The War of the Efficiencies: Understanding the Tension between Carbon and Energy Optimization

WALID A. HANAFY, University of Massachusetts Amherst, USA
ROOZBEH BOSTANDOOST, University of Massachusetts Amherst, USA
NOMAN BASHIR, Massachusetts Institute of Technology, USA
DAVID IRWIN, University of Massachusetts Amherst, USA
MOHAMMAD HAJIESMAILI, University of Massachusetts Amherst, USA
PRASHANT SHENOY, University of Massachusetts Amherst, USA

Major innovations in computing have been driven by scaling up computing infrastructure, while aggressively optimizing operating costs. The result is a network of worldwide datacenters that consume a large amount of energy, mostly in an energy-efficient manner. Since the electric grid powering these datacenters provided a simple and opaque abstraction of an unlimited and reliable power supply, the computing industry remained largely oblivious to the carbon intensity of the electricity it uses. Much like the rest of the society, it generally treated the carbon intensity of the electricity as constant, which was mostly true for a fossil fuel-driven grid. As a result, the cost-driven objective of increasing energy-efficiency — by doing more work per unit of energy — has generally been viewed as the most carbon-efficient approach. However, as the electric grid is increasingly powered by clean energy and is exposing its time-varying carbon intensity, the most energy-efficient operation is no longer necessarily the most carbon-efficient operation. However, the recent focus on exploiting the flexibility of computing's workloads—along temporal, spatial, and resource dimensions—to reduce carbon emissions, comes at the cost of either performance or energy efficiency. In this paper, we quantify the trade-offs between energy efficiency and carbon efficiency in exploiting computing's flexibility and show that blindly optimizing for energy efficiency is not always the right approach.[1]

CCS Concepts: • **Computer systems organization → Cloud computing**; • **Hardware → Renewable energy**; • **Social and professional topics → Sustainability**.

Additional Key Words and Phrases: Carbon Efficiency, Energy Efficiency, Sustainable Computing

1 INTRODUCTION

The demand for computing has experienced rapid growth and is expected to accelerate even further. However, the increase in computing demand has not resulted in a proportional increase in energy demand so far [3]. The growth in computing's energy consumption has been kept in check by massive gains in algorithmic efficiency, measured in cycles per unit of work, of its software and energy efficiency, measured in energy consumption per cycle, of its hardware [14]. However, as the algorithmic and energy efficiency gains slow down, an increase in computing demand directly increases the energy demand. A conservative estimate projects that the energy consumption of datacenters will increase by at least 10% per year till 2030 [12], much higher than an estimated increase of 1.65% per year in 2010s [30]. As society has begun to recognize the environmental impact of our activities, reducing the carbon footprint of this accelerating energy demand has attracted significant attention from academic researchers [15, 22, 26, 37, 40, 42] and industry leaders [9, 31, 34].

[1]This work was originally published in HotCarbon'23 [21]

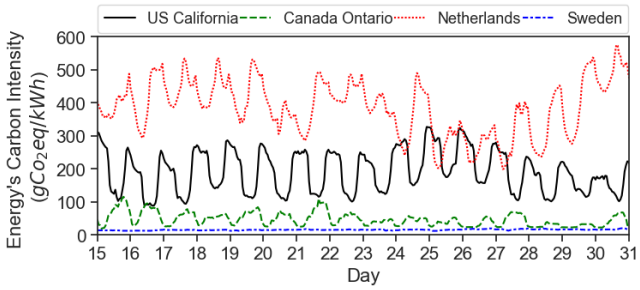


Fig. 1. *Energy's carbon intensity (05/15/2022 - 05/31/2022).*

The carbon footprint of computing depends on the computing's carbon efficiency, denoted as η_C, which is calculated by dividing the computing's energy efficiency η_E, measured as work done per unit of energy, by the energy's carbon intensity c, measured as the amount of emitted greenhouse gases (GHG) per kWh of energy. Traditionally, the electric grid has been powered by fossil fuels such as coal, and oil, which have similar carbon intensities of 1038 $g.CO_2eq/kWh$ and 1106 $g.CO_2eq/kWh$ [2]. Furthermore, even if energy's carbon intensity slightly varied across space and time, it was invisible to electricity consumers due to the simple and opaque abstraction exposed by the grid. As a result, the carbon intensity of electricity was viewed as constant, every unit of energy was the same, and a unit improvement in computing's energy efficiency meant a proportional improvement in computing's carbon efficiency. As the industry aggressively optimized for computing's energy efficiency—driven by the need to scale while reducing operational costs—it was only serendipitous that a cost-driven approach was also the environmentally conscious choice.

However, the evolution of the electric grid over the last decade has diversified the mix of energy sources used for electricity generation. With a higher penetration of renewable energy in the electric grid and advancements in traditional power plant technologies, such as combined heat and power (CHP) plants, the carbon intensity of electricity now varies widely over time and across locations [5, 39]. Figure 1 illustrates the carbon intensity of energy in gCO_2eq/kWh for four locations around the globe. Sweden exhibits a very low carbon intensity due to its reliance on hydropower, while the Netherlands has a higher carbon intensity due to its fossil fuel-heavy resource mix. Furthermore, Ontario and California experience diurnal changes in carbon intensity due to the increased use of solar

energy. The growing penetration of renewables in the electric grid has decreased the carbon intensity worldwide but also highlighted the importance of considering the timing and location of energy consumption. Computing workloads offer the flexibility to choose when and where to execute and consume energy. However, mechanisms that enable the exploitation of computing's flexibility tend to be energy-inefficient. Therefore, in most cases, achieving carbon efficiency requires sacrificing energy efficiency.

