IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 2 July 2013; revised 3 January 2014; accepted 10 January 2014. Date of publication 7 April 2014;
date of current version 30 July 2014.

Digital Object Identifier 10.1109/TETC.2014.2310455

Ensemble Learning for Large-Scale Workload Prediction

NIDHI SINGH, (Member, IEEE), AND SHRISHA RAO, (Senior Member, IEEE)

International Institute of Information Technology-Bangalore (IIIT-B), Bangalore 560100, India
CORRESPONDING AUTHOR: N. SINGH (nidhi.s.singh@ieee.org)

ABSTRACT Increasing energy costs of large-scale server systems have led to a demand for innovative methods for optimizing resource utilization in these systems. Such methods aim to reduce server energy consumption, cooling requirements, carbon footprint, and so on, thereby leading to improved holistic sustainability of the overall server infrastructure. At the core of many of these methods lie reliable workload-prediction techniques that guide in identifying servers, time intervals, and other parameters that are needed for building sustainability solutions based on techniques like virtualization and server consolidation for server systems. Many workload prediction methods have been proposed in the recent paper, but unfortunately they do not deal adequately with the issues that arise specifically in large-scale server systems, viz., extensive nonstationarity of server workloads, and massive online streaming data. In this paper, we fill this gap by proposing two online ensemble learning methods for workload prediction, which address these issues in large-scale server systems. The proposed algorithms are motivated from the weighted majority and simulatable experts approaches, which we extend and adapt to the large-scale workload prediction problem. We demonstrate the effectiveness of our algorithms using real and synthetic data sets, and show that using the proposed algorithms, the workloads of 91% of servers in a real data center can be predicted with accuracy > 89%, whereas using baseline approaches, the workloads of only 13%–24% of the servers can be predicted with similar accuracy.

INDEX TERMS Server workload prediction, sustainable server systems, sustainable computing, ensemble-based learning, machine learning.

I. INTRODUCTION

Growing energy consumption by large-scale server systems like data centers and server farms has caused increased interest in innovative methods for improving utilization of resources in these systems. Towards this goal, various methods based on virtualization, server consolidation, predictive capacity management, and similar, are proposed in recent literature [1]–[6]. These methods lead to improved sustainability of IT infrastructure (considered holistically) in many ways, such as by reducing server energy consumption, cooling requirements, and carbon footprints. Most of these methods require analyzing historical workload data of servers and predicting near-future workloads. To illustrate, consider a scenario where a server consolidation solution needs to be developed for a data center to reduce the energy costs and improve sustainability of some data center. In order to develop this solution, it must be determined which servers can be safely consolidated without disrupting operations of end-users. This can be done only by analyzing each server's historical workload and predicting its workload, with servers predicted to have low workloads in the near future being chosen for consolidation. Similarly, other sustainability solutions based on techniques like virtualization and dynamic capacity provisioning can be developed for large-scale systems using robust workload predictions.

Various approaches [7]–[10] have been proposed in the literature for predicting server workloads, but these approaches do not adequately deal with issues that arise specifically in large server systems. In this paper, we bridge this gap by addressing the problem of server workload prediction specifically for systems comprising large numbers of servers. We focus on two key problems in workload prediction for such systems, i.e., large-scale non-stationarity of workloads, and massive online data streams. We categorize workload non-stationarity into non-stationarity over time and non-stationarity across servers. *Non-stationarity of workload over*

time may be said to occur when workload patterns for a given server change unevenly with time (which may be due to reasons like changes in the ownership of servers, changes in the application profiles of servers, etc.). These kinds of changes are non-observable from the workload prediction perspective, and hence are difficult to capture in workload prediction models. *Non-stationarity of workloads across servers* may be said to occur when workload patterns in a server system differ greatly across servers due to differences in servers' utilization patterns, with the result that it is difficult to generalize one (or a few) workload prediction models over all servers in a large server system.

In a large-scale server infrastructure, these kinds of workload non-stationarities are observed on multitudes of servers, which makes it difficult to use statistical time-series based prediction models. This is so because each time the workload of a server exhibits non-stationarity, the time-series based prediction model needs to be re-trained to update the parameters so that the recent changes in workload patterns are taken into consideration. The retraining process of a prediction model is cumbersome and computationally expensive, especially if it requires some kind of manual intervention (e.g., for servers hosting critical applications). In large server infrastructures where workloads of multitudes of servers exhibit non-stationarity over time and/or across servers, it is not feasible to frequently retrain individual prediction models for many servers, due to the computational cost, service disruptions, and/or manual interventions required to perform such frequent retraining.

Massive online data streams adds another layer of complexity to the workload prediction problem, as it requires a prediction model to be capable of *incrementally* learning from every additional data point in a way that is fast as well as computationally efficient.

To address the aforementioned issues, we propose two ensemble methods for the large-scale workload prediction problem. These methods enable workload predictions with significantly high accuracies in scenarios where multitudes of servers exhibit extensive non-stationarities of workloads over time and/or across servers. Also, the proposed methods compute workload predictions in an online and computationally efficient way, so that massive scales of data do not as adversely influence the predictive performances of these methods. These capabilities make the proposed methods suitable for practical use in various solutions for server infrastructure management (e.g., dynamic capacity provisioning [11], [12] predictive server consolidation [13]–[15]) that would, in turn, lead to improved sustainability of server systems by reducing the total energy consumption by servers and cooling equipment, improving carbon footprint, reducing floor space requirements, and the like.

The two ensemble methods that we extend and adapt to workload prediction are Weighted Majority [16] and Simulatable Experts [17]. In ensemble methods, a set (or *ensemble*) of prediction models (or *experts*) is formed and their predictions are compounded in a well-defined way to yield a final prediction. Such ensemble methods for the large-scale workload prediction problem offer three advantages.

1) Using these methods, workload prediction of all servers in a server system can be done using one *global* ensemble of experts (where the number of experts in the ensemble is much smaller than the number of servers), thereby avoiding the need to create and fit a separate prediction model for each server.
2) The ensemble model parameters are updated in an incremental and computationally-efficient way which makes these algorithms well-suited to handling massive online data streams.
3) Unlike conventional time-series forecasting methods where only one prediction model is used, these ensemble learning algorithms use a set of prediction models for computing workload predictions, and hence are more consistent in predictive performance.

We experimentally evaluated the proposed workload prediction algorithms using large and real datasets of an enterprise data center, in which we took into consideration the workload datasets of 1570 actively utilized servers. We also used synthetic datasets in order to highlight important performance aspects of the proposed algorithms. Our experiments reveal that the proposed workload prediction approach achieves 91.63–91.95% accuracy, whereas baseline workload prediction algorithms achieve accuracies of only 69.3–78.3%, which indicates a gain of 13.65–22.3% in prediction accuracy by the proposed algorithms. It is also seen that using the proposed algorithms, the workloads of approximately 91% of the servers in the data center could be predicted with accuracies greater than 89%, whereas using baseline algorithms, the workloads of hardly 13–24% of the servers can be predicted with similar accuracy. Furthermore, we observed that increases in the number of servers do not adversely affect the predictive performance of the proposed algorithms. These evaluations demonstrate the efficacy of the proposed approaches in predicting workloads in large-scale server systems.

The rest of this paper is organized in the following manner. In Section II, we briefly review state-of-the-art methods for workload prediction and ensemble methods, and also describe the Simulatable Experts algorithm. Thereafter, we introduce the large-scale workload prediction problem in Section III. We present the proposed workload prediction approaches based on Weighted Majority and Simulatable Experts, in Section IV and Section V respectively. We experimentally evaluate the proposed workload prediction algorithms in Section VI, and conclude our work in Section VII.

II. BACKGROUND AND RELATED WORK

In this section, we review the literature in workload prediction, and briefly describe ensemble methods and the Simulatable Experts algorithm [17] which we extend to the workload prediction problem.

A. WORKLOAD CHARACTERIZATION AND PREDICTION

In the past, workload analysis and prediction for server systems and real-time applications has been attempted in many ways [10], [18]–[25]. Prediction models for dealing specifically with bursty or chaotic workloads have also been proposed and developed [4], [26], [27]. A large body of work also exists for workload prediction in grid and cloud computing environments. For instance, there are methods [5], [28]–[32] for predicting workloads of servers or applications in grid environments. In similar vein, there also are methods [2], [3] for resource prediction and allocation in cloud computing environments. However, none of these models and frameworks handle extensive non-stationarities of workloads, especially the non-stationarity across servers that is seen in large server systems. Thus, these models need retail, per-server fitting for overall predictions, which is infeasible in case of large systems. Also, the performance and computational efficiencies of these methods have not been assessed for massive online streaming data and their practical applicability in large scale systems thus remains unknown.

Many energy optimization techniques such as virtualization, server consolidation, and optimal workload placement have been built in recent years for server systems [1], [33]–[43]. We believe that these energy-optimization techniques for server systems can significantly benefit from our workload prediction models by better estimating servers' utilizations in the near-future, which in turn enables effective decision-making on energy-optimization measures and policies.

B. ENSEMBLE METHODS

In ensemble methods, a set (or *ensemble*) of prediction models is used to predict future outcomes in a problem domain. Each prediction model in the ensemble is referred to as an *expert*. The experts' predictions are compounded in a well-defined way to yield a final or *ensemble prediction*. It has been shown that the predictive performance of an ensemble method is much better than that of any of its constituent predictive models (or experts). Some of the oldest and yet most popular *off-line* ensemble methods are bagging [44] and boosting [45]. *Online* ensemble methods learn from one instance at a time, whereas offline methods require a batch of instances. A comprehensive survey of ensemble methods may be seen elsewhere [46], [47].

Online ensemble methods that can handle non-stationarity in problem environments, such as proposed in [16], [17],and [48]–[51], are most relevant in our context. In this paper, we extend and adapt two of these ensemble methods, viz., Weighted Majority [16] and Simulatable Experts [17], to deal with the issues of extensive non-stationarity and massive online streaming data that are faced in large-scale workload prediction. Also related to this work are prior papers [52] and [53] which briefly describe how Weighted Majority can be used in workload prediction, and the computational efficiency that can be achieved thereby. In this paper, we elaborate on the applicability of Weighted Majority to workload prediction quantitatively as well as qualitatively, and in addition also provide a theoretical extension of the Simulatable Experts model and improve its usefulness in large-scale workload prediction scenarios.

C. SIMULATABLE EXPERTS FOR BINARY OUTCOME SPACE

In this section, we describe the Simulatable Expert (SE) [17] algorithm which we later extend (in Section V) to suit large-scale workload prediction. In this algorithm, an ensemble of *simulatable experts* is used to predict future outcomes. A *simulatable expert* is a prediction model that can be simulated over any sequence of (hypothetical) data in order to compute its future predictions over that sequence of data. The SE algorithm *simulates* such experts in order to hypothesize about future outcomes and reduce overall prediction loss. In this algorithm, the ensemble predictions are assumed to be in the range [0, 1], but the actual outcomes are assumed to be binary, i.e., they are restricted to {0, 1}. The set of all possible actual outcomes is called an *outcome space* and is denoted by $\mathcal{Y} = \{0, 1\}$. (Since the actual outcomes correspond to server workloads in our prediction problem, the usual formulation of SE is not applicable to the workload prediction problem. Hence, in Section V, we extend SE to k-outcome space.)

Let P_{n-1} denote a string of the past $n-1$ actual outcomes. Then, since the outcome space is binary, the sequence of possible outcomes till n time steps can be represented as either $P_{n-1}0$ which denotes a string of n outcomes, with the first $n-1$ outcomes represented by P_{n-1} and the last outcome as 0; or $P_{n-1}1$ which denotes a string of n outcomes, with the first $n-1$ outcomes represented by P_{n-1} and the last outcome as 1.

