An Agile Pathway Towards Carbon-aware Clouds

Pratyush Patel
pratyush@cs.washington.edu
University of Washington
USA

Theo Gregersen
theoag@cs.washington.edu
University of Washington
USA

Thomas Anderson
tom@cs.washington.edu
University of Washington
USA

ABSTRACT

Climate change is a pressing threat to planetary well-being that can be addressed only by rapid near-term actions across all sectors. Yet, the cloud computing sector, with its increasingly large carbon footprint, has initiated only modest efforts to reduce emissions to date; its main approach today relies on cloud providers sourcing renewable energy from a limited global pool of options. We investigate how to accelerate cloud computing's efforts. Our approach tackles carbon reduction from a software standpoint by gradually integrating carbon awareness into the cloud abstraction. Specifically, we identify key bottlenecks to software-driven cloud carbon reduction, including (1) the lack of visibility and disaggregated control between cloud providers and users over infrastructure and applications, (2) the immense overhead presently incurred by application developers to implement carbon-aware application optimizations, and (3) the increasing complexity of carbon-aware resource management due to renewable energy variability and growing hardware heterogeneity. To overcome these barriers, we propose an agile approach that federates the responsibility and tools to achieve carbon awareness across different cloud stakeholders. As a key first step, we advocate leveraging the role of application operators in managing large-scale cloud deployments and integrating carbon efficiency metrics into their cloud usage workflow. We discuss various techniques to help operators reduce carbon emissions, such as carbon budgets, service-level visibility into emissions, and configurable-yet-centralized resource management optimizations.

CCS CONCEPTS

• **Computer systems organization** → **Cloud computing**.

KEYWORDS

cloud computing, carbon reduction, sustainability

ACM Reference Format:
Pratyush Patel, Theo Gregersen, and Thomas Anderson. 2023. An Agile Pathway Towards Carbon-aware Clouds. In *2nd Workshop on Sustainable Computer Systems (HotCarbon '23), July 9, 2023, Boston, MA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3604930.3605722

1 OVERVIEW

Nearly 1% of today's total global carbon emissions is attributable to datacenters [4]. This fraction is growing rapidly due to the increased demand for centralized computing, including recent advances in compute-intensive large language models (LLMs) [15, 28]. Rapid near-term actions are needed to reduce these emissions since carbon can otherwise accumulate and remain in the atmosphere for hundreds of years [63, 88]. One such action involves compliance with the Paris Agreement [64], which requires datacenters to reduce their emissions by 45% between 2020 to 2030 [88]. Driven by such political pressures, as well as social and market demands, several cloud providers and large-scale cloud users have made commitments to reduce their emissions and publish annual sustainability reports detailing their progress [32, 34, 48, 57, 71].

To offset or reduce datacenter emissions, cloud providers invest in renewable energy through long-term Power Purchase Agreements (PPAs) and time-based Renewable Energy Certificates (RECs) [4, 32]; they also incentivize cloud users to run applications in regions that primarily operate on renewables [31, 67]. These efforts help reduce the carbon intensity of the grid by funding sustainable energy generation and redirecting user compute demand towards greener energy sources. However, they incur three key limitations. First, since growth in hyperscale cloud energy demand far outpaces the rate of increase in renewable energy production [5, 6, 89], clouds running on renewables may displace other electricity consumers on the grid to carbon-intensive energy sources. Given the recent growth in demand for LLMs [15, 28], it may not even be feasible to source renewable energy for all datacenters in the near term. Second, even if providers match 100% of their annual or hourly energy demand with PPAs or RECs, clouds may still not be able to fully operate on renewable energy due to the supply variability of major renewable sources [19, 21, 92]. For example, solar and wind energy supplies are highly sensitive to weather patterns and the time of the day; consequently, their supply may not match the instantaneous cloud energy demand. Finally, using renewable energy for cloud operations does not address *embodied carbon*—the emissions from hardware manufacturing and supply chains—which constitutes a significant portion of the cloud carbon footprint [39, 61].

New approaches are needed to address these limitations, and we posit that providers alone can make limited progress. In particular, many carbon optimizations involve trade-offs that are opaque to providers due to the black-box nature of cloud user applications (e.g., deciding whether applications are delay tolerant to achieve higher carbon efficiency [93], or identifying application quality-of-service guarantees to perform efficient resource management [22]). Further, given that users dictate the cloud workload, they should also be given agency regarding their emissions.

Taking action to rapidly transition towards sustainable clouds therefore requires empowering cloud users with mechanisms to reduce both their energy and carbon footprints. Unfortunately, doing so is challenging because cloud users: (1) lack fine-grained visibility into and control over operational emissions, (2) lack visibility into

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

ACM ISBN 979-8-4007-0242-6/23/07.
https://doi.org/10.1145/3604930.3605722

datacenter hardware lifetimes and embodied emissions, (3) lack suitable development tooling and application deployment workflows for carbon reduction, and (4) encounter significant complexity in optimizing for carbon trade-offs due to renewable energy supply variability and growing cloud hardware heterogeneity.

Prior work argues that cloud providers could address some of these challenges by integrating a carbon API into the cloud abstraction to give users greater visibility and control over their emissions [13, 81]. Although powerful, this approach is complex for both providers and users because it requires carbon awareness to percolate throughout the application, platform, and hardware stacks. Specifically, it puts much of the complexity burden on application developers, who would need additional training and tooling to rewrite large-scale applications to make them carbon efficient [37, 38, 59]; further, it requires continuous development efforts since carbon trade-offs keep changing as technology evolves. Alas, developer time can already pose a bottleneck for business operations. Rewriting applications may not even be a viable option for developers because many cloud users deploy large-scale microservices that use complex components as black-box solutions [26, 87]. Thus, even if cloud providers were willing to expose low-level infrastructure details and a carbon API to users, this approach would likely take significant time, effort, and care to deploy, time that the planet may not have.