From a business standpoint, it does not make financial sense to be purposefully energy-inefficient as it costs money, especially in the absence of penalties on carbon emissions. However, there is a social incentive to reduce carbon footprint, and the computing industry is responding to this problem in two ways. First, assuming a time-varying carbon intensity, operators are leveraging the performance flexibility of workloads such as delaying or relocating workloads. Second, the industry is using various forms of carbon credits and offsets to reduce its estimated carbon emissions. Initially, carbon credits and power purchase agreements were used to offset carbon emissions on an annual basis. However, recently, the industry has started using stricter forms of carbon offsets that match the energy demand of datacenters with renewable energy generation on the same distribution grid on an hourly basis, known as 24/7 matching [6]. While this is a step in the right direction, it should not be construed as running the datacenters purely on zero-carbon energy because datacenters still rely on the electric grid. As the electric grid still uses carbon-intensive energy sources, no one can be fully carbon-free until the grid itself is carbon-free. In this futuristic world, the focus can shift back toward the gains in energy efficiency that the industry has helped achieve. In the meanwhile, blindly focusing on energy efficiency leaves many possible carbon-specific optimizations on the table. Unfortunately, just like the tension between alternating and direct current in the late nineteenth century, this debate has become a war of narratives and financial levers rather than a technical one [1, 8]. As grids worldwide still rely on carbon-emitting energy generation sources, we need to exploit all the flexibility of computing workloads such that we maximize carbon efficiency while optimizing energy efficiency.

Computing workloads, depending on the application, offer flexibility along multiple dimensions. They can be delayed, paused, and resumed (*temporal flexibility*). They can be assigned more or fewer resources (*resource scaling*). They can be scaled up or down using Dynamic Voltage and Frequency Scaling (DVFS) (*rate shifting*). Finally, they can be executed at a different geographical location (*spatial shifting*). These mechanisms for exploiting flexibility could be energy-inefficient to varying degrees and yield different amounts of carbon savings. As energy inefficiency costs money, we need to analyze the trade-offs between carbon efficiency and energy efficiency. This can not only guide the industry in estimating the cost of optimizing for carbon but also help regulators determine the appropriate incentives and penalties for carbon emissions.

The management of carbon emissions in cloud datacenters is receiving significant attention due to the growing impact of climate change [16, 17, 19, 20, 31, 32, 37, 39, 42]. While some studies have focused on embodied carbon, which refers to carbon emissions from the manufacturing and relocation of infrastructure, we are concentrating on the operational energy and carbon footprint of powering and cooling the infrastructure. Although our efficiency metrics can encapsulate other accounting methods, such as embodied carbon, by spreading the embodied carbon over the actual lifespan of the infrastructure, we did not use that combination since it does not comply with the GHG protocol [4] as highlighted by other researchers [13]. Nonetheless, our findings on the tradeoffs between energy efficiency and carbon efficiency remain valid.

To the best of our knowledge, our work is the first to explicitly quantify the energy-carbon trade-offs of various flexibility mechanisms. To demonstrate this trade-off, we consider real-world carbon traces and analytically-modeled applications and simulate the effect of carbon-aware scheduling mechanisms. We use a state-of-the-art energy-efficient execution as the baseline and demonstrate how carbon efficiency can be significantly increased by being energy inefficient, and blindly optimizing for energy efficiency is not always the right approach. We also highlight the trade-off breadth of different techniques and show that there exists a tipping point where carbon savings is not yet affected by the energy inefficiency of such mechanisms. Beyond this point, the carbon footprint of energy overheads overweighs the reduction in carbon savings from exploiting flexibility.

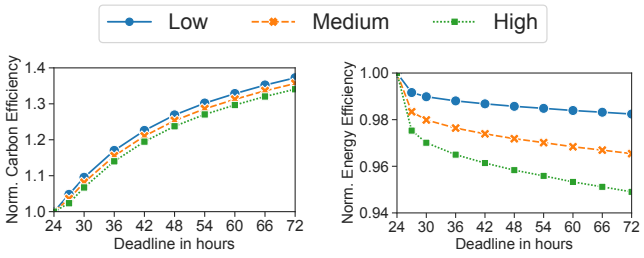
2 ILLUSTRATING EFFICIENCY TRADE-OFFS

In this section, we investigate the trade-off between carbon efficiency and energy efficiency of four commonly-used mechanisms for exploiting computing's flexibility: temporal shifting, resource scaling, rate shifting, and spatial shifting. We choose state-of-the-art energy-efficient execution as the baseline unless stated otherwise. Therefore, our carbon efficiency gains come from optimizing specifically for carbon and not from improving computing's energy efficiency. Furthermore, the term "*carbon efficiency*" refers to computing's carbon efficiency and not energy's carbon efficiency, which is referred to by the reciprocal term of carbon intensity.

