Forecasting Cloud Application Workloads with **CloudInsight** for Predictive Resource Management

In Kee Kim, *Member, IEEE,* Wei Wang, *Member, IEEE,* Yanjun Qi, *Member, IEEE,* and Marty Humphrey, *Member, IEEE*

Abstract—
Predictive cloud resource management has been widely adopted to overcome the limitations of reactive cloud autoscaling. The predictive resource management is highly relying on workload predictors, which estimate short-/long-term fluctuations of cloud application workloads. These predictors tend to be pre-optimized for specific workload patterns. However, such predictors are still insufficient to handle real-world cloud workloads whose patterns may be unknown a priori, may dynamically change over time and may be irregular. As a result, these predictors often cause over-/under-provisioning of cloud resources. To address this problem, we create CloudInsight, a novel cloud workload prediction framework, leveraging the combined power of multiple workload predictors. CloudInsight creates an ensemble model using multiple predictors to make accurate predictions for real workloads. The weights of the predictors in CloudInsight are determined at runtime with their accuracy for the current workload using multi-class regression. The ensemble model is periodically optimized to handle sudden changes in the workload. We evaluated CloudInsight with various real workload traces. The results show that CloudInsight has 13%–27% higher accuracy than state-of-the-art predictors. Moreover, the results from trace-based simulations with a cloud resource manager show that CloudInsight has 15%–20% less under-/over-provisioning periods, resulting in high cost-efficiency and low SLA violations.

Index Terms—Cloud Computing, Workload Prediction, Ensemble Prediction Model, Predictive Resource Management, Autoscaling, Performance Evaluation

1 Introduction

Over the past decade, cloud computing has become a popular infrastructure for industry and research organizations due to its appealing capabilities such as scalability, flexibility, pay-as-you-go billing model [1]. In particular, *elasticity* [2, 3] has attracted application developers to move towards clouds to deploy their applications. Autoscaling [4–6], offered by public cloud providers (e.g., AWS), is the most common approach for attempting to achieve elasticity [7–13]. Autoscaling mechanisms and triggers monitor the utilization and behavior of current resources and adjust the size/amount of resources according to the fluctuation of workloads (e.g., job[1] requests) and user-defined rules (e.g., upper-/lower-bound of CPU usage). However, autoscaling can often be sub-optimal because of its *reactive* nature [14]. The *reactive* nature often results in over- and under-provisioning of cloud resources, in turn, low cost-efficiency and high SLA (Service Level Agreement) violations. Therefore, many predictive approaches have been proposed [15–32] for addressing the limitations of reactive autoscaling.

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* component, which allocates/deallocates cloud resources and maps user workloads to specific cloud resources. To achieve desired resource utilization and SLA satisfaction, it is crucial that the workload predictor should be optimized for the behavior of application workloads.

Existing predictive autoscaling managers often create and/or use a single *static* (or *"one-size-fits-all"* style) workload predictor with a simple assumption that the target workload has a stable pattern (e.g., increasing, cyclic bursty, and on-and-off) over time. Therefore, this prediction model is typically built offline and often requires significant efforts and resources to build. Furthermore, since cyclic bursty is known as a typical workload pattern for cloud applications [33, 34], time-series based approaches are widely used as the *one-size-fits-all* workload predictor to handle cyclic workloads [15, 22–29, 35–44].

However, the *"one-size-fits-all"* approach is *insufficient* to address real-world cloud application workloads because the patterns of real workloads are usually unknown *a priori*. To confirm this problem, we quantified the accuracy of various predictors operating in different workload patterns. The measurement results (in Section 2) show that without prior knowledge of the workload pattern, a user is hard-pressed to pick a proper *one-size-fits-all* predictor for his/her workloads. Additionally, real workloads are complicated in that they may experience dynamic pattern-shift/fluctuation over time. This dynamic nature of real workloads can be confirmed with traces from real-world cloud applications [45–48]. Figure 1 illustrates the job arrival rate of the traces within Google [49] and Facebook [50]. Neither trace follows regular patterns, and both traces are more likely to

- *I. K. Kim is with the Department of Computer Science, University of Georgia, Athens, GA 30602. E-mail: inkee.kim@uga.edu*
- *W. Wang is with the Department of Computer Science, University of Texas at San Antonio, San Antonio, TX 78249. E-mail: wei.wang@utsa.edu*
- *Y. Qi and M. Humphrey are with the Department of Computer Science, University of Virginia, Charlottesville, VA 22904. E-mail: yanjun@virginia.edu, humphrey@cs.virginia.edu*

1. We use a job, user request, and workload interchangeably.

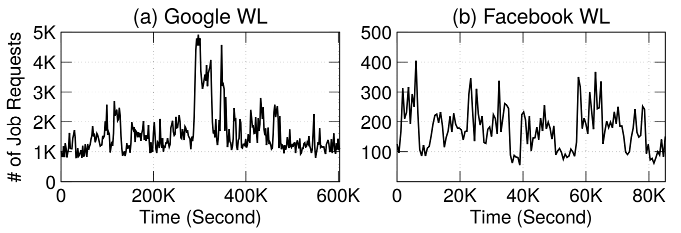

Fig. 1. Job arrival rate from cloud workload traces; (a) Google cluster trace [49] with 30-minutes of time interval and (b) Facebook Hadoop trace [50] with 5-minutes of time interval

be composed of short-lived interleaving patterns that have different characteristics. This dynamic fluctuation makes it difficult for a single static predictor to achieve high prediction accuracy through the lifetime of a cloud application. As Figure 1 shows, real workloads may not have clear and stable trend/seasonality. The lack of trend/seasonality implies that time-series models may not be the best choice for the cloud workloads in practice.

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 [51, 52]. Observing that different predictors excel at predicting different workload patterns, CloudInsight leverages the combined power of a diverse set of workload predictors. More specifically, CloudInsight dynamically creates an *ensemble model* that combines multiple predictors to predict the job arrival rate for the next time interval in the future. The weight of a predictor in the ensemble model is calculated at runtime based on the predictor's accuracy for the current workload at previous time intervals. 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. Since CloudInsight is an open-architecture, any different predictors from users' choice can be used in it.

This paper is based on our previous work [53], and we take a step further in the holistic performance evaluation of CloudInsight, particularly focused on how CloudInsight improves the overall performance in cloud resource management by minimizing the under-/over-provisioning state of the resources. More specifically, in this work, we provide an in-depth analysis of CloudInsight's contribution to cloud resource management by measuring cost-efficiency and SLA satisfaction based on trustworthy simulation with representative cloud resource management mechanisms. The improvement aims to obtain a complete understanding of predictive resource management and workload predictors' impacts in real clouds, as well as a better understanding of the effectiveness of our solution.

We have conducted comprehensive evaluations of the performance of CloudInsight with diverse real-world workload traces collected from cluster [49, 50], HPC (High-Performance Computing) [54], and web applications [55]. The experiment results show that CloudInsight outperforms existing time-series, machine learning, and specific custom predictors in every workload. CloudInsight has 13% to 27% better prediction accuracy with low overhead. Moreover, we also perform a trace-based simulation in combination with a representative resource management module. The results from the simulation study show that CloudInsight incurs 15% – 20% less under-/over-provisioning, resulting in 16% better cost-efficiency and 17% fewer SLA violations.

As a result, this work has the following contributions.

- **High accuracy and low overhead:** the CloudInsight framework is an online, multi-predictor based approach that performs highly accurate workload prediction with low overhead under dynamic cloud workloads with various patterns.
- **Online ensemble model:** a novel online mechanism to create an ensemble workload predictor. This mechanism dynamically assigns weights to each predictor by accurately estimating that the predictor's relative accuracy for the next time interval using multi-class regression.
- **Thorough performance evaluation:** we perform a comprehensive evaluation of the accuracy and overhead of CloudInsight with various workload traces collected from real cloud applications, including cluster, HPC, and web applications.
- **A simulation study of resource management:** a trace-based simulation with an autoscaling component confirms the actual benefit of CloudInsight to the resource management for cloud applications.

We structure the rest of the paper as follows. Section 2 describes the motivation of this work. Section 3 presents the framework details and implementation of CloudInsight. In Section 4, we evaluate CloudInsight with real-world workload traces. We further evaluate CloudInsight with cloud resource management in Section 5. In Section 6, we perform a sensitivity analysis of CloudInsight with more diverse predictors. Section 7 provides discussion and future work of CloudInsight. In Section 8, we summarize related work. Finally, Section 9 concludes this paper.

2 Motivation: No One Predictor Fits All Cloud Workload Patterns

We first investigated the degree to which a single existing predictor could be used across multiple typical cloud workload patterns. That is, if an existing predictor could perform moderately well even in a worst-case cloud workload environment, then a reasonable approach would be to configure this predictor, deploy the application, and hope for a decent-case cloud workload, knowing that in worst-case it will still suffice. To determine if such a predictor already exists in the community, we evaluated the prediction accuracy of 21 widely-used prediction algorithms, including two naive[2], six regression[3], seven time-series[4], and six machine-learning[5] models. We used four well-

2. *mean* and kNN (Nearest Neighbors).

3. Global and local models of linear, quadratic, and cubic regressions.

4. WMA (Weighted Moving Average), ES (Exponential Smoothing), BRDES (Brown's Double Exponential Smoothing), HWDES (Holt-Winters Double Exponential Smoothing), AR (Autoregressive), ARMA (Autoregressive and Moving Average), and ARIMA (Autoregressive Integrated Moving Average).