To address the urgency of reducing carbon emissions amid the rapid evolution of clean datacenter technologies, we suggest an agile approach [2]. We advocate that **the most pragmatic pathway towards achieving carbon-aware clouds is one that works within existing business practices while enabling swift and progressive reduction of carbon emissions.** Based on this philosophy, we recommend three measures to transition towards a carbon-aware cloud abstraction.

R1: Federate Responsibility for Carbon Efficiency into Existing Organizational Workflows. First, large-scale cloud users should leverage their existing organizational structure to integrate carbon efficiency into their current workflow. Rather than putting the entire burden of carbon awareness/reduction on developers, we argue that application operators are well positioned to monitor and navigate the complex trade-offs between cost, performance, reliability, and carbon at deployment time while ensuring that business-critical application objectives are met. Meanwhile, cloud providers are better suited to implement carbon-aware resource management measures and expose them to application operators given their better visibility into their own infrastructure and services. Furthermore, to meet carbon reduction targets, cloud users should implement a top-down carbon budget for the organization that is allocated to applications by project managers and enforced by operations teams: carbon budgets provide clear carbon goals to all stakeholders and align well with climate policy [64].

R2: Provide Actionable Visibility into Emissions. Second, to make more carbon-efficient decisions, cloud providers should empower application operators with coarse-grained visibility into the carbon footprints of their deployments; different providers should further collaborate to standardize reported metrics and accounting methodologies. Doing so casts carbon as an optimization metric that application operators can balance alongside traditional application objectives. It also helps operators define best practices for project managers to choose between cloud platforms and for application developers to select carbon-efficient services when they build or optimize software. For example, operators could help pinpoint which application services are "carbon hotspots" for developers to optimize; they could also help create an emissions ranking between different application service alternatives to guide developer decisions. Over time, as carbon optimizations become more nuanced, providers can improve the accuracy and granularity of their reported carbon metrics.

R3: Centralize Configurable Carbon Optimization Mechanisms. Finally, to simplify and enable progressive carbon reduction across users, cloud providers should centralize carbon optimization implementations as much as possible and offer them as configurable knobs to application operators. Operators could then reason about and expose application characteristics, possibly in real time, so the cloud provider could better schedule resources. These knobs would initially be coarse grained: for example, the main knob available today controls whether applications are run in a region powered by green energy if network latency from that region is acceptable [31]. As provider implementations mature, they can expose finer-grained knobs to operators (and eventually to developers) who would incrementally reduce emissions by making applications partially or fully carbon aware. For example, "green VMs" could become alternatives to spot VMs that are spun-up to run delay-tolerant workloads only when renewable energy is available [69, 93].

2 CARBON AWARENESS CHALLENGES

We now elaborate on the challenges that motivate the agile reduction of cloud carbon emissions.

2.1 Visibility into Carbon Emissions

Providing visibility into emissions can motivate and guide user actions [4, 9, 29]. However, carbon fundamentally differs from traditional application metrics like performance, availability, or correctness [10]. Cloud users can independently monitor and validate traditional metrics, which are typically exposed by the application itself. In contrast, carbon metrics, such as energy usage and emissions, must be externalized by the cloud provider. Addressing this requirement raises the following challenges due to the multi-tenant nature of the cloud.

C1: Operational Emissions Accounting. Visibility into operational carbon emissions requires user-level energy measurements; however, taking these measurements in a fine-grained manner is not possible with today's hardware interfaces. Most modern servers are equipped with full-node power monitoring through IPMI [45]. At the component level, energy can be monitored for CPU sockets and DRAM using the RAPL interface [44] and for GPUs using vendor-provided tools [8, 62]. However, most other server components, e.g., disks, NICs, and fans, typically do not expose energy measurement interfaces, making it difficult to get fine-grained visibility into user emissions. Furthermore, exposing raw energy counters directly to cloud users raises safety concerns since they are susceptible to side-channel attacks [52]. Even if there were safe energy measurement interfaces, it is difficult to map server- or component-level

Challenge	Today's Cloud	Ecovisor	Our Proposal R1: Federated Responsibility	Our Proposal R2: Actionable Visibility	Our Proposal R3: Centralized Management
C1: Operational carbon accounting	◐	●		●	◐
C2: Embodied carbon accounting	◐			●	◐
C3: Trust issues	◐		◐		
C4: Application complexity			◐		◐
C5: Developer tooling			◐	◐	
C6: Renewable energy variability	◐	●		◐	●
C7: Hardware heterogeneity				◐	●

Table 1: The extent to which different proposals address carbon-awareness challenges. Mostly addresses (●). Partially addresses (◐).

power measurements to consistent user-level emissions on shared, multi-tenant servers [10]. For example, since modern servers are not energy proportional [12], the same virtual machine (VM) running on an otherwise idle machine uses more energy than if it were colocated with other VMs, which the user has no control over.

C2: Embodied Emissions Accounting. Embodied carbon accounting is even more challenging since it incurs multi-tenancy in both space and time. Not all providers expose embodied emissions to users today [79], and those that do expose it only coarsely via post-hoc reports [54, 58]; in fact, cloud users can effectively hide their Scope 3 emissions, reporting on which is not required under existing standards [61, 68]. Since server lifetimes are not known in advance, it is unclear how to effectively account for and expose embodied emissions to cloud users. Current proposals account for embodied emissions based on electricity use prorated over anticipated hardware lifetimes [23, 38], factoring in the fraction of compute reserved [38, 85], or based on normalized cloud usage cost [54]. However, these accounting approaches fail to incentivize efforts to extend server lifetimes or use older hardware, which can help to reduce emissions [14, 18, 86].

C3: Trust Issues. The validity of cloud emissions metrics relies entirely on the provider's trustworthiness. Since providers can use cloud carbon efficiency as a competitive advantage to attract environmentally-conscious users [17], it is important to ensure that their emissions reporting is standardized and comparable. However, today's carbon footprint reports from cloud providers often lack details about accounting methodology, are unverified, and/or are subject to change [23, 58]. Furthermore, each provider uses their own carbon reporting methodology which makes it difficult for users to correctly choose carbon-efficient cloud platforms. To increase credibility and standardize a carbon reporting framework, providers should share their methodology with the community and cross-validate it with third-party stakeholders, which can facilitate greater carbon reduction through competition [82]. Currently, such transparency is not widespread.

