CloudInsight: Utilizing a Council of Experts to Predict Future Cloud Application Workloads

In Kee Kim
Computer Science
University of Virginia
ik2sb@virginia.edu

Wei Wang
Computer Science
University of Texas at San Antonio
wei.wang@utsa.edu

Yanjun Qi
Computer Science
University of Virginia
yanjun@virginia.edu

Marty Humphrey
Computer Science
University of Virginia
humphrey@cs.virginia.edu

***Abstract*—Many predictive approaches have been proposed to overcome the limitations of reactive autoscaling on clouds. These approaches leverage workload predictors that are usually targeted for a particular workload pattern and can fail to handle real-world cloud workloads whose patterns may be unknown a priori, may dynamically change over time, or may be irregular. The result is that resources are frequently under- and over-provisioned. To address this problem, we create a novel cloud workload prediction framework called *CloudInsight*, leveraging the combined power of multiple workload predictors that collectively provide a "council of experts". The weights of the predictors in this ensemble model are determined in real-time based on their accuracy for current workload using multi-class regression. Under real workload traces, *CloudInsight* has 13% – 27% better accuracy than state-of-the-art predictors. It also has low overhead for predicting future workload changes (< 100 ms) and creating a new ensemble workload predictor (< 1.1 sec.).**

***Index Terms*—Cloud Computing; Workload Prediction, Predictive Resource Management, Machine Learning, Ensemble Model**

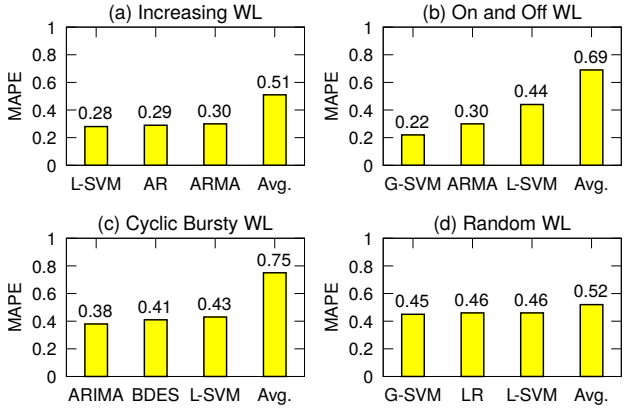


Fig. 1. The accuracy results of workload predictors under four different workload patterns. (L-SVM: Linear SVM, G-SVM: Gaussian SVM, BDES: Brown Double Exponential Smoothing, LR: Linear Regression)

I. Introduction

Autoscaling is a common approach for attempting to achieve elasticity in cloud applications [1–5]. However, autoscaling can often be sub-optimal because of its reactive nature. The reactive nature often results in over- and under-provisioning of cloud resources that causes low cost efficiency and high SLA (Service Level Agreement) violations. To overcome such limitations, many predictive approaches have been proposed [6–18]. The predictive approaches consist of two components; one is a workload predictor, which forecasts future job arrival time/rate; and the other is a resource management module, which allocates/deallocates cloud resources and maps user workloads to specific resources.

Existing predictive autoscaling managers often create and/or use a single static workload predictor with the simplifying assumption that their workload has a stable pattern (e.g., increasing, cyclic bursty, and on-and-off) over time. This predictor model is typically built offline and can sometimes require significant resources to build. Frequently, since cyclic bursty is known as a typical workload pattern for cloud applications [19], time-series based approaches are widely used as the static workload predictor to handle cyclic workloads [6, 11–17, 20–24].

We first investigated the degree to which a single existing predictor could be used across multiple typical cloud workload patterns, by evaluating the prediction accuracy of 21 widely used prediction algorithms: two naive, six regression, seven time-series, and six machine-learning models. We used four well-known/synthetic cloud workload patterns the community [25, 26]: increasing workload, on and off workload, cyclic bursty workload, and random workload. The performance of all predictors is measured by MAPE. Figure 1 shows the best three predictors and an average result from all predictors regarding the four different workloads. There is *no single best workload predictor* for all workload patterns – each workload pattern has its own best workload predictor. Moreover, the top three workload predictors for each workload pattern often show similar performance for the workload prediction, implying that best predictors could be changing if the workload contained more randomness or short-term burstiness [27]. It is also worth noting that in Figure 1, the best predictors usually contain non-time-series models, such as SVMs or linear regression, because of the lack of trend and seasonality in certain patterns.

Furthermore, the static approach is insufficient to address real-world cloud workloads because the patterns of real work-

loads are usually unknown a priori. Consequently, a new approach is required to improve the accuracy of workload prediction for real-world workloads that have a variety of workload patterns and dynamic fluctuations. To this end, we have created the *CloudInsight* framework, inspired by a "mixture-of-experts" problem [28]. Observing that different predictors excel at predicting different workload patterns, *CloudInsight* dynamically creates an ensemble model that combines multiple predictors to predict future job arrival rate/time. The weight of a predictor in the ensemble model is based on the predictor's accuracy for the current workload. To determine the weights, we design a novel evaluation approach based on a SVM (Support Vector Machine) multiclass regression model. The ensemble model is recreated periodically to handle the dynamic fluctuations in a workload.

We have conducted comprehensive evaluations of the performance (the prediction accuracy) of *CloudInsight* with diverse real-world workload traces collected from cluster [29, 30], HPC (High-Performance Computing) [31], and web applications [32]. The experiment results show that *CloudInsight* outperforms existing time-series, machine learning, and specific custom predictors in all evaluated workloads. *CloudInsight* has 13% to 27% better prediction accuracy than state-of-the-art approaches. While performing high prediction accuracy, *CloudInsight* shows low overhead for predicting future workload changes ($< 100\ ms$) and (re)creating a new ensemble model ($< 1.1\ sec.$).

