Peeling Back the Carbon Curtain: Carbon Optimization Challenges in Cloud Computing

Jaylen Wang
jaylenw@andrew.cmu.edu
Carnegie Mellon University

Udit Gupta
ugupta@cornell.edu
Cornell University

Akshitha Sriraman
asrirama@andrew.cmu.edu
Carnegie Mellon University

ABSTRACT

The increasing carbon emissions from cloud computing requires new methods to reduce its environmental impact. We explore extending data center server lifetimes to reduce embodied carbon emissions (from hardware manufacturing), rather than operational (from running hardware). Our experiments are the first to analyze a data center application's end-to-end performance on different server generations, to reveal that older hardware can preserve performance in certain conditions (e.g., low load).

Our observations show the need for a carbon-aware data center scheduler that schedules on older hardware when suitable. However, quantifying such a scheduler's carbon savings is challenging today due to the lack of practical carbon measurement metrics/tools. We identify gaps in current methods for measuring operational and embodied carbon and call upon the broader systems research community to take action and conduct research that can pave the way for future carbon footprint analysis in systems.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**; • **Applied computing → Data centers**; • **Social and professional topics → Sustainability**.

KEYWORDS

Data centers, sustainability, microservices

ACM Reference Format:
Jaylen Wang, Udit Gupta, and Akshitha Sriraman. 2023. Peeling Back the Carbon Curtain: Carbon Optimization Challenges in Cloud Computing. In *2nd Workshop on Sustainable Computer Systems (HotCarbon '23), July 9, 2023, Boston, MA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3604930.3605718

HotCarbon '23, July 9, 2023, Boston, MA, USA

ACM ISBN 979-8-4007-0242-6/23/07.
https://doi.org/10.1145/3604930.3605718

1 INTRODUCTION

Scientists have made it clear that anthropogenic climate change is one of the greatest threats to both global health and the planet's ecosystem [1, 4]. To mitigate climate change, systems researchers have a particular role to play, as recent studies estimate that 2% of global emissions are due to Information and Communication Technology (ICT) [24]. Critically, projections show that ICT could constitute 20% of anthropogenic emissions by 2030, with much of the increase attributable to an increased demand for cloud computing [22].

Previous research has focused on reducing *operational* carbon emissions in data centers, which are the emissions produced to power the data center. This focus on reducing operational emissions builds upon the extensive work in making data centers more energy and resource efficient [8, 14, 32] and managing data center power consumption [35, 38].

Operational emissions, however, are only one part of the story. Prior work has shown that up to 50% of data center emissions are "*embodied*," which result from hardware manufacturing and transport [19]. The fraction of emissions that are embodied will likely grow further, with cloud providers pledging to invest more in renewables [15, 28, 29].

In a step towards reducing data center emissions, we explore understanding how older hardware can be reused in a cloud setting while preserving end-to-end service performance. We analyze the performance impact when running a web service composed of microservices [13] on older server generations. Our results reveal operating conditions where older hardware can maintain service performance compared to running entirely on newer servers. For example, we observe that older hardware can achieve comparable tail latencies as newer hardware under low load conditions.

Our results reveal the need to schedule services while considering tradeoffs between embodied and operational characteristics of hardware generations. Upon considering these tradeoffs, however, we identify many challenges in measuring carbon emission reductions achieved by carbon-aware systems optimizations, such as carbon-aware scheduling. Operational emissions' complexity arises from handling the variability of a data center's "carbon intensity," (i.e., emissions per unit energy consumed). With embodied emissions,

