Estimating CO_2 emissions of distributed applications and platforms with SimGrid/Batsim

Gabriella Saraiva[1], Miguel Vasconcelos[2], Sarita Mazzini Bruschi[3], Danilo Carastan-Santos[4], Daniel Cordeiro[1]

[1]Escola de Artes, Ciências e Humanidades — Universidade de São Paulo (EACH–USP) — São Paulo, SP — Brasil

[2]University of Toulouse, CNRS, Toulouse INP, UT3 — Toulouse, France

[3]Instituto de Ciências Matemáticas e de Computação — Universidade de São Paulo (ICMC–USP) — São Carlos, SP — Brasil

[4]University Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG Grenoble — France

gabriella_saraiva@usp.br, miguel-felipe.silva-vasconcelos@irit.fr
sarita@icmc.usp.br, danilo.carastan-dos-santos@univ-grenoble-alpes.fr
daniel.cordeiro@usp.br

Abstract. *This work presents a carbon footprint plugin designed to extend the capabilities of the Batsim simulator by allowing the calculation of CO_2 emissions during simulation runs. The goal is to assess the environmental impact associated with task and resource management strategies in simulated environments. The plugin is developed within SimGrid—the underlying simulation framework of Batsim—and computes carbon emissions based on the simulated platform's energy consumption and carbon intensity factor of the simulated machines. Once implemented, it is integrated into Batsim, ensuring compatibility with existing simulation workflows and enabling researchers to assess the carbon efficiency of their scheduling strategies.*

1. Introduction

The rapid growth of digitalization across various sectors has increasingly demanded computational infrastructure, turning data centers into major electricity consumers. According to the International Energy Agency, these facilities account for approximately 2% of global energy consumption [International Energy Agency (IEA) 2024]. This is mainly due to the intensification of internet data traffic and the expansion of workloads and storage capacity. In the United States, for example, the energy consumption of data centers increased from 91 billion kWh in 2013 to almost double by 2020, highlighting the impact of this evolution [Abbas et al. 2021].

This increase has raised significant environmental concerns. In response to the energy and climate crisis, countries like China have established sustainability goals, including the commitment to peak carbon emissions by 2030 and achieve carbon neutrality by 2060 [Zhang et al. 2025]. Meanwhile, the industry has been striving to improve the energy efficiency of data centers. Despite increasing demand, global energy consumption by data centers increased only 6% between 2010 and 2018, as a result of efforts such as

technological modernization, improvements in cooling systems, and advances in workload management algorithms [Masanet et al. 2020].

However, the environmental impact of distributed computing infrastructures goes beyond energy consumption. The carbon footprint of data centers is determined not only by the amount of energy used but also by the carbon intensity of the electricity consumed, which varies significantly by region and over time. For example, Google's 2025 Environmental Report shows that while some of its data centers operate with almost 100% renewable energy, others, such as those in Japan, rely on grids with only 4% renewable share [Google 2025]. Microsoft's sustainability reports also highlight that, as the share of renewables increases, indirect emissions from manufacturing IT equipment become more relevant [Microsoft 2025].

Various strategies can be applied to reduce energy consumption in these environments. At the infrastructure level, adopting more efficient components, such as optimized CPUs or specialized accelerators (e.g., GPGPUs), already represents progress. Intelligent task and resource management play a key role at the software level. Among the most effective measures are to reduce task execution times, use available resources efficiently, and even turn off idle machines [Poquet 2017]. These decisions are guided mainly by scheduling algorithms that analyze usage patterns to strategically determine where, when, and how each task should be executed, to balance energy efficiency and performance.

Simulators are essential tools in this context, enabling researchers to evaluate the performance and energy consumption of scheduling strategies in controlled and reproducible environments before deployment. While several simulation frameworks model energy consumption, few offer native support for quantifying the resulting carbon footprint, which depends on the carbon intensity of the local energy grid. This gap makes it difficult to assess the complete environmental impact of computational workloads, as a strategy that is energy-efficient in one location may not be carbon-efficient in another.

