CarbonEdge: Leveraging Mesoscale Spatial Carbon-Intensity Variations for Low Carbon Edge Computing

Li Wu
University of Massachusetts Amherst
USA

Walid A. Hanafy
University of Massachusetts Amherst
USA

Abel Souza
University of California, Santa Cruz
USA

Khai Nguyen
University of Massachusetts Amherst
USA

Jan Harkes
Carnegie Mellon University
USA

David Irwin
University of Massachusetts Amherst
USA

Mahadev Satyanarayanan
Carnegie Mellon University
USA

Prashant Shenoy
University of Massachusetts Amherst
USA

ABSTRACT

The proliferation of latency-critical and compute-intensive edge applications is driving increases in computing demand and carbon emissions at the edge. To better understand carbon emissions at the edge, we analyze granular carbon intensity traces at intermediate "mesoscales," such as within a single US state or among neighboring countries in Europe, and observe significant variations in carbon intensity at these spatial scales. Importantly, our analysis shows that carbon intensity variations, which are known to occur at large continental scales (e.g., cloud regions), also occur at much finer spatial scales, making it feasible to exploit geographic workload shifting in the edge computing context. Motivated by these findings, we propose *CarbonEdge*, a carbon-aware framework for edge computing that optimizes the placement of edge workloads across mesoscale edge data centers to reduce carbon emissions while meeting latency SLOs. We implement *CarbonEdge* and evaluate it on a real edge computing testbed and through large-scale simulations for multiple edge workloads and settings. Our experimental results on a real testbed demonstrate that *CarbonEdge* can reduce emissions by up to 78.7% for a regional edge deployment in central Europe. Moreover, our CDN-scale experiments show potential savings of 49.5% and 67.8% in the US and Europe, respectively, while limiting the one-way latency increase to less than 5.5 ms.

CCS CONCEPTS

• **Computer systems organization → Distributed architectures**; • **Social and professional topics → Sustainability**.

KEYWORDS

Edge Computing, Sustainable Computing, Edge Placement, Edge Orchestration, Mesoscale Carbon Analysis

ACM Reference Format:
Li Wu, Walid A. Hanafy, Abel Souza, Khai Nguyen, Jan Harkes, David Irwin, Mahadev Satyanarayanan, and Prashant Shenoy. 2025. CarbonEdge: Leveraging Mesoscale Spatial Carbon-Intensity Variations for Low Carbon Edge Computing. In *The 34th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '25), July 20–23, 2025, Notre Dame, IN, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3731545.3731576

ACM ISBN 979-8-4007-1869-4/2025/07
https://doi.org/10.1145/3731545.3731576

1 INTRODUCTION

Data centers consumed more than 460 terawatt-hours (TWh) of energy in 2022, and are expected to consume more than 1000 TWh by 2026 [18]. As a result, data centers are already generating roughly 1% of global carbon emissions and could emit more than 2.5 billion metric tons of CO_2 by the end of the decade. The sustainable growth of data center capacity has emerged as a critical challenge in our society's transition to a low-carbon future, especially with the accelerating build-out of data center capacity to satisfy the growing demand for AI workloads. Historically, cloud operators have addressed data center sustainability issues by optimizing their energy efficiency (i.e., their computational work done per unit of energy consumed). However, optimizing energy-efficiency alone will likely not be sufficient to satisfy cloud platforms' carbon emissions targets [3]. In particular, since data center energy-efficiency is already highly optimized after years of research, further optimizations are expected to yield diminishing marginal improvements moving forward.

As a result, researchers have recently focused on several alternative approaches for reducing carbon footprint of cloud data centers and improving their carbon efficiency (i.e., their computational work done per unit of carbon emitted). Specifically, to optimize hyperscale data centers' carbon efficiency, cloud operators have deployed both supply- and demand-side approaches. On the supply-side, cloud operators have procured green energy (e.g., wind or solar) through long-term contracts to power their data center operations [6], while on the demand-side, researchers have explored techniques for modulating data center workload demand and its resulting carbon emissions to optimize their carbon footprint [41]. Since the electric grid in different regions uses different mixes of generation sources, grids with a higher penetration of low-carbon sources, such as hydro, solar, or wind, tend to produce lower-carbon electricity. Spatial workload shifting approaches exploit these regional differences in energy's carbon intensity by proactively shifting workloads to data center locations with lower-carbon energy, thereby performing the same computation while incurring fewer emissions. Recent research has shown that cloud workloads, such as machine learning training and batch processing, are amenable to such spatial shifting optimizations and can yield significant reductions in applications' carbon footprint [9, 13, 27, 38]. However, these optimizations typically incur large network delays to migrate workloads over long distances to a different data centers, and thus can increase user latency for interactive workloads that are latency-sensitive.

Since spatial differences in the grid's carbon intensity are clearly evident over large geographical distances, spatial shifting has largely been studied for cloud workloads at continental scales, i.e., across entire continents or between continents. For example, shifting workloads from eastern to western North America, or shifting workloads from North America or Asia to Europe. The differences in grid carbon intensity at these scales are due to the vastly different generation mixes at distant locations. *As a result, conventional wisdom has held that spatial workload shifting is unsuitable for edge data centers, since moving edge workloads over such long distances to distant edge data centers would result in unacceptable increases in the latency of edge applications.* Consequently, edge data centers thus far have not leveraged this key carbon optimization technique.

In this paper, we challenge this conventional wisdom and show that spatial workload shifting is a feasible carbon-optimization approach for edge data centers deployed in many, although not all, parts of the world. Our key insight is that spatial differences in grid carbon intensity do frequently occur even at "mesoscales" (i.e., smaller distances of tens to a few hundred kilometers), especially as the penetration of wind and solar renewables continues to grow. While, on average, variations in carbon intensity are certainly larger at longer distances, there are meaningful differences in energy's carbon intensity at short distances in many parts of the world. Such mesoscale differences open up new opportunities for spatial workload shifting across nearby edge data centers, enabling edge workloads to optimize their carbon footprint with limited performance impact on latency-sensitive applications. In contrast, as mentioned above, prior work has focused primarily on exploiting spatial differences in energy's carbon intensity at large continental scales, i.e., across a thousand kilometers or more, where variations in energy's carbon intensity arise from large environmental differences. For example, at continental scales, it may be daytime in one location with plentiful solar generation and nighttime in another with zero solar generation. Instead, mesoscale differences in carbon intensity generally arise from differences in a location's specific mix of various generation sources, e.g., hydro, coal, natural gas, oil, solar, wind, nuclear, etc., and types of generators. For example, a municipal utility that serves a small town may have its own low-carbon hydro-generating plant, while nearby towns are served by a private utility that generates most of its power from high-carbon fossil fuels. Such differences give rise to variations in energy's carbon intensity, even at relatively short distances.

Motivated by these observations, this paper presents *CarbonEdge*, a carbon-aware orchestration framework for distributed edge data centers that supports spatial workload shifting at mesoscales. *CarbonEdge* optimizes workload placement to significantly reduce carbon emissions of edge applications within a mesoscale region while satisfying latency constraints. Importantly, *CarbonEdge* considers the diversity of edge applications and resource heterogeneity when determining workload placement, which affects how applications consume energy at specific locations. This aspect is crucial because the carbon emissions of applications depend on both their energy consumption and the carbon intensity of the energy used. We hypothesize that small spatial-scale variations in carbon intensity can enable *CarbonEdge* to reduce the operational carbon footprint of edge applications without significantly impacting their low-latency benefits. In designing, implementing, and evaluating *CarbonEdge*, we make the following contributions.

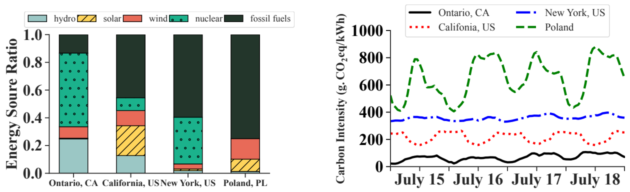
(1) **Mesoscale Carbon Analysis.** Our analysis is the first to demonstrate significant variations in grid carbon intensity at mesoscale distances, thereby making it feasible to deploy workload shifting optimizations in edge computing platforms. We present a detailed empirical analysis of the granular carbon intensity data and latency traces of 148 regions in the world (Section 3).
(2) ***CarbonEdge* Design and Implementation.** Based on our findings, we propose *CarbonEdge*, a carbon-aware placement framework to reduce carbon emissions from edge data centers at mesoscales. *CarbonEdge* integrates carbon intensity variations across edge data center locations and accounts for energy-efficiency differences among heterogeneous resources to intelligently distribute edge workloads to minimize carbon emissions (Section 4). Additionally, we implement a full prototype of a carbon-aware edge orchestration framework on top of Sinfonia, a Kubernetes-based framework for edge data centers, and plan to release it as open source (Section 5).
(3) **Experimental Evaluation.** We evaluate *CarbonEdge* in both edge testbeds and large-scale simulations, using real-world traces, edge workloads, and diverse edge settings. Our experimental results on real testbed demonstrate that *CarbonEdge* can reduce emissions by up to 78.7% in mesoscale regional edge deployments. Furthermore, our CDN-scale simulations indicate that *CarbonEdge* yields 49.5% and 67.8% savings in the US and Europe, respectively, while limiting the one-way latency increase to less than 5.5 ms (Section 6).

2 BACKGROUND

This section provides background on grid energy's carbon intensity, carbon-aware workload optimization, and edge data centers.