	Intel		AMD	
	Xeon E5-2660 v2	**Xeon E5-2660 v3**	**EPYC 7542**	**EPYC 7543**
Microarchitecture	Ivy Bridge (2012)	Haswell (2013)	Rome (2019)	Milan (2021)
Cores/Threads	10/20	10/20	32/64	32/64
Node	22 nm	22 nm	7 nm	7 nm
Base/Turbo (GHz)	2.2 / 3	2.6 / 3.3	2.9 / 3.4	2.8 / 3.7
LLC Cache Size	25 MB	25 MB	128 MB	256 MB
TDP (W)	95	105	225	225
RAM (DDR4)	256GB (1.6 GHz)	160GB (2.133 GHz)	256GB (3.2 GHz)	512GB (3.2 GHz)
Disk (SATA)	2 TB HDD	480 GB SSD	1.6 TB SSD	2 TB SSD
NIC	10Gb (PCIe v3)	10 Gb (PCIe v3)	25 Gb (PCIe v4.0)	25 Gb (PCIe v4.0)

Table 1: Attributes of the two generations (old on the left, new on the right) of Intel & AMD servers we study.

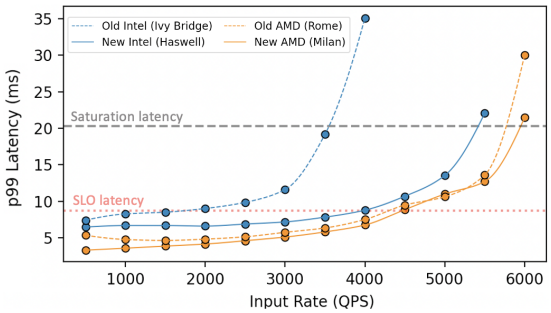

Figure 1: Tail (99^{th}%) latency across load conditions (in QPS) for older and newer Intel and AMD server SKUs. Older servers satisfy the SLO at low load conditions.

the main challenge is the lack of well-defined metrics or proxies, which makes it difficult to compare system designs.

Systems researchers must be able to analyze their designs' impact on carbon emissions. We take a step towards defining the challenges in enabling such analyses.

Our contributions are:

- Performing the first experiments to analyze a data center service's performance across server generations.
- Identifying conditions where older hardware does not increase tail latency, potentially allowing server lifetimes to be extended and reducing embodied carbon.
- Enumerating existing methods to measure carbon emissions and why they fall short in easily quantifying operational and embodied carbon costs of a systems decision, and suggesting ways to move forward.

In the rest of this paper, we discuss our initial results for extending data center server lifetimes (§2). We then describe the unique lessons and challenges we identify, and how they broadly apply, in handling operational (§3) and embodied (§4) carbon. Finally, we outline future areas of research that can pave the way for sustainable systems research (§5).

2 TOWARDS LONGER SERVER LIFETIMES

Previous research has proposed that extending server lifetimes could be effective to reduce data centers' carbon footprint by minimizing the embodied carbon caused by upgrading servers [18, 34, 37]. A key challenge in extending lifetimes is handling the added heterogeneity of having different server generations which have unique performance traits. Hence, to use servers for longer, we must determine the performance impact of running latency-sensitive web services (built using microservices) on different server generations.

Next, we explain how we study the performance impact of running web services on different server generations, our insights, and the challenges we face with quantifying the carbon savings produced by our proposed optimization.

A. Experimental setup

We compare a microservice-based application's performance on two AMD and Intel server generations. The servers of each type, i.e., Intel or AMD, are of the same SKU and only differ by their generation. The servers are located in CloudLab data centers [11]. Their characteristics are summarized in Table 1.

We study DeathStarBench's `Social Network` application [13], which allows users to create posts with text and images that are processed. This functionality is implemented as thirty core microservices that communicate with each other through Apache Thrift Remote Procedure Calls [31].

We set up our infrastructure in a way that allows us to evenly distribute the thirty microservices in `Social Network` across the number of nodes under test, pin each microservice to a single socket, and keep microservice placements constant. In a given server, we also constrain each microservice to the same amount of RAM. For all experiments, we use an open-loop load generator and sweep across low to high load conditions (measured in Queries Per Second) until a saturation throughput is reached, while recording tail latencies.

B. End-to-end service analysis across server generations

We first study how an end-to-end service behaves on different server generations. We distribute `Social Network`'s thirty microservices across fifteen nodes of the same server type (i.e., same SKU and generation).

