Multi-step Prediction of Worker Resource Usage at the Extreme Edge

Ruslan Kain
School of Computing, Queen's University
Kingston, ON, Canada
r.kain@queensu.ca

Sara A. Elsayed
School of Computing, Queen's University
Kingston, ON, Canada
selsayed@cs.queensu.ca

Yuanzhu Chen
School of Computing, Queen's University
Kingston, ON, Canada
yuanzhu.chen@queensu.ca

Hossam S. Hassanein
School of Computing, Queen's University
Kingston, ON, Canada
hossam@cs.queensu.ca

ABSTRACT

Democratizing the edge by leveraging the prolific yet underutilized computational resources of end devices, referred to as Extreme Edge Devices (EEDs), can open a new edge computing tech market that is people-owned, democratically managed, and accessible/lucrative to all. Parallel computing at EEDs can also move the computing service much closer to end-users, which can help satisfy the stringent Quality-of-Service (QoS) requirements of delay-critical and/or data-intensive IoT applications. However, EEDs are heterogeneous user-owned devices, and are thus subject to a highly dynamic user access behavior (i.e., dynamic resource usage). This makes the process of determining the computational capability of EEDs increasingly challenging. Estimating the dynamic resource usage of EEDs (i.e., workers) has been mostly overlooked. The complexity of Machine Learning (ML)-based models renders them impractical for deployment at the edge for the purpose of such estimations. In this paper, we propose the Resource Usage Multi-step Prediction (RUMP) scheme to estimate the dynamic resource usage of workers over multiple steps ahead in a computationally efficient way while providing a relatively high prediction accuracy. Towards that end, RUMP exploits the use of the Hierarchical Dirichlet Process-Hidden Semi-Markov Model (HDP-HSMM) to estimate the dynamic resource usage of workers in EED-based computing paradigms. Extensive evaluations on a real testbed of heterogeneous workers for multi-step sizes show an 87.5% prediction accuracy for the starting point of 2-steps and coming to as little as a 16% average difference in prediction error compared to a representative of state-of-the-art ML-based schemes.

CCS CONCEPTS

• **Computing methodologies** → **Model verification and validation**; • **Human-centered computing** → *Empirical studies in ubiquitous and mobile computing*.

MSWiM '22, October 24–28, 2022, Montreal, QC, Canada

ACM ISBN 978-1-4503-9479-6/22/10.
https://doi.org/10.1145/3551659.3559051

KEYWORDS

Multi-access Edge Computing, Extreme Edge, Edge Democratization, Hidden Semi-Markov Model, LSTM, Dynamic Resource Usage

ACM Reference Format:
Ruslan Kain, Sara A. Elsayed, Yuanzhu Chen, and Hossam S. Hassanein. 2022. Multi-step Prediction of Worker Resource Usage at the Extreme Edge. In *Proceedings of the International Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM '22), October 24–28, 2022, Montreal, QC, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3551659.3559051

1 INTRODUCTION

With the pervasive proliferation of the Internet of Things (IoT), 23.3 billion IoT devices are expected to be connected to the Internet by 2025 [7]. This substantial growth is expected to impose unprecedented demands on computing resources to satisfy the stringent Quality of Service (QoS) requirements associated with delay-critical and/or data-intensive IoT applications [12]. Such requirements can be challenging to meet in Cloud Computing (CC) due to the need to transmit massive amounts of data to remote data centers, which significantly increases latency and thrusts a large traffic load at backhaul links [18].

Muti-access Edge Computing (MEC) has emerged as a propitious computing paradigm that can alleviate the aforementioned issues by offering the computing service closer to end-users [19]. However, the vast majority of existing MEC models and platforms are dependent on infrastructure-based edge nodes that are exclusively governed by cloud service providers and/or network operators [20, 26]. Breaking this monopoly can be achieved by tapping into the profuse yet underutilized computational resources of Extreme Edge Devices (EEDs) such as PCs, Laptops, tablets, smartphones, and connected vehicles. Parallel processing at EEDs can democratize the edge and permit more players, such as Distributive[1], to establish and administer their own edge cloud [1, 5, 6, 16]. Edge Democratization can pave the way for a new tech market in edge computing that is people-owned, democratically managed, and accessible/lucrative to all. In addition, fostering parallel processing in EED-enabled computing environments can help bring the computing service even closer to end-users. This can drastically curtail the delay and facilitate significant compute-intensive and

[1] https://kingsds.network

latency-sensitive applications, such as virtual and augmented reality, autonomous vehicles, smart-grids, and smart-cities [20].

Despite the promising potential of EED-enabled computing, the viability of such a system is hindered by the heterogeneous nature of EEDs (i.e., workers) and the fact that EEDs are user-owned devices, making them subject to a highly dynamic user access behavior. In particular, users can access their devices at any given time to run an intensive application, such as streaming a video, playing a video game, augmented reality, etc. This can dynamically alter and affect the devices' available computational resources. Note that these resources are affected by the demands of the applications run, the operational conditions set, and the limits defined by the device's specifications [3]. Consequently, EED-enabled computing paradigms cannot rely on the same resource characterization techniques, methods, and algorithms used in distributed computing over infrastructure-based edge nodes or cloud servers. In fact, successful, consistent, and reliable task offloading and resource allocation services in EED-enabled computing that can adapt to the dynamic environment require new intelligent scheduling methods [22], and resource characterization techniques [11].

Resource characterization in EED-enabled computing environments, which requires running benchmark tasks on the devices, is used to estimate the performance of these dynamically accessed user-owned devices. The performance estimation, in terms of job completion time or throughput, would enable computational jobs scheduling algorithms to reduce delays, as shown in [11]. Thus, for EED-enabled computing environments, it is necessary to characterize the worker in each possible resource usage state with several benchmark tasks. As such, there is a need to identify the resource usage states and run the benchmarks in a timely manner to correctly characterize the worker. Such characterization can enable a "Dynamic Usage-Aware" scheduler to take into account the resource usage state of the worker when allocating tasks.