2.2 Carbon-aware Software Development

Carbon-aware software optimization typically leverages trade-offs being built into the application; for example, since 4G networks are more carbon intensive than WiFi, websites serving mobile networks could be designed to operate in a low carbon mode [43]. However, such optimizations are difficult to implement in large-scale cloud applications due to application complexity and developer tooling, which we now explore.

C4: Application Complexity. Large-scale user applications are likely to have the most significant impact on cloud carbon emissions. Such applications are complex and can be on the scale of millions of lines of code, meaning that developers typically have at most a partial understanding of the system. Cloud users manage this complexity by building hundreds or thousands of microservices across several different development teams [41, 87]; these microservices often rely on legacy code, external services, or libraries that must be treated as black boxes by developers [30]. Consequently, it is difficult, if not impossible, for developers to modify every application component to become carbon efficient in the near term.

C5: Lack of Developer Tooling. Even developers who want to write carbon-aware code today have little cloud-based tooling support available to do so. Developers cannot easily interpret or optimize the carbon efficiency of their cloud applications beyond adhering to high-level software design guidelines [37, 38, 59]; instead, they must rely on cloud providers to share energy and emissions metrics [56, 78]. Third-party tools like Scaphandre [42] and Cloud Carbon Footprint [85] offer some insight into carbon emissions, but they are still in early stages, use coarse-grained estimations, and cannot be easily integrated with large-scale user applications.

To examine how these issues might manifest, we review insights from the field of security and privacy. Specifically, many developers seeking to integrate privacy protections into the development cycle perceive them to contradict system requirements, have trouble verifying the correctness of their work, and/or lack knowledge on relevant practices [72]. The disconnect between modern development workflow and methodology adds further difficulty [50]. As a result, organizations trying to implement privacy engineering tend to face some developer resistance [40, 73]. The perceived usefulness and complexity of an approach particularly influence its adoption [73], with additional system overhead [80] and a lack of supportive tooling also having an impact. Viewing carbon efficiency as a non-functional requirement akin to privacy suggests that its integration will likely face similar development challenges.

2.3 Carbon-aware Resource Management

Carbon must be optimized alongside existing business-critical application objectives, i.e., cost, performance, availability, security, and more [60]. Yet, even with visibility into emissions, cloud users find it complex to develop mechanisms to reduce them. Furthermore, each user must develop their own carbon efficiency mechanisms which leads to redundant resource management optimizations. We discuss the two main sources of complexity below.

C6: Renewable Energy Variability. Since many major renewable energy sources today are relatively nascent and depend on the environment (i.e., solar and wind), there is considerable variability in the carbon intensity of the energy supply [19, 90, 92]. However, such variability does not imply variability in cloud energy usage. At a datacenter scale, the average difference between minimum and maximum energy demand is roughly 4%, a relatively small delta compared to changes in renewable energy supply [3]. To improve carbon efficiency, cloud stakeholders must therefore match supply variability to workloads that can adapt accordingly. Deciding how to schedule different applications to gain the highest marginal utility from variable renewable energy supply poses a challenging optimization problem [3, 69, 93], especially because it requires handling different timescales (e.g., solar supply varies with day/night cycles, but workload changes may occur within seconds). Geography requires consideration too; for instance, due to energy supply variability, it may be more carbon efficient to schedule workloads in a non-local region (e.g., by chasing the sun), as long as the transfer overheads are acceptable.

C7: Hardware Heterogeneity. Heterogeneity in the cloud manifests in several different dimensions spanning compute, memory, networks, and storage. For example, compute heterogeneity manifests across compute generations due to server lifetime extensions (e.g., older vs. newer CPUs) [86]; across compute types to achieve different performance and energy efficiency trade-offs (e.g., CPUs vs. GPUs vs. FPGAs; Intel vs. ARM vs. AMD CPUs) [20, 66, 77]; and across servers of the same type due to new failure mechanisms such as fail in place, where servers start to differ at a component level over time (e.g., CPUs with and without cache failures) [53]. Importantly, each compute backend offers different, yet often complementary, carbon trade-offs, and these trade-offs may change over time. For example, old CPUs past their standard lifetimes may be considered carbon efficient if their embodied emissions are already accounted for, but they may perform slower, use more energy, and be less reliable than newer counterparts [14, 83, 86]. Similarly, FPGAs, typically much more energy efficient than CPUs, spin up slower in response to bursty workloads, requiring substantial idle capacity to be provisioned to meet latency SLOs [66]. Increase in domain-specific architectures further complicates compute heterogeneity, with ASICs such as TPUs [49, 67] offering much better operational efficiency with the trade-off of a narrow case for allocation and potentially higher embodied carbon costs. Similar trade-offs also apply to other hardware technologies given recent advancements in memory disaggregation, flash storage, smartNICs, and optical networking. Deciding how best to schedule computation on increasingly heterogeneous resources to minimize carbon emissions while meeting other application objectives is another challenging and multi-dimensional optimization problem.

2.4 Summary

Importantly, the aforementioned challenges are rapidly evolving. Since carbon-aware software development changes will take time to design and implement, we believe that transparent carbon-aware resource management that leverages the role of application operators is a more likely near-term path to reduce carbon. Meanwhile, as developers gradually make applications more carbon aware, these resource management optimizations will in turn be impacted. Hence, from a cloud user standpoint, we seek to facilitate an agile transition towards carbon efficiency by integrating it into their cloud usage workflow [2].

