Empirical Evaluation of Workload Forecasting Techniques for Predictive Cloud Resource Scaling

In Kee Kim, Wei Wang, Yanjun Qi, and Marty Humphrey
Department of Computer Science
University of Virginia
{ik2sb, wwang}@virginia.edu, {yq2h, humphrey}@cs.virginia.edu

***Abstract*—Many predictive resource scaling approaches have been proposed to overcome the limitations of the conventional reactive approaches most often used in clouds today. In general, due to the complexity of clouds, these reactive approaches were often forced to make significant limiting assumptions in either the operating conditions/requirements or expected workload patterns. As such, it is extremely difficult for cloud users to know which – if any – existing workload predictor will work best for their particular cloud activity, especially when considering highly-variable workload patterns, non-trivial billing models, variety of resources to add/subtract, etc. To solve this problem, we conduct comprehensive evaluations for a variety of workload predictors under real-world cloud configurations. The workload predictors cover four classes of 21 predictors: naive, regression, temporal, and non-temporal methods. We simulate a cloud application under four realistic workload patterns, two different cloud billing models, and three different styles of predictive scaling. Our evaluation confirms that no workload predictor is universally best for all workload patterns, and shows that Predictive Scaling-out + Predictive Scaling-in has the best cost efficiency and the lowest job deadline miss rate in cloud resource management, on average providing 30% better cost efficiency and 80% less job deadline miss rate compared to other styles of predictive scaling.**

***Index Terms*—Predictive Cloud Resource Scaling; Cloud Workload Prediction; Job Arrival Time Prediction; Performance Evaluation;**

I. Introduction

When and how to scale-out and scale-in a cloud application can be one of the most difficult challenges for a cloud application designer/deployer. Fundamentally, basing this decision on the current state of resources (e.g., VMs) for a given cloud application is usually simple and can be effective but is limited due to its reactive nature. For example, a standard use of Amazon Web Services' auto-scaling mechanism is to configure it to add more "worker VMs" when the average current CPU utilization stays above a certain threshold for a short period time. However, fluctuating or generally unpredictable workload patterns can quickly eliminate any potential performance or monetary improvements of such a policy/mechanism. In other words, intuitively any changes to the cloud application system state/configuration assume that the near-past workload will continue for the near-future.

Next-generation scaling approaches attempt to move beyond the limitations of such reactive systems by instead attempting to predict the near-future workloads. In such systems, generally, there are two components. The *workload predictor* functionality, and the *resource scaling* functionality, which allocates/deallocates cloud resources and maps work requests to specific resources. Over the past years, many studies have proposed predictive scaling approaches [1–19]. In general, due to the complexity of cloud environments, these approaches were often forced to make significant limiting assumptions in either the operating conditions/requirements or expected workload patterns. As such, it is extremely difficult for cloud users to know *which – if any – existing workload predictor will work best for their particular cloud activity*, especially when considering highly-variable workload patterns, non-trivial billing models, variety of resources to add/subtract, etc.

The goal of this work is to comprehensively evaluate existing cloud workload predictors, holistically, and often in a broader context than in the authors' evaluation methodology. Because the most common metric to evaluate workload predictors is accuracy for the future job arrivals, the first question we seek to answer is:

Question #1: *Which existing workload predictor has the highest accuracy for job arrival time prediction, when applied for different workload patterns?*

After considering the workload predictor in isolation, we evaluate it in combination with the resource scaling component. Therefore, the second question we seek to answer is:

Question #2: *Which existing workload predictor has the best cost efficiency and deadline miss rate (which represents performance and SLA requirements), when applied for different workload patterns, different scaling styles and different pricing models?*

Furthermore, some previous work employed predictive scaling-out, some employed predictive scaling-in, and some employed both. Given these choices of applying predictive scaling, the third question we seek to answer is:

Question #3: *Which style of predictive scaling (predictive scaling-out, predictive scaling-in, or both) achieves the best cost and performance benefit for particular cloud configurations (e.g. billing model, job deadline)? And how much benefit can be achieved?*

To answer these questions, we conducted comprehensive evaluations with a wide range of configurations of predictive cloud resource scaling using 15 existing workload predictors [5–19]. We also included 6 well-known machine learning predictors that have not been used for the predictive scaling before. In total, we examined 21 predictors, covering naive,

regression, temporal (time series) and non-temporal methods. We used 24 randomly generated workloads covering four common types of job arrival patterns [20, 21], which are growing, on/off, bursty and random. We also examined scaling operations including **RR** (Scaling-out: **R**eactive + Scaling-in: **R**eactive), **PR** (Scaling-out: **P**redictive + Scaling-in: **R**eactive), **RP** (Scaling-out: **R**eactive + Scaling-in: **P**redictive) and **PP** (Scaling-out: **P**redictive + Scaling-in: **P**redictive). We also considered both hourly and minutely pricing models. In our experiments, each configuration then covered one workload predictor, one workload, one scaling operation and one billing model. As a result, more than 4K ($21 \times 24 \times 4 \times 2$) configurations were examined. We run each configuration using the PICS [22] simulators to evaluate each configuration's cost and deadline miss rate. We chose PICS because it can accurately simulate real public IaaS clouds in short amount of time. Without PICS, it is both timely and financially infeasible to conduct such extensive experiments on real IaaS clouds

Based on the experimental results, we successfully answer the three questions posed previously. Here we summarize our findings and answers to each question:

To find the best workload predictor in terms of the accuracy for diverse workload patterns (Question #1): the accuracies of different workload predictors vary considerably. The best workload predictors in terms of statistical accuracy are usually orders of magnitudes more accurate than the worst ones. However, no workload predictor is universally the best for all workload patterns. Each workload pattern has its own best workload predictor. We show the best workload predictors for each workload pattern in Section IV-A.

To find the best workload predictor in terms of cloud metrics such as cost efficiency and SLA satisfaction (Question #2): the workload predictor with the highest accuracy is not necessarily the best in terms of cost and deadline miss rate. Additionally, no workload predictor is universally the best for any workload pattern and billing model. However, we observe that the best predictor (in terms of cost and deadline miss rate) for a particular workload pattern is always one of the top 3 most accurate predictors (in terms of statistical accuracy) of that workload pattern. We also show the best workload predictors for each workload pattern in terms of cloud metrics in Section IV-B.

To find the best style of predictive resource scaling in terms of providing the best cost and performance benefits (Question #3): both predictive scaling-out (**PR**) and predictive scaling-in/out (**PP**) significantly reduces cost and deadline miss rate over purely reactive scaling (**RR**). However, predictive scaling-in (**RP**) performs similarly to **RR**. Overall, **PP** always provides the lowest cost and deadline miss rate for all workload patterns and billing models. On average, **PP** provides 30% less cost and 80% less deadline miss rate compared to **RR** or **RP**, and **PP** offers 14% less cost and 39% less deadline miss rate compared to **PR**.

A key finding from the answering those questions is that users, who want to design new algorithm for predictive resource scaling, should consider top 3 workload predictors depending on workload patterns, and use **PP** for their scaling operations in order to archive better cost efficiency and deadline miss rate.

The contributions of this paper include:

- An extensive evaluation of 21 workload predictors for their accuracy to predict the future job arrival time.
- A comprehensive evaluation of workload predictors in terms of cost and deadline miss rates. This evaluation considers various workload patterns, three styles of scaling operations, and two different billing models.
- Complete answers for the questions regarding what is the best workload predictor and what is the best style for predictive resource scaling operations.

The rest of this paper is organized as follows: Section II describes workload predictors used in this work. Section III contains the experimental design of this work. Section IV provides evaluation results for all predictors. Section V contains related work and Section VI concludes this paper.

II. Workload Predictors

We collect a total of 21 workload predictors via an extensive survey of previous research on predictive resource scaling. Each predictor is categorized into one of the following classes: 1) naive, 2) regression, 3) temporal, and 4) non-temporal predictor. The detailed description for all 21 workload predictors is described in Table I.

