

CARIBOU: Fine-Grained Geospatial Shifting of Serverless Applications for Sustainability

Viktor Gsteiger*, Pin Hong (Daniel) Long
Yiran (Jerry) Sun, Parshan Javanrood
Mohammad Shahrads
University of British Columbia

Abstract

Sustainability in computing is critical as environmental concerns rise. The cloud industry's carbon footprint is significant and rapidly growing. We show that dynamic geospatial shifting of cloud workloads to regions with lower carbon emission energy sources, particularly for more portable cloud workloads such as serverless applications, has a high potential to lower operational carbon emissions. To make the case, we build a comprehensive framework called CARIBOU that offloads serverless workflows across geo-distributed regions. CARIBOU requires no change in the application logic, nor on the provider side. It dynamically determines the best deployment plans, automatically (re-) deploys functions to appropriate regions, and redirects traffic to new endpoints. In reducing operational carbon through fine-grained, function-level offloading, CARIBOU does not undermine standard metrics such as performance and cost. We show how this approach can reduce the carbon footprint by an average of 22.9% to 66.6% across the North American continent. We demonstrate how a detailed specification of location constraints (e.g., to ensure compliance of one stage) can allow emission reductions for workflows (e.g., by offloading other stages). By showcasing the feasibility of carbon-aware geospatial application deployment, CARIBOU aims to push the boundaries of system techniques available to curtail cloud carbon emissions and provide a framework for future research.

CCS Concepts: • Computer systems organization → Cloud computing; • Social and professional topics → Sustainability.

Keywords: Sustainability, Geospatial Shifting, Carbon-Aware Scheduling, Serverless Computing, Cloud Computing

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org. SOSP '24, November 4–6, 2024, Austin, TX, USA

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-1251-7/24/11

<https://doi.org/10.1145/3694715.3695954>

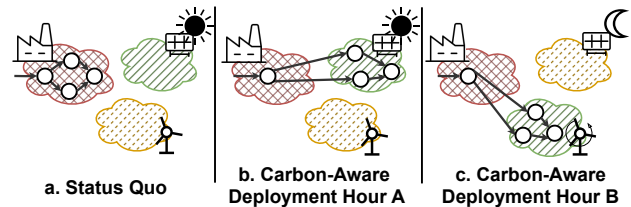


Figure 1. Serverless workflows by default have single-region deployments. This paper explores dynamic carbon-aware deployment, considering latency, cost, and data compliance.

ACM Reference Format:

Viktor Gsteiger, Pin Hong (Daniel) Long, Yiran (Jerry) Sun, Parshan Javanrood, and Mohammad Shahrads. 2024. CARIBOU: Fine-Grained Geospatial Shifting of Serverless Applications for Sustainability. In *ACM SIGOPS 30th Symposium on Operating Systems Principles (SOSP '24)*, November 4–6, 2024, Austin, TX, USA. ACM, New York, NY, USA, 18 pages. <https://doi.org/10.1145/3694715.3695954>

1 Introduction

The urgency of adopting sustainable computing practices has become increasingly evident in light of global energy disparities and the escalating environmental footprint of technological advancements. Estimations from recent studies have put the Information and Communication Technology (ICT) sector's carbon footprint at approximately 1.5% to 4% [1, 16, 23, 74, 76] of worldwide carbon emissions. Additionally, US electricity demand is rising again after a plateau around the mid-2000s, fueled mainly by the changes in computing demand due to more remote work, streaming, and artificial intelligence [31, 47, 73, 78]. Low-carbon energy sources cannot fully cover this increase in power demand [82]. This revelation underscores a critical need to address the carbon emissions associated with cloud applications.

While providers increasingly claim carbon neutrality for their operations, mostly by relying on Renewable Energy Credits (RECs), this still falls short of ensuring the direct use of renewable energy [2, 41, 66], as datacenters are tied to their local power grid. Due to different energy sources, each power grid has different carbon intensities, often fluctuating over daily and seasonal periods. This opens an avenue for

* Affiliated with ETH Zürich, Gsteiger conducted the research at UBC.

optimization by moving workloads to datacenters in low-carbon intensity regions. Providers might reallocate workloads within regions to optimize for carbon efficiency [42, 80]. However, cross-regional shifts, that could yield more substantial carbon savings, are not pursued [22, 107] due to a lack of tools for communicating and techniques for enforcing various constraints such as regulatory or end-to-end latency.

Today’s complex cloud applications often face geographical constraints due to regulatory requirements, such as GDPR in Europe, HIPAA in the US, or PIPEDA in Canada. In such cases, the state of the practice is to deploy the entire application to a "sticky" region, as shown in Fig. 1a. Even without compliance requirements, the latency requirements can limit geographical mobility. Emerging proposals for Sky Computing [19, 29, 106] enable multi-region and multi-cloud deployments, highlighting a shift towards more dynamism, but currently falls short of considering carbon. A carbon-aware framework must contend with the challenge of constantly fluctuating carbon intensity across regions. Migrating workloads from one region to another (Fig. 1b & 1c) to leverage low carbon swings, and doing so by considering latency, cost, transmission carbon, and changing workload profiles adds new dimensions to the problem. Additionally, data locality cannot be ignored since accessing remote data incurs carbon, latency, and potentially cost. In this regime, automation of such distributed deployments becomes a necessity rather than a luxury.

In response to the critical need for sustainable computing practices amidst growing environmental concerns, the complexity inherent in modern cloud applications, and the nascent opportunities for computational geospatial mobility, this paper introduces CARIBOU^{1,2}. CARIBOU is a framework for multi-region serverless application deployment that addresses both the environmental footprint of cloud computing and the regulatory and logistical challenges associated with data mobility across geographical boundaries. In addition to targeting carbon emissions of serverless, a popular cloud paradigm, CARIBOU is positioned as a case study of the potential for serverless to serve as a model for sustainable cloud practices. In doing so, it contributes to the ongoing community dialogue on the feasibility of integrating sustainability into the fabric of cloud computing infrastructures. This paper advances sustainable cloud computing with the following key contributions:

- We introduce a comprehensive framework, called CARIBOU, to help developers automatically deploy complex serverless workflows across globally distributed regions.
- We show how to leverage the grid carbon intensity variations across regions to reduce the carbon footprint of serverless workflows. **This is the first framework to**

holistically consider the carbon effect of data transmission, latency and cost implications of migration, and overhead of the control logic.

- We conduct a comprehensive evaluation of CARIBOU on the leading serverless offering, AWS Lambda. Using a range of benchmark applications with real carbon intensity data, we show that the framework can adapt to changing carbon intensity patterns and harness these differences to reduce the carbon emissions of cloud applications—by an average of 22.9% to 66.6% across the North American continent.

2 Background and Motivation

This section motivates a comprehensive cross-regional deployment framework.

2.1 Carbon Intensity Variances in Electrical Grids

Carbon intensity, measured in grams of CO_2 -*eq*, represents the equivalent carbon emission of different greenhouse gases and their potential for global warming over a common time period [77]. It is used to compare the environmental footprint of different energy production methods. The carbon intensity of different electrical grids can vary significantly due to the energy sources and transmission efficiency of each grid [26, 80]. What does this mean for cloud systems? Let us consider a leading cloud provider, Amazon Web Services (AWS), with six public regions in North America [12]. Fig. 2 illustrates the historical carbon intensity of four of these regions (ca-west-1 rolled out in 2024, us-east-1, and us-east-2 are on the same grid). It can be observed that:

- Grid carbon intensity varies widely due to the local energy sources of each location. ca-central-1 enjoys low intensity since its grid is mostly powered by hydroelectric [25]. us-east-1 and us-east-2 have the highest carbon intensity but are favored due to their proximity to large population centers. Switching from either of those to ca-central-1 or to one of the western regions enables using low-intensity grids. However, the former has data compliance implications (data leaving the US to Canada), and the latter incurs large data transmission overheads.
- Carbon intensity follows a diurnal pattern. This pattern is amplified in us-west-1 with a solar-heavy grid, leading to much greater carbon intensity at night compared to days. This motivates shifting workloads to daytime, or greener regions (e.g., ca-central-1 with consistently low carbon intensity). The possibility of both depends on the nature of the workload and its QoS expectations.
- Carbon footprints and patterns vary across nearby regions. us-west-1 (California) and us-west-2 (Oregon) are geographically close but observe different carbon intensity patterns.

These observations are even more pronounced globally, due to the increased diversity of energy sources, full daily lag for

¹Carbon Aware SeRverless GeospaTial Balancing DeploYment Utility

²Available at <https://github.com/ubc-cirrus-lab/caribou>

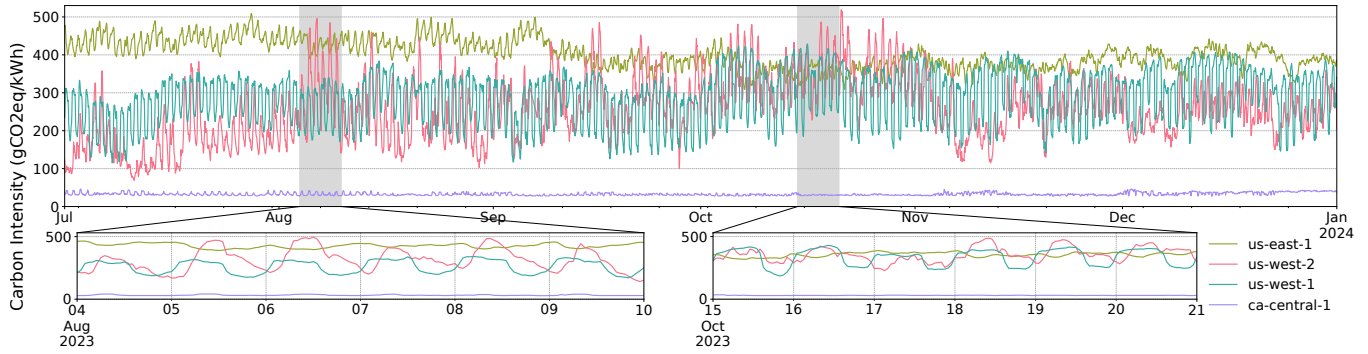


Figure 2. The carbon intensity of four of the six public AWS locations in North America from July 2023 to January 2024. Two week-long windows are chosen to better highlight short-term variability. (source: Electricity Maps [38])

solar generation, and opposite seasons in the northern and southern hemispheres. Similarly, there are many other cloud providers besides AWS operating globally.

2.2 State of the Art

The insights presented in §2.1 allude to optimization opportunities to reduce the carbon footprint of cloud workloads. The most straightforward is making static decisions, such as prioritizing low-carbon regions for building new datacenters or determining regions based on provider-reported carbon efficiency (Salesforce reported prioritizing locations based on Google’s reported carbon efficiency metric [57]). However, such static approaches can only bring about slow positive change and fall short of mitigating the exponential rise in carbon footprint associated with cloud services. Orthogonally, two classes of dynamic resource provisioning techniques offer paradigm change in carbon-aware optimizations in the cloud: **temporal** and **geospatial** workload shifting.

Temporal shifting leverages the variability in carbon intensity or load in a region over time by delaying the execution of latency-tolerant workloads to periods with lower carbon intensity. Recent studies have showcased its high potential as well as inherent constraints for carbon reduction [32, 46, 62, 64, 81, 87, 103]. Conversely, geospatial shifting deploys workloads in locations with lower carbon intensity, offering potential benefits to different classes of applications and broadening the optimization space. While not a new idea [4, 91, 114], geospatial shifting is only gaining serious research traction recently [29, 113] with additional work focused on its limitations and benefits [95, 96]. Unsurprisingly, geospatial shifting involves additional operational challenges around workload and data migration; the cost, performance, and carbon overhead of these can surpass the gains if done naively.

