Core Hours and Carbon Credits: Incentivizing Sustainability in HPC

Alok Kamatar
University of Chicago
Chicago, USA
alokvk2@uchicago.edu

Maxime Gonthier
University of Chicago
Chicago, USA
mgonthier@uchicago.edu

Valerie Hayot-Sasson
University of Chicago
Chicago, USA
valerie.hayot-sasson@etsmtl.ca

André Bauer
Illinois Institute of Technology
Chicago, USA
abauer7@iit.edu

Marcin Copik
ETH Zürich
Zurich, Switzerland
marcin.copik@inf.ethz.ch

Raul Castro Fernandez
University of Chicago
Chicago, USA
raulcf@uchicago.edu

Torsten Hoefler
ETH Zürich
Zurich, Switzerland
htor@inf.ethz.ch

Kyle Chard
University of Chicago
Chicago, USA
chard@uchicago.edu

Ian Foster
University of Chicago
Chicago, USA
Argonne National Laboratory (ANL)
Chicago, USA
foster@uchicago.edu

Abstract

Efforts to reduce the environmental impact of HPC often focus on resource providers, but choices made by users, e.g., concerning where to run, can be equally consequential. Here we present evidence that new accounting methods that charge users for energy used can incentivize significantly more efficient behavior. We first survey 300 HPC users and find that fewer than 30% are aware of their energy consumption, and that energy efficiency is a low priority concern. We then propose two new multi-resource accounting methods that charge for computations based on their energy consumption or carbon footprint, respectively. Finally, we conduct both simulation studies and a user study to evaluate the impact of these two methods on user behavior. We find that while only providing users feedback on their energy use had no impact on their behavior, associating energy with cost incentivized users to select more efficient resources, and use 40% less energy.

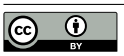
CCS Concepts

• **Hardware** → **Impact on the environment**; *Enterprise level and data centers power issues*; • **Human-centered computing** → User studies.

Keywords

Carbon-Aware Computing, Green Computing, Allocations

ACM Reference Format:
Alok Kamatar, Maxime Gonthier, Valerie Hayot-Sasson, André Bauer, Marcin Copik, Raul Castro Fernandez, Torsten Hoefler, Kyle Chard, and Ian Foster. 2025. Core Hours and Carbon Credits: Incentivizing Sustainability in HPC.



SC '25, St Louis, MO, USA

ACM ISBN 979-8-4007-1466-5/25/11
https://doi.org/10.1145/3712285.3759858

In *The International Conference for High Performance Computing, Networking, Storage and Analysis (SC '25), November 16–21, 2025, St Louis, MO, USA.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3712285.3759858

1 Introduction

HPC sustainability is a shared responsibility between infrastructure *providers* (who buy servers, source electricity, build and cool data centers, etc.) and *consumers* (who write software and decide what, where, and when to compute [5]). These roles should be complementary; however, they typically operate disjointly. Providers tout efficient facilities and investments in renewable/low-carbon energy [35, 54, 55] but are constrained by the amount, time, and location of consumer demand. Consumers are told that they can reduce environmental impact by using specific hardware or software, or by improving utilization, but lack information to accurately account for energy or carbon use. And while many tools exist to improve energy efficiency, these tools must first be adopted by users [10, 17, 59, 61] or require cooperation between users and providers [44, 53, 58, 60]. We argue and demonstrate that the key to achieving this cooperation is to charge users for energy/carbon used to incentivize more efficient behavior.

We first survey more than 300 HPC users to assess their knowledge, attitudes, and behaviors with respect to energy consumption. This pioneering survey of HPC users on energy awareness and sustainability reveals *a prevailing lack of awareness of energy use, lack of action to reduce energy, disregard for efficiency or sustainability rankings, and a low priority on energy efficiency.*

To address these issues, we investigate *adopting an application's environmental impact as the measure of its resource usage.* We formulate two versions of impact-based accounting: **Energy-Based Accounting (EBA)** which charges jobs based on energy used, and **Carbon-Based Accounting (CBA)**, which charges jobs based on estimated carbon footprint. Thus, a user might be allocated 10 kg carbon emissions ($kgCO_2e$), rather than 100 node-hours; be able to estimate the $kgCO_2e$ required for a computation on different

machines; and track $kgCO_2e$ rather than node-hours. We envisage using EBA and CBA as the basis for fungible multi-resource HPC allocations where users choose between one of several machines to which they have access [7]. Previous approaches for incorporating energy cost into cloud prices [2, 23, 30, 38] are not directly applicable to HPC: they aim to set cloud resource prices to recoup costs, not a primary concern in HPC allocations, and do not account for carbon emissions. We hypothesize that the adoption of such accounting schemes will incentivize users to develop more energy-efficient applications and select resources that are more energy efficient, allowing them to compute more for the same cost.

To enable experimentation with impact-based accounting across heterogeneous machines, we develop *Green-ACCESS*, which leverages a Function-as-a-Service (FaaS) interface on top of HPC infrastructure [9]. This system makes energy consumption transparent, seamlessly guides users to more efficient machines, and incentivizes sustainable use of computing resources. With this system, we illustrate trade-offs between energy, time, and carbon-footprint, where, for instance charging based on peak-performance leads to a cheaper price for a machine that uses twice the energy. Finally, we design a novel user-study using a web-based game to ascertain how energy information and impact-based accounting affect user behavior. We find that *showing users their energy-use has no impact on behavior, but linking price with energy can encourage users to run on more efficient devices.*

The contributions of this paper are:

- The first large-scale survey of 300+ HPC users focused on their awareness of energy and carbon use. We release the aggregate data to the community.
- Energy- and Carbon-Based Accounting (EBA and CBA) mechanisms for charging for computing based on environmental impact.
- A FaaS platform that implements EBA and CBA on diverse CPUs and GPUs. We open source the platform to be used as a plug-in on top of Globus Compute.
- A user study in which users interact with impact-based accounting via a web-based simulation game.

2 Survey

We first examine the prevailing attitudes of HPC users towards sustainable computing practices. Specifically, we developed a survey to understand users' awareness of existing sustainability metrics and techniques.

2.1 Survey Design

We employed Qualtrics, a platform for assembling and managing online surveys. We defined 33 questions that examine a participant's familiarity with their energy consumption and other sustainability-related questions. Our target participant was anyone who submits jobs or develops code for HPC machines. As such, we distributed the survey to HPC user groups, facilities, science collaborations known for use of HPC, and user groups of popular HPC tools.

2.2 Results

We received 316 responses, of which 192 completed 90% of the survey or more. 166 respondents were located in Europe, 104 in North America, 4 in Oceania, 4 in China, and 38 declined to share their location. 73 respondents were graduate students, 97 were early career researchers/engineers, and 99 were senior researchers/engineers. We ask respondents to self-identify as HPC users and report the amount they used HPC. There was no significant difference in responses based on career stage or node-hours used.

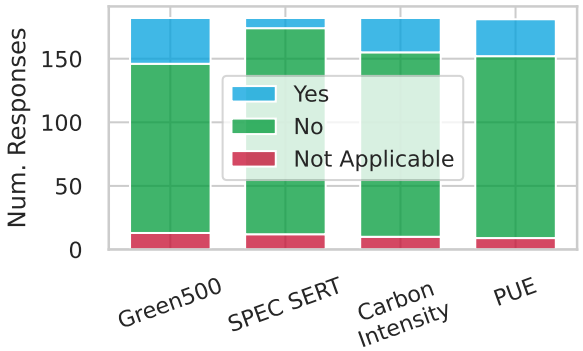


Figure 1: Responses to question "Are you aware of how the HPC resources you use perform on the following sustainability metrics?"

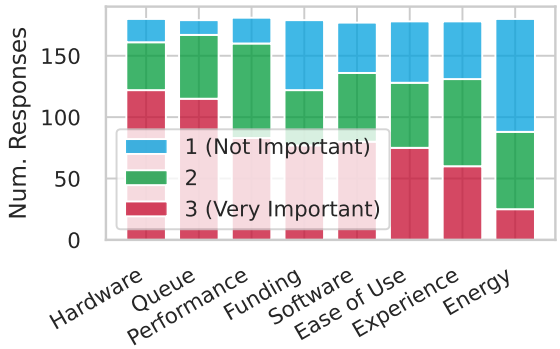


Figure 2: Energy efficiency is among the least important factors for users when choosing where to run a job.

We first examine HPC user awareness of their resource use. 73% of respondents (148 people) were aware of how many node-hours a job/workflow consumes, and 70% (142) indicated that they had taken steps to reduce the number of node-hours used. This aligns with the more than 80% of respondents (166) who were very or mildly concerned with finishing their jobs within their allocation, typically measured in node-hours. 77% of those concerned about completing their jobs had taken steps to reduce the node-hours their jobs consume. **Overall, users are aware of their node-hour usage, concerned about completing their jobs within their allocation, and take steps to reduce their node-hours as a result.**

In contrast, only 27% of respondents (51) were aware of the amount of energy their workload consumes and 30% (54) had taken steps to reduce energy use. Counterintuitively, there is not strong overlap between these groups. 39% of people who have taken steps to reduce their energy use were not aware of how much energy their jobs consume. **Few users are aware of their energy consumption or have taken steps to reduce it.**

We also examine familiarity with metrics that assess efficiency. Rankings like Green500 [19] and Graph Green500 [24], and benchmarks like the SPEC Server Efficiency Rating Tool (SERT) [31], distill machine efficiency to a single metric to help providers or users select resources. Carbon intensity captures carbon emissions from producing electricity, which varies based on facility location [18]. Power Usage Efficiency (PUE) captures the fraction of facility electricity that is used for computing rather than cooling or other needs [54]. Participants reported some familiarity with these metrics, with 51% (94) and 30% (55) of respondents knowing of Green500 and carbon intensity, respectively. However, this knowledge does not affect user behavior. As shown in Figure 1, few respondents were aware of how the resources they use perform on these metrics. For instance, of the 94 people familiar with the Green500 list, only 36 (20% of all respondents) knew how the machine they were using performed on that ranking. **While users are familiar with metrics designed to improve sustainability, most do not know how they apply to their own machines.**

Given these findings, it is not surprising that energy efficiency is among the least important metrics respondents used when selecting a machine. In most cases users have many options when choosing which machine to use: more than 70% of users have access to four or more machines. Figure 2 shows participant responses when asked how important various parameters were in selecting which machine to use. While 46% of respondents (83) said machine performance was very important, only 12% (25) said energy efficiency was very important. This is a challenge for providers looking to invest in more efficient options for users: without a shift in incentives, users will prioritize performance over efficiency. **In summary, users do not select machines based on energy efficiency, instead they prioritize hardware availability, queue times, performance and funding.**

3 Impact-Based Accounting

Our survey revealed that most users of HPC resources do not consider energy (or carbon) when deciding what to run or where to run it. To alleviate this issue, we explore resource accounting models in which the cost of using a machine is based on the energy or carbon consumed by the computation. As we discuss in the following, our approach combines solutions to two problems: (1) How to incorporate both energy and usage into an accounting method; and (2) How to account for carbon.

3.1 Background: Fungible Allocations

We design accounting methods for *fungible* allocations: allocations that are resource independent and may be redeemed on multiple resources. Such allocations are employed, for example, by the US National Science Foundation (NSF) Advanced Cyberinfrastructure Coordination Ecosystem: Services and Support ACCESS [7], Chameleon Cloud [28], and internally within Google [56]. Fungible allocations provide flexibility by allowing users to select between different machines, which may potentially have different efficiencies. Fungible allocations may be based only on time (e.g., Chameleon Cloud allocates users a number of node-hours that can be redeemed on any node type) or on a mix of time and performance (e.g., ACCESS grants users *service units* that can be exchanged for allocations on specific machines based on machine-specific exchange rates; Google tracks *Google Compute Units* which standardizes core-time to the same amount of computational power on any machine in the fleet).