Let $\mathcal{L}_s(P_n)$ denote the cumulative absolute loss of ensemble $\mathcal{E}$ for server s at time step n, which is defined as:

$$\mathcal{L}_s(P_n) = \sum_{t=1}^{n} |\widehat{p}_{s,t} - p_{s,t}| \tag{1}$$

where $\widehat{p}_{s,t}$ denotes the ensemble workload prediction for server s at time t, and $p_{s,t}$ denotes the actual workload for server s at time t. (Hereafter, we use the terms *loss* and *prediction error* interchangeably.)

Also, let $L_s^{(e)}(P_n)$ denote the cumulative absolute loss of expert e for server s at time step n, which is defined as:

$$L_s^{(e)}(P_n) = \sum_{t=1}^{n} |x_{s,t}^{(e)} - p_{s,t}| \tag{2}$$

where $x_{s,t}^{(e)}$ denotes the workload prediction of expert e for server s at time step t.

Now, given an ensemble $\mathcal{E}$ of simulatable experts, our objective is to find an optimal prediction at time step n such that the following *worst-case regret* $V_{s,n}(\mathcal{E})$ for server s is minimized:

$$V_{s,n}(\mathcal{E}) = \sup_{P_n \in \{0,1\}^n} (\mathcal{L}_s(P_n) - \min_{e \in \mathcal{E}} L_s^{(e)}(P_n)) \tag{3}$$

In order to minimize worst-case regret $V_{s,n}(\mathcal{E})$, we first minimize the following:

$$\max \begin{cases} \mathcal{L}_s(P_{n-1}) + \ell(\widehat{p}_{s,n}, 0) - \inf\limits_{e\in\mathcal{E}} L_s^{(e)}(P_{n-1}0), \\ \mathcal{L}_s(P_{n-1}) + \ell(\widehat{p}_{s,n}, 1) - \inf\limits_{e\in\mathcal{E}} L_s^{(e)}(P_{n-1}1) \end{cases} \tag{4}$$

where $\ell(\widehat{p}_{s,n}, 0)$ and $\ell(\widehat{p}_{s,n}, 1)$ are absolute losses computed as $|\widehat{p}_{s,n} - 0|$ and $|\widehat{p}_{s,n} - 1|$ respectively. Here, $P_{n-1}0$ is a string of n outcomes, with the first $(n-1)$ outcomes as P^{n-1} and the last (hypothetical) outcome as 0. So, in order to compute $L_s^{(e)}(P_{n-1}0)$ (or $L_s^{(e)}(P_{n-1}1)$), we simulate the experts in the ensemble over a *hypothetical* outcome of 0 (or 1) at time step n, so as to minimize the worst-case regret.

Now, minimizing (4) is equivalent to minimizing the following:

$$\max \begin{cases} \widehat{p}_{s,n} - \inf\limits_{e\in\mathcal{E}} L_s^{(e)}(P_{n-1}0), \\ 1 - \widehat{p}_{s,n} - \inf\limits_{e\in\mathcal{E}} L_s^{(e)}(P_{n-1}1) \end{cases} \tag{5}$$

If we define $A_n(P_n)$ by:

$$A_n(P_n) = -\inf_{e\in\mathcal{E}} L_s^{(e)}(P_n), \tag{6}$$

and $X_{1,0}^{(n-1)}$ by:

$$X_{1,0}^{(n-1)} = \frac{A_n(P_{n-1}1) - A_n(P_{n-1}0) + 1}{2}, \tag{7}$$

then the optimal prediction $\widehat{p}_{s,n}$ for server s at time step n, that minimizes the worst-case regret $V_{s,n}(\mathcal{E})$, can be computed as follows:

$$\widehat{p}_{s,n} = \begin{cases} 0 & \text{if } A_n(P_{n-1}0) > A_n(P_{n-1}1) + 1, \\ 1 & \text{if } A_n(P_{n-1}0) + 1 < A_n(P_{n-1}1), \\ X_{1,0}^{(n-1)} & \text{otherwise} \end{cases} \tag{8}$$

III. LARGE SCALE WORKLOAD PREDICTION PROBLEM

Large-scale workload prediction poses challenges concerning extensive non-stationarity in workloads and massive online streaming data, that are not seen in case of workload predictions for small-scale server infrastructures. Due to these challenges, the applicability of conventional workload prediction methodologies, which do perform well on small scales, becomes very limited in large-scale server systems. We proceed by describing these challenges in detail in the following sections.

A. NONSTATIONARITY OF WORKLOAD OVER TIME

Non-stationarity of workload over time may be said to occur when the workload of a server changes unpredictably with time, due to reasons like changes in the ownership of servers, changes in the dominant applications of servers (like a web server reconfigured as a database server), etc. Such changes are non-observable from the workload prediction perspective and can be interpreted as hidden context in the workload prediction problem. Any change in this hidden context, in turn, induces a change in the underlying distribution of server workloads, which leads to non-stationarity in server workloads over time.

In small-scale server systems, this kind of non-stationarity can be handled by cherry-picking those servers that exhibit such workload non-stationarity, and then retraining only their workload prediction models. However, such retraining of prediction models is computationally expensive, which makes this process infeasible for large-scale server infrastructures, where multitudes of servers may exhibit workload non-stationarity.

Example 1: We illustrate this phenomenon using sample workloads of three servers, s_1, s_2 and s_3, displayed in Fig. 1 wherein we plot time steps on the x-axis and server workloads on the y-axis. For server s_1, we see that initially, from time t_0 to t_{10}, the workload follows a gaussian distribution, denoted by $\mathcal{N}_1$, with mean as 5 and variance as 2. At t_{10}, due to changes in the hidden context, the server utilization increases such that until t_{20}, the corresponding distribution $\mathcal{N}_2$, though still gaussian, shifts to have a mean of 20 and variance of 15. At t_{20}, the underlying distribution of server workload again undergoes a change so that it now follows a beta distribution $\mathcal{B}$. For server s_2, the workload exhibits similar non-stationarity at time step t_5 and t_{20}, whereas for server s_3, the underlying distribution of workload changes from $\mathcal{B}$ to $\mathcal{N}_2$ at time step t_{10}, and from $\mathcal{N}_2$ to $\mathcal{N}_1$ at time step t_{20}. This scenario indicates that in order to predict workloads for servers s_1, s_2 and s_3, retraining of prediction models would have to be done at different times for each server, which clearly becomes a cumbersome task as the number of servers under consideration increases.

It is of note that though certain workload non-stationarity over time (e.g., seasonal or cyclical) may be handled using time-series based prediction models like ARIMA, other non-stationarity such as caused by changes of underlying distributions (as from $\mathcal{N}_2$ to $\mathcal{B}$ in Example 1) cannot be sufficiently approximated using such models.

Our proposed ensemble-based workload prediction models address this issue of extensive non-stationarity of workloads over time, by adapting to the non-stationarity through automatic update of its learning parameters, thereby avoiding any need for re-training of ensemble prediction models and/or manual intervention.

It is also of note that online learning problems in general, especially those that involve predictions of time-series data, are difficult. An important reason for this is that time-series data are generally non i.i.d. [54], so standard tools such as gradient methods (which require the i.i.d. assumption) do not work with them. Large-scale workload prediction is an instance of online learning, and it is further known that server workloads are characterized by data streams that are chaotic and change frequently with time [4], [27]. Hence also, classical machine-learning methods with the i.i.d. assumption do not work in this domain, and none of our methods make the assumption.

TABLE 1. Notations used in WMC, SE and kSE prediction models.

	Common notations
$\mathcal{E}$	Global ensemble of experts
S	Set of servers
s	Server in given set of servers
$\mathcal{E}$	Global ensemble of experts
e	Expert in global ensemble
t, n	Time steps
$D_T^{(s)}$	Subset of $\mathcal{D}_T$ containing workload data of single server s for past T time steps
T	Total number of time steps in training workload dataset $\mathcal{D}_T$
$\mathcal{D}_T$	Training dataset containing workload of all servers under consideration for past T time steps
$\mathcal{N}_1, \mathcal{N}_2$	Gaussian distributions from which sample server workload is drawn
$\mathcal{B}$	Beta distribution from which sample server workload is drawn
$x_{s,t}^{(e)}$	Workload prediction of expert e for server s at time t
$\widehat{p}_{s,t}$	Final or ensemble workload prediction for server s at time t
$p_{s,t}$	Actual workload of server s at time t
	WMC-specific notations
M	Number of experts in global ensemble $\mathcal{E}$
$w_{s,t}^{(e)}$	Weight of expert e in the ensemble for server s at time t
β	Fixed parameter between 0 and 1
F_s	Update factor which determines the rate at which experts' weights are updated for server s
$Q_{s,n}^{(e)}$	Total loss of expert e for server s till time step n
$\mathcal{Q}_{s,n}$	Total loss of the ensemble for server s till time step n
	SE and kSE-specific notations
$\mathcal{Y}$	Outcome space
P_n	String of outcomes till time step n
$P_n^{n-\alpha}$	String of past α outcomes from time step $(n-\alpha)$ to time step n
$P_n^{n-\alpha}y_i$	String of $\alpha+1$ outcomes with first α outcomes as $P_n^{n-\alpha}$ and last (hypothetical) outcome as y_i
$\mathcal{L}_s(P_n)$	Cumulative absolute loss of ensemble $\mathcal{E}$ for server s at time step n, given string of past outcomes P_n
$L_s^{(e)}(P_n)$	Cumulative absolute loss of expert e for server s at time step n, , given string of past outcomes P_n
$V_{s,n}(\mathcal{E})$	Worst-case regret of ensemble $\mathcal{E}$ for server s at time step n

B. NONSTATIONARITY OF WORKLOADS ACROSS SERVERS

Another kind of workload non-stationarity seen in server systems is *non-stationarity across servers*, which occurs when the utilization patterns differ across servers causing the workloads to vary from one server to another. Due to this, a prediction model that is effective for one server may not be so for another. Furthermore, even if a prediction model does seem to hold for different servers, its parameters may differ for each server and hence may need to be hand-tuned individually for each server.

Example 2: As an illustration, consider two different servers s_1 and s_2, each of which has workload that is strongly auto-correlated. Assume that the workload of s_1 has significant positive auto-correlation with lag 4, and the workload of s_2 has significant positive auto-correlation with lag 7. This implies that autoregressive models [55] serve as good prediction models for s_1 and s_2, but the parameters, or more specifically, the order of the autoregressive models, are different for each server, i.e., 4 for s_1 and 7 for s_2. Consider another server s_3 for which a support vector regression model [56] fits the workload data best. In this example scenario, different prediction models need to be identified and fitted for servers s_1, s_2 and s_3 due to non-stationarities in workloads across servers. This is clearly a difficult task, especially in large systems where this must be done for multitudes of servers.

In the proposed ensemble-based workload prediction models, we create a *global* ensemble of experts for the entire server system, with the number of experts being significantly less than the number of servers. Using this global ensemble, the proposed prediction models would predict the workload of each server by compounding workload predictions of experts in the ensemble, thereby obviating the use of a separate prediction model for each server in large server systems.