1. Naive workload predictors: two types workload predictors are included in this class – ***mean*** and ***recent mean***-based (kNN – k Nearest Neighbor) methods.

2. Regression-based workload predictors: the predictors in this class can be split into category of global and local regressions. Each category can include linear (1-degree) or polynomial (2 or more degrees) models. In total, we use six regression-based predictors, which are **global** and **local regression with linear, quadratic, and cubic models** [23].

3. Temporal (Time-Series)-based Workload Predictors: there exist various temporal (time-series) methods for the future workload prediction because these predictors [5–15, 18, 19, 24] are widely used to analyze workload patterns for cloud computing as well as other domains of computer systems research. We use four categories of temporal models: **ES** (Exponential Smoothing), **AR** (Autoregressive), **ARMA** (Autoregressive and Moving Average), and **ARIMA** (Autoregressive Integrated Moving Average) [1, 3, 8–15, 19].

4. Non-temporal Workload Predictors: the predictors in this class have not been applied to the cloud resource scaling before. These predictors, however, have provided accurate prediction results within a deterministic amount of time. We consider several non-temporal approaches to predict the next job arrival time and select three categories of non-temporal prediction approaches: **SVM**s (Support Vector Machines), **decision tree**, and **ensemble methods** [23].

TABLE I: The Description of All 21 Workload Predictors.

Class	Predictor	Description
Naive Predictors	***mean***	The **mean**-based predictor forecasts a next job arrival time based on a mean arrival time of all previous jobs. For the scaling-out operation, the cloud application prepares cloud resources as if the next job will be arrived at the predicted result based on mean. For the scaling-in operation, the cloud application waits until the predicted next job's arrival time when a VM running by the cloud application is idle in order to increase a possibility of reuse of this VM.
	recent-mean (k**NN**)	The **recent mean**-based predictor (kNN) is a similar approach with mean-based predictor, but this uses the arrival time of recent k jobs and predicts the next job's arrival time based on a mean arrival time of those recent jobs.
Regression -based Predictors	***Global Regression Models***	The **global regression** models forecast the next job arrival time by creating a linear or polynomial regression model [23] using features including all previous job arrival time. Here we only consider job arrival time as the main variable. Therefore, these approaches are a single variable regression models.
	Local Regression Models	The **local regression** models [23] to estimate the next job arrival time. These approaches consist of two steps: 1) applying a kernel function to select job arrival samples and 2) creating linear or polynomial regression model based on the samples. In this work, we use kNN (k Nearest Neighbor) as the kernel function for the local regression models to select proper samples. kNN calculates a distance between a target object (e.g. next job arrival time) and all possible samples (e.g. past job arrival time) by using absolute or Euclidean distance function. kNN then selects proper local samples (e.g. k recent jobs). Based on the selected samples from the kNN, a linear or polynomial regression model is created, and predicts the next job arrival time. The major difference between the global and local regression is the size and similarity of job arrival samples used for creating a regression model. Local regression uses smaller number of samples that are more similar to the prediction target. The local regression models often have less overhead for model creation and workload prediction.
Temporal (Time-Series) -based Predictors	**WMA**	**WMA** (Weighted Moving Average) [3] is a weighted sum of observed dataset (e.g. past job arrival information) and sum of weight for each observed data is 1. WMA is calculated by $\sum_{n=1}^{k} w_n x_{t+1-n}, s.t. \sum_{n=1}^{k} w_n = 1$
	EMA	**EMA** (Exponential Moving Average) [19] a similar approach as WMA, but it gives more weight to the most recent observation of job arrivals. EMA predicts the future job arrival time by $s_t = \alpha x_t + (1-\alpha)s_{t-1}$. α is a smoothing factor ($0 < \alpha < 1$). If α is close to 1, EMA has less smoothing effect and gives more weight to the recent data, and vice versa.
	Holt-Winters DES	**Holt-Winters DES** (Double Exponential Smoothing) predicts the next job arrival time by capturing a smoothing value at time t ($s_t = \alpha x_t + (1-\alpha)(s_{t-1} + b_{t-1}),\ where\ s_1 = x_1$) and estimating the trend at time t ($b_t = \beta(s_t - s_{t-1}) + (1-\beta)b_{t-1}$, $where\ b_1 = x_1 - x_0$). x_0 is the first observation of raw data, α is a smoothing factor ($0 < \alpha < 1$) and β is a trend smoothing factor ($0 < \beta < 1$). Then, the next job arrival time is calculated by $s_t + b_t$.
	Brown's DES	**Brown's DES** predicts the next job arrival time by calculating $(2 + \frac{\alpha}{1-\alpha})s_t^{'} - (1 + \frac{\alpha}{1-\alpha})s_t^{''}$. $s_t^{'}$ is the first order exponential smoothing model and is expressed by $s_t^{'} = \alpha x_t + (1-\alpha)s_t^{''}$. x_t is current job arrival and α is a smoothing factor ($0 < \alpha < 1$). $s_t^{''}$ is double-smoothed statistics and is expressed by $s_t^{''} = \alpha s_t^{'} + (1-\alpha)s_{t-1}^{''}$.
	AR	**AR** (Autoregressive) [1, 8] is a linear combination of previous data of the target object (e.g. job arrival time). AR(p) model is expressed in $X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-1} + \varepsilon_t$. p is the order of AR model, φ_i is the set of parameters of the model, c is constant, and ε_t is white noise.
	ARMA	**ARMA** (Autoregressive and Moving Average) [9–13]. is a combined model of AR and MA (Moving Average) and ARMA(p, q) is expressed in $X_t = \sum_{i=1}^{p} \varphi_i X_{t-1} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + c + \varepsilon_t$. The first term is AR($p$) model with the order of p. The second term is MA(q) model with the order of q.
	ARIMA	**ARIMA** [14, 15] is a generalization of ARMA and provides a reliable prediction for non-stationary time-series data by integrating AR and MA models. ARIMA is expressed as ARIMA(p, d, q), where p is the order of AR, q is the order of MA, and d is the order of differencing model.
Non-temporal Predictors	**SVM**	**SVM** (Support Vector Machine) [23] is an optimal margin-based classifier that tries to find a small number of support vectors (data points) that separate all data points of two classes with a hyperplane in a high-dimensional space. With kernel tricks, it can be extended as a nonlinear classifier to fit more complex cases. SVM can be applied to the case of regression as well which contains all the main features that characterize the maximum margin based algorithm. At testing time, the (positive or negative) distance of a data point to the hyper-plane is output as the prediction result for regression. We consider both linear and non-linear SVM. We use linear kernel for Linear-SVM and Gaussian kernel for non-linear-SVM (Gaussian-SVM). Linear-SVM is to focus on the workloads that have relatively clear trend factors and Gaussian-SVM is to predict the workloads with non-linear characteristics.
	Decision Tree	**Decision tree** [23] is a non-parametric learning algorithm and it has also been used for both classification and regression problems. Decision tree creates a classification or regression model by applying decision rules derived from features of dataset. Decision tree is known as its simple (tree-based) structure and fast execution time for numerical data.
	Ensemble Prediction Models	**Ensemble prediction methods** [23] employ multiple numbers of predictors to obtain better generalizability and increase performance. Ensemble methods use bagging or boosting approaches to reduce variance (bagging) or bias (boosting) on prediction results. We select three ensemble methods including gradient boosting, extra-trees, and random forest.

