KSwSVR: A New Load Forecasting Method for Efficient Resources Provisioning in Cloud

Rongdong Hu[†], Jingfei Jiang[†], Guangming Liu[†‡], Lixin Wang[†]
[†]School of Computer, National University of Defense Technology, Changsha, China
[‡]National Supercomputer Center, Tianjin, China
{rongdonghu, jingfeijiang, lixinwang}@nudt.edu.cn, liugm@nscc-tj.gov.cn

***Abstract*—Cloud provider should ensure QoS while maximizing resources utilization. One optimal strategy is to timely allocate resources in a fine-grained mode according to the actual resources demand of applications. The necessary precondition of this strategy is obtaining future load information in advance. We propose a multi-step-ahead load forecasting method, KSwSVR, based on statistical learning theory which is suitable for the complex and dynamic characteristics of the cloud computing environment. It integrates an improved support vector regression algorithm and Kalman smoother. Public trace data taken from multi-types of resources were used to verify its prediction accuracy, stability and adaptability, comparing with AR, BPNN and standard SVR. CPU allocation experiment indicated that KSwSVR can effectively reduce resources consumption while meeting Service Level Agreements requirement.**

Keywords- Load forecasting; Resources provisioning; Support vector regression; Kalman smoother; Cloud computing

I. INTRODUCTION

Cloud computing offers near-infinite amount of resources capacity at a competitive rate, and allows customers to obtain resources on-demand with pay-as-you-go pricing model. The proliferation of cloud computing has resulted in the establishment of large-scale data centers containing thousands of computing nodes and consuming enormous amounts of electrical energy. The reason for this extremely high energy consumption is not just the quantity of computing resources and the power inefficiency of hardware, but rather the inefficient usage of these resources. The data collected from more than 5000 production servers over a six-month period have shown that servers operate only at 10% ~ 50% of their full capacity most of the time [1]. Another problem is the narrow dynamic power range of servers. Even completely idle servers still consume about 70% of their peak power [2].

A number of practices are reported to be applicable to achieve energy efficiency, such as improvement of applications algorithms, energy efficient hardware, Dynamic Voltage and Frequency Scaling (DVFS), terminal servers and thin clients, etc. Cloud computing mainly leverages the virtualization technology to address the energy inefficiency problem [3]. It allows cloud providers to create multiple Virtual Machines (VM) on a host to improve the resources utilization. By using live migration [4], the VM can be dynamically consolidated to the minimal number of physical nodes according to their current resources requirements. The reduction in energy consumption can be achieved by switching idle nodes to low-power modes.

However, virtualization also creates a new problem. One essential requirement of a cloud computing environment is providing reliable QoS defined in terms of Service Level Agreements (SLA). Modern applications often experience highly variable workloads causing dynamic resources usage patterns. The consolidation of VMs can lead to performance degradation when an application encounters increasing demands resulting in an unexpected rise of resources usage. This may lead to SLA violation — increasing response times, time-outs, or failures. Over-provisioning may help to ensure SLA, but is clearly inefficient. The optimal strategy is to timely adjust resources provisioning according to the actual demands of applications. The precondition of this approach is to find out the future workload.

The focus of this work is on improving the efficiency of resources provisioning by forecasting the load of multi-types of resources in cloud. We propose a multi-step-ahead load prediction method, KSwSVR, mainly based on a statistical learning technology, Support Vector Regression (SVR), which is suitable for the complex and dynamic characteristics of the cloud computing environment. KSwSVR integrates our improved SVR algorithm and Kalman smoother. Experiments with public trace data have shown that, in comparison with AutoRegressive (AR), Back-Propagation Neural Network (BPNN) and standard SVR, KSwSVR always has the minimum prediction error. Furthermore, KSwSVR is very stable, i.e. its prediction error increases quite slowly as the predicted steps increasing. We also verified its broad adaptability with real trace data of various resources related to network, CPU, memory and storage systems. Finally, the usefulness of the prediction result is demonstrated in a CPU allocation experiment. With the assistance of KSwSVR, dynamic provisioning strategy can save 17.20% ~ 48.12% CPU capacities under different SLA levels, comparing with static provisioning.

The rest of this paper is organized as follows. Section 2 describes the background and motivation. Section 3 discusses the detailed design and implementation of the proposed approach and Section 4 presents the experimental evaluation. Section 5 examines the related work and Section 6 makes the conclusion.

This work is supported by the National Natural Science Foundation of China (NSFC) under Grant No.61272146.

978-0-7695-5026-8/13 $26.00 © 2013 IEEE
DOI 10.1109/SCC.2013.67

II. BACKGROUND AND MOTIVATION

A. Load in Cloud

With the proliferation of cloud data centers, it is quite common today to lease virtual machines to host applications instead of physical machines. Cloud users typically pay for a statically configured VM size, irrespective of the actual resources consumption of application (e.g., Amazon EC2). This charging mode is obviously unreasonable especially for applications with variable load. It is usually difficult for cloud users to figure out which size of VM is suitable for their applications as their loads are rarely constant. They certainly do not like to pay for the resources they hold but not use when the load is light. Furthermore, they have to face the risk of performance degradation when load is heavy. It would be highly desirable for cloud providers to provide dynamically finer-grained online scalable services that allocate resources according to application's demand. That could encourage the customer to pay a higher price for better service compared to paying a flat fee based on the size of their VMs. Moreover, cloud providers can have the flexibility to dynamically optimize the resources allocation, improve resources utilization and achieve maximum benefit.