5. Linear and Gaussian SVMs, Decision Trees, Extra Trees, Gradient Boosting, and Random Forest models.

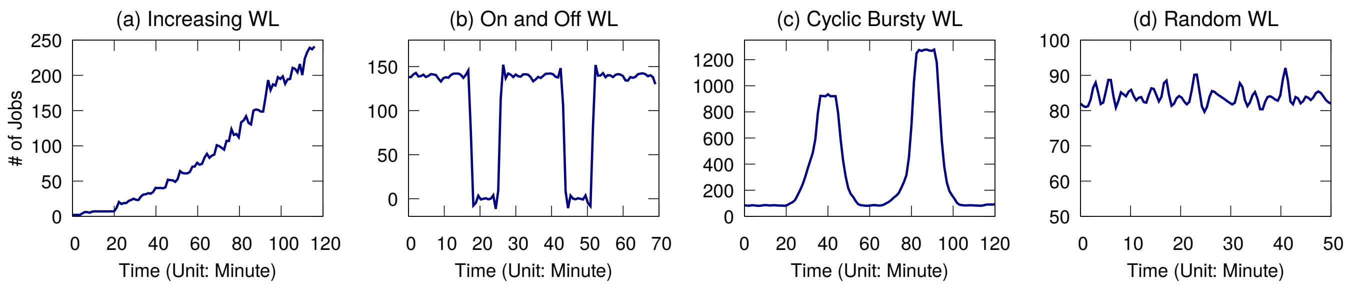

Fig. 2. Four synthetic cloud application workload patterns. The x-axis means the number of arrived jobs, and the y-axis indicates the time elapsed (unit: minutes).

known/synthetic cloud workload patterns [56, 57]. Figure 2 shows four patterns used in this evaluation. The x-axis in the graphs indicates the time elapsed (unit: minutes), and the y-axis means the number of arrived jobs. It is worth noting that Figure 2 only shows part of the workloads in order to confirm the characteristics of the four workload patterns. The total duration of each workload is from 6 to 12 hours.

The performance of all predictors with four workload patterns is measured by MAPE (Mean Absolute Percentage Error)[6]. The evaluation results are reported in Table 1, showing the best three predictors and an average accuracy from all the evaluated predictors regarding the four different workloads. The result shows that top predictors vary considerably for different workload patterns. 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 [3]. It is also worth noting that in Table 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.

These results indicate that a single predictor, i.e., the "*one-size-fits-all*" style predictor, is not sufficient to address diverse workload patterns that current cloud applications often experience. Additionally, if an application experiences a change in their workload, a *one-size-fits-all* predictor that performs well for the past workload may be suboptimal for the future workload. Based on the results, it can be argued that a new prediction mechanism is needed. The new mechanism should dynamically determine the best predictors for a given time and also quickly change this setting in the event of dynamic fluctuation and/or phased change. However, to design such a multi-predictor-based approach, further research questions must be answered:

- **RQ #1** – ***How to determine the top (most accurate) workload predictor(s)*** *when a workload is changing over time? (e.g., from a pattern to another one)*
- **RQ #2** – ***How to combine (ensemble) the top workload predictors*** *when it is unsafe to assume any of the top predictors is the best due to the random noises in a workload?*

In the rest of this paper, we will describe how our approach – CloudInsight – addresses these problems to predict dynamic and diverse workloads.

6. $MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Actual_i - Predicted_i}{Actual_i}\right| \times 100\%$

TABLE 1
MAPE results of workload predictors under four different workload patterns. (L-SVM: Linear SVM, G-SVM: Gaussian SVM, BRDES: Brown Double Exponential Smoothing, LR: Linear Regression)

Increasing Workload		**On and Off Workload**	
L-SVM	28%	G-SVM	22%
AR	29%	ARMA	30%
ARMA	30%	L-SVM	44%
Average	**51%**	**Average**	**69%**
Cyclic Bursty Workload		**Random Workload**	
ARIMA	38%	G-SVM	45%
BRDES	41%	LR	46%
L-SVM	43%	L-SVM	46%
Average	**75%**	**Average**	**52%**

3 Approach: CloudInsight

This section provides a detailed description of CloudInsight. Figure 3 illustrates the end-to-end architecture of CloudInsight. This framework 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 the *predictor pool*. The *model builder* is responsible for creating an ensemble prediction model by evaluating the performance of the predictors in the *predictor pool*. CloudInsight workload predictor provides the forecast for the near-future workload using an ensemble model created by the *model builder*. This prediction will be utilized by resource managers for predictive resource (e.g., VM) scaling.

3.1 Workflow of CloudInsight

Figure 4 depicts the workflow of CloudInsight. The workflow of CloudInsight runs through the following steps; 1) initial prediction and measurement ("initial step" in Figure 4), 2) ensemble model creation ("model creation" in Figure 4), and 3) workload prediction ("prediction" in Figure 4). Once the first step is finished, CloudInsight repeats the second and the third steps until the end of the workload. CloudInsight has two temporal interval parameters: (1) a prediction interval and (2) a model re-creation interval. The former is the interval to make a new future workload prediction, and the latter is the interval to re-create the ensemble model used by CloudInsight. The CloudInsight users set both intervals when

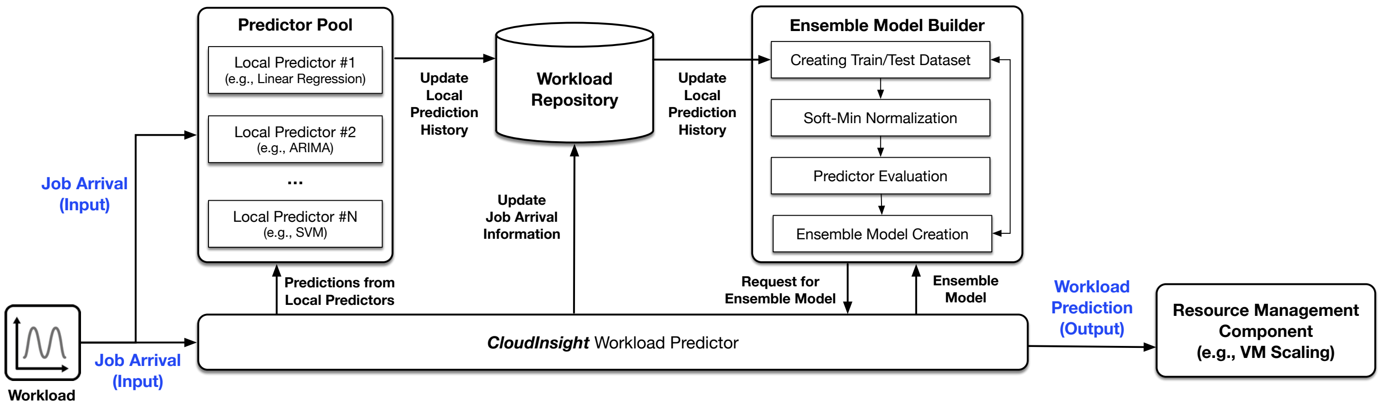
Predictor Pool
Local Predictor #1 (e.g., Linear Regression)
Local Predictor #2 (e.g., ARIMA)
...
Local Predictor #N (e.g., SVM)
Update Local Prediction History
Workload Repository
Update Local Prediction History
Ensemble Model Builder
Creating Train/Test Dataset
Soft-Min Normalization
Predictor Evaluation
Ensemble Model Creation
Job Arrival (Input)
Update Job Arrival Information
Predictions from Local Predictors
Request for Ensemble Model
Ensemble Model
Workload Prediction (Output)
Resource Management Component (e.g., VM Scaling)
CloudInsight Workload Predictor
Job Arrival (Input)
Workload

Fig. 3. Architecture of the CloudInsight Framework

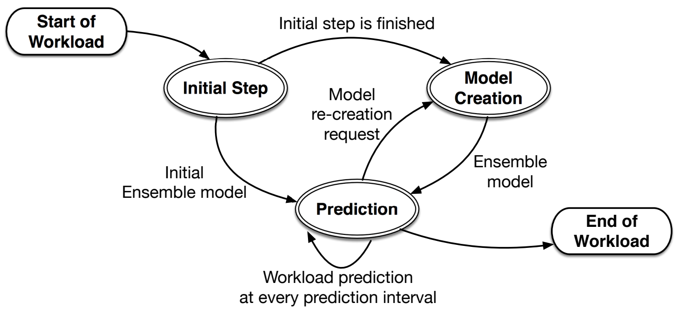

Fig. 4. Workflow of CloudInsight

the framework starts; normally, the prediction interval is shorter than the model creation interval.

Initial Prediction and Measurement. 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[7] 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). The use of a simple ensemble model here is because, during the initial period, CloudInsight does not have enough accuracy history for the local predictors. It is impossible to meaningfully assign weights to the predictors without this history.

Ensemble Model Creation: Once the initial (measurement) step finishes and initial accuracy history is collected, CloudInsight creates an ensemble prediction model based on the procedure described in Section 3.4. This ensemble model is used to predict future workload. After the expiration of the model re-creation interval, the ensemble model will be re-created.

Workload Prediction. This step is performed at every predefined 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.

7. The local predictors indicate the predictors in the *predictor pool*. Section 3.2 explains the local predictors in detail.

3.2 Predictor Pool

The *predictor pool* contains a set of workload predictors, called "local predictors." CloudInsight is designed to be generic and does not have any special dependency with a particular predictor. Consequently, its *predictor pool* can contain any predictors as long as those predictors can provide predictions for future workloads (e.g., job arrival rates). In other words, the outputs (prediction results) of local predictors in CloudInsight should be numerical values, and the outputs will be used to create a specific data structure for evaluating the performance of local predictors. This procedure will be further explained in Section 3.3. We have experimented with various workload predictors, including time-series, regressions, and machine-learning models. As shown later in Section 4, CloudInsight can properly handle all these types of predictors and considerably improve workload prediction accuracy. Because of CloudInsight's generality, users can add any workload predictors to the *predictor pool*. Note that, more local predictors may increase the overhead of workload prediction and ensemble model creation. Although CloudInsight's overhead is negligible in our evaluation with 21 predictors (in Section 6), users may want to limit the size of the *predictor pool* when there are hundreds of potential local predictors, and the overhead becomes non-trivial.

When actual workload comes to our target applications, all local predictors in the *predictor pool* make their predictions for future job arrival rates. However, because different local predictor works best for different workload patterns (or a particular part application workloads), the *model builder* would selectively consider the best ones and combine them with different contributions (weights). The *model builder* is responsible for properly selecting and combining them. We will describe how we address this issue.

