Reducing Datacenter Compute Carbon Footprint by Harnessing the Power of Specialization: Principles, Metrics, Challenges and Opportunities

Tamar Eilam, Pradip Bose, *Life Fellow, IEEE*, Luca P. Carloni, *Fellow, IEEE*, Asaf Cidon, Hubertus Franke, Martha A. Kim, Eun K. Lee, Mahmoud Naghshineh, Pritish Parida, *Member, IEEE*, Clifford S. Stein, and Asser N. Tantawi, *Life Senior Member, IEEE*

***Abstract*—Computing is an indispensable tool in addressing climate change, but it also contributes to a significant and steadily increasing carbon footprint, partly due to the exponential growth in energy-demanding workloads, such as artificial intelligence (AI). While hardware specialization has become the primary driver of operational energy efficiency improvements, it introduces new challenges including increased embodied emission, and a rise in complexity of operations of heterogeneous and dynamic datacenters. We posit that while specialization is necessary for sustainable computing, to fully harness its power, the academic and technical community must address the specific challenges arising from embracing it. We enumerate and analyze key challenges that specialization introduces across software, system design, and operations, and their potential impact on carbon cost, and propose a way forward for each identified area. Furthermore, we argue that intricate relationships exist across the life-cycle of compute systems, which must be understood, modeled, and analyzed to identify the most beneficial Pareto frontiers for carbon life-cycle efficiency. We analyze these trade-offs and offer an approach to address them using a unified metric and framework.**

***Index Terms*—Carbon footprint, chiplet design, energy efficiency, eBPF, low power consumption, runtime optimization, specialization, sustainable computing.**

I. Introduction

Climate change is arguably the biggest threat that humanity is facing today [1]. Datacenters are an increasing source of carbon emissions which has two components: *operational emissions*, comprising the carbon footprint of the machines plus cooling and power distribution overheads, and *embodied emissions*, incurred by the design and manufacturing of computing systems [2]. Datacenters already account for 1-2% of global electricity consumption, and their share is dramatically increasing. Global datacenter electricity consumption reached 460 TWh in 2022 (more than doubling since 2018) and is expected to exceed 620 TWh in 2026 [3].

A confluence of trends contribute to this acceleration in datacenter carbon emissions. First is an increased demand for computing in general, and particularly for power-hungry AI workloads, including large language models, foundation models, and generative AI [4]. Model sizes have grown steeply, from the 110 million parameters in the BERT base model (2018) [5], to 175 billion parameters in GPT-3 (2020) [6], and trillions of parameters (speculated) in GPT-4 (2023). The energy consumed in data preparation, training, adaptation and inference has grown with parameter count. Strubell et al. [7] equate the training of a single large language model with the carbon emissions of five cars through their life time. Recently there have been multiple attempts to quantify the energy usage and carbon emissions of models throughout all phases of their life cycle, including the manufacturing of specialized systems required to run AI, and operational phase activities including data preparation, training, alignment, and inference [8], [9], [10], [11].

The second trend is the demise of Dennard scaling [12] combined with the slowing of Moore's Law. In anticipation of this outlook, early work by Brooks et al. [13] highlighted the need to invest in power-aware microarchitecture design, aided by associated pre-silicon power-performance modeling methodologies. Much later, in a seminal paper [14], Horowitz established that we had reached the physical limits of power density, and thus energy had become the bottleneck in scaling performance of CMOS devices. This reality, coupled (of late) with the intense computational demands posed by AI, fuels the rush to design specialized systems that can reduce the energy per operation, albeit for a subset of computations. Accelerators such as Graphics Processing Units (GPUs), Tensor Processing Units (TPUs), AI Units (AIUs), and fixed-function hardware modules are much more efficient in AI training and inference than general-purpose CPUs. Consequently, integrated-circuit designers have moved towards system-on-chip (SoC) architectures that combine a heterogeneous mix of these components.

Given the exponential growth in energy-hungry workloads and the flattening of Dennard scaling, we posit that to fully

Manuscript received 21 February 2024; revised 6 June 2024; accepted 28 June 2024. Date of publication 31 July 2024; date of current version 20 November 2024. *(Corresponding author: Tamar Eilam.)*

Tamar Eilam, Pradip Bose, Hubertus Franke, Eun K. Lee, Mahmoud Naghshineh, Pritish Parida, and Asser N. Tantawi are with IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: eilamt@us.ibm.com; pbose@us.ibm.com; frankeh@us.ibm.com; eunkyung.lee@us.ibm.com; mahmoud@us.ibm.com; prparida@us.ibm.com; tantawi@us.ibm.com).

Luca P. Carloni, Asaf Cidon, Martha A. Kim, and Clifford S. Stein are with Columbia University, New York, NY 10027 USA (e-mail: luca@cs.columbia.edu; asaf.cidon@columbia.edu; martha@cs.colmbia.edu; cliff@ieor.columbia.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSM.2024.3434331.

Digital Object Identifier 10.1109/TSM.2024.3434331

harness the power of specialization, we must adopt a holistic approach addressing all phases of the compute life-cycle. In doing so, we must analyse and address the unique challenges that are introduced or exacerbated by specialization and the ensuing heterogeneity.

The paper is organized as follows. In Section II, we enumerate four main challenges that specialization and the ensuing heterogeneity introduce or exacerbate across the life-cycle of compute, and their potential negative effect on carbon cost: (1) life-cycle trade-offs, (2) design complexity, and cost, (3) Bloated, wasteful Operating System (4) heterogeneous data-center management complexity. The remaining sections are dedicated to addressing each of the challenges. Section VII concludes the paper.

II. Specialization: Complexity and Opportunities

As the demand for energy-efficient computing continues to grow, chip designers are increasingly turning to specialized architectures and SoC designs, where a plurality of accelerators are introduced along side general purpose CPUs on a single chip. This is a necessary step in order to improve the operational efficiency of a diverse set of high-intensity workloads that exhibit special characteristics, such as, signal and stream processing, Artificial Intelligence, and Cryptography. While specialization is necessary for sustained energy efficiency improvements, the following challenges that must be understood and addressed in order to fully harness its power.

