Fair-CO$_2$: Fair Attribution for Cloud Carbon Emissions

Leo Han
lxh4@cornell.edu
Cornell Tech
New York, New York, USA

Jash Kakadia
jkakadia@seas.upenn.edu
University of Pennsylvania
Philadelphia, Pennsylvania, USA

Benjamin C. Lee
leebcc@seas.upenn.edu
University of Pennsylvania
Philadelphia, Pennsylvania, USA

Udit Gupta
ugupta@cornell.edu
Cornell Tech
New York, New York, USA

Abstract

Fair-CO$_2$ is a system for fairly attributing operational and embodied carbon in cloud data centers to user workloads. It leverages the Shapley value, a game theory solution for fair shared cost attribution with theoretical fairness guarantees. We propose the standard Shapley value solution as a ground truth for attribution in cloud data centers, addressing two key gaps in existing carbon attribution methods that lead to unfair attributions: the effect of dynamic demand on embodied carbon, and interference effects on carbon attribution in colocated scenarios. However, the computational cost of the Shapley value solution scales exponentially with the number of workloads and becomes intractable for large systems. Using Monte Carlo simulations of different workload schedules and colocation scenarios, we show that Fair-CO$_2$ can approximate the ground truth Shapley attribution solution at scale. Fair-CO$_2$ comprises two core components: Temporal Shapley attribution that applies the Shapley value to demand-aware embodied carbon attribution with low computation complexity and an interference-aware resource cost attribution method. We also show how users, once provided a fair way of estimating their workload carbon footprint, can dynamically optimize workload deployment for carbon savings.

CCS Concepts

• **Hardware → Impact on the environment**; *Enterprise level and data centers power issues*; • **Computer systems organization → Cloud computing**.

Keywords

sustainable computing, cloud computing, carbon accounting

ACM Reference Format:
Leo Han, Jash Kakadia, Benjamin C. Lee, and Udit Gupta. 2025. Fair-CO$_2$: Fair Attribution for Cloud Carbon Emissions. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA '25), June 21–25, 2025, Tokyo, Japan.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3695053.3731023

1 Introduction

In 2021, the information and computing technology (ICT) industry represented 2.1% to 3.9% of global greenhouse gas (GHG) emissions [23], on par with the footprint of the entire aviation industry. Given the increasing demand in computing, ICT emissions are expected to grow annually by 10%, accounting for up to 8% of global emissions by 2030 [42, 50]. To combat the increasing environmental impact of computing, technology companies, including Amazon, Google, Microsoft, and Meta, have pledged to be carbon neutral by 2030, reducing their climate impact [51, 58, 70, 75].

In order to guide data-driven carbon optimizations, cloud providers must first understand and quantify the carbon impact of the myriad services and workloads running in data centers. Fine-grained accounting can open new opportunities for carbon-aware system design. For example, per-workload spatio-temporal shifting can maximize renewable energy use [1, 9, 11, 88]. Similarly, recent work has explored the design of carbon-aware workloads for prominent applications such as AI [38, 40, 73] by balancing algorithms, runtime scheduling, and hardware design.

To enable such accounting, data center operators and service providers have begun to implement carbon dashboards that provide users with individualized carbon footprint estimates. For example, Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP) have developed accounting tools to help users quantify the carbon impact of their cloud use [14, 52, 66]. The Green Software Foundation, a non-profit supported by many industry and academic partners, has also been proposing software attribution standards, such as the Software Carbon Intensity (SCI). SCI has already been adopted by some large corporations, such as Accenture [21, 22]. These carbon accounting methods have enabled companies like Google to quantify and report carbon emissions to individual enterprise and user products (e.g., Google Cloud, Workspace, Maps, Meet) [65]. Similarly, open-source tools such as Cloud Carbon Footprint [79] and CodeCarbon [64] help users quantify the climate impact of their software applications [45, 46, 56].

Challenges in carbon attribution. While cloud application and workload carbon accounting methods are beginning to emerge, existing methods do not fully capture the holistic emissions of workloads in cloud data centers. Specifically, existing methods suffer from three main challenges: directly accounting for operational and embodied emissions, accounting for dynamically varying data center resource demands on varying carbon costs, and accounting for workload interference and contention.

First, some existing cloud carbon dashboards [66] do not directly quantify *both* operational and embodied emissions. *Operational* emissions owe to the energy consumed by workloads and the corresponding carbon intensity of the data centers' power grid; *embodied* emissions owe to capital infrastructure investments such as chip fabrication, memory, storage, and data center construction. Attribution frameworks must explicitly account for both operational and embodied carbon.

Second, carbon attribution frameworks must account for dynamically changing levels of demand for cloud resources that drive hardware provisioning decisions. Due to this fluctuating demand driven by patterns (e.g., diurnal) and changes in cloud user activity, data center providers must provision resources such that resource requests are accommodated during peak demand periods, unfortunately leading to idle resources during times of low demand [7, 15, 32, 81]. Intuitively, workloads running during peak demand contribute more to resource capacity requirements and should be attributed higher emissions than workloads running during off-peak times. Similarly, batch workloads that allow temporal flexibility to smooth peak resource demand should be attributed less embodied carbon.

Third, carbon attribution models must account for resource contention and interference. Colocated workloads are disproportionately affected in terms of latency, power, and energy consumption [44] due to interference from shared resources (e.g., cores, memory), affecting both operational and embodied carbon.

Finally, live carbon signals are needed to guide real-time carbon-aware system and workload optimization. Although live operational carbon intensity signals exist, such signals do not exist for embodied carbon in available carbon accounting methodologies. Furthermore, providing live embodied carbon signals while considering dynamic demand, peak provisioning, and interference at scale is challenging due to the dynamic and diverse set of millions of workloads that all share the same data center infrastructure.

In this paper, we propose Fair-CO_2, a framework to fairly attribute cloud carbon emissions. We propose using the Shapley value [68] as a *ground truth* method for this attribution. The Shapley value guarantees several fairness properties and has been proven useful in many fair attribution applications in economics [41, 43, 53] and computer systems [17, 20, 34, 44]. However, computing Shapley values is intractable at scale, as (1) its computational cost grows exponentially with the number of workloads and (2) it retroactively attributes carbon after observing temporal variances in demand. Fair-CO_2 addresses both the challenges of attribution quality and fairness along with the challenges of scalability of the Shapley value by (1) directly quantifying operational and embodied carbon models based on fine-grained open source architectural carbon models [30], (2) considering dynamic demand variance to account for the total provisioned capacity, (3) adjusting operational and embodied estimates based on workload interference, and (4) generating live carbon intensity signals for operational and embodied carbon. Using a diverse suite of workloads (PBBS [4], PostgreSQL, H.265, LLAMA [25, 28], FAISS [18], Apache Spark) and Monte Carlo simulations of various workload schedules and colocation scenarios, we compare Fair-CO_2 to baseline methods such as *Software Carbon Intensity* [22] and operational carbon-based methods used by industry [65] in terms of *fairness*, defined as the deviation from the ground truth Shapley value solution.

The key contributions of this work include:

- Demonstration of the gaps in existing methods that lead to unfair carbon attribution (see Section 3). Current methods ignore peak demand, which drives hardware capacity provisioning and thus embodied carbon costs. Moreover, current methods ignore interference between workloads. We show, using a diverse suite of workloads, that current methods can unfairly attribute carbon by more than 30% as a result of ignoring interference effects.
- Use of the Shapley value as a ground truth for fairly attributing emissions by taking into account heterogeneity across hardware resources (e.g., compute, memory, storage), interference, and dynamic demand (see Section 4). However, while the Shapley value achieves fairness, it is computationally impractical at scale because of its exponentially scaling computational cost.
- Fair-CO_2, a framework that attributes operational and embodied carbon emissions taking into account dynamic demand, interference, and utilization of hardware resources. In hyperscalers with millions of virtual machines running over a month [15], Fair-CO_2 is over 600 000 × more computationally efficient than the ground truth Shapley value method. In addition, Fair-CO_2 produces a live carbon intensity signal that enables real-time workload and system-level carbon optimization.
- Evaluation of attribution fairness via Monte Carlo simulations across 10,000 workload schedules and 10,000 colocation scenarios, showing that Fair-CO_2 reliably approximates the ground truth Shapley value solution, providing attributions that are on average 4-6× fairer than existing attribution methods (see Section 7).
- Using PBBS [4], Apache Spark, and vector databases [18], we demonstrate how Fair-CO_2 can be used to guide carbon-aware workload optimization through intelligent resource allocation and runtime scheduling optimizations to balance performance and carbon (see Section 8). In an example batch-processing vector database, we demonstrate a reduction of 38.4% in carbon over one week of simulated deployment.

Open-source: To enable future investigation in fair carbon attribution methods and their application to carbon-aware system optimization, we open source Fair-CO_2 along with the scripts required to replicate all the experimental results shown in this paper. They can be accessed through Github (https://github.com/S4AI-CornellTech/fair-co2) or Zenodo (doi.org/10.5281/zenodo.15104035).

2 Background

The carbon footprint of computing hardware during its life cycle can be broken down into operational carbon and embodied carbon [31].

Operational carbon owes to a combination of static and dynamic energy consumption. Static energy, proportional to the number of servers provisioned, accounts for about 60% of server energy according to characterization in Google data centers [65]; dynamic energy, determined by workloads running on the underlying hardware, accounts for 40% of energy consumption [65]. Operational carbon emissions are the product of total energy consumption and

Component	TDP	Embodied Carbon	Ratio
DRAM	25 W	146.87 kgCO$_2$e	1 W : 9.7943 kgCO$_2$e
CPU	165 W	10.27 kgCO$_2$e	1 W : 0.0622 kgCO$_2$e

Table 1: The large difference between in TDP to embodied carbon ratios between CPU and DRAM shows that power and energy are poor proxies for embodied carbon.

the carbon of the energy grid. The carbon intensity of a grid energy source refers to how much greenhouse gasses are emitted per unit of energy, measured as gCO$_2$e/Joule or gCO$_2$e/kWh. Grid carbon intensity varies geographically and temporally based on the availability of renewable energy in the power grids [1, 11].

Embodied carbon result from both hardware manufacturing and capital infrastructure overheads (e.g., data center construction, racks) and comprises about half of all carbon emissions from data centers in hyperscalers such as Microsoft, Google and Meta [31, 86]. Recently, researchers have proposed various tools to quantify the embodied carbon of processors [30, 76, 89, 91], memory [30], SSDs [77], and data center-scale hardware [35, 86]. Despite these methods for quantifying the embodied emissions incurred during the design, manufacturing, and resource provisioning stages, there are few methods on how to attribute the shared operational and embodied emissions of workloads running in data centers, which is the focus of this paper.

Existing cloud carbon attribution tools. To enable service-level carbon accounting, the major public cloud providers — Microsoft Azure, Google Cloud Platform (GCP), and Amazon Web Services (AWS) — have developed carbon attribution tools [14, 52, 65, 66] that allow users to directly estimate their share of carbon emissions. Various open-source tools [64, 71, 72] and academic works [3, 12, 29, 33, 63, 74, 87] also support a combination of operational carbon, embodied carbon, and energy attribution.

Generally, existing tools and frameworks separately attribute operational and embodied carbon. For operational carbon, frameworks leverage hardware power and resource utilization telemetry to attribute energy at the granularity of workloads [64, 66, 71]; energy is converted to carbon by multiplying with the average or instantaneous carbon intensity, which varies geographically and temporally. Embodied carbon is attributed differently across dashboards. For example, AWS and CodeCarbon do not directly attribute embodied carbon. Systems that attribute embodied carbon typically use billing cost, energy usage, or resource utilization over time to quantify embodied carbon [14, 52, 63, 65, 79, 87]. However, neither energy use nor billing cost is representative of embodied carbon. Different components can have drastically different power to embodied carbon ratios, as shown in Table 1 for an example system (which is described further in Section 6.1). As such, we do not discuss these methods in this paper and focus on the resource utilization based attribution systems.

3 Baseline Methods: Resource Utilization Proportional Attribution

Existing resource utilization-based attribution methods [22, 63, 65, 87] share many similarities:

(1) Using CPU utilization over time as metric to attribute energy
(2) Using resource allocation over time as a metric to attribute embodied carbon

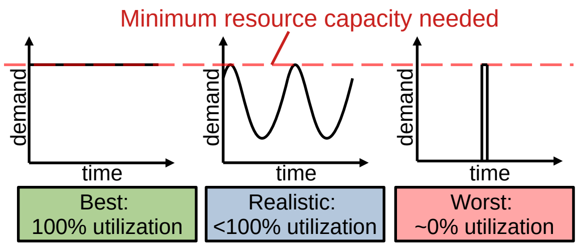


Figure 1: Minimum required resource capacity is determined by the peak demand. In the three scenarios shown above, the minimum required resource capacity is the same as given by the dashed line despite varying demand levels.

