



PDF Download
3773274.3774698.pdf
07 January 2026
Total Citations: 0
Total Downloads: 30

 Latest updates: <https://dl.acm.org/doi/10.1145/3773274.3774698>

RESEARCH-ARTICLE

Power Aware Cluster Orchestration: Taxonomy, Initial Results, and Challenges

ILEET MALLA, Umeå University, Umea, Västerbotten, Sweden

THIJS METSCH, Intel Corporation, Santa Clara, CA, United States

PAUL TOWNEND, Umeå University, Umea, Västerbotten, Sweden

Open Access Support provided by:

Intel Corporation

Umeå University

Published: 01 December 2025

[Citation in BibTeX format](#)

UCC '25: 2025 IEEE/ACM 18th
International Conference on Utility and
Cloud Computing
December 1 - 4, 2025
France, France

Conference Sponsors:
SIGARCH

Power Aware Cluster Orchestration: Taxonomy, Initial Results, and Challenges

Ileet Malla
Umeå University
Umeå, Sweden
ileet.malla@umu.se

Thijs Metsch
Intel Corporation
Neubiberg, Germany
thijs.metsch@intel.com

Paul Townend
Umeå University
Umeå, Sweden
paul.townend@umu.se

Abstract

Compute clusters are major power consumers in Cloud and Edge data centers, making it critical to reduce power usage and costs without compromising service levels objectives. Energy Performance Preference (EPP) settings and CPU frequency scaling can lower power but typically at the cost of reduced performance. When considering clusters with heterogeneous power profiles, it is essential to map workloads to the most suitable profile based on their quality-of-service constraints. Current orchestrators overlook power-profile heterogeneity; this is a particular concern at the Edge, where otherwise identical hardware may range from power-optimized to performance-oriented yet remain indistinguishable to schedulers. We present a taxonomy of power-aware orchestration, and extend the default Kubernetes scheduler with power-profile awareness. We evaluate the feasibility of this extended scheduler by comparing three power profile-aware scheduling strategies on a testbed running a microservices benchmark, with results showing that average power use can be reduced by up to 12% while maintaining application performance. We conclude with key challenges and future research directions.

CCS Concepts

• **Hardware** → **Power and energy**; • **Computer systems organization** → **Cloud computing**; • **Software and its engineering** → **Scheduling**.

Keywords

Power profiles, scheduling, cloud computing, cluster management, kubernetes, heterogeneous clusters

ACM Reference Format:

Ileet Malla, Thijs Metsch, and Paul Townend. 2025. Power Aware Cluster Orchestration: Taxonomy, Initial Results, and Challenges. In *2025 IEEE/ACM 18th International Conference on Utility and Cloud Computing (UCC '25)*, December 01–04, 2025, Nantes, France. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3773274.3774698>

1 Introduction

The power footprint of large-scale data centers [8] is a central concern for both industry and academia. Modern compute clusters consume substantial electricity, not only to run workloads but also

to sustain cooling systems and ensure availability. As cloud services, AI, and large-scale scientific applications expand, demand for computational power rises accordingly, driving costs and amplifying environmental impact. Minimizing power consumption while maintaining acceptable performance is thus a critical challenge [20].

A key factor complicating this challenge is the growing heterogeneity of power configurations across cluster nodes, especially at the Edge [2]. Even with uniform hardware, servers may differ in operating frequency ranges, Energy Performance Preference (EPP) settings, or dynamic voltage and frequency scaling (DVFS) policies. Some nodes favor energy efficiency at reduced frequencies, while others prioritize performance at higher frequencies. Such heterogeneity stems from incremental hardware upgrades and deliberate configuration choices balancing diverse workload needs.

Despite this, orchestration systems such as Kubernetes largely abstract away node-level power characteristics. Clusters are typically described in terms of coarse-grained resources (e.g. CPU cores, memory), assuming identical allocations yield equivalent performance. In practice, allocations can produce different outcomes depending on node power profiles. This abstraction gap introduces inefficiencies, as workloads may be placed in ways that create unnecessary power peaks, degrade performance, or inflate costs, requiring a shift in scheduling design. Beyond resources, schedulers must incorporate awareness of node power profiles and consider workload-specific trade-offs between power and performance. The central research question is therefore:

How can workloads be scheduled across power heterogeneous compute clusters to maximize cluster power efficiency while minimizing performance impact?

We first analyze how scheduling can be improved using only deployment metadata already available in standard manifests, establishing a baseline without modifying orchestration interfaces. We then examine how lightweight extensions, such as hints expressing performance or power preferences, can enable more informed scheduling. Together, these perspectives provide a pathway for integrating power-awareness into schedulers without excessive complexity or disruptive changes to workload models. The contributions of this work are:

- (1) A taxonomy of power-aware orchestration and identification of a key research gap.
- (2) Initial experiments validating the impact of power-aware scheduling on cluster power consumption and application performance.
- (3) A discussion of challenges and next steps for integrating power-awareness into existing scheduling frameworks.



