Requests Prediction in Cloud with a Cyclic Window Learning Algorithm

Min Sang Yoon  
Iowa State University

Ahmed E. Kamal  
Iowa State University, kamal@iastate.edu

Zhengyuan Zhu  
Iowa State University, zhuz@iastate.edu

Follow this and additional works at: https://lib.dr.iastate.edu/stat_las_conf

Part of the Statistics and Probability Commons

Recommended Citation
Yoon, Min Sang; Kamal, Ahmed E.; and Zhu, Zhengyuan, "Requests Prediction in Cloud with a Cyclic Window Learning Algorithm" (2016). Statistics Conference Proceedings, Presentations and Posters. 11. https://lib.dr.iastate.edu/stat_las_conf/11

This Conference Proceeding is brought to you for free and open access by the Statistics at Iowa State University Digital Repository. It has been accepted for inclusion in Statistics Conference Proceedings, Presentations and Posters by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Requests Prediction in Cloud with a Cyclic Window Learning Algorithm

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).

Keywords
Predictive models, Probability distribution, Data models, Maximum likelihood estimation, Histograms, Prediction algorithms, Linear regression

Disciplines
Statistics and Probability

Comments
This is a manuscript of a proceeding published as Yoon, Min Sang, Ahmed E. Kamal, and Zhengyuan Zhu. "Requests prediction in cloud with a cyclic window learning algorithm." In 2016 IEEE Globecom Workshops (GC Wkshps), (2016). DOI: 10.1109/GLOCOMW.2016.7849022. Posted with permission.
Requests Prediction in Cloud with a Cyclic Window Learning Algorithm

Min Sang Yoon\textsuperscript{a}, Ahmed E. Kamal\textsuperscript{b}, and Zhengyuan Zhu\textsuperscript{c}

\textsuperscript{a} Department of Electrical and Computer Engineering  
\textsuperscript{b} Department of Statistics  
\textsuperscript{c} Iowa State University, IA, 50010  
\{my222\textsuperscript{a}, kamal\textsuperscript{b}, zhuz\textsuperscript{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 an algorithm that predicts a 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 implemented about the number of task arrivals, CPU requests, and memory requests. Then the accuracy of prediction is measured with Mean Absolute Percentage Error (MAPE).

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. One advantage of Cloud systems is automatic scaling and management. Therefore, the accurate prediction of the requests in the Cloud systems will bring improvement in operating systems with more advanced resource management techniques.

Cloud systems consists of a lot of computing and network devices. Thus, activating the appropriate number of devices based on the requests is an important issue in terms of saving energy. Data centers consumed 91 billion kilowatt-hours of electricity in 2013 and 50 percent 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. If we inactivate overfull computing or network devices for energy saving purpose, it will cause a bottleneck and delay in handling requests.

There are many research that propose predicting an expected 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 fluctuation frequently. Thus, we propose a strategy to predict parameters of the probability distribution of the requests in every regular interval instead of predicting the actual amount of requests. An advantage of predicting the probability distribution is that we can make the more flexible prediction. There are always request bursts in networks. If the system predicts a certain amount of requests based on the average of past requests, Cloud systems cannot prepare 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, we collect the history requests data of the Cloud and observe the histogram of the data. Based on the histogram analysis of the data, we decide the probability distribution model fitted to the collected data. Then, we induce parameters of the probability distribution model with Maximum Likelihood Estimation (MLE) and save the parameter data to the prediction dataset. After we accumulate enough data 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 the time-dependent data, which means the parameters are changed as time goes by. So the parameters have obvious patterns during every interval. It will reveal the same pattern during a certain period in every day or every week. So our prediction model will make 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 with the 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 a large variance in its values, we use Mean Absolute Percentage Error (MAPE) to normalize the error rate and assess the prediction accuracy.

The rest of the paper is organized as follows. Previous works are reviewed in Section II. We will 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 machine learning technique for a predictive resource provisioning in Cloud [10]. Their prediction model is achieved with machine learning
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 one 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 $PPs$.

\[\text{Fig. 1: Prediction dataset}\]

To implement prediction, the prediction dataset saves the past data in a $m \times l$ matrix. $m$ represents the number of $TP$s on the $PP$ and $l$ denotes the number of $PP$s we accumulate. In Figure 3, 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 3, 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 $TP$s 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 Fig. 3.

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 model for MLE. The algorithm will employ LLR to predict the probability distribution parameters of the future target periods and MLE for updating dataset.

Poisson distribution has the only parameter \( \lambda \). Since the prediction model has data through observation, the number of task arrivals, CPU, and memory request, it is able to induce parameter \( \lambda \) by using MLE method. If we observe \( n \) independent datasets \( X_1, X_2, X_3, \ldots, X_n \) iid Poisson random variables, maximum likelihood function \( L(\lambda) \) will be:

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

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

\[
l(\lambda) = \sum_{i=1}^{n} (X_i \log \lambda - \lambda - \log X_i) = \log \lambda \sum_{i=1}^{n} X_i - n \lambda - \sum_{i=1}^{n} \log X_i! \tag{2}
\]

We find maximum of \( \lambda \) by finding the derivative of equation:

\[
l'(\lambda) = \frac{1}{\\lambda} \sum_{i=1}^{n} X_i - n = 0 \tag{3}
\]

which implies the \( \lambda \) that has closest distribution with observed histogram is:

\[
\hat{\lambda} = \frac{\sum_{i=1}^{n} X_i}{n} = \bar{X} \tag{4}
\]

C. Local Linear Regression (LLR)

LLR is one of the kernel smoother techniques for estimating a real value function, when no parametric model for this function is known. 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:

\[
K_{h\lambda}(X_u, X_i) = D_i(\frac{||X_i - X_u||}{h\lambda(X_u)}) \tag{5}
\]