(3) Amortizing embodied carbon uniformly over time and attributing the amortized carbon proportional to resource allocation

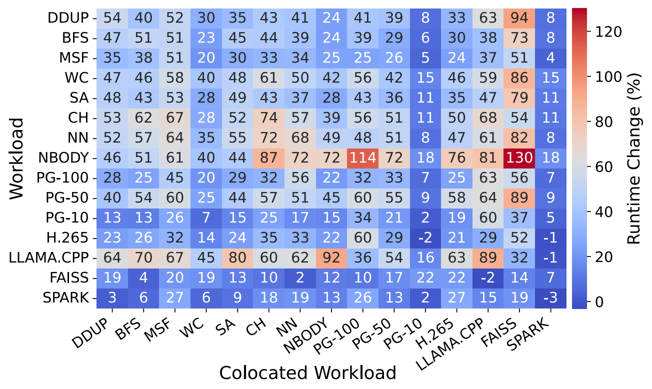
As a representative model, we use Google's production operational carbon accounting methodology as the reference baseline for operational carbon attribution [65]. Given the described methodology does not directly account for embodied carbon, we use the Green Software Foundation's Software Carbon Intensity (SCI) standard as a representative baseline [22]. This combination of Google's operational accounting and the Green Software Foundation's SCI methods provides a state-of-the-art baseline referenced as the **Resource Utilization Proportional Baseline (RUP-Baseline)** for the remainder of this paper. The RUP-Baseline is defined as:

(1) Attribute cluster per-resource static energy proportional to the workload's allocation of that resource over time
(2) Attribute cluster dynamic energy proportional to the workload's CPU utilization over time
(3) Attribute cluster per-resource embodied carbon proportional to the workload's allocation of that resource over time

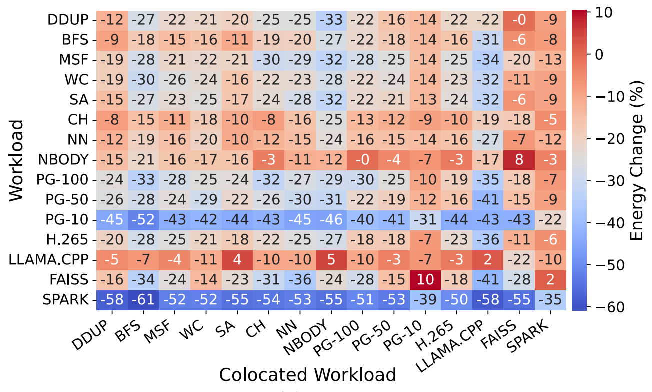
3.1 Limitations of Existing Attribution Methods

Dynamic resource demand is ignored. Data center resource utilization can exhibit strong diurnal and other patterns (hourly, weekly, monthly, etc.) [15], with low demand periods requiring much fewer resources than peak demand periods. Resources are provisioned largely on the basis of past demand and projected future demand. Peak demand represents the minimum quantity of resources that could have been provisioned while still being able to fully meet load. We apply the concept of peak pricing to attribute the carbon cost based on the impact on the minimum required resource capacity. Per Figure 1, three very different demand curves can have the same minimum resource capacity needed. Intuitively, an application that adds demand to the peak demand period increases the minimum resource capacity needed in a data center, increasing the embodied carbon. None of the existing models of carbon attribution address this relationship between dynamic resource demands and aggregate embodied carbon footprint.

Colocation effects are ignored. Although colocation can reduce carbon footprint by better amortizing idle energy and fixed embodied carbon costs, colocated workloads on a single node are susceptible to interference due to shared resources [44, 49]. With current methods for quantifying operational and embodied carbon



(a) Runtime



(b) Energy

Figure 2: Using the RUP-Baseline carbon attribution method, the carbon footprint of workloads can change drastically based on colocation neighbors. We measure the effects of colocating various PBBS [4], database, video encoding, and machine learning workloads and how their runtime and energy attribution changes versus if they were run in isolation.

based on resource utilization and time, a workload that is negatively impacted by interference can receive higher idle energy and embodied carbon shares due to increased occupancy time.

Attribution methodologies that only use per workload resource utilization ignore the effects of interference and can unfairly attribute workload emissions without accounting for external influences from colocated workloads. In the worst case, workloads that induce significant pressure on shared or contended resources (e.g., cores, caches, memory bandwidth) can cause other colocated workloads to suffer. As an example, Figure 2 shows the performance and energy impact of pairing workloads from a suite including a subset of the Problem-Based Benchmark Suite (PBBS) [4], PostgreSQL, H.265 encoding, Llama inference [25, 28], FAISS [18], and Apache Spark on server class Intel Xeon Gold 6240R CPUs (see Section 6.1). Figure 2(a) shows that colocating NBODY (the n-body problem) and CH (Convex Hull) results in 87% longer runtime for NBODY but only 39% longer runtime for CH. CH overall causes large runtime increases in its colocation partners' resource utilization, whereas NBODY has less of an effect. As a result, NBODY will be unfairly attributed extra carbon, while CH will unfairly reap the benefit of colocation. These asymmetric interference effects show that the RUP-Baseline method can lead to some workloads being unfairly attributed additional embodied and operational carbon that stem from colocation neighbors.

4 Shapley Value as a Fair Ground Truth

As described in Section 3, existing carbon attribution methods do not directly account for dynamic resource demands and the effects of colocating workloads which arise due to compute, memory and storage units being shared across workloads and services in cloud-scale systems. Just as hardware resources are shared, operational and embodied carbon is also shared between workloads. However, given the complexity of workloads that share hardware resources, it is challenging to *fairly* divide both operational and embodied carbon. To address the challenge, we propose using the Shapley value [68], a game theory solution to complex and fair division of cost problems.

Since its original formulation by Lloyd Shapley in 1951 [68], **the Shapley value** has been the core solution concept for fair attribution and collective welfare problems [53]. Shapley value provides four desirable properties of fair cost attribution:

(1) **Null Player.** Workloads that have no effect on data center carbon are attributed zero carbon.
(2) **Symmetry.** Workloads in the same equivalence class (i.e., with the same computational intensities, resource utilization profiles, and colocation characteristics) are attributed the same amount of carbon.
(3) **Efficiency.** The carbon footprint is fully attributed between all workloads and no carbon remains unattributed. Carbon is neither over-attributed nor under-attributed.
(4) **Linearity.** Linearity allows us to break down the problem of attributing data center carbon to each cloud user into smaller attribution subproblems (e.g., at rack or cluster scale).

By adhering to these four properties, the Shapley value has been shown to yield fair attribution for complex shared-cost problems. The Shapley value has a rich history of being used as a ground truth for fair attribution across many domains. In environmental economics, it is used to share pollution reduction costs between countries [57] and to attribute pollution costs between supply chain actors [13]. Other applications include cost-sharing for airport construction [43], electricity markets [37, 83], and corporate finance [41]. In networking and telecommunications, the Shapley value is used by seminal works in multicast transmissions [20] and network design [5] for fair cost sharing. The Shapley value has also been used in computer systems for energy attribution in mobile devices [17] and to attribute overhead power in servers [34]. The Shapley value is also widely used in explainable machine learning research for feature attribution [47, 62]. This work applies the Shapley value to attribute the carbon impact of data centers, where resources are shared and individual workloads can interfere with each other. This section formulates how carbon attribution can be framed as a cooperative game theory problem using the Shapley value.

4.1 Formulation

The Shapley value fairly splits costs or rewards among a set of *players* by looking at each player's marginal contributions to the total cost or reward. A set of players—workloads and users in our attribution problem—is defined as a coalition. To calculate the Shapley value fair attribution for each player, we evaluate all the possible ways of constructing the coalition by iteratively adding one player (e.g., workload) at a time. For each permutation, each player has a marginal contribution to the payoff when they are added. The Shapley value of a workload is the player's average marginal contribution across all such permutations. In the context of attributing carbon to colocated workloads, the Shapley value examines all the possible ways to construct a set of colocated scenarios by adding one workload at a time. In each permutation, a workload makes a marginal contribution to carbon costs by increasing the use of server hardware, necessary resources, and power. The Shapley value of a workload is the average of these marginal contributions across all permutations. Given the set N of n workloads with the carbon footprint function v, the formula for the Shapley value φ for workload $i \in N$ is:

$$\varphi_i(v) = \frac{1}{n} \sum_{S \subseteq N \setminus \{i\}} \binom{n-1}{|S|}^{-1} (v(S \cup \{i\}) - v(S)) \tag{1}$$

For each permutation, the embodied carbon is determined by the peak resource demand needed to run that permutation.

Similarly, each permutation will consume a different amount of energy depending on the static power consumption of the number of nodes needed and the dynamic power consumption characteristics of the workloads. The ground truth Shapley value method averages each workload's marginal contribution across permutations.

4.2 Fairness in Cloud Attribution via Shapley

The Shapley value provides a systematic method to attribute shared costs with strong theoretical foundations, providing intuitively fair attributions based on its four key properties: null player, symmetry, efficiency, and linearity. The Shapley value has been used as the gold standard for fair cost attribution, including in computer systems [5, 17, 20, 34] and pollution attribution [2, 13, 57].

We assign fair carbon costs to workloads that encapsulate their contribution to peak demand and their contribution to overall energy consumption, taking into account interference effects (see Section 3). Viewing the peak demand as the minimum required resource capacity, the Shapley value is calculated based on each workload's share of the minimum required resource capacity. The Shapley value also applies to interference-aware attribution of resource and energy costs, as it takes into account both a workload's sensitivity to interference and propensity to cause interference.

Limitations of Shapley value scalability While the ground truth Shapley guarantees fairness, it requires extensive offline profiling and evaluating counterfactual schedules for each and every coalition. The number of possible coalitions scales by 2^{N} for N workloads; exactly evaluating Shapley values becomes intractable at scale, such as in hyperscaler data centers with millions of virtual machines a month [15]. Motivated by the Shapley value as the ground truth, the subsequent sections develop a carbon attribution framework that provides both fairness and scalability.

5 Scalable and Fair Attribution with Fair-CO$_2$

To address the scalability issues of the ground truth Shapley value method, this section describes Fair-CO$_2$, a scalable attribution system that approximates the ground truth Shapley value. Fair-CO$_2$ breaks the task into two steps. First, Fair-CO$_2$ attributes carbon across time periods by casting each *time period* as a player in computing the Shapley value, as opposed to the ground truth Shapley value method which casts each workload as a player. This allows us to account for the impact of dynamic demand on fair carbon attribution. Within a time period, Fair-CO$_2$ generates rate-based carbon costs per hardware resource which can be used to assess the cost of individual workloads — circumventing the need for computing a Shapley value for *each* workload. Second, Fair-CO$_2$ adjusts the rate-based carbon cost to account for the effects of interference between workloads.

In this section, we detail the two key components of Fair-CO$_2$:

(1) Temporal Shapley for demand aware embodied attribution
(2) Interference-aware resource usage attribution

In addition, we also show how demand forecasting can be used in Fair-CO$_2$ to generate live carbon intensity signals for runtime optimization.

5.1 Temporal Shapley for Fixed Costs

Fair-CO$_2$ attributes embodied carbon and static operational carbon, which are fixed costs that scale with the data center capacity, by using the Shapley value to generate a dynamic carbon intensity signal. This carbon intensity signal determines how much carbon to attribute per unit of resource use at any specific time. Time periods with greater demand have a greater carbon intensity, and vice versa. For computational efficiency, Fair-CO$_2$ hierarchically attributes from coarser to finer granularity. Each time period is attributed carbon and then successively broken into smaller time periods until the desired granularity is reached. The Shapley value is used to define the contribution of each time period to the overall peak usage of a resource, Q. For a time period i defined as $t_{start,i} \leq t < t_{end,i}$, the peak demand is:

$$P(i) = \max(Q(t)), t_{start,i} \leq t < t_{end,i} \tag{2}$$

The peak function for a set S of time intervals is defined as the peak usage of a resource across all time periods within S.

$$p(S) = \max(P(i)), i \in S \tag{3}$$

To attribute contribution to the overall peak usage, we calculate the Shapley value of each time period using $p()$ as the payoff function.

$$\varphi_i(p) = \frac{1}{n} \sum_{S \subseteq N \setminus \{i\}} \binom{n-1}{|S|}^{-1} (p(S \cup \{i\}) - p(S)) \tag{4}$$

Resource usage and carbon during a time period is attributed proportional to that time period's Shapley value, attributing at a higher rate for high-demand periods and at a lower rate for lower-demand periods. Defining the average carbon intensity as $\overline{\gamma_i}$, for the time period i: $\overline{\gamma_i} \propto \varphi_i(p)$. Since the total amount of carbon C must be fully attributed, $\sum \overline{\gamma_i} q_i = C$, where $q_i = \int Q_i(t)dt, t_{start,i} \leq t < t_{end,i}$ is the total resource time. Thus:

$$\overline{\gamma_i} = \frac{\varphi_i}{\sum_k \varphi_k q_k} C \tag{5}$$

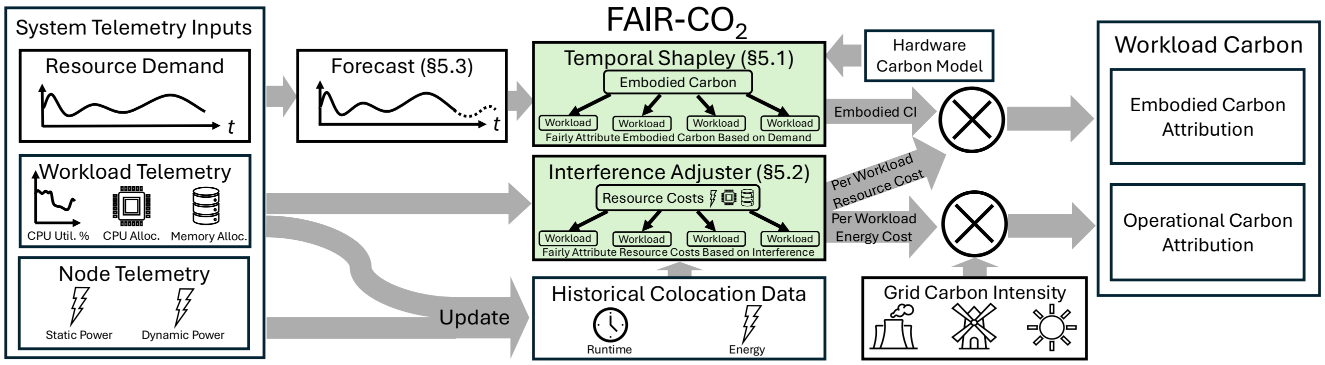


Figure 3: Architecture of Fair-CO$_2$, a scalable fair attribution framework accounting for dynamic demand and interference. Data center resource demand, along with demand forecasting, is used to a generate dynamic embodied carbon intensity signal. Workload and node telemetry along with historical colocation data are used to adjust for interference effects.