III. Experiment Design

A. Design of Cloud Resource Management System

To evaluate all workload predictors, we designed a cloud management system for resource scaling as shown in Figure 1. This system consists of three modules: job portal, resource management module (RMM) and predictive scaling module (PSM). The job portal is an entry for the workloads (jobs from end-users). A job's arrival triggers two other modules. A newly arrived job is passed to the RMM. The RMM selects a proper VM based on the job's duration and deadline. More specifically, the RMM creates a list of VMs that meets the deadline of the job by comparing the performance of different VM types with the job's duration and deadline. the RMM then selects the most cost efficient VM (i.e., cost/performance-ratio [25]) from the candidates. Note that the algorithm, used in this cloud resource management, focuses primarily on improving the job deadline satisfaction than reducing the cloud cost. Once a proper VM is selected, the RMM schedules the job to the selected VM via "Earliest Deadline First" scheduling. The selected VM is then used to execute the job. A new job's arrival activates the PSM as well. The job's arrival information is stored in the workload repository. The workload information from this repository is used by both predictive scaling-out and scaling-in.

Predictive Scaling-out Operation (Algorithm 1) is triggered when a job arrives. A prediction obtains proper amount of job arrival samples for prediction (**ln 1**) and forecasts the next job's arrival time in the future (**ln 2**). Based on the prediction result, this operation chooses a proper type of VM

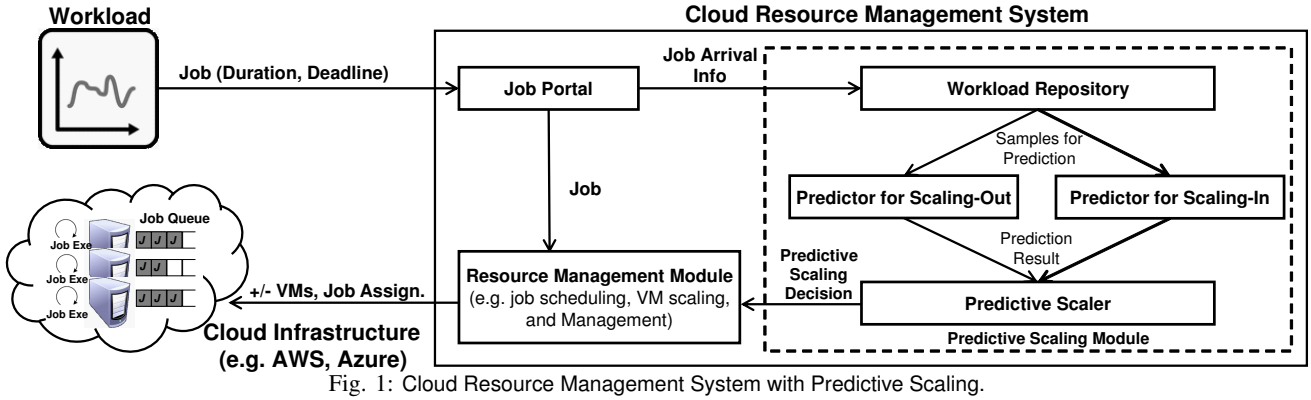


Fig. 1: Cloud Resource Management System with Predictive Scaling.

for the future job (**ln 3**) as explained in the previous paragraph (In Algorithm 1, we assume that the duration and deadline of the future job are known). This operation selects a list of currently running VMs to execute the next job (**ln 4**). If the list is empty (**ln 6**), then a new VM will be created for the next job (**ln 7**).

Algorithm 1 Predictive Scaling-Out

Require: A new job arrives
1: $samples \leftarrow$ get_samples_for_prediction ()
2: $next\_job \leftarrow$ predict_next_job_arrival ($samples$)
3: $vm\_type \leftarrow$ select_proper_vm_type ($next\_job$)
4: $vm\_list \leftarrow$ current_running_vms (vm_type)
5:
6: **if** vm_list is empty **then**
7: create_vm (vm_type, $time\_to\_start$)
8: **else**
9: do_nothing ()
10: **end if**

Algorithm 2 Predictive Scaling-In

Require: vm is idle
1: $samples$ = get_samples_for_prediction (vm)
2: $next\_job$ = predict_next_job_arrival ($samples$)
3:
4: **if** $next\_job$ arrival $\leq max\_startup\_delay$ **then**
5: $scale\_in\_time \leftarrow$ next billing boundary after $next\_job$ arrival
6: **else**
7: $scale\_in\_time \leftarrow$ this billing boundary
8: **end if**
9:
10: **repeat**
11: **if** $next\_job$ arrives **then**
12: go to new_job_processing_routine()
13: **end if**
14: **until** $scale\_in\_time$
15:
16: terminate (vm)

Predictive Scaling-in Operation (Algorithm 2) is triggered when a VM is idle – no jobs in both processor and work queue. The workload predictor for scaling-in operation estimates the next job arrival time to the idle VM (**ln 1–2**). Then we compare the predicted job arrival time with maximum VM startup delay [26]. If the job arrival time is smaller than the max startup delay, we choose to keep this VM for at least max startup delay time; otherwise, we choose to terminate this VM (**ln 4–8**). The rationale behind this choice is explained as follows. For any new job, starting a new VM for it takes (startup time + job duration) to execute it. However, if we use existing idle VM, it takes (job arrival time + job duration) to execute it. Therefore, if job arrival time is smaller than startup time, it is faster and cheaper to use the existing VM; otherwise, it is cheaper to use new VM. Moreover, we choose to terminate a VM only at nearest billing boundary, because we have already paid for this billing cycle. If the next job arrives within the predicted time, the job is assigned/processed by this VM (**ln 12**). If there is no more jobs until the predicted time, this VM is terminated (**ln 16**).

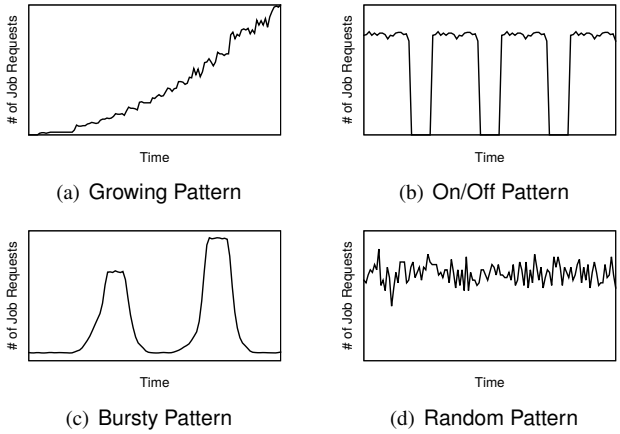


Fig. 2: Cloud Workload Patterns. X-axis means time and Y-axis means the number of requests (e.g. the number of jobs).

B. Cloud Workload Patterns

We generate synthetic workloads based on four cloud workload patterns (Figure 2). We create 6 workloads for each workload pattern with different mean and standard deviation of job arrival time/duration to reflect various and realistic cloud usage scenarios. The detail of each dataset is described in Table II. For the growing workload pattern, we first generate a quadratic base function and then apply Poisson distribution to randomize the arrival time of a particular job. The on/off workload pattern has four active periods and three inactive periods. For the active periods of the on/off workload pattern, we use growing and declining quadratic functions. The bursty workload pattern has 6 – 7 peak spikes periods and other stable periods. To generate the random workload pattern, we use Poisson distribution for the random job arrivals.

[1]This is job duration on smallest (worst performance) VM instance (small EC2 instance [27] in our experiment design). By using the job duration and deadline, the cloud resource management system (Section III-A) determines a proper VM type that can meet deadline.

TABLE II: Workload Dataset Information.

Workload	# of Jobs	Mean Job Arrival Time	Std. Dev. of Job Arrival Time	Job[1] Duration	Job Deadline
Growing	10K	25s	60.5	Average: 450s, Std.Dev.: 270	Average: 500s, Std.Dev.: 250
On/Off			375.5		
Bursty			35.3		
Random			270		

C. Implementation Details

We implemented the cloud resource management system on top of PICS [22], a Public IaaS Cloud Simulator. In addition to the simulation model, we implement all the predictors (described in Section II) and scaling-in/out mechanisms using numpy ver. 1.8, Pandas (Python Data Analysis Library), statsmodels, and scikit-learn machine learning packages.