The main objective of this work is to address this gap by proposing a carbon footprint plugin for the Batsim simulator, built upon SimGrid. The paper presents its design, implementation, and validation. The structure is as follows: Section 2 reviews related work, Section 3 describes the implementation, and Section 4 reports the validation results.

2. Background and Related Work

The optimization of energy consumption and the pursuit of sustainability in large-scale computational systems constitute an active and increasingly important area of research. One approach to address these challenges is simulation, which allows for the evaluation of resource management strategies in controlled and reproducible environments before their costly practical implementation.

Recent literature argues that an effective sustainability analysis must go beyond mere energy consumption, incorporating carbon emissions as a central metric. Comprehensive works, such as the theses of Orgerie (2020) [Orgerie 2020] and Vasconcelos (2023) [Vasconcelos et al. 2023], investigate strategies to make distributed systems more environmentally friendly, reinforcing the critical importance of considering both the energy source and the variation in carbon footprint by location.

In this context, several simulators have been proposed to model energy consumption in data centers. Tools such as CloudSim [Calheiros et al. 2011] and GreenCloud [Kliazovich et al. 2012] are widely used to study the efficiency of resource allocation algorithms. SimGrid [Casanova et al. 2014], in turn, is a well-established simulator in the scientific community, offering support for various distributed environments and already incorporating models for energy consumption and renewable energy generation. However, despite its capabilities, SimGrid lacks native support for calculating CO_2 emissions based on the carbon intensity of the electricity mix used by the infrastructure—a gap this work aims to fill.

This work introduces a carbon footprint plugin developed in SimGrid and integrated in Batsim [Dutot et al. 2016], with the objective of enabling a more comprehensive analysis of the environmental impact of different management strategies in distributed applications and platforms. Our plugin allows estimating CO_2 emissions associated with energy consumption during simulated executions in Batsim, considering variables such as the carbon intensity of the energy source used. Our plugin may be useful for researchers and practitioners to assess not only the energy efficiency of their solutions but also their environmental impacts, contributing to the development of more sustainable technologies in the context of high-performance computing. The source code for the plugin, as well as the documentation and tutorials, is openly available[1]

3. Carbon Footprint plugin

The carbon footprint plugin[2] for SimGrid was designed to enable simulation and monitoring of CO_2 emissions associated with the energy consumption of computational hosts in distributed systems. The plugin provides a flexible and extensible framework that integrates seamlessly with SimGrid's energy plugin, allowing users to quantify the environmental impact of their simulated workloads based on customizable carbon emission intensities.

3.1. Mathematical Model

An incremental model based on the energy consumption of each host was implemented to estimate the carbon footprint associated with computational workloads. This work applies the model exclusively to the machines (hosts) in the simulated environment. The energy consumption of network equipment and cooling systems is not considered at this stage and may be addressed in future work.

The model accounts for both the static component (energy consumed when the machine is idle) and the dynamic component (additional energy consumed when the machine executes tasks). Carbon emissions are periodically updated during the simulation, using the host's instantaneous power consumption and the environment's carbon intensity.

Let t_0 be the time of the last update and t_1 the current simulation time. At each update, the instantaneous power consumption of the host, P, is measured at time t_0 and is

[1]Plugin Home Page: `https://github.com/saraiva03/carbon-footprint-simgrid-batsim`

[2]Carbon footprint plugin repository: `https://github.com/saraiva03/simgrid/tree/carbon-footprint-calc`

assumed to remain constant during the interval $[t_0, t_1]$. Therefore, the energy consumed is given by:

$$E_{\text{step}} = P \cdot (t_1 - t_0) \tag{1}$$

$$E_{\text{step}}^{\text{kWh}} = \frac{E_{\text{step}}}{3.6 \times 10^6} \tag{2}$$

$$C_{\text{step}} = E_{\text{step}}^{\text{kWh}} \cdot CI_{\text{step}} \tag{3}$$

$$C_{\text{total}}(t_1) = C_{\text{total}}(t_0) + C_{\text{step}} \tag{4}$$

Where:

- E_{step}: energy consumed in the interval (in joules);
- $E_{\text{step}}^{\text{kWh}}$: energy in kilowatt-hours;
- P: instantaneous power consumption of the host at the time t_0 (in watts)
- CI_{step}: carbon intensity of the environment in the interval (in $\mathrm{g/kW\,h}$);
- C_{step}: carbon emitted in the interval (in grams);
- C_{total}: total carbon footprint up to time t_1.

