ASAP: A Self-Adaptive Prediction System for Instant Cloud Resource Demand Provisioning

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

***Abstract*—The promise of cloud computing is to provide computing resources instantly whenever they are needed. The state-of-art virtual machine (VM) provisioning technology can provision a VM in tens of minutes. This latency is unacceptable for jobs that need to scale out during computation. To truly enable on-the-fly scaling, new VM needs to be ready in seconds upon request.**

In this paper, We present an online temporal data mining system called *ASAP*, to model and predict the cloud VM demands. *ASAP* aims to extract high level characteristics from VM provisioning request stream and notify the provisioning system to prepare VMs in advance. For quantification issue, we propose *Cloud Prediction Cost* to encodes the cost and constraints of the cloud and guide the training of prediction algorithms. Moreover, we utilize a two-level ensemble method to capture the characteristics of the high transient demands time series. Experimental results using historical data from an IBM cloud in operation demonstrate that *ASAP* significantly improves the cloud service quality and provides possibility for on-the-fly provisioning.

***Keywords*-cloud service; time series prediction; service quality improvement.**

I. Introduction

A. Motivation

Cloud computing gradually becomes a ubiquitous choice for modern IT-solution of business activities. The infrastructure as a service (IaaS) paradigm of cloud computing service is an elastic and economical choice for business IT support. However, the long waiting time of VM provisioning hampers the future popularization of cloud computing. The state-of-art VM provisioning technology costs tens of minutes on average to provision a VM. Since computing resource cannot be instantly ready in cloud, some IT enterprise customers still prefer over-capacitated traditional data center for time-sensitive computing, even with a huge overhead cost for the hardware, infrastructure, maintenance, and energy [3]. Such hesitation from customers shows that there is still a big gap in providing *instant resource provisioning*.

From the perspective of hardware and software technologies, there is dim hope to immediately and significantly reduce the VM provisioning time. The technology of streaming VM [7] allows the customer to preview the VM before it is entirely ready. However, the VM is still not instantly available until enough proportion of it is ready. There will be some time before this technology can fully answer the need of instant provisioning. From the business perspective, a simple yet straightforward solution to this problem is to ask all the customers to provide future VM requests schedule so the cloud service provider can prepare all the VMs ahead of time. However, this is impossible for many reasons: (1) The customers have no obligation and usually unwilling to provide their schedule. (2) The customers themselves are unable to know when the computing resources are needed. (3) The constituents of customers are always changing. (4) The actual schedule may change at any time.

Facing these technology limitations and business constraints, we believe that the only practical, achievable and effective solution to provide instant cloud is to *predict the demands and prepare the VMs in advance*.

B. Our Contributions

Cloud resource prediction is a very challenging task due to the unbalanced demand distribution, dynamic changing requests, and continuous customer turnover. Our empirical studies show that applying traditional time series prediction techniques on cloud resource provisioning can not achieve acceptable performance. Moreover, traditional prediction techniques are unable to dynamically change the prediction strategy according to actual cloud environment. In this paper, we propose a self-adaptive prediction solution to enable the instant provisioning. Our contributions can be summarized as follows:

1) We introduce an asymmetric and heterogenous measurement called *Cloud Prediction Cost* in the context of cloud service to evaluate the quality of the prediction results.
2) We design *ASAP*, a three-module system that leverages two-level ensemble algorithm to predict the multi-type VM demands based on the request history.
3) We present a series of experiments on real data to demonstrate that our system is able to significantly

reduce the averaging provisioning time while also well control the workload of resource.

The rest of this paper is organized as follows. We discuss the problem of VM demands prediction and the framework of *ASAP* in next 2 sections. In section 4, we introduce the *Cloud Prediction Cost*. In section 5, we discuss the details of prediction oriented ensemble method. An extensive evaluation results of our system are reported in section 6. The related works are discussed in section 7. Finally, we summarize the paper and discuss the future work in section 8.

II. Cloud Resource Provisioning

A. Problem Description

In the IBM Smart Cloud Enterprise (SCE) platform, each VM request contains 21 features such as *Customer ID*, *VM Type*, *Request Start Time*, etc. For this study, we focus on the aggregated time series on features *VM Type* and *Request Start/End Time*. Further exploration of time series in other perspectives is one of our future directions.