Resource characterization that relies on capturing and estimating the highly dynamic resource usage in EED-based systems has been mostly overlooked. In addition, predictive models that are proactively used in infrastructure-based edge computing paradigms suffer from various drawbacks, namely high complexity in Machine Learning (ML)-based models [25] and low prediction accuracy in system identification-based methods [24]. Such drawbacks render them impractical to cope with the highly dynamic nature of user access behavior in EEDs. In this paper, we propose the Resource Usage Multi-step Prediction (RUMP) scheme. RUMP incorporates the use of the Hierarchical Dirichlet Process-Hidden Semi-Markov Model (HDP-HSMM) to predict the resource usage state of EEDs over multiple steps ahead.

In RUMP, the HSMM is used to proactively predict the resource usage state of the worker for a given period, more or less equivalent to the expected run-time of the benchmark, which necessitates multi-step prediction of the worker resource usage. In order to run multiple types of benchmark programs of various run-times, flexibility in the step-ahead prediction size is vital. The HSMM is used because it 1) inherits the accuracy of Hidden Markov Models (HMMs) and the added time-dimension predictability of Semi Markov Models (SMMs) and 2) requires only a single model for multi-step predictions making it more effective for inference in the context of EEDs, unlike ML models that require a separate model per step size.

We evaluate RUMP compared to a good representative of the state-of-the-art machine learning-based methods [23], in terms of mean absolute error and root mean square error. We also conduct a complexity analysis to study the complexity of RUMP compared to other methods. In contrast to most existing works in the literature that rely on simulations [13], we conduct extensive evaluations on a real testbed of heterogeneous workers to analyze the performance of RUMP for each of the resource usage patterns of workers and multi-step prediction accuracy.

Our contributions can be summarized as follows:

- Estimating worker resource usage to capture the highly dynamic user access behavior in EED-based systems and enable resource characterization. To the best of our knowledge, RUMP is the first scheme in EED-based computing environments that enables multi-step prediction of resource usage.
- In contrast to existing complex ML-based models that are typically used in infrastructure-based computing paradigms, RUMP enables practical and efficient estimation of resource usage to cope with the highly dynamic nature of EED-based systems by using the Hierarchical Dirichlet Process-Hidden Semi-Markov Model (HDP-HSMM).
- Employing a performance evaluation method that relies on conducting extensive evaluations on a realistic testbed of heterogeneous workers in different dynamic resource usage scenarios. In addition, we create a dataset of dynamic resource usage associated with running edge-native application labels to further catalyze EED-based research. The created dataset [9] is available to the research community and can also be accessed with the code via GitHub [2].

The remainder of the paper is organized as follows. Section 2 presents an overview of the related work. Section 3 presents a detailed description of RUMP, as well as an analysis of its complexity. Section 4 showcases the performance evaluation and simulation results. Finally, section 5 concludes the paper and suggests future research directions.

2 RELATED WORK

Methods for resource usage modeling and prediction, used as part of intelligent resource management mechanisms have been proposed for systems ranging from grid computing [21], volunteer computing [17], and cloud computing [14]. More recently, methods tailored for edge computing environments have been proposed for the benefit of management operations such as resource allocation [23]. However, coping with the highly dynamic user access behavior at EEDs and estimating the dynamic resource usage of workers in EED-enabled computing environments, has been mostly overlooked.

Existing resource characterization mechanisms in infrastructure-based edge paradigms can be categorized into reactive and proactive techniques [15]. Reactive techniques adapt to shifting resources by modifying computing task schedules and resource allocations when shifts are detected, some relying on system-identification [24], and others on threshold-based techniques [21]. In contrast, proactive techniques adapt to the anticipated future resource shifts

[2] https://github.com/RuslanKain/rump-ec

before they occur, predicted via machine learning [15, 23] and state-based methods [17]. Both reactive and proactive techniques enable adaptation to resource usage variability and availability at different expenses. Proactive techniques have been shown to achieve a much higher prediction accuracy, at the expense of higher computational complexity, compared to reactive techniques [15].

Proactive techniques that include the use of machine learning (ML) models and state-based models can indeed capture long-term dependencies and recurrence in the system. In Rusty [15], an LSTM is trained to predict lower-level architectural events and system-level characterization under interference to gain insights into the system state and guide the task scheduler. In [23], a mix of Long-Short Term Models (LSTM) and Convolutional Neural Networks (CNN) are used to predict the resource usage, where the architecture of the models is automatically designed to maximize performance using hyper-parameter search optimization methods, performing better and competitively in comparison to other ML models such as Support Vector Regression, Multiple Linear Regression, XGBoost, Deep Neural Networks. In particular, [23] uses a combination of Bayesian and particle swarm optimization for ML hyper-parameter search. Despite their high prediction power, such ML-based techniques are significantly complex and computationally intensive [25]. Thus, they suffer from long training times, and low adaptability to resource usage patterns continually changing in real-time. Consequently, they are rendered impractical for EED-based systems. In addition, ML-based models have limited usability within the context of EED-based systems due to their reliance on 1-step ahead predictions that fail to capture the highly dynamic nature of such systems. Multi-step ahead prediction of CPU loads using a pattern matching technique has been proposed in distributed systems [24]. However, it is unfit for multi-variate prediction of resource usage in EED-based systems, due to the explosive number of pattern combinations, which impacts predictive capabilities [8]. Using ML-based models for multi-step prediction has more potential. However, as previously mentioned, their drawbacks make them impractical for EED-based systems.

