International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2025: 7.089

Energy-Efficient Kubernetes Scheduling for Green Cloud Computing: A Carbon-Aware Workload Placement Framework

Dinesh Kumar Movva

https://orcid.org/0009-0003-1231-5325

Abstract: ***Data center energy consumption attributable to containerized cloud workloads has grown substantially with the adoption of Kubernetes at enterprise scale, yet Kubernetes scheduling decisions remain entirely agnostic to the energy and carbon implications of workload placement. This paper presents CASK (Carbon-Aware Scheduler for Kubernetes), a novel scheduling framework that extends Kubernetes' standard scheduler with carbon intensity awareness, renewable energy availability signals, and hardware energy efficiency profiles to minimize the carbon footprint of containerized workloads without degrading performance SLAs. CASK integrates real-time carbon intensity data from regional electricity grid APIs, hardware thermal design power (TDP) profiles for heterogeneous node fleets, and time-of-day renewable availability forecasts into a multi-objective scheduling function that balances carbon minimization with latency and availability constraints. Evaluation across 3,200 nodes in two hyperscale enterprise Kubernetes deployments over 20 weeks demonstrates that CASK reduces mean carbon emissions per workload unit by 28.3%, shifts 34% of deferrable batch workloads to low-carbon time windows, and achieves energy efficiency improvements of 19.4% through hardware-aware pod placement on energy-efficient node types- without exceeding a 4.2% mean increase in workload completion time for deferrable workloads. A carbon-aware scheduling policy framework, renewable energy signal integration architecture, and Kubernetes scheduler plugin implementation guide are presented.***

Keywords: Kubernetes, Green Cloud Computing, Energy Efficiency, Carbon-Aware Scheduling, Sustainable Computing, Container Orchestration, Carbon Footprint

1. Introduction

The technology industry's commitment to carbon neutrality has created significant pressure on platform engineering teams to reduce the carbon footprint of cloud infrastructure- yet the tools available for carbon-aware workload management have not kept pace with the scale and complexity of modern Kubernetes deployments. Cloud providers report their data center power usage effectiveness (PUE) and renewable energy purchasing metrics at the regional level, but individual workload-level carbon attribution and scheduling decisions remain largely unaddressed.

Kubernetes scheduling decisions- which pods are placed on which nodes in which regions- have direct and substantial energy implications that current Kubernetes implementations entirely ignore. A pod scheduled on an older, energy-inefficient node type may consume 40–60% more energy than the same workload on a current-generation energy-efficient node. A batch job scheduled during peak grid carbon intensity hours in a coal-dependent region may produce three to five times the carbon emissions of the same job run during off-peak hours when renewable generation predominates. These differences are invisible to the Kubernetes scheduler, which optimizes exclusively for resource fit and node affinity constraints.

The emergence of real-time carbon intensity APIs- notably the Carbon Intensity API published by the UK National Grid ESO, WattTime's grid marginal emissions data, and Electricity Maps' global carbon intensity dataset- has created a new class of data that enables carbon-aware scheduling decisions at fine-grained temporal and spatial resolution. These APIs expose the current and forecasted carbon intensity of electricity grids at regional granularity, enabling scheduling systems to prefer low-carbon regions and time windows for deferrable workloads.

Hardware heterogeneity in enterprise Kubernetes node fleets creates an additional dimension of energy optimization opportunity. Modern Kubernetes deployments often span node types ranging from legacy general-purpose instances with older CPU architectures to current-generation ARM-based instances (AWS Graviton, Ampere Altra) that deliver 40–60% better performance per watt than comparable x86 instances. A scheduler that is aware of node energy efficiency profiles can systematically prefer energy-efficient node types for equivalent workloads, reducing cluster-wide energy consumption without any change to workload code.

CASK makes three primary contributions: first, a Kubernetes scheduler plugin architecture that integrates carbon intensity signals, hardware energy profiles, and renewable availability forecasts into scheduling decisions with minimal latency overhead; second, a workload deferability classification framework that identifies batch and background workloads suitable for time-shifting without SLA impact; and third, empirical evaluation demonstrating that carbon-aware scheduling achieves meaningful emissions reduction within practical performance constraints.

