Empowering Sustainability: Energy Labeling of Digital Services Using Simulation

Saeedeh Baneshi[1], Anuj Pathania[1], Benny Akesson[1,3], Andy Pimentel[1], Ana-Lucia Varbanescu[2]
[1]*University of Amsterdam, The Netherlands*
[2]*University of Twente, The Netherlands*
[3]*TNO-ESI, Eindhoven, The Netherlands*
{s.baneshi, a.pathania, k.b.akesson, a.d.pimentel}@uva.nl, a.l.varbanescu@utwente.nl, k.b.akesson@tno.nl

***Abstract*—The energy consumption of digital services has become a concern for stakeholders committed to sustainability. Raising awareness of this consumption is essential to improve the energy efficiency of digital services. However, expressing the energy usage of digital services in an easily understandable and actionable way remains a challenge.**

We address this challenge by proposing a first operational energy labeling method for digital services in the computing continuum. Our approach enables stakeholders, including cloud and network providers, application developers, researchers, and end-users of digital services, to better understand and improve the energy efficiency of their applications.

Focusing on video surveillance digital services, and using the enhanced iFogSim framework, we propose an energy labeling scheme, and demonstrate its merits with extensive scenario analysis and simulation. We further discuss how our approach can help reduce energy consumption and/or improve performance, all without modifying the application's functional parameters or system architecture.

***Index Terms*—Application Energy Labels, Sustainable Digital Services, Computing Continuum, Energy Consumption**

I. Introduction

Digital services have become indispensable in our daily lives, driving advancements in communication, entertainment, healthcare, and online shopping. However, their growing energy demands raise sustainability concerns, given the fast-paced digitization processes and their significant contribution to global energy consumption and carbon emissions [1].

To address these concerns, detailed data collection and analysis are needed to accurately assess the energy consumption of digital services. However, the distributed nature of such services and the complex, heterogeneous, multi-tenant infrastructure they use make it difficult to measure and attribute energy consumption. The lack of widely accepted energy measurement frameworks for complex applications and systems further limits awareness and hinders informed decision-making for better energy consumption optimization. Finally, the limited understanding of the specialized data and metrics that various frameworks provide (e.g., FLOPS/Watt) further hinders the ability of stakeholders to understand when and how they can afford to reduce energy consumption.

Raising awareness of the energy consumption of digital services in a meaningful and action-ready way is crucial for sustainability in the Information and Communication Technology (ICT) sector. We argue that effectively expressing the *current* energy usage of digital services can benefit stakeholders, including cloud providers, application developers, and end-users, but remains a challenge. Thus, we investigate the merits of energy labels for digital services based on the idea that *energy labeling*, standardized for many appliances, helps consumers make informed choices with a better understanding of the trade-offs such choices imply. Adopting such a scheme in ICT is limited to a couple of attempts, for machine learning models and data centers [2]–[4], however, it holds promise to raise awareness and empower various stakeholders in making choices for more sustainable deployment and operation.

In this work, we propose a first transparent energy labeling system for digital services in the computing continuum. We demonstrate our energy labeling using a configurable video surveillance application, a workload with high data processing and energy demands. We construct the energy labels using energy consumption data from simulations conducted using an enhanced iFogSim framework on a four-tier computing continuum architecture, demonstrating how our the labeling scheme can differentiate and classify the simulated scenarios based on their energy consumption. Finally, we briefly discuss how these labels could reduce energy consumption.

Our work makes two major contributions: (1) we demonstrate how to use simulation models to characterize the energy consumption of digital services across deployment scenarios and application-specific parameters, and (2) we propose an operational framework that converts an application's energy profile into energy labels. The source code for this work is available at https://github.com/saeedehbaneshi/IFogSim.

II. Background

This section presents the simulation environment, the selected *configurable* workload (i.e., the case-study application), and the continuum architecture.