II. Related Work

The models for existing predictive approaches commonly rely on regression, time-series, and machine learning [33, 34]. Among them, time-series approaches are the most popular approach. (ES [11, 20], AR [12, 21], ARMA [13, 14, 22], ARIMA [15, 17, 23] and others [16, 24].) However, as we stated in the introduction, such single predictor-based approaches are not sufficient to address the dynamics and burstiness of cloud workloads and can show poor performance for unknown workload patterns.

Several custom predictive approaches are developed to address dynamic cloud workload patterns. PRESS [10] and CloudScale [7] employ a custom predictor that consolidates FFT and Markov model. FFT is used to detect a signature of workload patterns, and Markov model is to address a short-term change of the workloads. However, in practice, it is challenging for cloud users to determine the transition probability of the Markov model correctly. Wood et al. [6] developed a hybrid approach, combining robust linear stepwise regression and the model refinement. This technique requires an offline profiling for the initial linear model creation, but *CloudInsight* is a purely online model that does not require such offline profiling step.

Multi-predictor approaches are also proposed. e.g., Khan et al. [35], Herbst et al. [36], and Liu et al. [37]. These approaches employ a classification and clustering (e.g., decision tree) of incoming workloads and statically allocate the best predictor for the particular type of workloads to increase the performance of workload prediction. However, for real workloads without clear seasonality and trend, it is hard to enumerate all possible classes a priori.

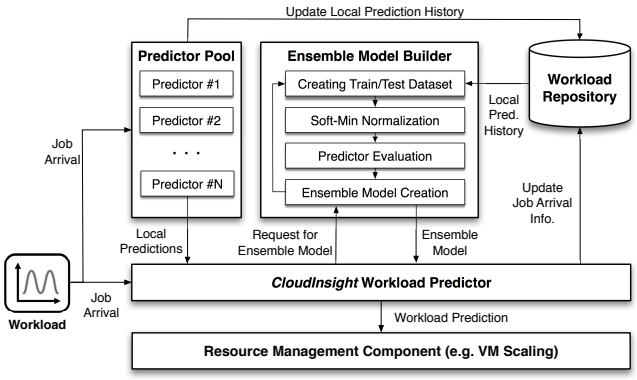


Fig. 2. Architecture of *CloudInsight*

ASAP [38] and Vadara [39] are two ensemble approaches with multiple workload predictors. These two approaches use a simple assumption to determine contributions from each individual predictor, i.e., recent the best predictors (e.g., the lowest cumulative error during the previous monitoring interval [18]) will perform the best for the near-future. However, we observe that this assumption is not always true. Especially, the workloads with short-term burstiness [27] can break this assumption. Unlike these approaches, we utilize a much longer history in *CloudInsight*, and employ multi-class regression model to predict the future accuracy of local predictors. Therefore, *CloudInsight* can provide more robust weights and more accurate predictions. We will show *CloudInsight*'s performance (prediction accuracy) in Section-IV.

III. CloudInsight Framework

The *CloudInsight* framework (Figure 2) consists of four main components: 1) a predictor pool, 2) a workload repository, 3) a model builder and 4) *CloudInsight* workload predictor. The input of this framework is the actual/current workloads (e.g., job arrivals) and the output is the prediction for a near-future workload. The predictor pool is a collection of workload predictors. The workload repository stores the job history of the workload and the prediction history of all local predictors in predictor pool. The model builder is responsible for creating an ensemble prediction model by evaluating the performance of the predictors in the predictor pool.

When jobs begin to arrive in a cloud application, a prediction for the future workload is also initiated. For the initial period (e.g., the first 30 minutes or 1 hour), *CloudInsight* either uses a simple ensemble workload prediction model that all local predictors have the equal contribution (weight) or relies on user's selection of the weights (the user can allocate a higher weight for a particular predictor). Once the initial (measurement) step finishes and initial accuracy history is collected, *CloudInsight* creates an ensemble prediction model based on the procedure described below. This ensemble model

TABLE I
LOCAL PREDICTORS IN THE PREDICTOR POOL OF *CloudInsight*

Category	Predictor	Description
Regression	Linear Regress.	This regression model forecasts the next job arrival rate by creating a linear function [40] using local history of the previous workload. In this work, the past job arrival rate is the only variable we consider, so this model is a single variable linear model. This model is also a local regression because we only use the limited amount of past workload history. To select the local samples, this model applied kNN as a kernel function [41]. Linear regression is included in the *predictor pool* because it can provide high accuracy for random workload pattern.
Time Series	WMA	Weighted Moving Average is a weighted sum of observed dataset (e.g., past workload information), and the sum of weight (ω) for each observed dataset is 1. WMA is calculated by $\sum_{n=1}^{k} \omega_n x_{t+1-n}$, $s.t. \sum_{n=1}^{k} \omega_n = 1$. We add this model since WMA shows good performance in overall.
	BDES	Brown's DES predicts the next job arrival rate 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 rate, and α is a smoothing factor ($0 < \alpha < 1$). BDES performs well for cyclic bursty workload pattern.
	AR	Autoregressive is a linear combination of previous data of the target object (e.g., job arrival rate). AR(p) model is expressed in $X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-1} + \varepsilon_t$, where 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. AR can provide high accuracy for increasing workload.
	ARMA	Autoregressive and Moving Average 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. ARMA shows good performance for increasing and on/off workload as well as provides high accuracy in overall.
	ARIMA	Autoregressive Integrated Moving Average is a generalization of ARMA and provides a reliable prediction of 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. ARIMA performs very well for the cyclic bursty workload that has strong trend and seasonality.
Machine Learning	Linear and Gaussian SVMs	Support Vector Machine is an optimal margin-based classifier [40] 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. 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 *Gaussian* SVMs. 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. Two SVM models work very well for both overall and specific workloads.

is used to predict future workload. After the expiration of the model re-creation interval, the ensemble model will be re-created. The workload prediction is performed at every pre-defined prediction interval with the ensemble model, which is created from the previous step. The ensemble model combines the predictions from the local predictors in the predictor pool. This prediction can then be used by a resource management component for resource scaling.

