time of the output value.

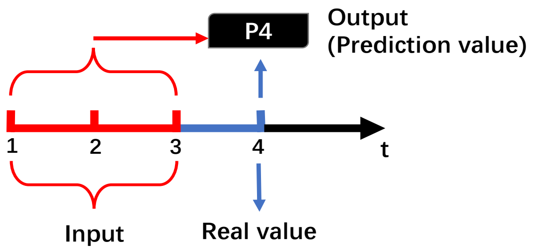


Fig. 2. An example of 0-gap prediction.

Figure 2 shows a simple example for the training and testing procedure of the 0-gap prediction. The numbers in the squares mean the time sequence. Take the testing procedure for instance, the red squares represent the input data of the testing and the number of red squares means the window size w. In this example, the $w = 3$. The blue square represents the real value of one time point. The black square represents the prediction value of the time point. Three data points (from the 1^{st} to the 3^{rd} time points) are used as input data to get the prediction value of the 4^{th} time point, and it is compared with the real value of the 4^{th} point to get the accuracy.

*2) **m-gap Prediction**:* In the above 0-gap prediction method, since there is no time gap between the inputs and the output. Then, the prediction model may output the predicted

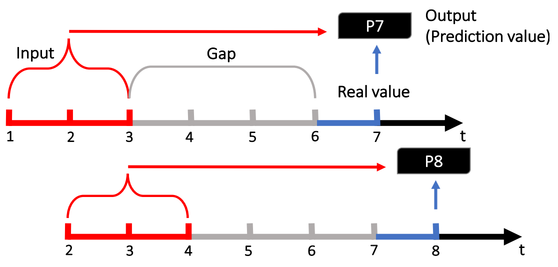


Fig. 3. An example of m-gap prediction.

value of V_n after the real workload of V_n already occurs. Even though the prediction can be completed before the real occurrence, there may leave little time for the scheduling before the real workload of V_n occurs. Therefore, we propose a method called m-gap prediction. That is, the w data points before the $n^{th} - m$ time point are used as inputs to predict the value at the n^{th} time point. For each task, we use the first 60 percent time of the trace as the training set and use the remaining 40 percent of the trace as the testing set.

Figure 3 shows the training and testing procedure of the m-gap prediction. Different from the 0-gap prediction, m-gap prediction has a time window gap m between the last time point of input data and the output time point, as shown by gray squares. In this example, the window size $w = 3$ and $gap = 3$. Three data points (from the 1^{st} to the 3^{rd} time points) are used as input data to get the prediction value of the 7^{th} time point, and it is compared with the real value of the 7^{th} point to get the accuracy.

D. Metrics

To illustrate the performance of the above methods, we use three metrics to determine the better results.

1) CDF for Accuracy: We use cumulative distribution function (CDF) to show the performance of accuracy. As in [40], the prediction accuracy is calculated by:

$$An = 1 - \frac{|P_n - R_n|}{R_n} \tag{1}$$

where A_n is the prediction accuracy of n^{th} prediction, P_n is the predicted value of n^{th} prediction and R_n is the real value in n^{th} prediction.

2) CDF for Accuracy with Different Window Sizes: To show the influence of the window size on the accuracy performance, we show the CDF for accuracy with different window sizes for the best model in accuracy.

E. Experimental Results

1) 0-gap Prediction Results: We first evaluate *0-gap prediction*. Figure 4 shows the CDF of the prediction accuracy of the CPU usage among the four comparison methods. The result follows SVR< ARIMA≈ LSTM<BRR. SVR has worse results than other methods. Since the size of Google cluster trace is large, SVR cannot achieve high prediction accuracy as indicated in [31]. ARIMA and LSTM have better performance than SVR but worse than BRR. For ARIMA, as mentioned previously that the Google cluster trace has some pathological data, since ARIMA creats a liner prediction model, it does not perform well for handling pathological datasets. For LSTM, its neural network mode can fit the non-linearities of a dataset. However, it does not perform well for short-term tasks that do not many data points for training as indicated in [33]. So the performance for LSTM is not good for the short-term tasks, which leads to its worse performance than BRR. BRR has the highest accuracy performance compared to other methods. BRR uses Levenberg-Marquardt algorithm [41] which is for non-linear datasets. Thus, BRR can achieve the best performance among these four methods in spit of the pathological dataset and the case of few training data points. Since the performance of SVR is worse than other three methods, we will not discuss this method in the rest of the experiments.

