DRUDGE: Dynamic Resource Usage Data Generation for Extreme Edge Devices

Ruslan Kain, Sara A. Elsayed, Yuanzhu Chen, and Hossam S. Hassanein
School of Computing, Queen's University, Kingston, ON, Canada
{r.kain, yuanzhu.chen}@queensu.ca
{selsayed, hossam}@cs.queensu.ca

GLOBECOM 2023 - 2023 IEEE Global Communications Conference | 979-8-3503-1090-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/GLOBECOM54140.2023.10437760

***Abstract*—Extreme Edge Computing (EEC) can drastically curtail the delay, reduce network bandwidth consumption, and enhance system performance by providing computing resources closer to the data-generating Internet of Things (IoT) devices. However, the use of Extreme Edge Devices (EEDs) in EEC presents unique challenges imposed by the inherent dynamic user-access behavior, which introduces highly dynamic resource usage. To tackle such challenges, it is crucial to enable accurate resource usage predictions, which in turn requires having reliable datasets. In this paper, we cultivate the Dynamic Resource Usage Data Generation for EEDs (DRUDGE) methodology. DRUDGE generates datasets that capture the resource usage dynamics of EEDs running diverse user-end applications in fine-grained intervals over extended periods. We present an in-depth characterization of resource utilization in EEDs and make the datasets publicly available to the research community. We examine the temporal variation of critical system metrics, such as CPU usage, memory usage, temperature, and network traffic. Furthermore, we apply various statistical tests to gain valuable insights into the data characteristics, including skewness, kurtosis, stationarity, volatility, cointegration, multi-collinearity, Granger causality, and Pearson correlation analysis. These insights inform model selection, feature engineering, and preprocessing techniques, leading to more accurate and reliable forecasts and analyses for EEC systems.**

***Index Terms*—Edge Computing, Internet of Things, Extreme Edge Devices, Dynamic Resource Usage, Data Generation**

I. Introduction

The rapid growth of Internet of Things (IoT) devices and their increasing integration into various aspects of our daily lives have resulted in an enormous amount of data being generated at the network's edge [1]. To effectively manage and process this data, there is a need to develop advanced computing solutions that can handle the unique challenges associated with IoT applications [2]. Centralized cloud computing, while offering significant advantages in terms of scalability and cost-effectiveness, are subject to limitations such as high latency, bandwidth constraints, and lack of location awareness. These limitations can be detrimental to the performance of latency-sensitive, data-intense, and location-dependent IoT applications, such as autonomous vehicles, remote surgery, and industrial automation [2].

To address these challenges, the computing landscape is evolving from centralized cloud computing toward a more decentralized model, known as edge computing [3]. Edge computing brings computation and data storage closer to the data source, i.e., the IoT devices, thereby reducing latency and improving data processing efficiency. Extreme Edge Computing (EEC) [4]–[7] is an extension of edge computing that pushes the boundaries of distributed computing even further, by enabling real-time processing and analysis of data directly on end devices, also referred to as Extreme Edge Devices (EEDs). This computing paradigm offers the potential for significant performance improvements in terms of low latency and location awareness [8]. In addition, it paves the way for a new tech market where industries, businesses, and individuals can capitalize their underused computational resources and maintain their own edge cloud [9].

Despite its promising prospects, the positive impact of EEC can be obstructed by the underlying dynamic user access behavior of EEDs. This dynamicity is triggered by the fact that EEDs are user-owned devices, which means that users can grab their devices at any time during the execution of offloaded tasks and run any intensive application, leading to a highly dynamic resource usage of EEDs and profoundly altering their available computational resources [10]. Thus, it is imperative to develop accurate resource usage prediction schemes to help make more informed resource allocation decisions. The accuracy of such predictions relies heavily on having reliable datasets. In [11], Deep Learning models are used to predict the resource usage of EEDs. However, the existing works fail to foster the provision of datasets that captures the resource usage dynamics of EEDs.

