The Illusion of Reduction: How Software Carbon Metrics Mask Global Footprints

PRATEEK SHARMA, Indiana University, United States
ABEL SOUZA, University of California, Santa Cruz, United States
MOHAMMAD SHAHRAD, University of British Columbia, Canada
CHANGYUAN LIN, University of British Columbia, Canada

While carbon-focused computing has emerged as a key approach for reducing the environmental footprint of cyberinfrastructure, widely used metrics that simultaneously incorporate operational and embodied impacts struggle to capture the behavior of modern, multi-layered, and multiplexed systems. Due to implicit assumptions in current carbon accounting methodologies, application-level optimizations do not necessarily induce proportional—or even monotonic—reductions in system-level carbon emissions, revealing a mismatch and lack of causal alignment between local actions and global outcomes. We identify a set of anomalies and inconsistencies spanning both embodied and operational carbon accounting that arise from under-specified system models and attribution mechanisms that rely on implicit assumptions. We synthesize these observations into a taxonomy of carbon measurement *paradoxes*, providing a foundation for more principled, transparent, and causally meaningful approaches to sustainable system design. To address these issues, we argue that carbon metrology must explicitly incorporate additional context information: how unused resources are accounted for, measurement boundaries, and workload and system elasticity. We also include a Carbon Footprint Context Card as a metadata template to support comparable carbon reporting.

CCS Concepts: • **Social and professional topics** → **Sustainability**; **Sustainability**; • **Hardware** → **Impact on the environment**; **Impact on the environment**; • **General and reference** → **Metrics**; **Metrics**; • **Computer systems organization** → *Cloud computing*.

1 Introduction

By 2030, data centers are estimated to consume more than 20% of U.S. electricity—placing a significant burden on the electrical grid and regular consumers [51]. To highlight and reduce the environmental impact of this cyberinfrastructure, *metrics* for capturing the carbon footprint of software applications will continue to play an important role. As suggested by the greenhouse gas protocol (GHG), carbon footprints of applications also include the environmental impact of the lifecycle of the hardware they run on [42]. Hardware manufacturing is an extremely energy and resource intensive process, with several hundreds of kg of CO_2 embedded in a typical server [23]. This accounting approach of including both the operational emissions (due to energy consumption) and the embodied emissions is widely adopted, including in the Software Carbon Intensity (SCI) metric, which has become an ISO standard for software carbon accounting [21].

Such holistic metrics are a double-edged sword. They expose new tradeoffs between embodied and operational emissions, opening up new avenues for carbon optimizations. For example, compute jobs can be scheduled based on the grid's carbon intensity, and in locations with more efficient hardware and lower embodied costs [56]. For many applications and workflows, traditional resource management and scheduling techniques, such as elastic scaling, have been shown to effectively reduce carbon emissions by considerable margins [25, 53]. Such techniques have been developed for a wide range of application classes, such as Machine Learning (ML) training [25], inference [58], serverless functions [22], web services [53], etc.

Compared to conventional metrics like performance, carbon metrics have a broader scope beyond the application, since they incorporate external system-level context such as hardware embodied emissions. While resource-management optimizations mentioned earlier may reduce carbon at the application-level, their impact on *system-level* outcomes and their generalizability across environments remains unclear. In this paper, we scrutinize carbon metrics and their associated metrology through a differential profiling approach, and investigate how optimization-induced changes in application-level carbon footprints affect the encapsulating system. Our approach is inspired by causal performance analysis [13], where the underlying principle is that performance improvements in one component (such as a thread) may not necessarily improve the overall performance of complex software systems.

Using a combination of analytical models and empirical measurements, we investigate the *causal strength* of different carbon optimizations. We find that both embodied and operational carbon accounting have many intricacies and implicit assumptions, and standard accounting methodology assumptions lead to a divergence between application and system-wide footprints. Because embodied costs are assigned proportionally to resource allocation, in under- or over-committed systems, this can lead to carbon not being conserved. Because of this lack of conservation, optimizations such as server consolidation and application right-sizing may end up reducing footprints at the application-level, not necessarily at the system-level. Additionally, carbon accounting amortizes past emissions into future workloads assuming a fixed hardware lifetime: this can lead to "sunk costs" impacting scheduling decisions [6].

Operational emissions are not immune to measurement and accounting challenges either, as energy is a shared resource among multiple applications and hardware components. In many scenarios, "local" energy reductions may not translate to equivalent global reductions due to the non-linearities in power conversion and distribution, and other physical infrastructure (such as cooling). Through empirical analysis, we show that energy measurement boundaries play a crucial role: whether energy is measured at the CPU, server, or wall-outlet level can have a significant impact on the absolute

Authors' Contact Information: Prateek Sharma, Indiana University, Bloomington, IN, United States, prateeks@iu.edu; Abel Souza, University of California, Santa Cruz, Santa Cruz, CA, United States, absouza@ucsc.edu; Mohammad Shahrad, University of British Columbia, Vancouver, BC, Canada, mshahrad@ece.ubc.ca; Changyuan Lin, University of British Columbia, Vancouver, BC, Canada, chlin@ece.ubc.ca.

and relative footprints after optimization. For example, while traditional power measurements often focus on the CPU, other system components and power conversion losses can increase the actual/-global power consumption several-fold. Most systems are designed to operate at maximum efficiency at high loads, and reducing the power consumption worsens the power-supply efficiency. This is applicable to data centers, as well as individual servers, where we show the reduced impact of vertical hardware scaling.

Ultimately, the goal of carbon metrics is to provide the right incentives to reduce the carbon footprint of the cyberinfrastructure. Our investigation reveals multiple gaps in current carbon metrology practices, making it challenging to formulate and test carbon optimizations. These gaps are due to simplifications, implicit assumptions about workloads and environmental context, under-specification, and fundamental challenges in energy attribution, and broadly occur in carbon metrics (not just SCI). Most critically, because carbon metrics are *under-specified*, they require researchers and practitioners to make their own important assumptions, often implicitly, leading to inconsistencies in measurement and comparison.

