EDITOR: Schahram Dustdar, dustdar@dsg.tuwien.ac.at

DEPARTMENT: INTERNET OF THINGS, PEOPLE, AND PROCESSES

Toward Sustainable Serverless Computing

Panos Patros, *University of Waikato, 3240 Hamilton, Aotearoa New Zealand*
Josef Spillner, *Zurich University of Applied Sciences, 8400 Winterthur, Switzerland*
Alessandro V. Papadopoulos, *Mälardalen University, 722 20 Västerås, Sweden*
Blesson Varghese, *Queen's University Belfast, BT7 1NN Belfast, U.K.*
Omer Rana, *Cardiff University, CF10 3AT Cardiff, U.K.*
Schahram Dustdar, *TU Wien, 1040 Vienna, Austria*

Although serverless computing generally involves executing short-lived "functions," the increasing migration to this computing paradigm requires careful consideration of energy and power requirements. Serverless computing is also viewed as an economically-driven computational approach, often influenced by the cost of computation, as users are charged for per-subsecond use of computational resources rather than the coarse-grained charging that is common with virtual machines and containers. To ensure that the startup times of serverless functions do not discourage their use, resource providers need to keep these functions hot, often by passing in synthetic data. We describe the real *power consumption characteristics of serverless, based on execution traces reported in the literature, and describe potential strategies (some adopted from existing VM and container-based approaches) that can be used to reduce the energy overheads of serverless execution. Our analysis is, purposefully, biased toward the use of machine learning workloads because: 1) workloads are increasingly being used widely across different applications; and 2) functions that implement machine learning algorithms can range in complexity from long-running (deep learning) versus short-running (inference only), enabling us to consider serverless across a variety of possible execution behaviors. The general findings are easily translatable to other domains.*

People and organizations are increasingly coming to terms with the urgent need to reverse the deleterious effects of climate change. The 2015 International Paris Agreement on Climate Change[a] mandated a temperature rise well below 2 °C—ideally capped at 1.5 °C. The UN proposed 17 Sustainable Development Goals (SDGs), such as "SDG7: Affordable and Clean Energy," "SDG9: Industry, Innovation, and Infrastructure," and "SDG13: Climate Action."[b] As our society's needs for computational power—and as such energy—increase, the software and computer engineering industries also need to decisively respond by adopting and encouraging sustainable operational paradigms. Serverless computing, as a new cloud computing paradigm, must also be made sustainable. As many predict serverless to be the next evolution of cloud systems,[1] ensuring that power and energy efficiency of such systems is adequately managed remains a crucial challenge.

[a] https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement

[b] https://sdgs.un.org/goals

1089-7801 © 2021 IEEE
Digital Object Identifier 10.1109/MIC.2021.3093105
Date of current version 6 December 2021.

Serverless computing expands on state-of-the-art cloud computing by further abstracting away software operations (ops) and parts of the hardware–software stack. One could consider functions, the execution unit of serverless computing, as "lightweight" containers, invoked with a set of inputs and expected to produce a set of outputs, when triggered. A key value proposition for serverless computing is its cost model, based on dynamic memory and CPU usage (connected directly to function invocations and as such, resource utilization and thus power/energy). This is unlike the more traditional cloud computing approaches, which charge based on the reservation of computing resources.

Data centers, and as such cloud and serverless computing, do have a significant impact on the world's total energy and power requirements. Estimates range from 200 to 500 TWh, which corresponds to 1–2.5% of the world's total energy usage. Additionally, this number is likely to increase as the demand for cloud computing increases: The estimated number of machines in data centers increased from 11 M in 2006 to 18 M in 2020. However, estimates are that only around 50% of this terawatt-hour energy consumption is used for actual computation; the other half is used on idling servers.[2] Serverless computing has a key role to play in this; this 50% waste in idling could in theory be completely reclaimed by this novel paradigm. Leading cloud providers have acknowledged the need to introduce real consumption pricing, something that becomes feasible with serverless architectures despite measurable overheads due to decomposition.[3] Serverless also provides a strong value proposition to users, who can pay for short time frames (less than a second), compared to reserving resources for an hour or more.

Apart from computation and memory, another energy-intensive computing task is networking. A 2015 metastudy estimated the Internet transmission energy to be 0.06 kWh/GB.[4] The problem is exacerbated in the era of Internet-of-Things (IoT) devices explosive growth: CISCO predicted 5ZB of IoT-related data to be transmitted in 2022.[5] This amount of IoT traffic will require 60 TWh of energy in 2022, essentially on par with 12–30% of data centers' energy needs.