Once an average carbon intensity $\overline{\gamma_i}$ has been found for time period i, we can successively divide time period i's carbon among the smaller constituent time periods for a more fine-grained carbon intensity signal.

In practice, it is infeasible to perform Temporal Shapley attribution starting from the entire carbon footprint of a fleet of servers, as that would require knowledge of resource demand over the entire lifetime of the hardware; such end-to-end lifetime attribution is only feasible after the end-of-life, in retrospect. Instead, we first amortize the entire carbon footprint of a server over its lifetime using a simple amortization scheme such as uniform amortization [36].

For example, the embodied carbon footprint of a server (see Section 6.1) can first be uniformly amortized over its lifetime to obtain a portion of carbon per month. Given a monthly share of carbon, Figure 4 shows how Temporal Shapley can generate a 30-day 5-minute carbon intensity signal for CPU embodied carbon using CPU demand data from the Azure 2017 VM data set [15]. As shown in Figure 4, using split ratios of 10, 9, 8, 12, Temporal Attribution can successively attribute carbon from 30 days → 3 days → 8 hours → 1 hour → 5 minutes. The number of calculations needed to calculate the 5-minute granularity carbon intensity is 10,378,240 which takes 27 seconds to calculate on a single core of a commercial desktop CPU. The Azure 2017 VM trace [15] contains around 2 million VMs. The ground truth Shapley requires $2^{2\times10^6}$ calculations, over 600 000× greater than Fair-CO$_2$.

For the ground truth Shapley value attribution method, where each workload is a player, the computational complexity of calculating the Shapley value for one workload in N workloads is $O(2^N)$. In Temporal Shapley attribution, total carbon is first attributed among M_1 time periods, and then within each of the M_1 time periods, we attribute among M_2 time periods, so on and so forth. The first iteration of Temporal Shapley with M_1 players requires $O(M_1 2^{M_1})$ calculations. In the second iteration, for each of the M_1 time periods, we further attribute among the M_2 time periods, for a computational complexity of $O(M_1 M_2 2^{M_2})$. We continue this for m iterations of Temporal Shapley, with split ratios of $M_1 ... M_m$. Once carbon intensity is calculated, computing each workload's carbon attribution is a simple $O(1)$ operation that multiplies the workload's resource use by the carbon intensity. The computational complexity of Temporal Shapley for N workloads and m iterations is:

$$O(N + \sum_{i=1}^{m} (2^{M_i} \prod_{j=1}^{i} M_j)) \tag{6}$$

To further simplify the calculation, we note that for any given subset $S \subseteq N \setminus \{i\}$, the marginal contribution of the time period i is nonzero only if the other $j \in S$ have lower peak resource utilization. Sorting the time periods, $T_1, T_2, ..., T_n$, in decreasing order with respect to peak resource utilization, we get:

$$\varphi_i(p) = \frac{1}{n}[P(T_i) + \sum_{j=i+1}^{n} \sum_{k=0}^{n-j+1} \frac{\binom{n-j+1}{k}}{\binom{n-1}{k}} (P(T_i) - P(T_j))] \tag{7}$$

Attributing carbon for M_1 time periods can be calculated in $O(M_1^2)$ time. Hierarchically, if we attribute to M_2 time periods within each M_1 period, the complexity is $O(M_1 M_2^2)$. Performing m iterations of Temporal Shapley with split ratios $M_1, M_2, ..., M_m$, the computational complexity is $O(N + \sum_{i=1}^{m} M_i^2 \prod_{j=1}^{i} M_j)$, a polynomial result in both the number of workloads and the split ratios.

Theoretical limits of Temporal Shapley. To evaluate the efficacy of Temporal Shapley, we introduce the unit resource-time approximation, based on results from Hadary et al. who show that most VMs are short and survive only for a few minutes [32] with a long tail of VMs that run almost indefinitely. Resource usage within a time period is attributed a fixed embodied and operational carbon intensity. While short-lived VMs fit within a time period, longer ones span multiple periods, and the unit resource-time approximation may cause over-attribution of embodied carbon.

To demonstrate the potential over-attribution to long-lived workloads, suppose we have N workloads running over T units of time. Of these, K workloads meet the unit resource-time approximation (e.g., runtime $\ll T$), while $N - K$ of the workloads run for $\approx T$ (e.g., long-running workloads). To perform the Temporal Shapley attribution, we split our T time units into intervals $T_1, ..., T_m$, and the total amount of embodied carbon C is evenly divided among each interval. Without loss of generality, we assume that the K short-lived workloads fit into T_1, and in each of the other time intervals $T_2, ..., T_m$, the peak resource demand is approximately $P \ll 1$. Temporal Shapley then gives $\varphi_2 = \varphi_3 = ... = \varphi_m = \frac{P}{m}$, and $\varphi_1 =$

$1 - \frac{m-1}{m}P$. The average carbon intensities are also proportional to these values. Assuming that for a given time period, each workload has the same demand, we conclude that short-lived workloads are attributed proportionally to $\frac{C}{N}[1 - (\frac{m-1}{m})P]$, and long-lived workloads are attributed proportionally to $\frac{C}{N}[1 - (\frac{m-1}{m})P] + \frac{CP(m-1)}{(N-K)m}$. If $K \approx N$, the additional embodied carbon attribution to these workloads is significant. Intuitively, while Temporal Shapley attributes more to the first time period, it is split among all the workloads, but the attributed carbon for the later time periods is split among fewer workloads. As the number of long-running workloads decreases, the overall impact of this second term increases, causing over-attribution of embodied carbon. Future work may consider discounting carbon for long-running workloads.

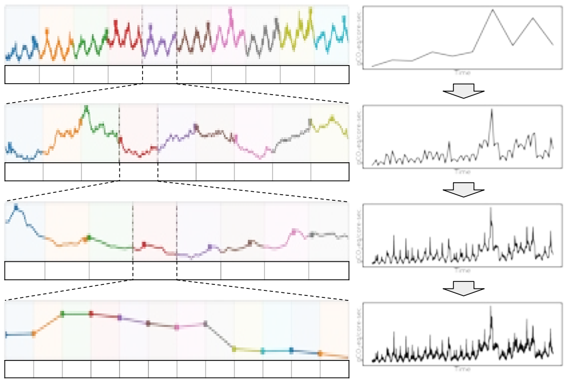


Figure 4: Temporal Shapley attributes carbon using the Shapley value from coarser time granularities to finer time granularities to generate a dynamic embodied carbon intensity signal that attributes more carbon at higher demand periods.

5.2 Fairness Adjustments for Interference

The ground truth Shapley value method as described in Section 4 inherently addresses the effects of luck in colocation partner by exploring all other possibilities for colocation partners; however, the method is not scalable. Fair-CO$_2$ approximates the ground truth Shapley method by looking at historical colocations to determine a workload's sensitivity to interference and its tendency to impose interference effects on colocation partners, similar to past work such as Bubble-Up [49] which characterizes interference characteristics of workloads in terms of sensitivity and pressure. The intuition behind this approach is based on the Shapley value calculation. In a permutation, when a workload is added to a node with an existing workload, the marginal contribution of the new workload to overall resource use is equal to its own resource use under colocation and the change in its colocation partner's resource use as a result of its colocation.

We perform this adjustment separately for runtime and dynamic energy. We define the attribution factor for workload i for embodied carbon for resource Q (e.g., CPU cores, GB of DRAM) as:

$$f_{Q,i} = (\alpha_{T,i} + \beta_{T,i}) \times Q_i \tag{8}$$

where α_{T_i} is the average historical slowdown suffered by i under colocation, β_{T_i} is the average slowdown inflicted by i on i's historical colocation partners and Q_i is the quantity of resource allocated to i. For a time slice, we attribute embodied carbon to each workload proportional to its attribution factor $f_{Q,i}$. Workload i's adjusted embodied carbon attribution intensity (in gCO$_2$e per resource-second) then becomes:

$$\gamma_{Q,i}(t) = \overline{\gamma_Q}(t)\frac{Q_{tot}(t)}{Q_i}\frac{f_{Q,i}}{\sum f_Q} \tag{9}$$

where $\overline{\gamma_Q}(t)$ is the embodied carbon intensity at time t and $Q_{tot}(t)$ is the aggregate resource demand.

Similar formulas can be used to adjust dynamic energy attribution:

$$f_{P,i} = (\alpha_{P,i} + \beta_{P,i}) \times P_{i,iso} \tag{10}$$

where α_{P_i} is the average RUP-Baseline dynamic energy attribution change suffered under colocation, β_{P_i} is the average RUP-Baseline dynamic energy change inflicted by i to i's historical colocation partners, and $P_{i,iso}$ is i's average power under isolation. The dynamic power attributed to i is then:

$$P_i(t) = P_{tot}(t)\frac{f_{P,i}}{\sum f_P} \tag{11}$$

where $P_{tot}(t)$ is the aggregate power consumption across all nodes.

5.3 Demand Forecasting

All existing carbon attribution methods attribute retroactively, with existing public cloud carbon attribution dashboards only providing aggregate carbon attribution for a user at a monthly granularity [14, 52, 66]. To enable users to dynamically optimize the carbon footprint of their workloads, demand projection can be integrated with Fair-CO$_2$ to provide live and projected embodied carbon intensity signals for data center resources. We can use time-series forecasting tools, such as Meta's Prophet tool [78]. Forecast tools work well to forecast data center demand due to periodic trends that exist in data center usage [15], as shown in Figure 5. Using historical data and projected data, we can use our attribution framework to generate carbon intensity signals both for the current time and for the projected future. By finetuning Meta Prophet's hyperparameters (e.g., frequency modes), we demonstrate that resource demand can be reasonably forecasted with only 21 days of historical data. In practice, cloud service providers can build more accurate forecasting predictors with access to more historical data and finer-grained per-service workload characteristics [15, 32]. Fair-CO$_2$ implements demand forecasting as a modular component, allowing cloud service providers to easily integrate existing demand forecasting tools with Fair-CO$_2$ to generate live carbon intensity signals.

6 Experimental Methodology

6.1 Hardware Infrastructure and Telemetry

We evaluate Fair-CO$_2$ on a server with two Intel Xeon Gold 6240R Cascade Lake CPUs comprising a total of 48 physical cores, 192 GB of DDR4 memory, and 480 GB of SSD storage. We estimate the carbon footprints of IC components using iMec [89] and ACT [30].

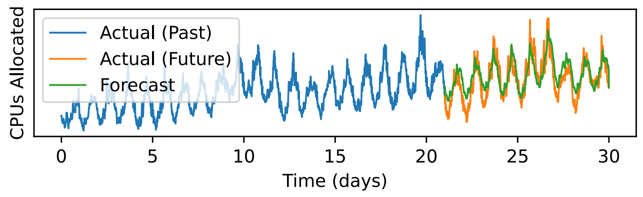
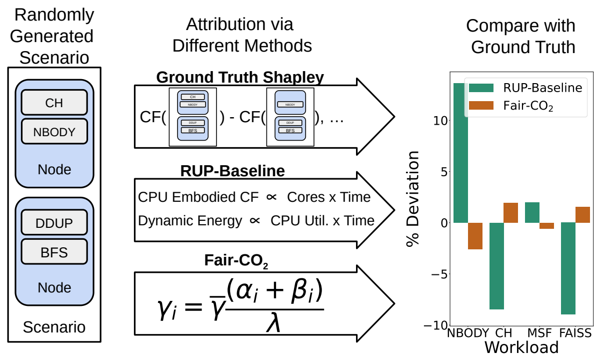


Figure 5: Due to patterns in aggregate data center CPU demand, forecasting tools such Meta Prophet can reasonably forecast demand days and weeks ahead, as shown on Azure VM 2017 traces [16]. Dynamic demand forecasting allows Fair-CO_2 to generate live embodied carbon intensity signals that take into account projected changes in demand.



Randomly Generated Scenario
CH
NBODY
Node
DDUP
BFS
Node
Scenario
Attribution via Different Methods
Ground Truth Shapley
CF( ) - CF( ), ...
RUP-Baseline
CPU Embodied CF ∝ Cores x Time
Dynamic Energy ∝ CPU Util. x Time
Fair-CO2
$\gamma_i = \bar{\gamma}\frac{(\alpha_i + \beta_i)}{\lambda}$
Compare with Ground Truth
RUP-Baseline
Fair-CO2
% Deviation
Workload
NBODY
CH
MSF
FAISS

Figure 6: The RUP-Baseline, Fair-CO_2, and the ground truth Shapley value method are used to create per-workload carbon attributions across 10,000 randomly generate colocation scenarios. For each scenario, we compare the RUP-Baseline attribution and the Fair-CO_2 attribution against the ground truth.