Our proposal for a carbon-aware cloud abstraction adopts this reasoning to address the challenges in an agile manner [2]. Table 1 summarizes the extent to which it does so, with comparisons to today's cloud and related work (Ecovisor [81]). Current cloud approaches address operational carbon accounting (C1) and embodied carbon accounting (C2) at a coarse level with post-hoc reports [23, 54, 58]; mechanisms such as audits provide limited support to address trust issues (C3) [1, 58]; and hourly RECs partially and indirectly handle renewable energy variability (C6) [32]. Ecovisor provides fine-grained emissions visibility for operational carbon accounting (C1) and renewable energy variability (C6). Both today's cloud and Ecovisor do not address application complexity (C4), developer tooling (C5), and hardware heterogeneity (C7). The following sections detail how implementing our proposal to federate carbon awareness across the organization addresses these challenges more comprehensively.

3 VIEWING CARBON AS AN OPERATIONS CONCERN

Large-scale cloud user organizations typically consist of managers, developers, and operators who collaborate to deploy applications with specific Service Level Objectives (SLOs) [16, 60]. Each role has different responsibilities and specializations: managers allocate budgets, coordinate between teams, and chart out application SLOs; developers focus primarily on implementing application functionality, optimizations, and ensuring correctness; and operators address deployment, monitoring, and reconfiguration to ensure that applications meet their SLOs[1].

To help application operators meet deployment objectives, cloud providers dedicate their own operations teams, such as Customer Reliability Engineers [70], to perform in-depth service monitoring and provide debugging and configuration guidance. Cloud operators may further communicate with cloud infrastructure teams about user needs to motivate new features and improvements to their underlying platforms.

Because application operators and cloud operators constitute a narrow-waist interface between cloud users and providers, we argue that they are best suited to introduce carbon awareness in the cloud (**R1**). Viewing carbon awareness as an operators-first responsibility integrates well into the typical cloud user workflow. For example, project managers could negotiate with operators to procure cloud resources while staying under carbon budgets; developers could build carbon-efficient optimizations by discussing design choices with operators since they have more insight into runtime application behavior; and application and cloud operators could provide each other greater visibility into carbon metrics and application characteristics to collaboratively make carbon-efficient resource management decisions.

[1] In some teams, the same individual might serve as both developer and operator. We discuss the roles separately for clarity.

3.1 Carbon Budgets

To meet specific carbon reduction targets, we propose that organizations define and adhere to an internal carbon budget over a specific time horizon (e.g., annual). Project managers should plan deliverables in accordance to this budget, while operations teams help monitor and enforce the budget. Carbon budgets are similar to financial budgets: a single quota is defined for an entire organization and subdivided for individual projects and/or teams over specific durations.

We believe that implementing well-defined carbon budgets would offer three key benefits. First, they would provide a clear vocabulary and common objective for cloud organizations, teams, and individuals. Second, they would keep carbon goals orthogonal to traditional SLOs, like performance and availability, which are typically defined based on business needs [16]. Third, they would align well with the carbon budgets commonly used in climate policy and hence provide a clear tracker for carbon reduction progress [63], e.g., relative to Paris Agreement requirements for the Information and Communication Technology (ICT) industry to reduce their overall emissions by 45% between 2020–2030 [88].

Note that carbon budgets are complementary to carbon fees that organizations like Microsoft internally impose on their teams: carbon fees serve as a "tax" that helps fund sustainability initiatives based on carbon emissions [94], whereas carbon budgets provide an organizational target/incentive for individual groups to make decisions about how to reduce their carbon emissions. Carbon budgets also differ from SLOs because they impact neither end-user experience nor correctness [16]. We encourage the research community to commit to studying the perception and effectiveness of carbon budgets in cloud organizations; over time, this would help refine and improve the budgeting process.

4 TAKING ACTION TO MAKE EMISSIONS MORE VISIBLE

Large-scale cloud users deploy thousands of microservices to manage their application needs [41, 87], meaning that there is substantial complexity even in identifying and optimizing the most carbon-intensive services. We therefore propose that cloud providers give application operators service-level visibility into carbon emissions and integrate emissions data into their typical cloud usage interfaces (e.g., monitoring dashboards [35, 36, 74] and container registries [24, 55, 76]). Doing so would help operators rapidly and easily develop techniques to monitor and reduce service emissions at scale (**R2**).

4.1 Service-level Carbon Metrics

Providers should expose real-time, service-level emissions metrics to operators to drive runtime carbon optimizations. For cloud-native microservices, such metrics can be integrated into the operator workflow by monitoring emissions in a black-box manner via sidecar proxies [25, 27] and displaying them using standard observability dashboards [35, 51, 74, 84]. For VM deployments, providers could expose emissions via a separate remote procedure call (RPC) or a hypervisor-mediated cloud API.

Dynamic service-level visibility into emissions would help for many reasons. First, it would let operators track whether their deployed services could meet their carbon budgets, even with constant development churn. Second, since emissions are exposed in a black-box manner through sidecars and service meshes, it would help operators implement carbon-aware resource optimizations without necessarily involving developers. Finally, it would help operators identify "carbon hotspots" on which to focus resource management (and optionally, development) efforts to reduce carbon emissions, making remediation efforts more effective and efficient.

Exposing accurate carbon metrics is challenging, as discussed in § 2.1. However, since carbon is a non-functional requirement, we maintain that accurate metrics are less necessary than approximate ones at present to help operators reduce emissions. Over time, as carbon optimization becomes more nuanced, providers could implement more accurate and safer carbon monitoring. Below, we discuss initial strategies to measure and export service-level emissions.

Operational Emissions. We propose two methods to track operational emissions: *measurement based* and *model based.* Under the measurement-based approach, server power measurements (via IPMI [45] or RAPL [44]) would be aggregated across coarse-grained time intervals and prorated according to component-level utilizations. For the model-based approach, performance counters for different services would be translated into energy measurements via a pretrained model [10]. In both cases, energy values would then be translated into carbon intensities using grid or datacenter-level carbon APIs [19, 81, 92]. These approaches incur different trade-offs. The measurement-based approach is likely to be more accurate, but energy counters may not be available on all machines/components. The model-based approach does not depend on hardware energy counters, but it may be less accurate and not scale well to cloud-scale server heterogeneity. Further research is needed to determine whether either poses a practical security risk due to side channels [52].

