2012 IEEE Ninth International Conference on Services Computing

Self-adaptive Cloud Capacity Planning

Yexi Jiang*, Chang-shing Perng†, Tao Li*, Rong Chang†
*School of Computing and Information Sciences
Florida International University, Miami, FL, 33199
Email: {yjian004, taoli}@cs.fiu.edu
†IBM T.J Watson Research Center, Hawthorne, NY, 10532
Email: {perng, rong}@us.ibm.com

***Abstract*—The popularity of cloud service spurs the increasing demands of cloud resources to the cloud service providers. Along with the new business opportunities, the pay-as-you-go model drastically changes the usage pattern and brings technology challenges to effective capacity planning.**

In this paper, we propose a new method for cloud capacity planning with the goal of fully utilizing the physical resources, as we believe this is one of the emerging problems for cloud providers. To solve this problem, we present an integrated system with intelligent cloud capacity prediction. Considering the unique characteristics of the cloud service that virtual machines are provisioned and de-provisioned frequently to meet the business needs, we propose an asymmetric and heterogeneous measure for modeling the over-estimation, and under-estimation of the capacity. To accurately forecast the capacity, we first divide the change of cloud capacity demand into provisioning and de-provisioning components, and then estimate the individual components respectively. The future provisioning demand is predicted by an ensemble time-series prediction method, while the future de-provisioning is inferred based on the life span distribution and the number of active virtual machines.

Our proposed solution is simple and computational efficient, which make it practical for development and deployment. Our solution also has the advantages for generating interpretable predictions. The experimental results on the IBM Smart Cloud Enterprise trace data demonstrate the effectiveness, accuracy and efficiency of our solution.

***Index Terms*—cloud service; capacity management; service quality maintenance.**

I. Introduction

The emergence of cloud service platforms like Amazon EC2 [4] and IBM Smart Cloud Enterprise [5] bring new challenges to IT infrastructure capacity management. As cloud computing promises pay-as-you-go flexible charging, resource demand becomes more volatile than that in traditional IT environments. Capacity management in cloud has critical impact on the service quality and the profitability of the clouds.

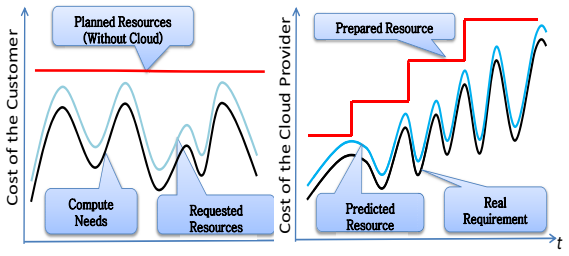
On one hand, underestimation of the cloud capacity can cause resource shortage and revenue loss. For the cloud providers, hardware resources still require long acquisition and deployment process. If the actual demand is higher than the existing capacity, the cloud has to postpone serving new customers and lose the potential revenue. If the shortage is severe, even provisioning requests from existing customers have to be rejected, which defeats the promise that application in cloud can scaling-up whenever workload increases.

On the other hand, overestimation of the cloud capacity may result in idled resources and unnecessary utility costs. Unused hardware not only causes under-utilized capital, but also results in more early purchase costs as the price of the same computing equipment is always going down. It is always better to purchase equipment later than earlier. Another associated cost with idled equipments is the cost of network, labor, facility (floor space, cooling systems, power systems, etc.), and utility (electricity and water) cost. These associated costs are usually proportional to the amount of hardware equipments [8].

Usually, the costs of labor, hardware, facilities, and network are relatively stable for a short term. These components are not suitable for dynamic adjustment at a fine granularity since the adjustment overhead is even more than the amount of cost savings. But the power supply can be adjusted readily. It is necessary to control the power consumption to reduce the cost of the cloud as well as reduce the carbon dioxide emission. The US Environment Protection Agency (EPA) estimates that the energy usage at cloud data centers is successively doubling every five years. In the year of 2011, the annual electricity cost at these data centers would be approximately $7.4 billion [15].

In this paper, we aim to find a satisfiable balance between reducing the power cost and maintaining a satisfactory service level by predicting and preparing the active server capacity. The motivating question for this study is that, since the Infrastructure-as-a-Service paradigm of cloud computing enables the IT infrastructure cost for customers to be elastic (as shown in Figure 1(a)), why not also makes the cloud cost for the providers to be elastic too? Figure 1(b) shows a providers' view of cloud capacity. Hardware capacity, indicated as the top line, is usually increased in a stepwise fashion as hardware components are purchased in batch to meet the growing demand. There is usually a large gap between the full capacity and the actual demand, as indicated by the margin between the top line and the lowest curves. It would be ideal if the utility and labor costs are proportional to the actual demand instead of the hardware capacity. We intend to predict and prepare resources as close to the actual demand as possible.

