Untangling the Carbon-Cost Tradeoffs and Stampede Effect Challenges in Cloud Computing

Walid A. Hanafy[a], Thanathorn Sukprasert[a], Abel Souza[b], David Irwin[a], and Prashant Shenoy[a]

[a]University of Massachusetts Amherst
[b]University of California, Santa Cruz

***Abstract*—As society explores new computing applications, data centers have seen a sharp rise in capacity and energy use, driving up data centers' carbon footprint and raising concerns about their sustainability. In response, efforts to reduce emissions now complement the traditional goals of cutting energy costs and boosting performance. Given the spatiotemporal variability of the grid carbon intensity, researchers are increasingly adopting workload shifting strategies to minimize the operational carbon footprint of cloud workloads. However, shifting from cost- and performance-centric operations to carbon-aware operations introduces performance penalties and cost overheads. Moreover, large-scale spatial and temporal shifts can cause stampede effects, with synchronized migration to the same low-carbon regions or times, thereby straining resources and creating imbalances. In this paper, we quantify the carbon-cost-performance trade-offs of carbon-aware scheduling and evaluate the risk of such stampede effects. We provide real-world examples illustrating these trade-offs for application and cloud providers. Lastly, we offer practical insights to help mitigate these challenges and guide sustainable scheduling decisions.**

***Index Terms*—Sustainable Computing, Carbon Efficiency, Cloud Computing, Decarbonization Trade-offs**

I. Introduction

In 2022, global data centers' electricity consumption was estimated to be between 240–340 TWh [1], more than that of Hong Kong or New Zealand in the same year [2], with expectations that it will reach 945 TWh by 2030 [3], more than that of the Netherlands. To reduce computing's energy demand, data centers have been primarily focusing on improving computing's energy efficiency through improved hardware design and energy management [4]. To date, there has been significant progress in improving energy efficiency, as the Green500 list shows a 20× increase in supercomputers' energy efficiency from 2013 to 2023 [5]. Despite notable improvements in energy efficiency, data centers' energy consumption has increased significantly. Such an increase is attributed to slowdowns in energy efficiency gains with little room for further improvements [6], [7], as well as the widespread use of machine learning (ML) and artificial intelligence (AI), such as ChatGPT [8], where data centers are expanding their compute fleets with AI accelerators that typically consume more energy [3].

The rapid increase in data centers' energy consumption raises concerns about their environmental sustainability, especially in electric grids and regions that still utilize significant amounts of non-renewable energy sources [2], resulting in a high operational carbon footprint. As a result, there has been growing interest in reducing data center carbon emissions [9] alongside original resource management and placement approaches that focus on reducing energy demand [10], [11], operational costs [12], [13], and increasing applications' performance. Recent work has explored *supply-side* and *demand-side* approaches to reduce carbon emissions. The supply-side approach involves investing in renewable energy sources such as wind or solar through power purchase agreements (PPAs) to match the consumption with carbon-free energy [14]. However, recent research has shown that supply-side approaches alone are insufficient, as increases in renewable penetration may lead to negligible carbon reductions unless they are paired with demand modulation [15], [16].

In contrast to supply-side adaptations that focus on decarbonization at the electric grid level, demand-side adaptations have also been used to reduce data center carbon emissions. For instance, a line of work has focused on scheduling techniques to shift compute workloads according to the availability of low-carbon energy [9], [17]–[26]. In this case, researchers have utilized the flexibility of batch jobs, which do not have strict deadlines and account for 40-60% of workload demand [27], can be delayed until low-carbon energy is available [9], [19], [21]. Researchers have also explored the scalability of batch workloads to better match the variations in carbon intensity without extending the deadline [9], [17], [18], [28]. Researchers have also used power capping at high carbon periods [23], [29]–[33]. For instance, [32] caps data center power consumption during high carbon periods and shows how different workloads respond to such capping. Another demand side approach is the use of batteries [9], [15], [34]–[36]. For instance, [9] shows how applications can discharge batteries during high-carbon periods and charge them during low-carbon periods. Lastly, a line of work has utilized approximate computing, in which applications are replaced with smaller or lower-energy versions [37]–[39]. For instance, [37] has shown how decreasing the size (and hence the quality) of LLMs at high carbon periods can significantly reduce the carbon emissions. In this paper, we focus on approaches that use carbon-aware scheduling and the trade-offs associated with it.

Although earlier works have been addressing the demand-side adaptations, the focus has been either on the applications' perspective [17]–[19], or the provider's perspective [15], [23], [40]. In the former case, applications interface with the cloud provider on a *pay-as-you-go* basis, considering only their

own performance, cost, and sustainability objectives. More importantly, while providers can estimate demand patterns, they cannot dictate how and when clients can use resources or react to these changes. On the other hand, when considering the provider perspective, researchers often assume that service providers have complete knowledge and full control over the workloads and the underlying resources [15], [23], [26], [40], where the provider schedules jobs in a way that only considers its operational objectives, such as throughput and total costs while considering infrastructure capacity, resource limits.

In this paper, we present an experimental study that redefines how we assess carbon-aware scheduling. Our analysis quantifies inevitable trade-offs that arise when deploying temporal and spatial carbon optimization and proposed policies to navigate these trade-offs. First, we consider the applications' perspective and show how carbon-aware resource management does not come for free and often involves performance and cost overheads. For example, carbon-aware schedulers, such as WaitAwhile [19] or GAIA [21], deliberately increase job waiting times so that the scheduler can search for time periods with low carbon intensity. Therefore, to be carbon-aware, users must tolerate the performance penalty (i.e., additional waiting time) to optimize their carbon emissions. In Section IV, we evaluate these trade-offs using real-world deployments as well as trace-driven simulations to show how state-of-the-art policies from the academic literature behave and to quantify the reasons and the range of such trade-offs.

Second, we examine the cloud provider's perspective, highlighting how large-scale spatial and temporal carbon-aware workload shifting can distort demand patterns and trigger stampede effects. This phenomenon—known as the thundering herd problem—occurs when many clients make similar scheduling decisions, leading to sharp demand spikes during low-carbon periods or in low-carbon regions, causing severe resource pressure and workload imbalances. Hence, our work highlights a growing challenge for cloud providers as applications become increasingly carbon-aware, enabling providers to anticipate and manage these shifts proactively. In Section V, we use real-world traces and policies to quantify the causes and magnitude of demand shifts in both single-cloud and multi-cloud settings. Lastly, in Section VI, we demonstrate how our proposed policies can navigate the application-perspective and cloud-perspective trade-offs.

In this paper, we evaluate the trade-offs and challenges in carbon-aware scheduling for cloud applications and providers. Our paper makes the following contributions:

1) First, we provide a quantitative and qualitative analysis of the trade-offs in carbon-aware scheduling methods for both interactive and batch workloads. We show how carbon-aware resource management is a multi-objective scheduling problem that must consider carbon emissions alongside other performance and cost overheads. For instance, our evaluation shows that achieving 63% carbon reductions in Europe for interactive applications requires extending the network latency by 17ms. Achieving 29.2% carbon reductions for batch jobs scheduled in California requires extending the scheduling deadline by 24 hours.
2) Second, we analyze the changes in demand patterns for cloud providers and the potential stampede effects when applications integrate carbon awareness into their decisions. Our results show that cloud providers may face an order-of-magnitude increase in their peak demand.
3) Lastly, we propose novel scheduling policies to navigate the trade-offs of carbon-aware scheduling for cloud applications and providers, thereby regulating overheads and avoiding stampede effects. Our results demonstrate that if applications forgo $\sim$4% of the carbon savings, they can reduce performance overheads by 50% and cost overheads by 9.2%.

II. Background

In this section, we provide a background on carbon emissions of cloud computing, energy's carbon intensity, and state-of-the-art approaches to optimize carbon emissions of computing.

A. Carbon Emissions of Cloud Computing

The GHG protocol [41] categorizes greenhouse gas (GHG) emissions sources into three scopes: Scope 1, Scope 2, and Scope 3. Scope 1 emissions occur from directly burning fuel and other fossil-based energy sources. Scope 2 emissions are from electricity use, while Scope 3 emissions are from other indirect GHG emissions. Data centers have little to no Scope 1 emissions. The majority of emissions in data centers are operational (Scope 2) and embodied (Scope 3) carbon emissions. The operational emissions of cloud data centers result from their electricity use, and recent studies estimated that the global carbon emissions of data centers from their electricity in 2024 to be 182.2 $Mt{\cdot}CO_2eq$ [3]. In contrast, embodied emissions represent the inherent carbon footprint from manufacturing servers and other components [42], [43]. Although both emissions are critical, they are optimized in distinct ways [42], [44]. In this work, we focus on reducing operational carbon emissions induced by applications' demand modulation, as cloud applications have no visibility or control over the provider's embodied emissions.

B. Cloud Applications Characteristics

Cloud data centers host latency-, data-, and compute-intensive applications that are highly heterogeneous in structure, resource demands, and quality-of-service (QoS) requirements. Workload-characterization studies of public traces from Google, Alibaba, and Microsoft Azure report mixes of long-running services, latency-sensitive web applications, batch analytics jobs, and increasingly AI and serverless workloads, each with distinct performance profiles [27], [45]–[47]. These workloads exhibit diurnal and weekly patterns, which directly affect carbon emissions and, because clusters are provisioned for peak demand, also enable temporal shifting of flexible jobs to lower-carbon or lower-cost periods. At the same time, modern applications are geo-distributed across data centers with different energy prices and sources. Hence, workload placement and regional demand patterns strongly influence

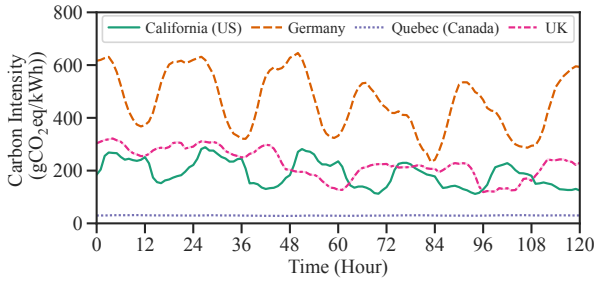


Fig. 1: Electricity grid carbon intensity of different locations from June 15th to 20th of 2022.

energy use, emissions, and cost [10], [11]. Finally, Service Level Agreements (SLAs) constrain how aggressively operators can exploit this temporal and spatial flexibility, effectively determining how much deferral or migration each application can tolerate. Taken together, temporal variation, spatial variation, and SLAs define the design space for carbon-aware cloud scheduling: time-shifting is feasible only when SLAs permit slack or deadline flexibility, while geo-shifting is viable only when end-to-end latency, data regulations, and region-specific SLAs remain satisfied.