A solution is to move small functions near the data instead of moving zettabytes of data to the data center; however, this adds significant extra development and operational burdens. Serverless computing, unlocks an easy migration toward easy-to-manage edge computing. Crucially, experimental evaluation on an IoT-driven video-analytics application suggests a 50% reduction in emissions is achievable if edge servers are used and data transmission to the data center is used sparingly.[6]

Therefore, to assure sustainable development for the Information Technology sector, there is an urgent call to establish energy- and power-aware design and operational strategies for the novel paradigm of serverless computing. We posit that the call for sustainable serverless computing can be split into three directions.

1. Sustainability techniques need to be designed and developed at the serverless platform level, such as power capping, scheduling, consolidation, and switching off policies. Crucially, to provide more room for maneuvering to the serverless Platform operator, serverless end-users need to be given incentives to minimize noncritical (deadline constrained) requirements, which can result in provisioning for sustainable service level agreements (SLAs).
2. The efficacy of serverless sustainability is closely coupled to workload patterns. The data center as a whole should avoid peak power consumption on its grid, as this leads to the use of emissions-heavy fossil-fuel-driven backup generators. As the world increasingly relies on artificial intelligence (AI) and machine learning (ML), the workload patterns generated by such smart systems must be studied and should (potentially) make use of relaxed SLAs, for instance during the training phase.
3. Connecting the above two topics, how can we, indeed, know how successful a serverless sustainability technique is? Sustainability oriented serverless benchmarks are needed to assess the quality of the proposed techniques, and these benchmarks need to be designed with realistic contemporary workloads, such as AI/ML, in mind. Crucially, as computation needs to move closer to the data, monolithic AI applications need to be replaced with function-oriented microservice architectures such that they can fit on low-powered edge devices and serverless operators can leverage the various aforementioned sustainability techniques.

Figure 1 illustrates data sources that *feed* data streams into serverless functions. Functions are frequently invoked with stream chunks as input, receiving data across different types of communication channels. A single data source, e.g., in-built environments, Industry 4.0, and electric mobility, may utilize different types of communication infrastructure.

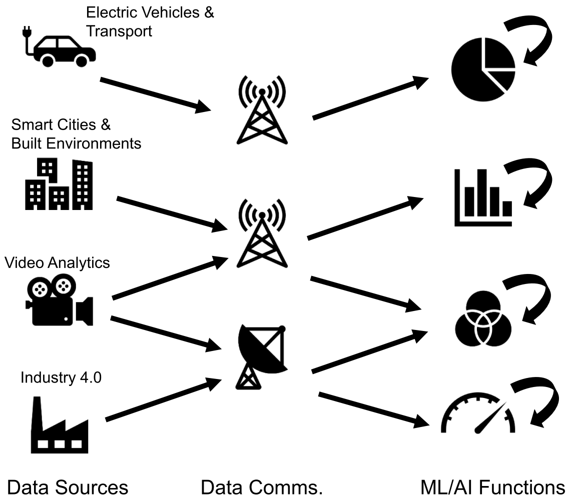

FIGURE 1. Serverless functions—responding to incoming data streams.

DESIGNING SUSTAINABLE SERVERLESS PLATFORMS

Various approaches can be used to limit the power consumption of serverless functions, ensuring more efficient use of energy of the associated infrastructure on which these functions are hosted. These techniques, which can be invoked transparently (from an end-user perspective) must be implemented by the serverless platform and can include: power capping of serverless deployments, use of scheduling strategies to make more effective use of the physical resources on which serverless functions are hosted, and mechanisms to minimize cold start times that can have significant power consumption requirements. Each of these approaches is described in this section, along with their benefit (and limitations).

Power Capping: This approach relies on limiting the power consumed by functions hosted within a specific container environment. Power capping techniques such as dynamic voltage and frequency scaling (DVFS) and running average power limit (RAPL) are hardware-based approaches that reduce CPU frequency and voltage to lower processor power consumption. However, this degrades the entire system performance and consequently the deployed application. Power cap violations are undesirable and need to be effectively managed, as the power benefit can be counterproductive—leading to applications running for longer time periods, which at times is the worst possible outcome from a sustainability perspective as it prevents shutting down under-utilized machines. More specifically, power cap violations occur when the total power consumed by a server exceeds a threshold defined by the server administrators.

Two power capping techniques are particularly relevant in the context of container-based functions: 1) DockerCap for Docker containers can make use of system power consumption obtained from a hardware power meter and RAPL. The CPU quota of all containers at different scheduling priority is reduced, thereby affecting the performance of all containers; and 2) DEEP-mon power monitoring can be used for Docker containers on the Kubernetes platform. This technique relies on RAPL and DVFS to manage power cap limits. It is demonstrated that RAPL affects the run-time performance of all containers on a server. RAPL enforces a power cap on the processor and DRAM by reducing the CPU frequency and thus degrading the overall system performance.