Timely dynamic fine-grained online scalability will greatly increase the pressure on management system to rapidly detect and resolve SLA violation problems. Typically, problem detection is done by specifying threshold tests. Examples include: "Do ping response times exceed 0.5s?" and "Are HTTP operations greater than 12 per second on server?" Unfortunately, once the detection occurs, there is often little time to take corrective actions. In addition, if the load changes dramatically, there will be frequent SLA violations. It is desirable that the resources can be acquired earlier than the time when the load actually increases. We need a proactive solution instead of a reactive strategy. This outcome would be possible only if the future load can be predicted. According to the predictive value, we can prepare for retrieving the upcoming idle resources, providing them to other users or configuring them into a power-saving mode beforehand; or we can add resources for the upcoming peak load in advance to ensure a stable QoS.

However, load forecasting is difficult in cloud computing environment for the following reasons. First, most modern applications have fluctuant loads which lead to complex behaviors in resources usages as their intensity and composition change over time. Load forecasting for such applications is not easy. Second, for security and privacy, cloud service providers are usually forbidden to access to the internal details of the applications. So, cloud management system cannot take advantage of the application's internal characteristics to forecast load. Third, unlike traditional computing environment, in cloud, the external environment which the applications have to face is dynamic. Interference among applications hosted on the same physical machine leads to complex resources usage behaviors as the applications compete for various types of resources which are hard to strictly partition. For instance, an application with constant load should have constant resources demand in exclusive non-virtualized environment. But in cloud where co-hosted applications compete for the shared last level cache or I/O bandwidth, the usage of resources that can be strictly partitioned and allocated (e.g. CPU or memory) will likely fluctuate.

B. Support Vector Machine

Modern cloud computing data centers comprise of heterogeneous and distributed components, making them difficult to manage piece-wise, let alone as a whole. Furthermore, the scale, complexity and growth rate of these systems render any heuristic and rule-based system management approaches insufficient. It is also infeasible to forecast load by modeling the behaviors of the various applications and their relationships to each other. In response to these challenges, statistics-based techniques for building gray or black box models of loads can better guide resources provisioning decision in cloud. Our study treats the load forecasting as a time series prediction problem, and makes use of statistical learning method – Support Vector Machine (SVM).

SVM was developed by Vapnik and used for many machine learning tasks such as pattern recognition, object classification, regression analysis and time series prediction [6]. It is based on the structural risk minimization principle which tries to control the model complexity as well as the upper bound of generalization risk. On the contrary, traditional regression techniques, including traditional Artificial Neural Networks (ANN) [31], are based on empirical risk minimization principle, which tries to minimize the training error only. Furthermore, its learning process is quite complex and inefficient for modeling, and the choices of model structures and parameters are lack of strict theory guide. So, it may suffer from over-fitting or under-fitting with ill-chosen parameters. In contrast, SVM has strict theory and mathematical foundation, doesn't have the problem of local optimization and dimensional disaster. It can achieve higher generalization performance especially for small samples set. It has a limited number of parameters to choose for modeling, and there exist fast and memory-efficient algorithms.

SVR is the methodology by which a function is estimated using observed data which in turn train the SVM. Its goal is to construct a hyper-plane that lays close to as many of the data points as possible. This goal is achieved by arriving at the most flat function which ensures that the error does not exceed a threshold ε. Flatness is defined in terms of minimum norm whereas the error threshold is introduced as a constraint. Slack variables are introduced to deal with the situations where the above definition followed in the strict sense leads to an infeasible solution [32].

When performing time series prediction by SVM, each input vector x_i is defined as a finite set of consecutive measurements of the series. The output vector y_i contains the $x_{(n+1)}$ observation data, where n determines the amount of history data. Each combination (x_i, y_i) constitutes a training point. There are N such training points used for fitting the SVR. SVM is a linear learning machine. The linear function is formulated in the high dimensional feature space, with the form of (1).

$$f(x)=w\phi(x)+b \tag{1}$$

Where x is nonlinearly mapped from the input space to a higher dimension feature space, via mapping function ϕ. Kernel function is used to simplify the mapping.

The goal is to find "optimal" weights w and threshold b as well as to define the criteria for finding an "optimal" set of weights. The First is the "flatness" of the weights, which can be measured by the Euclidean norm (i.e. minimize$\|w\|^2$). The second is the error R_{emp} generated by the estimation process of the value, also known as the empirical risk. The overall goal is to minimize the regularized risk R_{reg} as (2) and (3):

$$\begin{aligned} \textit{Minimize} \quad R_{reg} &= \frac{1}{2}\|w\|^2 + R_{emp} \\ &= \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} L^{\varepsilon}\left(y_i, f(x_i)\right) \end{aligned} \tag{2}$$

Where

$$L^{\varepsilon}\left(y_i, f(x_i)\right) = \begin{cases} \left|y_i - f(x_i)\right| - \varepsilon & \left|y_i - f(x_i)\right| \geq \varepsilon \\ 0 & \textit{otherwise} \end{cases} \tag{3}$$

R_{emp} in (2) is measured by the ε-insensitive loss function L^{ε}. C is the regularization constant determining the trade-off between the empirical risk and regularized risk. Introduction of the positive slack variables, ζ and ζ^*, which respectively denote the errors above and below ε, leads (2) to the following constrained function:

$$\begin{aligned} \textit{Minimize} \quad & R_{reg} = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\left(\zeta_i + \zeta_i^*\right) \\ \textit{Subject to} \quad & f(x_i) - w\phi(x_i) - b_i \leq \varepsilon + \zeta_i \\ & w\phi(x_i) + b_i - f(x_i) \leq \varepsilon + \zeta_i^* \\ & \zeta_i^{(*)} \geq 0 \end{aligned} \tag{4}$$

Training SVM is equivalent to solving a linearly constrained quadratic programming problem.