2.1 Electric Grid Carbon Intensity

The electricity supplied by the electric grid at a given location comes from a mix of generation sources, such as natural gas, coal, hydro, solar, and wind. The relative proportion of generation from each source varies from one region to another, depending on the types of generation sources present in each region. For example, as shown in Figure 1a, in the Ontario region of Canada, most energy comes from nuclear and hydroelectric energy sources. At the same time, eastern European countries such as Poland have a more significant proportion of coal and natural gas. The carbon intensity of electricity is defined as the total CO_2 emissions per unit of electricity generation and is measured in g·CO_2eq/kWh. For each location, it is computed as the weighted average of the carbon intensity of the



(a) Energy Mix **(b) Carbon Intensity**

Figure 1: Energy mix and carbon intensity of four regions.

source energy mix at that location. Figure 1b shows the carbon intensity of the energy supply in four different countries and regions — Ontario region of Canada, California and New York in the US, and Poland in Europe — and shows that there are significant differences in the carbon intensity of electricity at the spatial granularity of countries or large geographic regions.

As grid operators have begun to report their real-time energy generation mixes, third-party carbon information services, such as Electricity Maps [26] and WattTime [1], have begun exposing this carbon intensity data to data center operators and applications via real-time APIs and forecasting services. Our paper assumes the availability of such carbon intensity data for carbon optimizations in edge data centers.

2.2 Carbon-aware Workload Optimizations

The availability of real-time carbon intensity data has motivated cloud providers and applications to schedule workloads based on variations in the carbon intensity of electricity. The resulting carbon-aware scheduling approaches can be broadly viewed as workload shifting, which exploits temporal and spatial variations in the carbon intensity of electricity. Temporal workload shifting exploits temporal fluctuations in carbon intensity at a given location by scheduling (or delaying) jobs to periods of low carbon intensity. Such techniques are well-suited for batch workloads, which have temporal flexibility and can tolerate delays in their completion times. There have been numerous recent works that leverage temporal workload shifting to reduce the carbon emissions of batch workloads in the cloud [2, 16, 36, 38, 41].

In contrast, spatial workload shifting exploits spatial variations in carbon intensity across locations by moving jobs or requests to data center locations with a lower carbon intensity supply. Such spatial shifting optimizations have been studied in the cloud context by moving cloud workloads across data center locations that span large geographic regions, countries, or even continents [9, 11, 13, 27, 38]. For example, as depicted in Figure 1b, cloud workloads can be moved from the New York region of a public cloud to the Ontario region, whose electricity supply has a lower carbon intensity. In theory, spatial shifting can be implemented for both interactive workloads, such as web services, as well as batch workloads. In practice, however, migrating interactive requests to distant data centers increases their user-perceived latency, and hence, spatial shifting has been primarily utilized for batch applications, such as machine learning training [9]. Prior work has also shown that spatial shifting generally has much more potential for reducing carbon compared to temporal shifting [38]. This insight derives from the fact that there tend to be much larger differences in carbon between locations than within any one location over time.

2.3 Edge Data Centers

Edge computing involves deploying computing resources in the form of small server clusters at the network's edge close to end users. Edge computing is well-suited for low latency services since it avoids network delays incurred by traversing to more distant cloud data centers. For example, regional edge clusters, deployed by edge or even cloud providers, have been used to host latency-sensitive applications such as mobile offloading, augmented reality (AR), and deep learning inference [33, 34]. Another example is a content delivery network that operates large geo-distributed edge clusters and serves web and multimedia content to users from proximate edge locations.

While edge data centers can optimize their carbon footprint via temporal workload shifting, such methods are ill-suited for interactive or latency-sensitive applications that are prevalent at the edge, since such workloads cannot be delayed or time-shifted. In contrast, spatial shifting optimizations have traditionally been performed at larger continental or global scales, i.e., across entire continents or across multiple continents, to exploit carbon intensity variations present at that scale [9, 13, 38]. While these methods work well for cloud-based batch workloads, spatial shifting of interactive edge workloads at such large scales results in large latency increases. Hence, spatial shifting has not been considered for edge applications in prior work. We argue that spatial shifting is feasible even in the edge context by exploiting mesoscale variations in grid carbon intensity that are beginning to appear in today's energy grids.

3 MESOSCALE CARBON ANALYSIS

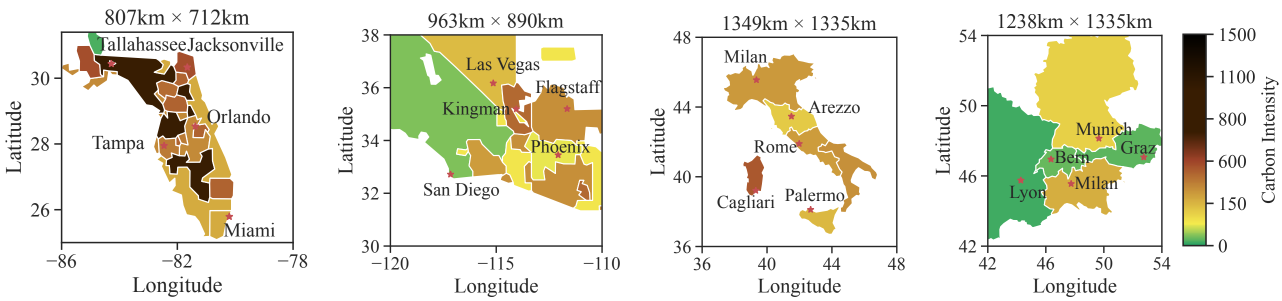
This section presents an empirical study of grid carbon intensity differences that occur over mesoscale geographic distances of tens to hundreds of kilometers. We also analyze the increases in network latency at these scales. Our empirical study seeks to answer two key questions.

(1) *How much does energy's carbon intensity vary within mesoscale regions that span tens to hundreds of kilometers, and are these differences large enough to warrant the use of spatial workload shifting in distributed edge data centers?*
(2) *How prevalent are these types of mesoscale variations in different parts of the world? Are they sufficiently common to warrant the broad deployment of carbon optimization techniques in edge data centers across the world?*

3.1 Carbon Intensity Analysis at Mesoscales

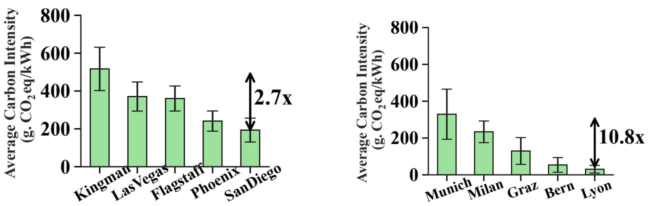
To understand the differences in grid carbon intensity that are seen at mesoscales, we conducted a measurement study where we collected carbon intensity traces for 148 carbon zones worldwide for an entire year (2023). For the purpose of our study, a carbon zone, or simply a *zone*, is a geographic area whose grid operator provides carbon intensity data. The geographic size of a carbon zone depends on the area served by the grid operator and can vary from a city to an entire state or even a small country. Further, we also collected round-trip latency traces from the WonderNetwork [32], which provides ping traces (in milliseconds) to cities across the world. We describe our data sources in Section 6.1.1.

To illustrate the carbon intensity differences at the mesoscale, we first select four specific mesoscale regions, each comprising five



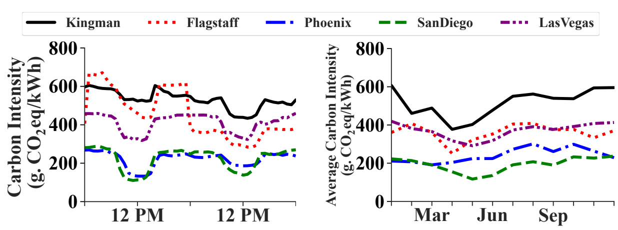
(a) Florida **(b) West US** **(c) Italy** **(d) Central EU**

Figure 2: Carbon intensity snapshots of four mesoscale regions, highlighting variations across zones.



(a) West US **(b) Central EU**

Figure 3: Yearly carbon intensity of two mesoscale regions.



(a) Two-day (Dec 25-27) **(b) Year-long (2023)**

Figure 4: Spatial-temporal variations in carbon intensity over two days and 12 months in 2023 in the West US.

carbon zones, across the United States and Europe. Figure 2 depicts a heat-map of the carbon intensity variations within each mesoscale region for a single hour in 2023, with darker colors representing higher carbon intensity values. We assume that each of the five carbon zones within a mesoscale region has an edge data center. The Florida region, for example, consists of five cities, each hosting an edge data center, that is a few hundred kilometers apart from one another. The figure shows significant differences in carbon intensity values even at this scale, with inter-zone variations of 2.5× in Florida, 7.9× in the west US, 2.2× in Italy, and 19.5× in Central Europe.

Figure 3 then plots the mean carbon intensity over the entire year for two regions. The figure confirms that the differences in carbon intensity persist across the year. Furthermore, the average difference between the maximum and minimum carbon intensity across zones in a region is 2.7× in the west US, and 10.8× in central Europe. Importantly, these differences compare favorably to those reported across global cloud regions. For instance, a recent study of spatial differences in carbon intensity across Amazon's cloud regions reported an order of magnitude difference across AWS cloud regions in Europe and Asia [38]. Moreover, since the relative mix of energy sources changes over time, Figure 4 shows temporal fluctuations in the carbon intensity of edge data centers within each mesoscale region within a day (Figure 4a) and across seasons (Figure 4b). For instance, Flagstaff, AZ (see Figure 4a) exhibits a daily difference of ~300g·CO_2eq/kWh. Figure 4b shows how monthly average carbon intensity changes. For example, Kingman, AZ, exhibits a ~200 g·CO_2eq/kWh change between March and November due to its reliance on solar energy.

Table 1: One-way network latency (ms).