C. Energy's Carbon Intensity

To reduce operational emissions, the first step is to track the carbon intensity of the electricity supply, measured in g·CO_2eq/kWh. The carbon intensity of the grid reflects the weighted average of the generation mix, which spans fossil-based sources such as coal or natural gas, renewable sources such as hydro, wind, and solar, and other non-carbon sources such as nuclear. However, since the availability of renewable energy sources varies throughout the day and across locations, the carbon intensity also varies temporally and spatially. For example, solar energy is unavailable at night, and energy generated from a fossil-based power plant will have a higher carbon intensity than energy generated from a renewable source. Previously, the information about the electricity grid was opaque; however, grid balancing authorities have begun releasing the data publicly via web APIs. The visibility into the electricity grid allows commercial services such as Electricity Maps [48] and WattTime [49] to expose the grid's current and forecasted carbon intensity to cloud providers and users.

Figure 1 illustrates the carbon intensity for five days across four carbon regions: California (US), Quebec (Canada), Germany, and the UK. As shown, energy's carbon intensity exhibits both temporal and spatial variations. For example, the carbon intensity in California follows the notable diurnal duck curve [50] due to its high dependency on solar energy. Moreover, the figure highlights the spatial variations between locations where Germany has high carbon intensity, as a large share of its electricity comes from fossil-based energy. At the same time, Quebec shows very low and stable carbon intensity as it is highly dependent on hydroelectric power, a stable source of energy. Next, we highlight how carbon variations motivate researchers to design policies and techniques to minimize computing workloads' carbon footprint.

D. Advances in Carbon-aware Scheduling

Since carbon intensity varies in time and location, researchers proposed various scheduling techniques to select the time and region with low carbon intensity to reduce operational emissions in computing workloads. This section presents an overview of temporal and spatial scheduling and their applications in various scenarios.

Temporal Shifting. The carbon intensity in some regions, including California (Figure 1) and South Australia [20], varies depending on the time of the day. To utilize the temporal variability of carbon intensity, various works have explored deferring the execution of delay-tolerant workloads, e.g., batch jobs, to the time period when low-carbon energy is available [9], [15], [17]–[21], [23], [28], [40], [51]. The essence of the majority of these temporal scheduling techniques is that *the longer the jobs are delayed, the more opportunity to select the time slots with the lowest carbon intensity*. For instance, the WaitAwhile policy [19] considers the job length and deadline and selects the time slots with the lowest carbon intensity. In this case, extending the deadline increases the chance of finding time slots with low carbon periods, resulting in lower carbon emissions.

Spatial Shifting. In addition to temporal variations, different regions exhibit different magnitudes in carbon intensity (see Figure 1). As a result, researchers have been exploring spatial migration and selection algorithms for workloads to run in regions with the lowest carbon footprint [20], [22], [24]–[26], [52]–[54]. Unlike temporal shifting, which the interactive workloads cannot utilize, spatial shifting applies to both batch and interactive workloads. To migrate the workloads spatially, the job schedulers search for the lowest carbon regions under latency constraints. In other words, the larger the allowed latency, the greater the selection of low-carbon destination regions to migrate the workloads. The fundamental concept of spatial workload shifting is that *the larger the search radius and the latency overheads, the higher the carbon savings*. For instance, [22], [54] considers routing requests to regions with the lowest carbon intensity while adhering to user-defined latency constraints.

III. Methodology

In this section, we present our experimental methodology. The goal of this work is to present quantitative and qualitative analysis of the inherent trade-offs of carbon-aware resource management. We start by highlighting the evaluation metrics that provide a holistic assessment of carbon-aware scheduling. We then explain the policies used throughout this paper, as well as our experimental scenarios, to demonstrate the trade-offs and approaches for navigating them. Finally, we list the assumptions we consider in our study.

A. Evaluation Metrics

We use the following metrics to evaluate the performance and overheads of carbon-aware scheduling. We consider:

Carbon Emissions. The total operational GHG emissions from a workload, defined as:

$$\mathbf{C} = \sum_{t \in T} E_t \times I_t \tag{1}$$

where E_t is the application energy consumption at time t, including the computing energy consumption and infrastructure overheads, I_t denotes the carbon intensity of the grid where this workload is running, and T is the execution duration.

Carbon Savings (%). The percentage reduction in carbon emissions, defined as:

$$\mathbf{CS} = \frac{\mathbf{C}_{agn} - \mathbf{C}_{aware}}{\mathbf{C}_{agn}} \times 100 \tag{2}$$

where $\mathbf{C}_{agn}$ is the original carbon emissions of the workload (i.e., carbon agnostic) and $\mathbf{C}_{aware}$ is the carbon emissions of the workload when applying carbon-aware scheduling.

Latency Overhead (ms). The increase in network latency when migrating a workload between data centers, defined as:

$$\mathbf{L} = L_{aware} - L_{agn} \tag{3}$$

where L_{aware} and L_{agn} are the end-to-end latencies of processing requests under carbon-aware and original data center selections.

Delay (hours). The increase in workload completion time that occurs when temporarily shifting the workload to a future low-carbon period. We define delay as:

$$\mathbf{D} = \mathbf{CT}_{aware} - \mathbf{CT}_{agn} \tag{4}$$

where $\mathbf{CT}_{aware}$ and $\mathbf{CT}_{agn}$ are the job completion times when processing workloads in a carbon-aware manner vs. a carbon-agnostic manner.

Application Compute Cost (\$). To analyze the compute cost for cloud applications, we analyze the compute time and, similar to modern billing in cloud approaches, consider a fixed cost model. In this case, given that cloud resources are typically billed on a pay-as-you-go basis. Application Compute Cost is defined as:

$$\mathbf{AC} = \sum_{t \in T} r_t \times p_c \tag{5}$$

where r_t is the amount of resources used at time slot t and p_c is the application resource cost per unit time.

Cost Overhead (%). The percentage overhead in application compute cost, defined as:

$$\mathbf{CO} = \frac{\mathbf{AC}_{aware} - \mathbf{AC}_{agn}}{\mathbf{AC}_{agn}} \times 100 \tag{6}$$

where $\mathbf{AC}_{agn}$ is the original application cost of the workload (i.e., carbon agnostic) and $\mathbf{AC}_{aware}$ is the application cost of the workload under carbon-aware scheduling.

Data Center Peak Demand Increase (×). We represent the stampede effect as the normalized increase in peak demand when considering the normal operations and carbon-aware operations, defined as:

$$\mathbf{PD} = \frac{N^{peak}_{aware}}{N^{peak}_{agn}} \tag{7}$$

where N^{peak}_{aware} and N^{peak}_{agn} are peak resource demand within a data center when considering carbon-aware and carbon agnostic scheduling. Note that our focus is on reporting the increase in peak demand independently of data center capacity, to illustrate how carbon-aware scheduling can influence peak demand patterns. However, we acknowledge that the data center's maximum capacity inherently bounds the actual increase in peak resource demand.

Data Center Energy Cost (\$). Data centers often exhibit two types of energy costs: energy charge and a monthly peak demand charge [55], [56], which can represent up to 40% of the total monthly cost [56]. Thus, in addition to reporting increases in peak demand, we report the total energy cost when considering a fixed power consumption per resource. $\mathbf{EC}(\$) = E_{total} \times p_e + P_{peak} \times p_p$, where E_{total} is the total energy consumption of the data center, including PUE, P_{peak} is the peak power demand, and p_e and p_p are the energy price and peak demand charge.

B. Carbon-Aware Scheduling Policies

To schedule cloud applications in a carbon-aware manner, we employ a wide range of state-of-the-art and proposed policies.

1) ***Scheduling Interactive Workloads****:* To schedule interactive workloads in a carbon-aware manner from the client perspective, we employed a greedy scheduling policy that forwards requests to the greenest location, i.e., the location with the lowest carbon intensity, at each hour, with the granularity of the carbon intensity traces. In this case, we constrain the search space by accounting for latency and regulatory constraints. The latency constraint is applied by setting a maximum performance penalty that the application is willing to accept. On the other hand, the regulatory constraint is enforced by limiting the search space to geographical boundaries, e.g., restricting destinations to regions within Europe.

2) ***Scheduling Batch Workloads****:* To schedule batch workloads in a carbon-aware manner, we consider two scheduling policies.

Suspend-Resume Policy. We first consider WaitAwhile [19], a policy that assumes knowledge of the future carbon intensity and schedules preemptive batch jobs in a suspend-resume (S-R) manner. To minimize the carbon footprint, this policy selects the time slots with the lowest carbon intensity between the submission time and the deadline.

Carbon Savings-per-Delay (CSD) Policy. In addition, we propose a scheduling policy that considers both the carbon savings and corresponding delay (i.e., performance overhead). Specifically, we redefine the scheduling objective as follows:

$$\arg\max_{s} \ \frac{\mathbf{CS}(s)}{\mathbf{D}(s) + \beta} \tag{8}$$

where $\mathbf{CS}(s)$ and $\mathbf{D}(s)$ are the carbon savings and delay for each schedule s, and $\beta \in [0, \infty)$ is the balancing factor that helps in navigating the carbon-delay trade-off. To select

TABLE I: Experimental Scenarios and Policies

Perspective	Workload	Policy	Evaluation Metrics
Application	Interactive	Greedy Selection	$\mathbf{CS}(\%)$, $\mathbf{L}(ms)$
	Batch Jobs	WaitAwhile [19]	$\mathbf{CS}(\%), \mathbf{D}(hour)$
	Elastic Batch Jobs	CarbonScaler [17]	$\mathbf{CS}(\%), \mathbf{CO}(\%)$
Application Navigation	Interactive	Greedy Selection	$\mathbf{CS}(\%)$, $\mathbf{L}(ms)$
	Batch Jobs	WaitAwhile [19] and CSD Policy	$\mathbf{CS}(\%), \mathbf{D}(hour)$
	Elastic Batch Jobs	CarbonScaler [17] and CSCO	$\mathbf{CS}(\%), \mathbf{CO}(\%)$
Provider	Interactive	Greedy Selection	$\mathbf{CS}(\%), \mathbf{PD}(\times)$
	Batch Jobs	WaitAwhile [19]	$\mathbf{CS}(\%), \mathbf{PD}(\times), \mathbf{EC}(\$)$
	Elastic Batch Jobs	CarbonScaler [17]	$\mathbf{CS}(\%), \mathbf{PD}(\times), \mathbf{EC}(\$)$
Provider Navigation	Interactive	Greedy Selection with CI limits	$\mathbf{CS}(\%), \mathbf{PD}(\times)$
	Batch Jobs	WaitAwhile [19] with $\mathcal{L}$ limits	$\mathbf{CS}(\%), \mathbf{PD}(\times)$
	Elastic Batch Jobs	CarbonScaler [17] with $\mathcal{L}$ limits	$\mathbf{CS}(\%), \mathbf{PD}(\times)$

the optimal schedule, our scheduling policy evaluates feasible schedules and selects the one that maximizes the objective value.

