Requests Prediction in Cloud with a Cyclic Window Learning Algorithm

Min Sang Yoon[a], Ahmed E. Kamal[b], and Zhengyuan Zhu[c]
Department of Electrical and Computer Engineering[a],[b], Department of Statistics [c]
Iowa State University, IA, 50010
{my222[a], kamal[b],zhuz[c]}@iastate.edu

***Abstract*—Automatic resource scaling is one advantage of cloud systems. Cloud systems are able to scale the number of physical machines depending on user requests. Therefore, accurate request prediction brings a great improvement in cloud systems' performance. If we can make accurate requests prediction, the appropriate number of physical machines that can accommodate predicted amount of requests can be activated and cloud systems will save more energy by preventing excessive activation of physical machines. Also, cloud systems can implement advanced load distribution with accurate requests prediction. We propose a prediction model that predicts probability distribution parameters of requests for each time interval. Maximum Likelihood Estimation (MLE) and Local Linear Regression (LLR) are used to implement this algorithm. An evaluation of the proposed algorithm is performed with the Google cluster-trace data. The prediction is achieved in terms of the number of task arrivals, CPU requests, and memory resource requests. Then the accuracy of prediction is measured with Mean Absolute Percentage Error (MAPE) and Normalized Mean Squared Error (NMSE).**

I. Introduction

Predicting users' requests has been playing an important role in operating network systems because the accurate prediction can prevent systems from wasting operational cost and optimize resource utilization. Resource utilization is an especially important issue in cloud systems because one of the advantages of cloud systems is automatic scaling and management. Therefore, the accurate prediction of the resource requests in the cloud systems brings a significant performance improvement of the system with more advanced resource management techniques.

Cloud systems are consist of a lot of computing and network devices and they are consuming a huge amount of the energy. Thus, activating the appropriate number of devices based on the resource requests is an important issue in terms of the energy saving. Data centers consumed 91 billion Kwh of electricity in 2013 and 50% of that is wasted due to lack of awareness of the traffic according to [1]. This data shows how the accurate prediction of the requests will be increasingly important in the future with an indisputable increase in energy consumption of data centers. Also, precise requests prediction has an effect on the performance of cloud systems. For example, if we deactivate overfull computing or network devices for energy saving purpose, this will cause a serious Service Level Agreement (SLA) violation.

There are many proposals for predicting a value of the requests with several methods. However, the prediction of the quantified number of requests is not the proper strategy for a stable operation of cloud systems because user requests have heavy and frequent fluctuation in cloud systems. The traffic burst and dynamic variation is common phenomenon in cloud or a large size computing cluster because the heterogeneity is the most notable characteristic of cloud workload. Thus, we propose a strategy to predict the probability distribution of the requests in every regular interval instead of predicting the actual value of requests. An advantage of predicting the probability distribution is that we can achieve more flexible prediction. There are always request bursts in networks, and if the system predicts a certain amount of requests based on the average of past requests, cloud systems cannot handle immediate variations of requests. However, if we are aware of the probability distribution of the requests during a given period, we are able to prevent request bursts by predicting the requests which corresponds to a high probability in the predicted probability distribution.

For the prediction function, we collect the requests data history of the cloud and observe the histogram of the data. Based on the histogram analysis of the data, we decide on a probability distribution model fitted to the collected data. Then, we evaluate parameters of the probability distribution model using Maximum Likelihood Estimation (MLE) and save the parameters data to the prediction dataset. After enough data is accumulated for the prediction, the prediction model estimates parameters of the probability distribution with Local Linear Regression (LLR) by using a cyclic window approach. Parameters of the probability distribution are time-dependent data, which means that parameters change with time. So the parameters have obvious patterns during every interval, and they exhibit the same pattern during a certain period every day or every week. So our prediction model will perform the prediction by using the accumulated dataset at every same time point through several periods. In order to reflect a change in trend of the requests, the prediction model maintains the dataset by replacing an old parameter data by recent data.