(a) Florida

Location	Miami	Orlando	Tampa	Tallah.
Jacksonville	3.64	5.32	6.86	3.42
Miami	-	4.5	3.37	7.2
Orlando		-	1.86	4.35
Tampa			-	4.14
Tallahassee				-

(b) Central EU

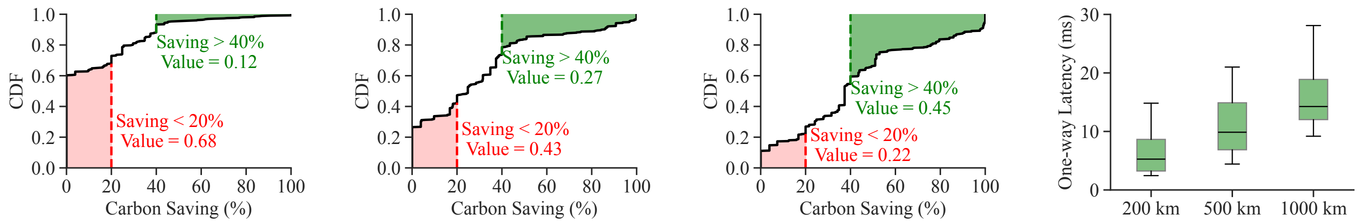
Location	Graz	Lyon	Milan	Munich
Bern, CH	8.78	6.28	6.45	3.985
Graz, AT	-	16.22	11.98	8.36
Lyon, FR		-	9.34	8.82
Milan, IT			-	8.65
Munich, DE				-

Finally, Table 1 shows the pairwise one-way network latency between edge data centers, within two mesoscale regions. The table shows that, unsurprisingly, the latency grows with geographic distance. However, the increase in latency due to shifting workload from one edge location to another ranges from a few milliseconds to ~16 ms, depending on the distance and the network topology between locations.

Key Takeaways. *Our results show significant differences in the carbon intensity of electricity at mesoscale distances, similar to those reported at continental scales between cloud regions. These mesoscale variations demonstrate the feasibility of using spatial workload-shifting optimizations for edge data centers.*

3.2 Mesoscale Analysis across Continents

Having shown that there can be significant differences in carbon intensity at the mesoscale, a key question is whether such differences are commonplace in different parts of the world or confined to a few specific locations. To answer this question, we conduct an analysis of carbon intensity traces across 496 Akamai edge data centers in the United States and Europe. For each edge data center, we find the location with the highest carbon intensity difference within a threshold radius distance D and compute the percentage difference in carbon intensity between the two locations. Figure 5



(a) D = 200 km **(b) D = 500 km** **(c) D = 1000 km** **(d) Radius-Latency**

Figure 5: Carbon savings with search radii of 200 km, 500 km, and 1000 km. (d) One-way latency across pairwise distances.

plots a CDF of the observed pairwise differences for different values of threshold radius D (from D = 200 km to D = 1000 km).

For a radius of 200 km, Figure 5a shows that 32% of the edge data centers have at least one data center with a carbon intensity difference of more than 20%, and 12% of locations have a data center with a carbon intensity difference of more than 40%. At the same time, 68% of the edge data centers do not have any location with a significant spatial carbon intensity difference (i.e., more than 20%). As the radius increases, the chances of finding an edge location with significant carbon intensity differences grow. As shown in Figure 5b and Figure 5c, increasing the radius to 500 km and 1000 km allows 57% and 78% of edge data centers to reduce their emissions by more than 20%. In addition, this increase enables 27% and 45% of edge data centers to *significantly* reduce their carbon emissions by more than 40% for the 500 km and 1000 km radius, respectively. The fraction of edge locations without any significant carbon intensity differences within its radius falls to 22% for D = 1000 km. Lastly, Figure 5d shows that the median increase in latency ranges from 5.3 ms for D = 200 km to 14.3 ms for D = 1000 km.

Key Takeaways. *More than 78% of the edge locations in Europe and North America see carbon intensity differences exceeding 20% within a radius of 1000 km, indicating that mesoscale carbon intensity variations are prevalent in many regions of the world.*[1]

4 *CARBONEDGE* DESIGN AND POLICIES

Motivated by our findings from the mesoscale carbon analysis, we introduce a carbon-aware framework for edge computing, named *CarbonEdge*, which employs the variations of carbon intensity across edge data centers to intelligently distribute edge applications while satisfying the low-latency demands. We formalize our carbon-aware edge placement problem with latency constraints and present an optimization approach to minimize carbon emissions at the edge. Lastly, we present our incremental placement algorithm to our edge placement optimization in a real-world edge system.

4.1 *CarbonEdge* Overview

CarbonEdge is a carbon-aware framework designed to reduce carbon footprint at the edge by spatially distributing workloads across edge data centers. It manages edge data centers dispersed at mesoscales, which have shown prevalently significant variations in carbon intensity (Section 3), and assumes that edge workloads

[1]Our analysis could not be extended to other continents (e.g., Asia, Australia) due to the unavailability of fine-grain spatial carbon intensity data, but we anticipate similar trends will persist as the adoption of renewables continues to grow globally.

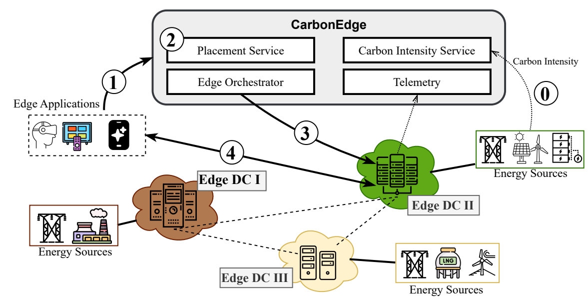


Figure 6: *CarbonEdge* design and exemplar workflow of placing offloading applications from IoT devices.

can shift across edge data centers at this scale. Edge servers are not only diverse in geolocations but also diverse in architecture and capacity, exhibiting significant differences in energy efficiency. As carbon emission is a function of energy consumption and carbon intensity of the grid, we further combine intensity variations across edge data center locations and energy efficiency differences across diverse edge servers to save carbon emissions at the edge. As a result, *CarbonEdge* reduces edge carbon emissions by placing edge applications on energy-efficient edge servers with a sustainable energy supply, respecting the low-latency and resource demands. Moreover, *CarbonEdge* manages the power states of edge servers to reduce emissions from idle servers.

Figure 6 shows an overview of our system. The telemetry and carbon intensity services continuously collect system metrics, energy consumption, and the electricity carbon intensity of edge data centers. In addition, the carbon intensity service periodically predicts the carbon intensity of all data centers (step ⓪). When edge workloads arrive, which can be applications offloaded from resource-limited IoT or mobile devices or applications to be redeployed when an edge server fails (step ①), the placement service instantly decides where to allocate the workloads using a carbon-aware placement policy (step ②). Once the placement decisions are made, the edge orchestrator deploys the applications accordingly (step ③) and establishes the connections to end users (step ④).

4.2 Carbon-aware Edge Placement

The section presents the carbon-aware placement with latency constraints problem. and our proposed policy used in *CarbonEdge*. Our carbon-aware policy minimizes the carbon footprint of edge data

Table 2: Notations used in *CarbonEdge*.

Decision variables:	
x_{ij}	True if application i is placed on server j; False otherwise.
y_j	True if server j is *powered-on*; False otherwise.
Inputs:	
C_j^k	Available capacity in server j of type k
$\bar{I}_j$	Average carbon intensity of server j
B_j	Base power usage of server j when it is *powered-on*
y_j^{curr}	Power state (on or off) of server j before placement optimization
R_{ij}^k	Resource demand of type k of application i when running on server j
E_{ij}	Energy usage of application i when running on server j
L_{ij}	Latency between application i and server j
ℓ_i	Latency requirement of application i
Optimization goal:	
f	Total carbon footprint of edge placement

centers using an incremental optimization-driven approach. Our approach holistically integrates three factors: 1) carbon intensity variations of mesoscale edge data centers, 2) workload and diversity in energy efficiency across heterogeneous edge servers, and 3) base power usage and power proportionality of servers.

Incremental carbon-aware placement. Our carbon-aware placement problem aims to minimize carbon emissions from incrementally placing arriving edge applications across edge data centers while meeting strict latency requirements. Given the current power state of the servers $y_j^{curr} : j \in \mathcal{S}$ and available resource capacity C_j^k, our objective is to place a batch of applications $\mathcal{A}$ on $\mathcal{S}$. First, we define two decision variables that represent the placement and power management decisions, and then define the carbon emissions associated with the placements based on these two variables. Finally, we present the carbon-aware placement problem formulation. Table 2 summarizes the notations.

Decision variables: Let $x_{ij} \in \{0, 1\}$ the placement of application $i \in \mathcal{A}$ on server $j \in \mathcal{S}$, and $y_j \in \{0, 1\}$ indicate whether server j is powered on. These variables are subject to the following four constraint classes.

1) *Multi-dimensional resource constraint*: Edge servers are typically computing, storage, and networking resource-limited and are diverse in capacity and resource types. We define C_j^k represents the available capacity of type k on server j. When application i runs the server j, its resource demands are R_{ij}^k. To ensure the application performance, the aggregated resource demands of applications allocated to a server must not exceed its available resources.

$$\sum_i x_{ij} \cdot R_{ij}^k \leq y_j \cdot C_j^k, \qquad \forall j, k \tag{1}$$

2) *Latency constraint*: Each application i comes from a certain geolocation and has a specific latency limit L_i. The latency from the application's source to the hosting server L_{ij} must remain below this limit.

$$x_{ij} \cdot L_{ij} \leq \ell_i, \qquad \forall i, j \tag{2}$$

3) *Placement constraint*: Each application i is assigned to exactly one server.

$$\sum_j x_{ij} = 1, \qquad \forall i \tag{3}$$

4) *Power state consistency*: An active server cannot be powered off during placement (e.g., to avoid service disruption):

$$y_j^{curr} \leq y_j, \qquad \forall j \tag{4}$$

Algorithm 1 *CarbonEdge* Incremental Placement Algorithm

Input: Arriving applications $\mathcal{A}$, Servers $\mathcal{S}$, Latency requirements ℓ, Resource demands $\mathbf{R}$, and Energy consumption $\mathbf{E}$
Output: Placement ($\mathbf{x}$) and Power ($\mathbf{y}$) decisions.