In the context of language-runtime-based functions, such as those supported by serverless platforms such as funcX, which runs on Python,[c] more fine-grain power capping can take place. Such algorithms could target specific subcomponents that might not need to run at full speed, such as resource-intensive dynamic memory management, aka., garbage collection.[7] Alternatively, the language runtime might be able to better characterize the resource requirements of its functions, enabling improved execution density via adaptive resource sharing among multitenant functions.[8]

The performance and execution behavior of a function is influenced by the power consumed by each function. Longer running functions can be terminated, for instance, if their power consumption exceeds the prespecified cap.

Network Power Saving: QUIC employs some of the basic mechanisms of TCP and TLS while keeping UDP as its underlying transport layer protocol. QUIC is, therefore, a combination of transport and security protocols by performing tasks including encryption, packet reordering, and retransmission. QUIC can be considered a user space, UDP-based (stream-oriented) protocol developed by Google—published by IETF in May 2021 as RFC9000. It is estimated that approx. 7% of Internet traffic employs QUIC. This protocol offers all the functionalities required to be considered a connection-oriented transport protocol, overcoming numerous problems faced by other connection-oriented protocols, such as TCP and SCTP. Specifically, the addressed problems are reducing the

[c] https://funcx.org/

connection setup overhead, supporting multiplexing, removing the head-offline blocking, supporting connection migration, and eliminating TCP half-open connections.

QUIC executes a cryptographic handshake that reduces connection establishment overhead by employing known server credentials learned from past connections. In addition, QUIC reduces transport layer overhead by multiplexing several connections into a single-connection pipeline. Furthermore, as QUIC uses UDP, it does not maintain connection status information in the transport layer. This protocol also eradicates the head-of-line blocking delays by applying a lightweight data-structure abstraction called *streams*. Due to its lightweight nature and support for data encryption, it is viewed as an important enabler for serverless functions. Using reduced overheads, the power consumption of QUIC is also reduced compared to other equivalent network protocols used for serverless deployment. The QUIC protocol can be also be used to preserve energy resources, especially between sleep and awake states that are often used by IoT devices. Maintaining a TCP connection requires use of keep-alive packets, which can consume energy and bandwidth. Understanding how this can undertake more efficiently is also an important approach to reduce energy use.[9]

Hotspot and Coldspot Migration: A common approach to reducing power consumption is the dynamic consolidation of virtual machines and containers on a smaller number of physical machines (PMs). This is based on the observation that PMs run at 10–50% of their maximum CPU usage and the majority of PMs are idle while still consuming about 70% of their peak power. This process involves migrating workload to enhance resource usage and minimize the use of machines that are underutilized within a data center—often turning these PMs OFF so that they do not consume power. Migration is expected to be transparent and beneficial when a physical server is highly overloaded (creating a hotspot) or underloaded (creating a coldspot). However, consolidation policies reduce energy consumption significantly but live VM migration results in increased violations of SLAs.

Many of these techniques, however, suffer from issues of instability and fluctuation—as migration of workload is often based on an instantaneous (or time-window-based) workload analysis. Only recently, time-series (machine-learning-based) forecasting techniques that take account of multiple criteria for estimating workloads are being used. Understanding where *cold spots* are likely to happen is as important as identifying the location of over utilized resources within a data center. A key challenge that differentiates this challenge within a serverless environment is the overhead of migrating workloads compared to: 1) the function execution time; and 2) the migration time and associated startup time of the function at the new location. Both of these aspects limit the benefit of migration for short-running functions—compared to longer running VMs or containers.

Power-OFF Strategies: As mentioned, traditional approaches in data center consolidation have focused on migrating long-running virtual machine instances to eventually power down idle hosts. More recently, these approaches have been suggested for reapplication in cloud-to-fog continuums.[10] In our view, such continuums will emerge everywhere due to the proliferation of sensing, and it would be short-sighted to assume conventional virtual machines as an execution technology. Instead, with a serverless computing approach, there are several advantages to simplify management and increase efficacy. First, short-running code can be left alone, and hosts can be switched OFF or suspended when none or even few instances remain. This greatly increases flexibility to decide when a switch-OFF shall occur. Second, the inherent event-driven nature of function invocation allows coupling with dynamic resumption such as Wake-on-LAN, in particular with fast-resuming and fast-booting technologies such as Coreboot[11] in conjunction with delay-tolerant function invocations. This way, hardware sensors along with virtualized fog nodes can be connected to as if they were permanently running, and yet they can power OFF in between. This programming simplicity resonates with the serverless computing mindset that infrastructural concerns are abstracted and largely hidden from application engineers.