Our goal is not to introduce new metrics—the intertwined anomalies suggest there is unlikely to be a one-size-fits-all solution. Instead, we aim to highlight the pitfalls and paradoxes of carbon measurement, educate the broader community to be attentive to the context of reported statistics and cautious about end-to-end effects, and ultimately motivate—and potentially guide—discussions toward improved measurement protocols for more consistent carbon metrology. As a concrete first step towards comparable carbon footprint reporting, we propose using Carbon Footprint Context Cards [**?**], which record the metadata required to correctly interpret a reported footprint, including attribution policies, measurement boundaries, data sources, and elasticity information.

2 Background

We consider two broad use-cases for carbon measurements: absolute quantification and relative comparison. Carbon accounting for these two can be relevant to various stakeholders in the supply chain, for instance, to i) software/service users and developers, ii) systems software and middleware operators, iii) infrastructure providers and cloud platforms, and iv) regulators and policymakers.

For the absolute quantification use-case, the goal is to provide an estimate of the total carbon footprint of the system (such as an application or an entire cloud platform). These carbon footprints are sometimes intended to increase awareness and transparency of an application's sustainability credentials. Increasingly, however, these are and will be needed for regulatory compliance purposes, such as the European Union's Corporate Sustainability Reporting Directive (CSRD) [17]. Since 2018, hyperscalers such as Amazon, Google, Meta, and Microsoft have been providing a yearly, coarse-grained breakdown of their total estimated emissions [1, 2]. For applications, quantifying their carbon footprint supports monitoring dashboards, raises awareness of emissions from commonly used software, and can ultimately enable developer-driven sustainability optimizations. Carbon monitoring dashboards have been valuable in highlighting the growing emissions of applications such as AI and video streaming, and in comparing these impacts to other human activities, such as driving [37].

For the rest of the paper, we use *application* to represent the entity whose footprint is being reported, such as a software, service, workflow, VM, container, or job. We use *system* to denote the larger boundary that hosts these applications, such as a server, cluster, or data center.

2.1 Software Carbon Intensity

For ease of exposition, we will focus on Software Carbon Intensity (SCI), which has been ratified as an ISO standard in March 2024 [28]. Based on the key principles behind the GHG protocol [41], it aims to be comprehensive in its carbon accounting, and it thus considers both the embodied and operational carbon emissions.

Embodied. SCI considers the embodied emission $\mathcal{E}$ of the underlying hardware platform. This is amortized over the lifetime L of the hardware, yielding the embodied "rate" $\mathcal{E}/L$. SCI attributes the embodied emissions *proportional* to the application's time and resource share of the hardware. If an application reserves r amount of the hardware of size R, and this reservation is for a time-duration t, then its embodied fraction is: $\frac{\mathcal{E}}{L}\frac{r}{R}$.

The attributes of $\mathcal{E}$ and L are hardware dependent. The embodied carbon $\mathcal{E}$ is mainly the carbon footprint associated with the manufacturing and disposal of the hardware components. For semiconductor hardware such as CPUs, SSDs, and memory, this embodied cost can be significant. For example, for mobile devices it can represent more than 50% of its total lifetime emissions (including using it over a 3-year period) [23]. It is often obtained from life-cycle-analysis (LCA) techniques [32], which aggregate the manufacturing emissions of various hardware components (such as CPU, memory, etc.). The lifetime L is usually around 5 years.

Operational. The operational part of the SCI is the application's energy (i.e., electricity) consumption. The energy consumption J is converted to carbon emissions using the grid's carbon emissions factor γ. The energy is empirically measured and can be decomposed to the average power W and the running time t of the application. SCI then adds the embodied and operational parts of the footprint:

$$\text{SCI} = \text{Embodied} + \text{Operational} = \frac{\mathcal{E}}{L} \cdot \frac{r}{R} \cdot \tau \; + \; W \cdot \gamma \cdot t \qquad (1)$$

While not all prior work on carbon accounting uses SCI explicitly, the underlying attribution policies are similar and use the same principles. For embodied emissions, attributing proportional to the share of resource allocation is common [3, 6–8, 24, 36, 43, 44, 48, 52, 54], and in some cases a 100% allocation is assumed [5, 16, 18, 26]. Operational emissions are often decomposed into an active component with consumption-based attribution and a static component with allocation-based attribution [33, 36].

3 Carbon Optimizations

The spatiotemporal flexibility of a software provides a broad set of control actions: resource-management mechanisms such as scaling (including hardware frequency scaling), migration, and scheduling, that operate over time, space, or both. This has led to the development of carbon-optimized versions of different applications [14,

35, 46, 53, 59], and general-purpose carbon-aware scheduling policies [8, 22, 54, 56, 58]. As such, these are all "local" optimizations and are tied to specific applications and deployment boundaries.

By contrast, "global" optimizations are operations with a larger scope and have the potential to affect the global metric of interest. In the performance realm, for example, service request latency serves as a key performance metric. For a typical distributed application composed of many micro-services, the global metric would be the average end-to-end latency for certain workflows, or alternatively, the expected latency weighted by workflow popularity. For instance, adding items to the shopping cart and searching would be two such workflows. In the above scenario, the local-vs-global question is: does improving the latency of a particular service improve the expected end-to-end latency? The strength of causality between the impact of optimizations on local and global metrics can be highly variable. For example, the service in question may not be on the critical path or be blocked by another slower service "in parallel". This has led to a range of causal-profiling [13] approaches, where the impact of individual components of the program (such as functions) is computed on the overall performance of the program.