This model is implemented in the `HostCarbonFootprint::update()` function, ensuring that the estimated carbon emissions reflect the host's dynamic behavior during the simulation. The carbon intensity CI can be configured according to the energy profile of the simulated environment.

3.2. Overview of the Plugin Functionality

The carbon footprint plugin was developed to extend SimGrid's host model, enabling the quantification of carbon emissions associated with energy consumption during simulation. An extension is attached to each `Host` object, responsible for monitoring and updating the accumulated emissions according to the host's activity to achieve this.

The plugin operates by intercepting host key simulation events, such as initialization (creation), shutdown (destruction), state transitions (power on/off), and changes in execution profile (task execution or updates to carbon intensity). Whenever one of these events occurs, the plugin updates the host's carbon footprint, considering the energy consumed since the last update and the current carbon intensity at that moment.

The carbon intensity, defined as the amount of CO_2 emitted per unit of energy consumed ($\mathrm{g/kW\,h}$), can be individually configured for each host via properties in the platform XML file, as illustrated in Listing 1. Furthermore, the plugin allows for the dynamic adjustment of this parameter during simulation, reflecting possible changes in the energy profile of the simulated environment.

SimGrid was modified to include callbacks on the main host events to enable this functionality, ensuring that the carbon footprint extension is notified and can perform the necessary updates. As a result, the original SimGrid model is enhanced to consider

not only energy consumption but also the environmental impact of simulated executions, providing detailed metrics on carbon emissions over time. The overall workflow of the plugin is illustrated in Figure 1.

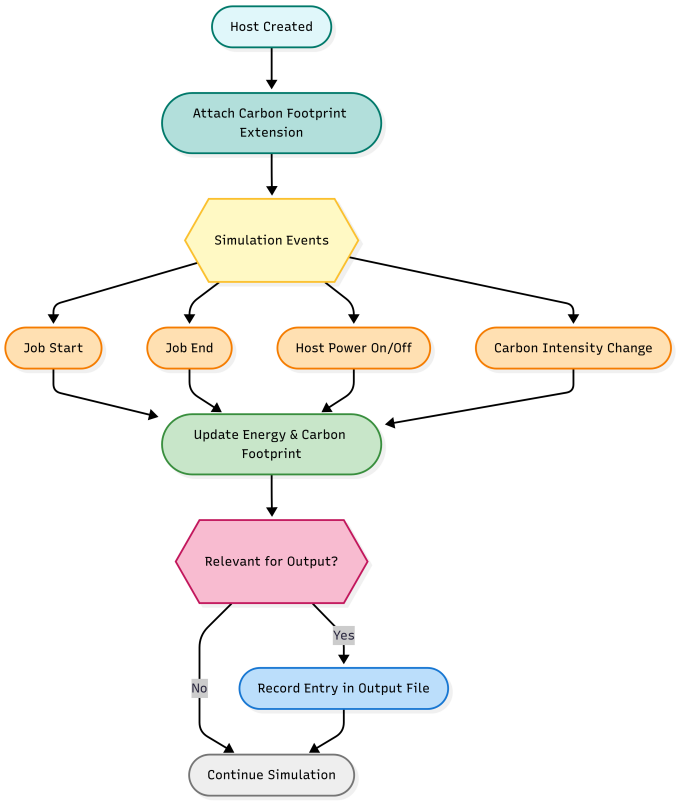

Figure 1. Flowchart illustrating the main steps of the carbon footprint plugin and its integration with Batsim.

3.2.1. Validation

The plugin was validated using different test case scenarios to explore the different states in SimGrid regarding the energy consumption of machines: i) when the machine is off; ii) when the machine is idle (the machine is powered on but not running any computations); and iii) when the machine performs computations, using one or more of its CPU cores.

