# Designing Cloud Servers for Lower Carbon

Jaylen Wang*, Daniel S. Berger†,‡, Fiodar Kazhamiaka†, Celine Irvene†, Chaojie Zhang†,
Esha Choukse†, Kali Frost†, Rodrigo Fonseca†, Brijesh Warrier†, Chetan Bansal†, Jonathan Stern†,
Ricardo Bianchini†, Akshitha Sriraman*

*Carnegie Mellon University †Microsoft ‡University of Washington

## 1 Introduction

To combat climate change, we must reduce carbon emissions from Information and Communication Technology (ICT), which can cause 20% of global emissions by 2030 [6]. Cloud computing will be a major part of ICT's emissions [6]. Thus, we must reduce cloud computing's emissions.

To reduce ICT's emissions from cloud computing, we must reduce the cloud's *operational emissions* (e.g., from producing electricity to run data centers) and *embodied emissions* (e.g., from semiconductor fabs that make server components) [6]. Today, embodied emissions account for 50%–82% of cloud emissions, due to energy optimizations and renewable energy use [6]. Thus, it is crucial to reduce both emission types.

To reduce cloud computing's operational and embodied emissions, we identify designing carbon-efficient cloud compute server Stock Keeping Units (SKUs) as a promising solution. Server SKU design is the process by which selected hardware components are composed into a server. Typically, cloud providers design compute server SKUs to meet performance and cost goals. To reduce emissions, we introduce a new way of designing carbon-efficient compute server SKUs, or "*GreenSKUs*," that trade off performance for lower carbon.

We find that deploying *GreenSKUs* is promising for three reasons. First, we use a data center carbon model [2] to find that compute servers at Azure cause most cloud emissions, as shown in the carbon breakdown in Fig. 1. Second, cloud servers are often underutilized [4], making a case for right-sizing server performance to save emissions. Third, *GreenSKU* deployment is more feasible today with the availability of carbon-efficient commodity server components, e.g., energy-efficient cores [1].

Due to *GreenSKUs*' promise, we design and build three *GreenSKUs* using low-carbon components that mitigate cloud compute servers' key sources of operational and embodied emissions[1]. Our *GreenSKUs* incrementally incorporate three low-carbon components: energy-efficient high-thread-count CPUs [1], reuse of old DRAM with Compute Express Link (CXL) [14], and reuse of old Solid State Drives (SSDs).

[1]This work was published in the proceedings of 2024 International Symposium on Computer Architecture [16].

*The 2nd Workshop on Hot Topics in System Infrastructure (HotInfra 2024), co-located with SOSP 2024, November 3, 2024, Austin, TX, USA*

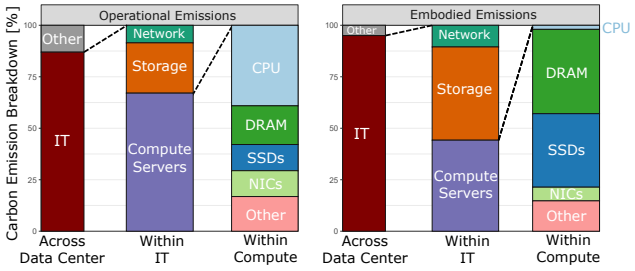


**Figure 1.** Carbon breakdown of general-purpose data centers at Azure.

While *GreenSKUs* promise carbon savings, we demonstrate several challenges that limit cloud providers from practically deploying them. First, a *GreenSKU* may compromise performance. Thus, it is challenging to design *GreenSKUs* while navigating their emissions vs. performance tradeoff, to justify deploying them at scale. Second, a *GreenSKU*'s operational and embodied emissions can have complex tradeoffs. For example, reusing older server components can reduce embodied emissions but may increase operational emissions due to lower energy efficiency [5]. We refer to these factors that impact a *GreenSKU*'s deployment potential, as the *GreenSKU*'s "adoption." It is critical and challenging for cloud providers to identify which *GreenSKU* designs are adoptable.

To address these challenges, we develop a novel framework, *GSF*, to enable cloud providers to evaluate a *GreenSKU*'s carbon savings in the cloud. *GSF* systematically considers the major factors, such as performance and adoption, that influence a *GreenSKU*'s benefits at scale. *GSF*'s components model each major factor and their relationships.