Fig. 1 shows the resulting plot of latency against Queries Per Second (QPS). Latency above the gray dotted line cannot meaningfully be measured as the system is under saturation, where queuing delays grow unbounded. The red dotted line shows a performance-guided Service Level Objective (SLO) target taken as the latency achieved at 75% of the saturation load for the best performing server ("New AMD").

Our results show that at certain lower load conditions, both of the older servers can still achieve the performance SLO (often within a latency margin of 2–3 milliseconds). This result indicates that upgrading and using newer server

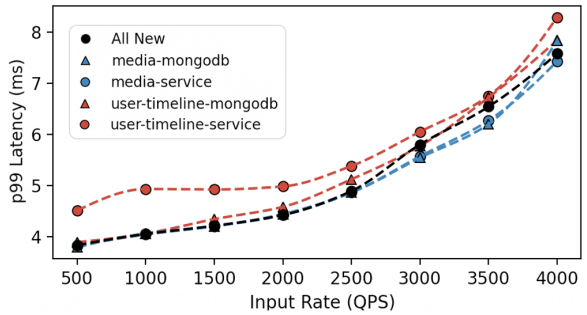

Figure 2: Impact on service tail (99^{th}%) latency upon placing a certain microservice on older hardware. Certain microservices are more (e.g., *media-service*) or less (e.g., *user-timeline-service*) tolerant to older hardware.

generations to serve lower load conditions is a carbon inefficiency, as older servers can still achieve SLO targets. Hence, it is worthwhile to explore if scheduling services on older hardware under lower loads can save embodied carbon emissions.

C. Microservice-based analysis across server generations

While Fig. 1 provides insights on the end-to-end service's performance tolerance when running on older hardware, it does not provide information on specific microservices' performance tolerance. Studying fine-grained microservice-level placement strategies that minimize performance impact might reveal further carbon optimization opportunities. To this end, we conduct a set of experiments where all microservices are initially placed on a set of fifteen newer servers under the same experimental setup as in §2. We then place one microservice on an older server, while keeping all other microservices on the newer nodes. We perform a QPS sweep and repeat for each of the thirty microservices. We report our results on only the two generations of AMD nodes for brevity.

In Fig. 2, we show the results for a set of representative microservices (those on the same call path are colored the same). Other microservices show similar trends; we omit them for brevity. "All New" shows latencies when running all microservices on newer AMD nodes, while other markers show latencies when that named microservice is the one scheduled on an older AMD node. We can see the effect on end-to-end service latency when a certain microservice is run on older hardware by comparing the latency to the "All New" latency.

We find that certain microservices, such as *user-timeline-service*, are less tolerant to being scheduled on older servers as they show consistently higher latencies than "All New". On the other hand, microservices such as *media-service* are more tolerant. A benefit of the microservice model is that individual components of a service can be optimized independently. While this benefit is often exploited for performance, the imbalance in microservices' tolerances to being placed on older servers suggests that individual microservices can be optimally scheduled to improve carbon efficiency.

D. Evaluating the solution

Driven by our study's insights, we aim to design carbon-efficient scheduling systems that can run performance-tolerant microservices on older hardware. To achieve this goal, we must turn qualitative intuitions about lifetime extension strategies into quantifiable metrics to evaluate carbon-efficiency. However, to quantify carbon-efficiency, we must answer two questions that each have their own challenges and potential solutions. The first question we address (§3) is: How can we monitor and quantify a data center system's (e.g., a carbon-efficient scheduling system's) operational carbon emissions? The second, and the one we have found more challenging to handle, is: How can we model and analyze a data center system's embodied carbon emissions in a way that is practical and accurate (§4)?

3 MEASURING OPERATIONAL CARBON

When identifying ways to quantify the differences in operational emissions for different server generations, we found useful existing approaches to enable this quantification as well as some deficiencies that we will detail.

