

Assessing Operational Carbon Footprint for Task-based Applications using Simulation

Otho José Sirtoli Marcondes, Rayan Raddatz, Lucas Mello Schnorr

¹ Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

ojsmarcondes@inf.ufrgs.br, rrdmatos@inf.ufrgs.br, schnorr@inf.ufrgs.br

Abstract. *HPC energy consumption translates into carbon emissions that vary by location and execution time, but assessing these emissions through real executions is costly. We propose a simulation-based method to predict the operational carbon footprint of task-based applications. A per-task CPU and GPU energy model, built from hardware measurements, is applied to StarPU-SimGrid traces, and the predicted energy is converted into the carbon footprint using regional data. We used Cholesky factorization as a case study on four partitions in Brazil and France, with simulations predicting energy with maximum errors from 6% (CPU) to 28% (GPU), and the same workload emitting up to $\times 3.7$ more carbon depending on the partition and region.*

Resumo. *O consumo de energia em HPC gera emissões de carbono que variam com o local e o horário, mas aferi-las com execuções reais é custoso. Propomos um método baseado em simulação para prever a pegada de carbono operacional de aplicações baseadas em tarefas. Um modelo de energia por tarefa para CPU e GPU, construído com base em medições de hardware, é aplicado a rastros StarPU-SimGrid, e a energia prevista é convertida para a pegada de carbono utilizando dados regionais. Nós utilizamos a fatoração de Cholesky em quatro partições no Brasil e na França, com as simulações prevendo a energia simulada com erros máximos de 6% (CPU) a 28% (GPU), e uma mesma carga de trabalho emitindo até $\times 3,7$ mais carbono em função da partição e região.*

1. Introduction

High-Performance Computing (HPC) is the backbone of modern scientific research across disciplines such as physics, engineering, and numerical analysis [Schulz et al. 2023] [Dongarra and Keyes 2024] [Portegies Zwart 2020]. As the complexity and scale of computational workloads continue to grow, so does the energy demand of HPC systems [Strohmaier et al. b] [Strohmaier et al. a] [Patterson et al. 2022] [Suarez et al. 2025]. This growing energy consumption translates directly into operational carbon emissions, making the environmental impact of HPC an increasingly critical concern [Gnibga et al. 2023] [Li et al. 2023] [Nguyen et al. 2022]. Task-based runtime systems offer an effective path to high performance in heterogeneous architectures by dynamically scheduling fine-grained tasks across CPUs and GPUs. Linear algebra kernels such as LU and Cholesky factorization are emblematic of this class of workloads [Xia et al. 2023], and are at the core of many real-world applications. Reducing the carbon footprint of these applications requires knowing their operational carbon intensity.

This value varies by location and time of day, depending on the local electricity grid’s energy mix. Identifying the best region or time window for running a workload is therefore a promising strategy for reducing emissions.

Performing this analysis through real executions is costly: it requires reserving full clusters, instrumenting software and hardware, and repeating runs under different configurations. Simulation offers an attractive alternative, enabling rapid assessment of application behavior, including energy and makespan, without the need for actual hardware [Heinrich 2019] [Saraiva et al. 2025]. However, simulating energy consumption in task-based runtimes presents a specific challenge: tasks execute sequentially within the simulator, whereas real hardware counters measure energy at socket or device granularity across concurrent tasks. Bridging this gap requires per-task energy consumption models. Previous work addresses exactly this need, proposing a method to attribute energy to individual tasks using an overdetermined linear system that links task execution intervals to energy measurements obtained from hardware counters [Risse et al. 2025]. Building on this foundation, simulation has been used to estimate the carbon footprint of CPU workloads [Saraiva et al. 2025]. However, to the best of our knowledge, no prior work has combined per-task energy models for both CPU and GPU with simulation-based carbon footprint estimation for task-based applications, nor has any study focused on South American regions [Kamatar et al. 2025] [Li et al. 2023].

This paper closes that gap with a methodology to predict the operational carbon footprint of task-based HPC applications using simulation, which we evaluate using the Cholesky factorization of Chameleon/StarPU as a case study. The methodology reuses an existing per-task energy solver [Risse et al. 2025], the StarPU-SimGrid simulator [Stanisic et al. 2014], and carbon intensity data from Electricity Maps [Electricity Maps 2026]. Our contributions are: 1/ a per-task energy model for CPU and GPU that predicts the power, duration, and energy of each task type at block sizes that were not measured, built from the solver’s output and StarPU’s performance models; 2/ a workflow that applies this model to StarPU-SimGrid traces to predict the whole-run energy of an execution through simulation, validated against hardware measurements on four HPC partitions; 3/ a conversion of the predicted energy into operational carbon footprint that reprices the same simulated run for any region and time; and 4/ an analysis of how the carbon footprint of the same workload varies between the South Brazilian and French electricity grids, across hours of the day and months of the year. Section 2 presents the background on task-based runtime systems and simulation frameworks and details our three-phase workflow. Section 3 presents and discusses the experimental results. Section 4 concludes with remarks and future work.

2. Methods & Materials

We present the background, the three-phase workflow implementation and validation.

2.1. Background on task-based and simulation frameworks

Task-based runtimes, such as StarPU [Augonnet et al. 2012], OmpSs [Duran et al. 2011], and PaRSEC [Bosilca et al. 2013], execute parallel programs on heterogeneous machines. The runtime scheduler receives a set of dependent tasks and schedules them during execution based on available computing resources. Chameleon [Agullo et al. 2010], the

linear algebra solverstack library we use in this work, utilizes this method through the StarPU runtime, which offers significant adaptability, especially in GPU-equipped nodes. Simulation frameworks enable understanding application behavior before actually making a real execution. Many tools, such as SimGrid [Casanova et al. 2025], BigSim [Zheng et al. 2004], SST [Janssen et al. 2010], and CODES [Mubarak et al. 2016], focus on simulating distributed applications at scale. SimGrid is a widely recognized tool used in scientific research to simulate distributed systems. It can quickly simulate many nodes, including network contention, making it particularly useful for studies that require analysis of large-scale network interactions. StarPU and Chameleon also provide validated support [Stanisic et al. 2014] [Stanisic et al. 2015] for the SimGrid framework.

