Challenges and Opportunities in Sustainable Serverless Computing

Prateek Sharma
Indiana University Bloomington
prateeks@iu.edu

Abstract

Serverless computing has rapidly emerged as a popular deployment model. However, its energy and carbon implications are unclear and require exploration. This paper takes a look at the fundamental distinguishing attributes of serverless functions, and shows how some of them make energy-efficiency challenging. The programming model and deployment requirements of serverless functions makes them terribly energy inefficient—consuming more than 15× energy compared to conventional web services. On the bright side, FaaS is still actively expanding, and there is also an opportunity for rethinking FaaS resource management and deployment models, and make carbon efficiency a primary consideration. We present a few such techniques: moving functions to energy-friendly locations in distributed edge clouds; machine learning based modeling and control; and function-level demand response that combines ideas from approximate computing. Together, these represent a possible path forward to making serverless cloud computing sustainable.

1 Introduction

Cloud computing is at the forefront of decarbonizing computing. Large scale cloud platforms, which host millions of different applications and consume more than 1% of global energy [24], have been the prime target for energy efficiency and renewable energy usage [10]. However, continually evolving cloud abstractions and usage patterns raise new challenges in energy and carbon-efficiency.

Serverless computing has rapidly emerged as a popular deployment model, where applications are partitioned into small, fine-grained, "functions", whose execution is managed by the cloud platform [28, 9, 3]. Application-code can be near-instantaneously deployed on the cloud with a true "pay for what you use" pricing model (millisecond granularity), and automatic elastic scaling to meet any workload fluctuations. A wide range of applications such as web and IoT services, ML inference, data analytics, and scientific computing, now use Functions as a Service (FaaS) offerings of cloud platforms such as Amazon Lambda, Azure and Google Functions, etc.

Given the sharp rise in its popularity, *what are the energy and carbon implications of FaaS?* While serverless computing has many benefits for *applications*, its programming model has imposed many resource management and optimization challenges for FaaS *providers* [28]. In the first part of this paper, we explore some of the key energy challenges that are a fundamental derivative of the FaaS programming and deployment models.

Our preliminary empirical investigation suggests that FaaS applications can be up to 15× more energy hungry than conventional web services. This energy and carbon (in)efficiency is unfortunately a fundamental attribute of serverless functions owing to their programming model and security isolation requirements. As FaaS usage continues its exponential growth, understanding and narrowing this energy gap will be vital for the carbon footprint of the overall computing ecosystem.

Our vision is to extend the notion of carbon and energy as a first class resource [8, 10] to serverless computing. This will require a precise understanding of where and how energy is consumed, so that functions can be "charged" for their carbon footprint, and for cloud providers to monitor and optimize their overall usage. However, we find that such accurate energy accounting and attribution will be especially challenging in FaaS owing to their distributed and resource-hungry *control planes* (such as OpenWhisk). Prior work on power management typically uses application or system level power and performance models, but these techniques are not directly applicable for serverless functions with distributed and highly variable resource (and hence energy) footprints.

All hope is not lost, however. Serverless functions can provide some unique opportunities to reduce cloud carbon footprint, because of their location independence and programming model. Many energy-first techniques such as workload migration and demand-response scaling, which

are challenging for conventional VMs and containers, can be significantly easier to develop and optimize for serverless functions that can be "run anywhere". FaaS can thus provide new energy knobs to cloud platforms for moving applications to carbon-friendly locations quickly and in a fine-grained manner, which will be especially beneficial for distributed edge clouds powered by intermittent renewable energy like solar and wind.

The FaaS programming model also allows for *function-level demand-response.* A function can have multiple implementations that trade off energy for output quality or performance. This can allow the cloud provider to run the appropriate function implementation based on energy/carbon availability and application preferences. Finally, functions are repeatedly invoked, and this permits data-driven and ML techniques such as transfer learning which can be used for coarse-grained and practical energy management.

Serverless computing is a rapidly developing area in both research and industry. Making it sustainable will be a significant challenge, but this is a timely opportunity to make energy and carbon efficiency a first-class design consideration through a combination of new mechanisms, policies, and user interfaces and incentives. The energy-first perspective of serverless computing will also help understand and prioritize the rapid advances in FaaS performance optimization. This also provides opportunities to look at long-standing challenges in many systems areas such as power management, approximate computing, transient computing, distributed edge clouds, cloud economics, and others.