A. How it's been done

Just as the Iron Law of processor performance uses three components to measure performance, we can quantify two components to measure an application's operational footprint over a time period: energy consumed and carbon intensity [18, 20]. Energy consumed can be measured in kilowatt-hours (kWh) or Joules/watt-seconds (Ws). The energy source's "carbon intensity" describes the carbon emissions produced by the energy source(s) used by the application, which is measured in grams or kilograms of carbon dioxide per unit of energy (e.g., g CO_2 per kWh). Measuring the first component of energy consumption is often relatively easy, given existing techniques/tools for data center- and hardware-level power monitoring and modeling [5, 21, 23, 27, 38].

The second component, carbon intensity, is more challenging to determine, as it depends on the time, location, and even data center operators' future renewable commitments. Prior work deals with these dependencies by making the carbon intensity a variable and then varying the intensities based on a system's operating conditions [2, 18]. Varying the carbon intensity provides insights into (1) how an optimization behaves in diverse data center scenarios (e.g., more or less renewables) and (2) how varying the carbon intensity changes the optimal decision to reduce emissions. Unless the carbon intensity remains constant over the system's lifetime,

treating it as a variable allows optimizations to evolve as the surrounding energy infrastructure inevitably changes.

B. What's in it for us?

To achieve our hardware lifetime extension goals, we must measure different server generations' operational emissions. We can measure our service's energy consumption via tools like RAPL [10], which record per-socket CPU and memory power measurements. We can even distribute microservices across sockets and use RAPL to determine per-microservice energy efficiency across server generations.

To deal with carbon intensity, we can use the stated variable method [2, 18], allowing our carbon-efficient scheduler to be deployed across long time scales and geographical regions. For example, when/where the carbon intensity is high, the scheduler should favor servers that optimize for energy-efficiency. In contrast, when/where carbon intensity is low, the scheduler should favor servers that are potentially less energy efficient but older, extending their possible lifetime.

C. Broader challenges with operational emissions

Operational carbon is easier to think about as the energy consumption component is directly related to resource efficiency, which is well-studied [9, 25, 35]. This direct relationship implies that energy and latency optimizations will result in operational reductions. However, systems researchers will still face challenges when handling operational emissions.

Handling variable carbon intensities. Often carbon intensity necessitates changing a system's optimization even if the workload and hardware is constant. This variability must be accounted for in two cases, to find an optimal design point that minimizes a system's net carbon footprint.

First, if the optimization operates in an environment where the carbon intensity varies over similar timescales as an application execution, it could complicate a system's decision-making. Often, temporal variability involves predicting how the carbon intensity will change in the future to schedule applications for future carbon intensity changes [2].

Second, if the application or system operates across geographical regions, then carbon intensity variations due to different grid sources must be considered, thereby affecting decisions on where to schedule applications and what optimizations are optimal in one region vs. another. If, however, the optimization does not face these two scenarios, then carbon intensity acts as a constant for energy consumption. Further, if the embodied carbon in such an optimization is constant (e.g., running an application on a single server configuration), then a holistic carbon optimization simply has to minimize the application's total energy consumption.

Exposing to cloud users. Since it is often possible to measure the two components of operational emissions at any period of time, we can expose operational emissions to users. Hence, large cloud providers use energy sourcing and application-level power monitoring information to expose operational emissions to users [16, 30]. However, the right way to apportion the energy consumption of virtualized resources such as memory and disk is still unclear. Even if users have this knowledge, identifying the right tools and APIs to give to users is an open area of research [33].

4 MODELING EMBODIED CARBON

Unlike a system's operational carbon, which can often be measured, it is unclear how or if the embodied carbon can be directly measured. We discuss current measurement methods and challenges that previous solutions have not yet solved.

