Dynamic Worker Availability Prediction at the Extreme Edge

Maria Kantardjian, Sara A. Elsayed, Hossam S. Hassanein
School of Computing, Queen's University, Kingston, ON, Canada
m.kantardjian@queensu.ca, selsayed@cs.queensu.ca, hossam@cs.queensu.ca

GLOBECOM 2023 - 2023 IEEE Global Communications Conference | 979-8-3503-1090-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/GLOBECOM54140.2023.10437966

***Abstract*—Leveraging the copious yet underutilized computational resources of end devices, also known as Extreme Edge Devices (EEDs), can significantly enhance the performance of various Internet of Things (IoT) applications. However, EEDs are heterogeneous and user-owned devices, which causes their availability to be highly unreliable. In this paper, we propose the Dynamic Worker Availability Prediction (DWAP) scheme. DWAP is the first scheme that predicts the availability of EEDs (i.e., workers) and adapts to the highly dynamic computing environment at the extreme edge. DWAP employs the Continuous-Time Markov Model (CTMC) to forecast the availability of workers in the upcoming time step. It does so while continuously fine-tuning the model parameters to incorporate newly available data. We use a dataset that consists of real-world Google cluster workload data traces. Extensive evaluations show that DWAP significantly outperforms a representative of state-of-the-art prediction schemes by up to 74% and 59% in terms of the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), respectively. In addition, DWAP yields 97% and 48% reduction in task drop rate compared to prominent availability-unaware and availability-based resource allocation schemes, respectively.**

***Index Terms*—Edge Computing, Extreme Edge, EEDs, Reliability, Availability**

I. Introduction

The widespread adoption of the Internet of Things (IoT) is expected to lead to a surge in connected devices, with an estimated 20 billion IoT devices by 2025 [1], which are expected to be a primary source of the estimated 175 ZB of global data generated [2]. This growth will trigger intensive demands on computational resources for applications such as smart cities, Tactile Internet, autonomous vehicles, and virtual and augmented reality, in order to meet the rigorous Quality of Service (QoS) requirements [3]. Cloud Computing (CC) is insufficient for these demands due to increased latency and congestion at back-haul links [4]. To address these challenges, Edge Computing (EC) has emerged as an auspicious paradigm. In EC, data processing is done at the network edge, significantly reducing communication latency. However, current EC platforms rely on dedicated edge servers that are controlled by cloud service providers and/or network operators [3]. Such monopoly can be evaded by leveraging the underutilized computational resources of Extreme Edge Devices (EED), such as smartphones, connected vehicles, and PCs. This equips more players to develop and manage their own edge cloud, creating a new tech market that is democratically governed, accessible, and remunerative to all [3]–[6].

The positive impact of EED-enabled computing may be hindered due to the unreliable and intermittently available nature of user-owned and heterogeneous EEDs, resulting from factors such as battery exhaustion, network disruption, and dynamic user access behavior. The latter pertains to the notion that at any given time, users may run computationally intensive applications on their devices, such as streaming videos or playing video-games, causing the EED to become unavailable and refrain from performing an offloaded task to preserve its resources for its own convenience. This negatively impacts reliability and increases task drop rate. Thus, availability-aware resource allocation schemes that account for the dynamicity of EEDs are crucial in EED-enabled computing environments.

Existing resource allocation schemes tend to overlook the dynamic availability of EED (i.e., workers). Some schemes allocate high-priority tasks to reliable workers, determined based on the ratio of successful task executions [7], [8]. Other schemes use static probabilistic techniques to predict resource availability by fitting traces of workers to a statistical distribution and calculating parameters for estimating future availability [9], [10]. However, these approaches fail to capture the worker dynamics, leading to suboptimal task offloading decisions. In this paper, we propose the Dynamic Worker Availability Prediction (DWAP) scheme. In contrast to existing schemes, DWAP accounts for the dynamic availability of EEDs. It incorporates the Continuous-Time Markov Chains (CTMC) to predict the availability of workers and adapt to the highly dynamic changes in the environment.