Figure 5 show the CDF of the prediction accuracy of the CPU usage in different window sizes with 1, 10 and 50. For these three cases, the results follow that window

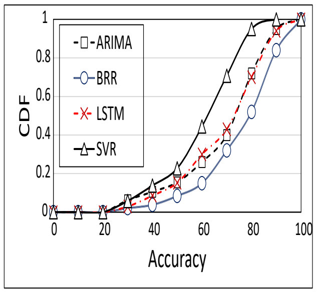
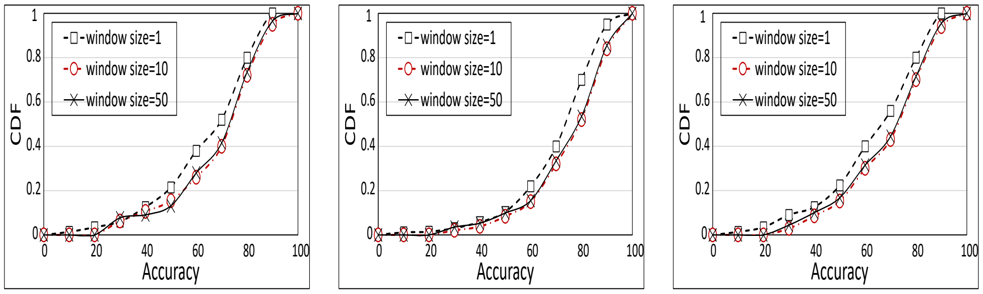


Fig. 4. CDF of CPU accuracy for 0-gap prediction.



(a) ARIMA (b) BRR (c) LSTM

Fig. 5. CDF of CPU accuracy with different window sizes for 0-gap prediction.

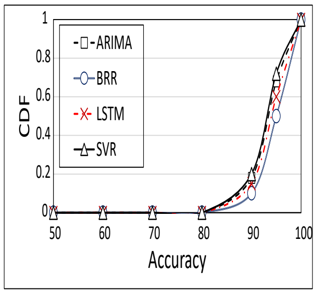
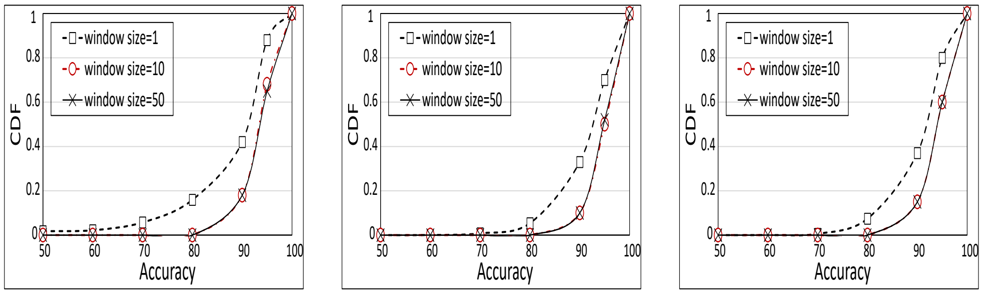


Fig. 6. CDF of memory accuracy for 0-gap prediction.



(a) ARIMA (b) BRR (c) LSTM

Fig. 7. CDF of memory accuracy with different window sizes for 0-gap prediction.

size=1<window size=10≈window size=50. Larger window size can introduce more input values in the model and then achieve more accurate results. However, the larger window size leads to much longer training and testing time cost [42]. Since window size=10 has similar prediction accuracy but lower overhead, we use window size=10 as unless otherwise specified.

Similar to Figures 4 and 5, Figures 6 and 7 show the results in memory prediction. The results follow the same trend and order for all the comparison methods and the different window sizes due to the same reasons.

2) m-gap Prediction Results: Figure 8 shows the CDF of the prediction accuracy of the CPU usage in *m-gap prediction*. The same as in *0-gap prediction*, the result follows ARIMA≈ LSTM<BRR. Due to the same reasons, for ARIMA, it does not perform well in handling pathological datasets. For LSTM, its neural network model can fit the non-linearities of dataset, but its performance is not good for the short-term tasks with not many training data points, which leads to its worse performance than BRR. BRR has the highest accuracy performance compared to other methods as it can handle pathological datasets and few training data points.