In this paper, we introduce the Dynamic Resource Usage Data Generation for EEDs (DRUDGE) methodology. DRUDGE presents a method for generating a dynamic resource usage dataset for EEDs. Note that in previous works [5] [7], we have evaluated the performance of resource usage prediction models on the generated datasets and the results validate the data's reliability. Our contributions can be summarized as follows:

- We introduce DRUDGE, a novel methodology that enables the generation of a comprehensive dataset that captures the dynamic resource usage of various user-end applications in extreme edge scenarios. To the best of our knowledge, DRUDGE is the first work in the literature that offers such a methodology. The code for DRUDGE is made available on GitHub[1].

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

- We present a detailed and annotated dataset that includes labels for running applications, such as video streaming, gaming, cryptocurrency mining, and augmented reality. Such various usage scenarios encompass the dataset allowing for a granular analysis of how each application impacts the system's resources to aid the development of efficient resource management strategies for IoT applications. The dataset also contains information about various system metrics, such as CPU usage, memory usage, temperature, network traffic, and Wi-Fi link quality, among others. The dataset is made up of more than 550,000 unique data points representing 768 hours of running applications on EEDs. It is made publicly available to the research community on Borealis [12] to provide a valuable resource for studying resource usage dynamics in EEDs and facilitate further research in this area.
- We conduct extensive data analysis using various statistical tests, including skewness, kurtosis, stationarity, volatility, cointegration, multicollinearity, Granger causality, and Pearson correlation analysis. These tests allow researchers to better understand the characteristics of the data, enabling informed decisions on model selection, feature engineering, and preprocessing techniques, leading to more accurate and reliable forecasts and analyses, and the development of more efficient and effective resource management strategies for IoT applications.

The remainder of the paper is organized as follows: Section II presents the related work. Section III presents the dataset generation method DRUDGE. Section IV describes the resource usage data. Section V presents the analysis conducted on the data. Finally, Section VI presents the conclusion and future research directions.

II. Related Work

Resource usage datasets for multi-tenant systems, such as CloudSuite [13], Google Cluster Data [14], and GWA-T-12 dataset [15], provide valuable insights into the behavior and performance of applications in cloud environments. These datasets include information on CPU, memory, and network utilization, enabling researchers and practitioners to study and optimize system performance. While CloudSuite [13] focuses on benchmarking cloud services and infrastructure, Google Cluster Data [14] provides large-scale datasets on resource usage patterns within data centers, and GWA-T-12 dataset [15] is comprised of resource usage traces for distributed computing systems. Additionally, a dataset generated by Kim et al. [16] provides experimental results on geographically distributed blockchain-based services in a cloud computing environment. The dataset contains timestamp logs, block and transaction information from the blockchain network, and resource usage data from AWS EC2 instances.

As opposed to existing datasets that fail to consider the highly dynamic nature of EEDs in EEC environments, we provide a dataset that is tailored for EEC applications, capturing resource usage dynamics of various applications under extreme edge scenarios. This dataset is unique in providing insights into the behavior and performance of EEDs in real-world scenarios and can be used to optimize system operation performance in EED-based environments. Although the work in [11] makes use of datasets in a similar environment, the authors have not made the data publicly available nor provided a reproducible data generation process. In contrast, we introduce a novel data generation method and make both the dataset and the code publicly available on the research data repository Borealis [12] and GitHub[1].