978-0-7695-4753-4/12 $26.00 © 2012 IEEE
DOI 10.1109/SCC.2012.8



(a) Cloud enable the IT cost of customers to be elastic (b) Enable the cloud self-elastic through capacity estimation

Fig. 1. Cloud Power Saving via Capacity Estimation

A. Challenges

The task of preparing an adequate cloud capacity for the demand is not trivial since the behavior of cloud customers tends to be more volatile. One of the unique features of the pay-as-you-go cloud service is that customers can ask for cloud resources whenever they need, and release the resources whenever they do not need them. The customers have no obligation to tell the service provider the amount of resources they need or the duration they would use the resources. Since the demand of resources is associated with the actual business activities, customers themselves are unable to figure out an accurate schedule. Similar to the scenario in retail industry, the behaviors of individual customers are unpredictable. Finding a way to aggregate information and use the information to improve the cloud service is a challenge for cloud provider.

How to handle workload burst is another challenge. There is no known way to effectively coordinate customers' behaviors, since they are independent to each other. As a consequence, the burst is unavoidable. In order to maintain the promised service quality such as fast provisioning and stable virtual machine performance, the number of prepared active servers should always be consistent with the demand, even when facing the possibility of workload burst.

We argue that one possible solution for effective capacity management is to enable the cloud environment to be burst-aware and self-scaling, which allow the system to prepare more active servers in advance for the possible incoming demand burst and to reduce its active capacity for the upcoming demand valley.

B. Our Contribution

In this paper, we focus on the problem of cloud capacity estimation and management. We handle capacity estimation using a two-stage approach: the estimating stage and the controlling stage. Figure 2 illustrates the framework of the capacity estimation system. The core of the system is the *Capacity Predictor*, which predicts the future cloud capacity based on the information of trace data in the database, and notifies the *Power Management Module* to prepare the predicted resources (include the active servers, active coolers, other auxiliaries, and scheduled labor) in advance. In real time, the *Capacity Controller* dynamically tunes the capacity according to the actual workload of the cloud environment.

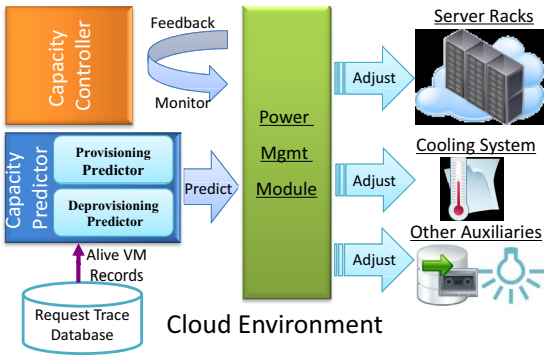


Fig. 2. System Framework

The scope of this paper is confined to cloud capacity prediction, real-time cloud capacity adjustment is one of our future works. In short, our contribution of this paper can be summarized as follows:

- We introduce a novel measure for prediction accuracy specifically for clouds. By incorporating this measure, the prediction can be dynamically tuned according to actual running status of the cloud environment.
- We decompose the cloud capacity into provisioning and de-provisioning components, and then estimate them separately for better accuracy.
- We conduct a series of experiments to test our methods on the trace data of IBM Smart Cloud Enterprise (SCE)[5], a real world cloud environment. The evaluation results demonstrate that our methods are effective and burst-tolerate.

To the best of our knowledge, this paper is the first work dedicated to the intelligent management of cloud capacity from the perspective of cloud capacity prediction and estimation.

C. Organization

The rest of this paper is organized as follows. We formulate the problem of capacity estimation in Section II. Section III introduces the detailed techniques for provisioning and de-provisioning demand estimation. We report the experimental evaluation of our method in Section IV. Finally, we discuss the related work and conclude our paper in Section V and Section VI, respectively.

II. Problem Statement

The goal of cloud capacity management is to ensure that the capacity of prepared resources in cloud satisfies the real demand in the near future. Without loss of generality, we model the cloud capacity as the number of VM units it can support. The VM unit is defined as the basic unit of cloud resources. Any VM a customer can apply must be the multiple of the VM unit. For example, IBM SCE defines one 64-bit VM unit as one 1.25 GHz Virtual CPUs with 2G main memory and 60G of storage space. The customer

can choose different sizes of VMs such as *Small*, *Medium*, *Large*, and *Jumbo* (16-core, 16G main memory and 2T+ storage space). We use VM(s) to denote the VM unit(s) for the remaining of this paper.

Let C_t be the future cloud capacity, $\hat{C}_t$ be the corresponding estimated capacity at time t, respectively. The problem of *estimating the future cloud capacity* is to find an estimation method, such that:

$$E_t = f(C_t, \hat{C}_t) \tag{1}$$

is minimized. In Equation (1), $f(\cdot, \cdot)$ denotes the cost function that is used to measure the error. The best capacity predictor should be the one that minimizes the sum of errors over multiple timestamps, i.e. $argmin \sum_t E$.

