Towards Sustainable Serverless Computing

Panos Patros[1], Josef Spillner[2], Alessandro Papadopoulos[3], Blesson Varghese[4], Omer Rana[5], and Schahram Dustdar[6]

[1] University of Waikato, Aotearoa New Zealand
[2] Zurich University of Applied Sciences, Switzerland
[3] Mälardalen University, Sweden
[4] Queen's University Belfast, UK
[5] Cardiff University, UK
[6] TU Wien, Austria

Abstract. 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-sub-second 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 towards the use of machine learning workloads as: (i) such workloads are increasingly being used widely across different applications; (ii) functions that implement machine learning algorithms can range in complexity from long-running (deep learning) vs. short-running (inference only), enabling us to consider serverless across a variety of possible execution behaviours. The general findings are also easily translatable to other domains.

Keywords: Serverless; Sustainability; Green Computing

1 Introduction & Context

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[7] 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"[8]. As our society's

[7] https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement
[8] https://sdgs.un.org/goals

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 [SSK+21], ensuring that power and energy efficiency of such systems is adequately managed remains a crucial challenge.

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 centres, and as such Cloud and Serverless Computing, do have a significant impact on the world's total energy and power requirements. Estimates range from 200TWh to 500TWh, 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 centres increased from 11M in 2006 to 18M in 2020. However, estimates are that only around 50% of this TWh energy consumption is used for actual computation; the other half is used on idling servers [Myt20]. 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 [PSR20]. 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 meta-study estimated the Internet transmission energy to be 0.06 kWh/GB [AMKF18]. The problem is exacerbated in the era of IoT devices explosive growth: CISCO predicted 5ZB of IoT-related data to be transmitted in 2022 [Cis20]. This amount of IoT traffic will require 60TWh of energy in 2022, essentially on par with 12%–30% of data centres.

A solution is to move small functions near the data instead of moving zettabytes of data to the data centre; however, this adds significant extra development and operational burdens. Serverless computing, however, unlocks an easy migration towards 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 centre is used sparingly [RdSVGdL21a].

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 manoeuvring to the Serverless Platform operator, serverless end-users need to be given incentives to minimise non-critical (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 centre 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 AI and 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 both these 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, receiving data across different types of communication channels. A single data source, e.g., in built environments, Industry 4.0 and electric mobility, may utilise different types of communication infrastructure.

2 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 minimise 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).

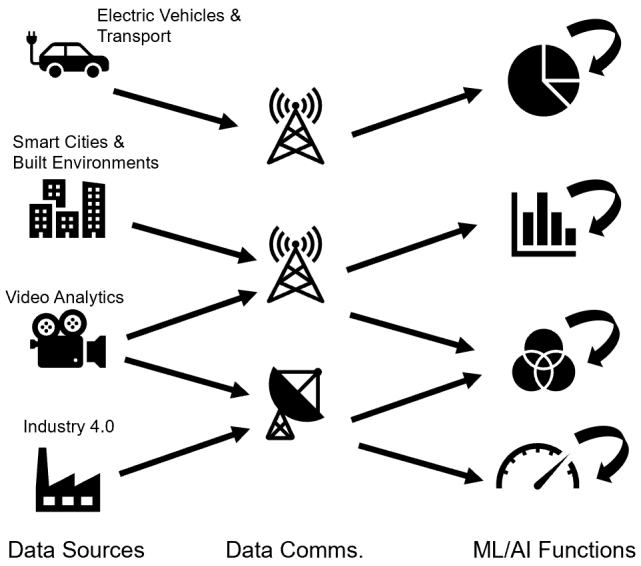

Fig. 1. Serverless functions – responding to incoming data streams

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 counter-productive—leading to applications running for longer time periods, which at times is the worse 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: (i) 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; (ii) 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 like funcx, which runs on Python[9], more fine-grain power capping can take place. Such algorithms could target specific sub-components that might not need to run at full speed, such as resource-intensive dynamic memory management, aka., garbage collection [PKD18]. 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 [PMK+19].