We use a 3-year-long carbon intensity trace for Ontario, Canada, from electricity map [7] spanning January 1, 2020, to December 31, 2022. The trace provides the hourly average intensity values, measured in gCO_2eq/kWh. Unless stated otherwise, we assume that the job starts at the hour boundary, and aggregate our results across jobs starting at each hour of the year. We report carbon efficiency and energy efficiency values, with the results normalized to the most energy-efficient for energy efficiency and the least carbon-efficient for carbon efficiency, unless otherwise specified. Furthermore, we present our analysis across a wide range of empirically-driven configurations that map to real-world server classes and application characteristics to ensure our results are broadly applicable and not tied to a particular application or hardware.

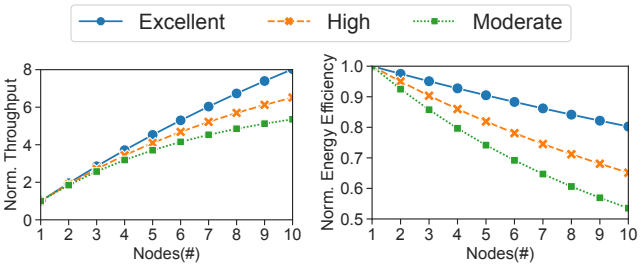
2.1 Temporal Shifting

The time-varying nature of electricity's carbon intensity creates green time periods, where the carbon intensity is significantly lower than the average carbon intensity for that location, as shown in Figure 1. The simplest, and most common, strategy to increase computing's carbon efficiency is to wait for such low carbon periods to arrive, execute the job during a given period, suspend the job at the



(a) Carbon Efficiency (b) Energy Efficiency

Fig. 2. *Carbon and energy efficiency of jobs with different overheads. Longer deadlines allow a higher use of checkpoint & restore to operate in low carbon periods.*



(a) Throughput (b) Energy Efficiency

Fig. 3. *As applications with sub-linear scaling characteristics scale, their energy efficiency reduces.*

end of this period, and resume its operation during the next low carbon period. However, this intermittent execution of jobs leverages checkpoint and restore techniques to save the state between low carbon periods, incurring energy overhead. The amount of overhead depends on the frequency of checkpoint and restore operations and the energy cost of a single checkpoint and restore operation, which, in turn, depends on the size of the state of the application [35].

Prior work on temporal shifting to reduce the carbon footprint has focused on configuring the thresholds for identifying low carbon periods, with or without future knowledge of carbon intensity, and the trade-offs between carbon savings and job completion times [34, 37, 42], while ignoring the overhead of suspend-resume. There is also a recent work that explores online algorithms, taking into account the overhead of suspend and resume methods [26]. However, these studies do not explicitly discuss or quantify the trade-off between the loss of energy efficiency and gains in carbon efficiency. We bridge this gap in an empirical study outlined next.

1. Experimental Setup. Our setup presumes multiple jobs with different overhead percentages, aiming to optimize carbon efficiency using state-of-the-art carbon-aware execution policies.

Applications. We consider an application that constantly performs computation, such as ML training, and requires a certain memory size to store the intermediate results. We assume that the job has performance flexibility and allows the operator to checkpoint & restore its state. We use three variants of this job represented by their checkpoint & restore overheads, which we configure as the time it takes to checkpoint or restore the memory state of the job. We set the overheads for the three variants as 5 minutes (low), 10 minutes (medium), and 15 minutes (high), which can be mapped to real-world applications by assuming ML training over different-sized models. The job also has temporal flexibility, aka slack, which we define as a multiple of the job's uninterrupted execution time. For example, a slack factor of 1.5× for a 24-hour job without interruption means that it takes 36 hours to finish. In our experiments, we consider a 24-hour job and vary the slack factor from 1× to 3× of the job's runtime.

Policy. We use a deadline-aware *suspend-resume* policy to execute the job that has been proposed in recent work to reduce the carbon footprint of jobs with temporal flexibility [42]. This policy assumes perfect future knowledge of carbon intensity and selects low carbon slots for executing the job such that it finishes before the deadline. It does not take into account the energy overhead of checkpoint & restore when determining the number and selection of slots for execution. However, we take into account the carbon overhead of intermittent execution and subtract that from carbon savings when calculating carbon efficiency gains.