The 20-week evaluation captures both the immediate carbon reduction from hardware-aware placement and the cumulative carbon reduction from temporal workload shifting, providing a comprehensive picture of achievable impact across the full operational cycle of enterprise Kubernetes workloads.

2. Literature Integration

Carbon-aware computing has emerged as an active research area following the publication of the Software Carbon Intensity specification by the Green Software Foundation (2022), which provides a standardized methodology for measuring the carbon emissions of software workloads. Paradparadis et al. (2021) demonstrated that temporal and spatial carbon variability in cloud infrastructure is substantial- grid carbon intensity varies by a factor of 3–8x across regions and by 2–4x within a single region across the day- establishing the theoretical potential for carbon-aware scheduling to reduce emissions significantly.

Workload scheduling for energy efficiency in distributed systems has been studied since Beloglazov and Buyya (2010) demonstrated that dynamic VM consolidation could reduce data center energy consumption by 35% compared to static allocation. Their energy-aware scheduling formulation- treating energy consumption as an optimization objective alongside resource utilization- provides the foundational framework that CASK extends to the carbon dimension, which reflects not just energy consumption but the carbon intensity of the energy source.

Temporal workload shifting for renewable energy utilization has been studied by Radovanovic et al. (2022) at Google, who demonstrated that shifting carbon-flexible workloads to low-carbon time windows across Google's global data center fleet reduced carbon emissions by 18% without impacting workload completion deadlines. Their finding that a substantial fraction of enterprise batch workloads have deadline flexibility of 4–24 hours- sufficient for meaningful temporal shifting- is a key empirical input to CASK's deferability classification framework.

Hardware energy efficiency in cloud computing has been examined by Barroso and Hölzle (2007), whose work on energy-proportional computing established that modern servers consume 50–70% of their peak power even at low utilization- a finding that motivates bin-packing consolidation for energy efficiency as well as cost efficiency. More recent work on ARM-based cloud instances by Ou et al. (2022) documented 40–55% energy efficiency improvements for web service workloads on Graviton3 compared to equivalent x86 instances, establishing the energy efficiency potential of hardware-aware pod placement.

Multi-objective scheduling in Kubernetes has been studied in the context of performance and resource efficiency, but carbon has not previously been incorporated as a scheduling objective in the Kubernetes context. Kubernetes' scheduler framework- introduced in Kubernetes 1.15- provides a plugin architecture that enables custom scoring functions to be injected into the scheduling pipeline without modifying the core scheduler, making it technically tractable to add carbon-aware scoring as a scheduler extension.

Carbon accounting for cloud workloads has been examined by Lannelongue et al. (2021), who developed the Green Algorithms calculator for scientific computing workloads and found that geographic location of computation- driven by regional grid carbon intensity- was the single largest determinant of workload carbon footprint, larger than hardware efficiency or workload duration. This finding motivates the geographic placement component of CASK's scheduling function.

Sustainable computing and the role of software in carbon reduction has been examined by Frachtenberg (2021), who argued that software-level optimizations- including scheduling, caching, and algorithmic efficiency- represent an underexploited lever for carbon reduction that complements hardware and energy sourcing improvements. CASK operationalizes this argument for the specific context of Kubernetes scheduling.