III. Approach

The main body of our multi-step-ahead load forecasting method is based on our improved ***SVR***. Rather than giving the same consideration to the training data within a sliding window in standard SVR, our multi-step-ahead load forecasting strategy gives more ***w***eight to more "important" data.

In order to enhance prediction accuracy, the ***K***alman ***S***moother is used for data pre-processing. We argue that Kalman smoother is suitable for the cloud application's load estimation because it was originally developed to estimate time-varying states in dynamic systems. This approach essentially uses a filtering technique to eliminate the noise of resources usage signal coming from error of measurement technique while still discovering its real main fluctuations.

We give this method a name, **KSwSVR**.

A. Kalman Smoother

The Kalman filter [7] has been widely used in the area of autonomous or assisted navigation. One of the main advantages of the filter is that it can estimate hidden parameters indirectly from measured data and can integrate data from as many measurements as are available, in an approximately optimal way. It estimates a process by using a form of feedback control: the filter estimates the process state and then obtains feedback in the form of (noisy) measurements.

But the Kalman filter only considers $P(x_t|y_{0:t})$ as the filtered estimate of x_t only takes into account the "past" information relative to x_t. By incorporating the "future" observations relative to x_t, we can obtain a more refined state estimate. That is why we choose the Kalman smoother as our noise reduction method. The Kalman smoother, which can be calculated from the Kalman filter results by recursions, estimates $P(x_t|y_{0:T},\ t<T)$, taking into account both past and future information. It is also computationally attractive, due to its recursive computation, since the production of the next estimate only requires the updated measurements and the previous estimations.

B. SVR with Weighted Training Data

Treating all data of a time series equally is clearly unreasonable. We should take advantage of the data according to their "importance" — usefulness of data for improving prediction accuracy.

In the time series of non-stationary system, the dependency between input variables and output variables gradually changes over the time. Specifically, the recent past data could provide more information than the distant past data. This conclusion is also true in this paper's cloud scenario. One of the most powerful arguments is *locality of reference*, also known as *principle of locality* [8]. As one of the cornerstones of computer science, locality of reference was born from efforts to make virtual memory systems work well, which is a phenomenon of the same value or related storage locations being frequently accessed. But today, this principal has been widely used beyond the virtual memory and directly influenced the design of processor caches, disk controller caches, storage hierarchies, network interfaces, database systems, graphics display systems, human-computer interfaces, individual application programs, search engines, web browsers, edge caches for web-based environments, and computer forensics. Therefore, we have good reason to believe that cloud load would follow the law as well. Hence we take the same principle that gives more weight to the more important recent historical data of the load. The newer the data are, the more important they are.

Another factor that influences the importance of data is their "credibility". In our multi-step-ahead load forecasting, there are two types of data — measured data and predicted data. Measured data refer to the true historical resources usage information collected by the system monitor. They are believed to have a high credibility. Predicted data, the result of load forecasting algorithm, have lower credibility as any prediction algorithm has prediction error. A multi-step-ahead prediction can be achieved by running one-step-ahead prediction iteratively. Time series data related to once m-step-ahead prediction are $\{\ldots, x_{t-2}, x_{t-1}, \hat{x}_t, \hat{x}_{t+1}, \hat{x}_{t+2}, \ldots, \hat{x}_{t+m-1}\}$. The prediction of $\hat{x}_{t+m-1}$ is based on the series

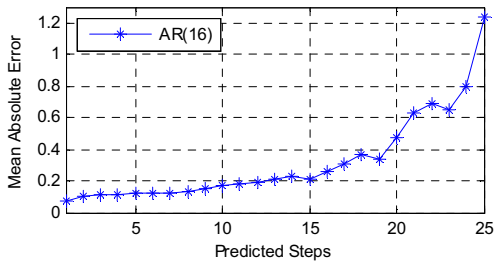

Figure 1 Relationship between MAE and predicted steps

$\{\ldots, x_{t-2}, x_{t-1}, \hat{x}_t, \hat{x}_{t+1}, \hat{x}_{t+2}, \ldots, \hat{x}_{t+m-2}\}$, where only $\{\ldots, x_{t-2}, x_{t-1}\}$ are the measured data and $\{\hat{x}_t, \hat{x}_{t+1}, \hat{x}_{t+2}, \ldots, \hat{x}_{t+m-2}\}$ are predicted data. This process is based on a significant hypothesis that predicted data are assumed to be measured data when performance next step prediction. However, due to prediction error and dynamic feature of cloud, this hypothesis cannot be satisfied all the time. Particularly, we need to address the accumulation of prediction errors. Every one-step-ahead prediction may cause an error. Therefore, using the former prediction results as the input data for next prediction will cause accumulation of errors. Fig. 1 depicts the relationship between prediction Mean Absolute Error (MAE) and predicted steps, where the load series we used is collected from a real world I/O trace of an On-Line Transaction Processing (OLTP) applications [13] and predictor is AR(16) [9]. It indicates that with the increase of predicted steps, the MAE is increasing drastically, i.e., the predicted data's credibility is decreasing.