A. The Enhanced iFogSim Framework

Simulation frameworks such as CloudSim [5], NS-3 [6], and iFogSim [7], assess energy consumption in fog computing. However, they often lack end-to-end energy analysis for the computing continuum, as they focus on either computation

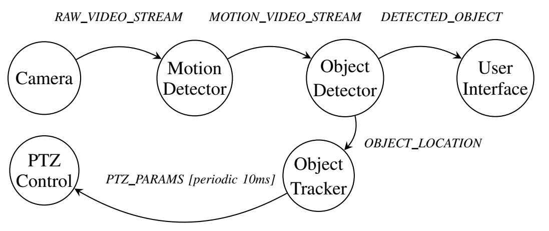

Fig. 1: Surveillance Application Architecture [7].

or networking energy consumption. Moreover, current energy models are often limited to device level, failing to account for multi-tenancy, device-sharing, or requiring low-level inputs [8], [9]. Among these tools, iFogSim models fog environments by placing application modules across continuum devices [7], [10]. However, out-of-the-box, it only reports computing energy consumption, ignoring networking and thus limiting end-to-end energy estimation for distributed applications, and reports only coarse, per-device data.

Recent work introduced an enhanced iFogSim that integrates computational and networking energy for end-to-end estimations in multi-application fog environments [9]. This enhanced iFogSim features a time-based model for estimating Network Interface Card (NIC) energy in end-user devices and a flow-based model for shared devices (fog nodes and cloud data centers), along with fine-grained energy reporting and per Virtual Machine (VM) calculations. These improvements enable detailed energy analysis and optimizations. Therefore, we use the enhanced iFogSim to collect our digital service energy consumption data.

B. Workload: Surveillance Application(s)

Our workload of choice for energy labeling is a surveillance application, a well-known case study in edge, fog, and cloud computing [7], [8]. The application consists of six interconnected modules: (1) **Cameras** continuously capture and transmit video frames; (2) the **Motion Detector** analyses the video streams and identifies moving objects; (3) the **Object Detector** extracts moving objects, calculating their coordinates for subsequent tracking; (4) the **Object Tracker** calculates optimal configurations for Pan-Tilt-Zoom (PTZ) cameras, which adjust via (5) a **PTZ Control** module in smart cameras; finally, (6) the **User Interface** module aggregates processed video feeds, including tracked objects, for the end-users.

Figure 1 presents the surveillance application architecture [7], showing data flow and module interactions. Application parameters such as input frame rate or deployment, significantly impact energy consumption, as seen in Section IV.

C. System: continuum architecture

The computing continuum architecture proposes a hierarchical view of the distributed system for surveillance data processing. In this work we employ a four-layer architecture, where *cameras* (end-user equipment and sensors) connect through a *router* and a *proxy server* to the *cloud*. Figure 2 illustrates this architecture and specifies device latencies.

The system configuration can be modified to assess the performance and energy footprint across various scenarios, as demonstrated in Section IV.

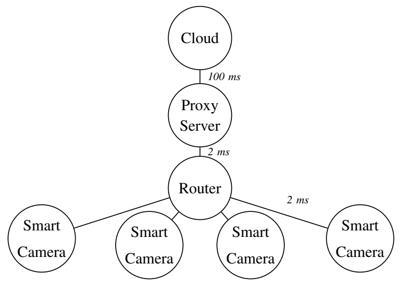

Fig. 2: Continuum architecture including links latencies.

III. Related work

There have been a few proposals for applying energy labels in ICT. For example, Kern et al. [2] introduced a sustainability labeling system for software products and websites, assessing *"Efficiency, Feasibility, and Perdurability"*. They propose various label formats, including energy footprint declarations. However, their framework is theoretical, with limited practical application. Castano et al. [4] analyzed the carbon footprint of Machine Learning (ML) models on the Hugging Face platform, identifying carbon emissions patterns and labeling models from A (most efficient) to E (least efficient) to promote transparency. In a separate study, Duran et al. [3] introduced GAISSALabel, a web-based tool for evaluating ML energy efficiency using weighted metrics like CO_2 emissions, power draw, and model size. Both studies are ML-specific. Moreover, although these studies demonstrate that *labels incentivize sustainable computing*, they overlook device diversity and deployment environments, which impact energy consumption.

Beyond labeling, many studies examine software environmental impacts. Tools like CodeCarbon [11] and Carbontracker [12] estimate software CO_2 emissions, while cloud platforms (AWS, GCP, and Azure) provide ML-specific emission tracking [3]. However, these approaches often overlook deployment scenarios and device diversity in the computing continuum. A prior study also using the enhanced iFogSim [9], demonstrated how deployment impacts energy consumption in video surveillance applications; however, we are taking the next step to introduce energy labeling as a simpler communication and incentives tool.