Two main scenarios were considered for the validation regarding the carbon emissions. First, we use a static value for the carbon emissions of the electricity grid, which can represent countries for which we only have the annual value of the carbon intensity. The second scenario considers values of carbon intensity that change over time, for example, to represent countries with intermittent renewable sources in their mix.

Figure 2 illustrates the variability over a single day for the different carbon intensities in the different countries we considered: i) the USA, which shows a high carbon intensity because it uses high-carbon sources such as coal; ii) France, which shows a low carbon intensity given the high presence of nuclear power; and iii) Brazil, which also shows a low carbon intensity, given the presence of renewable sources such as hydroelectric power.

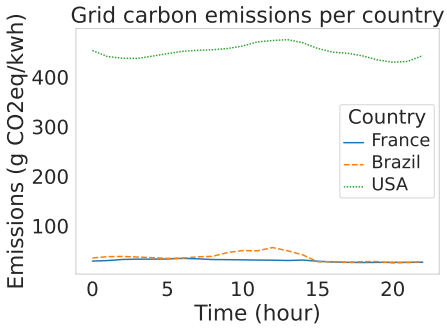

Figure 2. Grid emissions values for the different electricity mixes considered.

The validation step is important to ensure that the plugin remains working in the future when new code is incorporated into the SimGrid framework, such as including new functionalities or fixes to other problems in the code. Further details about the test scenarios and the inputs used can be found in the plugin's GitHub repository.

3.3. Integration with Batsim

The carbon footprint plugin was developed to monitor carbon emissions during simulations in Batsim. To use it, add the "`-C`" or "`--carbon-footprint`" option when running Batsim, along with the platform and workload files. This activates the plugin, which then begins tracking each host's energy consumption and carbon emissions in the simulation.

During execution, Batsim monitors key events such as the start and end of jobs, as well as changes in the power state of the hosts. Whenever one of these events occurs, the updated carbon emission value is retrieved for each host and recorded in an output file. This file contains information such as the event timestamp, the energy consumed, the total carbon emissions, the type of event, and the average emissions since the last recorded entry. The source code for the version of Batsim with the integrated carbon footprint plugin is available at `https://github.com/saraiva03/batsim/tree/carbon-footprint-calc`.

Once integrated into Batsim, an output file is generated that logs the environmental impact upon key events, such as job start and completion, and host power state changes. Each entry in this output file includes a timestamp, the cumulative energy and carbon emissions, the event type, and the average emission since the last record. This structured output allows for detailed temporal analysis of both energy consumption and carbon emissions throughout the simulation.

4. Evaluation and Comparison

The experiments aimed to compare the energy consumption and carbon emissions data between two distinct approaches: the actual execution of the inference benchmarks using machine learning models and the simulation of these same applications using Batsim. The implementation developed for the experiments can be accessed at `https://github.com/saraiva03/carbon-plugin-tests`.

The real executions involved 100 inferences for each model: ResNet18, BERT large (fine-tuned on the SQuAD dataset), and Deep Learning Recommendation Model

(DLRM). These tests were performed on a Dell G15 laptop with an 11th Gen Intel Core i5-11400H processor, featuring six physical cores and a base frequency of 2.70 GHz. The CodeCarbon[3] tool was used to monitor energy consumption, hardware usage, and environmental impact. The simulated versions of these applications were executed in Batsim, using a configuration model designed to represent the same hardware environment as the real executions. For both approaches—real and simulated—10 runs were performed. The following sections detail the experimental setup and analyze the results.

4.1. Experimental Setup

The experiments included several important considerations and assumptions. First, the `carbon_intensity` value was calculated as the ratio between carbon emissions (`emissions`) and energy consumed (`energy_consumed`), both extracted from the output file generated by CodeCarbon. This calculation yielded a value of $98.348\,\mathrm{g/kW\,h}$, consistent with the default value defined in the official CodeCarbon documentation. This value reflects the global average carbon intensity of electricity, as presented in the CodeCarbon repository, while considering country-specific variations. In this study, Brazil was used as the reference country.