Embodied Emissions. We propose amortizing the embodied carbon of servers over their expected lifetimes by following the depreciation model from financial accounting; that is, a server in its first year of use should incur higher embodied carbon per unit time than in its fourth year. Beyond the expected lifetimes, servers should incur no embodied carbon cost [83]. The same strategy could also be applied at the component level, since different server components have different expected lifetimes [53]. Our approach incentivizes the use of older hardware, thereby promoting hardware reuse. We note that older hardware might have a higher failure probability [53], implying a trade-off between carbon efficiency and reliability under our approach. Additional study is needed to investigate this trade-off and expose its implications to cloud users.

Trust Issues. Navigating trust is complex. We believe that a combination of legal regulations, audits, and public opinion will motivate cloud provider accountability. Third-party audits are already common for verifying sustainability practices; the same could be applied to verifying emissions measurements after making the methodology public [1, 54]. The research community could contribute a verification or attestation mechanism [33] so that users need not trust providers to report valid emissions metrics. If trust lapses, users could consider repatriating to private clouds for greater visibility and control over carbon reduction. While this might be cost effective for some cloud users today [91], it is unlikely to be a

carbon-efficient decision overall since it does not benefit from at-scale optimizations [17].

4.2 Carbon Knowledge Repository

Since carbon benchmarks are not yet common, operators in large-scale organizations would also benefit from static visibility into the energy usage of common services like key-value stores or machine learning models. Specifically, we propose that services expose their expected energy usage under standardized configurations via a shared carbon knowledge repository. Doing so would enable application operators and developers deploying new services to make informed initial choices about which service alternatives may be preferable. For instance, developers using machine learning serving frameworks would benefit from knowing the energy efficiencies of different models alongside their accuracy and performance.

We envision that such carbon knowledge repositories would be a collaborative effort across different cloud users and integrated into cloud-native container registries, image repositories, or machine learning model zoos [24, 55, 76, 95]. For uniformity, we expect that each carbon repository would be standardized with fixed benchmarks and hardware configurations, appropriate to the service type; deciding upon a set of standard configurations remains an open problem that could be guided by collaboratively-developed open-source benchmarks. Alternatively, cloud users could also create an internal repository of carbon emissions for their own service deployments in case of sensitive data/configurations. For willing users, request-level energy usage could be pulled directly through the metrics measurement interfaces described in § 4.1.

5 CENTRALIZING CARBON REDUCTION

Finally, to simplify operator efforts for cloud-managed services, we propose that providers centralize carbon-aware resource management mechanisms and expose configurable knobs to operators (**R3**). Optimizing cloud-managed services using black-box approaches quickly runs into limitations since many carbon optimizations involve multidimensional trade-offs beyond provider purview. For example, older hardware may run services slower and less reliably but offer higher carbon efficiency [83]; carbon-aware scheduling may delay workloads so they run during low carbon intensity time periods [69, 93]. To communicate these trade-offs, operators should be able tweak static or dynamic knobs that cloud providers can rely upon to make better carbon-aware optimizations. Centralizing optimizations particularly benefits agility by eliminating redundant optimization efforts across different services and across operators.

5.1 Static Optimizations

Operators should be able to communicate known static trade-offs to providers upon service creation or deployment. To implement static hints, providers could simply require operators to pass an appropriate flag when deploying cloud-managed microservices. For example, operators could indicate that a service is delay tolerant, letting providers schedule it when renewable energy is available [93]. Alternatively, providers could create new cloud platform offerings like "green VMs" which could be carbon-aware alternatives to spot VMs or harvest VMs that are spun up only when the carbon intensity of energy is low [7, 75]. Although carbon-aware scheduling already exists for internal workloads in some clouds today [69], we note that building such static optimizations into cloud-managed services enables users to benefit, as well.

5.2 Dynamic Eco Modes

Inspired by eco mode dials in modern cars [47], we propose that operators be able to dynamically specify simple hints to help the cloud platform improve carbon efficiency. For example, operators could signal that a service has latency slack, letting providers run it at lower clock frequencies (reducing operational emissions) or on older CPUs (reducing embodied emissions). Developers can further utilize eco modes to implement carbon-efficient application functionality, such as by using different eco modes to serve machine learning models that offer different accuracy, performance, and efficiency trade-offs for the same application task.

To enable eco mode settings in the cloud, providers must implement a communication interface between the application service and the cloud platform. As a first step, most existing cloud monitoring systems allow specification of custom metrics, and this interface could be extended to communicate eco modes. In the future, cloud providers could expose a dedicated eco mode interface to operators.

Each eco mode could capture different carbon trade-offs, and multiple modes could be combined for greater carbon reduction. For example, a delay-tolerant eco mode might use renewable-aware scheduling or lower compute clock frequencies [65]. A reliability-tolerant eco mode might schedule applications on older hardware generations to reduce carbon emissions [83, 86]. A balanced eco mode could schedule latency-sensitive applications interchangeably on heterogeneous compute workers to strike a middle ground between their spin-up latency, cost, and energy efficiency [11, 66].

5.3 User Incentives

To help operators make informed decisions about available static optimizations and eco mode trade-offs, providers could monitor historical user workloads, use simple simulators/predictors to determine potential carbon savings, and display savings on operator dashboards. For example, if shown that adding new FPGA implementations to existing CPU services could dramatically reduce carbon emissions while only slightly increasing costs [66], operators might be better positioned to make such recommendations within their enterprises based on their carbon and financial budgets. To motivate operators to enable carbon-reduction optimizations, providers could also introduce economic incentives by leveraging lowered operational costs due to reduced energy and power consumption. Prior work on market-mechanism-based power capping is a good starting point for future inquiry in this direction [46].

ACKNOWLEDGMENTS

We thank Sandy Kaplan for detailed feedback on our writing. We thank Esha Choukse, Tapan Chugh, Siddharth Gupta, Katie Lim, Bobbie Manne, Ashlie Martinez, Pulkit Misra, Bichlien Nguyen, Kushal Patel, and Akshitha Sriraman for insightful discussions that helped shape our perspective on sustainable clouds. This work is supported by NSF CNS-2104548, VMware, and Cisco Systems.