Challenge 1: Tradeoffs across the lifecycle By tailoring the hardware to the software's needs, heterogeneous designs can achieve significant improvements in energy efficiency during the operational stage of the compute life cycle. However, as the complexity of single-die SoCs grows, so is the carbon associated with its design and manufacturing. There are several factors that contribute to the increase in carbon cost, including reduced yield, increase in the number of manufacturing processing steps required, and increase in required gases, used for etching, photolithography, and deposition. While the discussion of sustainable manufacturing is beyond the scope of this paper, we argue that trade-offs across the life-cycle between carbon manufacturing cost (contributing to embodied emission) and operational energy-efficiency gains must be analyzed and understood in order to identify optimal design choices for lifetime carbon efficiency. In Section III we discuss these trade-offs and offer a unified metric and a conceptual framework to meaningfully address them.

Challenge 2: Sustainable design Heterogeneity increases the complexity of the design process in terms of hardware-software interactions, access to shared resources, and diminished regularity of the design [15]. This complexity contributes to an increase in energy spent on designing new specialized chips. The problem is exacerbated by the design irregularity and the lack of means for an effective reuse of existing intellectual property (IP) blocks in designing new chip by composition, rather than re-invention. In Section IV, we discuss an approach to address these challenges, by using an open-source platform that encourages and facilitates modularity and reuse in SoC design. We analyse the benefits of this approach with a real-life use case.

Challenge 3: Sustainable operating systems Existing datacenter system software, and in particular widely-used operating systems (OSes) and hypervisors, have no awareness of sustainability when allocating resources and scheduling applications. In fact, existing OS resource allocation policies, which generally aim to keep computing resources fully utilized, may lead to increased operational carbon emissions, for example by unnecessarily keeping "cold" pages in memory to maximize memory utilization. In addition, due to their generality and need to support legacy applications and interfaces, OSes are *bloated*, wasting many cycles synchronizing and traversing software layers, rendering them poorly-equipped to take advantage of specialized hardware. Indeed, it has been estimated that about 20% of datacenter cycles go to waste in Linux's various software layers alone [16]. In Section V, we explore a promising approach to make system software sustainability-aware, by exploiting *extensibility* frameworks (eBPF) supported by Linux and Windows. These frameworks can be used to expose sustainability-related metrics to applications, bypass expensive OS software layers, and to customize OS behavior to reduce operational emissions and take full benefit of heterogeneous datacenter resources to optimally run a diverse set of workloads.

Challenge 4: Sustainable datacenters A heterogeneous datacenter incorporates multiple different types of chips, memory, storage, and network, organized in layers spanning power sources, hardware systems, operating systems, management software, and applications frameworks. Such a datacenter is used to run a dynamic and a diverse set of workloads, where every workload is associated with a service level objective (SLO) defining latency, bandwidth, energy and cost goals. A careless allocation of resources to workloads may result in inefficiencies, due to underutilized, or stranded resources. Because hardware components span orders of magnitude in performance and efficiency, optimization algorithms and their implementations are essential at multiple levels: at the bottom, to control power caps and device partitions to respect power budgets, higher up to map workloads within and across racks for efficient cooling and job completion. An optimal use of datacenter resources must factor-in aspects such as the specific characteristics of the compute systems and workloads, as well as other factors such as cooling, which is a major source of energy cost. Section VI further delves into heterogeneity's impact on runtime optimization in datacenters, giving special attention to the unique challenges and opportunities introduced by AI workloads.

III. Addressing Trade-Offs Across the Life-Cycle

As single-die SoCs grow in complexity, so does the carbon associated with their manufacturing. The number of processing steps increases and fabrication yields drop, increasing the amortized carbon cost per chip. This situation is not unique to processing chips, but also applies to memory, storage and

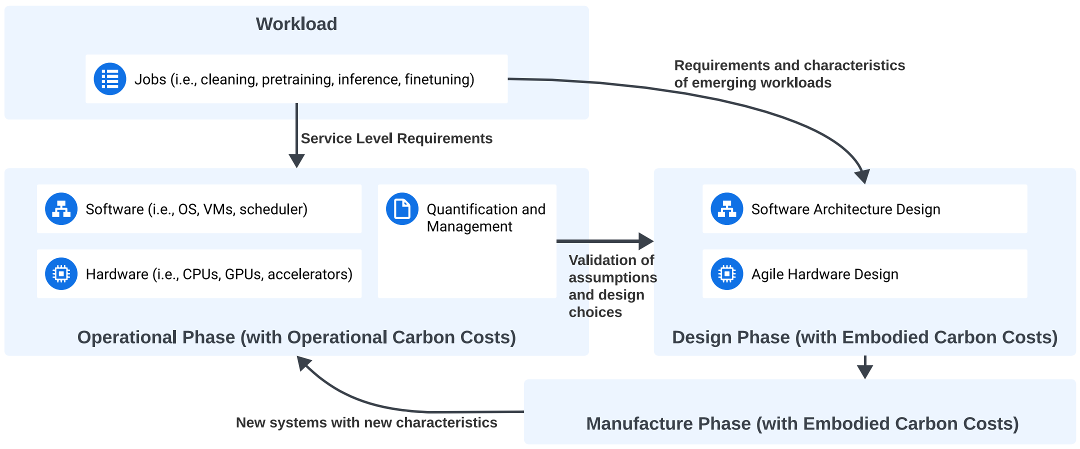
Workload
Jobs (i.e., cleaning, pretraining, inference, finetuning)
Requirements and characteristics of emerging workloads
Service Level Requirements
Software (i.e., OS, VMs, scheduler)
Hardware (i.e., CPUs, GPUs, accelerators)
Quantification and Management
Validation of assumptions and design choices
Software Architecture Design
Agile Hardware Design
Operational Phase (with Operational Carbon Costs)
Design Phase (with Embodied Carbon Costs)
New systems with new characteristics
Manufacture Phase (with Embodied Carbon Costs)