In contrast to existing schemes, our proposed RUMP scheme is tailored to model and capture the dynamic resource usage in EED-enabled computing environments by exploiting the use of the HDP-HSMM. HSMM combines the accuracy of Hidden Markov Models (HMMs) and the added time-dimension predictability of Semi-Markov Models (SMMs) that was shown to resource usage of computing systems [17]. The HSMM can be a more practical option for use in EED-based systems compared to ML since it 1) relies on estimating state transition probabilities between different statistical distributions and can thus render a lower complexity and impose a lower workload on an EED, 2) is flexible, since a single model is used for multi-step prediction for different step sizes, making it appropriate for benchmark tasks used for resource characterization, and 3) is adaptable, since new data can be continually input to the distribution statistics of the state without requiring significant re-training efforts [11]. As opposed to existing schemes that rely on simulations, we employ a performance evaluation method that relies on a realistic testbed of heterogeneous workers, and generate a dataset of dynamic resource usage in EED-based systems.

3 RESOURCE USAGE MULTI-STEP PREDICTION (RUMP)

In RUMP, we model the workers' dynamic resource usage using a Hidden Semi-Markov Model (HSMM) to enable a multi-step prediction of the workers' resource usage state. The implementation uses a Hierarchical Dirichlet Process (HDP) for a Bayesian nonparametric extension of the HSMM [10]. The extension HDP-HSMM enables the model to infer the number of hidden states using a weak-limit Gibbs sampling method without requiring prior knowledge. This is since the workers' usage behavior and the applications they run tend to vary from one user to another. Moreover, the use of explicit duration semi-Markov modeling overcomes the limitation of the resource usage data being non-Markovian, since the time when the next resource usage state is reached depends on the time spent in the current state and not just the state itself. To learn the model, the resource usage information of the worker, such as CPU time and memory percent usage, is collected at a fixed interval over a long enough period to capture all the possible resource usage states.

3.1 Hidden Semi-Markov Model

Much like the Hidden Markov Model, a Hidden Semi-Markov Model is composed of two layers; a hidden state later and an observation/emission layer, represented as random variables. However, HSMM additionally expresses the time duration of states, which is limited to a geometric distribution in HMM. We argue that running an application running on a worker, which from a MEC service provider's side would be similar to a hidden state, has a similar relation to the corresponding resources usage patterns that can be viewed as emission from a distribution of values previously observed. State transition patterns may be modeled probabilistically where each entered state is given an explicit duration, also drawn from a distribution. Such a Semi-Markov model is defined as an explicit duration Semi-Markov model. The observations can be represented by $y_t \in \mathbb{R}_{\geq 0}$ for $t \in \{1, 2, \ldots, T\}$, where T is the length of the observation sequence. The observations, in this case, are the values of the resource usage information such as CPU time, memory percent usage, and network rates. Moreover, the observations, drawn from a distribution corresponding to a hidden Semi-Markov state, are represented by a sequence of random variables $x_t \in \{1, 2, \ldots, N\}$ forming a Markov chain for the sequence of N possible states. The hidden states, in this case, represent the resource usage states associated with applications and processes running on the EED. The state transition matrix, which collects the transition probabilities in a row-stochastic matrix, is denoted by $\pi_{ij} = p(x_{t+1} = j | x_t = i)$. The emission distribution with parameters $\{\theta_i\}$ is represented by the probability $p(y_t | x_t, \theta_i)$.

HSMM is augmented with a random variable, denoted D_t representing state duration time drawn from a distribution specific to the entered state with the probability mass function $p(d_t, | x_t = i, , \omega_i)$, where $\{\omega_i\}$ are the duration distribution parameters. Once D_t of an entered state passes, a Markov transition occurs towards a different state. To simplify representation, a sequence of the same state is represented by "super-states" $z_s \in \{1, 2, \ldots, S\}$, for S possible super-states, which emits observations within the time D_t, thus avoiding state self-transition of super-states. More details of how the self-transitions are eliminated using an additional auxiliary

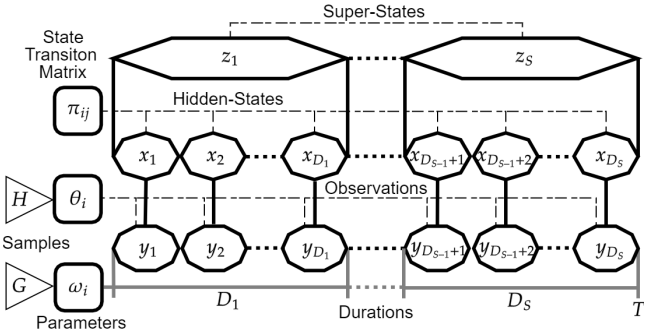


Figure 1: Components of the Hierarchical Dirichlet Process - Hidden Semi-Markov Model

variable can be found in [10]. Note that, we assume that the models are time-homogeneous. This means that the transition probabilities do not change with time. However, this can be modified using extensible Markov Models [4]. The worker dynamic resource usage scenario would allow the models to adapt with time to often regularly evolving worker usage behaviors. The state transition probabilities of the model are derived from a given sequence of observations/emissions using the standard message-passing inference of the forward-backward algorithm, also called smoothing.

3.2 Hierarchical Dirichlet Process

The HSMM generation method used relies on a hierarchical Dirichlet process (HDP) [10] which allows for Bayesian non-parametric inference of the hidden states and duration distributions, instead of the usual treatment from a non-Bayesian perspective by approximating parameters using the Expectation-Maximization algorithm. As such, the model parameters are treated as random variables having HDP priors $p(\pi|\alpha)$, $p(\{\theta_i\}|H)$, and $p(\{\omega_i\}|G)$, where $\alpha > 0$ is a concentration parameter specifying the shape of the distribution and where H and G are the base measures used to parameterise the emission distribution and duration distribution, respectively. The inference approach enables the modeling of uncertainty over the parameters and the prediction of observations and state sequences by integrating out all possible parameters. Thus, HDP acts as a prior over infinite-state transition matrices biased by a set of states that are consistently re-entered in each smoothing iteration, i.e., the draw of the forward-backward message passing algorithm, which is itself parameterized by a discrete measure β. The β parameter penalizes large numbers of states and reduces them to the states consistently visited in the sequence. A graphical representation of all components of HDP-HSMM is shown in Figure 1. A weak-limit Gibbs sampler is used to completely represent the transition matrix in a finite form based on L-Dimensional Dirichlet distributions. The L parameter can be sampled over without being set beforehand using the beam sampling technique as in [2]. Essentially the sampler constructs the parameters by drawing samples from the posterior probability $p(\{x_t\}, \{\theta_t\}, \{\pi_t\}, \{\omega_t\}|y_t, H, G, \alpha)$. Moreover, the sampler also accelerates mixing, i.e. estimating contributions of sources to a mixture, by allowing for block sampling of the entire state sequence simultaneously. Lastly, a parameter κ is introduced to allow for some control over duration statistics, i.e. encouraging longer or shorter periods, known as the sticky property.