3.3 Workload Repository

This component contains the prediction history of the all local predictors in the *predictor pool*. The prediction history is represented as a normalized performance vector.

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:

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

, where n is the number of the local predictors and m is the past (consecutive) 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 a window size of 50 for m in PV. This configuration works well for our case.)

Soft-Min Normalization. An issue of PV is that each element (PE) of the 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., 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 shown in Equation (2).

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

The input of Equation (2) is an element ($PE_{i,j}$) of PV. The numerator 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.) Also, 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 step, the sum of each column in a PV is 1. Intuitively, the soft-min value for a predictor at a particular prediction point can be viewed similarly as the probability of it being the best predictor for this prediction point.

3.4 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 [51, 52]. The essential insight of the MOE is that a collective result (e.g., ensemble) of all local experts is often better than a decision from a single expert [58]. In our case, each local predictor can be considered as a local expert in the MOE problem. Unlike a general MOE approach, which leverages a simple linear combination of all local experts, we create an ensemble model combined by the local predictors with different weights, because different local predictors work best for different workload patterns (or a particular part of dynamic real-world workloads). A higher weight of a local predictor indicates better performance for the current workload, and thus potentially better for the future.

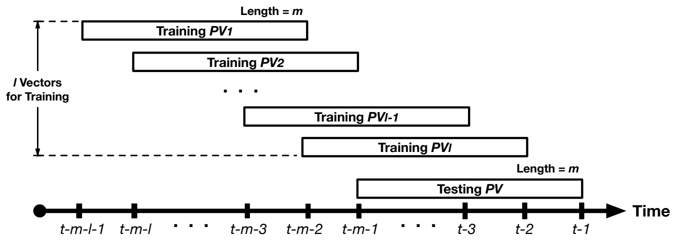

Fig. 5. Temporal coverage of PVs (Performance Vectors) for training and prediction input dataset. (m: length of row for PV, indicating m temporal points for past predictions, l: the size of training dataset, indicating the number of PVs in training dataset)

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 [59]. The following paragraphs give a detailed description of how the ensemble model is trained and used to make workload predictions.

Training Dataset and Prediction Inputs. Both training dataset and the prediction input dataset are represented as a collection of PVs as discussed in Section 3.3. However, these two datasets use separable PVs that cover different temporal windows. Suppose time t indicates the 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 dataset, 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 5 illustrates the temporal coverage of the training data set and prediction input data set.

Evaluation of Local Predictors. As we mentioned previously, we reduce the "evaluating local predictors" problem 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 can provide its projection for all local predictors. The output vector of this model is shown below.

$$Y = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_{n-1} \\ \omega_n \end{pmatrix} \tag{3}$$

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 a higher possibility to be the best predictor for workload prediction. Likewise, a lower value of ω_j (close to 0) suggests that jth predictor has a lower probability of being the best.

Creating an Ensemble Model for Workload Prediction. The ensemble workload predictor directly uses the output from the evaluation results of the local workload predictors. This ensemble model is constructed with Equation (4).

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

ω_i is the output from the previous step, and p_i is the prediction from a local predictor. By employing ω_i, this ensemble model gives higher weights to potentially more accurate local predictors and lowers weights to potentially less accurate local predictors. The result of this ensemble model is the prediction of the current and near-future job arrival rates, which can be utilized by other resource management components (e.g., VM scaling). Also, the predictions of the local predictors in the ensemble model will be updated to the *workload repository* for further evaluation of the local predictors. Note that the ensemble model is not necessarily created at every prediction point since this model requires the entire process of evaluating the local predictors, which is time-consuming. The ensemble model will be re-created periodically with a predefined time interval. In our evaluation (Section 4), we recreate the ensemble model after finishing every five predictions, which works well for our case.

3.5 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, a 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 [60] 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 BRDES, 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 take the same approach (grid search) with the way of tuning the local predictors, but we examine a 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 [56, 57] to guide the above processes of parameter selection. To ensure fair evaluation and avoid over-fitting, we did not use real workloads in parameter selection. Real workloads [45, 46, 54, 55] are only used to evaluate CloudInsight.

4 PERFORMANCE EVALUATION

4.1 Evaluation Setup

4.1.1 Workload Datasets

To evaluation CloudInsight, we used three categories of workload traces from real-world cloud applications: 1) Cluster workload traces from Google [49] and Facebook [50], 2) Scientific/HPC workloads from the Grid Workloads Archive [61] and 3) Wikipedia web traces from WikiBench [62]. These three groups of workload datasets allow us to evaluate CloudInsight with diverse scenarios of application deployment on clouds. We provide details on the characteristics of these workloads in Section 4.2.

4.1.2 Local Predictors

In this evaluation, we select 8 well-known workload predictors from the regression, time-series, and machine learning, and add them to the predictor pool in CloudInsight. The 8 local predictors are Linear regression [18, 63–66], WMA [28, 42, 43], BRDES [22, 23, 36], AR [24, 37, 67, 68], ARMA [25, 26, 38–40], ARIMA [27, 29, 41], Linear SVM [69], and Gaussian SVM [19, 70]. These predictors are chosen based on their performance with four different simulated workloads in the motivation measurement (Section 2). On average, these 8 predictors showed the highest accuracy for predicting four simulated workloads. More details about these local predictors are described in Appendix A.1.

4.1.3 Evaluation Goals

We have two goals in evaluating the performance of CloudInsight. First, we measure the accuracy of CloudInsight regarding the forecasting future job arrival rate (Section 4.3). We then evaluate the overhead of CloudInsight since prediction within deterministic time is a prerequisite of workload predictors (Section 4.4).

4.1.4 Performance Metrics

To measure the prediction accuracy of job arrive rate, we employ both MAPE and RMSE (Root Mean Square Error)[8]. Once we obtain RMSE results, all results are normalized over the result from CloudInsight to see the relative differences. 1.0 means the results of CloudInsight.

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.

4.1.5 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 [19, 27, 29, 69, 70] as well as the two best *"one-size-fits-all"* predictors that we have experimented

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

TABLE 2
Statistics of 13 evaluated workloads, collected from cluster, web, and HPC applications

	Workload	Duration	# Jobs	Predictor Setting	
				Prediction Interval	Model Recreation Interval
Cluster	Google	1 month	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	Wiki Global	3 days	823K	30 to 1200 sec.	At every 5 Preds.
	Wiki German		76.5K		
	Wiki Japanese		51K		
HPC	Grid 5000	22 days	62.5K	30 to 1200 sec. or 1 to 12 hrs (AuverGrid)	At every 5 Preds.
	NorduGrid	60 days	122K		
	AuverGrid	365 days	2.3M		
	SHARCNet	11 days	188K		
	LCG	33 days	435K		

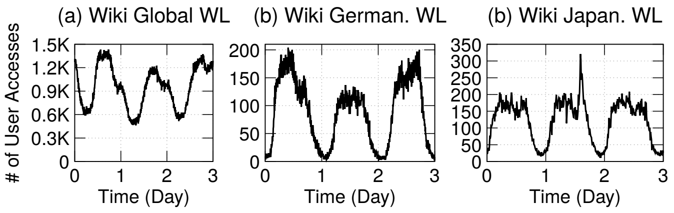

Fig. 6. Wikipedia traces with a 5-minutes of time interval.

with. We choose FFT [17, 21] and RSLR [15] from state-of-the-art approaches, which provide a robust and accurate prediction for cloud resource scaling.

4.2 Evaluated Workloads

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. Table 2 describes the summary of the characteristics of the workloads. We also choose workloads with a 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.

4.2.1 Cluster Workloads

These workloads represent the behaviors of cloud applications for cluster and big data analytics (e.g., Hadoop). We use Google Cluster trace [49] and Facebook Hadoop traces [50]. Google workload contains 2 millions of job arrival data for a one-month period. Facebook dataset contains 1 million of job submissions. Particularly for the Facebook workload, we use 4 sample traces (two traces from 2009 and two traces from 2010), each represents 1-day job submissions. The examples of visualizing such workloads are shown in Figure 1 in Section 1 (Introduction). We only show two of the cluster workload, but the other three workloads from Facebook have similar characteristics with Figure 1(b). The cluster workloads are varying significantly and have a dynamic nature of job submissions.

4.2.2 Web Workloads

These workloads represent the behaviors of web applications, which is a common application model on clouds.

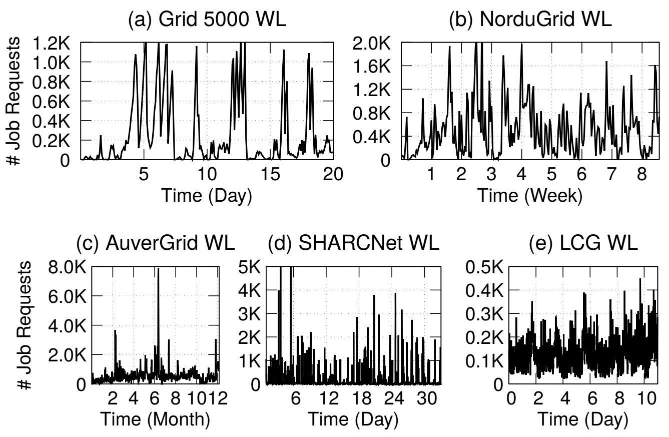

Fig. 7. HPC workload traces. (Grid 500 with 3600 sec. of time interval, NorduGrid with 7200 of time interval. AuverGrid with 12 hrs. of time interval, LCG with 600 sec. of time interval and SHARCNet with 900 sec. of time interval.)

We use three days of Wikipedia traces in September 2007. We focus on access log for (global) Wikipedia pages [71], German [72], and Japanese main page [73] of Wikipedia. The datasets have 823K (global), 76.5K (German), and 51K (Japanese) of user accesses. These Wikipedia workloads are illustrated in Figure 6. Wikipedia workloads generally show strong seasonality and trend characteristics, but the Japanese main page has an unexpected spike of user access.