TABLE I: Workers' Specifications and Labels

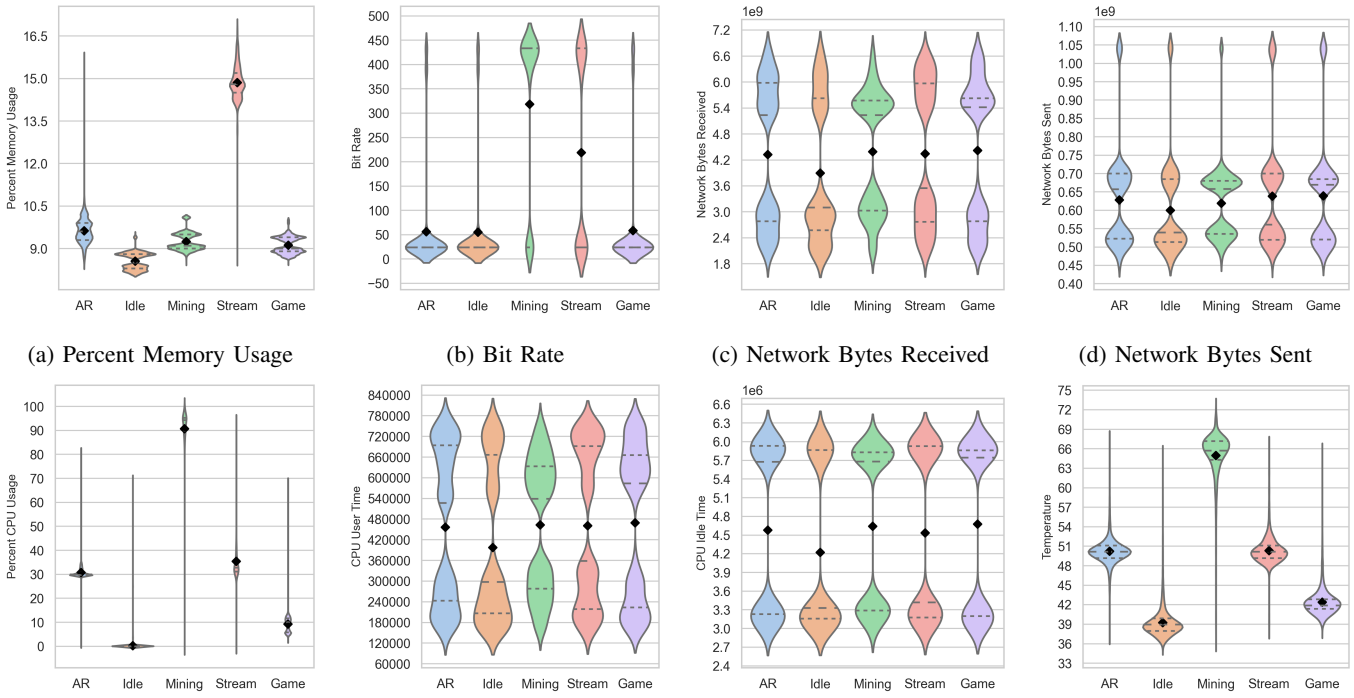
Worker	Raspberry Pi 4B Specifications	
	RAM (GB)	CPU Freq. (GHz)
A	8	1.8
B	4	1.5
C	2	1.5
D	2	1.2

III. Dynamic Resource Usage Data Generation for EEDs (DRUDGE)

DRUDGE is developed using Python; it can mimic the dynamic usage behavior of EEDs, also called workers, in a controlled and repeatable manner. This data generation methodology incorporates a diverse range of applications and it offers the flexibility to adjust the running duration of each application, CPU frequency, and network condition, providing a rich dataset that encompasses various usage scenarios.

The data generation process utilizes a selection of Raspberry Pi (RPi) 4B model devices with varying RAM sizes and CPU cycle frequencies, detailed in Table I. To further enhance the heterogeneity of the workers, the standard 1.5 GHz CPU frequency of the RPi 4B model is manually configured through overclocking and throttling. These devices are exposed to a sequence of applications running automatically designed to emulate dynamic resource usage scenarios, such as playing a video game (Doom), video streaming (YouTube), augmented reality emulation, and Duino-coin mining. The Doom game is run by simulating player movements through predefined actions. For the video streaming, a browser is launched to stream a YouTube video by navigating to its URL and initiating playback. Navigation and interaction with the Doom game and browser are achieved using the `pynput` library. Augmented Reality emulation is achieved using the `imutils` and `cv2` libraries, where source and input images with ArUco markers are processed, detected, and merged based on a computed homography matrix. Lastly, Duino-coin mining involves solving cryptographic problems making use of the SHA-1 cryptographic function for encryption on a hash-chain, a variant of a block-chain [17], to earn rewards. In addition, idle periods where no application is running are also included.