Figure 9 show the CDF of the prediction accuracy of the CPU usage in different window sizes with 1, 10 and 50 for *m-gap prediction*. For these three cases, even though the gap $m = 10$, the results follow that window size=1<window size=10≈window size=50 which is the same as in Figure 5.

Similar to Figures 6 and 7, Figures 10 and 11 show the results in memory prediction. The results follow the same trend and order for all the comparison methods and the different window sizes due to the same reasons.

IV. Clustering based Prediction Methods

Since different tasks have different workload features, it may be difficult for one model to capture the variable workload features and then predict the resource utilization with high accuracy. One prediction model can achieve higher accuracy for the tasks with similar workload features. Therefore, in order to overcome this problem for higher prediction accuracy, we propose clustering based prediction methods which are introduced below.

A. Clustering Methods

In the above, we build one model that is used for predicting the workloads of all tasks. Different tasks have different workload features. We would like to see if we build one model for similar tasks, whether the prediction accuracy can be improved. Therefore, we first cluster similar tasks to a group, and then build one model for each task cluster with the same machine learning algorithm. For a given task, we choose corresponding model of the task's category to predict its workload. We use two clustering methods explained below.

1) ***Prototype-based Clustering Method****:* Prototype-based clustering (PBC) is to find the shortest distance that between every tasks to the center. The number of clusters N means that the tasks (described by CPU and memory usage data) will be divided into N parts and for each part the total distance between each task description (or data point) to the center is the shortest.

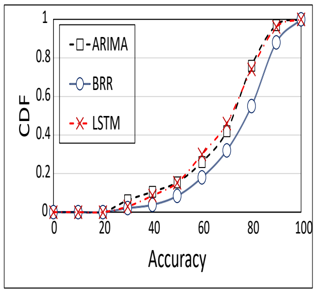
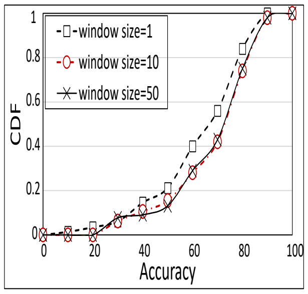
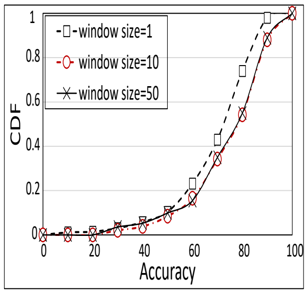


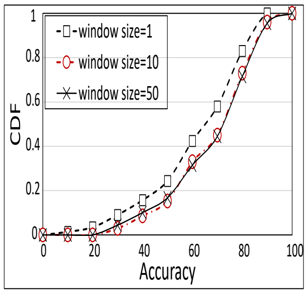
Fig. 8. CDF of CPU accuracy for m-gap prediction.



(a) ARIMA



(b) BRR



(c) LSTM

Fig. 9. CDF of different window sizes in CPU for m-gap prediction.

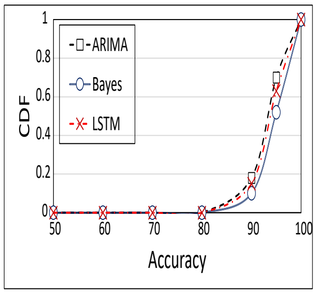
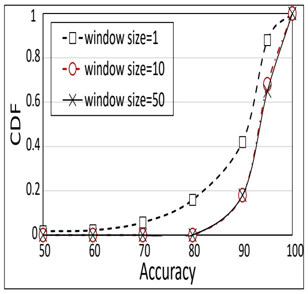
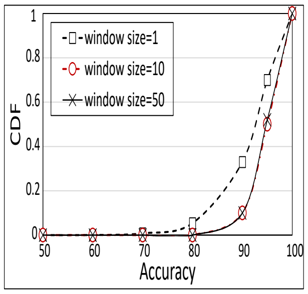


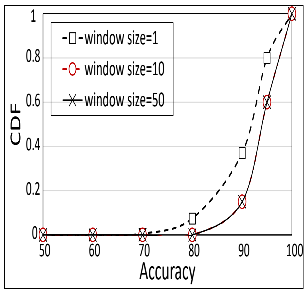
Fig. 10. CDF of memory accuracy for m-gap prediction.