Initialize latency matrix $\mathbf{L} \leftarrow \emptyset$
for each application $a_i \in \mathcal{A}$ **do**
  for each server $s_j \in \mathcal{S}$ **do**
    $\mathbf{L}_{ij} \leftarrow$ CalculateLatency(a_i, s_j) ▷ Get application-server latency
  end for
end for
$\mathcal{S}', \mathbf{L}' \leftarrow$ FilterFeasibleServers($\mathcal{A}, \mathcal{S}, \mathbf{L}, \ell$) ▷ Select servers satisfying latency limits
$\mathbf{C}, B, \bar{I}, \mathbf{y}^{curr} \leftarrow$ GetServerStates($\mathcal{S}'$) ▷ Available capacities $\mathbf{C}$, base power B, mean carbon intensity $\bar{I}$, current power states $\mathbf{y}^{curr}$
$\mathbf{x}, \mathbf{y} \leftarrow$ SolveOptimization($\mathcal{A}, \mathcal{S}', \ell, \mathbf{R}, \mathbf{E}, \mathbf{C}, B, \bar{I}, \mathbf{y}^{curr}, \mathbf{L}'$) ▷ Solve Equation 7
$\mathbf{C}' \leftarrow$ UpdateServerStates($\mathbf{x}, \mathbf{y}, C$) ▷ Update servers capacity
return $\mathbf{x}, \mathbf{y}$

where y_j^{curr} is the current power state. Additionally, assignments require active servers:

$$x_{ij} \leq y_j, \qquad \forall i, j \tag{5}$$

Carbon emissions: Carbon emissions from edge placement are from two main sources: application operation and server activation. Application operational emissions depend on the energy consumption of applications and the carbon intensity of hosting servers, given by $\sum_i \sum_j x_{ij} \cdot E_{ij} \cdot \bar{I}_j$, where E_{ij} is the energy consumption of application i on server j, and $\bar{I}_j$ is the average carbon intensity of the hosting server j. Note that carbon intensity varies over time depending on the mix of energy sources available in that area. $\bar{I}_j$ represents the average of the forecast carbon intensity values of server j. Server activation emissions depend on the base power B_j and the carbon intensity $\bar{I}_j$, represented as $\sum_j (y_j - y_j^{curr}) \cdot B_j \cdot \bar{I}_j$, where $(y_j - y_j^{curr})$ represents the newly activated server. The total carbon emissions for placing applications are:

$$f = \underbrace{\sum_i \sum_j x_{ij} \cdot E_{ij} \cdot \bar{I}_j}_{\text{Application operation}} + \underbrace{\sum_j (y_j - y_j^{curr}) \cdot B_j \cdot \bar{I}_j}_{\text{Server activation}} \tag{6}$$

Problem formulation: The carbon-aware placement policy identifies the optimal placement x_{ij}^* and power management y_j^* solutions to minimize carbon emissions at the edge while meeting the latency and resource constraints. The problem is formulated as follows:

$$\min_{x_{ij}^*, y_j^*} \quad f \qquad \text{s.t. Constraints 1-5 hold.} \tag{7}$$

Extensions. The above formulation focuses on reducing carbon emissions as a primary goal while considering edge latency as a constraint. Alternatively, a multi-objective optimization can be employed to minimize both carbon emissions and latency. Additionally, other objectives like energy usage can also be optimized alongside carbon in a similar fashion. In Section 6.4, we demonstrate the benefits of such a multi-objective optimization strategy, which navigates the trade-offs between carbon emissions and energy usage in edge computing.

4.3 *CarbonEdge* Placement Algorithm

Our placement algorithm assumes that latency-sensitive edge applications may arrive unpredictably and need to be placed onto edge data centers in a carbon-aware manner. To achieve this, *CarbonEdge* employs an incremental placement strategy, executing the algorithm periodically to process newly arriving applications as a batch, to ensure carbon-efficient placement without global recomputation, as shown in Algorithm 1.

The algorithm places a set of arriving applications $\mathcal{A}$ (with latency requirements ℓ, resource demands $\mathbf{R}$, and energy profiles $\mathbf{E}$) across edge servers $\mathcal{S}$ in mesoscale data centers. First, it computes application-server latency $\mathbf{L}$ and prunes (i.e., filters out) servers exceeding latency constraints, retaining only feasible candidates $\mathcal{S}' \in \mathcal{S}$ (line 1-8). Next, it retrieves server telemetry (available capacity $\mathbf{C}$, base power B, current power state y^{curr}) and mean carbon intensity $\bar{I}$ (line 9). Then, it solves the optimization in Equation 7 using the latency-compliant server set $\mathcal{S}'$ to ensure traceability. Placing incoming applications in small batches in real-time can be done efficiently—our result in Section 6.5 shows that incremental placement of a batch of 50 newly arriving applications across 400 servers completes in 3 seconds (line 10). Finally, *CarbonEdge* commits the resource allocation and power state transition, updating server states for the next iteration (line 11).

5 *CARBONEDGE* IMPLEMENTATION

This section describes the implementation of *CarbonEdge* (See Figure 6) on top of Sinfonia [35], a Kubernetes based open-source orchestrator for edge-native applications. Our implementation adds ~4k SLOC to the Sinfonia system and is available as open source at https://github.com/umassos/CarbonEdge.

5.1 *CarbonEdge* Prototype

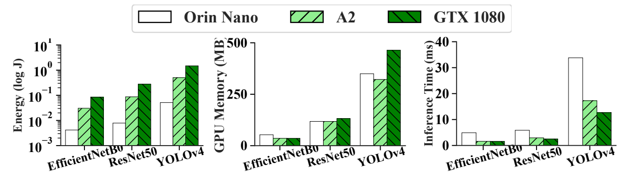
Our *CarbonEdge* consists of the following components that we added to Sinfonia.

Telemetry Service: Our telemetry service is integrated into Sinfonia's telemetry, where it collects static (e.g., location and IP address) attributes and real-time (e.g., utilization) metrics. Real-time metrics are collected based on the Prometheus monitoring stack[31]. We augment Sinfonia's monitoring with the following metrics:

(1) **Power Monitoring**: We measure the power consumption of CPU servers using RAPL [8], and we leveraged Prometheus's DGCM exporter for GPUs [28].
(2) **Carbon Intensity**: We integrate a carbon intensity service that replays historical traces from Electricity Maps [26] and uses the traces to provide real-time and forecast carbon intensity.
(3) **Carbon Monitoring**: We implement carbon monitoring based on energy usage and the carbon intensity of the selected edge sites, where we account for the base power (if the server is turned on) and applications' energy usage.
(4) **End-to-end latency**: In addition to latency across sites, we recorded end-to-end latency between users and their deployed applications.

Profiling Service: We implement an application profiling service that collects the application's performance metrics, such as latency, power consumption, resource demands, and other crucial information, to make accurate placement decisions across available resources. Our profiling service can be replaced with performance models that statically analyze the applications and predict the latency and energy consumption [5, 30].

Placement Service: Lastly, we implement our placement policy (Algorithm 1) on Sinfonia as a matching policy. The placement policy utilizes the system's real-time metrics, static attributes of



(a) Energy **(b) GPU Memory** **(c) Inference Time**

Figure 7: Energy consumption, memory usage, and inference time of ML workloads across devices.

different edge sites, and workload profiles to determine optimal placements and server activation. Our implementation batches deployment requests (e.g., every 5 minutes) and solves the optimization problem per application batch using the Google OR-Tools [29]. We demonstrate the effectiveness of our approach in Section 6.5. After computing the placement decisions, we utilize Sinfonia's orchestration capabilities to initiate the deployment sequence (Sinfonia RECIPE), which contains the necessary Kubernetes deployment files and helm charts, to the destination servers or activate servers if necessary, and inform the client(s) of the destination's address. Note that although Sinfonia and our system are packed with fault-tolerance and reconfiguration capabilities, evaluating them is beyond the scope of this paper.

5.2 *CarbonEdge* Edge Simulator

In addition to the prototype of *CarbonEdge*, we developed a simulator for larger-scale evaluations that is not feasible using an edge testbed. Our simulator supports simulating diverse edge settings with dynamic workloads and heterogeneous servers. This simulator represents the components of Sinfonia and follows the same decision process and metrics, where we implement our proposed carbon-aware placement policy and other baseline policies using Google OR-Tools [29]. The *CarbonEdge* simulator is implemented in Python using ~2k SLOC.

6 EVALUATION

In this section, we evaluate the performance of *CarbonEdge* using real experiments and large-scale simulations. We start with evaluating *CarbonEdge* in mesoscale edge deployments, showcasing its efficiency in reducing carbon emissions on a regional scale. Next, we extend our analysis to continental-scale edge data centers (e.g., a Content Delivery Network, CDN), highlighting that the benefits of saving carbon emissions with granular carbon intensity are commonplace. In doing so, we address the following questions:

(1) *What are the potential carbon savings of spatial shifting for an edge provider with multiple regional edge data center locations?*
(2) *How can a CDN exploit mesoscale variations for carbon-aware edge hosting across a large network of edge locations? How do latency limits affect potential carbon savings?*
(3) *How do seasonal variations, demand, and capacity affect carbon savings and placement decisions?*
(4) *How does the heterogeneity in edge resources impact savings? What are the carbon-energy trade-offs in these settings?*

Next, we detail our real-world datasets, experimental settings, baselines, and evaluation metrics.