CTMC is a mathematical model that describes stochastic processes, where the system transitions between states over time, based on transition rates [11]. In DWAP, we calculate the transition rates using the Mean Time Between Failure (MTBF) and the Mean Time to Repair (MTTR), which represent a worker's expected lifespan before experiencing a failure and the repair time after failure, respectively. To accurately capture the dynamic changes in the environment, we re-calculate MTBF and MTTR as new data becomes available using a dataset from Google's real-world application usage traces [12]. To the best of our knowledge, DWAP is the first scheme that accounts for the dynamic availability of workers by fostering dynamic availability prediction to allow for better resource allocation decisions that reduce the task drop rate.

The simulation results demonstrate that DWAP outperforms a representative of state-of-the-art prediction schemes, achieving up to 74% and 59% improvement in Mean Absolute

Authorized licensed use limited to: Queen's University. Downloaded on March 05,2024 at 13:19:19 UTC from IEEE Xplore. Restrictions apply.

Error (MAE) and Root Mean Squared Error (RMSE), respectively. Furthermore, when compared to prominent availability-unaware and availability-based resource allocation schemes, DWAP achieves a considerable reduction in task drop rate, with up to 97% and 48% fewer task drops, respectively.

The remainder of the paper is structured as follows. Section II provides an overview of some related work. Section III presents a detailed description of DWAP. Section IV discusses the performance evaluation and simulation results. Finally, Section V concludes our findings and discusses future work.

II. Related Work

Several methods have been proposed to mitigate the adverse impacts of intermittent availability and unreliability of workers in EED-enhanced EC. These methods can be broadly classified into two categories: *Reactive* and *Proactive*. Reactive approaches refer to measures taken after a failure occurs in order to minimize the negative impact on offloaded tasks. Proactive approaches, on the other hand, refer to measures taken to reduce the likelihood of failure before its occurrence.

A. Reactive Approaches

Common reactive methods for recovering from failures include task checkpointing [13], task replication [5], and task migration [14]. In [13], checkpointing is used to regularly save the task's state. In the event of failure, the task resumes from the last saved state. However, determining the optimal frequency of checkpointing is a challenging issue in such schemes. Creating checkpoints too frequently can lead to a waste of time and resources, whereas creating infrequent checkpoints can lead to significant data loss [13]. In [5], Amer et al. allocate multiple replicas of the task to multiple workers in order to increase the chance of continuing the task without interruption in the event of failure at one worker. However, task replication can lead to poor resource utilization and increased waste of resources [13]. In [14], Saleh et al. use service migration in peer-to-peer networks to transfer the remaining portion of the task from an unreliable worker to a different worker within the network. They propose two task migration approaches. The first approach migrates tasks to the first available worker, while the second approach waits for all sub-peers to finish their tasks before initiating migrations. The second approach is shown to be more reliable than the first approach. However, task migration can introduce additional overhead, latency, and energy consumption due to the transmission of the task from one worker to another.

In general, reactive approaches can often result in wasted resources, lost time, and decreased satisfaction for end-users due to increased delays [6]. Proactive approaches tend to render better performance results than reactive approaches [3].

B. Proactive Approaches

Several proactive techniques have been proposed to enable better decision-making concerning task allocation. In [15], McGough et al. utilize machine learning algorithms to forecast the idle periods of workers and allocate the tasks accordingly. They train the algorithms using several months' worth of traces and predict the idle times for the following month. In [11], Umer et al. develop a Markov chain-based model to predict worker states up to 15 days ahead. However, such schemes fail to account for the dynamicity of EED-enabled environments. This is since they rely on a static training process, which involves training the model on a fixed dataset and making predictions for days in advance. Additionally, the set of workers considered are fairly reliable, without much dynamic activity. As a result, it remains unclear how well these models would perform in environments with more frequent and significant changes in worker behavior. Some schemes explore statistics-based methods, such as modeling worker availability using probability distributions [10]. Incorporating reputation models to evaluate the reliability of workers is also used in other schemes [9], [16], whereby tasks are assigned to workers based on their reputation, determined by factors such as task failure rates and resource usage patterns. However, the approaches used in [9], [10], [16] are also static and fail to capture the factors that can influence worker availability in a volatile environment. Furthermore, in [17], Deng et al. propose a predictive task allocation mechanism by using the Exponential Moving Average (EMA) technique to forecast edge node capacities. However, the EMA may fail to capture sudden shifts due to its limitations.