Often, local optimizations are easier to perform and evaluate: the metrics can be more easily computed, which results in a tighter optimization loop. Thus, in this paper, we seek to answer the broader question: *what is the strength of causality for carbon optimizations?* A local optimization may be applying a vertical-scaling mechanism [25, 52], which leads to a reduction in application-level carbon footprint. In the context of distributed and shared computing infrastructures (such as cloud platforms), we want to understand how the different classes of carbon optimization may affect the global cloud or even data-center level carbon footprint.

3.1 Is it necessary to reduce global and system-wide carbon metrics?

Ideally, any local optimization for emissions, whether undertaken by clients, developers, or providers, should align with the broader goal of reducing global carbon output. This principle follows from the need for incentives to support core decarbonization objectives, including permanence (ensuring that emissions reductions persist over time rather than being deferred or reversed), conservation (avoiding shifts in emissions to other locations, actors, or time periods, also known as leakage), and additionality (ensuring that reductions would not have occurred in the absence of the intervention). In practice, however, such alignment is not always pursued or consistently achieved. Consider, for example, a public-sector organization such as a university or hospital that tracks only Scope 1 and 2 emissions. For them, offloading workloads to a remote cloud provider would appear to lower their reported emissions, since the associated carbon is now accounted for elsewhere. Yet at a global scale, this shift may actually increase total emissions. A similar effect arises when providers or researchers attempt to "game" emissions reporting models. For instance, they may assume that servers older than a fixed depreciation window incur zero embodied emissions and advocate for their continued use on that basis, even though doing so can increase actual energy consumption and total emissions due to reduced hardware efficiency.

We argue that any emission metric should ultimately serve the goal of global carbon reduction. Focusing only on local or accounting-based gains risks creating incentives that appear beneficial on paper but are counterproductive in aggregate. This framing guides our treatment of which emission sources and system boundaries can be safely abstracted or ignored when using such metrics.

4 Embodied Carbon Anomalies

The inclusion of embodied carbon in the total footprint can lead to many scenarios with inconsistent measurement outcomes, or the expected decrease in emissions after an intervention is different than expected. This section uncovers and categorizes such anomalies.

4.1 Carbon Conservation

Consider a simple scenario in which an application runs in a virtual machine (VM). This VM is allocated *half* of the resources of the underlying server (i.e., half the CPU, memory, I/O, etc.). The server is not hosting any other VM, which leaves half of its resources unallocated. This scenario thus looks at the relationship between the application's ($C(A)$) and system's carbon footprint ($C(S)$) respectively, when the system is underutilized and has surplus resources, a common occurrence in shared infrastructure.

Let us construct the carbon footprint of the application and the system for this scenario. The application is the VM, and the encompassing system is the server. For operational carbon, we have $O(A) = O(S)$, i.e., the system's operational footprint is determined entirely by this application. We assume there are no other overheads or losses in the virtualization layer: i.e., in this scenario, the hypervisor's energy contribution is negligible and ignored. This assumption is only for ease of exposition, and later scenarios will deal with the implications of system management overheads.

The embodied part is interesting. Since most carbon accounting metric (e.g., SCI) allocate embodied costs based on the *allocation*, and the application is allocated half of the resources, we have $E(A) = \frac{1}{2}E(S)$. Combining both the dimensions, and expressing in system quantities: we have $C(A) = O(S) + \frac{1}{2}E(S)$, and $C(S) = O(S) + E(S)$. Thus, in this under-allocation scenario, we have $C(A) < C(S)$. Unless the provider assumes responsibility for the remaining emissions (in this case $\frac{1}{2}E(S)$) the conservation principle [57] is violated.

Key takeaway. Under allocation-based embodied carbon accounting, *carbon is not necessarily conserved*: even when all system components are fully accounted for, application-level and system-level carbon footprints can diverge substantially.

4.1.1 Why does carbon conservation matter? The lack of conservation is due to "holes" in resource allocation, which are pervasive in shared environments. Resources can be under-allocated due to insufficient demands [12], or providers can keep spare resources to provide near-instantaneous on-demand resource allocation. Fully allocating all resources is also fundamentally challenging: stranded resources occur due to multi-dimensional bin-packing.

Validating Footprints. Conservation facilitates easier validation of footprints through multiple ways of accounting. Energy is typically measured with the top-down approach, where it is measured at the

server hardware level, and then disaggregated to the resident applications. That is, we start by measuring $C(S)$ first, and then partitioning it to $C(A_i)$. Conversely, the bottom-up approach aggregates fine-grained measurements (such as CPU RAPL [30, 60]) of all constituent components. Because energy is conserved, both the top-down and bottom-up approaches can be deployed, for improving power estimates, as well as for validating and testing the estimate. This is a common way for evaluating power profiling tools: the per-application energy is aggregated and compared against the server-level ground-truth measurement [4, 10, 15, 19, 20, 29, 31, 39, 40, 45, 55, 61]. However, when carbon is not conserved, it becomes challenging to obtain consistent carbon footprints and validate them.

Causality. Conservation also leads to causality. If $C(A)$ is always equal to $C(S)$, then for any intervention/optimization, $\Delta C(A) \rightarrow \Delta C(S)$. Without the conservation, there can be other scenarios (which we show later), in which $\Delta C(A) \neq \Delta C(S)$.

4.1.2 Optimizations affected. With embodied carbon, the efficiency of resource allocation governs the gap between application and system-wide footprints. As a result, optimizations which improve allocation-efficiency affect $C(A)$ and $C(S)$ differently, and may not have the intended outcome. One such common optimization is to **right-size** the server. In the cloud context, it entails running the VM on an appropriately-sized physical server to reduce the amount of wasted/unused resources. Suppose we 'fit' the application VM into a perfectly sized server where all its resources are allocated to the VM. In this case, we get $C'(A) = C'(S)$. Because we have switched to a smaller server, it decreases: $C'(S) < C(S)$, and we get a real reduction in carbon emissions, i.e., $\Delta C(S) < 0$, and we have realized system-wide savings as a result of this optimization.