4.2.3 HPC Workloads

The clouds are also actively used by many HPC applications to support diverse areas of scientific computing. Five workloads are used for HPC scenarios on the clouds. i.e., Grid5000 [74], NorduGrid [75], AuverGrid [76], SHARCNet [77], and LCG (LHC Computing Grid) [78]. These workloads respectively contain 62.5K jobs, 122K jobs, 2.3-million jobs, 188K jobs, and 435K jobs for various periods. These workloads are illustrated in Figure 7. HPC workloads have similar characteristics with two previous workloads but have more dynamic natures.

4.3 Prediction Accuracy of CloudInsight

To compare the accuracy of CloudInsight, with the baselines, we use various time intervals for the workload prediction,

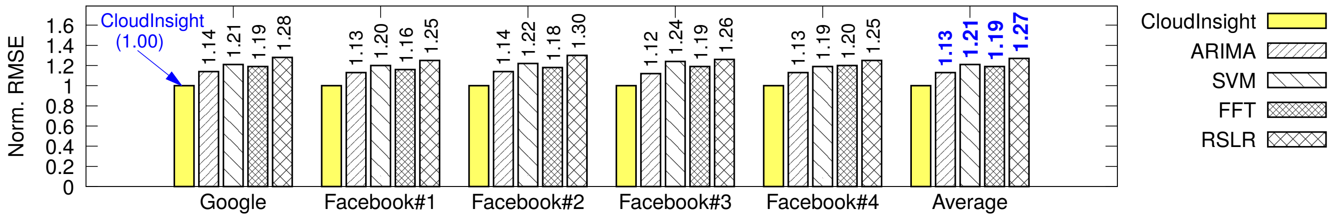

Fig. 8. Normalized RMSE results in cluster workloads (1.00 means the result from CloudInsight and higher values indicate worse performance. FB indicates Facebook. CI means CloudInsight.)

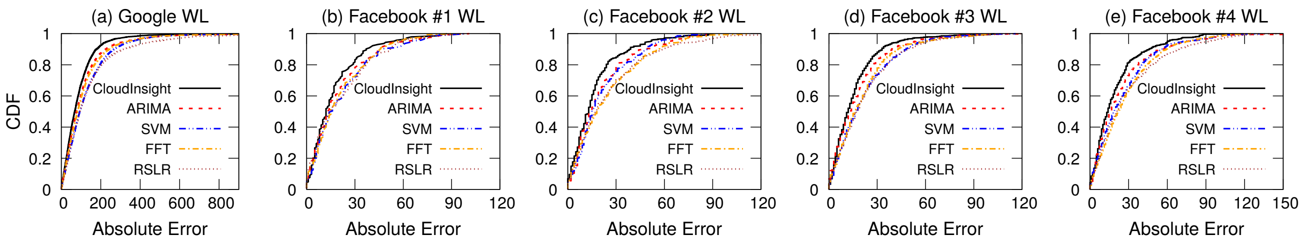

Fig. 9. CDF of prediction errors in cluster workloads (Absolute Error $= |Prediction_t - Actual_t|$)

TABLE 3
MAPE results with cluster workloads

Workload	Cloud-Insight	ARIMA	SVM	FFT	RSLR
Google	20%	28%	29%	34%	33%
Facebook #1	54%	62%	58%	65%	68%
Facebook #2	48%	55%	62%	70%	65%
Facebook #3	46%	56%	68%	51%	74%
Facebook #4	32%	38%	42%	50%	48%

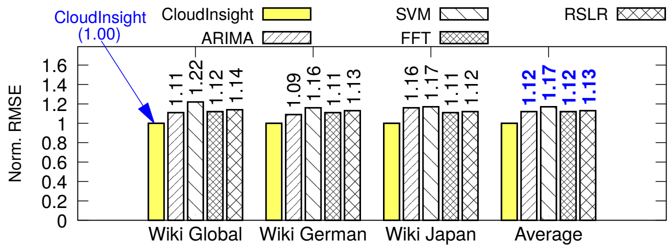

Fig. 10. Normalized RMSE results in web workloads

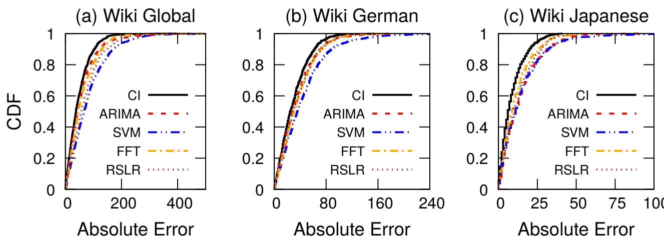

Fig. 11. Normalized RMSE results and CDF of prediction errors in web workloads

as shown in Table 2. 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.

4.3.1 *Accuracy with Cluster Workload*

The RMSE results of job arrival rate predictions of the five approaches are shown in Figure 8 (all RMSE 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. However, CloudInsight can keep adjusting the weights for each predictor and create a new ensemble model periodically to fit the changes in the workload. Therefore, CloudInsight can show better performance for workload prediction.

Table 3 reports MAPE results of all five predictors with cluster workloads. CloudInsight shows the lowest MAPE against four baselines and has 4% to 32% more accurate than others. While CloudInsight is the most accurate predictor, it also shows high prediction errors (over 40% of MAPE) for three cluster workloads (e.g., Facebook #1, #2, and #3). This is mainly due to the dynamic nature of cluster workloads, and the job arrival rate is highly fluctuating. For the three Facebook workloads, other baselines show even higher errors like 60% to 70% of MAPE.

Figure 9 shows the CDF (Cumulative Distribution Function) of prediction errors in cluster workloads. We only show four results (without Facebook #4) from cluster workloads due to page limitations. The results from Facebook #4 shows a similar graph with the other three Facebook workloads. The x-axis represents absolute prediction error, i.e., $|Prediction_t - Actual_t|$, while the y-axis gives the

TABLE 4
MAPE results with web workloads

Workload	CloudInsight	ARIMA	SVM	FFT	RSLR
Wiki Global	4%	6%	8%	6%	7%
Wiki German	15%	20%	23%	21%	20%
Wiki Japanese	19%	26%	34%	28%	24%

TABLE 5
MAPE results with HPC workloads

Workload	CloudInsight	ARIMA	SVM	FFT	RSLR
Grid 5000	54%	62%	58%	65%	68%
NorduGrid	20%	28%	29%	34%	33%
AuverGrid	48%	55%	62%	70%	65%
SHARCNet	46%	56%	68%	51%	74%
LGC	32%	38%	42%	50%	48%

cumulated probability of the errors. As Figure 9 shows, 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.

4.3.2 Accuracy with Web Workload

The RMSE results for web workloads are shown in Figure 10, 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, such characteristics can still change over time, albeit less abruptly. Also, as shown in the 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.

Table 4 describes the MAPE results with three web workloads. Similar to the RMSE results, CloudInsight outperforms all the baselines. In particular, CloudInsight shows 4%, 15%, and 19% of MAPE results with Wikipedia global page, Wikipedia German page, and Wikipedia Japanese page, respectively. These results indicate, on average, a 6% improvement over four baselines.

Figure 11 illustrates the CDF of prediction errors in web workloads. Similarly, the majority of CloudInsight's errors are still smaller than those of the baselines.

4.3.3 Accuracy with HPC Workload

Figure 12 shows the 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

TABLE 6
Prediction overhead of five approaches

	Cluster	Web	HPC	**Average**
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	**0.38** ms
FFT	4.2 ms	5.9 ms	8.8 ms	**6.3** ms
RSLR	22 ms	18 ms	22 ms	**21** ms

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.

Table 5 contains MAPE results with HPC workloads from the five predictors. In this evaluation, CloudInsight can maintain the lowest MAPE results among all the predictors by providing the most accurate predictions. Similar to the cluster workloads, CloudInsight also shows high MAPE results (over 40%) for Grid 5000, AuverGrid, and SHARCNet workloads. The high MAPE results are mainly due to the characteristics of workloads that job arrivals in HPC workloads can be dynamically fluctuating with several bursty (e.g., sudden traffic spike) and intermittent periods (e.g., no job submissions) so that predicting workloads with such characteristics are very challenging for all workload predictors. And it is also important to emphasize that, on average, MAPE of CloudInsight is 12.8% lower (more accurate) than MAPE from the baselines.

Figure 13 shows the CDF of the prediction errors in the HPC workloads. The results from HPC workload traces are similar to the two previous cluster and web workload types. The curves for CloudInsight are more skewed to the left, indicating that CloudInsight has fewer errors than other baselines. Please note that in Figure 13, we only show results from four HPC workload traces and exclude the CDF result from NorduGrid due to page limitation. However, the CDF of prediction errors in NorduGrid shows a similar trend with other HPC workloads.

4.4 Overhead of CloudInsight

Forecasting future workloads within a short and deterministic amount of time is also a very critical property for all predictive resource scaling systems. Here, 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 a 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.