III. Modeling and Prediction Approach

In this section, we first explain how we pick the suitable data for capacity prediction. Then we propose a novel measure for estimation evaluation. Finally, we present the techniques for provisioning estimation and de-provisioning estimation, respectively.

A. Time Series Selection

Given the cloud trace data, there are three kinds of time series available for capacity prediction: The original capacity time series, the capacity change time series, and the separated provisioning/de-provisioning time series.

The original capacity time series is shown in Figure 3. It can be easily observed that the capacity of the cloud gradually increases from the long-term perspective. Directly modeling such a non-stationary time series would bring extra difficulties for prediction.

If we take the first derivative of the original capacity time series, the capacity change time series is obtained (Figure 4). The trend of this time series shows it is more stable than the original capacity time series. It is true that we can directly use this time series for modeling, but irregularity of the trend implies that it is not easy to discover the temporal patterns.

If we further decompose this time series into provisioning part and de-provisioning part (Figure 5), we can easily discover a weekly periodic pattern by taking a quick glance. This indirectly shows its ease of modeling.

Incorporating the estimated provisioning and de-provisioning, the capacity change can be estimated according to: $\Delta_t = prov_t - deprov_t$, where Δ denotes the change of capacity each time unit, $prov_t$ / $deprov_t$ denote the number of provisioned VMs and number of de-provisioned VMs each time unit.

B. Estimation Quality Criteria

Over-estimation and under-estimation of demands have different consequences when estimating the provisioning and de-provisioning demand. Table I illustrates the cost matrix in different situations. For provisioning estimation, an over-estimation has no negative effect on the customer and only causes idled cloud resources, but an under-estimation degrades the service quality. For de-provisioning estimation, an over-estimation degrades the service quality, while an under-estimation causes idled resources and loss revenue. Traditional regression measures such as mean absolute error (MAE), mean square error (MSE), and mean absolute percentage error (MAPE) only focus on the absolute accuracy of the estimated results and ignore the differences between over-estimation and under-estimation.

	Provision	De-provision
Over-estimation	resource wastes	SLA penalty / revenue loss
Under-estimation	SLA penalty / revenue loss	resources waste

TABLE I
COST MATRIX FOR DEMAND ESTIMATION

Considering the characteristics of cloud demand prediction, we propose a novel error measure called Demand Estimation Error (DEE). DEE is an asymmetric and heterogeneous measure that models the over-estimation and under-estimation of the demand as different kinds of costs: the cost of idled resources and the penalty of SLA.

Suppose the real demand at time t is $v(t)$ and the estimated demand at time t is $\hat{v}(t)$. We use a cost function $P(v(t), \hat{v}(t))$ to quantify the penalty of SLA. Similarly, we also define a cost function $R(v(t), \hat{v}(t))$ to quantify the cost of idled resources. Concrete quantification of the costs may vary for different cloud environments, and our method can work with any cost functions that satisfy the following two properties:

1) $P(v(t), \hat{v}(t)) \geq 0$ and $R(v(t), \hat{v}(t)) \geq 0$ for all $v(t)$ and $\hat{v}(t)$.
2) For two different timestamps t_1 and t_2, if $v(t_1) - \hat{v}(t_1) \geq v(t_2) - \hat{v}(t_2)$, then $P(v(t_1), \hat{v}(t_1)) \geq P(v(t_2), \hat{v}(t_2))$. Similarly, if $\hat{v}(t_1) - v(t_1) \geq \hat{v}(t_2) - v(t_2)$, then $R(v(t_1), \hat{v}(t_1)) \geq R(v(t_2), \hat{v}(t_2))$.

Cost of SLA penalty: A SLA penalty would occur when the service quality violates the pre-defined agreement. Both the under-estimation of provisioning and over-estimation of de-provisioning would cause this penalty. For example, when the workload of the cloud is full, the unexpected VM requests should wait for new available physical resources. The extra waiting time would significantly increase the fulfillment time of VM provisioning, causing the violation of SLA.

For simplicity, we use the delay of request fulfillment time to quantify the SLA penalty. When available resources are enough, the resources associated to the requested VM can be immediately allocated and the request can be fulfilled in a short time $T_{available}$. On the other hand, if there is not enough resources, the request has to wait for T_{wait} until new resources are available. In general, $T_{available} \ll T_{wait}$.

One simple form of $P(v(t), \hat{v}(t))$ for provisioning estimation can be modeled as Equation (2). The penalty for de-provisioning estimation can be obtained by exchanging the positions of $v(t)$ and $\hat{v}(t)$.