2 Background

This section provides a brief background on serverless computing and introduces the key characteristics and components that will make energy-efficiency challenging. We also give an overview of the major challenges and solutions inherited from traditional power management, and explain why they may not be directly applicable.

2.1 Serverless Function Execution

Programming Model. Functions as a Service (FaaS) allows users to register small snippets of function code that get executed in response to some event or trigger (such as an HTTP request, message queue event, etc.). These functions must be stateless, and a new execution environment is created for every invocation (and can be destroyed after the function returns). The function code also contains all the necessary code and data dependencies (such as imported libraries and packages), and thus functions may spend significant time being *initialized* before the event-handling code can execute. Functions are executed inside virtual execution environments such as hardware virtual machines or OS containers. Function initialization, i.e., creating the execution environment and resolving code/data dependencies, can take 100s of milliseconds, and can significantly increase the latency of small functions [15].

Control Plane. All these steps are orchestrated by a control plane (such as OpenWhisk [1]) which handles all aspects of function execution. This control plane manages a cluster of servers to run functions on, and implements function scheduling and load-balancing, resource monitoring, function status tracking, storing function results, logging, etc. The control plane itself is highly distributed with many components such as API gateways, distributed message queues (such as Kafka), and databases. *Thus the resource and energy footprint of functions is spread out across function initialization and virtualization components, and the control plane itself.*

2.2 Power Management

Managing and optimizing power has been a long-standing challenge in computer systems, with many fundamental techniques, and has been increasingly receiving attention.

Measurement. Measuring the energy consumption of applications is non-trivial because of multi-processing, limited hardware support, and fairly attributing the consumption of shared resources (such as the OS) [20, 12]. Software power monitors [13, 14, 41] use application and system level power models that relates resource consumption (such as CPU time and frequency) to actual physical power [11]. Because of hardware heterogeneity and multi-processing, these models may need to be continuously monitored and updated. In FaaS, the set of functions running concurrently on a single server is large (100s) and highly dynamic since functions are typically very short lived. Thus, the classic software power accounting techniques may not be as effective.

Control. Controlling and restricting the energy consumption at the application and system level is also a major challenge. Hardware features like CPU frequency scaling and power states, and more modern features like RAPL (Running average power limit) are effective at system-level control [19]. Since capping power reduces performance of applications, doing so fairly and respecting quality-of-service [26, 40] again requires the use of power-vs-performance models, control theory, and machine learning [25]. Power capping can also be exposed through OS abstractions such as power containers [33] or sandboxes [18, 36]. The model-based per-application control and capping is again challenging for short-lived serverless functions, especially with a *distributed* control plane whose energy footprint is spread across servers.

Cloud-level. Managing energy at the data-center or cloud level requires additional policies and mechanisms. Workload shifting (either temporal or geographic) to run applications at energy-suitable locations is the key

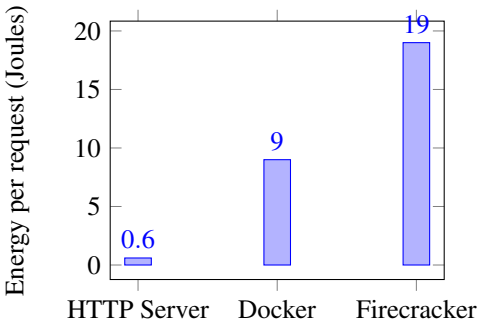

Figure 1: FaaS virtualization overheads can increase the energy consumption by more than 15x compared to conventional HTTP servers.

to match demand to variable renewable energy supply [30, 38, 5, 6, 39]. In general, this requires suspending and migrating applications, and many techniques from transient computing [34, 31, 35] are applicable. [8] makes the case for virtualizing the energy system and exposing detailed information to the applications in an exokernel-like approach. The FaaS control-plane is the ideal location for making intelligent policy decisions, since it has visibility into both application and infrastructure energy considerations.

3 Challenges in Energy-first FaaS

For energy and carbon to be first-class resources in cloud computing, they must be extended to the rapidly growing serverless/FaaS deployments and applications. That is, the primary consideration must be reducing, monitoring, and controlling FaaS energy usage. We now look at empirical results to highlight the severity of these challenges.

How energy efficient are functions? Function execution requires additional isolation and sandboxing, which is often provided through operating system or hardware virtualization. Using a container or a VM for every invocation adds a significant latency penalty of 100s of milliseconds. But how do these virtualization overheads impact energy efficiency?