2.2. Assessing Operational Carbon Footprint for Task-based Applications

Figure 1 depicts the three phases of the workflow to assess the carbon footprint of the executed applications. Phase 1 consists of the actual creation of the model through the execution of real experiments with energy measurements obtained from hardware counters. We build our model upon these collected data to create our energy model. Phase 2 uses our energy model to predict energy consumption based on the simulation traces. Phase 3 estimates the operational carbon footprint of a real run of that execution. The three phases are detailed as follows, with blue boxes representing the general inputs, orange boxes representing the general outputs and green ellipses highlighting the steps that are contributions of this work.

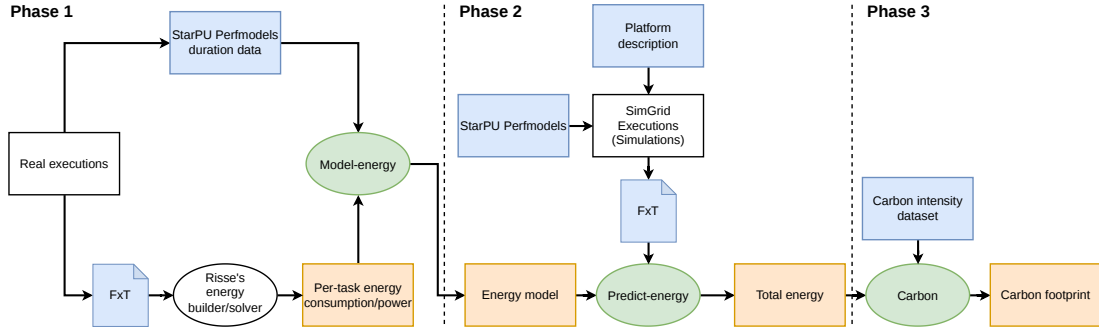


Figure 1. Carbon footprint prediction workflow.

Phase 1: Energy Model Creation. We build the energy model after executing a real batch of experiments that vary the block sizes (nb), with N repetitions, measuring the energy with hardware counters, and running Risse’s energy solver [Risse et al. 2025] to extract per-task energy values from the application. The model is built specifically for the machine that served as its input, as each machine has its own hardware configuration, but it is indexed by task type and processing unit rather than by application. Hence, another application on the same machine built from already-modeled kernels (e.g., other Chameleon routines based on `dgemm`, `dtrsm`, `dsyrk`, and `dpotrf`) can reuse it. A new kernel requires real executions that exercise it at several block sizes, while a new machine requires repeating this phase. For a given machine, the energy solver [Risse et al. 2025] outputs, for a single experiment, one power value $\hat{P}^{S_k}$ per state/task type S_k , including compute states such as `dgemm` and auxiliary states such as *Overhead* or *Sleeping* (extracted from an FxT (Fast User/Kernel Tracing) trace file of that execution). We build

upon this value and the StarPU history-based performance model to predict how power and duration, and consequently energy consumption, would behave at a different, unmeasured nb . This generalization across nb is one of the contributions of the work, and we detail it in five steps as follows.

1. *Grouping experiments.* We group experiments by the $(S_k, \text{domain_source})$ tuple, where domain_source can be either CPU (RAPL) or GPU (NVML) readings. We need to differentiate CPU and GPU because a task’s power is a property of one processing unit. For example, a `dgemm` executed on a CPU and a `dgemm` executed on a GPU have different power draw signatures even though they are the same task type. For each group g , let nb_i , $\hat{P}_i^{S_k}$, $\hat{D}_i^{S_k}$ denote the block size, per-experiment solved power (from the `energy-solver` output [Risse et al. 2025]), and per-experiment measured average task duration of the i -th experiment contributing to g . Before fitting, a solve $(nb_i, \hat{P}_i^{S_k})$ is dropped from g if $\hat{P}_i^{S_k} \notin [0, \text{TDP}_g]$, where TDP_g is the appropriate device TDP (CPU or GPU bound). As Risse’s energy solver can produce negative or large power estimates we discard any such implausible per-experiment results entirely before the model fit, rather than let one degenerate experiment’s solve distort every extrapolation built from it.

2. *Power as a function of block size.* For each group g with at least two distinct measured block sizes, we fit ordinary least squares of solved power against raw block size:

$$P_g(nb) = \beta_0 + \beta_1 nb \quad (1)$$

β_0 (W) is the fitted power at $nb = 0$ and β_1 (W per unit nb) is power’s sensitivity to block size. Where a group has fewer than two distinct block sizes, Eq. (1) has zero residual degrees of freedom and we fall back to the constant model $\beta_0 = \bar{P}^{S_k}$, $\beta_1 = 0$.

3. *Duration as a function of block size.* Unlike power, a compute state’s duration scales polynomially with nb (e.g. $O(nb^3)$ for `dgemm`-like kernels), so a straight-line fit on the raw scale underfits badly at large nb . We fit OLS on log-transformed axes instead:

$$\log D_g(nb) = \gamma_0 + \gamma_1 \log nb \iff D_g(nb) = C_g \cdot nb^{\gamma_1}, \quad C_g = e^{\gamma_0} \quad (2)$$

where γ_1 is the fitted exponent (expected ≈ 3 for `dgemm`); C_g (ms) is the fitted duration at $nb = 1$. As in Eq. (1), groups with fewer than two block sizes fall back to a constant, $C_g = \bar{D}^{S_k}$, $\gamma_1 = 0$. As StarPU’s performance model presents accurate execution time values, we use the duration from StarPU’s own history, only falling back to our own measured durations when the performance model is unavailable. The durations are resolved in the following priority order (where `device_type` is CPU or GPU): 1/ a real entry in StarPU’s performance model, per-task, per-host history for exactly $(S_k, \text{device_type}, nb)$; 2/ absent that, Eq. (2) fit to StarPU’s performance model aggregated history for $(S_k, \text{device_type})$ and evaluated at nb ; 3/ if StarPU has no performance model for $(S_k, \text{device_type})$, Eq. (2) fit to our own measured $\hat{D}_i^{S_k}$ as a last resort.