A. Predictor Pool

The predictor pool contains a variety of workload predictors, called "local predictors" – the predictor pool can contain any predictors as long as those predictors can provide predictions for future workloads (e.g., job arrival rates). We have experimented with various workload predictors, including time-series, regressions, and machine-learning models. In this work, *CloudInsight* leverages *eight* local predictors as described in Table I. As shown later, *CloudInsight* can properly handle all these types of predictors and considerably improve workload prediction accuracy. Altough the "eight" local predictors are selected for this work, users can add any workload predictors to the predictor pool because of *CloudInsight*'s generality. Note that, more local predictors may increase the overhead of workload prediction and ensemble model creation.

B. Workload Repository

Workload repository contains the prediction history of the all local predictors in the predictor pool. This history is represented as a normalized performance vector, which is described in the following paragraphs.

Performance Vector (*PV*): The PV is a fundamental element of training and prediction input datasets for the evaluation step and is a feature matrix composed of prediction errors of all local predictors for past prediction history. The performance vector is an $n \times m$ matrix as formulated below:

$$PV = \begin{bmatrix} PE_{1,1} & \cdots & PE_{1,m-1} & PE_{1,m} \\ PE_{2,1} & \cdots & PE_{2,m-1} & PE_{2,m} \\ \vdots & \ddots & \vdots & \vdots \\ PE_{n-1,1} & \cdots & PE_{n-1,m-1} & PE_{n-1,m} \\ PE_{n,1} & \cdots & PE_{n,m-1} & PE_{n,m} \end{bmatrix} \quad (1)$$

where n is the number of the local predictors and m is the past prediction points. $PE_{i,j}$, an element of PV matrix, is the prediction error of ith local predictor at jth prediction point. $PE_{i,j}$ is measured by squared error ($(Prediction_{i,j} - Actual_{i,j})^2$). A single PV represents a set of prediction errors of all n local predictors for past m prediction points. (In our evaluation, *CloudInsight* uses 50 for m of PV.

Soft-Min Normalization: An issue of PV is that each element (PE) of a PV matrix is the absolute squared error of each local predictor at certain prediction point. Since a PE represents the absolute prediction errors at a particular time interval, two PEs from two different time intervals cannot be directly compared to determine which is more accurate (i.e.,

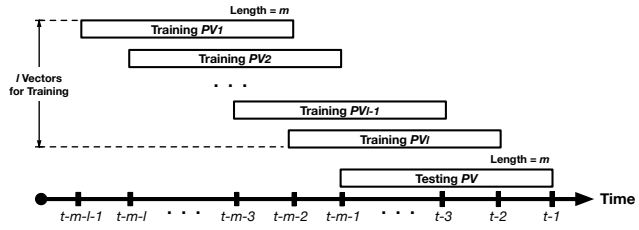


Fig. 3. Temporal coverage of PVs for training and prediction input dataset. (m: length of row for PV, meaning m temporal points for past predictions, l is the size of training dataset, meaning the number of PVs in training dataset.

which has smaller error). Therefore, we normalize PEs so that they can be directly compared. To normalize all PEs in a PV, we use a soft-min normalization function that transforms each element (PE) into a real number between 0 and 1. The soft-min function is as shown below:

$$SoftMin(PE_{i,j}) = 1 - \frac{e^{-PE_{i,j}}}{\sum_{k=1}^{n} e^{-PE_{k,j}}} \tag{2}$$

The input of Equation (2) is an element ($PE_{i,j}$) of PV. The numerator of the function is the exponentially inverse transform of the PE that we want to normalize. The dominator is the sum of exponentially inverse transforms for all PEs at a particular prediction point (a single column in the PV.)

This normalized value is subtracted from 1 so that higher values mean better performance (smaller prediction errors) of the local predictors. The upper bound of the normalized soft-min value is 1, while the lower bound is 0. A local predictor always has a soft-min value between 0 and 1. After this normalization, the sum of each column in a PV is 1. Intuitively, the soft-min value for a local predictor at particular prediction point can be viewed similarly as the probability of it being the best predictor for this prediction point.

C. Ensemble Predictor Builder

The model builder evaluates the local predictors, determines the best predictors among them, and creates an ensemble prediction model of top predictors with different weights. This model builder is inspired by a mixture-of-experts (MOE) problems [28]. Evaluating the local predictor is the most important step of the model builder to create an ensemble prediction model. We formulate this evaluation as a multiclass regression problem and use Gaussian SVM regression model [40].

Training Dataset and Prediction Inputs: Both training dataset and the prediction input dataset are represented as a collection of PVs. However, these two datasets use separable PVs that cover different temporal windows. Suppose time t indicates current prediction point, l is the size of training dataset, and m means the length of columns in PVs. The training dataset covers the history of the local predictors' performance between at $t-m-l-1$ and $t-2$. The training dataset is expressed as $\{PV_{t-m-l-1}, PV_{t-m-l}, ..., PV_{t-3}, PV_{t-2}\}$. The prediction input data-set, which is used to predict the job arrival rate at time t, is the PV at $t-1$ prediction point and is expressed as $\{PV_{t-1}\}$. Figure 3 illustrates the temporal coverage of training data set and prediction input data set.