This work is licensed under a Creative Commons Attribution 4.0 International License. *UCC '25, Nantes, France*

© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2285-1/25/12
<https://doi.org/10.1145/3773274.3774698>

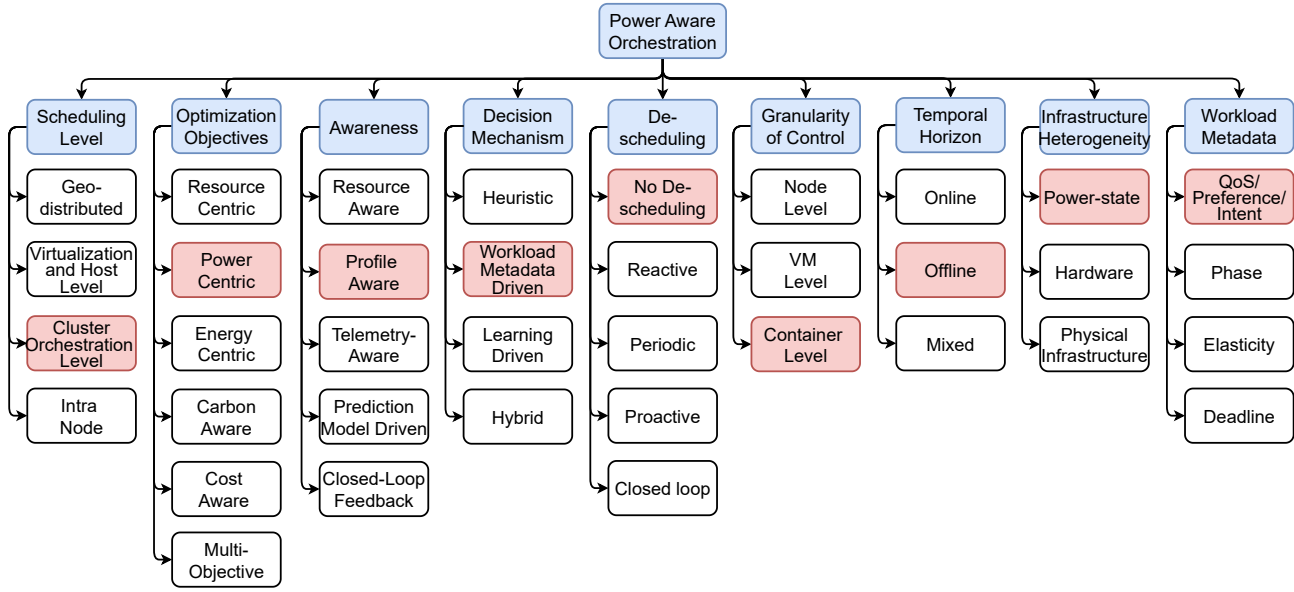


Figure 1: Taxonomy of Power-Aware Orchestration (categories of our proposed solution highlighted in pink)

2 Taxonomy of Power-Aware Orchestration

Power-aware orchestration in cloud and edge systems is inherently multi-dimensional, considering objectives (e.g. utilization, latency, energy reduction), scheduler knowledge (from static profiles to real-time telemetry), mechanisms for adapting to dynamic conditions (heuristics, learning-based models), workload characteristics, infrastructure heterogeneity, temporal granularity, and multi-tenancy constraints. This reveals how controls at the hardware level, consolidation at the cluster level, and workload distribution at the global scale jointly shape the energy footprint of computing infrastructures. Figure 1 captures these dimensions in a unified taxonomy of power-aware orchestration.

One central axis is the **level of scheduling**, which spans multiple layers of control. At the *global or inter-regional* scale, cloud providers decide where to run workloads across geographically distributed datacenters, balancing latency, cost, and grid carbon intensity. These decisions are coarse-grained yet have far-reaching effects on sustainability and cost. Early work on geographical load balancing reduced costs while meeting service guarantees [16], while later systems shifted batch jobs into renewable-heavy windows without missing deadlines [6]. The *virtualization and host* level concerns the mapping of virtual machines to physical servers and the use of mechanisms such as Dynamic Voltage and Frequency Scaling (DVFS), Energy Performance Preference (EPP) modes, and power capping. Systems like VirtualPower [17] is an example of how hypervisors can exploit these mechanisms to regulate server power states.

At the cluster-orchestration level, scheduling decisions are governed by management frameworks that combine infrastructure control with workload placement. On the provider side, cluster managers consolidate workloads, power down idle servers, and apply cluster specific optimization, directly shaping capacity and

energy footprint [4]. On the tenant side, orchestrators such as Kubernetes handle pod-to-node placement, autoscaling, and workload distribution within the available resources. While orchestrators cannot directly modify hardware states, they enforce scheduling strategies and policies such as taints and tolerations, affinity rules, Quality-of-Service classes, and autoscaling, which determine how diverse workloads are mapped onto nodes. Recent work explores energy-aware orchestration, including holistic scheduling that jointly considers placement and hardware-level modes [22], and learning-based techniques such as reinforcement learning for autoscaling [21]. Finally, at the intra-node level, operating systems and runtimes govern fine-grained behavior such as thread scheduling, core pinning, and accelerator states. Proposals such as Linux kernel extensions [15] and accelerator runtime adaptations regulate energy states at the sub-node level, complementing higher-level orchestration.