Figure 1 shows the average energy consumption of handling a request using three techniques. The conventional web server (python's builtin httpserver), which does not incur any virtualization overheads, requires around 0.6 Joules to serve a simple static "`GET /`" request. The FaaS approach requires using a Docker container, and the overhead of creating the container increases the energy footprint by 15×. Specifically, we measure the average energy consumption of a `docker run` command which spawns a container, and prints the contents of the home directory (equivalent to the above HTTP server's output), and destroys the container. Finally, we investigate full hardware virtualization, by spawning a Firecracker lightweight VM [4], and find its energy consumption to be more than 30× that of the simple HTTP server.

The energy consumption was measured on a Lenovo x1 carbon laptop running Ubuntu 20.04 by reading `/sys/class/power_supply/BAT0/sys/energy_now` which provides fine-grained battery energy information in micro Joules. We use an open-loop load generator for 1 minute, subtract the idle power consumption, and report the average energy to handle the request during the interval. For statistical robustness, we further average energy consumption across three repeated 1-minute runs.

We only capture the energy consumption at the server level, and do not account for the energy footprint of the distributed control plane such as OpenWhisk which can also be significant. We also measure the *cold-start* scenario for Docker and Firecracker, since we do not reuse the execution environment. However, the functions themselves do not import any code or data dependencies, so this is still the "best case cold-start". Keeping function execution environments in memory is likely to reduce the energy footprint just like it does the performance [29, 15], but the extra memory (and hence energy) consumption may offset those benefits. Thus, these energy numbers are approximate and may not be representative of all real scenarios. Nevertheless, they provide some sense of the scale of challenges involved in making FaaS energy efficient.

Performance Variability. Controlling energy consumption of applications is a tradeoff with their performance. For FaaS, function performance and energy footprint is affected significantly by the control plane, which is highly distributed and comprises of load-balancers, databases, and virtualization controllers at each server (such as docker). Assigning energy footprints to functions will also require fairly attributing these control-plane overheads to functions. However, this may be non-trivial, due to the high variance in control-plane overheads.

Table 1 shows the time spent in just two OpenWhisk components: the central controller that serves as the ingestion-point and load balancer, and the docker daemon on the server. We see that there is a very long tail of the latency overheads of these components: the 99.9 percentile is significantly higher than the average, even though the functions are performing identical work each time. This high variance makes it challenging to accurately attribute control plane overheads to functions, and also for controlling energy consumption. For instance, it may be hard to attribute slower function performance due to power-capping, or just control-plane jitter. The high and variable control plane overheads thus add significantly to FaaS energy consumption. *The energy footprint of serverless functions is high, distributed across many components, and shows very high variance. This will make it challenging to use conventional power management techniques and require new approaches.*

Component	Avg Latency (ms)	99.9%ile (ms)
Controller	0.002	13
Server	12	190

Table 1: FaaS control plane (OpenWhisk) overheads can be high, distributed across the controller and servers, and have very high variance.

4 Opportunities in Sustainable Serverless Computing

Many research opportunities can be found by leveraging the unique programming model of serverless functions for improving their efficiency, and rethinking the way they are used by applications and executed by FaaS providers. We describe three broad solutions that leverage different characteristics of functions, which we believe have not received enough attention. Not only are the energy challenges of serverless computing surmountable, but our proposed solutions can also address long-standing challenges in broader sustainable computing and perhaps improve the overall energy/carbon efficiency of cloud platforms.

4.1 Energy and Carbon Based Scheduling

Functions are not tied to any specific servers or locations, and can potentially be "run anywhere", as long as the execution platform has access to the function's code dependencies and the container/VM "image". By decoupling computation from its location, serverless computing allows us to run functions at the most energy-suitable location. *Thus even though individual functions may not be energy efficient, they can be run in carbon-friendly locations to achieve better carbon efficiency.*

This *location independence* can be an extremely potent technique for sustainable computing, but is often challenging for other workloads. Since renewable energy (such as solar and wind) can be intermittent, the availability of servers powered by them is only *transient* [35]. Handling resource unavailability in such transient computing scenarios requires applications be "agile" and migrate to locations where resources/energy is available [34]. Migration is often a challenge for most applications—however serverless functions are short-lived and have small disk images, and are more location-agile than traditional VMs and containers. This is an example of a situation where serverless computing can greatly simplify cloud-level sustainability compared to traditional workloads.