To ensure accuracy in estimating energy consumption, the `cpu_energy` metric was used instead of `energy_consumed`, since the real benchmarks were executed exclusively on the CPU of the designated machine. CodeCarbon enables a breakdown of energy usage by hardware components (such as CPU and GPU); therefore, only CPU-specific consumption was considered. Consequently, the Batsim simulation environment was configured to reflect the exact CPU specifications, as it was the only component involved in performing the inference tasks.

Throughout real executions, CodeCarbon recorded CPU power consumption ranging from $30\,\mathrm{W}$ to $40\,\mathrm{W}$, with occasional peaks reaching $50\,\mathrm{W}$. This consumption range corresponds to the values reported in the official processor documentation[4] and was used to define the different CPU power states in Batsim.

The number of floating point operations (FLOP) per inference was estimated using the THOP (PyTorch-OpCounter)[5] library, which is built on PyTorch. This library calculates the theoretical computational cost of a model's architecture based on input dimensions and model structure, independent of the actual input data. Therefore, the number of FLOP per inference remained constant across all runs, resulting in minimal variation in energy consumption and carbon emissions recorded during the 10 Batsim simulations.

For comparative analysis between real and simulated executions, standard statistical metrics were used, including the R^2 score, Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE).

Machine Description and Simulator Parameters

[3]CodeCarbon repository: `https://github.com/mlco2/codecarbon`

[4]Intel's official specification for the processor used (Intel® Core™ i5-11400H): `https://www.intel.com.br/content/www/br/pt/products/sku/213805/intel-core-i511400h-processor-12m-cache-up-to-4-50-ghz/specifications.html`

[5]THOP (PyTorch-OpCounter) repository: `https://github.com/Lyken17/pytorch-OpCounter`

In the Batsim simulator, the Intel Core i5-11400H processor was represented as a host with the following parameters:

```
<!-- Intel_i5_11400H -->
<host id="Intel_i5_11400H" speed="4.5Gf" pstate="0" core="6">
  <prop id="wattage_per_state" value="10:25:40" />
  <prop id="wattage_off" value="1.0" />
  <prop id="carbon_intensity" value="98.348" />
</host>
```

Listing 1. XML Configuration of Host used in the experiments

Host ID (`id`): Intel_i5_11400H

Speed (`speed`): Represents the processing capacity per core (in GigaFLOP per second).

Number of cores (`core`): 6

Power states (`wattage_per_state`): Indicates energy consumption (in watts) at different usage levels:

- **Idle:** When the processor is idle or under minimal activity, it consumes little energy.
- **Epsilon:** Represents an intermediate load, where some cores are in use or there is moderate activity.
- **AllCores:** Maximum usage state, with all cores active simultaneously, resulting in the highest energy consumption.

Power consumption in off state (`wattage_off`): 1.0 W

Carbon intensity (`carbon_intensity`): 98.348 g/kW h, the same value used in the real experiment, as specified in the CodeCarbon documentation.

The speed parameter was estimated individually for each benchmark by dividing the total FLOP required by the average execution time, resulting in a performance rate in GFLOP per second per core, the value shown in Listing 1 (4.5Gf) is a representative example corresponding to the BERT Large benchmark. Meanwhile, the values assigned to `wattage_per_state` were defined based on measurements of CPU energy consumption, obtained using the Intel Power Gadget software.

4.2. Results and Analysis

This section presents the results of the experiments, comparing the energy consumption and carbon emissions data obtained from real executions with the values generated by the Batsim simulation. The results, extracted from the `collected_results.csv` dataset, are analyzed using the previously mentioned metrics. The discussion is further supported by a visual analysis of the boxplots, illustrating the distribution and variability of the data for each benchmark.

ResNet18: The ResNet18 benchmark involved a total of 182 GFLOP across 100 inferences. With a total execution time of approximately 2.5 seconds, the CPU's effective processing rate was 72.8 GFLOP/s, or 12 GFLOP/s per core ($72.8 \div 6$), which was used as the speed value for the Batsim simulation. The simulation results for this model showed an R^2 score of -0.580, an RMSE of 0.001, and a MAPE of 23.11%.