4.4.1 Prediction Overhead

Table 6 gives the average prediction overhead for all the approaches. On average, 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

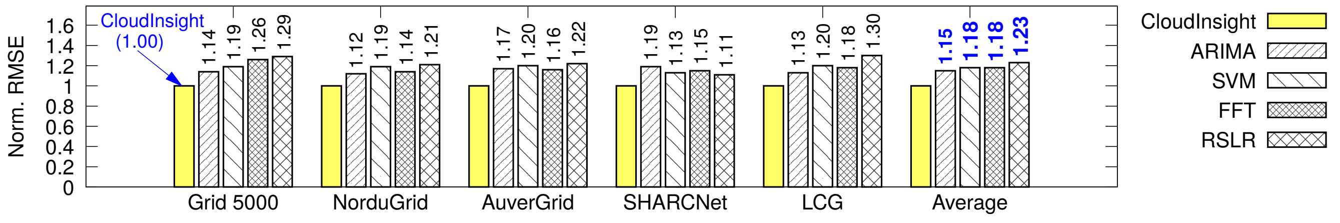

Fig. 12. Normalized RMSE results in HPC workloads

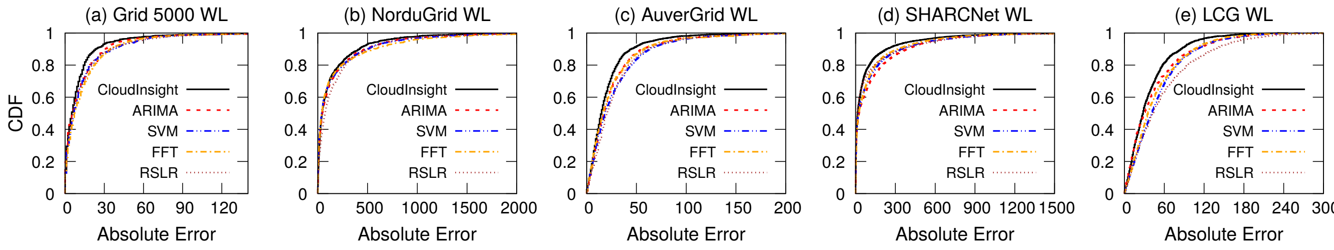

Fig. 13. CDF of prediction errors in HPC workloads

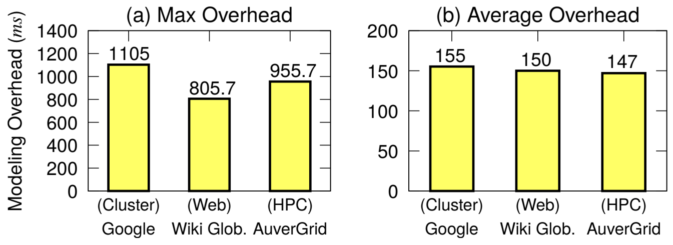

Fig. 14. Modeling overhead of CloudInsight

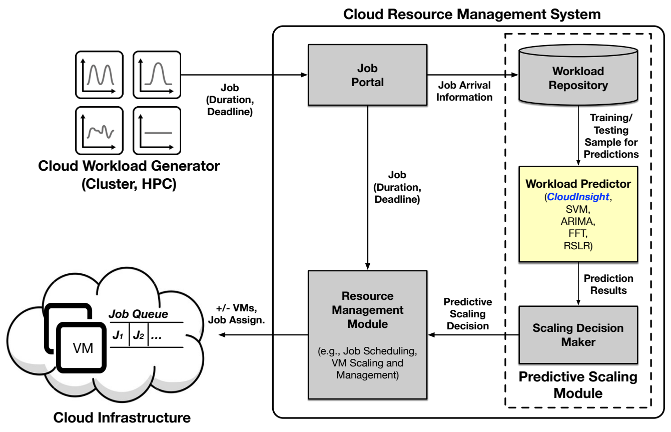
Cloud Resource Management System
Cloud Workload Generator (Cluster, HPC)
Job (Duration, Deadline)
Job Portal
Job Arrival Information
Workload Repository
Training/ Testing Sample for Predictions
Workload Predictor (CloudInsight, SVM, ARIMA, FFT, RSLR)
Prediction Results
Scaling Decision Maker
Predictive Scaling Module
Predictive Scaling Decision
Resource Management Module (e.g., Job Scheduling, VM Scaling and Management)
+/- VMs, Job Assign.
Job Queue
VM
Cloud Infrastructure

Fig. 15. Simulation model of a predictive cloud resource management system with workload predictors

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 8 local predictors, it takes only $12ms$ more time as compared to ARIMA. This prediction overhead ($34ms$) is achieved with parallel processing of all local predictors. When predicting the future job arrival, CloudInsight invokes all local predictors, and then each predictor forecasts the next job arrival individually. Thus, the prediction overhead of CloudInsight is significantly affected by the prediction overhead of the slowest local predictor. In our case, ARMA was the slowest predictor among the eight local predictors, and it has about $30ms$ of prediction overhead. Once all the predictors finish their prediction, CloudInsight calculates the weight of each predictor and generates the final prediction result.

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 [79]), autoscaling resource managers often reconfigure their resources at an interval higher than 30 seconds. As CloudInsight's prediction time is much shorter than the prediction interval, it imposes a very limited impact on an autoscaling resource manager.

4.4.2 Ensemble Modeling Overhead

The ensemble modeling overhead of CloudInsight is defined as the time to create a new ensemble model. The results of this overhead are shown in Figure 14. Limited by space, we only show the results from the largest workloads, which are Google, Wiki Global Main, and AuverGrid. The overheads results from the other workloads are 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 creates a new ensemble model and make predictions within the autoscaling resource reconfiguration intervals as stated previously.

5 Case Study: Predictive Resource Management with CloudInsight

This section performs further analysis of CloudInsight. Specifically, we conduct a simulation-based study with predictive resource management and evaluate the benefit brought by CloudInsight to cloud resource management.

5.1 Simulation Setup

We design a representative resource management system of cloud resources as shown in Figure 15. We embed CloudInsight and the four baselines (SVM, ARIMA, FFT, and RSLR) to the simulation model of the resource management module. All these experiments are conducted on a trace-based simulator [80], which closely resembles the behaviors of real cloud applications. Our simulation model is composed of three sub-components; (1) workload generator, (2) resource manager, and (3) cloud infrastructure (public cloud).

The **workload generator** creates user requests (job arrivals) as per the workload traces (e.g., cluster and HPC workloads) and uses the same configuration with Table 2. Each job is also associated with a randomly generated deadline. The **resource manager** is responsible for determining predictive scaling decisions to handle the incoming workloads. The workload predictors (e.g., CloudInsight, the four baselines) are part of this resource manager. For the scaling-out operation, the workload predictor forecasts the future job arrival rate, and the manager adds more resources if the demand is higher than the currently available VM resources. For the scaling-in operation, the manager checks the status of VMs in every minute and terminates the VM if the VM has no further jobs to process. For the **cloud infrastructure** (public cloud) setting, we use an on-demand homogeneous VM type for the simplicity of this evaluation and use a minute-based pricing model for the cost calculation.

5.2 Simulation Results

We used three metrics to evaluate the effectiveness of the resource manager; (1) cloud usage cost, (2) job deadline miss rate, and (3) under-/over-provisioning time. A higher cloud usage cost means that the VMs are more likely to be over-provisioned (poor cost-efficiency). A higher job deadline miss rate indicates that the VMs are more like to be under-provisioned (more SLA violations). Figure 16 and 17 depict the costs and job deadline miss rates of all approaches with cluster and HPC workloads. The results show that the resource manager using CloudInsight has the lowest cost and lowest job deadline miss rate. On average, CloudInsight shows 14% – 22% better cost-efficiency and 12% – 21% less job deadline miss rate as compared to the baselines. Table 7 shows the normalized under-/over-provisioning time of baselines. These accumulated under-/over-provisioning periods (from the baselines) are normalized over the results from CloudInsight. As the results show, CloudInsight has 15% – 19% less under-/over-provisioning periods compared to the baselines. These results imply that as CloudInsight increases the accuracy of future job arrival rate predictions, it can significantly reduce the under-/over-provisioning of the cloud resources. The low under- and over-provisioning periods also mean that the resource manager with CloudInsight always provisions the proper and accurate amount of VM resources according to the actual workload changes while retaining a high SLA satisfaction rate and cost-efficiency.

TABLE 7
Normalized sum of under-/over-provisioning time

	ARIMA	SVM	FFT	RSLR
Google	18%	27%	18%	32%
Facebook #1	9%	14%	13%	18%
Facebook #2	14%	13%	14%	17%
Facebook #3	17%	20%	17%	18%
Facebook #4	15%	17%	22%	27%
Grid 5000	13%	16%	12%	14%
NorduGrid	14%	15%	12%	16%
AuverGrid	16%	18%	14%	16%
SHARCNet	20%	17%	18%	14%
LCG	15%	16%	14%	19%
Average	**15%**	**18%**	**15%**	**19%**

6 Sensitivity Analysis of CloudInsight with More Predictors

In Section 4, we use only 8 local predictors to measure the performance (accuracy and overhead) of CloudInsight. However, the group of local predictors may affect the accuracy of CloudInsight. As CloudInsight has no limitation in leveraging the number of predictors, we perform a sensitivity analysis of CloudInsight with more number of local predictors to confirm how this change affects its performance.

To evaluate the impact of the composition of local predictors, we measured the accuracy and overhead of CloudInsight by adding 13 more local predictors (in total 21 local predictors) that are mentioned in Section 2. As we discussed in Section 2, these 13 predictors[9] showed lower average accuracy than the 8 predictors used by CloudInsight. In practice, it may be difficult for a user to determine which predictors are better than the others for his/her workload. Therefore, a user may be inclined to include all known predictors, many of which could have low accuracy. By including more predictors, this evaluation aims at determining whether CloudInsight can still correctly pick the best predictors from a large pool of mixed predictors.

6.1 Accuracy Sensitivity

Figure 18 shows the normalized accuracy of using 21 predictors. CloudInsight with 21 predictors has similar or even better accuracy than predictions using 8 predictors. In particular, when predicting Web and HPC workloads, CloudInsight with 21 predictors produces better accuracy than CloudInsight with 8 predictors. The improved results show that the *model builder* of CloudInsight can correctly evaluate the accuracy of local predictors and correctly assign higher weights for good predictors and lower weights for poor predictors. Furthermore, for certain parts of a workload, a predictor from the additional 13 predictors may perform better than any of the 8 averagely-best predictors. The higher accuracy in web and HPC workloads indicates that CloudInsight can correctly catch the accuracy benefits from the additional 13 predictors for certain parts of the workloads and utilize their better accuracy to improve overall prediction results. These

9. The 13 more predictors include mean-based model, *k*NN, polynomial and quadratic regression, local regression models, exponential moving average, Holt-Winters DES, decision tree, and three ensemble models (gradient boosting, extra-trees, random forest). More details about these additional 13 local predictors are described in Appendix A.2.