We implement *GSF* within Microsoft Azure's production constraints to evaluate our three *GreenSKUs*' carbon savings at scale. Using our *GSF* implementation, we show that our *GreenSKUs* reduce carbon emissions per core by 28% compared to currently-deployed cloud servers at Azure. When deploying *GreenSKUs* in a way that meets applications' performance goals, we reduce emissions by 15%. Finally, when incorporating overall data center overheads, our *GreenSKUs* reduce net cloud emissions by 8%, which is a significant reduction at scale.

## 2 Our GreenSKU Prototypes

We show the potential in building *GreenSKUs* by building prototype *GreenSKUs* that target the top three carbon contributors in compute servers: CPUs, DRAM, and SSDs (Fig. 1).

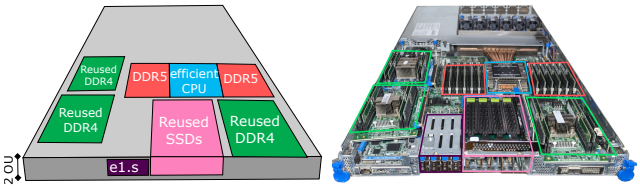


**Figure 2.** Our *GreenSKU-Full* design with AMD's efficient CPU, reused DDR4 DRAM (via CXL), and reused m.2 SSDs (via e1.s and PCIe adapters).

**Low-carbon components.** We use three low-carbon components in our prototypes. First, we use efficient CPUs with high core counts, which are now widely available and can reduce operational emissions [13]. We choose AMD Bergamo in our *GreenSKU* prototype, as it is on the market today and has full support for CXL.mem devices.

Next, at Azure, we find that numerous old DDR4 DIMMs can be reused, thus reducing embodied emissions, due to their host servers reaching the end of their deployment. Critically, internal studies indicate that these old DIMMs show no sign of increasing failure rates. Historically, reusing old DRAM was challenging as DDR generations are not backward compatible. However, today's commodity CXL controllers enable attaching old DDR4 to modern DDR5 systems.

Finally, we attach reused SSDs from decomissioned Azure servers via PCIe. While modern SSDs fail due to exhausting flash erasure cycles [11], even after seven years, most SSDs offer more than half of the guaranteed erasure cycles.

**Prototype SKUs.** We build three *GreenSKUs* by incrementally adding each carbon-efficient component (see Table 1): (1) *GreenSKU-Efficient* uses AMD's efficient Bergamo CPU, (2) *GreenSKU-CXL* adds reused, CXL-attached DDR4, and (3) *GreenSKU-Full* adds reused SSDs. Fig. 2 shows a logical diagram and an image of our *GreenSKU-Full* prototype.

**Performance characteristics.** Low-carbon components may have lower performance than the components of *baseline SKUs*, which are currently-deployed SKUs at Azure. For example, AMD Bergamo has a lower frequency and less LLC capacity than AMD generations deployed at Azure.

Reusing DDR4 memory via CXL incurs higher latency [8] and lower memory bandwidth than baseline SKUs. To reduce such slowdowns, we use Pond's approach [8]. This approach ensures that 98% of applications incur <5% slowdown with CXL, compared to running entirely with DDR5.

Reused SSDs also provide lower bandwidth and IOPS. We mitigate lower SSD performance using striped RAID sets that each offer more bandwidth and IOPS than the baseline configurations; thus, old SSDs have no adoption side effects.

Challenges such as these performance impacts may cause a *GreenSKU* to not be deployable, requiring a systematic way to evaluate a *GreenSKU*'s benefits at scale.