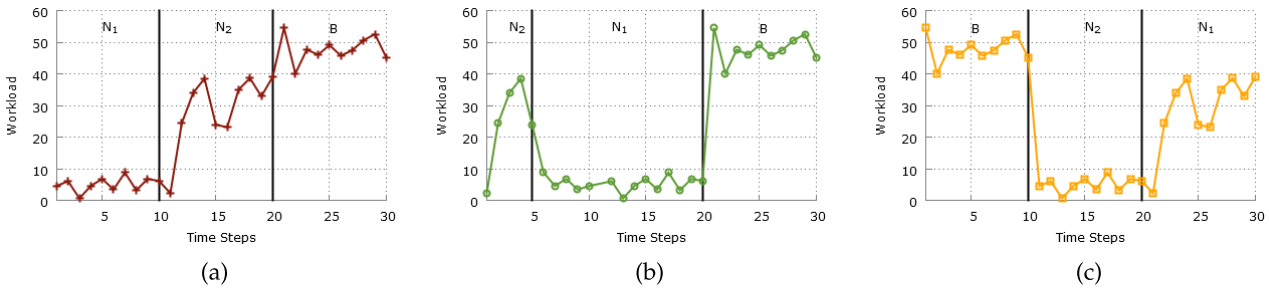


FIGURE 1. Sample workload demonstrating non-stationarity in workload over time due to changes in the underlying distribution. (a) Workload of server s_1. (b) Workload of server s_2. (c) Workload of server s_3.

C. MASSIVE ONLINE STREAMING DATA

With increases in the number of servers, the scale of the workload data increases at a rapid pace and continues to grow with time. For instance, consider a server system with 10000 servers, wherein for each server, a workload data point in collected by software/monitoring agents every 5 minutes. In this case, $(60/5)\times 24\times 10000 = 2.8 \times 10^6$ data points are collected every day in the system. It is easy to see that in such a system, the size of the workload dataset would reach a truly large size within a very short time.

Learning from this kind of massive online streaming data requires prediction models that satisfy the following two conditions:

1) The learning and prediction methods must be fast and computationally efficient, so that increasing sizes of streaming workload datasets does not negatively affect the performance of the prediction model.
2) Learning/update of model parameters, if needed, must be done in an incremental or online way on receiving additional streams of workload data.

The proposed ensemble-based workload prediction models meets both conditions, and hence are well-suited to the large-scale workload prediction problem.

IV. LARGE SCALE WORKLOAD PREDICTION USING WEIGHTED MAJORITY BASED ENSEMBLE LEARNING

In this section, we propose an online workload prediction approach based on ensemble learning, which addresses the issues of extensive workload non-stationarity and massive online data in the large-scale workload prediction problem. Specifically, we adapt a variant of the Weighted Majority algorithm [16], called as *WMC*, to workload prediction (Section IV-A), specify its error bounds (Section IV-B), show how it can be efficiently implemented for large-scale workload prediction (Section IV-C), and discuss the advantages of using this algorithm for our purpose (Section IV-D).

A. WMC ALGORITHM

The Weighted Majority variant that we adapt to the workload prediction problem is referred to as *WMC* [16]. One of the key features of WMC that makes it particularly suitable for handling non-stationarity is that it does not assume the data to be independently and identically distributed [16], [54]. It is due to this feature that non-stationarity scenarios such as those described in Example 1 can be easily handled by WMC. The WMC algorithm consists of three phases: building a global ensemble of experts; compounding experts' prediction using an ensemble combination rule; and incremental parameters/weights update—each of these is described in the following subsections.

1) BUILDING A GLOBAL ENSEMBLE OF EXPERTS

This is the first phase of WMC, in which we form a fixed-size *global ensemble* $\mathcal{E}$ of experts for a large-scale server system. As the name suggests, a global ensemble of experts is a common ensemble for all servers in the system. Experts are chosen in the global ensemble by analyzing servers' historical workload data, and identifying salient statistical properties of servers' workloads in the server system. For example, if workloads of significant fractions of servers show distinct increasing or decreasing trends, then experts could be built based on exponential smoothing methods [55] with different parameters. Such experts based on exponential smoothing methods would be capable of tracking any recent trends in workloads. In cases where workloads of significant fractions of servers exhibit complicated seasonal and/or non-seasonal patterns, appropriate ARIMA (Autoregressive Integrated Moving Average) models [55] could be identified after doing necessary data transformations like logging and differencing, and thereafter the experts could be created based on these models, with different statistical parameters. A fixed number M of experts are chosen in this way based on salient statistical properties of servers' workloads in the system and/or prior knowledge about workload patterns of servers, if available.

2) COMPOUNDING EXPERTS' PREDICTIONS USING ENSEMBLE COMBINATION RULE

After forming the global ensemble $\mathcal{E}$ consisting of M experts, we proceed to use this ensemble to predict the workload of each server in a large-scale server system. In order to do so, for each server s in the system, we associate a weight $w_{s,t}^{(e)}$ with each expert e in the global ensemble at time t. This weight indicates the accuracy of expert e in predicting the workload

of server s at time t, and is initially set equally (i.e., as $1/M$) for each expert in $\mathcal{E}$.

For server s, expert e in the ensemble provides workload prediction $x_{s,t}^{(e)}$ at time step t. These expert workload predictions are compounded using the following ensemble combination rule of WMC in order to compute the final or ensemble workload prediction $\widehat{p}_{s,t}$:

$$\widehat{p}_{s,t} = \frac{\sum_{e\in\mathcal{E}} w_{s,t}^{(e)} x_{s,t}^{(e)}}{\sum_{e\in\mathcal{E}} w_{s,t}^{(e)}} \tag{9}$$

In (9), the ensemble workload prediction $\widehat{p}_{s,t}$ for server s at time step t is computed as the weighted average of the workload predictions of all M experts in the ensemble. In this way, we compute an ensemble workload prediction for each server under consideration. Recall that for each server s, we had initialized the weight $w_{s,t}^{(e)}$ associated with expert e in the global ensemble as $1/M$. In the next phase, we proceed to update these experts' weights for server s based on the accuracy of each expert in predicting the workload of server s.

3) INCREMENTAL PARAMETERS OR WEIGHTS UPDATE

For each server s, after computing the ensemble workload prediction $\widehat{p}_{s,t}$ at time step t, we observe the server's actual workload $p_{s,t}$ and calculate the absolute loss (or prediction error) $|x_{s,t}^{(e)} - p_{s,t}|$ made by each expert e. Using this absolute loss, we determine an update factor F_s for server s that satisfies the following condition:

$$\beta^{|x_{s,t}^{(e)} - p_{s,t}|} \leq F_s \leq 1 - (1-\beta)|x_{s,t}^{(e)} - p_{s,t}| \tag{10}$$

where β is a fixed parameter between 0 and 1. The factor F_s determines the rate at which the weight of an expert e is updated in response to changes in expert's prediction accuracy. Thereafter, for server s, the weight of each expert e is updated in accordance with WMC as follows:

$$w_{s,t+1}^{(e)} = F_s w_{s,t}^{(e)} \tag{11}$$

In (10) and (11), the selection of F_s and multiplication of an expert's weight by factor F_s has the net effect of decreasing the relative weight of a low-performing expert, thereby decreasing its influence on the final or ensemble workload prediction of server s.

The updated weights are then normalized to sum to one, as follows:

$$w_{s,t+1}^{(e)} = \frac{w_{s,t+1}^{(e)}}{\sum_{e\in\mathcal{E}} w_{s,t+1}^{(e)}} \tag{12}$$

In this way, for each server in the server system, we incrementally learn and update the learning parameters, i.e., the weights of experts in the global ensemble. Using these updated parameters, we can then predict the workload of server s for future time steps, and continue to incrementally update the learning parameters of WMC.

It may be appear that WMC is similar to gradient boosting [46], [57], an existing machine learning technique for regression problems, which produces a prediction model in the form of an ensemble of weak prediction models. However, there is an important difference between these methods, viz., gradient boosting assumes the data distribution to be independently and identically distributed (i.i.d) whereas WMC makes no such assumption. The implication of this assumption is that the gradient boosting method cannot handle non-stationarity in data, which is key in our large scale workload prediction problem. Detailed analyses of WMC and gradient methods may be seen elsewhere [58].

B. ERROR BOUNDS

It can be proved that for any value of F_s that satisfies the condition specified in (10), the total prediction error made by WMC for each server s is bounded, as stated in the following theorem of Littlestone and Warmuth (see [16] for a detailed proof).

Theorem 1: Let $\mathcal{E}$ be an ensemble of M experts. For server s, let $Q_{s,n}^{(e)}$ denote the total loss of expert e till n time steps, i.e., $L_s^{(e)} = \sum_{t=1}^{n} |x_{s,t}^{(e)} - p_{s,t}|$ assuming that experts' predictions in the range of [0,1]. Also, let $\mathcal{Q}_{s,n}$ denote the total prediction error made by the ensemble for server s till n time steps, i.e., $\mathcal{Q}_{s,n} = \sum_{t=1}^{n} |\widehat{p}_{s,t} - p_{s,t}|$. Then, for server s, if WMC is applied to ensemble $\mathcal{E}$ with equal initial weights assigned to each expert, then the total prediction error $\mathcal{Q}_{s,n}$ made by the ensemble for server s satisfies the following inequality:

$$\mathcal{Q}_{s,n} \leq \frac{\ln M + Q_{s,n}^{(e)} \ln \frac{1}{\beta}}{1-\beta}, \quad \text{for } 1 \leq e \leq M \tag{13}$$

This theorem implies that using WMC, the total workload prediction errors for server s at time step n never exceed $\left(\frac{\ln M + Q_{s,n}^{(e)} \ln \frac{1}{\beta}}{1-\beta}\right)$.

C. IMPLEMENTATION

We now describe a high-level implementation of the proposed WMC-based workload prediction mechanism in Algorithm 1. The inputs to this algorithm are: a set of servers $S = \{s_1, s_2, \ldots, s_N\}$, a training dataset $\mathcal{D}_T$ containing workload data of each server in S for the past T time steps, and a fixed parameter β such that $0 \leq \beta < 1$. For simplicity of notation, we may omit the subscript of servers and denote a server simply by s (instead of s_1, s_2, and so on).

In Algorithm 1, we analyze training dataset $\mathcal{D}_T$ to identify salient statistical properties and patterns in servers' workloads, based on which we form a global ensemble of experts (lines 1–2). Lines 3–19 describe how ensemble parameters, i.e., the weights of the experts, can be learned for each server s, which can then be used for predicting workloads of server s for future time steps. For each server s, we begin by extracting a subset $D_T^{(s)}$ from training dataset $\mathcal{D}_T$ such that $D_T^{(s)}$ contains workload data of server s for past T time steps, i.e., $D_T^{(s)} = \{p_{s,1}, p_{s,2}, \ldots, p_{s,T}\}$ (Line 4). We initialize the weight $w_{s,0}^{(e)}$ of each expert e for server s as $1/M$ (Line 5–7).

