CPU Frequency Aware Power Modeling for IoT Edge Nodes

Vladimir Ostapenco*, Loïc Guégan†, Salma Tofaily†, Issam Raïs†, Laurent Lefèvre*
*EnsL, UCBL, CNRS, Inria, LIP, University of Lyon, Lyon, France
†Department of Computer Science, UiT The Arctic University of Norway, Tromsø, Norway
Corresponding authors: vladimir.ostapenco@ens-lyon.fr, {loic.guegan,salma.tofaily}@uit.no

***Abstract*—The Internet of Things (IoT) is used for various domains such as monitoring the environment, health care, and smart cities. Monitoring and measuring energy consumption of these systems is a crucial step in making them energy efficient.**

External Hardware-based power monitoring is not always available for IoT edge nodes. An alternative is to create an accurate power model that relates easy-to-monitor parameters (*e.g.,* instructions count, cache misses, node temperature, etc) to externally monitored power. This relationship helps to estimate the power drawn by the nodes.

IoT edge nodes have several power optimization leverages like Dynamic Voltage and Frequency Scaling (DVFS). When models calibration does not consider these leverages, the gap between power estimation and actual power usage increases.

In related works, several power models and corresponding Software-defined power meters do not consider CPU frequency on IoT edge nodes. These Software-defined power meters provide regression-based power models for IoT edge nodes. This work compares predictions made by these state of the art power models to accurate external power monitoring. We show that not considering CPU frequency can result in incorrect estimations.

We investigate and compare several methodologies for building power models, considering the CPU frequency, power, and energy leverage. Different performance metrics and regression methods are explored to estimate power usage. We demonstrate that linear and polynomial regression-based models are able to account for various CPU frequencies on IoT edge nodes. Using these models, we can predict the power consumed by IoT edge nodes running a specific workload, with a MAPE of 2% compared to accurate Hardware-based power meters.

***Index Terms*—sustainability, power monitoring, power models, energy efficiency, edge, DVFS**

I. Introduction

The Cloud has expanded beyond data centers. Fog and Edge paradigms leverage the capabilities of technological improvements such as smaller and cheaper node's hardware to answer timely challenges. IoT edge nodes are located at one end of these networks. All together, they form what is called "Internet of Things" [1], [2]. IoT edge nodes are small nodes that can range from simple micro-controllers to more complex single board computers. They can be (i) connected to sensors and actuators, (ii) connected to the cloud to send and receive data/tasks, (iii) close to the end-users, (iv) powered by wall sockets, or batteries.

IoT edge nodes can be powered by the electrical grid or by batteries. In the common case, they are close to the end users and have plenty of energy [3]. In the worst case, they can be deployed for long periods in remote locations where human intervention or energy harvesting are not possible [4]. Providing sustainable IoT systems by reducing energy consumption and carbon footprint is essential [5]. To build these energy efficient systems, achieving accurate quantification of power, energy consumption and remaining lifetime is crucial.

Measuring [6] and predicting [5] power is needed for developing energy efficient solutions. Several IoT edge nodes architectures do not have internal power sensors. Having power monitoring hardware [7] for a large scale distributed IoT edge system is challenging [8] due to cost, setup, and maintenance. Moreover, it is challenging to monitor individual processes running together using external hardware-based power meters [9].

Besides external power measurements, power can be estimated with power models [10]–[13]. Software-defined power meters use these models to estimate power usage of physical [9], [14], [15] and simulated [16]–[18] nodes.

To perform accurate studies, power models have to reflect the behavior of nodes. Modern Systems on Chip (SoC) multiprocessors, used in IoT edge nodes, offer adaptive frequency for energy efficiency [19]. Dynamic Voltage and Frequency Scaling (DVFS) permits to adapt the frequency and the supply voltage of nodes at run time [20]. It is a feature that impacts the energy consumption of IoT edge nodes [21], [22].

However, most power models and software-defined power meters do not consider CPU frequencies [23]–[25]. Up to our knowledge, this paper is the first one considering CPU frequencies when building power models for single-board computer based IoT edge nodes.

First, this paper compares a state of the art software-defined power meter predictions to real power measurements for CPU and RAM intensive workloads at different CPU frequencies. All experiments and calibrations are conducted on a testbed, comprising IoT edge nodes monitored by accurate external hardware-based power meters. Second, an empirical analysis is conducted to build accurate power models for IoT edge nodes. Models are built considering CPU frequency, temperature, and multiple hardware counters. The impact of these various system metrics on models accuracy is studied.

This paper presents the following contributions:

- Demonstrating how existing software power meters for IoT edge nodes (*e.g.,* PowerJoular) can have major inaccuracies in predictions due to overlooking available node configurations or power modes. We compare a state of the

art software-defined power meter and an accurate external power monitoring, on IoT edge nodes.
- Building power models by comparison of different prediction methods, considering several system metrics and regression methods. By taking into account CPU frequency changes for IoT edge nodes, we thus propose CPU frequency aware power models.

The paper is structured as follows. Section II presents the state of the art. Section III introduces the experimental setup. Section IV details the experimental protocol. Section V analyzes the results of software power meters estimations with DVFS. Section VI presents results analysis of model training and evaluation. A discussion about the work is given in Section VII. Finally, Section VIII concludes the work.