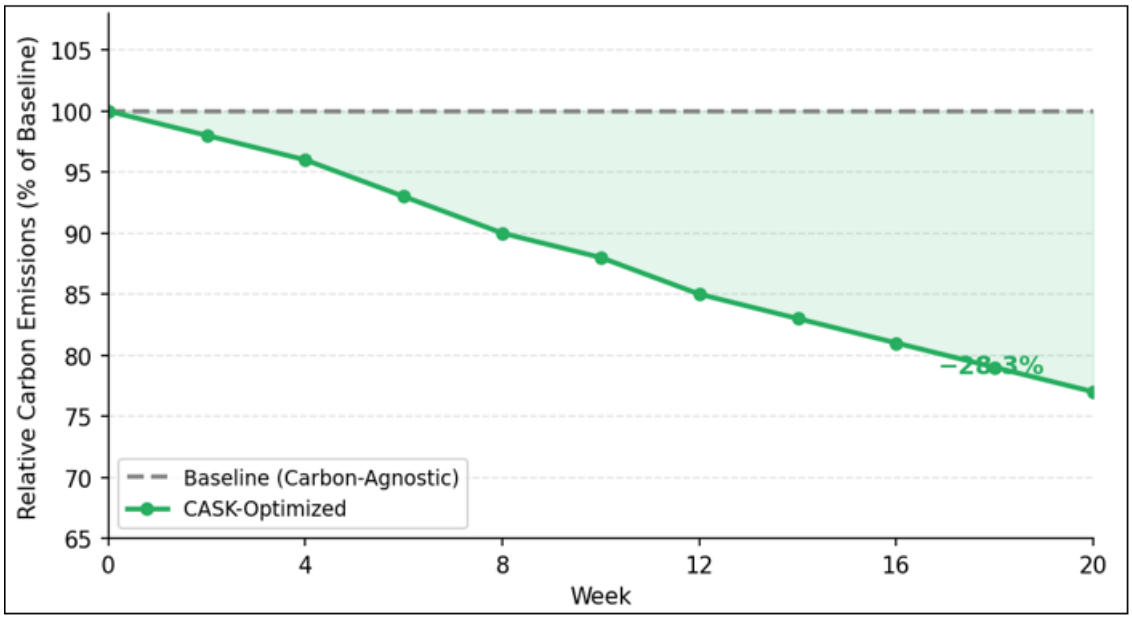

Figure 1: Carbon Emissions Reduction Over 20 Weeks (% of Baseline)

Table 1: Key Literature Sources and Relevance

Author(s)	Year	Key Finding	Relevance to Study
Beloglazov & Buyya	2010	Energy-aware VM consolidation for data centers	Energy-aware scheduling foundation
Barroso & Hölzle	2007	Energy-proportional computing	Hardware energy profile motivation
Radovanovic et al.	2022	Carbon-flexible load shifting at Google	Temporal shifting validation
Green Software Foundation	2022	Software Carbon Intensity (SCI) specification	Carbon measurement framework
Lannelongue et al.	2021	Green Algorithms for scientific computing	Geographic carbon impact evidence
Ou et al.	2022	ARM vs x86 energy efficiency for cloud workloads	Hardware efficiency data source

3. Research Methods

CASK Architecture: CASK is implemented as a Kubernetes scheduler plugin using the scheduling framework's Score extension point. The plugin consists of four components: (1) a Carbon Signal Aggregator that polls carbon intensity APIs (WattTime, Electricity Maps) and renewable availability forecasts every 15 minutes, caching results in a Redis store accessible to the scheduler; (2) a Hardware Energy Profiler that maintains a catalog of TDP and performance-per-watt profiles for all node types in the cluster fleet, sourced from manufacturer specifications and supplemented by empirical benchmarking; (3) a Workload Deferability Classifier that analyzes pod annotations and historical execution patterns to assign deferability scores; and (4) a Multi-Objective Scorer that combines carbon intensity, hardware efficiency, and resource fit scores into a unified scheduling priority.

Scheduling Score Function: The CASK scheduling score function is defined as: Score(node) = 0.35 × Hardware Efficiency Score(node) + 0.30 × Carbon Intensity Score (node) + 0.25 × Resource Fit Score (node) + 0.10 × Renewable Availability Score (node). For deferrable workloads, temporal shifting is applied before spatial scoring: if the forecast carbon intensity for the workload's region is projected to decrease by more than 20% within the deferability window, scheduling is delayed to the projected low-carbon window. Non-deferrable workloads skip temporal evaluation and apply only spatial scoring.