Algorithm 1: WMC Workload Prediction Algorithm

Input: Set of servers S, training dataset $\mathcal{D}_T$, fixed parameter β

1 Perform statistical analysis of training dataset $\mathcal{D}_T$;
2 Construct global ensemble $\mathcal{E}$ of M experts based on statistical analysis results;
3 **foreach** $s \in S$ **do**
4 Extract workload dataset $D_T^{(s)}$ of server s from $\mathcal{D}_T$;
5 **foreach** $e \in \mathcal{E}$ **do**
6 Set initial weight $w_{s,0}^{(e)}$ to $1/M$;
7 **end**
8 **foreach** $t = 1, 2, \ldots, T$ **do**
9 Compute workload prediction $x_{s,t}^{(e)}$ of each expert e;
10 Compute ensemble prediction $\widehat{p}_{s,t} = \frac{\sum_{e \in \mathcal{E}} w_{s,t}^{(e)} x_{s,t}^{(e)}}{\sum_{e \in \mathcal{E}} w_{s,t}^{(e)}}$;
11 Obtain actual server workload $p_{s,t}$ from $D_T^{(s)}$;
12 **foreach** $e \in \mathcal{E}$ **do**
13 Calculate absolute loss as: $|x_{s,t}^{(e)} - p_{s,t}|$;
14 Select F such that the following condition is satisfied: $\beta^{|x_{s,t}^{(e)} - p_{s,t}|} \leq F_s \leq 1 - (1 - \beta)|x_{s,t}^{(e)} - p_{s,t}|$;
15 Update weight of expert e as: $w_{s,t+1}^{(e)} = F_s w_{s,t}^{(e)}$;
16 **end**
17 Compute sum of weights of all experts $\sum_{e \in \mathcal{E}} w_{s,t+1}^{(e)}$;
18 Normalize weight of each expert e as: $w_{s,t+1}^{(e)} = \frac{w_{s,t+1}^{(e)}}{\sum_{e \in \mathcal{E}} w_{s,t+1}^{(e)}}$;
19 **end**
20 Repeat Steps 8–16 for predicting workload of s for future time steps $T + 1$, $T + 2$, and so on;
21 **end**

In lines 8–19, we learn the parameters of the WMC-based prediction model (i.e., the weights of the experts in ensemble $\mathcal{E}$) for server s, using historical data $D_T^{(s)}$. We begin the learning procedure for server s by computing workload predictions of each expert e for time step t included in the historical time-series $D_T^{(s)}$ (line 9). The predictions of all experts are then compounded using the ensemble combination rule specified in (9), to constitute the ensemble workload prediction $\widehat{p}_{s,t}$ (line 10) for server s at time step t. Thereafter, we obtain the actual workload of server s at time step t from $D_T^{(s)}$ (line 11) and use it to evaluate the absolute loss of each expert (line 13). Using this absolute loss and fixed parameter β, we determine the update factor F_s such that the condition in (10) is satisfied (line 14). We then update the weight of each expert e based on F_s using (11) (line 15). Finally, the weights of all experts are normalized to add to 1 (lines 17–18). This learning procedure is repeated for each time step in the training set $D_T^{(s)}$ of server s. Having learned these ensemble parameters, we then predict workloads of server s for future time steps $T+1$, $T+2$, and so on (line 20) using the procedure outlined in lines 8–19.

D. DISCUSSION

The proposed WMC-based workload prediction model offers three advantages in large scale workload prediction, which indicate the efficacy of WMC-based workload prediction model in practical settings.

First, it inherently handles the non-stationarity in workloads over time by dynamically updating the learning parameters, i.e., the weights of experts, in response to changes in the underlying distributions of server workloads. For instance, in Example 1 in Section III-A, when the underlying distribution of workload of server s_1 changes from one gaussian distribution $\mathcal{N}_1$ to another gaussian distribution $\mathcal{N}_2$, the weight update rule of WMC causes changes in experts' weights such that the weight of experts based on (or approximately following) $\mathcal{N}_1$ distribution is reduced and the weight of expert(s) based on (or approximately following) $\mathcal{N}_2$ distribution is increased. This obviates the need to retrain the prediction model of each server individually, and likewise renders hand-tuning and manual intervention unnecessary.

Second, the proposed workload prediction model also handles non-stationarity across servers, by forming a global ensemble and maintaining a set of experts' weights for each server so that experts with higher accuracies in workload predictions are assigned greater weights. Thus, the need to create and fit a separate prediction model for each server in a large scale system is avoided.

Third, the learning/update of experts' weights, and the predictions of workload of each server, are done in an incremental and computationally efficient way, as and when new workload data for servers are available.

V. WORKLOAD PREDICTION USING SIMULATABLE EXPERTS BASED ENSEMBLE LEARNING

In this section, we propose another ensemble-based workload prediction model by extending the Simulatable Experts (SE) algorithm [17] described in Section II-C. In previous work, SE is only formulated for problem scenarios in which the actual outcomes are binary, i.e., the *outcome space* $\mathcal{Y}$ is denoted by $\{0, 1\}$. (It may be mentioned that in workload prediction scenarios, outcomes correspond to actual server workloads).

In this work, we extend SE in two ways to make it applicable to the large-scale workload prediction problem. (Hereafter, we call the extended SE as *kSE*). First, since the outcome space in the workload prediction problem is not limited to 0 and 1, we extend SE to a k-outcome space. In this setting, the outcome space can be denoted as $\mathcal{Y} = \{y_i\}_{i=0}^{k-1}$, $y_i \in [0, 1]$, and $y_i = \frac{i}{k-1}$ $\forall i = 0, \ldots, (k-1)$. Second, we limit the workload instances that would be considered for minimization of the worst-case regret function in such a

way that in (3), only the most-recent α instances would be considered (as opposed to all n instances in SE).

A. KSE ALGORITHM

We now proceed to mathematically formulate kSE with these proposed extensions. The first step in kSE remains the same as that in WMC, i.e., building a fixed-size global ensemble $\mathcal{E}$ of M experts based on statistical analyses of historical workload data of servers in a system. At time step n, let $P_n^{n-\alpha}$ denote the most-recent α outcomes, i.e., from time step $n-\alpha$ to time step n. Let $\mathcal{L}_s(P_n^{n-\alpha})$ denote the cumulative absolute loss of ensemble $\mathcal{E}$ for server s at time step n for the last α outcomes, which is defined as:

$$\mathcal{L}_s(P_n^{n-\alpha}) = \sum_{t=n-\alpha}^{n} |\widehat{p}_{s,t} - p_{s,t}| \tag{14}$$

where $\widehat{p}_{s,t}$ denotes the ensemble workload prediction for server s at time t, and $p_{s,t}$ denotes the actual workload for server s at time t.

Also, let $L_s^{(e)}(P_n^{n-\alpha})$ denote the cumulative absolute loss of expert e for server s at time step n, which is again defined using the most-recent α outcomes as:

$$L_s^{(e)}(P_n^{n-\alpha}) = \sum_{t=n-\alpha}^{n} |x_{s,t}^{(e)} - p_{s,t}| \tag{15}$$

where $x_{s,t}^{(e)}$ denotes the workload prediction of expert e for server s at time step t.

The *worst-case regret function* in kSE, denoted by $V_{s,n}(\mathcal{E})$, remains the same as in SE, except that the outcome space $\mathcal{Y}$ now consists of k outcomes (as opposed to binary outcomes in SE), and we now consider only α instances (as opposed to n in SE) in computing the cumulative loss of the global ensemble and of its experts. So, the worst-case regret function can now be written as:

$$V_{s,n}(\mathcal{E}) = \sup_{P_n^{n-\alpha} \in \mathcal{Y}^\alpha} (\mathcal{L}_s(P_n^{n-\alpha}) - \min_{e \in \mathcal{E}} L_s^{(e)}(P_n^{n-\alpha})) \tag{16}$$

In order to determine the optimal workload prediction $\widehat{p}_{s,n}$ for server s at time step n that minimizes the worst-case regret $V_{s,n}(\mathcal{E})$, we first minimize the following:

$$\max \begin{cases} \mathcal{L}_s(P_{n-1}^{n-1-\alpha}) + \ell(\widehat{p}_{s,n}, y_0) - \inf\limits_{e \in \mathcal{E}} L_s^{(e)}(P_{n-1}^{n-1-\alpha} y_0), \\ \mathcal{L}_s(P_{n-1}^{n-1-\alpha}) + \ell(\widehat{p}_{s,n}, y_1) - \inf\limits_{e \in \mathcal{E}} L_s^{(e)}(P_{n-1}^{n-1-\alpha} y_1) \\ \vdots \\ \mathcal{L}_s(P_{n-1}^{n-1-\alpha}) + \ell(\widehat{p}_{s,n}, y_{k-1}) - \inf\limits_{e \in \mathcal{E}} L_s^{(e)}(P_{n-1}^{n-1-\alpha} y_{k-1}) \end{cases}$$

Minimizing this equation, in turn, is equivalent to minimizing the following:

$$\max \begin{cases} |\widehat{p}_{s,n} - y_0| - \inf\limits_{e \in \mathcal{E}} L_s^{(e)}(P_{n-1}^{n-1-\alpha} y_0), \\ |\widehat{p}_{s,n} - y_1| - \inf\limits_{e \in \mathcal{E}} L_s^{(e)}(P_{n-1}^{n-1-\alpha} y_1), \\ \vdots \\ |\widehat{p}_{s,n} - y_{k-1}| - \inf\limits_{e \in \mathcal{E}} L_s^{(e)}(P_{n-1}^{n-1-\alpha} y_{k-1}) \end{cases} \tag{17}$$

In order to determine optimal prediction $\widehat{p}_{s,n}$ that minimizes (17), we first define $A_{n-1}(P_{n-1}^{n-1-\alpha}))$ as:

$$A_{n-1}(P_{n-1}^{n-1-\alpha}) = -\inf_{e \in \mathcal{E}} L_s^{(e)}(P_{n-1}^{n-1-\alpha}) \tag{18}$$

and then define a comparison operator '$\succ$' on $A_n(P_n^{n-\alpha} y_i)$ as follows:

$$A_n(P_n^{n-\alpha} y_i) \succ A_n(P_n^{n-\alpha} y_j) \implies A_n(P_n^{n-\alpha} y_i) > A_n(P_n^{n-\alpha} y_j) + Q(y_j) \tag{19}$$

where

$$Q(y_i) = \begin{cases} y_i & \text{if } y_i \geq 0.5, \\ 1 - y_i & \text{otherwise.} \end{cases} \tag{20}$$

In (20), we chose 0.5 as it is the midpoint of the outcome space [0,1]. Next, we define $X_{y_i,y_j}^{(n-1-\alpha,n-1)}$ as:

$$X_{y_i,y_j}^{(n-1-\alpha,n-1)} = \frac{A_n(P_{n-1}^{n-1-\alpha} y_i) - A_n(P_{n-1}^{n-1-\alpha} y_j) + 1}{2} \tag{21}$$

Using (21) in (18), (19), and (21), we compute $\widehat{p}_{s,n}$ for the k-outcome space (based on the last α outcomes) that minimizes the worst-case regret as follows:

$$\begin{cases} y_i & \text{if } A_n(P_{n-1}^{n-1-\alpha} y_i) \succ A_\alpha(P_{n-1}^{n-1-\alpha} y_j), \\ & i \neq j \\ X_{y_i,y_j}^{(n-1-\alpha,n-1)} & \text{if } A_n(P_{n-1}^{n-1-\alpha} y_i) \not\succ A_n(P_{n-1}^{n-1-\alpha} y_j) \wedge \\ & A_n(P_{n-1}^{n-1-\alpha} y_i) \succ A_n(P_{n-1}^{n-1-\alpha} y_q) \wedge \\ & A_n(P_{n-1}^{n-1-\alpha} y_j) \succ A_n(P_{n-1}^{n-1-\alpha} y_q), \\ & i \neq j \neq q \end{cases} \tag{22}$$

B. IMPLEMENTATION

We now describe how kSE can be implemented for predicting servers' workloads in a large-scale server system. We outline a high-level implementation of kSE in Algorithm 2, which requires the following inputs: a set of servers S, a training dataset $\mathcal{D}_T$ containing workload data of each server in S for the past T time steps, and a fixed parameter α which denotes the number of past workload instances that would be taken into consideration while minimizing the worst-case regret function $V_{s,n}(\mathcal{E})$ for each server s. We begin the same way as with WMC, by analyzing training dataset $\mathcal{D}_T$ to identify salient statistical properties and patterns in servers workload based on which we form a global ensemble of experts (lines 1–2). Lines 3–14 describe how optimal workload prediction can be computed using kSE for each server s in a server system.