3.3 Complexity Analysis

The most computationally intensive part of HDP-HSMM results from the message passing step using the forward-backward algorithm, i.e. the smoothing process. The algorithm has a complexity of $O(T^2 N_{\text{States}} + T N_{\text{States}}^2)$, where N_{States} is the number of states and T is the observation sequence length. However, not all steps in the sequence need to be considered, thus it is enough to consider the steps where a change in the super-state is most likely, i.e. the change points. The complexity is reduced to $O(T_{\text{Change}}^2 N_{\text{States}} + T_{Change} N_{\text{States}}^2)$ if change-point detection is run when using the weak-sampler on the observations, where T_{Change} is the number of possible change points, which is much less than T. We reinforce the aforementioned efficiency and lower complexity of the HDP-HSMM by comparing it to the baseline machine learning approach for resource usage prediction called Hybrid Bayesian Particle Swarm Hyper-Parameter Optimization (HBPSHPO) [23]. The LSTM-CNN based model in HBPSHPO uses the Particle Swarm meta-heuristic (PS) to optimize the model architecture until it stops according to termination criteria with value C. Thus, the number of PS rounds is a function of the termination criteria value $N_{\text{PSO}} = f(C)$. HBPSHPO also uses Bayesian optimization to optimize the selection of activation and loss functions within N_{BO} number of iterations per particle in the PS. Moreover, an LSTM-CNN model needs to be trained for each step size, numbered N_{Steps}, and so the number LSTM-CNN models trained is $N_{\text{Models}} = N_{\text{PSO}} \times N_{\text{BO}} \times N_{\text{Steps}}$. Each model has a complexity of $O(UT)$, where U is the number of LSTM-CNN units and, as before, T is the input data sequence length. It follows that the complexity of the HBPSHPO method is $O(TUN_{\text{Models}})$, which exceeds the complexity of the HDP-HSMM of $O(T^2 N_{\text{States}})$ for a sizeable observation sequence length T, in our case on the order of 10^4, since $U \gg T \gg T_{\text{Change}} \gg N_{\text{Models}} \gg N_{\text{States}}$ where the number of LSTM-CNN units U is on an order of magnitude of 10^9.

4 PERFORMANCE EVALUATION

We evaluate RUMP in comparison to a state-of-the-art hyper-tuned LSTM/CNN model, called Hybrid Bayesian Particle Swarm Hyper-parameter Optimization Model (HBPSHPO) [23], in terms of the average mean absolute error and root mean square error of the user, system, and idle time in seconds. As well as the memory percent usage, for both random and patterned state sequences, over different step sizes. The original HBPSHPO model was originally implemented for 1-step ahead prediction, but for the experiments, we modified it to enable multi-step ahead predictions. In RUMP, the resource usage information (i.e., CPU Time, memory usage, etc.) are represented as observations, while the resource usage states (i.e., "game," "mining", "idle", etc.) are modeled as hidden states. As for the HDP-HSMM parameters in RUMP, they may all be derived from the data. However, the κ parameter and the number of sampling iterations of the weak Gibbs sampler need to be selected, which may impact the HSMM model. The parameters may be selected using a grid search method, but once they are selected for the model, the same values can be used for all other models, unlike the hyper-parameters of the HBPSHPO model, which need to be optimized for every training cycle. The values of κ and the number of sampling iterations used for the experiments are 0.1 and 800, respectively. The evaluations of the models, trained on the training data, are

performed by applying inference of the observations in the testing data for the different step sizes.

4.1 Experimental Setup

To test our proposed method on heterogeneous workers representative of EEDs in different dynamic usage scenarios. We use a set of four computers, the Raspberry Pi (RPi) 4B model, of different RAM sizes and CPU cycle frequencies. The processor is a 64-bit quad-core Cortex-A72 (ARM v8). Note that an RPi 4B model's regular CPU frequency is 1.5 GHz, however to increase heterogeneity, the RPi is overclocked and throttled. As such, the CPU frequencies used are 1.8, 1.5, and 1.2 GHz, and the RAM sizes are 8, 4, and 2 GB. The specifications of the workers are described in Table 1.

Table 1: Worker specifications and labels

Worker Label	Raspberry Pi 4B Specifications	
	RAM Size (GB)	CPU Cycle Freq. (GHz)
A	8	1.8
B	4	1.5
C	2	1.5
D	2	1.2