Evaluation of Local Predictors: The "evaluating local predictors" problem is reduced to the "multiclass regression" problem. A multiclass regression problem gives the probabilities of whether an observation belongs to a set of categories. Consequently, with a "multiclass regression" model, we can evaluate the probability that a local predictor is the most accurate predictor for the future workload. More specifically, we employ Gaussian SVM model for this classification problem. The evaluation with the SVM model follows a typical machine learning process; training and prediction. The SVM model is trained with the aforementioned training dataset. After training, this model provides its projection for all local predictors. The output vector of this model is shown below.

$$Y^T = \left[\omega_1, \omega_2, \cdots, \omega_{n-1}, \omega_n\right] \tag{3}$$

The output of this SVM model is a $n \times 1$ matrix. Thanks to the soft-min normalization, all items in this output matrix are real numbers (ω) between 0 and 1. A higher value of ω_i (close to 1) indicates that ith predictor has higher possibility to be the best predictor for workload prediction.

Creating an Ensemble Model for Workload Prediction: The ensemble is constructed with Equation (4).

$$\text{Ensemble Model: } \frac{\sum_{i=1}^{n} \omega_i p_i}{\sum_{i=1}^{n} \omega_i} \tag{4}$$

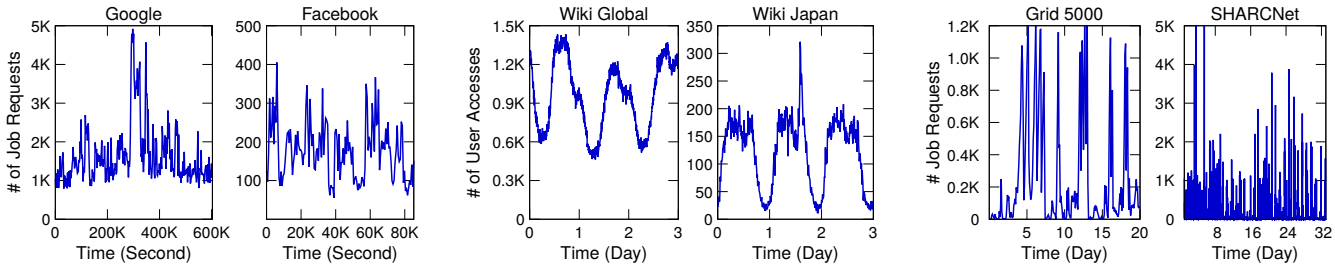
where ω_i is the output from the previous step and p_i is the prediction from a local predictor. The predictions of the local predictors used in this ensemble model will be updated to the workload repository to be used for further evaluation of the local predictors. The ensemble model is re-created periodically with a predefined time interval.

D. Implementation of CloudInsight

We implemented *CloudInsight* with Python 2.7 on Ubuntu 16.04 LTS. To implement the local predictors in the predictor pool and the evaluation step of the model builder, the following statistics and machine learning libraries are used; NumPy, Statsmodels, Pandas, and scikit-learn.

For implementing the local predictors, while our goal is to improve/maximize the prediction accuracy, deterministic processing time of the local predictors is desirable. This requirement is because *CloudInsight* collaborates with a resource manager that should adequately prepare cloud resources before the actual job arrives. We use a grid search [42] to determine the parameters for the local predictors with a tradeoff between the accuracy and the prediction overhead. We consider parameters of $0 < \alpha < 1$ for BDES, 1st to 3rd order for other time-series models (AR, ARMA, ARIMA) and $10e^{-3}$ to $10e^{3}$ for soft margin and kernel parameters in SVMs.

For the implementation of the ensemble predictor builder (the SVM multi-regression model), we aim more at improving the performance of an ensemble model. To this end, we also



(a) Cluster Workloads (b) Web Workloads (c) HPC Workload

Fig. 4. Example traces of cloud workloads used in this evaluation; (a) Cluster Workloads: Google cluster trace with 30 minutes of time-interval and Facebook Hadoop trace with 5 minutes of time-interval (b) Web Workloads: Two Wikipedia traces with 5 minutes of time-interval (c) HPC Workloads: Grid 500 with 3600 seconds of time-interval and SHARCNet with 900 seconds of time-interval.

take the same approach (grid search) with the way of tuning the local predictors, but we examine broader range for soft margin and kernel parameters of SVM model $10e^{-6}$ to $10e^{6}$ to yield better results.

We use various synthetic workloads [25, 26] to guide the above two parameter selection processes. To ensure fair evaluation and avoid over-fitting, we did not use real workloads in parameter selection. Real workloads [29–32] are only used to evaluate the performance of *CloudInsight*.

IV. Performance Evaluation

A. Evaluation Setup

Local Predictors: The predictor pool in *CloudInsight* has *eight* well-known workload predictors; linear regression [37, 43], WMA [16, 24], BDES [11], AR [12, 21], ARMA [13, 14, 22], ARIMA [15, 17, 23], and two SVMs (both linear and Gaussian models) [8]. These predictors are described in Table I in Section-III.

Evaluation Goals: Our goal is to evaluate following properties of *CloudInsight*. First, we measure the accuracy of *CloudInsight* regarding the forecasting future job arrival rate in above datasets. We then evaluate the overhead of *CloudInsight* since prediction within deterministic time is a prerequisite of any workload predictors.

Performance Metrics: To measure the prediction accuracy of job arrive rate, we employ RMSE (Root Mean Square Error)[1]. Lower RMSE means the better performance and vice versa.

To evaluate the overhead of *CloudInsight*, we define the processing overhead as the *time* for "job arrival rate prediction process" and "ensemble model recreation process." We measure the actual processing time on a Linux Server with 8 CPUs (AMD Opteron Processor 4386) and 16G RAM.