REFERENCES

[1] 3Degrees. 2022. Google Cloud Services Carbon Footprint Methodology Review. Retrieved May 20, 2023 from https://services.google.com/fh/files/misc/3degrees_cloud_services_review_statement_final.pdf.

[2] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, and Juhani Warsta. 2017. Agile Software Development Methods: Review and Analysis. *arXiv preprint arXiv:1709.08439*.

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

[4] International Energy Agency. 2022. Data Centres and Data Transmissions Networks. Retrieved May 20, 2023 from https://www.iea.org/reports/data-centres-and-data-transmission-networks.

[5] International Energy Agency. 2022. Global Data Centre Energy Demand by Data Centre Type. Retrieved May 21, 2023 from https://www.iea.org/data-and-statistics/charts/global-data-centre-energy-demand-by-data-centre-type.

[6] International Energy Agency. 2022. Renewable Electricity. Retrieved May 21, 2023 from https://www.iea.org/reports/renewables-2022/renewable-electricity.

[7] Pradeep Ambati, Íñigo Goiri, Felipe Frujeri, Alper Gun, Ke Wang, Brian Dolan, Brian Corell, Sekhar Pasupuleti, Thomas Moscibroda, Sameh Elnikety, Marcus Fontoura, and Ricardo Bianchini. 2020. Providing SLOs for Resource-Harvesting VMs in Cloud Platforms. *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.

[8] AMD. ROCm Open Software Platform for GPU Compute. Retrieved May 20, 2023 from https://www.amd.com/en/graphics/servers-solutions-rocm.

[9] Nina Amenta and Angela Sanguinetti. 2020. Adding Carbon to the Equation in Online Flight Search. *UC Davis: National Center for Sustainable Transportation*.

[10] Vaastav Anand, Zhiqiang Xie, Matheus Stolet, Roberta De Viti, Thomas Davidson, Reyhaneh Karimipour, Safya Alzayat, and Jonathan Mace. 2022. The Odd One Out: Energy is not like Other Metrics. *Workshop on Sustainable Computer Systems (HotCarbon)*.

[11] Thomas Anderson, Adam Belay, Mosharaf Chowdhury, Asaf Cidon, and Irene Zhang. 2022. Treehouse: A Case For Carbon-Aware Datacenter Software. *Workshop on Sustainable Computer Systems (HotCarbon)*.

[12] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. 2018. The Datacenter as a Computer: Designing Warehouse-Scale Machines. *Synthesis Lectures on Computer Architecture*.

[13] Bashir, Noman and Guo, Tian and Hajiesmaili, Mohammad and Irwin, David and Shenoy, Prashant and Sitaraman, Ramesh and Souza, Abel and Wierman, Adam. 2021. Enabling Sustainable Clouds: The Case for Virtualizing the Energy System. *ACM Symposium on Cloud Computing (SoCC)*.

[14] Rabih Bashroush, Nour Rteil, Rich Kenny, and Astrid Wynne. 2020. Optimizing Server Refresh Cycles: The Case for Circular Economy with an Aging Moore's Law. *IEEE Transactions on Sustainable Computing*.

[15] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*.

[16] Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy. 2016. *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, Inc.

[17] Daniel Bizo. 2019. The Carbon Reduction Opportunity of Moving to Amazon Web Services.

[18] Rani Borkar. 2022. Learn How Microsoft Circular Centers are Scaling Cloud Supply Chain Sustainability. Retrieved May 20, 2023 from https://azure.microsoft.com/en-us/blog/learn-how-microsoft-circular-centers-are-scaling-cloud-supply-chain-sustainability/.

[19] California ISO. Retrieved May 20, 2023 from https://www.caiso.com/.

[20] Adrian Caulfield, Eric Chung, Andrew Putnam, Hari Angepat, Jeremy Fowers, Michael Haselman, Stephen Heil, Matt Humphrey, Puneet Kaur, Joo-Young Kim, Daniel Lo, Todd Massengill, Kalin Ovtcharov, Michael Papamichael, Lisa Woods, Sitaram Lanka, Derek Chiou, and Doug Burger. 2016. A Cloud-scale Acceleration Architecture. *International Symposium on Microarchitecture (MICRO)*.

[21] Jacques De Chalendar. 2019. Why '100% Renewable Energy' Pledges are Not Enough. Retrieved May 20, 2023 from https://www.ft.com/content/d75f49d0-103f-11ea-a225-db2f231cfeae.

[22] Shuang Chen, Christina Delimitrou, and José F Martınez. 2019. PARTIES: QoS-aware Resource Partitioning for Multiple Interactive Services. *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[23] Google Cloud. Carbon Footprint Reporting Methodology. Retrieved May 20, 2023 from https://cloud.google.com/carbon-footprint/docs/methodology.

[24] Docker. Docker Hub Container Image Library. Retrieved May 20, 2023 from https://hub.docker.com/.

[25] NGINX Docs. Sidecar Proxy Injection. Retrieved May 20, 2023 from https://docs.nginx.com/nginx-service-mesh/guides/inject-sidecar-proxy/.

[26] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. 2017. Microservices: Yesterday, Today, and Tomorrow. *Present and Ulterior Software Engineering*.

[27] Envoy Proxy. Retrieved May 20, 2023 from https://www.envoyproxy.io/.

[28] International Institute for Strategic Studies. 2023. Large Language Models: Fast Proliferation and Budding International Competition. *Strategic Comments*.

[29] Jon Froehlich, Leah Findlater, and James Landay. 2010. The Design of Eco-feedback Technology. *SIGCHI Conference on Human Factors in Computing Systems (CHI)*.

[30] Andrei Furda, Colin Fidge, Olaf Zimmermann, Wayne Kelly, and Alistair Barros. 2017. Migrating Enterprise Legacy Source Code to Microservices: On Multitenancy, Statefulness, and Data Consistency. *IEEE Software*.