Wake-on-LAN concepts have already reached beyond LANs and are commonly used in Internet-wide device management, including with custom protocols such as Apple's Bonjour Sleep Proxy (Multicast DNS, RFC 6762). For messaging-based triggers, protocol wrapping will allow a device to be booted or resumed before answering a request. For time-based triggers, an external time source needs to be added. Figure 2 shows the sequence of events, including eventual suspend and resume actions by the device or virtualized runtimes, based on rules or machine-learned patterns.

According to our early work experiments on event-driven power switching of a FaaS platform triggered by occasional IoT events, this approach added on average 0.95s execution time per request, within the delay tolerance to most batch jobs. In return, the platform could be suspended for 73% of the time, leading

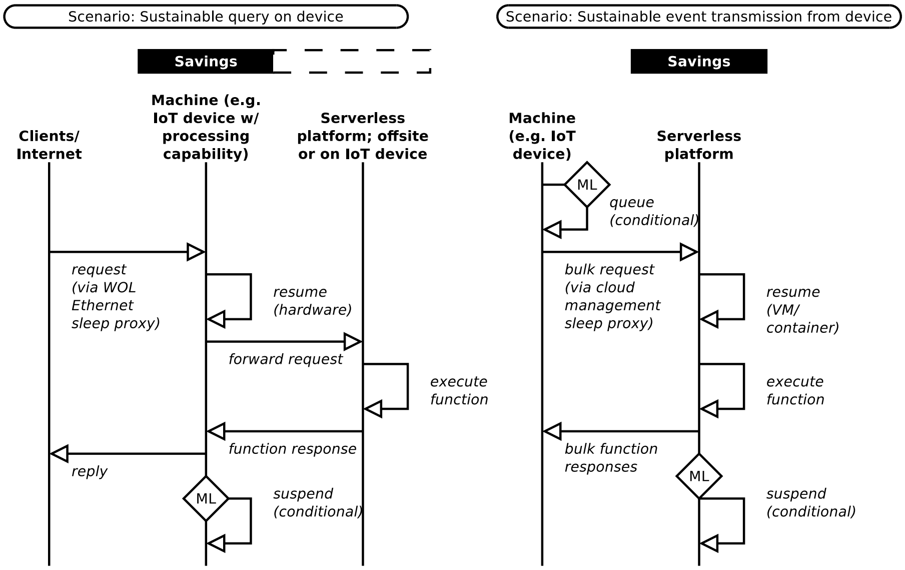
Scenario: Sustainable query on device
Savings
Clients/ Internet
Machine (e.g. IoT device w/ processing capability)
Serverless platform; offsite or on IoT device
request (via WOL Ethernet sleep proxy)
resume (hardware)
forward request
execute function
function response
reply
ML
suspend (conditional)
Scenario: Sustainable event transmission from device
Savings
Machine (e.g. IoT device)
Serverless platform
ML
queue (conditional)
bulk request (via cloud management sleep proxy)
resume (VM/ container)
execute function
bulk function responses
ML
suspend (conditional)

FIGURE 2. Sustainability approaches for Internet-wide control of device states and virtualized runtime lifecycle based on serverless event processing.

to great savings in power consumption. Figure 3 summarizes the suspend/resume pattern over a window of 6 min. The research challenge is, then, to learn and predict messaging patterns to optimize the suspend/resume or switch-OFF/switch-ON actions.

EFFECT OF WORKLOAD PATTERNS

We consider machine learning workloads consisting of deep neural networks (DNN), which comprise a sequence of layers and is a general term that covers all neural networks with multiple hidden layers (that is multiple layers between the input and output layers).

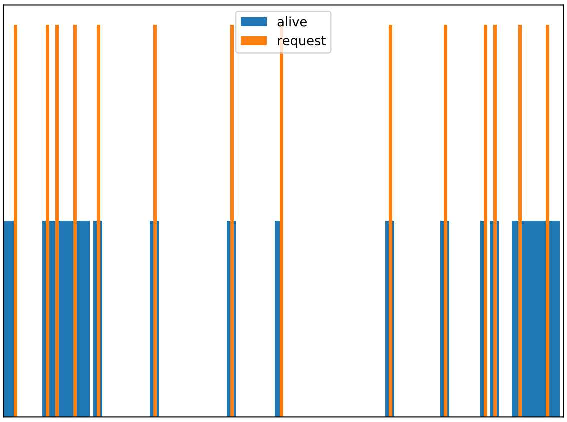

FIGURE 3. Event-driven suspend/resume patterns leading to power consumption savings.