6.1 Experimental Methodology

6.1.1 Real World Traces.

This section describes our real-world traces and how we combined them in our evaluations.

Carbon Intensity Traces. We utilize the carbon intensity traces for 2023 from Electricity Maps [26]. The trace contains the hourly carbon intensity, measured in g·CO_2eq/kWh, for 148 carbon zones worldwide, including 54 and 45 in the US and Europe, respectively. Electricity Maps define each zone according to the structure of the regional electricity grid. For example, the results show that the area of carbon zones can be as small as 123.73 km^2 (Tallahassee, Florida).

Latency Traces. To incorporate realistic network latencies between edge data centers. We used the round-trip latency traces from WonderNetwork [32], which provides ping times (in milliseconds) between 246 major cities worldwide. The data covers 64 cities in the US and 64 cities in Europe. Each city is associated with longitude and latitude coordinates. The data highlights that in the US, latency can range from 0.93 ms to 184.67 ms, with an average of 43.17 ms, while in Europe, it ranges from 1.12 ms to 156.74 ms, with an average of 36.94 ms.

Edge Workloads. We utilize two types of compute-intensive edge workloads: a CPU-based edge application that emulates edge sensor data processing and a GPU-based model-serving application that emulates edge AI inference. The CPU-based application is implemented using Python and numpy v1.26, while the model-serving application uses TensorRTv10.2 and CUDA 12.1. Figure 7 depicts the three selected models that cover different tasks and resource requirements: EfficientNetB0 [40], ResNet50 [17], and YOLOv4 [4]. Figure 7a highlights the diversity of our workloads, where energy consumption can reach 45× across models in the same device, and 2× across devices for the same model. Similarly, Figure 7b shows that memory also differs across models and devices. We evaluate the effect of heterogeneity in Section 6.3.5. Unless mentioned otherwise, we assume a round-trip network latency constraint of 20 ms (~500km) in our experiments.

Edge Data Centers. To emulate real-world edge deployment, we utilize Akamai CDN traces, which include the location of edge data centers globally identifiable by their coordinates.

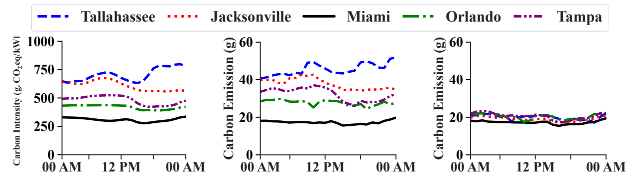
Integrating Traces. Since the availability of granular data differs between traces, we integrate the traces using the following steps:

(1) We map each data center in the Akamai trace to its corresponding carbon intensity zone using its coordinates.
(2) We compute the cross-data center latency by mapping each data center to the nearest city. We assume that users exhibit the same latency as their original edge data center.
(3) We limit our evaluations to Akamai CDN edge data centers where the carbon intensity and latency traces are available.
(4) In the case of multiple data centers in the same city, we combine them into a single data center.

6.1.2 Experimental Setup

We evaluate our *CarbonEdge* under two deployment scenarios:

Mesoscale Regional Edge Deployment. We first evaluate the performance of *CarbonEdge* in mesoscale edge deployments. In our experiments, we use eleven servers to emulate a mesoscale edge network, which comprises five edge data centers distributed across five cities. Each data center is represented by a server and associated with an end device, also represented by a server, for issuing application placement and service requests. *CarbonEdge* operates on a separate server to prevent any interference. The eleven servers are Dell PowerEdge R630, each equipped with a 40-core Xeon E5-2660v3 CPU, 256GB of memory, and a 1Gb/s network connection. Additionally, each edge server contains an NVIDIA A2 GPU that has 1280 CUDA cores, 16GB of memory, and 60W of maximum power consumption, enabling us to evaluate *CarbonEdge* on a GPU cluster. Lastly, we used a workload generator based on Locust[2] and used the Linux traffic control tool (*tc*[3]) to emulate network latency across edge data centers.



(a) Carbon Intensity **(b) Latency-aware** **(c) CarbonEdge**

Figure 8: Carbon intensity and emissions across edge data centers in Florida.

Continental-scale CDN Edge Deployment. In addition to mesoscale evaluations, we show how mesoscale variations can help decarbonize CDN edge deployment that spans an entire continent (e.g., Akamai CDN and AWS Local Zones). In this case, we utilize trace-driven simulations to evaluate the year-long global behavior of the CDN across the US and Europe. In Section 6.3.5, we show the impact of heterogeneity across data centers, where we profiled the ML workloads mentioned above on NVIDIA A2 (1280 CUDA cores, 16GB memory, 60W), NVIDIA Jetson Nano (1024 CUDA cores, 8GB memory, and 15W), and NVIDIA GTX 1080 (2560 CUDA cores, 8GB memory, 180W).

6.1.3 Baselines

We evaluate *CarbonEdge* against multiple baselines.

(1) `Latency-aware`: This policy allocates workloads to the nearest edge data centers to minimize latency overhead, a strategy commonly employed in edge computing [42].
(2) `Energy-aware`: This policy distributes workloads to energy-efficient edge data centers to decrease energy consumption [12, 22]. We implemented this policy by minimizing energy usage while adhering to latency and resource constraints.
(3) `Intensity-aware`: This policy greedily assigns workloads to the greenest edge data centers with the lowest carbon intensity values while respecting the latency and resource constraints.

6.1.4 Evaluation Metrics

We evaluate *CarbonEdge* with three key metrics: Carbon Emissions, Response Time, and Energy Consumption, where we report absolute values as well as carbon savings (%), *round-trip* latency increases (ms), and energy consumption compared to the `Latency-aware` baseline.

[2] Locust: https://locust.io/
[3] https://man7.org/linux/man-pages/man8/tc.8.html

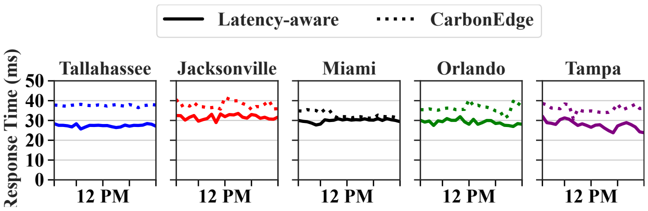
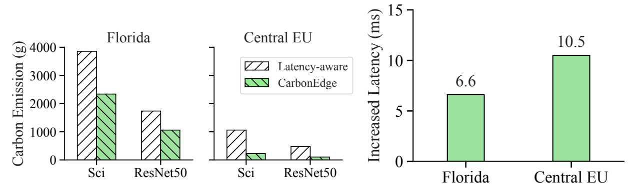
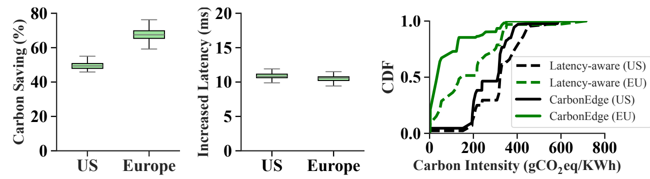


Figure 9: End-to-end response times of applications across edge data centers in Florida.



(a) Carbon Emissions (b) Latency Increases

Figure 10: Performance of *CarbonEdge* across applications, policies, and locations.



(a) Carbon (b) Latency (c) Load Distribution

Figure 11: Carbon savings, latency increases, load distribution in the US and Europe.

6.2 Mesoscale Evaluation

We start by evaluating the performance of the *CarbonEdge* prototype for two regional deployments (in Florida and Central Europe) over a 24-hour period. We compare the performance of *CarbonEdge* to the `Latency-aware` baseline. Figure 8 illustrates the carbon intensity and emissions of the CPU-based application across five zones within the Florida region. As explained in Section 3, zones in Florida exhibit high variations (see Figure 8a). Figure 8b shows the carbon emissions of the `Latency-aware` policy, which highly resemble each zone's carbon intensity in Figure 8a. In contrast, Figure 8c shows how *CarbonEdge* places all applications in the greenest zone (Miami), resulting in an equivalent 20-23 g·CO_2eq of emissions for all applications. Figure 9 lists how the response time changes between `Latency-aware` and *CarbonEdge* across different data centers. As expected, increases in response time are limited due to the proximity of different data centers, where the response time increases are <10.1 ms, with an average increase of 6.61 ms.

Figure 10 depicts the aggregate emissions and latency overheads across regions for the CPU-based and GPU-based applications (ResNet50). As shown in Figure 10a, carbon emissions vary significantly across regions and applications. For instance, in Central Europe, total carbon emissions are reduced by up to 2.6× and 10.3× for the `Latency-aware` and *CarbonEdge* policy compared to those in Florida, where total emission is a function of the average carbon intensity, as highlighted in earlier research [16]. The figure also highlights how power consumption impacts total carbon emissions. For instance, the GPU-based application emits 54.7% less carbon, which is proportional to the differences in power consumption between the CPU-based application and the GPU-based application. However, since the proposed system implements the same placement decisions apart from the application requirements, the carbon savings and latency increases remain consistent. Overall, *CarbonEdge* lowers carbon emissions by 39.4% in Florida and 78.7% in Central EU. Meanwhile, response time increased by 6.6 ms for Florida and 10.5 ms for Central EU (shown in Figure 10b).