In summary, the problem of VM demands prediction can be described below:

1) Given a requests record stream S, an efficient and proper mechanism is required to properly extract the high level aggregated information from the stream and transform them into multiple time series $\mathcal{T} = (T_1, T_2, ..., T_k)$, one for each image type.
2) Getting the time series set $\mathcal{T}$, a model selection and training mechanism is required to accurately, periodically and timely generate the corresponding prediction models $\mathcal{P}$.
3) Acquiring the up-to-date prediction model set $\mathcal{P}$, an effective and accurate time series predictor is required to predict the amount of demands of time t_{n+1} based on historical request records.

B. Challenges in Cloud Resource Provisioning

Through experimental study, we found the following difficulties for VM requests prediction: (1) The distributions of different VM types demands are heavily unbalanced and always changing. (2) In the presence of dynamic varying customer group, for long-term perspective, some of the time series contain random factors that can mislead the prediction algorithms. (3) There is a variety of different types of time series with quite different characteristics. No single existing prediction algorithm is able to perform well on all of these time series. As a result, the requests prediction tasks are not as simple as a straightforward time series prediction task.

III. The System Framework

Figure 1 shows the framework of *ASAP* system. As depicted, our system can be divided into three modules.

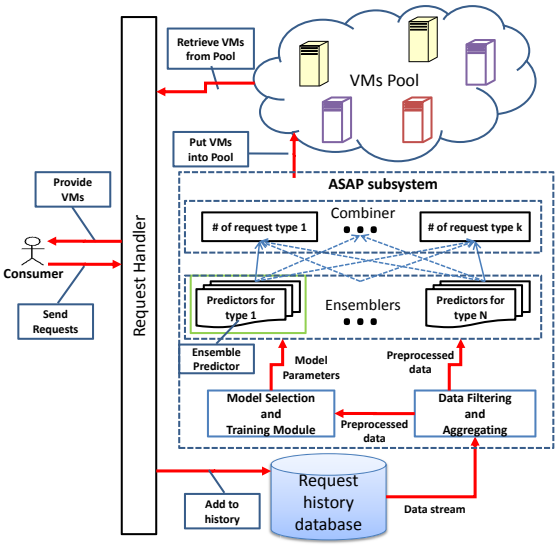


Figure 1. The framework of *ASAP* system

1) Raw Data Filtering and Aggregating Module: This module conducts the preprocessing works for the follow-up modules. It handles tasks such as filtering out unnecessary information from the raw data and extracting high-level characteristics from the filtered data to generate the time series.

2) Model Generation Module: Model generation module is responsible for building the models of separate predictors. It periodically selects and trains the most suitable models of each predictor based on the latest data feed by the raw data preprocessing module. Once the work is finished, the new parameters of the trained models are stored into a particular file and the demand prediction module is notified.

3) Demand Prediction Module: Demand prediction module reconstructs the prediction models according to the information stored in the model file. It then utilizes regression ensemble and correlation ensemble, a two-stage ensemble mechanism, to predict the future demands.

IV. Cloud Prediction Cost

In cloud demands prediction, the consequence of over-prediction and under-prediction is totally different. An over-prediction has no negative effect to the customer but causes resource waste of the cloud service provider, while an under-prediction saves the cost of resource but lower the service quality. Considering the uniqueness of cloud demands prediction, we propose *Cloud Prediction Cost* (CPC), an asymmetric and heterogenous cost measurement to guide the model selection and training.

Suppose for a certain type of VM, the real demands at time t is $v(t)$ and the predicted number at time t is $\hat{v}(t)$, where $\hat{v}(t)$ is a function of the historical values $v(t-1)$, ..., $v(t-i)$. We define two functions $T(v(t), \hat{v}(t))$ and

$R(v(t), \hat{v}(t))$ to represent the cost of service quality and the cost of idled resource respectively.

Cost of service quality: When a VM provisioning request arrives, the system can simply transfer the ownership to the customer and change related records if a VM of the same type is already prepared in advance. This only take a few seconds at most. We denote this short provisioning time as T_{hit}, which is similar to the "hit" in system cache scenario. If there is no prepared VM to satisfy customer's request, the VM has to be prepared on-the-fly, which we call it a "miss". The wait time is denoted as T_{miss} where $T_{miss} \gg T_{hit}$. For current real cloud service system, the on-the-fly provisioning time, T_{miss}, is tens of minutes for mechanical operations alone. For enterprise clouds, quality assurance process for software integrity and security may even take days if there is any manual process involved.