Baselines: We compare *CloudInsight* against four predictors; ARIMA, SVM, FFT (Fast Fourier Transform), and RSLR (Robust Stepwise Linear Regression). We choose ARIMA and SVM from the local predictors because they are widely used in many predictive approaches [8, 15, 17] as well as the two best "*static*" predictors that we have experimented with. We choose FFT [7, 10] and RSLR [6] from state-of-the-art approaches, which provide a robust and accurate prediction for cloud resource scaling.

[1] $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Predicted_i - Actual_i)^2}$

TABLE II
Statistics of evaluated workloads

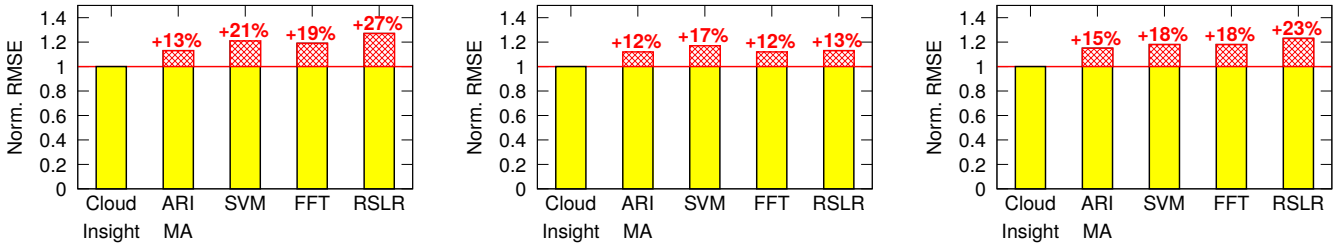
Worload	Trace	Duration	# Jobs	Predictor Setting	
				Predict. Interval	Model Recreat. Interval
Cluster [29, 30]	Google	1 mon.	2M	30 to 1200 sec.	At every 5 Preds.
	Facebook #1	1 day	5.9K		
	Facebook #2		6.6K		
	Facebook #3		24K		
	Facebook #4		25K		
Web [32, 44]	Wiki Glob.	3 days	823K	30 to 1200 sec.	At every 5 Preds.
	Wiki Germ.		76.5K		
	Wiki Japan.		51K		
HPC [31, 45]	Grid 5000	22 days	62.5K	30 to 1200 sec. or 1 to 12 hr. (AuverGrid)	At every 5 Preds.
	NorduGrid	60 days	122K		
	AuverGrid	365 days	2.3M		
	SHARCNet	11 days	188K		
	LCG	33 days	435K		

B. Workloads Used for Evaluation

To evaluation *CloudInsight*, we used three categories of workload traces from real-world cloud applications: 1) Cluster traces from Google [29] and Facebook [30]. 2) Wikipedia web traces from WikiBench [32, 44], and 3) Scientific/HPC workload traces from the Grid Workloads Archive [31, 45]. These three groups of workload datasets allow us to evaluate *CloudInsight* with diverse scenarios of cloud applications.

While the workload datasets contain various characteristics, this work focuses on its temporal characteristics. i.e., job arrival rate. We extract temporal behaviors of job submissions in the workloads. Example workload traces from three different application models are illustrated in Figure 4 and and Table II describes the summary of the characteristics of the workloads. We also choose workloads with variable length of duration (lifetime) and density of job arrivals to show the generality of *CloudInsight*. The following paragraphs outline the backgrounds of such workloads.

Cluster Workloads: Example traces of cluster workloads are shown in Figure 4(a). Google workload contains 2 millions of job arrival data for a 1-month period. Facebook dataset contains 1 millions of job submissions. Particularly for the



(a) Cluster Workload (b) Web Workload (c) HPC Workload

Fig. 5. Normalized RMSE results in cluster workloads (1.00 means the result from CloudInsight and higher values indicate worse performance); (a) Normalized RMSE of cluster workloads, (b) Normalized RMSE of web workloads, (c) Normalized RMSE of HPC workloads

Facebook workload, we use 4 sample traces from 2009 and 2010, each represents 1-day job submissions.

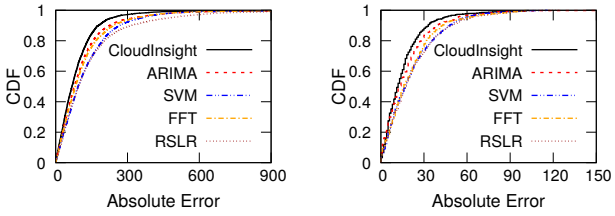
Web Workloads: We use three days of Wikipedia traces in September 2007. We focus on access log for (global) Wikipedia pages, German, and Japanese main page of Wikipedia. The datasets have 823K (global), 76.5K (German), and 51K (Japanese) of user accesses. Examples of Wikipedia workloads are illustrated in Figure 4(b). Wikipedia workloads generally show strong seasonality and trend characteristics, but Japanese main page has an unexpected spike of user access.

HPC Workloads: Five workloads are used for HPC scenarios on the clouds. i.e., Grid5000, NorduGrid, AuverGrid, SHARCNet, and LCG (LHC Computing Grid). These workloads respectively contain 62.5K jobs, 122K jobs, 2.3-million jobs, 188K jobs, and 435K jobs for various periods. Examples of HPC workloads are illustrated in Figure 4(c). HPC workloads have similar characteristics with two previous workloads, but have more dynamic natures.