Choosing the parameters and the training sample size are very crucial to all workload predictors in order to provide the best possible prediction performance. For the decision of the training sample size for predictors, it is obvious that a predictor should use as many sample as possible to maximize the accuracy of the prediction. However, large size of training samples increases the overhead of prediction. A constraint for the prediction is that the predictors should be able to forecast the next job arrival time before the actual job comes to our cloud application. To this end, we tested a wide range of sample size and determine the size based on a tradeoff between the prediction overhead and the prediction accuracy. In this work, all predictors (except global regression approach) uses 50 – 100 of most recent job arrival samples for forecasting the future job arrival time prediction.

For the parameter selection of the workload predictors, we use either a performance-based or a grid search approach. For AR, ARMA, and ARIMA model, we employ the performance-based parameter selection, and we choose the first order of these three models. (e.g. AR(1), ARMA$(1,1)$, ARIMA$(1,1,1)$) Higher-order of these models is not desirable because these three workload predictors require high computation time. It is often impossible for the higher order of these models to predict the next job arrival time before the actual job arrives. For other temporal-based workload predictors (e.g. EMA and DES), we leverage a grid search approach for the parameter selection that tries every possible parameter within its range constraints (e.g. $0 < \alpha < 1$ and $0 < \beta < 1$ for Holt-Winters DES). Moreover, SVM predictors require soft and kernel parameters (e.g. Gaussian-SVM needs both parameters and Linear-SVM requires only the soft margin parameter). We choose these parameters that result in best prediction performance. The range we have considered is from $10e^{-6}$ to $10e^{3}$ for both parameters.

IV. Evaluation

A. Evaluation of Accuracy for Workload Predictors

As the first evaluation of this work, we measure all 21 workload predictors' accuracy for predicting the future job arrival time under four different workload patterns. We employ MAPE (Mean Absolute Percentage Error) [23] to statistically measure the prediction accuracy.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{T_{actual,i} - T_{predicted,i}}{T_{actual,i}}\right| \quad (1)$$

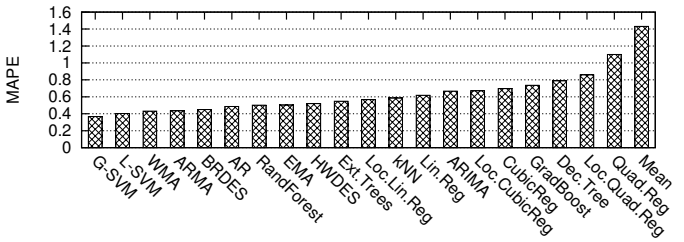


Fig. 3: MAPE Results of All 21 Predictors.

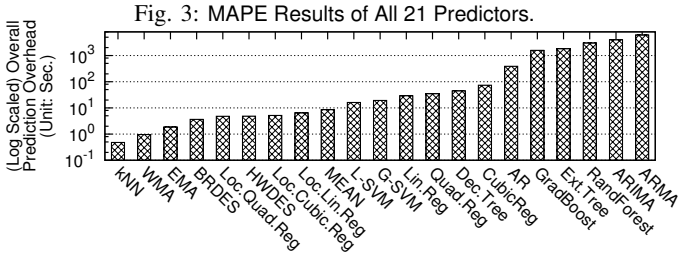


Fig. 4: Prediction Overhead (Log-scaled Sum of Total Prediction Time) for All the 21 Workload Predictors. (Sample Size: 50 recent jobs)

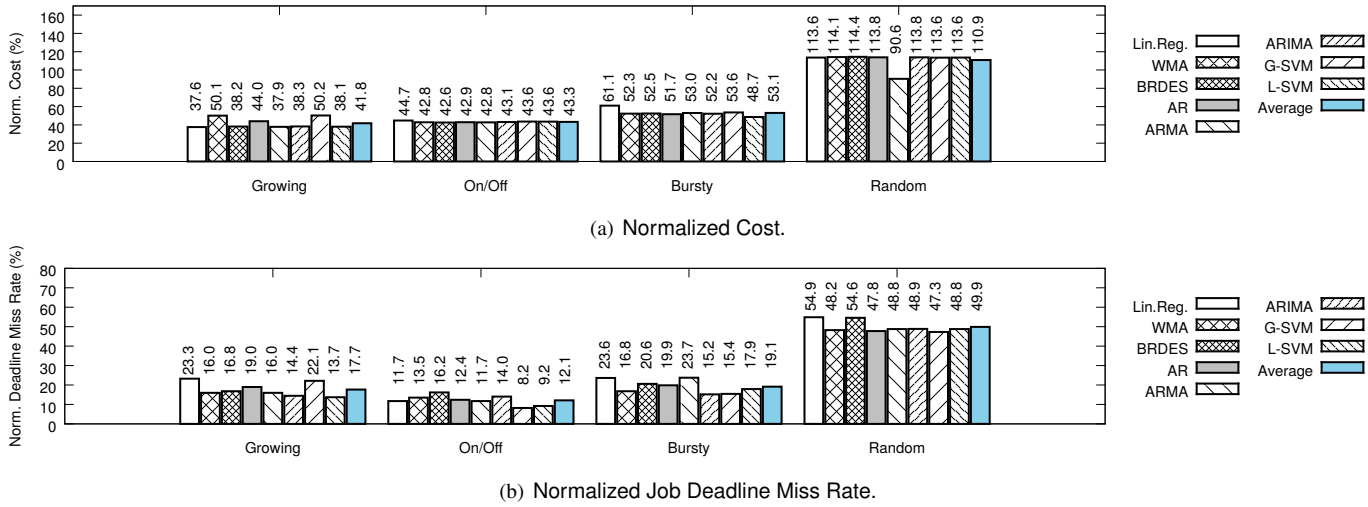
Figure 3 shows MAPE results of all 21 predictors. Average of the MAPE for all 21 predictors is 0.6360. Overall, two SVM approaches have the best MAPE results. Other three best predictors are WMA, ARMA, and Brown's DES. The MAPE value of Gaussian-SVM predictor (the best predictor) is 0.3665, which is 42% less result than average of all predictors. However, the best predictors in overall do not necessarily mean the best predictor for each workload pattern, so we also present the performance of all predictors for each workload pattern.

Table III shows the MAPE results for each workload pattern. Due to page limitation, Table III only contains the best 3 predictors, average results, and the worst predictor. As shown in Table II, different workload patterns have different best predictors: **Linear-SVM** (growing), **Gaussian-SVM** (on/off), **ARIMA** (bursty), and **Gaussian-SVM** (random). Table III also shows that the MAPE results of the top three predictors are very similar to each other for growing and bursty workloads. These workloads have clear trend patterns, and many good workload predictors can successfully detect these patterns when using enough job arrival samples. The MAPE results of random workload are lower compared to other workloads, indicating random workload is the most difficult to predict. This difficulty is primarily caused by the fact that job arrival times in random workloads do not have clear trend pattern for predictors to discover.