A. How it's been done

Prior work on data center carbon optimizations create simple models for embodied carbon [7, 18]. Some models compare design choices by considering a lifetime, such as three to four years, in which all hardware components can reasonably remain functional. Such models allow us to account for the single purchase of hardware required to run over a given time period [7]. This accounting uses published energy consumption values for manufacturing these components. The embodied modeling then accounts for the different designs that require differing types and amounts of hardware to support a service.

ACT considers services that run on a single processor, such as a machine learning inference [18]. To account for embodied carbon, ACT uses a simple model in which a service's embodied emissions are modeled as $\frac{T}{L}E_{CF}$ where T is the service runtime, L is the estimated device lifetime, and E_{CF} is the estimated emissions from manufacturing the device. Similar to the previous approach [7], ACT assumes a constant lifetime for diverse devices (e.g., three years for servers and two years for mobile devices). Then, the embodied carbon footprint of the different devices are counted once for the lifetime while also weighing the service run time.

B. What's in it for us?

Since our carbon-aware scheduler must consider lifetime extensions, where a key carbon saving comes from minimizing server upgrades, and therefore manufacturing, the previous modeling approaches [7, 18] of considering a single lifetime period do not cover our system's timescales. This drawback is because prior research [7, 18] does not focus on lifetime extension, but on considering the carbon and performance tradeoffs between two devices.

Furthermore, as in the ACT model [18], there are inherent issues with including the latency in the embodied component when measuring carbon footprint. This latency discounting treats a device's lifetime as expendable and fungible, and therefore provisions the embodied emissions over its lifetime according to the time spent on a service.

One issue with the ACT model [18] is that it assumes that a device is running at 100% utilization throughout its lifetime. This assumption could be valid for a device that handles fixed-size workloads, but might not apply for typical data center services as these services rarely involve fixed-size operations. Another issue is that using average latency as a weight on the embodied emissions does not fit well for distributed web services. Modern web services are not always provisioned to achieve a certain average latency over their requests, but are more concerned with tail-latency effects. Hence, web systems are provisioned to provide acceptable latencies for worst-case loads. This assumption also does not account for cases where lifetime depends on the optimization itself and is hence not known a priori, as with our proposed carbon-efficient scheduling system.

For our scheduling system, existing embodied carbon modeling methods do not answer a key question: How can we measure the carbon emissions of a scheduling system that better accounts for server heterogeneity? Under certain operating conditions, such a scheduler will allow older generations to remain in a data center longer while achieving similar performance to a data center with the latest servers. To determine the carbon savings of such a scheduler, we require a baseline implementation of a data center scheduler. This baseline can then be compared against to determine carbon emissions avoided by using our carbon-aware scheduler. Possible baselines are schedulers that are unaware of performance heterogeneity and hence make naïve placements that can cause older hardware to be unused in cases where they might be useful, precipitating premature server upgrades.

C. Broader challenges with embodied emissions

Reasoning about the challenges that we must consider when measuring embodied carbon for our carbon-aware scheduler has helped us identify why measuring embodied emissions is challenging for any new system design optimization.

Embodied emissions are not continuous. First, unlike a service's operational footprint, it is unclear what a service's embodied emissions are over its typical runtime. While the two components of operational emissions are continuous functions, embodied emissions are an outcome of manufacturing, which is a result of discrete instances of procuring hardware. The timescales of typical service runtimes and of making system-level decisions, such as scaling up or scaling out, are typically not long enough (on the order of seconds to days) to capture general trends of server upgrade and procurement cycles (on the order of months to years). Hence, monitoring the embodied emissions of purchased servers in a data center would not be representative of changes in the embodied emissions during a service's runtime or a scheduler's policy decision.

	Apple Mac Pro	HPE ProLiant DL360	Dell R740
Assumed lifetime (years)	"3 or 4"	4	4
Assumed utilization	Undisclosed	30% load, 100% of time	100% load, 10% of time 50% load, 35% of time 10% load, 30% of time, Idle mode, 25% of time
Assumed location	Undisclosed	EU	US and EU
CPU(s) Used	1x M1	1x (Undisclosed)	2x Intel Xeon
SSD	256 GB	Undisclosed	8x 3.84 TB
IC Breakdown	No	No	Yes

Table 2: LCA differences across server vendors.