Building on this foundation, our work explores a broad range of deployment scenarios and establishes an energy labeling system to qualify consumption. The labels facilitate data analysis and enhance stakeholder understanding and participation in promoting sustainability within the ICT sector.

IV. Energy labeling methodology

This section presents our energy labeling approach and demonstrates its feasibility for a surveillance application running in the computing continuum. Intuitively, the labeling process involves two steps: establishing the application's energy consumption range using the enhanced iFogSim framework and categorizing it into energy classes.

TABLE I: Application inter-module parameters [7]

Tuple type	CPU load [MI]	Data Size [B]
RAW_VIDEO_STREAM	1000	20000
MOTION_VIDEO_STREAM	2000	2000
DETECTED_OBJECT	500	2000
OBJECT_LOCATION	1000	100
PTZ_PARAMS	100	28

TABLE II: Fog devices for our continuum architecture [7].

Device	Computational	RAM	Power [W]	
type	Capacity [MIPS]	[GB]	Max	Idle
Cloud	44800	40	1648.0	1332.0
Proxy server	2800	4	107.3	83.4
Router	2800	4	107.3	83.4
Smart camera	500	1	87.5	82.4

A. Energy consumption range

We define a *scenario* as a deployment mapping, i.e., a placement of modules to system components. To define the application's energy consumption range, we identify the best and worst application scenarios. We considered three main options: (1) a generic min/max definition (roughly based only on the system architecture), (2) an application-specific range, and (3) a workload-specific range. The main trade-off between these approaches is generality (one range for all applications) versus scenario-specific differentiation (one range per workload configuration). We adopt the application-aware options ((2) and (3)), thus defining a unified range for multiple configurations.

Studies show that energy consumption decreases when processes run on devices closer to the user and increases when they are moved to the cloud [9]. Therefore, we assume that the *Cloud-based* scenario (all modules on the cloud) is the worst case, and the *Edge-based*, (all modules except the user interface on smart cameras) is the best case. We use the enhanced iFogSim framework to simulate these scenarios, using the application base configuration (i.e., data sizes and processing requirements) in Table I, and the processing capacities in Table II. The camera emission interval is set to 20 ms, directly impacting the application workload.

Because Quality of Experience (QoE) is crucial for digital services, we evaluate all deployment scenarios where the total delay meets QoE requirements and placement restrictions (i.e., user-interface on the cloud), as presented in Table III. The energy consumption results for all these scenarios confirm that the *Cloud-based* scenario is the worst case. However, the *Router-based* scenario, where all modules except the user interface run on the router, consumes the least energy, making it the new best case. Thus, we adjust the application's energy range based on these minimum and maximum values.

B. Energy classes and labels

Once the range is defined, we employ Equal-Width Binning [13] to classify energy consumption into seven classes (A-G), with every bin width calculated by dividing the application energy range by seven. Each scenario is assigned a label based on the corresponding consumption interval. Class **A** contains the most energy-efficient scenarios, while Class **G** includes the least efficient ones. For example, our base application scenarios range from Class A (46.07-95.96 J/s) to G (345.41-395.27 J/s).