The data is collected from the aforementioned resource usage scenarios over multiple 48-hour periods, with monitoring intervals set at 5 seconds. Note that DRUDGE prompts user inputs for various parameters, including the monitoring interval size, CPU frequency, dataset name, and application



(a) Percent Memory Usage (b) Bit Rate (c) Network Bytes Received (d) Network Bytes Sent

(e) Percent CPU Usage (f) CPU User Time (g) CPU Idle Time (h) Temperature

Fig. 1: Distribution of Resource Usage for different states in random sequence for Worker A. The bottom and top dashed lines in the violin plots indicate the 25th and 75th quantiles, respectively, the middle dashed line represents the median, and the black dot indicates the mean.

sequence type (random or pattern). DRUDGE generates a list of state time lengths depending on the chosen dataset type. For the patterned sequence, the lengths are determined based on the total run time, multiple application run-time lengths, and the maximum application run-time length.

DRUDGE executes a series of applications with the running duration of each application determined by the previously generated state time lengths. The resource usage is monitored and recorded continuously in a separate thread at the specified interval using the `psutil` library [18], ensuring that data collection does not interfere with other processes. The recorded data is then saved in CSV format. If needed, DRUDGE can also switch between different network connections, mimicking the varying network conditions in real-world IoT scenarios.

IV. Data Description

The generated dataset in DRUDGE provides a comprehensive picture of system resource utilization over an extended period, enabling researchers to analyze temporal trends and variations in resource usage. The high granularity of the dataset makes it suitable for studying fine-grained system behavior and building system performance models. It consists of dynamic resource usage information related to various applications running on the set of workers detailed in Table I.

Each row in the dataset corresponds to a specific timestamp and captures the state of the system, including the CPU frequency (Hz), CPU utilization (%), CPU user, system, and idle time (seconds), memory utilization (%), network data sent and received (bytes), network upload and download rates (bytes/second), temperature (degree Celsius), WiFi frequency (MHz), bit rate (Mbps), link quality and maximum link quality (unitless), signal level (dBm), and the resource usage state associated with the running application (categorical). The resource usage states are represented by the labels *Game*, *Stream*, *Mining*, *Augmented_Reality*, and *Idle*. We provide the details of each of these measurements in the Appendix.

In addition to the rich set of measurements, system metrics and resource usage states, the dataset also contains pre-processed versions of the data divided into training (70%) and testing (30%). The dataset is made up of more than 550,000 unique data points representing 768 hours of running applications on EEDs. The size of the dataset is approximately 444 MB, and it is divided into 74 CSV files, which are publicly available on Borealis [12].

V. Data Analysis

In this section, we present a thorough analysis of worker A's resource usage data for a random application sequence. Our analysis includes Resource Usage by State, Cointegration (Johansen's), Multicollinearity (VIF), Skewness, Kurtosis, Stationarity, Volatility (EWMA), Causality (Granger), and Correlation (Pearson) [19]. Due to space constraints, we present only worker A's results from our comprehensive investigation of all workers' resource usage during random and patterned

sequences, and omit additional analyses such as periodogram (power spectral density vs. frequency), distribution, autocorrelation, seasonal decomposition, autocorrelation/partial autocorrelation functions, and Hurst parameter estimation; however, they can be found in our GitHub repository[1].