Embodied metrics are not well-defined. Unlike operational emissions which have two clear metrics to optimize, there is no consensus on what metrics to optimize for to minimize embodied emissions. One option is to track embodied footprint as a running average of emissions caused by continuous purchasing as a result of needing additional hardware, where the window that is averaged over is long enough to capture hardware refresh cycles. This rate method is useful when making "run-time" decisions, where steeper embodied emission rates can be detected. Detection could then trigger a mitigation strategy, such as changing a scheduling decision to shift to existing hardware and to shift the tradeoffs more towards saving embodied rather than performance or throughput. Another approach to minimize the total embodied carbon is to instead constrain the optimization time-period to one where discrete decisions can be modeled.

LCA standards are lacking. Ultimately, any approach that tracks embodied emissions relies on accurate estimations of emissions generated by manufacturing hardware. Of late, companies have released lifecycle assessments (LCAs) on server-class and commodity products [3, 6, 12]. However, as shown in Table 2, each company has its own standards, which makes mixing LCAs difficult. In addition, companies are not required to release LCAs for all products, making it difficult to find details for more bleeding-edge components that systems designers often use.

"Embodied proxies" are not well-defined. Standardized LCAs could take years to come to fruition, so we must focus on defining easily quantifiable metrics that can serve as proxies for embodied carbon. Such proxies could include (1) a server fleet's average age, (2) a service's compute, memory, and bandwidth requirements, and (3) a server fleet's mass. Each proxy would have its own tradeoffs in terms of ease of measurement, accuracy, and granularity.

For example, a server fleet's average age is simple to measure given the dates of when servers were manufactured. Age accounts for a device's original embodied emissions being amortized over its lifetime, allowing it to follow general trends of a system's embodied emissions. However, the age only operates at the granularity of a group of servers, and

would hence not account for embodied carbon differences of individual servers.

Ultimately, a good proxy should incentivize system designers and cloud companies to promote lowered carbon, even if the proxy itself does not exactly match carbon emissions.

5 FUTURE AREAS OF WORK

We discuss future directions for both carbon optimizations and improving system carbon measurement.

Reliability. Further research is needed to quantify how system reliability, across the stack, can lead to carbon emissions reductions. For example, increased reliability at the distributed systems level can lead to improved uptime, which can reduce overprovisioning for handling outages. Similarly, increased hardware reliability can further extend lifetimes and reduce embodied emissions. These examples reveal the need for techniques that can lead to drastic reliability improvements in systems.

Disaggregation/pooling. Recent research has shown that pooling can lead to quantifiable reductions in hardware [17, 26]. Future optimizations in this area are promising, as they reduce both operational (by minimizing resource utilization) and embodied (by reducing the required hardware resources to achieve good performance) emissions.

Characterization of performance trends. Further insights into performance trends are needed to understand how systems should adapt to be carbon efficient. For example, software optimizations like kernel bypass [39] could further emphasize hardware performance differences across generations. Such optimizations could make it more beneficial to exploit cases where older hardware outperforms newer ones.

Transparent embodied carbon in public clouds. More exposure of embodied carbon and incentives to reduce embodied emissions for public cloud users could lead to further carbon reductions. Such rewards could be in the form of (1) price incentives for using older generations of hardware and (2) cloud providers exposing helpful carbon proxies.

Other embodied concerns. There are other environmental concerns that are not captured by modeling carbon emissions. These concerns include raw material consumption and hardware waste [36]. Determining how these concerns should be factored into systems design is an open question.

The time is now. We believe that given prompt action, we can lay the necessary groundwork for sustainability research to come, to allow continuous growth in computational power and further global access to critical web services.