*3) **Scheduling Elastic Batch Workloads**:* To schedule elastic batch workloads in a carbon-aware manner, we consider two scheduling policies.

Elastic-Scheduling Policy. To scale jobs in a carbon-aware manner, we employ a recently proposed carbon- and elasticity-aware scheduling policy named CarbonScaler [17]. CarbonScaler is a deadline-aware policy for preemptive jobs that assumes knowledge of future carbon intensity and job scalability profiles. This policy creates a carbon-aware schedule by selecting the optimal execution scale between m and M servers that maximizes throughput per unit carbon, where $m = 0$ denotes suspension and M is a user-defined maximum scale. Moreover, setting $M = 1$ is equivalent to the suspend-resume policy explained above.

Carbon Savings-per-Cost Overhead (CSCO) Policy. To navigate this carbon–cost trade-off introduced by scaling [17], we develop a carbon-cost-aware scheduling policy that selects workload schedules maximizing the ratio of carbon savings to cost overhead, i.e., $\mathbf{CS}(\%)/\mathbf{CO}(\%)$. The policy works by evaluating feasible carbon-aware schedules and selecting the one that yields the highest savings per unit cost overhead, while ensuring that the carbon savings exceed the cost overhead $\mathbf{CS}(\%) > \mathbf{CO}(\%)$.

C. Experimental Scenarios

To demonstrate the trade-offs and overheads of carbon-aware scheduling on cloud applications and providers, we construct a set of scenarios and show the behavior across the policies listed in Section III-B across real-world workload and carbon intensity traces. Table I presents a summary of the scenarios, workload types, and evaluation metrics per scenario. Our experiments utilize the average carbon intensity traces between 2020 and 2022 from Electricity Maps [48]. To generalize our results, we sample various start times and report the average carbon savings and other metrics. We organize the scenarios as follows:

Application Perspective. First, we evaluate trade-offs from the perspective of the cloud application (see Section IV). We simulate two types of workloads: *interactive* (e.g., web services), where clients migrate applications to the greenest regions (i.e., the greedy policy), and *batch* (e.g., ML training), which can be shifted in time or scaled via the WaitAwhile and CarbonScaler policies. In both cases, we demonstrate that carbon-aware scheduling introduces inherent trade-offs among emissions, performance, and cost.

Provider Perspective. Second, we evaluate the implications of carbon-aware scheduling from the cloud providers' perspective (see Section V). Similarly, we consider both *interactive* and *batch* workloads and policies, focusing on the trade-off between carbon emissions and possible stampede effects. This is particularly relevant when many applications implement similar policies, leading to high demand during low-carbon periods and locations, measured by changes in peak demand.

Navigating Trade-offs. Lastly, in Section VI, we highlight how our proposed policies can help applications navigate the complex trade-offs in carbon-aware resource management, balancing emissions, performance, and cost. In addition, we consider how limiting the number of applications (e.g., by workload length $\mathcal{L}$) or imposing limits on carbon intensity can affect the trade-offs from the provider's perspective.

D. Assumptions

Our analysis considers the following assumptions:

1) First, to isolate the benefits of carbon-aware scheduling from differences in data center energy efficiency, we assume that resources are homogeneous. Considering heterogeneous resources and the associated trade-offs is left to future work.
2) Similarly, while cloud resource prices can vary by location and time of use [12], [55], [57], we focus on the cost overheads of using carbon-aware scheduling and assume a uniform cost per resource and per energy unit across all data centers.
3) Lastly, we assume that the processing speed is fixed across data centers. This assumption enables us to focus on the additional network latency introduced by spatial shifting across data centers.

IV. Carbon, Cost, and Performance Tradeoffs for Cloud Applications

Carbon-aware scheduling exploits the inherent spatiotemporal flexibility of computing workloads to match the availability of low-carbon energy across time and location. However, carbon-aware workload scheduling requires modifications to the typical scheduling decisions, which could incur costs and performance penalties. This section presents three practical cases where carbon-aware scheduling introduces additional performance penalties, such as latency, waiting time, and additional costs when scheduling interactive and batch workloads.

A. Interactive workloads

Interactive workloads, such as web and machine learning inference requests, are latency-sensitive workloads. Still, they have the potential to exploit spatial flexibility by getting redirected to regions with lower carbon intensity for servicing. However, such redirections are not feasible for some interactive workloads due to regulatory and latency constraints. For instance, regulatory constraints, privacy laws, and regulations

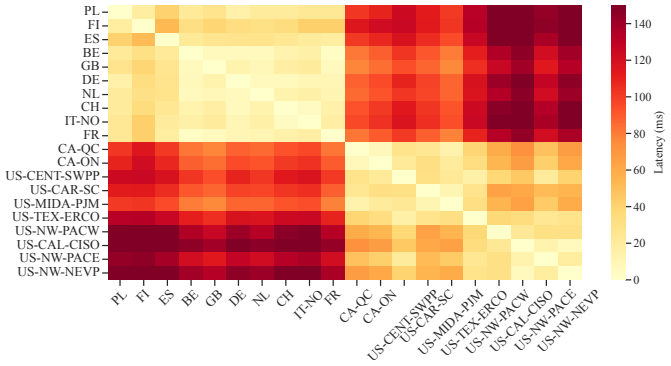


Fig. 2: Latency heatmap across GCP Data Centers, from [58].

may prohibit applications and data from moving outside a specific geographic jurisdiction [59]. Furthermore, some applications, such as AI-driven autonomous vehicles, require immediate, real-time responses and cannot tolerate any degradation in latency. In contrast, some interactive workloads, such as LLM inference services, exhibit tolerance to delays and can accommodate relaxed latency targets, enabling requests to be served in alternative low-carbon regions.

*1) **Experimental Setup**:* Our analysis demonstrates the trade-off introduced by carbon-aware scheduling for interactive workloads. Specifically, we show that for interactive workloads, achieving carbon savings $\mathbf{CS}(\%)$ incurs latency overheads $\mathbf{L}(ms)$. To quantify this trade-off, we conduct real-world experiments on Google Cloud Platform (GCP) and large-scale trace-driven simulation using multiple origin–destination data center pairs across diverse regions, evaluating performance under varying latency and regulatory constraints that limit the spatial flexibility of applications.

Policy. To schedule requests in a carbon-aware manner, we employed the greedy scheduling policy described in Section III-B when considering the hourly varying carbon intensity.

Latency, Workload, and Carbon Traces. We use the latency statistics from the GCP Cloud Platform, which report the round-trip latencies between GCP data centers worldwide. These traces cover 32 GCP locations across North America, Europe, and Asia. Figure 2 reports the latency across data centers in North America and Europe. As shown, the figure highlights two major clusters capable of shifting load across Europe and the US without introducing significant latency overhead that violates QoS.

Lastly, to compute carbon emissions when running the workload across data centers, we map each data center to its corresponding geographic location and compute emissions based on processing time, carbon intensity, and data center- and season-specific PUEs, as reported by Google [60]. To quantify the latency–carbon trade-off under realistic conditions, we built and deployed a geodistributed web service that exposes a matrix-multiplication API across all 10 Google Cloud Platform regions in North America. In each time slot, a Nevada-based client consults our server-selection policy, chooses the region with the lowest carbon intensity subject to latency constraints, and dispatches the request, while logging both the latency and carbon emissions.

Moreover, to generalize our findings, we extend the real-world experiments with trace-driven simulation, using carbon-

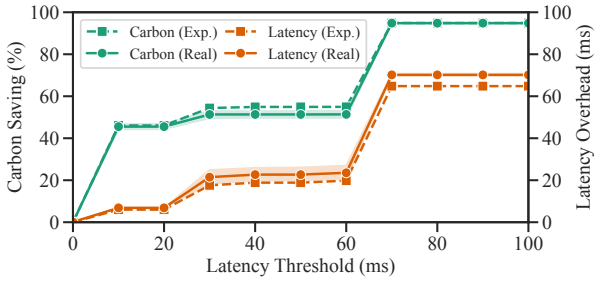


Fig. 3: Real and Expected Carbon savings and Latency overheads across Latency thresholds for a client in Nevada, US.

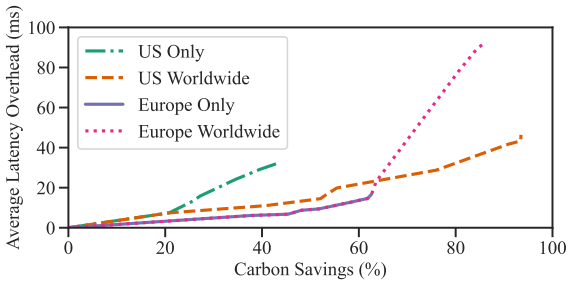


Fig. 4: Carbon savings and Latency Overhead

intensity data from 2020 to 2022. In this case, clients are assumed to be located in the US or Europe and may be served by any of the 32 data centers. Specifically, we consider the clients to be from eight US regions—Iowa, South Carolina, Virginia, Texas, Oregon, California, Utah, and Nevada—and ten European regions—Poland, Finland, Spain, Belgium, the UK, Germany, the Netherlands, Switzerland, Italy, and France. Among these, we select the data center with the greenest energy, subject to latency and geographical constraints on data center migration. We assume each client is geographically close enough to their original GCP region such that the observed latency aligns with the reported GCP latency measurements. Lastly, we create the workload using the Wikipedia request rate traces [61]. For simplicity, we assume a uniform request rate across all regions.