3.2 Energy-Based Accounting Model

Our **Energy-Based Accounting (EBA)** model *charges users for energy used rather than time spent computing.*

Intel and AMD CPUs support accurate CPU energy readings [29] that can be collected by a cluster management tool [3]. Although not available for this work, Baseboard Management Controllers (BMCs) can on some machines be used to measure energy used by a whole node. To account for differences in data-center design and cooling, the measured energy could be multiplied by the PUE. Many HPC centers already implement monitoring of power consumption [21, 52], so these energy could be incorporated into accounting models without monetary overhead.

A challenge with charging for energy consumed is the trade-off between *potential* and *actual* usage. When accounting for time, i.e., node-hours, providers charge for the entire allocated resource, regardless of how it is used, because the resource cannot also be allocated to another user. That is, they charge based on potential rather than actual usage. An accounting model that charges based on energy consumed rather than time spent results in the end price depending on user activity. This is a double-edged sword: users who write more efficient software are rewarded with reduced costs, but so are users who do not fully use allocated hardware, even though the provider cannot charge others for the resource.

To address this issue, we incorporate the potential use of a node into our calculation. We use a processor's Thermal Design Power (TDP)—the maximum sustained power that it can dissipate—as a surrogate for a node's full utilization. We charge users for the average of the actual energy used and the energy that would have been consumed had they used the resource fully. For a job j on resource R that uses energy e_j and takes duration d_j, the cost based on the **energy charge** is:

$$\hat{e}_j = (e_j + \cdot d_j \cdot \mathrm{TDP}_R)/2. \quad (1)$$

As a refinement, the second term could be weighted by a parameter $\beta < 1$ for scenarios where the TDP of a device is much greater than the typical power use, skewing the cost. However, we do not employ that refinement here.

Factoring potential use into EBA, ensures that energy efficiency is balanced with job duration. The average between energy used and TDP is both simple and transparent.

3.3 Carbon-Based Accounting Model

Energy consumption is not the only factor that contributes to the sustainability of a computing facility [32]. Estimating the carbon footprint of a computation gives us a mechanism to account for more holistic impacts. To that end, our **carbon-based accounting (CBA)** model charges users based on the gCO_2e, i.e., grams of carbon dioxide equivalent, used. This footprint encompasses the *operational carbon* of the electricity used to run a job and a portion of the machine's *embodied carbon* that we attribute to a job.

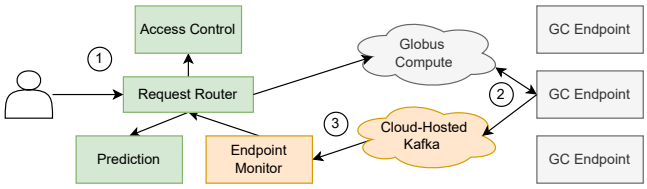


Figure 3: Architecture of the *green-ACCESS* prototype. The three principal system components, shaded green, grey, and orange, are described in the text.

We use the carbon intensity, $I_f(t)$, of the electricity grid at facility f at time t to estimate the operational carbon of a job. Carbon intensity captures the carbon emissions produced in generating electricity, in gCO_2e per kWh. This measure depends on the generation source, which varies by location and time; estimates can be obtained from grid operators or public APIs [18].

Embodied carbon provides a principled method of weighting potential use of a resource against actual use. To attribute embodied carbon to a job, we differ from the standard practice of allocating the embodied carbon of a device linearly based on time [50]. Instead we treat the embodied carbon like a capital expense invested in the machine that depreciates over time. Our rationale for this approach is that the embodied carbon allocated to a job should be proportional to the utility derived from using the machine. Users demanding the latest technologies drive providers to acquire new machines. Alternatively, users who leverage older technology allow hardware providers to extend refresh cycles [36]. Thus, our model charges a higher rate to users who use a machine earlier in its lifespan.

Specifically, we employ a form of accelerated depreciation called double declining balance [15]. In line with typical refresh periods [36], we assume an HPC machine has a depreciation period of five years, which corresponds to a 40% annual depreciation rate. For a machine with C_f total embodied carbon, the unaccounted-for carbon y years after the machine was installed is:

$$R_f(y) = C_f \cdot (1 - 0.4)^y;$$

the embodied carbon allocated to year t is:

$$D_f(y) = 0.4 \cdot R_f(y);$$

and the carbon-rate per hour of resource utilization is

$$D_f(y)/(24 * 365).$$

Thus the total **carbon charge** for a job j run at facility f with carbon intensity at time t of $I_f(t)$ that uses e_j kWh is:

$$c_j = e_j \cdot I_f(t) + d_j \cdot D_f(y)/(24 * 365). \quad (2)$$

4 *Green-ACCESS* System Implementation

We implemented *green-ACCESS*, a HPC-FaaS platform that provides fungible allocations across machines with EBA and CBA.

4.1 *Green-ACCESS* Design

We implement *green-ACCESS* with a FaaS interface that provides a flexible and adaptive runtime, allowing us to efficiently realize impact-based allocations. Recent work shows that FaaS can improve resource utilization and accelerate applications on HPC systems [9, 12, 27]. FaaS simplifies migration of applications between different systems which we leverage to show the differences between impact-based accounting and other accounting mechanisms. Our system comprises three major components (see Figure 3).

(1) The frontend supports user interactions, with accounting and admission control. Users can employ a Web or Python interface to access a prediction service that provides estimates of the energy consumption of their jobs. Jobs are submitted via a FaaS interface.

(2) *Green-ACCESS* uses Globus Compute as the FaaS platform to run Python functions on HPC systems [9]. *green-ACCESS* calculates expected costs before forwarding requests to Globus Compute. Registering a machine with *green-ACCESS* requires deploying a Globus Compute Endpoint (GCE) equipped with a monitor that polls data from the RAPL interface, reads hardware counters, and communicates those data back to *green-ACCESS*. This integration with Globus Compute allows *green-ACCESS* to be deployed within (or beside) existing cluster management systems.

(3) Energy and performance counter data are transferred via Kafka to *green-ACCESS*, where they are consumed by the *green-ACCESS* endpoint monitor, a streaming consumer based on the Faust library [51]. This monitor disaggregates per-node power measurements from the RAPL subsystem [13, 29] into user jobs, which are provisioned by core. To this end, we collect per-process hardware performance counters and periodically fit a power model between performance counters and measured energy [20, 46]. Per-process estimates are aggregated to obtain the energy used by a task. For GPUs, we assume that an entire GPU is allocated to each job.

4.2 Comparison of Accounting Methods

Using *green-ACCESS*, we evaluate the price of executing functions on CPUs and GPUs with five accounting methods:

- **Runtime**: Price is determined only by the core-time used, not accounting for heterogeneity. This is similar to the model used by Chameleon Cloud [28].
- **Energy**: Price is determined only by the energy used, without accounting for device capacity.
- **Peak**: Price is determined by core-time used, multiplied by machine peak performance. This metric accounts for heterogeneous devices by charging more for higher performance systems, similar to ACCESS [7].
- **EBA**: The proposed Energy-Based Accounting.
- **CBA**: The proposed Carbon-Based Accounting.

Green-ACCESS allows us to calculate the cost of an invocation using any of the accounting methods.

4.2.1 Green-ACCESS on CPUs. We start by executing five applications from the SeBS [11] benchmark and two scientific applications [42, 57] on CPU nodes with *green-ACCESS*. We select nodes of different generations and with varied performance characteristics: A Desktop with an i7-10700 CPU, a *Cascade Lake* node with 2 Intel Xeon 6248R, and an *Ice Lake* node with 2 Intel Platinum 8380 CPUs, and a *Zen3* node with 2 AMD EPYC 7763 processors. Figure 4 shows the runtime and energy consumed for each application and highlights the different energy/performance tradeoffs that exist between applications and nodes.

Table 1 shows the normalized costs of the Cholesky decomposition application for each accounting method. We chose this application as it most clearly demonstrates unexpected tradeoffs

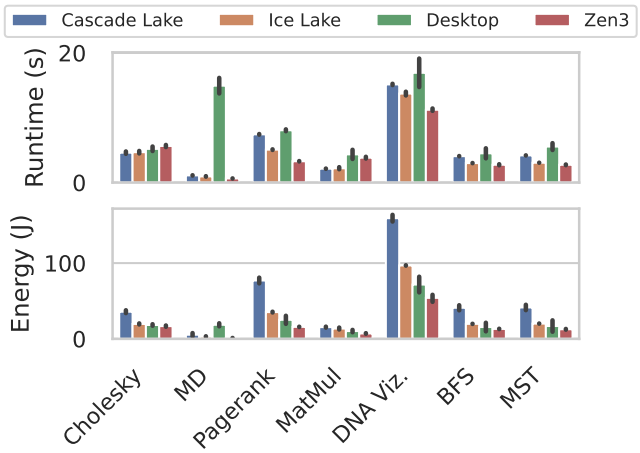


Figure 4: Runtime and energy consumption of seven applications running on four different nodes reveal different performance/energy tradeoffs across machines.

Table 1: Runtime, energy consumption, and costs on different CPU nodes running a Cholesky Decomposition.

Machine	Metrics		Normalized Costs		
	Runtime (s)	Energy (J)	EBA	CBA	Peak
Desktop	5.20	18.3	**1.0**	**1.0**	1.43
Cascade Lake	4.68	35.8	1.90	1.20	**1.0**
Ice Lake	**4.60**	19.8	1.10	1.10	1.06
Zen3	5.65	**16.8**	1.05	1.15	1.36

and differences between accounting methods. The table shows the undesirable properties of the *Runtime* and *Peak* accounting methods. As the Cholesky application runs fastest on the Cascade Lake and Ice Lake systems, charging based solely on *Runtime* assigns the lowest cost on those machines. However, in this case, those machines also use the most energy. Weighting by peak performance, as in *Peak* does not rectify this. In this case, Zen3 and Desktop have the highest peak performance per thread [39] but are the slowest systems for this task. As a result, the Cascade Lake machine is the cheapest under the Peak accounting method, even though it uses more than 2× as much energy as the Zen3 machine. **The *Runtime* and *Peak* accounting methods both indirectly incentivize higher energy use.**

In contrast, the *EBA* price mirrors task energy usage. Desktop, which has the 2nd lowest energy use, has the lowest *EBA* cost. Zen3 uses the least energy but has a slightly higher *EBA* cost than Desktop due to pricing for utilization in Equation 1: Zen3 has a higher TDP than Desktop, so it costs more to use per time. *CBA* shares some properties of EBA, with Desktop having the lowest cost and Cascade Lake the highest. However, the newer Zen3 is charged a higher price for embodied carbon. **Overall, we see that running on a more efficient machine costs less under *EBA* and *CBA*.**

4.2.2 Green-ACCESS on GPUs. The accounting methods and tradeoffs that we have demonstrated extend to GPUs. We run a tiled

Table 2: Specifications of GPU nodes and carbon rate for different number of GPUs. GFlop/s is manufacturer reported. The average carbon intensity of all nodes was 53 gCO_2e/kWh. Embodied carbon was calculated using SCARIF [25].

GPU	Year	GFlop/s	TDP	Carbon Rate			
# GPUs				1	2	4	8
P100	2018	6700	250	8.5	9.1		
V100	2019	14000	250	19	20	23	28
A100	2021	18000	400	87	93	106	131

Table 3: Comparison of runtime, energy use, and costs under different accounting schemes for Cholesky Decomposition with different types and number of Nvidia GPUs.