Key Takeaways. *In mesoscale edge settings, CarbonEdge can highly optimize the carbon emissions resulting in 39.4% and 78.7% carbon savings for Florida and Central EU, respectively.*

6.3 Mesoscale Evaluation for a CDN

In this section, we evaluate *CarbonEdge* for deploying edge applications in a continental scale CDN using a year-long simulation. We focus our evaluation on US and European CDN edge data centers only since fine-granular carbon intensity information for other continents was unavailable. In a CDN, edge applications arrive at edge data centers over time. Each edge application comes from a specific zone, and its placement is limited to a subset of edge data centers within a certain radius (or latency) of that site.

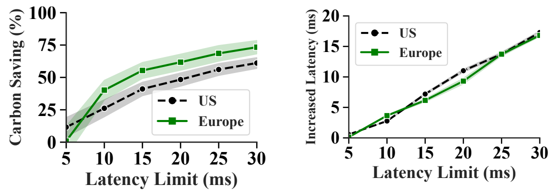
6.3.1 Year-long Performance Evaluation

Figure 11 presents the year-long performance of *CarbonEdge* in terms of carbon savings and latency overheads when considering a latency limit of 20 ms (~500 km). As shown, *CarbonEdge* reduces carbon emissions by 49.5% in the US and 67.8% in Europe. Meanwhile, the average latency increases by 10.8 ms in the US and 10.5 ms in Europe. We note that Europe experiences higher carbon savings as the European data centers reside in greener zones, and the carbon intensity variations across these data centers are larger than those in the US. In addition, Figure 11c illustrates the workload shifting with `Latency-aware` and *CarbonEdge* in the US and Europe. The results highlight that *CarbonEdge* shifts workloads toward low-carbon edge locations. For instance, compared to the `Latency-aware` baselines, *CarbonEdge* increases application execution at 200g·CO_2eq by 40% and 33.9% for the US and Europe, respectively. Moreover, the figure highlights examples where edge data centers do not have any of their load shifted as they are far away from other greener regions. For instance, in the US, the edge data center in Salt Lake City, Utah, does not offload any of its load.

Key Takeaways. *By shifting the demand towards low carbon zones, CarbonEdge decreases carbon emissions by 49.5% and 67.8% for the US and Europe, respectively, while increasing the round-trip latency by less than 11 ms.*

6.3.2 Impact of Latency Tolerance

Carbon savings are a function of placement flexibility, where applications with no latency requirements can be placed in locations with zero carbon intensity [38]. However, in practice, edge applications have tight latency requirements. Figure 12 depicts the carbon savings and latency overheads across different round-trip latency limits in the US and Europe. As shown in Figure 12a, allowing a



(a) Carbon Savings (b) Latency Increases

Figure 12: Effect of latency tolerance on carbon savings and latency increases.

round-trip latency tolerance of 10 ms can yield 28% and 44.8% carbon savings in the US and Europe, respectively, while raising the latency limit to 20 ms can increase these carbon savings by 23% and 23.4%. These increasing carbon savings come from placing more workload in greener regions that meet the latency limits. Moreover, the figure emphasizes that increasing latency limits leads to diminishing returns. For example, in Europe, increasing the latency limit from 5 ms to 10 ms increases savings by 43.8%, whereas increasing the limit from 25 ms to 30 ms only yields an extra 4% savings. Figure 12b shows that performance overheads increase linearly with rising latency limits. Importantly, the results indicate that the benefits consistently outweigh the overheads. For instance, the figure demonstrates that the *CarbonEdge* reduces carbon emissions by 74.7% while incurring only a 17.2 ms increase in round-trip latency.
Key Takeaways. *For a 10 ms increase in latency, CarbonEdge derives 28% and 44.8% carbon savings in the US and Europe, respectively. Notably, the results demonstrate that the benefits constantly outweigh the overheads, where CarbonEdge reduces carbon emissions by 74.7% while incurring only a 17.2 ms increase in round-trip latency.*

6.3.3 Impact of Seasonality

To better understand the effect of seasonality on spatial decisions, Figure 13 illustrates the fluctuations in carbon savings and latency increases over 12 months in the US and Europe. Figure 13a shows that carbon savings in the US exhibit minimal changes, with a maximum difference of 3.3% in carbon savings (i.e., between July and April). In contrast, carbon savings vary significantly in Europe, resulting in a 9.9% difference between February and June. Furthermore, Figure 13b indicates that latency overheads see only slight changes, with only 1.2 ms differences in both the US and Europe.

Figure 13c and Figure 13d further detail how seasonal variations in carbon intensity impact placement decisions in *CarbonEdge*, demonstrating the need for migrating long-running applications across regions. As shown in Figure 13c, the changes in carbon intensity vary significantly between locations. For example, Zagreb, HR exhibits a 102 g·CO_2eq/kWh difference between April and May, whereas Oslo, NO only sees a 2.4 g·CO_2eq/kWh change. Reflecting these changes in carbon intensity, the number of applications assigned to these data centers fluctuates greatly. As indicated in Figure 13d, the number of applications assigned to Paris varies by 1.3× between July and August, while in Oslo, it changes by 3× between December and November. Additionally, the figure indicates that variations in demand might be reflected in the carbon intensity of nearby areas rather than in the region itself. For example, Oslo, which exhibits the most significant fluctuation in application assignments, demonstrates little change in carbon intensity. Conversely, Vienna, with the largest shifts in carbon intensity, sees only minor changes in the volume of assigned applications.
Key Takeaways. *The seasons' changes in carbon intensity highly affect the carbon savings that change by up to 10% across months. The intertwined relations between regions change across seasons, resulting in up to 3× change in resource allocation.*

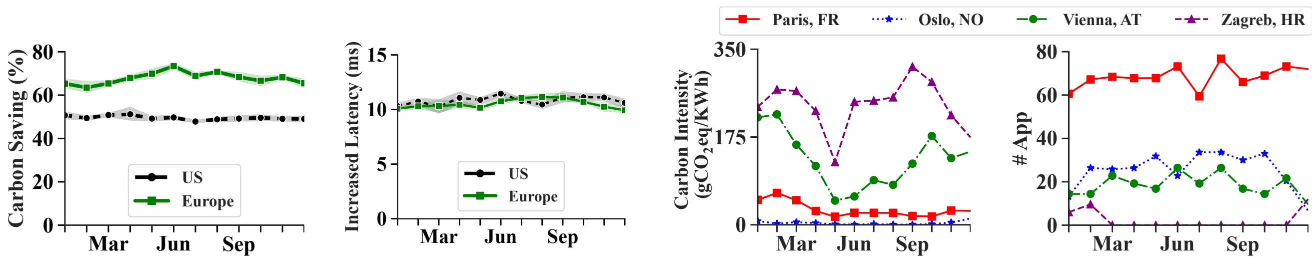
6.3.4 Impact of Demand and Capacity

Carbon savings from edge placements are affected by regional demand and resource capacity. This section evaluates the impact of demand and capacity using population data as a proxy for such differences. Our intuition behind this is that locations with high populations typically have high demand. Similarly, edge providers tend to increase their capacities near them. Figure 14 demonstrates the impact of demand and capacity variations on carbon savings. The demand represents the case where the workload across data centers is proportional to the population across cities, while keeping the capacity fixed, while the capacity scenario changes the capacity distribution according to the population density, while keeping the demand fixed. Lastly, for reference, we add the homogeneous scenario (labeled as Homo), where the demand and capacity are constant across data centers. As shown, in the US, changes in demand and capacity can limit the flexibility to do spatial shifting and decrease carbon savings. For instance, changes in capacity per the population reduce carbon emissions by 6%. This is because high carbon intensity locations (e.g., FL) have no nearby green regions to shit workloads. In contrast, in Europe, the population is more evenly distributed within the data centers we utilize, where carbon savings changes are <1.6% and latency changes by <0.6 ms.
Key Takeaways. *Changes in demand and capacity can impact carbon savings based on the carbon intensity of their origin.*

6.3.5 Impact of Heterogeneity

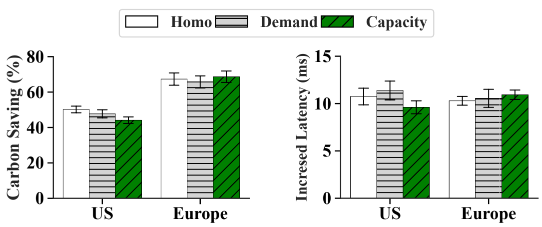
Heterogeneity is an inherent property of edge computing that appears in applications and systems [43]. In this section, we evaluate the performance of *CarbonEdge* with diverse edge applications and heterogeneous edge servers and compare it to three baselines: `Latency-aware`, `Energy-aware`, and `Intensity-aware`.

Figure 15 illustrates the carbon emissions and energy consumption of a mix of applications, including EfficientNetB0, ResNet50, and YOLOv4, across three different resources (Orion Nano, A2, and GTX 1080) and a mix of them (labeled as Hetero.). Figure 15a shows `Energy-aware`, `Intensity-aware`, and *CarbonEdge* can reduce carbon emissions compared to `Latency-aware`. The figure highlights the energy efficiency of different hardware, showing that serving the same load using Orion Nano uses 95.6% less energy than GTX 1080. However, *CarbonEdge* achieves 53% and 62% carbon reductions for Orion Nano and GTX 1080, respectively. This is because, although the GTX 1080 has higher energy consumption, its low inference time (see Figure 7c) can enlarge the potential of spatial shifting, allowing requests to be offloaded to low-carbon locations. Importantly, when considering heterogeneous resources, *CarbonEdge* can further reduce carbon emissions by interplaying the differences in energy efficiency, carbon intensity, and processing speed, decreasing carbon emissions by 98.4%, 79%, and 63% reductions compared to the `Latency-aware`, `Intensity-aware`, and `Energy-aware` baselines, respectively. Figure 15b highlights



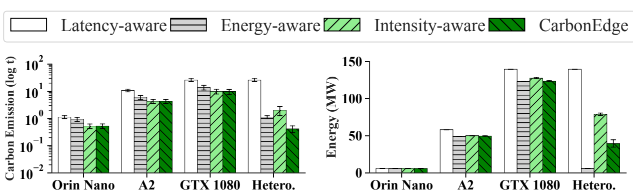
(a) Carbon Savings (b) Latency Increases (c) Carbon Intensity (d) Placements

Figure 13: Effect of seasonality on carbon savings and latency overhead across the US and Europe.