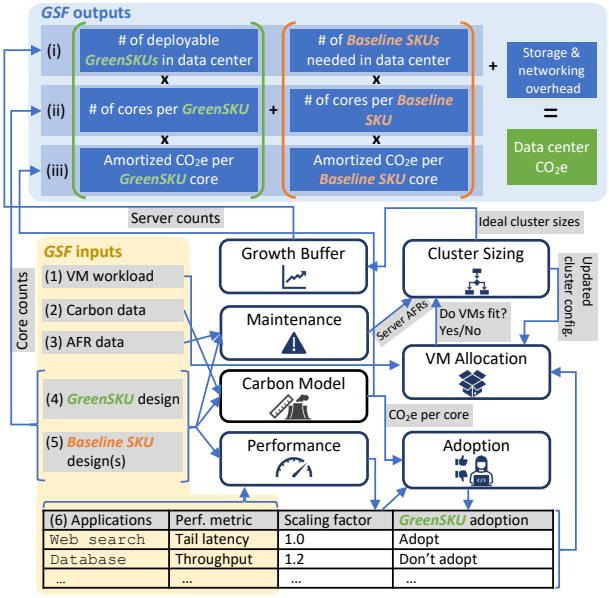


**Figure 3.** Overview of the *GreenSKU* Framework (*GSF*).

## 3 GSF: The GreenSKU Framework

To evaluate a *GreenSKU*, *GSF* estimates a data center's emissions from deploying a *GreenSKU*. *GSF* considers seven major factors that influence a *GreenSKU*'s carbon savings. It considers each factor using distinct *components*, as shown in Fig. 3. *GSF* specifies initial inputs (in yellow) to calculate intermediate outputs (in blue boxes) for the *GreenSKU* and baseline SKUs, which produce the final output, i.e., data center emissions from deploying a *GreenSKU*.

We organize *GSF*'s components into three levels, i.e., the server-, rack-, and data center-level, based on which level of the data center the component models. We discuss each component by specifying the factor it considers and how we implement it under Azure's production constraints.

**Carbon model.** *GSF* requires a carbon model that computes a SKU's emissions. Our novel carbon model implementation systematically aggregates embodied and operational emissions from server, rack, and data center components. As in prior work [15], we consider reused server components to be in their "second life," with zero embodied emissions. For more details, we have open-sourced our carbon model [2].

**Performance.** Since an application running on a *GreenSKU* may face a lower performance-per-core, it may scale up/out to suitably serve the target workload. *GSF*'s performance component must quantify such performance effects.

To implement this component, we benchmark the performance of 20 open- and closed-source applications that span the six application classes that run on most VMs in Azure [12]. We measure a *GreenSKU*'s performance by setting a Service Level Objective (SLO) based on the performance of three deployed baseline SKU generations, Gen 1, 2, 3, where Gen3 is a future generation that would be deployed

instead of a *GreenSKU*. To achieve comparable performance, we scale up the number of VM cores on the *GreenSKU* and compare its performance against the baseline SKU. We then calculate the *scaling factor*, which defines how many *GreenSKU* cores per baseline SKU core are needed to meet an application's performance goals.

**Maintenance.** Server failures require an overhead of additional servers [10]. As *GreenSKUs* may influence a server's Annual Failure Rate (AFR), we must calculate this overhead. We estimate this overhead via an open-source maintenance model [2] that uses component AFRs to calculate server maintenance overheads across all SKUs.

**Adoption.** This component helps decide which applications can run on a *GreenSKU* while meeting deployment goals (e.g., performance goals). To implement this component, we calculate the carbon required to service the application on a *GreenSKU*. To this end, we multiply the scaling factor from the performance component by the $CO_2$e-per-core determined from the carbon model. We also calculate this value for the baseline SKU. For each application, we model that an application will adopt a *GreenSKU* if the calculated carbon value to run the application on the *GreenSKU* is lower than the baseline SKU's value.

**VM allocation and packing.** Server design impacts how well VMs can be packed into a cluster [7]. We account for this factor using an internal VM allocation simulator with VM arrival/departure traces from multiple Azure data centers.

Since the applications running in VMs are opaque in production traces, we assign VMs to one of our representative benchmark applications. We determine whether a VM can run on a *GreenSKU* using our adoption model and the VM's application assignment. If a VM adopts the *GreenSKU*, we apply the relevant scaling factor to the VM's resources.

**Cluster sizing.** This component finds how many baseline SKUs and *GreenSKUs* are required to service a data center's VM workload. We use a search algorithm via our allocation simulator to find the number of each SKU that minimizes emissions while supporting our VM workload.

**Growth buffer.** Cloud providers maintain a *growth buffer*, i.e., extra server capacity to handle spikes in VM demand. We account for this capacity as an overhead of baseline SKUs.

## 4 Evaluating our GreenSKUs Using GSF

We use *GSF* at Azure to evaluate our *GreenSKUs* from §2.

**Evaluating GreenSKUs' performance and adoption.** To evaluate our *GreenSKUs*' performance, we measure the $95^{th}$% tail latency across different Queries Per Second (QPS) for each application. We set an application's SLO as the $95^{th}$% tail latency achieved at 90% of the peak saturation throughput when the application is run on the baseline SKU [9]. We conduct three trials and report 99% confidence intervals.