Fig. 1. Our vision for sustainability, where the coupled concerns of chip design, manufacture, and operation are considered together to reduce carbon emissions in a comprehensive manner.

network systems. Take as an example a solid-state drive (SSD), whose reliability depends on the number and frequency of write operations and thus drops over time. This shortens its useful lifespan and increases the amortized carbon cost per operation. Redundancy is a common remedy for this situation. By incorporating redundant components, such as extra memory cells or controller circuits, designers can ensure that a device continues to function even if some components fail. This, in turn, amortizes the embodied emissions associated with the device's production over a longer lifetime, thereby improving its lifetime carbon efficiency at the expense of increased carbon cost during manufacturing. A holistic life-cycle carbon analysis is thus necessary to meaningfully identify the optimal design point, balancing the carbon cost of redundant hardware with the increased amortization thanks to an extended device lifetime. Generalizing this principle, we argue that it is important to analyse system design choices, by taking into account both the expected operational efficiency gains and the embodied cost losses leading to optimal *life-cycle carbon efficiency*.

In a comprehensive carbon reduction ecosystem (depicted in Figure 1), operational and manufacturing concerns are analyzed together to co-evolve hardware and software to minimize the life-cycle carbon footprint. During the *Operational Phase*, management software will allocate resources to arriving jobs, based on the type of job, its SLO, and the properties of the resources. Such management software must rely on online quantification (such as [17]) to understand the current state and behavior of the infrastructure. In the *Design Phase*, system designers will design the next iteration of hardware and software, potentially optimizing either one to run a particular class of workloads based on understanding of their runtime characteristics. At this point, it is crucial for designers to anticipate the impact of introducing a new hardware or software product. Will the runtime reductions in carbon emissions be enough to offset the carbon emissions from developing the product? Or, will the carbon cost of creating the product in the first place swamp any savings when the product comes online?

To inform new system design, we require comprehensive metrics that reflect the shared metrics reflecting operational carbon and embodied carbon costs. Here we put forth expressions that quantify and relate embodied and operational carbon footprint, both of which we quantify using units of *carbon dioxide equivalent* (gCO_2e).

We start with the *job sustainability cost (JSC)*, defined by Gandhi et al. [18]. The JSC of a job includes the total carbon footprint of all of the systems participating in the execution of that job, including indirect support systems that provide cooling and power distribution. Because a job might run on multiple machines, we quantify its JSC piece by piece.

The expression below, $JCS(j, m)$ is the operational carbon emissions of the portion of job j that runs on machine m. It starts with the number of operations to be executed on m ($Ops(j, m)$) and multiplies that number by the efficiency of machine m ($\frac{J_{needed}(m)}{Op}$). This product gives the total Joules needed by job j on machine m, which is then converted to Joules needed by factoring in the *power usage efficiency* ($\frac{J_{used}}{J_{needed}}$) of the datacenter. This energy cost is converted to carbon cost based on the cleanliness of the energy source, i.e., the mix of energy sources and their associated carbon emissions ($\frac{gCO_2e}{J_{used}}$).

$$JSC(j, m) = Ops(j, m) \times \frac{J_{needed}(m)}{Op} \times \frac{J_{used}}{J_{needed}} \times \frac{CO_2}{J_{used}}$$

The first two of these terms are within the sphere of control for hardware and software designers. Thus, their objective is to minimize the total energy needed for the job (J_{needed}, the product of the first two terms). In so doing, system designers can minimize carbon emissions regardless of the datacenter design (third term) or energy source (fourth term).

To this operational carbon cost, the *amortized sustainability cost (ASC)* [18] layers in the amortized carbon cost of manufacturing each machine m involved in completing job j:

$$ASC(j) = \sum_{m \in j} \left(JSC(j, m) + \frac{Ops(j, m)}{Ops(m)} \times EMC(m) \right)$$

Here, $EMC(m)$ is the embodied cost for manufacturing m. So, to the baseline JSC, we are adding a share of the embodied carbon cost of m, with this share calculated as a proportion of the operations executed on m due to job j ($Ops(j, m)$) to the total operations completed by m over its lifetime ($Ops(m)$). These definitions can be generalized to factor in different types of operations performed on m in the execution of a job j, where each operation type consumes a different amount of energy.

Finally, Dennard et al. [12] extend the notion of an embodied cost from hardware to software, defining the *embodied product cost (EPC)*, which quantifies the carbon cost of a software product. For example, an AI model m is a software product whose EPC is the sum of all $ASC(j)$ for all j involved in the development, test, pre-training, and other jobs executed in order to deliver m.

To understand how these expressions capture the key dynamics, let us consider a real-world example of hardware reuse. DDR4 memory devices are not supported by the memory bus interface of modern server platforms. However, major datacenter operators, including Meta [20] and Microsoft [21], plan to reuse old DDR4 DIMMs by connecting them via the new CXL memory interface, and leaving the memory bus to newer DDR5 memory. In this particular example, if DDR4 CXL memory comprises 40% of the total server memory capacity, which is projected to be a typical server configuration in certain datacenters [22], operators expect to reduce server emissions by more than 20% thanks to the recycled DDR4s [21]. Recycling DIMMs in this way effectively extends their lifetime. If this is anticipated early, it can be reflected in an increased $Ops(m)$ value. If it is not anticipated, the embodied cost of the DIMM is paid off over the original lifetime, after which the only carbon cost to using the DIMM is the operational JSC.

IV. Agile Design for Sustainability

In the late CMOS design era, energy-efficient performance is achieved by realizing domain-specific SoC architectures that are heterogeneous because they combine CPU cores with a large variety of specialized hardware *accelerators*. However, as discussed in Section I, the price of heterogeneity is an increase in design complexity. Indeed, the net cost of design, programming and verification of heterogeneous SoCs is a major contributor to embodied carbon costs (Figure 1).

