Breaking Barriers: From Black Box to Intent-Driven, User-Friendly Power Management

Thijs Metsch[1][a] and Adrian Hoban[2][b]
[1]*Intel Labs, Intel Corporation, Santa Clara, CA, U.S.A.*
[2]*Network and Edge Group, Intel Corporation, Santa Clara, CA, U.S.A.*

Keywords: Edge Computing, Cloud Computing, Power Management, Energy Efficiency, Intent-Driven-Systems.

Abstract: Power management in cloud and edge computing platforms is challenging due to the need for domain-specific knowledge to configure optimal settings. Additionally, the interfaces between application owners and resource providers often lack user-friendliness, leaving efficiency potentials unrealized. This abstraction also hinders the adoption of efficient power management practices, as users often deploy applications without optimization considerations. Efficient energy management works best when user intentions are clearly specified. Without this clarity, applications are treated as black boxes, complicating the process of setting appropriate throttling limits. This paper presents an application intent-driven orchestration model that simplifies power management by allowing users to specify their objectives. Based on these intentions, we have extended Kubernetes to autonomously configure system settings and activate power management features, enhancing ease of use. Our model demonstrates the potential to reduce power consumption in a server fleet within a range of $\approx$ 5-55% for a sample AI application. When applied broadly, the research offers promising potential to address both economic and environmental challenges. By adopting this model, applications can be more efficiently orchestrated, utilizing advanced resource management techniques to mitigate the power usage surge that is in part driven by applications such as AI and ML.

1 INTRODUCTION

Managing power efficiently in cloud and edge computing is increasingly challenging due to the growing power demands of applications such as AI and ML based applications. As the need for environmentally friendly deployments grows, understanding the roles and motivations of different users/stakeholders – application owners and resource providers – becomes crucial. Current power management features often require domain-specific knowledge and platform-level configurations, which are either not easily accessible or deliberately restricted (e.g. for security reasons) to application owners. This results in sub-optimal deployments and inefficiencies in platform management and leave untapped savings for the resource providers. Overall, this hinders the adoption of the green edge-cloud continuum.

Moreover, hardware features often automatically boost to higher performance modes when detecting activity, leading to increased power draw. Without adequate context, these boosts can be inefficient and unnecessary. By providing context through application intents, power management can become more precise and efficient, aligning performance boosts with actual needs. Additionally, system-wide power configurations generally result in higher power draw compared to fine-tuned, per-application configurations. Resource providers should only flexibly perform per-application tuning, and space and time shift them if they understand the application intents. This aligns with the concept of "*tell me what you want, not how to do it*", demonstrating how resource providers benefit more from understanding user intentions rather than following a strict set of instructions.

This paper uses an Intent-Driven Orchestration (IDO) model as a novel approach to address these challenges. By allowing application owners to specify their intents through a set of objectives instead of low-level resource requirements, intent-driven systems bridge the gap between user-friendly interfaces and complex power management configurations. The obtained experimental data indicates an enhanced ef-

[a] https://orcid.org/0000-0003-3495-3646
[b] https://orcid.org/0009-0001-4970-889X

Metsch, T. and Hoban, A.
Breaking Barriers: From Black Box to Intent-Driven, User-Friendly Power Management.
DOI: 10.5220/0013150000003953
In *Proceedings of the 14th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS 2025)*, pages 24-35
ISBN: 978-989-758-751-1; ISSN: 2184-4968

ficiency in the computing continuum.

Intent-driven systems ensure that applications are not treated as black boxes but are managed based on intentions, enabling more effective power optimization. Key benefits include:

- **Semantic Application Portability** - Declaring intents with targeted application objectives remain consistent across platforms, unlike declaring specific resource requests, which can result in significantly different performance and power characteristics depending on the platform context.
- **Invariance** - Intents allow resource providers to dynamically modify system resource allocations in response to changing contexts, such as available energy, optimizing overall system setup over time as long as the intents can be met.
- **Context** - Intents provide context for resource providers to maximize performance and energy efficiency by understanding the application owners' objectives.

Throughout this paper, the implementation and benefits of IDO models for various applications are demonstrated. By leveraging Intel®'s IDO extension for Kubernetes[1], we enable autonomous configuration of systems based on user-specified intents, simplifying power management and enhancing ease of use.

The rest of the paper is structured as follows: Section 2 introduces efficient power management for feature-aware advanced users. Section 3 showcases the benefits of an IDO model. Section 4 discusses the results of providing intents to a orchestration stack, followed by related work in Section 5 and conclusions in Section 6.

2 PLATFORM LEVEL POWER MANAGEMENT

Intel® Xeon® processors have an extensive set of power management capabilities that were created to automatically, or under user and systems administrators preference, adjust the compute performance to power consumption ratio dynamically in response to the needs of the applications or the resource provider.