To sum up the above arguments, in multi-step-ahead load forecasting of cloud, the importance of input data series gradually increases and then decreases. The inflection point is between last measured data and first predicted data.

From (2), it can be observed that the performance of SVR is sensitive to the regularization constant *C*. A small value for *C* will under-fit the training data because the weight placed on the training data is too small thus resulting in large value of prediction error. On the contrary, when *C* is too large, SVR will over-fit the training data, leading to deterioration of generalization performance.

By using a fixed value of *C* in the regularized risk function, standard SVR assigns equal weights to all the ε-insensitive errors between the measured and predicted data, treating two types of data equally. For illustration, the empirical risk function in standard SVR is expressed by (5).

$$R_{emp\_std}=C\sum_{i=1}^{n}\left(\zeta_i+\zeta_i^*\right) \tag{5}$$

As discussed above, this is unreasonable. Thus, it is beneficial to place different weight on the ε-insensitive errors according to the importance of training data. So, we add a weight coefficient w_i to the regularization constant, translating the empirical risk function to (6).

$$R_{emp\_w}=C\sum_{i=1}^{n}w_i\left(\zeta_i+\zeta_i^*\right)$$
$$w_i=\begin{cases}f_{measured}(i) & i<t\\ f_{predicted}(i) & i\geq t\end{cases} \tag{6}$$

Where $f_{measured}(i)$ is monotonically increasing function for measured data while $f_{predicted}(i)$ is monotonically decreasing function for predicted data. $f_*(i)$ may be a linear function, an exponential function, or others meeting the monotonicity requirement. Obviously, the choice of $f_*(i)$ will affect the subsequent prediction accuracy. The optimal solution is taking into account characteristics of the data. This is another point worth studying.

C. Resources Provisioning Based on KSwSVR

Directly taking the predicted value as resources provisioning value may lead to unacceptable SLA violation because any prediction algorithm has an error range. We update the predicted value for two considerations. First, load spike, which will lead to under-provisioning, is difficult to predict especially in dynamic cloud. Second, we need a flexible mechanism to deal with different SLA levels in this multi-tenant environment. The actual allocation value is computed as (7).

$$x_t^{alloc}=s\hat{x}_t+\frac{1}{k}\sum_{i=t-1}^{t-k}d_i,\quad d_i=max\left\{0,\ x_i^{use}-x_i^{alloc}\right\} \tag{7}$$

x_t^{use}, $\hat{x}_t$ and x_t^{alloc} separately represent real resources usage, predicted value and actual resources allocation value at time *t*. With d_i, we make use of the information of under-provisioning in the last *k* periods, while ignoring over-provisioning. This allows system to quickly respond to load spike. *s* is an incremental coefficient which is highly correlated with QoS of cloud. Bigger *s* means allocating more resources and fewer SLA violations.

IV. Evaluation

In this section, the performance of KSwSVR will be evaluated by using various types of real-world trace data and comparing with other typical load forecasting technology. We prefer using public trace data rather than historical data generated by ourselves for the purpose of giving comparable and reproducible results.

A. Prediction Algorithm

In order to highlight the prediction performance of KSwSVR, two widely used prediction methods are chosen for comparison. They are typical representatives of linear prediction algorithm and machine learning technology – AR and BPNN. Meanwhile, standard SVR is also our comparison object.

- Linear Prediction – AR

Dinda et al. [9] studied different linear load forecasting models including AR, moving average, autoregressive moving average, autoregressive integrated moving average and autoregressive fractionally integrated moving average models. Their conclusion is that the simple AR model is the best model and is appropriate and sufficient to be used for load prediction.

AR is a basic linear time series prediction algorithm in which the current value can be represented by the sum of a linear combination of previous values and an error ε. The general expression of AR(*p*) model can be denoted as (8).

$$x_t = \sum_{i=1}^{p} \varphi_i x_{t-i} + \varepsilon_t \quad (8)$$

We adopt Dinda's recommendation that AR(16) is best in consideration of both overhead and prediction accuracy.

- Machine Learning – BPNN

As SVR, ANN [31] is also a typical machine learning strategy in the category of regression computation. It can generalize the characteristics of load by proper training. A standard multilayer feed-forward neural network consists of an input layer, a hidden layer and an output layer. The BPNN is the most popular and the oldest supervised learning feed-forward neural network algorithm [31]. It learns by calculating the errors of the output layer to find the errors in the hidden layers. The algorithm is highly suitable for the solution to problems in which no relationship is found between the output and inputs. Due to its flexibility and learning capabilities, BPNN has been successfully used in wide range of applications. Therefore, we chose it as one of the comparison object and empirically configure it with six input neurons, one hidden layer with ten hidden neurons, as considering both prediction overhead and accuracy.

- SVR and KSwSVR

Parameter configuration of standard SVR and KSwSVR in this work is as below. We choose the RBF kernel as it is easier to compute and has fewer parameters to adjust.