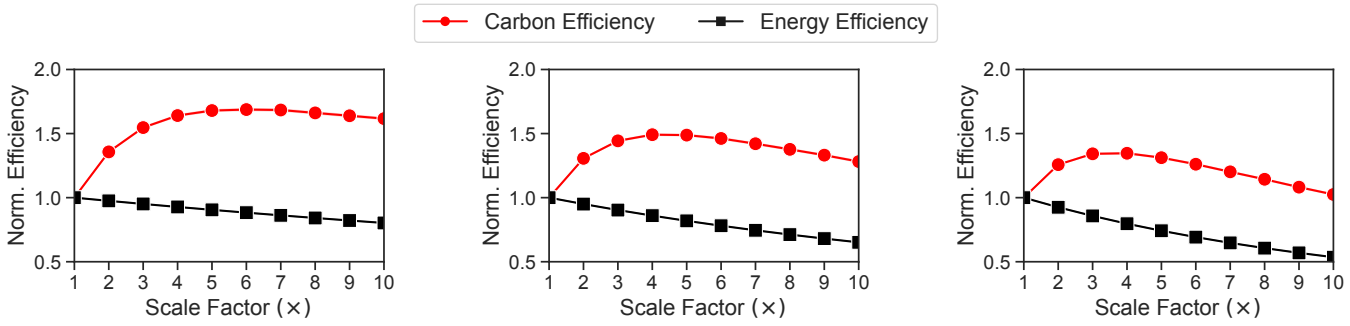
2. Experimental Results. The results in Figure 2a show that a higher degree of flexibility (higher slack) can lead to greater reductions in carbon emissions (increased carbon efficiency). However, flexibility comes at the cost of energy efficiency. As shown in Figure 2b, larger slacks allow applications to checkpoint & restore more often, increasing the energy and carbon overhead, which reduces energy efficiency. In Figure 2b, the y-axis limit is set between 1 and 0.94 to ensure clear visibility of the lines representing normalized energy efficiency. It is important to note that the magnitude of carbon savings and the impact on energy efficiency are application-specific, but their relationship is fundamental and will hold across application characteristics and carbon intensity profiles.

3. Key Takeaways. *Higher temporal flexibility enables applications to increase their carbon efficiency, but the gains depend on how often applications incur the energy and carbon overhead of the checkpoint and restore mechanism to take advantage of low carbon periods.*

2.2 Resource Scaling

Resource scaling is the method of adding or removing resources to a given job to speed up or slow down the speed of execution, respectively. In the context of carbon-aware computing, resource scaling can be used as an antidote to increasing job completion time under *suspend-resume* execution [22, 37]. Instead of resuming the job at 1× during low-carbon periods, it can be scaled up to k× to compensate for the time spent in suspend state. However, the effectiveness of scaling depends on the application characteristics, such as the size of sequential barriers modeled by Amdahl's law [11]. As a result, as the allocated resources increase, the speed up increases sub-linearly, reducing the energy efficiency of execution.

1. Experimental Setup. In this experiment, we demonstrate the carbon efficiency and energy efficiency trade-off for an application that leverages resource scaling to reduce its carbon footprint. To



(a) Excellent (b) High (c) Moderate

Fig. 4. *Normalized carbon and energy efficiency for different scalability profiles when running in Ontario, Canada.*

characterize the energy overheads of scaling alone, we focus on scenarios where jobs must finish without increasing the job completion time, i.e., no slack.

Applications. Since applications' scalability dictates the energy overhead, we consider different scalability in terms of the throughput reduction per additional node. Figure 3 shows the scalability characteristics for three applications with excellent scaling (5% reduction in normalized throughput per additional node), high scaling (10% reduction per additional node), and moderate scaling (15% reduction per additional node) characteristics. This behavior is similar to many real-world distributed applications, as shown in [22, 33] and is considered a result of the relations between workloads' communication and computation [22]. We assume that the energy consumption of each additional node is the same. As a result, as shown in Figure 3b, the energy efficiency of the computing decreases due to reduced normalized throughput per unit energy after scaling.

Policy. Souza et al. [37] propose a carbon-aware scaling policy called *Wait&Scale*, which selects an application-specific scale factor based on its scalability characteristics. It assumes perfect knowledge of future carbon intensity. Similar to a carbon-aware suspend-resume policy, it generates a schedule that suspends the job during high-carbon periods and resumes it at the application-specific scale factor during low-carbon periods. In this scenario, when the scale factor is set to 1 and since job completion time is set as the job length, the job will execute in an uninterrupted manner. As a result, jobs do not gain carbon savings from temporal flexibility.

2. Experimental Results. Figure 4 shows the normalized carbon efficiency and energy efficiency of the three scalable applications with scalability characteristics shown in Figure 3. The results indicate that maximizing energy efficiency does not necessarily result in maximum carbon efficiency. Rather, the increase in carbon efficiency depends on the flexibility that comes with a decrease in energy efficiency. Furthermore, the results in Figure 4 demonstrate that the application's scalability plays a significant role in carbon efficiency gains and energy efficiency losses. For instance, the job with excellent scaling (Figure 4a) was able to increase its carbon efficiency by 68% at an energy efficiency loss of 15%. Conversely, the moderately scalable job showed only a 34% increase in carbon efficiency but paid more than a 25% loss in energy efficiency.

It is also worth noting that, in all three cases, Figure 4 illustrates that increasing the scaling factor does not always improve carbon efficiency. Beyond a certain scaling factor, the gain in throughput during high carbon periods does not overcome the carbon cost due to low energy efficiency at high scales. Hence, the reciprocal behavior that loss of energy efficiency does not always lead to carbon efficiency should also be considered while scaling applications.

3. Key Takeaways. *Carbon-aware application of scaling policies can yield significant gains in carbon efficiency. However, the high energy overhead of scaling means that applications must be highly judicious in choosing their scale factor as gains become marginal at high scales.*

2.3 Rate Shifting

Dynamic Voltage and Frequency Scaling (DVFS) has been widely used for energy optimization for servers in datacenters by leveraging the non-linear relationship between power consumption and application throughput [24, 29, 41, 44]. DVFS can also be used for optimizing carbon efficiency where the application runs faster using a higher frequency during low carbon periods and saves energy by lowering the CPU frequency and its execution speed during high carbon periods, possibly at a lower energy-efficiency. DVFS can especially be helpful for uninterruptible applications that cannot be suspended, as it makes progress at all times without suspension.