TABLE III: Application Deployment Scenarios

Scenario	**Application Module**	**Target Device**
Router_Based	motion_detector, object_detector, object_tracker user_interface	Router Cloud
Router_Router_Proxy	motion_detector, object_detector object_tracker user_interface	Router Proxy Cloud
Router_Router_Cloud	motion_detector, object_detector object_tracker, user_interface	Router Cloud
Router_Proxy_Proxy	motion_detector object_detector, object_tracker user_interface	Router Proxy Cloud
Router_Only	motion_detector object_detector, object_tracker user_interface	Camera Router Cloud
Router_Proxy	motion_detector object_detector object_tracker user_interface	Camera Router Proxy Cloud
Router_Cloud	motion_detector object_detector object_tracker, user_interface	Camera Router Cloud
Router_Cloud_Cloud	motion_detector object_detector, object_tracker, user_interface	Router Cloud
Proxy_Only	motion_detector object_detector, object_tracker user_interface	Camera Proxy Cloud
Proxy_Cloud	motion_detector object_detector object_tracker, user_interface	Camera Proxy Cloud
Cloud_Only	motion_detector object_detector, object_tracker, user_interface	Camera Cloud
Edge_Edge_Router	motion_detector, object_detector object_tracker user_interface	Camera Router Cloud
Edge_Edge_Proxy	motion_detector, object_detector object_tracker user_interface	Camera Proxy Cloud
Edge_Edge_Cloud	motion_detector, object_detector object_tracker, user_interface	Camera Cloud
Edge_Based	motion_detector, object_detector, object_tracker user_interface	Camera Cloud
Proxy_Based	motion_detector, object_detector, object_tracker user_interface	Proxy Cloud
Proxy_Proxy_Cloud	motion_detector, object_detector object_tracker, user_interface	Proxy Cloud
Proxy_Cloud_Cloud	motion_detector object_detector, object_tracker, user_interface	Proxy Cloud
Cloud_Based	motion_detector, object_detector, object_tracker, user_interface	Cloud

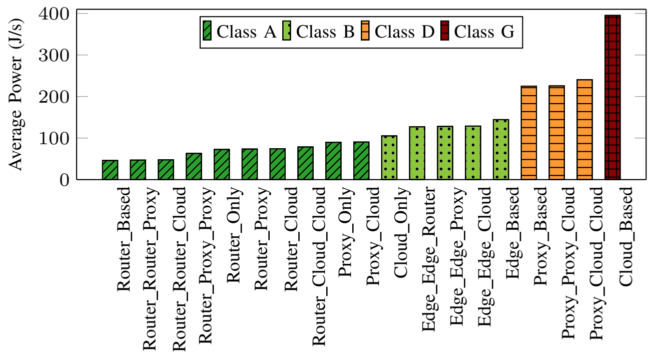

Fig. 3: Labeling scenarios with Equal-Width Binning.

Figure 3 illustrates the labeling scheme for the base application, based on the total energy consumption (computing and networking) for 20,000ms simulation time. Scenarios are sorted by average power and classified using a color-coded approach from green (A) to red (G). The labels facilitate energy-consumption assessment, as users can quickly see how switching from *Proxy-Based* to *Router-Based* reduces consumption, because labels change from class D to A.

C. From Workload to Application

We extend our labeling system from workload-specific to application-specific by incorporating multiple workloads (i.e.,

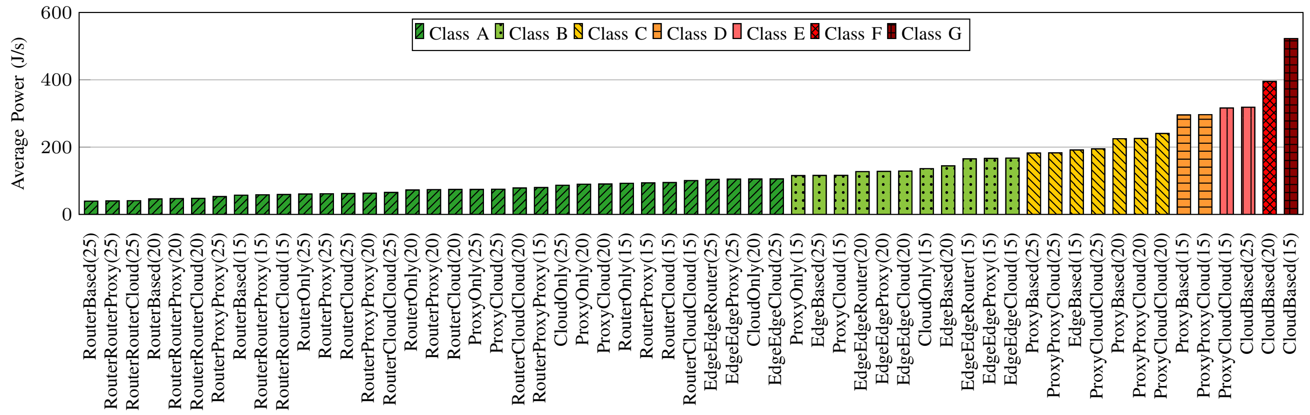

Fig. 4: Integrated Representation of Labeling Energy Consumption of Deployment Scenarios with different Workload Intensities

different application configurations). This expansion broadens the energy range and accounts for workload variation across scenarios, adjusting label boundaries as needed.