SVM-Type:	ε-regression
Kernel Function:	Radial Basis Function (RBF)
Cost (C):	1, penalty parameter of the error term
Gamma($\gamma = \sigma^{-2}$):	0.0625, parameter of the RBF
Epsilon(ε):	0.1, ε-insensitive loss function
$f_*(i)$:	linear function, for KSwSVR in (6)

It is worth emphasizing that we do not have tuned parameters of BPNN, SVR and KSwSVR for specific prediction object, but set the same parameters for all trace data according to domain knowledge. Therefore, the experimental result of this paper is representative.

B. Experiment Setup

In order to highlight the adaptability of KSwSVR, we have collected public trace data of various types of resources, involving the network, CPU, memory and storage systems (see Fig. 2):

- Gcpu/Gmem: 7 hours of CPU and memory usage data in Google cluster (TraceVersion1) [11]. We randomly selected a long duration job with jobID 1485896354.
- CScpu: CPU load trace of a big memory compute server in the CMCL (Computers, Media, and Communication Laboratory) at Carnegie Mellon University [12].
- OLTPio: Storage system data request rate derived from I/O trace of an OLTP applications running at large financial institutions [13].
- SEio: Storage system data request rate derived from I/O trace of a search engine [14].
- WC98: Client request rate observed in World Cup 98 web servers, from 1998-06-22:00.00 to 1998-06-22:23.59 [15].

First, we have evaluated the forecast performance of KSwSVR. The first 2000 points of each trace is used as our experiment data and translated to mean value of five intervals, except Google cluster data for its limited amount. As previously mentioned, our prediction work is for timely and dynamic fine-grained online scalability of cloud, so that the overhead of each prediction would be as small as possible. In order to ensure forecasting speed, six training data were used each time in BPNN, standard SVR and KSwSVR in our experiment.

Then, by simulating dynamic CPU allocation, we have shown the high efficiency of KSwSVR in resources provisioning.

C. Experimental Results

- Prediction Accuracy Evaluation

Before the training process begins, data normalization is performed by linear transformation as (9).

$$x_i^n = \frac{x_i - x_{min}}{x_{max} - x_{min}} \quad (9)$$

Where x_i^n and x_i represent the normalized and original measured data respectively, and x_{min} and x_{max} represent the minimum and maximum value of the original measured data. The evaluation of prediction performance is based on the MAE which is a widely used error metric for evaluating results of time-series forecasting, as showed in (10).