For SSD storage, we estimate the carbon footprint at a rate of 0.16 $kgCO_2e$/GB[77]. We estimate the carbon footprint of the mainboard, chassis, and cooling using reference values from the Dell R740 LCA [55] and by scaling the power and cooling components by system TDP. We measure system power and resource utilization using Intel PCM and per-workload resource utilization using Docker.

6.2 Workloads

We use the following workloads to evaluate Fair-CO_2:

Problem-Based Benchmark Suite (PBBS) We choose the following eight PBBS, parallel workloads[4]:

- DDUP: remove duplicates from a list of 2 billion random integers
- BFS - breadth-first search on a 640 million node directed graph
- MSF - find the minimum spanning forest on an undirected weighted graph of 120 million nodes and 2.4 billion edges
- WC - count the number of occurrences of each word in a string of 500 billion characters
- SA - generate the suffix array of a string of 500 billion characters
- CH - calculate convex hull from 1 billion points in 2-D space
- NN - find 10 nearest neighbors for 50 million 3-D points
- NBODY - calculate gravitational forces of 10 million, 3-D points

PostgreSQL Benchmark (PG) We use pgbench to simulate concurrent clients that generate database traffic. We tested 100 clients (PG-100), 50 clients (PG-50), and 10 clients (PG-10).

Video Encoding (H.265) We use x.265 to perform H.265 video encoding on a 2.6 GB 4K video.

Llama.cpp Inference (LLAMA) We run Llama 3 8B [28] inference on the CPU using Llama.cpp's llama-bench tool [25] with batch size = 1, prompt size = 128 tokens and output size = 64 tokens.

Facebook AI Similarity Search (FAISS) FAISS [18] is a library used for document retrieval. We run retrieval benchmarks on two types of indices: Inverted File Index (IVF) and Hierarchical Navigable Small World (HNSW) graphs [8].

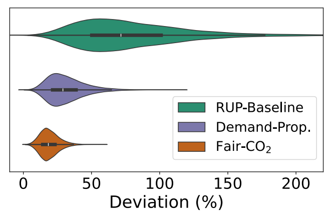
Apache Spark (SPARK) We run Apache Spark on a local node via PySpark, performing SQL queries on a scaled version of the STORE_SALES table from TPC-DS [59, 85].

6.3 Evaluation Against the Ground Truth via Monte Carlo Simulation

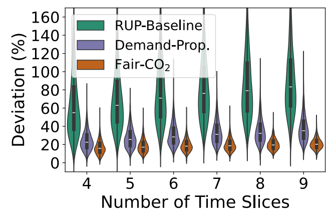
To evaluate Fair-CO_2 and the RUP-Baseline, we use percentage deviation in attribution from the ground truth Shapley method as a measure for fairness. As the ground truth Shapley value attribution is done by counter-factually permuting across all coalitions and evaluating the marginal carbon footprint of each workload, the deviations between the ground truth attributions and those from other methods will vary case-by-case. To effectively evaluate against the ground truth, we simulate carbon attributions across a large number of different possible scenarios and coalitions and characterize the deviation from the ground truth.

Demand evaluation methodology. To evaluate fairness in accounting for dynamic demand, we generate 10,000 different workload schedules with dynamic demand over time. For each workload schedule, we determine the embodied carbon attribution within the schedule using the ground truth approach and other methods. We then compare the carbon attribution for each workload for each method to the ground truth attribution. In addition to Fair-CO_2 and the RUP-Baseline, we also evaluate a demand-proportional attribution scheme for comparison. The demand-proportional method attributes carbon such that the embodied carbon intensity and static operational carbon intensity are directly proportional to the resource demand at each moment in time. Due to the exponential scaling of the ground truth Shapley method, we limit the number of workloads within a schedule to 22. The generated schedule contains between 4 and 9 time steps, with each time step containing between 1 and 5 workloads running simultaneously. Each workload uses 8, 16, 32, 48, 64, 80, or 96 CPU cores and runs for anywhere between 1 and 3 time steps. As described in Section 4, the ground truth Shapley method treats each workload as a player and finds each workload's marginal contribution to the peak demand across all permutations. RUP-Baseline attributes carbon based solely on the workload resource allocation over time; the demand proportional method and Fair-CO_2 look at both the workload's resource allocation and the dynamic demand.

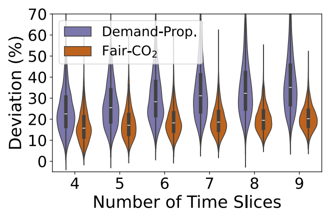
Interference evaluation methodology. To evaluate the effects of interference on attribution for the different methods, we simulate 10,000 sets of colocated pairs of workloads. Each set contains



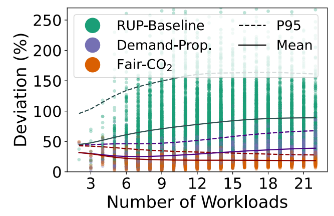
(a) Average deviation across all schedules



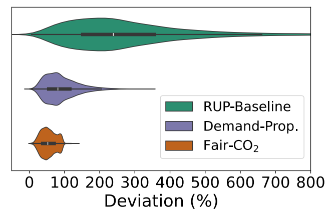
(b) Average deviation for different lengths of schedules



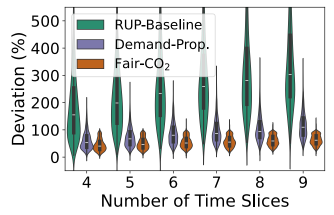
(c) Average deviation for different lengths of schedules



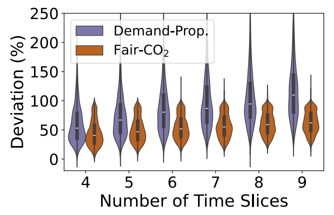
(d) Average deviation for varying number of workloads



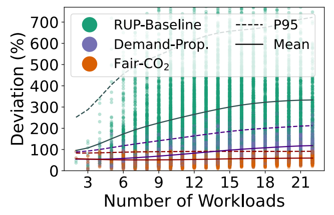
(e) Worst case deviation across all schedules



(f) Worst case deviation for different lengths of schedules



(g) Worst case deviation for different lengths of schedules



(h) Worst case deviation for different number of workloads

Figure 7: Monte Carlo simulation results for workload schedules with dynamically varying demand to evaluate fairness with respect to peak demand. Across 10,000 simulated scenarios, Fair-CO$_2$ consistently attributes closer to the fair ground truth than the RUP-Baseline. The top plots look at average attribution deviation across all workloads in each simulated scenario. The bottom plots look at the least fair attribution for any one workload in each simulated scenario.

anywhere from 4 workloads to 100 workloads, where each workload can be any of the eight PBBS workloads, pgbench with 100, 50, or 10 clients, H.265 video encoding, Llama inference, FAISS, or Apache Spark. We characterize the effects of interference for each of the workloads in all possible pairwise colocations, as shown in Figure 2. Each workload is allocated 48 CPU cores and 96 GB of memory, which are half the resources of a node. The ground truth Shapley method permutes across all possible colocations, providing a carbon attribution to each workload that is an average of that workload's marginal contributions across the permutations. Figure 6 shows the evaluation process for one mock set of workloads.

7 Evaluation

In this section, we evaluate our method and the RUP-Baseline against the ground truth Shapley (see Section 4) around two key dimensions: demand-aware and interference-aware attribution.

7.1 Attribution Fairness with Dynamic Demand

Figure 7 shows the fairness of three embodied carbon attribution methods: the RUP-Baseline, a demand proportional method, and the Temporal Shapley method implemented by Fair-CO$_2$. The demand proportional method simply attributes carbon at any moment in time proportional to the dynamic demand at that time. We evaluate it as a demand-aware baseline.

We evaluate the different attribution methods in terms of average deviation (top) from the ground truth Shapley value in each scenario. We define the average deviation for each scenario as the average attribution deviation from the ground truth attribution across all workloads in that scenario. For each simulated scenario, we also look at the "worst case" deviation (bottom), as in cloud environments, individual users care about their personal footprint as opposed to the average fidelity of the attribution. The "worst-case" deviation is defined as the maximum single workload attribution deviation from the ground truth within a scenario. Figure 7 evaluates the attribution schemes across all scenarios (a, e), different schedule lengths (b, c, d, g), and different number of workloads (d, h).

Fairness comparison across all scenarios of dynamic demand. Figure 7(a, e) show the overall comparison of the RUP-Baseline method, the demand proportional method, and Fair-CO$_2$ across all simulated schedules. We find that the RUP-Baseline method deviates from the fair, ground truth Shapley attribution by around 80% on average and 279% in the worst case. This is due to the fact that the RUP-Baseline method does not account for time-varying demand and attributes carbon across resource usage uniformly. The demand proportional attribution method deviates from the ground truth by approximately 31% on average and 90% in the worst case. In comparison, Figure 7(a) shows Fair-CO$_2$ minimizes deviation from ground truth compared to the other methods. Fair-CO$_2$'s Temporal Shapley deviates around 19% from the ground truth on average, with a smaller spread than the RUP-Baseline and the demand-proportional method. Similarly, even in the worst case performance, Fair-CO$_2$'s Temporal Shapley method deviates by around 55%. The Temporal Shapley method approximates the ground truth significantly better than even the demand proportional approach because it employs the Shapley value attribution mechanism.

Fairness under varying schedule lengths. While we cannot evaluate the ground truth Shapley attribution method at scale due to its computational complexity, we analyze the impact of "scheduling length" to understand how the different attribution methods scale. Figures 7(b, c) show the three attribution methods and their average deviation from the ground truth Shapley. Scaling the number of time slices worsens the methods' deviation from the ground truth. For example, the average deviation of the RUP-Baseline method increases from 65% to around 93% as the number of time slices increases from 4 to 9. The demand proportional deviation increases from 24% to 38% going from 4 to 9 time slices. Across the spectrum, we find Fair-CO_2 deviates by 20% or less even as the number of time slices increases. Similar to the overall deviation across all schedules, the worst-case deviation is significantly worse for the RUP-Baseline (up to 150%) compared to Fair-CO_2.

Fairness under varying number of workloads. Finally, we perform a similar scaling analysis for the number of workloads, as shown in Figure 7(d, h). Fair-CO_2 scales much better than the RUP-Baseline and the demand proportional method. As the number of workloads increases, the counterfactual analysis performed for the ground truth Shapley approach becomes more complex, and the baseline simple attribution schemes begin to diverge from the ground truth. In contrast, Fair-CO_2 uses the same Shapley value mechanism of counterfactual attribution, so it performs consistently well as the number of workloads increases. The average fairness of the demand proportional method and Fair-CO_2 is similar for few workloads; however, past 15 workloads, the demand proportional method's average deviation from the ground truth becomes even worse than Fair-CO_2's 95th percentile deviation. This suggests that Fair-CO_2 will continue to attribute fairly at scale and serves as a scalable approximation of the ground truth.

7.2 Attribution Fairness under Interference

Similar to the analysis for dynamic demand fairness, we evaluate the deviation of the methods compared to the ground truth Shapley value over 10,000 simulations of different sets of colocated workloads, as shown in Figure 8. We evaluate the different attribution methods in terms of average (top) and worst-case deviation (bottom) to understand their limits in at-scale environments. Figure 8 shows the overall deviation across all scenarios (a, e), across historical sampling rates (b, f), across the number of colocated workloads (c, g), and across varying grid intensities (d, h).

Fairness across all scenarios of colocated workloads. As shown in Figure 8a, Fair-CO_2's interference-aware method is significantly better at following the ground truth Shapley value attribution than the RUP-Baseline. Fair-CO_2's attribution deviates on average by only 1.72%; on the other hand, the RUP-Baseline method deviates by 9.7% on average. Note that the worst-case deviations, as shown in Figure 8e, show a large gap with RUP-Baseline deviating by around 31.7% across all scenarios and Fair-CO_2 deviating by around 5.0%.

Robustness to sparse historical data availability. Fair-CO_2's interference-aware attribution methods relies on historical colocation data to build a profile of each workload's sensitivity to colocation and each workload's propensity to cause interference. In practical scenarios, workloads may only have a sparse set of historical data to condition on. To simulate what would happen with limited data, for each workload in each simulation scenario, we randomly choose to only use a subset of the colocation data shown in Figure 2 to generate the attribution for that workload. We uniformly randomly choose to use anywhere from only one sample (6.7% sampling rate) to all 15 samples (100% sampling rate). Figures 8b and 8f show the impact of the sampling rate on the attribution method. The RUP-Baseline method does not account for interference or look at historical data. For Fair-CO_2, even conditioning on 1 data sample is sufficient to achieve significant improvements in fairness; even though the effect of interference experienced by a workload can differ significantly depending on its partner, one or two samples are often enough to estimate the workload's sensitivity to interference and propensity to cause interference.

Fairness under varying number of workloads. Figures 8c and 8g show the average and worst-case deviations as we scale the number of workloads, respectively. At few workloads, the RUP-Baseline has a greater spread for both the average and worst case. As the number of workloads increases, the RUP-Baseline's average deviation from the ground truth converges to around 9.5%. As the number of workloads in a scenario increases, the chance of any one pairing being particularly unfair increases; therefore, the worst case for RUP-Baseline increases as the number of workloads increases. In contrast, Fair-CO_2 is capable of consistently providing fair attributions because it uses historical data to inform attribution, effectively averaging over history and negating the randomness in the colocations that arise from any one scenario.