The energy savings in using DVFS come from the non-linear relationship between a processor's power demand (P) and its frequency (f) and voltage (V) governed by the following equation,

$$P = CfV^2 + P_{\texttt{static}}. \tag{1}$$

Here, C and $P_{\texttt{static}}$ are processor architecture-specific constants. As shown in this equation, power demand has a dynamic and a static range. The dynamic range is dictated by the linear relation with its frequency; reducing the operating frequency by 50% will reduce the dynamic range by 50%. However, higher savings in the dynamic range come from the non-linear relationship with the operating voltage of the processor; decreasing the operating voltage to half reduces the dynamic power consumption of the processor by a factor of 4. It is worth mentioning that a decrease in the power consumption of a processor does not lead to a proportional decrease

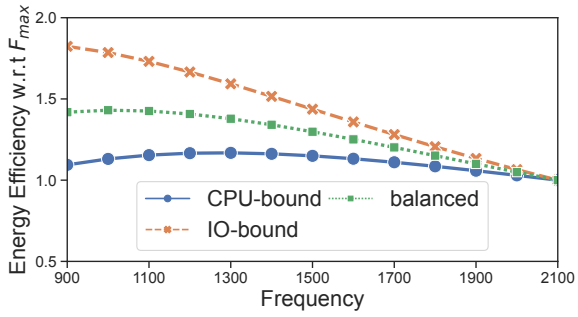


Fig. 5. *Normalized energy efficiency across different application profiles defined by their IO-intensiveness.*

in its performance. The decrease in performance, denoted by S, depends on the CPU-boundedness of applications and can be modeled using Amdahl's law [11] by the following equation:

$$S = \frac{1}{io + {}^{(1-io)}/_{f'_n}}. \tag{2}$$

where io is the fraction of time an application spends accessing Input/Output (IO) peripherals at the maximum operating frequency (F_{max}), which is one of the major reasons affecting the application's slowdown due to their sequential nature. f'_n is the normalized frequency w.r.t. the highest frequency $f'_n = f_n/F_{max}$, where $f_n \in [F_{\min}, F_{\max}]$.

Considering only the dynamic range, by assuming P_{static} to be 0 and C to be 1, a 50% decrease in operating frequency for an application with 50% IO yields only a 34% reduction in the application's performance, making it more efficient. While reducing frequency alone yields a 50% power savings, more savings will be achieved if the operating voltage is decreased as well. However, many modern processors do not allow direct and separate control of the processor's operating voltage. Instead, changing the operating frequency alters the operating voltage to a pre-determined voltage level. As a result, applications may not have a full range of parameters available to them to optimize energy efficiency using DVFS. Furthermore, depending on the practical values for P_{static} and C, energy efficiency gains may be further limited.

1. Experimental Setup. We next evaluate the impact of DVFS for different application characteristics under different operation frequencies to highlight the tension between carbon and energy efficiency.

Server Configurations. Figure 5 shows our modeled server, modeled after a local server in our lab, that has an Intel-Xeon Processor E5-2620 v4 with DVFS enabled. The server dynamic power range was controlled through the frequency range of (0.9 to 2.1GHz, 0.1 MHz step), voltage range of 0.8 to 1.2V. The CPU's transistor gate's capacitance (3×10^{-2}), while the static power of the server is 30W, which is approximately 25% of its operational power. In practice, frequency levels and voltage values are tied together. Thus, we split the voltage range into the same number of steps as frequency and establish a direct relationship between frequency and voltage levels.

Applications. We model three different applications that map to real-world: "CPU-bound" application such as matrix multiplication in ML training (0% time spent on input/output), "IO-bound"application of text processing, e.g., Hadoop (70% time spent on I/O), and "balanced" application such as in-memory data processing, e.g., Spark, (40% time spent on I/O). In all of these examples, I/O time is configured when the processor is running at the highest frequency. The application-specific energy efficiency profile is created based on its I/O%, under different frequencies, by equations 1 and 2.

Policy. There is no prior work on leveraging DVFS to optimize for carbon efficiency. Therefore, we devise a simple strategy to explore the carbon and energy efficiency trade-off. Our policy uses the application-specific profile to operate at a high frequency, often less energy efficient, during low carbon periods and at a low frequency, e.g., the highest energy-efficient frequency, during high carbon periods. The complex decision space necessitated evaluating the policy against all frequency permutations. In this case, the policy is given all permutations of $F_1, F_2 \in [900, ..., 2100]$, and we also use the mean carbon intensity μ_c, during the expected execution period, as the threshold. In this case, at a time slot i, the application runs at frequency F_1 if the carbon intensity c_i is less than or equal to the threshold μ ($c_i \leq \mu$), and it runs at frequency F_2 otherwise.