	#	Metrics		Normalized Costs		
		Runtime (s)	Energy (kJ)	EBA	CBA	Perf.
P100	1	2321	889	1.20	1.40	**1.0**
	2	1396	**635**	**1.0**	**1.0**	1.20
V100	1	1494	1316	1.23	2.07	1.34
	2	1190	1194	1.26	1.88	2.14
	4	917	916	1.25	1.44	3.30
	8	926	944	1.85	1.49	6.67
A100	1	1405	2100	1.83	3.35	1.62
	2	926	1427	1.46	2.28	2.14
	4	841	1320	1.76	2.11	3.89
	8	**838**	1325	2.59	2.13	7.76

Cholesky decomposition on a 42 GB single precision matrix, using the StarPU runtime system to orchestrate the application across different Nvidia GPUs [4] (see Table 2 for specifications). Table 3 shows the energy consumed by different generations and numbers of GPUs. We observe that: (1) Energy consumption decreases as we scale up to four GPUs, but then stabilizes from four to eight GPUs as the application problem size is too small to saturate all GPUs. Furthermore, we see that both runtime and energy remain similar, implying good scaling performance with respect to energy. (2) Recent GPUs consume more energy for modest performance gains (specifically between the V100 and A100). For the Cholesky application, the critical path and data transfers limit the benefits of the additional cores available on newer GPUs. Indeed, the newest GPU (A100) solves the problem 6% faster than the previous generation (V100), but consumes 60% more energy.

Table 3 also illustrates how *EBA* and *CBA* manage this trade-off compared to other accounting methods. Eight A100 devices provide the best performance, but consume twice the energy of the P100s. On the other hand, *EBA* and *CBA* both prioritize using two P100 GPUs, which provide the lowest energy consumption and have the lowest embodied carbon rate compared to newer GPUs. Meanwhile, a *Peak* charges the least for using one P100 GPU even as it uses 40% more energy and 66% more time than two P100 GPUs. **EBA and CBA intentionally balance the trade-offs between the higher performance of modern GPUs and the lower energy/carbon use of older GPUs.**

Table 4: Comparison of operational carbon vs. different methods of attributing embodied carbon: linear depreciation, and accelerated depreciation (ours). Compared to linear depreciation, accelerated depreciation charges less for older machines.

Machine	Age	Operational ($mgCO_2e$)	Embodied ($mgCO_2e$)	
			Linear	Accel.
Desktop	3	2.1	1.5	0.6
Cascade Lake	4	2.8	1.0	0.3
Ice Lake	2	0.9	1.4	1.0
Zen3	1	1.2	1.3	1.6

4.3 Comparison of Embodied Carbon

In Table 4, we compare different methods for accounting for carbon using the same Cholesky decomposition application as above. Specifically, we compare the operational carbon to two different methods of charging for embodied carbon: a linear method of depreciation where the embodied carbon rate is constant over the lifetime of the resource [50], and our accelerated depreciation method.

Bashir et al. [6] claim that optimizing for embodied carbon has the potential to increase overall carbon emissions. They argue that the embodied carbon cannot be reduced, so the most sustainable policy is to use the machine with the lowest operational emissions. This possibility is reflected in the table: the Cascade Lake node uses 2.8 mg, the most operational carbon of any node, but is attributed the least embodied carbon, 1.0 mg based on linear depreciation and 0.3 mg based on accelerated depreciation. Including embodied carbon makes the price of the least efficient machine closer to the other machines. However, newer hardware is acquired at least partly in response to user behavior: users that demand and use (only) the latest hardware drive providers to retire older machines [36]. Thus, we argue that this observation motivates the use of accelerated depreciation instead of linear depreciation to allocate more embodied carbon earlier in a resource's lifespan and extend the productive lifespan of these machines. The accelerated depreciation attributes 0.7 mg less embodied carbon than linear depreciation on the older Cascade Lake machine, but 0.3 mg more embodied carbon on the newer Zen3 machine. As longer lifespans will lengthen refresh cycles and lessen the number of devices manufactured, accounting for embodied carbon will reduce emissions over the long run.

5 Simulation Studies

The implementation of *green-ACCESS* demonstrates that trade-offs between energy consumption and performance can lead to different machines being cheaper under **EBA** and **CBA**. To analyze EBA and CBA at scale, we conduct simulations of real workloads and machines. We modify an existing batch simulator to use the proposed accounting methods on multiple machines [22]. Our goals are to (1) understand how different choices affect the cost of running a workload under EBA and CBA, (2) examine how different accounting models translate to resources used, and (3) explore how low-carbon scenarios may affect the accounting models.

5.1 Machines

We characterize in Table 5 the machines used in our simulation. For each, we indicate: Cores per Node; CPU Thermal Design Power (the maximum heat in watts that a CPU can dissipate); Idle Power (power consumed by the CPUs when running only the monitoring code); Carbon Rate (see Section 3.3, where the embodied carbon was calculated using manufacturers datasheets where available or SCARIF [25]). For carbon intensity, we retrieved data at an hourly resolution assuming the simulation starts in January 2023, and report the yearly average. Three of the machines are in HPC centers: FASTER a cluster at Texas A&M University available through NSF ACCESS [7], Midway an Institutional Cluster (IC) at the University of Chicago, and Theta a machine at Argonne Leadership Computing Facility. The fourth is a personal computer referred to here as Desktop.

5.2 Workload

We use a published dataset of per-job energy use from two HPC clusters [40]. In order to infer the energy characteristics, we assume jobs run by the same user with the same requested resources are repetitions of the same app and have the same cross-platform characteristics. Discarding jobs that lack an associated energy value reduces the dataset from ~84k to 71,190 jobs. We repeat each execution twice to generate a workload of 142,380 jobs. Any job can run anywhere, except that 17% of jobs require more cores than are available on the one-node Desktop.

The dataset that we use to construct this workload provides, for each job, energy consumption only on one machine. We must extrapolate these data to our machines. To begin, we assume that the reported runtime and energy consumption correspond to running on IC, the system most similar to the source dataset. We then adapt a method to predict energy use and runtime had jobs been run on the other machines [43]. First, we generate realistic values for hardware performance counters (i.e., LLC Misses/sec., Instructions/sec) for each job using a Gaussian Mixture Model trained on data collected on IC. We then use a KNN [43] trained on a set of benchmark applications [11] to estimate runtime and power consumption on the other machines.

5.3 Policies

To simulate user choice, we define eight machine selection *policies* that select which machine to submit a job to based on the system state (queue times) and job profile (energy use, runtime, or cost). Each simulated user is limited to one running job on a cluster at a time. The eight policies are:

- **Greedy**: Select machine that minimizes cost, according to the accounting method (EBA or CBA) used.
- **Energy**: Select machine that minimizes energy used.
- **Mixed**: Balance runtime and cost. Select machine with the least allocation cost (in terms of EBA or CBA) *unless* another machine can complete the job in half the time, in which case select that machine.
- **EFT** (earliest finish time): Select machine that minimizes completion time, i.e., queue time + runtime.
- **Runtime**: Select machine with shortest runtime.
- **Theta**, **IC**, and **FASTER**: Always select that machine.

Table 5: Machines used for simulation. TDP=Thermal Design Power, in Watts. Idle power is for all sockets on the node.

Machine	Year Deployed	CPU Model	# of Cores	CPU TDP (W)	Idle Power (W)	Carbon Rate (gCO_2e/h)	Avg. Carbon Intensity (gCO_2e/kWh)
TAMU FASTER	2023	2 × Intel Xeon 8352Y	64	205	205	105.2	389
Desktop	2022	Intel Core i7-10700	16	65	6.51	12.2	454
Institutional Cluster	2021	2 × Intel Xeon 6248R	48	205	136	16.7	454
ALCF Theta	2017	Intel KNL 7320	64	215	110	2.0	502

Table 6: Energy and carbon used when deploying each policy while computing the workload. We show how the *Greedy* and *Mixed* policy behaved both under *EBA* and *CBA*.

Policy	Energy (MWh)	Carbon ($KgCO_2e$) Operational	Attributed
Greedy - EBA	328	88	322
Greedy - CBA	491	167	**228**
Mixed - EBA	407	132	319
Mixed - CBA	494	172	275
Energy	**321**	**83**	345
EFT	486	169	315
Runtime	501	170	237

5.4 Results with Energy-Based Accounting

Figure 5a depicts the amount of work (in core-hours) completed by a user when applying each policy with a fixed allocation. Specifically, we consider work to be the average number of core hours required to run a job across all machines. This places higher weight on larger and longer jobs, without favoring any one machine.

In general, the single-machine policies (*Theta*, *IC*, and *FASTER*) and policies that do not consider energy (*EFT* and *Runtime*) perform less work for the same cost. A user using *Theta* is severely disadvantaged by EBA because they submit all jobs to a machine that is inefficient for most tasks. Using *Greedy*, a user is able to complete the most work, because, by definition, they use the cheapest machine for every task. This results in completing 28% more work than when using the performance focused *EFT* policy. A user employing *Energy* can complete 99% of the work done using *Greedy* because the most efficient machine is often the cheapest machine. **Under EBA, by optimizing for energy, a user is indirectly optimizing for cost; in other words, the cost incentivizes users to be more energy efficient.**

We now consider only the multi-machine policies and examine how each policy affects total energy consumption. Table 6 shows the energy consumed using each policy over the workload. Logically, we see that using *Energy* consumes the least energy. A user optimizing cost instead of the energy by using *Greedy* resulted in only 2% more energy use. In contrast, applying *EFT* or *Runtime* results in 51% or 56% more energy used, respectively. Much like we saw in real hardware, performance and efficiency are not always aligned. **A user prioritizing the speed in terms of either makespan or core-hours may make inefficient decisions.**

Next, we examine how the different policies relate to completion time. Using a single machine is detrimental in terms of completion time because of long queue times. This is visible in Figure 5b, which shows the jobs completed by different policies over time. We also see that using *Mixed* a user is able to compute 100,000 jobs as fast as using *EFT*. **By balancing energy efficiency and performance, users can reduce cost under EBA without impacting completion time.**

Finally, we investigate how EBA might affect the usage of each machine. Figure 5c shows how jobs are distributed with each policy. *Greedy* and *Energy* policies distribute jobs similarly to FASTER, Desktop, and IC. *Greedy* and *Energy* allocate no tasks to Theta because it is neither the cheapest nor the most energy efficient. However, *Mixed* distributes tasks to all four machines to avoid long queues and reduce the completion time. **Overall, EBA may lower utilization of inefficient machines to reduce energy consumption, but users may choose to use less efficient machines at a higher cost to reduce completion time.**

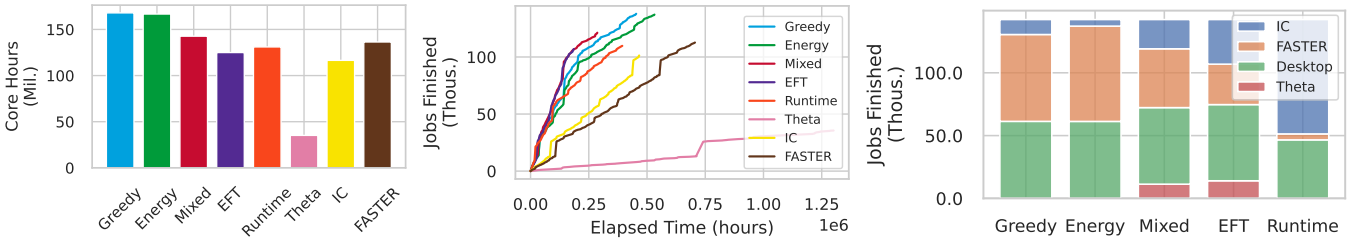
5.5 Results with Carbon-Based Accounting

Next, we analyze CBA as described in Section 3.3. For Figure 6, we allow a user employing *Greedy* to run the same amount of work as in Figure 5a. We see that using *Energy*, a user can run 22% fewer jobs than under EBA, while using *Runtime*, a user can run 23% more. This is due to the high carbon rate of FASTER: while often selected by *Energy* because it is the most efficient machine (see Figure 5c), it has a much higher embodied carbon rate. Since *Energy* favors FASTER and *Runtime* favors IC, deploying *Runtime* policy allows a user to complete more work in the same allocation. *Greedy* adapts to the new accounting scheme and submits 50% of its workload to IC and only 11% to FASTER.