A. Resource Usage by State

This section analyzes the impact of the different applications (states) on various resource usage values. Fig. 1 presents the distribution of resource usage values for each state using violin plots. We examine memory usage, bit rate, network bytes sent/received, CPU usage, CPU user/idle time, and temperature for each state. This information can be useful for optimizing system performance and balancing the resources allocated to various applications in different states. It can be observed that the impact of the different states on resource usage, in ascending order, is as follows:

- The Augmented Reality (AR) state is exeeded only by the Stream state in mean memory usage (9.6). The mean bit rate (55.97) is similar to that of the Idle state. Its mean network bytes sent (6.28×10^8) ranks below that of the Stream and Game states, and its mean network bytes received (4.32×10^9) is similar to the Game state. The AR state's mean CPU usage (30.74) and mean temperature (50.22) fall below that of the Mining and Stream states. The mean CPU user time (456259) exceeds only that of the Idle state, and its CPU idle time (4.58×10^6) also exceeds that of the Stream state.
- The Idle state naturally shows the lowest mean values: memory usage (8.6), bit rate (55), network bytes sent (5.99×10^8) and received (3.89×10^9), CPU usage (0.24), CPU user time (397308) and idle time (4.22×10^6), and temperature (39.23).
- The Mining state leads in mean bit rate (318.34), network bytes received (4.39×10^9), CPU usage (90.68), and temperature (64.96), compared to all other states. Its mean memory usage (9.24) is less than that of the Stream and AR states. The mean CPU user time (463005) and idle time (4.64×10^6) exceeds all states except the Game state. Lastly, the Mining state only exceeds the Idle state in mean network bytes sent (6.19×10^8).
- The Stream state leads in mean memory usage (14.86). Its mean bit rate (218.75), network bytes received (4.34×10^9), CPU usage (35.41), and temperature (50.32) are only lower than that of the Mining state. The mean network bytes sent (6.38×10^8) is only lower than the Game state, while its CPU user time (460640) falls between those of the Game and Mining states. The CPU idle time (4.53×10^6) only exceeds that of the Idle state.
- The Game state has the highest mean network bytes sent (6.39×10^8), and CPU user time (469068) and idle time (4.68×10^6). Its mean bit rate (58.26) is less than that of the Mining and Stream states. Its mean memory usage (9.12), network bytes received (4.32×10^9), CPU usage (9.25), and temperature (42.39) exceed only that of the Idle state.

In summary, resource usage is highest in the Mining and Stream states, moderately high in the AR and Game states, with the Idle state showing the least resource consumption.

B. Cointegration and Multicollinearity

Table II shows the results of the Johanson's Cointegration Test and Variance Inflation Factor (VIF) for the resource usage data. Johanson's Cointegration is useful for forecasting by identify if two or more time series share a common long-term trend. Most features exhibit a significant relationship at a 95% confidence level - indicating a long-term equilibrium relationship - except for network download rate, temperature, and bit rate, suggesting they might not have a long-term relationship with other features.

The Variance Inflation Factor (VIF) assesses multicollinearity among predictors in a regression model. Multicollinearity occurs when predictor variables in a statistical model are highly correlated, leading to unreliable estimates and reduced model interpretability. A VIF value greater than 10 indicates significant multicollinearity. Table II reveals high multicollinearity for CPU user time, CPU system time, CPU idle time, network bytes received, and temperature suggesting that their combined effect on the dependent variable may be overestimated. Addressing this issue may involve removing or combining highly correlated features or using techniques like Principal Component Analysis (PCA) [20].

Resource parameters such as CPU frequency, percent CPU and memory usage, network bytes sent, network upload and download rates, and bit rate have VIF values below 10, indicating a lower degree of multicollinearity. Thus, these parameters are more reliable predictors in a regression model, since they provide independent information.