4. *Predicted single-task energy at an unmeasured nb .* Following the power equation $E = P \times t$, where E is the energy (J), P is the power (W), and t is the time (s), predicted energy for one instance of state/task S_k at block size nb is

$$\hat{E}^{S_k}(nb) = P_g(nb) \times \frac{D(nb)}{1000} \quad (3)$$

with $D(nb)$ resolved via the three-tier lookup above and the 1000 converting ms to s.

5. *Predicted whole-run energy.* A run executes many instances of each state/task across its execution. Rather than sum Eq. (3) instance-by-instance, which would additionally require a separate model of instance *count* at unmeasured nb , we fit, per group g , the aggregate time already summed over every instance of S_k within an experiment:

$$\log T_g(nb) = \delta_0 + \delta_1 \log nb \iff T_g(nb) = S_g \cdot nb^{\delta_1} \quad (4)$$

fit only for groups with at least three experiments with positive total time. This is applied uniformly to compute states and to the auxiliary states (*Idle*, *Sleeping*, *Overhead*, *Scheduling*, *FetchingInput*, *Executing*, *Progressing*). Predicted whole-run energy is then given by

$$\hat{E}_{\text{run}}(nb) = \sum_{S_k} P_g(nb) \times T_g(nb) \quad (5)$$

summed over every state/task type present in the group’s history, group power at nb times the group’s aggregate predicted time at nb .

Phase 2: Predict Energy with Simulation. To run simulations with StarPU-SimGrid, it is imperative to have performance models generated from real executions and the platform description of the cluster. By setting the simulation execution to gather traces, it is possible to obtain important task information, such as the duration of each task and the types of tasks executed during that run. This trace information will later be necessary to predict the energy consumption via this simulation run. After simulating with the traces collection activated, it is necessary to transform the FxT trace file into readable parquet files. Using these new readable trace files, we can assess the task types and their respective durations. Using our model, we can predict energy consumption based on the task type, the block size (workload) of that task, and the type of worker (CPU or GPU). The total energy consumption of the run is then obtained by applying our model to the tasks.

Phase 3: Estimate Carbon Footprint. With the whole-run energy previously predicted with simulation, this phase is responsible for converting this energy into the actual operational carbon footprint of the execution. In order to do so, we rely on carbon intensity data from Electricity Maps [Electricity Maps 2026], which, for a given region and time point, indicates how much CO₂ is emitted per unit of electricity generated. Given the region and the timestamp of the execution, we retrieve the carbon intensity $I(\text{region}, t)$ (in gCO₂eq/kWh) closest to that moment, and convert the predicted energy E (in Joules) into carbon following

$$\text{gCO}_2 = \frac{E}{3,600,000} \times I(\text{region}, t) \quad (6)$$

with 3,600,000 J/kWh converting E to kWh. We consult Electricity Maps’ direct (combustion-only) signal, rather than its lifecycle basis, as our goal is to assess the operational footprint, so we do not include emissions from plant construction and fuel supply chain. By applying Eq. (6) to the energy predicted for each task, and aggregating this result, we obtain the carbon footprint on different granularities: per task, per task type and per run, this last one being the total operational carbon footprint of the simulated execution. As the same predicted energy can be repriced against any region and timestamp without the need to re-run the simulation, we can analyze how the footprint of a given

execution changes across regions and across time of the day and year which is essential to establish relocation or rescheduling mechanisms to mitigate its environmental impact.

2.3. Implementation and Validation Procedures

We employ two partitions of the PCAD cluster (`pc:cei` and `pc:poti`) at INF/UFRGS and two from the Grid’5000 testbed (`g5k:chuc` and `g5k:grouille`) in the experiments. Table 1 specifies the hardware of one node of each partition. Although some partitions feature more than one GPU per node, we used only a single GPU in our experiments. From a software perspective, we use StarPU with commit `919c107f` from the `energy-counters` branch (which enabled us to implement energy measurements via hardware counters) and Chameleon version 1.4.0. The complete software stack is kept stable by the Guix package manager, which maintains a manifest file that contains all software configurations. The source code and scripts used in this work are publicly available at <https://github.com/othomarcondes/starpu-simgrid-energy>.

Table 1. Hardware specification of the used partitions.

Name	CPU	RAM	GPU
<code>pc:cei</code>	2 x Intel(R) Xeon(R) Silver 4116 2.10 GHz, 24 cores/CPU	96 GB	
<code>pc:poti</code>	Intel(R) Core(TM) i7-14700KF 3.4 GHz, 20 cores/CPU	96 GB	NVIDIA GeForce RTX 4070 12 GB
<code>g5k:chuc</code>	AMD EPYC 7513 32 cores/CPU	512 GiB	4 x Nvidia A100 (40 GiB)
<code>g5k:grouille</code>	2 x AMD EPYC 7452 32 cores/CPU	128 GiB	2 x Nvidia A100 (40 GiB)

Design of Experiments for Phase 1: Energy Model Creation. The first phase of our experiments consists of running multiple repetitions and collecting energy measurements for the selected applications with different block sizes to obtain sufficient data to build our model. For the `pc:cei` and `pc:poti` partitions, we performed a Cholesky factorization (`dpotrf`) with a fixed input double precision square matrix of order 30720. For the `g5k:chuc`, `g5k:grouille` partitions, we performed the same Cholesky factorization but with a size of 92160 due to their higher processing speed. For the `pc:cei` partition (CPU only), the block sizes are as follows: 320, 384, 480, 512, 640, 768, 960, 1024. For the `pc:poti` partition, which possesses a user-grade GPU, we also added the block sizes: 1280, 1536 and 1920. For the `g5k:chuc` and `g5k:grouille` partitions, which run the larger matrix, we use the block sizes 960, 1024, 1152, 1280, 1440, 1536, 1920, 2048 and 2304. This differentiation is due to GPUs’ high computational capability, which enables them to work better with larger block sizes than CPUs. For all partitions, 10 repetitions were performed for each configuration. In this phase, we also validate whether Risse’s energy solver [Risse et al. 2025] correctly estimates the energy consumption of tasks. To verify it, we sum the energy of the application tasks based on its output and compare it with the actual hardware counter measurements.