C. Prediction Accuracy of CloudInsight

To compare the accuracy of *CloudInsight*, with the baselines, we use various time interval for the workload prediction as shown in Table II. This prediction interval may affect its prediction accuracy because a longer prediction interval may provide a smoothing effect on the workload patterns. Evaluation with the various time interval minimizes this impact and averaging them offsets the variation. In general, we use the prediction intervals from 30 seconds to 1200 seconds with a step of 30 seconds. Especially for AuverGrid with one year period, we use the prediction intervals with a range from 3600 seconds (1 hour) to 86400 (24 hours) with a step of 3600 seconds. For the model re-creation interval, we recreate and update the SVM model for evaluating the local predictors at every five predictions of future workloads.

Figure 5(a) shows the RMSE results of job arrival rate predictions of the five approaches with cluster workloads. (all results are normalized to *CloudInsight*). On average, *CloudInsight* is 13% – 27% more accurate than the four baselines. Because the cluster workloads do not have a stable seasonality and a trend, it is difficult for a single model (ARIMA, SVM, FFT, or RSLR) to accurately detect certain patterns from the cluster workloads to predict future changes.



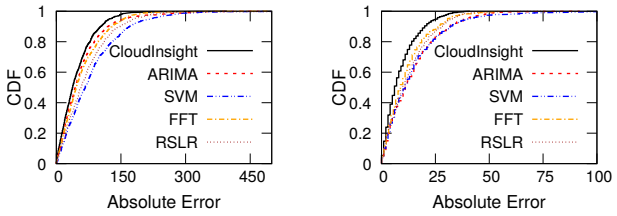
(a) Google WL (b) Facebook #3 WL

Fig. 6. CDF of prediction errors in two cluster workloads

However, *CloudInsight* can keep adjusting the weights for each predictor and create new ensemble model periodically to fit the changes in the workload. Therefore, *CloudInsight* can show better performance for workload prediction.

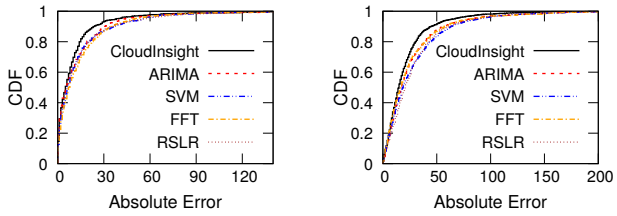
Figure 6 shows the CDF (Cumulative Distribution Function) of prediction errors in two cluster workloads. Note that due to page limitation, we only show the CDF results with Google and Facebook #4 workloads. The x-axis represents absolute prediction error, i.e., $|Prediction_t - Actual_t|$, while the y-axis gives the cumulated probability of the errors. As shown in the Figure, the curves for *CloudInsight* are skewed to the left than the baselines, meaning the majority of *CloudInsight*'s prediction errors are smaller than the baselines. Also, the results from the baselines have longer tails, indicating they yield more extreme prediction errors.

The averge RMSE results for web workloads are shown in Figure 5(b) and *CloudInsight* outperforms the baselines again. On average, *CloudInsight* has 12% – 17% of fewer errors than the baselines. Because web workloads usually have strong seasonality and trends, all baselines perform better than when predicting cluster workloads. Especially, both FFT and LRSR have a significant improvement in their accuracy. However, although web workloads have relatively stable seasonality and trends, the seasonality and trends can still change over time, albeit less abruptly. Also, as shown in Wiki Japanese workload, web workload could have a sudden spike of user accesses. *CloudInsight* can identify the seasonality and trends as well as detect the changes (or spikes) in them. Thus, it can provide better prediction results. Figure 7 illustrates the CDF of prediction errors in two web workloads (Wikipedia global main page and Wikipedia Japanese page). Similarly, the



(a) Wikipedia Global Main WL (b) Wikipedia Japanese WL

Fig. 7. CDF of prediction errors in two web workloads



(a) Grid5000 WL (b) AuverGrid WL

Fig. 8. CDF of prediction errors in two HPC workloads

majority of *CloudInsight*'s errors are still smaller than those of the baselines.

Figure 5(c) shows the average (normalized) RMSE results in the five HPC workloads; Grid 5000, NorduGrid, AuverGrid, LCG, and SHARCnet. On average, *CloudInsight* is 19% more accurate than the four baselines with all the five HPC workloads. Our HPC workloads exhibit a broad range of characteristics. The Grid 5000, AuverGrid, and SHARCNet workloads are bursty and random. They also lack seasonality and trends. The NorduGrid and LCG workloads have relatively clearer seasonality (among HPC workloads), although it is much more bursty and noisier than web workloads. The measurement results with various HPC workloads indicate that *CloudInsight* can correctly assign weights to the local predictors based on current workloads behaviors so that it can predict with the best predictors for the future workload. Figure 8 shows the CDF distribution of the prediction errors in the two HPC workloads. The results are similar with the two previous cluster and web workload types. The curves for *CloudInsight* are more skewed to the left, indicating that *CloudInsight* has less errors than other baselines.

D. Overhead of CloudInsight

We evaluate the overhead of *CloudInsight* and the four baselines. For the baselines, we only consider "prediction overhead" for this evaluation as there is no additional overhead due to the ensemble model reconstruction for them. We define "prediction overhead" as the time that it takes to make predictions at given time point. For *CloudInsight*, we measure both "prediction overhead" and "modeling overhead." We define the modeling overhead as the time that *CloudInsight* takes to create a new ensemble prediction model.