Connectivity between the layers and propagation of an error function differentiates the different types of DNN architectures. Overall, a DNN may include: 1) fully connected layers, where each node in the network is connected to nodes at layer+1 and layer-1. A DNN may also include nodes that are not fully connected, or where connections may skip layers; 2) convolution layers convolve the input to produce feature maps of inputs to learn features. This is generally undertaken by identifying X-by-Y windows that are moved over a stride of Z. A convolution filter is chosen to identify key properties observed within the input dataset; 3) pooling layers apply a predefined function (maximum or average) to downsample the input; 4) an activation layer applies nonlinear functions and the most commonly used is the rectified linear unit (ReLu); and 5) a Softmax layer is generally used for classification to generate a probability distribution over the possible classes. The complexity of the DNN model is dependent on the number of nodes, the interconnectivity structure, and the choice of hyperparameters (such as X, Y, Z for convolution layers and learning rate).

Two different ML deployment scenarios can be considered: 1) workloads that are distributed and 2) workloads that are *miniaturization*. Traditionally, a DNN is trained at a data center and, then, deployed as a monolithic application on other resources where they need to be trained. More recently, it has been demonstrated that DNNs can be partitioned and deployed across different tiers of resources spanning the cloud, edge, and user devices to preserve privacy and achieve performance and energy efficiency in distributed systems.[12,13,14] In this scenario, the layers of a partitioned DNN can be mapped onto serverless functions that are invoked on-demand for inference on both resource-abundant (data center) and resource-constrained (edge servers or user devices) tiers. Such an approach can meet the power cap requirements on different resources. Since training is typically a long-running task, traditional mechanisms such as containers or VMs may be suited for deployment.

In the second scenario, machine learning workloads that need to fit on resource-constrained resources that are located outside conventional data centers closer to where data are generated are considered. The energy consumed by both the networking and compute infrastructure can be reduced. During inference, the energy consumption of the networking infrastructure is naturally conserved if limited data are transferred to geographically distant servers and can be processed at the edge of the network (up to a 50% reduction in the carbon footprint when processing data at the edge has been demonstrated[15]). In addition, there are opportunities to reduce energy consumption on a compute resource.

Consider the example of a real-time video analytics application, such as identifying objects on different frames of a video stream. A different DNN model from a portfolio of models can be employed for each frame to maximize the accuracy of detection.[16] This is achieved by leveraging the metacharacteristics of each video frame, such as the size of the object and the speed of movement of the object. Certain DNN models are more accurate when detecting fast moving objects but may have higher power requirements. Conversely, low-power models may deliver sufficient accuracy for slow-moving objects. Since the models contained in the portfolio have different power requirements, serverless computing can leverage the accuracy and power tradeoff to deliver a transprecision-based approach that maximizes accuracy.

QUALITY ASSESSMENT OF SERVERLESS SUSTAINABILITY

While the sustainability aspect of serverless computing has gained a lot of attention, the same cannot be said about approaches and methodologies for the quality assessment of serverless sustainability. Kistowski *et al.* define a benchmark as a "Standard tool for the competitive evaluation and comparison of competing systems or components according to specific characteristics, such as performance, dependability, or security."[17] The SPEC Cloud IaaS 2018 benchmark,[d] for example, focuses on four key benchmark metrics: 1) replicated application instances, 2) performance score, 3) relative scalability, and 4) mean instance provisioning time, none of which includes sustainable-related metrics. This is the typical focus of most of benchmarks in cloud computing,[18] and the ones developed for serverless computing.[19,20]

Including sustainability in benchmarks for serverless computing is challenging, yet extremely important, especially when considering the fast growth of AI applications deployed in the cloud. In a study conducted by Strubell *et al.*,[21] it has been found that *training* a single deep learning model can generate up to 284,000 kg of CO_2 emissions. This corresponds to the total lifetime carbon footprint of approximately five cars. But this is not a one-off cost, concluding with the training of the ML algorithm—that could be potentially mitigated using transfer learning techniques. Amazon estimates that 90% of production ML infrastructure costs are for *inference*, not training.[22] NVIDIA estimated that 80%–90% of the energy cost of neural networks deployed in data centers lie in inference processing.[e]

In addition, a benchmark should not just focus on the raw numbers of energy consumption, but rather on where the energy comes from. If an AI model were trained using electricity generated primarily from renewables, its carbon footprint would be correspondingly lower. For instance, Google Cloud Platform's power mix is more heavily weighted toward renewables than the AWS Platform (56% versus 17%, according to Strubell *et al.*[21]). Lacoste *et al.*[23] developed an ML CO_2 calculator[f] that practitioners can use to estimate the carbon footprints of their deployment based on the following factors: 1) hardware type, 2) hours used, 3) cloud provider, and 4) geographical region. The last factor can have a significant impact on carbon emission, as different locations may have different access to greener energy sources.