Design of Experiments for Phase 2 and 3: Carbon Footprint Estimation. Our second phase uses the models created in Phase 1 to predict the carbon footprint of executions via simulations with SimGrid. In this step, we perform the same design of experiments as in Phase 1, but with only one repetition, as the simulations always generate the same trace. With the application trace, we can extract the tasks’ durations and use our model to predict the total energy consumption of a real run via simulation. With this information, we convert the energy values into the actual carbon footprint in Phase 3, which consults Electricity Maps’ datasets to verify the carbon intensity based on the time and location of execution. This enables us to analyze how different clusters (across different regions and execution times) can mitigate the environmental impact of the applications. For this analysis, we use SimGrid-MPI, using 4 nodes per partition in order to compare their energy consumption and carbon footprint.

3. Experimental Results & Discussion

In this section we evaluate our approach. We first validate our energy model and simulations, then present the carbon footprint case studies and the cost of our workflow.

3.1. Validation of Energy Model and Simulation Results

We perform two types of validation before analyzing the application’s carbon footprint. First, we revalidate that the per-task energy information retrieved from Risse’s energy solver yields consistent results with the total energy consumption measured by the hardware counters. We conduct this comparison using the Mean Absolute Percentage Error (MAPE), which is well-suited for our context because it expresses prediction error as a relative percentage, making it interpretable and comparable across partitions with different absolute energy scales. The second validation consists of applying our energy model to simulations to predict the energy consumption of a run and comparing the results with actual energy measurements from a real execution. This validation demonstrates the accuracy of our energy prediction model and workflow methodology, enabling us to assess a run’s energy consumption with a reduced computational cost.

Validation of Energy Models. We execute the application, varying the block size with 10 repetitions, measuring energy through hardware counters, and building the application’s energy models using the energy solver. Afterward, we compare the measured hardware energy consumption with the solver’s output to verify that the solver provides accurate solutions for the runs. Figure 2 presents the MAPE of the energy solver output compared to the actual raw hardware measurements. In this and the next two figures, the Y-axis represents the MAPE, the X-axis represents the block size of the executions, each partition has a facet, and the Y-axis is limited at 30% to better visualize the MAPE. The energy consumption calculated by the energy solver presents a MAPE of less than 1% for the executions on the `pc:cei` partition (CPU only). As for the `pc:poti` partition, which has a GPU, the prediction was less accurate, yielding a MAPE of $\approx 4\%$, consistent with the results reported in Risse’s work. This deterioration in results when using a GPU stems from the inaccuracy of the NVML energy measurements, which can report drastically wrong power readings at times and also provide greater granularity than the CPU ones. The `g5k:chuc` and `g5k:grouille` both present a MAPE of $\approx 5\%$ or lower for all block sizes, which are aligned with the results of the `pc:poti` partition. This can be explained as the smaller block sizes have longer run executions (as the workloads of tasks

are too small for the GPU) and a great part of the energy is actually compounded by the CPU in the *Overhead* and *Sleeping* states and the GPU in the *Idle* state compared to larger block sizes. As the GPU energy consumption during the total run increases, so does the MAPE, since NVML energy measurements are more unreliable than RAPL measurements. Figure 3 presents the MAPE when using our model output against real executions and comparing it to the actual raw hardware measurements. As we can see, the `pc:cei` partition presents a MAPE of at most $\approx 4\%$, and the GPU partitions (`pc:poti`, `g5k:chuc`, and `g5k:grouille`) present a MAPE of at most $\approx 12\%$ for the worst cases. This means that our model can predict the application’s energy consumption on these machines with relative accuracy. An alternative to enhance the model for these partitions would be to start with larger block sizes and problem sizes.

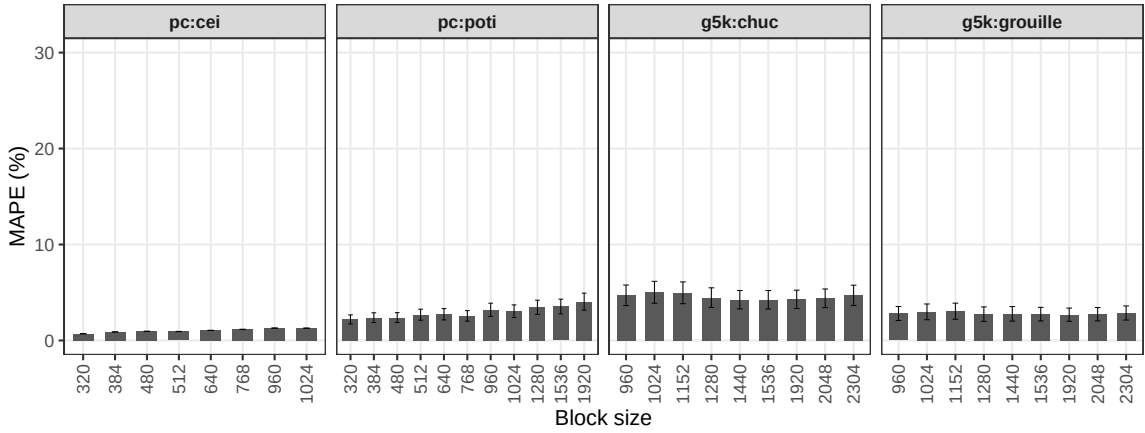


Figure 2. MAPE by cluster and counter domain of the energy-solver. Error bars show the standard deviation across the 10 repetitions.