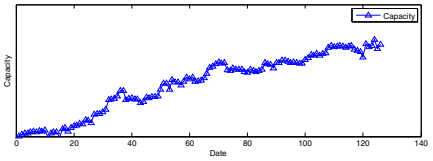


Fig. 3. The original capacity time series

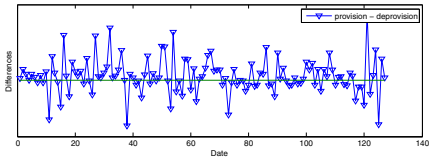


Fig. 4. The capacity change time series

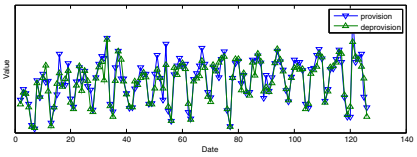


Fig. 5. The provision/de-provision time series

$$P(v(t), \hat{v}(t)) = min(v(t), \hat{v}(t))T_{available} + max(0, v(t) - \hat{v}(t))T_{wait}. \quad (2)$$

Based on the definition, the penalty is proportional to the seriousness of under-estimation. More sophisticated forms of the P function can also be used. For example, a common SLA typically specifies a penalty threshold for provisioning time. The time cost of a non-violated request in this case would have zero value for P function.

Cost of idled resources: This is the non-billable part of resources including the cost of electricity and the labor etc. We use R_{vm} to denote the cost of a single VM unit. The R function is defined as Equation (3).

$$R(v(t), \hat{v}(t)) = max(0, \hat{v}(t) - v(t))R_{vm}, \quad (3)$$

The de-provisioning version can be obtained by exchanging the positions of $\hat{v}(t)$ and $v(t)$.

Combining Equation (2) and Equation (3), the total cost of estimation is quantified as:

$$C = \beta P(v(t), \hat{v}(t)) + (1-\beta)R(v(t), \hat{v}(t)) = \begin{cases} \beta v(t)T_{available} + (1-\beta)(\hat{v}(t) - v(t))R_{vm} & \text{if } v(t) < \hat{v}(t), \\ \beta(\hat{v}(t)T_{available} + (v(t) - \hat{v}(t))T_{wait}) & \text{if } v(t) \geq \hat{v}(t). \end{cases} \quad (4)$$

For different clouds, the trade-off between SLA penalty and idled resources can be different. We use the parameter β to tune the preference between P function and R function. As mentioned in Section II, both provisioning and de-provisioning estimation aim to minimize the total cost quantified by Equation (4).

C. Prediction of Provisioning Demand

Predicting the provisioning demand is the first step for capacity estimation. Little research has been conducted on the cloud customer behavior as the cloud service has been appeared for less than 5 years. Similar to the retail supply scenario, individual customer may come at any time to purchase arbitrary amount of VMs, there is no trivial way to infer the demands of individuals. In the absence of deep understanding of customer behavior, one possible way of estimating the provisioning demand is to leverage the time series prediction/forecasting techniques to model and infer the global behavior of all the customers from the historical data. Figure 6 plots the demand time series of the total demand (the top time series) as well as the most frequent requested customers (the bottom three time series). This figure clearly shows that the time series of the total demand is more regular than those of individual customers. This irregularity of the bottom three time series is caused by the unknown behavior of the customers.

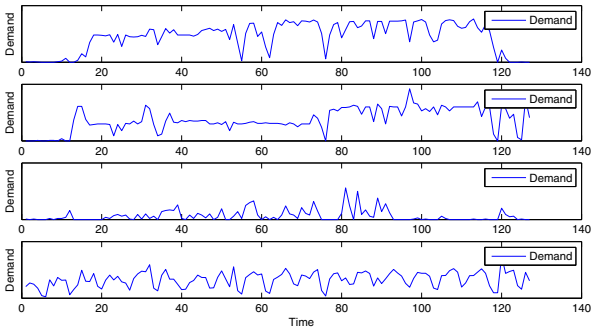


Fig. 6. The provisioning demand time series. The top three subfigures show the demand of three most frequent requested customers, the bottom one shows the time series of the total demand. Note that the scale of y-axis is not normalized for the sake of better visualization. For confidential issue, the values on y-axis are removed.

To better capture the temporal dynamics of the provisioning demand, we propose to use the ensemble method that combines the power of multiple predictors for prediction. Three reasons indicates why we utilize the ensemble method: (1) The robustness of ensemble method mitigates the risk of large deviation for prediction results. (2) The precision of ensemble method is on average better than the individual predictor. Without knowing the data characteristics, the ensemble method is a safe choice. (3) By changing the weights of individual predictors, we can dynamically tune the preference of prediction according to the actual running status of the cloud system. For example, we can assign large weights to the over-predictor when the workload of a cloud system is low, or assign small weights to the under-predictors while the workload is high.

The ensemble method proposed in our work is motivated from the online majority voting algorithm for classification [3]. However, different from classification, the provisioning time series contains the temporal information and the labels of prediction are continuous (not discrete classes). Therefore, it is not suitable to directly apply the classification ensemble for predicting provisioning demands.