The energy and carbon consumed while using each policy are detailed in Table 6. In the table, we decompose the operational carbon emissions (that is, the carbon intensity of the grid multiplied by the energy used), from the attributed carbon emissions, which includes a portion of the embodied carbon of the machine. Based off the attribution of CBA, the direct carbon emissions are between 24% and 72% of the total carbon attributed to a job. In particular, *Energy* shows the largest share of its carbon footprint attributed to embodied carbon as a result of heavily using the latest hardware. **Minimizing cost allows a user to compute the most work while using the least attributed carbon, incentivizing the selection of *efficient* and *older* machines.**

5.6 Results with Reduced Carbon Intensity

As countries transition to green electricity sources, grids have a higher temporal variation in carbon intensity [60]. We simulate



(a) Work completed with fixed allocation. (b) Number of jobs completed over time. (c) Distribution of jobs over machines by policy.

Figure 5: Results from simulating EBA. The *Energy* and *Greedy* policies complete the most work by prioritizing FASTER over IC for efficiency. The *Mixed* policy trades some efficiency for faster completion time.

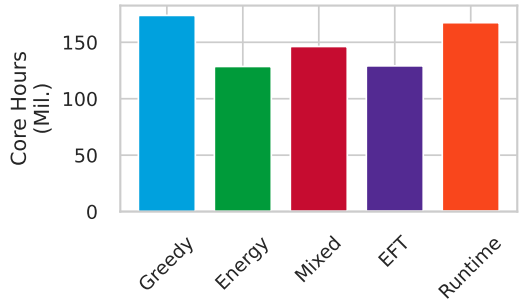


Figure 6: Results from a simulating of CBA, showing the amount of work a user completed for a fixed cost by employing different machine selection policies .

a scenario involving low-carbon grids with high temporal variation. We assign each of our systems to a grid with high variability in carbon intensity: Southern Australia (IC); Ontario, Canada (FASTER); Bornholm, Denmark (Theta); and Southern Norway (Desktop) [18, 53]. We do not adjust the embodied carbon rate in this simulation.

In Figure 7a, we show the work completed by each policy under this simulation. As before, the carbon-aware *Greedy* completes significantly more work than the other, non-carbon-aware policies. The interesting difference is in how *Greedy* exploits the variability in carbon. Throughout the day, the carbon intensity varies with the electricity source in each region. The carbon intensity on an arbitrary day in the simulation is shown in Figure 7b. This variation affects the distribution of the cheapest cost endpoint for each job: see Figure 7c. The cheapest location shifts from Theta to IC later in the day, as carbon intensity increases in Denmark and renewable generation comes online in Southern Australia. In the simulation (as well as above), we do not allow job migration: once a job has been started on a machine, it cannot move even as the carbon intensities change. **CBA incentivizes users to spatially and temporally align their jobs with periods of renewable energy generation.**

6 User Study

In this final study we examine whether and how user behavior changes under a modified accounting method.

6.1 Design

We built a JavaScript game to emulate decisions faced by users of a *green-ACCESS*-like platform: see Figure 8. Participants were asked to imagine that they were a computational scientist who had to finish all of their jobs within a time and allocation limit, and they had four machines to use to do so. These instructions are a proxy for realistic HPC use cases where a researcher is granted a limited allocation but faces pressure to meet a deadline. In our game, a job could be "scheduled" by dragging and dropping it onto a specific machine. Every job was randomly assigned one of four priorities between "Low" and "Very High." The priority functioned as a placebo metric. No instructions were given on how to treat priorities, so users had to weigh priority against a jobs time, energy, and cost. The machines reflected those used in the simulation, and the resources a job used were inferred using the same mechanism as the simulation. More jobs "arrived" as jobs were scheduled, to reflect the time-dependent nature of real workloads. The jobs were the same for all participants. Each participant was randomly assigned to one of three game versions:

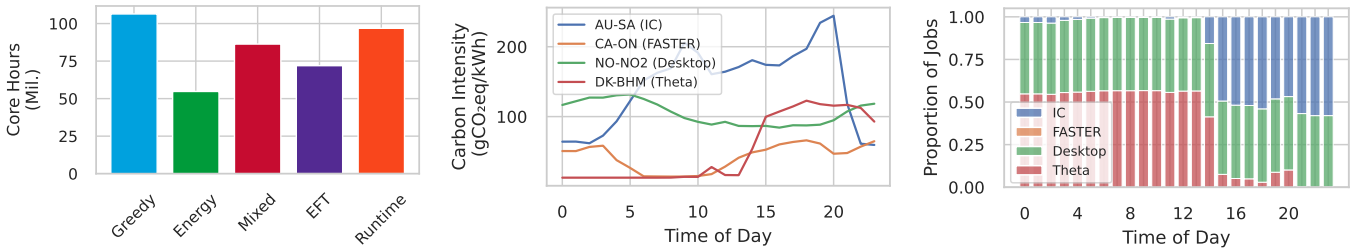
- **V1**: Job cost is proportional to runtime, and no information on job energy consumption is shown. This reflects current standard practice: see Section 2.2.
- **V2**: Cost is the same as V1, but job energy consumption is displayed next to time and cost.
- **V3**: Cost is given based on the EBA formula.

Participants played the game at least twice, with the first iteration, intended for familiarization with the game mechanics. The version remained the same between the first and second play of the game, but was randomized after that. The game was distributed using the same channels as the survey.

6.2 Results

After discarding every users first time playing the game, we received 207 instances of the game played by 90 unique users. We discarded 15 instances in which participants finished the game in less than one minute as these also had a disproportionately high amount of allocation remaining.

We first assess the impact of displaying energy information (V2) and using EBA (V3) on total energy consumption: see Figure 9. On average, participants using EBA (V3) consumed 1928 kWh in the simulation, while those with V1 or V2 consumed 3262 kWh and 3142 kWh, respectively. V3 is significantly lower than V1 or V2 ($p=0.00$), but there was no significant difference between V1 (the control) and V2 (with energy information). **Information regarding energy consumption alone created no change in how much energy a participant consumed.**



(a) Work completed with a fixed CBA allocation and with reduced carbon intensity.

(b) Variations in carbon intensity during a single day of the simulation.

(c) Distribution of of the lowest cost endpoint as the carbon intensity varies.

Figure 7: Results from simulating CBA under a low-carbon scenario. As the carbon intensity varies, the cheapest machine for a job can change. Efficiency becomes less important when carbon intensity is low, so Theta is often the cheapest machine.

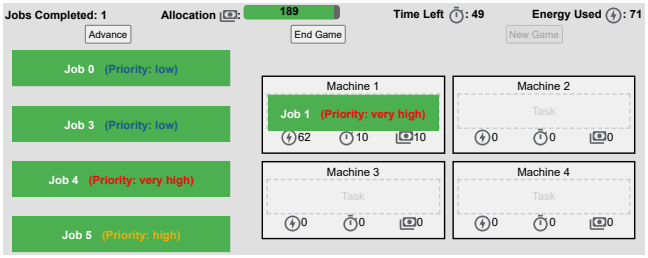


Figure 8: Scheduling game. The green "job" boxes can be "scheduled" by dragging them to the light-grey "machine" boxes. Job time and cost, was shown by hovering over a job.

However, in V3 (using EBA), participants also completed fewer jobs overall. Users with the changed allocation scheme completed an average of 9.7 jobs compared to 14.5 in V1 and 14.9 in V2. Although we attempted to give an equivalent sized allocation to users in V3 as V1 and V2, the differing accounting method meant an exact conversion was impossible. To distinguish the effect of EBA from a smaller allocation, in Figure 9c we examine the amount of energy used, stratified by the number of jobs completed. The figure shows for any number of jobs completed, **users who saw an EBA version of the game used significantly less energy**.

To use less energy while running the same number of jobs, a user had two options: run less energy intensive jobs, or use more efficient machines to run the same jobs. In order to distinguish these two mechanisms, we looked at the probability that a user selected to run a job. Since users may have ran out of time or allocation before seeing a job, we calculate the probability as (# of users who completed job i / # of users who saw job i). If participants reduced energy by not running energy intensive jobs, we would see a (negative) correlation between the probability a job was run and its energy use. We plot this correlation in Figure 10. While V3 participants were less likely to run a job in general (since they completed fewer jobs on average), there is no correlation between job energy use and the probability that a user ran a job. Thus, **the decision to run a job was not influenced by energy consumed, even when cost depended on energy.**

The second mechanism for consuming less energy was to employ more efficient resources. We assess the efficiency of resources used by measuring the average energy used to run a job by all participants playing a specific version of the game. Lower average energy consumption means more participants chose more efficient machines. For 16 of the 20 jobs, the average energy used by participants in V3 of the game was the lowest of the three versions. This suggests that **under EBA, when participants elected to run a job, they selected a more efficient machine to do so.**

7 Limitations

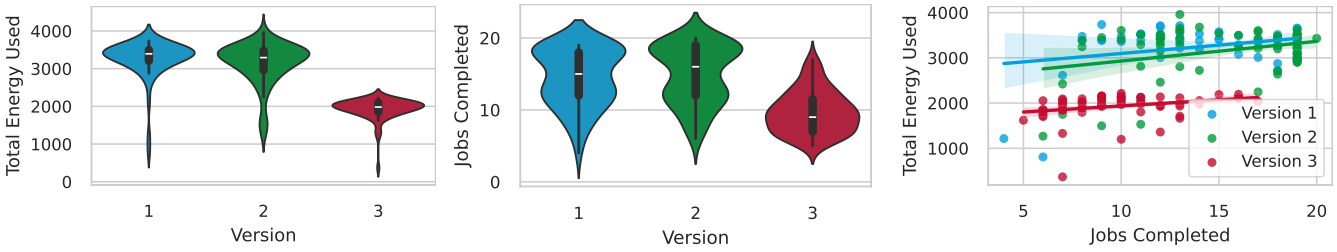
We assumed that applications are able to be migrated between machines and providers will cooperate to enable efficient and sustainable use of computing. Here, we address the validity and limitations of these assumptions.

Reliance on Portability. Performance portability is an open challenge and our results are further evidence of the need for portability solutions. Already many HPC users are agnostic to underlying machine details. For instance, users writing high-level PyTorch code can efficiently leverage most GPU devices without modifying their code. In such cases, selecting between machines based on energy or carbon can already yield substantial reduction. Ongoing research into libraries that abstract different machine-specific backends and containers that ease the deployment of applications will facilitate more portable applications. Even in situations where only a single machine is available or suitable for an application, EBA/CBA can incentivize the adoption of energy-aware schedulers [27, 59] and other tools [10]. As we find in our survey, these tools have not been well adopted, possibly arising from a lack of incentive structures.