We use Google cluster-trace data to test our prediction model, which is real measurements of usage requests in Google cluster. One week data is employed for a training data and the following week data is used for testing in order to verify whether the prediction model is accurate. We selected random time point to start collecting data so that the prediction model can be tested in more practical environment. There are three types of data for prediction, the number of task arrivals, CPU requests, and memory requests. Since various user requests are predicted and have large variances in their values, we use Mean Absolute Percentage Error (MAPE) to normalize the error rate and assess the prediction accuracy. Also, the proposed algorithm is compared with previous works by using Normalized Mean Squared Error (NMSE) measurement.

The rest of the paper is organized as follows. Previous work is reviewed in Section II. We introduce mathematical methods and the prediction algorithm in Section III. A system model for an experiment will be demonstrated in Section IV and the

experiment results will be shown in Section V. The conclusions are given in Section VI.

II. Related work

Requests prediction in cloud system is accomplished by many researchers for the automatic scaling of systems.

Akindele A. Bankole *et al*. employ a machine learning technique for a predictive resource provisioning in cloud [9]. Their prediction model is achieved with machine learning techniques: Neural Network, Linear Regression, and Support Vector Machine. They predict the CPU utilization, response time, and throughput based on collected data from virtual machines web servers and database servers. The prediction model generates prediction values in every given minute with machine learning techniques and measure the error rate with MAPE and Root Mean Squared Error (RMSE). However, the prediction model did not show the high prediction accuracy. Their results show 24% prediction error at a certain point of time in predicting the CPU utilization and a 21% error in the response time prediction.

Sedeka Islam *et al*. present more advanced machine learning techniques for predicting resource usage in [4]. The error correction Neural Network (ECNN) and the Linear Regression techniques are employed for prediction. They included a sliding window method to reflect the current state of the system. They generated prediction values based on window sizes and evaluated them with MAPE, PRED(25), RMSE, and R^2prediction accuracy. The CPU utilization data is collected from the Amazon EC2 cloud through the TPC-W benchmark and prediction values are generated with the ECNN and Linear Regression method. The prediction values of CPU utilization has around 19% error rate without the sliding window and has minimum 18.6% error rate when they employ the sliding window.

Many statistical approaches are also applied to prediction in the cloud. Bruno Lopes Dalmazo *et al*. propose the traffic prediction approach based on the statistical model where observations are weighted with Poisson distribution inside a moving window [8]. They consider the past information by means of a sliding window of size λ and this window is applied by weighting the past observations according to the Poisson distribution with parameter λ. Dropbox trace data is employed for testing their prediction model and Normalized Mean Square Error (NMSE) evaluation method is utilized for the error measurement. The prediction model could achieve NMSE values between 0.044 and 0.112. The prediction is ideal when NMSE value is equal to zero and worse when it is greater than one. They could achieve a reasonably accurate prediction with this approach.

They also propose a traffic prediction model with a dynamic window approach [7]. The sliding window size is changed based on variance of the previous window size. A small variance indicates the predicted data is close to the actual data while a high variance means the predicted data is spread out from the mean. Therefore, they update the window size in every prediction interval by considering the size of the variance in the previous prediction. The prediction accuracy is improved from 7.28% to 495.51% compared to the previous statistical model.

III. Algorithm

The prediction model estimates parameters of the probability distribution of future user requests in every predetermined period. We make the assumption that user requests have obvious patterns and the patterns are repeated periodically. For example, the request pattern of all Mondays will be a similar. Hence, the prediction model adopts the history parameter data at the same time point in order to make a prediction about the future.

We introduce three time scales for the prediction periods: Pattern Period (PP), Target Period (TP) and Utilization Period (UP). The PP is a cyclic interval that exhibits pattern repetition. The TP is a unit duration for which we want to make a prediction. The UP is a cyclic window that we use for predicting the activities in TP. In the example, when we intend to predict the request distribution during a certain Monday by assuming the same pattern is repeated in every week. Then, we can set the TP to a day and the PP to a week because we assume the pattern of a day is repeated every week. If we only use the past Mondays′ data for the prediction, the UP becomes a day.