But what about the application? Its footprint does not change, i.e., $C(A) = C'(A)$. The operational cost remains the same in both cases. Initially on the original server we have $E(A) = \frac{1}{2}E(S)$ since it gets half the server allocation. After moving to the new half-sized server, we have $E'(A) = E'(S) = \frac{1}{2}E(S) = E(A)$. Thus, $\Delta C(A) = 0$, and the application has no incentive to participate in right-sizing.

4.2 Sunk-Costs

Temporal and spatial shifting of workloads is a major category of carbon optimizations. In temporal shifting, the job execution is delayed from the original execution time with carbon intensity γ_1, to a different time with lower intensity γ_2. A similar difference in intensity also drives spatial job migration (move the job to a lower carbon intensity location). Recent work has shown that naively incorporating embodied emissions into scheduling decisions (or any operational decision such as spatial workload shifting) can lead to the sunk carbon fallacy [6]. This issue can be mitigated either by focusing solely on operational carbon or by consistently accounting for the embodied emissions of all unselected alternatives [6].

We argue that the sunk carbon fallacy can be perceived more clearly from the lens of carbon conservation. Consider a simple scenario involving server-consolidation where VMs are moved to a smaller number of physical servers. Consider two application VMs running on two servers with each using half the server resources, and then packed into a single server. The system comprises of two servers S_1 and S_2, and $C(S) = C(S_1) + C(S_2)$. The initial total system-wide footprint is $C(S) = (E_1 + E_2) + (O_1 + O_2)$ (time and γ omitted since they are invariant in this scenario). After we consolidate, the total embodied and operational emissions remain the same: embodied carbon E_1 and E_2 was already emitted when the servers were manufactured, and so $C'(S) = C(S)$. Application footprints tell a similar story. $C(A_1) = E_1/2 + O_1$, and $C(A_2) = E_2/2 + O_2$. After consolidating, from the applications perspective, the system has shrunk, so $C'(A_1) = E_1/2 + O_1$, and $C'(A_2) = E_2/2 + O_2$. The application footprint has also not changed, since $C(A_1) + C(A_2) = C'(A_1) + C'(A_2)$.

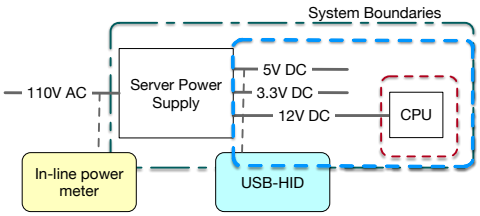

Fig. 1. System boundaries for power/energy measurements significantly impact the operational footprints.

Key takeaway. Carbon conservation holds across abstraction boundaries – system-level and application-level footprints remain consistent – which exposes the sunk carbon fallacy: consolidating workloads cannot reduce total carbon footprint because embodied carbon is already spent at manufacturing time.

5 Operational Carbon Accounting Challenges

We now investigate common scenarios and optimizations to reveal the gaps between local and global operational emissions. Since the operational footprint is energy times the carbon intensity, our focus will be on energy. Energy flows through many shared hardware and software components in complex cyberinfrastructure. From a system-control perspective, this results in a large amount of buffering/capacitance, causing lag in the system response. That is, interventions and optimizations which locally reduce energy consumption may not immediately result in a corresponding decrease in system-wide energy.

5.1 Energy Measurement Boundaries and Conversion Overheads

The energy-measurement boundary determines both the absolute operational footprint and the relative impact of optimizations. Consider the simplest case of measuring the power consumption of a single application running on a server. The AC power is first converted by the power supply unit (PSU) to different DC voltages for the different hardware components, as shown in Figure 1.

The input AC power is measured by a SpecPower-approved power meter (Instek GPM 8310) which we read at 4Hz via the meter's telnet/SCPI interface. The PSU is a Corsair HX1000i which provides output current information on each of the three voltage rails (3.3V, 5V, and 12V). The system has a AMD Ryzen 5 2400G CPU and 64 GB

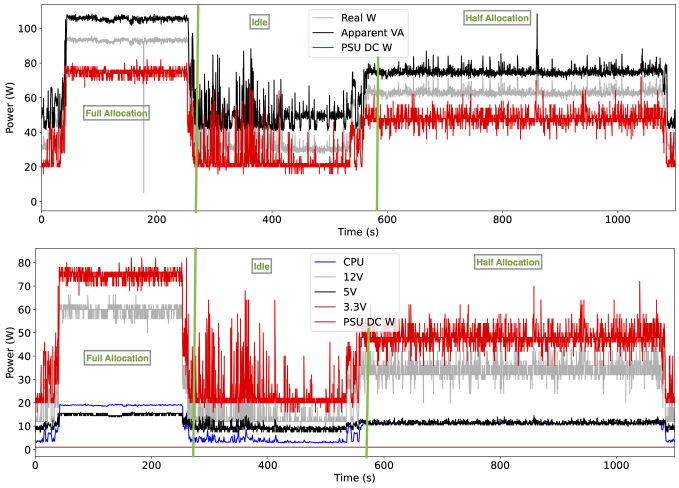

Fig. 2. Power consumption during kernel-compile and idling. The CPU, DC, and AC power consumption all differ significantly due to system-load, which non-linearly affects PSU conversion and cooling overhead.

of DDR4 RAM, and the OS is Ubuntu 24.04—CPU power is measured through the RAPL interface.