Agile SoC design methodologies [23] can play a crucial role in carbon footprint reduction by simplifying design reuse and increasing the productivity of the design team. The ESP project [15], [19], [24], developed at Columbia University, has evolved rapidly to become an open-source design platform that enables the design, emulation, and silicon tape-out of complex heterogeneous SoCs in less than thirty person-months [25], [26]. Figure 2 shows a high-level view of the ESP methodology, which supports a number of front-ends for design specifications: from register-transfer level (RTL) specifications expressed in hardware description languages such as SystemVerilog, to higher-level specification expressed in SystemC that can be synthesized to hardware with high-level synthesis tools. Furthermore, for the critical domain of AI, ESP leverages the *hls4ml* open-source project [27], [28] to provide a domain-specific design flow from higher-level specifications expressed in frameworks like PyTorch. A key enabler of agile design is the support for the reuse of IP blocks through libraries of hardware and software components that combine ESP-native elements (developed with the various design flows) with external IP blocks acquired from the open-source world. The ESP methodology provides almost "push-button" capabilities for integrating multiple, heterogeneous IP blocks into a single, tiled-based SoC architecture [24]. Tiles are connected by a scalable, multi-plane twp-dimensional mesh network-on-chip, which provides them with a set of pre-designed platform services [15]. ESP simplifies the generation of an integrated bitstream for FPGA implementation; or, eventually, an integrated RTL that drives the backend ASIC design flow [29]. The agility of the ESP approach was demonstrated with the realization of the EPOCHS-1 chip in a 12nm technology process [26]. The design of this SoC, which features a variety of open-source hardware components (14 different types of accelerators next to 4 RISC-V cores capable of running many simultaneous applications on top of a Linux-SMP OS) was carried out by a small team of PhD students, postdocs, and industry researchers in 3 months.

To compensate the slow-down of technology scaling and maintain competitive growth in performance, the sizes of chips tend to increase, especially processor chips for high-end server- and mainframe-class processors. An increase in chip size negatively affects the yield, thus resulting in cost escalation. This is clearly not a sustainable proposition for the chip industry. In this context, chiplet-oriented design with heterogeneous integration (HI) packaging technology has the potential of starting a new era of computing [30]. This cost-effective and carbon-friendly design trend promises to sustain the next-generation growth in datacenter footprint across the globe. Taking advantage of chiplet/HI packaging technology requires that agile SoC design methodologies evolve to support system-in-package (SiP) designs with advanced degrees of design automation. In particular, agile SoC+SiP design methodologies can reduce operational carbon costs by enabling modular and scalable on-system power-management architectures (Figure 1). A proof point of this claim is BlitzCoin [31], the novel distributed hardware power manager (DHPM) that was implemented in the EPOCHS-1 chip by using the ESP methodology [26] BlitzCoin gets rid of the centralized bottleneck posed by classical global power-management architectures in multi-core processors. As it fosters scalability, DHPM is the way forward for many-core designs with or without heterogeneity. With this approach, each processor or accelerator tile is designed to support its own power control mechanism, guided by its allocation of power tokens. The DHPM manages the dynamic exchange of tokens across neighboring tiles, driven by changing workload demands. Its novel algorithm maximizes system-level performance for a given power budget, which is determined by the maximum number of power tokens in the system [31].

In ongoing work that builds on the BlitzCoin approach, we have shown that the decentralization of power management can be architected hierarchically, from chips to servers and

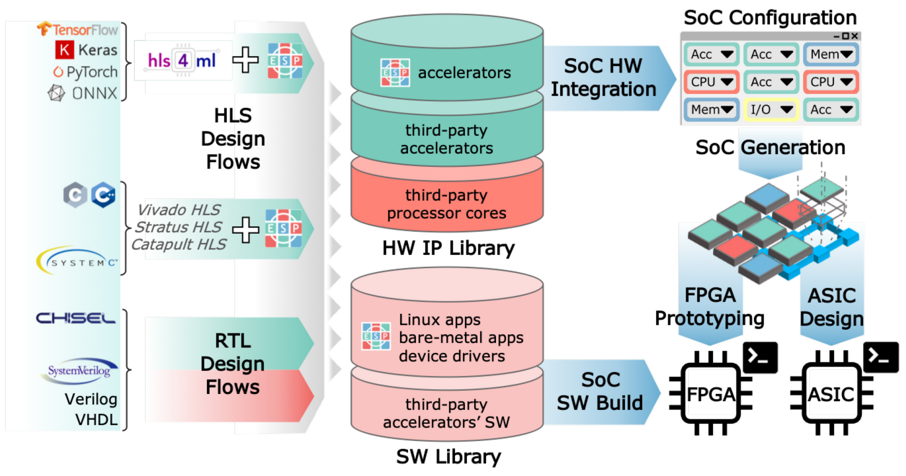

Fig. 2. Agile SoC design with the ESP platform [19].

all the way up to the datacenter level. With the onset of the modular, chiplet-oriented design era, we envision the use of reusable power management chiplets that would be instantiated in a SiP to monitor and control the power consumption of the other compute and memory chiplets. The modular reuse of chiplets, coupled with heterogeneous integration, points to cost-effective design with reduced carbon emission over time.