Existing proactive approaches are not entirely applicable to dynamic environments due to their reliance on static models that analyze historical data and provide estimations over days or even months. In contrast, we propose a proactive approach that is specifically tailored to dynamic EED-enabled environments and enables efficient resource allocation in real time. We foster dynamic prediction of worker availability for optimized resource allocation. We also use a large Google-generated dataset of real-world applications and preprocess it to capture the volatile nature of EEDs [12].

III. Dynamic Worker Availability Prediction (DWAP)

In DWAP, the system consists of three major components; workers, requesters, and an orchestrator. User-owned devices, known as workers, contribute their resources to the system in exchange for some incentives provided by the orchestrator. Requesters send the tasks that need to be offloaded for execution to the orchestrator. The latter plays a central role in the system by allocating the incoming tasks to participating workers based on their capabilities and availability within certain constraints and limitations. The orchestrator predicts the availability of workers using the Continuous-time Markov Chain (CTMC) model. Based on the estimated availability, the orchestrator makes the resource allocation decision by applying a refined optimization-based approach that is adopted in various schemes [4], [6].

A. System Model

Consider a set of m workers $W = \{w_1, w_2, \ldots, w_m\}$ and a set of n tasks $T = \{t_1, t_2, \ldots, t_n\}$. Each task $t_i \in T$ is

Authorized licensed use limited to: Queen's University. Downloaded on March 05,2024 at 13:19:19 UTC from IEEE Xplore. Restrictions apply.

associated with a workload q_i (in CPU cycles). Each worker $w_j \in W$ has a maximum CPU cycle frequency c_j^{max} (in CPU cycles/sec). The maximum number of tasks that can be executed in parallel on each worker w_j, is denoted β. One of the key responsibilities of the orchestrator is to assign workers with tasks that they can complete successfully. Note that the successful completion of tasks is highly dependent on the availability of workers, which is predicted by the orchestrator to be subsequently used in the resource allocation process. In the latter, the orchestrator strives to minimize the execution time of tasks. For each task t_i allocated to worker w_j, the execution time depends on both the workload q_i and the CPU cycle frequency, denoted c_{ij} (in CPU cycles/sec), dedicated by w_j to execute t_i. The CPU cycle frequency c_{ij} is given by Eq. 1.

$$c_{ij} = \frac{c_j^{max}}{\beta} \tag{1}$$

The execution time of executing task t_i on worker w_j is thus denoted by γ_{ij} and is given by Eq. 2.

$$\gamma_{ij} = \frac{q_i}{c_{ij} * S_j} \tag{2}$$

Each worker w_j is estimated to be available for a certain duration α_j. Consequently, the orchestrator only assigns tasks to workers whose estimated availability can accommodate the task duration. To estimate the availability of any given worker, historical data on the worker's availability behavior is used and fitted into a Continuous-time Markov Chain (CTMC) model. More details on how this estimation is performed is discussed in the following section.

B. Prediction Methodology

This section provides information about the dataset utilized, as well as the steps taken in preprocessing it. In addition, we present the prediction process by providing a detailed description of the CTMC model used.

1) Dataset and Preprocessing

We utilize a large usage trace of real-world applications made available by Google [12]. The dataset encompasses 8 Google computing clusters, also referred to as cells, collected in May 2019. Each cell comprises roughly 12k machines (i.e., workers) and provides detailed information on various aspects, including the characteristics of individual machines, events occurring on those machines, the tasks they perform, and the utilization of resources. Each task can be in one of six possible states at any given time, namely Submitted, Queued, Scheduled, Evicted, Failed, or Finished. Machines can also be in one of three possible states indicating their operational status, namely Add (the machine is added to the cluster), Remove (the machine is removed from the cluster), or Update (the machine's available resources is updated). Additionally, the machines in the cells exhibit a significant level of heterogeneity. Notably, among all the cells, cell g is the most heterogeneous [18], comprising 13 different types of machines with varying CPU and memory capabilities. Thus, we use data obtained solely from cell g.