Since we assume patterns are repeated on the every PP, any time duration that we want to make a prediction corresponds to a certain TP on the PP. Therefore, we can predict the request distribution during any time interval by corresponding them to a certain TP on the PP. For the precise prediction, we accumulate data for several PPs. Although patterns of the traffic distributions will be similar on the same time point in every PP, the traffic amount will be different. In other words, we can say the distribution of the traffic shows similar form in every Monday, but we cannot ensure that the amounts of traffic will be same. Therefore, the prediction model can achieve higher prediction accuracy by accumulating data during several PPs.

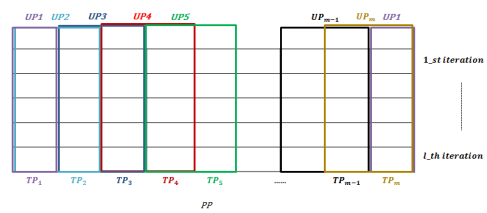


Figure 1: Prediction dataset

To implement prediction, the prediction dataset saves the past data in a $m \times l$ matrix. m represents the number of TPs on the PP and l denotes the number of PPs we accumulate. In Figure 1, each vertical block corresponds to saved parameters of the probability distribution in each TP during a PP. We start to stack the data from the first block of the first iteration. If we reach the TP_m's block, which is the last TP, we move to the second iteration and stack the data from the first block of the second iteration, which means we have saved data during a PP. When the matrix is full, we go back to the first block of the first iteration and replace the old data to reflect the tendency of recent requests.

Any time duration that we want to predict the traffic distribution for can be related to a certain TP on the matrix. In order to make a prediction, we employ the UP data. In Figure 1, we can see that distribution parameters of the TP_m can be predicted by using the UP_m. We can set the size of the UP depending on how many previous TPs will affect to the state of the current TP. For example, if we set the UP to two days, we can say previous Sunday and Monday′s history parameters will affect to next Monday′s traffic distribution as we can see in Figure 1.

In this paper, we forecast the number of task arrivals during each target period. The first step of prediction, the prediction model constructs a histogram of user requests in every target period to observe distributions of requests. Then, it fits a probability distribution to the distribution of requests. When the probability distribution kernel is decided for fitting, we will adopt MLE in order to obtain parameters of the probability distribution in every observed period and the parameters are saved on the dataset. After the dataset accumulates enough data for prediction, it is able to predict parameters of a following target period. Local Linear Regression (LLR) will be employed to predict parameters of the future requests.

A. Histogram

A histogram is the graphical representation of the distribution of data. We can observe frequencies and overall distribution of given data through a graphical representation.

Histograms of requests are constructed in every regular interval to observe the distribution of requests. After observe the histograms, the prediction model decides which of the probability distribution model will be the closest to the actual distribution of requests.

B. Maximum Likelihood Estimation (MLE)

MLE estimates parameters of a probability distribution when there are data corresponding to the probability distribution model. Our prediction model employ the Poisson distribution based on the histogram observation of the experiment data. Depending on accumulated request data, the prediction model will induce the Poisson distribution parameter by using MLE in every TP.

Poisson distribution has the only parameter λ. Since the prediction model has data through observation, the number of task arrivals, CPU, and memory request, it is able to induce parameter λ by using MLE method. If we observe n independent datasets $X1$, $X2$, $X3$, ..., Xn $i.i.d$ Poisson random variables, maximum likelihood function L(λ) will be:

$$L(\lambda) = \frac{\lambda^{X_1} e^{-\lambda}}{X_1} \frac{\lambda^{X_2} e^{-\lambda}}{X_2} \cdots \frac{\lambda^{X_n} e^{-\lambda}}{X_n} = \prod_{i=1}^{n} \frac{\lambda^{X_i} e^{-\lambda}}{X_i} \quad (1)$$

If we take log in the equation, log likelihood function becomes:

$$\begin{aligned} l(\lambda) &= \sum_{i=1}^{n} (X_i \log \lambda - \lambda - \log X_i!) \\ &= log\lambda \sum_{i=1}^{n} X_i - n\lambda - \sum_{i=1}^{n} \log X_i! \end{aligned} \quad (2)$$