We first evaluate *GreenSKU-Efficient*'s performance and adoption. For brevity, in Fig. 4, we show results for one

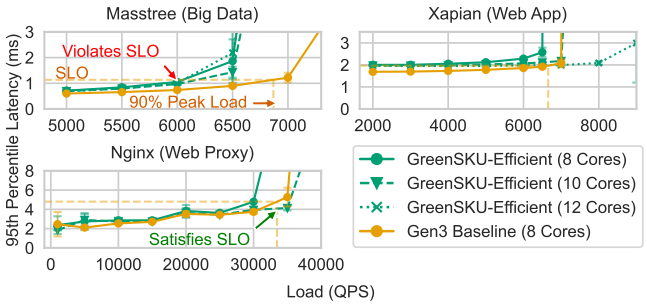


**Figure 4.** $95^{th}$% tail latency vs. load (QPS) for applications spanning three of our six application classes. Tail latency is shown for 8-core configurations for the Gen3 baseline SKU (in orange). The dotted orange line indicates an SLO set using Gen3's latency at 90% of peak load. For some applications (e.g., `Xapian` and `Nginx`), *GreenSKU-Efficient* achieves the SLO with scaling; for other applications (e.g., `Masstree`), the scaling required outweighs carbon savings.

representative application in three of our classes. We show results up to the minimum number of cores on *GreenSKU-Efficient* that achieve a peak saturation throughput closest to that of our Gen3 baseline SKU. We omit results for Gen1 and Gen2, as they consistently perform worse than Gen3.

For applications such as `Masstree`, even with 12 cores, *GreenSKU-Efficient* cannot match Gen3's peak throughput. However, for other applications, like `Xapian` and `Nginx`, *GreenSKU-Efficient* achieves SLOs with 10–12 cores. Thus, *GreenSKU-Efficient* effectively meets the performance goals of several latency-critical applications, albeit with scaling.

We repeat this study for all 20 applications, calculating the scaling factors required relative to the three baseline SKUs. For seven applications, *GreenSKU-Efficient* meets Gen3's SLO without scaling. For another nine applications, scaling by 25% is required to achieve Gen3's SLO. Our *GSF* adoption component's implementation notes the applications that cannot be run on the *GreenSKU* if they offset carbon savings. Thus, through *GSF*, we see that the *GreenSKU* meets the performance goals of most applications while saving carbon.

Next, we study *GreenSKU-CXL*'s performance to evaluate older memory reuse via CXL. We find that 20.2% of our applications, weighted by fleet core-hours, do not face significant performance penalties when running on *GreenSKU-CXL*.

**Evaluating GreenSKUs' impact on VM packing.** We now evaluate how well we can pack VMs on our *GreenSKUs*. We run VM packing simulations and our cluster-sizing algorithm on 35 production VM traces.

In particular, when packing VMs on our *GreenSKUs*, we use our simulator to validate whether there is enough untouched memory to back on CXL memory, thus mitigating CXL-induced performance loss. To this end, each VM reports the maximum amount of allocated memory that it uses over its lifetime. To aggregate the maximum memory usage across all VMs on each server, we periodically take snapshots of

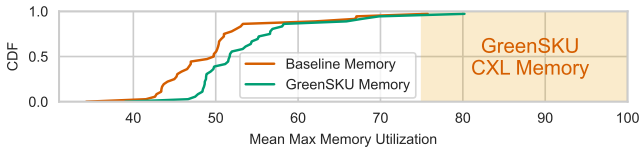


**Figure 5.** CDF of the mean per-server max memory utilization across all servers for a cluster of baseline SKUs and *GreenSKU-CXLs*. The shaded region shows memory accessed through CXL on the *GreenSKU*. Almost all servers can service their VMs' memory demand with local DDR5 memory.

| SKU Config. | # Cores | # × DIMM (GB) | # × SSD (TB) | Operational Savings | Embodied Savings | Total Savings |
|---|---|---|---|---|---|---|
| Baseline | 80 | 12 × 64 | 6 × 2 | - | - | - |
| *GreenSKU-Efficient* | 128 | 12 × 96 | 5 × 4 | 29% | 14% | 23% |
| *GreenSKU-CXL* | 128 | 12 × 64<br>8 × 32 CXL | 5 × 4 | 23% | 25% | 24% |
| *GreenSKU-Full* | 128 | 12 × 64<br>8 × 32 CXL | 2 × 4<br>12 × 1 Reuse | 17% | 43% | 28% |

**Table 1.** Per-core operational, embodied, and total carbon savings relative to our Gen3 baseline SKU for three incremental GreenSKU configurations.