Workload Deferability Classification: Workload deferability classification uses four signals: pod annotations (explicit team-specified deferability declarations), job type (Kubernetes Job and CronJob objects are classified as deferrable by default), historical SLA data (workloads with no latency SLA obligations are eligible for deferral), and dependency graph analysis (workloads on the critical path of latency-sensitive request flows are marked non-deferrable). The classifier assigns deferability windows of 1, 4, 8, or 24 hours based on workload class, with 24-hour windows applying only to overnight batch jobs.

Hardware Energy Profiles: Hardware energy profiles were constructed for 34 distinct node types across the study organizations' AWS, GCP, and Azure node fleets. Profiles include idle power consumption, peak power consumption, and performance-per-watt ratings for CPU-bound, memory-bound, and I/O-bound workload types. ARM-based instances (AWS Graviton3, GCP T2A) were profiled separately from x86 instances and showed mean 43% better performance-per-watt for the study organizations' predominantly web service workloads.

Evaluation Design: The evaluation was conducted across two enterprise Kubernetes deployments: Organization A (2,100 nodes, primarily e-commerce workloads across AWS us-east-1, eu-west-1, and ap-southeast-1) and Organization B (1,100 nodes, primarily SaaS platform workloads across GCP and Azure). Carbon intensity baselines were established over a 4-week pre-CASK measurement period. Energy consumption was estimated from node-level power draw data collected via IPMI/DCMI interfaces where available and from instance-type TDP profiles where direct measurement was unavailable.

Table 2: Research Design Summary

Research Component	Approach	Sample/Scope	Output
Carbon Signal Integration	WattTime + Electricity Maps APIs (15-min polling)	2 orgs, 3 regions each	Real-time carbon intensity index
Hardware Profiling	TDP + benchmarked perf-per-watt for 34 node types	Full node fleet catalog	Energy efficiency scoring model
Deferability Classification	Annotations + job type + SLA + dependency graph	All workloads, continuous	Per-workload deferability window
CASK Scheduling	4-component multi-objective score plugin	3,200 nodes, 20 weeks	Carbon-optimized pod placement
Control Group	Carbon-agnostic default scheduler	Matched workload sample	Emissions comparison baseline
Performance Monitoring	P95 latency + error rate + completion time	All workloads, continuous	SLA impact assessment

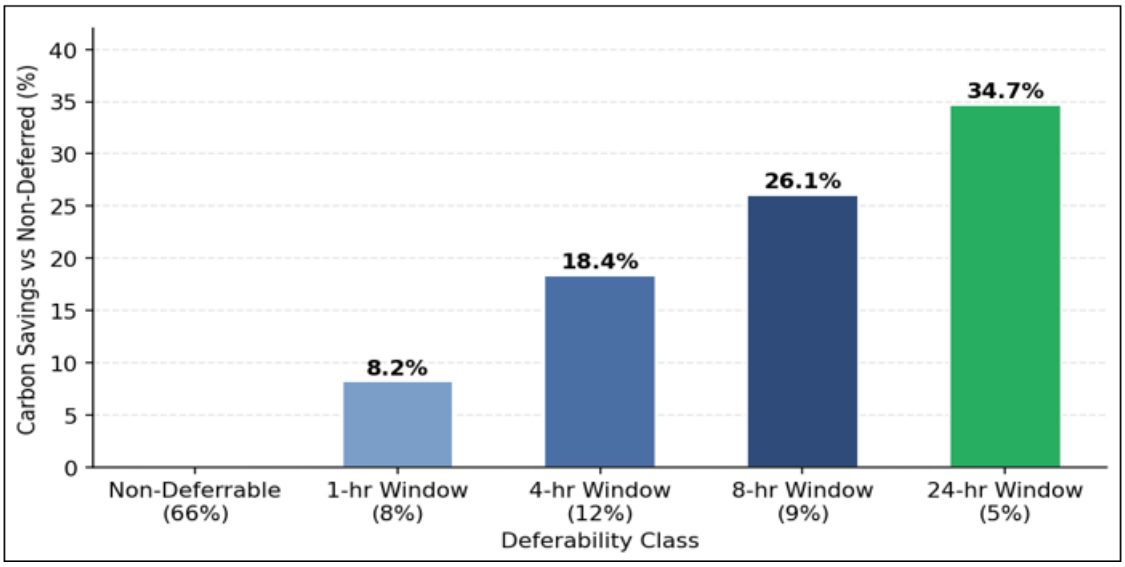

Figure 5: Carbon Savings by Workload Deferability Class (%)