Prior research on geospatial shifting has predominantly viewed the challenge through the lens of cloud providers and workload balancing [3, 4, 26, 60, 61, 99, 111]. Some providers report to migrate internal workloads geospatially to reduce

emissions [43]. This feature is currently not offered to end-users. Migrating client workloads beyond the boundaries of one region remains a complex, underexplored frontier.

2.3 Viability of Geospatial Shifting

The feasibility of geospatial shifting hinges on a myriad of factors, each with its own set of implications.

Compatibility: The varying hardware and software compatibility between cloud providers and between regions of the same provider can preclude the migration of workloads.

Cost: A viable geospatial shifting strategy must take into account the cost implications of migration. There are differences in the cost of computation between providers and even across regions of the same provider [75]. There are also cost implications for data transfer, in the form of egress fees [49, 55] charged even for intra-provider transfers.

Latency: The latency variation introduced by different hardware, software, and loads (relative pressure by co-tenants) in various regions and the transmission latency due to sticky data or traffic source/sink cannot be ignored.

Carbon: A geospatial shifting strategy aware of or optimized for carbon footprint reduction must consider the balance between carbon savings from shifting computing, the carbon cost of data transmission, and any added operational overhead, where the carbon savings from offloading execution might not justify the transmission or overhead carbon costs.

Compliance: It is crucial to consider legal and political boundaries, such as data residency laws. These impose restrictions on certain data/compute movements, necessitating a fine-grained deployment framework capable of navigating these complexities.

Complexity: Modern cloud applications are complex networks of interconnected services and dependencies within and beyond application boundaries. Accounting for this complexity is crucial for a viable shifting solution.

The need for a holistic approach. A viable geospatial shifting strategy must consider these complex and dynamic factors. These factors vary between different applications,

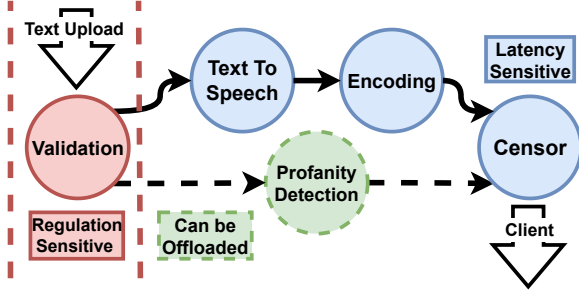


Figure 3. Text2Speech Censoring Serverless Application. Dashed functions and edges are off the critical path and can be offloaded without increasing end-to-end latency.

at different times, and for different users. Therefore, a holistic solution that intricately balances the trade-offs between carbon efficiency and the operational intricacies of complex cloud applications is desired.

2.4 Serverless: A Great Candidate for Geo-Shifting

Given the diverse service models offered by cloud service providers, it is almost impossible to devise a one-size-fits-all framework to universally unlock the potential of geospatial shifting. As *"perfect is the enemy of good"*, we believe the field would benefit from paving the way with a focus on a suitable cloud workload first, to develop necessary techniques and uncover challenges. With this philosophy, this work investigates one of the most suitable cloud service models for geospatial shifting: serverless. Here is why we believe so:

1. Serverless is a popular paradigm, currently used by the majority of organizations using cloud [33]. Serverless applications are more energy-hungry than native execution (more than 15x [90]), ideal for emission reduction.
2. Serverless functions are primarily stateless, and storage is often disaggregated from compute. This allows for fast and reliable re-deployment to new regions.
3. Serverless functions are short-lived and popular applications with high traffic volume [51, 65, 89]. For frequently invoked, lightweight workloads (e.g., serverless), rapid learning from past executions and tolerating suboptimal offloading are far simpler than with infrequent, heavy-weight workloads (e.g., training recommendation models on GPU clusters).
4. The complexity of serverless applications is on the rise. For instance, serverless directed acyclic graph (DAG) adoption in Azure grew 600% between 2019 and 2022 [65]. Increased complexity presents opportunities for exploitation, such as the ability to delay execution of functions not on the critical path of latency [86].

To illustrate the challenges described in §2.3, let us introduce an example workflow that turns text input into speech while censoring profanity (Fig. 3). Each text undergoes validation to ensure it contains no illegal content and is region-restricted to minimize liability. The text is then turned into

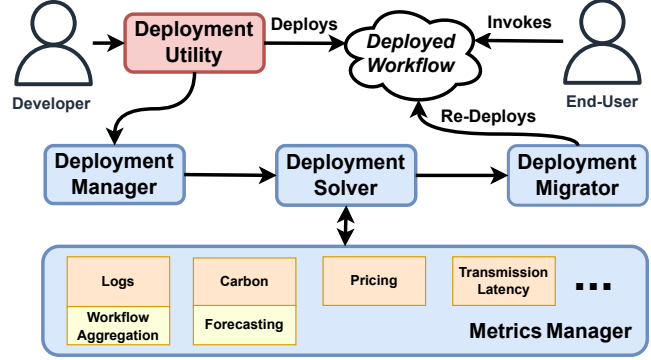


Figure 4. CARIBOU framework overview: Red indicates the user-facing component and blue represents framework components associated with geospatial shifting. Orange shows raw data sources and yellow indicates data-processing.

speech and converted into wav format. The workflow identifies profanities in parallel. The longer path is computationally more expensive and is on the application’s critical path, limiting the ability to offload with stringent QoS constraints due to network latency. A fine-grained offloading framework can (depending on input) assign the profanity identification task to lower carbon regions with no QoS violations.

3 CARIBOU: A Holistic Framework for Geospatial Shifting

To leverage the grid carbon variations across regions while providing a feature-rich framework that enables a large host of future research, we present CARIBOU, a middleware orchestration framework that transparently facilitates managing the relationship between providers, developers, and end-users. This Framework, illustrated in Fig. 4, provides the required infrastructure to answer the following questions for carbon-aware geospatial shifting of serverless workflows:

- **Structure:** What application **structures** (§4) can be defined, and how does this structure relate to the geospatial deployment?
- **Policy: How** (§5.1) and **when** (§5.2) should the framework determine a new geospatial deployment that enables the most, overhead aware, carbon reductions given the workflow constraints?
- **Enforcement:** How to materialize the above policy to geospatially **deploy** (§6.1) and **invoke** (§6.2)?
- **Metrics:** What **metrics** (§7.1) should the policy rely on? What **data sources** (§7.2) can provide those? What steps are required to gather metrics from raw data?
- **Interface:** How to define intuitive and powerful **APIs and interfaces** (§8) for such a framework?

CARIBOU is implemented in Python and Go, with 22K and 4.3K SLOC, respectively. All framework components can be

deployed as individual serverless functions. The components interact asynchronously through a distributed key-value store. The current implementation is fully tested on AWS, the leading serverless provider.

4 Workflow Model

CARIBOU uses a workflow model that encompasses a broad range of serverless applications, from those with a single function to complex ones with many conditionally interconnected functions. In doing so, we consider the structure, data, and control flow of target applications as well as requirements for geospatial shifting. The structure of a workflow is implicitly defined by a developer using our API (§8) and a workflow is then extracted from the source code through static code analysis at initial deployment of a workflow (§6.1).

A workflow is defined as a DAG $G = (N, E)$, where N is a set of nodes and E is a set of edges. The edge $e_{ij} \in E$ represents execution dependencies between nodes $n_i, n_j \in N$ where n_j depends on the execution of n_i . Each source code function (s_k) can be associated with multiple execution stages in the workflow. To make the workflow representation acyclic, each execution stage is represented as a separate node (n_i) in the DAG. Each node (n_i) has an associated deployment region $r_i \in R$, where R is the set of regions. Let $\psi : N \mapsto R$ be the mapping of DAG nodes to regions, referred to as a *deployment plan (DP)* in this paper.

Every node (n_i) has a set of incoming and a set of outgoing edges, $E_{in}(n_i)$ and $E_{out}(n_i)$, respectively. If $E_{in}(n_i) = \emptyset$, the node is a *start node*. We only consider workflows with exactly one start node since this is the most common structure. Additionally, if $|E_{in}(n_i)| > 1$, the node is a *synchronization node*. Any edge e_{ij} is annotated with a Boolean value that captures if the edge is invoked ($C : E \mapsto \{0, 1\}$).

Synchronization nodes: When invoking a synchronization node n_j through e_{ij} , the predecessor invocation of n_i is required to atomically update an annotation associated with e_{ij} . After updating the annotation atomically, n_i has to check if the condition for executing n_j is true, and if so, invoke n_j ; else do nothing. The condition for executing the synchronization node is:

$$(\forall e_{ij} \in E_{in}(n_j), C(e_{ij}) \neq \emptyset) \wedge (\exists e_{kj} \in E_{in}(n_j) : C(e_{kj}) = 1) \quad (4.1)$$

This process is shown in the right section of Fig. 5, where n_1 and n_2 invoke the synchronization node (n_3) through storing intermediate data, small-scale information passed between DAG nodes to communicate data passed through remote storage [65], and annotation in the key-value store. In this example, n_1 is the node checking the condition last and is responsible for invoking n_3 . The synchronization node then loads the intermediate data from the predecessors from the key-value store.

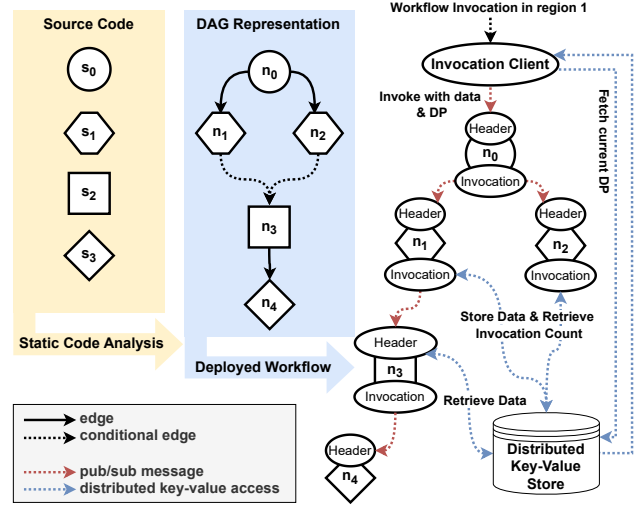


Figure 5. CARIBOU converts the annotated workflow to a DAG and deploys each stage to an optimal region.

Conditional DAGs: The above semantics enable conditional branches, needed to support serverless DAGs with conditional edges [63, 85]. Supporting conditional DAGs adds some complexity. Handling the case where all incoming edges to a node are unconditional, i.e., always taken ($\forall e_{ij} \in E_{in}(n_j), C(e_{ij}) = 1$), is straightforward: n_j is executed when all predecessor nodes are executed. For a conditional edge (e_{ij}), a predecessor node (n_i) marks $C(e_{ij}) = 0$ when the trigger condition of the edge is not satisfied. In this case: (1) for all paths between n_j to any synchronization node n_s , set $C(e_{ts}) = 0$ if n_t is on $Path(n_j, n_s)$ and has edge e_{ts} ; (2) n_i checks if n_s now fulfills the condition to be invoked.