The performance and execution behaviour 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 pre-specified 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 re-ordering, 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 connection setup overhead, supporting multiplexing, removing the head-of-line 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 undertaken more efficiently is also an important approach to reduce energy use [KD19].

Hotspot & Coldspot Migration: A common approach to reducing power consumption is the dynamic consolidation of virtual machines & containers on a smaller number of physical machines. This is based on the observation that physical machines (PMs) run at 10%–50% of their maximum CPU usage and a

[9] https://funcx.org/

majority of PMs are idle, whilst still consuming about 70% of their peak power. This process involves migrating workload to enhance resource usage and minimise the use of machines that are underutilised within a data centre – 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 over-loaded (creating a hotspot) or under-loaded (creating a coldspot). However, consolidation policies reduce energy consumption significantly but live VM migration results in increased violations of Service Level Agreements.

Many of these techniques however suffer from issues of instability and fluctuation – as migration of workload is often based on 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 utilised resources within a data centre. A key challenge that differentiates this challenge within a serverless environment is the overhead of migrating workloads compared to: (i) the function execution time; (ii) the migration time and associated startup time of the function at the new location. Both of these 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 centre consolidation have focused on migrating long-running virtual machine instances to eventually power down idle hosts. More recently, these approaches have been suggested for re-application in cloud-to-fog continuums [OCY+17]. 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 [SJRZ15] in conjunction with delay-tolerant function invocations. This way, hardware sensors along with virtualised 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. Fig. 2 shows the sequence of events, including eventual

suspend and resume actions by the device or virtualised runtimes, based on rules or machine-learned patterns.

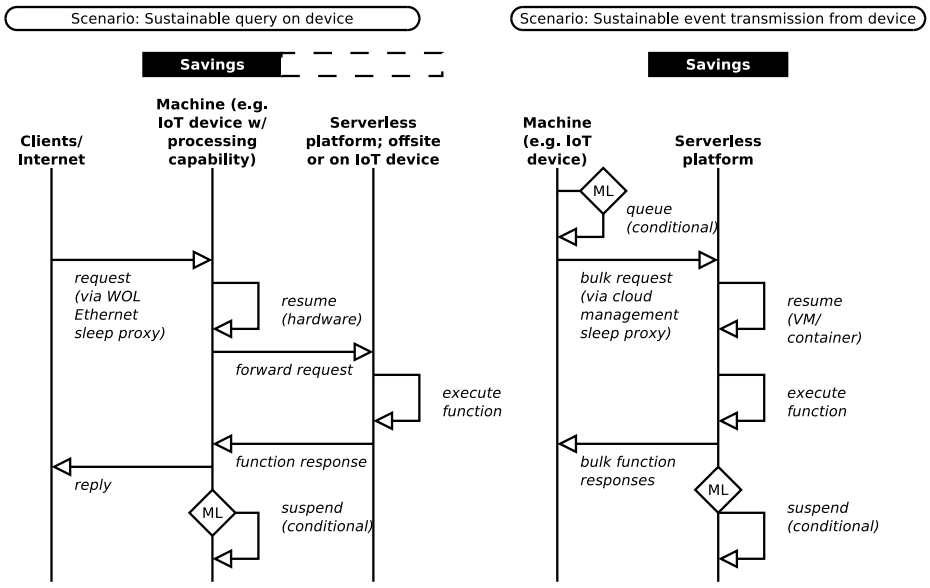

Fig. 2. Sustainability approaches for Internet-wide control of device states and virtualised runtime lifecycle based on serverless event processing

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 0.95 s 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 to great savings in power consumption. Fig. 3 summarises the suspend/resume pattern over a window of six minutes. The research challenge is then to learn and predict messaging patterns to optimise the suspend/resume or switch-off/switch-on actions.