Simplified Provider Behavior. Our study of EBA/CBA assumes providers will cooperate to incentivize users to make sustainable choices. Since HPC systems are typically built/funded by governments/universities for the public interest, this cooperation is nominally aligned with the goals of the centers. However, we do not consider latent motivations such as competition between facilities for future funding that make modeling provider behavior complex. Furthermore, EBA/CBA may increase the energy use or carbon footprint of a single machine in order to reduce the overall impact, which could make sites reluctant to adopt these approaches. We hope incentive structures like EBA and CBA will stimulate the discussion on energy/carbon efficiency of scientific computing at a global level.

8 Related Work

Motivating sustainable behavior. Limited work studies user incentives and behavior in regards to HPC use and energy. Georgiou



(a) Energy used across different versions of the game. On average, participants used significantly less energy in V3 (with EBA).

(b) Jobs completed with different versions. V3 participants ran fewer jobs, despite allocation intended to be equivalent to V1 and V2.

(c) Energy use by the jobs completed per user across versions. For any given number of jobs completed, users in V3 used less energy.

Figure 9: Comparison of the energy used during the three different versions of the game.

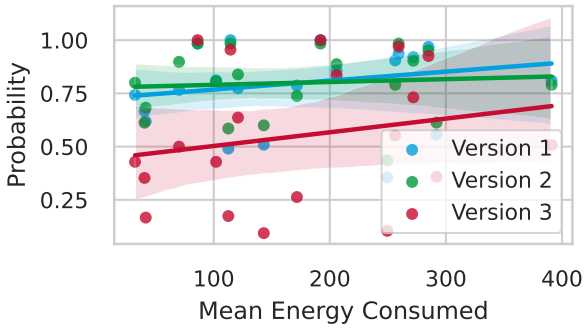


Figure 10: Proportion of participants who ran a job vs. average energy consumed by job. Energy use was not correlated with probability of running a job in any version of the game.

et al. suggest incorporating energy into the cluster scheduling algorithm to prioritize energy-efficient users [21]. Kishwar et al. propose a contract-based mechanism to provide HPC operators flexibility in scheduling jobs by rewarding users, but ignore the effects of renewable electricity generation and of heterogeneous hardware [1]. Fugaku implemented a points based system to incentivize users to enable power-control knobs [52]. Jobs that use less than the "standard power" on a node are rewarded with additional node-hours. Di Pietrantonio et al. suggest that the Thermal Design Power (TDP) of a device can be used to calculate a static cost-ratio between GPU and CPU nodes [16]. Other works have proposed incorporating the price of electricity into the cost of cloud VMs [2, 23, 30, 38]. These works emphasize deferring costs rather than incentivizing sustainability, and do not examine the same trade-offs as here. Margery et al. propose attributing CO_2 emissions to VMs [37].

Sustainable data centers. Prior work examined adapting data-center capacity or demand to reduce power, cost, and carbon emissions [14, 34, 53], for example by scheduling jobs to data-centers currently powered by renewable energy. The Zero-Carbon Cloud project examines running data-centers on "stranded" power—excess power generated when grid supply exceeds load [60]. Such scheduling requires incentives for users to provide jobs that can be delayed or moved.

Green serverless. The problem of emissions has been noted by the FaaS community [41, 48]. GreenCourier implements carbon-aware scheduling of serverless functions based on cluster location [8], and GreenWhisk [47] extends load balancing with the grid's carbon intensity and energy status of off-grid executors. Sharma and Fuerst improve the accuracy of software power meters for serverless functions [49]. Lin and Mohammed similarly propose carbon aware pricing for serverless, but consider optimizations to function configuration rather than heterogeneity between systems [33]. Roy et al. examine the carbon footprint of FaaS, highlighting the impact of keep-alive time [45], but keep-alive time is not applicable to HPC environments, Globus Compute, or *greenACCESS*. EcoLife uses multi-generation hardware to reduce the carbon footprint of serverless platforms [26]. While EcoLife focuses on scheduling and keep-alive time, we focus on user behavior.

9 Summary

The responsibility for sustainable computing is shared by producers and consumers. Yet we find that more than 70% of HPC users remain unaware of the energy consumed by their computations. With the goal of realizing shared responsibility, we introduced Energy- and Carbon-Based Accounting (EBA and CBA), two mechanisms for charging for computing based on environmental impact. We built a prototype FaaS-HPC platform to implement EBA and CBA and demonstrated that both incentivize more sustainable decisions on real hardware. We then used simulations to show that impact-based charging allows an energy-conscious user to process 28% more workload than a performance-focused user with the same allocation. Finally, we demonstrated that users in a simulated environment use less energy under EBA than traditional time-based accounting. **By linking charging to energy use and carbon emissions, EBA and CBA incentivize users to prioritize sustainable computing. Thus our work identifies concrete steps that the HPC community can take to reduce its environmental impact.**

Acknowledgments

Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation. This work used FASTER at Texas A&M University through allocation CIS230030 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by U.S. National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296. The GPU experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

References

[1] Kishwar Ahmed, Jesse Bull, and Jason Liu. 2018. Contract-Based Demand Response Model for High Performance Computing Systems. In *2018 IEEE Intl Conf on Parallel and Distributed Processing with Applications, Ubiquitous Computing and Communications, Big Data and Cloud Computing, Social Computing and Networking, Sustainable Computing and Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*. IEEE, New York, NY, USA, 580–589. https://doi.org/10.1109/BDCloud.2018.00091

[2] Mohammad Aldossary and Karim Djemame. 2018. Energy-based Cost Model of Virtual Machines in a Cloud Environment. In *5th International Symposium on Innovation in Information and Communication Technology*. IEEE, New York, NY, USA, 1–8. https://doi.org/10.1109/ISIICT.2018.8613288

[3] Danny Auble, Thomas Cadeau, Yiannis Georgiou, and Martin Perry. 2013. SLURM Energy Accounting and External Sensors Plug-In. https://slurm.schedmd.com/SUG13/energy_sensors.pdf.

[4] Cédric Augonnet, Samuel Thibault, Raymond Namyst, and Pierre-André Wacrenier. 2011. StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures. *Concurrency and Computation: Practice and Experience, Special Issue: Euro-Par 2009* 23 (2011). Issue 2. https://doi.org/10.1002/cpe.1631

[5] AWS. 2023. *Sustainability Pillar - AWS Well-Architected Framework*. Retrieved February 26, 2024 from https://docs.aws.amazon.com/wellarchitected/latest/sustainability-pillar/sustainability-pillar.html

[6] Noman Bashir, Varun Gohil, Anagha Belavadi Subramanya, Mohammad Shahrad, David Irwin, Elsa Olivetti, and Christina Delimitrou. 2024. The Sunk Carbon Fallacy: Rethinking Carbon Footprint Metrics for Effective Carbon-Aware Scheduling. In *Proceedings of the 2024 ACM Symposium on Cloud Computing* (Redmond, WA, USA) *(SoCC '24)*. Association for Computing Machinery, New York, NY, USA, 542–551. https://doi.org/10.1145/3698038.3698542

[7] Timothy J. Boerner, Stephen Deems, Thomas R. Furlani, Shelley L. Knuth, and John Towns. 2023. ACCESS: Advancing innovation: NSF's advanced cyberinfrastructure coordination ecosystem: Services & support. In *Practice and Experience in Advanced Research Computing* (Portland, OR, USA) *(PEARC '23)*. Association for Computing Machinery, New York, NY, USA, 173–176. https://doi.org/10.1145/3569951.3597559

[8] Mohak Chadha, Thandayuthapani Subramanian, Eishi Arima, Michael Gerndt, Martin Schulz, and Osama Abboud. 2023. GreenCourier: Carbon-Aware Scheduling for Serverless Functions. In *9th International Workshop on Serverless Computing*. 18–23.

[9] Ryan Chard, Yadu Babuji, Zhuozhao Li, Tyler Skluzacek, Anna Woodard, Ben Blaiszik, Ian Foster, and Kyle Chard. 2020. FuncX: A federated function serving fabric for science. In *29th International Symposium on High-Performance Parallel and Distributed Computing* (Stockholm, Sweden) *(HPDC '20)*. Association for Computing Machinery, New York, NY, USA, 65–76. https://doi.org/10.1145/3369583.3392683

[10] Jae-Won Chung, Yile Gu, Insu Jang, Luoxi Meng, Nikhil Bansal, and Mosharaf Chowdhury. 2024. Perseus: Reducing Energy Bloat in Large Model Training. arXiv:2312.06902 [cs.LG] https://arxiv.org/abs/2312.06902

[11] Marcin Copik, Grzegorz Kwasniewski, Maciej Besta, Michal Podstawski, and Torsten Hoefler. 2021. SeBS: A Serverless Benchmark Suite for Function-as-a-Service Computing. In *22nd International Middleware Conference* (Québec city, Canada) *(Middleware '21)*. Association for Computing Machinery, New York, NY, USA, 64–78. https://doi.org/10.1145/3464298.3476133

[12] Marcin Copik, Konstantin Taranov, Alexandru Calotoiu, and Torsten Hoefler. 2023. rFaaS: Enabling High Performance Serverless with RDMA and Leases. In *IEEE International Parallel and Distributed Processing Symposium*. IEEE, New York, NY, USA, 897–907. https://doi.org/10.1109/IPDPS54959.2023.00094

[13] Howard David, Eugene Gorbatov, Ulf R Hanebutte, Rahul Khanna, and Christian Le. 2010. RAPL: Memory power estimation and capping. In *16th ACM/IEEE International Symposium on Low Power Electronics and Design*. IEEE, New York, NY, USA, 189–194.

[14] Wei Deng, Fangming Liu, Hai Jin, Bo Li, and Dan Li. 2014. Harnessing renewable energy in cloud datacenters: Opportunities and challenges. *IEEE Network* 28, 1 (2014), 48–55. https://doi.org/10.1109/MNET.2014.6724106

[15] Department of the Treasury Internal Revenue Service 2023. *How To Depreciate Property*. Department of the Treasury Internal Revenue Service. https://www.irs.gov/pub/irs-pdf/p946.pdf.

[16] Cristian Di Pietrantonio, Christopher Harris, and Maciej Cytowski. 2021. Energy-based Accounting Model for Heterogeneous Supercomputers. *arXiv preprint arXiv:2110.09987* (2021).

[17] Jonathan Eastep, Steve Sylvester, Christopher Cantalupo, Brad Geltz, Federico Ardanaz, Asma Al-Rawi, Kelly Livingston, Fuat Keceli, Matthias Maiterth, and Siddhartha Jana. 2017. Global Extensible Open Power Manager: A Vehicle for HPC Community Collaboration on Co-Designed Energy Management Solutions. In *High Performance Computing*. Springer International Publishing, Cham, 394–412.

[18] Electricity-Maps 2022. Electricity Maps. https://app.electricitymaps.com/map.