V. Sustainability-Aware Operating System

While hardware systems have always been designed with power (and energy) as a first class citizen, system software has almost no awareness of power or energy. In particular, the OS, which sits at the core of datacenter resource allocation, whose primary function is to assign hardware resources (CPU, memory, I/O bandwidth) to applications, does not take sustainability into account in its decisions. Instead, the implicit goal of the OS is typically to keep all hardware resources (e.g., CPU/GPU cores, memory) fully utilized, while maintaining some degree of fairness across applications. While in some cases a policy of high hardware resource utilization does lead to lower carbon emissions, in many cases it does not. For example, over time, the Linux OS will end up allocating all of its available memory capacity to long-running applications, even if the application "working set" (i.e., the amount of memory they need to achieve reasonable performance) is much smaller than the total memory capacity. In contrast, a sustainability-aware OS may choose to use a lower amount of memory, and even power down idle energy-hungry DRAM banks. Such a policy could even allow the datacenter operator to deploy fewer DRAM DIMMs (or reuse older, low capacity DIMMs), reducing not only operational carbon emissions, but also embedded ones.

Making matters worse, existing OSes introduce many software layers of abstraction that lead applications to waste a significant percentage of their CPU cycles on "useless" computation that is not related to the actual application. Examples include copying memory unnecessarily from one layer of the OS stack to another, synchronizing across different layers, and serialization and deserialization. Our prior work shows that when reading small requests from storage in Linux, about half of CPU cycles are wasted simply traversing the multiple layers of Linux's software stack [32]. The Linux networking stack is notoriously even more wasteful [33], [34]. This "bloated" software stack leads to both increased operational and embedded carbon emissions, causing datacenter operators to over-provision their clusters.

Finally, with the proliferation of specialized hardware, the task of an OS becomes even harder. It is very difficult for a monolithic OS, such as Linux or Windows, to anticipate every possible hardware configuration it will be deployed with, in order to hide the complexity of those hardware configurations from applications, and to optimally utilize hardware. While one approach to solve this problem is to design a "clean slate" OS that will be sustainability-aware as well as be less wasteful, it would exorbitantly expensive to migrate the entire application ecosystem to a new OS.

Fortunately, mainstream OSes (Linux and Windows) now support a technology, called eBPF [35], that allows us to *customize and extend* these OSes. eBPF allows applications to run custom functions in the kernel. This might allows applications for example to modify the OS' scheduling policy to be sustainability-aware [36], [36], [37], use eBPF to bypass expensive software layers that lie on the critical path of the application [32], [38], or use eBPF to take advantage of new types of programmable hardware accelerators, such as a smartNIC [39] or smartSSD [40].

Therefore, we envision that by using eBPF, we can transform standard monolothic OSes like Linux to become sustainability-aware, reduce their overhead and better utilize the plethora of hardware accelerators that will become available in the coming years. However, there are significant open obstacles in realizing this vision.

We group these obstacles into three categories: (a) lack of software-level telemetry to measure sustainability, (b) new integrations with Linux, and (c) limitations of existing eBPF framework. The first obstacle is the lack of visibility into metrics related to sustainability at the software level. Today, applications have no knowledge of how much energy they consume or carbon emissions they generate. There have been initial efforts in this direction, including the open source Kepler project [17], which provides applications with visibility on how much energy they consume, utilizing existing power estimation tools provided by Intel and AMD CPUs. However, there remain significant gaps: energy does not automatically translate to carbon (especially embodied carbon) emissions, and right now we only have visibility on the energy usage of CPUs, but not of memory, storage or the network.

The second obstacle in making Linux a "sustainable OS" using eBPF is that it currently has no mechanisms to incorporate energy (and carbon) in its resource allocation and scheduling policies. For example, to make Linux sustainability-aware requires navigating its millions-of-line code base to find appropriate places to add new tracepoints (Linux functions that can be observed externally by an application) and hooks (points in the kernel code where custom eBPF function can be inserted in runtime) that can accommodate sustainability-aware policies. For example, to allow Linux to take advantage of a particular programmable network or storage device, we need to add hooks in Linux's I/O stack that would allow it to offload I/O computations to the external device. Or in order to make Linux incorporate energy-consumption into its scheduling decisions, we need to be able to add hook into its scheduling functions to take into account energy as an additional metric.

The final obstacle is that eBPF as a programming framework itself imposes certain limitations, which limit its functionality, such as the length of functions, its ability to dynamically allocate and reference memory, etc. These limitations exist in order for functions to be verified before they run in privileged kernel space. However, some of these limitations may constrain our ability to customize Linux to be more sustainable. For example, in order to offload functions and data dynamically to external energy efficient devices (e.g., programmable NICs), we may need an ability to dynamically allocate memory in eBPF, and to more efficiently synchronize and share memory between the kernel and userspace applications.

We have taken a first step in the journey of making Linux more sustainable, by making the storage datapath be significantly more energy efficient [32], [41]. Storage is a significant source of carbon emissions in datacenters, both directly (i.e., when accessing SSD devices), but also indirectly, by generating a significant percentage of datacenter network traffic in the case of networked storage (e.g., accounting for 70% of all network traffic in Azure datacenters [42]), and consuming a significant percentage of CPU when traversing the Linux storage stack for I/O intensive applications [32]. We made the observation that storage-intensive applications (e.g., databases, analytics systems) issue series of dependent storage accesses when they navigate a large on-disk data structure. Today, these storage accesses require excessive traversals of the OS storage stack, and even worse, in the case of accessing a remote networked storage device, they cause many back-and-forth network requests.

To avoid these back-and-forth requests, we created a new framework called XRP (eXpress Resubmission Path), which allows applications to run their storage traversals as eBPF functions within the operating system. In the case of locally-attached storage, this reduces unnecessarily traversing Linux's storage stack, wasting CPU cycles. In the case of remote storage, this reduces a significant amount of network traffic (and its processing). We integrated several storage systems with XRP, including the widely-used RocksDB and WiredTiger, and showed that XRP can directly reduce the CPU's energy consumption by 37% [41]. Furthermore, due to its much more efficient operation, XRP allows datacenter operators to deploy their storage clusters with far less DRAM than they do today without hurting application performance, thereby significantly reducing operational emissions and embedded emissions.