Optimization objectives reflect the diversity of goals pursued by orchestration frameworks. Conventional systems maximize utilization or throughput, whereas power-centric mechanisms aim to limit instantaneous draw to comply with power delivery or cooling limits. Energy-centric schedulers instead minimize cumulative energy over time, improving proportionality between load and power use. Economic and sustainability objectives add new layers: cost-aware approaches leverage dynamic pricing or spot-market signals [18], while carbon-aware strategies align execution with renewable generation or low-carbon grid periods [9]. Increasingly, orchestrators must pursue *multi-objective optimization*, balancing competing criteria such as energy, cost, latency, and throughput simultaneously [14]. These objectives are dynamic rather than static; changing with user priorities, workload criticality, or regional grid conditions—thus requiring adaptive, context-aware orchestration.

The **scheduler’s awareness** reflects the extent of information it possesses about system conditions, directly influencing the quality and precision of its decisions. Resource-only designs consider CPU and memory requests but ignore energy implications. Profile-aware systems extend this with static knowledge of node capabilities (e.g., frequency ranges, energy-performance profiles). Telemetry-aware schedulers integrate real-time signals such as utilization, power, and temperature, while predictive designs anticipate workload demand or renewable availability. Closed-loop feedback mechanisms dynamically adjust placement and scaling to maintain power budgets or service objectives [5].

Decision mechanisms define how awareness translates into action. *Heuristic strategies* (e.g., power-first or best-fit) follow fixed rules that are computationally efficient but often too rigid for dynamic environments. *Model-driven approaches* use analytical formulations to evaluate trade-offs between power, migration, and performance costs [4], offering greater transparency but at higher computational overhead. *Workload-metadata-driven mechanisms* integrate application-level information directly into scheduling logic, allowing placement and scaling decisions that align with workload semantics [12]. *Learning-based approaches* [21] employ reinforcement or supervised learning to adapt over time to changing workloads and system states, while *hybrid mechanisms* combine heuristic stability, model-based reasoning, and learning-based adaptability in tiered or multi-phase decision pipelines. The optimal choice of mechanism depends on system scale, workload volatility, and the acceptable balance between runtime overhead, explainability, and adaptability.

The **de-scheduling dimension** characterizes how orchestration revisits or revises scheduling decisions. Static schedulers execute one-shot placements and never re-evaluate them. *Reactive strategies* trigger migrations or reconfiguration in response to SLA violations or overloads. *Periodic strategies* re-evaluate allocations at fixed intervals, while *proactive* or *predictive* systems use workload and energy forecasts to act ahead of changes. *Closed-loop* designs perform continuous adaptation, adjusting workloads in real time to maintain target power or performance levels. Power capping controllers [11] exemplify continuous adaptation, while intent-driven orchestration extends this paradigm further by dynamically reconfiguring deployments to uphold declared service intents such as latency bounds or energy budgets [12]. Such adaptivity is particularly valuable for cloud-native and edge environments, where workload volatility and power constraints are both high.

Granularity of control and temporal horizon influence how quickly and precisely orchestrator can respond. Coarse-grained node-level control offers stability and lower overhead but lacks flexibility to react to micro-level variations. Fine-grained orchestration at the VM or container level supports precise adaptation and micro-scaling but increases scheduling complexity and migration cost. Temporal granularity similarly defines the reaction timescale: *online* schedulers react immediately to telemetry, *offline* or batch schedulers optimize over long planning windows, and *hybrid* systems combine both—using offline optimization for global trends and online loops for local responsiveness.

Infrastructure heterogeneity complicates orchestration but also opens new opportunities for optimization. Beyond architectural diversity across CPUs, accelerators, and memory hierarchies,

an increasingly relevant form is *power-state heterogeneity*, where nodes operate under different EPP modes or CPU frequency caps, leading to distinct power-performance envelopes [13]. Exploiting this requires schedulers to account for node-specific power profiles when placing workloads—mapping latency-critical services to high-performance nodes and energy-sensitive or background tasks to power-efficient ones. This perspective extends to physical-infrastructure diversity, including cooling technologies, renewable-powered clusters, and network topology, all of which influence both energy efficiency and sustainability. Treating such characteristics as active scheduling dimensions enables orchestrators to reason about power and performance as dynamic, context-dependent attributes rather than static capacities.

Finally, **workload metadata** enriches orchestration with signals ranging from Quality-of-Service (QoS) classes to preferences and explicit intents. Different workloads exhibit varying sensitivities to latency, throughput, and energy throttling. QoS classes provide a baseline mechanism to categorize workloads by priority or resource guarantees, but they express limited semantics. *Preferences* extend this model by conveying soft hints or relative priorities (e.g., “prefer energy-efficient nodes” or “favor low latency”), enabling schedulers to optimize according to user or application tendencies without enforcing strict guarantees. *Intents* go further by representing declarative goals that specify desired outcomes—such as “maintain p99 latency <200 ms” or “stay within a 100 W power budget” [12]. The orchestration system then interprets and enforces them through adaptive placement and scaling decisions. Additional metadata, such as workload *phase* (e.g., training vs. inference) or elasticity characteristics (e.g., scale-out/scale-in-friendly, stateful or stateless), enable even finer differentiation. Embedding such metadata transforms orchestration from a purely resource-based process into a context-aware control loop that aligns system behavior with user objectives and situational constraints.