3 The 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). Connectivity between the layers and propagation of an error function differentiates the different types of DNN architectures. Overall, a DNN may include: (i) 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

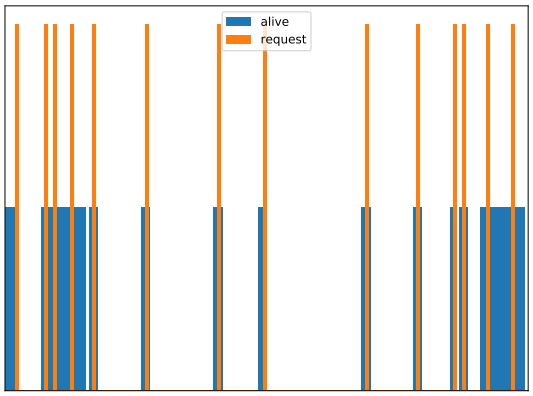

Fig. 3. Event-driven suspend/resume patterns leading to power consumption savings

that are not fully connected, or where connections may skip layers; (ii) 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 data set; (iii) Pooling layers apply a pre-defined function (maximum or average) to downsample the input; (iv) an Activation layer applies non-linear functions and the most commonly used is the rectified linear unit (ReLu); and (v) 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: (i) workloads that are distributed; (ii) the *miniaturisation* of the workload. Traditionally, a DNN is trained at a data centre 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 [LHI+20,ALHV21,KHG+17]. 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 centre) 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 to resource-constrained resources that are located outside conventional data centres closer to where data is 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 is 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 [RdSVGdL21b]). 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 maximise the accuracy of detection [LVWV21]. This is achieved by leveraging the meta-characteristics 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 trade-off to deliver a trans-precision-based approach that maximises the accuracy.

4 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" [vKAH+15]. The SPEC Cloud IaaS 2018 benchmark[10], for example, focuses on four key benchmark metrics: (i) Replicated Application Instances, (ii) Performance Score, (iii) Relative Scalability, and (iv) Mean Instance Provisioning Time, none of which includes sustainable-related metrics. This is the typical focus of most of the benchmarks in cloud computing [PVB+19], and the ones developed for serverless computing [vESE+20, CKB+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. [SGM20] it has been found that *training* a single deep learning model can generate up to 284 000kg of CO_2 emissions. This corresponds to the total lifetime carbon footprint of approximately five cars. But this is not a one-off cost, that concludes with the training of the ML algorithm—that could be potentially mitigated by the use of transfer learning techniques. Amazon estimates that 90% of production ML infrastructure costs are for *inference*, not training [Jas18]. Also NVIDIA estimated that 80-90% of the energy cost of neural networks deployed in data centres lies in inference processing[11].

[10] `https://www.spec.org/cloud_iaas2018/`

[11] `https://www.forbes.com/sites/moorinsights/2019/05/09/google-cloud-doubles-down-on-nvidia-gpus-for-inference/`

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 correspondingly lower. For instance, Google Cloud Platform's power mix is more heavily weighted toward renewables than the AWS Platform (56% vs 17%, according to [SGM20]). Lacoste et al. [LLSD19] developed an ML CO_2 calculator[12] that practitioners can use to estimate the carbon footprints of their deployment based on the following factors: (i) hardware type, (ii) hours used, (iii) cloud provider, and (iv) 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 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 vs. 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 be achieve improved computational density both on cloud and on edge servers. As such, consider for simplicity a multilayer DNN [Bur19]. Each layer will have outputs $\boldsymbol{u} \in \mathbb{R}^m$, weights $\boldsymbol{W} \in \mathbb{R}^{m \times n}$, biases $\boldsymbol{b} \in \mathbb{R}^m$, activation function g and connected with inputs $\boldsymbol{u} \in \mathbb{R}^n$ will execute the following function:

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

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:

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

Consequently, from a serverless perspective, the instructions required to be transmitted in order 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