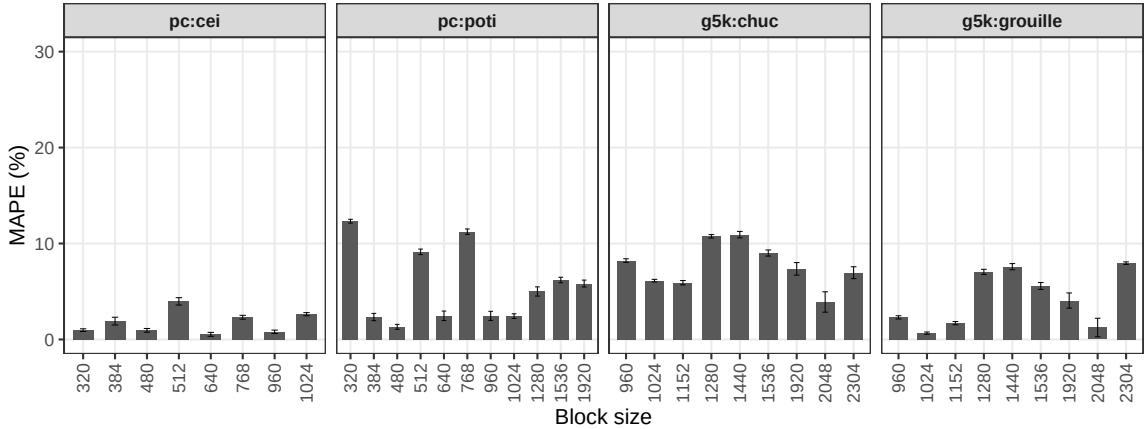


Figure 3. MAPE by cluster and counter domain of our energy model. Error bars show the standard deviation across the 10 repetitions.

Validation of Simulations carried out with SimGrid. We now validate our model when using simulations through the StarPU-SimGrid framework. Figure 4 presents the MAPE of our simulations when using our model output compared to the actual raw hardware measurements. As we can see, the `pc:cei` partition presents a MAPE of at most $\approx 6\%$. The `pc:poti` partition presents a MAPE of at most $\approx 15\%$ when comparing our model’s simulations with real-execution energy measurements. The `g5k:chuc` and

g5k:grouille partitions yielded worse results, with max MAPE values of $\approx 23\%$ and $\approx 28\%$, respectively. We hypothesize that this degradation comes from the idle time between tasks, as their GPU hardware executes tasks faster than the other two partitions, a pattern that is usually smoothed out in the SimGrid traces.

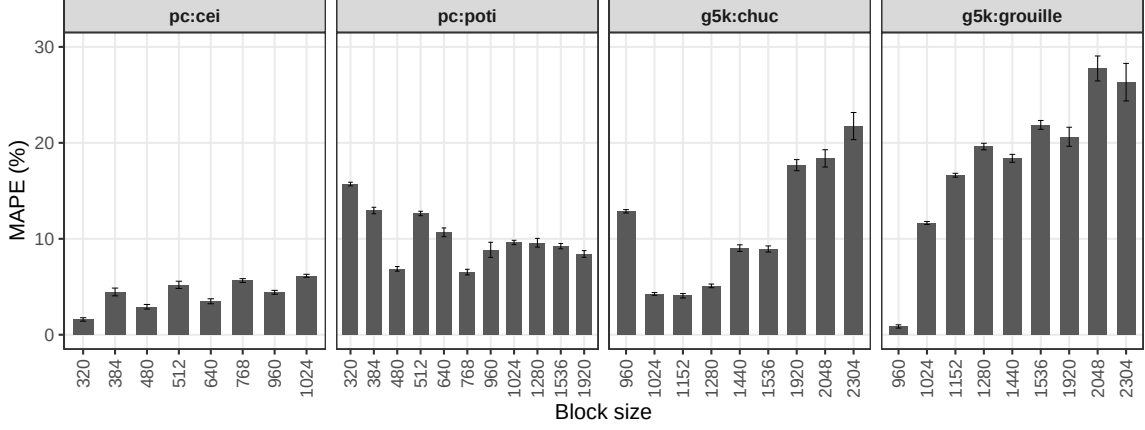


Figure 4. MAPE by cluster and counter domain of the simulations. Error bars show the standard deviation across the 10 repetitions.

3.2. Carbon Footprint Estimation Case-studies

In this section, we first delve into the carbon intensity and the specific characteristics of the energy sources in the zones of the selected partitions where we conducted our experiments. We use carbon intensity data and related information for this analysis from Electricity Maps [Electricity Maps 2026], using 2025 datasets. Through this analysis, we aim to understand the overall behavior of the carbon footprint across different zones and times of day/year, which we can use to inform our choice of where and when our application should run to mitigate the overall carbon footprint. The partitions from pc (pc:cei and pc:poti) are located at the South Brazilian zone, while the g5k partitions (g5k:chuc and g5k:grouille) are located at the France zone.