The power conversion and delivery hardware is not 100% efficient—resulting in larger system-wide energy consumption. PSU conversion overheads are an important, and often overlooked, source of non-linearity, since the PSU efficiency-vs-load curves are inverted U-shaped. Desktop PSUs obtain a peak efficiency of around 85% at 85% of the maximum output load, and decline to 80% efficiency at higher and lower loads. The conversion efficiency (ratio of output DC power to the input AC power) is especially poor at low loads, often dipping to 60 – 80% at loads of 20% and less [9, 11, 27], which has implications for both measurement and optimization. In our own measurements (Figure 2), the efficiency was even lower.

The absolute and relative differences between the different power measurement boundaries for a three-part workload is shown in Figure 2. In the middle idle part, the system is idle, but power consumption varies from 3W to more than 30W depending on the measurement boundary. At this low load, the PSU also has a poor power factor, and the apparent power [1] is 50W. In the third phase, we run kernel-compile on only half the CPU resources. This is an example of vertical scaling to reduce power consumption. Compared to the full allocation case, the CPU power is halved (10 instead of 20W), but the AC power only drops from 110 to 77W because of the fixed idle power. This again illustrates that the change in application footprints can be different from the system-level footprint.

5.2 Non-linear Power Dynamics

Multiple factors contribute to energy-consumption nonlinearities, where the change in application-level energy ($\Delta(A)$) does not cause an equal change in the system-wide energy ($\Delta(S)$). This makes establishing the causal strength of power optimizations even more challenging.

[1]Total power flowing through alternating current (AC).

CPU. CPU power non-linearities due to frequency scaling are well known—however, auxiliary cooling and power-delivery components can also have a significant contribution. To empirically illustrate this, we run the kernel-compile benchmark with two different resource configurations. In the first case (also the first phase in Figure 2), we use all the 8 cores of the CPU and 8 compilation threads, representing an application that has 100% of the server. In the second case (third phase in the figure), we allocate half the resources (i.e., 4 threads, with the rest of the 4 cores idle).

The CPU power consumption is the stand-in for application footprint, and for the system footprint, we use the AC real power. In the full-allocation case, we get $\Delta(A) = 20 - 3 = 17W$, and $\Delta(S) = 90 - 30 = 60W$. In other words, the system-level footprint rose nearly 3.5× higher than the application-level CPU power—illustrating the non-proportional power consumption.

Memory. Another contributor to non-linear power dynamics is DRAM memory, which also has a high idle power consumption [34]. In Figure 2, the DRAM is powered by the 5V line, and consumes 15W during the full-allocation and idling at around 8W. Since DRAM contributes to a large fraction of the data center power consumption, reducing its use can yield significant power benefits. To emulate this, we conduct an experiment where we incrementally add 16GB DIMMs to the motherboard, starting at 16 and ending at 64GB. The power consumption for this memory scaling experiment is shown in Figure 3—an idle workload is used in all cases. The change in power is minimal—when going from 64 to 16GB, the reduction in AC power is only 2W. DDR4 specification indicates around 2W per 16GB DIMM, and adding 3 DIMMs should have increased the power consumption by 6W. This is another example of the system-wide change being less than the component-level change, indicative of high system capacitance and inertia. This affects energy-saving optimizations, which reduce memory consumption, like in [43]—the decrease in system-wide energy is much lower than expected.

Data centers also have load-dependent efficiency. As observed in [50], the data center power usage effectiveness (PUE) improves with increasing IT load. Data center power delivery and cooling infrastructure is engineered to have maximum efficiency near the peak load, and has high fixed costs at low loads. Due to this, the PUE of the state-of-the-art academic MGHPCC data center improved from 1.5 to 1.3 as the IT load increased from 0.6 to 1 MW. Failing to account for this nonlinearity can undermine carbon optimizations such as migrating workloads to lower-carbon locations: the resulting reduction in energy use at the source data center may be noticeably smaller than the energy footprint of the migrated VM.

Key takeaway. Power measurement boundaries significantly affect operational footprints, by as much as 10×. Non-linear power efficiencies are pervasive and highly load-dependent—which makes causal strength of energy optimizations challenging to ascertain.

6 Toward Comparable Carbon Footprint Reporting

The anomalies we discussed in §4 and §5 show that carbon reporting numbers can be misleading when reported without their system context. SCI is simple and *under-specified*—which has made it easy to compute and adopt. However, it also requires many implicit assumptions, which lead to the anomalies we have identified. To help resolve the anomalies and inconsistencies, this section provides

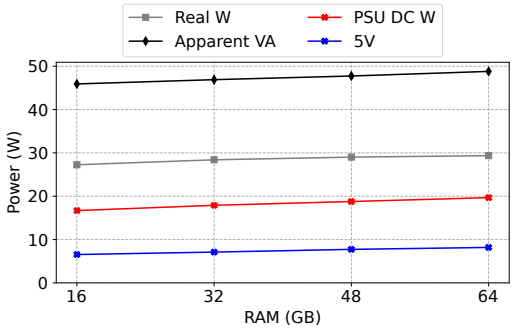

Fig. 3. The increase in power consumption on systems with different physical memory capacity. The observed increase is much lower than expected (8W for DDR4 memory).

some guidelines which could be adopted in the near-term, and proposes some directions for longer-term improvements in carbon and energy accounting.

For capturing the measurement and accounting assumptions, we propose that SCI-type footprints be accompanied by **C**arbon **F**ootprint **C**ontext **C**ards (CFCC). Similar to model cards for machine learning models which provide model capabilities [38], these context cards can provide measurement context and interpretability to their accompanying footprints. These context cards include the key embodied and operational accounting assumptions (shown in Figure 4 and elaborated in [49]). These cards can be "filled in" whenever the footprint is computed—akin to artifact evaluation metadata and scripts required for many computer systems research publications [47].