In order to compute the optimal workload prediction for server for a future time step t, we proceed as follows. We first extract a subset $D_\alpha^{(s)}$ from training dataset $\mathcal{D}_T$ such that $D_\alpha^{(s)}$ contains the time-series workload data of server s for the past α time steps, i.e., $D_\alpha^{(s)} = \{p_{s,t-1}, p_{s,t-2}, \ldots, p_{s,t-\alpha}\}$ (line 5). Then for each expert e in the global ensemble $\mathcal{E}$, we obtain its workload predictions for server s for the past α time steps, and then calculate its cumulative loss over all possible k outcomes in $\mathcal{Y}$, i.e., $L_s^{(e)}(P_{n-1}^{n-1-\alpha} y_0), \ldots, L_s^{(e)}(P_{n-1}^{n-1-\alpha} y_{k-1})$ (lines 6–9). Having calculated the cumulative losses of all experts in $\mathcal{E}$

Algorithm 2: kSE Workload Prediction Algorithm

Input: Set of servers S, training dataset $\mathcal{D}_T$, fixed parameter α

1 Perform statistical analysis of the dataset $\mathcal{D}_T$;
2 Construct ensemble $\mathcal{E}$ of M experts based on statistical analysis results;
3 **foreach** $s \in S$ **do**
4 **foreach** $t = T+1, T+2, \ldots$ **do**
5 Extract workload dataset $D_\alpha^{(s)}$ of server s from $\mathcal{D}_T$;
6 **foreach** $e \in \mathcal{E}$ **do**
7 Compute workload predictions $x_{s,t-1}^{(e)}, \ldots, x_{s,t-\alpha}^{(e)}$ of expert e for past α time steps;
8 Calculate cumulative loss of expert e (for last α time steps) over k outcomes in $\mathcal{Y}$, i.e., $L_s^{(e)}(P_{n-1}^{n-1-\alpha} y_0), \ldots, L_s^{(e)}(P_{n-1}^{n-1-\alpha} y_{k-1})$;
9 **end**
10 **foreach** $y_i \in \mathcal{Y}$ **do**
11 Set $A_t(P_{t-1}^{t-1-\alpha} y_i) = -\inf_{e \in \mathcal{E}} L_s^{(e)}(P_{t-1}^{t-1-\alpha} y_i)$;
12 **end**
13 Compute final workload prediction $\widehat{p}_{s,t}$ as

$$\begin{cases} y_i & \text{if } A_t(P_{t-1}^{t-1-\alpha} y_i) \succ A_\alpha(P_{t-1}^{t-1-\alpha} y_j), \\ & i \neq j \\ X_{y_i,y_j}^{(t-1-\alpha,t-1)} & \text{if } A_t(P_{t-1}^{t-1-\alpha} y_i) \not\succ A_t(P_{t-1}^{t-1-\alpha} y_j) \wedge \\ & A_t(P_{t-1}^{t-1-\alpha} y_i) \succ A_t(P_{t-1}^{t-1-\alpha} y_q) \wedge \\ & A_t(P_{t-1}^{t-1-\alpha} y_j) \succ A_t(P_{t-1}^{t-1-\alpha} y_q), \\ & i \neq j \neq q \end{cases}$$

;
14 **end**
15 Repeat Steps 5–11 until a stopping criterion is met;
16 **end**

over all possible outcomes in $\mathcal{Y}$, we proceed to find the infimum of the loss term $L_s^{(e)}(P_{t-1}^{t-1-\alpha} y_i)$ over all experts for each outcome, which is used to initialize $A_t(P_{t-1}^{t-1-\alpha} y_i)$ (lines 10–12). Finally, we compute the optimal workload prediction $\widehat{p}_{s,t}$ for server s at time step t using (22) (line 13). Lines 3–14 can be repeated for workload predictions of server $s \in S$ for future time steps until a stopping criterion is met (line 15).

It may be noted that this algorithm (as with Algorithm 1) is easily extensible to multi-step predictions, as they can be used for predictions of multiple time intervals either in sequence or in parallel. This further increases the applicability of the proposed kSE and WMC models to those scenarios that require workload predictions over a window of time intervals.

C. DISCUSSION

The kSE workload prediction model effectively deals with the key issues of large scale workload prediction, i.e., extensive workload non-stationarity and massive scale of online data, in the following three ways.

First, any non-stationarity in the workload of server s at time step t would cause the losses of experts (i.e., $L_s^{(e)}(P_t^{t-\alpha})$) to change, which would affect the value of $A_t(P_t^{t-\alpha})$. The updated $A_t(P_t^{t-\alpha})$ would, in turn, cause the ensemble prediction $\widehat{p}^{s,t+1}$ to change in accordance with the workload non-stationarity. For instance, in Example 1 of Section III-A, when the underlying distribution of the workload of server s_1 changes from $\mathcal{N}_1$ to $\mathcal{N}_2$, the expert(s) based on (or approximately following) distribution $\mathcal{N}_1$ would show significant increases in absolute loss, whereas the expert(s) based on (or approximately following) distribution $\mathcal{N}_2$ would show decreases in loss. This would affect the value of $A_t(P_t^{t-\alpha})$ which, in turn, would induce adjustments in the ensemble prediction $\widehat{p}^{s,t+1}$ for the next time step. In this way, kSE handles workload non-stationarity over time without any need of hand-tuning or retraining of the prediction model.

Second, since we create and use one global ensemble for workload predictions of all servers in the system, and the optimal workload prediction of a server is computed based on predictions of an expert in a way which minimizes the worst-case regret, we are able to compute optimal workload predictions for each server without the need to assess and create different prediction models for each server in the system.

Third, the computational cost of kSE is kept at a minimum by following an online procedure for determining optimal predictions, wherein only the last α workload instances (as opposed to all n instances in SE) are taken into consideration. These key advantages make kSE attractive for practical use in large-scale workload prediction scenarios.

However, it may be noted that kSE is computationally more expensive as compared to WMC, primarily because at each time step t, kSE computes the cumulative loss of each expert over the last α instances for each outcome in $\mathcal{Y}$ (i.e., Step 8 in Algorithm 2). But, as we will see in Section VI, kSE outperforms WMC in terms of predictive accuracy on large datasets.

VI. EXPERIMENTAL EVALUATION

In this section, we evaluate the effectiveness of the proposed ensemble-based workload prediction models, kSE and WMC, using datasets from a large enterprise data center. We perform three kinds of experimental evaluation on these workload prediction models: macro-averaged prediction error evaluation, percentage-of-servers based evaluation, and scale-and-composition based evaluation. In macro-averaged error evaluation (see Section VI-B), we evaluate the *macro-averaged* errors of kSE and WMC on large non-stationary datasets of 1570 servers. We determine the macro-averaged error of a prediction model by first computing the prediction error of the model for each server, and then averaging the errors over all servers in the data center. This gives a summary statistic which indicates how effective a prediction model is in predicting workloads of servers in the given data center.

In percentage-of-servers based evaluation (see Section VI-C), we analyze the percentage of servers in the data center for which workload can be predicted, using

each prediction model under consideration, within specific error limits, e.g., <9 or <11. This kind of evaluation is of significance while choosing a workload prediction model in practical business scenarios.

Finally, in scale-and-composition based evaluation (see Section VI-D), we show the impact of increases in the number of servers in the data center on the predictive performance of proposed kSE and WMC prediction models. We also assess whether the composition of server infrastructure, in terms of homogeneity and heterogeneity of servers, affects the predictive performances of kSE and WMC.

A. DATASETS AND EXPERIMENTAL SETUP

We use datasets collected from a real data center consisting of 2400 servers, of which we chose a subset of 1570 actively-utilized servers for our analyses. The data center under consideration runs loads of various kinds, many of which are CPU-intensive. Therefore, for the selected 1570 servers, we collected CPU workload data for a period of six months using operating system-based software agents which were installed on each server. Each software agent collected one data point every ten minutes, for its server. This data point gave the percentage of time in the previous ten minutes for which that server's CPU was in active state. These data points were then aggregated to form hourly workload time series.

The workload dataset of the selected 1570 servers was analyzed to gain insights into important characteristics of the datasets. The analyses revealed that most servers under consideration had significant auto-correlation for lags of 6^{th} and 7^{th} day every week, indicating weekly trends in the dataset. It was observed that such servers had lower workloads on the 6^{th} and 7^{th} day of the week as compared to the rest of the days in the week. A few servers also showed significant auto-correlation for a lag of the 3^{rd} day every month, indicating a monthly trend in these servers. Apart from seasonality, non-stationarity over time was observed frequently for many servers, mainly due to reallocation of one or more cluster of servers to different user-groups by the data center administrators, reconfiguration of servers, etc. A detailed investigation into workload of a random sample of 200 servers belonging to different user groups also revealed massive non-stationarity in the workloads across servers, due to differences in their utilization.

We denote the large scale non-stationary dataset of 1570 servers as G(1570). It was manually ensured that only actively-utilized servers were included in G(1570), many of which have significantly non-stationary workloads. This dataset was partitioned into a training dataset G-train(1570) consisting of servers' workload data for five consecutive months, and a test dataset G-test(1570) consisting of workload data for one month.

In order to particularly ensure that the test dataset G-test(1570) has significant non-stationarity with respect to the training dataset G-train(1570), we created a synthetic test dataset G′-test(1570). Many servers in G(1570) showed distinct patterns of low workloads during each weekend. So, to create a synthetic test dataset with distinct non-stationarity, we introduced an *inverted* trend of high workloads during each weekend in G′-test(1570). As a result, the synthetic test dataset G′-test(1570), with inverted high workload pattern for weekends, becomes non-stationary with respect to the training set G-train(1570) that has distinctly low workload patterns during weekends. It may be noted that in the real world the distribution of a training dataset often differs to some extent from the distribution of a test dataset, for various reasons. Hence, it is important to perform this kind of evaluation in order to assess the applicability of kSE and WMC in practical scenarios.

In our experiments, we primarily use Mean Absolute Percentage Error (MAPE) as the measure of prediction accuracy but, for macro-averaged analysis (in Section VI-B), we also use relative error and order statistics for an in-depth analysis of predictive performance of the proposed models. This kind of accuracy-based evaluation is useful in scenarios like dynamic capacity provisioning [11], [12] of server systems, wherein workload predictions are used to enable decision-making regarding optimal capacity allocation to server server systems to achieve energy efficiency and improve the sustainability of server systems. It is also useful in predictive server consolidation [13]–[15], where based on server workload predictions, decisions are taken regarding consolidation of servers (that are predicted to have low workloads in the near future), to efficiently manage system resources and minimize energy consumption. In such scenarios, the effectiveness of decisions depend highly on the accuracy of workload predictions, which makes accuracy-based evaluation of workload prediction methodologies desirable considering these scenarios.

For comparative evaluation of kSE and WMC, we use three baseline prediction models: MA(4), MA(24) and random walk model (RW). MA(4) and MA(24) are based on the Moving Average (MA) model [55] with the order of moving average as 4 (hours) and 24 (hours) respectively. We do not use seasonal models as baselines, because such models tend to perform poorly on synthetic test dataset G′-test(1570) as the seasonal/weekly trend parameters learned by such models from the training dataset G-train(1570) are no longer valid in the test dataset G′-test(1570) due to inversion of weekly trends. On the other hand, MA models, i.e., MA(4) and MA(24), are not affected significantly by inversion of trends in G′-test(1570), and hence provide a fair comparison and consistent performances across both test datasets, G-test(1570) and G′-test(1570). Also, seasonal models (e.g., SARIMA) typically require computationally-expensive retraining whenever seasonal trends change for a server under consideration, which makes them infeasible for large-scale server systems, whereas MA-based models do not require such retraining as they are inherently incremental and online.

1) Parameter Selection and Ensemble Formation

In WMC, parameter β influences the rates at which the weights of experts are updated in response to their prediction accuracies—the higher the value of β, the lower the rate of updates. In a large-scale workload prediction scenario, if the workloads of most servers change slowly over time, then a lower rate of update might be desirable, and β may be set to a high value like 0.8 or 0.9. In our dataset, workloads of most servers exhibit significant non-stationarity, implying the need for relatively faster updates, and hence the value of β was set to 0.5 in our experiments.