Easy movement of functions opens up the possibility of energy-efficient distributed "routing", placement, scheduling, and load-balancing. These policies can be developed and applied at many levels, and are likely to yield interesting insights into system design in general, as we briefly describe below:

Global routing. We envision a geo-distributed FaaS control-plane that runs functions on the most energy-appropriate location/server. *Distributed edge clouds* can offer different energy/carbon tradeoffs depending on location, time, and resource and hardware availability. Functions can be run on the edge [17] to provide low latency. The decision of whether to run a function on the edge or cloud data center is non-trivial. Running functions on remote locations with ample renewable energy may be suitable for carbon-sensitive functions, if the extra network latency is tolerable. This will require new *global* energy-based grid and placement algorithms. The task will be made more challenging by the highly heterogeneous and bursty FaaS workloads [29].

Edge Clouds. The energy vs. performance tradeoff adds a new dimension to the discussion on future cloud architectures. While edge clouds may have performance and security/privacy advantages, their energy benefits need additional analysis. Most hyperscale cloud datacenters are extremely efficient and increasingly use renewable energy [2, 32]. However, edge locations may not have the economies of scale or sufficient statistical multiplexing to be powered by renewable energy alone. They may either only have intermittent and limited energy availability, in which case we need to be careful about which function to offload to the edge. Thus edge clouds may incur higher carbon footprint than the data-centers, if they are not co-designed with energy-first FaaS software.

Thus, thinking of energy and carbon efficiency can throw new light into the tradeoffs of various cloud and systems designs. Given the relatively nascent nature of both serverless and edge clouds, this seems to be the right time to investigate these issues.

Cluster-level load-balancing and scaling. At a cluster-level, we need to decide on which server to run a function. This placement/load-balancing is already challenging because it needs to consider locality and performance tradeoffs [16]. Energy considerations will make this more complex, due to lack of power-proportionality of servers. Thus, load-balancing needs to be tightly integrated with energy-aware horizontal scaling. It may be more beneficial to turn some servers off completely and run the remaining servers at higher load. The performance vs. energy tradeoffs in this scenario presents an exciting exploration opportunity: overcommitting some servers may mean imbalance server loads but perhaps result in better overall energy efficiency. Hardware heterogeneity trends will lead to clusters with multiple types of servers with different hardware configurations and architectures. These servers will have different power and performance tradeoffs, making energy-aware scheduling more challenging.

Server-level scheduling. Energy-aware scheduling of functions at a *server* level is also a rich exploration space. Keep-alive and function snapshots can reduce cold-start

overheads, but their extra memory (and hence energy) footprint can be significant, and thus policies should be cognizant of the total energy footprint. Capping individual server energy consumption can also be accomplished by delaying some function and absorb function workload bursts, which is a key characteristic of FaaS workloads [29]. Conventional power management techniques using control-based power/performance tradeoffs can be used—however, the large number of co-located functions and their heterogeneous nature makes this challenging.

Interfaces and Incentives. The energy vs. performance optimizations will ultimately depend on FaaS SLAs. Designing practical and effective "user interface" and energy-incentives for functions is likely to be a major challenge. A running average carbon limit can be set by users, or more explicit carbon-pricing mechanisms can be viable starting points. Energy-pricing for functions can also fix the current misaligned incentives of current FaaS pricing: since functions are charged by running time, cloud providers dont have any incentive to improve their performance. Energy pricing will incentive users to write more energy-efficient code, and cloud operators to improve per-function energy consumption.

4.2 Learning-based energy management

Functions have repeated invocations, often with nearly identical execution characteristics. This opens the door to using online statistical learning techniques that can incrementally learn from prior executions. The predictability of function invocations and execution has already been successfully used in various aspects of FaaS resource management such as keep-alive [29, 15]. Even the success of snapshot optimizations [37, 7, 27] is built on the assumption that execution characteristics of functions across invocations is nearly identical (modulo large changes in function input characteristics).

This predictability can be leveraged to develop data and AI-driven energy management. Instead of measuring or optimizing the energy footprint of a specific invocation, providers can apply policies over larger intervals of time. For instance, while energy accounting and attribution has many challenges as shown in the previous section, it may be much easier to estimate average energy consumption of a function over an hour/day. The repeated invocations also open the door to game-theoretic techniques for attribution such as Shapley values [12].