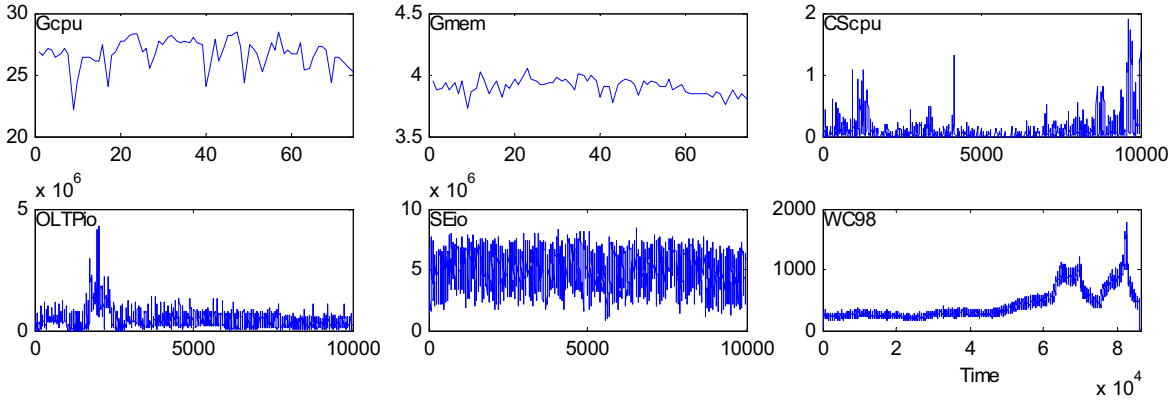

Figure 2 Trace data used in this work

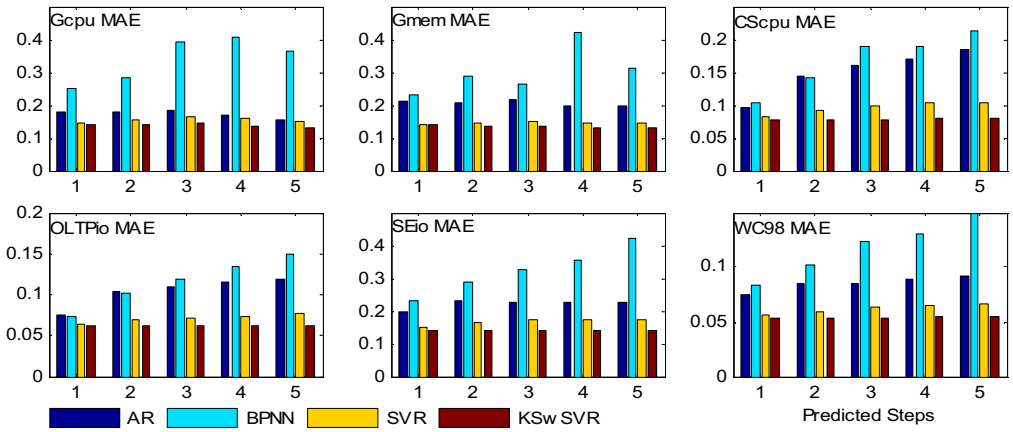

Figure 3 Load forecasting MAE comparison

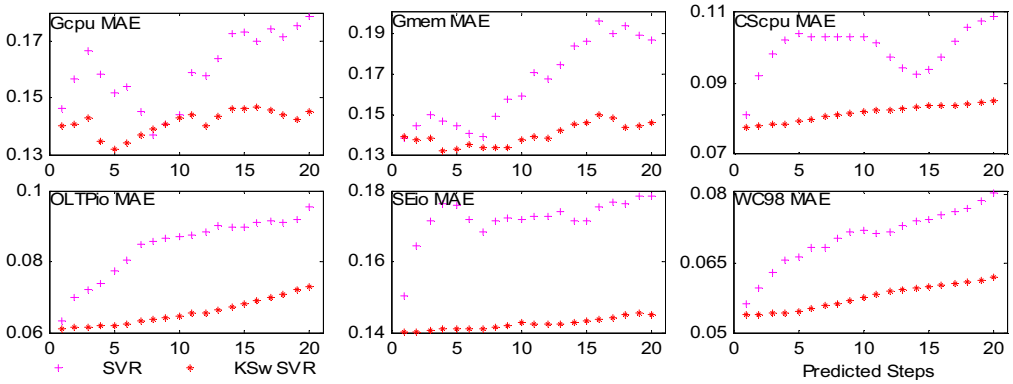

Figure 4 MAE comparison between KSwSVR and standard SVR

$$MAE=\frac{1}{n}\sum_{i=1}^{n}\left|\hat{x}_i-x_i\right| \tag{10}$$

Where $\hat{x}_i$ and x_i represent the predicted and original data respectively, and n is the number of predicted data points.

The detailed experimental results are shown in Fig. 3. Because the total number of samples is limited, performance trend feature of Google's two traces fluctuate slightly. But we can still reach this conclusion: KSwSVR has the best prediction accuracy, followed successively by standard SVR and AR, and BPNN.

AR, as a typical representative of the linear prediction technology that cannot be well adapted to the nonlinear regression problem, has a relatively high prediction error. Furthermore, when data have high volatility (e.g. CScpu and OLTPio), the error cumulative effect of AR is quite obvious. This makes it unsuitable for multi-step-ahead long-term load forecasting in cloud.

As an excellent machine learning technology, BPNN should have a better nonlinear regression performance. However, as mentioned before, timely and dynamic fine-grained online scalability of cloud needs a fast and efficient load forecasting algorithm. For this reason, the training data set for a predicted data should not be too large. Suffering from this restriction, the performance of BPNN deteriorates significantly, and its prediction accuracy is even lower than that of AR. Its error cumulative effect is also large.

In contrast, the theory foundation of SVM determines its excellent performance in the face of small training samples set. For all six traces, SVR and KSwSVR have always maintained a relatively high prediction accuracy and stability and their error cumulative effect is quite slight.

We further compared KSwSVR and standard SVR with more predicted steps, see Fig. 4. Benefiting from Kalman smoother and weight technology, KSwSVR outperforms standard SVR all the time, except two points of Gcpu for its small samples set. Furthermore, the prediction error of KSwSVR increases quite more slowly as predicted steps increases. That means its prediction accuracy is very stable. In contrast, the performance of standard SVR fluctuates with predicted steps. Specifically, improved MAE of KSwSVR is in Table 1.

- Dynamic CPU Allocation Based on KSwSVR

We demonstrated the advantage of accurate load forecasting for resources provisioning with a real CPU load trace, CScpu [12]. To be clearer, we only use its first half in this experiment. Static CPU allocation with fixed allocation value was compared with our dynamic CPU allocation based on KSwSVR.

We assume that SLA only requires no CPU under-provisioning. Any SLA violation rate can be achieved by adjusting incremental coefficient s in (7). The detailed experimental results are shown in Fig. 5. Under the same

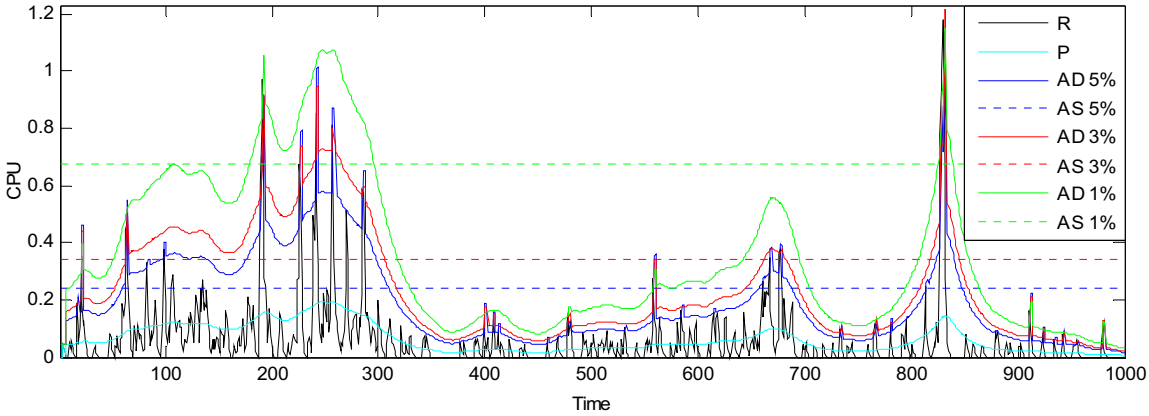

Figure 5 CPU allocation of Dynamic/ Static strategy

R - Real usage; P - Predicted value; AD/AS - Actual Dynamic/ Static allocation value; *x*%: SLA violation rate.