We find maximum of λ by finding the derivative of equation:

$$l'(\lambda) = \frac{1}{\lambda} \sum_{i=1}^{n} X_i - n = 0 \quad (3)$$

, which implies the λ that has closest distribution with observed histogram is:

$$\hat{\lambda} = \frac{\sum_{i=1}^{n} X_i}{n} = \bar{X} \quad (4)$$

C. Local Linear Regression (LLR)

LLR is one of the most accurate kernel smoother techniques for estimating a real value function, when no parametric model for this function is known. Since the prediction is achieved using short range of time period which does not guarantee linearity or any parametric model, we employ the non-parametric statistical model for the prediction. LLR combines much of the simplicity of linear least square regressions by fitting the line about the given k number of points with the N number of observed points. In the prediction model, the k is corresponded to the number of TPs on UPs and N is equivalent to the number of history parameters we will employ for the prediction, which is called to bandwidth. After fitting the line at every given point, the estimation functions $\hat{Y}(TP_k)$ are achieved as a value function with the k numbers of values. $K_{h\lambda}(X_u, X_i)$ be a kernel defined by:

$$K_{h\lambda}(X_u, X_i) = D(\frac{\|X_i - X_u\|}{h_\lambda(X_u)}) \quad (5)$$

The $D()$ is a positive real valued function in (5), which is decreasing when the distance between X_i and X_u increases. The X_u is the given points and the X_i is one of the observed data around X_u. Commonly used kernels include the Epanechnikov, biweigh and Gaussian function. For one dimension data, least-square method is employed for obtaining function value on the X_u.

$$\min_{\alpha(X_u),\beta(X_u)} \sum_{i=1}^{N} K_{h\lambda}(X_u, X_i) \Big(Y(X_i) - \alpha(X_u) - \beta(X_u) X_i \Big)^2 \quad (6)$$

The N is the number of history parameter near X_u, that we will employ in (6). Since we obtain parameters in each TP_i by using MLE, the minimum of $\alpha(X_u)$ and $\beta(X_u)$ can be achieved by solving the weighted least square problem (6). If we assume the estimation function on X_u is $\hat{Y}(X_u) = \alpha(X_u) + \beta(X_u)X_u$, the closed form solution of the estimation function is like:

$$\hat{Y}(X_u) = (1, X_u)(B^T W(X_u) B)^{-1} B^T W(X_u) y \quad (7)$$

where:

$$y = (Y(X_1), \ldots, Y(X_N))^T \quad (8)$$

$$W(X_0) = diag\Big(K_{h\lambda}(X_u, X_i) \Big)_{N \times N} \quad (9)$$

$$B^T = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ X_1 & X_2 & \ldots & X_N \end{pmatrix} \quad (10)$$

By repeating this process about all given k points, X_u, we can get real value estimation functions $\hat{Y}(X_u)$ about k points.

D. Cyclic Window Learning Algorithm

We describe the algorithm that predicts parameters of the probability distribution of the future target periods by using a cyclic window approach. We assume the dataset has enough past data for the prediction and determined a probability distribution model for MLE. The algorithm will employ LLR to predict the probability distribution parameters of the future target periods and MLE for updating dataset.

The Algorithm 1 obtains the predicted parameters of the target period (PT_t) for the prediction and actual parameters of target period (AT_t) to update dataset at every time period at time t. The m means the number of TPs included in a UP, which is equivalent to a window size. The n is the number of TPs during a PP. The l represents how many cycles of PPs will be stacked on the prediction dataset ($PData$). The $PData$ is the prediction dataset has the $m \times l$ dimension. The $TPdata_t$ means observed requests during an interval of the TP at time

Algorithm 1 Cyclic Window Learning Algorithm

Require: $PData_{m\times l}$,$TPdata_t$, m, n, and l
Ensure: PT_t and AT_t

1: $t = 1$,$p = 1$, and $w = 1$
2: **while** System operate **do**
3: **if** $p < n$ **then**
4: $UT_t = PData_{(m-n-p)\times l} : PData_{m\times l}$, $PData_{1\times l} : PData_{p\times l}, \forall l$
5: **else**
6: $UT_t = PData_{(p-n)\times l} : PData_{p\times l}, \forall l$
7: **end if**
8: Implement LLR in terms of UT_t
9: $PT_t = \hat{Y}(UT_p)$ and select kth value
10: Update databased with actual parameter
11: $AT_p = MLE(TPdata_t)$
12: Update databased with AT_p
13: $PData(p, w) = AT_p$
14: $t = t + 1$
15: **if** $p < m$ **then**
16: $p = p + 1$
17: **else**
18: $p = 1$
19: **if** $w < l$ **then**
20: $w = w + 1$
21: **else**
22: $w = 1$
23: **end if**
24: **end if**
25: **end while**

t. The prediction model will obtain parameters of the TP at time t with MLE by using this data, $TPdata_t$.