Carbon footprint characterization. In Figure 5, we can observe, in the left plot, the carbon intensity throughout the year by zone. The X-axis represents the carbon intensity ($\text{gCO}_2\text{eq/kWh}$), and the Y-axis represents the number of occurrences for each carbon intensity range of the regional zones where we conducted our experiments. As we can see, the French region has a lower carbon intensity throughout the year compared to the Brazilian zone. The right plot explains why this happens. In this second plot, the Y-axis represents the share of total electricity generation (%), and the X-axis represents the zones. As we can observe, all regions have a high share of renewable energy (nuclear, hydro, solar, wind). However, the share of Unknown sources is what really drives the carbon intensity of the South Brazilian region up. Electricity Maps states that these Unknown sources are assumed to be derived from thermal sources (coal, gas, and oil) and have a high emission factor. In Figure 6, we can observe the variations of carbon intensity by the months of the year and also by the hours of the day. In the left plot, the X-axis represents the months of the year, and the Y-axis represents the carbon intensity ($\text{gCO}_2\text{eq/kWh}$). The colors represent the zones. As we can see, the South Brazilian region presents higher carbon intensity throughout almost the entire year than the French zone. In the right plot,

the Y-axis represents the carbon intensity ($\text{gCO}_2\text{eq/kWh}$), and the X-axis represents the hour of the day in UTC, with zones identified by different colors. For the South Brazilian region, the first hours of the day (4-10 hours) present a peak in its carbon intensity.

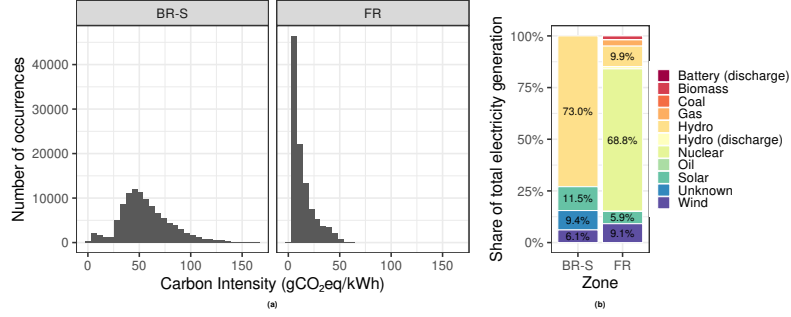


Figure 5. Carbon intensity and sources of the electricity for each zone: (a) distribution of the 2025 carbon intensity values; (b) share of total electricity generation by source.

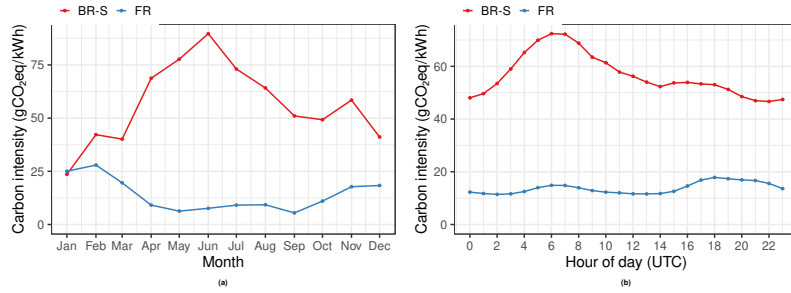


Figure 6. Carbon intensity by month (a) and by hour of the day (b) for each zone.

SimGrid-MPI simulations. In Figure 7, we present two plots, the first showing the predicted energy consumption for each partition and the second showing the carbon footprint of each partition. In the left plot, we can observe total energy consumption on the Y-axis and block size on the X-axis; the partitions are differentiated by color. As we can see, older hardware, such as the `pc:cei`, is less energy-efficient than the other (more recent) partitions, which aligns with the work of [Li et al. 2023]. We used the most recent carbon intensity data we had for the analysis in the second plot (27th July, 2026 at 23:00), which reported carbon intensities of 52 and 117 $\text{gCO}_2\text{eq/kWh}$ for the FR and BR-S zones, respectively. In the right plot, we can observe the carbon footprint on the Y-axis and the block size on the X-axis; each partition is differentiated by the color of its bar. As we can see, although both `pc:poti` and `g5k:chuc` reported similar energy consumption, the difference in the carbon intensity of their regions more than doubles the carbon footprint of the `pc:poti` compared to the `g5k:chuc`. Also, although `g5k:grouille` is second in energy consumption, it reported a lower carbon footprint than every other partition except `g5k:chuc`, showing that cluster location definitely matters for carbon mitigation.

For a more practical use case of our method, and to analyze the carbon footprint of the different partitions over one week, we selected a time span across both regions, from January 11th to January 18th of 2025. We simulated all partitions with a block size of 1024. As we can see in Figure 8, both regions show low carbon intensity on the

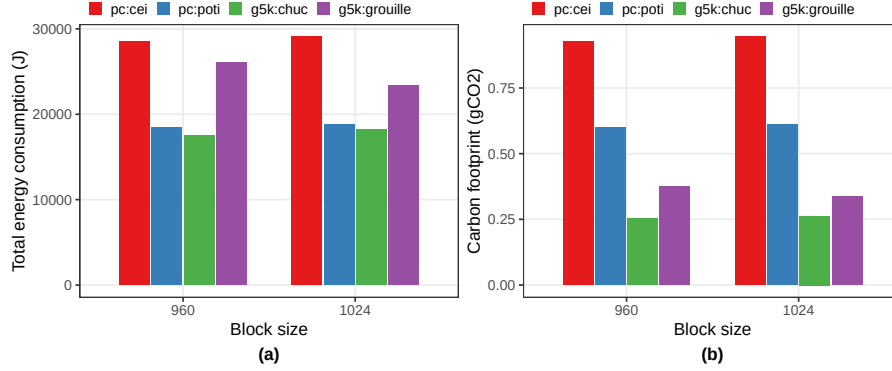


Figure 7. Energy and carbon footprint of simulations per partition and block size: (a) predicted total energy consumption; (b) resulting carbon footprint.

first day (January 11th), then increase until a pattern seems to develop for the rest of the week (starting on January 13th). It is possible to observe that the first half of the day usually shows a more pronounced peak in carbon intensity in both regions, whilst the South Brazilian region shows a decay and stabilized carbon intensity in the second half of the day, and the French region shows a new increase. Therefore, having recognized a pattern for this week of the carbon intensity in both regions, we could mitigate the overall carbon footprint of an application using the clusters from the South Brazilian region during the second half of the day (UTC), whilst in the France region, the first hours of the day and the noon period should be preferred.