[31] Steren Giannini. 2021. Helping You Pick the Greenest Region for Your Google Cloud Resources. Retrieved May 20, 2023 from https://cloud.google.com/blog/topics/sustainability/pick-the-google-cloud-region-with-the-lowest-co2.

[32] Google. 2018. Moving Toward 24x7 Carbon-Free Energy at Google Data Centers: Progress and Insights.

[33] Google. 2022. Remote Attestation of Disaggregated Machines. Retrieved May 20, 2023 from https://cloud.google.com/docs/security/remote-attestation.

[34] Google. Sustainability Reports. Retrieved May 20, 2023 from https://sustainability.google/reports/.

[35] Google Cloud. Operations Suite. Retrieved May 20, 2023 from https://cloud.google.com/products/operations.

[36] Grafana: The Open Observability Platform. Retrieved May 20, 2023 from https://grafana.com/.

[37] Green Software Foundation. Retrieved May 20, 2023 from https://greensoftware.foundation/.

[38] Green Software Foundation. 2023. Software Carbon Intensity (SCI) Specification. Retrieved May 20, 2023 from https://github.com/Green-Software-Foundation/sci.

[39] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. 2021. Chasing Carbon: The Elusive Environmental Footprint of Computing. *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*.

[40] Irit Hadar, Tomer Hasson, Oshrat Ayalon, Eran Toch, Michael Birnhack, Sofia Sherman, and Arod Balissa. 2018. Privacy by Designers: Software Developers' Privacy Mindset. *Empirical Software Engineering*.

[41] Harris, Chandler. Microservices vs. Monolithic Architecture. Retrieved May 20, 2023 from https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith.

[42] hubblo-org. Scaphandre. Retrieved May 20, 2023 from https://github.com/hubblo-org/scaphandre.

[43] Asim Hussain. 2020. Carbon-Aware vs. Carbon-Efficient Applications. Retrieved May 20, 2023 from https://devblogs.microsoft.com/sustainable-software/carbon-aware-vs-carbon-efficient-applications/.

[44] Intel. 2023. Intel® 64 and IA-32 Architectures Software Developer's Manual.

[45] Intel, Hewlett Packard, NEC, and Dell. 2013. Intelligent Platform Management Interface Specification.

[46] Mohammad A Islam, Xiaoqi Ren, Shaolei Ren, Adam Wierman, and Xiaorui Wang. 2016. A Market Approach for Handling Power Emergencies in Multi-tenant Data Center. *IEEE International Symposium on High Performance Computer Architecture (HPCA)*.

[47] Yavor Ivanov, Rosen Ivanov, Georgi Kadikyanov, Gergana Staneva, and Igor Danilov. 2019. A Study of the Fuel Consumption of Hybrid Car Toyota Yaris. *Transport Problems*.

[48] Joppa, Lucas. 2021. Made to Measure: Sustainability Commitment Progress and Updates. Retrieved May 20, 2023 from https://blogs.microsoft.com/blog/2021/07/14/made-to-measure-sustainability-commitment-progress-and-updates/.

[49] Norman P. Jouppi, Doe Hyun Yoon, Matthew Ashcraft, Mark Gottscho, Thomas B. Jablin, George Kurian, James Laudon, Sheng Li, Peter Ma, Xiaoyu Ma, Thomas Norrie, Nishant Patil, Sushma Prasad, Cliff Young, Zongwei Zhou, and David Patterson. 2021. Ten Lessons from Three Generations Shaped Google's TPUv4i. *International Symposium on Computer Architecture (ISCA)*.

[50] Blagovesta Kostova, Seda F. Gürses, and Carmela Troncoso. 2020. Privacy Engineering Meets Software Engineering. On the Challenges of Engineering Privacy ByDesign. *ArXiv*, abs/2007.08613.

[51] Linkerd. Retrieved May 20, 2023 from https://linkerd.io/.

[52] Moritz Lipp, Andreas Kogler, David Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. 2021. PLATYPUS: Software-based Power Side-channel Attacks on x86. *IEEE Symposium on Security and Privacy (S&P)*.

[53] Jialun Lyu, Daniel S. Berger, Marisa You, Celine Irvene, Mark Jung, Tyler Narmore, Jacob Shapiro, Luke Marshall, Savyasachi Samal, Ioannis Manousakis,

Ashish Raniwala, Ricardo Bianchini, and Bianca Schroeder. 2023. Hyrax: Fail-in-Place Operation in Cloud Platforms. *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.

[54] Microsoft. 2021. A New Approach for Scope 3 Emissions Transparency. Retrieved May 20, 2023 from https://go.microsoft.com/fwlink/p/?linkid=2161861.

[55] Microsoft. Azure Container Registry. Retrieved May 20, 2023 from https://azure.microsoft.com/en-us/products/container-registry/.

[56] Microsoft. Emissions Impact Dashboard. Retrieved May 20, 2023 from https://www.microsoft.com/en-us/sustainability/emissions-impact-dashboard.

[57] Microsoft. 2022. Environmental Sustainability Report. Retrieved May 20, 2023 from https://aka.ms/SustainabilityReport2022.

[58] Microsoft. 2023. Microsoft Cloud for Sustainability API Calculation Methodology. Retrieved May 20, 2023 from https://learn.microsoft.com/en-us/industry/sustainability/api-calculation-method.

[59] Microsoft. The Principles of Sustainable Software Engineering. Retrieved May 20, 2023 from https://aka.ms/sse/learn.

[60] Jeffrey Mogul and John Wilkes. 2019. Nines are Not Enough: Meaningful Metrics for Clouds. *Workshop on Hot Topics in Operating Systems (HotOS)*.

[61] David Mytton. 2020. Hiding Greenhouse Gas Emissions in the Cloud. *Nature Climate Change*.

[62] NVIDIA. System Management Interface. Retrieved May 20, 2023 from https://developer.nvidia.com/nvidia-system-management-interface.

[63] Intergovernmental Panel on Climate Change. 2022. Sixth Assessment Report. Retrieved May 20, 2023 from https://www.ipcc.ch/assessment-report/ar6/.