We initialize variables: t, p, and w (line 1). The t represents how many unit periods are passed after the algorithm starts and the p is a corresponding position of the TP at time t. Since we return to the initial position on the PP when the p reaches the end of the PP, the p becomes a cyclic number from 1 to m. The w is a row position on the $m \times l$ $PData$. Too much data requires the complexity of the prediction and consumes too much time for the prediction. Therefore, we stack only the appropriate number of cycles on $PData$ by replacing the old data. The w is which row position in the $PData$ will be updated.

First, the algorithm collects the data for the prediction from the $PData$ to the UT_t (line 2 $-$ line 7). If the position p is less than the window size n, we need to employ the data from the end of the $PData$ because the p is cyclic. So we collect the data from $m-n-p$th to mth columns' data and from the first to pth columns of $PData$ (line 3 $-$ line 4). If the p is greater than the window size n, we collect previous n columns data from the point p on the $PData$ (line 5 $-$ line 7).

We implement LLR about the collected data, UT_t, and obtain the prediction value about time t (line 8 $-$ line 9). In order to update the $PData$, we need to obtain the actual probability distribution parameters of observed requests. We apply MLE about the data $TPdata_t$ and update corresponding data block in $PData$ (line 10 $-$ line 13). We update the position on the $PData$ for next prediction. We update the t for predicting next time point (line 14). We just increase the p if the position of the p is still on PP. If the p exceeds the size of PP, m, it goes back to the initial position of the PP. We also update the w when p goes back to the initial position because that means one row of $PData$ is filled with new data. The w increases when the p increases. However, the w becomes one if the w exceeds l, which is vertical size of the matrix $PData$ (line 15 $-$ line 23).

IV. System model

A. Google Cluster Data collection (Arriving Tasks, CPU, and Memory)

The experiment is implemented with Google cluster-usage traces data [13]. Google cluster is a set of computing resources composed of thousands of machines. A job is composed of several tasks which can be processed separately. So each task will be a unit of a process. We consider the number of task arrivals, CPU requests, and Memory requests of tasks. Each task has a timestamp which represents when the task arrives at the cluster. Therefore, the distribution of the number of task arrivals can be observed by using the timestamp. The cluster data also contains the CPU and memory requests of each task. The CPU requests show core counts or core-seconds/second of tasks and the memory requests represent how much bytes each task requires. The cluster starts measurement 600 seconds after the system is operated and has accumulated data for one month approximately. We select a random point to collect data for the prediction model. One weak data is sampled as a training dataset for the prediction modeling and the following week data is employed to test the accuracy of the prediction model.

V. Experiment

A. Histogram analysis of data

The purpose of the prediction is forecasting the traffic of short period for dynamic system management. However, we found that too short TP is not enough to observe apparent patterns of distribution of requests according to our experiment. Based on our empirical observation, the histogram represents the obvious pattern when we have accumulated at least 30 minutes data. Thus, we decide to set the prediction model makes the prediction in every 30 minutes.

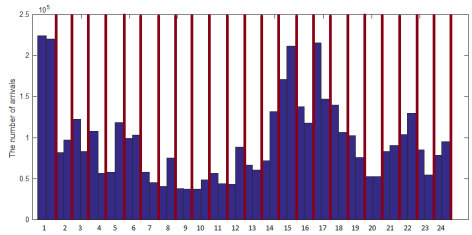


Figure 2: The number of tasks during the day

As patterns are observed in Figure 2, the number of task arrivals show an apparent pattern depending on the time. Figure 2 represents the number of task arrivals in every 30 minute during the first day. For the first 4 hours and the last 10 hours, Figure 3 presents high rate of incoming tasks.