II. State of the art

Multiple energy efficiency leverages exist in the literature [26]. DVFS is one of the most widely used. In opposition to its wide adoption in supercomputers and cloud literature [27], DVFS is still an open and promising leverage for IoT edge nodes [21], [22], [28]–[30].

A. DVFS energy leverage at the edge

At the edge, nodes can spend an important time ON but operating at low CPU utilization (*e.g.,* edge caching [31], media steaming [32]. In [20], authors study power management strategies for IoT edge nodes with long idle periods. Low CPU utilization and workload fluctuations make DVFS promising for energy efficiency. DVFS is becoming available on an increasing number of micro-controllers [33] and single board computers [21]. It is an important leverage that is gaining interest in IoT and edge systems [21], [28], [34]. It is suitable for memory bound workloads and for workloads with long idle periods and target deadlines [17], [20].

In the IoT edge context, DVFS is studied to optimize energy consumption [21], [35], but additional research is needed in such context [20].

In [28], authors co-optimize DVFS and offloading parameters via deep reinforcement learning to improve the energy efficiency of IoT edge nodes. Experiments are conducted on several physical NVIDIA IoT edge nodes supporting DVFS. Authors from [34] show how DVFS performs on IoT edge nodes with common governors (on-demand and conservative) by performing experiments on Jetson Nano and Raspberry Pi 4B. The relation between CPU utilization and power usage is explored for one CPU frequency. However, important details about the experimental protocol are not described. The power monitoring setup and important experimental parameters (*e.g.,* benchmark, etc) are not detailed.

Related works stress the need to study DVFS to preserve quality of service while saving energy for IoT edge nodes [35]. Accurate tools are needed to facilitate such studies.

B. Power models and CPU frequency

In [36], a self-calibrating software-defined power meter (SelfWatts) is proposed for estimating energy consumption of processes and containers of heterogeneous High Performance Computing nodes. It is based on CPU and DRAM power models. These models are built using nodes energy counters in Intel's Running Average Power Limit (RAPL) and hardware performance metrics. This is done at runtime to allow continuous energy predictions for dynamic workloads or node configurations. Calibration is not needed. Several IoT edge nodes do not have RAPL support. Up to our knowledge, there is no similar methodology that automatically selects system metrics for training and self calibration for IoT edge nodes.

In [24], an automated power modeling approach is proposed for generating and sharing power models of IoT edge nodes. Benchmarks are run on a set of Raspberry Pi. These measurements are used to generate power models by polynomial and linear regression. For all nodes, the error percentage using linear regression is higher than the error percentage using polynomial regression. However, the models overlook important node configurations like CPU frequency, and the experimental protocol for the models validation is not detailed.

In [25], power models are built for several IoT edge nodes (*e.g.,* CubieBoard3, Odroid-C2, and Raspberry Pi 3B). Different node components are modeled.

In [37], a framework called WattEdge is proposed to analyze energy consumption of Raspberry Pi 3B+. Energy measurements are collected and used in linear regressions to build energy models. Models are validated by comparing real power measurements to model predictions.

In [15], a software-defined power monitoring (PowerJoular) is proposed for several platforms, including single-board computers. It is based on power models for CPU and GPU components. Power models are proposed per node types. For Raspberry Pi, power models are built using polynomial regression. The work reports that PowerJoular automatically detects computer configuration. Although multiple CPU frequencies are supported on the nodes, CPU frequency is overlooked in benchmarking experiments.

In [23], authors investigate building unified power models for Raspberry Pi 3B and 3B+. The work shows that representing maximum CPU frequency in the model can increase its portability. Different statistical modeling methods (Support Vector Regression, Gaussian Regression, Kernel Ridge Regression, and Linear Regression) are compared. Linear regression is considered as a baseline. Hardware counters and power measurements are collected while executing two benchmarks: NAS and RODINIA. Collected hardware counters are L2 Data Cache Misses, L2 Data Cache Access, Storage Instructions, Branch Instructions, Total Instructions, and Total Executed Cycles. SVG has the minimum mean error. But its maximum error is higher than the linear regression one. However, the experimental protocol is not fully described (*e.g.,* kernel used in NAS is not specified). Node configurations such as CPU frequency are overlooked. The importance of using different hardware metrics on model accuracy is not investigated. In [11], power models for micro-controller edge node (ESP32) are contributed. CPU instructions are used to build the model. Benchmarking experiments are done for several components,

including CPU and RAM. Experiments are conducted for multiple CPU frequencies and utilization levels. Two methods are compared to build the models: polynomial regression and random forest. Reported random forest model accuracy is slightly higher than polynomial models accuracy. However, the variables used in the polynomial models are not fully detailed in the paper. No experimental artifacts are given, making it difficult to reproduce and compare.

In [38], a power model is proposed for an ARM11 microprocessor. The work acknowledges that cache misses are widely used in literature to build power models. CPU Pipeline stalls are collected instead of cache misses. No comparison is made between using pipeline stalls and cache misses.

Multiple studies in the state of the art aim at building and improving power models for IoT edge nodes. Several works overlook important node configurations like CPU frequency. In an IoT edge system, errors in power estimations can be cumulative. Unexpected error in predictions can have a major impact on energy budgets [6], [39].