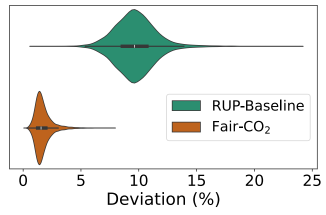
Fairness under varying grid carbon intensity. We also look at fairness across different grid carbon intensities — Fair-CO_2 similarly outperforms the RUP-Baseline. As the grid carbon intensity increases, the weighting of embodied and operational carbon changes, causing the operational carbon to become more dominant. The RUP-Baseline performs worse at low grid carbon intensities, meaning that the attributions are less fair for embodied carbon than operational carbon. Fair-CO_2 performs consistently well across different grid carbon intensities, attributing embodied and operational carbon equally fairly.

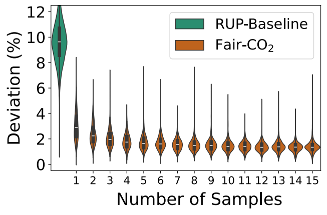
Attribution equity across different types of workloads. Fair-CO_2 not only greatly improves attribution fairness on average, it also reduces variance within and across different workloads. In Figure 9, the top two plots show the distribution of each workload's attribution deviations from the ground truth. The two bottom plots show the distribution of each workload's partner's deviation from the ground truth, with RUP-Baseline on the left and Fair-CO_2 on the right. The RUP-Baseline attributes carbon unfairly to workloads running on an isolated node, which is seen as the upper band in Figures 8g and 8h. The different peaks in the RUP-Baseline plots represent specific colocation pairings. Fair-CO_2 virtually eliminates the effects of different workloads on their partner workloads, yielding fair carbon attribution.

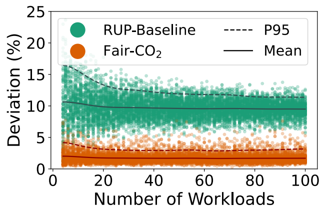
Robustness to varying workload load. In production data centers, workloads can serve a varying number of clients over time exhibiting distinct hardware utilization, impacting carbon cost. For instance, Figure 2 illustrates PostgreSQL with 10, 50, and 100 clients exhibiting varying interference patterns when colocated with the suite of Fair-CO_2's workloads. Figure 8 shows the evaluation of the three load scenarios for PostgreSQL separately yields commensurate fairness as other workloads.

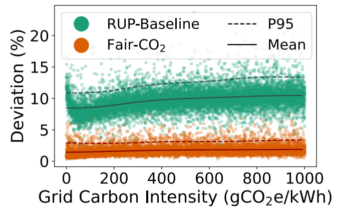
7.3 Live Carbon Intensity Evaluation

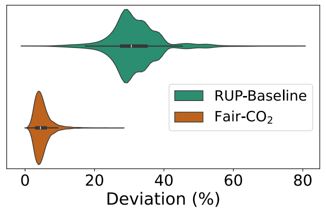
Figure 11 shows the impact of errors in demand forecasting on the live embodied carbon intensity signals generated by Fair-CO_2. Using Meta Prophet and 21 days of historical data, we forecast 9

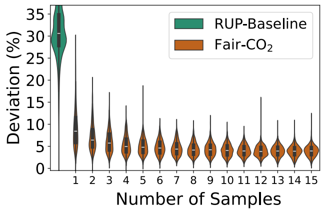

(a) Average deviation across all scenarios

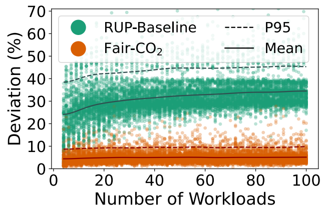

(b) Average deviation for varying historical sample rate


(c) Average deviation for different number of workloads


(d) Average deviation for different grid CI


(e) Worst case deviation across all scenarios


(f) Worst case deviation for varying historical sample rate


(g) Worst case deviation for different number of workloads

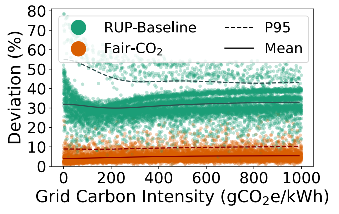

(h) Worst case deviation for different grid CI

Figure 8: Monte Carlo simulation results for sets of colocated workloads under interference. The top plots look at average attribution deviation across all workloads in each simulated scenario. The bottom plots look at the least fair attribution for any one workload in each simulated scenario.

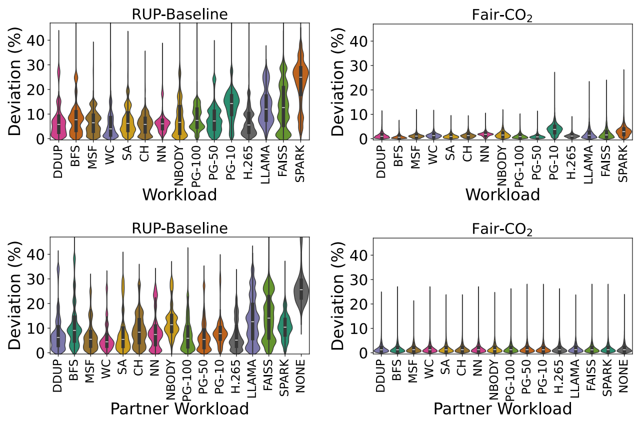


Figure 9: Deviation from the ground truth for specific workloads and partner workloads. Top row shows the distribution of deviation from the ground truth for each workload. Bottom row shows the distribution of attribution deviation for each workload's partner workload. The left column is the RUP-Baseline method; the right column is Fair-CO$_2$. Fair-CO$_2$ largely eliminates unfair carbon attribution biases for different workloads which are present under the RUP-Baseline method.

days of demand to generate live embodied carbon intensity signals at 5-minute intervals. We compare the carbon intensity generated from the forecast with the embodied carbon intensity generated from the true data. We find that the mean absolute percentage error (MAPE) caused by the forecast error is 2.30 % for the forecast days. Furthermore, the worst-case embodied carbon intensity error caused by forecast errors is only 15.72 %. As such, even with limited demand forecasting knowledge based on publicly available resource estimates (see Section 5.3), Fair-CO$_2$ is able to accurately generate live carbon intensity signals to guide online carbon-aware system optimization.

8 Case Study: Workload Carbon Optimization

Using Fair-CO$_2$, we enable users to reduce their carbon footprint by varying workload configurations. For instance, for PBBS and Apache Spark we can vary the number of CPU cores, parallel threads, and memory allocation to determine the workload's resource use. Algorithmic changes are also possible for workloads with multiple algorithms (e.g., IVF versus HNSW search for FAISS).

For each of the workloads listed in 6.2, we sweep across workload configuration parameters, measuring energy, resource utilization, and runtime for each configuration. For the PBBS workloads and Spark, we sweep across various CPU allocations from 8 to 96 cores and various memory allocations from 8 GB to 192 GB. For FAISS, for both IVF and HNSW indices, we sweep across CPU allocations from 8 to 96 cores and batch sizes from 8 to 1024.

Optimizing workload configuration for minimizing carbon footprint. We investigate how the carbon footprint of different workload configurations change as grid carbon intensity changes. Different workload configurations not only have different performance, they have differences in CPU resource and memory resource requirements along with runtime and energy consumption changes.

Figure 10 shows that as grid carbon intensities change, the carbon-optimal workload configuration also changes. For each workload, we plot the footprint of different configurations (e.g., energy optimal, embodied optimal, overall carbon optimal) normalized to the performance-optimal configuration. Although the minimal overall carbon configuration depends on the grid carbon intensity, the energy and embodied carbon optimal configurations do not. Configuration changes are shown using different shaded regions.

PBBS workloads and Spark experience good but sub-linear parallel scaling, especially at higher core counts. Embodied carbon,

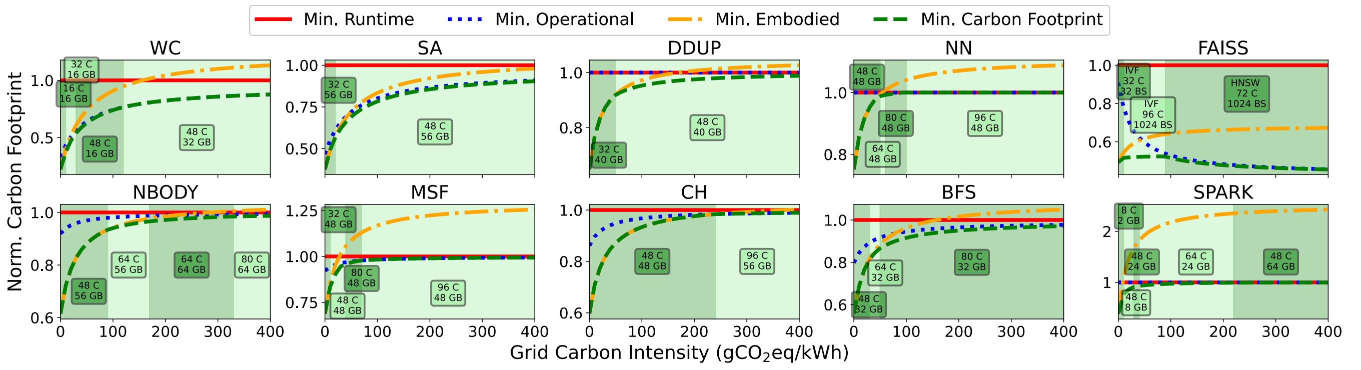


Figure 10: As grid carbon intensities change, the carbon-optimal workload configuration changes, providing up to 65% carbon savings compared to optimizing for performance. Carbon footprints are normalized to the carbon footprint of the performance optimal configuration (solid red line).

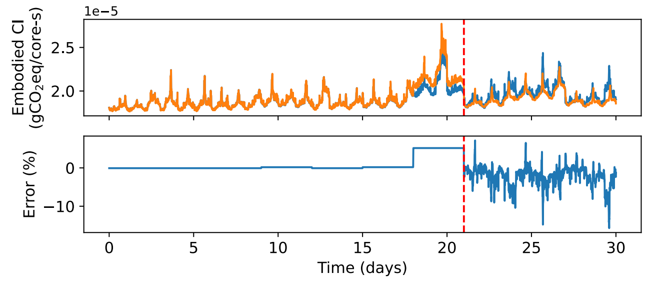


Figure 11: Fair-CO$_2$'s embodied carbon intensity signal remains stable with demand forecasting error. The top plot shows embodied carbon intensity generated from the full 30-day Azure trace [16] versus a 21-day trace with a 9-day forecast (see Figure 5). The bottom plot shows the error in embodied carbon intensity with the 9-day forecast.

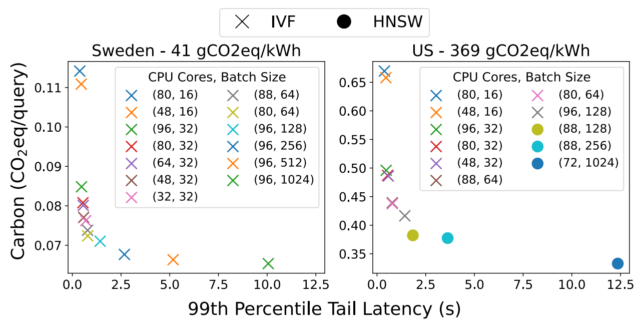


Figure 12: FAISS carbon-latency trade-off Pareto fronts. The Pareto-optimal set of core allocation, batch size, and index choice is changes with the grid carbon intensity.

proportional to core-seconds and GB-seconds, goes up with higher resource utilization. Operational carbon, on the other hand, decreases as the static energy consumption decreases with a faster runtime. We also observe that the dynamic energy to CPU utilization ratio, in J/%-s, decreases as more cores are used due to simultaneous multithreading. Thus, the core count for the carbon optimal configuration increases as grid intensity increases and operational carbon makes up a larger portion of the overall footprint. Some workloads — WC, NBODY, and Spark — are amenable to varying memory allocation, providing an additional parameter to optimize carbon.

We explore varying CPU cores, batch size, and search algorithm for FAISS. IVF experiences better core scaling, using up to all 96 cores, which allows faster performance at the expense of greater energy use. HNSW stops scaling in performance past 88 cores. The index size of IVF is smaller, using 77.7 GB of memory versus 180.8 GB for HNSW. Because of HNSW's lower power consumption and larger memory footprint, HNSW has a greater embodied carbon to operational carbon ratio. At around 90 gCO$_2$e/kWh, the carbon-optimal algorithm switches from IVF to HNSW as the overall carbon footprint becomes more operational carbon dominated.

Performance-carbon trade-offs. When the workload configuration changes, both the carbon footprint and the performance change. Figure 12 shows the Pareto-optimal trade-off space between tail latency and carbon across two grid carbon intensities for FAISS. IVF runs faster for small batch sizes with low-latency configurations. At low latency, performance gains in tail latency reduction come at large costs to carbon footprint — in the Sweden scenario, going from 32 cores with a batch size of 32 to the latency-optimal configuration of 80 cores with a batch size of 16 reduces tail latency by 45% at a cost of a 50% increase in carbon footprint.