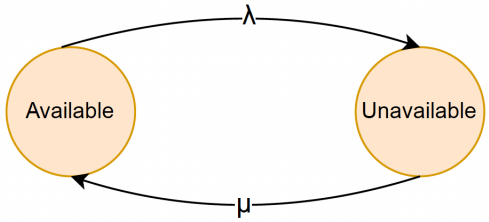


Fig. 1: State diagram of worker transitions

Although the machines exhibit heterogeneity in terms of their CPU and memory capabilities, it is worth noting that they are dedicated machines and not user-owned machines, which makes them less volatile and more reliable. Thus, for our purposes, we sample the 100 most volatile and unreliable machines, while ensuring representation of all 13 CPU and memory capability combinations and proportionality to the initial population. After extracting the availability traces of the sampled machines, we preprocess the events by categorizing the Add and Update events as "Available" and the Remove events as "Unavailable", since our system is only concerned with the two states of availability and unavailability. Moreover, we preprocess the time stamps into 15-minute intervals.

2) CTMC Model

Markov processes, specifically CTMC, enable modeling the system as a set of states and transitions between states that occur probabilistically over continuous time intervals, making it possible to analyze the probabilities of different system states at various points in time. The transitions between states are specified through rates at which workers transfer from one state to another, either due to failure (i.e. leaving the system) or due to repair (i.e. rejoining the system). The state diagram of worker transitions is illustrated in Fig. 1.

Let λ denote the worker failure rate. We express the failure rate in terms of the Mean Time Between Failure (MTBF), as shown in Eq 3. MTBF is a metric that measures the expected time between component failures. In our setting, it refers to the expected duration between workers leaving.

$$\lambda = \frac{1}{MTBF} \tag{3}$$

Similarly, let μ denote the worker repair rate. We express the repair rate in terms of the Mean Time to Repair (MTTR), as shown in Eq 4. MTTR is a metric that measures the expected time to repair a component after it fails. In our setting, it reflects the expected time for a worker to rejoin the system.

$$\mu = \frac{1}{MTTR} \tag{4}$$

The state-transition matrix for the described Markov chain is given by Q, as follows:

Authorized licensed use limited to: Queen's University. Downloaded on March 05,2024 at 13:19:19 UTC from IEEE Xplore. Restrictions apply.

$$Q = \begin{bmatrix} -\lambda & \lambda \\ \mu & -\mu \end{bmatrix}, \quad \text{where } \lambda, \mu > 0$$

Let P(t) denote the transition probability matrix, where $P_{i,j}(t)$ is the transition probability from state *i* to state *j* within time *t*. The state transition equations are calculated by Eq. 5.

$$\frac{d}{dt}P(t) = P(t).Q \tag{5}$$

By considering "Available" as state 1 and "Unavailable" as state 2, and assuming that $P_{1,1}(0) = 1$, solving the differential equations yields the following:

$$P_{1,1}(t) = \frac{\lambda}{\lambda+\mu}e^{-(\lambda+\mu)t} + \frac{\mu}{\lambda+\mu} \tag{6}$$

$$P_{1,2}(t) = -\frac{\lambda}{\lambda+\mu}e^{-(\lambda+\mu)t} + \frac{\lambda}{\lambda+\mu} \tag{7}$$

$$P_{2,2}(t) = \frac{\mu}{\lambda+\mu}e^{-(\lambda+\mu)t} + \frac{\lambda}{\lambda+\mu} \tag{8}$$

$$P_{2,1}(t) = -\frac{\mu}{\lambda+\mu}e^{-(\lambda+\mu)t} + \frac{\mu}{\lambda+\mu} \tag{9}$$

Using Eq. 6, 7, 8, and 9, a worker's state is predicted for a specified number of time steps ahead. A worker's availability duration, α_j, is calculated by aggregating the duration of consecutive time intervals when it is predicted to be available within these time steps.