III. Experimental Setup

A. Hardware environment

The IoT-to-Extreme-Edge testbed is used as an experimental environment [7]. This testbed comprises Single Board Computer (SBC) nodes along with external power meters. In this work, 10 Raspberry Pi model 4 revision 1.4 (detailed in Table I) are used for our experiments. Each node runs Debian 11 (Bulleyes). These nodes are monitored by INA260 power meters. During experiments, power measurements are collected at regular intervals through I^2C by BeagleBone Black SBCs. The effective delay between measurement samples is less than 10ms. More details about the power monitoring architecture and features are given in [7].

To ensure proper time synchronization between experimental phases and power measurement samples, nodes of the testbed are synchronized through NTP (via *chrony*) before each round of the experiment. The time reference is provided by a local node being part of the testbed (the front-end).

B. Software tools

This work uses PowerJoular, a state of the art software power meter in the context of IoT edge nodes under DVFS. In addition, to build the proposed CPU frequency aware power models, a metric collector software is developed to gather various system metrics exposed by the Linux kernel. The following system metrics are gathered: 1) Temperature from *sysfs* 2) CPU Usage (global and per core) from *procfs* 3) Cores frequencies from *sysfs* 4) Performance events from the Linux kernel *perf* subsystem (`PERF_COUNT_HW_INSTRUCTIONS` and `PERF_COUNT_HW_CACHE_MISSES`).

Cache misses stall the CPU and cause memory access. They thus impact power and execution time. They are highly used in literature to build power models [38]. Total instructions count provides an overview of conducted computation in the CPU.

This subset of critical metrics is gathered to limit the impact of the metric collector on the energy consumption. These system metrics are collected repeatedly at an average frequency of 10Hz (one retrieval of all system metrics every 100ms). This metric collector is available online [40].

C. Workloads

Two types of workloads are considered in this work. The first one consists of two kernels and one pseudo application from the NAS Parallel Benchmarks v3.4.2: 1) Embarrassingly Parallel (EP), computationally intensive 2) Multi-Grid (MG), memory intensive 3) Lower-Upper Gauss-Seidel solver (LU), common computational kernel with computations and memory accesses. In several related works, power models are built using several benchmarks, including NAS benchmark on IoT edge nodes [23].

The second workload is a C program developed to accurately stress CPU cores at a specific load (percentage) for a specific duration. This tool executes a user specified computation (*e.g.,* square root, multiplication, or sum) while aiming for a targeted CPU load. The approach used to maintain a steady CPU load is to alternate between carefully timed compute and rest phases. These phases execute during CPU Loader slots from Figure 1(b) (yellow boxes).

IV. Experimental protocol

A. Performance metrics

To evaluate software power meters, various metrics are used. The first is the average power given by the average value of n power measurements during a benchmark on a given node:

$$\bar{P} = \frac{1}{n}\sum_{i=1}^{n} P_i \tag{1}$$

From n power measurements, the energy consumption of nodes can be estimated using the trapezoidal rule:

$$E = \int_0^t P(t) \approx \sum_{i=1}^{n-1} \frac{(t_{i+1} - t_i)(P_i + P_{i+1})}{2} \tag{2}$$

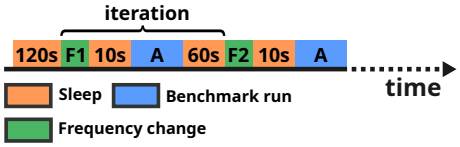
To evaluate the power predictions given by the regression models, the mean absolute percentage error (MAPE) between the predictions $P_{predicted}$ and the measurements $P_{measured}$. MAPE is defined as follows:

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n} \left| \frac{P_{measured} - P_{estimated}}{P_{measured}} \right| \tag{3}$$

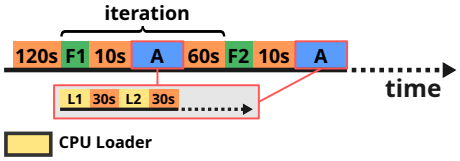
B. Power prediction models

This work investigates the power prediction capabilities of two common regression models, namely: 1) Linear 2) Polynomial of degree two. These predictions are made according to various system metrics that provide information about the workload and the current operating CPU frequency.

Linear regression is simple to implement and interpret, as it looks for simple linear relationship between independent variables. Polynomial regression is more complex but provides more flexibility as it can fit more closely to data with non-linear relationship.



(a) Experiment phases for the NAS parallel benchmarks.



(b) Experiment phases for the CPU Loader benchmarks.

Fig. 1. Experiment protocol occurring on a node for the NAS parallel benchmark and CPU loader workloads. The protocol is composed of multiple frequency iteration that represents a benchmark run (in blue) after setting a new frequency (in green). It also contains resting periods (in orange).

C. Experiment 1: Software Power Meters overlooking CPU frequencies

As discussed in section II, most of the software power meters overlook CPU frequencies. We take PowerJoular as an illustrative example of a software power meter that does not take into account CPU frequencies. The first experiments evaluate the software power meter called PowerJoular v0.7.0, while varying CPU frequency. The workload consists of the three NAS parallel benchmarks introduced previously (EP, MG and LU). To be able to dynamically change the core frequencies, the governor of each core is set to *performance*. For each benchmark, the core frequencies are varied from 600MHz to 1 800MHz by steps of 100MHz using the *scale_freq_max* parameter. During the experiments, power measurements are collected from the software power meter (*i.e.,* PowerJoular) and the hardware-based power meters (*i.e.,* INA260). Both are compared to evaluate the accuracy of PowerJoular in contexts where CPU frequency varies. Each experiment is executed 10 times on each node to account for results variability. The experiment protocol is detailed in Figure 1(a).