*2) **Results**:* We first evaluate the performance of our real-world deployment, with a client and a server in Nevada.[1] We randomly select 120 time slots between 2020 and 2022 and replay the carbon traces at 30-second intervals. Figure 3 depicts our expected and real carbon savings and latency overheads across different latency thresholds. As shown, increasing the network latency threshold by 10ms can reduce emissions by 45%, as the client can be served by the Google data center in Los Angeles. As the threshold increases, the system opportunistically reallocates requests to regions with lower carbon intensity, further reducing emissions. For instance, at a 70ms threshold, requests are typically shifted to Ontario/Quebec, yielding a 94% reduction in emissions. More importantly, the figure also highlights that the expected and actual values closely track each other across thresholds, indicating that our accounting and end-to-end measurement pipeline is well calibrated, which provides a reliable foundation for the trace-driven results that follow.

[1]Client and Server are deployed in GCP `us-west4` across different zones.

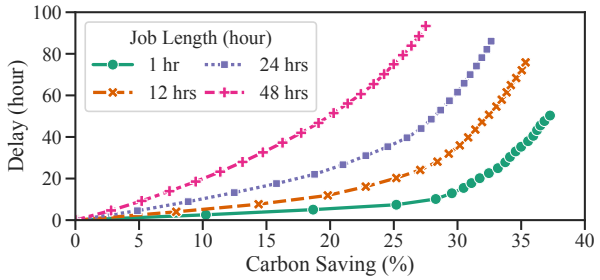


Fig. 5: Carbon savings and delay across waiting times and job lengths.

In generalizing our findings, Figure 4 shows the average carbon savings $\mathbf{CS}(\%)$ and their corresponding latency overheads $\mathbf{L}(ms)$. We analyze the carbon savings for clients from two geographical regions, namely across the US and Europe, with the capability to send requests to any of the 32 GCP locations worldwide. As the results indicate, higher carbon savings are strictly achieved by increases in latency, since savings come by forwarding requests to greener regions, which are often located further away. In addition, we explore the limitations of carbon savings and the latency penalties associated with enforcing geographical (e.g., regulatory) boundaries. The results show that the maximum achievable savings for scheduling requests in the US and Europe are 43% and 63%, respectively. These carbon savings come at the overhead of a 32ms and 17ms increase in round-trip latency. The reason is that within geographical regions, round-trip latency is often low, whereas carbon savings are constrained by the greenest location.

In contrast, when dropping the region constraint, both regions are able to achieve almost 100% carbon-free execution, where all requests get redirected to Quebec and Ontario (Canada), where the electricity grid is mostly carbon-free due to the large penetration of nuclear and hydroelectric energy. However, we note that the added latency is a function of the distance between the source region and the destination, as evidenced by the latency penalty observed from the US or Europe.

Key Takeaways. *For interactive workloads, carbon-aware spatial optimizations incur a latency overhead that is a function of the distance between the source and the permissible greenest destination. For instance, reducing carbon emissions by 63% in Europe comes with an average latency overhead of 17ms.*

B. Batch Jobs

Batch workloads, such as machine learning training and HPC-style jobs, often have temporal flexibility due to their long runtime. Users expect such jobs to wait in a scheduler's job queue until resources are available and run for multiple hours or days. Carbon-aware policies have utilized this temporal flexibility to execute jobs according to the variations in electricity's carbon intensity.

1) ***Experimental Setup****:* To demonstrate trade-offs of carbon-aware scheduling for batch jobs, we construct four synthetic batch jobs of different lengths and simulate their execution in a carbon-aware manner. In particular, we consider jobs with job length from 1 hour to 48 hours and show the possible carbon savings across different waiting times. We extend the waiting time, also known as slack, from 0 to 100 hours, and report the carbon savings $\mathbf{CS}(\%)$ and the corresponding delay $\mathbf{D}(hour)$.

Policy. To schedule jobs in a carbon-aware manner, we employ WaitAwhile [19] as detailed in Section III-B.

2) ***Results****:* Figure 5 displays the carbon savings and delay of temporal shifting when scheduling various jobs, considering the carbon intensity of California, US. The figure shows the average between carbon savings, $\mathbf{CS}(\%)$, and the delay $\mathbf{D}(hour)$, where every point represents the allowed waiting time that the scheduler uses the allowed flexibility to decrease the total carbon emissions. For example, when scheduling a 1-hour job given a 6-hour maximum waiting time, the scheduler selects the best hour to begin within 7 hours. As illustrated in the figure, shorter jobs yield higher savings as they can be freely moved and fit into the lowest time slots, yielding 37% average carbon savings and an average delay of 50.3 hours when allowing jobs to wait for up to 100 hours. In contrast, the longer the job duration, the longer it takes to reach a certain saving percentage. For example, a one-hour job needed an extra 15 hours to gain 25% average carbon savings, while the 12-, 24-, and 48-hour jobs require an additional 25, 45, and 85 hours, respectively, to get the same level of savings. Nonetheless, all job lengths show the same trend of diminishing returns, where each carbon saving percentage comes with an increasing performance penalty, and extending the waiting time unveils slightly better periods. For example, for the 12-hour task, gaining 25% carbon savings comes at a 25-hour extra waiting time. However, improving from the 25% carbon saving levels by 5% and 10% comes with 2× and 4× additional waiting times, respectively. Lastly, we highlight that the relation between carbon savings and penalty depends on the carbon intensity variations within the electricity grid, where higher variations could lead to higher savings in less time, as highlighted in earlier research [17], [20].

Key Takeaways. *Increasing the carbon savings comes at an exponentially growing performance (waiting time) penalty. A 24-hour waiting time extension for batch workloads reduces emissions by up to 29.2%.*

C. Elastic Batch Jobs

Although temporal shifting is a key approach in carbon-aware scheduling, carbon savings come with a considerable performance penalty, as shown earlier. Thus, researchers have explored resource elasticity [62], available in many frameworks, e.g., PyTorch and Spark, to increase carbon efficiency without paying such performance penalties. The key idea of such approaches is to better match valleys of the carbon traces by increasing the demand through scaling [9], [17], [18], [28] while stopping at higher carbon periods. However, scaling batch jobs is subject to reductions in marginal speedup, a direct manifestation of Amdahl's law, e.g., doubling the number of nodes will not double the throughput. Running at a higher scale increases the compute time (run-time × resources) beyond the initial compute time, which increases the operational cost.

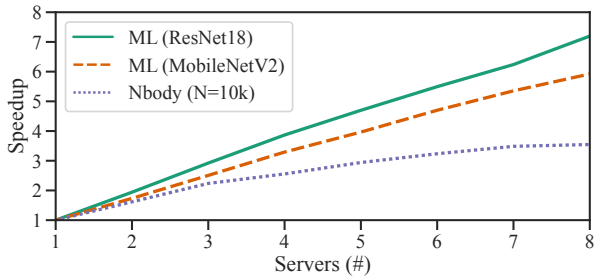


Fig. 6: Application Scalability of machine learning training and HPC-style MPI workloads.

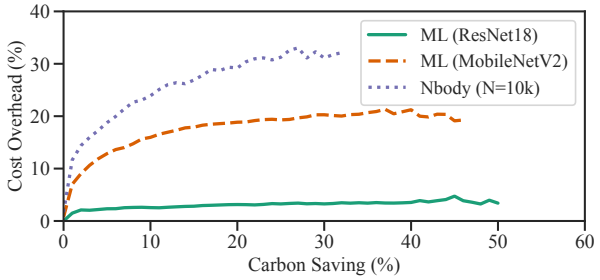


Fig. 7: Carbon savings and Cost overhead achieved by CarbonScaler [17] across different scales when following California, US carbon intensity.

*1) **Experimental Setup**:* To demonstrate the trade-offs between carbon savings and cost increases, we profile real-world elastic applications across different numbers of servers.

Policy. In this section, we schedule elastic jobs using the CarbonScaler policy [17] explained in Section III-B.

Workload. Figure 6 depicts the profiles of two machine learning applications and one MPI application that execute an N-body simulation. The machine learning training applications use PyTorch machine learning framework and are profiled on Amazon Web Services (AWS) using 8 p2.xlarge instances. The MPI application is profiled on a local cluster consisting of 8 servers, each equipped with a 16-core Intel Xeon CPU E5-2620 and connected through a 10Gb network. As shown, applications have different scalability behaviors, which determine the carbon-cost relationship, as we explain next.

*2) **Results**:* Figure 7 shows the average cost overhead $\mathbf{CO}(\%)$ for each achievable carbon saving $\mathbf{CS}(\%)$, when scheduling 24-hour jobs without extending the delay, i.e., slack $= 0$. The results include all carbon savings across various flexibility levels by changing the maximum scale, M, from 2 to 8. As the figure demonstrates, *no carbon savings is achieved for free*. However, the magnitude of carbon savings and cost overhead highly differs between jobs due to their scalability differences. For instance, highly scalable jobs, e.g., ML (ResNet18), can save up to 55% while the average and maximum cost overheads are 3% and 11%, respectively. In contrast, moderately scalable jobs can achieve up to 46% carbon savings at an average cost increase of 15%, while low-scalability jobs achieve only up to 33% carbon savings at a staggering cost increase of up to 36%. Notably, applications with limited or no scalability yield no carbon savings and cost overheads, as the algorithm only scales a job if it results in carbon savings.

Key Takeaways. *Elastic scheduling can achieve high carbon savings without extending the deadline by better exploiting variations in carbon intensity. However, savings come with operational cost increases, depending on the job scalability profile.*

V. Quantifying Stampede Effects for Cloud Providers

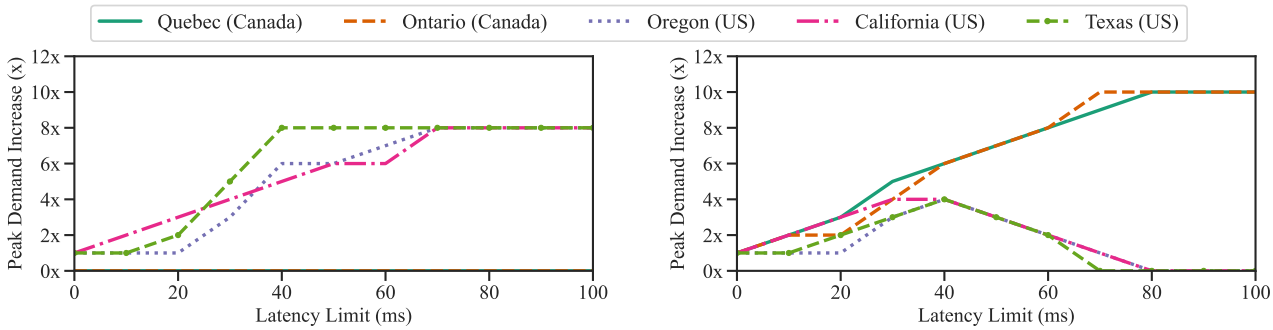
Recent research has focused solely on the perspectives of applications and systems with complete control over the underlying workloads and scheduling decisions. However, in the case of commercial cloud services, applications interact with the cloud provider on a pay-as-you-go basis and only consider their own performance, cost, and sustainability objectives, where the cloud provider has no knowledge or control over how and when the users will run their workloads. Although carbon-aware scheduling, i.e., demand-side adaptation, is essential in carbon optimization, recent research [15], [21] has shown that it increases peak demand above expected levels. When applications choose to be carbon-aware, it becomes very likely that they will execute analogous scheduling decisions, leading to stampede effects or thundering herd problems. Thus, cloud providers must consider the dynamics of carbon emissions and account for anticipated changes in demand patterns primarily driven by the availability of low-carbon energy.