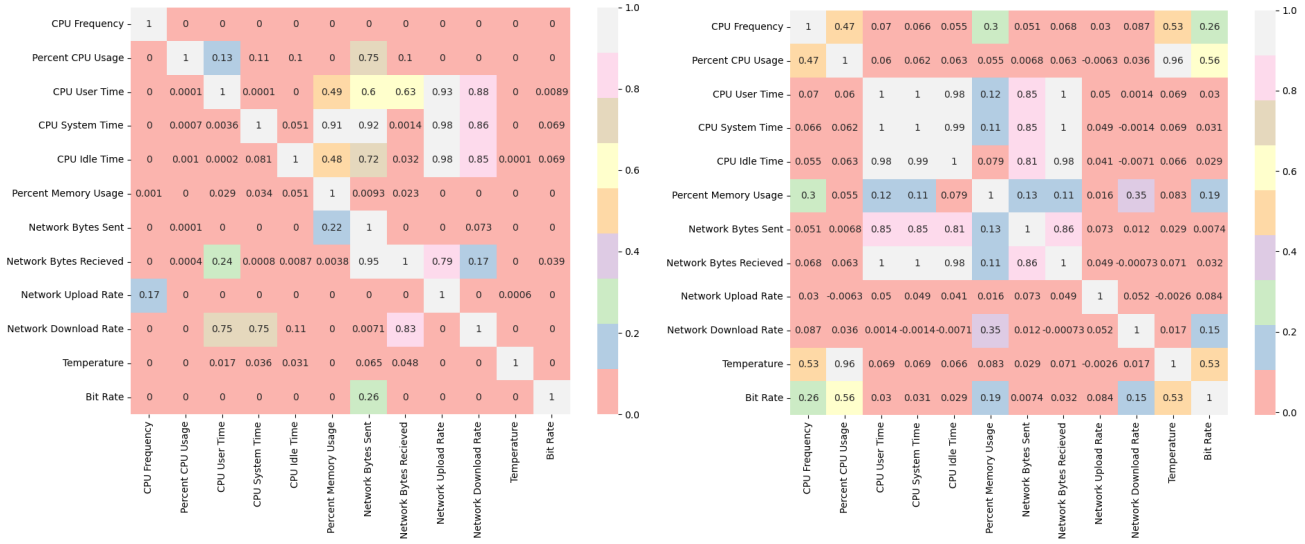
C. Skewness, Kurtosis, Stationarity, and Volatility

Skewness and kurtosis, the third and fourth moments, help understand data distributions. Skewness reflects data asymmetry, with positive skewness indicating a long right tail and negative skewness a long left tail. Kurtosis reveals the "tailedness" of the distribution, with high kurtosis signifying heavy tails and more outliers, while low kurtosis suggests lighter tails and fewer outliers. The insights into data distribution can guide model choice and preprocessing to improve performance. The network upload and download rates have high skewness, indicating that their distributions are not symmetrical, and high kurtosis, indicating heavy tails and more outliers.

Volatility analysis measures data variation over time. Data with high volatility requires models that effectively capture fluctuations, such as GARCH or ARIMA models. In terms of the Exponential Weighted Moving Average (EWMA) Volatility, CPU frequency and network bytes sent show higher mean values, indicating higher average metric levels. Network upload and download rates demonstrate close to zero averages. Standard deviation values offer insights into each feature's fluctuation. High values, such as those for CPU user time and network bytes received, imply substantial variability, making forecasting and risk management more challenging.

TABLE II: Combined Results: Cointegration, Multicollinearity (VIF), Skewness (3^{rd} Moment), Kurtosis (4^{th} Moment), EWMA Volatility (Mean & Std), ADF p-value, and Stationarity

Feature	Cointegration	VIF	Skewness	Kurtosis	EWMA Mean	EWMA Std	ADF p-value	Stationary
CPU Frequency	True	1.57	-1.43	0.15	232.28	258.52	6.83e-27	True
Percent CPU Usage	True	5.27	0.8	-0.78	4.14	6.06	1.05e-26	True
CPU User Time	True	470.01	0.04	-1.68	53.38	1006.87	0.9345	False
CPU System Time	True	4350.06	0.01	-1.76	5.92	159.28	0.8930	False
CPU Idle Time	True	178.35	0	-1.96	332.19	14193.39	0.8037	False
Percent Memory Usage	True	1.28	1.5	0.56	0.16	0.39	4.79e-28	True
Network Bytes Sent	True	5.1	1.59	2.88	51048.23	821249.30	0.9771	False
Network Bytes Received	True	13784.72	0.01	-1.70	454703.1	10346860	0.9026	False
Network Upload Rate	True	1.02	23.07	553.07	0	0	2.22e-30	True
Network Download Rate	False	1.17	7.98	75.43	0	0	2.12e-30	True
Temperature	False	173.97	0.6	-0.8	0.86	0.8	4.10e-26	True
Bit Rate	False	1.54	0.83	-1.3	122.33	73.42	2.54e-29	True