Moreover, obtaining the prediction results in a deterministic

TABLE III: The MAPE Results of Workload Predictors Under Four Different Workload Patterns. (WL: Workload, GR: Growing, OO: On/Off, BR: Bursty, RN: Random)

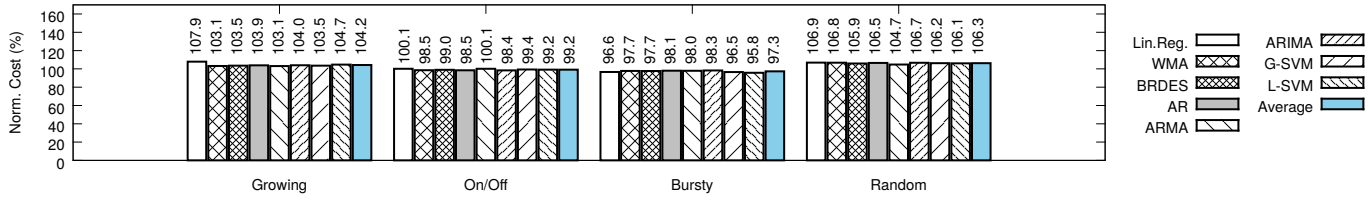
WL	Rank	Predictor	MAPE	WL	Rank	Predictor	MAPE
GR	**1**	**L-SVM**	**0.28**	**OO**	**1**	**G-SVM**	**0.22**
	2	AR	0.29		2	ARMA	0.30
	3	ARMA	0.30		3	L-SVM	0.44
	Avg.	–	0.51		Avg.	–	0.69
	Worst	Qua.Reg	2.75		Worst	Loc.Cub.Reg	1.25
BR	**1**	**ARIMA**	**0.38**	**RN**	**1**	**G-SVM**	**0.45**
	2	BRDES	0.41		2	Lin.Reg	0.46
	3	L-SVM	0.43		3	L-SVM	0.46
	Avg.	–	0.75		Avg.	–	0.52
	Worst	mean	3.35		Worst	Dec.Tree	0.62



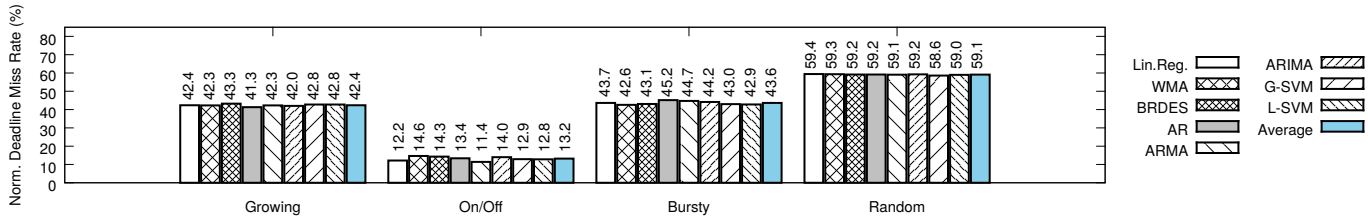
(a) Normalized Cost.

(b) Normalized Job Deadline Miss Rate.

Fig. 5: **Case #1** – Normalized Cost and Job Deadline Miss Rate of **PR** (Scaling-Out:**P**redictive + Scaling-In:**R**eactive) – Hourly Pricing Model.



(a) Normalized Cost.



(b) Normalized Job Deadline Miss Rate.

Fig. 6: **Case #1** – Normalized Cost and Job Deadline Miss Rate of **PR** (Scaling-Out:**P**redictive + Scaling-In:**R**eactive) – Minutely Pricing Model.

amount of time is a critical issue for the predictive resource scaling. We also measured computation overhead of predictors (Figure 4). kNN, WMA, and EMA are the fastest predictors. The overall prediction time for 10K jobs of these three predictors are 0.48 (kNN), 0.96 (WMA), and 1.86 seconds (EMA). However, some temporal approaches (AR, ARMA, and ARIMA) and ensemble methods (extra trees, gradient boosting, and random forest) have longer prediction time. The highest overhead predictor is ARMA, which takes 6031.52 seconds for 10K jobs.

B. Performance of Different Styles of Predictive Scaling

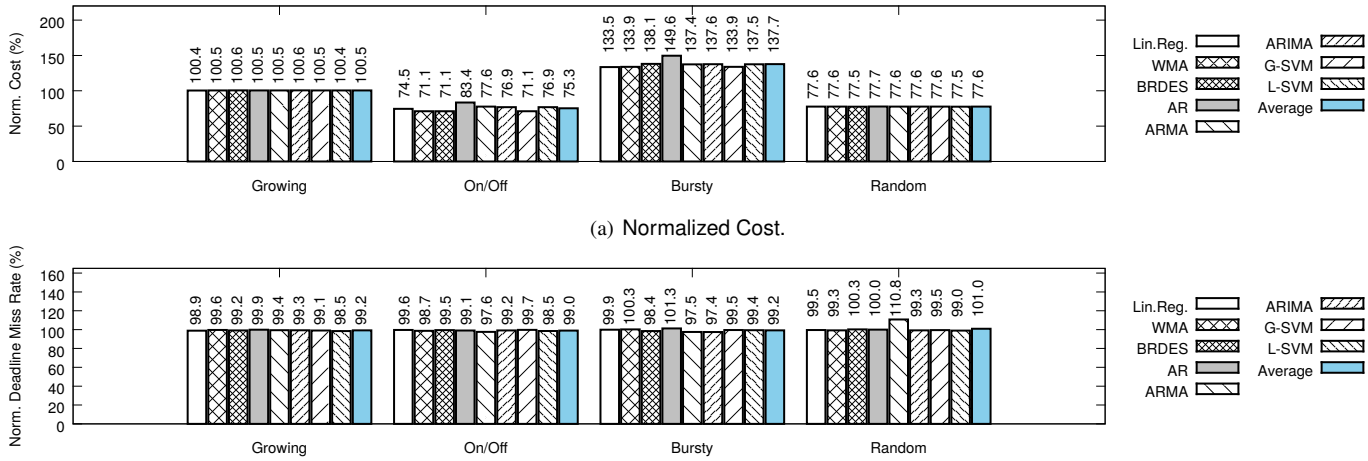
We measure the performance of four different styles of scaling operations for cloud resource management:

- **Baseline**: **RR** (Scale-Out: **R**eactive + Scale-In: **R**eactive)
- **PR** (Scale-Out: **P**redictive + Scale-In: **R**eactive)
- **RP** (Scale-Out: **R**eactive + Scale-In: **P**redictive)
- **PP** (Scale-Out: **P**redictive + Scale-In: **P**redictive)

PR is the most common style of predictive scaling for cloud application. **PR** predictively scales out cloud resources and reactively scales in cloud resources. **RP** is another way of predictive scaling, and it uses a reactive way for scaling-out and a predictive approach for scaling-in. **PP** is a combination of **PR** and **RP**, and it leverages a workload predictor for both scaling-out and scaling-in operations. For this evaluation, we use **RR** (Scaling-Out: **R**eactive + Scaling-In: **R**eactive) as a baseline. **RR** adds a new VM (scaling-out) when a new job needs an extra VM, and terminates an idle VM (scaling-in) at its billing boundary. (i.e., hourly bound or minutely bound).

Due to page limitation, we only show the results of the predictive scaling operations with the most accurate 8 workload predictors from Section IV-A. These predictors are Linear-SVM, Gaussian-SVM, ARMA, AR, WMA, ARIMA, Brown's DES, Linear regression, which cover the overall best 5 predictors and the best 3 predictors for each workload pattern.