(a) ARIMA



(b) BRR



(c) LSTM

Fig. 11. CDF of memory accuracy with different window sizes for m-gap prediction.

The K-means [43] and Gaussian Mixture Clustering [44] are the two main methods for PBC. K-means is a distance-based iterative algorithm. It clusters the whole data points observation instances into N clusters so that each observation instance is smaller than the center point of the cluster in which it is located, compared to other cluster center points. In order to minimize the squared error, the K-means algorithm uses iterative optimization to approximate the target. In Gaussian Mixture clustering, it uses probability model to express clustering prototype. The problem here is that the lengths of all tasks are not exactly the same while the PBC clustering methods require the same length of all data points. To solve this problem, we append 0 in the end to make all the tasks have the same length. In our experiment, we find out that the performance of K-means is better than Gaussian Mixture Clustering. So we use K-means in our prediction methods. The input of the PBC method is all tasks in the dataset. We cluster the tasks into different subsets. For each subset, we use one of the three prediction algorithms (i.e., ARIMA, BRR and LSTM) to build a model. As a result, N subsets lead to N models. Larger N leads to high computation overhead but lower N leads lower prediction accuracy since the lower number of subsets may not capture the workload feathers. We use $N = 5$ in default because we found it achieves a better tradeoff between prediction accuracy and computer overhead from our experiments.

*2) **Density-based Clustering Method:*** Density-based clustering (DBC) method [45] is a density-based clustering non-parametric method. It first finds the points within high density area and then groups together points that are close. Meanwhile, for the rest outliers points that lie alone in low-density area, each point is assigned into the nearest group. Finally, all the points are clustered. One difference between K-means and DBC is that the number of groups, N, can be set manually in K-means but is determined by DBC itself. DBC clusters all the tasks into 5 groups.

Also since DBC discovers clusters by continuously connecting high-density points in the neighborhood, it only needs to define the neighborhood size and density thresholds, so clusters of different shapes and sizes can be found. We use DBC in our prediction methods. The input of DBC is all the tasks in the dataset. We cluster the tasks into different subsets and directly use the previous prediction algorithms on each subset. Using this method, we get the clustered tasks with same pattern and train each subset for better prediction accuracy.

B. *Prediction Procedure*

Combining the different clustering methods and prediction methods, we finally can get the methods denoted by PBC-ARIMA, DBC-ARIMA, PBC-BRR, DBC-BRR, and PBC-LSTM and DBC-LSTM. The clustering method clusters all the tasks into several groups. Then, we use the data in each group for training to build a model. To predict a task's workload, we map the task to a task group based on its features and then use the model for the corresponding group to conduct prediction.

C. *Experimental Results*

Now we evaluate the performance of PBC and DBC based prediction methods. Since there is no big difference for predic-

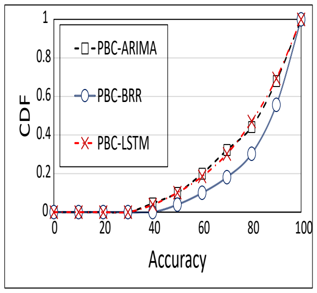
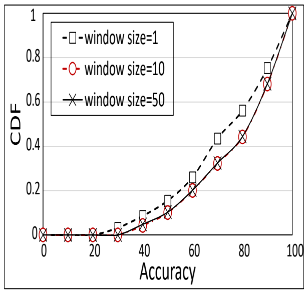
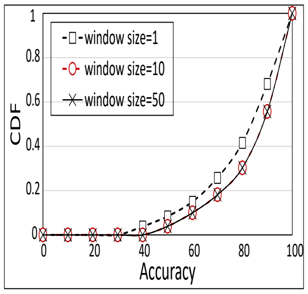


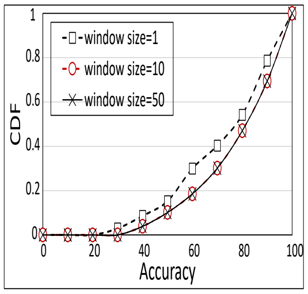
Fig. 12. CDF of CPU accuracy in PBC-based prediction.



(a) PBC-ARIMA



(b) PBC-BRR



(c) PBC-LSTM

Fig. 13. CDF of CPU accuracy with different window sizes in PBC-based prediction.