Most of these efforts focus on long-lived applications that may not fully exploit the potential of serverless computing. Researchers and practitioners can try to focus on how to optimize their deployments and executions of ML applications. However, more fundamental long-term solutions are needed to automate and optimize the sustainability of ML applications. This could be achieved through the main features of serverless computing and the development of suitable

[d] https://www.spec.org/cloud_iaas2018/
[e] https://www.forbes.com/sites/moorinsights/2019/05/09/google-cloud-doubles-down-on-nvidia-gpus-for-inference/
[f] https://mlco2.github.io/impact/

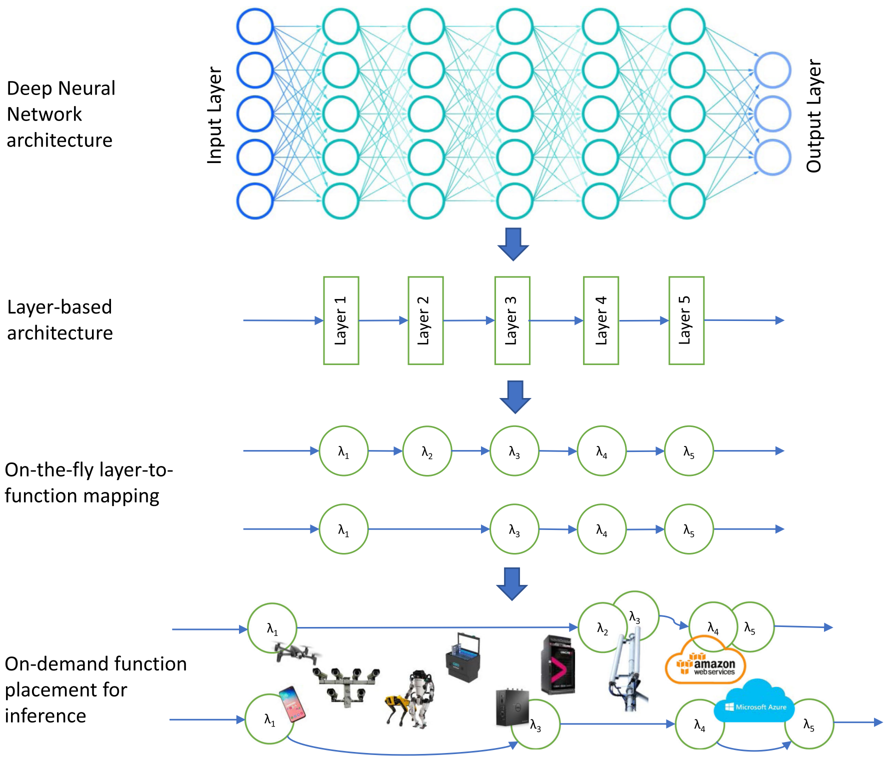

FIGURE 4. Serverless approach for the sustainable execution of deep AI inference.

management techniques and cost models that can promote greener computation.

In Figure 4, we display our proposed approach for enabling serverless computing for AI-intensive workloads. As major energy requirements for AI workloads are due to inference (i.e., during usage rather than training—where training is only needed occasionally or once), we also focus our attention on the actual operation of deep learning algorithms versus their training. A key observation is that DNNs do not need to run as monolithic structures; instead, we propose that each layer of a DNN be segmented into a function suitable for fine-grain deployment and scheduling—which can achieve improved computational density both on cloud and on edge servers. As such, consider for simplicity a multilayer DNN.[24] Each layer will have outputs $u \in R^m$, weights $W \in R^{m \times n}$, biases $b \in R^m$, activation function g, and connected with inputs $u \in R^n$ will execute the following function:

$$\lambda^{\mathrm{DNN}} := y = g(W \cdot u + b).$$

Thus, a DNN with depth k can be recursively split into independent functions that can be *stacked* as follows, considering that λ_0^{DNN} describes the raw data inputs to whole DNN:

$$\lambda_k^{\mathrm{DNN}} := \gamma = g(W \cdot \lambda_{k-1}^{\mathrm{DNN}} + b).$$

Consequently, from a serverless perspective, the instructions required to be transmitted to execute one of these λ^{DNN} functions reduces to the weights, biases, and type of activation function. From a cluster management perspective, the serverless provider can now make more informed decisions on where and when each λ^{DNN} instance should run, taking into consideration data locality to minimize network energy, renewable, and off-peak power availability to reduce the stress on the grid, depending on their sparsity, place them on machines with the appropriate level of hardware parallelism (e.g., CPU versus GPU), as well as existing utilization of computing resources to maximize utilization of power-on resources while keeping as many machines as possible powered OFF. Additionally, smart reusable algorithms could be created that easily combine existing structures to efficiently deploy novel DNN architectures by essentially leveraging this microservice-oriented design of DNNs.