[19] Wu-chun Feng and Kirk Cameron. 2007. The Green500 list: Encouraging sustainable supercomputing. *Computer* 40, 12 (2007), 50–55. https://doi.org/10.1109/MC.2007.445

[20] Guillaume Fieni, Romain Rouvoy, and Lionel Seinturier. 2020. SmartWatts: Self-calibrating software-defined power meter for containers. In *20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing* (Melbourne, Australia). IEEE, New York, NY, USA, 479–488. https://doi.org/10.1109/CCGrid49817.2020.00-45

[21] Yiannis Georgiou, David Glesser, Krzysztof Rzadca, and Denis Trystram. 2015. A Scheduler-Level Incentive Mechanism for Energy Efficiency in HPC. In *15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, New York, NY, USA, 617–626. https://doi.org/10.1109/CCGrid.2015.101

[22] Maxime Gonthier, Elisabeth Larsson, Loris Marchal, Carl Nettelblad, and Samuel Thibault. 2024. Data-Driven Locality-Aware Batch Scheduling. In *26th Workshop on Advances in Parallel and Distributed Computational Models*. IEEE, New York, NY, USA. https://inria.hal.science/hal-04500281

[23] Mauro Hinz, Guilherme Piegas Koslovski, Charles C Miers, Laércio L Pilla, and Maurício A Pillon. 2018. A cost model for IaaS clouds based on virtual machine energy consumption. *Journal of Grid Computing* 16 (2018), 493–512.

[24] Torsten Hoefler. 2012. The Green Graph500. http://www.unixer.de/~htor/publications/.

[25] Shixin Ji, Zhuoping Yang, Stephen Cahoon, Alex K Jones, and Peipei Zhou. 2024. SCARIF: Towards Carbon Modeling of Cloud Servers with Accelerators. In *IEEE Computer Society Annual Symposium on VLSI*. IEEE, New York, NY, USA, 1–6.

[26] Yankai Jiang, Rohan Basu Roy, Baolin Li, and Devesh Tiwari. 2024. EcoLife: Carbon-Aware Serverless Function Scheduling for Sustainable Computing. arXiv:2409.02085 [cs.DC] https://arxiv.org/abs/2409.02085

[27] Alok Kamatar, Valerie Hayot-Sasson, Yadu Babuji, Andre Bauer, Gourav Rattihalli, Ninad Hogade, Dejan Milojicic, Kyle Chard, and Ian Foster. 2024. GreenFaaS: Maximizing Energy Efficiency of HPC Workloads with FaaS. *arXiv preprint arXiv:2406.17710* (2024).

[28] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S. Gunawi, Cody Hammock, Joe Mambretti, Alexander Barnes, François Halbach, Alex Rocha, and Joe Stubbs. 2020. Lessons Learned from the Chameleon Testbed. In *USENIX Annual Technical Conference*. USENIX Association, Boston, MA, USA.

[29] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. 2018. RAPL in action: Experiences in Using RAPL for power measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems* 3, 2, Article 9 (mar 2018), 26 pages. https://doi.org/10.1145/3177754

[30] Mascha Kurpicz, Anne-Cécile Orgerie, and Anita Sobe. 2016. How Much Does a VM Cost? Energy-Proportional Accounting in VM-Based Environments. In *24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, New York, NY, USA, 651–658. https://doi.org/10.1109/PDP.2016.70

[31] Klaus-Dieter Lange and Michael G. Tricker. 2011. The design and development of the server efficiency rating tool (SERT). In *2nd ACM/SPEC International Conference on Performance Engineering* (Karlsruhe, Germany) *(ICPE '11)*. Association for Computing Machinery, New York, NY, USA, 145–150. https://doi.org/10.1145/1958746.1958769

[32] Baolin Li, Rohan Basu Roy, Daniel Wang, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. 2023. Toward Sustainable HPC: Carbon Footprint Estimation and Environmental Implications of HPC Systems. In *International Conference for High Performance Computing, Networking, Storage and Analysis* (<conf-loc>, <city>Denver</city>, <state>CO</state>, <country>USA</country>, </conf-loc>) *(SC '23)*. Association for Computing Machinery, New York, NY, USA, Article 19, 15 pages. https://doi.org/10.1145/3581784.3607035

[33] Changyuan Lin and Mohammad Shahrad. 2024. Bridging the Sustainability Gap in Serverless through Observability and Carbon-Aware Pricing. *HotCarbon'24* (2024).

[34] Zhenhua Liu, Minghong Lin, Adam Wierman, Steven Low, and Lachlan L. H. Andrew. 2015. Greening Geographical Load Balancing. *IEEE/ACM Transactions on Networking* 23, 2 (2015), 657–671. https://doi.org/10.1109/TNET.2014.2308295

[35] LUMI 2024. LUMI: Sustainable Future. https://www.lumi-supercomputer.eu/sustainable-future/

[36] Jialun Lyu, Jaylen Wang, Kali Frost, Chaojie Zhang, Celine Irvene, Esha Choukse, Rodrigo Fonseca, Ricardo Bianchini, Fiodar Kazhamiaka, and Daniel S. Berger. 2023. Myths and Misconceptions Around Reducing Carbon Embedded in Cloud Platforms. In *2nd Workshop on Sustainable Computer Systems* (Boston, MA, USA) *(HotCarbon '23)*. Association for Computing Machinery, New York, NY, USA, Article 7, 7 pages. https://doi.org/10.1145/3604930.3605717

[37] David Margery, David Guyon, Anne-Cecile Orgerie, Christine Morin, Gareth Francis, Charaka Palansuriya, and Kostas Kavoussanakis. 2017. A CO2 Emissions Accounting Framework with Market-based Incentives for Cloud Infrastructures. In *6th International Conference on Smart Cities and Green ICT Systems* (Porto, Portugal) *(SMARTGREENS 2017)*. SCITEPRESS - Science and Technology Publications, Lda, Setubal, PRT, 299–304. https://doi.org/10.5220/0006356502990304

[38] Akshay Narayan and Shrisha Rao. 2014. Power-Aware Cloud Metering. *IEEE Transactions on Services Computing* 7, 3 (2014), 440–451. https://doi.org/10.1109/TSC.2013.22

[39] PassMark. 2024. CPU Benchmarks. https://www.cpubenchmark.net/

[40] Tirthak Patel, Adam Wagenhäuser, Christopher Eibel, Timo Hönig, Thomas Zeiser, and Devesh Tiwari. 2020. What does Power Consumption Behavior of HPC Jobs Reveal?: Demystifying, Quantifying, and Predicting Power Consumption Characteristics. In *IEEE International Parallel and Distributed Processing Symposium*. IEEE, New York, NY, USA, 799–809. https://doi.org/10.1109/IPDPS47924.2020.00087

[41] Panos Patros, Josef Spillner, Alessandro V Papadopoulos, Blesson Varghese, Omer Rana, and Schahram Dustdar. 2021. Toward sustainable serverless computing. *IEEE Internet Computing* 25, 6 (2021), 42–50.

[42] J. Gregory Pauloski, Valerie Hayot-Sasson, Maxime Gonthier, Nathaniel Hudson, Haochen Pan, Sicheng Zhou, Ian Foster, and Kyle Chard. 2024. TaPS: A Performance Evaluation Suite for Task-based Execution Frameworks. In *IEEE 20th International Conference on e-Science*. IEEE, New York, NY, USA, 1–10. https://doi.org/10.1109/e-Science62913.2024.10678702

[43] Thanh-Phuong Pham, Juan J. Durillo, and Thomas Fahringer. 2020. Predicting Workflow Task Execution Time in the Cloud Using A Two-Stage Machine Learning Approach. *IEEE Transactions on Cloud Computing* 8, 1 (2020), 256–268. https://doi.org/10.1109/TCC.2017.2732344

[44] Ana Radovanovic. 2020. *Our data centers now work harder when the sun shines and wind blows*. Google. Retrieved February 26, 2024 from https://blog.google/inside-google/infrastructure/data-centers-work-harder-sun-shines-wind-blows/

[45] Rohan Basu Roy, Raghavendra Kanakagiri, Yankai Jiang, and Devesh Tiwari. 2024. The Hidden Carbon Footprint of Serverless Computing. In *Proceedings of the 2024 ACM Symposium on Cloud Computing* (Redmond, WA, USA) *(SoCC '24)*. Association for Computing Machinery, New York, NY, USA, 570–579. https://doi.org/10.1145/3698038.3698546

[46] Norbert Schmitt, Lukas Iffländer, André Bauer, and Samuel Kounev. 2019. Online power consumption estimation for functions in cloud applications. In *IEEE International Conference on Autonomic Computing*. IEEE, Umea, Sweden, 63–72. https://doi.org/10.1109/ICAC.2019.00018

[47] Jayden Serenari, Sreekanth Sreekumar, Kaiwen Zhao, Saurabh Sarkar, and Stephen Lee. 2024. GreenWhisk: Emission-aware computing for serverless platform. *arXiv preprint arXiv:2409.03029* (2024).

[48] Prateek Sharma. 2023. Challenges and opportunities in sustainable serverless computing. *ACM SIGENERGY Energy Informatics Review* 3, 3 (2023), 53–58.

[49] Prateek Sharma and Alexander Fuerst. 2024. Accountable Carbon Footprints and Energy Profiling For Serverless Functions. In *Proceedings of the 2024 ACM Symposium on Cloud Computing* (Redmond, WA, USA) *(SoCC '24)*. Association for Computing Machinery, New York, NY, USA, 522–541. https://doi.org/10.1145/3698038.3698531

[50] Software Carbon Intensity (SCI) Specification 2024. Software Carbon Intensity (SCI) Specification. https://sci.greensoftware.foundation/

[51] Solem, Ask and Goel, Vineet. 2019. *Faust User Manual*. Robinhood Markets. https://faust.readthedocs.io/en/latest/.

[52] Ana Luisa Veroneze Solórzano, Kento Sato, Keiji Yamamoto, Fumiyoshi Shoji, Jim M. Brandt, Benjamin Schwaller, Sara Petra Walton, Jennifer Green, and Devesh Tiwari. 2024. Toward Sustainable HPC: In-Production Deployment of Incentive-Based Power Efficiency Mechanism on the Fugaku Supercomputer. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, New York, NY, USA, 1–16. https://doi.org/10.1109/SC41406.2024.00030

[53] Thanathorn Sukprasert, Abel Souza, Noman Bashir, David Irwin, and Prashant Shenoy. 2023. Quantifying the Benefits of Carbon-Aware Temporal and Spatial Workload Shifting in the Cloud. arXiv:2306.06502 [cs.DC]

[54] sustainability-report-aws 2022. 2022 Amazon Sustainability Report. https://sustainability.aboutamazon.com/2022-sustainability-report.pdf.

[55] sustainability-report-aws-google 2023. Google Environmental Report 2023. https://www.gstatic.com/gumdrop/sustainability/google-2023-environmental-report.pdf.

[56] Muhammad Tirmazi, Adam Barker, Nan Deng, Md E. Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. 2020. Borg: the next generation. In *Proceedings of the Fifteenth European Conference on Computer Systems* (Heraklion, Greece) *(EuroSys '20)*. Association for Computing Machinery, New York, NY, USA, Article 30, 14 pages. https://doi.org/10.1145/3342195.3387517

[57] Logan Ward. 2021. ML-in-the-loop molecular design with Parsl. https://github.com/ExaWorks/molecular-design-parsl-demo/tree/main. Accessed: 2024-01-24.