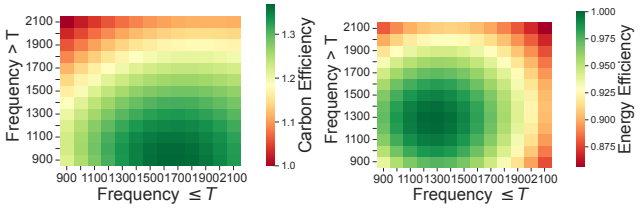
2. Experimental Results. We model the energy efficiency behavior of our applications by computing the normalized energy consumption at the highest frequency with respect to other frequencies. Figure 5 shows that across application classes, I/O-bound applications observe the most gains since an I/O-heavy application does not utilize the CPU to its full extent and is not affected by the slower processing speed of the CPU at lower frequencies, while a CPU-bound application barely sees any gains as its throughput is highly affected by the processing speed. The efficiency gains from DVFS were further in prior work [25].

Figure 6 shows the carbon and energy efficiency, of the three applications denoted as CPU-bound (0% IO), IO-Bound (70% IO), and balanced (40% IO). The results clearly indicate that energy-efficient configurations do not always yield the highest carbon efficiencies. For example, Figure 6a shows that the highest carbon efficiency is achieved by running fast (~1.7GHz) when energy's carbon intensity is low while running slow (~1GHz) otherwise, contrarily, always running at (~1.3GHz) yields highest energy efficiency. Figure 6c shows another example where energy and carbon efficiency are more correlated as the energy and carbon efficiency increase from lowering the frequency. We point out that other configurations resulted in similar conclusions, but we had to leave them out due to space constraints.

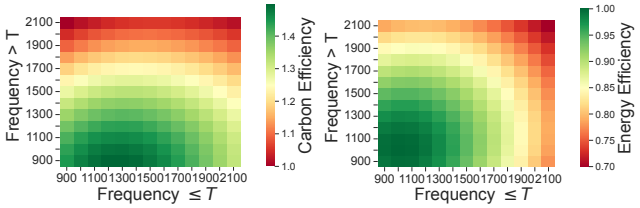
3. Key Takeaways. *Using DVFS in a carbon-aware manner can greatly improve carbon efficiency. Adjusting the operating frequency can result in high-efficiency gains that might bridge the gap between energy and carbon-efficient computing.*

2.4 Spatial Shifting

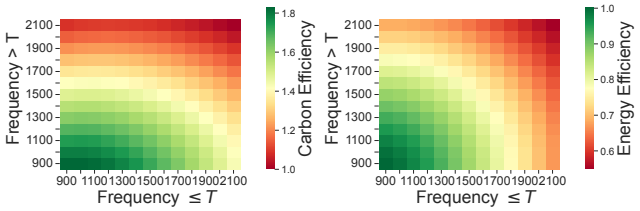
Migrating services across the network in order to optimize cost or latency has been widely discussed [27, 36, 39, 43]. Similarly, migrations can be utilized to increase carbon efficiency by transferring jobs to a region where the carbon intensity of energy is lower. For example, a task can be migrated across the globe by following the



(a) CPU-bound Task



(b) Balanced Task



(c) IO-bound Task

Fig. 6. *Carbon and energy efficiency of different tasks.*

availability of carbon-free solar energy. However, flexibly transferring jobs between locations decreases energy efficiency as it involves energy overheads from checkpointing, transferring, and restoring execution state as well as application data. For instance, for migrating a data processing application (e.g., ML training), the application state must be checkpointed at the source, moved across the network, and restarted at the destination. Also, the data must be moved or cloned in both locations which incurs extra energy consumption. The process of checkpointing and restoring depends on the application size, as explained in section 2.1, while the migration depends on the state and data size, and cloning also comes with energy and financial cost overheads. A full analysis of spatial shifting trade-offs is left to future work.

3 DISCUSSION

Increasing computing carbon efficiency through carbon-aware scheduling involves a wide range of trade-offs in terms of performance, energy, and cost. For instance, temporal and spatial shifting often requires performance reductions, either by extending the completion time [23, 39, 42] or by utilizing distant but greener resources, which increases the response time [17, 18, 38, 39]. Carbon-aware scheduling also tends to increase costs, not only through implicit cost increases but also due to higher energy consumption from flexible execution. For example, researchers have highlighted that carbon-aware scheduling often introduces increases in peak, as well as acute changes in computing and energy demand [10, 23, 28]. These trade-offs are present in other sustainability approaches, where reducing embodied carbon by amortizing servers means running at lower energy efficiency compared to state-of-the-art servers [13]. Finally, it is worth noting that despite these trade-offs, researchers have highlighted scenarios and approaches where the benefits outweigh the overheads [15, 23, 26, 38].