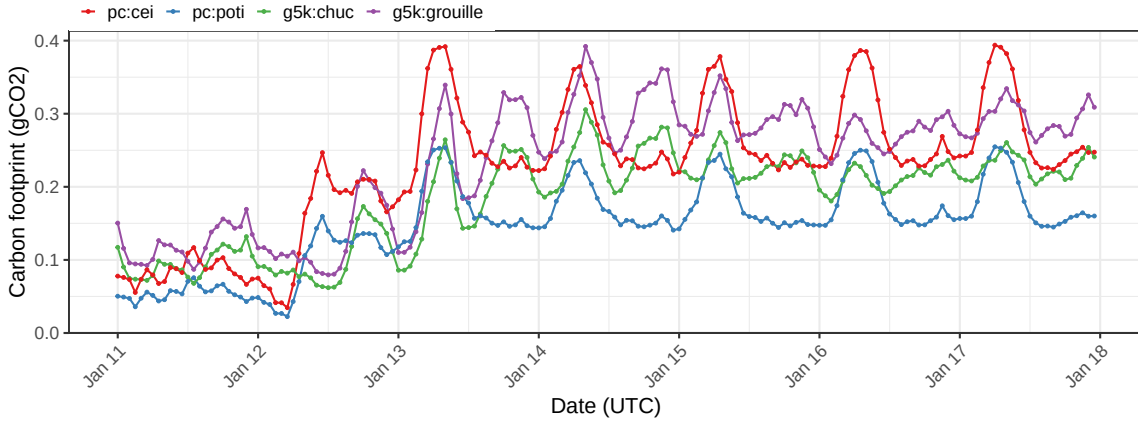


Figure 8. Carbon footprint of simulations for a selected week, per partition.

3.3. Execution Time of the Workflow

Finally, we perform an evaluation of our workflow cost in order to make carbon footprint predictions through simulations using our methodology. In Figure 9, the left plot shows the time for the full workflow on each partition (with all three phases), whereas the right plot shows the time spent on Phases 2 and 3 only for a finer-grain look. The X-axis represents the partitions and the Y-axis represents the time (in seconds) for both plots. In the left plot, we can observe that Phase 1 steps dominate the total time of our workflow, with the real energy-measured executions being responsible for most of this time, followed by

Risse’s energy-solver [Risse et al. 2025]. In the right plot, we can see that the SimGrid simulations often dominate the time span of the second and third phases of our workflow, followed by the Predict-energy step. This makes sense, as the SimGrid runs need to generate the trace files and the Predict-energy step needs to read those files in order to extract the necessary task information, while the Energy-to-carbon step is only a simple conversion. It is important to remark that Phase 1 (the most costly one) needs to be done only a single time in order to create our model, while Phases 2 and 3 can be repeated countless times using the saved model from the previous phase.

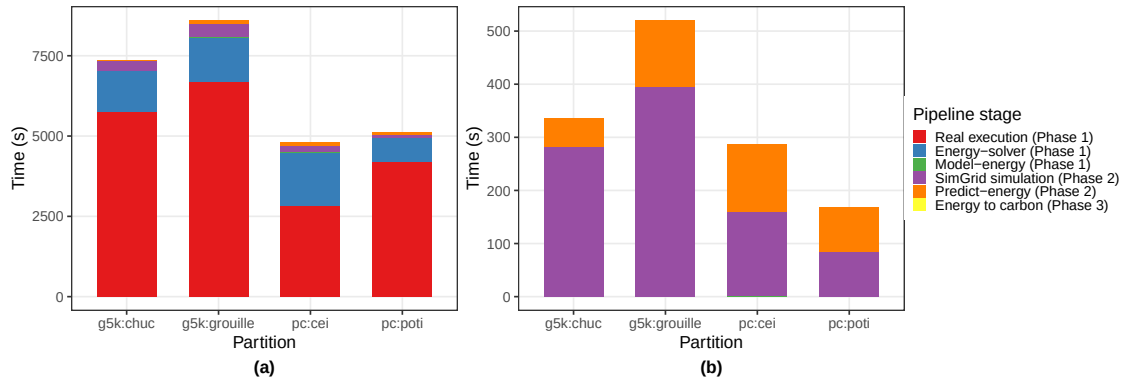


Figure 9. Execution time of each step of our workflow: (a) full workflow, with all three phases; (b) Phases 2 and 3 only.

4. Conclusion

Assessing the carbon footprint of applications enable us to mitigate the environmental impact related to the use of HPC systems. Our main contribution is a method to predict the carbon footprint of task-based applications through simulations. With the predicted energy assessed from a simulation with our model, we can quantify the carbon footprint of an execution, which enables us to analyze where and when we could run an application to mitigate the carbon footprint of our experiment. By verifying that our methodology can predict, with relative accuracy, the energy consumption and carbon footprint of clusters with both CPUs and GPUs, we expect to enable other researchers to quickly assess the carbon footprint of their runs using simple simulations, without the need to allocate real hardware for their experiments. As future work, we expect to enhance our model and prediction steps using other applications and also use the forecast carbon intensities provided by Electricity Maps, aiming to provide a tool which can inform the optimal cluster and execution time to mitigate the environmental impact of a run in the near future.

Acknowledgments. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Finance Code 001. This work was also partially funded by Petrobras under Agreement no. 2020/00182-5 e 2018/00263-5 with UFRGS. We also acknowledge the Brazilian National Council for Scientific Technological Development (CNPq) for their financial scholarship support (process number 132750/2025-2). Experiments were carried out on the Grid’5000 testbed (<https://www.grid5000.fr>) and on the PCAD infrastructure (<http://pcad.inf.ufrgs.br>) at INF/UFRGS.