5 Policy

5.1 Determining Optimal Deployment Plan

The fundamental policy decision CARIBOU makes is the deployment plan (DP in §4), i.e., where to deploy each workflow stage. The *Deployment Solver* determines the optimal DP based on the DAG structure and the corresponding performance, carbon, and cost metrics (§7.1). To capture the diurnal carbon patterns, 24 plans are generated per solve—one for each hour, given sufficient carbon budget (§5.2). For a workflow with execution stages N and available regions R , the search space is $|R|^{|N|}$, growing exponentially with increasing workflow complexity and regions. Compliance constraints can narrow this search space, but the complexity explosion persists. Since the goal of CARIBOU is to reduce the carbon footprint of serverless applications, the DP solver must also have low computational requirements. A simple approach to tame the search space is to limit the deployment of all DAG nodes to the same region, reducing the solver complexity to $O(|R|)$. However, this approach can be globally suboptimal

Algorithm 1 HBSS algorithm used for finding optimal DP. α is number of iterations, β is bias, and γ is temperature.

```

1: function HBSS(Deployments, HomeRegionDeployment)
2:    $\alpha \leftarrow |N| \times |R| \times 6$ ,  $\beta \leftarrow 0.2$ 
3:    $\gamma \leftarrow 1.0$ ,  $i \leftarrow 0$ ,  $CD \leftarrow \text{HomeRegionDeployment}$ 
4:   while  $i < \alpha$  do
5:      $ND \leftarrow \text{GENNEWDEPLWBIAS}(CD, \beta)$ ,  $i++$ 
6:     if  $\text{TOLERANCEVIOLATED}(ND)$  then continue
7:     if  $ND.\text{metric} < CD.\text{metric}$  or  $\text{MUT}(\gamma, CD, ND)$  then
8:        $CD \leftarrow ND$ ,  $\gamma \leftarrow \gamma \times 0.99$ ,  $\text{Deployments.add}(ND)$ 
9:       if  $\text{len}(\text{Deployments}) == |R|^{|N|}$  then break
10:  function  $\text{MUT}(\gamma, CD, ND)$  ▷ Stochastic Mutation
11:     $\Delta \leftarrow \forall |CD.\text{metric} - ND.\text{metric}|$ 
12:    return  $\text{Random} < e^{-\frac{\Delta}{\gamma}}$ 

```

for not 1) deploying nodes off the critical path to remote, low-carbon regions and 2) deploying nodes without data compliance requirements to foreign, low-carbon regions.

Implementing the solver using a breadth-first search (BFS) strategy proved intractable and resource-inefficient. We decided on a *Heuristic-biased Stochastic Sampling* (HBSS) algorithm [24, 45], outlined in Alg. 1. It employs heuristics to explore new deployments and tests for improvement, leveraging the information obtained as a region bias. The search terminates either after the complete exploration of the search space or after α iterations, where α depends on DAG size and region count and is dynamically adjusted to fit in AWS Lambda’s 900-second limit when deployed as a serverless function. The algorithm’s hyper-parameters were determined empirically. The developer indicates their preferred optimization priority between carbon, cost, or latency (§8). This is then used to determine the best solutions from the generated deployments.

5.2 Dynamic Triggering of Policy Determination

CARIBOU incurs overheads primarily from DP generation (§5.1) that is computationally heavy, and metrics collection (§7.2) and deployment migration (§6.1) that are data transmission heavy. To offer net gains, the carbon overhead of these factors must remain lower than the carbon savings achieved by geo-shifting the workload. On the one hand, the framework’s deployment region and the frequency of generating DPs influence the overhead carbon emissions. On the other hand, the carbon differential between the home and offload region(s), together with the workflow traffic volume, affects the potential carbon savings. A dynamic control mechanism to trigger the deployment determination is required to balance the two varying effects. To hit a balance between generating new, potentially more optimal DPs and the system overhead, CARIBOU uses a token bucket algorithm to self-regulate DP generation frequency. This frequency is enforced through token check times; when a check is due and

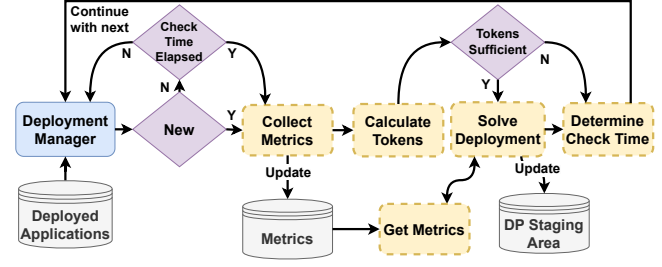


Figure 6. CARIBOU dynamically determines the DP generation trigger frequency with a token bucket algorithm. Blue signifies system components, violet condition checks, and yellow actions taken by the system.

a pre-determined deployment exists that deployment is expired, and all traffic is routed to the home region. DPs expire to account for the dynamic factors influencing optimality, such as fluctuation in carbon intensities, changes in pricing, and varying workflow distributions. Tokens represent the carbon budget for system overhead, and the framework consumes them when the bucket contains enough tokens for determining new deployments. Tokens are earned dynamically based on invocations in past periods and realized workflow carbon savings. The cost of a DP generation is determined based on workflow structure and the framework’s region carbon intensity. The granularity of the generated DP depends on the carbon budget; it can vary from a daily deployment based on daily carbon forecasts to more granular hourly deployments using hourly forecasts.

Fig. 6 illustrates the self-adaptive process, orchestrated by the *Deployment Manager* (DM). The DM regularly iterates over all deployed workflows, going through the following process: If a workflow is new or an existing workflow DP requires a token check, workflow metrics will be collected. Functions with higher invocation counts and longer runtimes accumulate more tokens. Each token represents the carbon intensity differential between target regions, allowing us to estimate the carbon budget. This calculation assumes that the number of invocations and average runtime in the next period will be similar to the last period, without considering any earlier periods, effectively applying a sliding-window to catch sudden changes in metrics. Similarly, the cost of generating a new DP is estimated by the complexity of the application, with more complex DAGs incurring higher generation costs. The current carbon budget is then compared to the DP generation costs. If there are sufficient tokens, a new DP is generated. Regardless of the outcome, the next token check time is determined by the difference between the token generation rate and current bucket content, smoothed by a sigmoid function, to ensure that the next check aligns with the invocation rate of the past period.

6 Enforcement

6.1 Automated Cross-Regional Deployment

CARIBOU automatically manages cross-regional function execution and (re-)deploys the functions to appropriate regions.

Initial Deployment. Developers can initialize the deployment of a workflow using the *Deployment Utility* Command Line Interface (CLI) of the *caribou* Python package. CARIBOU automatically packages the source code into Docker images to facilitate seamless cross-regional deployment, support more complex dependencies, and establish a shared execution environment irrespective of the deployed region (§2.3 Compatibility). This process automatically deploys a workflow for the first time to the developer-defined *home region* that precedes the framework’s dynamic deployments and acts both as a fallback and a baseline. The home region is ideally the default deployment location without our adaptive framework. The deployment utility then works through the following steps to deploy the workflow functions:

1. Through static analysis on the source code directory, it generates the workflow DAG (Fig. 5);
2. The utility creates the necessary Identity and Access Management (IAM) roles and permissions, pushes the Docker image to the container registry, and creates the function and messaging services (e.g., AWS SNS) where each function is subscribed to one topic in its respective region;
3. It uploads metadata, such as DP for this workflow, required by other framework components to the distributed key-value store.

Re-Deployment. Automating migrations to new regions is necessary to maximize carbon emission reductions without burdening developers. The *Deployment Migrator* solves this technical challenge by checking whether the migration of a function to a new region based on the latest DP is required. If a re-deployment is warranted, the migrator replays steps (2) and (3) outlined in **Initial Deployment**, where in step (2), the migrator uses crane [72], a lightweight library for image migration between arbitrary container registries, to copy the image from the home region to a new region. This low-overhead solution is to not have to rebuild function images. A new DP is considered *active* once all functions are redeployed by updating the corresponding value in the distributed key-value store. If any function re-deployment fails, the framework defaults to the home region deployment, such that the framework does not route any invocation through an invalid deployment. This process catches potential issues with deployment, including region unavailability due to increased traffic. The Migrator periodically retries the rollout of any non-activated DP until it is replaced by a new one.

6.2 Flexible, cross-regional workflow execution

CARIBOU must be able to handle and enforce cross-regional workflow execution and traffic routing. No new DP should

necessitate changing the source code. Additionally, the challenge of control flow invocations and synchronization nodes needs to be solved to enable a well-rounded framework. The execution and traffic routing interface must be noninvasive to the developer’s source code, and thus, all the complexity of cross-regional execution is hidden away as part of the function wrapper provided by the *caribou* Python package.

Invoking a workflow. The rightmost section of Fig. 5 outlines the process of workflow invocation. End-user workflow invocations can occur either by sending requests directly to the entry function in the home region, which is then automatically re-routed if required, or through a proxy provided by *caribou*’s CLI utility. In any case, the wrapper routes 10% of the workflow invocations to be fully executed at the home region for performance benchmarking and metric collection. On a workflow invocation, the initial node or the CLI utility fetches the current DP from the distributed key-value store, as indicated in the figure’s blue arrows.

Traffic Routing. The DP determines the current workflow traffic routing by providing the current mapping $\psi : N \mapsto R$. Each node can determine the deployment information of all successors by knowing ψ , the DAG structure, and its location in the DAG. Determining the current DAG location during an invocation is not trivial. The information should be passed on from the predecessor, starting from the known starting point of the initial node; any subsequent node location is determined inductively. When invoking a successor, the decorator provided in the wrapper copies the DP and notes the location of the successor, piggybacking it on the invocation’s intermediate data sent to the successor. The successor handler decorator retrieves the DP, and the data is passed to the defined function, ensuring it receives the data as forwarded by its predecessor.

Pub/sub for invocations. The wrapper invokes function calls by posting a message to the respective function’s publisher/subscriber (pub/sub) messaging topic. By posting the message to the correct region and topic that is part of the deployment information retrieved from the DP, the wrapper routes the invocation to the correct successor deployment without requiring any code changes. Similar to a recent related work [86], we use pub/sub as a geospatial offloading glue due to its availability at all major cloud service providers (e.g., AWS SNS, Azure Service Bus, and Google Pub/Sub) and its ability to support many programming languages, allowing for future portability. Furthermore, they provide a level of reliability by requiring subscriber acknowledgment. If no acknowledgment is received, the pub/sub service automatically retries to deliver the messages. Lastly, pub/sub services seamlessly integrate with serverless functions as invocation triggers, ensuring dependable message delivery. The red arrows in Fig. 5 illustrate this process of message posting and successor invocation.

7 Metrics

7.1 Workflow Metrics and Models

To make policy decisions, CARIBOU requires end-to-end workflow latency, cost, and carbon metrics. The *Metric Manager* (MM) component retrieves/models per-node/-edge DAG metrics and combines them to form workflow-level metrics.

Latency. Transmission latency between regions is captured as a latency distribution for various input sizes, derived from historical data. In the absence of historical data, the MM defaults to using CloudPing [70] to estimate transmission latency. Similarly, the execution time of a workflow stage in a given region is captured as a distribution (as opposed to average) from historical data. When such information is unavailable for a new region, MM defaults to using the home region’s execution time distribution. This distribution exists for any nodes with a non-trivial invocation probability.

Cost. The MM calculates the execution cost based on execution time (t) in seconds, configured memory size (mem) in megabytes, a fixed per-invocation fee [9], along with additional DynamoDB accesses introduced by CARIBOU for geospatial shifting [6]. Moreover, the transmission cost is calculated from the associated outbound data transfer (egress) [7] as well as the costs of SNS messaging used by our framework for function orchestration [8]. We do not consider the implications of the free tier provided by AWS.

Carbon. Carbon footprint has operational and embodied components. The MM only models the operational carbon. Considering embodied carbon is essential for carbon accounting [59], but including it in scheduling decisions is not always accurate. In our setting, as long as capacity is available to host a function execution, embodied carbon for both the current and future host nodes will be incurred regardless of CARIBOU’s offloading decision. In economic terms, this is referred to as sunk cost. And if there is no capacity to offload, there won’t be any offloading. Furthermore, without extensive publicly available data (including for building construction, land, etc.) to reliably model the embodied carbon for each region, the most meaningful approach would be to associate the same embedded carbon per unit of resource to all regions. Adding the resulting equal embodied carbon baseline to all regions does not affect their relative carbon differential, the element leveraged by CARIBOU.