SLA violation rate, the total CPU consumption of dynamic strategy is always less than static method. For example, when the SLA violation rate is 5%, dynamic CPU allocation strategy based on the KSwSVR (blue solid line) can reduce the CPU consumption by 17.20%, compared with the static CPU allocation strategy (blue dashed line). Detailed data is shown in Table 2. With the decrease of SLA violation rate, the benefit produced by KSwSVR becomes more significant. Specifically, the total allocated CPU amount even reduces by nearly half (48.12%) when SLA violation rate is 1%.

By cooperating with other technologies such as virtualization and DVFS, our method can effectively improve resources utilization and energy-saving.

V. Related Work

We classify the related work into two categories: forecasting technology in grid, cloud and virtualization environment, and SVM/SVR related to our work.

The Network Weather Service (NWS) [10] is the most famous system designed to provide dynamic Grid resources performance forecasts. The predictive methods currently used include running average, sliding window average, last measurement, adaptive window average, median filter, adaptive window median, α -trimmed mean, stochastic gradient, and AR. Xu et al. [16] use fuzzy logic to model and predict the load of virtualized web applications. The VCONF project has studied using reinforcement learning combined with ANN to automatically tune the CPU and memory configurations of a VM [17]. Roy et al. [18] used a second order autoregressive moving average to forecast load in cloud. Jiang et al. [19] proposed a self-adaptive prediction solution to enable the instant cloud resources provisioning. They employed moving average, AR, ANN, SVM and Gene Expression Programming as the base predictors, and proposed a prediction ensemble method to combine the power of individual prediction techniques. It is not suitable for fine-grained online forecasting for its large amount of calculation. In [20], PRESS first employs FFT to identify repeating patterns. If repeating pattern is not discovered, PRESS employed discrete-time Markov chain to capture short term patterns of load. Based on similar characteristics of web-traffic, Caron et al. [21] proposed a pattern matching algorithm to identify closest resembling patterns similar to the present resources utilization pattern in a set of past usage traces of the cloud client. The resulting closest patterns were then interpolated by using a weighted interpolation to forecast approximate future values that were going to follow the present pattern. However, searching for similar patterns each time over the entire set of historical data is inefficient. Islam et al. [22] explored error correction neural network and linear regression to predict resources requirement in the cloud. Bey et al. [23] presented a modeling approach to estimating the future value of CPU load. It combined an adaptive network-based fuzzy inference systems and a clustering process. Liang et al. [24] presented a long term prediction model applying Fourier transform to exploit the periods of the CPU waves and using tendency-based methods to predict the variation. Wu et al. [25] adopted an adaptive hybrid model for long term load prediction in computational grid. It is an improvement of their previous work [26]. Both are based on AR and confidence interval estimations.

SVM/SVR is a relatively new forecasting technology. Its application related to our work is comparatively rare at present. Prem et al. [27] directly used SVM to forecast resources derived from NWS. SVR was used to build models predicting response time given a specified load for individual workloads co-hosted on shared storage system in [29]. SVR has also been used to model power consumption as a function of hardware and software performance counters in [29]. Kundu et al. [30] proposed to use SVR and ANN to model the relationship between the resources provisioning to a virtualized application and its performance. But, their work

Table 1 Improvement of prediction error

Trace Data	Gcpu	Gmem	CScpu	OLTPio	SEio	WC98
MAE Improvement	12.9%	17.9%	22.0%	28.1%	20.8%	22.5%

Table 2 Comparison of total CPU consumption

Incremental Coefficient (*s*)	3.0	3.8	5.6
SLA Violation Rate	5%	3%	1%
(Static-Dynamic)/Static	17.20%	29.81%	48.12%

was based on one assumption that application has static workloads and its behavior is stable. Moreover, their offline performance modeling cannot timely respond to changes of environment and load. Niehörster et al. [5] used SVM to enable the SaaS provider to predict the resources requirements of an application. Different from our method, theirs is coarse-grained and off-line. In addition, feature extraction of their work needs to know the internal detail of applications, but this is generally prohibited in cloud.

VI. CONCLUSION

In order to achieve efficient resources provisioning in cloud, we propose a multi-step-ahead load prediction method, KSwSVR, based on statistical learning technology. It integrates an improved SVR algorithm and Kalman smoother, and does not require access to the internal details of application. The improved SVR gives more weight to more important data than standard SVR, using the historical information more reasonably. Kalman Smoother is employed to eliminate the noise of resources usage data coming from measurement error. Public trace data of various resources were used to verify the excellent prediction accuracy, stability and adaptability. This statistical learning-based approach is not designed for a specific forecast object, so we believe it will exhibit outstanding performance when faced with various objects (application, VM, host and cloud system) and resources (computing, storage and network). CPU allocation experiment has shown that accurate load forecasting of KSwSVR could significantly reduce resources consumption while ensuring QoS. It is beneficial to improve resources utilization.