In this section, we experimentally analyze the possible stampede effects in cloud data centers. We demonstrate how applications *synchronized* decisions to schedule workloads in a carbon-aware manner can increase peak demand, which is crucial for determining the operational cost and embodied emissions of the datacenter. For instance, increases in peak demand increase the energy peak charge—a monthly fee determined by the monthly maximum power usage and can be two orders of magnitude higher than regular energy prices [55]. Moreover, it forces data centers to purchase more servers, increasing their embodied emissions or at least relocating them [26], [63]. We evaluate the demand peak using carbon-aware scheduling techniques mentioned in Section IV on latency-sensitive and batch workloads. Finally, we acknowledge that data centers are often over-provisioned and typically operate at 40-60% utilization [27] and demonstrate how utilization affects flexibility and carbon savings.

A. Interactive workloads

As noted in Section IV-A, interactive workloads exploit spatial flexibility by forwarding requests to regions with lower carbon intensity. However, with the expansion of such flexibility and when most cloud users follow a carbon-aware schedule, workloads will be scheduled in concert, impacting providers operating in green regions that would face an unprecedented increase in demand.

*1) **Experimental Setup**:* To demonstrate the relationship between different latency limits, which correlate with carbon savings, and peak demand in the greenest data centers, we conducted an experiment using real-workload traces and distributed requests in a carbon-aware manner, and reported the peak demand across all data centers.



(a) United States (b) North America

Fig. 8: Increase in peak demand with latency limit

Policy. We follow the greedy scheduling policy as explained in Section IV-A. This policy redirects requests to regions with the lowest carbon intensity. The permissible destinations are constrained within the maximum allowed latency as well as the regulatory limits.

Latency, Workload, and Carbon Traces. We leverage the Wikipedia request rate trace [61] and consider 10 clients and data centers in North America, of which 8 are located in the US, and 2 are located in Canada. Similar to Section IV-A, we utilize latency traces of the Google Cloud Platform (GCP) [58] and assume that request rates are fixed per client and that clients experience similar latency as the GCP cross data center latency.

2) ***Results****:* Figure 8 depicts normalized increases in peak demand $\mathbf{PD}(\times)$ across different latency and regional constraints. Figure 8 shows the variation in demand across 3 representative data centers, which exhibit the largest changes, and considers 8 clients and data centers in the US. Note that the carbon savings and actual latency increases for these two cases are the same as those reported in Figure 4 for the US-only and US Worldwide cases, because the carbon intensity in Canada is consistently lower than in European regions. As shown, as customers increase their latency tolerance, the greenest regions receive higher request rates, with peak demand increasing by $8\times$ the original level; i.e., there is a time slot during which all requests go to a single region. Interestingly, multiple regions reach peak demand due to carbon-intensity overlaps or crossings (e.g., across seasons or time of day). A carbon-aware application might switch between the two regions throughout the day, which is feasible mainly when serving stateless applications, e.g., Content Delivery Networks [26].

We then extend our destination regions to include Quebec and Ontario, which have low and stable carbon intensity. As expected, with the increase in latency limits, all customers choose to run their workloads in Quebec and Ontario due to their low intensity. Moreover, the peak demand shows the same trend as in the US case due to the carbon intensity between Quebec and Ontario overlapping over the day. Finally, our experiments assume that the load is homogeneous; however, in practice, data centers receive heterogeneous loads that can amplify the increases in peak demand.

Key Takeaways. *Regions with low carbon intensity will serve as destinations for carbon-aware customers, depending on the overhead they are willing to incur. Cloud providers in North America may face an increase of $3\times$ and $10\times$ in peak demand when customers endure 20ms and 70ms latency overheads.*

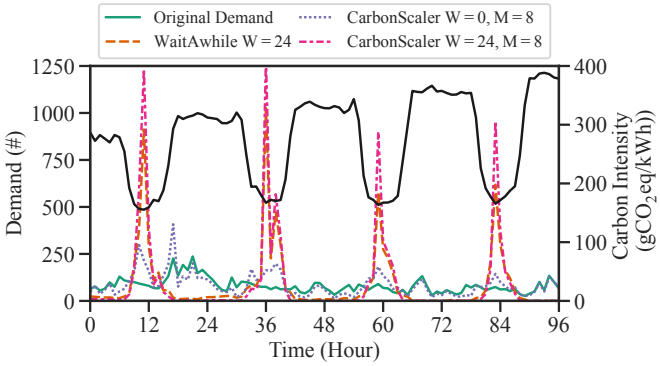


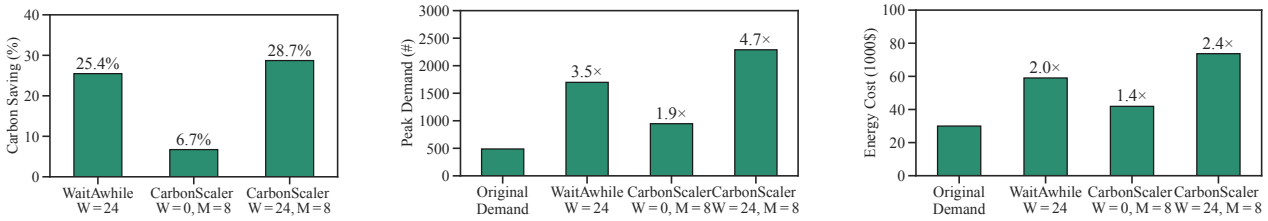
Fig. 9: Carbon intensity for the first three days of March 2020 in California, US, and demand changes across different policies when scheduling the *Alibaba-PAI* trace during the same period.

B. Batch and Elastic Batch Jobs

Carbon-aware scheduling modifies the typical demand patterns as customers try to match the demand with the lowest carbon-intensity periods throughout the day [21]. This behavior shifts the demand from high-carbon periods to low ones, creating high-demand accumulations at low-carbon periods. Nonetheless, recent research has only focused on the attained carbon savings and never reported the increases in peak demand [9], [17], [19], [64]. This section bridges this gap and reports both carbon savings, increase in peak demand, and increases in peak demand cost, when scheduling batch jobs using temporal shifting and scaling.

1) ***Experimental Setup****:* To illustrate the increases in peak demand when scheduling batch jobs, we utilize the Alibaba Platform for Artificial Intelligence, *Alibaba-PAI*, trace [45], and simulate the behavior of a carbon-aware scheduler assuming a homogeneous cluster of various policies and settings, which we detail next.

Policies. We utilize three carbon-aware scheduling policies and settings to schedule the trace in a carbon-aware manner. We use the WaitAwhile [19] and CarbonScaler [17] batch-scheduling policies. We configure the waiting time W for WaitAwhile to be 24 hours $(W = 24)$ and run CarbonScaler



(a) Carbon Savings. (b) Peak Demand (increase × on top). (c) Total Energy Cost (increase × on top).

Fig. 10: Carbon savings ($\mathbf{CS}(\%)$, peak demand increase $\mathbf{PD}(\times)$, and energy cost $\mathbf{EC}(\$)$ across carbon-aware policies when scheduling the *Alibaba-PAI* trace and following California, US carbon intensity.

under two settings. We first consider scaling without waiting by setting a maximum scale factor M to be $(M = 8, W = 0)$. In addition, we consider the case where jobs wait for 24 hours and scale, i.e., $(M = 8, W = 24)$. We evaluate each policy when simulated against the entire trace and report the carbon savings and peak demand.

Workload Traces. We use the Alibaba Platform for Artificial Intelligence, *Alibaba-PAI*, and the trace [45], which contains traces of machine learning training and serving workloads spanning two months. Our analysis includes only training jobs and is performed on a single GPU. This resulted in a trace with $158k$ jobs. Next, we randomly assigned these jobs to an elasticity profile from Figure 6. Finally, to compute the peak demand charge and total energy cost, we consider the energy costs from PG&E [56].

2) Results: Figure 9 depicts the carbon intensity and the demand when applying the carbon-aware policies: WaitAwhile $(W = 24)$, CarbonScaler $(W = 0, M = 8)$, and CarbonScaler $(W = 24, M = 8)$, as well as the original trace over the first three days of March 2020. The results show that all policies create changes in workload demands. However, policies that permit jobs to wait introduce more significant changes. These policies cluster all workloads around the time slots with the lowest carbon intensity. On the other hand, policies that only scale the job and don't extend the waiting time introduce fewer changes. This is because most jobs are short, e.g., 95% of the jobs are less than 6 hours, where scaling introduces less flexibility as it's less likely to find very low carbon slots to scale the job. In contrast, adding a 24-hour waiting time highly expands the scheduling range, allowing all jobs, even the short ones, to be freely moved.

To understand the effects of such demand changes, Figure 10 shows the average carbon savings $\mathbf{CS}(\%)$, peak demand, and energy costs (including energy charge and peak demand charge) when scheduling the entire trace and applying the carbon-aware policies: WaitAwhile $(W = 24)$, CarbonScaler $(W = 0, M = 8)$, and CarbonScaler $(W = 24, M = 8)$. As expected, the better the policy matches the demand with carbon intensity, the higher the carbon saving it gains and the peak demand it exhibits. As shown in Figure 10a, the CarbonScaler $(W = 24, M = 8)$ policy, which has the highest degree of flexibility, can achieve higher savings, reaching 28.7% savings. However, this comes at a 4.7× increase in peak demand; see Figure 10b. Moreover, the figure shows that since most jobs are short, WaitAwhile $(W = 24)$ can achieve compatible results, where it reduces carbon emissions by 25.4% while increasing the peak demand by 3.5×. Lastly, Figure 10c highlights the impact of carbon-aware scheduling on the total energy costs. As shown, total energy costs are much higher for carbon-aware workloads, by up to 2.4× for approaches that use both scaling and temporal shifting, i.e., the CarbonScaler $(W = 24, M = 8)$ policy.