Motivated by the classification scenario, we propose a weighted linear combination strategy for provisioning demand prediction. Suppose the predicted value for predictor $p \in \mathcal{P}$ at time t is $\hat{v}_p(t)$ and its corresponding weight at

time t is $w_p^{(t)}$, the predicted value $\hat{v}^{(t)}$ at time t is

$$\hat{v}^{(t)} = \sum_p w_p^{(t)} \hat{v}_p(t), \text{subject to } \sum_p w_p^{(t)} = 1. \quad (5)$$

Initially (t = 0), all the individual predictors have the same contributions to the prediction result, i.e. $w_p^{(0)} = \frac{1}{|\mathcal{P}|}$. Since the prediction result is continuous, the updating strategy should be carefully quantified. To update the weight, we first calculate the relative error $e_p^{(t-1)}$ of predictor p at time $t-1$ according to Equation (6).

$$e_p^{(t)} = \frac{\sum_p c_p^{(t-1)}}{c^{(t-1)}} w_p^{(t-1)}, \quad (6)$$

where $c_p^{(t-1)}$ is the prediction cost/error and can be given by any kind of cost functions, like MAE, MSE, MAPE and DEE (as defined in this section).

Note that the relative errors cannot be used as the new weights of the predictors since the sum of the errors is unbounded. As the final predicted value is the linear combination of all the results of individual predictors, we use Equation (7) defined below to make the constraint $\sum_p w_p^{(t)} = 1$ holds:

$$w_p^{(t)} = \frac{e^{(t)}}{\sum_p e_p^{(t)}}. \quad (7)$$

As for individual predictors, we employ five different time series techniques, their names and categories are listed in Table II. The new predictors can be added or old predictors can be removed freely due to the flexibility of the ensemble framework.

Method Name	Category
Moving Average	Naive
Auto Regression	Linear Regression
Artificial Neural Network	Non-linear Regression
Support Vector Machine	Linear Learner with Non-linear Kernel
Gene Expression Programming	Heuristic Algorithm

TABLE II
TIME SERIES PREDICTION TECHNIQUES USED FOR ENSEMBLE

D. Estimation of De-provisioning Demand

The future demand of de-provisioned VMs is bounded by the number of total active VMs in the cloud. At any time, any customer cannot de-provision more VMs than they requested. The ensemble technique is still workable for de-provisioning estimation, but we have an alternative way to estimate the de-provisioning demand more preciously.

Since we have all the information of the potential de-provisioned VMs, we can build profiles on the aspect of temporal dynamics for the VM image types. Rather than the observed time series, the temporal characteristics provides more meaningful and interpretable context information of the de-provisioning, and it can be used to facilitate the estimation of future demand.

Through exploration, we find that knowing the current life time of individual VM is helpful for estimating the de-provisioning. That is, we can infer when a certain VM would be de-provisioned through the life span distribution of the images. Since the true distribution of the image life span is not available, we need to estimate it from the historical data first.

Under the stationarity assumption [1], the life span distribution of the VMs does not depend on the time the VMs are provisioned. Suppose $life(VM)$ is the current life time of a VM, and n_i is the frequency of VMs with life span t_i. We estimate the empirical cumulative distribution function(CDF) of the life span with a step-wise function in Equation (8).

$$\begin{aligned} \hat{F}(x) &= P(life(VM) \le x) \\ &= \begin{cases} n_1 / \sum_i n_i & t_1 \le t < t_2, \\ (n_1 + n_2) / \sum_i n_{t_1} & t_2 \le t < t_3, \\ ..., & ... \\ (\sum_{i=1}^{n-1} n_i) / \sum_i (n_i) & t_n \le t. \end{cases} \end{aligned} \quad (8)$$

The output of the estimated CDF denotes the probability of a VM that would be de-provisioned when its current life time is t_i. Utilizing $\hat{F}(x)$, the de-provisioning demand can be estimated by

$$\sum_{i \in VM_{active}} \hat{F}(life(i) \le t_{now} - t_{start(i)}),$$

where t_{now} denotes the current time and $t_{start(i)}$ denotes the provisioned time of VM i.

IV. EXPERIMENTAL EVALUATION

To evaluate the effectiveness of our system, we use the real VM trace log of IBM Smart Cloud Enterprise (SCE) product to conduct the experiments. The trace data we obtained records the VM requests for more than 4 months, which contains tens of thousands of VM records with more than 100 different VM image types. In SCE's trace data, each VM request record contains 21 features such as *Customer ID*, *VM Type*, *Request Start Time*, and *Request End Time*, etc. For this study, we only focus on the features of *VM Type*, *Request Start Time* and *Request End Time*.

The experiments are trying to answer the following questions:

- Whether the estimation of provisioning based on ensemble is reliable?
- Whether the measure DEE is practical and flexible for cloud capacity prediction?
- Whether the estimation of de-provisioning demand based on the life span distribution outperforms the time series prediction method?
- To what extent can our method estimate the future capacity?

A. The Choice of Aggregation Granularity

Before the estimation, we need first pre-process the raw trace data. Two reasons indicate that the raw data is not suitable for estimation: (1) The raw trace data records the

requests in low-level representation. Such data format is not suitable to directly feed the estimation/prediction models. (2) The irrelevant features of the records would affect the precision of the estimation and bring unnecessary overhead for the learning procedure. The time series used in our work is generated by aggregating the request records via request timestamps. The aggregation can be conducted at different time granularities.