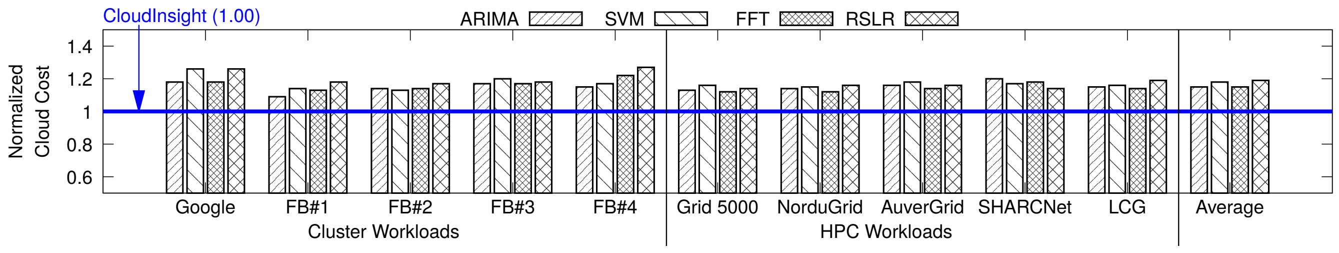

Fig. 16. Cost usage of the resource management with five approaches.

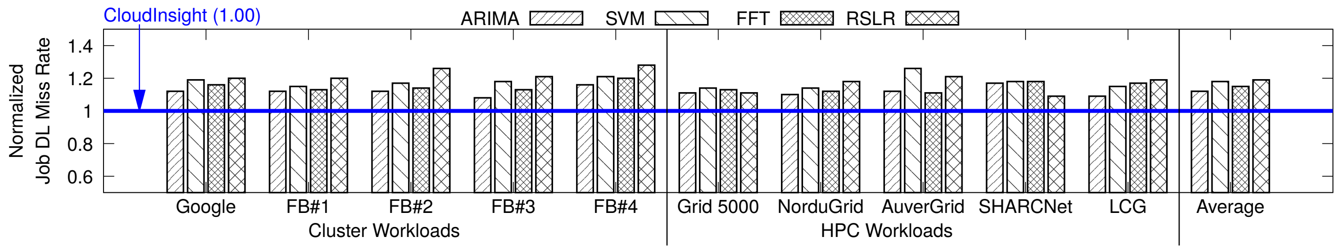

Fig. 17. Job deadline miss rate of the resource management with five approaches.

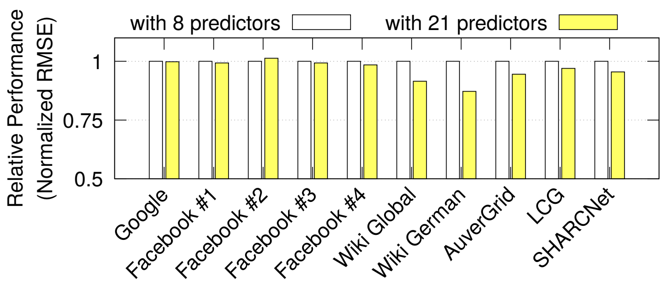

Fig. 18. Result of sensitivity analysis of CloudInsight with the more number of local predictors. Note that lower values mean better accuracy and vice versa

results also show that CloudInsight is a generic framework that can be used with a myriad of types of predictors.

However, we also experience that CloudInsight with 21 predictors can perform slightly worse than CloudInsight with 8 averagely-best predictors. This case is found in the evaluation with Facebook #2, and CloudInsight with 21 predictors shows a slightly lower (1.3%) performance than the original CloudInsight. This worse performance may happen when evaluating the future accuracy of local predictors (described in Section 3.4), CloudInsight may incorrectly weight more on poor predictors, which slightly impact on the overall prediction accuracy of the final ensemble model. This case indicates that CloudInsight still has room for improvement.

6.2 Overhead with More Predictors

While using more predictors does not hurt (sometimes even improves) prediction accuracy, using more predictors increases the prediction overhead of CloudInsight. With 21 predictors, CloudInsight was 3.85x slower than only using 8 predictors. We observe that most overhead comes from re-creating the ensemble model: more local predictors exponentially increases the overhead for model re-creation. It is worth noting that even though using 21 predictors is slower, the absolute time for prediction and model re-creation is still much smaller than the prediction and model re-creation intervals (or resource reconfiguration interval). Therefore, the overhead of 21 predictors has a limited impact on practical usage. However, if a user has a much larger set of potential local predictors, he/she may want to limit the number of predictors employed to reduce overhead. In the future, we will investigate how to determine the optimal types/numbers of local predictors for CloudInsight.

7 Discussion and Future Work

7.1 Determining the Prediction and Modeling Intervals

CloudInsight requires two user inputs, which are an ensemble model re-creation interval and a prediction interval (described in Section 3.1). These two intervals play an important role since they may impact on the overall accuracy of and the overhead of CloudInsight. Regarding the model re-creation interval, a smaller value of the re-creation interval is better for CloudInsight than a larger value of the interval because CloudInsight will create the ensemble model more frequently (with a smaller interval), in turn, can have a more accurate final model. However, to determine the interval of the model re-creation, the overhead caused by CloudInsight also needs to be considered. So we recommend users to determine a tradeoff value for this interval. Regarding the decision of the prediction interval, the CloudInsight users should consider application dynamics (as you commented), resource management and job scheduling, as well as workload dynamics.

To help CloudInsight users' decisions on two intervals, we will develop an automated approach, which is to find optimal (or near-optimal) parameters and dynamically tune/update the values for the intervals. The automated approach is not only to maximize the overall prediction accuracy but also overhead caused by CloudInsight needs to be manageable. We will investigate Bayesian Optimization with hyperparameter tuning [81] and/or reinforcement learning [82] to optimize and tune the values for two intervals at run-time.

7.2 Deployment of CloudInsight on Production Cluster

When deploying CloudInsight on a production cluster to predict future workload to real-world applications, CloudInsight may need to collaborate with open-source monitoring tools like Sysdig [83], cAdvisor [84], and Prometheus [85]. These monitoring tools offer capabilities to monitor network IOs of managed applications, and the network IOs can be a feature to be interpreted as job arrivals. Thus, we will first investigate whether the change of network IOs has a correlation with job arrivals in the managed applications. If there is a correlation between network IO changes and job arrival, the monitoring tools can be potentially integrated with CloudInsight. Then, we can develop CloudInsight's APIs to communicate with target monitoring tools and translate network IOs into (numerical) forms, so that local predictors can take the network IOs as input parameters. With these two steps, in the near future, we will improve CloudInsight in order to provide proper instructions and APIs that can support various open-source monitoring tools to support user applications running on production clusters.

8 Related Work

A wide variety of predictive approaches has been proposed with an emphasis on accurately forecasting temporal and other properties (e.g., resource needs, data size) of cloud workloads. Most notably, employing a "*one-size-fits-all*" style workload predictor is the mainstream of such proposals. Diverse models from statistical and machine learning domains are used to design the workload predictor. These models are generally relying on regression, time-series, and machine learning [69, 70, 86, 87]. Among them, time-series approaches are the most popular approach. (ES [22, 23, 35, 36], AR [24, 37, 67, 68], ARMA [25, 26, 38–40], ARIMA [27, 29, 41] and others [28, 42, 43].) However, as discussed in Section 2, 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. Further, several custom predictive approaches are developed to address dynamic cloud workload patterns. PRESS [21] and CloudScale [17] employ a custom predictor composed of FFT and Markov model. FFT is used to detect a signature of workload patterns, and the 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. [15] developed a hybrid approach, combining robust linear stepwise regression and model refinement. This technique requires offline profiling for the initial linear model creation, but CloudInsight is a purely online model that does not require such offline profiling steps.

To overcome the limitations of the "*one-size-fits-all*" predictors, multi-predictor approaches are also proposed. These approaches are in the spirit similar to CloudInsight because they are designed to provide more generic and adaptive nature to their target application and resource management infrastructure. Khan et al. [88], Herbst et al. [89], Liu et al. [63], and Züfle et al. [44] proposed an adaptive model for workload prediction. These approaches employ a classification, clustering (e.g., decision tree), and/or recommendation system based on incoming workloads, and these approaches 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*. Therefore, these approaches are not general enough to handle unknown, varying, and/or non-typically-patterned real workloads. CloudInsight, on the other hand, makes no assumption about the class/type/patterns and thus can handle these real workloads. Moreover, the extra step of classification and clustering adds additional overhead to each job arrival prediction.

ASAP [90] and Vadara [91] 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 [30]) 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 [3] can break this assumption. Unlike these approaches, we utilize a much longer history in CloudInsight and employ a multi-class regression model to predict the future accuracy of local predictors. Therefore, CloudInsight can provide more robust weights and more accurate predictions.

Due to the proliferation of deep learning [92], there are several approaches to apply deep learning models to cloud workload prediction [93–98]. In particular, LSTM [99] or LSTM-variants [93, 95, 98] are considered as a powerful tool for predicting future workloads because of LSTM's capability for time-series prediction. However, deep learning-based models, including LSTM, require a massive amount of (workloads) training dataset and computing resources. On the other hand, CloudInsight is a lightweight (as evaluated in Section 4.4.1 and 4.4.2), and it can be promptly adapted to sudden changes in the cloud workloads.

9 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 (contributions) 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 (e.g., cluster, web, and HPC workloads). Our evaluation results show that CloudInsight has 13% – 27% of better accuracy than state-of-the-art *one-size-fits-all* style predictors, and it also has low overhead for predicting future workload changes ($< 100ms$) and (re)creating a new ensemble model ($< 1.1sec.$).

In conclusion, the mechanism and evaluation results of CloudInsight show that our approach is capable of addressing real-world cloud workloads that have dynamic and high variable nature. This work will help other cloud researchers and practitioners design a new predictive method for managing and scaling cloud resources autonomously.