Furthermore, on the actual device that executes one of these λ^{DNN} functions, an autonomic manager operated by the serverless platform can enable run-time-specific energy-aware optimizations. For example, consider multitenancy of DNN lambdas, i.e., the secure sharing of equal λ^{DNN} functions used by multiple tenants—the statelessness of these functions allows for this optimization—which can save energy costs by reducing unneeded context switches or thrashing the hardware caches. Additionally, if users are willing to sacrifice a bit of accuracy for improved energy efficiency, even slightly different λ^{DNN} could be shared, for example, by using the average weights of the multitenant users or one user accepting to use λ^{DNN} of another.

CONCLUSIONS

The serverless computing paradigm enables abstracting away hardware resource management and resource operations, which transfers the burden of energy innovation to the serverless platform provider. With an urgent call for worldwide sustainable development, serverless platforms must also be designed to be energy- and power-aware.

We highlight the need for sustainable serverless computing, which we posit can be achieved via: 1) serverless platform design and infrastructure, 2) improved characterization of novel IoT- and AI-driven workloads, which are bound to dominate the world's computing needs, paired with smarter decision-making at the application-design level, and 3) automated methodologies that assess the sustainability efficacy of such power and energy-aware methods.

Finally, people, developers and end-users must also contribute to this effort of sustainable serverless computing! For instance, a user might need to consider turning ON the "eco-mode" for their functions, relaxing the requirements just enough so that the serverless provider has enough time to schedule them during an off-peak time or can keep that extra server in reserve turned OFF. "Human brains can do amazing things with little power consumption. The bigger question is how can we build such machines."[25]

ACKNOWLEDGMENTS

Note: This contribution is based on discussions initiated at a Daghstuhl event on "Serverless Computing" (16–21 May 2021).

Details of the event can be found at: *https://www.dagstuhl.de/en/program/calendar/semhp/?semnr=21201*

REFERENCES

1. J. Schleier-Smith *et al.*, "What serverless computing is and should become: The next phase of cloud computing," *Commun. ACM*, vol. 64, no. 5, pp. 76–84, 2021.
2. D. Mytton, "How much energy do data centers use?" Hypertext document, 2020. [Online]. Available: https://davidmytton.blog/howmuch-energy-do-data-centers-use/
3. A. Poth, N. Schubert, and A. Riel, "Sustainability efficiency challenges of modern IT architectures—A quality model for serverless energy footprint," in *Systems, Software and Services Process Improvement*, M. Yilmaz, J. Niemann, P. Clarke, and R. Messnarz, Eds. Cham, Switzerland: Springer, 2020, pp. 289–301.
4. J. Aslan, K. Mayers, J. G. Koomey, and C. France, "Electricity intensity of internet data transmission: Untangling the estimates," *J. Ind. Ecol.*, vol. 22, no 4, pp. 785–798, 2018.
5. Cisco, Cisco annual internet report (2018–2023), San Jose, CA, USA, White paper, 2020.
6. B. Ramprasad, A. da Silva Veith, M. Gabel, and E. de Lara, "Sustainable computing on the edge: A system dynamics perspective," in *Proc. 22nd Int. Workshop Mobile Comput. Syst. Appl.*, 2021, pp. 64–70.
7. P. Patros, K. B Kent, and M. Dawson, "Mitigating garbage collection interference on containerized clouds," in *Proc. IEEE 12th Int. Conf. Self-Adaptive Self-Organizing Syst.*, 2018, pp. 168–173.
8. V. Podolskiy, M. Mayo, A. Koay, M. Gerndt, and P. Patros, "Maintaining SLOs of cloud-native applications via selfadaptive resource sharing," in *Proc. IEEE 13th Int. Conf. Self-Adaptive Self-Organizing Syst.*, 2019, pp. 72–81.
9. P. Kumar and B. Dezfouli, "Implementation and analysis of QUIC for MQTT," *Computer Netw.*, vol. 150, pp. 28–45, 2019.
10. O. Osanaiye, S. Chen, Z. Yan, R. Lu, K. K. R. Choo, and M. Dlodlo, "From cloud to fog computing: A review and a conceptual live VM migration framework," *IEEE Access*, vol. 5, pp. 8284–8300, 2017.
11. J. Sun, M. Jones, S. Reinauer, and V. Zimmer, "Building coreboot with Intel FSP," in *Embedded Firmware Solutions*. Berkeley, CA, USA: Apress, 2015, pp. 55–95.
12. L. Lockhart, P. Harvey, P. Imai, P. Willis, and B. Varghese, "Scission: Performance-driven and context-aware cloud-edge distribution of deep neural networks," in *Proc. IEEE/ACM 13th Int. Conf. Utility Cloud Comput.*, 2020, pp. 257–268.
13. H. Ahn, M. Lee, C.-Ho Hong, and B. Varghese, "Scissionlite: Accelerating distributed deep neural networks using transfer layer," 2021, *arXiv:2105.02019*.
14. Y. Kang *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proc. 22nd Int. Conf. Architectural Support Program. Lang. Oper. Syst.*, 2017, pp. 615–629.