BERT Large (fine-tuned for SQuAD): The BERT Large benchmark involved a total of 1,250 GFLOP across 100 inferences, completed in about 46 seconds. The CPU demonstrated an effective processing rate of 27 GFLOP/s, or approximately 4.5 GFLOP/s per core $(27 \div 6)$, the speed value used in the simulator. The simulation of this model produced an R^2 score of -0.146, an RMSE of 0.004, and a notably lower MAPE of 8.32%.

Deep Learning Recommendation Model (DLRM): The DLRM benchmark, the lightest model tested, required 13 GFLOP for 100 inferences, completed in approximately 0.45 seconds. The effective processing rate was 28.8 GFLOP/s, or 4.8 GFLOP/s per core $(28.8 \div 6)$. The simulation metrics for DLRM were an R^2 score of -0.019, an RMSE of 0.000, and a MAPE of 16.13%, an intermediate error value.

Analyzing the results obtained in the experiments allows us to assess the accuracy of the simulation performed in Batsim compared to the real executions monitored by CodeCarbon. Overall, the simulation was able to estimate the average energy consumption and carbon emissions reasonably close to the real values, especially for heavier models such as BERT Large. However, there are important differences in the variation of the results, reflected in the statistical metrics calculated.

For the ResNet18 model, the mean absolute percentage error (MAPE) was approximately 23.11%, indicating a significant deviation between the real and simulated values. However, it is important to note that the power consumption of processors of the same model can vary up to 20% due to variability in manufacturing [Chasapis et al. 2019]. A MAPE of 23.11% is therefore close to an acceptable range. Furthermore, the coefficient of determination (R^2) showed a negative value (-0.580), revealing that the simulation could not reproduce the fluctuations observed in real executions. This behavior can be explained by the fact that the model is relatively lightweight and runs quickly, making it more susceptible to consumption variations caused by background processes and dynamic frequency adjustments in the processor. In contrast, the Batsim simulation maintained stable values that did not reflect these occasional consumption peaks.

In the case of BERT Large, the results were more consistent. The MAPE was only 8.32%, demonstrating a reasonable proximity between the simulated and experimentally measured values. Although it remained negative (-0.146), indicating a low correlation with the variations in different runs, the average consumption and emissions estimated by Batsim were quite accurate. This suggests that for heavier applications with longer execution times, the simulation model represents the energy behavior of the system in a more reliable way.

Finally, for the DLRM model, the MAPE recorded was 16.13%, an intermediate error compared to the other two benchmarks. Like ResNet18, R^2 was practically zero (-0.019), again evidencing the difficulty in capturing variations in fast executions. This result is consistent considering that execution times less than one second are more sensitive to measurement inaccuracies and fluctuations in the real system, which reduces the correlation with the simulated values.

The boxplots in Figure 3 and Figure 4 visually illustrate the patterns observed in the quantitative metrics (MAPE and R^2). It is observed that, for all models, the simulated values show low variability, with boxes that are nearly collapsed and no presence of outliers. This reflects the deterministic nature of the Batsim simulation, which, by assuming