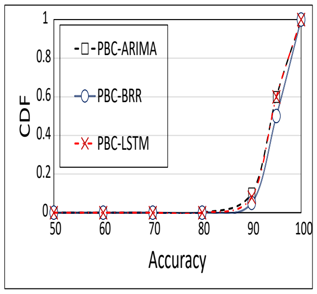
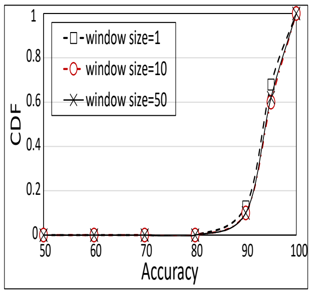
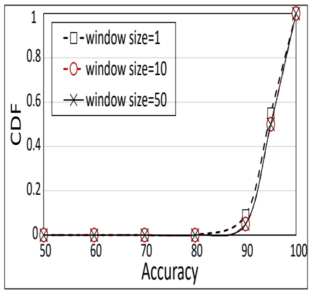


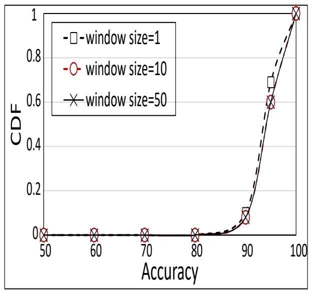
Fig. 14. CDF of memory accuracy in PBC-based prediction.



(a) PBC-ARIMA



(b) PBC-BRR



(c) PBC-LSTM

Fig. 15. CDF of memory accuracy with different window sizes in PBC-based prediction.

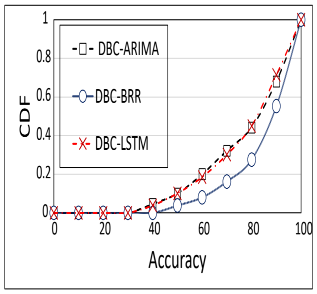
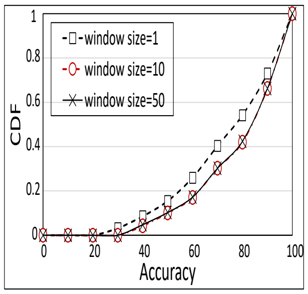
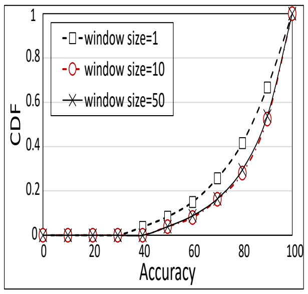


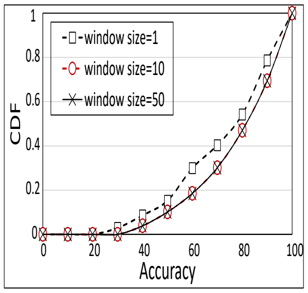
Fig. 16. CDF of CPU accuracy in DBC-based prediction.



(a) DBC-ARIMA



(b) DBC-BRR



(c) DBC-LSTM

Fig. 17. CDF of CPU accuracy with different window sizes in DBC-based prediction.

tion accuracy between *0-gap prediction* and *m-gap prediction*. We use *0-gap prediction* below for these clustering based methods.

1) PBC-based Prediction: Figure 12 shows the CDF of the prediction accuracy for the CPU usage using the PBC-based prediction methods. The results follow PBC-ARIMA≈PBC-LSTM<PBC-BRR. For PBC-ARIMA, as we discussed before, ARIMA cannot achieve better performance when the data is not stable. Since the pathological data randomly exists in all the tasks [28], even after PBC that clusters similar tasks for modeling, the performance of ARIMA is still worse and similar to the performance of PBC-LSTM. PBC-LSTM's advantage is to predict the workload for long-term tasks (with many training data points) because of the neural network involved in LSTM, but it is not good in predicting the workload of short-term tasks that do not have many data points for training. Even after PBC, PBC-LSTM still performs worse than PBC-BRR. For PBC-BRR, it can handle non-linear datasets and few training datasets. After PBC, PBC-BRR can achieve higher prediction accuracy than other two methods. Thus, PBC-BRR still has the best performance among three methods.

Figure 13 show the CDF of the prediction accuracy for the CPU usage using different window sizes for all the three methods. The results follow the same trend and order as in Figure 5 due to the same reasons.