4 CONCLUSION

For a long-time, energy efficiency has been a key objective for cost-effective and sustainable computing. The necessity to decrease computing's operating costs and the environmental impact of computing made energy efficiency a first-class citizen in computing. However, the wide adoption of clean energy in electrical grids, along with increasing public awareness about energy sources, and the enabled visibility of the time-varying carbon intensity, has resulted in a shift where the most energy-efficient operations may no longer be considered the most sustainable or socially acceptable choice. For these reasons, carbon efficiency (the amount of work per unit of carbon) appeared as a "true" sustainability metric. The key idea in increasing carbon efficiency is to exploit computing workloads' flexibility by adjusting execution time (Temporal Shifting), speed (Scaling and Rate Shifting), and location (Spatial Shifting) according to the grid's carbon intensity. In this paper, we highlighted an inevitable tension between carbon and energy efficiency. We explored the core mechanisms used in carbon-efficient computing along with policies from the state-of-the-art in a wide range of scenarios. The paper demonstrated qualitatively and quantitatively that "striving for maximum energy efficiency is not always the most sustainable (carbon-efficient) approach". The gains and overheads of combining multiple carbon-aware flexibility mechanisms are left for future work.

ACKNOWLEDGEMENTS

We thank the HotCarbon reviewers for their valuable comments, which improved the quality of this paper. We also thank WattTime and electricityMap for providing the carbon-intensity data. This research is supported by NSF grants 2213636, 2136199, 2106299, 2102963, 2105494, 2021693, 2020888, 2045641, 2211302, 2211888, US Army contract W911NF-17-2-0196, DOE award DE-EE0010143, and VMware, and Amazon Web Services.

REFERENCES

[1] 2014. The War of the Currents: AC vs. DC Power. https://www.energy.gov/articles/war-currents-ac-vs-dc-power.
[2] 2022. Annual Electric Power Industry Report. https://www.eia.gov/electricity/data/eia861/
[3] 2022. Global Trends in Internet Traffic, Data Centre Workloads and Data Centre Energy Use, 2010-2019. https://www.iea.org/data-and-statistics/charts/global-trends-in-internet-traffic-data-centre-workloads-and-data-centre-energy-use-2010-2019.
[4] 2022. Greenhouse Gas Protocol. https://ghgprotocol.org/.
[5] 2022. Share of Cumulative Power Capacity by Technology, 2010-2027. https://www.iea.org/data-and-statistics/charts/share-of-cumulative-power-capacity-by-technology-2010-2027.
[6] 2023. 24/7 by 2030: Realizing a Carbon-free Future. https://www.gstatic.com/gumdrop/sustainability/247-carbon-free-energy.pdf.
[7] 2023. Electricity Map. https://www.electricitymap.org/map.
[8] 2023. War of the Currents. https://en.wikipedia.org/wiki/War_of_the_currents.

[9] 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. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*.

[10] 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. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[11] Gene M Amdahl. 1967. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In *Proceedings of the Spring Joint Computer Conference*.

[12] Srini Bangalore, Arjita Bhan, Andrea Del Miglio, Pankaj Sachdeva, Vijay Sarma, Raman Sharma, and Bhargs Srivathsan. 2023. Investing in the Rising Data Center Economy. https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/investing-in-the-rising-data-center-economy.

[13] Noman Bashir, David Irwin, and Prashant Shenoy. 2023. On the Promise and Pitfalls of Optimizing Embodied Carbon. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems (HotCarbon)*.

[14] Noman Bashir, David Irwin, Prashant Shenoy, and Abel Souza. 2022. Sustainable Computing – Without the Hot Air. In *Proceedings of the First Workshop on Sustainable Computer Systems Design and Implementation (HotCarbon)*.

[15] Roozbeh Bostandoost, Adam Lechowicz, Walid A. Hanafy, Noman Bashir, Prashant Shenoy, and Mohammad Hajiesmaili. 2024. LACS: Learning-Augmented Algorithms for Carbon-Aware Resource Scaling with Uncertain Demand. In *The 15th ACM International Conference on Future and Sustainable Energy Systems (e-Energy '24), June 4–7, 2024, Singapore, Singapore.* https://doi.org/10.1145/3632775.3661942

[16] A. Chien. 2021. Driving the Cloud to True Zero Carbon. *CACM* 64, 2 (February 2021).

[17] Jesse Dodge, Taylor Prewitt, Remi Tachet des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A. Smith, Nicole DeCario, and Will Buchanan. 2022. Measuring the Carbon Intensity of AI in Cloud Instances. In *2022 ACM Conference on Fairness, Accountability, and Transparency (FAccT '22)*.

[18] Peter Xiang Gao, Andrew R. Curtis, Bernard Wong, and Srinivasan Keshav. 2012. It's Not Easy Being Green. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (Helsinki, Finland) *(SIGCOMM '12)*. 211–222. https://doi.org/10.1145/2342356.2342398

[19] Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S. Lee, David Brooks, and Carole-Jean Wu. 2022. ACT: Designing Sustainable Computer Systems With An Architectural Carbon Modeling Tool. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (New York) *(ISCA '22)*. 784–799. https://doi.org/10.1145/3470496.3527408

[20] 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. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*.