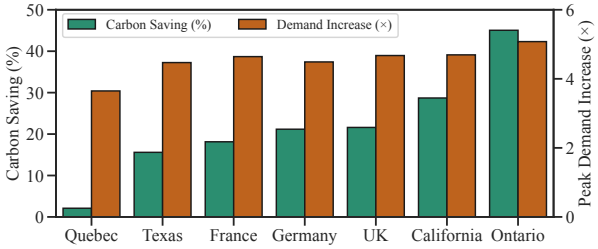


Fig. 11: Carbon Savings $\mathbf{CS}(\%)$ and peak demand increase $\mathbf{PD}(\times)$ when scheduling the *Alibaba-PAI* using scaling and temporal flexibility, i.e., CarbonScaler $(W = 24, M = 8)$, across locations.

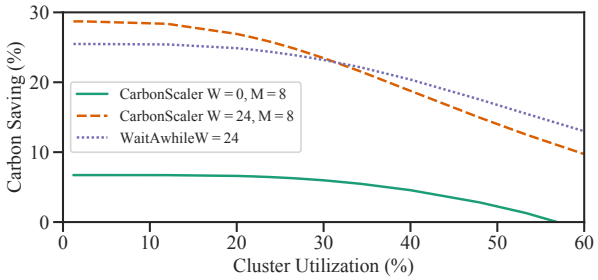
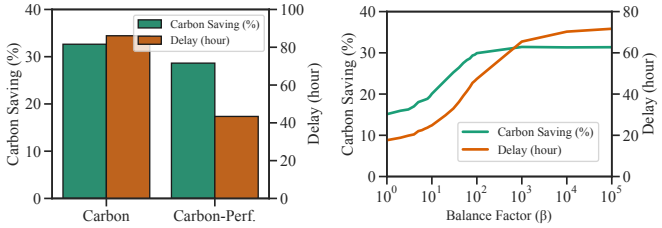


Fig. 12: Effect of utilization carbon savings across policies in California, US.

Figure 11 generalizes our findings when scheduling the *Alibaba-PAI* where jobs wait for 24 hours and scale, i.e., CarbonScaler $(W = 24, M = 8)$, across different locations with various carbon intensity profiles [20]. As shown, the carbon savings vary a lot between locations, reflecting the variability of the carbon trace. For instance, implementing carbon-aware scheduling in Quebec (Canada), known for its stable carbon intensity, leads to low carbon savings of 2%. In contrast, in Ontario (Canada), known for its variable carbon intensity, carbon-aware scheduling can reduce carbon emissions by 45%. Nonetheless, carbon-aware scheduling always increases the peak demand aside from the aggregate savings. For instance, carbon-aware scheduling increases the peak demand in Quebec and Ontario (Canada) by 3.6× and 5×, despite a 43% difference in carbon savings.



(a) $\beta = 70$. (b) Effect of β.

Fig. 13: Navigating the carbon-performance trade-offs of a 24-hour batch job and 100-hour waiting time following California, US carbon intensity.

Lastly, while our earlier experiments considered the demand in a data center with no capacity limit. In practice, however, data centers have a capacity limit and typically have a utilization between 40-60% [27], which highly limits the flexibility. In Figure 12, we evaluate cluster utilization by varying the cluster sizes. We then schedule workloads in a carbon-aware manner, while resolving resource pressures in an FCFS fashion. As shown, when the cluster is highly utilized (i.e., less flexible), carbon savings are lower. Nonetheless, as the cluster size increases and utilization decreases, the policies increase the carbon savings. The figure shows that increasing the cluster size (i.e., lower utilization) yields diminishing returns. Lastly, the figure shows how carbon-aware policies perform under resource pressure: policies that use scaling yield lower savings at high utilization (lower cluster size) and may even increase total emissions. This resulting stampede effect is directly related to how scaling often replenishes the cluster during low-carbon periods, leading to less-efficient executions.

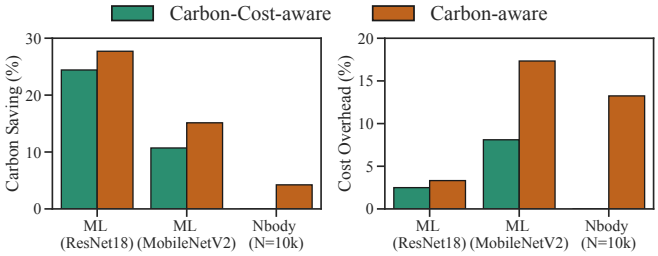
Key Takeaways. *Carbon-aware scheduling of batch jobs introduces unprecedented demand variations and spikes—even when resulting in small carbon savings— resulting in demand peaks that are up to 5× higher than the original demand peaks.*

VI. Navigating The Trade-offs

Although the performance and cost trade-offs are rooted in carbon-aware scheduling, in this section, we discuss multiple approaches that applications and providers can employ to navigate these trade-offs.

A. Navigating Application Trade-offs

Carbon-aware scheduling can implement aggressive scheduling decisions, where carbon savings can come at very high overheads. For instance, in Figure 4, Figure 5, and Figure 7, carbon savings increase at a diminishing rate as performance/cost overheads rise. Thus, the scheduling approach must not surpass the knee of the curve—a cutoff point of diminishing returns—in their scheduling decisions, as it is no longer worth the cost of further savings. In this case, applications should maximize the gains from carbon-aware scheduling by *co-optimizing* both the carbon and the overheads by using multi-objective metrics [21], [65]. Thus, to balance the trade-offs in carbon-aware resource management, we employ the proposed navigation policies presented in Section III-B.



(a) Carbon Savings (%). (b) Cost Overhead (%).

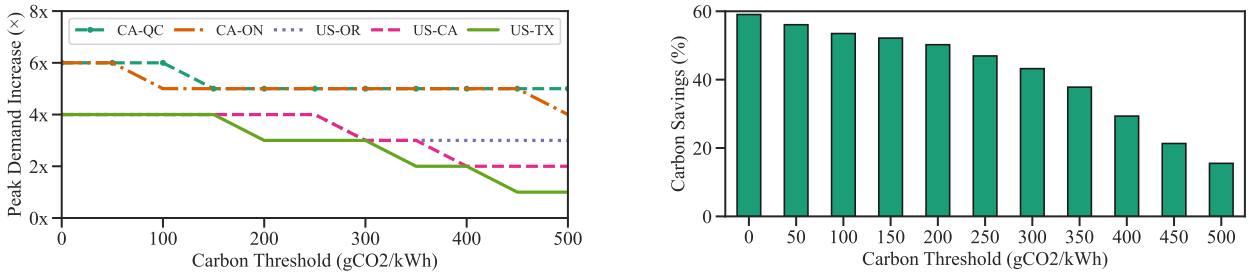
Fig. 14: Navigating the carbon-cost trade-offs for 24-hour elastic batch jobs following California, US carbon intensity.

Batch Workloads. Carbon-aware scheduling of batch workload introduces a carbon-performance trade-off where delaying applications is essential to reduce emissions. To navigate these trade-offs, we consider the proposed Carbon Savings-per-Delay (CSD) policy, which creates a schedule that maximizes the carbon savings per delay in hours.

Figure 13 demonstrates the ability to navigate the carbon-performance trade-off of a 24-hour batch when considering a 100-hour waiting time following the carbon intensity of California, US. As shown, by considering both the carbon and the performance, applications can retain their carbon savings while significantly reducing their performance overheads. Figure 13a compares carbon-aware schedule to carbon-performance-aware schedule when considering $\beta = 70$. As shown, the carbon-performance aware schedule sacrifices 4% of the carbon savings and in return reduces the delay by 50%. Figure 13b shows how users can utilize different values for β to navigate the trade-off. For instance, a low value of β prioritizes delay reduction over carbon savings. In contrast, a large value of β, prioritizes carbon reductions, resulting in higher carbon savings and delay. Nonetheless, the application can select a value of β that yields the highest carbon reductions for the least savings (i.e., at the knee of the curve).

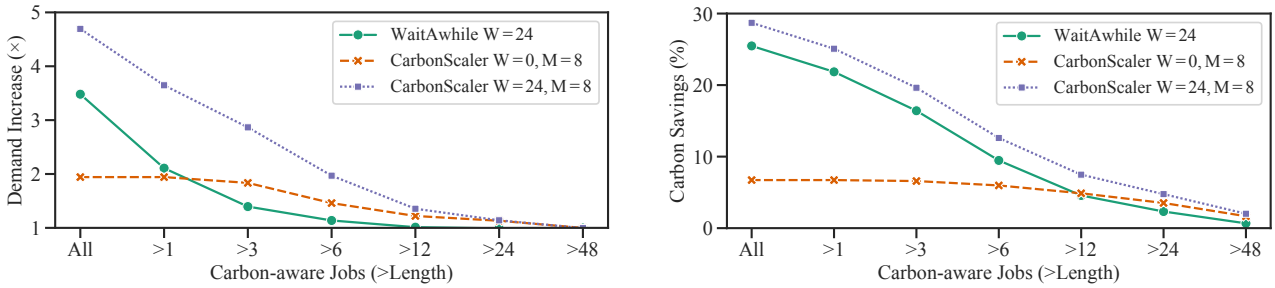
Elastic Batch Workloads. Similarly, carbon-aware scheduling of elastic batch workflows introduces a carbon–cost trade-off. Thus, to navigate this trade-off, we consider the proposed Carbon Savings-per-Cost Overhead (CSCO) metric in Section III-B, which maximizes carbon savings per cost overhead and ensures that carbon savings exceed cost overheads.