A dynamic usage scenario is implemented by automatically running a set of applications in sequence for different periods. The applications are a video game, streaming a Youtube video on a browser, emulating real-time augmented reality, and mining a crypto-currency. The augmented reality application is emulated by continuously imposing on ArUco markers detected on an input of a sequence of image frames, much like a live video stream. The crypto-currency mining is for the Duino-coin, which is designed for low-power devices, such as Arduinos and RPis. This is since the coin uses the easily implemented SHA-1 cryptographic function for encryption on a variant of a blockchain, called a hash-chain. Additionally, the computer is left idle, i.e. no application runs, for some periods. To create varying usage scenarios (i.e., sequence and periods of running applications and idling, which we refer to as resource usage states), two types of sequence are used for each worker; a random sequence, and a patterned sequence. The patterned sequence consists of a recurrent sequence of applications that is repeated over a specific period. We generate four resource usage information datasets for each worker over a 48-hour duration, a 5-second monitoring interval, and two pairs of sequence types labeled "random" and "patterned". The resource usage information is collected as user, system, and idle CPU time, memory percent usage, network upload and download size and rates, disk IO, etc. For modeling, we focus on the user, system, idle CPU time, and memory percent usage. Note that other resources can be modeled as well. The resource usage information is associated with the resource usage states in the datasets using the labels "game", "stream", "augmented reality", "mining", and "idle". To input the labels into the prediction model, they are given a numerical label. The step sizes used are 1, 2, 5, 10, 15, 30, and 60 steps, which are equivalent to 5, 10, 25, 50, 75, 150, and 300 seconds, respectively. Moreover, the datasets are divided into training and testing sets, to use the same terms of ML modeling, using a 70-30% split.

4.2 Results and Analysis

We first evaluate RUMP and HBPSHPO in terms of the average mean absolute error and root mean square error of the user, system, and idle time in seconds, as well as the memory percent usage, for both random (R) and patterned (P) state sequences, over different step sizes. As shown in Figure 2, there is a gradual increase in error for larger step sizes for both models. RMSE is always larger than MAE. This is since RMSE is more influenced by outlier values (i.e., values much larger than the average). Moreover, it is observable that the difference between RMSE and MAE is larger for the predictions made by RUMP than those made by HBPSHPO, which indicates that there are more inaccurate outlier values predicted by RUMP. The large differences way the inference is done by RUMP, which creates predictions of observations drawn from a probability distribution rather than a more precise function that maps inputs to predicted outputs. This does not mean that the inferences made by RUMP are less useful since the hidden state corresponding to the predicted observation may still be correct, which is corroborated by the results shown in Figure 3 (as explained later).

Figure 2(a) and Figure 2(b) show that the predictions for the user and idle CPU times have similar patterns when comparing the two models for the different state sequence types. Meanwhile, Figure 2(c) and 2(d) show that the predictions using RUMP for the system CPU time and memory percent usage are similar in that the error for the random sequence is less than the patterned sequence, but not for the predictions made using HBPSHPO. Except when predicting the memory percent usage for step sizes 15, 30, and 60. Thus, HBPSHPO sees an improved accuracy for the sequence of states that follows a repetitive pattern, while RUMP does not improve as much. This indicates that in most cases the HBPSHPO model might be overfitting to the patterned sequence data, thus leading to a gain in accuracy at the expense of model generalization. The variance in MAE for RUMP is on average 31.54 % for both the random (26.09%) and patterned (36.99%) sequence data. In contrast, for the HBPSHPO model, the average variance is 35.06% for both the random (36.82%) and patterned (33.29%) sequence data. It follows that the predictive power of RUMP is comparable to the more computationally complex HBPSHPO model, with an overall average percent difference of 28.03% in MAE and 66.01% in RMSE. The average percent difference between RUMP and HBPSHPO for the patterned sequence data is 39.13% and 72.78% in terms of MAE and RMSE, respectively, and 16.93% and 59.25% in MAE and RMSE, respectively for the random sequence data, in favor of the HBPSHPO model.

Figure 3 presents a time-series plot of the predictions made using HBPSHPO (Figure 3(a)) and RUMP (Figure 3(b)), taking 5-step prediction of the user CPU time on worker C as a sample. As shown in the Figure, the predictions made by RUMP tend to fluctuate more than those made by HBPSHPO, which explains the higher RMSE, but nonetheless closely follow the level of the actual observations, which explains the lower values of MAE.

Figure 4 depicts the distribution of the percent difference in MAE and RMSE for each of the state sequence types (random and patterned) when comparing the predicted observations of RUMP to that of HBPSHPO. Note that a positive percent value indicates that HBPSHPO has a lower error and vice versa. The results show an average of 16% and 40% difference in MAE in favor of the HBPSHPO