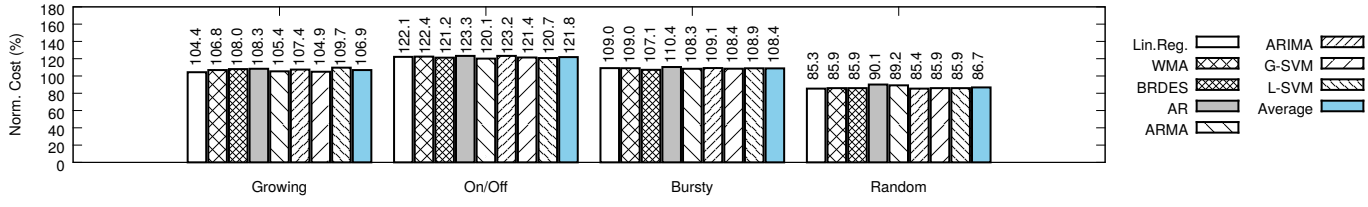
To evaluate the predictive scaling operations, we use two common cloud metrics (cost and job deadline miss rate) and two different billing models (hourly and minutely pricing model). Cost is to evaluate each scaling operations' cost efficiency, and job deadline miss rate represents the SLA-



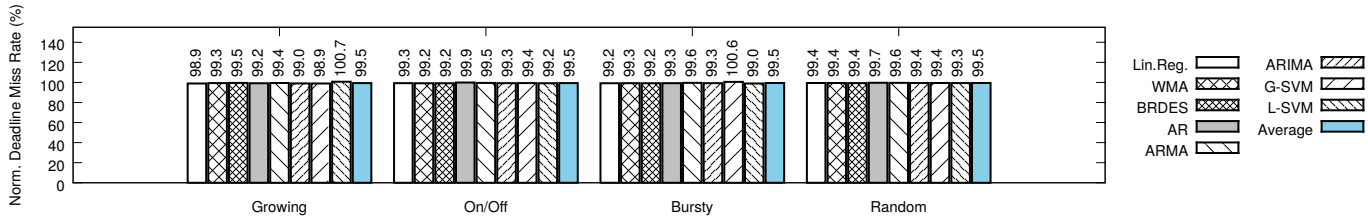
(a) Normalized Cost.

(b) Normalized Job Deadline Miss Rate.

Fig. 7: **Case #2** – Normalized Cost and Job Deadline Miss Rate of **RP** (Scaling-Out:**R**eactive + Scaling-In:**P**redictive) – Hourly Pricing Model.



(a) Normalized Cost.



(b) Normalized Job Deadline Miss Rate.

Fig. 8: **Case #2** – Normalized Cost and Job Deadline Miss Rate of **RP** (Scaling-Out:**R**eactive + Scaling-In:**P**redictive) – Minutely Pricing Model.

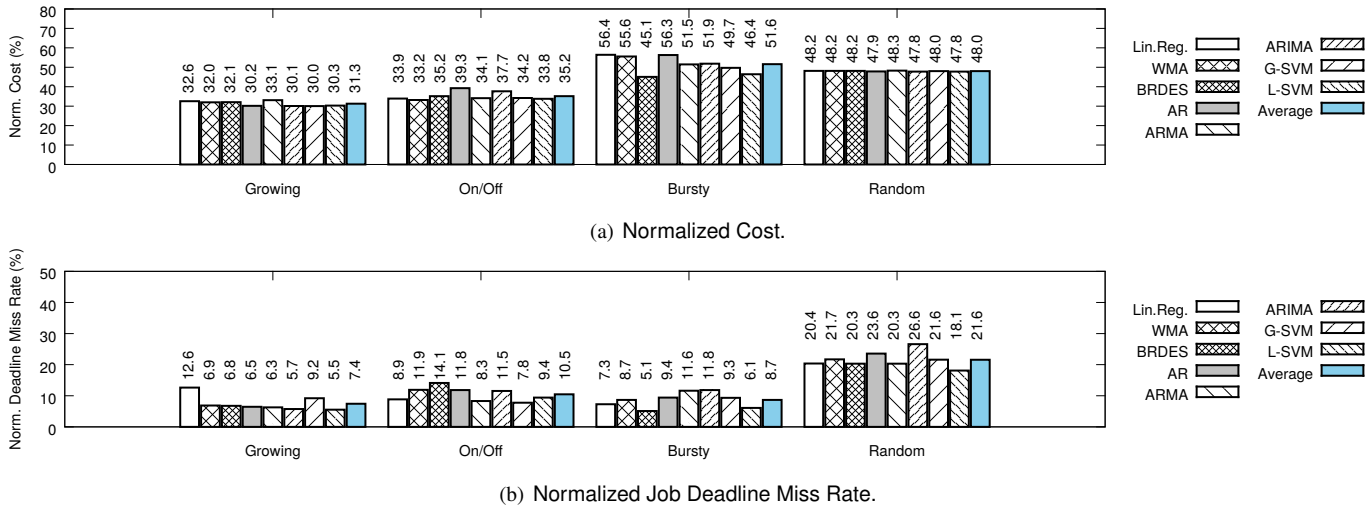
satisfaction requirement. We use two different billing models for cloud infrastructure, because major commercial IaaS clouds employ either hourly (e.g. AWS) or minutely (e.g. MS Azure) pricing model. We also use four different workload patterns (growing, on/off, bursty, and random workload).

The goals of this evaluation are:

- Measuring the actual benefits from predictive scaling.
- Determining the best style of predictive scaling.
- Finding the best workload predictor for each workload pattern in terms of cloud metrics.

Case #1 – PR (Scale-Out: **P**redictive + Scale-In: **R**eactive): Figure 5 shows a normalized cost and job deadline miss rate (all results are normalized to **RR**) of **PR** for hourly pricing model. The results show that **PR** can improve 47%–58% of cost efficiency for growing, on/off, and bursty workloads. However, for random workload, **PR** has 11% of worse cost efficiency over the baseline. In terms of job deadline miss rate, **PR** has 50%–88% of less job deadline miss rate over the **RR** (baseline). For the hourly pricing model, **PR** shows relatively poor performance for random workload in both cost efficiency and job deadline miss rate. This is because random workload is harder to predict than the other workload patterns. We rank the workload predictors for the **PR** based on the deadline miss rate, because it is more important to ensure that jobs meet their deadlines. Only after the deadline requirements are met, our cloud resource manager in Section III-A optimizes for cost efficiency. The best workload predictors for **PR** are: **Linear-SVM** (13.7%) for growing, **Gaussian-SVM** (8.2%) for on/off, **ARIMA** (15.2%) for bursty, and **Gaussian-SVM** (47.3%) for random workload.

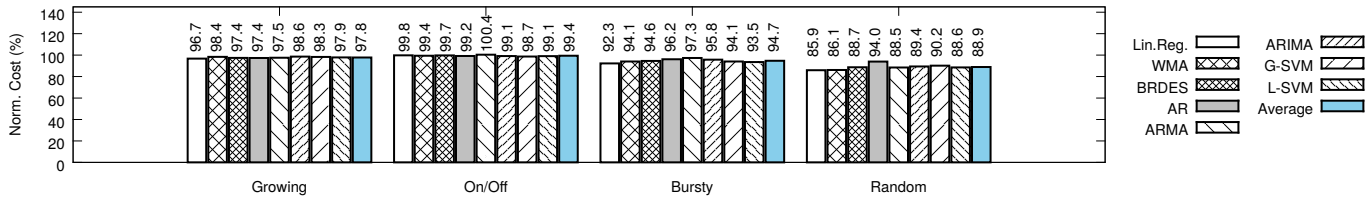
Figure 6 shows evaluation results of **PR** for minutely pricing model under four workload patterns. **PR** has similar cost efficiency with **RR**, but it has 41%–87% of less job deadline miss rate than the baseline. Thus, **PR** provides better job deadline satisfaction without dramatically increasing cost. The reason that **PR** has similar cost efficiency with **RR** is that the minutely pricing model is designed to provide better cost efficiency than hourly model to the user. So it is very hard to improve the cost efficiency for the minutely pricing model even though we have a good predictor. The best workload predictors for **RP** with



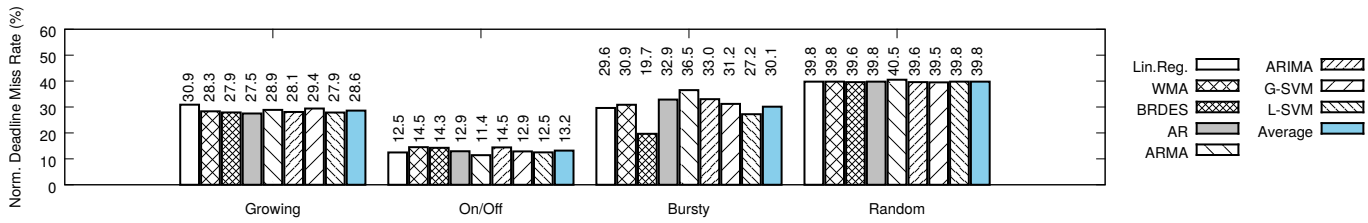
(a) Normalized Cost.