Histograms of each target period are different depending on the total number of task arrivals. Histograms have high peaks in the more right side during the busy hours and they have high peaks in the more left side during the free hours.

Figure 3 is a histogram of the first hour. The first histogram exhibits the distribution of task arrivals during the first half hour and the second histogram presents the distribution of the second half hour. The first histogram has a peak in more right

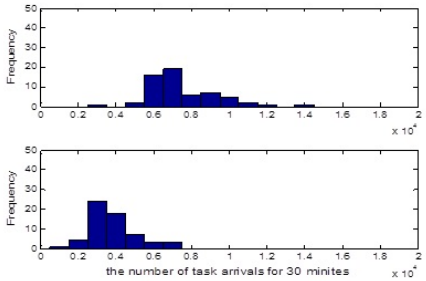


Figure 3: Distribution of the first day

side than the second histogram because the cluster received higher task arrivals during the first half hour than the second half hour.

This trend is similar to the Poisson distribution. The Poisson distribution has a high peak in the more right side when the rate parameter λ is high. Therefore, the observed data will be fitted to the Poisson distribution by using MLE to obtain parameters of the Poisson distribution in every duration.

B. Maximum Likelihood Estimation of distributions

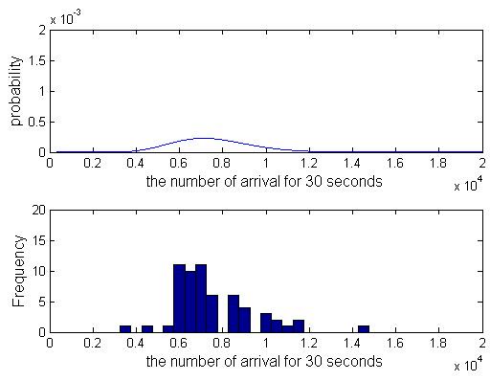


Figure 4: MLE of the first hour

MLE is employed to obtain parameters of the Poisson distribution in every target period. The estimated Poisson distribution are achieved about the first half hour histogram by using MLE in Figure 4. The first graph shows the Poisson distribution with estimated parameters and the second graph is a histogram of task arrivals during the first half hour. We can observe the estimated Poisson distribution has a similar distribution with the histogram of data. The prediction model implements MLE in every 30 minute and saves them to the prediction dataset.

C. Time Dependent Parameter Estimation

Parameters are induced from the request data in every 30 minute. Parameters generated at the same time point on the PP are stacked in the same column of the prediction dataset. Predicted parameters are achieved by implementing LLR about the corresponding UP. For example, if we implement LLR about the UP including the first to the 10th TP to predict the duration corresponding to the 10th TP, the last point value of LLR function becomes the prediction value of the 10th target period. Prediction values are changed depending on how to set the utilization periods and how much bandwidth we adopt for LLR. The bandwidth represents how many near data are included when we predict the function value of a certain point.

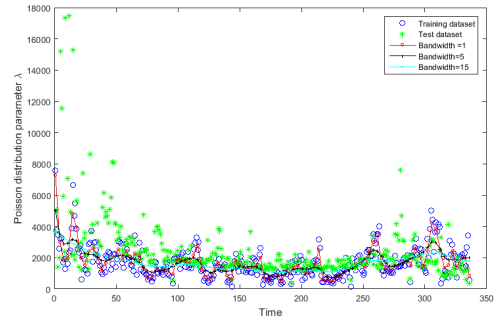
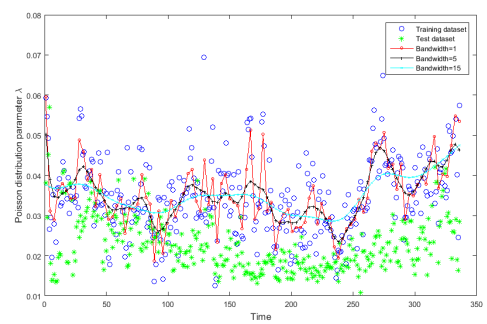
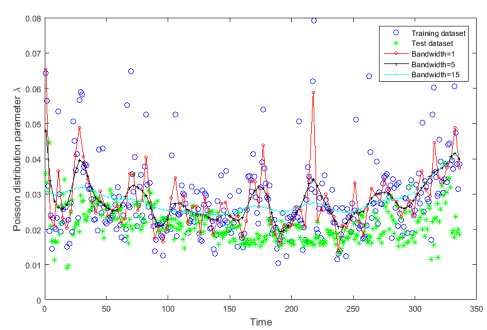