[58] Philipp Wiesner, Ilja Behnke, Dominik Scheinert, Kordian Gontarska, and Lauritz Thamsen. 2021. Let's wait awhile: How temporal workload shifting can reduce carbon emissions in the cloud. In *22nd International Middleware Conference* (Québec city, Canada) *(Middleware '21)*. Association for Computing Machinery, New York, NY, USA, 260–272. https://doi.org/10.1145/3464298.3493399

[59] Daniel C. Wilson, Siddhartha Jana, Aniruddha Marathe, Stephanie Brink, Christopher M. Cantalupo, Diana R. Guttman, Brad Geltz, Lowren H. Lawson, Asma H. Al-rawi, Ali Mohammad, Fuat Keceli, Federico Ardanaz, Jonathan M. Eastep, and Ayse K. Coskun. 2021. Introducing application awareness into a unified power management stack. In *IEEE International Parallel and Distributed Processing Symposium*. IEEE, Virtual, 320–329. https://doi.org/10.1109/IPDPS49936.2021.00040

[60] Fan Yang and Andrew A. Chien. 2016. ZCCloud: Exploring Wasted Green Power for High-Performance Computing. In *IEEE International Parallel and Distributed Processing Symposium*. IEEE, New York, NY, USA, 1051–1060. https://doi.org/10.1109/IPDPS.2016.96

[61] Jie You, Jae-Won Chung, and Mosharaf Chowdhury. 2023. Zeus: Understanding and Optimizing GPU Energy Consumption of DNN Training. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 119–139. https://www.usenix.org/conference/nsdi23/presentation/you

Appendix: Artifact Description/Artifact Evaluation

Artifact Description (AD)

A Overview of Contributions and Artifacts

In this work, we propose energy-based and carbon-based accounting to incentivize users to prioritize efficiency in HPC environments.

A.1 Paper's Main Contributions

C_1 A large-scale survey of 300+ HPC users focused on their awareness of energy and carbon use.
C_2 Energy- and Carbon-Based Accounting mechanisms for charging for computing based on environmental impact.
C_3 A comparison of EBA and CBA on diverse CPUs and GPUs.
C_4 A user study in which users interact with impact-based accounting via a web-based simulation game

A.2 Computational Artifacts

The computational artifacts of the paper are provided in a single mono-repo (https://doi.org/10.5281/zenodo.16943778, https://github.com/AK2000/core-hours-artifact/releases/tag/sc25-release). For clarity, we decompose the mono-repo into its components:

A_1 Survey, Analysis Scripts and Data (survey/)
A_2 Prototype HPC-FaaS platform using CBA. This artifact contains two components available in two folders: The Green-Access Service and Experiments (green-ACCESS/green-access), and the Endpoint (green-ACCESS/green-faas-endpoint).
A_3 GPU measurements (gpu-experiments/)
A_4 EBA and CBA Simulations (batch-simulator/)
A_5 User Study Game Interface, Data and Analysis Scripts (carbon-credits-game/)

The table shows how each artifact is related to the contributions of this work, and the elements in the paper.

Artifact ID	Contributions Supported	Related Elements
A_1	C_1	Figures 1-2
A_2	C_2, C_3	Figure 4, Tables 2, 5
A_3	C_3	Table 4
A_4	C_2	Figures 5, 6, 7, Table 6
A_5	C_4	Figure 8-10

B Artifact Identification

B.1 Computational Artifact A_1

Artifact A_1 includes the survey questionaire, the scripts used for analysis, and anonymized and aggregate data from the responses collected. This is primarily a data artifact.

Relation To Contributions

The survey is the first study to reveal how HPC users think about energy and sustainability of their computation (C_1). This is both a resource for the community, and a motivation for tying sustainability to the cost of computation (C_2).

Expected Results

We include the survey that was administered, as well as the aggregate data of the responses received. The analysis scripts will produce Figures 1 and 2 in the paper, as well as the data that is used in Section 2.

Expected Reproduction Time (in Minutes)

Less than 10 minutes.

Artifact Setup (incl. Inputs)

Hardware. The analysis scripts were run on a commodity desktop with an Intel Core i7-10700 CPU and 16 GB Memory. Any commodity machine should be able to support the analysis.

Software. The scripts require Python 3.10.13 and Jupyter Notebook.

Datasets/Inputs. The responses from the survey are included in the artifact, and are the only data needed.

Installation and Deployment. No installation required.

Artifact Execution

To execute the analysis script, open the `Analysis.ipynb` notebook and run all cells.

Artifact Analysis (incl. Outputs)

The notebook will produce figures identical to the paper. These results confirm the conclusions in the paper that users are not aware of their energy, and do not prioritize efficiency.

B.2 Computational Artifact A_2

Artifact A_2 is the prototype of the GreenAccess platform. The platform setup includes deploying the web-service, and deploying multiple endpoints across heterogeneous hardware.

Relation To Contributions

This experiment is used to demonstrate traditional accounting models may incentivize inefficiency compared to EBA and CBA. Specifically, we illustrate an example where using a machine with a faster runtime leads to much higher energy consumption. In this example, current accounting mechanisms charge less for this endpoint even though it is inefficient.

Expected Results

First, a user should be able to deploy FaaS endpoints to their own infrastructure and monitor the energy consumption while executing functions. Second, on any heterogeneous infrastructure, we expect differences in performance and energy consumption based off the computational characteristics of the functions being run, similar to Figure 4 in the paper. Finally, if using the same hardware as the paper, the energy and performance trade-offs of the provided Cholesky function should mirror the trade-offs seen in Table 2.

Expected Reproduction Time (in Minutes)

Approximately 1.5 hours: 45 minutes to configure and deploy the service, and 45 minutes to run experiments.

Artifact Setup (incl. Inputs)

Hardware. The GreenAccess server was deployed on a Chameleon Cloud KVM Large instance with 4 vCPUs and 8GBs of RAM. In addition, GreenAccess requires deploying a function as a service execution endpoint on multiple servers with different hardware characteristics. We expect the artifacts to be functional on any server with an Intel CPU as recent as Cascadelake or any AMD CPU as recent as Zen3. To produce the results in the article, we used four servers with the following configurations:

- **Desktop:** Intel Core i7-10700, 16 GB Memory
- **Cascade Lake:** 2 × Intel Xeon 6248R, 192 GB Memory,
- **Ice Lake:** 2 × Intel Platinum 8390, 256 GB Memory
- **Zen3:** 2 ×AMD EPYC 7763, 256 GB Memory

In the rest of this description we refer to the KVM instance as the "service node" and each of the servers as "endpoints". All experiments were launched from the service node.

Software. The service node was configured with Ubuntu 20.04. We used Python 3.10.13 which was installed inside a Conda environment (conda 24.1.2). A complete list of the Python packages used is provided in the artifact (`reproducibility/requirements.txt`). The service also uses a MySQL (version 8.0.36) database to store allocation information, and Redis (version 7.2.4) for in-memory communication between processes.

Each endpoint was configured with Ubuntu 22.04. We use Python 3.10 installed inside a Conda environment. A complete conda specification is provided in the repository (`reproducibility/endpoint_environment.yaml`). The endpoint also requires libpfm4 (https://sourceforge.net/projects/perfmon2/files/libpfm4/libpfm-4.13.0.tar.gz), which was built with gcc 11.4.1.

Datasets / Inputs. No dataset or inputs are required for this artifact.
Installation and Deployment.
Deploying the artifact consists of 3 steps:

T1: Deploy GreenAccess Web Service by cloning the repository, connecting and authenticating with Kafka, and running the Faust application for online analysis of collected data. Then start the web service using `uvicorn`.

T2: Deploy GreenAccess endpoints on each of the systems listed above using the provided conda environment. Authenticate each endpoint with Globus Auth.

T3: Configure the experiment file with the URL of the deployed service and the IDs of the newly created endpoints.

Artifact Execution

The experiment can be launched with the command:

```
$ green-access-experiment profile \
    --config <path to config>  -n 32
```

The parameter *n* is used to launch 32 repetitions of each function.

Artifact Analysis (incl. Outputs)

The experiment will populate the MySQL database with information about the runtime and energy use of each function. To fetch these results, run,

```
$ python green_access/experiments/fetch_jobs.py \
    <database connection string>
```

which will output a file call `job_log.csv`. This file is used by `notebooks/Plot Results.ipynb`, where the average of all executions are used to create Table 2. Then, the carbon attributed to each function is re-calculated using different methods, and the averages are reported in Table 5.

B.3 Computational Artifact A_3

Artifact A_3 measures how execution time and energy consumption for the Cholesky factorization vary across different generations of GPUs.

Relation To Contributions

This set of experiments demonstrates how different generations of GPUs, running the same workload under the same runtime system, can exhibit a trade-off between execution time and energy. Newer GPUs deliver higher performance but consume more energy. We then show that our EBA and CBA policies can balance a good trade-off between the throughput of modern GPUs and the reduced energy consumption of older GPUs.

Expected Results

Newer generation GPU should complete the application faster. Increasing the number of GPU should reduce the execution time. Older generation GPU should consume less energy than newer ones.

Expected Reproduction Time (in Minutes)

Approximately 15 minutes to set up and install the runtime, and 3 hours to run the application on all three type of GPUs.

Artifact Setup (incl. Inputs)

Hardware. We expect that any system with Nvidia GPU will output similar result. The compute nodes we used were hosted on the Grid'5000 resource provider and are as follows:

- **P100:** one node with 2 x Nvidia Tesla P100-PCIE, 16 GB Memory
- **V100:** one node with 8 x Nvidia Tesla V100-SXM2, 32 GB Memory
- **A100:** one node with 8 x Nvidia A100-SXM4, 40 GB Memory

Software. The runtime we used is StarPU version 1.4 (https://gitlab.inria.fr/starpu/starpu). The compute nodes ran on Debian 5.10.209-2 and we compiled StarPU with cmake version 3.18.4.

Datasets/Inputs. StarPU contains examples that generate matrices to be used for Cholesky decomposition. These examples are compiled when installing StarPU, so no additional data is needed.

Installation/Deployment. Deployment consists of two steps. First, installing StarPU on the target system, and second, calibrating the StarPU performance models. Building StarPU uses `autotools` as well as other packages. Use the `Install_packages.sh` script from our repository to install the necessary dependencies. Then, run `autogen.sh` from the StarPU repository to configure the build environment. Calibrating the performance model is done by running warm-up iterations of the Cholesky kernel with GPUs being evaluated.

Artifact Execution

The workflow consists of three tasks. T_1 runs the Cholesky factorization on the compute node through a bash script. For each different type of GPU, T_1 is repeated. T_2 collects runtime and energy-consumption data (in our case, we have a command-line specific to the Grid'5000 resource provider). T_3 processes those data and parses them to produce the values shown in Table 4.

Artifact Analysis (incl. Outputs)

The file `gpu_experiments/results.csv` will contain the parsed results of the experiments, specifically the execution time, the energy consumed, and the number of GPUs used.

B.4 Computational Artifact A_4

Artifact A_4 is the simulator used to evaluate EBA and CBA.

Relation To Contributions

The simulator demonstrates that on a real job trace, EBA rewards users who choose the most energy-efficient machines (C_1, C_2). This illustrates the proposed accounting method incentivizes the behavior we desire, as well as illustrates the effects that the accounting scheme may have on how each machine is used. We extend the analysis to CBA by simulating the differences between EBA and CBA (C_1). We also simulate the cost of CBA with low-carbon electricity and discuss the relationship to sustainability (C_3).

Expected Results

The simulation results should match those presented in Section IV of the article.

Expected Reproduction Time (in Minutes)

Approximately 1 hour. 20 minutes for setup, 25 minutes to preprocess the data and 10 minutes for execution and analysis.

Artifact Setup (incl. Inputs)

Hardware. We expect that any system equipped with the proper software can run this simulation. The system we used was a Dell Inc. Precision 3580 with 16GB of memory and an Intel Core i7-1355U. The operating system used was a 64-bit Ubuntu 22.04.1.

Software. We used Python 3.10.12 for the scripts and gcc 11.4.0 and GNU Make 4.3 for the simulator written in C. A complete Python requirements file is included in the artifact (`requirements.txt`).