C. Problem Formulation

Upon estimating the availability duration of workers, the orchestrator uses such estimations to make the resource allocation decision. During resource allocation, the objective is to minimize the total execution time of all tasks. We formulate the resource allocation problem as an Integer Linear Program (ILP) problem, where the binary decision variable x_{ij} is set to 1 if task t_i is allocated to worker w_j , and 0 otherwise, as given by Eq. 10.

$$x_{ij} = \begin{cases} 1 & \text{if task } t_i \text{ is allocated to worker } w_j \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

The problem formulation is given by Eq. 11.

$$\min_{x_{ij}} \sum_{j \in W} \sum_{i \in T} x_{ij}\gamma_{ij} \tag{11}$$

subject to:

C1: $\sum_{j \in W} x_{ij} = 1 \qquad \forall\ i \in T$

C2: $\sum_{i \in T} x_{ij}c_{ij} \leq c_j^{max} \qquad \forall\ j \in W$

C3: $x_{ij}\gamma_{ij} \leq \alpha_j \qquad \forall\ i \in T, \forall\ j \in W$

The objective given by Eq. 11 is subject to the constraints C1-C3. Constraint C1 ensures that each task is assigned to one worker. This guarantees that each task is assigned only once. Constraint C2 specifies that the CPU cycle frequency used to execute all the tasks assigned to a worker should not exceed its maximum capacity. Constraint C3 indicates that the execution time of each task assigned to each worker does not exceed the estimated availability duration α_j of the worker. This is critical to ensure that a worker is not assigned a task that takes longer to complete than the time it is available to work on.

IV. Performance Evaluation

In this section, we evaluate the performance of DWAP compared to a representative of state-of-the-art availability-naive resource allocation schemes [4], [6], referred to as the Availability-unaware Resource Allocation (AURA) scheme. Note that AURA adopts the same optimization problem used in DWAP but without Constraint C3. In order to assess the impact of the prediction model used in DWAP, we compare it to the prominent Exponential Moving Average (EMA) prediction model [17]. Note that EMA calculates the weighted average of past availability observations, with more weight given to recent observations. We apply EMA to estimate the availability duration of workers and then apply the same availability-aware resource allocation approach used in DWAP. We refer to such a combination as the Exponential Moving Average Resource Allocation (EMARA) scheme.

We use the following performance metrics: 1) the average execution time, which is the average time taken to execute tasks successfully, 2) the task drop rate, which is the ratio of the total number of tasks that get dropped (i.e., fail to be executed) to the total number of allocated tasks, 3) the Mean Absolute Error (MAE), which measures the average absolute difference between predicted and actual values, and 4) the Root Mean Square Error (RMSE), which measures the square root of the average of the squared differences between predicted and actual values.

A. Simulation Setup

For the prediction system, we first split the data into training and testing sets, with the training set initially comprising the first 80% of the data and the testing set comprising the remaining 20%. We use this initial split to compute the failure and repair rates of each worker up until the current time step.

To train the CTMC-based model and make predictions on worker availability, we use the Expanding Window Cross-Validation technique [19]. This involves gradually increasing the size of the training data by incorporating more historical data through an expanding window and making predictions on a fixed window of test data. We set the window size to 30 minutes, meaning that predictions are made for 30 minutes into the future. Afterward, the model parameters (failure and repair rates) are updated using the latest training set, which now includes the data from the most recent 30-minute window. This process of retraining the model and making predictions is repeated for each subsequent 30-minute window of test data.

DWAP, EMARA, and AURA are all implemented using using Python, which is integrated with Gurobi [20] to generate

Authorized licensed use limited to: Queen's University. Downloaded on March 05,2024 at 13:19:19 UTC from IEEE Xplore. Restrictions apply.