D. Experiments 2: Model training and evaluation

To build our CPU frequency aware power models, offline training is performed. System metrics need to be collected during the execution of the benchmarks. The system metrics used to train the models are fetched by the metrics collector (detailed in Section III-B). The benchmarks used to build the models are the NAS parallel benchmarks and the CPU loader detailed in Table I. For the CPU loader experiments, cores are loaded gradually up to 100%, one after the other, until all cores are fully loaded. Each step lasts 30s. A cool down period of 30s is added between each step (see Figure 1(b)).

Three datasets are built to evaluate the models, each containing the system metrics collected from the following scenarios: (i) compute and memory intensive: with NAS parallel benchmark, (ii) varying load: with the created CPU loader and (iii) hybrid: a fair share distribution of data collected, where half of the workload comes from (i) and half from (ii). Each of these datasets is used to train and test the models. These datasets are split into training (80%) and test (20%) sets.

TABLE I
OVERVIEW OF EXPERIMENTAL PARAMETERS

Hardware	
Node count	10
Nodes hardware	Raspberry Pi 4 model B Rev 1.4
Hardware-based power meter	INA260
CPU Frequency	
Scaling governors	performance
Cores scale_freq_min	600MHz
Cores scale_freq_max	600MHz-1 800MHz by 100MHz
Workload	
NAS parallel benchmarks	EP, MB and LU
CPU load targets range	1%-10% by 1% 10%-100% by 10%

V. Results analysis of Software Power Meter estimations, using various CPU frequencies

Following the experimental protocol described in IV-C, the software power meter estimations, when modifying CPU frequency, are analyzed in this section.

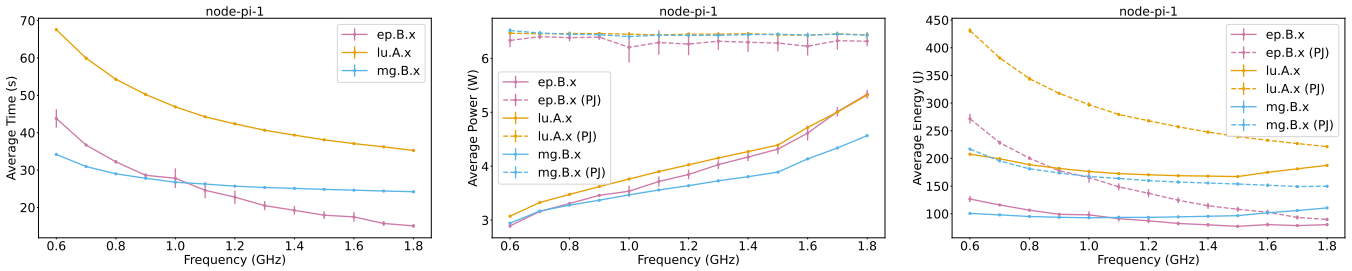
Figure 2 shows the results for the three NAS parallel benchmarks while varying CPU frequencies. The execution time, power measurements, power predictions (PowerJoular) and energy consumption are reported. Each point on the graphs is an average of 10 runs. Standard deviation is also shown.

Figure 2(a) focuses on the execution time of each benchmark (see Table I). Overall, average execution time decreases when frequency increases. Because of the property of each benchmark (computational and memory intensive), the slopes of the curves are different since the impact of CPU frequency varies across benchmarks.

Figure 2(b) shows the average power for each NAS benchmarks with both PowerJoular predictions and the INA260 measurements. For the three benchmarks, average power measured increases as the CPU frequency increases. However, power estimations given by PowerJoular show high and stable values. It shows that PowerJoular do not account for the CPU frequency changes on the node to make power predictions.

Figure 2(c) shows the average energy consumption for each benchmark with PowerJoular and the INA260. Energy consumption estimated by PowerJoular and measured by external monitoring differ for all benchmarks. PowerJoular energy consumption estimations always decrease. It is not the case for energy consumption derived from the measured power. For example, with LU.A.x benchmark, energy consumption measured by external power monitoring decreases at start as CPU frequency increases (up until 1.5 GHz), then increases.

For EP.B.x benchmark, at the highest frequency, measured and estimated energy consumption do not intersect (but are close to each other). In studied scenarios, execution time is



(a) Execution time (b) Average power (c) Energy consumption

Fig. 2. Experiments run on one IoT edge node with different CPU frequency for the three NAS parallel benchmarks. Power is measured by external monitoring and estimated by PowerJoular. Each point is an average over 10 runs. The standard deviation is also reported.

relatively small (lower than 60s). In other scenarios where execution time is higher, this difference in energy could be more significant.

Figure 3 shows execution time, average power and energy consumption for 10 nodes. Each point on the graphs is an average of 100 runs (10 per nodes). The standard deviation is also reported. These aggregated results show that Figure 2 is an accurate overview of the nodes average power, energy consumption and benchmarks execution times. Trends and values are similar and thus same conclusions can be derived.

For the entire set of experiments, the MAPE between values given by PowerJoular and external monitoring is 39.62%. PowerJoular gives inaccurate results in this context.

Most power estimators for IoT edge nodes like PowerJoular do not consider CPU frequency changes, thus differ from external power meter measurements. Extracting accurate knowledge about energy efficiency of an edge system when CPU frequency changes is not possible. Having CPU frequency aware power prediction models would allow to study edge system scenarios where CPU frequency is used as leverage (e.g. for energy efficiency).