TABLE 2. Macro-averaged MAPE analysis of prediction models using real dataset, G-test(1570), and synthetic dataset with increased non-stationarity, G′-test(1570).

Prediction model	Macro-averaged Error		Difference (in %)
	G-test(1570)	G′-test(1570)	
kSE	8.05	8.3	-3.1
WMC	8.37	8.2	2.03
MA(4)	22.7	25.1	-10.5
MA(24)	26.4	28.3	-7.19
RW	29.4	30.7	-4.4

In kSE, parameter α denotes the number of most-recent instances that would be taken into consideration for minimizing the regret $V_{s,n}(\mathcal{E})$ of ensemble $\mathcal{E}$. In our experiments, the value of α is set to (24×7), implying one week of hourly workload data points. Setting α to higher values may result in more reliable server workload predictions, but would increase the computational time required to execute kSE. On the other hand, setting α to low values like 24 (implying just one day's hourly workload data points) may significantly decrease the required computational time for kSE, but would adversely affect the accuracy of the model.

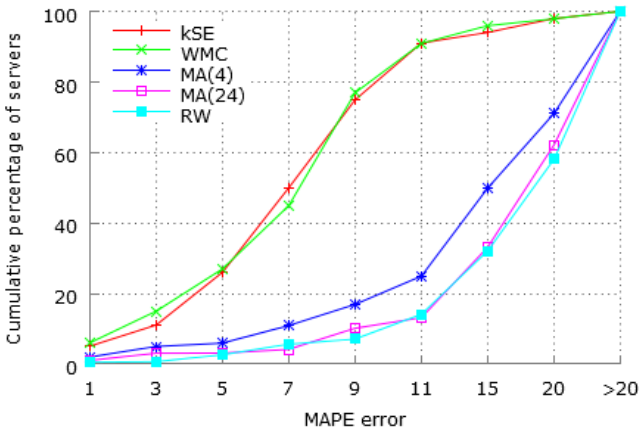


FIGURE 3. Percentage-of-servers based evaluation using real dataset G-test(1570).

In order to determine the experts to be included in the global ensemble, data analysis can be done on a smaller, random sample of servers using conventional time-series based models like exponential smoothing, auto-regression and ARIMA. Based on the analysis results, an initial set of experts can be chosen and the predictive performance of the ensemble can be assessed using a validation set [57]. Validation set may contain workload data of all servers for a small time period, e.g., a week. (We refer the readers to [57] and [58] for relevant details). Depending on the performances of experts based on a time-series model, say exponential smoothing, more experts based on that model can be included (or excluded) with varying statistical and time parameters. The updated set of experts can again be assessed on the same validation set as before, and after a few iterations, a final set of experts can be formed. In our experiments, this procedure concluded in a set of 90 experts which formed the global ensemble.

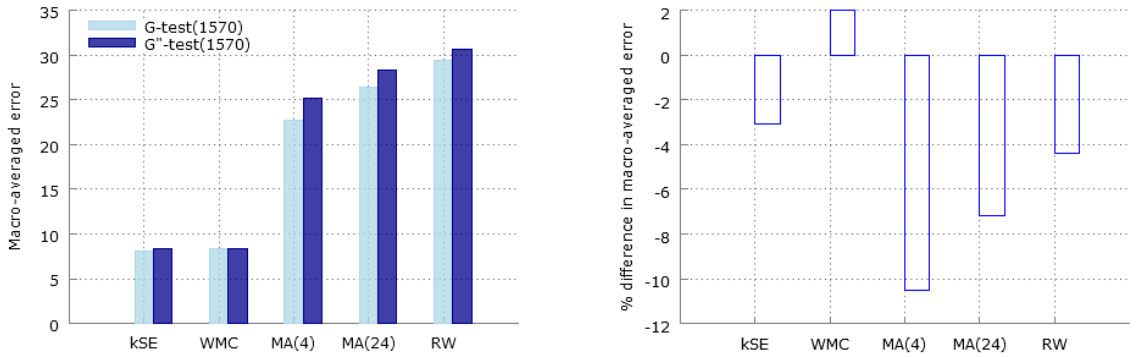
B. MACRO-AVERAGED PREDICTION ERROR ANALYSIS ON LARGE SCALE NONSTATIONARY DATASET G(1570)

In this section, we present comparative evaluations of macro-averaged prediction error of the ensemble-based prediction models, kSE and WMC, and the baseline prediction models, MA(4), MA(24) and random walk (RW). In order to do so, we trained each of these prediction models on the G-train(1570) dataset and then tested them on G-test(1570) and G′-test(1570) datasets.

Fig. 2 shows the macro-averaged errors of the kSE, WMC and baseline prediction models, evaluated over real dataset G-test(1570) as well as synthetic dataset G′-test(1570). Recall that G′-test(1570) has greater non-stationarity due to inverted trends of workloads that we introduced for each weekend in the test period. It can be easily seen that for both G-test(1570) and G′-test(1570) datasets, the macro-averaged MAPE error of kSE and WMC is in the range of 8.05–8.37 which is significantly lower than the macro-averaged error of 22.7–30.7 for the baseline prediction models.

The detailed numerical results of macro-averaged MAPE evaluation for each prediction model are shown in Table 2. The table also shows a comparison (in percentage terms) between macro-averaged MAPE computed over G-test(1570) and G′-test(1570), so as to highlight the effect of additional non-stationarity in G′-test(1570) on the prediction accuracy of proposed and baseline prediction models. This percentage comparison is illustrated pictorially in Fig. 2, where we see that due to increased non-stationarity in G′-test(1570), the macro-averaged error of kSE increases by 3.1%, but surprisingly, the macro-averaged error of WMC decreases by 2.03%. On the other hand, the macro-averaged error of baseline algorithms MA(4), MA(24) and RW, show more significant increases of 10.5%, 7.19% and 4.4% respectively.

We further evaluated the accuracies of proposed models using relative errors and order-statistics based metrics, and present the evaluation results in Table 3. We first compute the relative error of a prediction model for server s as $\frac{\sum_t |\widehat{p}_{s,t}-p_{s,t}|}{\sum_t p_{s,t}}$, and then average it over all servers in order to determine macro-averaged relative error. In Table 3, it is



(a) Macro-averaged error analysis (b) Difference in macro-averaged error

FIGURE 2. (a) Macro-averaged error analysis using real dataset, G-test(1570), and synthetic dataset, G′-test(1570); (b) Percentage difference in macro-averaged prediction error due to increased non-stationarity in G′-test(1570).

TABLE 3. Macro-averaged relative error and order statistics analysis of prediction models using real dataset, G-test(1570), and synthetic dataset, G′-test(1570).

Prediction model	Relative Error		Median		90^{th} percentile	
	G-test(1570)	G′-test(1570)	G-test(1570)	G′-test(1570)	G-test(1570)	G′-test(1570)
kSE	0.07	0.08	0.04	0.04	0.30	0.31
WMC	0.08	0.07	0.05	0.04	0.32	0.31
MA(4)	0.20	0.23	0.10	0.11	0.50	0.53
MA(24)	0.23	0.25	0.11	0.13	0.55	0.58
RW	0.26	0.27	0.21	0.20	0.61	0.64

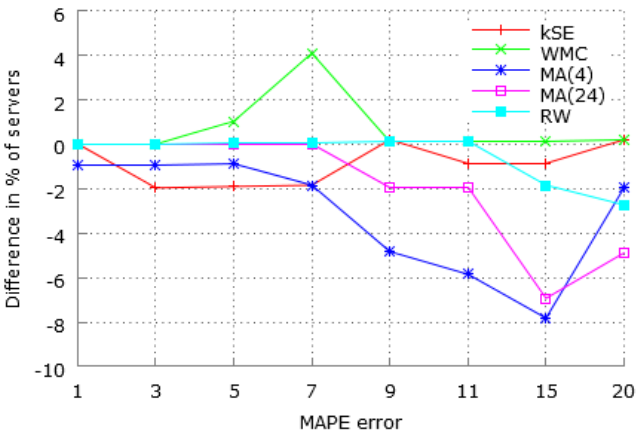


FIGURE 4. Percentage difference in cumulative number of servers for each error limit, caused due to increased non-stationarity of G′-test(1570).

shown that the proposed ensemble-based workload prediction models have macro-averaged relative errors in the range of 0.07–0.08, whereas the baseline models have significantly higher macro-averaged relative errors in the range of 0.20–0.27. We also evaluated the proposed models based on median and 90^{th} percentile values of their error distributions, where error is simply computed as $\frac{|\hat{p}_{s,t}-p_{s,t}|}{p_{s,t}}$. We present macro-averaged median and macro-averaged 90^{th} percentile values of the error distribution for each prediction model in Table 3, where we see that the macro-averaged median errors of our proposed workload prediction models fall in the range of 0.04–0.05, whereas macro-averaged median errors of the baseline models are comparatively higher, i.e., in the range of 0.10–0.20. Similarly, 90^{th} percentile errors of the proposed models are in the range of 0.30–0.31, whereas the 90^{th} percentile errors of the baseline models are again significantly higher, in the range of 0.50–0.64.

These results indicate that with increased non-stationarity of workloads, the proposed ensemble-based workload prediction models outperform the baseline models by a significant extent. More specifically, kSE outperforms the baselines as well as WMC (albeit by a small margin) in terms of prediction accuracy. However, recall that kSE is computationally more expensive than WMC. Therefore, in large scale-workload prediction scenarios where predictive accuracy is of utmost importance and computational cost is not considered a constraint, kSE may be preferred to WMC. On the other hand, in scenarios where computational time or resources are limited, WMC may be preferred over kSE.

TABLE 4. Prediction error evaluation with total number of servers in different error ranges using real dataset G-test(1570) and synthetic dataset with increased non-stationarity G′-test(1570).

Dataset	Prediction model	Range of MAPE error								
		<1	1–3	3–5	5–7	7–9	9–11	11–15	15–20	>20
G-test(1570)	kSE	79	94	**235**	**376**	**392**	**253**	47	62	32
	WMC	94	141	188	**282**	**502**	**220**	79	31	33
	MA(4)	31	47	15	78	94	126	**392**	**332**	**455**
	MA(24)	16	32	1	16	93	48	**314**	**455**	**595**
	RW	8	2	31	47	23	110	**282**	**408**	**659**
G′-test(1570)	kSE	79	63	**236**	**377**	**424**	**236**	47	79	29
	WMC	94	141	**204**	**330**	**440**	**220**	79	32	30
	MA(4)	16	47	16	63	47	110	**361**	**424**	**486**
	MA(24)	16	32	1	16	63	47	**236**	**487**	**672**
	RW	8	2	32	47	24	110	**251**	**394**	**702**

C. PERCENTAGE-OF-SERVERS BASED EVALUATION ON LARGE-SCALE NONSTATIONARY DATASET G(1570)

We now present detailed evaluation results of the proposed ensemble-based prediction models based on the percentages of servers for which workloads may be predicted within specific error limits like <5, <11, etc. For this evaluation, we use MAPE as the error measure for comparing the predictive performances of the models. In business scenarios, this evaluation is useful while making decisions on the choice of workload prediction models for a given server infrastructure, especially when a target prediction error limit is specified by a third party or end-users.