VI. Sustainability Runtime Optimization

The goal of minimizing the carbon footprint of datacenters as they process various types of workloads, without compromising performance or cost, has been considered at various levels of the datacenter hierarchy [43], [44]. Generically, datacenter power management is layered, where each layer has its own, unique controls that are set by independent optimizing controllers.

Significant work has been done addressing datacenter power management. However, the proliferation of AI workloads and the associated specialized hardware, has introduced new optimization problems. The new specialized hardware tends to have high power consumption. AI workloads have special characteristics in terms of resource demand, performance, and energy. Innovations at the specialized devices, their drivers, and software stacks, create opportunities to better utilize them. We aim to tackle some of these many challenges.

AI workloads, especially driven by large language models (LLMs), may be categorized into three types: training, fine-tuning, and inference. The needs of these three types of workloads differ dramatically. Training jobs employ hundreds, if not thousands, of GPU units for long duration such as weeks or months. Fine-tuning jobs utilize a few or tens of AI accelerator units for minutes or hours. Inference workloads serve a stream of query requests, with varying intensity over time, each using a whole, multiples, or a fraction of an accelerator, with millisecond-scale response times required via SLO. Horizontal, as well as vertical, auto-scaling of the inference service is needed to react to surges in the request rate with the goal of achieving guaranteed quality of service for the various classes of requests. Horizontal auto-scaling involves the adjustment of the number of replicas serving a particular LLM inference model, whereas vertical auto-scaling involves the re-assignment of GPU types, or slice size, to replicas.

Managing the resources for AI workloads opens a number of optimization problems, which are compounded in the real world where these workloads are co-mingled. The usage pattern of workloads may incite the sharing of an accelerator unit among multiple tasks [45]. Finding the optimal partitioning, whether through time/space-sharing or slicing, among multiple tasks is a challenging optimization problem [46], [47].

Modern accelerators offer the ability to impose a power cap - trading off power and performance. Orthogonally, many accelerators can be spatially partitioned (or "sliced" using NVIDIA's Multi-Instance GPU or MIG feature [48]) to allocate sub-accelerator slices to independent workloads. Configuring even one accelerator optimally requires understanding how a workload will respond power and

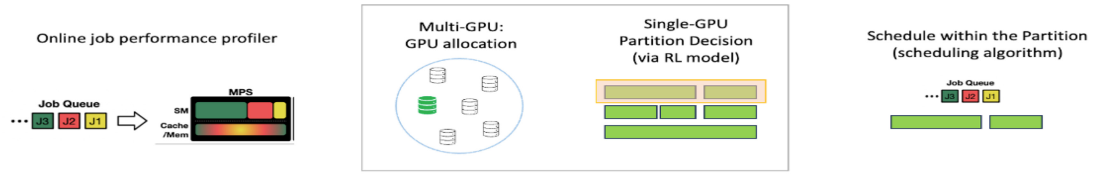
Online job performance profiler
Job Queue
J3 J2 J1
MPS
SM
Cache /Mem
Multi-GPU: GPU allocation
Single-GPU Partition Decision (via RL model)
Schedule within the Partition (scheduling algorithm)
Job Queue
J3 J2 J1

Fig. 3. Energy efficient scheduling using MIG.

performance-wise to these settings, and then configuring and allocating the resources to best match the current workload. The space becomes richer when multiple accelerator types are available, as is typical in a datacenter. Furthermore, the scheduling algorithm that sends a task to a slice of an accelerator, a whole accelerator, or multiple accelerators, must be efficient and scalable. Figure 3 illustrates job scheduling on multiple GPU units, where each unit may be sliced to improve sharing and minimize power consumption, while satisfying job performance objectives. Slicing configuration is determined through profiling, and once a configuration is selected and deployed, a scheduling algorithm decides on the dispatching of jobs to the sliced partitions.

We have experimented with the impact of paring down a GPU via slicing and power capping to understand the relationship between performance and energy for training workloads [49]. Our experiments suggest that power increases proportionally to the size of a GPU slice to a point, after which it saturates despite the share of the GPU continuing to increase. This suggests that the optimal operational point for a GPU is at full utilization, whether via multiple small slices, or one larger workload that can saturate the device. Further work is sought to examine a wider array of AI accelerator types.

A hybrid technique of using dynamic workload balancing and static slicing to arrive at an optimal configuration, that is based on machine learning, has been suggested [46]. Additionally, once a particular partitioning configuration is effected, a scheduling algorithm is needed to take advantage of the heterogeneity in service rate and arrival pattern [50], [51]. Further investigation and experimentation with scheduling algorithms that are both performance and power aware to schedule AI jobs on accelerator units with slicing are needed. In general, optimizing accelerator allocations involve multi-dimensional bin packing problems, as other resources such as CPU, memory, switches, and accelerator memory have to be considered, otherwise stranded resources and/or wasted energy may result [52].

As noted, experimental results of profiling power consumption as a function of utilization suggest that power increases linearly up to a certain utilization then it saturates, making the saturation region a desirable operating point. Selecting an optimal configuration depends on the workload mix as well as the overhead and penalty involved in conducting a change. We have conducted experimental research by running training jobs on a variety of MIG configurations and measured their corresponding running time and power consumption [49]. Characterizing the tradeoff between these two measures is crucial in selecting an optimal configuration for a given workload mix, as well as in job scheduling.

Pods are units of scheduling that consist of multiple containers, each placed on a set of resources in a node in the cluster. At a higher level, the scheduling of pods impacts the power consumption of the datacenter. With AI workloads and high energy accelerator units, the Kubernetes scheduler needs to be energy-aware, as well as aware of power capping and accelerator sharing [53]. Job schedulers, where a job is a collection of pods, need to consider the trade-off between energy consumption and job performance, among other factors. Scheduling across clusters, which may span different geographical areas, involves taking the carbon intensities into consideration, and not only energy. Optimized dispatching across multiple clusters may lead to delaying some jobs, within some specified timing constraints, to take advantage of periods of low carbon intensity at some geographical locations [54]. For inference workloads, hierarchical balancing across clusters and within a cluster, while considering workload characteristics, multiple locations with varying carbon intensities and multiple types of accelerators, is a challenging problem [55], [56].