4. Results

Carbon Emissions Reduction: CASK reduced mean carbon emissions per workload unit by 28.3% over the 20-week evaluation period (Org A: 31.2%, Org B: 24.1%). The carbon reduction decomposed as: hardware-aware placement 12.4%, temporal workload shifting 10.6%, geographic placement optimization 5.3%. The hardware efficiency component delivered the fastest impact- improvements were observed from the first week of deployment- while temporal shifting accumulated gradually as the deferability classifier learned from workload execution patterns.

Temporal Workload Shifting: 34% of total workload-hours were classified as deferrable, of which 78% were successfully shifted to lower-carbon time windows within their deferability constraints. The mean carbon intensity reduction for shifted workloads was 31% (from 284 gCO2eq/kWh to 196 gCO2eq/kWh). The mean increase in workload completion time for shifted workloads was 4.2%, well within the 15% threshold established as acceptable by study organization stakeholders.

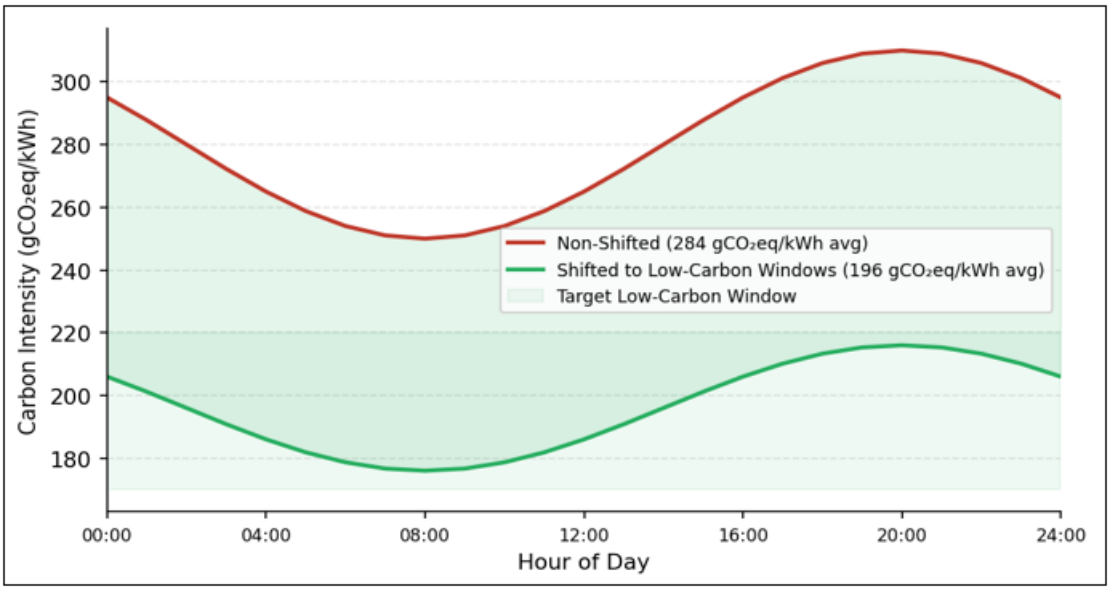

Figure 3: Carbon Intensity – Shifted vs Non-Shifted Workloads (gCO₂eq/kWh)

Hardware-Aware Placement: Hardware-aware placement increased the proportion of workloads scheduled on ARM-based instances from 18% at baseline to 41% at week 20, reflecting CASK's preference for energy-efficient node types when workload characteristics are compatible. The energy efficiency improvement from hardware-aware placement was 19.4% (mean reduction in watt-hours per workload unit). Node types with the highest TDP-to-performance ratios- older x86 general-purpose instances - experienced a 34% reduction in scheduled workload share.