The power/performance models and utility curves of functions can also be incrementally obtained, without depending on expensive and impractical offline profiling, which is required for most other applications today. The FaaS programming model also gives providers enough visibility into the application to use statistic program analysis and transfer learning [42] techniques. For example, the energy footprint of a function could be predicted based on other similar functions, and a database of energy fingerprints could be collaboratively built. Some of the techniques from widely studied areas like NILM [23] (Non Intrusive Load Monitoring) can be adapted, since the problem of function energy attribution in analogous.

4.3 Function Demand Response

Demand-response is a fundamental technique in energy-aware systems where the energy consumers alter their consumption patterns (i.e., demand), in response to changes in energy supply [22]. The FaaS programming model provides a unique opportunity for implementing function-level demand response. Multiple *variants* of a function can be implemented with different energy footprints, and the provider can choose to execute "low-energy" variants under energy pressure.

Many kinds of applications are amenable to such demand response: a smaller and less accurate ML model may be chosen (common in vision processing tasks); the number of loop iterations could be reduced; or the output quality could be reduced (e.g., compression). The output-quality vs. energy tradeoff also provides a rich design and exploration space. This tradeoff is also fundamental to approximate computing [21], and its techniques can be adapted as a starting point. The tradeoff is exciting and is not generally available in traditional VM/container based deployments where the provider is restricted to crude controls like stopping the application outright or reduce parallel replicas.

Function-variants will be a powerful mechanism for enabling energy-agility. Designing *policies* using this mechanism also presents many challenges. Presumably, the FaaS middleware can determine if a low-energy variant should be executed. But this may require new pricing models and incentive schemes that allow users to concisely specify conditions and preferences for this "degraded" execution mode.

5 Conclusion

Serverless computing is a rapidly developing area in both research and industry. Making it sustainable will be a significant challenge, due to fundamental inefficiencies and control plane overheads. We show how energy and carbon efficiency can be made a first-class design consideration through a combination of new mechanisms, policies, and user interfaces and incentives. This is a timely opportunity to make energy a first-class resource in the rapidly evolving ecosystem and influence it, and will require a cross-community effort involving FaaS users and providers, operating systems, and hardware.

References

[1] Apache OpenWhisk: Open Source Serverless Cloud Platform. https://openwhisk.apache.org/, 2020.

[2] ACUN, B., LEE, B., MAENG, K., CHAKKARAVARTHY, M., GUPTA, U., BROOKS, D., AND WU, C.-J. A Holistic Approach for Designing Carbon Aware Datacenters. *arXiv:2201.10036 [cs, eess]* (Jan. 2022). arXiv: 2201.10036.

[3] ADZIC, G., AND CHATLEY, R. Serverless computing: economic and architectural impact. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (2017), pp. 884–889.

[4] AGACHE, A., BROOKER, M., IORDACHE, A., LIGUORI, A., NEUGEBAUER, R., PIWONKA, P., AND POPA, D.-M. Firecracker: Lightweight virtualization for serverless applications. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)* (2020), pp. 419–434.

[5] AGARWAL, A., SUN, J., NOGHABI, S., IYENGAR, S., BADAM, A., CHANDRA, R., SESHAN, S., AND KALYANARAMAN, S. Redesigning Data Centers for Renewable Energy. *HotNets* (2021), 8.

[6] ANDERSON, T., BELAY, A., CHOWDHURY, M., CIDON, A., AND ZHANG, I. Treehouse: A Case For Carbon-Aware Datacenter Software. *arXiv:2201.02120 [cs]* (Jan. 2022). arXiv: 2201.02120.

[7] AO, L., PORTER, G., AND VOELKER, G. M. Faasnap: Faas made fast using snapshot-based vms. In *Proceedings of the Seventeenth European Conference on Computer Systems* (2022), pp. 730–746.

[8] BASHIR, N., GUO, T., HAJIESMAILI, M., IRWIN, D., SHENOY, P., SITARAMAN, R., SOUZA, A., AND WIERMAN, A. Enabling Sustainable Clouds: The Case for Virtualizing the Energy System. In *Proceedings of the ACM Symposium on Cloud Computing* (Seattle WA USA, Nov. 2021), ACM, pp. 350–358.

[9] CASTRO, P., ISHAKIAN, V., MUTHUSAMY, V., AND SLOMINSKI, A. The rise of serverless computing. *Communications of the ACM 62*, 12 (2019), 44–54.