Dynamic workload adjustment. To maximize savings, users can dynamically configure their workload in response to live carbon intensities. As shown in Figure 12, carbon savings plateau once the FAISS indices reach a tail-latency of 2 to 2.5 seconds. The shaded regions (top plot) represent different retrieval algorithms (i.e., IVF, HNSW) run to minimize carbon footprint. MLPerf's benchmarks for LLM-based Q&A and Text Generation implement a server latency target of 2 seconds [61] where FAISS indices are frequently used in RAG-based LLMs. Thus, we set a tail-latency target of 2 seconds for our dynamic optimization case study with FAISS. Figure 13 shows how the optimal configuration for FAISS changes over time under this constraint. Responding to the real carbon intensities of the grid from California, USA [48] and the embodied carbon intensities generated from the Azure 2017 VM traces [15], we see

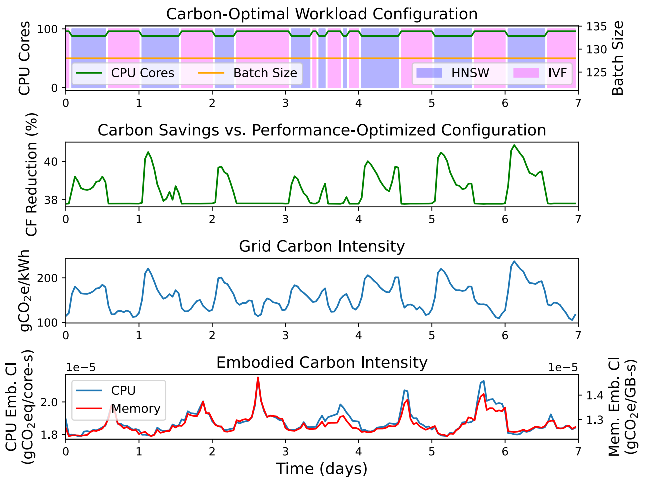


Figure 13: FAISS's configuration (top) is dynamically optimized over one week in response to grid carbon intensity (third from top) and Fair-CO$_2$ embodied carbon intensity (bottom) changes, yielding 38.4% carbon savings (second from top) compared to the performance-optimal configuration.

that the optimal algorithms switch between IVF and HNSW; the optimal algorithm switches to HNSW when the intensity of the grid is higher and the embodied intensity is lower.

9 Related Works

Hardware carbon accounting and modeling. Tools such as SCARIF [35], ACT [30], FOCAL [19], GreenSKU [86], and others [1, 76, 77, 86, 89, 91] model and optimize the operational and embodied carbon footprint of computer hardware. However, these works do not address the attribution of carbon footprint to workloads.

Software carbon and energy accounting. We compare against SCI [22] for embodied carbon attribution and against Google's carbon attribution method [65] for operational carbon attribution. Other works [3, 33, 63, 64, 71, 87] look at attributing carbon or energy using resource allocation and utilization metrics. We address these methods and their gaps in Section 3.

QSEAS [34] uses the Shapley value to attribute overhead power consumption (i.e., power from UPSs, cooling, etc.) with quadratic complexity, but it cannot be applied to server power attribution or other resource costs. Dong, Lan, and Zhong [17] use the Shapley value to attribute energy on mobile devices. However, they are limited to a small set of workloads due to the computational complexity of calculating the Shapley value. No prior work uses the Shapley value to attribute embodied carbon.

Fair resource allocation. Karma [84], Dominant Resource Fairness [26], and other works [27, 60, 69] address fairness issues in resource allocation for shared resources. However, they do not address fair cost attribution. Cost attribution via the Shapley value has been explored for networks [20]. Fair resource allocation and colocation schemes such as Cooper [44] provide fairness; however, they limit possible scheduler and colocation optimizations that can provide more efficient resource allocation (i.e. stranding minimization). In contrast, Fair-CO$_2$ provides fair carbon attributions that are agnostic to the choice of scheduler.

Interference-aware scheduling. Our approach to workload interference is similar to Bubble-Up's [49] notions of sensitivity and pressure. Cooper [44] uses game theory to devise a scheme for fair pairwise colocations between workloads, accounting for interference. Like Cooper, other works [10, 24, 39, 54, 80, 82, 90] provide solutions for interference-aware workload scheduling and colocation, but do not address fair cost attribution.

10 Conclusion and Future Work

Fair-CO$_2$ is a first-of-its-kind framework to leverage cooperative game theory to fairly attribute both operational and embodied carbon. Central to Fair-CO$_2$ are its two key components — Temporal Shapley and interference-aware attribution — that enable scalable carbon attribution taking into account dynamic demand and interference. Fair-CO$_2$ also generates live carbon intensity signals to guide real-time carbon optimization. This work opens new research directions for the use of fair, scalable carbon attribution for carbon-aware systems. In the following, we outline some salient directions:

Greater coverage of all data center emissions. Data centers implement myriad components beyond servers, including networking systems, power delivery, cooling, UPS systems and mechanical components (e.g., racks). Complex interactions occur across co-designing server architectures with overall data center infrastructure. For example, recent work has documented that resource stranding can account for up to 15% of memory capacity [6]. Future works must consider attributing responsibility of all data center components and overheads, including stranding, software schedulers, and hardware cooling and power delivery.

Request-level attribution and function-level attribution. Fine-grained attribution for microservices and serverless platforms is future work that can leverage the demand-aware and interference-aware elements of Fair-CO$_2$. Such fine-grained attribution can open unique opportunities to guide carbon-aware service design considering embodied and operational emissions.

Privacy and security. Fair-CO$_2$ requires as input fine-grained resource utilization statistics which are already tracked for VM-level and node-level telemetry in production data center for smart scheduling [15, 32] and for carbon attribution [52, 65, 67]. Fair-CO$_2$ also requires tracking the average runtime and energy variance under colocation per workload, as well as the impact of workloads on their colocation partners. This can introduce opportunities for power and resource side channels leaking private data for sensitive applications. Future efforts must investigate privacy-aware interfaces to realize fair carbon attribution.

Acknowledgments

We would like to thank Michael Shen for his help and expertise with setting up the FAISS workload. We would also like to thank Zhanqiu Hu and Yueying Li for their feedback and comments on drafts of this paper. This work is supported by the National Science Foundation via the Expeditions in Computing program (grant nos. 2326606 and 2326608) and the EAGER program (grant no. 2335795). This work is also supported by Chameleon Cloud (grant no. 232660) and Google.

A Artifact Appendix

A.1 Abstract

Fair-CO2 is a scalable and fair framework for attributing both the operational and embodied carbon of cloud data centers to cloud workloads. This artifact contains code and instructions on how to reproduce the experimental results presented in the paper, including colocation characterization results, Monte Carlo simulation results, demand forecasting evaluation results, and the dynamic workload optimization case study. The workloads used are PBBS, Apache Spark, Llama.cpp, x256, pgbench, and FAISS.

A.2 Artifact check-list (meta-information)

- **Output:** Figures
- **How much disk space required (approximately)?:** 600 GB
- **How much time is needed to prepare workflow (approximately)?:** 1-2 weeks
- **How much time is needed to complete experiments (approximately)?:** 2 weeks
- **Publicly available?:** Yes
- **Workflow automation framework used?:** Yes: shell scripts, python scripts, docker containers.
- **Archived (provide DOI)?:** 10.5281/zenodo.15104035

A.3 Description

A.3.1 How to access. The artifact can be accessed at https://doi.org/10.5281/zenodo.15104035.

A.3.2 Hardware dependencies. We recommend testing on a bare-metal system with root access with the following requirements:

- **Number of CPUs:** 2
- **Physical cores per CPU:** 24
- **Logical cores per CPU:** 48
- **RAM size:** 192 GiB
- **Storage capacity:** > 600 GB
- **Operating system:** Ubuntu 22.04

A.3.3 How to run. Follow the installation and experiment instructions in the included README.md as it contains more details and may be easier to follow due to better formatting compared to this appendix. Alternatively, you may follow the instructions in the following sections.

A.4 Installation

Download and unzip the artifact in the home directory. Run the setup scripts located in `/setup-scripts`:

```
source setup-scripts/env.sh
source setup-scripts/create_large_swap.sh
source setup-scripts/install_pcm.sh
source setup-scripts/install_docker.sh
source setup-scripts/install_conda.sh
source setup-scripts/setup_workloads.sh
source setup-scripts/delete_swap.sh
```

We strongly recommend reviewers to download and use the pre-generated data rather than using the generation scripts due to the time required for generation.

Specifically, we provide the HNSW and IVF FAISS indices required for the FAISS workload and the store_sales.csv required for the Apache Spark workload, accessible at this Google Drive link.

Download and copy `100M_IVF16K_SQ8.faiss` and `100M_HNSW.faiss` to `/workloads/faiss/indices/`

Download and copy `store_sales.csv` to `/workloads/spark/data/`

Create and activate the conda environment:

```
conda env create -f environment.yml
conda activate fair-co2
```

A.5 Experiment workflow

A.5.1 Using Provided Experimental Data. To reproduce the paper results without running all workload experiments on hardware, partially pre-processed experimental data for all experiments is provided. When applicable, instructions to use this data for analysis and figure generation are noted throughout this document.

A.5.2 Pairwise Colocation Profiling (Figure 2). This set of experiments runs all pairwise colocations along with isolated runs of each workload.

To use the provided data, copy the content from `colocation/ref-results` to `colocation/results` and skip steps 1a and 1b.

Colocation Experiments Run the following script on the test server to start the experiment. Results are stored in `colocation/results`. The run time is around one day.

```
python3 colocation/colocation_sweep.py --model \
Meta-Llama-3-8B-F16.gguf
```

To run a smaller subset of the experiments (for demo purposes only), use the `–small` argument:

```
python3 colocation/colocation_sweep.py --small \
--model Meta-Llama-3-8B-F16.gguf
```

Processing Colocation Experiment Logs

```
python3 colocation/process_colocation_sweep.py
```

1c. Generating Colocation Matrix Figures
Generate figures 2a and 2b (as shown in the paper) by running:

```
python3 colocation/gen_colocation_sweep_figures.py
```

The figures will be saved in the `figures` folder.

A.5.3 Monte Carlo Simulations (Figures 7, 8, 9). This set of experiments runs Monte Carlo simulations of randomly generated dynamic demand/workload schedules and colocation scenarios.

Using Provided Experimental Data You must have processed the colocation experiment data (steps 1a and 1b) to run the Monte Carlo simulations. Alternatively, use the provided pre-processed colocation data by copying the contents of `colocation/ref-results` to `colocation/results`.

2a. Dynamic Demand/Workload Schedule Simulations Run the dynamic demand Monte Carlo simulation:

```
python3 monte-carlo-simulations/dynamic-demand/\
dynamic_demand_sim.py --trials 10000 \
--max_workloads 22 --min_time_slices 4 \
--max_time_slices 10 --num_workers 20
```

To run a faster simulation, reduce the simulation scale:

```
python3 monte-carlo-simulations/dynamic-demand/\
dynamic_demand_sim.py --trials 1000 \
--max_workloads 18 --min_time_slices 4 \
--max_time_slices 8 --num_workers 20
```

Generate figure 7:

```
python3 monte-carlo-simulations/dynamic-demand/\
gen_dynamic_demand_sim_figures.py
```

2b. Colocation Scenario Simulations
Run the colocation scenario Monte Carlo simulation:

```
python3 monte-carlo-simulations/colocation/\
colocation_sim.py --trials 10000 --min_workloads 4 \
--max_workloads 100 --min_grid_ci 0 --max_grid_ci \
1000 --min_samples 1 --max_samples 15 --num_workers 20
```

To run a faster simulation, reduce the simulation scale:

```
python3 monte-carlo-simulations/colocation/\
colocation_sim.py --trials 1000 --min_workloads 4 \
--max_workloads 50 --min_grid_ci 0 --max_grid_ci \
500 --min_samples 1 --max_samples 15 --num_workers 20
```

Generate figures 8 and 9:

```
python3 monte-carlo-simulations/colocation/\
gen_colocation_sim_figures.py
```

A.5.4 Demand Forecasting Evaluation (Figures 5 and 11). This step evaluates how Fair-CO2 responds to forecasting error. It uses the Microsoft Azure 2017 VM traces and includes a 30-day resource allocation time series from `forecast/azure-time-series.csv`. Using Meta's Prophet tool, the last 9 days of CPU core allocation are forecasted from the first 21 days of data. Fair-CO2 is applied to both the full 30-day trace and the generated 30-day trace (with 9 days forecast data), and the resulting embodied carbon intensity signals are compared.

Run the following script:

```
python3 forecast/gen_forecast_eval_figures.py
```

Figures 5 and 11 will be saved in the `figures` folder.

A.5.5 Workload Carbon Optimization (Figures 10, 12, and 13). This step runs runtime parameter sweep experiments for the workload carbon optimization case study. These experiments can take several days to over a week. Workloads may crash or stall on low-memory configurations, so frequent monitoring may be required.

General Setup for Optimization Experiments
A large swap space is required since physical memory is constrained in experiments 4a and 4b. If storage is a concern, FAISS indices may need to be deleted.

```
rm -rf workloads/faiss/indices
source setup-scripts/create_large_swap.sh
```

4a. Apache Spark Parameter Sweep
Run the script to sweep across CPU cores and memory for Apache Spark. Results are saved in `workload-optimization/results/spark` (this may take over a day).

```
python3 workload-optimization/spark/spark_sweep.py
```

To run a small subset for demo purposes:

```
python3 workload-optimization/spark/spark_sweep.py \
--small
```

Process the Spark sweep logs:

```
python3 workload-optimization/spark/process_spark.py
```

Using Provided Experimental Data: Alternatively, copy the contents from `workload-optimization/ref-results/spark` to `workload-optimization/results/spark`.