The operational carbon footprint of execution ($Carbon_{ex}$) can be estimated by the power consumed adjusted by power usage effectiveness (PUE) and the carbon intensity of grid (I_{grid}) [34, 56, 98, 114]. I_{grid} can either be the average carbon intensity (ACI) or marginal carbon intensity (MCI). Prior work has used both ACI [46, 96, 103] and MCI [26, 67] for scheduling-related tasks and analysis. There is growing interest in using MCI for carbon-aware optimization, but it can lead to different decisions [94], highlighting the importance of continued research. We opted to use ACI due to the high uncertainty of the MCI signals [103] and the difficulty of

verifying MCI as opposed to ACI, which can be measured from electricity production data [37].

During the execution, CPU and memory are the predominant carbon contributors, as modeled in previous research [56, 93]. The contribution of other components is shown to be relatively negligible [56]. We use a PUE of 1.11, the average of the 1.07-1.15 range reported for AWS datacenters [5].

$$Carbon_{ex} = I_{grid} \times (E_{proc} + E_{mem}) \times PUE \quad (7.1)$$

We use the 3.725e-4 kW/GB, estimated and used by the community [53, 56], as a basis for energy usage associated with the serverless function’s memory usage (P_{mem}):

$$E_{mem} = P_{mem} \times (mem/1024) \times t/3600 \quad (7.2)$$

The number of vCPUs of an AWS Lambda function is based on its memory size ($n_{vcpu} = mem/1,769$) [11]. CARIBOU uses AWS Lambda Insights [13] to collect cpu_total_time , needed to calculate average vCPU utilization [27, 102]. The power consumption per vCPU (P_{vcpu}) is then derived using a linear utilization-based power model [35, 39]. A prior study [98] estimated that the average power draw per core in AWS datacenters is 7.5e-4 kW when idle (P_{min}) and 3.5e-3 kW when fully utilized (P_{max}).

$$P_{vcpu} = P_{min} + \frac{cpu_total_time}{t \times n_{vcpu}} \times (P_{max} - P_{min}) \quad (7.3)$$

$$E_{proc} = P_{vcpu} \times n_{vcpu} \times t/3600 \quad (7.4)$$

The operational carbon footprint of data transmission $Carbon_{tran}$ can be estimated by the size of data moved (S), the energy consumption of the data transfer (EF_{trans}), and the average carbon intensity of the route between source and destination I_{route} . This methodology was used in prior network energy characterization studies [15, 30, 40], and it is a simplified version of a state-of-the-art methodology proposed by Tabaeiaghdaei et al. [97]. We refer the reader to the CARIBOU’s repository for further implementation details.

$$Carbon_{tran} = I_{route} \times EF_{trans} \times S \quad (7.5)$$

Estimates of EF_{trans} vary greatly across studies [15, 20, 40, 68, 69, 79, 88, 104]. As the energy efficiency of data transfer doubles approximately every two years [15, 20], we extrapolate EF_{trans} based on these studies to be in the range of 0.001 to 0.005 kWh/GB. To account for this uncertainty due to limitations of today’s network energy models, we included a best-case scenario for offloading (0.001 kWh/GB for any transmission) and a worst-case scenario (0.005 kWh/GB inter- and 0 kWh/GB intra-region transmission). Future more accurate models will most likely not be out of this range for the next few years. CARIBOU’s Metrics Manager can seamlessly integrate alternative models, as demonstrated by a sensitivity study of different transmission energy factors (§9.3).

End-To-End Metric Estimation Estimating the end-to-end latency, cost, and carbon emissions of complex conditional application DAGs is a challenging task [21] warranting more research, especially when dealing with highly variable

serverless traffic [89]. Prior work [36, 58] has used Monte Carlo simulations [71] as an effective strategy to tame this complexity. To support conditional DAGs (§4), the original DAG’s edge invocation probabilities are sampled to determine if edges are taken in each simulation. By sampling each function’s historical execution time and each edge’s transmission latency distributions, the critical path of the partial DAG—and consequently, the end-to-end execution time—is determined. Cost and carbon emissions are also calculated for each simulated scenario. The Monte Carlo simulations are conducted in batches of 200, continuing until reaching a low coefficient of variation below 0.05 for the distributions of end-to-end latency, cost, and carbon footprint or until a maximum of 2,000 samples. For these distributions, the mean represents the "average case" used for DP ordering, and the 95th percentile is the "tail case" used to determine tolerance violations for satisfying the QoS.

7.2 Metric Acquisition and Forecasting

The metrics are based on various sources, requiring different methods of acquisition, processing, and possibly forecasting. Additionally, different metric sources may demand varying acquisition periods (e.g., cloud prices do not change as frequently as carbon intensity). While workflow metrics are collected upon request by the DM, carbon and cost data can be collected periodically.

Learning from Past Invocations. Logs from all function executions are collected, aggregated, and managed by the MM. The MM maintains a list of the daily invocations of every workflow in a distributed storage for the last thirty days and, at most, for the 5,000 latest workflow executions. If more than 5,000 invocations are currently stored, it starts selectively forgetting the oldest invocations: only invocations representing DAG information (e.g., region-to-region latency) not present in new data are maintained, and others are removed in a FIFO manner. This ensures the framework adapts to dynamic workloads without losing data, even if certain DAG information is no longer pursued. MM also gathers metrics from AWS Lambda Insights to calculate each function’s average vCPU utilization and total network traffic.

External Data Sources. The MM also gathers and aggregates data from other external sources, including carbon intensity from *Electricity Maps* [38], *AWS Price List* [14] for latest prices, and *CloudPing* [70] for fresh inter-region latency estimation.

From Learning to Forecasting. Another responsibility of the MM is to perform forecasting with acquired data. Currently, the manager performs this process for forecasting carbon intensities, that are later used by the data modeling process. Carbon forecasting is needed as the diurnal patterns exhibited by several regions mean that the best deployment for carbon objectives might shift throughout

```

1 from caribou import Workflow
2 workflow = Workflow(name="example", version="0.1")
3 @workflow.serverless_function(
4     name = "Example_Function",
5     regions_and_providers = {"allowed_regions": [{"
6         region": "region\1"}]}
7 )
8 def example_function(event):
9     results = workflow.get_predecessor_data()
10    workflow.invoke_serverless_function(
11        next_function, intermediate_data, conditional)

```

Listing 1. Example demonstrating our Python API. The API consists of one class and three functions.

the day; however, solving the ideal DP for every workflow DAG at every hour might be infeasible as the overhead incurred can outweigh the gains. In that case, to ensure that infrequently solved deployments may still benefit from the CARIBOU framework, carbon forecasting is used for reasonable future data prediction. MM accomplishes this by using Holt-Winters Forecasting Exponential Smoothing [52] once every day using the hourly carbon intensities of the previous week as input.

8 Developer Interface

Developers interact with the framework in two ways: 1) the API provided by the *caribou* Python package, and 2) the deployment manifest, where they can provide further deployment details. CARIBOU currently supports Python, a leading language in serverless [33, 51].

API: To declare a workflow, developers use the lightweight API provided by the *caribou* Python package (Listing 1). In a main file named *app.py*, the workflow is instantiated (line 2) to provide helper methods to define the DAG. The function handler to be invoked on requests is registered using a decorator (lines 3-6). The developer can specify function-level configurations, such as region constraints (allow/disallow), to enforce function-level data compliance (line 5). The function parameter (*event*), holds the value passed in the invocation.

A call to *invoke_serverless_function* (line 9) corresponds to a DAG edge. When calling, the handler of the other function and intermediate data is passed as an argument. The developer can optionally pass a Boolean variable to indicate a conditional invocation. This variable is dynamically evaluated when the function is executed, and the condition’s outcome is stored in the distributed annotation table if the successor is a synchronization node.

The developer can indicate a synchronization node in the source code by calling *get_predecessor_data* (line 8). Upon execution, this API call will retrieve intermediate data from the predecessors in the distributed table.

Deployment Manifest: In addition to declaring the workflows implicitly within the code and providing function-level configurations, developers must also configure their workflows in the `iam_policy.json` and `config.yml` file. The `iam_policy.json` file specifies identity and access management (IAM) permissions. These policies are associated with every role linked to the deployed serverless workflow (one per function deployment region).

The `config.yml` allows developers to define workflow-level objectives and tolerances. The developer specifies the "home region"—the initial deployment region of the workflow. Additionally, the developer can define the tolerances on end-to-end latency, carbon emission, and cost per invocation in this file. These are enforced at DP generation.

Lastly, to enforce regulatory compliance on a workflow level, developers can specify regions or providers eligible or prohibited for deployment, where function-level configurations supersede workflow-level ones. If no regions are explicitly allowed, the framework defaults to considering all potential regions.

9 Evaluation

9.1 Methodology

We evaluate CARIBOU end-to-end to demonstrate realistic results in real settings and highlight its capabilities and limitations. The carbon data period is from the 15th to the 21st of October 2023. Due to space limits, we limit the evaluation to the North American regions `us-east-1`, `us-west-1`, `us-west-2`, and `ca-central-1`.

Benchmark workflows: We use five workflows that exemplify serverless workflows from simple one-step functions to complex applications to evaluate CARIBOU:

1. DNA Visualization [28]: a simple single-step workflow that, given a DNA sequence file, generates the corresponding visualization.
2. RAG Data Ingestion [100]: a two-stage pipeline that, given an input PDF document, extracts document metadata and then generates bedrock embeddings for use as a part of a "Document Chat" LLM application.
3. Image Processing [54]: a fan-out application that, given an image and a list of transformations, performs those transformations in parallel.
4. Text2Speech Censoring [36]: similar to the example introduced in §2.4, but with a simplified validation stage.
5. Video Analytics [101]: an application that recognizes objects in video frames by splitting the video into chunks, processing them in parallel, and then joining the results. We evaluated this benchmark using input data from INO's Video Analytics Dataset [48].

Table 1 highlights the differences in DAG structures of these workflows, whether they have synchronization nodes and/or conditional branches (§4), and tested input sizes. We use






Benchmark	DAG Structure	Sync	Cond	Inputs
DNA Visualization		✗	✗	69KB / 1.1MB
RAG Data Ingestion		✗	✗	33 / 115 Pages
Image Processing		✗	✓	222KB / 2.4MB
Text2Speech Censoring		✓	✗	1 KB / 12 KB
Video Analytics		✓	✓	206KB / 2.4MB

Table 1. We used benchmark workflows with different structures, features, and input sizes to evaluate CARIBOU. Benchmark code and input data are publicly available in the CARIBOU GitHub repository.

small and large input sizes to show the sensitivity of our results to input variability. In practice, input sizes may vary greatly and undergo distribution shifts. CARIBOU captures these shifts by learning from the most recent invocations and adapts the deployment plan if necessary.

Workload Invocation and Traffic: We use a uniform invocation pattern to evaluate trade-offs and high-level aspects. Continuous evaluations (§9.5, §9.7) used the 2021 Azure Functions invocation trace [17, 110].