Prediction Overhead: On average (Table III), *CloudInsight* takes $34ms$ to make a prediction, while other methods show lower prediction overhead. i.e., ARIMA takes $26ms$, RSLR takes $21ms$, FFT takes, $6.3ms$ and SVM takes $0.38ms$. Although SVM has the lowest overhead among the approaches, it has less accuracy than *CloudInsight* and the others for the majority of our workloads. Even though *CloudInsight* leverages eight local predictors, it takes only $12ms$ more time as compared to ARIMA. The reason is that these eight predictors compute the prediction in parallel. The prediction overhead of *CloudInsight* is determined by the highest prediction time of its eight local predictors. It is worth noting

TABLE III
PREDICTION OVERHEAD OF FIVE APPROACHES

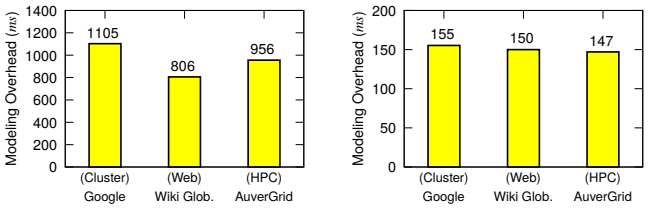
Predictor	Cluster WL	Web WL	HPC WL	Avg.
CloudInsight	29 ms	37 ms	36 ms	**34** ms
ARIMA	25 ms	24 ms	29 ms	**26** ms
SVM	0.35 ms	0.4 ms	0.4 ms	**0.4** ms
FFT	4.2 ms	5.9 ms	8.8 ms	**6.3** ms
RSLR	22 ms	18 ms	22 ms	**21** ms

that although *CloudInsight* has the longest prediction time among five predictors, the absolute prediction time ($34ms$) is still *negligible* compared to workload prediction intervals and resource reconfiguration intervals. Because the overhead from cloud infrastructure is usually higher than 30 seconds (e.g., VM startup time), autoscaling resource managers often reconfigure their resources at an interval higher than 30 seconds. As *CloudInsight*'s prediction time is much smaller than the prediction interval, it imposes very limited impact to an autoscaling resource manager.

Modeling Overhead: The results of this overhead are shown in Figure 9. Limited by space, we only show the results from the largest workload for each type of workloads: Google, Wiki Global Main, and AuverGrid. This overhead for the other workloads is usually smaller than these three workloads. In the worst cases, it takes 0.8-1.1 seconds to create a new ensemble model. The average modeling time is less than $155ms$. This overhead is still negligible in practice because *CloudInsight* can create a new ensemble model and make predictions within the autoscaling resource reconfiguration intervals as stated previously.

V. CONCLUSION

This paper presents *CloudInsight– an online workload prediction framework to address dynamic and highly variable cloud workloads*. *CloudInsight* employs a number of local predictors and creates an ensemble prediction model with them by dynamically determining the proper weights of each local predictor. To determine the weights, we formulate this problem as a multi-class regression problem with a SVM classifier. We have performed a comprehensive study to measure the performance and overhead of this framework with a broad range of real-world cloud workloads. Our evaluation results show that *CloudInsight* has 13% – 27% of better accuracy than state-of-the-art static predictors and it also has low overhead



(a) Max Overhead (b) Average Overhead

Fig. 9. Modeling overhead of *CloudInsight*

for predicting future workload changes ($< 100\ ms$) and (re)creating a new ensemble model ($< 1.1\ sec.$).

ACKNOWLEDGMENT

This work was partially supported by the National Science Foundation under grant CCF-1617390. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied of NSF.