[10] CHIEN, A. A. Driving the cloud to true zero carbon. *Communications of the ACM 64*, 2 (Jan. 2021), 5–5.

[11] COLMANT, M., ROUVOY, R., KURPICZ, M., SOBE, A., FELBER, P., AND SEINTURIER, L. The next 700 CPU power models. *Journal of Systems and Software 144* (Oct. 2018), 382–396.

[12] DONG, M., LAN, T., AND ZHONG, L. Rethink energy accounting with cooperative game theory. In *Proceedings of the 20th annual international conference on Mobile computing and networking* (Maui Hawaii USA, Sept. 2014), ACM, pp. 531–542.

[13] FIENI, G., ROUVOY, R., AND SEINTURIER, L. SmartWatts: Self-Calibrating Software-Defined Power Meter for Containers. *arXiv:2001.02505 [cs]* (Jan. 2020). arXiv: 2001.02505.

[14] FIENI, G., ROUVOY, R., AND SEITURIER, L. SelfWatts: On-the-fly Selection of Performance Events to Optimize Software-defined Power Meters. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)* (Melbourne, Australia, May 2021), IEEE, pp. 324–333.

[15] FUERST, A., AND SHARMA, P. Faascache: Keeping serverless computing alive with greedy-dual caching. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (New York, NY, USA, 2021), ASPLOS 2021, Association for Computing Machinery, pp. 386–400.

[16] FUERST, A., AND SHARMA, P. Locality-aware Load-Balancing For Serverless Clusters. In *Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing* (New York, NY, USA, 2022), HPDC 2022, Association for Computing Machinery.

[17] GADEPALLI, P. K., PEACH, G., CHERKASOVA, L., AITKEN, R., AND PARMER, G. Challenges and Opportunities for Efficient Serverless Computing at the Edge. In *2019 38th Symposium on Reliable Distributed Systems (SRDS)* (Oct. 2019), pp. 261–2615. ISSN: 2575-8462.

[18] GERHORST, L., REIF, S., HERZOG, B., AND HNIG, T. Energy-Budgets: Integrating Physical Energy Measurement Devices into Systems Software. In *2020 X Brazilian Symposium on Computing Systems Engineering (SBESC)* (Nov. 2020), pp. 1–8. ISSN: 2324-7894.

[19] GULIANI, A., AND SWIFT, M. M. Per-Application Power Delivery. In *Proceedings of the Fourteenth EuroSys Conference 2019* (Dresden Germany, Mar. 2019), ACM, pp. 1–16.

[20] GUO, L., XU, T., XU, M., LIU, X., AND LIN, F. X. Power sandbox: power awareness redefined. In *Proceedings of the Thirteenth EuroSys Conference* (Porto Portugal, Apr. 2018), ACM, pp. 1–15.

[21] HOFFMANN, H. JouleGuard: energy guarantees for approximate applications. In *Proceedings of the 25th Symposium on Operating Systems Principles* (Monterey California, Oct. 2015), ACM, pp. 198–214.

[22] IRWIN, D., SHARMA, N., AND SHENOY, P. Towards continuous policy-driven demand response in data centers. In *Proceedings of the 2nd ACM SIGCOMM workshop on Green networking* (2011), pp. 19–24.

[23] MAKONIN, S., AND POPOWICH, F. Nonintrusive load monitoring (nilm) performance evaluation. *Energy Efficiency 8*, 4 (2015), 809–814.

[24] MASANET, E., SHEHABI, A., LEI, N., SMITH, S., AND KOOMEY, J. Recalibrating global data center energy-use estimates. *Science 367*, 6481 (2020), 984–986.

[25] MISHRA, N., IMES, C., LAFFERTY, J. D., AND HOFFMANN, H. CALOREE: Learning Control for Predictable Latency and Low Energy. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems* (Williamsburg VA USA, Mar. 2018), ACM, pp. 184–198.

[26] PATEL, T., AND TIWARI, D. PERQ: Fair and Efficient Power Management of Power-Constrained Large-Scale Computing Systems. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing* (Phoenix AZ USA, June 2019), ACM, pp. 171–182.

[27] SAXENA, D., JI, T., SINGHVI, A., KHALID, J., AND AKELLA, A. Memory deduplication for serverless computing with medes. In *Proceedings of the Seventeenth European Conference on Computer Systems* (2022), pp. 714–729.