Fair and Controlled Experiments: In all evaluations, we report the relative carbon emissions for comparative arguments about deployments to specific regions. Additionally, we take the following steps to ensure a fair evaluation:

- (1) All benchmarks access external storage and services at or close to their home region. Fixing these external data and services is important because, in the real world, some applications may need to access fixed data or services that cannot be migrated. Migrating these applications can significantly impact performance, cost, and carbon, especially when data is not migrated;
- (2) All experiments evaluate the service time of the application from the moment the request is first received by the first function to the end time of the last function(s) of a workflow. This is to ensure a consistent comparison of workflow performance among alternative deployment plans (§9.4) and orchestration approaches (§9.6);
- (3) We choose random subsamples of inputs from input sources (in indicated size). While CARIBOU is able to handle more complex as well as changing workload patterns, we use this approach to simplify the plots while highlighting the input's impact;
- (4) All experiments show both the best- and worst-case model regarding transmission carbon (§7.1);

9.2 Insights on Effectiveness of Geospatial Shifting

Fig. 7 shows the carbon savings compared with running everything in `us-east-1` using manual static deployment and CARIBOU with different sets of regions. Our insights are:

I1: Static deployment to regions with lower carbon intensity does not necessarily reduce a workflow's carbon

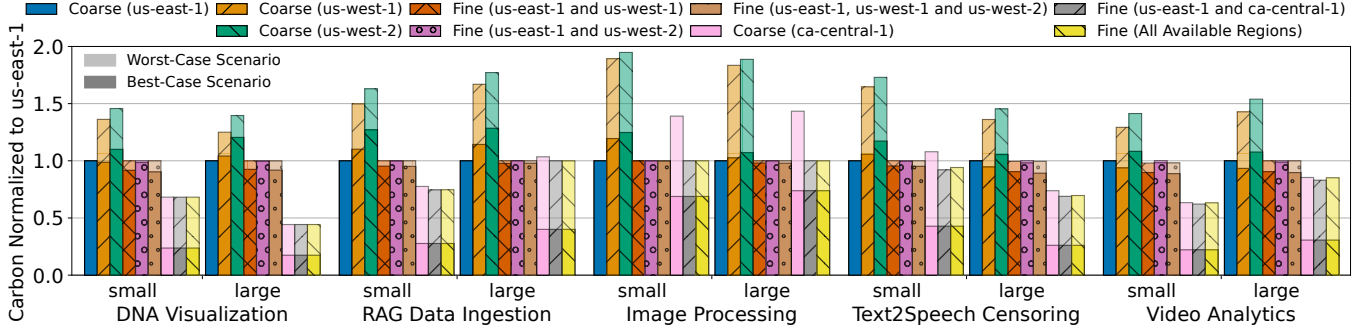


Figure 7. Normalized relative carbon to deploying the workflow in us-east-1 with the two sets of input sizes (small and large) and the two data transmission carbon scenarios (translucent being energy factor 0.005 kWh/GB, no within region carbon and fully colored being 0.001 kWh/GB and same within region).

emissions. us-west-1 and ca-central-1 have 6.1% and 91.5% lower average carbon intensity than us-east-1 in the experiment period (us-west-2 has comparable average intensity). Coarse grained, single-region deployment of workflows can worsen carbon emissions depending on the input size and the transmission carbon scenario. Additionally, any static deployment misses out on leveraging cross-regional carbon intensity variations (§2.1).

I2: An adaptive framework is needed to control geoshifting across workflows, or even for different inputs of the same workflow. CARIBOU tames the spikes in excess carbon emissions through not offloading data transmission heavy applications such as Image Processing in the worst-case, where a non-adaptive framework can cause significant overhead. This shows that adaptiveness is also needed based on applications’ input size. For example, RAG Data Ingestion under the worst-case model, given a small input, could benefit from being singularly offloaded to ca-central-1; however, for the large input, that strategy generates more carbon than simply staying in the home region.

I3: Benefits of geospatial shifting depend on the mix of available regions. This is due to the different and varying carbon intensity patterns in different regions (§2.1). If we zoom into the three two-region combinations for the Text2Speech Censoring benchmark, we observe that depending on the combination, the reductions vary but are all more than any individual involved region. If we extend the two-region combinations with a third, low carbon region such as ca-central-1, we see additional gains. This offloading would not have happened with compliance constraints.

I4: The effectiveness of geospatial shifting depends on the application’s compute-to-transmission ratio. When geospatially shifting, the effectiveness depends on input size and workflow makeup. Transmission-heavy workflows benefit less from geospatial shifting than compute-heavy ones; migrating more data can offset carbon savings from regions

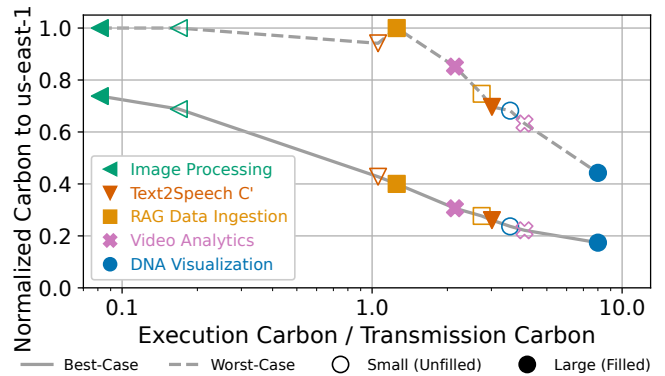


Figure 8. Geospatial shifting offers more carbon savings with increased Execution / Transmission ratio.

with cleaner power grids. Fig. 8 additionally showcases this relationship, where we compare the normalized carbon over the execution to transmission ratio. We calculate the ratio using our modeled energy usage data based on collected workflow execution data.

I5: Holistic geospatial shifting offers substantial gains. CARIBOU considers novel angles to determine carbon-optimal deployments (§5.1). Achieving a carbon reduction of an average (geometric mean) of 22.9% to 66.6%, respectively, for the worst-case and best-case transmission carbon scenarios (Fig. 7), CARIBOU demonstrates the significant potential of carbon-aware geospatial shifting approaches. More research is needed to unlock even more savings for serverless workflows and, more importantly, pave the way for extending techniques presented in this work to new workloads.

9.3 Transmission Energy Intensity to Carbon Saving

Fig. 9 shows the average (geometric mean) results for the five studied workflows given different transmission energy factors and demonstrates the adaptability of the framework to changing carbon accounting and modeling approaches. We evaluated the scenario where the transmission factor is

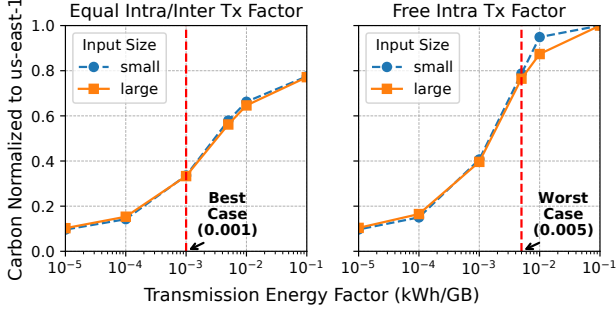


Figure 9. Geometric mean of normalized operational carbon for different transmission energy factors when using CARIBOU for us-east-1, us-west-1, us-west-2, and ca-central-1 regions.

equal between all regions (left sub-figure, *Scenario 1*) and where intra-region data transmission between the same region is free (right sub-figure, *Scenario 2*). The red vertical line shows the best- and worst-case scenarios for shifting (caption of Fig. 7) in the left and right sub-figures, respectively. This study shows that geospatial offloading, while already achieving significant carbon savings, especially in *Scenario 1*, will become increasingly more beneficial as the energy consumption of data transfer continues to decrease [15, 20], the benefit of this increase in energy efficiency is more evident in *Scenario 2*, where the normalized carbon decreases significantly as the transmission energy factor approaches $10e-4$ kWh/GB. As transmission energy factor approaches zero, geospatial shifting frameworks such as CARIBOU can offer carbon reduction of 91.2% (geometric mean), nearly fully leveraging the difference in carbon intensities between power grids (Fig. 2), hindered by differences in application execution time between regions.

9.4 Carbon Efficiency and Latency Tolerance

Incorporating tolerances on DP generation, such as workflow end-to-end latency (outlined in §8) on the generated DPs, enables developers to meet Quality of Service (QoS) requirements while still benefiting from carbon reductions. Fig. 10 illustrates how varying workflow runtime tolerances lead to different deployments with different normalized carbon emissions by evaluating the ratio of the 95-percentile tail service time of deployment over QoS, defined as 95-percentile tail service time of a single region deployment in us-east-1 augmented by runtime tolerance. This ratio, shown in the figure as Relative Time, can be used to visualize if a workflow meets or violates QoS constraints (higher than 1.0 signifies violation). We examine this effect on the DNA Visualization, where we see that as the latency tolerance of workflow execution increases (from 0% to 10% over the deployment in the home region us-east-1), the freedom of the framework to offload parts or the whole application to regions with lower carbon intensity also increases. The framework

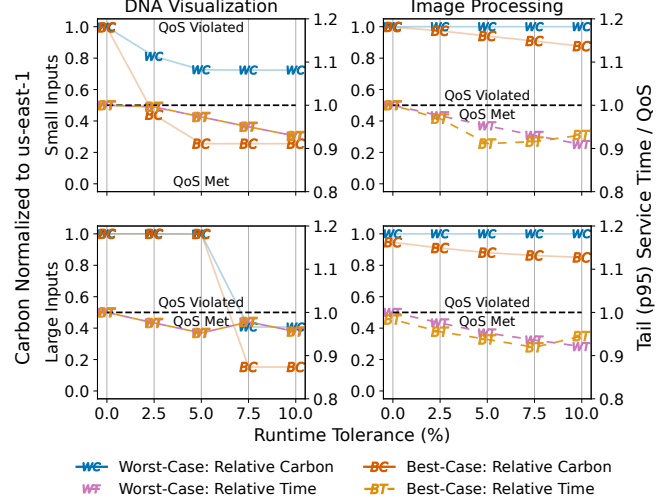


Figure 10. Carbon emissions under different latency tolerances when using CARIBOU to determine optimal DPs for us-east-1, us-west-1, us-west-2, and ca-central-1.

generally observes the QoS tolerances while reducing the relative carbon emissions of the workflow. The decisions CARIBOU makes reflect a conservative end-to-end latency modeling, where for low tolerances, the algorithm generally does not offload, especially since DNA Visualization as a one function application does not offer any offloading of stages off the critical path. Very short-running workflows such as Image Processing are more sensitive to the data transmission scenario (Fig. 12), where the algorithm progressively offloads more and more fan-out nodes as runtime tolerances increase (starting from shorter running nodes to longer running nodes) to adhere to QoS requirements. Additionally, the algorithm generally decides to remain in the home region and thus does not cause any runtime overhead in the worst-case scenario.

9.5 Efficiency of Self-Adaptive Re-Deployment

Fig. 11 captures the week-long operation and visualizes the decisions made by the deployment solver over time. New DP generation points are marked with vertical lines. Initially, the framework enters a learning phase, optimizing deployment regions daily and subsequently transitioning to a lower frequency schedule. Due to a stable input distribution, CARIBOU's DP converges to a relatively stable number of DPs where all three DP generations generate similar 24-hour DPs. However, the frequency of metric collection remains unchanged, given that carbon data and workloads may not always remain stable, and drastic changes may result in previously optimal DP's being suboptimal or, in a more extreme case, more carbon-intense. Generally, CARIBOU generates DPs that offload the workflow to the lowest-carbon region for that time. This is true both under the best-case scenario

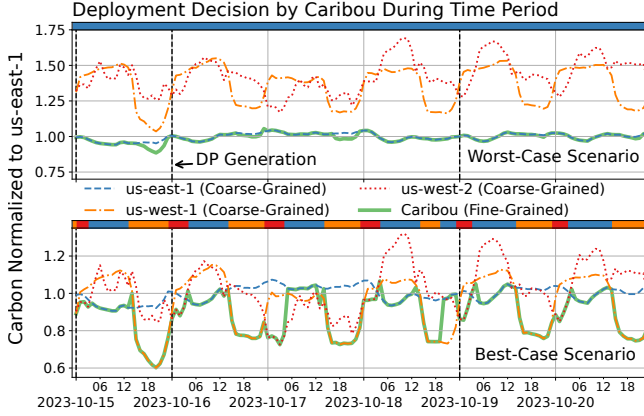


Figure 11. Showcasing CARIBOU’s DP decisions for Text2Speech Censoring using the large input size. The top line of each plot shows our framework’s decisions, highlighting where most workflow nodes are deployed.