Prior taxonomies and surveys have provided valuable perspectives on energy-efficient orchestration but differ in focus and granularity. The Cloud-to-Things taxonomy [23] emphasizes orchestration focusing on automated deployment and run-time management rather than power and performance aspect. The sustainable IoT computing taxonomy [1] focuses on device-level energy optimization, while the carbon-aware container orchestration survey [24] classifies sustainability strategies by hardware, software, and policy scope. In contrast, our taxonomy unifies these perspectives by explicitly connecting low-level power management mechanisms (e.g., DVFS, EPP, power capping) with orchestration-level constructs such as QoS, preferences, and intents. It also aligns awareness dimensions (resource, profile, telemetry) with their corresponding decision mechanisms (heuristic, analytical, learning-driven, closed-loop), providing a coherent framework for analyzing and comparing orchestration systems.

Overall, this taxonomy demonstrates that power efficiency, stability, and predictability arise not from individual techniques but from their coordinated interplay across system layers. By examining scheduling levels, objectives, awareness, mechanisms, and heterogeneity together, it provides a conceptual foundation for designing future orchestration frameworks that intelligently balance power, performance, and sustainability in heterogeneous, dynamically evolving cloud and edge environments.

3 Research Gap

Although research on power-aware orchestration has advanced considerably, three key dimensions remain insufficiently addressed in combination: *infrastructure heterogeneity*, *scheduler awareness of that heterogeneity*, and *expressive workload metadata*. Heterogeneity differs in its interpretation across prior studies. Most existing work [19] considers it primarily in terms of hardware diversity, such as variations in processor architectures, accelerators, or networks. PAX [3], a performance and energy-aware Kubernetes scheduler designed for heterogeneous clusters, employs Bayesian optimization to jointly tune pod scaling and placement. While it accounts for heterogeneity, its focus lies on hardware-generation variation, primarily by combining older and newer servers.

By contrast, *power-profile heterogeneity* is under-explored. Even with identical hardware, nodes may run under different EPP modes or frequency caps, producing distinct power–performance envelopes. Scheduler-level control of CPU states directly influences energy and performance [13], yet orchestrators typically ignore these differences, treating nodes as equivalent once CPU and memory are satisfied. Recognizing and leveraging this latent form of heterogeneity is critical for achieving fine-grained, power-aware orchestration.

On the workload side, schedulers rely mainly on *resource-centric metadata*, typically CPU and memory requests. While useful for admission control, these values offer no view of workload sensitivity to power–performance trade-offs. Kubernetes maps these into *Quality-of-Service (QoS) classes*—Guaranteed, Burstable, and BestEffort [10]—which may distinguish latency-critical from background tasks, but only if developers specify accurate limits. This burdens developers, whose manual tuning is error-prone and often fails to reflect real priorities such as latency or business importance. As a result, QoS alone often misaligns with application objectives.

A richer path is to extend metadata with *preferences*, expressing soft hints (e.g., “favor efficiency”), and *intents*, capturing explicit service goals (e.g., p99 latency < 200 ms). Intents can guide adaptive orchestration [12], while recent systems such as GreenKube [21] apply machine learning and forecasting to enhance scheduling. Yet these systems either tune server configurations per workload or overlook combining expressive metadata with scheduler awareness of node-level power diversity.

This missing intersection — expressive metadata plus heterogeneous profiles — defines the gap we address. We develop strategies that exploit node-level power profiles while being guided by workload metadata from QoS classes to preferences. We show schedulers can save power without sacrificing performance and without complex predictive models or disruptive workflow changes. While our study focuses on intra-cluster scheduling, the same principles extend to multi-cluster or geo-distributed settings, where additional factors (e.g. inter-cluster heterogeneity, carbon intensity, electricity price, regional latency) apply. Addressing power-profile heterogeneity at the single-cluster level provides both a concrete foundation and a step toward broader orchestration.

Within this scope, scheduling strategies can be grouped into two categories. *Static strategies* are heuristic and rule-based, e.g. the *Power-First* approach, which fills low-power nodes first. This reduces power draw but risks poor performance for latency-sensitive workloads and leaves high-performance nodes idle [4]. In contrast,

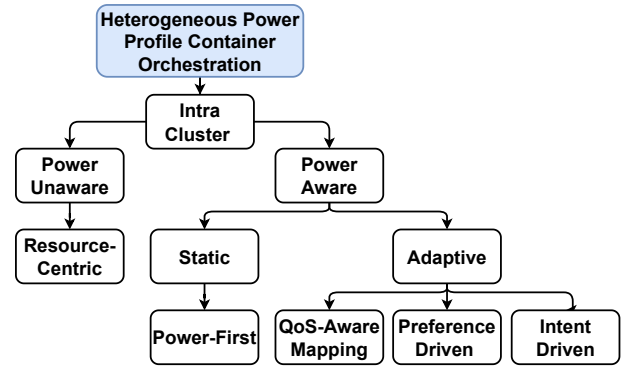


Figure 2: Orchestration in a Heterogeneous Power-Profile Cluster

adaptive strategies incorporate workload metadata to guide placement more intelligently, e.g. QoS-aware mapping (aligning QoS categories with node profiles but relying on accurate specification), preference-driven scheduling (using soft hints to balance trade-offs), and intent-driven scheduling (explicit service objectives are matched to the minimum-power profile that satisfies them). Adaptive approaches thus offer stronger trade-offs between energy efficiency and predictable performance.