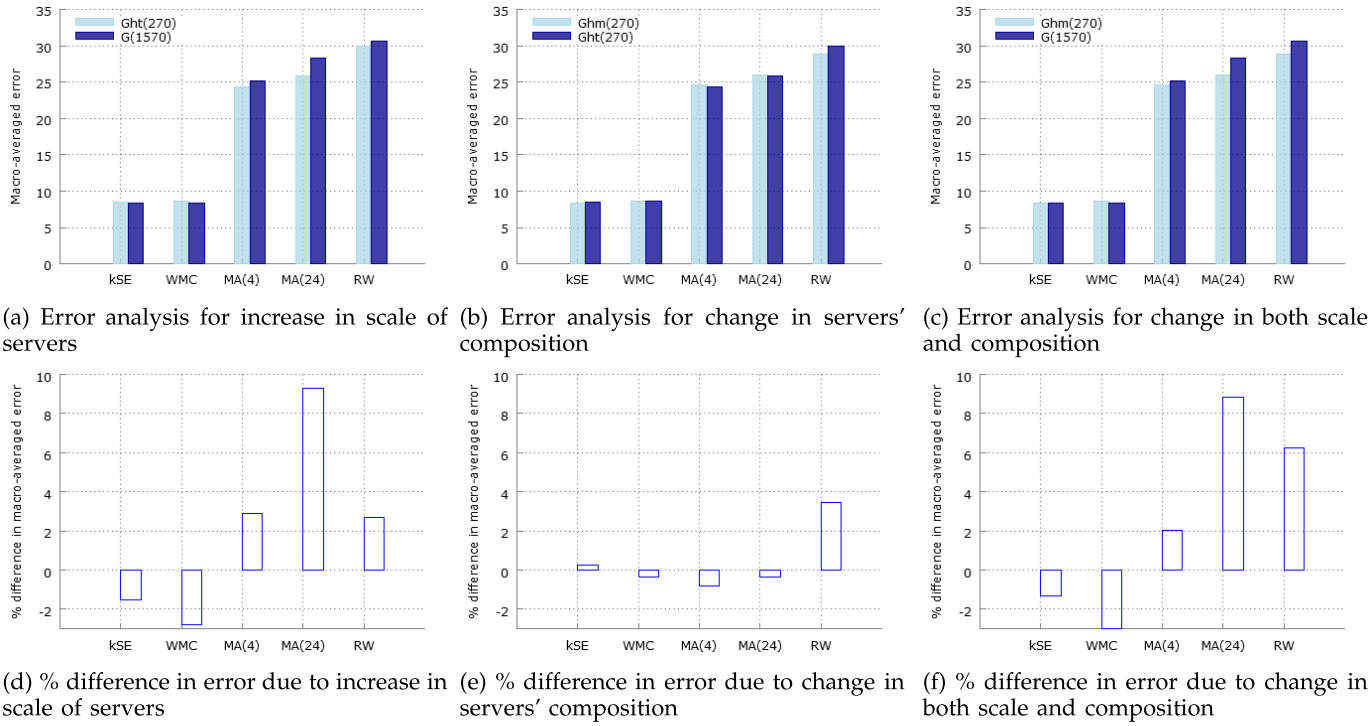
Table 4 shows evaluations based on percentages of servers, on the real dataset G-test(1570) and the synthetic dataset G′-test(1570). We see that for G-test(1570), using kSE and WMC leads to a majority of servers being in the error range 3–5, 5–7, 7–9, and 9–11 (highlighted in the table). This implies that using the proposed ensemble-based prediction models, we can obtain workload predictions with errors less than 11 for a majority of servers in the data center. However, this is not true for baseline prediction models, using which prediction error of most of the servers are in higher ranges of 11–15, 15–20 and >20 (highlighted in Table 4).

In Fig. 3, we further analyze this difference quantitatively by plotting the cumulative *percentage* of servers for each error limit specified in Table 4. We see that, using kSE and WMC, we obtain prediction errors of less than 11 for approximately 91% of servers, whereas using the baseline models, we achieve prediction errors of less than 11 for only 13%–24% of servers. This is an important result, as it shows that using the proposed ensemble-based prediction models, the workloads of approximately 91% of servers in the data center can be predicted with accuracy greater than 89%, whereas using baseline algorithms, the workloads of hardly 13%–24%, servers can be predicted with this accuracy. If the desired error limit is moved to stricter values of 9 or 5, the ensemble-based prediction models still outperform the baseline models by a significant extent.

Table 4 also shows similar evaluation results on the synthetic dataset G′-test(1570). We see that using kSE and WMC results in the prediction errors of a majority of servers being less than 11, as compared to the baselines, in which a majority of servers have MAPE in the ranges of 11–15, 15–20 and >20 (highlighted in Table 4). In order to illustrate the effect of added non-stationarity in G′-test(1570), we calculated for each prediction model the percentage difference in the number of servers that is caused due to increased non-stationarity in G′-test(1570), and plot the results in Fig. 4. We see that for an error limit of 11, the kSE and WMC prediction models do not show significant decrease in the cumulative number of servers, whereas the baseline models MA (4) and MA (24) do show decreases of approximately 2% and 6% respectively. For stricter error limits like 9 or 5, similar results can be seen in Fig. 4. This indicates that the proposed workload prediction models are comparatively more robust at handling increased workload non-stationarity.

TABLE 5. Prediction error evaluation with changes in scale and/or composition of server infrastructure.

	Prediction model	Macro-averaged Error over different datasets		Difference (in %)
		Ght(270)	G(1570)	
Increase in servers′ scale	kSE	8.43	8.3	-1.54
	WMC	8.61	8.37	-2.79
	MA(4)	24.4	25.1	2.87
	MA(24)	25.9	28.3	9.27
	RW	29.9	30.7	2.68
		Ghm(270)	Ght(270)	
Change in servers′ composition	kSE	8.41	8.43	0.24
	WMC	8.64	8.61	-0.35
	MA(4)	24.6	24.4	-0.81
	MA(24)	26	25.9	-0.38
	RW	28.9	29.2	3.46
		Ght(270)	G(1570)	
Change in both scale and composition	kSE	8.41	8.3	-1.31
	WMC	8.64	8.37	-3.13
	MA(4)	24.6	25.1	2.03
	MA(24)	26	28.3	8.85
	RW	28.9	30.7	6.23



(a) Error analysis for increase in scale of servers

(b) Error analysis for change in servers' composition

(c) Error analysis for change in both scale and composition

(d) % difference in error due to increase in scale of servers

(e) % difference in error due to change in servers' composition

(f) % difference in error due to change in both scale and composition

FIGURE 5. (a)–(c) Macro-averaged error analysis for changes in scale and/or composition of servers; (d)–(f) Percentage difference in macro-averaged error due to changes in scale and/or composition of servers.

D. PREDICTION ERROR EVALUATION CONSIDERING SCALE AND COMPOSITION OF SERVER INFRASTRUCTURE

In this section, we discuss how we investigated if the prediction accuracies of kSE and WMC tend to change when the scale of server infrastructure is increased, and/or when the composition of server infrastructure changes from homogenous (in terms of operating systems and software applications hosted) to heterogenous. In order to perform this evaluation, we created two more datasets Ghm(270) and Ght(270) where Ghm(270) consists of workload data of 270 homogeneous servers, and Ght(270) consisting of workload data of 270 heterogenous servers. For this evaluation, we used macro-averaged MAPE as the error measure for comparing predictive performance of the proposed and baseline models.

We first evaluated the effect of increases in the number of servers on the predictive performances of kSE and WMC, using the Ght(270) and G(1570) datasets. We show the comparison results in Fig. 5(a), wherein we plot the macro-averaged error for each prediction model over Ght(270) and G(1570) datasets. We see that as the number of servers increase from 270 in Ght(270) to 1570 in G(1570), the macro-averaged error of kSE and WMC show a minor, but definite, decrease, whereas the macro-averaged errors of the baseline models, especially MA(24), show considerable increases. This difference is further highlighted in Fig. 5(d), where we show the percentage differences in macro-averaged errors of the prediction models due to increase in the scale of servers. We see that while kSE and WMC show minor decreases of approximately 1–3% in their macro-averaged errors, the baseline prediction models show considerable increases in their macro-averaged errors. Table 5 shows a summary of this comparison, wherein we see that as a result of increases in the number of servers, the macro-averaged errors of kSE and WMC decreased by 1.54% and 2.79% respectively, whereas macro-averaged error of the baseline prediction models increased by 2.68–9.27%. This result indicates that as the scale of server infrastructure increases, the proposed workload prediction models perform significantly better than the baseline prediction models.

In order to demonstrate the effect of changes in composition of server infrastructure (in terms of homogeneity and heterogeneity of servers) on the performances of kSE and WMC, we use the Ghm(270) and Ght(270) datasets. The evaluation results are shown in Fig. 5(b) and Table 5 wherein we see that none of the prediction models show significant increases or decreases in their respective macro-averaged errors. We compute the percentage differences in macro-averaged errors of prediction models, and plot the same in Fig. 5(e), and see that none of the prediction algorithms (except RW) show greater than 1% difference in their macro-averaged errors. This indicates that the composition of server infrastructure does not have any notable impact on the performance of kSE, WMC and baseline prediction models.

For the sake of completeness, we also computed changes in the accuracies of the kSE and WMC prediction models when scale and nature of composition of server infrastructure

are simultaneously changed. We performed this evaluation by comparing macro-averaged errors of the prediction models using workload data of 270 homogenous servers in Ghm(270), and 1570 heterogenous servers in G(1570). Fig. 5(c) and (f) and Table 5 show the results of this comparison wherein we see that kSE and WMC show decreases of 1.3% and 3.1% respectively in macro-averaged errors, whereas the baseline prediction models show considerable increases of 2.03–8.84% in macro-averaged errors, primarily due to increases in the number of servers.

To summarize, the experimental evaluation of proposed kSE and WMC workload prediction models indicate the following:

1) In a large data center with 1570 servers, kSE and WMC achieved significantly lower macro-averaged errors of 8.05–8.37% as compared to errors of 22.7–30.7% for the baseline prediction models, indicating the effectiveness of proposed workload prediction models in large-scale scenarios.
2) Using kSE and WMC, workloads of approximately 91% of servers in a large real data center can be predicted with accuracy greater than 89%, whereas using baseline algorithms, the workloads of only 13–24% of servers can be predicted with similar accuracy.
3) As the scale of server infrastructure increases, the macro-averaged errors of the proposed kSE and WMC prediction models show minor decreases of 1.54–2.79%, whereas the macro-averaged errors of baseline prediction models increase considerably by 2.68–9.27%. This implies that as the number of servers is increased in a server system, kSE and WMC can easily outperform the baseline prediction models.
4) Changes in the nature of composition of server infrastructure, in terms of homogenity or heterogenity of servers, does not significantly affect the accuracy of workload prediction models under consideration.

VII. CONCLUSION

In this work, we addressed the problem of large-scale workload prediction by extending and adapting two online ensemble learning methods, i.e., Weighted Majority and Simulatable Experts. In doing so, we developed a generic theoretical extension of the classical Simulatable Experts, from binary outcome space to k-outcome space, thereby making Simulatable Experts suitable for any k-class learning problem. We demonstrated the effectiveness of the proposed ensemble learning algorithms using large datasets of 1570 servers, and showed that using the proposed algorithms, workloads of approximately 91% servers can be predicted with accuracies greater than 89%, whereas using baseline algorithms, workloads of only 13–24%, servers can be predicted with similar accuracies. In future, we see scope for other enriched variants of Weighted Majority (like Dynamic Weighted Majority [48]) and/or Simulatable Experts to be adapted to yield further benefits in terms of accuracy and computational efficiency in the large-scale workload prediction problem, and also the applicability of the proposed ensemble-based methods in grid and cloud environments.

REFERENCES

[1] T. Lu, M. Chen, and L. L. H. Andrew, "Simple and effective dynamic provisioning for power-proportional data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1161–1171, Jun. 2013.

[2] L. Jian *et al.*, "Reducing operational costs through consolidation with resource prediction in the cloud," in *Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2012, pp. 793–798.