VI. Model Training and Evaluation Results

A. Single system metric based models

Results for models trained with a single system metric are presented in Figure 4 for temperature (T), CPU frequency (F), CPU usage (U), Instruction count (I) and Cache misses count (C) metrics (introduced in III-B).

The figure shows results for both models trained on three datasets as mentioned in Table I. The mean and median of the MAPE are given for each model (linear and polynomial). The standard deviation is also represented.

The results with the NAS parallel benchmarks dataset show that using the cores CPU frequency alone gives the largest error between the predicted power and the measured one. Similarly, temperature is not the best system metric to predict power, with an average median error of 10% with the real measurements. In this case, CPU usage and instruction count provide the lowest MAPE.

For the CPU Loader workload, cores CPU frequency and cache misses system metrics lead to the largest error. As this workload is purely compute-bound, CPU frequency gives a lower mean and median MAPE compared to the NAS dataset. On the other hand, cache misses give larger errors compared to the NAS dataset. The compute-bound nature of the workload makes the instruction count system metric the most accurate for power estimation with a MAPE of 5%.

For NAS benchmarks, using CPU usage or instructions count to build the models gives similar MAPE. However, for CPU Loader workload, using instructions count gives lower MAPE than using CPU usage. The relevance of the metrics to train the models can thus change depending on the workload.

Combining both datasets provides a trade-off between the results of each dataset (NAS and CPU loader). This is the case for each system metric. Still, CPU frequency, temperature, and cache misses lead to the largest error.

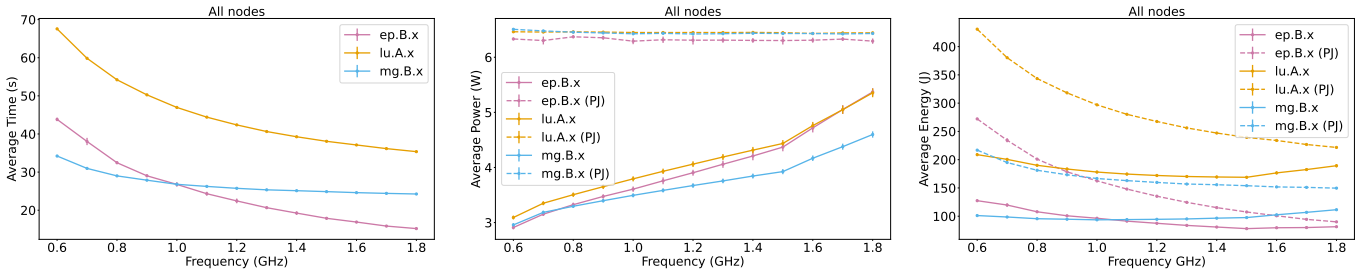
Overall, the linear regression model has a higher MAPE compared to the polynomial. This difference is in the order of 3%. The median MAPE is smaller than the mean one in most cases. This is because the mean MAPE captures potential outliers, leading to a larger error. Standard deviations are low, indicating that, overall, the reported MAPE is a good indicator of the model predictions MAPE over the various training and test phases. To try to improve prediction accuracy, multivariate regression must be investigated. The following section investigates the use of multiple system metrics to train the model.

B. Combined system metrics based models

Figure 5 depicts the MAPE of power predictions with models trained on different combinations of system metrics. For space reasons, only a subset of all possible system metric combinations is reported.

Results for the NAS dataset show that combining temperature and frequency achieves a lower MAPE than frequency alone, but worse than temperature alone. More generally, as the number of relevant system metrics used to train the model increases, the model MAPE decreases. Combining all system metrics produces a median (and mean) MAPE less than 2%, for the polynomial model.

Combining system metrics with CPU loader workload shows similar trends. Overall, the MAPE is smaller with this workload, as it is purely compute-bound.



(a) Execution time (b) Average power (c) Energy consumption

Fig. 3. Experiments run on 10 nodes with different CPU frequency for the three NAS parallel benchmarks. Power is measured by external monitoring and estimated by PowerJoular. Each point is an average over 100 runs (10 per nodes). The standard deviation is also reported.

Using the dataset with both workloads (NAS and CPU loader) compared to CPU loader dataset alone, the MAPE increases when combining temperature and frequency. However, by combining all system metrics (referred to in Figure 5 as T_U_F_I_C) to train the models, a MAPE of around 2% can be achieved. This suggests that adding more metrics could help to generalize the models to other workloads while maintaining a low MAPE. But, combining numerous metrics could lead to overhead in monitoring and low improvement of MAPE compared to scenarios with at least the temperature, CPU usage, and CPU frequency system metrics (referred in Figure 5 as T_U_F), highlighting a trade-off that needs to be studied.

VII. Discussions

A. CPU frequency aware power model

In the literature, DVFS is a commonly used leverage to save energy. Accounting for CPU frequency change in software power meters is thus crucial. The results presented in this work demonstrate that predicting the power draw of an IoT edge node under various CPU frequencies with a regression model is possible. Depending on the dataset used to train the model and the combination of system metrics used, a reasonable MAPE is achieved. The results show that using datasets from various workloads could help to generalize the models.