References

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A View of Cloud Computing. *Communications of the ACM*, 53(4):50–58, 2010.

[2] Nikolas Roman Herbst, Samuel Kounev, and Ralf Reussner. Elasticity in Cloud Computing: What It Is, and What It Is Not. In *International Conference on Autonomic Computing (ICAC '13)*, San Jose, CA, USA, June 2013.

[3] Sadeka Islam, Srikumar Venugopal, and Anna Liu. Evaluating the Impact of Fine-scale Burstiness on Cloud Elasticity. In *ACM Symposium on Cloud Computing (SoCC '15)*, Hawaii, August 2015.

[4] AWS auto scaling. https://aws.amazon.com/autoscaling, 2019.

[5] Microsoft azure autoscale. https://azure.microsoft.com/en-us/features/autoscale, 2019.

[6] Google. Google cloud platform – autoscaling groups of instances. https://cloud.google.com/compute/docs/autoscaler, 2019.

[7] Luwei Cheng, Jia Rao, and Francis C.M. Lau. vScale: Automatic and Efficient Processor Scaling for SMP Virtual Machines. In *ACM European Conf. on Computer Sys. (EuroSys)*, London, UK, Apr. 2016.

[8] Tayler H. Hetherington, Mike O'Connor, and Tor M. Aamodt. MemcachedGPU: Scaling-up Scale-out Key-value Stores. In *ACM Symposium on Cloud Computing (SoCC '15)*, Kohala Coast, Hawaii, USA, August 2015.

[9] Bailu Ding, Lucja Kot, Alan Demers, and Johannes Gehrke. Centiman: Elastic, High Performance Optimistic Concurrency Control by Watermarking. In *ACM Symposium on Cloud Computing (SoCC '15)*, Kohala Coast, Hawaii, USA, August 2015.

[10] Thomas Heinze, Lars Roediger, Andreas Meister, Yuanzhen Ji, Zbigniew Jerzak, and Christof Fetzer. Online Parameter Optimization for Elastic Data Stream Processing. In *ACM Sympo. Cloud Computing (SoCC '15)*, Kohala Coast, Hawaii, August 2015.

[11] Christina Delimitrou and Christos Kozyrakis. Quasar: Resource-Efficient and QoS-Aware Cluster Management. In *International Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Salt Lake City, UT, March 2014.

[12] Ganesh Ananthanarayanan, Christopher Douglas, Raghu Ramakrishnan, Sriram Rao, and Ion Stoica. True Elasticity in Multi-Tenant Data-Intensive Compute Clusters. In *ACM Symposium on Cloud Computing (SoCC '12)*, San Jose, CA, USA, October 2012.

[13] Herodotos Herodotou, Fei Dong, and Shivnath Babu. No One (Cluster) Size Fits All: Automatic Cluster Sizing for Data-intensive Analytics. In *ACM Symposium on Cloud Computing (SoCC '12)*, Cascais, Portugal, October 2011.

[14] Marco A. S. Netto, Carlos Cardonha, Renato L. F. Cunha, and Marcos D. Assuncao. Evaluating Auto-scaling Strategies for Cloud Computing Environments. In *IEEE International Symp. Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '14)*, Paris, France, September 2014.

[15] Timothy Wood, Ludmila Cherkasova, Kivanc M. Ozonat, and Prashant J. Shenoy. Profiling and Modeling Resource Usage of Virtualized Applications. In *The 9th ACM Middleware Conference (Middleware '08)*, Leuven, Belgium, December 2008.

[16] Waheed Iqbala, Matthew N. Daileya, David Carrerab, and Paul Janeceka. Adaptive resource provisioning for read intensive multi-tier applications in the cloud. *Future Generation Computer Systems*, 27(6):871–879, 2011.

[17] 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 '11)*, Cascais, Portugal, October 2011.

[18] Nohhyun Park, Irfan Ahmad, and David J. Lilja. Romano: Autonomous Storage Management using Performance Prediction in Multi-Tenant Datacenters. In *ACM Symposium on Cloud Computing (SoCC '12)*, San Jose, CA, USA, October 2012.

[19] Neeraja J. Yadwadkar, Ganesh Ananthanarayanan, and Randy Katz. Wrangler: Predictable and Faster Jobs using Fewer Resources. In *ACM Symposium on Cloud Computing (SoCC)*, Seattle, WA, November 2014.

[20] Daniel Jacobson, Danny Yuan, and Neeraj Joshi. Scryer: Netflix's predictive auto scaling engine. The Netflix Tech Blog, November 2013. Posted at http://techblog.netflix.com/2013/11/scryer-netflixs-predictive-auto-scaling.html.

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

[22] 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 '13)*, Denver, CO, USA, November 2013.

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

[24] 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 *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, CA, USA, April 2008.

[25] 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 *ACM European conference on Computer Systems (Eurosys '09)*, Nuremberg, Germany, April 2009.

[26] 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 '11)*, Washington DC, USA, June 2011.

[27] Rodrigo N. Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications' QoS. *IEEE Transactions on Cloud Computing*, 3(4):449–458, 2015.

[28] Anshul Gandhi, Mor Harchol-Balter, Ram Raghunathan, and Michael A. Kozuch. AutoScale: Dynamic, Robust Capacity Management for Multi-Tier Data Centers. *ACM Transactions on Computer Systems*, 30(4):14:1–14:26, 2012.

[29] 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.

[30] Hector Fernandez, Guillaume Pierre, and Thilo Kielmann. Autoscaling Web Applications in Heterogeneous Cloud Infrastructures. In *IEEE International Conference on Cloud Engineering (IC2E '14)*, Boston, MA, USA, March 2014.

[31] Maryam Amiri, Leili M. Khanli, and Raffaela Mirandola. An online learning model based on episode mining for workload prediction in cloud. *Future Gen. Comp. Syst.*, 87:83–101, 2018.

[32] Jitendra Kumar and Ashutosh Kumar Singh. Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Gen. Comp. Syst.*, 81:41–52, 2018.

[33] 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 *IEEE International Conf. on Cloud Engineering (IC2E '14)*, Boston, MA, March 2014.

[34] Yexi Jiang, Chang S. Perng, Tao Li, and Rong N. Chang. Cloud Analytics for Capacity Planning and Instant VM Provisioning. *IEEE Trans. on Net. and Service Manage.*, 10(3):312–325, Aug. 2013.

[35] Eyal Zohar, Israel Cidon, and Osnat Mokryn. The Power of Prediction: Cloud Bandwidth and Cost Reduction. In *ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '11)*, Toronto, ON, Canada, August 2011.

[36] 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 Computing Data Centers. In *IEEE International Conference on Services Computing (SCC '10)*, Maiami, FL, USA, July 2010.

[37] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Black-box and Gray-box Strategies for Virtual Machine Migration. In *USENIX Symp. on Networked Systems Design and Implementation (NSDI)*, Cambridge, MA, April 2007.

[38] Wei Fang, ZhiHui Lu, Jie Wu, and ZhenYin Cao. RPPS: A Novel Resource Prediction and Provisioning Scheme in Cloud Data Center. In *IEEE International Conference on Services Computing (SCC '12)*, Honolulu, HI, USA, June 2012.