(b) Normalized Job Deadline Miss Rate.

Fig. 9: **Case #3** – Normalized Cost and Job Deadline Miss Rate of **PP** (Scaling-Out:**P**redictive + Scaling-In:**P**redictive) – Hourly Pricing Model.



(a) Normalized Cost.



(b) Normalized Job Deadline Miss Rate.

Fig. 10: **Case #3** – Normalized Cost and Job Deadline Miss Rate of **PP** (Scaling-Out:**P**redictive + Scaling-In:**P**redictive) – Minutely Pricing Model.

minutely pricing model are: **AR** (41.3%) for growing, **ARMA** (11.4%) for on/off, **WMA** (42.6%) for bursty, and **Gaussian-SVM** (58.6%) for random workload.

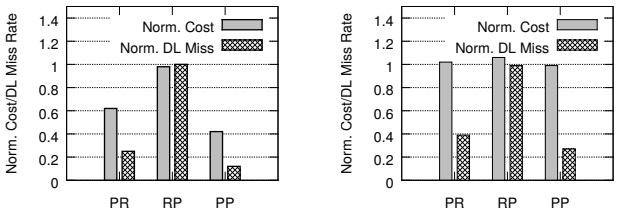
Case #2 – RP (Scale-Out: **R**eactive + Scale-In: **P**redictive): Figure 7 and 8 show the evaluation results of **RP** for both pricing models under four workload patterns. The results indicate that **RP**'s benefit to the cloud system is not as much as the benefits from **PR**. The only benefit from the **RP** is the improved cost efficiency (on/off and random workloads for hourly pricing model, random workload for minutely pricing model). The improvement of cost efficiency is 12%–25%, but it has no benefits for job deadline miss rate.

Case #3 – PP (Scale-Out: **P**redictive + Scale-In: **P**redictive): Figure 9 shows normalized costs and job deadline miss rates of **PP** for hourly pricing model under four workload patterns. The results show that **PP** improves 48%–69% of cost efficiency for the four workloads over **RR**, and has 78%–93% of less job deadline miss rate than **RR**. The best workload predictors for the **PP** with hourly pricing model are: **Linear-SVM** (5.5%) for growing, **Gaussian-SVM** (7.8%) for on/off, **Brown's DES** (5.1%) for bursty, and **Linear-SVM** (18.1%) for random workload.

Figure 10 shows evaluation results of **PP** for minutely pricing model under four workload patterns. The results show that **PP** slightly improves cost efficiency ($\leq$ 11%) of the cloud system and has huge improvement (60%–87%) for the job deadline miss rate. The best workload predictors of **PP** with minutely pricing model for each workload pattern are: **AR** (27.5%) for growing, **ARMA** (11.4%) for on/off, **Brown's DES** (19.7%) for bursty, and **Gaussian-SVM** (39.5%) for random workload.

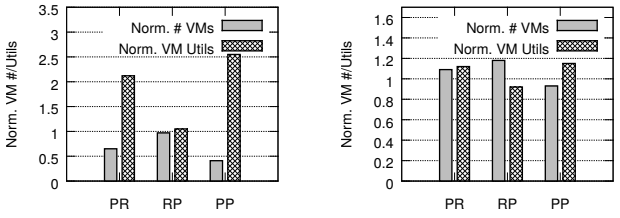
Comparison of Three Predictive Scaling Styles: So far we have separately evaluated three predictive scaling styles of the cloud resource management. In the following paragraphs, we present the overall benefit of predictive scaling. And we compare the results of these three predictive scaling styles to determine the best one for cloud resource management. Figure 11 shows that the comparison of average results of normalized cost and job deadline miss rate for **PR**, **RP**, and **PP** in both hourly and minutely pricing models.

For the hourly pricing model (Figure 11(a)), we found that **PP** is the best style for cloud resource management in terms



(a) Hourly Pricing Model (b) Minutely Pricing Model

Fig. 11: Comparison of Average Results of Normalized Cost and Job Deadline Miss Rate of Three Scaling Styles (**PR**, **RP**, and **PP**).



(a) Hourly Pricing Model (b) Minutely Pricing Model

Fig. 12: Comparison of VM Numbers/Utilization of Three Scaling Styles.

of better cost efficiency and less job deadline miss rate. **PP**'s cost efficiency is 20% (compared to **PR**) and 56% (compared to **RP**) better than other two approaches. Moreover, **PP**'s job deadline miss rate is 13% (compared to **PR**) and 88% (compared to **RP**) lower than others. This result is interesting, because although predictive scaling-in does not improve cost and deadline miss rate by itself (as shown in Case #2), it provides considerable improvement for both metrics when combined with predictive scaling-out. These results show that predictive scaling in/out approach (**PP**) (with a good workload predictor) helps to improve the performance of the cloud resource management.

For the minutely pricing model (Figure 11(b)), the job deadline miss rate of **PP** outperforms other two styles of predictive scaling operations. **PP** has 12% (compared to **PR**) and 72% (compared to **RP**) of less job deadline miss rate. **PP** also improves cost efficiency over **PR** and **RP**. These results suggest **PP** can significantly reduce deadline miss rate without cost overhead.

To understand the reasons of 1) **PP** significantly improves cost efficiency (hourly pricing model) and deadline miss rate (both pricing model) and 2) **RP** does not improve the performance by itself, we analyze the number of created VMs and VM utilization of three styles. Figure 12 represents the VM numbers and utilization of three scaling styles for both pricing models. For the both pricing models, **PP** creates the less number of VMs and has higher utilization than others. The reasons that **PP** has high VM utilization and lower number of created VMs are:

- Predictive scaling-out of **PP** uses more currently running VMs for the (near) future jobs, and creates less VMs for the (near) future jobs.
- Predictive scaling-in of **PP** keeps VMs alive for (further) future jobs, which further reduces the new VM creations, and increases the utilizations of existing VMs.

Moreover, the reason that **RP** cannot improve the cloud metrics is related to reactive scaling-out of **RP**. Reactive scaling-out operation creates VMs when jobs actually arrive, so **RP** has to create better performance VMs (more expensive VMs) in order to meet the jobs' deadline. This is because **RP** has no advance preparation for eliminating the overhead of the VM creation (e.g. startup delay). Also most of VMs should be terminated after processing the current job because they are not used for future jobs. So, predictive scaling-in of **RP** does not help in this case because most of VMs should be destroyed.

V. Related Work

A large body of work has been conducted for the predictive cloud resource management for dynamic workload patterns. There are two major branches in predictive resource management in the clouds. First branch is to focus on predicting the future resource usages (e.g. CPU, memory, and I/O) based on past resource usage history [1–4, 28]. Second branch is a workload predictive approach for cloud resource management. More specifically, this branch focuses on predicting the future job arrival time by using various workload predictors. Our work is closer to the second branch because we have evaluated prediction techniques for the future job arrivals to the clouds. In order to predict the next job arrival time, previous works employ a variety of predictors. They applied regression [16, 17, 29–31], time-series methods (e.g. ES [5–7, 18, 19], AR [8, 24], ARMA [9–13] and ARIMA [14, 15]) to their cloud systems to effectively manage the cloud resources. Herbst et al [32] proposed WCF (Workload Classification and Forecasting) framework, which is a self-adaptive approach for proactive resource management. However, the purpose of our work is different from the goal of previous works (proposing a new predictive scaling mechanism). We are more geared toward evaluating the performance of all existing prediction techniques for cloud resource management (e.g. scaling) by employing various/realistic workload patterns [20, 21], diverse cloud configurations (e.g. billing model), and common performance metrics. Furthermore, we have concentrated on the evaluation of different styles of predictive scale operations: predictive scaling-out only, predictive scaling-in only, and both predictive scaling-in/out. Therefore, our goal is to determine the best workload predictor for the cloud users' specific workload pattern and cloud configurations as well as the best scaling style of the predictive scaling.