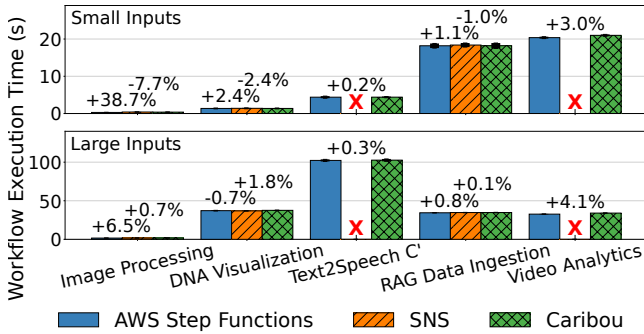


Figure 12. Comparison of workflow execution time between AWS Step Functions, Amazon SNS, and CARIBOU. SNS is included as the function-to-function messaging channel in CARIBOU, but by itself does not support synchronization.

where all three regions are viable offloading options, and under the worst-case scenario where, due to the high transmission carbon overhead of the large input, keeping the majority of the nodes of the workflow in us-east-1 presents the sole feasible solution. In the worst-case scenario, the fine-grained orchestration nature of CARIBOU allows some less transmission intensive nodes to be offloaded to regions with a lower carbon intensity, as observed on October 15 between 2 pm-11 pm. Nonetheless, we observe a notable limitation in carbon emission forecasting on October 17, 6 am-12 pm in the best-case scenario, attributed to a carry-over from the previous day’s inaccurate prediction.

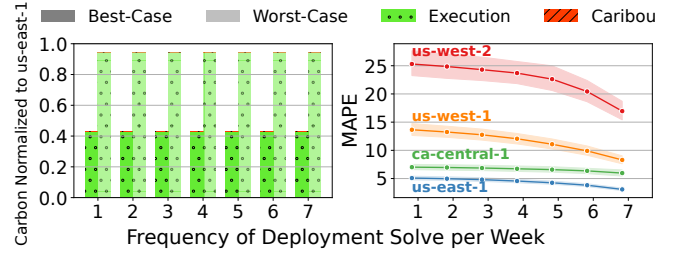


Figure 13. (a) Increasing total carbon per invocation and (b) improved carbon forecasting with more frequent updates.

9.6 Performance Overhead Characterization

We compare the performance overhead of CARIBOU with AWS Step Functions and basic orchestration via SNS to invoke subsequent functions. We compare to AWS Step Functions, as it is a first-party service explicitly designed to provide serverless orchestration. We additionally compare to SNS as CARIBOU uses it, allowing us to assess the additional overhead imposed by our framework. Fig. 12 shows that performing function orchestration on AWS Step Functions results in the lowest workflow execution time, arising from the faster transition between application stages, likely due to the proprietary optimization techniques employed by AWS Step Functions. AWS Step Function is, on average, 12.8% and 2.17% (geometric mean) faster for small and large input sizes, respectively, than similar implementations in SNS. CARIBOU’s function orchestration introduces less than 1% (geometric mean) overhead for small and large input sizes over similar implementations in SNS. When compared to AWS Step Function, CARIBOU incurs 5.72% and 2.71% (geometric mean) additional workflow execution time for small and large input sizes, respectively. Notably, the latency overhead of CARIBOU decreases with increasing application execution duration, as seen from the drop of relative overhead for large input sizes over smaller ones. Conversely, the overhead increases with increasing DAG complexity, as exemplified by contrasting the negligible overhead of a one-stage application of DNA Visualization to the complex application with fan outs and synchronization branches exemplified by Video Analytics.

9.7 Adaptive Learning and Solving Period

We disable the dynamic triggering policy (§5.2) and evaluate the sensitivity of the carbon overhead to different policy determination frequencies. We use the Text2Speech Censoring benchmark with a small input, where the deployment solver runs for a total of ~534s for 24-hour granularity (24 solves) DP generation and incurs ~1.98e-2 gCO₂eq when executed in ca-central-1, with a break-even point of 91 invocations in the worst-case scenario. We sweep the DP generation frequency in a week: from once to seven times. For this study, the number of invocations affects the framework cost per invocation. We chose the 5th percentile DAG reported from

Framework	Objectives	Deployment Granularity	Dynamic Workflow Migration	Geo-spatial	Multi-Stage	Control Flow	Sync Nodes	Transmission Overhead	Supported Providers
AWS Step Functions [10]	✗	Coarse	✗	✗	✓	✓	✓	✗	AWS
GCP Workflows [44]	✗	Coarse	✗	✗	✓	✓	✓	✗	Google
Azure Logic Apps [18]	✗	Coarse	✗	✗	✓	✓	✓	✗	Azure
Serverless Multicloud [112]	Latency Cost	Fine	✗	✗	✓	✗	✗	✗	AWS, Google, Alibaba
BPMN4FO [109]	✗	Coarse	✗	✗	✗	✓	✗	✗	AWS, Azure, IBM
xAFCL [84]	Latency Cost	Fine	✗	✓	✓	✓	✗	✗	AWS, Azure, IBM, Google, Alibaba
OpenTOSCA [105]	✗	Coarse	✗	✗	✓	✓	✓	✗	AWS, Azure, IBM, Google,...
Carbon Aware GSLB [67]	Carbon	Coarse	✗	✓	✓	✗	✗	✗	Azure
GreenCourier [26]	Carbon	Coarse	✗	✓	✗	✗	✗	✗	Google
CARIBOU	Carbon Latency Cost	Fine	✓	✓	✓	✓	✓	✓	AWS

Table 2. Taxonomy of different capabilities of frameworks for serverless cloud workflow deployment.

Azure characterizations [65], with $\sim 1.6K$ average daily invocations. For DAGs with a different number of invocations, the framework overhead would be amortized proportionally. Fig. 13(a) shows that more frequent deployment updates do not incur any significant framework overhead relative to carbon savings but also do not significantly decrease the carbon emissions from running the workflow. Fig. 13(b) shows forecast quality does not worsen linearly with increasing forecast window (leftward X-axis). To reduce forecasting overhead, we re-implemented the Monte Carlo simulation in Go, doubling performance compared to Python while maintaining forecast accuracy. In the case of Text2Speech Censoring, it drops the DP generation time to only $\sim 276s$ for 24-hour granularity (24 solves).

10 Limitations

Based on our discussions with power and sustainable computing experts, we are aware of one limitation of our proposed geospatial shifting vision. Our work assumes that the carbon intensity of electrical grids is independent of workload shifting. This assumption only holds if the power consumption of the shifted workload is small enough compared to the scale of the energy provider. While this dynamic can (and should) be modeled, unfortunately, there is currently no way to associate the marginal carbon intensity of grids with a specific consumer. This remains an open research question for energy and computing experts.

11 Related Work

The background in this space was extensively discussed in §2.2. Table 2 compares CARIBOU with other frameworks for serverless workflow deployment. These frameworks can be broadly categorized into provider-specific solutions for workflow deployment [10, 18, 44], offering a rich toolset for complex applications with additional concepts such as fan-outs. An additional cluster of frameworks concerns themselves with deployment and provisioning of workflow choreographies [84, 109] and topologies [105], offering rich environments for modeling of cloud workflows. Many have also specifically extended their frameworks to deploy serverless workflows in multi-cloud settings [112], but are more

in a proof-of-concept stage of development. Distributing workloads geospatially is an active research area, particularly in ML training [83, 92, 108]. However, the challenge of allocating resources dynamically based on collected data differs significantly from allocating them for one-time use. Moreover, existing solutions are neither comprehensive nor specific enough to support the deployment of arbitrary, complex serverless computation graphs and do not account for the transmission overhead of workflows. Lastly, a host of recent works consider carbon for geospatial workflow deployment [26, 67] or across different generations of hardware [50]. They are not fine-grained, focus on offloading the execution of singular serverless functions geospatially, do not consider transmission carbon, and do not support DAGs.

12 Conclusion

CARIBOU is a framework to reduce the operational carbon emissions of serverless workflows through geospatial shifting. It is the first framework to holistically consider the carbon effect of data transmission, latency and cost implications of migration, and overhead of the control logic. Our findings reveal the untapped potential of geospatial shifting in reducing emissions for seemingly complex workflows. Future research is needed to enhance the accuracy of data transfer emission models, better understand the limits of geospatial shifting, and expand the benefits to broader workloads.

Acknowledgments

We thank Ana Klimovic, Margo Seltzer, Changyuan Lin, Norman Bashir, Arshia Moghimi, Nima Nasiri, Foteini Strati, Juntong Luo, and Mohammadamin Baqers Shahi for their feedback on this work. We also thank the anonymous reviewers, and specially our shepherd, Philip Levis, for helping us improve the paper. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Mitacs Globalink Research Award program, the Swiss-European Mobility Program (SEMP), the Zeno Karl Schindler Foundation, and the Institute for Computing, Information and Cognitive Systems (ICICS) at UBC. This work was enabled by cloud resources from the Digital Research Alliance of Canada (RAS and RAC allocations).