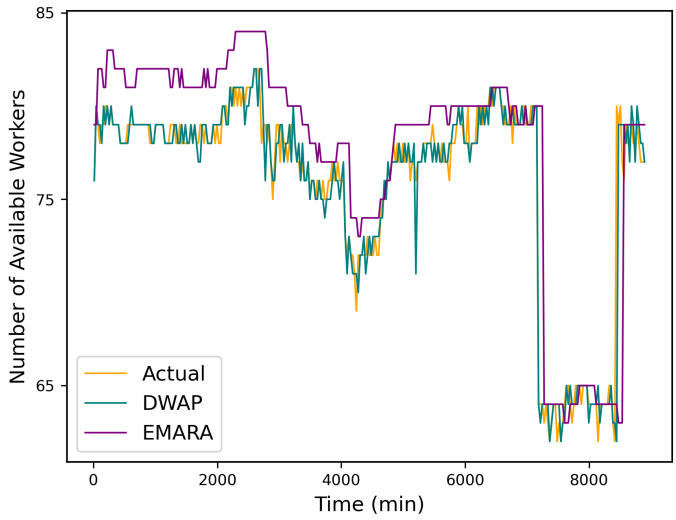


Fig. 2: Actual and predicted number of available workers in DWAP and EMARA at each time interval

the optimal solution. The total number of tasks at every time step is set to 200. The computation intensity of tasks is uniformly distributed in the range of [50,000, 70,000]. Unless otherwise specified, the maximum number of tasks that workers can execute in parallel is set to 4. The maximum CPU cycle frequency for each worker is extracted from the dataset. Note that in the dataset, the values provided are normalized relative to the most powerful worker. Thus, we set the maximum possible value to 600 CPU cycles/sec and scale the values accordingly. Finally, the smoothing factor in EMARA, which determines the weight given to past observations relative to new observations, is set to 0.1.

B. Results and Analysis

In our experiments, we evaluate the performance of DWAP, EMARA, and AURA over varying β (i.e., maximum number of parallel tasks) and average task intensity q. In addition, we evaluate the performance of the prediction models used in DWAP and EMARA over varying time intervals. All experiments are repeated 150 times and the final results from each scheme are averaged across all time steps and workers.

1) Prediction Results

Fig. 2 depicts the actual and the predicted number of available workers in DWAP and EMARA at each time interval. It can be observed that the predicted values in DWAP closely follow the actual data, which shows the ability of DWAP to effectively capture the overall trend of worker availability. In contrast, predictions made using EMARA substantially deviate from the actual values, indicating that it is significantly less accurate than DWAP in predicting worker availability.

Notably, DWAP is also able to capture short-term fluctuations in the data, which EMARA is unable to do. This demonstrates that DWAP is not only more accurate, but also more adaptable to changes. In order to provide a more comprehensive evaluation of the models' performance, we assess their MAE and RMSE. As depicted in Table I, DWAP outperforms EMARA in terms of both metrics, achieving an improvement of more than 74% in MAE and 59% in RMSE. This can be attributed to DWAP's utilization of state transitions, consideration of time intervals between events, and dynamic parameter adjustments, which effectively capture data dynamics and temporal dependencies. EMARA, on the other hand, employs a weighted average of previous data points, which may fall short in capturing complex patterns or sudden shifts.

TABLE I: MAE and RMSE of DWAP and EMARA

Scheme	MAE	RMSE
DWAP	0.58	1.36
EMARA	2.31	3.39

2) Impact of Maximum Number of Parallel Tasks (β)