[12] `https://mlco2.github.io/impact/`

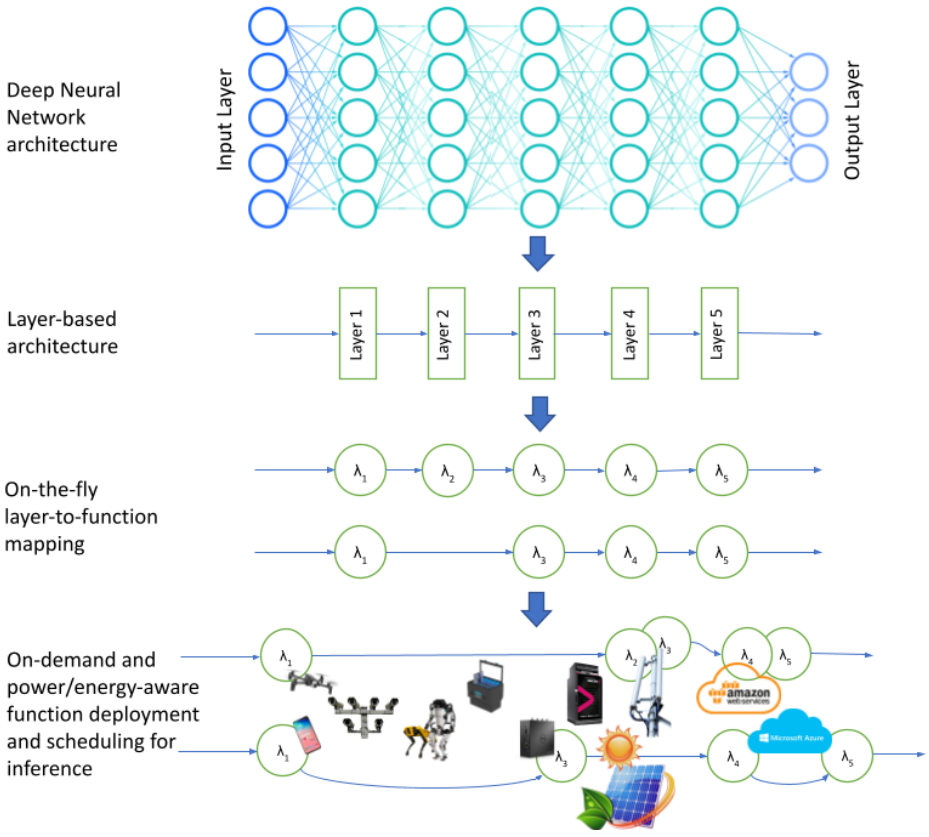

Fig. 4. A serverless approach for the sustainable execution of deep AI inference.

the stress on the grid, depending on their sparsity, place them on machines with the appropriate level of hardware parallelism (e.g., CPU vs. 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 micro-service 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 runtime-specific energy-aware optimizations. For example, consider multi-tenancy 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 the λ^{DNN} of another.

5 Conclusion

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: a) serverless platform design and infrastructure, b) 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 c) 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." [Hao19].

References

[ALHV21] Hyunho Ahn, Munkyu Lee, Cheol-Ho Hong, and Blesson Varghese. Scissionlite: Accelerating distributed deep neural networks using transfer layer, 2021.

[AMKF18] Joshua Aslan, Kieren Mayers, Jonathan G Koomey, and Chris France. Electricity intensity of internet data transmission: Untangling the estimates. *Journal of Industrial Ecology*, 22(4):785–798, 2018.

[Bur19] Andriy Burkov. *The hundred-page machine learning book*, volume 1. Andriy Burkov Canada, 2019.

[Cis20] U Cisco. Cisco annual internet report (2018–2023) white paper, 2020.

[CKB+20] Marcin Copik, Grzegorz Kwasniewski, Maciej Besta, Michal Podstawski, and Torsten Hoefler. Sebs: A serverless benchmark suite for function-as-a-service computing, 2020.

[Hao19] Karen Hao. Training a single ai model can emit as much carbon as five cars in their lifetimes. *MIT Technology Review*, 2019.

[Jas18] Andy Jassy. Amazon AWS ReInvent keynote, 2018. https://www.youtube.com/watch?v=ZOIkOnW640A.

[KD19] Puneet Kumar and Behnam Dezfouli. Implementation and analysis of QUIC for MQTT, arxiv: 1810.07730, 2019.

[KHG+17] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, page 615–629, 2017.