References

- Agullo, E., Augonnet, C., Dongarra, J., Ltaief, H., Namyst, R., Thibault, S., and Tomov, S. (2010). Faster, Cheaper, Better – a Hybridization Methodology to Develop Linear Algebra Software for GPUs. In *GPU Computing Gems*, volume 2. Morgan Kaufmann.
- Augonnet, C., Aumage, O., Furmento, N., Namyst, R., and Thibault, S. (2012). Starpu-mpi: Task programming over clusters of machines enhanced with accelerators. In *European MPI Users' Group Meeting*, pages 298–299. Springer.
- Bosilca, G., Bouteiller, A., Danalis, A., Faverge, M., Hérault, T., and Dongarra, J. J. (2013). Parsec: Exploiting heterogeneity to enhance scalability. *Computing in Science & Engineering*, 15(6):36–45.
- Casanova, H., Giersch, A., Legrand, A., Quinson, M., and Suter, F. (2025). Lowering entry barriers to developing custom simulators of distributed applications and platforms with SimGrid. *Parallel Computing*, 123:103–125.
- Dongarra, J. and Keyes, D. (2024). The co-evolution of computational physics and high-performance computing. *Nature Reviews Physics*, 6(10):621–627.
- Duran, A., Ayguadé, E., Badia, R. M., Labarta, J., Martinell, L., Martorell, X., and Planas, J. (2011). Ompss: a proposal for programming heterogeneous multi-core architectures. *Parallel processing letters*, 21.02.
- Electricity Maps (2026). Reduce carbon emissions with actionable electricity data.
- Gnibga, W. E., Blavette, A., and Orgerie, A.-C. (2023). Latency, energy and carbon aware collaborative resource allocation with consolidation and qos degradation strategies in edge computing. In *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 2630–2639. IEEE.
- Heinrich, F. (2019). *Modeling, prediction and optimization of energy consumption of MPI applications using SimGrid*. PhD thesis, Université Grenoble Alpes.
- Janssen, C. L., Adalsteinsson, H., Cranford, S., Kenny, J. P., Pinar, A., Evensky, D. A., and Mayo, J. (2010). A simulator for large-scale parallel computer architectures. *International Journal of Distributed Systems and Technologies (IJDST)*, 1(2):57–73.
- Kamatar, A., Gonthier, M., Hayot-Sasson, V., Bauer, A., Copik, M., Castro Fernandez, R., Hoefer, T., Chard, K., and Foster, I. (2025). Core hours and carbon credits: Incentivizing sustainability in hpc. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 870–887.
- Li, B., Basu Roy, R., Wang, D., Samsi, S., Gadepally, V., and Tiwari, D. (2023). Toward sustainable hpc: Carbon footprint estimation and environmental implications of hpc systems. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 1–15.
- Mubarak, M., Carothers, C. D., Ross, R. B., and Carns, P. (2016). Enabling parallel simulation of large-scale hpc network systems. *IEEE Transactions on Parallel and Distributed Systems*, 28(1):87–100.
- Nguyen, T., MacLean, C., Siracusa, M., Doerfler, D., Wright, N. J., and Williams, S. (2022). Fpga-based hpc accelerators: An evaluation on performance and energy efficiency. *Concurrency and Computation: Practice and Experience*, 34(20):e6570.

- Patterson, D., Gonzalez, J., Hölzle, U., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D. R., Texier, M., and Dean, J. (2022). The carbon footprint of machine learning training will plateau, then shrink. *Computer*, 55(7):18–28.
- Portegies Zwart, S. (2020). The ecological impact of high-performance computing in astrophysics. *Nature Astronomy*, 4(9):819–822.
- Risse, J., Guermouche, A., and Trahay, F. (2025). Fine-grain energy consumption modeling of hpc task-based programs. In *2025 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–12. IEEE.
- Saraiva, G., Vasconcelos, M., Bruschi, S., Carastan-Santos, D., and Cordeiro, D. (2025). Estimating co2 emissions of distributed applications and platforms with `simgrid_batsim`. In *Anais do XXVI Simpósio em Sistemas Computacionais de Alto Desempenho*, pages 109–120, Porto Alegre, RS, Brasil. SBC.
- Schulz, M., Ruefenacht, M., Kranzlmüller, D., and Schulz, L. B. (2023). Accelerating hpc with quantum computing: It is a software challenge too. *Computing in Science & Engineering*, 24(4):60–64.
- Stanisic, L., Agullo, E., Buttari, A., Guermouche, A., Legrand, A., Lopez, F., and Videau, B. (2015). Fast and accurate simulation of multithreaded sparse linear algebra solvers. In *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*, pages 481–490. IEEE.
- Stanisic, L., Thibault, S., Legrand, A., Videau, B., and Méhaut, J.-F. (2014). Modeling and simulation of a dynamic task-based runtime system for heterogeneous multi-core architectures. In *European Conference on Parallel Processing*, pages 50–62. Springer.
- Strohmaier, E., Meuer, H. W., Dongarra, J., and Simon, H. D. Green500. <https://www.top500.org/lists/green500/>. accessed November 10,2025.
- Strohmaier, E., Meuer, H. W., Dongarra, J., and Simon, H. D. Top500. <https://www.top500.org/lists/top500/>. accessed November 10,2025.
- Suarez, E., Amaya, J., Frank, M., Freyermuth, O., Girone, M., Kostrzewa, B., and Pfalzner, S. (2025). Energy efficiency trends in hpc: what high-energy and astrophysicists need to know. *Frontiers in Physics*, 13:1542474.
- Xia, Y., Jiang, P., Agrawal, G., and Ramnath, R. (2023). End-to-end lu factorization of large matrices on gpus. In *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, PPOPP ’23, page 288–300, New York, NY, USA. Association for Computing Machinery.
- Zheng, G., Kakulapati, G., and Kalé, L. V. (2004). Bigsim: A parallel simulator for performance prediction of extremely large parallel machines. In *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, page 78. IEEE.