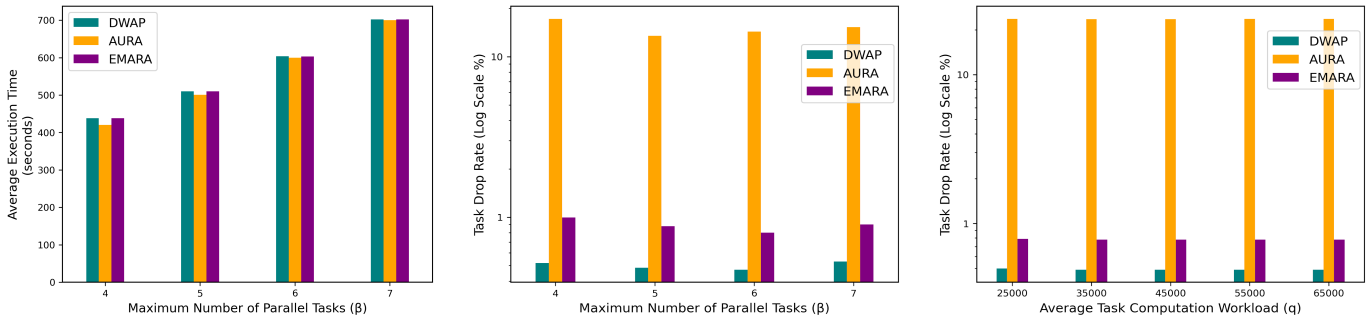
Fig. 3a depicts the performance of DWAP, AURA, and EMARA in terms of the average execution time over varying β. The results show that as the value of β increases, the average execution time increases for all of the schemes. This is because the dedicated computational capability c_{ij} is reduced for each individual task, resulting in a more time-consuming execution. Furthermore, the three schemes yield fairly comparable results. AURA marginally outperforms DWAP and EMARA by up to 4%. This can be attributed to the fact that AURA focuses solely on minimizing the execution time without considering worker availability. In contrast, DWAP and EMARA take both factors into consideration, which results in a more constrained selection of available resources.

Fig. 3b shows the task drop rate of DWAP, AURA, and EMARA over varying β. It can be observed that DWAP significantly outperforms AURA, with an improvement of up to 97%. This is because in contrast to AURA, DWAP accounts for worker availability and only assigns tasks to workers that are predicted to be available for the entire duration of task execution, thus reducing the risk of tasks being dropped due to intermittent availability of workers. In addition, DWAP significantly outperforms EMARA, yielding an improvement of up to 48%. This is since DWAP provides more accurate estimations of workers' availability than EMARA, thus reducing the risk of assigning tasks to workers that are estimated to be available longer than they actually are.

3) Impact of Average Task Intensity (q)

We investigate how DWAP, AURA, and EMARA respond to increasing workload intensity of tasks and whether this affects their task allocation decisions. Fig. 3c depicts the drop rate of DWAP, AURA, and EMARA over varying workload intensity (q). Our findings indicate that increasing the workload intensity of tasks does not significantly impact the task allocation decision process. Instead, it only makes the tasks more computationally intensive, which affects the execution time, but not the task allocation decision. Thus, as q increases, the task drop rate of DWAP, AURA, and EMARA remains the same.

Authorized licensed use limited to: Queen's University. Downloaded on March 05,2024 at 13:19:19 UTC from IEEE Xplore. Restrictions apply.



(a) Average execution time over varying β (b) Task drop rate over varying β (c) Task drop rate over varying q

Fig. 3: Performance results of DWAP, AURA and EMARA

V. Conclusions and Future Work

The ownership and management of Extreme Edge Devices (EEDs) by users introduce inherent uncertainties that demand novel resource allocation mechanisms capable of adapting to dynamically changing worker availability. In this paper, we have proposed the Dynamic Worker Availability Prediction (DWAP) scheme. DWAP incorporates the use of historical data into the Continuous-Time Markov Chain (CTMC) model to dynamically predict the availability of workers in the subsequent time step. DWAP continually adjusts the model parameters to incorporate new incoming data, enabling it to make informed and effective resource allocation decisions for the volatile environment of EED-enabled EC. Extensive evaluations on real-world data have shown that DWAP yields significant improvements of 74% and 59% in terms of MAE and RMSE, respectively, compared to a prominent prediction scheme. Furthermore, evaluations have shown that DWAP significantly outperforms prominent availability-unaware and availability-based resource allocation schemes, by up to 97% and 48%, respectively, in terms of task drop rate. In the future, we plan on further improving the prediction accuracy of DWAP by applying uncertainty quantification techniques.

Acknowledgement

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

References

[1] M. Hung, "Leading the iot, gartner insights on how to lead in a connected world," *Gartner Research*, vol. 1, pp. 1–5, 2017.