4b. PBBS Parameter Sweep
Run the following script to sweep across CPU cores and memory for eight PBBS workloads. (this may take over a week).

```
python3 workload-optimization/pbbs/pbbs_sweep.py
```

For a small subset:

```
python3 workload-optimization/pbbs/pbbs_sweep.py --small
```

Process the PBBS sweep logs:

```
python3 workload-optimization/pbbs/process_pbbs.py
```

Using Provided Experimental Data: Alternatively, copy the contents from `workload-optimization/ref-results/pbbs` to `workload-optimization/results/pbbs`.

4c. FAISS Parameter Sweep (Figures 12 and 13)
Sweep and Process FAISS Results
Run the FAISS parameter sweep:

```
python3 workload-optimization/faiss/faiss_sweep.py
```

For a small subset:

```
python3 workload-optimization/faiss/faiss_sweep.py \
--small
```

Process the FAISS sweep logs:

```
python3 workload-optimization/faiss/process_faiss.py
```

Using Provided Experimental Data: Alternatively, copy the contents from `workload-optimization/ref-results/faiss` to `workload-optimization/results/faiss`.

Generating Dynamic Workload Optimization Figures 12 and 13

```
python3 workload-optimization/gen_dyn_wl_figure.py
```

4d. Generate Workload Optimization Summary Figure (Figure 10)
Run these scripts in order:

```
python3 workload-optimization/faiss_spark_grid_ci_sweep.py
python3 workload-optimization/gen_sweep_summary_figure.py
```

A.6 Evaluation and expected results

The experiments and reproducible results include the workload pairwise colocation characterization (Fig. 2), the Monte Carlo simulations and attribution fairness evaluation for different colocation scenarios (Fig. 8, 9), the Monte Carlo simulations and attribution fairness evaluation for different dynamic demand scenarios (Fig. 7), demonstration and evaluation of attribution with demand forecasting (Fig. 5, 11), and the case study on dynamic workload optimization (Fig. 10, 12, 13).

A.7 Notes

You may alternatively choose to follow the included README.md which can be easier to follow due to Markdown's better formatting. It also provides more details, including how to generate datasets.

References

[1] Bilge Acun, Benjamin Lee, Fiodar Kazhamiaka, Kiwan Maeng, Udit Gupta, Manoj Chakkaravarthy, David Brooks, and Carole-Jean Wu. 2023. Carbon Explorer: A Holistic Framework for Designing Carbon Aware Datacenters. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (Vancouver, BC, Canada) *(ASPLOS 2023)*. Association for Computing Machinery, New York, NY, USA, 118–132. https://doi.org/10.1145/3575693.3575754

[2] Johan Albrecht, Delphine François, and Koen Schoors. 2002. A Shapley decomposition of carbon emissions without residuals. *Energy Policy* 30, 9 (2002), 727–736. https://doi.org/10.1016/S0301-4215(01)00131-8

[3] Marcelo Amaral, Huamin Chen, Tatsuhiro Chiba, Rina Nakazawa, Sunyanan Choochotkaew, Eun Kyung Lee, and Tamar Eilam. 2023. Kepler: A Framework to Calculate the Energy Consumption of Containerized Applications. In *2023 IEEE 16th International Conference on Cloud Computing (CLOUD)*. 69–71. https://doi.org/10.1109/CLOUD60044.2023.00017

[4] Daniel Anderson, Guy E. Blelloch, Laxman Dhulipala, Magdalen Dobson, and Yihan Sun. 2022. The problem-based benchmark suite (PBBS), V2. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (Seoul, Republic of Korea) *(PPoPP '22)*. Association for Computing Machinery, New York, NY, USA, 445–447. https://doi.org/10.1145/3503221.3508422

[5] Elliot Anshelevich, Anirban Dasgupta, Jon Kleinberg, Éva Tardos, Tom Wexler, and Tim Roughgarden. 2008. The Price of Stability for Network Design with Fair Cost Allocation. *SIAM J. Comput.* 38, 4 (2008), 1602–1623. https://doi.org/10.1137/070680096 arXiv:https://doi.org/10.1137/070680096

[6] Hugo Barbalho, Patricia Kovaleski, Beibin Li, Luke Marshall, Marco Molinaro, Abhisek Pan, Eli Cortez, Matheus Leao, Harsh Patwari, Zuzu Tang, Tamires Santos, Larissa Rozales Gonçalves, David Dion, Thomas Moscibroda, and Ishai Menache. 2023. Virtual Machine Allocation with Lifetime Predictions. In *MLSys*. https://www.microsoft.com/en-us/research/publication/virtual-machine-allocation-with-lifetime-predictions/

[7] Andrea Borghesi, Carmine Di Santi, Martin Molan, Mohsen Seyedkazemi Ardebili, Alessio Mauri, Massimiliano Guarrasi, Daniela Galetti, Mirko Cestari, Francesco Barchi, Luca Benini, Francesco Beneventi, and Andrea Bartolini. 2023. M100 ExaData: a data collection campaign on the CINECA's Marconi100 Tier-0 supercomputer. *Scientific Data* 10, 1 (May 2023), 288. https://doi.org/10.1038/s41597-023-02174-3

[8] James Briggs. 2022. *Faiss: The Missing Manual*. Pinecone. https://www.pinecone.io/learn/series/faiss/

[9] Mohak Chadha, Thandayuthapani Subramanian, Eishi Arima, Michael Gerndt, Martin Schulz, and Osama Abboud. 2023. GreenCourier: Carbon-Aware Scheduling for Serverless Functions. In *Proceedings of the 9th International Workshop on Serverless Computing* (Bologna, Italy) *(WoSC '23)*. Association for Computing Machinery, New York, NY, USA, 18–23. https://doi.org/10.1145/3631295.3631396

[10] Quan Chen, Shuai Xue, Shang Zhao, Shanpei Chen, Yihao Wu, Yu Xu, Zhuo Song, Tao Ma, Yong Yang, and Minyi Guo. 2020. Alita: comprehensive performance isolation through bias resource management for public clouds. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Atlanta, Georgia) *(SC '20)*. IEEE Press, Article 32, 13 pages.

[11] Andrew A. Chien, Richard Wolski, and Fan Yang. 2015. The Zero-Carbon Cloud: High-Value, Dispatchable Demand for Renewable Power Generators. *The Electricity Journal* 28, 8 (2015), 110–118. https://doi.org/10.1016/j.tej.2015.09.010

[12] Sunyanan Choochotkaew, Chen Wang, Huamin Chen, Tatsuhiro Chiba, Marcelo Amaral, Eun Kyung Lee, and Tamar Eilam. 2023. Advancing Cloud Sustainability: A Versatile Framework for Container Power Model Training. In *2023 31st International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. 1–4. https://doi.org/10.1109/MASCOTS59514.2023.10387542

[13] Francesco Ciardiello, Andrea Genovese, and Andrew Simpson. 2020. A unified cooperative model for environmental costs in supply chains: the Shapley value for the linear case. *Annals of Operations Research* 290, 1-2 (July 2020), 421–437. https://doi.org/10.1007/s10479-018-3028-3

[14] Google Cloud. 2023. Carbon Footpint. Retrieved May 2, 2024 from https://cloud.google.com/carbon-footprint

[15] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles* (Shanghai, China) *(SOSP '17)*. Association for Computing Machinery, New York, NY, USA, 153–167. https://doi.org/10.1145/3132747.3132772

[16] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles* (Shanghai, China) *(SOSP '17)*. Association for Computing Machinery, New York, NY, USA, 153–167. https://doi.org/10.1145/3132747.3132772

[17] Mian Dong, Tian Lan, and Lin Zhong. 2014. Rethink energy accounting with cooperative game theory. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking* (Maui, Hawaii, USA) *(MobiCom '14)*. Association for Computing Machinery, New York, NY, USA, 531–542. https://doi.org/10.1145/2639108.2639128

[18] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. (2024). arXiv:2401.08281 [cs.LG]

[19] Lieven Eeckhout. 2024. FOCAL: A First-Order Carbon Model to Assess Processor Sustainability. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (La Jolla, CA, USA) *(ASPLOS '24)*. Association for Computing Machinery, New York, NY, USA, 401–415. https://doi.org/10.1145/3620665.3640415

[20] Joan Feigenbaum, Christos H. Papadimitriou, and Scott Shenker. 2001. Sharing the Cost of Multicast Transmissions. *J. Comput. System Sci.* 63, 1 (2001), 21–41. https://doi.org/10.1006/jcss.2001.1754

[21] Green Software Foundation. 2023. How Accenture Implemented the SCI Specification Score to Track Software Emissions. https://greensoftware.foundation/articles/how-accenture-implemented-the-sci-specification-score-to-track-software-emissions

[22] Green Software Foundation. 2024. SCI Guidance. https://sci-guide.greensoftware.foundation/

[23] Charlotte Freitag, Mike Berners-Lee, Kelly Widdicks, Bran Knowles, Gordon S. Blair, and Adrian Friday. 2021. The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations. *Patterns* 2, 9 (2021), 100340. https://doi.org/10.1016/j.patter.2021.100340

[24] Joshua Fried, Zhenyuan Ruan, Amy Ousterhout, and Adam Belay. 2020. Caladan: Mitigating Interference at Microsecond Timescales. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, 281–297. https://www.usenix.org/conference/osdi20/presentation/fried

[25] Georgi Gerganov and other contributors. 2025. Llama.cpp. Github repository. Retrieved February 21, 2025 from https://github.com/ggml-org/llama.cpp

[26] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. 2011. Dominant Resource Fairness: Fair Allocation of Multiple Resource Types. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*. USENIX Association, Boston, MA. https://www.usenix.org/conference/nsdi11/dominant-resource-fairness-fair-allocation-multiple-resource-types

[27] Robert Grandl, Mosharaf Chowdhury, Aditya Akella, and Ganesh Ananthanarayanan. 2016. Altruistic Scheduling in Multi-Resource Clusters. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. USENIX Association, Savannah, GA, 65–80. https://www.usenix.org/conference/osdi16/technical-sessions/presentation/grandl_altruistic

[28] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini,

Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783

[29] Xiaoding Guan, Noman Bashir, David Irwin, and Prashant Shenoy. 2024. WattScope: Non-intrusive Application-level Power Disaggregation in Datacenters. *SIGMETRICS Perform. Eval. Rev.* 51, 4 (feb 2024), 24–25. https://doi.org/10.1145/3649477.3649491

[30] Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S. Lee, David Brooks, and Carole-Jean Wu. 2022. ACT: Designing Sustainable Computer Systems with an Architectural Carbon Modeling Tool. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (New York, New York) *(ISCA '22)*. Association for Computing Machinery, New York, NY, USA, 784–799. https://doi.org/10.1145/3470496.3527408

[31] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. 2021. Chasing carbon: The elusive environmental footprint of computing. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 854–867.

[32] Ori Hadary, Luke Marshall, Ishai Menache, Abhisek Pan, David Dion, Esaias E Greeff, Star Dorminey, Shailesh Joshi, Yang Chen, Mark Russinovich, and Thomas Moscibroda. 2020. Protean: VM Allocation Service at Scale. In *OSDI*. USENIX. https://www.microsoft.com/en-us/research/publication/protean-vm-allocation-service-at-scale/

[33] Hongyu Hè, Michal Friedman, and Theodoros Rekatsinas. 2023. EnergAt: Fine-Grained Energy Attribution for Multi-Tenancy. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems* (Boston, MA, USA) *(HotCarbon '23)*. Association for Computing Machinery, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3604930.3605716

[34] Mohammad A. Islam and Shaolei Ren. 2016. A New Perspective on Energy Accounting in Multi-Tenant Data Centers. In *USENIX Workshop on Cool Topics on Sustainable Data Centers (CoolDC 16)*. USENIX Association, Santa Clara, CA. https://www.usenix.org/conference/cooldc16/workshop-program/presentation/islam

[35] Shixin Ji, Zhuoping Yang, Xingzhen Chen, Stephen Cahoon, Jingtong Hu, Yiyu Shi, Alex K. Jones, and Peipei Zhou. 2024. SCARIF: Towards Carbon Modeling of Cloud Servers with Accelerators. arXiv:2401.06270 [cs.DC] https://arxiv.org/abs/2401.06270

[36] Shixin Ji, Zhuoping Yang, Alex K Jones, and Peipei Zhou. 2024. Advancing Environmental Sustainability in Data Centers by Proposing Carbon Depreciation Models. *arXiv preprint arXiv:2403.04976* (2024).

[37] Max Junqueira, Luiz Carlos da Costa, Luiz Augusto Barroso, Gerson C. Oliveira, Luiz Mauricio Thome, and Mario Veiga Pereira. 2007. An Aumann-Shapley Approach to Allocate Transmission Service Cost Among Network Users in Electricity Markets. *IEEE Transactions on Power Systems* 22, 4 (2007), 1532–1546. https://doi.org/10.1109/TPWRS.2007.907133

[38] Baolin Li, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. 2023. Clover: Toward Sustainable AI with Carbon-Aware Machine Learning Inference Service. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '23)*. ACM, 1–15. https://doi.org/10.1145/3581784.3607034

[39] Suyi Li, Wei Wang, Jun Yang, Guangzhen Chen, and Daohe Lu. 2023. Golgi: Performance-Aware, Resource-Efficient Function Scheduling for Serverless Computing. In *Proceedings of the 2023 ACM Symposium on Cloud Computing* (Santa Cruz, CA, USA) *(SoCC '23)*. Association for Computing Machinery, New York, NY, USA, 32–47. https://doi.org/10.1145/3620678.3624645

[40] Yueying Li, Zhanqiu Hu, Esha Choukse, Rodrigo Fonseca, G Edward Suh, and Udit Gupta. 2025. EcoServe: Designing Carbon-Aware AI Inference Systems. *arXiv preprint arXiv:2502.05043* (2025).