Figure 14 uses the same settings as Section IV-C to compare the carbon-aware scheduling policy (CarbonScaler [17]) with our proposed carbon-cost-aware scheduling policy. The figure shows carbon savings $\mathbf{CS}(\%)$ and cost overheads $\mathbf{CO}(\%)$ for both schedules. As shown, incorporating cost considerations preserves most carbon savings while substantially reducing overheads. For example, for ML (MobileV2)—a moderately scalable application—the carbon-cost-aware policy reduces carbon savings by only 4.3% but cuts cost overheads by 9.2%. Moreover, workload scalability influences these trade-offs, resulting in different balancing points. For instance, highly scalable jobs, e.g., ML (ResNet18), already incur low cost overheads, so co-optimization has minimal effect. In contrast, for highly non-scalable jobs, e.g., Nbody (N=10k), where carbon savings typically come with high cost overheads, co-optimization opts not to scale, resulting in no carbon savings.



(a) Demand Increase w.r.t. Carbon-Agnostic. (b) Carbon Savings(%).

Fig. 15: Peak demand increases and Carbon savings (%) across different carbon thresholds when considering a latency limit of 40ms in Figure 8b.



(a) Demand Increase w.r.t. Carbon-Agnostic. (b) Carbon Savings(%).

Fig. 16: Peak demand increases and Carbon savings (%) when different application lengths implement carbon-aware scheduling when scheduling the *Alibaba-PAI* trace and following California, US carbon intensity.

Key Takeaways. *Multi-objective optimization and understanding the trade-offs returns can maximize the gains of carbon-aware scheduling with minimal overheads. Our results demonstrate that by sacrificing ∼4% of the carbon savings, we reduce performance overheads by 50% and cost overheads by 9.2%.*

B. Navigating Provider Trade-offs

Cloud providers face several challenges when optimizing their carbon footprint. For example, there is a trade-off between over-provisioning resources to enhance performance and ensuring resilient execution, which can lead to increased embodied carbon emissions in data centers. Carbon-aware scheduling often exacerbates this trade-off, because it incentivizes all workloads to take similar actions simultaneously and which may worsen peak demand, compelling data centers to overprovision their peak capacity further, resulting in higher embodied carbon emissions.

To address this issue, cloud providers must understand their workload characteristics and how different applications contribute to overall carbon emissions and demand fluctuations. They can then incentivize users to shift resource usage to off-peak periods through mechanisms like spot instances, which offer resources at discounted rates during low utilization periods, which can help smooth demand and reduce sudden usage spikes. In this context, we present two case studies that demonstrate how balancing carbon savings with the associated demand spikes to reduce the stampede effects while still achieving meaningful emission reductions.

Our first case study shows how applications can optimize their selection criteria by including the benefits of carbon-aware load-balancing decisions. In this case, applications migrate only if the difference in carbon intensity between the application's current location and destination region is bigger than the carbon threshold in g·CO_2eq/kWh. Figure 15 shows the effect of increasing the carbon threshold and how it affects the increases in peak demand (see Figure 15a) and total carbon savings (see Figure 15b) in North America. Note that in Figure 15a, we only show regions that exhibit an increase in their peak demand, however, we are considering 10 data centers, as noted in Section V-A. The results indicate that by introducing a carbon threshold of 100 g·CO_2eq/kWh, we can reduce the peak demand for Ontario, Canada, while still achieving 90% of the potential carbon savings.

The second case study utilizes the fact that although short jobs constitute the majority of job traces, they contribute little to the total carbon emissions [21]. Figure 16 illustrates scenarios where carbon-aware scheduling is implemented only for jobs longer than a certain duration. Additionally, we include a case where all jobs implement carbon-aware scheduling, referred to as "All". The figure shows both the increase in peak demand (see Figure 16a), and the carbon savings achieved (see Figure 16b). As expected, Figure 16a shows that short jobs (<1 hour) have minimal effect on the total carbon emissions. However, more importantly, these jobs have a high effect on peak demand (Figure 16(a)), where cloud providers should divert very short jobs from following carbon-aware scheduling. For instance, when considering the WaitAwhile ($W = 24$) policy, a carbon-aware scheduling for only jobs

$>$1 hour decreases the peak demand by 40% while retaining 85% of the carbon savings. Lastly, the figure highlights how different job lengths $\mathcal{L}$ benefit from different policies.

Key Takeaways. *Prioritizing applications with the highest carbon savings helps balance the trade-off between emission reductions and peak demand increases. Our results show that restricting carbon-aware scheduling to jobs exceeding one hour decreases peak demand by 40% while preserving 85% of the potential carbon savings.*

VII. Conclusion

Researchers and business leaders have always acknowledged the cost of sustainability [21], [66], [67], where responding to environmental challenges comes at a cost, which we must pay, but consciously. In this article, we illustrate the inevitable costs and performance overheads of carbon-aware computing in a wide range of scenarios. Our results show that decarbonization costs are context-specific, and greater decarbonization comes with higher costs. Below, we list our key findings that should aid future research in carbon-aware resource management:

1) **No Free Lunch:** *Carbon footprint reductions do not come for free and involve many trade-offs.*
2) **Diminishing Gains:** *Increases in performance and cost overheads come with diminishing gains in carbon savings.*
3) **Carbon-cost trade-offs are context-specific:** *Carbon savings per unit of performance degradation or cost increase* can vary substantially based on application and carbon characteristics.
4) **Stampede effects can be significant:** *Stampede effects from deploying temporal or spatial workload shifting can increase peak demand by an order of magnitude.*
5) **Co-Optimization is Key:** *Co-optimizing the carbon savings and the overheads maximizes the gains from carbon-aware scheduling with minimal overheads.*

Acknowledgment

This research is supported by National Science Foundation (NSF) grants 2213636, 2105494, 2211302, 2211888, 2325956, 23091241, 2105494 and U.S. Department of Energy award DE-EE0010143. This work used CloudBank, which is supported by NSF grant 19250001.

References

[1] International Energy Agency, "Data Centres and Data Transmission Networks," 2023.
[2] H. Ritchie, P. Rosado, and M. Roser, "Energy Production and Consumption," *Our World in Data*, 2020. https://ourworldindata.org/energy-production-consumption.
[3] International Energy Agency, "Energy and AI," 2025.
[4] L. A. Barroso and U. Hölzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. 2009.
[5] Top500.org, "Green500," 2023.
[6] N. Bashir, D. Irwin, P. Shenoy, and A. Souza, "Sustainable Computing – Without the Hot Air," in *HotCarbon'22*, 2022.
[7] D. Irwin *et al.*, "A Vision for Computational Decarbonization of Societal Infrastructure," *IEEE Internet Computing*, pp. 1–7, 2025.
[8] T. Brown *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
[9] A. Souza, N. Bashir, J. Murillo, W. Hanafy, Q. Liang, D. Irwin, and P. Shenoy, "Ecovisor: A Virtual Energy System for Carbon-Efficient Applications," in *ASPLOS'23*, pp. 252–265, March 2023.
[10] R. Mandal *et al.*, "Design and implementation of an SLA and energy-aware VM placement policy in green cloud computing," in *2022 IEEE Globecom Workshops (GC Wkshps)*, pp. 777–782, 2022.
[11] M. Cardosa, A. Singh, H. Pucha, and A. Chandra, "Exploiting Spatio-Temporal Tradeoffs for Energy-Aware MapReduce in the Cloud," *IEEE Transactions on Computers*, vol. 61, no. 12, pp. 1737–1751, 2012.
[12] K. Oh, A. Chandra, and J. Weissman, "A Network Cost-aware Geo-distributed Data Analytics System," in *CCGRID'20*, pp. 649–658, 2020.
[13] M. T. Islam, H. Wu, S. Karunasekera, and R. Buyya, "SLA-Based Scheduling of Spark Jobs in Hybrid Cloud Computing Environments," *IEEE Transactions on Computers*, vol. 71, no. 5, pp. 1117–1132, 2022.
[14] "One small step for RFPs, one giant leap for clean energy," 2023.
[15] B. Acun *et al.*, "Carbon Explorer: A Holistic Framework for Designing Carbon Aware Datacenters," in *ASPLOS'23*, p. 118–132, 2023.
[16] W. J. Cole *et al.*, "Quantifying the challenge of reaching a 100% renewable energy power system for the United States," *Joule*, vol. 5, no. 7, pp. 1732–1748, 2021.
[17] W. A. Hanafy, Q. Liang, N. Bashir, D. Irwin, and P. Shenoy, "Carbonscaler: Leveraging cloud workload elasticity for optimizing carbon-efficiency," in *SIGMETRICS/PERFORMANCE '24*, p. 49–50, 2024.
[18] W. A. Hanafy, R. Bostandoost, N. Bashir, D. Irwin, M. Hajiesmaili, and P. Shenoy, "The War of the Efficiencies: Understanding the Tension between Carbon and Energy Optimization," in *HotCarbon '23*, 2023.
[19] P. Wiesner *et al.*, "Let's Wait Awhile: How Temporal Workload Shifting Can Reduce Carbon Emissions in the Cloud," in *Middleware'21*, p. 260–272, December 2021.
[20] T. Sukprasert, A. Souza, N. Bashir, D. Irwin, and P. Shenoy, "On the Limitations of Carbon-Aware Temporal and Spatial Workload Shifting in the Cloud," in *EuroSys '24*, p. 924–941, 2024.
[21] W. A. Hanafy, Q. Liang, N. Bashir, A. Souza, D. Irwin, and P. Shenoy, "Going Green for Less Green: Optimizing the Cost of Reducing Cloud Carbon Emissions," in *ASPLOS '24*, p. 479–496, 2024.
[22] A. Souza *et al.*, "CASPER: Carbon-Aware Scheduling and Provisioning for Distributed Web Services," in *IGSC'23*, 10 2023.
[23] A. Radovanovic *et al.*, "Carbon-Aware Computing for Datacenters," *IEEE Transactions on Power Systems*, vol. 38, no. 2, pp. 1270–1280, 2022.
[24] M. Chadha, T. Subramanian, E. Arima, M. Gerndt, M. Schulz, and O. Abboud, "GreenCourier: Carbon-Aware Scheduling for Serverless Functions," in *WoSC '23*, p. 18–23, 2023.
[25] B. Li, S. Samsi, V. Gadepally, and D. Tiwari, "Clover: Toward Sustainable AI with Carbon-Aware Machine Learning Inference Service," in *SC '23*, 2023.
[26] J. Murillo, W. A. Hanafy, D. Irwin, R. Sitaraman, and P. Shenoy, "CDN-Shifter: Leveraging Spatial Workload Shifting to Decarbonize Content Delivery Networks," in *SoCC '24*, p. 505–521, 2024.
[27] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes, "Borg: The Next Generation," in *EuroSys'2020*, 2020.
[28] R. Bostandoost *et al.*, "LACS: Learning-Augmented Algorithms for Carbon-Aware Resource Scaling with Uncertain Demand," in *e-Energy '24*, p. 27–45, 2024.
[29] S. Li *et al.*, "Thunderbolt: Throughput-Optimized, Quality-of-Service-Aware Power Capping at Scale," in *OSDI'20*, pp. 1241–1255, 2020.
[30] C. Zhang *et al.*, "Flex: High-Availability Datacenters With Zero Reserved Power," in *ISCA'21*, pp. 319–332, 2021.
[31] M. Savasci, A. Souza, L. Wu, D. Irwin, A. Ali-Eldin, and P. Shenoy, "SLO-Power: SLO and Power-aware Elastic Scaling for Web Services," in *2024 IEEE 24th International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pp. 136–147, 2024.
[32] J. Xing, B. Acun, A. Sundarrajan, D. Brooks, M. Chakkaravarthy, N. Avila, C.-J. Wu, and B. C. Lee, "Carbon responder: Coordinating demand response for the datacenter fleet," 2023.
[33] M. Savasci, A. Souza, D. Irwin, A. Ali-Eldin, and P. Shenoy, "PADS: Power Budgeting with Diagonal Scaling for Performance-Aware Cloud Workloads," in *IGSC'24*, pp. 14–21, 2024.
[34] V. Gupta, P. Shenoy, and R. K. Sitaraman, "Efficient solar provisioning for net-zero Internet-scale distributed networks," in *COMSNETS'18*, pp. 372–379, 2018.
[35] B. Aksanli, E. Pettis, and T. Rosing, "Architecting Efficient Peak Power Shaving Using Batteries in Data Centers," in *MASCOTS 2013*, pp. 242–253, 2013.