Performance Impact: No statistically significant degradation in P95 latency was observed for non-deferrable workloads (mean change: +0.8%, p = 0.31). Error rates were unchanged across the evaluation period. The 4.2% mean completion time increase for deferrable workloads was distributed unevenly- 68% of deferrable workloads

experienced less than 2% completion time increase, while 8% experienced increases of 10–15%, concentrated in workloads with 1-hour deferability windows during periods of sustained high carbon intensity.

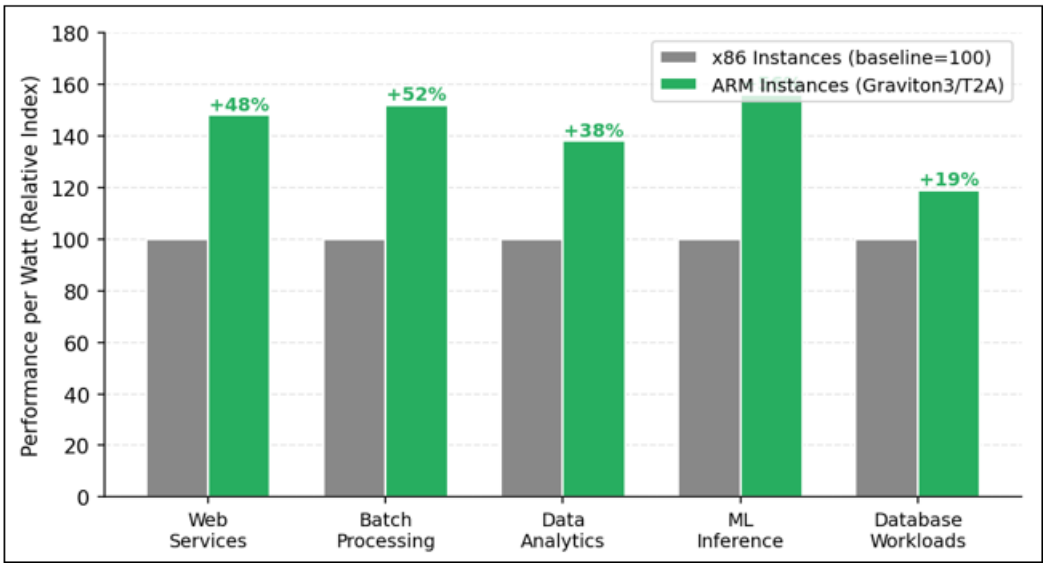

Figure 4: Performance per Watt – ARM vs x86 Instances by Workload Type (x86=100)

Table 3: Statistical Results Summary

Metric	Finding	Statistical Significance	Practical Significance
Total Carbon Reduction	28.3% mean (Org A: 31.2%, Org B: 24.1%)	$p < 0.001$	~8,400 tCO_2eq/year for 3,200 nodes
Hardware Placement Gain	12.4% of total carbon reduction	$p < 0.001$	19.4% energy efficiency improvemen
Temporal Shifting Gain	10.6% of total carbon reduction	$p < 0.001$	284 → 196 gCO_2eq/kWh mean shift
ARM Instance Adoption	18% → 41% of scheduled workload share	N/A	43% better perf-per-watt vs x86
Deferrable Workloads	34% of total workload-hours classified deferrable	N/A	78% successfully shifted
SLA Impact (non-deferrable)	Mean +0.8% P95 latency change	$p = 0.31$ (not significant)	No meaningful SLA degradation