[41] Yadong Li, Dimitri Offengenden, and Jan Burgy. 2019. *Reduced Form Capital Optimization*. Papers 1905.05911. arXiv.org. https://ideas.repec.org/p/arx/papers/1905.05911.html

[42] Paul Lin, Robert Bunger, and Victor Avelar. 2023. *Quantifying Data Center Scope 3 GHG Emissions to Prioritize Reduction Efforts*. Technical Report.

[43] S. C. Littlechild and G. F. Thompson. 1977. Aircraft Landing Fees: A Game Theory Approach. *The Bell Journal of Economics* 8, 1 (1977), 186–204. http://www.jstor.org/stable/3003493

[44] Qiuyun Llull, Songchun Fan, Seyed Majid Zahedi, and Benjamin C. Lee. 2017. Cooper: Task Colocation with Cooperative Games. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 421–432. https://doi.org/10.1109/HPCA.2017.22

[45] Alexandra Sasha Luccioni, Yacine Jernite, and Emma Strubell. 2023. Power Hungry Processing: Watts Driving the Cost of AI Deployment? arXiv:2311.16863 [cs.LG]

[46] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. 2023. Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model. 24 (2023), 1–15.

[47] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 4765–4774. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[48] Electricity Maps. 2024. US-CAISO 2023 Hourly Carbon Intensity Data (Version January 17, 2024). Website. Retrieved Nov. 22, 2024 from https://www.

electricitymaps.com/data-portal/

[49] Jason Mars, Lingjia Tang, Robert Hundt, Kevin Skadron, and Mary Lou Soffa. 2011. Bubble-Up: increasing utilization in modern warehouse scale computers via sensible co-locations. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture* (Porto Alegre, Brazil) *(MICRO-44)*. Association for Computing Machinery, New York, NY, USA, 248–259. https://doi.org/10.1145/2155620.2155650

[50] McKinsey and Company. 2023. *Investing in the rising data center economy.* Technical Report.

[51] Meta. 2024. Growing Our Commitment to Carbon Removal with the U.S. Department of Energy. Retrieved November 13, 2024 from https://sustainability.atmeta.com/blog/2024/10/11/growing-our-commitment-to-carbon-removal-with-the-u-s-department-of-energy/

[52] Microsoft. 2023. Emissions Impact Dashboard. Retrieved December 15, 2023 from https://www.microsoft.com/en-us/sustainability/emissions-impact-dashboard

[53] Hervé Moulin. 2003. *Fair Division and Collective Welfare.* The MIT Press. https://doi.org/10.7551/mitpress/2954.001.0001

[54] Dejan Novaković, Nedeljko Vasić, Stanko Novaković, Dejan Kostić, and Ricardo Bianchini. 2013. DeepDive: transparently identifying and managing performance interference in virtualized environments. In *Proceedings of the 2013 USENIX Conference on Annual Technical Conference* (San Jose, CA) *(USENIX ATC'13)*. USENIX Association, USA, 219–230.

[55] Thinkstep on behalf of Dell Technologies. 2019. *Life Cycle Assessment of Dell R740.* Technical Report.

[56] Himanshu Pandey. 2023. *Software tactics for sustainable cloud : an empirical study.* Ph. D. Dissertation. https://lutpub.lut.fi/handle/10024/166174 Accepted: 2023-08-08T06:45:04Z.

[57] Leon Petrosjan and Georges Zaccour. 2003. Time-consistent Shapley value allocation of pollution cost reduction. *Journal of Economic Dynamics and Control* 27, 3 (2003), 381–398. https://doi.org/10.1016/S0165-1889(01)00053-7

[58] Sundar Pichai. 2020. Our third decade of climate action: Realizing a carbon-free future. Retrieved December 15, 2023 from https://blog.google/outreach-initiatives/sustainability/our-third-decade-climate-action-realizing-carbon-free-future/

[59] Meikel Poess, Bryan Smith, Lubor Kollar, and Paul Larson. 2002. TPC-DS, taking decision support benchmarking to the next level. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data* (Madison, Wisconsin) *(SIGMOD '02)*. Association for Computing Machinery, New York, NY, USA, 582–587. https://doi.org/10.1145/564691.564759

[60] Lucian Popa, Gautam Kumar, Mosharaf Chowdhury, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. 2012. FairCloud: sharing the network in cloud computing. *SIGCOMM Comput. Commun. Rev.* 42, 4 (Aug. 2012), 187–198. https://doi.org/10.1145/2377677.2377717

[61] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, et al. 2020. Mlperf inference benchmark. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 446–459.

[62] Benedek Rozemberczki, Lauren Watson, Péter Bayer, Hao-Tsung Yang, Olivér Kiss, Sebastian Nilsson, and Rik Sarkar. 2022. The Shapley Value in Machine Learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 5572–5579. https://doi.org/10.24963/ijcai.2022/778 Survey Track.

[63] Andreas Schmidt, Gregory Stock, Robin Ohs, Luis Gerhorst, Benedict Herzog, and Timo Hönig. 2023. carbond: An Operating-System Daemon for Carbon Awareness. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems* (Boston, MA, USA) *(HotCarbon '23)*. Association for Computing Machinery, New York, NY, USA, Article 2, 6 pages. https://doi.org/10.1145/3604930.3605707

[64] Victor Schmidt, Kamal Goyal, Aditya Joshi, Boris Feld, Liam Conell, Nikolas Laskaris, Doug Blank, Jonathan Wilson, Sorelle Friedler, and Sasha Luccioni. 2021. CodeCarbon: Estimate and track carbon emissions from machine learning computing. https://doi.org/10.5281/zenodo.11097062

[65] Ian Schneider and Taylor Mattia. 2024. Carbon accounting in the Cloud: a methodology for allocating emissions across data center users. arXiv:2406.09645 [cs.SE] https://arxiv.org/abs/2406.09645

[66] Amazon Web Services. 2023. AWS Customer Carbon Footprint Tool Overview. Retrieved May 2, 2024 from https://aws.amazon.com/aws-cost-management/aws-customer-carbon-footprint-tool/

[67] Amazon Web Services. 2024. Understanding your carbon emission estimations. Retrieved May 2, 2024 from https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/ccft-estimation.html

[68] Lloyd S. Shapley. 1951. *Notes on the N-Person Game; II: The Value of an N-Person Game.* RAND Corporation, Santa Monica, CA. https://doi.org/10.7249/RM0670

[69] David Shue, Michael J. Freedman, and Anees Shaikh. 2012. Performance Isolation and Fairness for Multi-Tenant Cloud Storage. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*. USENIX Association, Hollywood, CA, 349–362. https://www.usenix.org/conference/osdi12/technical-sessions/presentation/shue

[70] Brad Smith. 2020. Microsoft will be carbon negative by 2030. Retrieved December 15, 2023 from https://blogs.microsoft.com/blog/2020/01/16/microsoft-will-be-carbon-negative-by-2030/

[71] Green Coding Solutions. 2024. Green Metrics Tool. Github repository. Retrieved May 2, 2024 from https://github.com/green-coding-solutions/green-metrics-tool

[72] Emily Sommer, Mike Adler, John Perkins, Joshua Thiel, Hilary Young, Chelsea Mozen, Dany Daya, and Katherine Sundstrom. 2020. Cloud Jewels: Estimating kWh in the Cloud. https://www.etsy.com/codeascraft/cloud-jewels-estimating-kwh-in-the-cloud?utm_source=OpenGraph&utm_medium=PageTools&utm_campaign=Share

[73] Yonglak Son, Udit Gupta, Andrew McCrabb, Young Geun Kim, Valeria Bertacco, David Brooks, and Carole-Jean Wu. 2025. GreenScale: Carbon Optimization for Edge Computing. *IEEE Internet of Things Journal* (2025).

[74] Abel Souza, Noman Bashir, Jorge Murillo, Walid Hanafy, Qianlin Liang, David Irwin, and Prashant Shenoy. 2023. 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* (Vancouver, BC, Canada) *(ASPLOS 2023)*. Association for Computing Machinery, New York, NY, USA, 252–265. https://doi.org/10.1145/3575693.3575709

[75] Amazon Staff. 2023. Everything you need to know about The Climate Pledge. Retrieved December 15, 2023 from https://www.aboutamazon.com/news/sustainability/what-is-the-climate-pledge

[76] Chetan Choppali Sudarshan, Nikhil Matkar, Sarma Vrudhula, Sachin S. Sapatnekar, and Vidya A. Chhabria. 2024. ECO-CHIP: Estimation of Carbon Footprint of Chiplet-based Architectures for Sustainable VLSI. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 671–685. https://doi.org/10.1109/HPCA57654.2024.00058

[77] Swamit Tannu and Prashant J. Nair. 2023. The Dirty Secret of SSDs: Embodied Carbon. *SIGENERGY Energy Inform. Rev.* 3, 3 (oct 2023), 4–9. https://doi.org/10.1145/3630614.3630616

[78] Sean J. Taylor and Benjamin Letham. 2018. Forecasting at Scale. *The American Statistician* 72, 1 (2018), 37–45. https://doi.org/10.1080/00031305.2017.1380080 arXiv:https://doi.org/10.1080/00031305.2017.1380080

[79] Thoughtworks. 2023. Cloud Carbon Footprint. Retrieved May 2, 2024 from https://www.cloudcarbonfootprint.org/

[80] Huangshi Tian, Suyi Li, Ao Wang, Wei Wang, Tianlong Wu, and Haoran Yang. 2022. Owl: performance-aware scheduling for resource-efficient function-as-a-service cloud. In *Proceedings of the 13th Symposium on Cloud Computing* (San Francisco, California) *(SoCC '22)*. Association for Computing Machinery, New York, NY, USA, 78–93. https://doi.org/10.1145/3542929.3563470

[81] Muhammad Tirmazi, Adam Barker, Nan Deng, Md Ehtesam Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. 2020. Borg: the Next Generation. In *EuroSys'20*. Heraklion, Crete.

[82] Nedeljko Vasić, Dejan Novaković, Svetozar Miučin, Dejan Kostić, and Ricardo Bianchini. 2012. DejaVu: accelerating resource allocation in virtualized environments. *SIGPLAN Not.* 47, 4 (March 2012), 423–436. https://doi.org/10.1145/2248487.2151021

[83] Simon Voswinkel, Jonas Höckner, Abuzar Khalid, and Christoph Weber. 2022. Sharing congestion management costs among system operators using the Shapley value. *Applied Energy* 317 (2022), 119039. https://doi.org/10.1016/j.apenergy.2022.119039

[84] Midhul Vuppalapati, Giannis Fikioris, Rachit Agarwal, Asaf Cidon, Anurag Khandelwal, and Éva Tardos. 2023. Karma: Resource Allocation for Dynamic Demands. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*. USENIX Association, Boston, MA, 645–662. https://www.usenix.org/conference/osdi23/presentation/vuppalapati

[85] Gengliang Wang, Reynold Xin, and Jules Damji. 2018. Benchmarking Apache Spark on a Single Node Machine. Retrieved Sep 4, 2024 from https://www.databricks.com/blog/2018/05/03/benchmarking-apache-spark-on-a-single-node-machine.html

[86] Jaylen Wang, Daniel S. Berger, Fiodar Kazhamiaka, Celine Irvene, Chaojie Zhang, Esha Choukse, Kali Frost, Rodrigo Fonseca, Brijesh Warrier, Chetan Bansal, Jonathan Stern, Ricardo Bianchini, and Akshitha Sriraman. 2024. Designing Cloud Servers for Lower Carbon. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. 452–470. https://doi.org/10.1109/ISCA59077.2024.00041

[87] Richard Westerhof, Richard Atherton, and Vasilios Andrikopoulos. 2023. An Allocation Model for Attributing Emissions in Multi-tenant Cloud Data Centers. http://arxiv.org/abs/2305.10439 arXiv:2305.10439 [cs].

[88] 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 *Proceedings of the 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

[89] Laurent Van Winckel, Lizzie Boakes, Benjamin Vanhouche, I-Yun Liu, William Ragnarsson, Vincent Schellekens, Cédric Rolin, Lars Åke Ragnarsson, and Swamy

Asha. [n. d.]. *imec.netzero*. https://netzero.imec-int.com/

[90] Hailong Yang, Alex Breslow, Jason Mars, and Lingjia Tang. 2013. Bubble-flux: precise online QoS management for increased utilization in warehouse scale computers. In *Proceedings of the 40th Annual International Symposium on Computer Architecture* (Tel-Aviv, Israel) *(ISCA ’13)*. Association for Computing Machinery, New York, NY, USA, 607–618. https://doi.org/10.1145/2485922.2485974

[91] Yujie Zhao, Yang (Katie) Zhao, Cheng Wan, and Yingyan (Celine) Lin. 2024. 3D-Carbon: An Analytical Carbon Modeling Tool for 3D and 2.5D Integrated Circuits. In *Proceedings of the 61st ACM/IEEE Design Automation Conference* (San Francisco, CA, USA) *(DAC ’24)*. Association for Computing Machinery, New York, NY, USA, Article 178, 6 pages. https://doi.org/10.1145/3649329.3658482