Figure 2 represents orchestration strategies in a single-cluster, contrasting static heuristics with metadata-driven adaptive strategies. Our work targets this underexplored intersection, building schedulers exploiting configuration diversity while guided by metadata can achieve meaningful power savings without compromising performance.

4 Solution

Our proposed solution addresses the underexplored scheduling gap at the intersection of workload metadata and node-level power-profile diversity. Our design exposes heterogeneous power profiles to the orchestration layer and extends scheduling with workloads metadata. The taxonomy shown in Figure 1 highlights the categories of our solution and Figure 3 shows the overall system architecture. At the application layer, workloads are encapsulated in pods (the fundamental unit of Kubernetes deployment) containing one or more tightly coupled containers that share the same network namespace and storage volumes.

Each pod is described by a declarative manifest specifying container images, replicas, storage volumes, and CPU/memory requests. These requests guide admission control and map into QoS classes which provide a coarse measure of workload criticality. QoS is limited, so our design allows manifests to carry additional metadata: *preferences* (e.g., “favor efficiency” or “favor performance”) and *intents* encoding explicit objectives such as p99 latency < 200 ms. This model ensures scheduling reflects both quantitative requirements and qualitative workload priorities.

Pods are submitted through the *kube-apiserver*, the central gateway of the control plane. The API server exposes them to other components, and persists state in *etcd*, a strongly consistent distributed key–value store. Worker nodes then host pod execution.

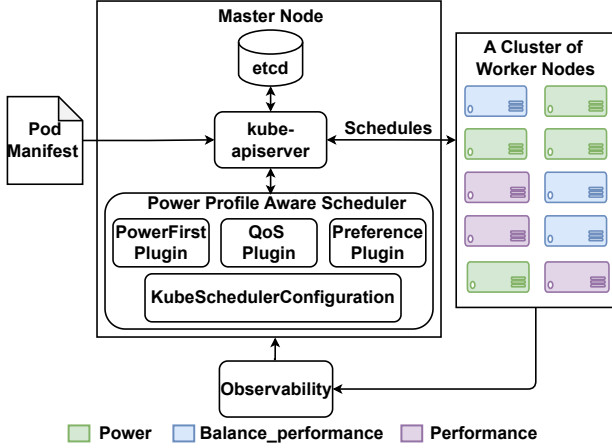


Figure 3: System Architecture of the Power Profile Aware Scheduler

Although architecturally identical, nodes operate under distinct EPP modes and CPU frequency caps, exposing *power profiles* as node labels. This provides a lightweight mechanism for incorporating heterogeneity into orchestration decisions without kernel or runtime modifications.

Scheduling is handled by a *power profile aware scheduler*, extending Kubernetes’ *Filter*, *Score*, and *Bind* phases. Nodes failing CPU/memory requirements are excluded in filtering, and custom plugins then combine node labels with pod metadata to score candidates. The top-scoring node is bound, with plugin registration managed through the *KubeSchedulerConfiguration*, enabling multiple strategies to be tested within one framework.

Three plugins represent distinct strategies: *PowerFirst*, a static heuristic preferring low-power nodes; *QoS*, mapping QoS classes to profiles; and *Preference*, interpreting user annotations to bias toward efficiency or performance. Intent-aware scheduling remains a forward-looking extension, envisioned to allow explicit service-level objectives to be matched with the minimum-power profile that can meet them.

Finally, an observability layer collects telemetry on resource use, power, and application performance. Currently used for offline analysis, it lays the foundation for future closed-loop scheduling where real-time feedback can drive adaptive placement and migration.

5 Implementation

We utilize a Kubernetes (k8s)-based testbed (version v1.29.15) with one master node and three worker nodes, each equipped with an Intel(R) Core(TM) i5-7600T CPU @ 2.80 GHz. Hardware is identical; heterogeneity is introduced through distinct EPP modes and CPU frequency capping. We use 3 power profiles, summarized in Table 1. Profile A caps frequency at 1600 MHz to emulate a low-power configuration; Profile B operates at a base frequency of 2800 MHz; Profile C scales up to 3700 MHz under Turbo Boost for performance-oriented workloads.

Pods are deployed using declarative manifests specifying container images, replicas, volumes, and resource requirements. In our

Table 1: Node power profile configurations

Profile	Min (MHz)	Max (MHz)	EPP Mode
Profile A	800	1600	Power
Profile B	800	2800	Balance Performance
Profile C	800	3700	Performance

implementation, manifests may also include metadata such as QoS classes, preferences, or intents, which are interpreted by scheduler plugins during node scoring.