For reporting use-cases, we have distilled it to five questions covering the embodied attribution, operational measurements, and data provenance. For computer-systems researchers developing carbon or energy optimizations, the last question (Q6) is intended to address the strength of causality between local and global reductions. Context cards are not limited to just serving as post-facto metadata; they can also guide the carbon metrology and create more comprehensive and explicit measurement plans.

7 Conclusion

Through first-principles and empirical measurements, we have identified multiple inconsistencies and anomalies, where carbon-reducing optimizations applied to an application do not have an equivalent system-wide effect. We posit that one of the fundamental issues is that the SCI-type metrics are *underspecified*. They require many implicit assumptions to be made, especially when used in modern shared cyberinfrastructure, which causes the footprints to diverge and be inconsistent.

As a first step to avoid these common pitfalls, carbon accounting should incorporate additional meta-data and the system context to disambiguate key assumptions about unallocated and shared resources and measurement boundaries. The Carbon Footprint Context Card is one potential way to expose these assumptions as carbon reporting metadata. CFCC's provide an incremental way to improve carbon accounting and can be used in a tiered reporting model: higher levels require progressively richer metadata, enabling and incentivizing organizations to mote from basic carbon-intensity estimates to more transparent, reproducible, and optimization-friendly measurements.

Carbon Footprint Context Card

(1) How are hardware embodied emissions attributed to applications?
- ☐ Proportional to allocations
- ☐ Proportional to usage
- ☐ Other (describe if applicable): _____

(2) Who is responsible for embodied emissions of unallocated resources? (choose one option and describe)
- ☐ Tenant(s): _____
- ☐ Host: _____
- ☐ Unaccounted (justify): _____

(3) At what level is power measured? (all that apply)
- ☐ Sub-chip power domain ☐ Chip ☐ Board
- ☐ Node/motherboard ☐ rack ☐ Cluster ☐ Data center

(4) How are emissions from shared services and components reported? Describe the methodology (if applicable).
- ☐ Not applicable (no external dependencies)
- ☐ Fair sharing of local software components (e.g., OS): _____
- ☐ Fair sharing of shared network hardware and services: _____
- ☐ Software production emissions included (e.g., amortized model training costs): _____

(5) What are the underlying information and data sources for carbon and energy?
- – Carbon intensity provider: _____
- – Carbon intensity methodology: ☐ Avg. ☐ Marginal
- – Power monitoring interface (e.g., Intel RAPL): _____
- – Embodied carbon sources: _____

If accompanying an emissions-reducing solution:

(6) Is resource allocation affected by your solution (e.g., VM right-sizing, spatial or temporal shifting, scaling)?
- ☐ Yes (allocation reduction/increase)
- ☐ No (e.g., improved energy management on a device)

(6a) If yes, what is the demand elasticity of the system?
- ☐ Unknown (risk of reporting false gains)
- ☐ Linear
- ☐ Sub-linear
- ☐ Exponential
- ☐ Other (estimated): _____

Fig. 4. Carbon Footprint Context Card for reporting measurement assumptions and methodology. The card records attribution policies, measurement boundaries, data sources, shared-component accounting, and elasticity assumptions that affect the interpretation and comparison of the reported carbon footprint.

Acknowledgments. This work is supported in part by an NSF CAREER award CNS-2340722 and by the Natural Sciences and Engineering Research Council of Canada (NSERC), through research grants RGPIN-2021-03714, DGECR-2021-00462, and the Canada Graduate Scholarship (CGS D) program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Google 2023 environmental report. https://sustainability.google/reports/google-2023-environmental-report/, 2023.

[2] Meta 2023 sustainability report. https://sustainability.fb.com/2023-sustainability-report/, 2023.

[3] Hanan Awwad, Changyuan Lin, Rabab Ward, and Mohammad Shahrad. Estimating the carbon footprint of serverless functions on a public cloud platform. In *Proceedings of the 3rd Workshop on SErverless Systems, Applications and MEthodologies*, pages 12–20, 2025.

[4] Timur Babakol, Anthony Canino, Khaled Mahmoud, Rachit Saxena, and Yu David Liu. Calm energy accounting for multithreaded Java applications. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 976–988, Virtual Event USA, November 2020. ACM.

[5] Michail Bachras and Hans-Arno Jacobsen. Beyond performance: Measuring the environmental impact of analytical databases. *arXiv preprint arXiv:2504.18980*, 2025.

[6] Noman Bashir, Varun Gohil, Anagha Belavadi Subramanya, Mohammad Shahrad, David Irwin, Elsa Olivetti, and Christina Delimitrou. The sunk carbon fallacy: Rethinking carbon footprint metrics for effective carbon-aware scheduling. In *Proceedings of the 2024 ACM Symposium on Cloud Computing*, pages 542–551, 2024.

[7] Will Buchanan, John Foxon, Daniel Cooke, Sangeeta Iyer, Elizabeth Graham, Bill DeRusha, Christian Binder, Kin Chiu, Laura Corso, Henry Richardson, et al. Carbon-aware computing. 2023.

[8] Mohak Chadha, Thandayuthapani Subramanian, Eishi Arima, Michael Gerndt, Martin Schulz, and Osama Abboud. Greencourier: Carbon-aware scheduling for serverless functions. In *Proceedings of the 9th International Workshop on Serverless Computing*, pages 18–23, 2023.

[9] CLEAResult. Check all 80 plus(r) certified psus. https://www.clearesult.com/80plus/certified-psus/all-certified-psus, Accessed Jun 2026.

[10] Maxime Colmant, Romain Rouvoy, Mascha Kurpicz, Anita Sobe, Pascal Felber, and Lionel Seinturier. The next 700 CPU power models. *Journal of Systems and Software*, 144:382–396, October 2018.