One simple form of $T(v(t), \hat{v}(t))$ can be modeled as the total provisioning time of all VMs. When over-prediction, i.e. $v(t) \leq \hat{v}(t)$, all provisioning requests can be immediately fulfilled. When under-prediction, i.e. $v(t) > \hat{v}(t)$, all the under-predicted portion of VMs need to be prepared on-the-fly. Therefore, the cost of VM provisioning time is:

$$T(v(t), \hat{v}(t)) = min(v(t), \hat{v}(t))T_{hit} + max(0, v(t) - \hat{v}(t))T_{miss}. \quad (1)$$

Cost of idled resources: This is the non-billable cost of resources including cost of disk spaces, network bandwidth, electricity and the labor cost etc. In this paper, we assume the idled resource cost for each VM in a unit time is identical regardless of the type of VM images. We define the cost function for idled resources as:

$$R(v(t), \hat{v}(t)) = max(0, \hat{v}(t) - v(t))R_{vm}, \quad (2)$$

where R_{vm} denotes the cost of a single idled VM.

Combining Eq.(1) and Eq.(2), a simple cost function for prediction quality is quantified by Eq.(3).

$$C = \begin{cases} \beta v(t)T_{hit} + (1-\beta)(\hat{v}(t) - v(t))R_{vm}, & \text{if } v(t) < \hat{v}(t) \\ \beta(\hat{v}(t)T_{hit} + (v(t) - \hat{v}(t))T_{miss}), & \text{if } v(t) \geq \hat{v}(t). \end{cases} \quad (3)$$

where β is used to tune the importance between service quality cost and idled resources cost.

The best predictor of cloud resource demands prediction is the one that achieves the minimum total cost between the predict value $\hat{v}(t)$ and the real value $v(t)$. Cost functions vary for each cloud. Our method can work with any cost function that satisfies the following two properties:

1) **Non-negative**: $T(v(t), \hat{v}(t)) \geq 0$ and $R(v(t), \hat{v}(t)) \geq 0$ for all $v(t)$ and $\hat{v}(t)$.
2) **Monotonic**: If $v_1(t) - \hat{v}_1(t) \geq v_2(t) - \hat{v}_2(t)$, then $T(v_1(t), \hat{v}_1(t)) \geq T(v_2(t), \hat{v}_2(t)$. Similarly, if $\hat{v}_1(t) - v_1(t) \geq \hat{v}_2(t) - v_2(t)$, then $R(v_1(t), \hat{v}_1(t)) \geq R(v_2(t), \hat{v}_2(t))$.

A. Periodical Model Updating

The philosophy of "train once, predict forever" is not suitable for cloud VM demands prediction. Due to the unique characteristic of cloud service, the time series of VM type is highly transient. In order to capture the up-to-date characteristics of the time series, *ASAP* periodically updates the prediction models based on the new requests history. For each updating, *ASAP* utilizes grid search strategy and 10-fold validation method to pick the best parameters combination of each prediction model for each VM type.

V. Demands Prediction

The core of the demands prediction module is a two-level ensemble algorithm. The first level is a regression based ensemble that combines the results of different prediction models of the same VM type. The second level ensemble considers the relationship between different VM types, and utilizes their correlation to help improve the robustness of prediction.

A. Base Prediction Methods

In *ASAP*, we employ a set of different prediction techniques as the base predictors. The prediction techniques are listed in Table I.

Prediction Method	Description
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 I
Time Series Prediction Algorithms

B. prediction ensemble

For the sake of well handling the prediction task, we propose a prediction ensemble method to combine the power of individual prediction techniques. Our method is inspired by the classification based online ensemble algorithm [1]. The goal of the ensemble method is to *make the ensemble predictor more robust and to make its precision close to the best predictor for different types of time series.*

Suppose the predicted value for predictor $p \in \mathcal{P}$ is $\hat{v}_p$ and its corresponding weight at time t is $w_p^{(t)}$, the predicted value for a certain VM type at time t is

$$\hat{v}^{(t)} = \sum_p w_p^{(t)} \hat{v}_p, \quad \text{subject to } \sum_p w_p^{(t)} = 1. \quad (4)$$

Initially ($t = 0$), we have $w_p^{(0)} = \frac{1}{|\mathcal{P}|}$ for every predictor p, so the predictors have the same contributions to the combined prediction result.