We analyze the impact of workload variation by adjusting the camera emission intervals to 15 and 25 milliseconds. Figure 4 shows the updated energy labeling system, based on the extended energy range and revised energy classes for all scenarios and different emission intervals. We observe that increasing the camera emission interval from 15 to 25 ms reduces workload intensity and, in some scenarios (e.g., *Proxy-Based*), may lead to a transition from D to C, i.e., lower energy consumption. Conversely, higher processing workloads increase energy consumption, pushing scenarios into higher energy classes. These observations highlight the effect of application specification on energy consumption and the need for a flexible and comprehensive labeling system.

D. Use and limitations

Energy labels provide a quick grasp of the different energy classes, and incentivize users to aim for a greener class for their application deployment. To allow for action toward energy reduction, we envision a framework that assists the user in selecting the acceptable trade-off - between energy class and QoE - for their deployment. This approach is feasible due to our rapid, simulation-based scheme, where QoE metrics such as latency and delay are also reported for each scenario.

Although we only showed the method is effective for surveillance applications, the labeling approach is applicable to other services, e.g. in online shopping or personalized healthcare, only requiring energy-consumption range calibration for each application(-domain). We currently explore the pros and cons of generalization beyond application-aware labeling, which will accommodate more applications, but may lead to unacceptably coarse energy classes. We further note that hardware-specific variations, although not included here, can also be considered, with the same caveat: range recalibration.

V. Conclusion and Future Directions

In this work, we proposed an energy labeling method for digital services deployed in the computing continuum. Our scheme classifies deployment scenarios into energy classes (A to G) using Equal-Width Binning. This labeling system increases awareness, helping stakeholders assess energy efficiency across different applications and system configurations. Standardizing the approach across diverse digital services can further enhance energy transparency and sustainability.

Beyond energy analysis, this labeling system can support energy-efficient scheduling and optimization strategies. For future work, we aim to extend the system to broader application domains and account for hardware diversity to improve accuracy; we also explore integrating the labeling method into energy-efficient scheduling and optimization strategies.

References

[1] Y. Li *et al.*, "End-to-end energy models for edge cloud-based iot platforms: Application to data stream analysis in iot," *Future Generation Computer Systems*, vol. 87, pp. 667–678, 2018.

[2] E. Kern *et al.*, "Labelling sustainable software products and websites: ideas, approaches, and challenges," *EnviroInfo ICT Sustain.*, pp. 82–91, 2015.

[3] P. Duran *et al.*, "Gaissalabel: A tool for energy labeling of ml models," in *Companion Proc. of the 32nd ACM Int. Conf. on FSE*, 2024, pp. 622–626.

[4] J. Castaño *et al.*, "Exploring the carbon footprint of hugging face's ml models," in *Proc. of 2023 ACM/IEEE Int. Symp. on ESEM*, 2023, pp. 1–12.

[5] R. N. Calheiros *et al.*, "Cloudsim: A toolkit for modeling cloud computing environments," *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23–50, 2011.

[6] H. Wu *et al.*, "An energy framework for ns-3," in *4th Int. Conf. on Simulation Tools and Techniques*, 2012.

[7] H. Gupta *et al.*, "ifogsim: A toolkit for modeling and simulation of resource management in iot, edge, and fog computing," *Softw. Pract. Exp.*, vol. 47, no. 9, pp. 1275–1296, 2017.

[8] S. Baneshi *et al.*, "Estimating energy consumption of applications in the computing continuum with ifogsim," in *Int. Conf. on High Performance Computing*, 2023, pp. 234–249.

[9] ——, "Analyzing per-application energy consumption in a multi-application computing continuum," in *Proc. of 2024 9th Int. Conf. on Fog and Mobile Edge Computing*, 2024, pp. 30–37.

[10] E. Ahvar *et al.*, "Estimating energy consumption of cloud, fog, and edge infrastructures," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 2, pp. 277–288, 2019.