[11] Corsair. Psu: Efficiency ratings explained. https://help.corsair.com/hc/en-us/articles/14641912717453-PSU-Efficiency-Ratings-Explained, 2023.

[12] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 153–167, 2017.

[13] Charlie Curtsinger and Emery D Berger. Coz: Finding code that counts with causal profiling. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 184–197, 2015.

[14] Yi Ding and Tianyao Shi. Sustainable llm serving: Environmental implications, challenges, and opportunities. In *2024 IEEE 15th International Green and Sustainable Computing Conference (IGSC)*, pages 37–38. IEEE, 2024.

[15] Thanh Do, Suhib Rawshdeh, and Weisong Shi. pTop: A Process-level Power Profiling Tool. *HotPower*, page 5, 2009.

[16] Jesse Dodge, Taylor Prewitt, Remi Tachet des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A Smith, Nicole DeCario, and Will Buchanan. Measuring the carbon intensity of ai in cloud instances. In *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, pages 1877–1894, 2022.

[17] European Financial Reporting Advisory Group (EFRAG). European sustainability reporting standard e1. https://www.efrag.org/sites/default/files/media/document/2025-12/November_2025_ESRS_E1.pdf, Accessed Jan 2026.

[18] Brad Everman, Trevor Villwock, Dayuan Chen, Noe Soto, Oliver Zhang, and Ziliang Zong. Evaluating the carbon impact of large language models at the inference stage. In *2023 IEEE international performance, computing, and communications conference (IPCCC)*, pages 150–157. IEEE, 2023.

[19] Guillaume Fieni, Romain Rouvoy, and Lionel Seinturier. SmartWatts: Self-Calibrating Software-Defined Power Meter for Containers. *arXiv:2001.02505 [cs]*, January 2020. arXiv: 2001.02505.

[20] Guillaume Fieni, Romain Rouvoy, and Lionel Seiturier. SelfWatts: On-the-fly Selection of Performance Events to Optimize Software-defined Power Meters. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 324–333, Melbourne, Australia, May 2021. IEEE.

[21] Green Software Foundation. GSF Releases Alpha Version of SCI Specification. https://greensoftware.foundation/articles/the-green-software-foundation-releases-alpha-version-of-software-carbon-intensity.

[22] Viktor Urban Gsteiger, Pin Hong Long, Yiran Sun, Parshan Javanrood, and Mohammad Shahrad. Caribou: Fine-grained geospatial shifting of serverless applications for sustainability. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, pages 403–420, 2024.

[23] Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S. Lee, David Brooks, and Carole-Jean Wu. ACT: designing sustainable computer systems with an architectural carbon modeling tool. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pages 784–799, New York New York, June 2022. ACM.

[24] Leo Han, Jash Kakadia, Benjamin C Lee, and Udit Gupta. Fair-co2: Fair attribution for cloud carbon emissions. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, pages 646–663, 2025.

[25] Walid A Hanafy, Qianlin Liang, Noman Bashir, David Irwin, and Prashant Shenoy. Carbonscaler: Leveraging cloud workload elasticity for optimizing carbon-efficiency. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 7(3):1–28, 2023.

[26] Geerd-Dietger Hoffmann and Verena Majuntke. Improving carbon emissions of federated large language model inference through classification of task-specificity. *ACM SIGENERGY Energy Informatics Review*, 4(5):44–50, 2024.

[27] Intel. Atx version 3 multi rail desktop platform power supply design guide. https://www.intel.com/content/www/us/en/content-details/336521/atx-version-3-multi-rail-desktop-platform-power-supply-design-guide.html, 2023.

[28] ISO Central Secretary. Information technology — software carbon intensity (sci) specification. Standard ISO/IEC 21031:2024, International Organization for Standardization, Geneva, CH, 2024.

[29] Mathilde Jay, Vladimir Ostapenco, Laurent Lefevre, Denis Trystram, Anne-Cécile Orgerie, and Benjamin Fichel. An experimental comparison of software-based power meters: focus on CPU and GPU. In *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 106–118, May 2023.

[30] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. RAPL in Action: Experiences in Using RAPL for Power Measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 3(2):1–26, June 2018.

[31] Kashif Nizam Khan, Filip Nyback, Zhonghong Ou, Jukka K. Nurminen, Tapio Niemi, Giulio Eulisse, Peter Elmer, and David Abdurachmanov. Energy Profiling Using IgProf. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 1115–1118, Shenzhen, China, May 2015. IEEE.

[32] Randolph E Kirchain Jr, Jeremy R Gregory, and Elsa A Olivetti. Environmental life-cycle assessment. *Nature Materials*, 16(7):693–697, 2017.

[33] Sven Köhler, Benedict Herzog, Henriette Herzog, Manuel Vögele, Lukas Wenzel, Andreas Polze, and Timo Hönig. Carbon-aware memory placement. *ACM SIGENERGY Energy Informatics Review*, 4(3):39–45, 2024.

[34] Seunghak Lee, Ki-Dong Kang, Hwanjun Lee, Hyungwon Park, Younghoon Son, Nam Sung Kim, and Daehoon Kim. Greendimm: Os-assisted dram power management for dram with a sub-array granularity power-down state. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 131–142, 2021.

[35] Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. Sprout: Green generative ai with carbon-efficient llm inference. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21799–21813, 2024.

[36] Changyuan Lin and Mohammad Shahrad. Bridging the sustainability gap in serverless through observability and carbon-aware pricing. *ACM SIGENERGY Energy Informatics Review*, 4(5):120–126, 2024.

[37] Stephen Makonin, Laura U Marks, Radek Przedpełski, Alejandro Rodriguez-Silva, and Ramy ElMallah. Calculating the carbon footprint of streaming media: Beyond the myth of efficiency. In *Eighth workshop on computing within limits 2022*. LIMITS, 2022.