VII. Summary and Open Questions

In this paper, we offer a vision of how to harness the power of specialization to reduce the energy and carbon associated with computing. We posit that in order to harness the full power of specialization, we must approach the problem area holistically across the system and software life-cycle and address the specific challenges that specialization introduces or exacerbates. We discuss broadly four problem areas: life cycle efficiency, sustainable design, sustainable operating system, and sustainable software management. In each of these areas we analyze the challenges and opportunities, and propose some direction for future research. This work is based on a collaboration program between Columbia University and IBM Research. Our hope is to inspire multi-disciplinary research, involving algorithms, systems, and AI to address an urgent need to significantly reduce energy consumption and carbon emission associated with computing in datacenters.

References

[1] (U. N. Press, New York, NY, USA). *Climate Change 'Biggest Threat Modern Humans Have Ever Faced'*. 2021. [Online]. Available: https://press.un.org/en/2021/sc14445.doc.htm
[2] X. Shao, Z. Zhang, P. Song, Y. Feng, and X. Wang, "A review of energy efficiency evaluation metrics for data centers," *Energy Build.*, vol. 271, Sep. 2022, Art. no. 112308. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378778822004790
[3] "Electricity 2024." 2024. [Online]. Available: https://www.iea.org/reports/electricity-2024
[4] R. Bommasani et al., "On the opportunities and risks of foundation models," 2022, *arXiv:2108.07258*.
[5] J. Devlin et al., "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[6] T. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.

[7] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," 2019, *arxiv.1906.02243*. [Online]. Available: http://

[8] C.-J. Wu et al., "Sustainable AI: Environmental implications, challenges and opportunities," in *Proc. Mach. Learn. Syst.*, 2022, pp. 795–813. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2022/file/462211f67c7d858f663355eff93b745e-Paper.pdf

[9] D. Patterson et al., "Carbon emissions and large neural network training," 2021, *arXiv:2104.10350*.

[10] X. Wang et al., "Energy and carbon considerations of fine-tuning BERT," 2023, *arXiv:2311.10267*.

[11] T. Eilam et al., "Towards a methodology and framework for AI sustainability metrics," in *Proc. 2nd Workshop Sustain. Comput. Syst.*, 2023, pp. 1–7.

[12] R. H. Dennard et al., "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE Solid-State Circuits Soc. Newslett.*, vol. 12, no. 1, pp. 38–50, 2007.

[13] D. M. Brooks et al., "Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors," *IEEE Micro*, vol. 20, no. 6, pp. 26–44, Jul./Aug. 2020.

[14] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC)*, 2014, pp. 10–14.

[15] L. P. Carloni, "The case for embedded scalable platforms," in *Proc. Design Autom. Conf.*, pp. 1–6.

[16] S. Kanev et al., "Profiling a warehouse-scale computer," in *Proc. 42nd Annu. Int. Symp. Comput. Archit.*, 2015, pp. 158–169.

[17] M. Amaral et al., "Kepler: A framework to calculate the energy consumption of Containerized applications," in *Proc. 16th Int. Conf. Cloud Comput.*, 2023, pp. 69–71.

[18] A. Gandhi et al., "Metrics for sustainability in data Centers," *SIGENERGY Energy Inform. Rev.*, vol. 3, no. 3, pp. 40–46, 2023. [Online]. Available: https://doi.org/10.1145/3630614.3630622

[19] "Columbia SLD Group." ESP. 2019. [Online]. Available: www.esp.cs.columbia.edu

[20] (Tech Radar, London, U.K.). *AMD, Meta are Working on Revolutionary Tech that Could Recycle Petabytes Worth of RAM*. 2023. [Online]. Available: https://www.techradar.com/pro/amd-meta-are-working-on-revolutionary-tech-that-could-recycle-petabytes-worth-of-ram-cxl-could-help-save-hyperscalers-tens-of-millions-of-dollars-while-improving-performance

[21] J. Lyu et al., "Myths and misconceptions around reducing carbon embedded in cloud platforms," in *Proc. HotCarbon Workshop*, 2023, pp. 1–7.

[22] D. S. Berger et al., "Design tradeoffs in CXL-based memory pools for public cloud platforms," *IEEE Micro*, vol. 43, no. 2, pp. 30–38, Apr. 2023.

[23] A. Rautakoura and T. Hamalainen, "Does SoC hardware development become agile by saying so: A literature review and mapping study," *ACM Trans. Embed. Comput. Syst.*, vol. 22, no. 3, Apr. 2023.

[24] P. Mantovani et al., "Agile SoC development with open ESP," in *Proc. 39th Int. Conf. Comput.-Aided Design (ICCAD)*, 2020, pp. 1–9. [Online]. Available: https://doi.org/10.1145/3400302.3415753

[25] T. Jia et al., "A 12nm agile-designed SoC for swarm-based perception with heterogeneous IP blocks," in *Proc. Eur. Solid State Circuits Conf. (ESSCIRC)*, 2022, pp. 269–272.

[26] M. Cassel dos Santos et al., "A 12nm Linux-SMP-capable RISC-V SoC with 14 accelerator types, distributed hardware power management and flexible NoC-based data orchestration," in *Proc. Int. Solid State Circuits Conf. (ISSCC)*, 2024, pp. 262–264.

[27] F. Fahim et al., "hls4ml: An open-source codesign workflow to empower scientific low-power machine learning devices," in *Proc. Tinyml Res. Symp.*, 2021, pp. 1–8.

[28] "hls4ml," [Online]. Available: https://fastmachinelearning.org/hls4ml

[29] M. Cassel dos Santos et al., "A scalable methodology for agile chip development with open-source hardware components," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2022, pp. 1–9.