In order to update the weights, we calculate the relative error $e_p^{(t-1)}$ caused by predictor p at time $t-1$ according to

$$e^{(t)} = \frac{c^{(t-1)}}{\sum_p c_p^{(t-1)}} w^{(t-1)}, \tag{5}$$

where $c_p^{(t-1)}$ is the prediction cost and can be given by any kind of cost functions, like MAE, LSE, MAPE and CPC (as defined in Section IV).

Note that the relative errors cannot be used as the new weights of the predictors since they are not normalized. As the final predicted value is the linear combination of all the results of individual predictors, $w^{(t)} = \frac{e^{(t)}}{\sum_p e_p^{(t)}}$ is used to make the constraint $\sum_p w_p^{(t)} = 1$ holds.

C. Correlation Ensemble

In order to mitigate the disturbance of the "noisy" data in real cloud service scenario, we also make use of the correlation information between time series to help prediction. For instance, the demands time series of the same software with different platform (32-bit and 64-bit) are highly correlated.

In *ASAP*, the correlations between time series are used to help post-process the prediction. Suppose $cov_{ij}^{(t)}$ denotes the covariance between resource types i and its jth correlated resource, the post-processed predicted demand for type i at time t should be

$$\hat{u}_i^{(t)} = \frac{\sum_{j=1}^{k} cov_{ij}^{(t-1)} s_{ij} \hat{v}_k^{(t)}}{\sum_{j=1}^{k} s_{ij} cov_{ij}}, \tag{6}$$

where $s_{ij} = \bar{v}_i / \bar{v}_j$ denotes the different of scale between two time series and k is the number of strongly correlated time series. In our current design, we only considered the positively correlated time series, the negative influence consideration is one of our future works.

D. Reservation Controller

Reservation Controller is a module for *ASAP* to communicate with the VM provisioning system. It reserves the unused VMs and only notifies the VM provision system to prepare new VMs when all the reserves of the requested type are used up. The reservation controller provides a good buffer mechanism that effectively reduces the waste caused by over-prediction.

VI. System Evaluation

Our evaluation is based on the real historical VM requests data obtained from IBM's current cloud service platform. The historical demands data contains tens of thousands of VM requests with more than 100 different types. The data is spread over more than 3 month, starting from 02/13/2011 to 06/02/2011.

The goal of the evaluation is to answer the following questions: (1) Whether the ensemble method is more robust and effective than separate prediction techniques. (2) To what extent can *ASAP* decrease the provisioning time and how much idled resource would be caused by *ASAP*? (3) Whether the *Cloud Prediction Cost* is practical and flexible for cloud demands prediction?

A. Data Filtering and Aggregating

Data preprocessing is the first step of demands prediction. The raw request records in the data stream cannot be directly used for demands prediction for two reasons: (1) No prediction algorithm is able to build model directly on the low-level representation. (2) The raw data contains either useless request records or irrelevant features that can cause the prediction to be imprecise.

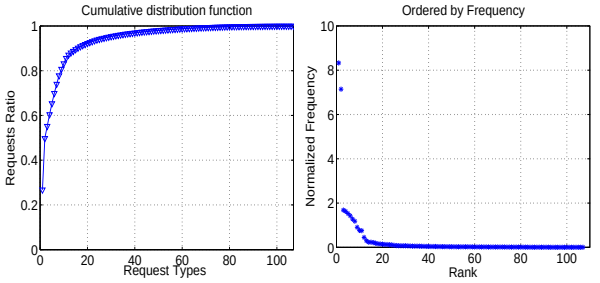
Not all the 21 features of the request record are useful. In our current work, we only consider *VM Type*, which illustrates the type of VM the customer requests; and *Request Start Time*, which indicates the time that the customer sends the request. Some other features like *Customer ID*, *Customer Priority* and *Data Center* will be considered in our future work for the research of personalized service quality improvement.

1) Time Series Aggregation Granularity: All the time series are generated by aggregating the request records via the VM types and request timestamps. The VM requests can be aggregated by hourly, daily and weekly. It is trivial to know that the coarser the granularity, the larger the demands in each time slot. If the time series is aggregated at a coarse granularity, i.e. weekly, it requires the system to prepare too much VMs and most of the prepared VMs should be idled for a long time. Moreover, compared with finer granularity, even a smaller portion of over-prediction would cause larger idled resources.