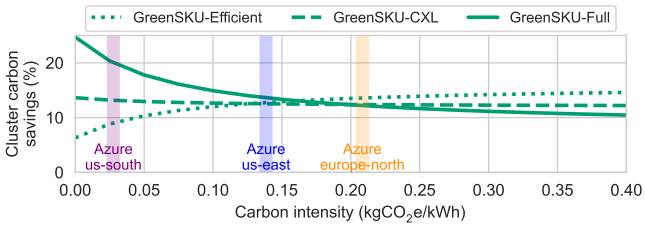


**Figure 6.** End-to-end cluster-level carbon savings relative to all baseline SKU clusters across carbon intensities evaluated for our three *GreenSKUs*. Vertical lines mark estimated carbon intensities for three Azure regions [3]. The best *GreenSKU* depends on a data center's operating conditions.

the servers. We then average across servers and snapshots to identify a cluster's average maximum memory utilization.

We show a CDF of the mean per-server maximum memory utilization in Fig. 5 for clusters with both baseline-only SKUs and *GreenSKU-CXLs*. The shaded portion is the 25% of memory reused via CXL. In most traces, we see a maximum memory utilization below 60%, which can be accommodated by *GreenSKU-CXL*'s local memory. Only 3% of traces have a memory utilization that would require using CXL memory.

Moreover, we can leverage our observation that CXL does not cause a performance loss for 20% of our applications, to schedule such applications to use CXL-backed memory. Thus, we show that reusing old DRAM via CXL does not significantly impact *GreenSKU-CXL*'s adoption at Azure.

**Evaluating GreenSKUs' carbon savings.** We now evaluate our *GreenSKUs*' carbon savings using the average grid Carbon Intensity (CI), i.e., the amount of carbon emitted per unit of electricity generated, across major Azure data center regions. Table 1 shows our *GreenSKUs*' per-core operational and embodied emissions savings over the Gen3 baseline SKU. *GreenSKU-Full*, with our three carbon-efficient components, emits 28% less carbon than the baseline SKU.

We now evaluate our *GreenSKUs*' cluster-level carbon savings using *GSF*'s output. In Fig. 6, we show the cluster-level carbon savings achieved when introducing each of our *GreenSKUs* compared to a cluster of all baseline SKUs. We evaluate across a spectrum of CI values. We show the CI values for three Azure data center regions.

The cluster with *GreenSKUs* saves up to 25% carbon. The lowest-carbon *GreenSKU* varies with a region's CI. At higher CI, *GreenSKU-Efficient* is effective due to not having energy-inefficient reused components. At lower CI, embodied emissions dominate, so *GreenSKU-Full* is effective. These subtle tradeoffs between performance, carbon, and other *GSF* factors, motivate the need for *GSF*, to help evaluate *GreenSKUs*.

Even when factoring emissions sources beyond compute servers, deploying *GreenSKUs* reduces cluster-wide savings by 15% and net cloud emissions by 8%, which is a significant reduction at scale.

## 5 Discussion

We analyze *GreenSKUs*' monetary cost and discuss the need to develop run-time schedulers for *GreenSKUs*.

**GreenSKU cost analysis.** *GSF* can be adapted to analyze *GreenSKUs*' effect on Total Cost of Ownership (TCO) by replacing the carbon model with a TCO model. To this end, our TCO analysis reveals that a cost-efficient server SKU is only 5% less costly compared to our carbon-efficient *GreenSKU*. This relatively small TCO loss may be tolerable to a cloud provider seeking to meet aggressive decarbonization targets. Moreover, cloud providers can use *GSF* to evaluate other SKUs that achieve the required carbon vs. cost tradeoff.

**Scheduling applications on GreenSKUs.** Run-time schedulers that leverage *GreenSKUs*, post-deployment, can help realize a *GreenSKU*'s full carbon-saving potential. To this end, we are developing a carbon-aware scheduling system that can suitably schedule components of distributed applications on *GreenSKUs* [17]. At run-time, our scheduling system decides how many *GreenSKU* and baseline SKU VMs are required to schedule each component of a service, to minimize carbon while meeting the service's performance goals.

## Acknowledgement

We thank the anonymous reviewers for their insightful feedback. This work was partially supported by an NSF CAREER Award CCF-2340042. Jaylen Wang was supported by an NSF Graduate Research Fellowship, a Benjamin Garver Lamme/Westinghouse Graduate Fellowship, and a Jack and Mildred Bowers Scholarship in Engineering.