[28] SCHLEIER-SMITH, J., SREEKANTI, V., KHANDELWAL, A., CARREIRA, J., YADWADKAR, N. J., POPA, R. A., GONZALEZ, J. E., STOICA, I., AND PATTERSON, D. A. What serverless computing is and should become: The next phase of cloud computing. *Commun. ACM 64*, 5 (Apr. 2021), 7684.

[29] SHAHRAD, M., FONSECA, R., GOIRI, ., CHAUDHRY, G., BATUM, P., COOKE, J., LAUREANO, E., TRESNESS, C., RUSSINOVICH, M., AND BIANCHINI, R. Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider. *arXiv:2003.03423 [cs]* (June 2020). arXiv: 2003.03423.

[30] SHARMA, N., BARKER, S., IRWIN, D., AND SHENOY, P. Blink: managing server clusters on intermittent power. In *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems* (2011), pp. 185–198.

[31] SHARMA, P., LEE, S., GUO, T., IRWIN, D., AND SHENOY, P. Spotcheck: Designing a derivative iaas cloud on the spot market. In *Proceedings of the Tenth European Conference on Computer Systems (EuroSys)* (2015), ACM.

[32] SHARMA, P., PEGUS II, P., IRWIN, D., SHENOY, P., GOODHUE, J., AND CULBERT, J. Design and operational analysis of a green data center. *IEEE Internet Computing 21*, 4 (2017), 16–24.

[33] SHEN, K., SHRIRAMAN, A., DWARKADAS, S., ZHANG, X., AND CHEN, Z. Power containers: an OS facility for fine-grained power and energy management on multicore servers. *ASPLOS* (2013), 12.

[34] SINGH, R., IRWIN, D., SHENOY, P., AND RAMAKRISHNAN, K. Yank: Enabling Green Data Centers to Pull the Plug. In *NSDI* (April 2013).

[35] SINGH, R., SHARMA, P., IRWIN, D., SHENOY, P., AND RAMAKRISHNAN, K. Here Today, Gone Tomorrow: Exploiting Transient Servers in Data Centers. *IEEE Internet Computing 18*, 4 (July/August 2014).

[36] SNOWDON, D. C., LE SUEUR, E., PETTERS, S. M., AND HEISER, G. Koala: a platform for OS-level power management. In *Proceedings of the fourth ACM european conference on Computer systems - EuroSys '09* (Nuremberg, Germany, 2009), ACM Press, p. 289.

[37] USTIUGOV, D., PETROV, P., KOGIAS, M., BUGNION, E., AND GROT, B. Benchmarking, analysis, and optimization of serverless function snapshots. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (2021), pp. 559–572.

[38] WIESNER, P., BEHNKE, I., SCHEINERT, D., GONTARSKA, K., AND THAMSEN, L. Let's Wait Awhile: How Temporal Workload Shifting Can Reduce Carbon Emissions in the Cloud. *Proceedings of the 22nd International Middleware Conference* (Dec. 2021), 260–272. arXiv: 2110.13234.

[39] ZHANG, C., KUMBHARE, A. G., MANOUSAKIS, I., ZHANG, D., MISRA, P. A., ASSIS, R., WOOLCOCK, K., MAHALINGAM, N., WARRIER, B., GAUTHIER, D., KUNNATH, L., SOLOMON, S., MORALES, O., FONTOURA, M., AND BIANCHINI, R. Flex: High-Availability Datacenters With Zero Reserved Power. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)* (June 2021), pp. 319–332. ISSN: 2575-713X.

[40] ZHANG, H., AND HOFFMANN, H. Maximizing Performance Under a Power Cap: A Comparison of Hardware, Software, and Hybrid Techniques. *ASPLOS* (2016), 15.

[41] ZHANG, X., SHEN, Z., XIA, B., LIU, Z., AND LI, Y. Estimating Power Consumption of Containers and Virtual Machines in Data Centers. In *2020 IEEE International Conference on Cluster Computing (CLUSTER)* (Sept. 2020), pp. 288–293. ISSN: 2168-9253.

[42] ZHUANG, F., QI, Z., DUAN, K., XI, D., ZHU, Y., ZHU, H., XIONG, H., AND HE, Q. A comprehensive survey on transfer learning. *Proceedings of the IEEE 109*, 1 (2020), 43–76.