(a) Granger causality Matrix

(b) Pearson Correlation Coefficients Matrix

Fig. 2: Heatmap of a) Granger Causality Matrix and b) Pearson Correlation Coefficients Matrix for the resource usage states in random sequence for Worker A.

Conversely, low values, like those for temperature, percent memory usage, and network upload and download rates, suggest more predictable behavior.

The Augmented Dickey-Fuller (ADF) test checks whether a time series is stationary. A stationary time series has a constant mean, variance, and autocorrelation, which is important for many time series prediction models. The ADF test results shown in Table II reveal that several features are stationary, exhibiting p-values much lower than the 0.05 significance level. However, some features are non-stationary, such as CPU user, system, and idle times and network bytes sent and received, with p-values greater than 0.05. Features with high volatility and non-stationarity may require additional preprocessing. For instance, when addressing non-stationarity, the first-order difference can be used to stabilize the mean. When addressing high volatility, logarithmic transformation can be employed to make the data more amenable to time series modeling. While other features with lower volatility and stationarity can be directly used without adjustments.

D. Causality and Correlation

In the analysis of Worker A's resource usage states, we computed and visualized two distinct matrices as heatmaps in Fig.2. Fig.2a showcases the Granger causality matrix, signifying feature predictability, while Fig. 2b displays the Pearson correlation coefficients matrix, denoting linear relationships between feature pairs.

The Granger causality matrix (Fig. 2a) offers insights into predictive relationships between features. Notably, significant causality relationships exist, such as CPU user and system Time predicting network upload and download rates, percent memory usage, and network bytes sent. Conversely, weak Granger causality relationships are observed for some features, such as CPU frequency, memory percent usage, and temperature, meaning they are not predictive of the other features.

The Pearson correlation analysis (Fig. 2b) uncovers strong positive correlations between several feature pairs, including

Temperature and percent CPU usage, CPU user time and CPU idle time, and CPU system time and network bytes received. On the other hand, weak correlations are observed between other feature pairs, such as network upload and download rates, bit rate and CPU user time, and percent CPU and memory usage.

VI. Conclusion

In this paper, we have proposed the Dynamic Resource Usage Data Generation for Extreme Edge Devices (DRUDGE) methodology. DRUDGE provides an effective way to generate resource usage datasets to study the impact of the dynamic access behavior of EEDs on the available resources. We have made the generated dataset publicly available on Borealis for researchers and practitioners and conducted extensive analysis to gain insights into the resource usage patterns of EED applications, and to develop and validate methods for system optimization, energy efficiency improvement, and other research areas in Extreme Edge Computing (EEC). In the future, we plan on extending the dataset to include multiple applications running concurrently on diverse EEDs, such as the NVIDIA Jetson Nano. In addition, we will investigate techniques to reduce the impact of dynamic usage on the performance of the EEC system to better support IoT applications.

Acknowledgment

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] J. Steward, "21 Internet of Things Statistics, Facts & Trends for 2021," Oct 2021. [Online]. Available: https://findstack.com/internet-of-things-statistics/

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[3] M. Satyanarayanan, "The Emergence of Edge Computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[4] R. Kain and S. Sorour, "Worker Resource Characterization Under Dynamic Usage in Multi-access Edge Computing," in *International Wireless Communications and Mobile Computing (IWCMC)*, 2022, pp. 1070–1075.

[5] 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*, ser. MSWiM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 25–32.