Tuning the proportion of each workload present in the training dataset is important. In this work, the training dataset was evenly split between NAS and CPU loader workload. Training the model on multiple workloads and fine-tuning the proportion of data for each workload, to ensure accuracy and appropriate generalization, are strong improvement areas.

B. Metric collection overhead on IoT edge node

IoT edge nodes are usually limited in computing resources. Monitoring many metrics concurrently, and at high frequency, can induce overhead. This work does not study the overhead of the metric collection process. Studying this point would help in the assessment of the prediction's accuracy. To mitigate the impact of the metric collection process, reducing the frequency at which system metrics are collected is important. A potential trade-off between overhead, chosen metrics, and metric collection frequency could be found.

C. Power estimation overhead on IoT edge node

Power estimation itself has an impact on the IoT edge node when done locally. The power estimation requirements in terms of computation vary depending on the estimation model type and the number of metrics used. Polynomial models provide better accuracy than linear models but require more computation, not only when training them but also when using them. Even though polynomial models produce better results in the vast majority of cases, linear models can be useful in environments where power estimation overheads must be kept as low as possible. Power estimation should have low impact on node energy consumption, especially in energy frugal contexts.

Additionally, the metrics considered by power estimation models have an impact on accuracy. But this accuracy does not increase linearly with the addition of metrics. As a result, an optimal set of metrics can be identified to provide a trade-off between power estimation accuracy and overhead.

This work does not study the overhead of the power estimation process. To reduce this overhead, the estimation frequency, model used, and metrics considered can be adjusted, and a trade-off between them must be found.

VIII. Conclusion

With the importance of IoT edge systems in today's infrastructures, being able to study IoT edge nodes energy consumption is crucial to design energy frugal systems. To the best of our knowledge, most of the software-defined power meters do not account for CPU frequency, which is an important energy leverage. This work demonstrates that a state of the art software power meter for IoT edge nodes does not account for potential fluctuations of CPU frequency. In scenarios with varying CPU frequencies, this leads to wrong power estimations for IoT edge nodes.

We demonstrate that linear and polynomial regression-based power prediction models are able to account for various CPU frequencies on IoT edge nodes. Results show that polynomial regression provides better accuracy compared to linear regression-based models. By using the correct system metrics, these models can predict the power of IoT edge nodes running a specific workload with a MAPE of 2% compared

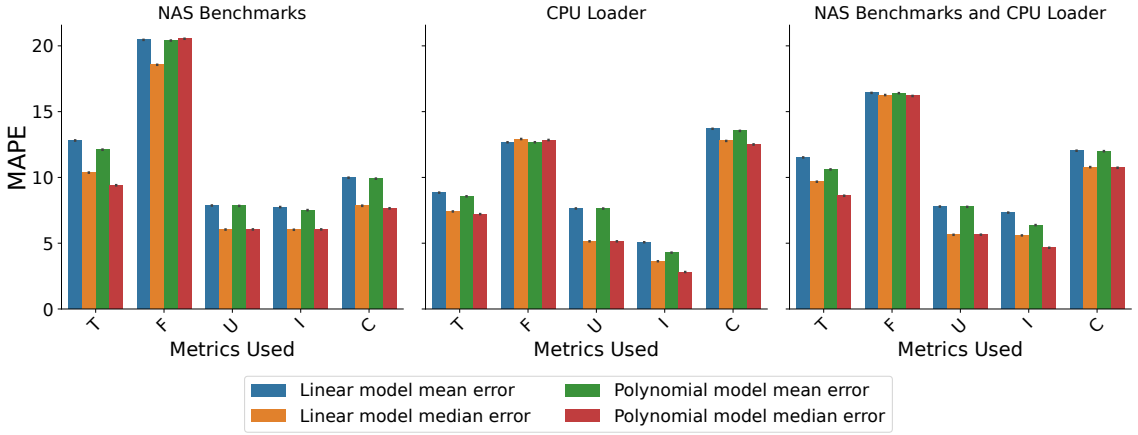


Fig. 4. Mean and median MAPE while training model with different system metrics: Temperature (T), CPU Frequency (F), CPU Usage (U), Instruction Count (I), Cache Misses Count (C).

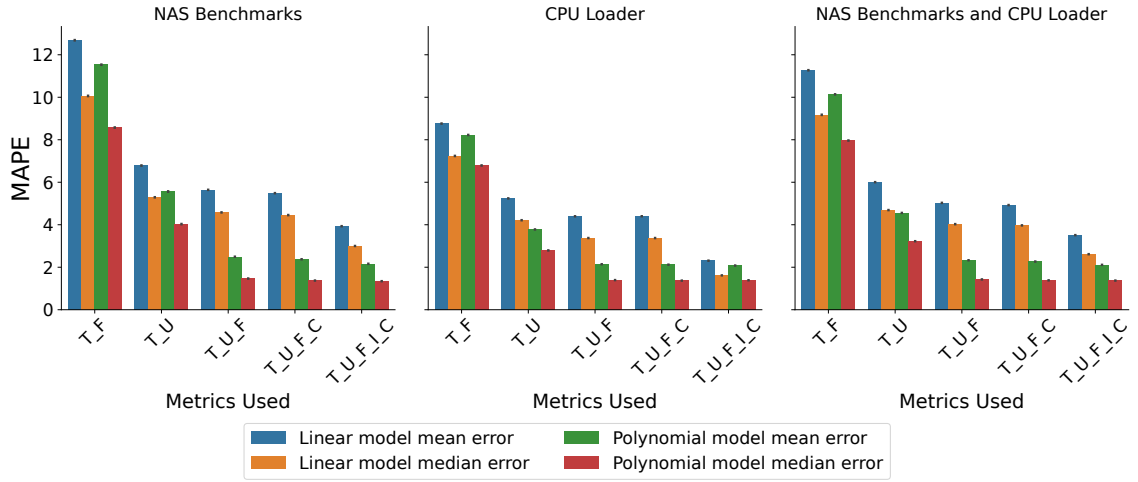


Fig. 5. Mean and median MAPE while training model with different combination of system metrics: Temperature (T), CPU Frequency (F), CPU Usage (U), Instruction Count (I), Cache Misses Count (C).