[64] United Nations Framework Convention on Climate Change. 2015. Adoption of the Paris Agreement. *FCCC/CP/2015/L.9/Rev.1*.

[65] Pratyush Patel, Zibo Gong, Syeda Rizvi, Esha Choukse, Pulkit Misra, Thomas Anderson, and Akshitha Sriraman. 2023. Towards Improved Power Management in Cloud GPUs. *IEEE Computer Architecture Letters*.

[66] Pratyush Patel, Katie Lim, Kushal Jhunjhunwalla, Ashlie Martinez, Max Demoulin, Jacob Nelson, Irene Zhang, and Thomas Anderson. 2023. Hybrid Computing for Interactive Datacenter Applications. *arXiv preprint arXiv:2304.04488*.

[67] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R So, Maud Texier, and Jeff Dean. 2022. The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink. *Computer*.

[68] Greenhouse Gas Protocol. 2015. A Corporate Accounting and Reporting Standard. Retrieved May 20, 2023 from https://ghgprotocol.org/corporate-standard.

[69] Ana Radovanović, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, Saurav Talukdar, Eric Mullen, Kendal Smith, MariEllen Cottman, and Walfredo Cirne. 2022. Carbon-aware Computing for Datacenters. *IEEE Transactions on Power Systems*.

[70] Dave Rensin. 2016. Introducing Google Customer Reliability Engineering (CRE). Retrieved May 20, 2023 from https://cloud.google.com/blog/products/devops-sre/introducing-a-new-era-of-customer-support-google-customer-reliability-engineering.

[71] Salesforce. 2023. Stakeholder Impact Report. Retrieved May 20, 2023 from https://stakeholderimpactreport.salesforce.com/.

[72] Awanthika Senarath and Nalin A. G. Arachchilage. 2018. Why Developers Cannot Embed Privacy into Software Systems? An Empirical Investigation. *International Conference on Evaluation and Assessment in Software Engineering (EASE)*.

[73] Awanthika Senarath, Marthie Grobler, and Nalin Asanka Gamagedara Arachchilage. 2019. Will They Use It or Not? Investigating Software Developers' Intention to Follow Privacy Engineering Methodologies. *ACM Transactions on Privacy and Security (TOPS)*.

[74] Amazon Web Services. Amazon CloudWatch. Retrieved May 20, 2023 from https://aws.amazon.com/cloudwatch/.

[75] Amazon Web Services. Amazon EC2 Spot Instances. Retrieved May 20, 2023 from https://aws.amazon.com/ec2/spot/.

[76] Amazon Web Services. Amazon Elastic Container Registry (ECR). Retrieved May 20, 2023 from https://aws.amazon.com/ecr/.

[77] Amazon Web Services. AWS Graviton Processor: Enabling the Best Price Performance in Amazon EC2. Retrieved May 20, 2023 from https://aws.amazon.com/ec2/graviton/.

[78] Amazon Web Services. Customer Carbon Footprint Tool. Retrieved May 20, 2023 from https://aws.amazon.com/aws-cost-management/aws-customer-carbon-footprint-tool/.

[79] Amazon Web Services. 2023. Understanding Your Carbon Emission Estimations. Retrieved May 20, 2023 from https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/ccft-estimation.html.

[80] Supreeth Shastri, Vinay Banakar, Melissa Wasserman, Arun Kumar, and Vijay Chidambaram. 2020. Understanding and Benchmarking the Impact of GDPR on Database Systems. *Proceedings of the VLDB Endowment*.

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

[82] Ion Stoica and Scott Shenker. 2021. From Cloud Computing to Sky Computing. *Workshop on Hot Topics in Operating Systems (HotOS)*.

[83] Jennifer Switzer, Gabriel Marcano, Ryan Kastner, and Pat Pannuto. 2023. Junkyard Computing: Repurposing Discarded Smartphones to Minimize Carbon. *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[84] The Istio Service Mesh. Retrieved May 20, 2023 from https://istio.io/.

[85] Thoughtworks. Cloud Carbon Footprint. Retrieved May 20, 2023 from https://www.cloudcarbonfootprint.org/.

[86] Amanda Tomlinson and George Porter. 2022. Something Old, Something New: Extending the Life of CPUs in Datacenters. *Workshop on Sustainable Computer Systems (HotCarbon)*.

[87] Uber. 2020. Introducing Domain-Oriented Microservice Architecture. Retrieved May 20, 2023 from https://www.uber.com/blog/microservice-architecture/.

[88] International Telecommunication Union. 2020. Greenhouse Gas Emissions Trajectories for the Information and Communication Technology Sector Compatible with the UNFCCC Paris Agreement. *ITU-T L.1470*.

[89] International Telecommunication Union and World Benchmarking Alliance. 2022. Greening Digital Companies: Monitoring Emissions and Climate Commitments.

[90] US Energy Information Administration. Open Data. Retrieved May 20, 2023 from https://www.eia.gov/opendata/index.php.

[91] Sarah Wang and Martin Casado. 2021. The Cost of Cloud, a Trillion Dollar Paradox. Retrieved May 20, 2023 from https://a16z.com/2021/05/27/cost-of-cloud-paradox-market-cap-cloud-lifecycle-scale-growth-repatriation-optimization/.

[92] WattTime. Retrieved May 20, 2023 from https://www.watttime.org/.

[93] Philipp Wiesner, Ilja Behnke, Dominik Scheinert, Kordian Gontarska, and Lauritz Thamsen. 2021. Let's Wait Awhile: How Temporal Workload Shifting Can Reduce Carbon Emissions in the Cloud. *International Middleware Conference (Middleware)*.

[94] Elizabeth Willmott. 2022. How Microsoft Is Using An Internal Carbon Fee to Reach its Carbon Negative Goal. Retrieved May 20, 2023 from https://www.microsoft.com/en-us/industry/blog/sustainability/2022/03/24/how-microsoft-is-using-an-internal-carbon-fee-to-reach-its-carbon-negative-goal/.

[95] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art Natural Language Processing. *Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*.