[3] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107–1117, Jun. 2013.

[4] W. Zhikui *et al.*, "AppRAISE: Application-level performance management in virtualized server environments," *IEEE Trans. Netw. Service Manag.*, vol. 6, no. 4, pp. 240–254, Dec. 2009.

[5] N. Doulamis, A. Doulamis, A. Litke, A. Panagakis, T. Varvarigou, and E. Varvarigos, "Adjusted fair scheduling and non-linear workload prediction for QoS guarantees in grid computing," *Comput. Commun.*, vol. 30, no. 3, pp. 499–515, 2007.

[6] T. Enokido, A. Aikebaier, and M. Takizawa, "A model for reducing power consumption in peer-to-peer systems," *IEEE Syst. J.*, vol. 4, no. 2, pp. 221–229, Jun. 2010.

[7] T. Vercauteren, P. Aggarwal, W. Xiaodong, and L. Ta-Hsin, "Hierarchical forecasting of web server workload using sequential Monte Carlo training," *IEEE Trans. Signal Process.*, vol. 55, no. 4, pp. 1286–1297, Apr. 2007.

[8] T.-H. Li, "A hierarchical framework for modeling and forecasting web server workload," *J. Amer. Statist. Assoc.*, vol. 100, no. 471, pp. 748–763, 2005.

[9] J. Moore, J. Chase, K. Farkas, and P. Ranganathan, "Data center workload monitoring, analysis, and emulation," in *Proc. 8th Workshop Comput. Archit. Eval. Commercial Workloads*, Feb. 2005, pp. 1–8.

[10] F. Kluge, S. Uhrig, J. Mische, B. Satzger, and T. Ungerer, "Dynamic workload prediction for soft real-time applications," in *Proc. IEEE 10th Int. Conf. CIT*, Bradford, U.K., Jun. 2010, pp. 1841–1848.

[11] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska, "Online dynamic capacity provisioning in data centers," in *Proc. 49th Annu. Allerton Conf. Commun. Control Comput.*, 2011, pp. 1159–1163.

[12] S. Zhang, Y. Liu, B. Wang, and R. Zhang, "Analysis and modeling of dynamic capacity provisioning problem for a heterogeneous data center," in *Proc. 5th ICUFN*, 2013, pp. 785–790.

[13] M. Chen, H. Zhang, Y.-Y. Su, X. Wang, G. Jiang, and K. Yoshihira, "Effective VM sizing in virtualized data centers," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag.*, May 2011, pp. 594–601.

[14] L. Lu, H. Zhang, E. Smirni, G. Jiang, and K. Yoshihira, "Predictive VM consolidation on multiple resources: Beyond load balancing," in *Proc. IEEE/ACM 21st Int. Symp. Qual. Service*, Jun. 2013, pp. 1–10.

[15] C. Jeonghwan, S. Govindan, J. Jinkyu, B. Urgaonkar, and A. Sivasubramaniam, "Power consumption prediction and power-aware packing in consolidated environments," *IEEE Trans. Comput.*, vol. 59, no. 12, pp. 1640–1654, Dec. 2010.

[16] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Inf. Comput.*, vol. 108, no. 2, pp. 212–261, Feb. 1994.

[17] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. New York, NY, USA: Cambridge Univ. Press, 2006.

[18] S. Casolari and M. Colajanni, "Short-term prediction models for server management in internet-based contexts," *Decision Support Syst.*, vol. 48, no. 1, pp. 212–223, Dec. 2009.

[19] M. Lin, A. Wierman, L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 1098–1106.

[20] A. W. Lewis, N.-F. Tzeng, and S. Ghosh, "Runtime energy consumption estimation for server workloads based on chaotic time-series approximation," *ACM Trans. Archit. Code Optim.*, vol. 9, no. 3, pp. 15:1–15:26, Oct. 2012.

[21] J. Oh and K.-D. Kang, "A predictive-reactive method for improving the robustness of real-time data services," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 5, pp. 974–986, May 2013.

[22] M. Beltran, A. Guzman, and J. L. Bosque, "A new cpu availability prediction model for time-shared systems," *IEEE Trans. Comput.*, vol. 57, no. 7, pp. 865–875, Jul. 2008.
[23] C. Dupont, "Renewable energy aware data centres: The problem of controlling the applications workload," in *Proc. 2nd Int. Workshop Energy-Efficient Data Centers*, 2013.
[24] J. Martinez and E. Ipek, "Dynamic multicore resource management: A machine learning approach," *IEEE Micro*, vol. 29, no. 5, pp. 8–17, Oct. 2009.
[25] X. Zhu *et al.*, "What does control theory bring to systems research?" *SIGOPS Oper. Syst. Rev.*, vol. 43, no. 1, pp. 62–69, Jan. 2009.
[26] D. Krishnamurthy, J. Rolia, and X. Min, "WAM—The weighted average method for predicting the performance of systems with bursts of customer sessions," *IEEE Trans. Softw. Eng.*, vol. 37, no. 5, pp. 718–735, Oct. 2011.
[27] G. Casale, M. Ningfang, L. Cherkasova, and E. Smirni, "Dealing with burstiness in multi-tier applications: Models and their parameterization," *IEEE Trans. Softw. Eng.*, vol. 38, no. 5, pp. 1040–1053, Oct. 2012.
[28] Y. Wu, K. Hwang, Y. Yuan, and W. Zheng, "Adaptive workload prediction of grid performance in confidence windows," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 7, pp. 925–938, Jul. 2010.
[29] N. D. Anastasios, A. Doulamis, A. Panagakis, K. Dolkas, and T. Varvarigou, "Workload prediction of rendering algorithms in grid computing," in *Proc. Eur. Multigrid Conf. GRID Comput.*, Oct. 2002, pp. 7–12.
[30] A. Litke, K. Tserpes, and T. Varvarigou, "Computational workload prediction for grid oriented industrial applications: The case of 3D-image rendering," in *Proc. 5th IEEE Int. Symp. Cluster Comput. Grid*, Washington, DC, USA, May 2005, pp. 962–969.
[31] K. Dolkas, D. Kyriazis, A. Menychtas, and T. Varvarigou, "E-business applications on the grid: A toolkit for centralized workload prediction and access," *Concurrency Comput., Pract. Exper.*, vol. 19, pp. 867–883, Apr. 2007.
[32] P. Xiong, Z. Wang, S. Malkowski, Q. Wang, D. Jayasinghe, and C. Pu, "Economical and robust provisioning of N-tier cloud workloads: A multi-level control approach," in *Proc. 31st ICDCS*, Jun. 2011, pp. 571–580.
[33] B. J. Watson, M. Marwah, D. Gmach, Y. Chen, M. Arlitt, and Z. Wang, "Probabilistic performance modeling of virtualized resource allocation," in *Proc. 7th ICAC*, New York, NY, USA, 2010, pp. 99–108.
[34] M. V. D. Berge, G. D. Costa, M. Jarus, A. Oleksiak, W. Piatek, and E. Volk, "Modeling data center building blocks for energy-efficiency and thermal simulations," in *Proc. 2nd Int. Workshop Energy-Efficient Data Centers*, 2013.
[35] P. Kahn and M. Mashaly, "Automatic energy efficiency management of data center resources by load-dependent server activation and sleep modes," in *Proc. 2nd Int. Workshop Energy-Efficient Data Centers*, 2013.
[36] K. Wang, M. Lin, F. Ciucu, A. Wierman, and C. Lin, "Characterizing the impact of the workload on the value of dynamic resizing in data centers," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 1, pp. 405–406, Jun. 2012.
[37] T. Heath, B. Diniz, E. Carrera, W. Meira, and R. Bianchini, "Energy conservation in heterogeneous server clusters," in *Proc. 10th ACM SIGPLAN Symp. Principles Pract. Parallel Program.*, New York, NY, USA, 2005, pp. 186–195.
[38] X. Wang and Y. Wang, "Coordinating power control and performance management for virtualized server clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 2, pp. 245–259, May 2010.
[39] J. F. Botero *et al.*, "A data center control architecture for power consumption reduction," in *Proc. 2nd Int. Workshop Energy-Efficient Data Centers*, 2013.
[40] T. Enokido, A. Aikebaier, and M. Takizawa, "Process allocation algorithms for saving power consumption in peer-to-peer systems," *IEEE Trans. Ind. Electron.*, vol. 58, no. 6, pp. 2097–2105, Jun. 2011.
[41] V. Mathew, R. K. Sitaraman, and P. J. Shenoy, "Energy-aware load balancing in content delivery networks," in *Proc. INFOCOM*, 2012, pp. 954–962.
[42] A. Lee, "Energy data center," in *Proc. 2nd Int. Workshop Energy-Efficient Data Centers*, 2013.
[43] P. Padala *et al.*, "Automated control of multiple virtualized resources," in *Proc. 4th ACM Eur. Conf. Comput. Syst.*, New York, NY, USA, 2009, pp. 13–26.
[44] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.
[45] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th ICML*, 1996, pp. 148–156.
[46] L. Rokach, "Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography," *Comput. Statist. Data Anal.*, vol. 53, no. 12, pp. 4046–4072, 2009.
[47] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits Syst. Mag.*, vol. 6, no. 3, pp. 21–45, Sep. 2006.
[48] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, Dec. 2007.
[49] A. Pocock, P. Yiapanis, J. Singer, M. Luján, and G. Brown, "Online non-stationary boosting," in *Proc. 9th Int. Conf. MCS*, 2010, pp. 205–214.
[50] M. Muhlbaier, A. Topalis, and R. Polikar, "Learn $^{++}$.NC: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 152–168, Jan. 2009.
[51] S. Chen and H. He, "Towards incremental learning of non-stationary imbalanced data stream: A multiple selectively recursive approach," *Evol. Syst.*, vol. 2, no. 1, pp. 35–50, 2011.
[52] N. Singh and S. Rao, "Meta-learning based architectural and algorithmic optimization for achieving green-ness in predictive workload analytics," in *Proc. 28th Annu. ACM Symp. Appl. Comput.*, Mar. 2013, pp. 1169–1176.
[53] N. Singh and S. Rao, "Online ensemble learning approach for server workload prediction in large datacenters," in *Proc. 11th ICMLA*, Boca Raton, FL, USA, Dec. 2012, pp. 68–71.
[54] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Cambridge, MA, USA: MIT Press, 2012.
[55] G. Box, G. M. Jenkins, and G. Reinsel, *Time Series Analysis: Forecasting and Control*, 3rd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1994.
[56] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. N. Vapnik, "Support vector regression machines," in *Advances in Neural Information Processing Systems 9*. Cambridge, MA, USA: MIT Press, 1996, pp. 155–161.
[57] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997.
[58] A. Smola and S. V. N. Vishwanathan, *Introduction to Machine Learning*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

NIDHI SINGH has worked with Xerox Research Center Europe and IBM Labs for several years, and has been involved in research and development of innovative algorithms for optimization of servers' energy consumption in large scale systems. Her research interests include supervised machine learning, online learning, ensemble methods, and convex optimization as applied to green computing (especially for data centers) and short-text analytics. She has received several accolades for her work in these areas. She is a member of the IEEE, the ACM, and the Computer Society of India.

SHRISHA RAO received the Ph.D. degree in computer science from the University of Iowa, and the M.S. degree in logic and computation from Carnegie Mellon University. He is an Associate Professor with the International Institute of Information Technology-Bangalore. His research interests include distributed computing, specifically algorithms and approaches for concurrent and distributed systems, and include solar energy and microgrids, cloud computing, energy-aware computing (green IT), and demand side resource management. He is a member of the IEEE Computer Society, the Association for Computing Machinery, the American Mathematical Society, and the Computer Society of India.