For evaluation, we use the Google Microservices Benchmark [7], a realistic service-oriented application of ten interdependent microservices. This workload stresses the system with diverse latency and throughput requirements, making it suitable for studying trade-offs between energy consumption and responsiveness under different scheduling strategies. Cluster-wide power consumption is defined as:

$$P_{\text{cluster}} = \sum_{i=1}^N P_{\text{worker}_i}, \quad (1)$$

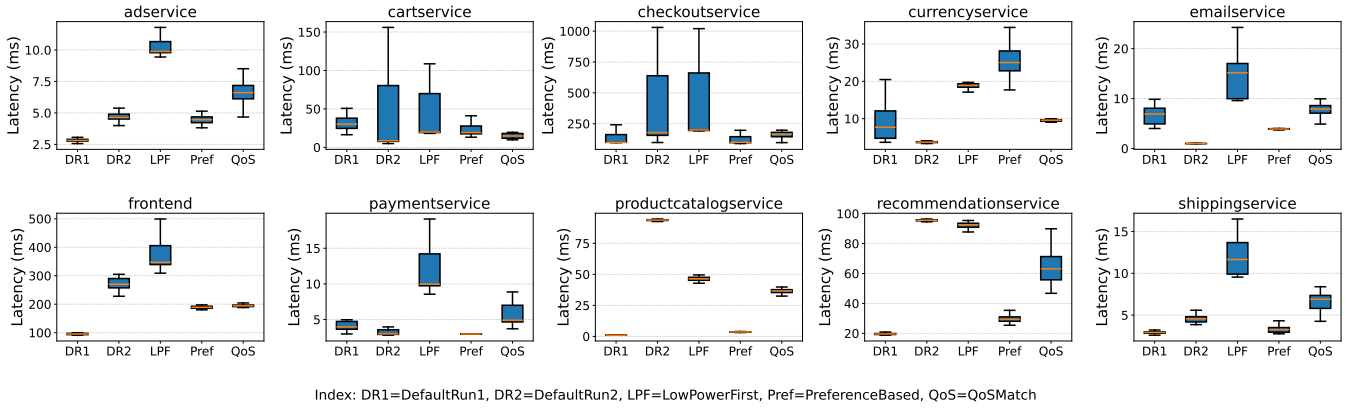
where P_{cluster} is total cluster power, P_{worker_i} the draw of the i^{th} worker node, and N the number of worker nodes. Both active and idle states are included, since powered-on nodes consume energy even when idle. The master node is excluded because its control-plane services (API server, etcd, scheduler) contribute negligible power relative to workers. Power is measured with Intel’s Running Average Power Limit (RAPL) interface and cross-validated against a physical meter.

Three custom score plugins are implemented to realize taxonomy categories. *PowerFirst* embodies a static heuristic, always prioritizing low-power nodes; *QoS* aligns pod QoS classes with node profiles, representing an adaptive strategy leveraging workload classification; *Preference* interprets explicit metadata annotations to bias placement toward efficiency or performance. All plugins are registered through *KubeSchedulerConfiguration* and invoked during filtering or scoring. The default k8s scheduler serves as a baseline, representing a power-unaware, resource-centric approach.

Monitoring and observability are provided by an instrumentation stack. Prometheus collects pod and node level metrics, including CPU utilization, scheduling events, and power readings. Grafana offers dashboards for real-time visualization, and Linkerd, a lightweight service mesh, instruments inter-service communication to expose latency, throughput, and failures. Together, these tools provide comprehensive visibility into power consumption and application-level performance, enabling evaluation of each scheduling strategy.

6 Initial Results and Evaluation

We evaluate our taxonomy of scheduling strategies using the Google Microservices Benchmark running on our heterogeneous k8s cluster. Among the 10 microservices, *frontend* is most critical as it is the entry point, so we primarily evaluate this microservice. Fig. 4 presents the distribution of the 99th percentile (P99) latency for all microservices and Fig. 5 presents total cluster power consumption. For clarity, extreme outliers are removed from both figures.



Index: DR1=DefaultRun1, DR2=DefaultRun2, LPF=LowPowerFirst, Pref=PreferenceBased, QoS=QoSMatch

Figure 4: P99 latency distribution

Across 10 runs with the default k8s scheduler (power-unaware and resource-centric), the 2 most extreme variations are shown in the first two box plots and represent the baseline scheduler. While both executions produced comparable power consumption (28-32W), their latency distributions differ considerably. In one case, the interquartile range (IQR) is narrow (median below 100 ms), indicating stable performance. In the other, IQR widens substantially (shifting the median above 200 ms), and the whiskers extended beyond 250 ms. This pattern is reflected in Table 2, where the mean latency of DefaultRun1 is 100.76 ms (standard deviation of 18.57 ms), while DefaultRun2 has a much higher mean latency of 303.11 ms (standard deviation of 105.20 ms), **highlighting the instability of the baseline scheduler.**

The LowPowerFirst strategy (a static heuristic) achieves the lowest total cluster power consumption of 18W by prioritizing scheduling on nodes configured with energy-saving profiles. However, latency distribution shifts unfavorably: median latency increases to nearly 400 ms, IQR widens, and whiskers indicate larger variability. The table confirms this trend, with a mean latency of 383.25 ms and a standard deviation of 79.16 ms, reflecting the limitation

of static heuristics, which reduce power consumption but disregard workload heterogeneity, resulting in significant performance degradation.