VI. Conclusion

In order to help the cloud users select the best workload predictor, we comprehensively evaluated 21 workload predictors using statistical and cloud metrics. We then evaluated the prediction accuracy of all workload predictors, and cost/deadline miss rate of three following styles of predictive scaling under diverse workload patterns and different cloud configurations.

- **PR** (Scale-Out: **P**redictive + Scale-In: **R**eactive)
- **RP** (Scale-Out: **R**eactive + Scale-In: **P**redictive)
- **PP** (Scale-Out: **P**redictive + Scale-In: **P**redictive)

We found that to design a new predictive cloud resource scaling, the cloud users should consider top 3 workload predictors depending on their workload patterns, and use **PP** approach for their scaling operations in order to maximize the scaling performance.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of this paper.

REFERENCES

[1] Peter A. Dinda and David R. O'Hallaron. Host Load Prediction using Linear Models. *Cluster Computing*, 3(4):265–280, 2000.

[2] Zhenhuan Gong, Xiaohui Gu, and John Wilkes. PRESS: PRedictive Elastic ReSource Scaling for cloud systems. In *International Conference on Network and Service Management (CNSM)*, 2010.

[3] Zhen Xiao, Weijia Song, and Qi Chen. Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment. *IEEE Transactions on Parallel and Distributed Systems*, 24(6), 2013.

[4] Akindele A. Bankole and Samuel A. Ajila. Cloud Client Prediction Models for Cloud Resource Provisioning in a Multitier Web Application Environment. In *IEEE International Symposium on Service Oriented System Engineering (SOSE)*, 2013.

[5] Shuangcheng Niu, Jidong Zhai, Xiaosong Ma, Xiongchao Tang, and Wenguang Chen. Cost-effective Cloud HPC Resource Provisioning by Building Semi-Elastic Virtual Clusters. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2013.

[6] Ching Chuen Teck Mark, Dusit Niyato, and Tham Chen-Khong. Evolutionary Optimal Virtual Machine Placement and Demand Forecaster for Cloud Computing. In *IEEE International Conference on Advanced Information Networking and Applications (AINA)*, 2011.

[7] Haibo Mi, Huaimin Wang, Gang Yin, Yangfan Zhou, Dianxi Shi, and Lin Yuan. Online Self-reconfiguration with Performance Guarantee for Energy-efficient Large-scale Cloud Comp. Data Centers. In *IEEE International Conference on Services Computing (SCC)*, 2010.

[8] Norman Bobroff, Andrzej Kochut, and Kirk Beaty. Dynamic Placement of Virtual Machines for Managing SLA Violations. In *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2007.

[9] Nilabja Roy, Abhishek Dubey, and Aniruddha Gokhale. Efficient Autoscaling in the Cloud using Predictive Models for Workload Forecasting. In *IEEE International Conference on Cloud Computing (CLOUD)*, 2011.

[10] Juan M. Tirado, Daniel Higuero, Florin Isaila, and Jesus Carretero. Predictive Data Grouping and Placement for Cloud-based Elastic Server Infrastructures. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2011.

[11] Mohit Dhingra, J. Lakshmi, S. K. Nandy, Chiranjib Bhattacharyya, and K. Gopinath. Elastic Resources Framework in IaaS, preserving performance SLAs. In *IEEE International Conference on Cloud Computing (CLOUD)*, 2013.

[12] Upendra Sharma, Prashant Shenoy, and Sambit Sahu. A Flexible Elastic Control Plane for Private Clouds. In *ACM Cloud and Autonomic Computing Conference (CAC)*, 2013.

[13] Shun-Pun Li and Man-Hon Wong. Data Allocation in Scalable Distributed Database Systems Based on Time Series Forecasting. In *IEEE International Congress on Big Data (BigData Congress)*, 2013.

[14] Rodrigo N. Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyy. Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications' QoS. *IEEE Transactions on Cloud Computing*, 3(4), 2015.

[15] Hong Xu Di Niu, Baochun Li, and Shuqiao Zhao. Quality-Assured Cloud Bandwidth Auto-Scaling for Video-on-Demand Applications. In *IEEE International Conference on Computer Communications (INFOCOM)*, 2012.

[16] Peter Bodik, Rean Griffith, Charles Sutton, Armando Fox, Michael Jordan, and David Patterson. Statistical Machine Learning Makes Automatic Control Practical for Internet Datacenters. In *USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2009.

[17] Jingqi Yang, Chuanchang Liu, Yanlei Shang, Bo Cheng, Zexiang Mao, Chunhong Liu, Lisha Niu, and Junliang Chen. A Cost-aware Autoscaling Approach Using the Workload Prediction in Service Clouds. *Information Systems Frontiers*, 16(1), 2014.

[18] Sou Koyano, Shingo Ata, Ikuo Oka, and Kazunari Inoue. A High-grained Traffic Prediction for Microseconds Power Control in Energy-aware Routers. In *IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, 2012.

[19] Prasad Saripalli, GVR Kiran, Ravi Shankar R, Harish Narware, and Nitin Bindal. Load Prediction and Hot Spot Detection Models for Autonomic Cloud Computing. In *IEEE International Conference on Utility and Cloud Computing (UCC)*, 2011.

[20] Christoph Fehling et al. Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications. 2014.

[21] Workload Patterns for Cloud Computing. http://watdenkt.veenhof.nu/2010/07/13/workload-patterns-for-cloud-computing/.

[22] In Kee Kim, Wei Wang, and Marty Humphrey. PICS: A Public Iaas Cloud Simulator. In *IEEE International Conference on Cloud Computing (CLOUD)*, 2015.

[23] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The Element of Statistical Learning: Data Mining, Inference, and Prediction. 2011.

[24] Abhishek Chandra, Weibo Gong, and Prashant Shenoy. Dynamic Resource Allocation for Shared Data Centers Using Online Measurements. In *International Workshop on Quality of Service (IWQoS)*, 2003.

[25] In Kee Kim, Jacob Steele, Yanjun Qi, and Marty Humphrey. Comprehensive Elastic Resource Management to Ensure Predictable Performance for Scientific Applications on Public IaaS Clouds. In *IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, 2014.

[26] Ming Mao and Marty Humphrey. A Performance Study on the VM Startup Time in the Cloud. In *IEEE International Conference on Cloud Computing (CLOUD)*, 2012.

[27] Amazon Web Services. http://aws.amazon.com.

[28] Zhiming Shen, Sethuraman Subbiah, Xiaohui Gu, and John Wilkes. CloudScale: Elastic Resource Scaling for Multi-Tenant Cloud Systems. In *ACM Symposium on Cloud Computing (SoCC)*, 2011.

[29] Waheed Iqbal, Matthew N. Dailey, David Carrera, and Paul Janecek. Adaptive Resource Provisioning for Read Intensive Multi-tier Applications in the Cloud. *Future Generation Computer Systems*, 27(6), 2011.

[30] Sireesha Muppala, Xiaobo Zhou, and Liqiang Zhang. Regression Based Multi-tier Resource Provisioning for Session Slowdown Guarantees. In *IEEE International Performance Computing and Communications Conference (IPCCC)*, 2010.

[31] Sourav Dutta, Sankalp Gera, Akshat Verma, and Balaji Viswanathan. SmartScale: Automatic Application Scaling in Enterprise Clouds. In *IEEE International Conference on Cloud Computing (CLOUD)*, 2012.

[32] Nikolas Roman Herbst, Nikolaus Huber, Samuel Kounev, and Erich Amrehn. Self-Adaptive Workload Classification and Forecasting for Proactive Resource Provisioning. In *ACM/SPEC International Conference on Performance Engineering (ICPE)*, 2013.