Figure 7 shows the time series of the provisioning demand aggregated by week, day and hour. This figure shows that coarser the granularity, larger the demand in each time slot. This fact implies that the weekly aggregated time series requires the system to prepare the most VMs for each time slot. Compared with a finer granularity, a smaller portion of prediction deviation for weekly aggregated time series would result in large cost. Moreover, the life span of most of the VMs are shorter than one week based on our exploration on the trace data. The weekly aggregated values cannot reflect the real capacity required.

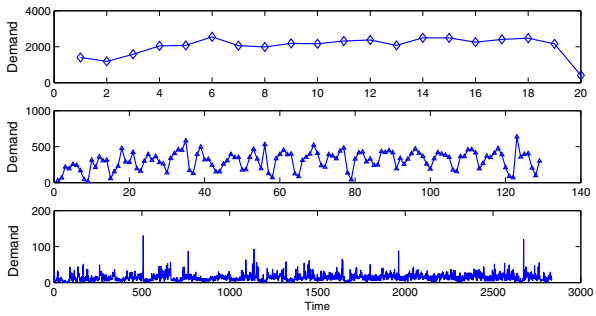


Fig. 7. Time series with different granularity. The top one is aggregated by week, the middle on is aggregated by day, the bottom one is aggregated by hour.

On the contrary, it is also not suitable to aggregate the records by hour, since the life span of the VMs cannot be so short. A fine granularity would make the value on each timestamp lack of statistical significance. Therefore, we aggregate the time series in day in our solution.

B. Provisioning Estimation Accuracy Evaluation

To evaluate provisioning estimation, we compare the accuracy of our ensemble predictor with the individual predictors mentioned in Table II. In this experiment, DEE is used as the measure. For parameter, we set $\beta = 0.5$, $T_{wait} = 1200$ (the time for on-demand preparation, include server boot up, VM generation, and configuration etc.), $T_{available} = 10$ (resource is available instantly), and $R_{vm} = 500$ (the cost of one idled VM).

1) Prediction Precision Comparison: We partition the time series horizontally into two parts: the records before May 2011 and the records after May 2011, and then we use the data before May for training. The precision of these predictors are evaluated on the date of May, June, and July, respectively. For each individual predictor, the grid search strategy is used to find the best parameters. To eliminate the randomness of some predictors (Random Guess, Artificial Neural Network and Gene Expression Programming), the results are computed by averaging 10 runs of each predictor. The evaluation results are shown in Table III. They clearly illustrate that the best predictor is different for different test data sets. But on average, the ensemble method achieves the best performance.

Predictor	May	June	July	Average
Random Guess	2281555	3507600	3080320	2956491
Moving Average	1295550	1293620	1293620	1294263
Auto Regression	504912	760110	1047275	770765
Artificial Neural Network	980780	1095102	1577127	1217669
Gene Expression Programming	866117	640405	1037705	848075
Support Vector Machine	3746005	2199010	1147240	2364085
Ensemble	538302	626585	1072840	**745909**

TABLE III
THE COST OF DIFFERENT PREDICTORS MEASURED BY DEE

Besides the comparison of prediction techniques, we also evaluate the effectiveness of weight updating using different error measures. We run the ensemble methods with measures MAE, MSE, MAPE and DEE. The results in Table IV show that our DEE causes the least error in the context of cloud capacity estimation. This is because the consequence of under- and over-estimation of cloud capacity is different. DEE provides a fine-grained configuration to differentiate the penalty for under-predictor and over-predictor to mitigate the potential risk of future under-estimation. However, MAE, MSE and MAPE only focus on the absolute difference between the real and predicted capacity and ignore the asymmetric issue.

Measure	MAE	MSE	MAPE	DEE
Cost	1270307	1299287	1300322	**745909**

TABLE IV
COST COMPARISON FOR DIFFERENT MEASURES

2) Effectiveness of Parameter Tuning: As mentioned before, the parameter β can be used to tune the preference of the predictor (optimistic vs. pessimistic). A higher β results in less SLA penalty risk but increases the idled resources, while a lower β can reduce the idled resources but increases SLA penalty. This experimental results shown in Figure 8 confirm that β is an effective parameter to set the trade-off between the SLA penalty and idled resources. When β equals to 0, the SLA penalty cost is totally ignored, therefore the idled resource decreases to the minimum. When β equals to 1, the cost of idled resources is huge but the SLA penalty decreases to the minimum. In practice, β can be tuned on-the-fly.