REFERENCES

[1] 2021. *Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change.*

[2] Bilge Acun, Benjamin Lee, Fiodar Kazhamiaka, Kiwan Maeng, Manoj Chakkaravarthy, Udit Gupta, David Brooks, and Carole-Jean Wu. 2023. Carbon Explorer: A Holistic Approach for Designing Carbon Aware Datacenters. *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (2023).

[3] Apple. 2019. Mac Pro Product Environmental Report. (Dec. 2019).

[4] Lukoye Atwoli, Abdullah H. Baqui, Thomas Benfield, Raffaella Bosurgi, Fiona Godlee, Stephen Hancocks, Richard Horton, Laurie Laybourn-Langton, Carlos Augusto Monteiro, Ian Norman, Kirsten Patrick, Nigel Praities, Marcel G.M. Olde Rikkert, Eric J. Rubin, Peush Sahni, Richard Smith, Nick Talley, Sue Turale, and Damián Vázquez. 2021. Call for Emergency Action to Limit Global Temperature Increases, Restore Biodiversity, and Protect Health. *New England Journal of Medicine* (Sept. 2021).

[5] D. Brooks, V. Tiwari, and M. Martonosi. 2000. Wattch: a framework for architectural-level power analysis and optimizations. In *Proceedings of 27th International Symposium on Computer Architecture.*

[6] Andreas Busa, Malcolm Hegeman, Jeff Vickers, Natalia Duque-Ciceri, and Constantin Herrmann. 2019. Life Cycle Assessment of Dell R740.

[7] Jichuan Chang, Justin Meza, Parthasarathy Ranganathan, Amip Shah, Rocky Shih, and Cullen Bash. 2012. Totally green: evaluating and designing servers for lifecycle environmental impact. *ACM SIGPLAN Notices* (March 2012).

[8] Gong Chen, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, and Feng Zhao. 2008. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation.*

[9] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles.*

[10] Howard David, Eugene Gorbatov, Ulf R. Hanebutte, Rahul Khanna, and Christian Le. 2010. RAPL: memory power estimation and capping. In *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design.*

[11] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. 2019. The design and operation of cloudlab. In *Proceedings of the 2019 USENIX Conference on Usenix Annual Technical Conference.*

[12] Hewlett Packard Enterprise. 2021. HPE product carbon footprint – HPE ProLiant DL360 Gen10 server data sheet.

[13] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, Kelvin Hu, Meghna Pancholi, Yuan He, Brett Clancy, Chris Colen, Fukang Wen, Catherine Leung, Siyuan Wang, Leon Zaruvinsky, Mateo Espinosa, Rick Lin, Zhongling Liu, Jake Padilla, and Christina Delimitrou. 2019. An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems. In *International Conference on Architectural Support for Programming Languages and Operating Systems.*

[14] Jiechao Gao, Haoyu Wang, and Haiying Shen. 2020. Smartly Handling Renewable Energy Instability in Supporting A Cloud Datacenter. In *IEEE International Parallel and Distributed Processing Symposium.*

[15] Google. 2020. Google Environmental Report 2020. (2020).

[16] Google. 2023. Carbon Footprint.

[17] Zhiyuan Guo, Yizhou Shan, Xuhao Luo, Yutong Huang, and Yiying Zhang. 2022. Clio: a hardware-software co-designed disaggregated

memory system. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*.

[18] Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S. Lee, David Brooks, and Carole-Jean Wu. 2022. ACT: designing sustainable computer systems with an architectural carbon modeling tool. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*.

[19] 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 *IEEE International Symposium on High-Performance Computer Architecture*.

[20] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning. (Jan. 2020).

[21] Majid Jalili, Ioannis Manousakis, Íñigo Goiri, Pulkit A Misra, Ashish Raniwala, Husam Alissa, Bharath Ramakrishnan, Phillip Tuma, Christian Belady, Marcus Fontoura, et al. 2021. Cost-Efficient Overclocking in Immersion-Cooled Datacenters. In *ACM/IEEE Annual International Symposium on Computer Architecture*.