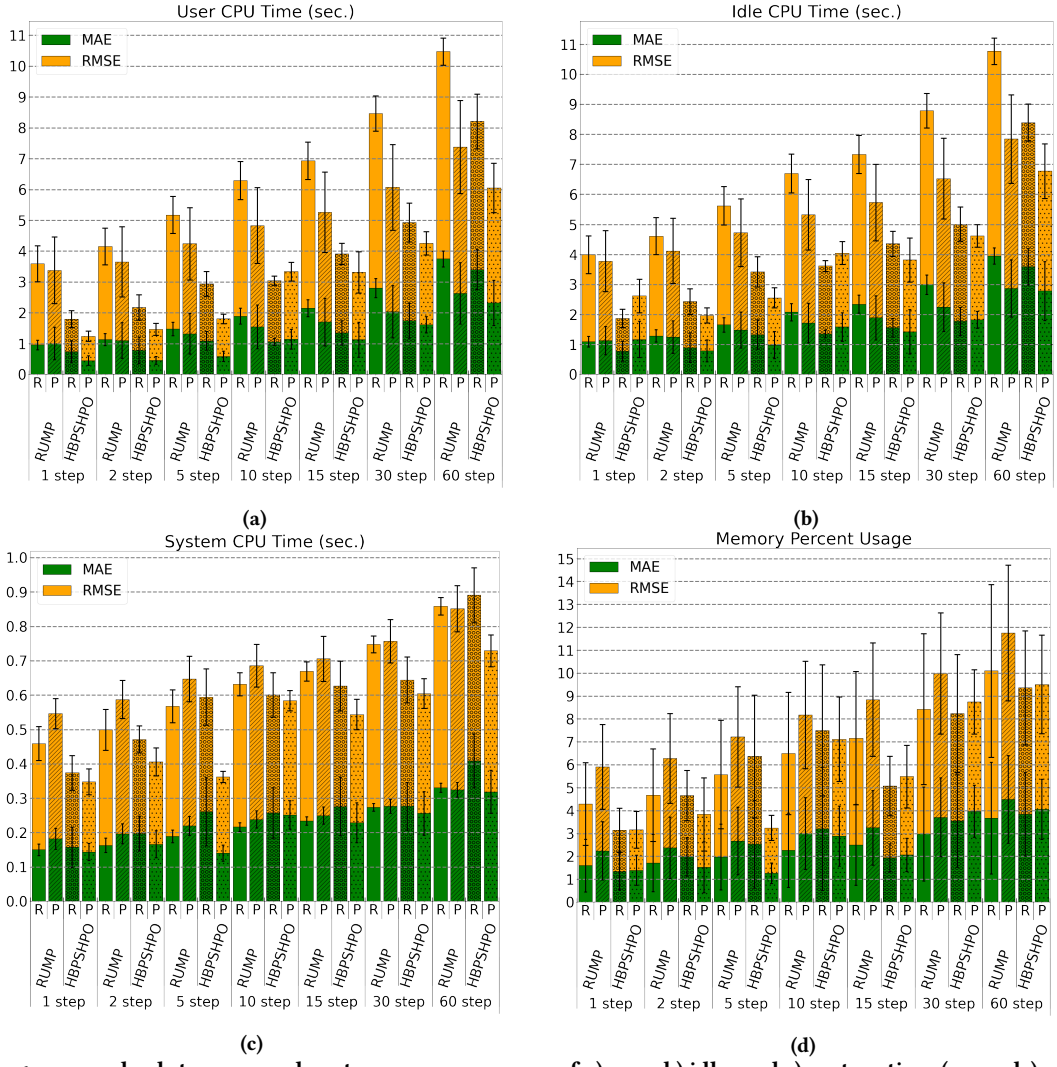


Figure 2: Average mean absolute error and root mean square error of a) user, b) idle, and c) system time (seconds) and d) memory percent usage predicted using RUMP and HBPSHPO for all workers for both random (R) and patterned (P) state sequences

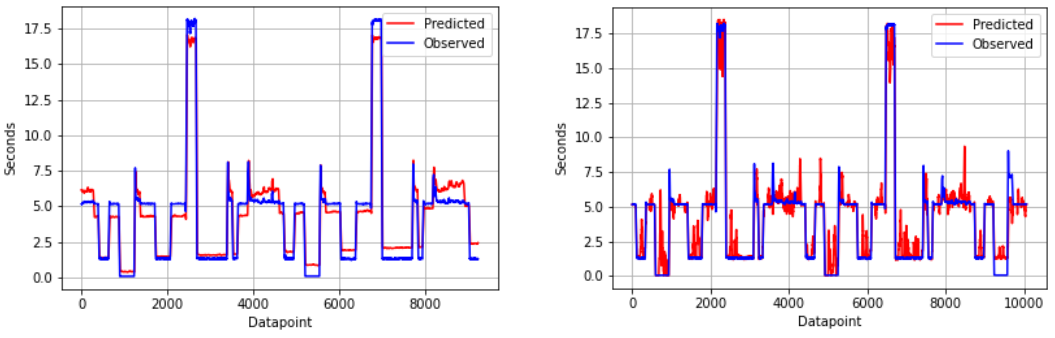


Figure 3: Observed (blue) and predicted (red) user CPU time for worker C and by a) HBPSHPO b) and RUMP

for the random and patterned sequence data, respectively. Moreover, the results show an average of 60% and 70% difference in RMSE in favor of the HBPSHPO for the random and patterned sequence data, respectively. In some cases, RUMP even outperforms HBPSHPO, for example, the minimum difference in MAE for the random sequence

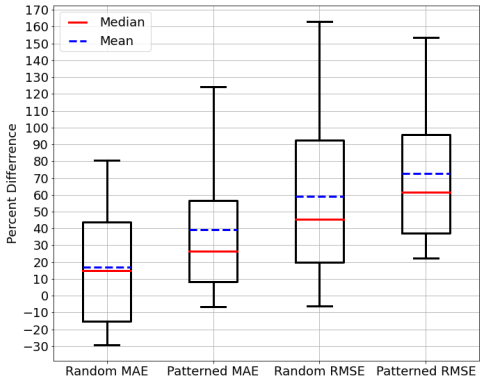


Figure 4: Percent error difference in MAE and RMSE for resource usage prediction using RUMP and HBPSHPO, the blue dashed line is the mean and the red line is the median

data is -30% (i.e., in favor of RUMP). Reflecting on the results obtained, in general, the accuracy of RUMP overall remains lower than HBPHSPO but nonetheless benefits from lower computational complexity and higher efficiency. This is since HBPSHPO is ML-based, and thus has better predictive powers, but at the expense of extra training, since each step-ahead size requires a new model, and re-training must be applied when new resource usage data is collected. Moreover, HBPSHPO renders higher complexity (as illustrated in Section 3.3), due to the number of hyper-parameters searches for the model architecture optimization technique.

Lastly, RUMP's accuracy in predicting the resource usage state of each worker is evaluated in terms of percent accuracy matched to the dataset's true labels. The resource usage state of a worker in HSMM terms is represented by the hidden state. However, the predicted hidden states do not necessarily match the labels that relate to the running applications. This is since other worker background processes may run at any time, thus impacting the resource usage. For simplification, we assume that the hidden state predicted by RUMP that most often coincides with the dataset label is the predicted resource usage state. The results of the evaluation for each worker are shown in Figure 5 for both the training and testing data and for the random 5(a) and patterned 5(b) sequence data over the different step sizes for all workers. Note that the HSMMs in RUMP are generated for 1-step size predictions, and thus the state prediction accuracy for the training data corresponds to 1-step prediction only and is represented by the horizontal lines. The scattered dots indicate the accuracy achieved when using the model to infer on the testing data for each step size. The average training-phase accuracy for all workers for random and patterned state sequences shown in Figure 5(a) is 95.3% and in Figure 5(b) is 93.8%. As for the multi-step inference on the unseen testing data for random state sequences the average prediction accuracy starts from 90.4% for a 1-step size, 88.4% for a 2-step size, 78% for a 10-steps prediction, and down to 55% for a 60-steps size. While for the patterned state sequences data, the accuracy starts from 88.5% for a 1-step size, 86.6% for a 2-step size, 78.6% for a 10-steps prediction, and down to 59.6% for a 60-steps size. The decrease in prediction accuracy has a reduced steepness for longer step sizes, which suggests a non-linear relation with step size due to the error propagation from the prediction step to the next. In Figure 5b, the accuracy results of workers B and D are slightly lower starting accuracy, i.e. for 1-step prediction.