to accurate hardware-based power meter measurements. The models designed in this work are available online [40].

Future works include investigating accurate generalization of the models to different workloads, further experiments with other ML models, and evaluating their accuracy. Studying the trade-offs between models accuracy, sample rate at which the metric collector operates, number of metrics used, and estimation overhead is also planned.

IX. Acknowledgment

This joint work is supported by the Aurora Mobility Program from the French Ministry of Europe and Foreign Affairs (MEAE) and the French Ministry of Higher Education and Research (MESR) and the Research Council of Norway.

References

[1] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "Enorm: A framework for edge node resource management," *IEEE transactions on services computing*, vol. 13, no. 6, pp. 1086–1099, 2017.

[2] L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, P. Zeng, M. Khan, and S. K. Das, "Edge-computing-driven internet of things: A survey," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–41, 2022.

[3] M. Kamal, M. Atif, H. Mujahid, T. Shanableh, A.-R. Al-Ali, and A. Al Nabulsi, "Iot based smart city bus stops," *Future Internet*, vol. 11, no. 11, p. 227, 2019.

[4] M. J. Murphy, M. Tveito, E. F. Kleiven, I. Raïs, E. M. Soininen, J. M. Bjørndalen, and O. Anshus, "Experiences building and deploying wireless sensor nodes for the arctic tundra," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2021, pp. 376–385.

[5] P. Dibal, E. Onwuka, S. Zubair, E. Nwankwo, S. Okoh, B. A. Salihu, and H. Mustaphab, "Processor power and energy consumption estimation techniques in iot applications: A review," *Internet of Things*, vol. 21, p. 100655, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660522001366

[6] S. Tofaily, I. Raïs, and O. Anshus, "Quantifying the variability of power and energy consumption for iot edge nodes," in *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. IEEE, 2023, pp. 577–584.

[7] L. Guegan, S. Tofaily, and I. Raïs, "Design and Evaluation of Single-Board Computer Based Power Monitoring for IoT and Edge Systems," in *2023 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE*

Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics). Danzhou, China: IEEE, Dec. 2023.

[8] M. Kasioulis, M. Symeonides, G. Pallis, and M. D. Dikaiakos, "Power estimation models for edge computing devices," in *Euro-Par 2023: Parallel Processing Workshops*, D. Zeinalipour, D. Blanco Heras, G. Pallis, H. Herodotou, D. Trihinas, D. Balouek, P. Diehl, T. Cojean, K. Fürlinger, M. H. Kirkeby, M. Nardelli, and P. Di Sanzo, Eds. Cham: Springer Nature Switzerland, 2024, pp. 257–269.

[9] G. Fieni, R. Rouvoy, and L. Seinturier, "Smartwatts: Self-calibrating software-defined power meter for containers," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. IEEE, 2020, pp. 479–488.

[10] F. Kaup, P. Gottschling, and D. Hausheer, "Powerpi: Measuring and modeling the power consumption of the raspberry pi," in *39th Annual IEEE Conference on Local Computer Networks*. IEEE, 2014.

[11] E. O. Lange, J. M. Jose, S. Benedict, and M. Gerndt, "Automated energy modeling framework for microcontroller-based edge computing nodes," in *Advanced Network Technologies and Intelligent Computing*, I. Woungang, S. K. Dhurandher, K. K. Pattanaik, A. Verma, and P. Verma, Eds. Cham: Springer Nature Switzerland, 2023, pp. 422–437.

[12] F. Kaup, S. Hacker, E. Mentzendorff, C. Meurisch, and D. Hausheer, "Energy models for nfv and service provisioning on fog nodes," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–7.

[13] L. Ardito and M. Torchiano, "Creating and evaluating a software power model for linux single board computers," in *2018 IEEE/ACM 6th International Workshop on Green And Sustainable Software (GREENS)*, 2018, pp. 1–8.

[14] M. Jay, V. Ostapenco, L. Lefèvre, D. Trystram, A.-C. Orgerie, and B. Fichel, "An experimental comparison of software-based power meters: focus on cpu and gpu," in *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 2023, pp. 106–118.

[15] A. Noureddine, "Powerjoular and joularjx: Multi-platform software power monitoring tools," in *2022 18th International Conference on Intelligent Environments (IE)*, 2022, pp. 1–4.

[16] L. Guegan, I. Raïs, and O. Anshus, "Validation of esds using epidemic-based data dissemination algorithms," in *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. IEEE, 2023, pp. 277–284.