In contrast, the Preference-Based and QoS-Match strategies (adaptive schedulers) achieve more favorable trade-offs. By incorporating workload metadata, they consistently align power profiles with workload requirements. The box plots show narrower latency distributions, with medians just above 200 ms and reduced variability compared to LowPowerFirst and baseline runs. PreferenceBased maintains a mean latency of 191.10 ms with a standard deviation of 13.04 ms, while QoSMatch has a mean of 213.33 ms with a standard deviation of 73.99 ms. **Both strategies keep cluster power close to 27W. These adaptive approaches reduced power consumption by up to 12% compared to the default scheduler while maintaining satisfactory latency.**

Table 2: Latency of frontend and cluster power (Mean ± Std Dev)

Scenario	Latency (ms)	Power (W)
DefaultRun1	100.76 ± 18.57	31.56 ± 5.59
DefaultRun2	303.11 ± 105.20	28.30 ± 6.33
LowPowerFirst	383.25 ± 79.16	17.46 ± 3.13
PreferenceBased	191.10 ± 13.04	27.50 ± 6.68
QoSMatch	213.33 ± 73.99	26.78 ± 4.87

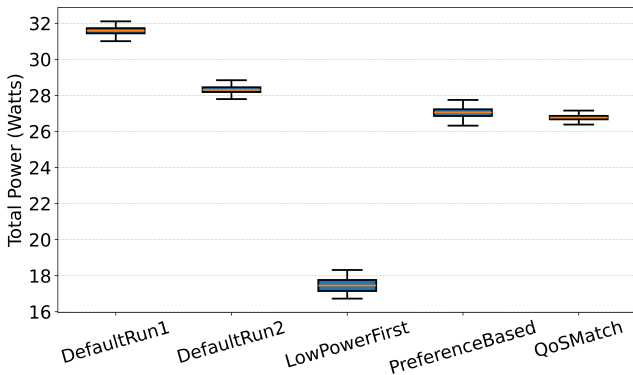


Figure 5: Total cluster power

Overall, our results highlight three key insights. First, power-unaware scheduling produces unpredictable latency distributions, as seen in the divergence of medians and the large standard deviation between baseline runs. Second, static heuristics like LowPowerFirst minimize cluster power but at the cost of a substantial upward shift in both median and mean latency, accompanied by wide variability. Third, adaptive strategies such as Preference-Based and QoS-Match strike the most effective balance; they narrow latency distributions, maintain mean latency around 200 ms with reduced variance, and achieve lower mean power consumption.

7 Challenges and Future Directions

Our taxonomy and experiments highlight the potential of power profile-aware scheduling in clusters, but several challenges must be addressed before deployment in production systems; these define promising avenues for future work.

Workload characterization and metadata. Current frameworks expose only CPU and memory requests, insufficient for power-aware decisions. Richer metadata (QoS, preferences, intents) and simple interfaces are needed to capture performance sensitivity without burdening developers.

Mapping and fulfilling intents. Translating metadata into scheduling decisions is challenging. Heterogeneous nodes vary under load, and static profiles may not always meet goals. Dynamic profiling and recommendation mechanisms could ensure workload–profile suitability.

Adaptivity to evolving workloads. Applications rarely operate under fixed conditions; latency or throughput targets shift with demand and changes in the cluster. Future schedulers must adapt continuously through low-overhead reallocation.

Node-level power visibility. Although hardware exposes DVFS, EPP, and RAPL, these signals remain hidden from orchestration layers. Bridging this gap is key for scheduling.

Balancing efficiency, performance, and complexity. Static heuristics are simple but often sacrifice performance, whereas ML and optimization methods capture richer trade-offs yet introduce runtime overhead and deployment complexity. The key challenge is designing strategies that achieve meaningful power savings while sustaining performance.

Capacity planning Cluster power efficiency depends on how nodes are configured. Determining which power profiles to assign to nodes based on expected workloads, performance goals, and power capping remains an open challenge. Effective cluster capacity planning should ensure optimal utilization without over-provisioning, while also meeting performance thresholds.

Ecosystem integration and scalability. Adoption requires compatibility with existing platforms, autoscalers, and multi-cluster frameworks, with extensions kept minimally invasive. Our current evaluation is limited to a small testbed and single workload; future work must assess larger clusters and diverse workloads (e.g., batch jobs, AI/ML) to validate at scale.