[6] R. F. El Khatib, S. A. Elsayed, N. Zorba, and H. S. Hassanein, "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*, 2022.

[7] R. Kain, S. A. Elsayed, Y. Chen, and H. S. Hassanein, "RUMP: Resource Usage Multi-Step Prediction in Extreme Edge Computing," vol. 210, 2023, pp. 45–57.

[8] H. Gedawy, K. A. Harras, K. Habak, and M. Hamdi, "Femtoclouds Beyond the Edge: The Overlooked Data Centers," *IEEE Internet of Things Magazine*, vol. 3, no. 1, pp. 44–49, 2020.

[9] Distributive, 2023. [Online]. Available: https://distributive.network/

[10] S. Dey, A. Mukherjee, H. S. Paul, and A. Pal, "Challenges of Using Edge Devices in IoT Computation Grids," in *2013 International Conference on Parallel and Distributed Systems*, 2013, pp. 564–569.

[11] J. Violos, T. Pagoulatou, S. Tsanakas, K. Tserpes, and T. Varvarigou, "Predicting Resource Usage in Edge Computing Infrastructures with CNN and a Hybrid Bayesian Particle Swarm Hyper-parameter Optimization Model," in *Intelligent Computing*. Springer, 2021, pp. 562–580.

[12] R. Kain, S. A. Elsayed, Y. Chen, and H. S. Hassanein, "Resource Usage of Applications Running on Raspberry Pi Devices," August 2022. [Online]. Available: http://dx.doi.org/10.5683/SP3/GOZAJE

[13] "CloudSuite - Benchmarking the Cloud," http://cloudsuite.ch, accessed: 2023-05-04.

[14] "Google Cluster Data," https://github.com/google/cluster-data, accessed: 2023-05-04.

[15] "GWA-T-12," http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains, accessed: 2023-05-04.

[16] T. Kim and H.-J. Kim, "Experimental dataset for the Performance Evaluation for Geographically Distributed Blockchain-based Services in a Cloud Computing Environment," 2022. [Online]. Available: https://dx.doi.org/10.21227/b7mg-yb75

[17] R. Piotrowski, "Duino-Coin White Paper," Tech. Rep., 2021.

[18] G. Sforna, "Psutil: A Cross-platform Process and System Utilities Module for Python," GitHub repository, 2009. [Online]. Available: https://github.com/giampaolo/psutil

[19] J. D. Hamilton, *Time series analysis*. Princeton university press, 2020.

[20] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

Appendix

- time_stamp: precise time when the measurement is taken
- time: time as a floating-point number expressed in seconds since the epoch (UTC)
- cpu_freq: system-wide CPU cycle frequency (MHz)
- cpu: system-wide CPU utilization as a percentage (%)
- cpu_user_time: time spent by normal processes executing in user mode, including guest time (seconds)
- cpu_idle_time: time spent doing nothing (seconds)
- cpu_system_time: time spent by processes executing in kernel mode (seconds)
- memory: memory currently in use or very recently used, and so it is in RAM (%)
- net_sent: number of overall bytes sent
- net_recv: number of overall bytes received
- net_upload_rate: number of bytes sent in last time interval / time interval (MBytes/s)
- net_download_rate: number of bytes received in last time interval / time interval (MBytes/s)
- temp: current temperature of the CPU (Degree Celsius)
- wifi_freq: operating frequency of WiFi (GHz)
- bit_rate: speed at which bits are transmitted over the medium (Mbps)
- link_quality: overall quality of the link. May be based on the level of contention or interference, the bit or frame error rate, how good the received signal is, some timing synchronization, or other hardware metric.
- link_quality_max: maximum quality of the link (unitless)
- signal_level: strength of the Wi-Fi signal received by the network interface (dBm)
- (resource)_diff: resource (CPU times, memory usage, etc.) usage in the last interval (difference between current aggregate and previously measured aggregate values)
- net: label feature for type of network access technology used
- state: label feature for resource usage state of the device associated with running application