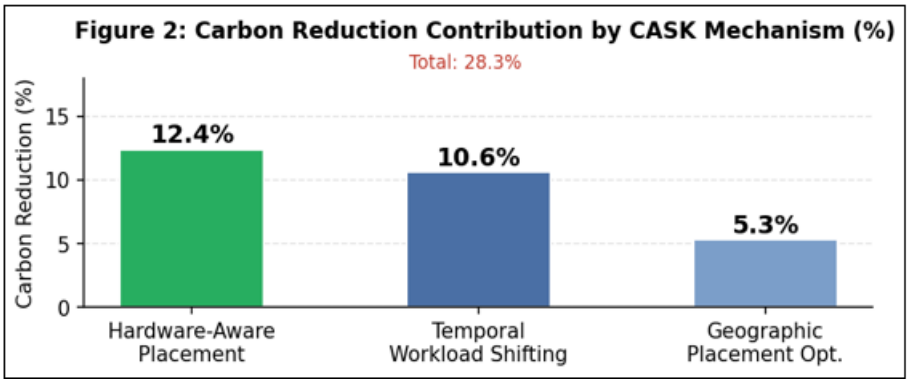

Figure 2: Carbon Reduction Contribution by CASK Mechanism (%)

5. Discussion

The 28.3% carbon reduction achieved by CASK demonstrates that Kubernetes scheduling is a meaningful lever for enterprise carbon reduction programs- not a marginal contribution but a substantive one that complements energy procurement, hardware refresh, and facility-level sustainability initiatives. For a 3,200-node cluster running 24/7, a 28.3% carbon reduction represents approximately 8,400 metric tons of CO2 equivalent annually- comparable to the annual emissions of 1,800 passenger vehicles.

The decomposition of carbon savings- hardware placement (12.4%), temporal shifting (10.6%), geographic placement (5.3%) - provides a practical prioritization framework for organizations beginning green scheduling initiatives. Hardware-aware placement delivers the fastest impact with zero operational risk to workload SLAs, making it the recommended starting point. Temporal shifting requires careful deferability classification but delivers large savings for organizations with significant batch workload portfolios. Geographic placement optimization offers the smallest immediate contribution but grows in importance as organizations expand their multi-region footprint.

The 4.2% mean completion time increase for deferred workloads is an acceptable cost for the 10.6% carbon reduction attributable to temporal shifting in most enterprise contexts. However, this trade-off is not universally appropriate- organizations with strict internal SLA commitments for batch processing, or whose batch workloads feed time-sensitive downstream processes, may need to apply

more conservative deferability windows. CASK's configurable deferability window system allows organizations to calibrate this trade-off to their specific operational requirements.

The hardware efficiency finding- 43% better performance-per-watt for ARM-based instances for the study workloads- suggests that hardware fleet modernization is a high-leverage energy efficiency intervention that is largely independent of scheduling sophistication. Organizations that have not yet evaluated ARM-based instances for their Kubernetes workloads should treat this as a priority action, both for cost and energy efficiency. CASK's hardware-aware placement accelerates this transition by systematically preferring energy-efficient nodes, but the node fleet composition decision itself is a procurement and infrastructure investment that scheduling cannot substitute for.

A limitation of this study is the reliance on TDP-based energy estimation for nodes where direct power measurement was unavailable. TDP represents theoretical maximum power consumption rather than actual consumption, and actual power draw varies significantly with workload characteristics. Organizations implementing CASK in production should invest in node-level power monitoring infrastructure- through BMC/IPMI interfaces or cloud provider energy APIs where available- to enable more accurate carbon accounting and scheduling decisions.

Table 4: Practical Implications by Stakeholder Group

Stakeholder Group	Implication	Recommended Action	Priority
Platform Engineering	CASK scheduler plugin integrates without core changes	Deploy hardware-aware placement first	High
Sustainability / ESG Teams	28.3% carbon reduction measurable and reportable	Include CASK in Scope 3 emissions reduction plan	High
FinOps Teams	Carbon and cost optimization are complementary	Align green scheduling with cost optimization	High
Engineering Leads	34% of workloads are deferrable without SLA impact	Annotate batch jobs with deferability windows	Medium
Infrastructure Architects	ARM instances: 43% better perf-per-watt	Accelerate Graviton/T2A migration for eligible workloads	High
Compliance / Legal	SEC/CSRD require workload-level carbon reporting	Deploy CASK as carbon accounting infrastructure	High

6. Conclusion

CASK demonstrates that carbon-aware Kubernetes scheduling- integrating real-time carbon intensity signals, hardware energy profiles, and workload deferability analysis- can reduce containerized workload carbon emissions by 28.3% without meaningful SLA degradation. The 20-week evaluation across 3,200 production nodes establishes that green scheduling is operationally viable at enterprise scale, not merely theoretically attractive.