[39] Juan M. Tirado, Daniel Higuero, Florin Isaila, and Jesus Carretero. Predictive Data Grouping and Placement for Cloud-based Elastic Server Infrastructures. In *IEEE/ACM Intl' Symp. on Cluster, Cloud and Grid Comp. (CCGrid)*, Newport Beach, CA, May 2011.
[40] Gueyoung Jung, Matti A. Hiltunen, Kaustubh R. Joshi, Richard D. Schlichting, and Calton Pu. Mistral: Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures. In *International Conference on Distributed Computing Systems (ICDCS '10)*, Genova, Italy, June 2010.
[41] 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 *International Conference on Autonomic Computing (ICAC '12)*, San Jose, CA, USA, September 2012.
[42] Andrew Krioukov, Prashanth Mohan, Sara Alspaugh, Laura Keys, David E. Culler, and Randy H. Katz. NapSAC: Design and Implementation of a Power-Proportional Web Cluster. *Computer Communication Review*, 41(1):102–108, 2011.
[43] 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 *IEEE Conference on Computer Comm. (INFOCOM)*, Toronto, ON, Canada, April 2014.
[44] Marwin Züfle, André Bauer, Veronika Lesch, Christian Krupitzer, Nikolas Herbst, Samuel Kounev, and Valentin Curtef. Autonomic forecasting method selection: Examination and ways ahead. In *2019 IEEE International Conference on Autonomic Computing, ICAC 2019, Umeå, Sweden, June 16-20, 2019*, 2019.
[45] Charles Reiss, Alexey Tumanov, Gregory R. Ganger, Randy H. Katz, and Michael A. Kozuch. Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis. In *ACM Symposium on Cloud Computing (SoCC '13)*, San Jose, CA, USA, October 2013.
[46] Yanpei Chen, Archana Ganapathi, Rean Griffith, and Randy Katz. The Case for Evaluating MapReduce Performance Using Workload Suites. In *IEEE International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '11)*, Singapore, July 2011.
[47] Yanpei Chen, Sara Alspaugh, and Randy Katz. Interactive Analytical Processing in Big Data Systems: A Cross-Industry Study of MapReduce Workloads. *VLDB Endowment*, 5(12), August 2012.
[48] Zujie Ren, Xianghua Xu, Jian Wan, Weisong Shi, and Min Zhou. Workload Characterization on a Production Hadoop Cluster: A Case Study on Taobao. In *IEEE International Symposium on Workload Characterization (IISWC)*, Portland, OR, Sep. 2013.
[49] John Wilkes. More Google cluster data. Google research blog, November 2011. Posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.
[50] SWIMProjectUCB. Workloads repository. https://github.com/SWIMProjectUCB/SWIM/wiki/Workloads-repository, 2019.
[51] 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.
[52] Robert A. Jacobs. Methods for Combining Experts' Probability Assessments. *Neural Computation*, 7(5):867–888, 1995.
[53] In Kee Kim, Wei Wang, Yanjun Qi, and Marty Humphrey. CloudInsight: Utilizing a Council of Experts to Predict Future Cloud Application Workloads. In *IEEE International Conference on Cloud Computing (CLOUD '18)*, SF, CA, USA, July 2018.
[54] Alexandru Iosup, Hui Li, Mathieu Jan, Shanny Anoep, Catalin Dumitrescu, Lex Wolters, and Dick H.J. Epema. The Grid Workloads Archive. *Future Generation Comp. Sys.*, 24(7):672–686, 2008.
[55] Erik-Jan van Baaren. WikiBench: A Distributed, Wikipedia based Web App. Benchmark. *MS Thesis, VU Univ. Amsterdam*, 2009.
[56] In Kee Kim, Wei Wang, Yanjun Qi, and Marty Humphrey. Empirical Evaluation of Workload Forecasting Techniques for Predictive Cloud Resource Scaling. In *IEEE International Conference on Cloud Computing (CLOUD '16)*, 2016.
[57] Christoph Fehling, Frank Leymann, Ralph Retter, Walter Schupeck, and Peter Arbittera. Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications. 2014.
[58] Yu Hen Hu, Surekha Palreddy, and Willis J. Tompkins. A Patient-Adaptable ECG Beat Classifier Using a Mixture of Experts Approach. *IEEE Trans. on Biomedical Engineering*, 44(9):891–900, 1997.
[59] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The Element of Statistical Learning: Data Mining, Inference, and Prediction. 2011.
[60] James Bergstra, Remi Bardenet, Yoshua Bengio, and Balazs Kegl. Algorithms for Hyper-Parameter Optimization. In *Neural Information and Processing Systems (NIPS)*, Granada, Spain, Dec. 2011.
[61] The Grid Workloads Archive. http://gwa.ewi.tudelft.nl, 2019.
[62] WikiBench. http://www.wikibench.eu, 2019.
[63] 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.
[64] Jingqi Yang, Chuanchang Liu, Yanlei Shang, Zexiang Mao, and Junliang Chen. Workload Predicting-Based Automatic Scaling in Service Clouds. In *IEEE International Conference on Cloud Computing (CLOUD '13)*, Santa Clara, CA, USA, June 2013.
[65] Peter Bodik, Rean Griffith, Charles A. Sutton, Armando Fox, Michael I. Jordan, and David A. Patterson. Statistical Machine Learning Makes Automatic Control Practical for Internet Datacenters. In *USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '09)*, Santa Diego, CA, USA, June 2009.
[66] 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)*, Santa Diego, CA, June 2010.
[67] Abhishek Chandra, Weibo Gong, and Prashant Shenoy. Dynamic Resource Allocation for Shared Data Centers Using Online Measurements. In *International Conference on Quality of Service (IWQoS '03)*, Monterey, CA, USA, June 2003.
[68] Peter A. Dinda and David R. O'Hallaron. Host Load Prediction using Linear Models. *Cluster Computing*, 3(4):265–280, 2000.
[69] Ali Yadavar Nikravesh, Samuel A. Ajila, and Chung-Horng Lung. Towards an Autonomic Auto-Scaling Prediction System for Cloud Resource Provisioning. In *Int'l Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '15)*, Firenze, Italy, May 2015.
[70] Akindele A. Bankole and Samuel A. Ajila. Cloud Client Prediction Models for Cloud Resource Provisioning in a Multitier Web Application Environment. In *IEEE Int'l Symposium on Service-Oriented System Engineering (SOES '13)*, Firenze, Italy, May 2013.
[71] Wikipedia. https://www.wikipedia.org/, 2019.
[72] Wikipedia german page. https://de.wikipedia.org/, 2019.
[73] Wikipedia japanese page. https://ja.wikipedia.org/, 2019.
[74] Grid5000. https://www.grid5000.fr, 2019.
[75] NorduGrid. http://www.nordugrid.org, 2019.
[76] AuverGrid. http://www.auvergrid.fr, 2019.
[77] SHARCNet. https://www.sharcnet.ca, 2019.
[78] CERN. LHG Computing Grid. http://wlcg.web.cern.ch, 2019.
[79] Ming Mao and Marty Humphrey. A Performance Study on the VM Startup Time in the Cloud. In *IEEE International Conference on Cloud Computing (CLOUD '12)*, Honolulu, HI, USA, 2012.
[80] In Kee Kim, Wei Wang, and Marty Humphrey. PICS: A Public IaaS Cloud Simulator. In *IEEE International Conference on Cloud Computing (CLOUD '15)*, New York, NY, USA, 2015.
[81] JJacob R. Gardner, Matt J. Kusner, Zhixiang Eddie Xu, Kilian Q. Weinberger, and John P. Cunningham. Bayesian Optimization with Inequality Constraints. In *31th International Conference on Machine Learning (ICML 2014)*, Beijing, China, June 2014.
[82] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *33nd International Conference on Machine Learning (ICML 2016)*, NYC, NY, June 2016.
[83] Sysdig. https://sysdig.com/, 2020.
[84] cadvisor. https://github.com/google/cadvisor, 2020.
[85] Prometheus. https://prometheus.io/, 2020.
[86] 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):155–162, 2012.
[87] Sheng Di, Derrick Kondo, and Walfredo Cirne. Host Load Prediction in a Google Compute Cloud with a Bayesian Model. In *International Conference on High Performance Computing Networking, Storage and Analysis (SC)*, Salt Lake City, UT, November 2012.
[88] Arijit Khan, Xifeng Yan, Shu Tao, and Nikos Anerousis. Workload Characterization and Prediction in the Cloud: A Multiple Time Series Approach. In *IEEE Network Operations and Management Symposium (NOMS '12)*, Maui, Hi, USA, April 2012.
[89] 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 '13)*, Prague, Czech Republic, April 2013.

[90] Yexi Jiang, Chang-Shing Perng, Tao Li, and Rong N. Chang. ASAP: A Self-Adaptive Prediction System for Instant Cloud Resource Demand Provisioning. In *IEEE Int'l Conference on Data Mining (ICDM '11)*, Vancouver, BC, Canada, December 2011.
[91] Joao Loff and Joao Garcia. Vadara: Predictive Elasticity for Cloud Applications. In *IEEE International Conference on Cloud Computing Technology and Science (CloudCom '14)*, Singapore, December 2014.
[92] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep Learning. *Nature, 521*, 521, 2015.
[93] Chanh Nguyen, Cristian Klein, and Erik Elmroth. Multivariate LSTM-Based Location-Aware Workload Prediction for Edge Data Centers. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID '19*, Larnaca, Cyprus, May 2019.
[94] Siddhant Kumar, Neha Muthiyan, Shaifu Gupta, Dileep A.D., and Aditya Nigam. Association Learning based Hybrid Model for Cloud Workload Prediction . In *IJCNN '18*.
[95] Xiaoyong Tang. Large-Scale Computing Systems Workload Prediction Using Parallel Improved LSTM Neural Network. *IEEE Access*, 7, 2019.
[96] Jing Bi, Shuang Li, Haitao Yuan, Ziyan Zhao, and Haoyue Liu. Deep Neural Networks for Predicting Task Time Series in Cloud Computing Systems. In *ICNSC '19*.
[97] Qingchen Zhang, Laurence T. Yang, Zheng Yan, Zhikui Chen, and Peng Li. An Efficient Deep Learning Model to PredictCloud Workload for Industry Informatics. *IEEE Trans. on Industrial Informatics*, 14, 2018.
[98] Hoang Minh Nguyen, Gaurav Kalra, and Daeyoung Kim. Host load prediction in cloud computing using Long Short-Term Memory EncoderDecoder. *Journal of Super Computing*, 2019.
[99] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation, 9(8)*, 1997.
[100] 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 4th International Conference on Utility and Cloud Computing (UCC 2011), Melbourne, Australia, December 5-8*, pages 397–402, 2011.

In Kee Kim received a Ph.D. in Computer Science from University of Virginia in 2018. He is currently an Assistant Professor at the Department of Computer Science of the University of Georgia. His research areas include cloud computing, large-scale distributed systems, IoT/edge computing, and machine learning systems. He is a member of the IEEE.

Wei Wang holds a Ph.D. in computer science from University of Virginia in 2015. He is currently an Assistant Professor at the Computer Science Department of the University of Texas at San Antonio. His research interests include system software, cloud computing, computer architecture and software engineering. He is a member of the IEEE.

Yanjun Qi is an assistant professor at University of Virginia, Department of Computer Science from summer 2013. Her research interests are in machine learning, data mining, and passionate in applying learning techniques to real-world problems of significant biomedical impacts. She obtained her Ph.D. degree from School of Computer Science at Carnegie Mellon University in 2008 and her Bachelor with high honors from Computer Science Department at Tsinghua University, Beijing. She was a researcher in the Machine Learning Department at NEC Labs America, Princeton, NJ from 2008 to 2013. She has served as PCs and reviewers for multiple renowned international conferences/journals and has co-chaired the NIPS Machine Learning for Computational Biology workshops. Dr. Qi has received CAREER award from NSF, a Best Paper Award at NIPS workshop for Transparent and interpretable Machine Learning and a Best Paper Award at International Conference on BodyNet.

Marty Humphrey is an Associate Professor in the Department of Computer Science at the University of Virginia. He received a B.S. and M.S. degree in Electrical Engineering from Clarkson University in 1986 and 1989, respectively. He received his Ph.D. degree in Computer Science from the University of Massachusetts in 1996. From 1996-1998, he was an Assistant Professor of Computer Science and Engineering at the University of Colorado at Denver. From 1998-2002, he was a Research Assistant Professor at UVA. He has co-authored over 80 publications and has been a principal investigator on a number of projects funded through government agencies (such as the National Science Foundation and the Department of Energy) and private sector (such as Sun Microsystems and Microsoft Corporation).