15. B. Ramprasad, A. da Silva Veith, M. Gabel, and E. de Lara, "Sustainable computing on the edge: A system dynamics perspective," in *Proc. 22nd Int. Workshop Mobile Comput. Syst. Appl.*, 2021, pp. 64–70.
16. J. K. Lee, B. Varghese, R. Woods, and H. Vandierendonck, "TOD: Transprecise object detection to maximise real-time accuracy on the edge," in *Proc. 5th IEEE Int. Conf. Fog Edge Comput.*, 2021.
17. J. von Kistowski, J. A. Arnold, K. Huppler, K.-D. Lange, J. L. Henning, and P. Cao, "How to build a benchmark," in *Proc. 6th ACM/SPEC Int. Conf. Perform. Eng.*, 2015, pp. 333–336.
18. A. V. Papadopoulos *et al.*, "Methodological principles for reproducible performance evaluation in cloud computing," *IEEE Trans. Softw. Eng.*, to be published.
19. E. van Eyk, J. Scheuner, S. Eismann, C. L. Abad, and A. Iosup, "Microbenchmarks: The SPEC-RG vision for a comprehensive serverless benchmark," in *Proc. Companion ACM/SPEC Int. Conf. Perform. Eng.*, 2020, pp. 26–31.
20. M. Copik, G. Kwasniewski, M. Besta, M. Podstawski, and T. Hoefler, "SeBS: A serverless benchmark suite for function-as-a-service computing," 2020, *arXiv:2012.14132*.
21. E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for modern deep learning research," *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 9, pp. 13693–13696, Apr. 2020.
22. A. Jassy, Amazon AWS ReInvent keynote, 2018. [Online]. Available: https://www.youtube.com/watch?v=ZOIkOnW640A
23. A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, Quantifying the carbon emissions of machine learning," 2019, *arXiv:1910.09700*.
24. A. Burkov, *The Hundred-Page Machine Learning Book*, volume 1. Québec, QC, Canada: Andriy Burkov, 2019.
25. K. Hao, "Training a single AI model can emit as much carbon as five cars in their lifetimes," MIT Technol. Rev., 2019.

PANOS PATROS is currently a Senior Lecturer with the Department of Software Engineering, University of Waikato, Aotearoa, New Zealand. He leads the Cloud and Adaptive Systems (Ohu Rangahau Kapua Aunoa) ORKA Lab and is a member of the Ahuora Smart Energy Systems Centre. He received the Ph.D. degree in computer science from the University of New Brunswick, Fredericton, NB, Canada, on multitenancy, performance, and modeling of cloud systems. Contact him at panos.patros@waikato.ac.nz.

JOSEF SPILLNER is currently an Associate Professor with the Zurich University of Applied Sciences, Winterthur, Switzerland, currently investigating distributed application computing paradigms and nation-scale system designs. He is a Senior Member of IEEE. Contact him at josef.spillner@zhaw.ch.

ALESSANDRO V. PAPADOPOULOS is currently an Associate Professor in Computer Science with Mälardalen University, Västerås, Sweden. His research interests include real-time and embedded systems, distributed systems, self-adaptive systems, and control theory. He received the Ph.D. degree in information technology (systems and control) from Politecnico di Milano, Milan, Italy. He is a Senior Member of IEEE. Contact him at alessandro.papadopoulos@mdh.se.

BLESSON VARGHESE is currently an Associate Professor in Computer Science with Queen's University Belfast, Belfast, U.K. His current research interests include distributed systems and machine learning spanning across the cloud and edge. He received the Ph.D. degree in computer science from the University of Reading, Reading, U.K. Contact him at b.varghese@qub.ac.uk.

OMER RANA is currently a Professor of Performance Engineering with Cardiff University, Cardiff, U.K., with research interests in distributed systems (cloud/edge and IoT) and data analytics (machine learning). He received the Ph.D. degree in neural computing and parallel architectures from Imperial College, London, U.K. Contact him at ranaof@cardiff.ac.uk.

SCHAHRAM DUSTDAR is currently a Full Professor of Computer Science (Informatics) with a focus on Internet Technologies heading the Distributed Systems Group, TU Wien, Vienna, Austria. He is a member of the Academia Europaea: The Academy of Europe. He is a Fellow of the IEEE. He is the corresponding author of this article. Contact him at dustdar@dsg.tuwien.ac.at.