[36] R. Urgaonkar, B. Urgaonkar, M. J. Neely, and A. Sivasubramaniam, "Optimal power cost management using stored energy in data centers," in *SIGMETRICS '11*, p. 221–232, 2011.

[37] P. Wiesner, D. Grinwald, P. Weiß, P. Wilhelm, R. Khalili, and O. Kao, "Carbon-Aware Quality Adaptation for Energy-Intensive Services," in *E-Energy '25*, p. 415–422, 2025.

[38] S. Tuli *et al.*, "HUNTER: AI based holistic resource management for sustainable cloud computing," *Journal of Systems and Software*, vol. 184, p. 111124, 2022.

[39] B. O. Turkkan *et al.*, "Greenabr: energy-aware adaptive bitrate streaming with deep reinforcement learning," in *MMSys '22*, p. 150–163, 2022.

[40] L. Lin and A. A. Chien, "Adapting Datacenter Capacity for Greener Datacenters and Grid," in *e-Energy '23*, p. 200–213, 2023.

[41] W. R. Institute, "Green house gas protocol," 2023.

[42] N. Bashir, D. Irwin, and P. Shenoy, "On the Promise and Pitfalls of Optimizing Embodied Carbon," in *Proceedings of the 2nd Workshop on Sustainable Computer Systems (HotCarbon)*, 2023.

[43] U. Gupta *et al.*, "ACT: Designing Sustainable Computer Systems With An Architectural Carbon Modeling Tool," in *ISCA '22*, p. 784–799, 2022.

[44] N. Bashir *et al.*, "The Sunk Carbon Fallacy: Rethinking Carbon Footprint Metrics for Effective Carbon-Aware Scheduling," in *SoCC '24*, p. 542–551, 2024.

[45] Q. Weng, W. Xiao, Y. Yu, W. Wang, C. Wang, J. He, Y. Li, L. Zhang, W. Lin, and Y. Ding, "MLaaS in the wild: Workload analysis and scheduling in large-scale heterogeneous GPU clusters," in *NSDI'22*, pp. 945–960, 2022.

[46] E. Cortez *et al.*, "Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms," in *SOSP '17*, p. 153–167, 2017.

[47] M. Shahrad *et al.*, "Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider," in *ATC' 20*, pp. 205–218, July 2020.

[48] E. Maps, "Electricity Map." https://www.electricitymap.org/map, Accessed September 2022.

[49] WattTime, "WattTime." https://www.watttime.org/, Accessed March 2022.

[50] California Independent System Operator (CAISO), "What the duck curve tells us about managing a green grid?," 2016. https://www.caiso.com/documents/flexibleresourceshelprenewables_fastfacts.pdf.

[51] J. Thiede, N. Bashir, D. Irwin, and P. Shenoy, "Carbon Containers: A System-Level Facility for Managing Application-Level Carbon Emissions," in *SoCC '23*, p. 17–31, 2023.

[52] J. Zheng, A. A. Chien, and S. Suh, "Mitigating Curtailment and Carbon Emissions through Load Migration between Data Centers," *Joule*, vol. 4, no. 10, pp. 2208–2222, 2020.

[53] J. Dodge *et al.*, "Measuring the carbon intensity of ai in cloud instances," in *FAccT '22*, pp. 1877–1894, 2022.

[54] D. Maji *et al.*, "Bringing Carbon Awareness to Multi-cloud Application Delivery," in *HotCarbon'23*, pp. 6:1–6:6, 2023.

[55] Z. Liu, A. Wierman, Y. Chen, B. Razon, and N. Chen, "Data center demand response: Avoiding the coincident peak via workload shifting and local generation," *Performance Evaluation*, vol. 70, no. 10, pp. 770–791, 2013. Proceedings of IFIP Performance 2013 Conference.

[56] S. Govindan, A. Sivasubramaniam, and B. Urgaonkar, "Benefits and limitations of tapping into stored energy for datacenters," in *ISCA '11*, p. 341–352, 2011.

[57] S. Qi, D. Milojicic, C. Bash, and S. Pasricha, "SHIELD: Sustainable Hybrid Evolutionary Learning Framework for Carbon, Wastewater, and Energy-Aware Data Center Management," in *IGSC '23*, p. 56–62, 2024.

[58] AT&T Center for Virtualization at Southern Methodist University, "Google Cloud Inter-Region Latency and Throughput." https://lookerstudio.google.com/u/0/reporting/fc733b10-9744-4a72-a502-92290f608571/page/70YCB, May 27th 2023.

[59] European Parliament and Council of the European Union, "Regulation (EU) 2016/679 of the European Parliament and of the Council." https://data.europa.eu/eli/reg/2016/679/oj, 2016.

[60] Google, "Power usage effectiveness – google data centers." https://datacenters.google/efficiency/, 2025. Accessed: 2025-12-09.

[61] W. Foundation, "Wikimedia's grafana installation," 2023. https://grafana.wikimedia.org/.

[62] W. Fox, D. Ghoshal, A. Souza, G. P. Rodrigo, and L. Ramakrishnan, "E-HPC: A Library for Elastic Resource Management in HPC Environments," in *Proceedings of the 12th Workshop on Workflows in Support of Large-Scale Science*, pp. 1–11, 2017.

[63] Y. Liu, P. Li, D. Wong, and S. Ren, "Geographical Server Relocation: Opportunities and Challenges," in *Proceedings of HotCarbon'24*, 2024. Accessed: 2024-10-08.

[64] C. Zhang and A. A. Chien, "Scheduling Challenges for Variable Capacity Resources," in *Job Scheduling Strategies for Parallel Processing* (D. Klusáček, W. Cirne, and G. P. Rodrigo, eds.), pp. 190–209, 2021.

[65] A. Gandhi, D. Lee, Z. Liu, S. Mu, E. Zadok, K. Ghose, K. Gopalan, Y. D. Liu, S. R. Hussain, and P. Mcdaniel, "Metrics for Sustainability in Data Centers," *SIGENERGY Energy Inform. Rev.*, vol. 3, p. 40–46, Oct. 2023.

[66] R. A. Clarke, J. L. Stavins, Robert N. Greeno, J. L. Bavaria, F. Cairncross, D. C. Esty, B. Smart, J. Piet, R. P. Wells, R. Gray, K. Fischer, and J. Schot, "The Challenge of Going Green," *Harvard Business Review*, 1994.

[67] N. Walley and B. Whitehead, "It's Not Easy Being Green," *Harvard Business Review*, 1994.

Walid Abdelrahman Hanafy is a Postdoctoral Research Associate at the College of Information and Computer Sciences, University of Massachusetts Amherst. He earned his Ph.D. and M.Sc. in Computer Science from UMass Amherst in 2023 and 2025, respectively. Before that, he earned his B.Sc. and M.Sc. in Computer Engineering from Helwan University, Egypt, in 2011 and 2018, respectively. His research lies at the intersection of cloud computing, sustainable computing, and AI systems. Walid's research has been recognized at top venues, including a Best Student Paper award at ACM SIGMETRICS 2024.

Thanathorn Sukprasert is a PhD student in the College of Information and Computer Sciences at the University of Massachusetts, Amherst. She received her B.S. in Computer Engineering from the University of Massachusetts Amherst in 2022. Her research focuses on cloud and edge computing, with an emphasis on power management and sustainability. In particular, her work explores grid-aware power management, demand-response integration, and system-level mechanisms for coordinating computing workloads with power availability to improve efficiency and reliability.

Abel Souza is an Assistant Professor in the Department of Computer Science and Engineering at the University of California, Santa Cruz. He earned his Ph.D. and Licentiate in Computing Science from Umeå University in 2020 and 2018, respectively, and his B.Sc. in Computer Science from Universidade Federal Fluminense in 2014. His research focuses broadly on distributed computing systems, encompassing resource planning and the deployment of optimal strategies for distributed and data-intensive systems, with a recent focus on energy and carbon-aware approaches to designing and managing cloud data centers.

David Irwin is a Professor in the Department of Electrical and Computer Engineering at the University of Massachusetts Amherst. He received his Ph.D. and M.S. in Computer Science from Duke University in 2007 and 2005, respectively, and his B.S. in Computer Science and Mathematics from Vanderbilt University in 2001. His research interests are broadly in experimental computing systems with a particular emphasis on sustainability.

Prashant Shenoy is currently a Distinguished Professor of Computer Science at the University of Massachusetts, Amherst. He received a B.Tech degree in Computer Science and Engineering from the Indian Institute of Technology, Bombay, in 1993, and the M.S. and Ph.D. degrees in Computer Science from the University of Texas, Austin, in 1994 and 1998, respectively. His current research focuses on distributed systems and networking. He is a fellow of the ACM, IEEE, AAIA, and AAAS.