Figures 14 and 15 show the CDF of the prediction accuracy in memory. These figures demonstrate that all the methods achieves the similar prediction accuracy performance with varied window sizes. The reason is that, after clustering, all the methods can achieve similar prediction accuracy performance even with smaller window size. Comparing these results with

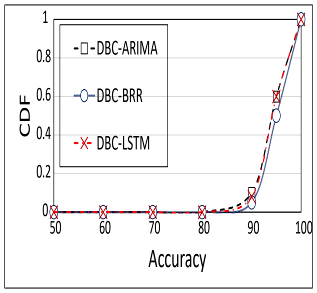
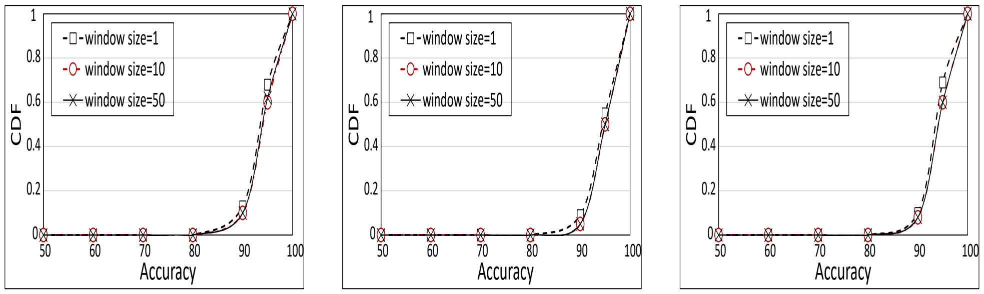
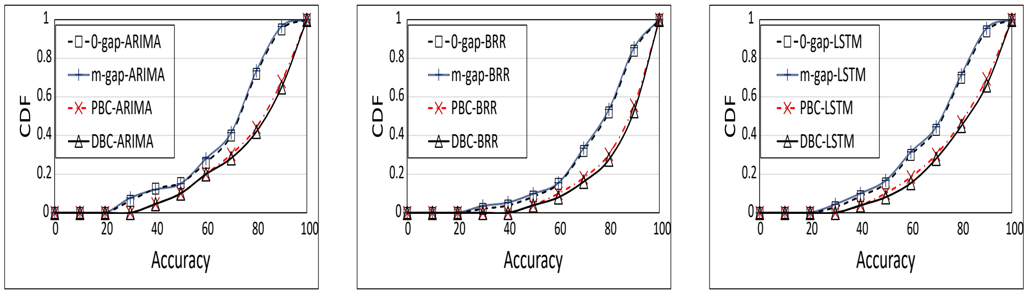


Fig. 18. CDF of memory accuracy in DBC-based prediction.



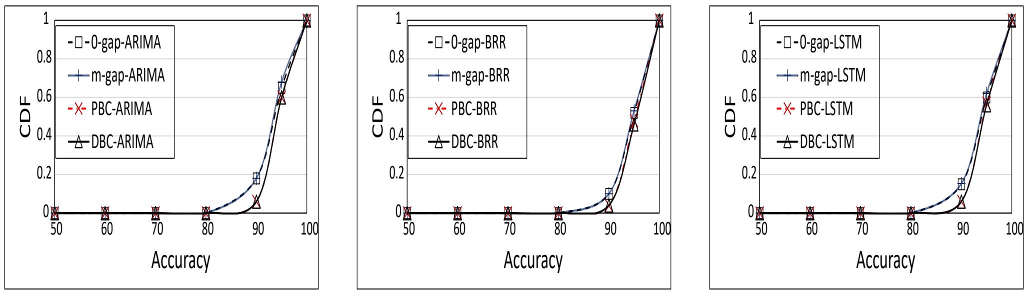
(a) DBC-ARIMA (b) DBC-BRR (c) DBC-LSTM

Fig. 19. CDF of memory accuracy with different window sizes in DBC-based prediction.



(a) ARIMA (b) BRR (c) LSTM

Fig. 20. CDF of CPU accuracy of enhanced prediction methods.



(a) ARIMA (b) BRR (c) LSTM

Fig. 21. CDF of memory accuracy of enhanced prediction methods.

Figure 7, we notice that the accuracy of window size 1 is greatly improved. Therefore, PBC can highly improve the prediction accuracy.