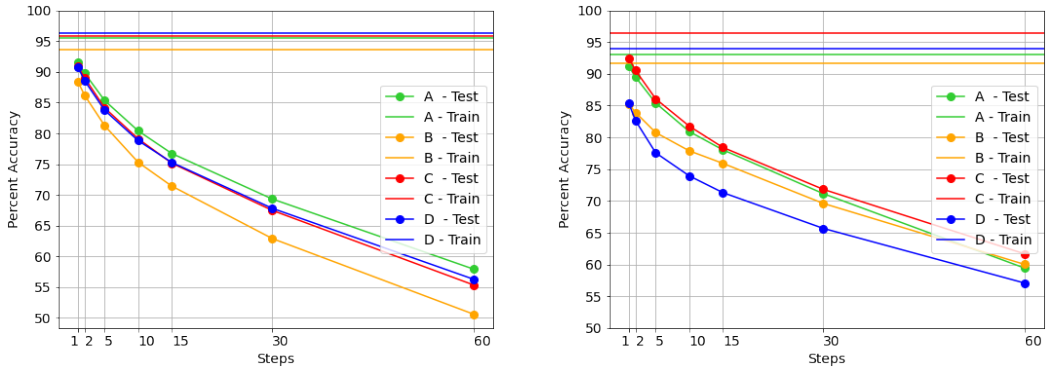
Table 2: Distribution of 1-step predictions in terms of percent correct state prediction (✓), incorrect prediction as other known states (×), and prediction of unknown states (?) for testing-phase patterned sequence data for "Stream", "Game", and "Mining" resource usage states for all workers

Worker Label	Resource Usage State								
	Stream			Game			Mining		
	✓	×	?	✓	×	?	✓	×	?
A	81.1	9.8	9.1	96.6	0.6	2.8	77	2.7	20.3
B	77	2	21	64.3	17.7	18	83.4	7.1	9.5
C	85.1	2.9	12	92.6	0.4	7	92.4	2.5	5.1
D	67.2	18.3	14.5	88.5	2.7	8.8	61.6	9.7	28.7

Table 2 helps explain the source of the aforementioned deviations by presenting the 1-step prediction accuracy on the unseen patterned sequence data for the "Steam", "Game", and "Mining" resource usage states for each worker. The table shows that accuracy is relatively low in the "Steam" state for workers B and D, "Game" for worker B, and "Mining" state for workers A and D. Besides having the state incorrectly classified as a different resource usage state, another source of error is the prediction of a hidden state that does not directly correspond to the labeled resource usage state. This is due to the model generating "extra" hidden states that are impacted by some unknown processes instead of the running application. These processes may include automatic operating system processes which may run at any time or the application itself uses resources abnormally due to the application's operation being affected by an overloaded system. All of the resource usage states have been sometimes predicted as these "unknown" hidden states, in addition to incorrect predictions as other hidden states. This does not mean that the model is inaccurate, but instead, the issue is that the labels in the dataset do not completely capture the unknown resource usage states reflected in the resource usage patterns which also impact a worker's performance that the model can detect. Therefore, the model's prediction accuracy would increase if the resource usage associated with the unknown processes is successfully labeled.

5 CONCLUSION

In this paper, we have proposed the Resource Usage Multi-step Prediction (RUMP) scheme to model and predict the resource usage of heterogeneous workers in EED-enabled computing environments. RUMP deals with the highly dynamic user access behavior in such environments by using a Hierarchical Dirichlet Process-Hidden Semi-Markov Model (HDP-HSMM) to provide a relatively high predictive power without sacrificing computational efficiency or rendering too much complexity. Extensive evaluations on a realistic testbed have shown that RUMP is comparable to a representative of the state-of-the-art machine learning-based models in terms of multi-step and multi-variant prediction accuracy, while rendering a much lower computational complexity. In particular, evaluations have shown that RUMP yields a 16% gap in prediction error in comparison to the more complex ML-based model. Further performance studies have substantiated the modeling capability of RUMP for workers' dynamic resource usage patterns, starting with a 87.5%



(a) (b)

Figure 5: Multi-step prediction accuracy for training and testing phase for all workers for a) random and b) patterned sequences

accuracy for 2-step prediction. In the future, we plan on further improving the prediction accuracy by using predictive modeling and analytics that exploit both historical and real-time data. Moreover, conduct further performance evaluations compared to other resource usage prediction methods and formulate an economical model to handle the trade-off between accuracy and efficiency when selecting prediction models.

ACKNOWLEDGMENTS

This research is supported by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant number: ALLRP 549919-20.

REFERENCES

[1] Ibrahim M. Amer and Sameh Sorour. 2022. Cost-based Compute Cluster Formation in Edge Computing. *2022 IEEE International Conference on Communications (ICC): IoT and Sensor Networks Symposium, IEEE ICC'22 - IoTSN Symposium.*

[2] Michael Dewar, Chris Wiggins, and Frank Wood. 2012. Inference in hidden Markov models with explicit state duration distributions. *IEEE Signal Processing Letters* 19, 4 (2012), 235–238.