References

- [1] ACM Technology Council. 2021. Computing and Climate Change. <https://dl.acm.org/doi/pdf/10.1145/3483410>. Accessed: 2024-04-17.
- [2] Bilge Acun, Benjamin Lee, Fiodar Kazhamiaka, Aditya Sundarajan, Kiwan Maeng, Manoj Chakkaravarthy, David Brooks, and Carole-Jean Wu. 2023. Carbon dependencies in datacenter design and management. *ACM SIGENERGY Energy Informatics Review* 3, 3 (2023), 21–26.
- [3] Iftikhar Ahmad, Muhammad Imran Khan Khalil, and Syed Adeel Ali Shah. 2020. Optimization-based workload distribution in geographically distributed data centers: A survey. *International Journal of Communication Systems* 33, 12 (2020), e4453.
- [4] Sherif Akoush, Ripduman Sohan, Andrew Rice, Andrew W Moore, and Andy Hopper. 2011. Free lunch: Exploiting renewable energy for computing. In *13th Workshop on Hot Topics in Operating Systems (HotOS XIII)*.
- [5] Amazon Web Services. 2020. Four trends driving global utility digitization. <https://aws.amazon.com/blogs/industries/four-trends-driving-global-utility-digitization/>. Accessed: 2024-04-17.
- [6] Amazon Web Services. 2024. Amazon DynamoDB Pricing. <https://aws.amazon.com/dynamodb/pricing/>. Accessed: 2024-05-07.
- [7] Amazon Web Services. 2024. Amazon EC2 On-Demand Pricing. <https://aws.amazon.com/ec2/pricing/on-demand/>. Accessed: 2024-04-17.
- [8] Amazon Web Services. 2024. Amazon SNS Pricing. <https://aws.amazon.com/sns/pricing/>. Accessed: 2024-05-07.
- [9] Amazon Web Services. 2024. AWS Lambda Pricing. <https://aws.amazon.com/lambda/pricing/>. Accessed: 2024-04-17.
- [10] Amazon Web Services. 2024. AWS Step Functions. <https://aws.amazon.com/step-functions/>. Accessed: 2024-04-17.
- [11] Amazon Web Services. 2024. Configure Lambda function memory. <https://docs.aws.amazon.com/lambda/latest/dg/configuration-memory.html>. Accessed: 2024-04-17.
- [12] Amazon Web Services. 2024. Global Infrastructure Regions & AZs. https://aws.amazon.com/about-aws/global-infrastructure/regions_az/. Accessed: 2024-04-17.
- [13] Amazon Web Services. 2024. Lambda Insights. <https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/Lambda-Insights.html>. Accessed: 2024-04-17.
- [14] Amazon Web Services. 2024. What is AWS Price List? <https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/price-changes.html>. Accessed: 2024-04-17.
- [15] Joshua Aslan, Kieren Mayers, Jonathan G Koomey, and Chris France. 2018. Electricity intensity of internet data transmission: Untangling the estimates. *Journal of industrial ecology* 22, 4 (2018), 785–798.
- [16] Seth Ayers, Sara Ballan, Vanessa Gray, and Rosie McDonald. 2024. Measuring the Emissions & Energy Footprint of the ICT Sector: Implications for Climate Action. *World Bank* (2024).
- [17] Azure. 2021. Azure Functions Invocation Trace 2021. <https://github.com/Azure/AzurePublicDataset/blob/master/AzureFunctionsInvocationTrace2021.md> Accessed: 2024-04-17.
- [18] Azure. 2024. Azure Logic Apps. <https://learn.microsoft.com/en-us/azure/logic-apps/>. Accessed: 2024-04-17.
- [19] Ataollah Fatahi Baarzi, George Kesidis, Carlee Joe-Wong, and Mohammad Shahrad. 2021. On Merits and Viability of Multi-Cloud Serverless. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC '21)*. ACM, 600–608.
- [20] Jonathan Barnsley, Jh enelle A Williams, Simon Chin-Yee, Anthony Costello, Mark Maslin, Jacqueline McGlade, Richard Taylor, Matthew Winning, and Priti Parikh. 2023. Location location location: a carbon footprint calculator for transparent travel to the UN Climate Conference 2022. *UCL Open Environment* 5 (2023).
- [21] Sanjoy Baruah and Alberto Marchetti-Spaccamela. 2023. The Computational Complexity of Feasibility Analysis for Conditional DAG Tasks. 10, 3, Article 14 (9 2023), 22 pages. <https://doi.org/10.1145/3606342>
- [22] Noman Bashir, Tian Guo, Mohammad Hajiesmaili, David Irwin, Prashant Shenoy, Ramesh Sitaraman, Abel Souza, and Adam Wierman. 2021. Enabling sustainable clouds: The case for virtualizing the energy system. In *Proceedings of the ACM Symposium on Cloud Computing*. 350–358.
- [23] Lotfi Belkhir and Ahmed Elmeligi. 2018. Assessing ICT global emissions footprint: Trends to 2040 & recommendations. *Journal of cleaner production* 177 (2018), 448–463.
- [24] John L Bresina. 1996. Heuristic-biased stochastic sampling. In *Proceedings of the thirteenth national conference on Artificial intelligence-Volume 1*. 271–278.
- [25] Canada Energy Regulator. 2024. Provincial and Territorial Energy Profiles – Quebec. <https://www.cer-rec.gc.ca/en/data-analysis/energy-markets/provincial-territorial-energy-profiles/provincial-territorial-energy-profiles-quebec.html>. Accessed: 2024-04-17.
- [26] Mohak Chadha, Thandayuthapani Subramanian, Eishi Arima, Michael Gerndt, Martin Schulz, and Osama Abboud. 2023. Green-Courier: Carbon-Aware Scheduling for Serverless Functions. In *Proceedings of the 9th International Workshop on Serverless Computing*. 18–23.
- [27] Xinghan Chen, Ling-Hong Hung, Robert Cordingly, and Wes Lloyd. 2023. X86 vs. ARM64: An Investigation of Factors Influencing Serverless Performance. In *Proceedings of the 9th International Workshop on Serverless Computing*. 7–12.
- [28] Marcin Copik, Grzegorz Kwasniewski, Maciej Besta, Michal Podstawski, and Torsten Hoeffler. 2021. SeBS: A serverless benchmark suite for function-as-a-service computing. In *Proceedings of the 22nd International Middleware Conference*. 64–78.
- [29] Robert Cordingly, Jasleen Kaur, Divyansh Dwivedi, and Wes Lloyd. 2023. Towards Serverless Sky Computing: An Investigation on Global Workload Distribution to Mitigate Carbon Intensity, Network Latency, and Cost. In *2023 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 59–69.
- [30] Vlad C Coroama, Lorenz M Hilty, Ernst Heiri, and Frank M Horn. 2013. The direct energy demand of internet data flows. *Journal of Industrial Ecology* 17, 5 (2013), 680–688.
- [31] John D. Wilson and Zach Zimmerman. 2023. The Era of Flat Power Demand is Over. <https://gridstrategiesllc.com/wp-content/uploads/2023/12/National-Load-Growth-Report-2023.pdf> Accessed: 2024-04-17.
- [32] Mehیار Dabbagh, Bechir Hamdaoui, Ammar Rayes, and Mohsen Guizani. 2017. Shaving data center power demand peaks through energy storage and workload shifting control. *IEEE Transactions on Cloud Computing* 7, 4 (2017), 1095–1108.
- [33] Datadog. 2023. The State of Serverless 2023. <https://www.datadoghq.com/state-of-serverless/>. Accessed on 2024-03-29..
- [34] Benjamin Davy. 2021. Building an AWS EC2 Carbon Emissions Dataset. <https://medium.com/teads-engineering/building-an-aws-ec2-carbon-emissions-dataset-3f0fd76c98ac/>. (2021). Accessed: 2024-04-17.
- [35] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. 2015. Data center energy consumption modeling: A survey. *IEEE Communications surveys & tutorials* 18, 1 (2015), 732–794.
- [36] Simon Eismann, Johannes Grohmann, Erwin Van Eyk, Nikolas Herbst, and Samuel Kounev. 2020. Predicting the costs of serverless workflows. In *Proceedings of the ACM/SPEC international conference on performance engineering*. 265–276.
- [37] Electricity Maps. 2022. Marginal vs average: which one to use for real-time decisions? <https://www.electricitymaps.com/blog/marginal-vs-average-real-time-decision-making>. Accessed: 2024-08-10.

- [38] Electricity Maps. 2023. Real-time Electricity Map. <https://www.electricitymaps.com>. Accessed: 2024-04-17.
- [39] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. 2007. Power provisioning for a warehouse-sized computer. *ACM SIGARCH computer architecture news* 35, 2 (2007), 13–23.
- [40] Marion Ficher, Françoise Berthoud, Anne-Laure Ligozat, Patrick Sigonneau, Maxime Wisslé, and Badis Tebbani. 2021. Assessing the carbon footprint of the data transmission on a backbone network. In *2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, 105–109.
- [41] Wedan Emmanuel Gnibga, Anne Blavette, and Anne-Cécile Orgerie. 2023. Renewable energy in data centers: the dilemma of electrical grid dependency and autonomy costs. *IEEE Transactions on Sustainable Computing* (2023).
- [42] Google. 2021. We now do more computing where there’s cleaner energy. <https://blog.google/outreach-initiatives/sustainability/carbon-aware-computing-location/> Accessed: 2024-04-17.
- [43] Google. 2022. Our data centers now work harder when the sun shines and wind blows. <https://blog.google/inside-google/infrastructure/data-centers-work-harder-sun-shines-wind-blows/>. Accessed: 2024-08-25.
- [44] Google Cloud Platform. 2024. GCP Workflows. <https://cloud.google.com/workflows/>. Accessed: 2024-04-17.
- [45] Alex Grasas, Angel A Juan, Javier Faulin, Jéscica De Armas, and Helena Ramalhinho. 2017. Biased randomization of heuristics using skewed probability distributions: A survey and some applications. *Computers & Industrial Engineering* 110 (2017), 216–228.
- [46] 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 (2023), 1–28.
- [47] Camilla Hodgson. 2024. Booming AI demand threatens global electricity supply. <https://www.ft.com/content/b7570359-f809-49ce-8cd5-9166d36a057b> Accessed: 2024-04-17.
- [48] INO. 2023. Video Analytics Dataset. <https://www.ino.ca/en/technologies/video-analytics-dataset/>. Accessed: 2024-04-17.
- [49] Paras Jain, Sam Kumar, Sarah Wooders, Shishir G Patil, Joseph E Gonzalez, and Ion Stoica. 2023. Skyplane: Optimizing transfer cost and throughput using Cloud-Aware overlays. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 1375–1389.
- [50] Yankai Jiang, Rohan Basu Roy, Baolin Li, and Devesh Tiwari. 2024. EcoLife: Carbon-Aware Serverless Function Scheduling for Sustainable Computing. arXiv:2409.02085 [cs.DC] <https://arxiv.org/abs/2409.02085>
- [51] Artjom Joosen, Ahmed Hassan, Martin Asenov, Rajkarn Singh, Luke Darlow, Jianfeng Wang, and Adam Barker. 2023. How Does It Function? Characterizing Long-Term Trends in Production Serverless Workloads. In *Proceedings of the 2023 ACM Symposium on Cloud Computing (SoCC ’23)*. ACM, 443–458.
- [52] Prajakta S Kalekar et al. 2004. Time series forecasting using holt-winters exponential smoothing. *Kanwal Rekhi school of information Technology* 4329008, 13 (2004), 1–13.
- [53] Alexey Karyakin and Kenneth Salem. 2017. An analysis of memory power consumption in database systems. In *Proceedings of the 13th International Workshop on Data Management on New Hardware*. 1–9.
- [54] Jeongchul Kim and Kyungyong Lee. 2019. FunctionBench: A suite of workloads for serverless cloud function service. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*. IEEE, 502–504.
- [55] Iwona Kotlarska, Andrzej Jackowski, Krzysztof Lichota, Michal Welnicki, Cezary Dubnicki, and Konrad Iwanicki. 2023. InftyDedup: Scalable and Cost-Effective Cloud Tiering with Deduplication. In *21st USENIX Conference on File and Storage Technologies (FAST 23)*. 33–48.
- [56] Loïc Lannelongue, Jason Grealey, and Michael Inouye. 2021. Green algorithms: quantifying the carbon footprint of computation. *Advanced science* 8, 12 (2021), 2100707.
- [57] Daphne Leprince-Ringuet. 2021. How clean is cloud computing? New data reveals how green Google’s data centers really are. <https://www.zdnet.com/article/how-clean-is-cloud-computing-new-data-reveals-how-green-googles-data-centers-really-are/>. Accessed: 2024-04-17.
- [58] Changyuan Lin, Nima Mahmoudi, Caixiang Fan, and Hamzeh Khazaei. 2023. Fine-Grained Performance and Cost Modeling and Optimization for FaaS Applications. *IEEE Transactions on Parallel and Distributed Systems* 34, 1 (2023), 180–194.
- [59] Changyuan Lin and Mohammad Shahrad. 2024. Bridging the Sustainability Gap in Serverless through Observability and Carbon-Aware Pricing. In *3rd Workshop on Sustainable Computer Systems (HotCarbon ’24)*.
- [60] Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H Low, and Lachlan LH Andrew. 2011. Geographical load balancing with renewables. *ACM SIGMETRICS Performance Evaluation Review* 39, 3 (2011), 62–66.
- [61] Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H Low, and Lachlan LH Andrew. 2011. Greening geographical load balancing. *ACM SIGMETRICS Performance Evaluation Review* 39, 1 (2011), 193–204.
- [62] Zhenhua Liu, Adam Wierman, Yuan Chen, Benjamin Razon, and Nianguan Chen. 2013. Data center demand response: Avoiding the coincident peak via workload shifting and local generation. In *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*. 341–342.
- [63] Pedro García López, Aitor Arjona, Josep Sampé, Aleksander Slominski, and Lionel Villard. 2020. Triggerflow: trigger-based orchestration of serverless workflows. In *Proceedings of the 14th ACM International Conference on Distributed and Event-Based Systems (DEBS ’20)*. ACM, 3–14.
- [64] Jianying Luo, Lei Rao, and Xue Liu. 2013. Temporal load balancing with service delay guarantees for data center energy cost optimization. *IEEE Transactions on Parallel and Distributed Systems* 25, 3 (2013), 775–784.
- [65] Ashraf Mahgoub, Edgardo Barsallo Yi, Karthick Shankar, Eshaan Minocha, Sameh Elnikety, Saurabh Bagchi, and Somali Chaterji. 2022. WISEFUSE: Workload Characterization and DAG Transformation for Serverless Workflows. *Proc. ACM Meas. Anal. Comput. Syst.* 6, 2, Article 26 (6 2022), 28 pages.
- [66] Diptyaroop Maji, Noman Bashir, David Irwin, Prashant Shenoy, and Ramesh K Sitaraman. 2024. The Green Mirage: Impact of Location- and Market-based Carbon Intensity Estimation on Carbon Optimization Efficacy. In *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems (e-Energy ’24)*. ACM, 256–267.
- [67] Diptyaroop Maji, Ben Pfaff, Vipin PR, 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*. 1–6.
- [68] Jens Malmodin. 2020. Science & Society Forum: Växande IKT-sektor och fler datacenter – hur påverkas elförsörjningen? (2020). https://www.youtube.com/watch?v=Xo0PB5i_b4Y&t=2520s IVA-conference Science & Society Forum.
- [69] Jens Malmodin, Nina Lövehagen, Pernilla Bergmark, and Dag Lundén. 2024. ICT sector electricity consumption and greenhouse gas emissions – 2020 outcome. *Telecommunications Policy* (2024), 102701.
- [70] Matt Adorjan. 2024. AWS Latency Monitoring. <https://www.cloudping.co/grid>. Accessed: 2024-04-17.