[LHI+20] Luke Lockhart, Paul Harvey, Pierre Imai, Peter Willis, and Blesson Varghese. Scission: Performance-driven and context-aware cloud-edge distribution of deep neural networks. In *IEEE/ACM 13th International Conference on Utility and Cloud Computing*, pages 257–268, 2020.

[LLSD19] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

[LVWV21] JunKyu Lee, Blesson Varghese, Roger Woods, and Hans Vandierendonck. Tod: Transprecise object detection to maximise real-time accuracy on the edge. In *5th IEEE International Conference on Fog and Edge Computing*, 2021.

[Myt20] David Mytton. How much energy do data centers use? hypertext document, 2020. Available electronically at https://davidmytton.blog/how-much-energy-do-data-centers-use/.

[OCY+17] Opeyemi Osanaiye, Shuo Chen, Zheng Yan, Rongxing Lu, Kim-Kwang Raymond Choo, and Mqhele Dlodlo. From cloud to fog computing: A review and a conceptual live vm migration framework. *IEEE Access*, 5:8284–8300, 2017.

[PKD18] Panagiotis Patros, Kenneth B Kent, and Michael Dawson. Mitigating garbage collection interference on containerized clouds. In *2018 IEEE 12th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 168–173. IEEE, 2018.

[PMK+19] Vladimir Podolskiy, Michael Mayo, Abigail Koay, Michael Gerndt, and Panos Patros. Maintaining SLOs of cloud-native applications via self-adaptive resource sharing. In *2019 IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 72–81. IEEE, 2019.

[PSR20] Alexander Poth, Niklas Schubert, and Andreas Riel. Sustainability efficiency challenges of modern it architectures – a quality model for serverless energy footprint. In Murat Yilmaz, Jörg Niemann, Paul

Clarke, and Richard Messnarz, editors, *Systems, Software and Services Process Improvement*, pages 289–301, Cham, 2020. Springer International Publishing.

[PVB+19] Alessandro V. Papadopoulos, Laurens Versluis, André Bauer, Nikolas Herbst, Jóakim von Kistowski, Ahmed Ali-Eldin, Cristina L. Abad, José Nelson Amaral, Petr Tůma, and Alexandru Iosup. Methodological principles for reproducible performance evaluation in cloud computing. *IEEE Transactions on Software Engineering*, 2019.

[RdSVGdL21a] Brian Ramprasad, Alexandre da Silva Veith, Moshe Gabel, and Eyal de Lara. Sustainable computing on the edge: A system dynamics perspective. In *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications*, pages 64–70, 2021.

[RdSVGdL21b] Brian Ramprasad, Alexandre da Silva Veith, Moshe Gabel, and Eyal de Lara. Sustainable computing on the edge: A system dynamics perspective. In *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications*, page 64–70, 2021.

[SGM20] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for modern deep learning research. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(09):13693–13696, Apr. 2020.

[SJRZ15] Jiming Sun, Marc Jones, Stefan Reinauer, and Vincent Zimmer. *Building coreboot with Intel FSP*, pages 55–95. Apress, Berkeley, CA, 2015.

[SSK+21] Johann Schleier-Smith, Vikram Sreekanti, Anurag Khandelwal, Joao Carreira, Neeraja Jayant Yadwadkar, Raluca Ada Popa, Joseph E. Gonzalez, Ion Stoica, and David A. Patterson. What serverless computing is and should become: the next phase of cloud computing. *Commun. ACM*, 64(5):76–84, 2021.

[vESE+20] Erwin van Eyk, Joel Scheuner, Simon Eismann, Cristina L. Abad, and Alexandru Iosup. Beyond microbenchmarks: The spec-rg vision for a comprehensive serverless benchmark. In *Companion of the ACM/SPEC International Conference on Performance Engineering*, ICPE '20, page 26–31, New York, NY, USA, 2020. Association for Computing Machinery.

[vKAH+15] Jóakim von Kistowski, Jeremy A. Arnold, Karl Huppler, Klaus-Dieter Lange, John L. Henning, and Paul Cao. How to build a benchmark. In *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*, ICPE '15, pages 333–336, New York, NY, USA, 2015. Association for Computing Machinery.