C. De-provisioning Estimation Accuracy Evaluation

To find out the best de-provisioning estimation method, we try four variations of the life span distribution methods to estimate the de-provisioning demands. For all the variations, the testing are conducted on the last 60 days of de-provisioning data. All the other data can be used as the training data. The detail of the four different variations are listed as follows:

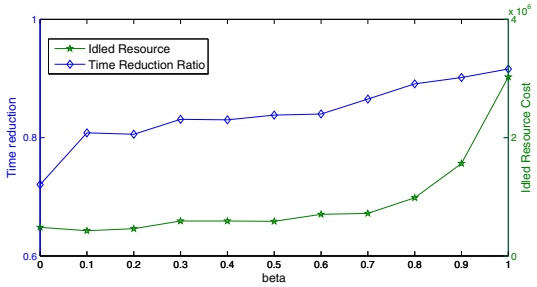


Fig. 8. The effect of tuning β

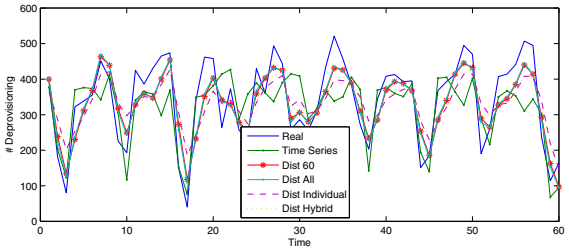


Fig. 9. Estimation result of de-provisioning

1) **Using the global life span distribution estimated by all the historical data.** We estimate the distribution as well as the CDF with all the training data. For each day, we estimate the expected number of VMs that would be de-provisioned based on their living time. We name this method as *Dist All*.
2) **Using the global life span distribution estimated by the latest 60 days of data before the testing data.** This variation is the same as the first one, but with only 60 days of training data. We name this method as *Dist 60*.
3) **Using the individual life span distributions estimated by all the historical data.** For this variation, we estimate the empirical CDF for each image type. This method is a finer granularity version of the first method. We name this method as *Dist Individual*.
4) **Combine the global and individual life span distribution estimated by all the historical data.** We combine the estimation of global distribution and individual distribution together, and weight them as $\alpha\hat{F}_i + (1-\alpha)\hat{F}_g$, where α equals to the fraction between the frequency of specified image type and the most popular image type. We name this method as *Dist Hybrid*.

Besides the four variations, we also leverage ensemble method to predict the future de-provisioning. As illustrated in Figure 9, all of these methods successfully discover the periodic patterns of the de-provisioning trend. But, the ensemble methods has limited ability to capture the scale of the peaks. Among these methods, *Dist All* has the highest accuracy since it can capture the scale of peaks the best. The accuracy of *Dist 60*, *Dist Individual*, and *Dist Hybrid* are close to that of *Dist All*. All of these distribution based methods outperform the time series prediction method. The reason is that they all estimate the de-provisioning from the empirical life span distributions, rather than simply from the observed time series.

Figure 10 shows the errors of these methods for prediction. Besides the methods mentioned above, we also calculate the errors of two naive methods. The first method does no pre-action for the capacity planing (we name it as *None*). It only prepares the resources when request is coming. The second method always prepares the maximum capacity for the cloud (we name it as *Maximum*). Based on the experiment results, any method that takes the pre-action is significantly better than *None*. The *Dist All* makes least error among all the methods.

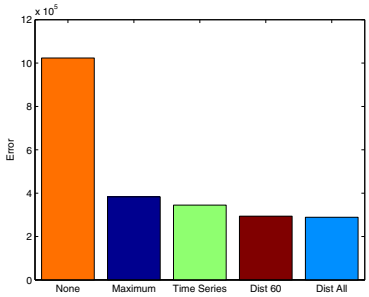


Fig. 10. Errors of different methods

V. Related Works

A. System Oriented Data Mining

The increasing complexity and scale of modern computing systems make the analytic difficulty far beyond the capability of human being. The emergence of system auxiliary technologies liberates the heavy burden of system administrators by leveraging the state-of-art data mining technologies. There are mainly two types of system auxiliary technologies: the system analytical techniques and the system autonomous techniques. For the first type, [10] [16] [17] utilized temporal mining and encoding theory to discover and analyze the event interaction behaviors from systems logs and then summarizes them in a brief way. Xu et al [18] introduced an anomaly detection algorithm to detect the system faults by utilizing the system source code and logs. These above methods are all off-line algorithms and are unable to tell the system to take reactions on-the-fly. For the second type, [15] used motif mining to model and optimize the performance of data center chillers. [7] proposed a signature-driven approach for load balance in the cloud environment with the help of utilization data. Our proposed solution can be categorized as an system autonomous technique. Different from existing works, the capacity estimation problem we face lies on the whole cloud environment scale.

B. Behavior Modeling and Prediction for System

In operating system area, caching is one of the techniques to improve the performance of the system through forecasting. *Partitioned Context Modeling* (PCM) [11] and

Extended Partitioned Context Modeling (EPCM) [12] are two statistical based caching algorithms, they models the file accesses patterns to reliably predict upcoming requests. These two method shows much better performance over the traditional caching algorithms like *Least Recent Used* (LRU), *Least Frequent Used* (LFU) and their extensions. While in the area of modern large-scale system behavior prediction, there is only little related works. Different from the traditional shared memory scenario, the virtual machines in the cloud environment can not be shared and reused due to the security issue. Therefore, there should be multiple copies of virtual machines prepared for multiple requests. The work of [6] also studied the resource prediction problem in cloud. But it focuses on predicting VM resource (CPU, memory, etc) for individual VMs. For our work, rather than predicting the resource usage within VMs, we aim to predict the VMs demands of the whole cloud.