On the contrary, aggregation at a fine granularity, i.e. hourly, would make the time series lack of statistical significance and difficult to predict. Hence in later experiments, aggregation is performed daily.

2) VM Type Selection: We plot the corresponding cumulative distribution function (CDF) (Figure 2(a)) and rank the VM based on their requests frequency (Figure 2(b)). The CDF shows that the VM requests obey the 80/20 rule, that is, *more than* 80% *of the requests concentrates on less than 20% of VM types*. In the ranking plot, we observe that there is an inflection point between the 12th and 13th types, which explicitly divides the types into frequent and infrequent groups. The measurements like Coefficient of Variance, Skewness and Kurtosis on these minority time series also show higher values than those majority time series, indicating that the time series of these minority types are not regular enough to be modeled and predicted.

Based on the results of the above studies, we design the data filtering and aggregating module to periodically checks the change of rank, and only filters the frequent types before inflection point to generate their corresponding time series.



(a) CDF of VM Types (b) Ordering Types by Frequency

Figure 2. CDF plot and the requests frequency plot (normalized data)

Predictor	1st VM Type	2nd VM Type	3rd VM Type	Top 12 Avg
MA	437305	532132	76370	152402
AR	300687	391630	83292	137046
ANN	388535	247085	37052	106113
GEP	154440	473347	33745	112149
SVM	1010422	890447	88737	234225
Ensemble	158862	254190	53327	88679

Table II
THE COST OF PREDICTION ALGORITHMS UNDER DIFFERENT MEASUREMENTS

For those minority VM types, We can simply apply naive strategies, e.g. prepare a fixed number of VMs for each infrequent VM type. Since these VM types are infrequent, the cost of prediction error is insignificant.

B. Prediction Precision

We compare the prediction performance of base predictors with our proposed ensemble predictor under CPC measurement. For the parameters of CPC, we set $\beta = 0.5$ (treat the cost of provisioning time and the cost of idled VMs equally), $T_{miss} = 1040$ (if one VM is not prepared in advance, the customer need to wait for more than 10 minutes), $T_{hit} = 5$ (if one VM is prepared, the customer can get it in 5 seconds), $R_{vm} = 400$ (the cost of one idled VM).

In order to evaluate the robustness of the predictor, we pick the time series of the top 12 (before the inflation point in Figure 2(b)) most frequent VM types for experiments. All the time series are partitioned into two sets, the time series of last 30 days are used for testing, while the remains are used for training. For base predictors, grid search strategies are used to seek the best parameter combinations.

Table II shows the CPC cost of all the predictors on all the time series. Due to the space limit, we only list the CPC cost of the top 3 time series and the average cost of all the time series. It can be easily observed that the best predictor is different for different VM types. For example, GEP performs the best on the 1st VM type; ANN and GEP achieves good results on the 2nd VM type and the 3rd VM type, respectively. Moreover, the winner predictor of one VM type can also perform badly for other VM types. For example, ANN obtains a poor performance on the 1st VM type.

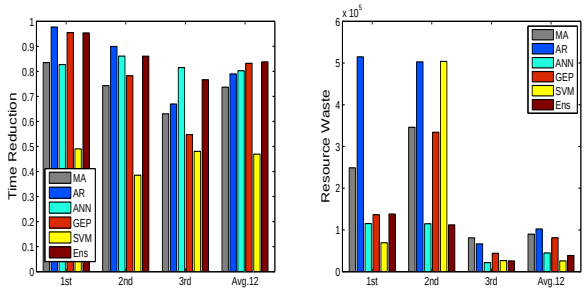
For our ensemble predictor, although it does not perform the best on any single VM type, it is very robust as its performance is always close to the winner predictor on all the types. The average CPC shows that our ensemble predictor has the best average performance, indicating its self-adaptation capability.

C. Detail Cost

1) Provisioning time reduction: The most important criteria to evaluate the performance is how much provisioning time can be saved. We calculate the proportion of saved time obtained by predictors based on Eq. (7).

$$P_{save} = \frac{\sum_t (v(t)T_{miss} - T(v(t), \hat{v}(t)))}{\sum_t v(t)T_{miss}}. \quad (7)$$

Figure 3(a) shows the proportion of time saved by each predictor. It is good to see that most of the predictors can significantly decrease more than 60% of the provisioning time. In the presence of large variations across time series, the saved time achieved by the predictor is not stable for different VM types. On average, our ensemble predictor performs the best due to its strong self-adaptation ability.