[38] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019.

[39] Lev Mukhanov, Pavlos Petoumenos, Zheng Wang, Nikos Parasyris, Dimitrios S. Nikolopoulos, Bronis R. De Supinski, and Hugh Leather. ALEA: A Fine-Grained Energy Profiling Tool. *ACM Transactions on Architecture and Code Optimization*, 14(1):1–25, April 2017.

[40] Adel Noureddine, Romain Rouvoy, and Lionel Seinturier. A review of energy measurement approaches. *ACM SIGOPS Operating Systems Review*, 47(3):42–49, 2013.

[41] Greenhouse Gas Protocol. Greenhouse Gas Protocol. https://ghgprotocol.org/, Accessed May 2025.

[42] Barath Raghavan and Justin Ma. The energy and emergy of the internet. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks - HotNets '11*, pages 1–6, Cambridge, Massachusetts, 2011. ACM Press.

[43] Rohan Basu Roy, Raghavendra Kanakagiri, Yankai Jiang, and Devesh Tiwari. The hidden carbon footprint of serverless computing. In *Proceedings of the 2024 ACM Symposium on Cloud Computing*, pages 570–579, 2024.

[44] Andreas Schmidt, Gregory Stock, Robin Ohs, Luis Gerhorst, Benedict Herzog, and Timo Hönig. carbond: An operating-system daemon for carbon awareness. *ACM SIGENERGY Energy Informatics Review*, 4(3):52–57, 2024.

[45] Norbert Schmitt, Lukas Iffländer, André Bauer, and Samuel Kounev. Online Power Consumption Estimation for Functions in Cloud Applications. In *2019 IEEE International Conference on Autonomic Computing (ICAC)*, pages 63–72, June 2019. ISSN: 2474-0756.

[46] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Communications of the ACM*, 63(12):54–63, November 2020.

[47] Mohammad Reza Saleh Sedghpour, Alessandro Vittorio Papadopoulos, Cristian Klein, and Johan Tordsson. Artifact evaluation for distributed systems: Current practices and beyond, 2024.

[48] Prateek Sharma and Alexander Fuerst. Accountable carbon footprints and energy profiling for serverless functions. In *Proceedings of the 2024 ACM Symposium on Cloud Computing*, pages 522–541, 2024.

[49] Prateek Sharma, Changyuan Lin, Abel Souza, and Mohammad Shahrad. Carbon footprint context cards: A metadata framework for software carbon accounting. *Second Workshop on Measurements, Modeling, and Metrics for Carbon-Aware Computing (CarbonMetrics)*, 2026.

[50] Prateek Sharma, Patrick Pegus II, David Irwin, Prashant Shenoy, John Goodhue, and James Culbert. Design and operational analysis of a green data center. *IEEE Internet Computing*, 21(4):16–24, 2017.

[51] Arman Shehabi, Sarah Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Inês Azevedo, and William Lintner. *United States Data Center Energy Usage Report.* Number LBNL–1005775, 1372902. December 2024.

[52] Abel Souza, Noman Bashir, Jorge Murillo, Walid Hanafy, Qianlin Liang, David Irwin, and Prashant Shenoy. Ecovisor: A virtual energy system for carbon-efficient applications. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 252–265, 2023.

[53] Abel Souza, Shruti Jasoria, Basundhara Chakrabarty, Alexander Bridgwater, Axel Lundberg, Filip Skogh, Ahmed Ali-Eldin, David Irwin, and Prashant Shenoy. Casper: Carbon-aware scheduling and provisioning for distributed web services. In *Proceedings of the 14th International Green and Sustainable Computing Conference*, pages 67–73, 2023.

[54] Thanathorn Sukprasert, Abel Souza, Noman Bashir, David Irwin, and Prashant Shenoy. On the limitations of carbon-aware temporal and spatial workload shifting in the cloud. In *Proceedings of the Nineteenth European Conference on Computer Systems*, pages 924–941, 2024.

[55] Linus Wagner, Maximilian Mayer, Andrea Marino, Alireza Soldani Nezhad, Hugo Zwaan, and Ivano Malavolta. On the Energy Consumption and Performance of WebAssembly Binaries across Programming Languages and Runtimes in IoT. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, pages 72–82, Oulu Finland, June 2023. ACM.

[56] Philipp Wiesner, Ilja Behnke, Dominik Scheinert, Kordian Gontarska, and Lauritz Thamsen. Let's wait awhile: how temporal workload shifting can reduce carbon emissions in the cloud. In *Proceedings of the 22nd International Middleware Conference*, pages 260–272, 2021.

[57] Wikipedia. Conservation law. https://en.wikipedia.org/wiki/Conservation_law/, Accessed May 2026.

[58] Li Wu, Walid A Hanafy, Abel Souza, Khai Nguyen, Jan Harkes, David Irwin, Mahadev Satyanarayanan, and Prashant Shenoy. Carbonedge: Leveraging mesoscale spatial carbon-intensity variations for low carbon edge computing. In *Proceedings of the 34th International Symposium on High-Performance Parallel and Distributed Computing*, pages 1–13, 2025.

[59] Kaiqiang Xu, Decang Sun, Han Tian, Junxue Zhang, and Kai Chen. {GREEN}: Carbon-efficient resource scheduling for machine learning clusters. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*, pages 999–1014, 2025.

[60] Huazhe Zhang and Henry Hoffmann. A Quantitative Evaluation of the RAPL Power Control System. *Feedback computing*, page 6, 2015.

[61] Xusheng Zhang, Ziyu Shen, Bin Xia, Zheng Liu, and Yun Li. Estimating Power Consumption of Containers and Virtual Machines in Data Centers. In *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 288–293, September 2020. ISSN: 2168-9253.