C. Virtual Environment Management

Virtual environment management technologies are mainly based on control theory, meaning that the actions are taken after the event happened. Autoscaling proposed by Amazon [2] is the customer side auto-scaling management components, which allow customers to set up their own rules to manage the capacity of their cloud resources. This method focuses on the customer side post-processing of capacity tuning rather than the provider side. For the provider side, [9] proposed a self-adaptive system based on temporal data mining to reduce the VM provisioning time. In [14], the resources allocation problem is modeled as a stochastic optimization problem with the objective of minimizing the number of servers as well as the SLA penalty. [13] proposed a technique to enable the auto-scaling of visualized data center management servers. All these works are focusing on the resource allocation and scheduling with a fix amount of resources, while our work aims at estimating the total amount of resources for the whole cloud environment.

VI. Conclusion

Cloud service enables the IT infrastructure cost of the companies to be utilization efficient, and the capacity estimation makes the cloud service platform to be elastic itself. In this paper, we proposed an effective system for cloud capacity planning based on the knowledge learnt from the requests history. We leveraged the time series ensemble method to predict the provisioning demands and utilized the life span distribution of VM to estimate the de-provisioning requests. The experimental results demonstrate that our method is precise on capacity estimation and can effectively reduce the maintenance cost of the cloud environment.

There are several future directions to improve the precision of capacity estimation. First of all, there are many other historical trace data available and it is interesting to incorporate them to improve capacity estimation. Second, the capacity estimation method proposed in this paper is a kind of mid-term estimation, which can only reduce the maintenance cost of the cloud environment. A challenging problem is to extend the current work for long-term estimation to reduce the hardware cost. Moreover, we can also combine the utility monitor data from the hypervisor to realize the short-term capacity estimation.

Acknowledgement

The work is partially supported by National Science Foundation (NSF) under grant IIS-0546280.

References

[1] Stationarity Assumption. http://en.wikipedia.org/wiki/Stationary_process.

[2] Amazon AutoScaling. http://aws.amazon.com/autoscaling/.

[3] Avrim Blum. On-line algorithms in machine learning. In *Proc. of the workshop on Online Algorithms*, 1996.

[4] Amazon EC2. http://aws.amazon.com/ec2/.

[5] IBM Smart Cloud Enterprise. http://www-935.ibm.com/services/us/igs/cloud-development/.

[6] Zhenhuan Gong, Xiaohui Gu, and John Wilks. Press: Predictive elastic resource scaling for cloud systems. In *CNSM*, 2009.

[7] Zhenhuan Gong, Prakash Ramaswamy, Xiaohui Gu, and Xiaosong Ma. Siglm: Signature-driven load management for cloud computing infrastructures. In *IWQoS*, 2009.

[8] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. The cost of a cloud: Research problems in data center networks. *Computer Communication Review*, 2009.

[9] Yexi Jiang, Chang-Shing Perng, Tao Li, and Rong Chang. Asap: A self-adaptive prediction system for instant cloud resource demand provisioning. In *ICDM*, 2011.

[10] Jerry Kiernan and Evimara Terzi. Constructing comprehensive summaries of large event sequences. In *Proc. KDD*, 2008.

[11] Thomas M. Kroeger and Darrel D.E Long. The case for efficient file access pattern modeling. In *Proc. HotOS*, 1999.

[12] Thomas M. Kroeger and Darrel D.E. Long. Design and implementation of a predictive file prefetching algorithm. In *USENIX*, 2002.

[13] Shicong Meng, Ling Liu, and Vijayaraghavan Soundararajan. Tide: Achiving self-scaling in virtualized datacenter managment middleware. In *MiddleWare*, 2010.

[14] Swarna Mylavarapu, Vijay Sukthankar, and Pradipta Banerjee. An optimized capacity planning approach for virtual infrastructure exhibiting stochastic workload. In *SAC*, 2010.

[15] Debprakash Patnaik, Manish Marwah, Ratnesh Sharma, and Naren Ramakrishnan. Sustainable operation and management of data center chillers using temporal data mining. In *Proc. KDD*, 2009.

[16] Wang Peng, Haixun Wang, Majin Liu, and Wei Wang. An algorithmic approach to event summarization. In *Proc. SIGMOD*, 2010.

[17] Wei Peng, Chang-Shing Perng, Tao Li, and Haixun Wang. Event summarization for system management. In *Proc. KDD*, 2008.

[18] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Micheal I. Jordan. Detecting large-scale system problems by mining console logs. In *SOSP*, 2009.