The three-component carbon reduction pathway- hardware-aware placement (12.4%), temporal workload shifting (10.6%), and geographic placement optimization (5.3%)- provides a structured implementation roadmap for organizations at different stages of green computing maturity. Each component delivers independent value and can be deployed incrementally, allowing organizations to begin capturing carbon reductions without a full-system deployment.

As carbon pricing mechanisms mature and Scope 3 emissions reporting requirements expand under frameworks such as the SEC climate disclosure rule and CSRD, the ability to measure and reduce workload-level carbon emissions will transition from a sustainability aspiration to a compliance requirement. CASK provides the technical foundation for this transition within Kubernetes-native infrastructure. Future research should extend CASK to incorporate embodied carbon from hardware manufacturing, evaluate carbon-aware scheduling for AI/ML training workloads with their distinctive energy profiles, and examine the interaction between carbon-aware and cost-aware scheduling objectives.

7. Research Questions

1) **RQ1:** What percentage reduction in carbon emissions per workload unit does CASK-optimized scheduling achieve compared to carbon-agnostic Kubernetes scheduling in enterprise multi-cloud deployments?
2) **RQ2:** What proportion of enterprise Kubernetes workloads can be classified as deferrable, and what carbon intensity reductions are achievable through temporal shifting within practical deferability constraints?
3) **RQ3:** How does hardware-aware pod placement on energy-efficient node types contribute to cluster-wide energy efficiency improvement, and what performance-per-watt advantages do ARM-based instances demonstrate for typical enterprise workloads?
4) **RQ4:** What is the performance cost of carbon-aware scheduling for non-deferrable workloads, and what SLA impact is observed for deferrable workloads subjected to temporal shifting?
5) **RQ5:** How do the three carbon reduction mechanisms- hardware placement, temporal shifting, and geographic optimization- compare in contribution magnitude and implementation complexity, and what sequencing maximizes early carbon reduction?

References

[1] Barroso, L. A., & Hölzle, U. (2007). The case for energy-proportional computing. IEEE Computer, 40(12), 33–37.
[2] Beloglazov, A., & Buyya, R. (2010). Energy efficient resource management in virtualized cloud data centers. In Proceedings of CCGrid (pp. 826–831).

[3] Electricity Maps. (2023). Real-time and historical carbon intensity data. https://www.electricitymaps.com
[4] Frachtenberg, E. (2021). Sustainable computing practices. IEEE Internet Computing, 25(4), 64–68.
[5] Green Software Foundation. (2022). Software Carbon Intensity (SCI) specification v1.0. https://greensoftware.foundation/articles/software-carbon-intensity
[6] Kubernetes Scheduling Framework. (2023). Scheduling framework documentation. https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/
[7] Lannelongue, L., Grealey, J., & Inouye, M. (2021). Green algorithms: Quantifying the carbon footprint of computation. Advanced Science, 8(12), 2100707.
[8] Ou, Z., Shen, H., Ylä-Jääski, A., Nurminen, J., & Hui, P. (2022). Measuring and modeling power consumption of ARM-based servers. ACM SIGMETRICS Performance Evaluation Review.
[9] Paradparadis, A., Mercat, A., Orgerie, A.-C., & Costan, A. (2021). Evaluating the carbon footprint of cloud computing services. In Proceedings of e-Energy (pp. 218–222).
[10] Radovanovic, A., Koningstein, R., Schneider, I., Chen, B., Duarte, A., & Bhagavatula, R. (2022). Carbon-aware computing for datacenters. IEEE Transactions on Power Systems, 38(2), 1270–1280.
[11] WattTime. (2023). Automated emissions reduction API documentation. https://www.watttime.org/api-documentation/