REFERENCES

[1] Luwei Cheng, Jia Rao, and Francis C.M. Lau. vScale: Automatic and Efficient Processor Scaling for SMP Virtual Machines. In *EuroSys '16*.
[2] Tayler H. Hetherington, Mike O'Connor, and Tor M. Aamodt. MemcachedGPU: Scaling-up Scale-out Key-value Stores. In *SoCC '15*.
[3] Bailu Ding, Lucja Kot, Alan Demers, and Johannes Gehrke. Centiman: Elastic, High Performance Optimistic Concurrency Control by Watermarking. In *SoCC '15*.
[4] Thomas Heinze, Lars Roediger, Andreas Meister, Yuanzhen Ji, Zbigniew Jerzak, and Christof Fetzer. Online Parameter Optimization for Elastic Data Stream Processing. In *SoCC '15*.
[5] Ganesh Ananthanarayanan, Christopher Douglas, Raghu Ramakrishnan, Sriram Rao, and Ion Stoica. True Elasticity in Multi-Tenant Data-Intensive Compute Clusters. In *SoCC '12*.
[6] Timothy Wood, Ludmila Cherkasova, Kivanc M. Ozonat, and Prashant J. Shenoy. Profiling and Modeling Resource Usage of Virtualized Applications. In *Middleware '08*.
[7] Zhiming Shen, Sethuraman Subbiah, Xiaohui Gu, and John Wilkes. CloudScale: Elastic Resource Scaling for Multi-Tenant Cloud Systems. In *SoCC '11*.
[8] Neeraja Yadwadkar, Ganesh Ananthanarayanan, and Randy Katz. Wrangler: Predictable and Faster Jobs using Fewer Resources. In *SoCC '14*.
[9] Daniel Jacobson, Danny Yuan, and Neeraj Joshi. Scryer: Netflix's predictive auto scaling engine. The Netflix Tech Blog, 2013. http://techblog.netflix.com/2013/11/scryer-netflixs-predictive-auto-scaling.html.
[10] Zhenhuan Gong, Xiaohui Gu, and John Wilkes. PRESS: PRedictive Elastic ReSource Scaling for cloud systems. In *CNSM '10*.
[11] Shuangcheng Niu, Jidong Zhai, Xiaosong Ma, Xiongchao Tang, and Wenguang Chen. Cost-effective Cloud HPC Resource Provisioning by Building Semi-Elastic Virtual Clusters. In *SC '13*.
[12] Gong Chen, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, and Feng Zhao. Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services. In *NSDI '08*.
[13] Pradeep Padala, Kai-Yuan Hou, Kang G. Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, and Arif Merchant. Automated Control of Multiple Virtualized Resources. In *Eurosys '09*.
[14] Nilabja Roy, Abhishek Dubey, and Aniruddha Gokhale. Efficient Autoscaling in the Cloud using Predictive Models for Workload Forecasting. In *CLOUD '11*.
[15] Rodrigo N. Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications' QoS. *IEEE Trans. on Cloud Computing*, 4, 2015.
[16] Anshul Gandhi, Mor Harchol-Balter, Ram Raghunathan, and Michael A. Kozuch. AutoScale: Dynamic, Robust Capacity Management for Multi-Tier Data Centers. *ACM Trans. on Computer Systems*, 30(4), 2012.
[17] 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 '12)*, Orlando, FL, USA, June 2012.
[18] Hector Fernandez, Guillaume Pierre, and Thilo Kielmann. Autoscaling Web Applications in Heterogeneous Cloud Infrastructures. In *IC2E '14*.
[19] Ahmed Ali-Eldin, Ali Rezaie, Amardeep Mehta, Stanislav Razroev, Sara Sjostedt de Luna, Oleg Seleznjev, Johan Tordsson, and Erik Elmroth. How will your workload look like in 6 years? Analyzing Wikimedia's workload. In *IC2E '14*.
[20] Eyal Zohar, Israel Cidon, and Osnat Mokryn. The Power of Prediction: Cloud Bandwidth and Cost Reduction. In *SIGCOMM '11*.
[21] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Black-box and Gray-box Strategies for Virtual Machine Migration. In *NSDI '07*.
[22] Wei Fang et al. RPPS: A Novel Resource Prediction and Provisioning Scheme in Cloud Data Center. In *SCC '12*.
[23] Qi Zhang, Mohamed Faten Zhani, Shuo Zhang, Quanyan Zhu, Raouf Boutaba, and Joseph L. Hellerstein. Dynamic Energy-Aware Capacity Provisioning for Cloud Computing Environments. In *ICAC '12*.
[24] Yang Peng, Kai Chen, Guohui Wang, Wei Bai, Zhiqiang Ma, and Lin Gu. HadoopWatch: A First Step Towards Comprehensive Traffic Forecasting in Cloud Computing. In *INFOCOM '14*.
[25] Workload patterns for cloud computing. http://watdenkt.veenhof.nu/2010/07/13/workloadpatterns-for-cloudcomputing, 2018. ONLINE.
[26] Christoph Fehling, Frank Leymann, Ralph Retter, Walter Schupeck, and Peter Arbittera. Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications. 2014.
[27] Sadeka Islam, Srikumar Venugopal, and Anna Liu. Evaluating the Impact of Fine-scale Burstiness on Cloud Elasticity. In *SoCC '15*.
[28] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, 1991.
[29] John Wilkes. More Google cluster data. Google research, 2011. http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.
[30] SWIMProjectUCB. Workloads repository. https://github.com/SWIMProjectUCB/SWIM/wiki/Workloads-repository, 2018. ONLINE.
[31] Alexandru Iosup, Hui Li, Mathieu Jan, Shanny Anoep, Catalin Dumitrescu, Lex Wolters, and Dick H.J. Epema. The Grid Workloads Archive. *Future Generation Computer Systems*, 24(7):672–686, 2008.
[32] Erik-Jan van Baaren. WikiBench: A Distributed, Wikipedia based Web Application Benchmark. *Master Thesis, VU Univ. Amsterdam*, 2009.
[33] Sadeka Islam, Jacky Keung, Kevin Lee, and Anna Liu. Empirical Prediction models for Adaptive Resource Provisioning in the Cloud. *Future Generation Computer Systems*, 28(1), 2012.
[34] Sheng Di, Derrick Kondo, and Walfredo Cirne. Host Load Prediction in a Google Compute Cloud with a Bayesian Model. In *SC'12*.
[35] Arijit Khan, Xifeng Yan, Shu Tao, and Nikos Anerousis. Workload Characterization and Prediction in the Cloud: A Multiple Time Series Approach. In *NOMS '12*.
[36] Nikolas Roman Herbst, Nikolaus Huber, Samuel Kounev, and Erich Amrehn. Self-Adaptive Workload Classification and Forecasting for Proactive Resource Provisioning. In *ICPE '13*.
[37] Chunhong Liu, Chuanchang Liu, Yanlei Shang, Shiping Chen, Bo Cheng, and Junliang Chen. An Adaptive Prediction Approach based on Workload Pattern Discrimination in the Cloud. *Journal of Network and Computer Applications*, 80, 2017.
[38] Yexi Jiang, Chang-Shing Perng, Tao Li, and Rong N. Chang. ASAP: A Self-Adaptive Prediction System for Instant Cloud Resource Demand Provisioning. In *ICDM '11*.
[39] Joao Loff and Joao Garcia. Vadara: Predictive Elasticity for Cloud Applications. In *CloudCom '14*.
[40] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The Element of Statistical Learning: Data Mining, Inference, and Prediction. 2011.
[41] 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 *UCC '14*.
[42] James Bergstra, Remi Bardenet, Yoshua Bengio, and Balazs Kegl. Algorithms for Hyper-Parameter Optimization. In *NIPS '11*.
[43] Jingqi Yang, Chuanchang Liu, Yanlei Shang, Zexiang Mao, and Junliang Chen. Workload Predicting-Based Automatic Scaling in Service Clouds. In *CLOUD '13*.
[44] WikiBench. Wikipedia Traces. http://www.wikibench.eu, 2017. ONLINE.
[45] TU Delft. The Grid Workloads Archive. http://gwa.ewi.tudelft.nl, 2018.