[2] D. R.-J. G.-J. Rydning, J. Reinsel, and J. Gantz, "The digitization of the world from edge to core," *Framingham: International Data Corporation*, vol. 16, 2018.

[3] R. Kain, S. A. Elsayed, Y. Chen, and H. S. Hassanein, "Multi-step prediction of worker resource usage at the extreme edge," in *Proceedings of the 25th International ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems*, 2022, pp. 25–32.

[4] M. De'bas, S. A. Elsayed, and H. S. Hassanein, "Multitiered worker-oriented resource allocation at the extreme edge," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*, 2022, pp. 5674–5679.

[5] I. M. Amer, S. M. Oteafy, S. A. Elsayed, and H. S. Hassanein, "Qos-based task replication for alleviating uncertainty in edge computing," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*, 2022, pp. 5147–5152.

[6] R. F. El Khatib, S. A. Elsayed, N. Zorba, and H. S. Hassanein, "Optimal proactive resource allocation at the extreme edge," in *ICC 2022-IEEE International Conference on Communications*, 2022, pp. 5657–5662.

[7] G. A. McGilvary, A. Barker, and M. Atkinson, "Ad hoc cloud computing," in *2015 IEEE 8th international conference on cloud computing*, 2015, pp. 1063–1068.

[8] A. Celestini, A. Lluch Lafuente, P. Mayer, S. Sebastio, and F. Tiezzi, "Reputation-based cooperation in the clouds," in *Trust Management VIII: 8th IFIP WG 11.11 International Conference, IFIPTM 2014, Singapore, July 7-10, 2014. Proceedings 8*, 2014, pp. 213–220.

[9] Y. Alsenani, G. Crosby, and T. Velasco, "Sara: A stochastic model to estimate reliability of edge resources in volunteer cloud," in *2018 IEEE international conference on EDGE computing (EDGE)*, 2018, pp. 121–124.

[10] B. Javadi, D. Kondo, J.-M. Vincent, and D. P. Anderson, "Discovering statistical models of availability in large distributed systems: An empirical study of seti@ home," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 11, pp. 1896–1903, 2011.

[11] A. Umer, A. N. Mian, and O. Rana, "Predicting machine behavior from google cluster workload traces," *Concurrency and Computation: Practice and Experience*, vol. 35, no. 5, p. e7559, 2023.

[12] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes, "Borg: the next generation," in *Proceedings of the fifteenth European conference on computer systems*, 2020, pp. 1–14.

[13] S. Jayasekara, S. Karunasekera, and A. Harwood, "Optimizing checkpoint-based fault-tolerance in distributed stream processing systems: Theory to practice," *Software: Practice and Experience*, vol. 52, no. 1, pp. 296–315, 2022.

[14] E. Saleh and C. Shastry, "Task migration in volunteer computing systems," in *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 2022, pp. 2076–2079.

[15] A. S. McGough, M. Forshaw, J. Brennan, N. Al Moubayed, and S. Bonner, "Using machine learning to reduce the energy wasted in volunteer computing environments," in *2018 Ninth International Green and Sustainable Computing Conference (IGSC)*, 2018, pp. 1–8.

[16] Y. Alsenani, G. V. Crosby, T. Velasco, and A. Alahmadi, "Remot reputation and resource-based model to estimate the reliability of the host machines in volunteer cloud environment," in *2018 IEEE 6th international conference on future internet of things and cloud (FiCloud)*, 2018, pp. 63–70.

[17] X. Deng, J. Li, E. Liu, and H. Zhang, "Task allocation algorithm and optimization model on edge collaboration," *Journal of Systems Architecture*, vol. 110, p. 101778, 2020.

[18] Y. Lin, A. Barker, and S. Ceesay, "Exploring characteristics of inter-cluster machines and cloud applications on google clusters," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 2785–2794.

[19] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[20] Gurobi, "Gurobi optimizer reference manual," https://www.gurobi.com/, 2022, accessed on March, 2023.

Authorized licensed use limited to: Queen's University. Downloaded on March 05,2024 at 13:19:19 UTC from IEEE Xplore. Restrictions apply.