[30] G. H. Loh and R. Swaminathan, "The next era for Chiplet innovation," in *Proc. Design Autom. Test Eur. Conf. Exhibit.*, 2023, pp. 1–6.

[31] M. Cochet et al., "BlitzCoin: Fully Decentralized hardware power management for accelerator-rich SoCs," in *Proc. Int. Symp. Comput. Archit. (ISCA)*, 2024.

[32] Y. Zhong et al., "XRP: In-kernel storage functions with eBPF," in *Proc. 16th USENIX Symp. Oper. Syst. Design Implement. (OSDI 22)*, 2022, pp. 375–393. [Online]. Available: https://www.usenix.org/conference/osdi22/presentation/zhong

[33] I. Zhang et al., "The Demikernel datapath OS architecture for microsecond-scale datacenter systems," in *Proc. 28th Symp. Oper. Syst. Princ.*, 2021, pp. 195–211. [Online]. Available: https://doi.org/10.1145/3477132.3483569

[34] G. Prekas, M. Kogias, and E. Bugnion, "ZygOS: Achieving low tail latency for microsecond-scale networked tasks," in *Proc. 26th Symp. Oper. Syst. Princ.*, 2017, pp. 325–341.

[35] "eBPF." https://ebpf.io/

[36] K. Kaffes et al., "Syrup: User-defined scheduling across the stack," in *Proc. ACM SIGOPS 28th Symp. Oper. Syst. Princ.*, 2021, pp. 605–620.

[37] J. T. Humphries et al., "ghOSt: Fast & flexible user-space delegation of Linux scheduling," in *Proc. 28th Symp. Oper. Syst. Princ.*, 2021, pp. 588–604.

[38] "XDP." https://www.iovisor.org/technology/xdp

[39] M. S. Brunella et al., "hXDP: Efficient software packet processing on FPGA NICs," in *Proc. Symp. Oper. Syst. Design Implement. (OSDI)*, 2020, pp. 973–990. [Online]. Available: https://www.usenix.org/conference/osdi20/presentation/brunella

[40] N. Hedam et al., "Delilah: EBPF-offload on computational storage," in *Proc. 19th Int. Workshop Data Manage. New Hardw.*, 2023, pp. 70–76.

[41] I. Zarkadas et al., "BPF-oF: Storage function pushdown over the network," 2023, *arXiv:2312.06808*.

[42] W. Bai et al., "Empowering azure storage with RDMA," in *Proc. 20th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2023, pp. 49–67. [Online]. Available: https://www.usenix.org/conference/nsdi23/presentation/bai

[43] B. Acun et al., "Carbon explorer: A holistic framework for designing carbon aware datacenters," in *Proc. Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, 2023, pp. 118–132. [Online]. Available: https://doi.org/10.1145/3575693.3575754

[44] T. Anderson et al., "Treehouse: A case for carbon-aware datacenter software," *SIGENERGY Energy Inform. Rev.*, vol. 3, no. 3, pp. 64–70, Oct. 2023. [Online]. Available: https://doi.org/10.1145/3630614.3630626

[45] J. Zhang et al., "Model-Switching: Dealing with fluctuating workloads in machine-learning-as-a-service systems," in *Proc. 12th USENIX Workshop Hot Topics Cloud Comput.*, 2020, pp. 1–8. [Online]. Available: https://www.usenix.org/conference/hotcloud20/presentation/zhang

[46] B. Li et al., "MISO: Exploiting multi-instance GPU capability on multi-tenant GPU clusters," in *Proc. 13th Symp. Cloud Comput.*, 2022, pp. 173–189. [Online]. Available: https://doi.org/10.1145/3542929.3563510

[47] F. Xu et al., "iGniter: Interference-aware GPU resource provisioning for predictable DNN inference in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 3, pp. 812–827, Mar. 2023.

[48] "NVIDIA multi-instance GPU user guide." 2022. [Online]. Available: https://docs.nvidia.com/datacenter/tesla/mig-user-guide/index.html

[49] C. Espenshade et al., "Characterizing training performance and energy for foundation models and image classifiers on multi-instance GPUs," in *Proc. 4th Workshop Mach. Learn. Syst.*, 2024, pp. 47–55. [Online]. Available: https://doi.org/10.1145/3642970.3655830

[50] Q. Weng et al., "Beware of fragmentation: Scheduling GPU-sharing workloads with fragmentation gradient descent," in *Proc. USENIX Annu. Tech. Conf.*, 2023, pp. 995–1008. [Online]. Available: https://ww.usenix.org/conference/atc23/presentation/weng

[51] P. Oberholzer, "Scheduling for MIG-capable GPUs: Accelerator-aware operating system scheduling," M.S. thesis, ETH Zurich, Dept. Comput. Sci., Zürich, Switzerland, 2021.

[52] J. R. Gunasekaran et al., "Cocktail: A multidimensional optimization for model serving in cloud," in *Proc. 19th USENIX Symp. Netw. Syst. Design Implement.*, 2022, pp. 1041–1057. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/gunasekaran

[53] "PEAKS: Power efficiency aware Kubernetes scheduler." 2024. https://github.com/sustainable-computing-io/peaks

[54] T. Bahreini, A. Tantawi, and A. Youssef, "A carbon-aware workload dispatcher in cloud computing systems," in *Proc. Int. Conf. Cloud Comput. (CLOUD)*, 2023, pp. 212–218.

[55] A. Dhakal, S. G. Kulkarni, and K. K. Ramakrishnan, "GSLICE: controlled spatial sharing of GPUs for a scalable inference platform," in *Proc. ACM Symp. Cloud Comput.*, 2020, pp. 492–506. [Online]. Available: https://doi.org/10.1145/3419111.3421284

[56] L. Wang et al., "Morphling: Fast, near-optimal auto-configuration for cloud-native model serving," in *Proc. ACM Symp. Cloud Comput.*, 2021, pp. 639–653. [Online]. Available: https://doi.org/10.1145/3472883.3486987