Figure 5: Poisson distribution parameter λ estimation of arrival tasks

Figure 5 represents the prediction of Poisson distribution parameter λ of arriving tasks during week. Since we set the TP to 30 minutes, we have 336 target periods during the week. Blue points represent parameter values of training dataset in each TP and green points are parameter values of test dataset in each TP. Solid lines represent parameter prediction values for different values of bandwidth. We set the UP to 25 hours in Figure 5, which means that prediction value is obtained base on the last 25th hours data. Parameter λ is equivalent to mean number of arrivals during 30 minutes. We can observe that the graph has a regularly repeated pattern. It has seven high peaks in the graph, which means similar patterns repeated during the week.



(a)



(b)

Figure 6: Poisson distribution parameter λ estimation (a)CPU (b)Memory

Prediction values of parameters are obtained about the CPU and memory requests as well in the same method. Figure 6 is the parameter prediction of the CPU and memory requests. We

obtain prediction values in the same method with task arrivals.

D. Error assessment

In order to quantify an accuracy of the prediction, we measure Mean Absolute Percentage Error (MAPE) between the prediction data and the test dataset. MAPE expresses an error rate as a percentage. So we can compare the prediction accuracy of task arrivals, CPU requests, and memory requests with a normalized error rate value.

$$MAPE = \frac{1}{n}\sum_{j=1}^{n}\frac{|P_j - T_j|}{T_j} \tag{11}$$

The P_j is a predicted value of a target value, T_j. MAPE value is equal to zero when the prediction model is the perfect fit to the target value and increased when the prediction is not properly fit to target values.

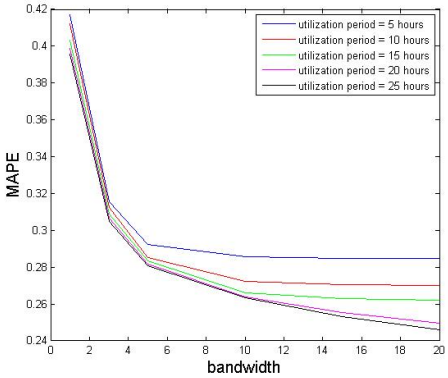


Figure 7: MAPE measurement of task arrivals prediction

Figure 7 is MAPE measurement graph of the Poisson distribution parameter λ. The parameter λ has MAPE range between 0.3885 and 0.5194. If we consider the ideal state of MAPE is zero, the prediction model has enough prediction accuracy. The prediction model could achieve a higher accuracy with the longer UP, which is equivalent the window size because increasing the UP means employing more previous data for the prediction. However, the large UP requires more complexity of a computation and consumes more time. In other words, a proper selection of the UP is required to satisfy both of the prediction accuracy and the computation time. Choosing the best bandwidth is also an important issue in order to reduce the prediction error. Too small bandwidth causes very spiky estimates while large bandwidth leads over smoothing. If data values are spread widely, the smaller bandwidth will not acquire the higher prediction accuracy.

The proposed algorithm is compared with fractal differential equation modeling based prediction method proposed in [14]. The Mean Square Error (MSE) of the proposed algorithm and comparison prediction model is measured for CPU and memory requests and normalized by baseline prediction algorithm. Auto-regressive predictor is used for baseline model by employing 16 previous time slot value, which is equivalent to using 16 UP in the proposed algorithm. In memory request prediction, the proposed model achieves 25% reduced MSE compare to fractional modeling based predictor and 84% reduced MSE than the auto-regressive predictor. In CPU request prediction, the proposed algorithm shows slightly advanced prediction accuracy than fractal modeling based predictor, 3% reduced MSE but it reduces 75% MSE compare to the baseline prediction model.