[21] Walid A. Hanafy, Roozbeh Bostandoost, Noman Bashir, David Irwin, Mohammad Hajiesmaili, and Prashant Shenoy. 2023. The War of the Efficiencies: Understanding the Tension between Carbon and Energy Optimization. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems* (Boston, MA, USA) *(HotCarbon '23)*. Article 19, 7 pages. https://doi.org/10.1145/3604930.3605709

[22] Walid A. Hanafy, Qianlin Liang, Noman Bashir, David Irwin, and Prashant Shenoy. 2023. CarbonScaler: Leveraging Cloud Workload Elasticity for Optimizing Carbon-Efficiency. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 7, 3, Article 57 (December 2023), 28 pages. https://doi.org/10.1145/3626788

[23] Walid A. Hanafy, Qianlin Liang, Noman Bashir, Abel Souza, David Irwin, and Prashant Shenoy. 2024. Going Green for Less Green: Optimizing the Cost of Reducing Cloud Carbon Emissions. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (ASPLOS '24)*. 479–496. https://doi.org/10.1145/3620666.3651374

[24] Jakub Krzywda, Ahmed Ali-Eldin, Trevor E Carlson, Per-Olov Östberg, and Erik Elmroth. 2018. Power-Performance Tradeoffs in Data Center Servers: DVFS, CPU Pinning, Horizontal, and Vertical scaling. *Future Generation Computer Systems* (2018).

[25] Etienne Le Sueur and Gernot Heiser. 2010. Dynamic Voltage and Frequency Scaling: The Laws of Diminishing Returns. In *Proceedings of the 2010 International Conference on Power Aware Computing and Systems*.

[26] Adam Lechowicz, Nicolas Christianson, Jinhang Zuo, Noman Bashir, Mohammad Hajiesmaili, Adam Wierman, and Prashant Shenoy. 2023. The Online Pause and Resume Problem: Optimal Algorithms and An Application to Carbon-Aware Load Shifting. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 7, 3, Article 53 (Dec 2023), 36 pages. arXiv:2303.17551

[27] Mathieu Lemay, Kim-Khoa Nguyen, Bill St. Arnaud, and Mohamed Cheriet. 2012. Toward a Zero-Carbon Network: Converging Cloud Computing and Network Virtualization. *IEEE Internet Computing* (2012).

[28] Liuzixuan Lin and Andrew A Chien. 2023. Adapting Datacenter Capacity for Greener Datacenters and Grid. In *Proceedings of the 14th ACM International Conference on Future Energy Systems* (Orlando, FL, USA) *(e-Energy '23)*. 200–213. https://doi.org/10.1145/3575813.3595197

[29] David Lo, Liqun Cheng, Rama Govindaraju, Parthasarathy Ranganathan, and Christos Kozyrakis. 2015. Heracles: Improving Resource Efficiency at Scale. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*.

[30] Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith, and Jonathan Koomey. 2020. Recalibrating Global Data Center Energy-use Estimates. *Science* (2020).

[31] 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* 55, 7 (2022), 18–28. https://doi.org/10.1109/MC.2022.3148714

[32] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. *Carbon Emissions and Large Neural Network Training*. Technical Report. arXiv.

[33] Hang Qi, Evan R. Sparks, and Ameet Talwalkar. 2017. Paleo: A Performance Model for Deep Neural Networks. In *Proceedings of the International Conference on Learning Representations*.

[34] 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. 2023. Carbon-Aware Computing for Datacenters. *IEEE Transactions on Power Systems* (2023).

[35] Prateek Sharma, Tian Guo, Xin He, David Irwin, and Prashant Shenoy. 2016. Flint: Batch-Interactive Data-Intensive Processing for Transient Servers. In *ACM European Conference on Computer Systems (EuroSys)*.

[36] Zhiming Shen, Qin Jia, Gur-Eyal Sela, Ben Rainero, Weijia Song, Robbert van Renesse, and Hakim Weatherspoon. 2016. Follow the Sun through the Clouds: Application Migration for Geographically Shifting Workloads. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*.

[37] 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. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*.

[38] Abel Souza, Shruti Jasoria, Basundhara Chakrabarty, Alexander Bridgwater, Axel Lundberg, Filip Skogh, Ahmed Ali-Eldin, David Irwin, and Prashant Shenoy. 2023. CASPER: Carbon-Aware Scheduling and Provisioning for Distributed Web Services. In *Proceedings of the 14th International Green and Sustainable Computing Conference (IGSC), Toronto, ON, Canada*.

[39] Thanathorn Sukprasert, Abel Souza, Noman Bashir, David Irwin, and Prashant Shenoy. 2024. On the Limitations of Carbon-Aware Temporal and Spatial Workload Shifting in the Cloud. In *Proceedings of the Nineteenth European Conference on Computer Systems (EuroSys '24)*. 924–941. https://doi.org/10.1145/3627703.3650079

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

[41] Mark Weiser, Brent Welch, Alan Demers, and Scott Shenker. 1996. Scheduling for Reduced CPU Energy. *Mobile Computing*.

[42] 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. In *Proceedings of the 22nd International Middleware Conference (Middleware)*.

[43] Timothy Wood, K.K. Ramakrishnan, Prashant Shenoy, and Jacobus Van der Merwe. 2011. CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines. In *International Conference on Virtual Execution Environments (VEE)*.

[44] Chaojie Zhang, Alok Kumbhare, Ioannis Manousakis, Deli Zhang, Pulkit Misra, Rod Assis, Kyle Woolcock, Nithish Mahalingam, Brijesh Warrier, David Gauthier, Lalu Kunnath, Steve Solomon, Osvaldo Morales, Marcus Fontoura, and Ricardo Bianchini. 2021. Flex: High-Availability Datacenters With Zero Reserved Power. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*.