[17] M. A. Awan and S. M. Petters, "Race-to-halt energy saving strategies," *Journal of Systems Architecture*, vol. 60, no. 10, pp. 796–815, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762114001295

[18] S. Tofaily, I. Raïs, and O. Anshus, "Representing power variability of an idle iot edge node in the power state model," in *2023 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*. IEEE, 2023, pp. 540–546.

[19] E. Nogues, M. Pelcat, D. Menard, and A. Mercat, "Energy efficient scheduling of real time signal processing applications through combined dvfs and dpm," in *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, 2016.

[20] A. Di Mauro, H. Fatemi, J. P. de Gyvez, and L. Benini, "Idleness-aware dynamic power mode selection on the i.mx 7ulp iot edge processor," *Journal of Low Power Electronics and Applications*, vol. 10, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2079-9268/10/2/19

[21] S. K. Panda, M. Lin, and T. Zhou, "Energy-efficient computation offloading with dvfs using deep reinforcement learning for time-critical iot applications in edge computing," *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 6611–6621, 2023.

[22] T. Zhou and M. Lin, "Deadline-aware deep-recurrent-q-network governor for smart energy saving," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 6, pp. 3886–3895, 2022.

[23] J. M. Paniego, L. Libutti, M. P. Puig, F. Chichizola, L. De Giusti, M. Naiouf, and A. De Giusti, "Unified power modeling design for various raspberry pi generations analyzing different statistical methods," in *Argentine Congress of Computer Science*. Springer, 2019, pp. 53–65.

[24] H. Kanso, A. Noureddine, and E. Exposito, "Automated power modeling of computing devices: Implementation and use case for raspberry pis," *Sustainable Computing: Informatics and Systems*, vol. 37, 2023.

[25] F. Kaup, S. Hacker, E. Mentzendorff, C. Meurisch, and D. Hausheer, "The progress of the energy-efficiency of single-board computers," *Tech. Rep. NetSys-TR-2018-01*, 2018.

[26] I. Raïs, "Discover, model and combine energy leverages for large scale energy efficient infrastructures," Ph.D. dissertation, Université de Lyon, 2018.

[27] X. Mei, Q. Wang, and X. Chu, "A survey and measurement study of gpu dvfs on energy conservation," *Digital Communications and Networks*, vol. 3, no. 2, pp. 89–100, 2017.

[28] Z. Zhang, Y. Zhao, H. Li, C. Lin, and J. Liu, "Dvfo: Learning-based dvfs for energy-efficient edge-cloud collaborative inference," *IEEE Transactions on Mobile Computing*, 2024.

[29] H. Ali, U. U. Tariq, L. Liu, J. Panneerselvam, and X. Zhai, "Energy optimization of streaming applications in iot on noc based heterogeneous mpsocs using re-timing and dvfs," in *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2019.

[30] J. L. Bez, E. E. Bernart, F. F. dos Santos, L. M. Schnorr, and P. O. A. Navaux, "Performance and energy efficiency analysis of hpc physics simulation applications in a cluster of arm processors," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 22, 2017.

[31] J. Xu, K. Ota, and M. Dong, "Saving energy on the edge: In-memory caching for multi-tier heterogeneous networks," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 102–107, 2018.

[32] H. Ali, U. U. Tariq, L. Liu, J. Panneerselvam, and X. Zhai, "Energy optimization of streaming applications in iot on noc based heterogeneous mpsocs using re-timing and dvfs," in *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2019.

[33] Y. H. Yassin, M. Jahre, P. G. Kjeldsberg, S. Aunet, and F. Catthoor, "Fast and accurate edge computing energy modeling and dvfs implementation in gem5 using system call emulation mode," *Journal of Signal Processing Systems*, vol. 93, no. 1, pp. 33–48, 2021.

[34] T. Zhou, H. Wang, X. Li, and M. Lin, "Profiling and understanding cpu power management in linux," in *2023 IEEE Smart World Congress (SWC)*. IEEE, 2023, pp. 1–8.

[35] M. H. Alsharif, A. H. Kelechi, A. Jahid, R. Kannadasan, M. K. Singla, J. Gupta, and Z. W. Geem, "A comprehensive survey of energy-efficient computing to enable sustainable massive iot networks," *Alexandria Engineering Journal*, vol. 91, pp. 12–29, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110016824001091

[36] G. Fieni, R. Rouvoy, and L. Seiturier, "Selfwatts: On-the-fly selection of performance events to optimize software-defined power meters," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 2021, pp. 324–333.

[37] M. S. Aslanpour, A. N. Toosi, R. Gaire, and M. A. Cheema, "Wattedge: A holistic approach for empirical energy measurements in edge computing," in *Service-Oriented Computing: 19th International Conference, ICSOC 2021, Virtual Event, November 22–25, 2021, Proceedings 19*. Springer, 2021, pp. 531–547.

[38] W. Wang and M. Zwolinski, "An improved instruction-level power model for arm11 microprocessor," *High Performance Energy Efficient Embedded Systems (HIP3ES)*, 2014.

[39] T. Trathnigg, M. Jürgen, and R. Weiss, "A low-cost energy measurement setup and improving the accuracy of energy simulators for wireless sensor networks," in *Proceedings of the workshop on Real-world wireless sensor networks*, 2008, pp. 31–35.

[40] V. Ostapenco, L. Guégan, S. Tofaily, I. Raïs, and L. Lefèvre, "Models Git Repositories," https://gitlab.com/manzerbredes/aurora-team-2023, [Accessed 18-12-2024].