[3] Swarnava Dey, Arijit Mukherjee, Himadri Sekhar Paul, and Arpan Pal. 2013. Challenges of using edge devices in IoT computation grids. In *2013 International Conference on Parallel and Distributed Systems*. IEEE, 564–569.

[4] Margaret H Dunham, Yu Meng, and Jie Huang. 2004. Extensible markov model. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*. IEEE, 371–374.

[5] Rawan F. El Khatib, Sara A. Elsayed, Nizar Zorba, and Hossam S. Hassanein. 2022. Optimal Proactive Resource Allocation at the Extreme Edge. *2022 IEEE International Conference on Communications (ICC): IoT and Sensor Networks Symposium, IEEE ICC'22 - IoTSN Symposium.*

[6] Ana Juan Ferrer, Joan Manuel Marquès, and Josep Jorba. 2019. Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing. *ACM Computing Surveys (CSUR)* 51, 6 (2019), 1–36.

[7] GSMA. 2022. The mobile economy 2022. https://www.gsma.com/mobileeconomy/

[8] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. 2007. Frequent pattern mining: current status and future directions. *Data mining and knowledge discovery* 15, 1 (2007), 55–86.

[9] Ruslan Kain Sara A. Elsayed Yuanzhu Chen Hossam S. Hassanein. 2022. Resource Usage of Applications Running on Raspberry Pi Devices. http://dx.doi.org/10.5683/SP3/GOZAJE

[10] Matthew James Johnson and Alan S Willsky. 2013. Bayesian nonparametric hidden semi-Markov models. (2013).

[11] Ruslan Kain and Sameh Sorour. 2022. Worker Resource Characterization Under Dynamic Usage in Multi-access Edge Computing. *Accepted in the International Wireless Communications and Mobile Computing Conference - Mobile Computing Symposium* (2022).

[12] Jing Li, Weifa Liang, Wenzheng Xu, Zichuan Xu, and Jin Zhao. 2020. Maximizing the Quality of User Experience of Using Services in Edge Computing for Delay-Sensitive IoT Applications. In *Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems* (Alicante, Spain) *(MSWiM '20)*. Association for Computing Machinery, New York, NY, USA, 113–121. https://doi.org/10.1145/3416010.3423234

[13] Quyuan Luo, Shihong Hu, Changle Li, Guanghui Li, and Weisong Shi. 2021. Resource scheduling in edge computing: A survey. *IEEE Communications Surveys & Tutorials* 23, 4 (2021), 2131–2165.

[14] Karl Mason, Martin Duggan, Enda Barrett, Jim Duggan, and Enda Howley. 2018. Predicting host CPU utilization in the cloud using evolutionary neural networks. *Future Generation Computer Systems* 86 (2018), 162–173.

[15] Dimosthenis Masouros, Sotirios Xydis, and Dimitrios Soudris. 2020. Rusty: Runtime interference-aware predictive monitoring for modern multi-tenant systems. *IEEE Transactions on Parallel and Distributed Systems* 32, 1 (2020), 184–198.

[16] Duncan J. Mays, Sara A. Elsayed, and Hossam S. Hassanein. 2022. Decentralized Data Allocation via Local Benchmarking for Parallelized Mobile Edge Learning. In *2022 IEEE International Wireless Communications and Mobile Computing Conference: IoT and Sensor Networks Symposium, IEEE IWCMC'22*. Dubrovnik, Croatia.

[17] Tessema M Mengistu, Dunren Che, Abdulrahman Alahmadi, and Shiyong Lu. 2018. Semi-markov process based reliability and availability prediction for volunteer cloud systems. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 359–366.

[18] Talal H Noor, Sherali Zeadally, Abdullah Alfazi, and Quan Z Sheng. 2018. Mobile cloud computing: Challenges and future research directions. *Journal of Network and Computer Applications* 115 (2018), 70–85.

[19] Larry Peterson, Tom Anderson, Sachin Katti, Nick McKeown, Guru Parulkar, Jennifer Rexford, Mahadev Satyanarayanan, Oguz Sunay, and Amin Vahdat. 2019. Democratizing the network edge. *ACM SIGCOMM Computer Communication Review* 49, 2 (2019), 31–36.

[20] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge computing: Vision and challenges. *IEEE internet of things journal* 3, 5 (2016), 637–646.

[21] Niyazi Sorkunlu, Varun Chandola, and Abani Patra. 2017. Tracking system behavior from resource usage data. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 410–418.

[22] Kundjanasith Thonglek, Kohei Ichikawa, Keichi Takahashi, Hajimu Iida, and Chawanat Nakasan. 2019. Improving resource utilization in data centers using an LSTM-based prediction model. In *2019 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 1–8.

[23] John Violos, Tita Pagoulatou, Stylianos Tsanakas, Konstantinos Tserpes, and Theodora Varvarigou. 2021. Predicting Resource Usage in Edge Computing Infrastructures with CNN and a Hybrid Bayesian Particle Swarm Hyper-parameter Optimization Model. In *Intelligent Computing*. Springer, 562–580.

[24] Dingyu Yang, Jian Cao, Jiwen Fu, Jie Wang, and Jianmei Guo. 2013. A pattern fusion model for multi-step-ahead CPU load prediction. *Journal of Systems and Software* 86, 5 (2013), 1257–1266.

[25] Yuanshun Yao, Zhujun Xiao, Bolun Wang, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. 2017. Complexity vs. Performance: Empirical Analysis of Machine Learning as a Service. In *Proceedings of the 2017 Internet Measurement Conference* (London, United Kingdom) *(IMC '17)*. Association for Computing Machinery, New York, NY, USA, 384–397. https://doi.org/10.1145/3131365.3131372

[26] Amr M. Zaki and Sameh Sorour. 2022. Proactive Migration for Dynamic Computation Load in Edge Computing. In *2022 IEEE International Conference on Communications (ICC): IoT and Sensor Networks Symposium, IEEE ICC'22 - IoTSN Symposium*. Seoul, Korea.