[22] Nicola Jones. 2018. How to stop data centres from gobbling up the world's electricity. *Nature* (2018).

[23] Vijay Kandiah, Scott Peverelle, Mahmoud Khairy, Junrui Pan, Amogh Manjunath, Timothy G. Rogers, Tor M. Aamodt, and Nikos Hardavellas. 2021. AccelWattch: A Power Modeling Framework for Modern GPUs. In *IEEE/ACM International Symposium on Microarchitecture*.

[24] Bran Knowles. 2021. *ACM TechBrief: Computing and Climate Change*.

[25] Alok Kumbhare, Reza Azimi, Ioannis Manousakis, Anand Bonde, Felipe Vieira Frujeri, Nithish Mahalingam, Pulkit Misra, Seyyed Ahmad Javadi, Bianca Schroeder, Marcus Fontoura, and Ricardo Bianchini. 2021. Prediction-Based Power Oversubscription in Cloud Platforms. In *USENIX Annual Technical Conference*.

[26] 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 *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*.

[27] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. 2009. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *IEEE/ACM International Symposium on Microarchitecture*.

[28] Meta. 2021. *2021 Sustainability Report*. Technical Report.

[29] Microsoft. 2021. 2021 Environmental Sustainability Report. (2021).

[30] Microsoft. 2023. Calculating My Carbon Footprint - Microsoft Sustainability.

[31] Krzysztof Rakowski. 2015. *Learning Apache Thrift*. Packt Publishing Ltd.

[32] Junaid Shuja, Kashif Bilal, Sajjad A. Madani, Mazliza Othman, Rajiv Ranjan, Pavan Balaji, and Samee U. Khan. 2016. Survey of Techniques and Architectures for Designing Energy-Efficient Data Centers. *IEEE Systems Journal* (2016).

[33] Abel Souza, Noman Bashir, Jorge Murillo, Walid Hanafy, Qianlin Liang, David Irwin, and Prashant Shenoy. 2023. Ecovisor: A Virtual Energy System for Carbon-Efficient Applications. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*.

[34] Jennifer Switzer, Rob McGuinness, Pat Pannuto, George Porter, Aaron Schulman, and Barath Raghavan. 2021. *TerraWatt: Sustaining Sustainable Computing of Containers in Containers*.

[35] Rahul Urgaonkar, Ulas C. Kozat, Ken Igarashi, and Michael J. Neely. 2010. Dynamic resource allocation and power management in virtualized data centers. In *IEEE Network Operations and Management Symposium*.

[36] WHO. 2021. Children and digital dumpsites: e-waste exposure and child health.

[37] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga Behram, James Huang, Charles Bai, Michael Gschwind, Anurag Gupta, Myle Ott, Anastasia Melnikov, Salvatore Candido, David Brooks, Geeta Chauhan, Benjamin Lee, Hsien-Hsin S. Lee, Bugra Akyildiz, Maximilian Balandat, Joe Spisak, Ravi Jain, Mike Rabbat, and Kim Hazelwood. 2022. Sustainable AI: Environmental Implications, Challenges and Opportunities. In *arXiv*.

[38] Qiang Wu, Qingyuan Deng, Lakshmi Ganesh, Chang-Hong Hsu, Yun Jin, Sanjeev Kumar, Bin Li, Justin Meza, and Yee Jiun Song. 2016. Dynamo: Facebook's Data Center-Wide Power Management System. In *ACM/IEEE International Symposium on Computer Architecture*.

[39] Irene Zhang, Amanda Raybuck, Pratyush Patel, Kirk Olynyk, Jacob Nelson, Omar S. Navarro Leija, Ashlie Martinez, Jing Liu, Anna Kornfeld Simpson, Sujay Jayakar, Pedro Henrique Penna, Max Demoulin, Piali Choudhury, and Anirudh Badam. 2021. The Demikernel Datapath OS Architecture for Microsecond-scale Datacenter Systems. In *ACM SIGOPS Symposium on Operating Systems Principles*.