## References


[1] [n. d.]. AMD adds 128-core Bergamo and 3D V-Cache Genoa CPUs to Zen 4 Epyc lineup. https://www.xda-developers.com/amd-128-core-bergame-genoa-epyc-cpu/. (Accessed on 04/26/2024).

[2] [n. d.]. GreenSKU-Model. https://github.com/Azure/AzurePublicDataset.

[3] [n. d.]. Microsoft Datacenter Statistics and Sustainability Fact Sheets. https://datacenters.microsoft.com/. (Accessed on 04/26/2024).

[4] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In *Symposium on Operating Systems Principles*.

[5] Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S. Lee, David Brooks, and Carole-Jean Wu. 2022. ACT: Designing Sustainable Computer Systems With An Architectural Carbon Modeling Tool. In *International Symposium on Computer Architecture*.

[6] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. 2021. Chasing Carbon: The Elusive Environmental Footprint of Computing. In *International Symposium on High-Performance Computer Architecture*.

[7] Ori Hadary, Luke Marshall, Ishai Menache, Abhisek Pan, Esaias E Greeff, David Dion, Star Dorminey, Shailesh Joshi, Yang Chen, Mark Russinovich, and Thomas Moscibroda. 2020. Protean: VM Allocation Service at Scale. In *Symposium on Operating Systems Design and Implementation*.

[8] Huaicheng Li, Daniel S. Berger, Lisa Hsu, Daniel Ernst, Pantea Zardoshti, Stanko Novakovic, Monish Shah, Samir Rajadnya, Scott Lee, Ishwar Agarwal, Mark D. Hill, Marcus Fontoura, and Ricardo Bianchini. 2023. Pond: CXL-Based Memory Pooling Systems for Cloud Platforms. In *International Conference on Architectural Support for Programming Languages and Operating Systems*.

[9] David Lo, Liqun Cheng, Rama Govindaraju, Luiz André Barroso, and Christos Kozyrakis. 2014. Towards Energy Proportionality for Large-Scale Latency-Critical Workloads. In *International Symposium on Computer Architecture*.

[10] Jialun Lyu, Marisa You, Celine Irvene, Mark Jung, Tyler Narmore, Jacob Shapiro, Luke Marshall, Savyasachi Samal, Ioannis Manousakis, Lisa Hsu, Preetha Subbarayalu, Ashish Raniwala, Brijesh Warrier, Ricardo Bianchini, Bianca Schroeder, and Daniel S. Berger. 2023. Hyrax: Fail-in-Place Server Operation in Cloud Platforms. In *Symposium on Operating Systems Design and Implementation*.

[11] Sara McAllister, Benjamin Berg, Julian Tutuncu-Macias, Juncheng Yang, Sathya Gunasekar, Jimmy Lu, Daniel S Berger, Nathan Beckmann, and Gregory R Ganger. 2021. Kangaroo: Caching Billions of Tiny Objects on Flash. In *Symposium on Operating Systems Principles*.

[12] Anjaly Parayil, Jue Zhang, Xiaoting Qin, Íñigo Goiri, Lexiang Huang, Timothy Zhu, and Chetan Bansal. 2024. Towards Cloud Efficiency with Large-scale Workload Characterization. *arXiv preprint* (2024).

[13] Mohammad Shahrad, Jonathan Balkind, and David Wentzlaff. 2019. Architectural Implications of Function-as-a-Service Computing. In *International Symposium on Microarchitecture*.

[14] Debendra Das Sharma, Robert Blankenship, and Daniel S Berger. 2023. An Introduction to the Compute Express Link (CXL) Interconnect. *arXiv preprint arXiv:2306.11227* (2023).

[15] Jennifer Switzer, Gabriel Marcano, Ryan Kastner, and Pat Pannuto. 2023. Junkyard Computing: Repurposing Discarded Smartphones to Minimize Carbon. In *International Conference on Architectural Support for Programming Languages and Operating Systems*.

[16] Jaylen Wang, Daniel S. Berger, Fiodar Kazhamiaka, Celine Irvene, Chaojie Zhang, Esha Choukse, Kali Frost, Rodrigo Fonseca, Brijesh Warrier, Chetan Bansal, Jonathan Stern, Ricardo Bianchini, and Akshitha Sriraman. 2024. Designing Cloud Servers for Lower Carbon. In *International Symposium on Computer Architecture*.

[17] Jaylen Wang, Udit Gupta, and Akshitha Sriraman. 2023. Peeling Back the Carbon Curtain: Carbon Optimization Challenges in Cloud Computing. In *Workshop on Sustainable Computer Systems*.