The \( D(\cdot) \) 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 \( X_u \).

\[
\min_{\alpha(X_u), \beta(X_u)} \sum_{i=1}^{N} K_{h\lambda}(X_u, X_i) \left( Y(X_i) - \alpha(X_u) - \beta(X_u)X_i \right)^2 \tag{6}
\]

The \( N \) is the number of history parameter near \( X_u \), that we will employ in (6). Since we obtain parameters in each \( TP \) 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:

\[
\hat{Y}(X_u) = (1, X_u)(B^T W(X_u) B)^{-1} B^T W(X_u) y \tag{7}
\]

where:

\[
y = (Y(X_1), \ldots, Y(X_N))^T \tag{8}
\]

\[
W(X_u) = diag(K_{h\lambda}(X_u, X_i))_{N \times N} \tag{9}
\]

\[
B^T = \begin{pmatrix} 1 & X_1 & X_2 & \cdots & X_N \end{pmatrix} \tag{10}
\]

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 target period \((PT)\) for the prediction and actual parameters of target period \((AT)\) to update dataset at every time period. 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 \( PP \)'s will be stacked on the prediction dataset \((PData)\). The \( PData \) is the prediction dataset has the \( m \times l \) dimension. The \( TData \) means observed requests during an interval of the \( TP \) at time \( t \). The prediction model will obtain parameters of the \( TP \) at time \( t \) with MLE by using this data, \( TData \).

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 \( TP \) about 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 \( UT \) (line 2 – line 7). If the position \( p \) is
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: $PT_t = Y(UT_p)$ and select $k_{th}$ 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

IV. System Model

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

The experiment is implemented with Google cluster usage traces data [14]. 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. Patterns of data

All experiments are conducted by Matlab. Basically, user requests have a regular pattern. Requests increases during the daytime and weekdays more than the nighttime and weekend. We could observe some patterns by analyzing the Google trace data. The start time of trace is randomly chosen in data. So we do not know what the exact date or time of measurement points. However, our assumption is arriving tasks will show regular patterns in every interval.

![Fig. 2: Daily Pattern Analysis](image)

Figure 2 shows patterns of task arrivals during a week. The number of task arrivals is counted for every one hour, so we could observe how many tasks arrive at the cluster in every hour during the week. The hourly pattern shows high peaks for the first 4 hours and the last 10 hours. So we can assume daytime starts around the tenth hours from the measurement because the number of tasks is increased from the tenth hour and end at the fourth hour in Figure 2. If we see the daily pattern, the first two days and the last two days show higher requests. In the same way, we can assume weekdays start on the fourth day from the measurement by assuming requests increases during weekdays rather than the weekend.

The daily pattern analysis of task arrivals presents obvious patterns of the incoming requests. Therefore, the cyclic data collection should be an effective approach for the prediction.

The daily pattern analysis of the CPU and memory requests also have demonstrated the similar patterns with task arrivals. The CPU and memory requests have had high requests when the number of task arrivals increases and had low requests during free periods.

According to our observation, we have decided to set the $PP$ to one week because we could observe hourly and daily patterns of requests repeatedly during each week.
B. Histogram analysis of data

According to the experiment, we found that too short $TP$ is not enough to observe apparent patterns of distribution of requests. Thus, the $TP$ is set to 30 minutes based on empirical observation. The prediction model predicts distribution parameters in every 30 minutes.

As patterns are observed in Figure 2, the number of task arrivals show an apparent pattern depending on the time. Figure 3 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 4 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 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 $\lambda$ 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.

C. Maximum Likelihood Estimation of distributions

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 5. 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.

D. 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 $10_{th}$ $TP$ to predict the duration corresponding to $10_{th}$ $TP$, the last point value of LLR function becomes the prediction value of the $10_{th}$ 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 6 represents the prediction of Poisson distribution parameter $\lambda$ 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 Fig. 4, which means that prediction value is obtained based on the last $25\times 2$ hours data. Parameter $\lambda$ 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.

Figure 7: Poisson distribution parameter $\lambda$ estimation
(a)CPU (b)Memory

Prediction values of parameters are obtained about the CPU and memory requests as well in the same method. Figure 7 is the parameter prediction of the CPU and memory requests. We obtain prediction values in the same method with task arrivals.

E. 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.

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

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 8 is MAPE measurement graph of the Poisson distribution parameter $\lambda$. The parameter $\lambda$ 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.

Figure 9: MAPE measurement of task arrivals prediction

We compare the prediction accuracy of cyclic window learning algorithm with Poisson based statistical algorithm that suggested in [9]. We selected the prediction values obtained with bandwidth 20 and compared them with the prediction result of Poisson based statistical algorithm in Figure 9. In the result, we could observe that the cyclic window learning algorithm presents slightly better prediction accuracy when we employed small $UP$s. However, the cyclic window learning algorithm could improve the performance up to 13.6% when we employ more $UP$s data.

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. Minka, Beyond newton’s method, research.microsoft.com/ minka/ papers/newton.html. 2000.
[3] T. P. Minka, Estimating a gamma distribution, research.microsoft.com/ minka/ papers/newton.html. 2002.
[4] J. Fan, Local linear regression smoothers and their minmax efficiencies, Ann. Statist, vol 21, No. 1, pp 196-216. November 1993.
[5] 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.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] 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.
[11] T. Wood, L. Cherkasova, K. Ozonat, and P. Shenoy, Profiling and modeling resource usage of virtualized application, Proceeding of the 9th ACM/SIGP/USENIX International conference on middleware, pp 386-387. December 2008.
[12] 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.
[13] 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.
[14] Google cluster-trace data, https://code.google.com/p/googleclusterdata/