(a) Carbon Savings (b) Increased Latency

Figure 14: Effect of demand and capacity.



(a) Carbon Emissions (log t) (b) Energy Consumption

Figure 15: Carbon emissions and energy consumption across workloads on heterogeneous resources.

the carbon-energy trade-off, where carbon-aware placement increases the total energy consumption. Compared to the energy-aware placement, `Intensity-aware` and *CarbonEdge* can use 12× and 5.5× more energy.

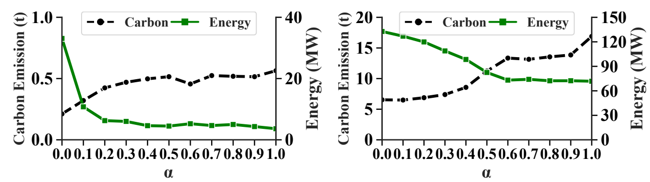
Key Takeaways. *By interplaying the differences in energy efficiency, carbon intensity, and processing speed, CarbonEdge can reduce carbon emissions by 98%, 79%, and 63% compared to the* `Latency-aware`, `Intensity-aware`*, and* `Energy-aware` *baselines, respectively.*

6.4 Navigating the Carbon-Energy Trade-off

Despite the importance of carbon emissions from edge computing, it is crucial not to ignore the trade-off between lowering carbon emissions and energy consumption, as energy typically incurs a monetary cost. To understand the breadth of the carbon-energy trade-off, we augment the optimization objective in Equation 7 as follows:

$$\min_{x^*_{ij}, y^*_j} \quad \alpha \cdot p + (1 - \alpha) \cdot f \tag{8}$$

where p is the total energy consumption, f is the total carbon footprint and α is a weighting factor, where $\alpha = 0$ results in the vanilla *CarbonEdge* policy, while $\alpha = 1$ is the `Energy-aware` policy.



(a) Low Utilization (b) High Utilization

Figure 16: *CarbonEdge* with carbon-energy trade-offs.

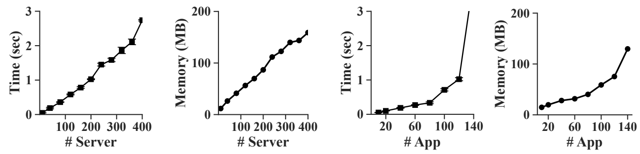
Note that to navigate the trade-off seamlessly, we normalize the carbon intensity and energy consumption between $[0, 1]$ using min-max normalization.

Figure 16 depicts how changing α can affect carbon emissions and energy consumption within two scenarios: low and high resource utilization. As shown, as the utilization increases, the magnitude of carbon emissions and energy highly increases, where carbon emissions and energy consumption increase by 15.5× and 20× from the low to the high utilization scenarios. Moreover, in both cases, *CarbonEdge* significantly reduces carbon emissions, where it reduces carbon emissions by 98.4% and 90.5% for the low and high utilization compared to the `Latency-aware`, respectively. Nonetheless, in both cases, carbon-efficient placement ($\alpha = 0$) increases energy consumption compared to energy-efficient placement ($\alpha = 1$) by 9.1× and 1.84× for the low and high utilization scenarios, respectively. Lastly, the figure highlights that, in both cases, a balance point exists where carbon reductions come at a lower energy cost. For instance, in the low utilization scenario (Figure 16a), using $\alpha = 0.1$ allows *CarbonEdge* to retain 97.5% of its carbon savings while decreasing energy consumption by 67%. In contrast, in the high utilization scenario (Figure 16b), the trade-off is more prominent, where using $\alpha = 0.5$ allows *CarbonEdge* to retain 83.7% of its carbon savings while increasing energy by 15%.

Key Takeaways. *The inherent carbon-energy trade-off is pronounced in heterogeneous edge settings. By augmenting the objective function with energy-awareness, CarbonEdge can maintain 97.5% of its carbon savings while decreasing energy consumption by 67%.*

6.5 System Overhead

We evaluate the runtime performance of *CarbonEdge* in the mesoscale regional edge deployments. When a workload arrives, *CarbonEdge* requires approximately 3.3 *ms* to determine the placement and 1.01 *s* to initiate the application deployment. We further analyze the overhead of our incremental placement algorithm,



(a) Number of Servers **(b) Number of Applications**

Figure 17: Scalability of *CarbonEdge* to input parameters.

which directly affects system performance. Figure 17 shows how the runtime and memory usage of our algorithm scale with two key parameters: the number of servers and the number of applications. By isolating these parameters (varying one while fixing the other), we demonstrate that our incremental placement algorithm scales efficiently to 400 servers and 140 applications, completing computations within 3 seconds while consuming less than 200 MB of memory.

7 DISCUSSION

We have shown the potential of mesoscale carbon intensity information in decarbonizing edge computing. In this section, we reflect on the adoption of *CarbonEdge* for futuristic edge infrastructures and the limitations.

Adopting *CarbonEdge*. Although the carbon intensity in many mesoscale regions is still opaque, we have shown that mesoscale spatial shifting can effectively reduce carbon emissions. In addition, the advances in carbon accounting and the continuing adoption of local renewable energy will enable fine-grained carbon information at a regional and even at the edge data center level. In this case, as highlighted in earlier research [2, 13, 44], carbon-aware spatial shifting becomes more crucial in the decarbonization of computing.

Limitations. Currently, *CarbonEdge* does not automatically redeploy applications to adapt to dynamic workloads and changes in carbon intensity, ensuring low computational overhead and service continuity. Additionally, while carbon-aware spatial shifting has potential and applicability for edge applications, our evaluations were limited to edge data centers with available data from the Akamai CDN traces and the WonderNetwork traces[32].

Holistic Emissions Reduction. In this paper, we only focused on the *operational emissions* of edge workloads from energy consumption, while the *embodied emissions* from manufacturing servers is beyond the scope of this paper[10, 14, 15, 20]. Nonetheless, we note that *CarbonEdge* does not require increases in the number of servers, and earlier research has shown how spatial shifting can help extend the hardware lifespan, amortizing its embodied emissions [24, 39].

8 RELATED WORK

Researchers have analyzed the potential of spatial shifting for batch workloads [9, 23, 38, 44]. For instance, [9] have shown how spatial shifting can reduce the carbon emissions of machine learning training workloads, while in [44], researchers have analyzed how spatial shifting can utilize curtailed energy, which increases the utility of renewable energy. In addition to batch workloads, researchers have shown how interactive workloads can benefit from spatial shifting [7, 11, 13, 21, 25, 27, 37, 38]. For instance, in [37, 38], the authors demonstrated that spatial shifting across geographically dispersed cloud data centers can reduce the carbon emissions of web requests. Moreover, [21] analyzed how spatial shifting can be used for machine learning inference, where the authors showed how combining spatial shifting with model selection can reduce carbon emissions further. In this paper, we underscore the potential of geospatial workload shifting across mesoscale edge data centers, where the benefits of carbon savings outweigh the cost of latency increases. Moreover, we propose a carbon-aware framework for optimizing edge application placements to reduce emissions at the edge. Lastly, many researchers have highlighted many prevalent trade-offs in carbon-aware optimizations to include carbon-performance trade-offs [37, 41], carbon-energy trade-offs [15, 19, 21], carbon-accuracy[21], carbon-cost trade-offs [11, 27]. This paper considers the carbon-performance and carbon-energy trade-offs that are more prevalent in edge computing.

9 CONCLUSION

In this paper, we analyzed fine-grained carbon intensity traces at intermediate "mesoscales," such as within a single U.S. state or neighboring countries in Europe, and showed that intelligently distributing workload at these mesoscales can reduce carbon without violating latency SLOs. To build upon this observation, we presented *CarbonEdge*, a carbon-aware placement framework for edge applications, which optimizes workload placement and power management decisions across edge data centers within a mesoscale region to minimize carbon emissions while satisfying latency SLOs. Our evaluation highlights that our *CarbonEdge* can exploit the mesoscale carbon intensity variations and present carbon savings that outweigh the latency overhead. Current work does not consider the cost of data movement across edge locations. In future work, we will enhance *CarbonEdge* to consider the data movement cost, such as latency and the carbon emissions associated with storage and transfer.

ACKNOWLEDGMENTS

We thank the anonymous HPDC reviewers for their valuable insight and feedback, and Electricity Maps for access to their carbon intensity data. This research is supported by National Science Foundation (NSF) grants 2213636, 2105494, 2211302, 2211888, 2325956, the U.S. Department of Energy Award DE-EE0010143, and support from VMware. This work used Amazon Web Services through the CloudBank, which is supported by NSF grant 19250001. The creation of Sinfonia was supported by the NSF under grant number CNS-2106862.

REFERENCES

[1] 2024. WattTime. https://www.watttime.org/.

[2] 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 *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 118–132.

[3] Noman Bashir, David Irwin, Prashant Shenoy, and Abel Souza. 2022. Sustainable Computing – Without the Hot Air. In *Proceedings of the First Workshop on Sustainable Computer Systems Design and Implementation (HotCarbon)*.