2) DBC-based Prediction: Figure 16 shows the CDF of the prediction accuracy for the CPU usage using the DBC-based prediction methods. After DBC, the results follow the same trend and order as in Figure 12 due to the same reasons.

Figure 17 show the CDF of the prediction accuracy for the CPU usage using different window sizes for all the three methods. The results follow the same trend and order as in Figure 13 due to the same reasons.

Figures 18 and 19 show the CDF of the prediction accuracy for the memory usage. These figures follow the same trend and order as in Figures 16 and 17 due to the same reasons.

3) Comparison Performance Evaluation: In this section, we compare our proposed clustering based methods and the three state-of-art prediction methods in *0-gap prediction*. We also include the results of the comparison methods in *m-gap prediction* for reference. Figures 20 and 21 show the CDF of the prediction accuracy in the CPU and memory usage. The results follow *0-gap prediction* $\approx$ *m-gap prediction* $<$ *PBC* $\approx$ *DBC* where the improvement is from 75% to over 90% in CPU usage prediction and from 92% to 95% in memory usage prediction. In the trace, since different tasks have much different patterns, *0-gap prediction* and *m-gap prediction* cannot achieve better prediction performance using only one model. *PBC* and *DBC* help the prediction methods achieve better prediction performance since the clustering methods cluster the tasks with similar patterns into a group for training. Then, the model corresponding to the group of tasks can capture the pattern easily and predict the performance more accurately. The result indicates that clustering methods can highly improve the prediction accuracy as much as 15%

compared with the previous prediction methods.

V. Conclusion

Accurate task workload prediction is crucial in cloud resource management. In this paper, we first measured and compared the state-of-the-art statistical and machine learning methods in the task workload prediction using the Google cluster trace. Then, we suggested the *m-gap prediction* method to do workload prediction a certain time before the predicted time point to leave enough time for task scheduling based on predicted workload. We further proposed a clustering based workload prediction method for higher prediction accuracy. This method clusters tasks with similar workload patterns, builds a workload prediction model for each cluster, and uses corresponding model to predict the upcoming workload of a task. This method achieves higher prediction accuracy compared to the traditional prediction methods. In the future work, we will focus on improving the architecture of deep learning algorithms with clustering based model to achieve higher prediction accuracy.

Acknowledgements

This research was supported in part by U.S. NSF grants NSF-1827674, CCF-1822965, OAC-1724845, CNS-1733596, Microsoft Research Faculty Fellowship 8300751, and AWS Machine Learning Research Awards.

References

[1] L. Vaquero, L.and Rodero-Merino and M. Caceres, J.and Lindner, "A break in the clouds: towards a cloud definition," in *Proc. of SIGCOMM*, 2008.

[2] H. Shen and L. Chen, "Distributed autonomous virtual resource management in datacenters using finite-markov decision process," *Trans. on TON*, 2017.

[3] A. Josep, R. Katz, A. Konwinski, L. Gunho, D. Patterson, and A. Rabkin, "A view of cloud computing," *Communications of the ACM*, 2010.

[4] C. Qiu, H. Shen, and L. Chen, "Probabilistic demand allocation for cloud service brokerage," in *Proc. of INFOCOM*, 2016.

[5] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *Trans. on TPDS*, 2013.

[6] H. Wang, H. Shen, and Z. Li, "Approaches for resilience against cascading failures in cloud datacenters," in *Proc. of ICDCS*, 2018.

[7] W. Wei, H. Fan, X.and Song, and J. Fan, X.and Yang, "Imperfect information dynamic stackelberg game based resource allocation using hidden markov for cloud computing," *Trans. on SC*, 2018.

[8] M. Xu and R. Buyya, "Brownout approach for adaptive management of resources and applications in cloud computing systems: A taxonomy and future directions," *ACM Computing Surveys (CSUR)*, 2019.

[9] Y. Yu, F. Jindal, V.and Bastani, F. Li, and I. Yen, "Improving the smartness of cloud management via machine learning based workload prediction," in *Proc. of COMPSAC*, 2018.

[10] M. Hassan, H. Chen, and Y. Liu, "Dears: A deep learning based elastic and automatic resource scheduling framework for cloud applications," in *Proc. of UBICOMP*, 2018.

[11] H. Wang and H. Shen, "Proactive incast congestion control in a datacenter serving web applications," in *Proc. of INFOCOM*, 2018.