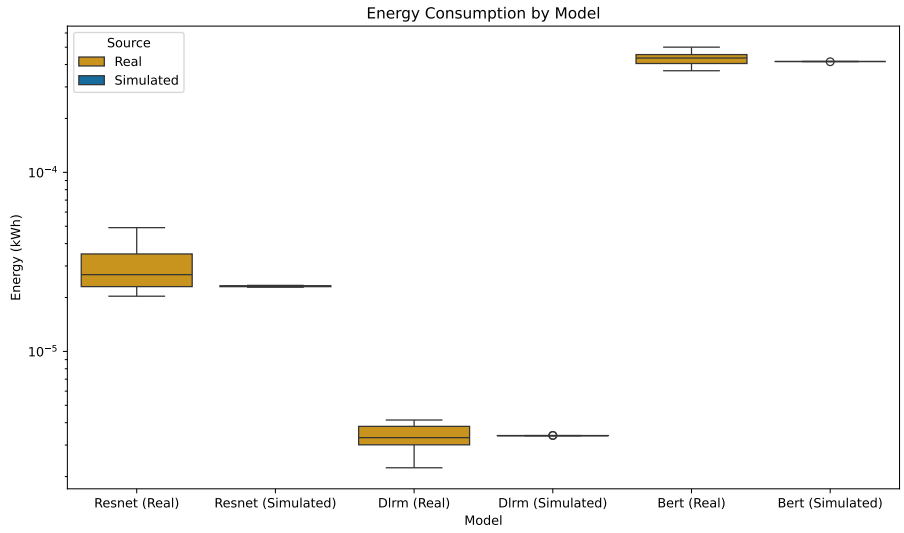

Figure 3. Energy consumption (kWh) by model, comparing real and simulated data. Log scale in the y axis.

constant consumption per power state, generates very stable results across executions.

In the case of ResNet18, there is a marked difference between the dispersion of real and simulated data, both in terms of energy consumption and carbon emissions. The real data present a wider interquartile range, indicating greater variation between executions, which aligns with its short runtime and susceptibility to system noise (such as frequency fluctuations and background processes). The median of the simulated values is also visibly lower than the real median, suggesting an underestimation of the energy load by the simulation model.

For DLRM, a behavior similar to that of ResNet18 is observed. Although overall consumption is even lower, the dispersion in real data remains evident, and the gap between the real and simulated medians is slightly smaller than in ResNet18, which is consistent with the intermediate mean absolute percentage error previously reported (MAPE $\approx 16\%$). The presence of outliers in the real data further reinforces the influence of external factors in short executions.

In the case of BERT Large, the boxplots reveal a closer match between real and simulated values. Both the medians and interquartile ranges are similar, especially in the energy consumption graph. Although there is still some variation in the real data, it is significantly smaller in relative terms, given the larger magnitude of the consumption. These visual results support the low MAPE (8.32%) and reinforce that the simulation model is more reliable for heavy workloads and longer durations, in which the impact of system noise is mitigated.

The results indicate that the simulation-based approach is suitable for estimating average energy consumption and carbon emissions, especially for more intensive and longer-running workloads such as BERT Large. However, there are limitations for lightweight and fast models, since small variations in the real environment significantly affect measurements, while the simulation keeps values constant. More refined adjustments to the power states (`wattage_per_state`) or the introduction of dynamic load

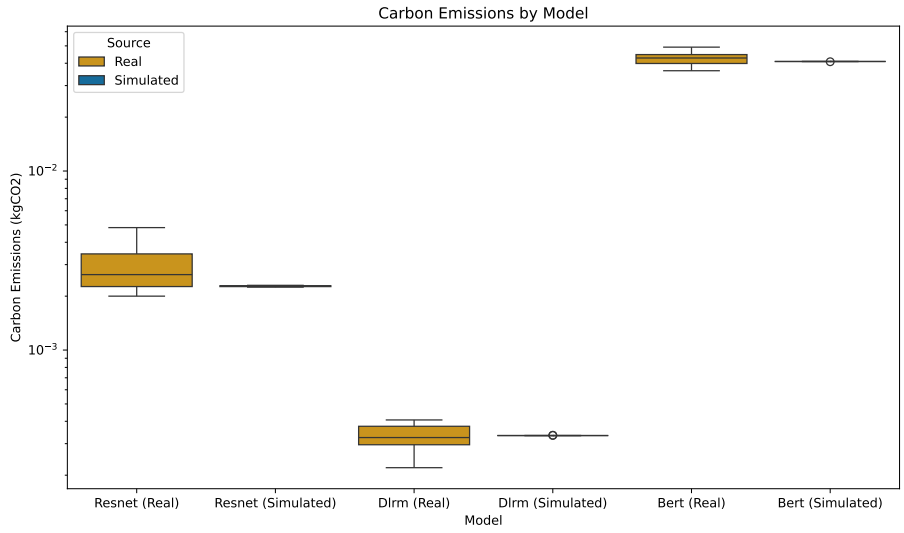

Figure 4. Carbon emissions (kg CO_2) by model, comparing real and simulated data. Log scale in the y axis.

models could improve the accuracy of the simulation, bringing it even closer to real execution.