[11] B. Courty *et al.*, "mlco2/codecarbon: v2.4.1," may 2024. [Online]. Available: https://doi.org/10.5281/zenodo.11171501

[12] L. F. W. Anthony *et al.*, "Carbontracker: Tracking and predicting the carbon footprint of training deep learning models," *arXiv preprint arXiv:2007.03051*, 2020.

[13] R. Kerber, "Chimerge: Discretization of numeric attributes," in *Proc. 10th Nat. Conf. on Artificial Intelligence*, 1992, pp. 123–128.

APPENDIX

ARTIFACT DOCUMENTATION

Abstract—**This document provides an overview of the artifacts accompanying our paper. Our research introduces an energy labeling framework for digital services in the computing continuum using an enhanced iFogSim simulation framework. To facilitate reproducibility, we provide our modified iFogSim code, sample workloads, execution scripts, and dataset, all archived in a persistent repository.**

Our submission applies for the **Open Research Objects (ORO)** and **Reusable/Research Objects Reviewed (ROR)** badges for our work on "Empowering Sustainability: Energy Labeling of Digital Services Using Simulation"[1].

The provided artifacts (Zenodo Repository) include:

- **Pre-compiled JAR:** `LabellingDCNS.jar` for direct execution of scenarios.
- **Source Code:** Enhanced iFogSim framework with a provided Java archive (JAR) file for execution.
- **Dependencies:** All required JARs are included in the 'jars/' folder.
- **Datasets:** Energy consumption data used for validation and comparison.
- **Execution Instructions:** A detailed guide on running simulations, reproducing results, and generating energy labels.
- **Python Script:** Scripts for automating simulation execution, parsing results, normalizing energy data, and generating labeled outputs.

A. Software and Data Repository

- **Permanent Repository:** Zenodo Repository
- **GitHub (Development):** https://github.com/saeedehbaneshi/IFogSim
- **License:** Apache 2.0

B. Software Details

- **Programming Language:** Java, Python
- **Simulation Framework:** iFogSim (based on CloudSim)
- **Execution Environment:** Linux/Ubuntu 22.04.5
- **Expected Runtime:** 5 minutes per scenario

C. Dependencies:

- **Java Version:** OpenJDK 17 (required).
- **Python Version:** Python 3.10 (required).
- **Jupyter Notebook:** Required for running automated scenario execution
- **Required Python Packages:**

```
$ pip install numpy pandas matplotlib seaborn
  scipy networkx jupyterlab
```

[1]For convenience, we append the short paper draft for submission #97 at the end of this document.

D. Data Description

- **Dataset Format:** Excel files.
- **Dataset Structure:** Includes total energy consumption per scenario (Computation + Networking) for each three workload intensities (15ms, 20ms, and 25ms).
- **Data for Figure 3:** The required data for the energy labeling of the base application with an interval of 20ms is available in Path: `/Report_Saeedeh/Defining_labels_data_after_fixing_simulator_CCGRID/Final/Rate_20_normalized_total_energy_per_scenario.xlsx`
- **Data for Figure 4:** The file containing normalized energy of all scenarios with different workload intensities (extended labeling) is available in Path: `/Report_Saeedeh/Defining_labels_data_after_fixing_simulator_CCGRID/Final/merged_normalized_energy_with_equal_width_classes_sorted.xlsx`

E. Running a Single Scenario with Pre-compiled JAR

1) Download the Zenodo package and extract it:

```
$ wget https://zenodo.org/record/14969629/
  files/IFogSim-EnergyLabelling-v2.zip
$ unzip IFogSim-EnergyLabelling-v2.zip
$ cd IFogSim-EnergyLabelling-v2
```

2) Run simulation for a single scenario:

```
java -jar LabellingDCNS.jar <Scenario_name>
```

This executes the simulation for the specified scenario. A list of valid scenario names and their definitions can be found in Table 3 of the paper.

F. Running a Single Scenario with Python

We also provide the simulation using the Python notebook `LabellingResultParser.ipynb`. In the first code box A *run_command* is defined to explicitly specify the dependencies required for running the application via the system command line. ***The user must update the Java path and dependency directories in this command based on the main directory of the iFogSim on their machine.***

The following code box contains the *run* function, which utilizes this command to execute the target application ("LabellingDCNS") with the specified scenario and store the output report log in *report_path*. The remainder of the script automates execution for all scenarios, as described in the next section.