(a) Provisioning Time Reduction (b) Resource Waste

Figure 3. Average provisioning time reduction and average resource waste (normalized data)

2) Idled resources: The cost of idled resources is another evaluation criterion of the quality of prediction. It is true that an always over-predicted predictor can save a lot of provisioning time, but such a predictor would also waste a lot of resources. Figure 3(b) shows the amount of idled resource caused by each predictor. On average, the best resource saver is SVM, but its performance in time reduction is the worst. GEP achieves a good performance on time reduction, but it wastes the resources twice as much as our method.

D. Effect of β

As mentioned before, the parameter β is used to tune the importance between cost of service quality and that of idled resource. A higher β leads the ensemble predictor to assign larger weight to the over-predictor. The time reduction/idled resources curve in Figure 4 illustrates how β affects the balance between these two factors. As expected, when β increases, both the ratio of time reduction and the idled

resource increase. When β equals to 0, the cost of service quality loss is totally ignored, and the ensemble method predicts the demands very pessimistically. When β equals to 1, the cost of idled resources is huge but the average provisioning time reduction is the highest.

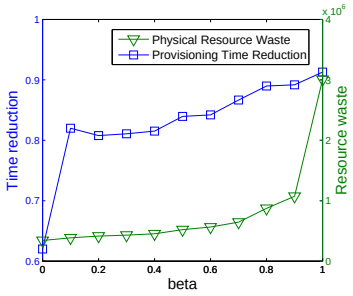


Figure 4. The effect of β

This experimental result confirms that β is an effective parameter to set the trade-off between the service quality loss and the cost of idled resources. In practice, it can be dynamically tuned whenever needed.

VII. Related Work

A. Data mining on Computer System Data

Due to the increasing complexity and scale of systems, the difficulty of system analysis is already far beyond the capability of human being. Researchers begin to make use of the data mining technology to help the system administrators to analyze, monitor and detect abnormal behaviors of the systems. [4] [5] [9] [10] utilized temporal mining and encoding theory to discover, analyze and summarize the event interaction behaviors from systems logs. [8] used motif mining to model and optimize the performance of data center chillers. [11] proposed a large-scale console logs mining system to detect the abnormal behaviors of distributed system. For all prior work, the patterns of normal behaviors are well defined and the analysis is based on relatively static and stable environments. While in our scenario, in the presence of the unstable customer constituency and volatile demand, we cannot directly use these methods for request prediction.

B. System behavior prediction

In traditional operating system area, the performance of cache heavily relies on the cache pre-fetching algorithm. [6] proposed a new algorithms called *Extended Partitioned Context Modeling* (EPCM) to model file accesses patterns to reliably predict upcoming requests. Their method showed much better performance over the classic cache algorithms and their extensions. While in the area of modern large-scale system behavior prediction, there is only little related works. The reason is partly due to its complexity. The work of [2] also studied the resource prediction problem in cloud. But their focus is 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.

VIII. Conclusion and Future Work

The long waiting time of VM provisioning hampers the further popularization of cloud computing. In order to support the on-demand provisioning, we design and implement an instant resource provisioning system called *ASAP* to handle the cloud VM demand prediction. Due to the unique cost structure of cloud service, we propose *CPC* to measure the performance of prediction model. A series of experiments demonstrate that *ASAP* can effectively reduce the waiting time while not causing much idled resource.

There are several future directions for this work. Firstly, we currently only consider the time series aggregated by time and VM type. Other stored features like customer priority and data center location may also provide clues for better prediction. Moreover, we can reasonably assume some VM requests are the scale-out attempts to alleviate workload from overloaded VMs. So we believe by monitoring resource utilization, we can predict the imminent provisioning requests and prepare the right types of VMs in advance.

Acknowledgement

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

References

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

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

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

[4] Yexi Jiang, Chang-Shing Perng, and Tao Li. Natural event summarization. In *CIKM*, 2011.

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

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

[7] Francois Labonte, Peter Mattson, Ian Buck, Christos Kozyrakis, and Mark Horowitz. The stream virtual machine. In *Proc. ICPACT*, 2004.

[8] 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.

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

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

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