5. Conclusion

This work introduced a carbon footprint plugin developed in SimGrid and integrated into Batsim, enabling the simulation and monitoring of CO_2 emissions based on the energy consumption of computing hosts. By allowing per-host configuration of carbon intensity and logging emissions during key scheduling events, the plugin extends Batsim's capabilities to support more environmentally aware simulations.

A validation study was conducted by comparing simulated executions of three machine learning workloads against experimental results on a single machine. The findings show the plugin's ability to reproduce average energy and carbon emission profiles with reasonable accuracy, particularly for heavier, longer-running workloads. The study also highlighted limitations in capturing the variability of short-running tasks, where system noise has a greater impact on experimental measurements.

These results establish the plugin as a valuable tool for researchers to evaluate the environmental impact of scheduling strategies in a controlled environment. Future work includes refining the plugin to improve simulation accuracy for lighter workloads and using Grid'5000 testbed to evaluate distributed large-scale scenarios. Furthermore, efforts will be directed toward making this plugin an official SimGrid extension, fostering long-term maintenance, and encouraging broader adoption by the research community.

6. Acknowledgments

This project was partially funded by FAPESP (procs. 19/26702-8, 23/00811-0, 24/09487-4, and 24/23163-7), by the Science for Development Center (CCD) 'Carbon Neutral Cities' (FAPESP 24/01115-0), and by the CNRS International Research Center (IRC) 'Transitions'.

References

Abbas, A., Huzayyin, A., Mouneer, T., and Nada, S. (2021). Effect of data center servers' power density on the decision of using in-row cooling or perimeter cooling. *Alexandria Engineering Journal*, 60(4):3855–3867.

Calheiros, R. N., Ranjan, R., Beloglazov, A., Rose, C. A. F. D., and Buyya, R. (2011). Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50.

Casanova, H., Giersch, A., Legrand, A., Quinson, M., and Suter, F. (2014). Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917.

Chasapis, D., Moretó, M., Schulz, M., Rountree, B., Valero, M., and Casas, M. (2019). Power efficient job scheduling by predicting the impact of processor manufacturing variability. In *Proceedings of the ACM International Conference on Supercomputing*, pages 296–307.

Dutot, P.-F., Mercier, M., Poquet, M., and Richard, O. (2016). Batsim: a Realistic Language-Independent Resources and Jobs Management Systems Simulator. In *20th Workshop on Job Scheduling Strategies for Parallel Processing*.

Google (2025). Google environmental report 2025.

International Energy Agency (IEA) (2024). Analysis and forecast to 2026. Technical report, IEA, France.

Kliazovich, D., Bouvry, P., and Khan, S. U. (2012). Greencloud: A packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62(3):1263–1283.

Masanet, E., Shehabi, A., Lei, N., Smith, S., and Koomey, J. (2020). Recalibrating global data center energy-use estimates. *Science*, 367(6481):984–986.

Microsoft (2025). Microsoft sustainability report.

Orgerie, A.-C. (2020). *From Understanding to Greening the Energy Consumption of Distributed Systems*. Habilitation à diriger des recherches (HDR), Ecole Normale Supérieure de Rennes.

Poquet, M. (2017). *Simulation approach for resource management*. Data structures and algorithms, Université Grenoble Alpes.

Vasconcelos, M., Cordeiro, D., da Costa, G., Dufossé, F., Nicod, J.-M., and Rehn-Sonigo, V. (2023). Optimal sizing of a globally distributed low carbon cloud federation. In *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 203–215.

Zhang, X., Zhong, J., Zhang, X., Zhang, D., Miao, C., Wang, D., and Guo, L. (2025). China can achieve carbon neutrality in line with the Paris agreement's 2°C target: Navigating global emissions scenarios, warming levels, and extreme event projections. *Engineering*, 44:207–214.