For energy labeling, we run the simulation for three different camera emission intervals: 15ms, 20ms, and 25ms. The emission interval is set in line 530 of the file `src/org/fog/test/perfeval/LabellingDCNS.java`, within the add camera function. The default emission interval is 20ms that is used for base application labeling.

G. Labeling Base Application: [Figure 3 of The Paper]

For each emission interval, we execute the simulation for all possible scenarios listed in Table 3 of the paper. To automate this process and organize the output results into a structured CSV file, we provide the Python notebook `LabellingResultParser.ipynb`.

The script automatically classifies energy consumption using an equal-width binning approach (this script also has another labeling approach and plotting delay data, which is for more analysis and is not used in the paper).

Steps to Run All Scenarios

1) **Open the Jupyter Notebook**:

```
$ jupyter notebook LabellingResultParser.ipynb
```

2) **Run the script to**:

- Iterate over `scenarios_list` and call the `run` function for each scenario:

```
for scenario in scenarios_list:
   run("LabellingDCNS", scenario)
```

- Parse energy and delay data from simulation output log for all scenarios using the `parse` function to prepare the data for labeling analysis.
- Apply energy labeling: Scenarios are classified into energy efficiency classes (A-G) based on total energy consumption using an **equal-width binning** approach and generate a labeled figure:

```
plt.title('Normalized Energy per Scenario by
Energy Class (Equal-Width Binning)', fontsize=14)
```

This corresponds to Figure 3 in the paper. This Figure is for emission interval of 20ms.

- **Store parsed energy data** for all simulated scenarios of the selected emission interval in:

```
Report_Saeedeh/Labelling_Results
```

We renamed each output Excel file to include the emission interval in its filename and moved it to the following directory: `Report_Saeedeh/Defining_labels_data_after_fixing_simulator_CCGRID/Final/Rate_20_normalized_total_energy_per_scenario.xlsx`

Repeat the process for other emission intervals. After running the notebook for each emission interval, we obtain the complete energy dataset needed for extended analysis and labeling. It saves the energy results of each workload intensity across all scenarios into a excel file.

H. Extended Energy Labeling: [Figure 4 of The Paper]

To generate the extended **Energy Labeling**, across workload intensities, we merge their results data, sort them and apply energy labeling. For this purpose we provide another Python notebook: `Merging_different_rate_results_labelling.ipynb`.

This Python notebook:

- **Merge Results for Different Intervals:** it loads and merges energy data from three workload intensity (15ms, 20ms, 25ms) into a single DataFrame:

```
rate_15_df = pd.read_excel(
'Rate_15_normalized_energy_per_scenario.xlsx')

rate_20_df = pd.read_excel(
'Rate_20_normalized_total_energy_per_scenario.xlsx')

rate_25_df = pd.read_excel(
'Rate_25_normalized_energy_per_scenario.xlsx')
```

- **Sorting and Normalizing Energy:** The script sorts scenarios based on their normalized energy:

```
merged_df = merged_df.sort_values(by=
'normalized_energy_per_sec').reset_index(drop=True)
```

- **Applying Energy Labeling:** The equal-width binning approach is applied to classify each scenario into energy labels A-G:
- **Generating Figure 4:** The energy class distribution is visualized using a color-coded bar chart titled: `'combined_energy_performance_charts_equal_width_fixed.png'`
- **Saving the Processed Data:** The dataset is saved as:

```
merged_df.to_excel('merged_normalized_energy_with_
equal_width_classes_sorted_fixed.xlsx', index=False)
```

This artifact provides the necessary code, data, and instructions to reproduce our energy labeling framework. By following the provided steps, reviewers and researchers can validate our claims, explore different configurations, and extend our work. We believe this submission supports open, transparent, and reproducible research in sustainable computing.

ACKNOWLEDGMENT

This research was conducted as part of the PhD program at the University of Amsterdam (UvA). It was supported by the Parallel Computing Systems (PCS) group at UvA.