REFERENCE

[1] L. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[2] X. Fan, W. Weber, and L. Barroso, "Power provisioning for a warehousesized computer," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 13–23, 2007.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5. ACM, 2003, pp. 164–177.

[4] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 273–286.

[5] O.Niehorster, A. Krieger, J. Simon, and A. Brinkmann, "Autonomic resource management with support vector machines," in *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*. IEEE Computer Society, 2011, pp. 157–164

[6] V. Vapnik, "An overview of statistical learning theory," *Neural Networks, IEEE Transactions on*, vol. 10, no. 5, pp. 988–999, 1999.

[7] R. Kalman *et al.*, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[8] P. Denning, "The locality principle," *Communications of the ACM*, vol. 48, no. 7, pp. 19–24, 2005.

[9] P. Dinda and D. O'Hallaron, "Host load prediction using linear models," *Cluster Computing*, vol. 3, no. 4, pp. 265–280, 2000.

[10] M. Swany and R. Wolski, "Multivariate resource performance forecasting in the network weather service," in *Supercomputing, ACM/IEEE 2002 Conference*. IEEE, 2002, pp. 11–11.

[11] (2012) The Google cluster trace data. [Online]. Available: http://code.google.com/p/googleclusterdata/

[12] (2012) CPU load trace from CMCL. [Online]. Available: http://people.cs.uchicago.edu/~lyang/Load/abyss10000.dat

[13] (2012) OLTP applications. [Online]. Available: http://skuld.cs.umass.edu/traces/storage/Financial1.spc.bz2

[14] (2012) Search engine. [Online]. Available: http://skuld.cs.umass.edu/traces/storage/WebSearch2.spc.bz2

[15] (2012) World Cup 98 web servers. [Online]. Available: http://ita.ee.lbl.gov/html/contrib/WorldCup.html

[16] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, "Autonomic resource management in virtualized data centers using fuzzy logic-based approaches," *Cluster Computing*, vol. 11, pp. 213–227, 2008.

[17] J. Rao, X. Bu, C. Xu, L. Wang, and G. Yin, "Vconf: a reinforcement learning approach to virtual machines auto-configuration," in *Proceedings of the 6th international conference on Autonomic computing*. ACM, 2009, pp. 137–146.

[18] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in Cloud Computing (CLOUD), 2011 IEEE International Conference on. IEEE, 2011, pp. 500–507.

[19] Y. Jiang, C. Perng, T. Li, and R. Chang, "Asap: A self-adaptive prediction system for instant cloud resource demand provisioning," in Data Mining (ICDM), 2011 IEEE 11th International Conference on. IEEE, 2011, pp. 1104–1109.

[20] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in Network and Service Management (CNSM), 2010 International Conference on. IEEE, 2010, pp. 9–16.

[21] Caron, Eddy, Frédéric Desprez, and Adrian Muresan. "Forecasting for grid and cloud computing on-demand resources based on pattern matching." Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. IEEE, 2010.

[22] 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, no. 1, pp. 155–162, 2012.

[23] K. Bey, F. Benhammadi, A. Mokhtari, and Z. Guessoum, "Cpu load prediction model for distributed computing," in Parallel and Distributed Computing, 2009. ISPDC'09. Eighth International Symposium on. IEEE, 2009, pp. 39–45.

[24] J. Liang, J. Cao, J. Wang, and Y. Xu, "Long-term cpu load prediction," in Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on. IEEE, 2011, pp. 23–26.

[25] Y. Wu, K. Hwang, Y. Yuan, and W. Zheng, "Adaptive workload prediction of grid performance in confidence windows," Parallel and Distributed Systems, IEEE Transactions on, vol. 21, no. 7, pp. 925–938, 2010.

[26] Y. Wu, Y. Yuan, G. Yang, and W. Zheng, "Load prediction using hybrid model for computational grid," in Grid Computing, 2007 8th IEEE/ACM International Conference on. IEEE, 2007, pp. 235–242.

[27] H. Prem and N. Raghavan, "A support vector machine based approach for forecasting of network weather services," Journal of Grid Computing, vol. 4, no. 1, pp. 89–114, 2006.

[28] S. Uttamchandani, L. Yin, G. Alvarez, J. Palmer, and G. Agha, "Chameleon: a self-evolving, fully-adaptive resource arbitrator for storage systems," in USENIX Annual Technical Conference, 2005, pp. 75–88.

[29] J. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. Snoeren, and R. Gupta, "Evaluating the effectiveness of model-based power characterization," in USENIX Annual Technical Conf, 2011.

[30] S. Kundu, R. Rangaswami, A. Gulati, M. Zhao, and K. Dutta, "Modeling virtualized applications using machine learning techniques," ACM SIGPLAN Notices, vol. 47, no. 7, pp. 3–14, 2012.

[31] S. Theodoridis and K. Koutroumbas, "Pattern recognition, fourth edition," Academic, San Diego, 2008.

[32] Tao, Tingye, Fei Gao, and Zhaofu Wu. "Gross error detection and correction based on wavelet transform and support vector machine." International Symposium on Spatial Analysis, Spatial-temporal Data Modeling, and Data Mining. International Society for Optics and Photonics, 2009.