8 Conclusion

This work presents a profile-aware scheduling approach for power-heterogeneous clusters, where nodes differ in power–performance trade-offs via EPP modes and DVFS scaling. We propose a taxonomy of power-aware orchestration and corresponding scheduling strategies spanning from power-unaware placement to adaptive, metadata-driven methods guided by QoS, preferences, and intents. Our implementation extends the Kubernetes scheduler to expose node-level power profiles and interpret workload metadata during scheduling, achieving power awareness with minimal changes. Experiments with the Google Microservices Benchmark show that adaptive strategies reduce cluster power use by up to 12% while maintaining stable performance. In contrast, the default scheduler exhibits unstable latency and power behavior, while static heuristics save power but degrade performance. Combining node-level

visibility with workload semantics enables efficient mapping of workloads to suitable profiles—maximizing cluster power efficiency while minimizing performance impact. Future work should focus on richer metadata, incorporating intent-based objectives, enabling dynamic profile switching in response to workload and system dynamics, integrating power telemetry into capacity planning and autoscaling, and evaluating these strategies across larger clusters and diverse workloads. Addressing these directions will enable production-grade power-aware scheduling that adapts to workload variability and reduces costs without compromising performance.

Acknowledgments

This work was supported by the European Commission through the Horizon Europe project SovereignEdge.Cognit under Grant Agreement 101092711.

References

- [1] Mohammed H. Alsharif et al. 2024. A comprehensive survey of energy-efficient computing to enable sustainable massive IoT networks. *Alexandria Engineering Journal* 91 (2024), 12–29.
- [2] Mohammad Sadegh Aslanpour et al. 2024. Load balancing for heterogeneous serverless edge computing: A performance-driven and empirical approach. *Future Generation Computer Systems* 154 (2024), 266–280.
- [3] H. Dong et al. 2025. Towards Performance and Energy Aware Kubernetes Scheduler. *SIGENERGY Energy Inform. Rev.* 5, 2 (Aug. 2025), 7 pages.
- [4] Xiaobo Fan et al. 2007. Power provisioning for a warehouse-sized computer. In *ACM ISCA*. 13–23.
- [5] Anshul Gandhi et al. 2009. Optimal power allocation in server farms. In *ACM ICAC*. 157–166.
- [6] I. Goiri et al. 2011. GreenSlot: scheduling energy consumption in green datacenters. In *ACM HPDC*. 317–326.
- [7] GoogleCloudPlatform. 2025. microservices-demo. Accessed on: 2025-09-15.
- [8] Nicola Jones. 2018. How to stop data centres from gobbling up the world’s electricity. Accessed on: 2025-09-15.
- [9] Young Geun Kim et al. 2023. GreenScale: Carbon-Aware Systems for Edge Computing. *arXiv preprint arXiv:2304.00404* (2023).
- [10] Kubernetes. 2023. Configure Quality of Service for Pods. <https://kubernetes.io/docs/tasks/configure-pod-container/quality-service-pod/>. Accessed: 2025-11-04.
- [11] Charles Lefurgy et al. 2007. Server-level power control. In *IEEE Int. Conference on Autonomic Computing*. 4–11.
- [12] Thijs Metsch et al. 2023. Intent-Driven Orchestration: Enforcing Service Level Objectives for Cloud Native Deployments. *SN Computer Science* 4, 5 (2023).
- [13] Thijs Metsch and David Hoban. 2025. Breaking Barriers: From Black Box to Intent-Driven, User-Friendly Power Management. In *International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*.
- [14] Preethika Pradeep and Eyhab Al-Masri. 2025. Energy-Optimized Scheduling for AIoT Workloads Using TOPSIS. *arXiv preprint arXiv:2506.04902* (2025).
- [15] Feitong Qiao, Yiming Fang, and Asaf Cidon. 2024. Energy-Aware Process Scheduling in Linux. In *HotCarbon Workshop*.
- [16] A. Qureshi et al. 2009. Cutting the electric bill for internet-scale systems. In *ACM SIGCOMM*. 123–134.
- [17] K. Schwan R. Nathuji. 2007. VirtualPower: coordinated power management in virtualized enterprise systems. In *ACM SOSP*. 265–278.
- [18] Samuel Rac et al. 2024. Cost-aware Service Placement and Scheduling in the Edge-Cloud Continuum. *ACM Trans. Archit. Code Optim.* 21, 2, Article 29 (March 2024), 24 pages.
- [19] Isabelly Rocha et al. 2019. HEATS: Heterogeneity- and Energy-Aware Task-based Scheduling. *arXiv:1906.11321* [cs.DC]
- [20] S. Sarkar et al. 2024. Power Aware Container Placement in Cloud Computing with Affinity and Cubic Power Model. *arXiv:2408.01176* [cs.DC]
- [21] Theodoros Theodoropoulos et al. 2023. GreenKube: Towards Greener Container Orchestration using Artificial Intelligence. In *2023 IEEE Int. Conf. on Service-Oriented System Engineering (SOSE)*.
- [22] Paul Townsend et al. 2019. Improving Data Center Efficiency Through Holistic Scheduling In Kubernetes. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. 156–15610.
- [23] Amjad Ullah et al. 2023. Orchestration in the Cloud-to-Things Compute Continuum: Taxonomy, Survey and Future Directions.
- [24] Jialin Yang et al. 2025. A Survey on Task Scheduling in Carbon-Aware Container Orchestration. *arXiv:2508.05949* [cs.SE]