[12] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *Proc. of NOMS*, 2012.

[13] Y. Jiang, C. Perng, T. Li, and R. Chang, "Asap: A self-adaptive prediction system for instant cloud resource demand provisioning," in *Proc. of ICDM*, 2011.

[14] A. Morais, V. Brasileiro, V. Lopes, A. Santos, W. Satterfield, and L. Rosa, "Autoflex: Service agnostic auto-scaling framework for iaas deployment models," in *Proc. of CCGrid*, 2013.

[15] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *Proc. of ICNSM*, 2010.

[16] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Trans. on FGCS*, 2012.

[17] A. Bankole and S. Ajila, "Cloud client prediction models for cloud resource provisioning in a multitier web application environment," in *Proc. of SOSE*, 2013.

[18] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proc. of ICC*, 2011.

[19] A. Nikravesh, S. Ajila, and C. Lung, "Measuring prediction sensitivity of a cloud auto-scaling system," in *Proc. of CSAC*, 2014.

[20] Y. Nikravesh, S. Ajila, and C. Lung, "Towards an autonomic auto-scaling prediction system for cloud resource provisioning," in *Proc. of SEAS*, 2015.

[21] F. Qiu, B. Zhang, and J. Guo, "A deep learning approach for vm workload prediction in the cloud," in *Proc. of SNPD*, 2016.

[22] J. Gao, H. Wang, and H. Shen, "Task failure prediction in cloud data centers using deep learning," *Proc. of IEEE Bigdata*, 2019.

[23] W. Zhang, P. Duan, L. Yang, F. Xia, Z. Li, Q. Lu, W. Gong, and S. Yang, "Resource requests prediction in the cloud computing environment with a deep belief network," *Software: Practice and Experience*, 2017.

[24] Q. Zhang, L. Yang, Z. Yan, Z. Chen, and P. Li, "An efficient deep learning model to predict cloud workload for industry informatics," *Trans. on TII*, 2018.

[25] J. Kumar, R. Goomer, and K. Singh, "Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters," *Trans. on Computer Science*, 2018.

[26] B. Song, Y. Yu, Y. Zhou, Z. Wang, and S. Du, "Host load prediction with long short-term memory in cloud computing," *Journal of Supercomputing*, 2018.

[27] J. Gao, H. Wang, and H. Shen, "Smartly handling renewable energy instability in supporting a cloud datacenter," *Proc. of IPDPS*, 2020.

[28] C. Reiss, J. Wilkes, and J. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., White Paper*, 2011.

[29] T. Imam, F. Miskhat, R. Rahman, and A. Amin, "Neural network and regression based processor load prediction for efficient scaling of grid and cloud resources," in *Proc. of ICCIT*, 2011.

[30] F. Farahnakian, P. Liljeberg, and J. Plosila, "Lircup: Linear regression based cpu usage prediction algorithm for live migration of virtual machines in data centers," in *Proc. of EuroSEAA*, 2013.

[31] T. Joachims, "Training linear svms in linear time," in *Proc. of SIGKDD*, 2006.

[32] G. Shyam and S. Manvi, "Virtual resource prediction in cloud environment: a bayesian approach," *Journal of Network and Computer Applications*, 2016.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 1997.

[34] S. Das, *Time series analysis*. Princeton University Press, Princeton, NJ, 1994.

[35] R. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using arima model and its impact on cloud applications' qos," *Trans. on CC*, 2015.

[36] A. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, 2004.

[37] L. Chen, H. Shen, and K. Sapra, "Rial: Resource intensity aware load balancing in clouds," in *Proc. of INFOCOM*, 2014.

[38] T. Park and G. Casella, "The bayesian lasso," *Journal of the American Statistical Association*, 2008.

[39] D. Basak, S. Pal, and D. Patranabis, "Support vector regression," 2007.

[40] "http://www.acheronanalytics.com/acheron-blog/how-to-measure-the-accuracy-of-predictive-models, [Accessed in APR 2019]."

[41] J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*, 1978.

[42] C. Richard, J. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *Trans. on SP*, 2009.

[43] K. Krishna and N. Murty, "Genetic k-means algorithm," *Trans. on SMC*, 1999.

[44] G. Celeux and G. Govaert, "Gaussian parsimonious clustering models," 1995.

[45] J. Sander, M. Ester, H. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," 1998.