Datasets / Inputs. Two datasets are needed:

(1) The job trace with power consumption information, available for download from (https://zenodo.org/records/3666632).
(2) The dataset collected from Desktop, Theta, IC, and FASTER needed for extrapolation of the workload to different machines, available in the repo.
(3) The hourly carbon trace from 2023 for US-MISO, US-ERCOT, US-PJM, DK-BHM, CA-ON, AU-SA grids, sourced from electricity maps, and *not* redistributed. These are available to download with a free account.

Installation and Deployment. The simulator is included in the artifact repository under the `batch-simulator/` folder. In the `batch-simulator/` folder, run `make -C src/` to build the simulator.

Artifact Execution

T1: Pre-process the input datasets to remove unwanted data and jobs without power values. Download the job trace files and unpack them. This may take 20+ minutes. Replace `path` with the folder path in which the data has been unpacked and run the following script:

```
batch-simulator/Simulator$ bash parse_database.sh path
```

This will output the files `Simulator/inputs/workload/meggie-raw` and `Simulator/inputs/workload/emmy-raw`.

T2: Interpolate the pre-processed trace files to the unseen machines. Run the script,

```
batch-simulator/Simulator$ python3 \
  extrapolate_job_trace.py
```

This will create a new file called `job-trace-extrapolated.csv`. Depends on **T1**.

T3: Reformat the data for use in the simulator using,

```
batch-simulator/Simulator$ bash write_workload_all.sh
```

that gives the file `meggie_and_emmy_count_from_database` that will be used as an input in the next step. Depends on **T2**. This data file has been committed to the repository.

Tasks **T4-6** are automated with the script:

```
batch-simulator/Simulator$ bash plot_all.sh
```

which performs the operations below:

T4: Run the simulation with EBA. Depends on **T3**.
T5: Run the simulation with CBA. Depends on **T3**.
T6: Run the simulation with CBA in a low-carbon scenario. Depends on **T3**.

Artifact Analysis (incl. Outputs)

To analyze the results use the file `Simulator/notebooks/PlotResults.ipynb`. You will see plots for Figure 5,6, and 7. The `plot_all.sh` script should have created data in the `plot/` folder for Table 6.

B.5 Computational Artifact A_5

Computational artifact A_5 is the web-based user study, including the responses and scripts used to analyze the results.

Relation To Contributions

The web-based game mimics the trade-offs that HPC users see when choosing resources in order to asses how changing the information and the cost of resources affects behavior. This study validates that (1) more information does not lead users to change their behavior, and (2) impact-based pricing incentivizes users to choose more sustainable resources.

Expected Results

The artifact allows you to reconstruct the analysis from the data collected during the user study.

Expected Reproduction Time (in Minutes)

5 minutes is necessary to reconstruct the plots.

Artifact Setup (incl. Inputs)

Hardware. The web game was served from a Chameleon Cloud KVM Large instance with 4 vCPUs and 8GBs of RAM. Analysis of the results was performed on a Desktop with an Intel Core i7-10700 and 16 GB Memory.

Software. The game is built using JavaScript and served using Python 3.10.12 and Flask (v3.1) and stores data into a MongoDB instance.

Datasets/Inputs. The data file for the jobs used in the game is included in the repository and served as a static file. The responses from the game saved from the database and used for analysis are also included in the artifact.

Installation and Deployment. Only `jupyter notebook` is required for analysis.

Artifact Execution

The responses are produced by a user study, so there is no execution of this artifact.

Artifact Analysis (incl. Outputs)

The analysis of the results is done by running all cells in the `PlotResults.ipynb` notebook. This will generate Figures 9 and 10 in the paper.

Artifact Evaluation (AE)

C.1 Computational Artifact A_1

Artifact A_1 is the survey and responses.

Artifact Setup (incl. Inputs)

(1) Download the aggregated responses from the dataset Zenodo link (also included in the repository).
(2) Install Jupyter notebook and the required analysis libraries (pandas, seaborn).

Artifact Execution

As the artifact only includes analysis of the survey results, no additional execution is required.

Artifact Analysis (incl. Outputs)

Run all cells in the `PlotAggregateResults.ipynb` notebook. The artifact also includes a `SurveyAnalysis.ipynb` which was used to process the raw results into the data in the paper. This notebook will not be runnable as we agreed to only share aggregate results in our IRB. The artifact also includes `aggregate_responses.py` which was used to compile the raw results into the aggregate results. This will also not be runnable.

The output of the notebook will produce figures 1 and 2 in the paper.

C.2 Computational Artifact A_2

Artifact A_2 is the GreenAccess service.

Artifact Setup (incl. Inputs)

Deploy GreenAccess Service (30 minutes)

T1: Clone the GreenAccess repository and install the Python package by running,

```
$ pip install .
```

inside the `service/` folder.

T2: Authenticate with `diaspora-event-sdk` and create topics for producing events. In a Python terminal, run:

```
from diaspora_event_sdk import Client
c = Client()
c.register_topic(
    "ga-resources-<user>"
)
c.register_topic(
    "ga-predictions-<user>"
)
```

Running those commands will require authentication with Globus Auth. Once authenticated, we create Kafka topics that are used for passing resource information from the endpoints to the service.

T3: Start MySQL with default configuration and create database and user. Once credentials have been created, Update service/-common.py with the connection string.

T4: Start Redis with default configuration.

T5: To start the faust consumer, run

```
$ faust -A monitor -l info worker
```

T6: To start the web service, run

```
$ uvicorn router:app
```

Deploy a GreenAccess Endpoint: (20 minutes each)

T1: Install libpfm4 (https://sourceforge.net/projects/perfmon2/files/libpfm4/libpfm-4.13.0.tar.gz).

T2: Install `diaspora-event-sdk` and authenticate. To produce to the Kafka topics, we are first required to authenticate this machine with the Diaspora Event Fabric. Documentation on this authentication can be found here: https://github.com/globus-labs/diaspora-event-sdk/blob/main/docs/troubleshooting.md#key-migration. The AWS keys created on the service node must be copied from the service node to the endpoint using the command

```
c.put_secret_key(
    "<access-key>",
    "<secret-key>",
    "<brokers>"
)
```

T3: Install GreenAccess Endpoint conda environment from the repository by running

```
$ conda env create \
    -f reproducibility/endpoint_environment.yml
```

This environment will install the Green-Access endpoint.

T4: Configure the endpoint. First run,

```
$ globus-compute-endpoint configure <endpoint name>
```

where endpoint name is any display name. This will create a file `/.globus_compute/<endpoint name>/config.yaml`. Replace this file with `reproducibility/endpoint_config.py` from the GreenAccess service repository. The configuration will need to be updated to set the number of workers equal to the number of cores on the machine used. In addition, replace the kafka topics inside of the configuration with the kafka topics that were created above.

T5: Start the endpoint with the command:

```
$ globus-compute-endpoint start <endpoint name>
```

T6: Update the configuration file with the endpoint ID from above.

Artifact Execution

The experiment can be launched with the command:

```
$ green-access-experiment profile \
    --config <path to config> \
    -n 32
```

This command submits 32 repitions of each FaaS functions defined in the `experiments` folder to the `GreenAccess` service.

Artifact Analysis (incl. Outputs)

The experiment will populate the MySQL database with information about the runtime and energy use of each function. To fetch these results, run,

```
$ python green_access/experiments/fetch_jobs.py \
    <database connection string>
```

which will output a file call `job_log.csv`. This file is used by `notebooks/Plot Results.ipynb`, where the average of all executions are used to create Table 2. Then, the carbon attributed to each function is re-calculated using different methods, and the averages are reported in Table 5.

The expected results depend on the system used for evaluation. At a minimum, you should observe a variation in the profiled power use by function type.

C.3 Computational Artifact A_3

Artifact A_3 is the GPU measurements.

Artifact Setup (incl. Inputs)

T1: Install dependencies:

```
<repo>/gpu_experiments $ Install_packages.sh
```

T2: Build StarPU

```
~/$ git clone https://gitlab.inria.fr/starpu/starpu.git
~/$ cd starpu/
~/starpu$ ./autogen.sh
~/starpu$ ./configure \
    --prefix=/path/to/starpu/folder/starpu \
    --enable-maxcudadev=<NUM_GPUS>
~/starpu$ make -j 100
~/starpu$ make install
~/starpu$ sudo ldconfig
```

Artifact Execution

Calibration must be done on every new compute node before running experiments. Run the command below:

```
~$ source <repo>/gpu_experiments/config_env.sh
~$ cd starpu/
~/starpu$ ./examples/cholesky/cholesky_implicit -size \
    $((2880*50)) -nblocks $((50)) -niter 5
```

To start the experiments do:

```
~/$ bash core-hours-artifact/gpu_experiments/run.sh \
    <NUMGPUS>
```

Replace <NUMGPUS> with the number of GPUs you want to evaluate.

Artifact Analysis (incl. Outputs)

If you are using Grid'5000 (as we did), you can gather metrics like this:

```
$ curl 'https://api.grid5000.fr/stable/sites/<site>/\
    metrics?job_id=<job_id>' > data.txt
$ python3 parse_bmc.py data.txt results.csv
```

And replace with the site used and <job_id> with your job's ID. This will report the average time to run the Cholesky experiment along with the average power collected from the BMC during the run. To calculate the figures shown in the paper, you need to input the TDP, age, FLOPS and embodied carbon of your device. The calculation of the cost from the time is shown in the notebook `PlotResults.ipynb`

C.4 Computational Artifact A_4

Artifact A_4 is the simulation.

Artifact Setup (incl. Inputs)

The simulator is included in the artifact at `batch-simulator/`. The workload dataset is included in the repository. The carbon traces should be downloaded and placed in the Simulator/inputs/carbon directory.

In the batch-simulator/Simulator folder, run

```
$ make -C src/
```

to build the simulator.

To prepare the Python environment run

```
$ pip install Simulator/requirements.txt
```

Artifact Execution

T1: Pre-process the input datasets to remove unwanted data and jobs without power values. Download the job trace files and unpack them. This may take 20+ minutes. Replace `path` with the folder path in which the data has been unpacked and run the following script:

```
batch-simulator/Simulator$ bash parse_database.sh path
```

This will output the files `Simulator/inputs/workload/meggie-raw` and `Simulator/inputs/workload/emmy-raw`.

T2: Interpolate the pre-processed trace files to the unseen machines. Run the script,

```
batch-simulator/Simulator$ python3 \
    extrapolate_job_trace.py
```

This will create a new file called `job-trace-extrapolated.csv`. Depends on **T1**.

T3: Reformat the data for use in the simulator using,

```
batch-simulator/Simulator$ bash write_workload_all.sh
```

that gives the file `meggie_and_emmy_count_from_database` that will be used as an input in the next step. Depends on **T2**.

Tasks **T4-6** are automated with the script:

```
batch-simulator/Simulator$ bash plot_all.sh
```

which performs the operations below:

T4: Run the simulation with EBA. Depends on **T3**.

T5: Run the simulation with CBA. Depends on **T3**.

T6: Run the simulation with CBA in a low-carbon scenario. Depends on **T3**.

Artifact Analysis (incl. Outputs)

In the folder `batch-simulator/Simulator/plot/` you will find the figures named after Table 6 from the paper because we just reported the values into a table instead of showing the figures. To analyze the results, run the file `Simulator/notebooks/PlotResults.ipynb`. Figures 5, 6 and 7, will be in folder `Simulator/notebook/`.

C.5 Computational Artifact A_5

Artifact A_5 is the user study. We detail how to reproduce the figures from the collected data.

Artifact Setup (incl. Inputs)

Download the collected data from the artifact repository.

Artifact Execution

There is no additional execution.

Artifact Analysis (incl. Outputs)

Run `Analysis.ipynb` to create Figures 9 and 10 from the aggregate data.