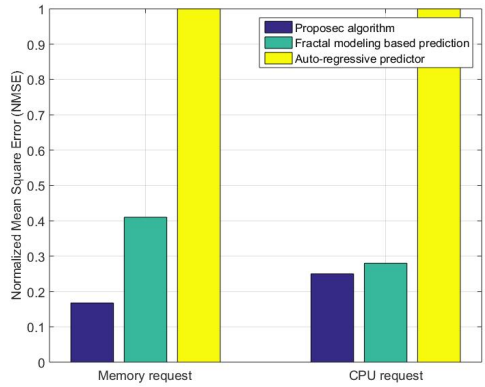


Figure 8: MAPE measurement of task arrivals prediction

VI. Conclusions

We propose a novel approach for the request prediction in cloud systems. Instead of predicting an actual amount of requests, our prediction model estimates parameters of the probability distribution during the given period. We accumulate the historical data of the system and a cyclic window approach to utilize data with MLE and LLR. In the experiment with Google cluster-trace data, we could ensure advanced performance of the prediction algorithm. Our prediction model achieves the very low level of error rates in predicting the probability distribution parameters.

References

[1] J. Whitney and P. Delforge, *Data center efficiency assessment*, Issue paper on NRDC (The Natural Resource Defense Council). August 2014.

[2] T. P. Minksa, *Estimating a gamma distribution*, research.microsoft.com/ minka/ papers/newton.html. 2002.

[3] J. Fan, *Local linear regression smoothers and their minmax efficiencies*, Ann. Statist, vol 21, No. 1, pp 196-216. November 1993.

[4] S. Islam, J. Keung, K. Lee, and A. Liu, *Empirical prediction models for adaptive resource provisioning in the cloud*, Future Generation Computer Systems, vol 28, pp 155-162. January 2012.

[5] E. Caron and F. Desprez, *Forecasting for grid and cloud computing on demand resources based on pattern matching*, Proceeding on IEEE Cloud Computing Technology and Science (CloudCom), pp 456-463. November 2010.

[6] M. Li and S. C. Lim, *Modeling network traffic using generalized Cauchy process*, Physica A: Statistical Mechanics and its application, vol 387, pp 2584-2594. April 2008.

[7] B. L. Dalmazo, J. P. Vilela, and M. Curado, *Onlinetraffic prediction in the cloud: a dynamic window approach*, Proceeding on IEEE Cloud and Green Computing (CGC), pp 9-14. August 2014.

[8] B. L. Dalmazo, J. P. Vilela, and M. Curado, *Predicting traffic in the cloud: a statistical approach*, Proceeding on IEEE Cloud and Green Computing (CGC), pp 121-126. October 2013.

[9] A. A. Bankole and S. A. Ajila, *Predicting cloud resource provisioning using machine learning techniques*,Proceeding on IEEE Electrical and Computer Engineering (CCECE), pp 1-4. May 2013.

[10] T. Wood, L. Cherkasova, K. Ozonat, and P. Shenoy, *Profiling and modeling resource usage of virtualized application*,Proceeding of the 9th ACM/IFIP/USENIX International conference on middleware, pp 386-387. December 2008.

[11] L. Lio, X. Jin, G. Min, and L. Xu, *Real-Time diagnosis network anomaly based on statistical traffic analysis*,Proceeding on IEEE Trust, Security, and Privacy in Computing and Communications (TrustCom), pp 264-270. June 2012.

[12] J. Yang, C. Liu, Y. Shag, Z. Mao, and J. Chen, *Workload predicting based automatic scaling in service clouds*,Proceeding on IEEE 6th International Conference on Cloud Computing, pp 810-815. June/July 2013.

[13] Google cluster-trace data, https://code.google.com/p/googleclusterdata/

[14] S. Chen, M. Ghorbani, Y. Wang, P. Bogdan, M. Pedram, *Trace-based analysis and prediction of cloud computing user behavior using the fractal modeling technique*, 2014 IEEE International Congress on Big Data, pp 733-739. June/July 2014.

[15] R. N. Calheiros, E. Masoumi, R. Ranjan, R. Buyya, *Workload prediction using ARIMA model and its impact on cloud applications' QoS*, IEEE Transactions on Cloud Computing, Vol. 3, No. 4, pp 449-458. October-December 2015.