- [71] Nicholas Metropolis and Stanislaw Ulam. 1949. The Monte Carlo Method. *J. Amer. Statist. Assoc.* 44, 247 (1949), 335–341.
- [72] Michael Sauter. 2020. Crane - Lift containers with ease. <https://michaelsauter.github.io/crane/index.html>. Accessed: 2024-04-17.
- [73] Janine Morley, Kelly Widdicks, and Mike Hazas. 2018. Digitalisation, energy and data demand: The impact of Internet traffic on overall and peak electricity consumption. *Energy Research & Social Science* 38 (2018), 128–137.
- [74] David Mytton and Masaō Ashtine. 2022. Sources of data center energy estimates: A comprehensive review. *Joule* 6, 9 (2022), 2032–2056.
- [75] Deepak Narayanan, Keshav Santhanam, Fiodar Kazhemiaka, Amar Phanishayee, and Matei Zaharia. 2020. Analysis and exploitation of dynamic pricing in the public cloud for ML training. In *VLDB DISPA Workshop 2020*.
- [76] Thong Trung Nguyen, Thu Anh Thi Pham, and Huong Thi Xuan Tram. 2020. Role of information and communication technologies and innovation in driving carbon emissions and economic growth in selected G-20 countries. *Journal of environmental management* 261 (2020), 110162.
- [77] Intergovernmental Panel on Climate Change (IPCC). 2023. *Annex VII: Glossary*. Cambridge University Press, 2215–2256.
- [78] Brad Plumer and Nadja Popovich. 2024. A New Surge in Power Use Is Threatening U.S. Climate Goals. <https://www.nytimes.com/interactive/2024/03/13/climate/electric-power-climate-change.html> Accessed: 2024-04-17.
- [79] Lorenzo Posani, Alessio Paochia, and Marco Moschettini. 2018. The carbon footprint of distributed cloud storage. *arXiv preprint arXiv:1803.06973* (2018).
- [80] Ana Radovanović, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, et al. 2022. Carbon-aware computing for datacenters. *IEEE Transactions on Power Systems* 38, 2 (2022), 1270–1280.
- [81] Sabidur Rahman, Abhishek Gupta, Massimo Tornatore, and Biswanath Mukherjee. 2017. Dynamic workload migration over backbone network to minimize data center electricity cost. *IEEE Transactions on Green Communications and Networking* 2, 2 (2017), 570–579.
- [82] Joseph Rand, Rose Strauss, Will Gorman, Joachim Seel, Julie Mulvaney Kemp, Seongeun Jeong, Dana Robson, and Wiser Ryan. 2022. Queued Up: Characteristics of Power Plants Seeking Transmission Interconnection As of the End of 2022. https://emp.lbl.gov/sites/default/files/emp-files/queued_up_2022_04-06-2023.pdf Accessed: 2024-04-17.
- [83] Ray. 2024. Welcome to Ray. <https://docs.ray.io/en/latest/>. Accessed: 2024-08-11.
- [84] Sasko Ristov, Stefan Pedratscher, and Thomas Fahringer. 2021. xAFCL: Run scalable function choreographies across multiple FaaS systems. *IEEE Transactions on Services Computing* (2021).
- [85] Francisco Romero, Mark Zhao, Neeraja J. Yadwadkar, and Christos Kozyrakis. 2021. Llama: A Heterogeneous & Serverless Framework for Auto-Tuning Video Analytics Pipelines. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC '21)*. ACM, 1–17.
- [86] Ghazal Sadeghian, Mohamed Elsakhawy, Mohanna Shahrad, Joe Hattori, and Mohammad Shahrad. 2023. UnFaaSener: Latency and Cost Aware Offloading of Functions from Serverless Platforms. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. 879–896.
- [87] Trever Schirmer, Nils Japke, Sofia Greten, Tobias Pfandzelter, and David Bermbach. 2023. The Night Shift: Understanding Performance Variability of Cloud Serverless Platforms. In *Proceedings of the 1st Workshop on SErverless Systems, Applications and MEthodologies*. 27–33.
- [88] Anders SG Andrae. 2020. New perspectives on internet electricity use in 2030. *Engineering and Applied Science Letter* 3, 2 (2020), 19–31.
- [89] Mohammad Shahrad, Rodrigo Fonseca, Inigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. 2020. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In *2020 USENIX annual technical conference (USENIX ATC 20)*. 205–218.
- [90] Prateek Sharma. 2023. Challenges and opportunities in sustainable serverless computing. *ACM SIGENERGY Energy Informatics Review* 3, 3 (2023), 53–58.
- [91] Zhiming Shen, Qin Jia, Gur-Eyal Sela, Ben Rainero, Weijia Song, Robbert van Renesse, and Hakim Weatherspoon. 2016. Follow the sun through the clouds: Application migration for geographically shifting workloads. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*. 141–154.
- [92] Foteini Strati, Paul Elvinger, Tolga Kerimoglu, and Ana Klimovic. 2024. ML Training with Cloud GPU Shortages: Is Cross-Region the Answer?. In *Proceedings of the 4th Workshop on Machine Learning and Systems*. 107–116.
- [93] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2020. Energy and Policy Considerations for Modern Deep Learning Research. *Proceedings of the AAAI Conference on Artificial Intelligence* 09 (2020), 13693–13696.
- [94] Thanathorn Sukprasert, Noman Bashir, Abel Souza, David Irwin, and Prashant Shenoy. 2024. On the Implications of Choosing Average versus Marginal Carbon Intensity Signals on Carbon-aware Optimizations. In *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems*. 422–427.
- [95] Thanathorn Sukprasert, Abel Souza, Noman Bashir, David Irwin, and Prashant Shenoy. 2023. Spatiotemporal Carbon-aware Scheduling in the Cloud: Limits and Benefits. In *Companion Proceedings of the 14th ACM International Conference on Future Energy Systems*.
- [96] 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 *Nineteenth European Conference on Computer Systems (EuroSys), Athens, Greece*.
- [97] Seyedali Tabaeiaghdaei, Simon Scherrer, Jonghoon Kwon, and Adrian Perrig. 2023. Carbon-Aware Global Routing in Path-Aware Networks. In *Proceedings of the 14th ACM International Conference on Future Energy Systems*. 144–158.
- [98] Thoughtworks. 2023. Cloud Carbon footprint Methodology. <https://www.cloudcarbonfootprint.org/docs/methodology/>. Accessed: 2024-04-17.
- [99] Adel Nadjaran Toosi, Chenhao Qu, Marcos Dias de Assunção, and Rajkumar Buyya. 2017. Renewable-aware geographical load balancing of web applications for sustainable data centers. *Journal of Network and Computer Applications* 83 (2017), 155–168.
- [100] UBC Cloud Innovation Centre. 2024. <https://github.com/UBC-CIC/document-chat/tree/main>. Accessed: 2024-09-11.
- [101] vHive Ecosystem. 2024. vSwarm: A suite of representative serverless cloud-agnostic (i.e., dockerized) benchmarks. <https://github.com/vhive-serverless/vSwarm> Accessed: 2024-04-17.
- [102] Liang Wang, Mengyuan Li, Yinqian Zhang, Thomas Ristenpart, and Michael Swift. 2018. Peeking behind the curtains of serverless platforms. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. 133–146.
- [103] 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*. 260–272.
- [104] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. 2022. Sustainable AI: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems* 4 (2022), 795–813.

- [105] Michael Wurster, Uwe Breitenbücher, Kálmán Képes, Frank Leymann, and Vladimir Yussupov. 2018. Modeling and automated deployment of serverless applications using TOSCA. In *2018 IEEE 11th conference on service-oriented computing and applications (SOCA)*. IEEE, 73–80.
- [106] Zongheng Yang, Zhanghao Wu, Michael Luo, Wei-Lin Chiang, Romil Bhardwaj, Woosuk Kwon, Siyuan Zhuang, Frank Sifei Luan, Gautam Mittal, Scott Shenker, and Ion Stoica. 2023. SkyPilot: An Intercloud Broker for Sky Computing. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 437–455.
- [107] Xianyu Yu, Yuezhi Hu, Dequn Zhou, Qunwei Wang, Xiuzhi Sang, and Kai Huang. 2023. Carbon emission reduction analysis for cloud computing industry: can carbon emissions trading and technology innovation help? *Energy Economics* 125 (2023), 106804.
- [108] Binhang Yuan, Yongjun He, Jared Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy S Liang, Christopher Re, and Ce Zhang. 2022. Decentralized training of foundation models in heterogeneous environments. *Advances in Neural Information Processing Systems* 35 (2022), 25464–25477.
- [109] Vladimir Yussupov, Jacopo Soldani, Uwe Breitenbücher, and Frank Leymann. 2022. Standards-based modeling and deployment of serverless function orchestrations using BPMN and TOSCA. *Software: Practice and Experience* 52, 6 (2022), 1454–1495.
- [110] Yanqi Zhang, Íñigo Goiri, Gohar Irfan Chaudhry, Rodrigo Fonseca, Sameh Elnikety, Christina Delimitrou, and Ricardo Bianchini. 2021. Faster and cheaper serverless computing on harvested resources. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*. 724–739.
- [111] Yanwei Zhang, Yefu Wang, and Xiaorui Wang. 2011. Greenware: Greening cloud-scale data centers to maximize the use of renewable energy. In *Middleware 2011: ACM/IFIP/USENIX 12th International Middleware Conference, Lisbon, Portugal, December 12-16, 2011. Proceedings 12*. Springer, 143–164.
- [112] Haidong Zhao, Zakaria Benomar, Tobias Pfandzelter, and Nikolaos Georgantas. 2022. Supporting Multi-Cloud in Serverless Computing. In *2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*. IEEE, 285–290.
- [113] 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.
- [114] Zhi Zhou, Fangming Liu, Yong Xu, Ruolan Zou, Hong Xu, John CS Lui, and Hai Jin. 2013. Carbon-aware load balancing for geo-distributed cloud services. In *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, 232–241.