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv:2004.10934* (2020). https://arxiv.org/abs/2004.10934

[5] Ermao Cai, Da-Cheng Juan, Dimitrios Stamoulis, and Diana Marculescu. 2017. Neuralpower: Predict and Deploy Energy-efficient Convolutional Neural Networks. In *Asian Conference on Machine Learning*.

[6] Justine Calma. 2024. Google to power data centers in Europe with offshore wind farms. *The Verge* (2024). https://www.theverge.com/2024/2/1/24056819/google-offshore-wind-farms-data-centers-europe Accessed: 2024-02-01.

[7] 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)*. 18–23. https://doi.org/10.1145/3631295.3631396

[8] Howard David, Eugene Gorbatov, Ulf R Hanebutte, Rahul Khanna, and Christian Le. 2010. RAPL: Memory Power Estimation and Capping. In *ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*.

[9] Jesse Dodge, Taylor Prewitt, Remi Tachet des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A. Smith, Nicole DeCario, and Will Buchanan. 2022. Measuring the Carbon Intensity of AI in Cloud Instances. In *2022 ACM Conference on Fairness, Accountability, and Transparency (FAccT '22)*. 1877–1894.

[10] 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)*. 401–415. https://doi.org/10.1145/3620665.3640415

[11] Peter Xiang Gao, Andrew R. Curtis, Bernard Wong, and Srinivasan Keshav. 2012. It's Not Easy Being Green. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (Helsinki, Finland) *(SIGCOMM '12)*. 211–222. https://doi.org/10.1145/2342356.2342398

[12] Mohammad Goudarzi, Huaming Wu, Marimuthu Palaniswami, and Rajkumar Buyya. 2021. An Application Placement Technique for Concurrent IoT Applications in Edge and Fog Computing Environments. *IEEE Transactions on Mobile Computing* 20, 4 (2021), 1298–1311. https://doi.org/10.1109/TMC.2020.2967041

[13] Viktor Urban Gsteiger, Pin Hong (Daniel) Long, Yiran (Jerry) Sun, Parshan Javanrood, and Mohammad Shahrad. 2024. Caribou: Fine-Grained Geospatial Shifting of Serverless Applications for Sustainability. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles* (Austin, TX, USA) *(SOSP '24)*. 403–420. https://doi.org/10.1145/3694715.3695954

[14] 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)*. 784–799. https://doi.org/10.1145/3470496.3527408

[15] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S. Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. 2022. Chasing Carbon: The Elusive Environmental Footprint of Computing. *IEEE Micro* 42, 4 (jul 2022), 37–47. https://doi.org/10.1109/MM.2022.3163226

[16] Walid A. Hanafy, Qianlin Liang, Noman Bashir, David Irwin, and Prashant Shenoy. 2023. CarbonScaler: Leveraging Cloud Workload Elasticity for Optimizing Carbon-Efficiency. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 7, 3, Article 57 (December 2023), 28 pages. https://doi.org/10.1145/3626788

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

[18] International Energy Agency. 2024. *Electricity 2024*. Technical Report. IEA, Paris. https://www.iea.org/reports/electricity-2024

[19] Yankai Jiang, Rohan Basu Roy, Baolin Li, and Devesh Tiwari. 2024. EcoLife: Carbon-Aware Serverless Function Scheduling for Sustainable Computing. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis* (Atlanta, GA, USA) *(SC '24)*. IEEE Press, Article 12, 15 pages. https://doi.org/10.1109/SC41406.2024.00018

[20] Baolin Li, Rohan Basu Roy, Daniel Wang, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. 2023. Toward Sustainable HPC: Carbon Footprint Estimation and Environmental Implications of HPC Systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Denver, CO, USA) *(SC '23)*. Article 19, 15 pages. https://doi.org/10.1145/3581784.3607035

[21] 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* (Denver, CO, USA) *(SC '23)*. Article 20, 15 pages. https://doi.org/10.1145/3581784.3607034

[22] Yuanzhe Li and Shangguang Wang. 2018. An Energy-Aware Edge Server Placement Algorithm in Mobile Edge Computing. In *2018 IEEE International Conference on Edge Computing (EDGE)*. 66–73. https://doi.org/10.1109/EDGE.2018.00016

[23] Liuzixuan Lin and Andrew A. Chien. 2023. Adapting Datacenter Capacity for Greener Datacenters and Grid. In *Proceedings of the 14th ACM International Conference on Future Energy Systems* (Orlando, FL, USA) *(e-Energy '23)*. 200–213. https://doi.org/10.1145/3575813.3595197

[24] Yejia Liu, Pengfei Li, Daniel Wong, and Shaolei Ren. 2024. Geographical Server Relocation: Opportunities and Challenges. In *Proceedings of HotCarbon'24*. https://hotcarbon.org/assets/2024/pdf/hotcarbon24-final20.pdf Accessed: 2024-10-08.

[25] Diptyaroop Maji, Ben Pfaff, Vipin P. R, Rajagopal Sreenivasan, Victor Firoiu, Sreeram Iyer, Colleen Josephson, Zhelong Pan, and Ramesh K. Sitaraman. 2023. Bringing Carbon Awareness to Multi-cloud Application Delivery. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems, HotCarbon 2023, Boston, MA, USA, 9 July 2023*. 6:1–6:6.

[26] Electricity Maps. 2024. Electricity Maps. https://www.electricitymap.org/map.

[27] Jorge Murillo, Walid A. Hanafy, David Irwin, Ramesh Sitaraman, and Prashant Shenoy. 2024. CDN-Shifter: Leveraging Spatial Workload Shifting to Decarbonize Content Delivery Networks. In *Proceedings of the 2024 ACM Symposium on Cloud Computing* (Redmond, WA, USA) *(SoCC '24)*. Association for Computing Machinery, New York, NY, USA, 505–521. https://doi.org/10.1145/3698038.3698516

[28] NVIDIA. 2023. DCGM Exporter. https://github.com/NVIDIA/dcgm-exporter.

[29] Laurent Perron and Vincent Furnon. 2024. *Google OR-Tools v9.10*. Google. https://developers.google.com/optimization/

[30] Qi, Evan R. Sparks, and Ameet S. Talwalkar. 2017. Paleo: A Performance Model for Deep Neural Networks. In *The International Conference on Learning Representations (ICLR'17)*.

[31] Björn Rabenstein and Julius Volz. 2015. Prometheus: A Next-Generation Monitoring System (Talk). USENIX Association, Dublin.

[32] Paul Reinheimer. 2020. A day in the life of the Internet. https://wonderproxy.com/blog/a-day-in-the-life-of-the-internet/. Accessed: (2023-04-14).

[33] Mahadev Satyanarayanan. 2017. The Emergence of Edge Computing. *Computer* 50, 1 (2017), 30–39. https://doi.org/10.1109/MC.2017.9

[34] Mahadev Satyanarayanan, Paramvir Bahl, Ramon Caceres, and Nigel Davies. 2009. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing* 8, 4 (2009), 14–23. https://doi.org/10.1109/MPRV.2009.82

[35] Mahadev Satyanarayanan, Jan Harkes, Jim Blakley, Marc Meunier, Govindarajan Mohandoss, Kiel Friedt, Arun Thulasi, Pranav Saxena, and Brian Barritt. 2022. Sinfonia: Cross-Tier Orchestration for Edge-Native Applications. *Frontiers in the Internet of Things* 1 (2022), 1025247.

[36] 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 *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 252–265.

[37] Abel Souza, Shruti Jasoria, Basundhara Chakrabarty, Alexander Bridgwater, Axel Lundberg, Filip Skogh, Ahmed Ali-Eldin, David Irwin, and Prashant Shenoy. 2023. CASPER: Carbon-Aware Scheduling and Provisioning for Distributed Web Services. In *Proceedings of the 14th International Green and Sustainable Computing Conference (IGSC), Toronto, ON, Canada*.

[38] Thanathorn Sukprasert, Abel Souza, Noman Bashir, David Irwin, and Prashant Shenoy. 2024. On the Limitations of Carbon-Aware Temporal and Spatial Workload Shifting in the Cloud. In *Proceedings of the Nineteenth European Conference on Computer Systems (EuroSys '24)*. 924–941. https://doi.org/10.1145/3627703.3650079

[39] Jennifer Switzer, Gabriel Marcano, Ryan Kastner, and Pat Pannuto. 2023. Junkyard Computing: Repurposing Discarded Smartphones to Minimize Carbon. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (Vancouver, BC, Canada) *(ASPLOS 2023)*. 400–412. https://doi.org/10.1145/3575693.3575710

[40] Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 6105–6114. https://proceedings.mlr.press/v97/tan19a.html

[41] 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 (Middleware)*. 260–272.

[42] Shanhe Yi, Zijiang Hao, Qingyang Zhang, Quan Zhang, Weisong Shi, and Qun Li. 2017. Lavea: Latency-aware video analytics on edge computing platform. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. 1–13.

[43] Daniel (Yue) Zhang, Tahmid Rashid, Xukun Li, Nathan Vance, and Dong Wang. 2019. HeteroEdge: taming the heterogeneity of edge computing system in social sensing. In *Proceedings of the International Conference on Internet of Things Design and Implementation* (Montreal, Quebec, Canada) *(IoTDI '19)*. 37–48. https://doi.org/10.1145/3302505.3310067

[44] Jiajia Zheng, Andrew A. Chien, and Sangwon Suh. 2020. Mitigating Curtailment and Carbon Emissions through Load Migration between Data Centers. *Joule* 4, 10 (2020), 2208–2222. https://doi.org/10.1016/j.joule.2020.08.001