2.1 Core Sleep/Idle States (c-State)

The Advanced Configuration and Power Interface (ACPI) specification[2] defines the processor power state, known as its C-state. The C-state is sometimes colloquially known as the processor "idle" state on a per-core basis or on a CPU package basis. Core and package C-states coordination is managed by the CPU Power Control Unit. C-state values range from C0 to Cn, where n is dependent on the specific processor. When the core is active and executing instructions, it is in the C0 state. Higher C-states indicate how deep the CPU idle state is.

Higher C-states consume less energy when resident in that state but require longer latency times to transition into the active C0 state. The BIOS/UEFI configuration can be configured to restrict how deeply the cores and package can idle, e.g., it is possible to restrict access to the deepest C-state.

In Linux, C-state management is implemented on modern Intel® Xeon® processors with the *intel_idle* driver that is part of the CPU idle time management subsystem. Linux categorizes a CPU core as being idle if there are no tasks to run on it except for the "idle" task. Note that C-state information is introduced here for completeness and to draw a distinction with P-State, however the experiments outlined in this paper did not leverage C-state capabilities.

2.2 Core Frequency/Voltage State (P-State)

Intel® Xeon® processors include the ability to alter the processor operating frequency and voltage between high and low levels. The frequency and voltage pairings are defined in the ACPI specifications as the processor Performance State (P-state). P-states are SKU-specific settings ranging from the low-end of the operating frequency and voltage pairings with minimums defined by Pn to the nominal operating condition Base Frequency (P1), to the maximum single core turbo (P01).

Intel® Xeon® processors with HW P-state Management (HWP) can manage P-state transitions automatically with some tuning configuration provided to it by the operating system. With the Linux operating system the CPU Performance Scaling Subsystem (*CPUFreq*)[3] is responsible for providing the OS-level inputs to the processor P-State management capability. The *CPUFreq* layer is composed of core foundational capabilities, and scaling governors that contain the algorithms for selecting P-States and scaling drivers for communicating the desired P-States to the processor. Users/administrators can configure the CPU scaling with the *sysfs* filesystem and using utili-

[1] https://github.com/intel/intent-driven-orchestration/

[2] https://uefi.org/htmlspecs/ACPI_Spec_6_4_html/

[3] https://www.kernel.org/doc/html/v6.7/admin-guide/pm/cpufreq.html

ties such as the CPUFreq Governor[4]. Note, the Intel® Xeon® processors' internal logic will be the final adjudicator of what P-State the processor should be in.

The *intel_pstate* driver[5] is the CPU scaling driver part of the *CPUFreq* subsystem for Intel® Xeon® processors. When configured in the default *active* mode with HWP enabled there are effectively two governor modes exposed:

- A *performance* governor mode that promotes maximum performance.
- A *powersave* governor mode that promotes maximum power savings with a reduction in peak performance.

Intel® Xeon® processors can take as an input to their P-state management an Energy Performance Preference (EPP) setting. When in *performance* mode, EPP is set to its maximum performance setting (0). In *powersave* model, the EPP setting determines how aggressively the CPU pursues power saving configurations. Furthermore, uncore frequency scaling is a power management mechanism in modern CPUs that adjusts the frequency of the uncore domain, which includes components like the memory controller, last-level cache, and interconnects. Unlike P-states and C-states, which operate on a per-core basis, uncore frequency scaling independently regulates the performance of shared subsystems.

2.3 Power Manager for Kubernetes

The Intel® Power Manager for Kubernetes software[6] is a Kubernetes Operator that has been developed to provide Kubernetes cluster users with a mechanism to dynamically request adjustment of worker node power management settings applied to cores allocated to the Pods running the applications. The power management-related settings can be applied to individual cores or to groups of cores, and each may have different policies applied. It is not required that every core in the system be explicitly managed by this Kubernetes power manager. When the Power Manager is used to specify core power related policies, it overrides the default settings. The container deployment model in scope is for containers running on bare metal (i.e., host OS) environments.

The Power Manager for Kubernetes software has two main components, the Configuration Controller and the Power Node Agent, which in turn has a dependency on the Intel® Power Optimization Library[7] which in turn configures the *intel_pstate* driver.

The Configuration Controller deploys, sets, and maintains the configuration of the Power Node Agent. By default, it applies four cluster-administrator or user-modifiable PowerProfile settings to the Power Node Agent. This enables Pods to select between a performance, balance-performance, balance-power profile, and a default profile, to be configured on cores allocated to the Pod. Note, in the experiments in this paper, Profiles A-D were defined as per 1.

The Power Manager for Kubernetes organizes CPU cores into logical "pools", including a shared pool, which represents the Kubernetes shared CPU pool, and a default pool, a subset of system-reserved cores within the shared pool. Cores in the default pool are excluded from power setting configurations, while those in the shared pool are assigned a power-saving profile.

The Power Manager for Kubernetes also supports the notion of an "exclusive" pool. This pool is used to group cores allocated to Guaranteed QoS class of Kubernetes Pods. When cores are allocated to Guaranteed Pods, they are moved from the Power Manager Shared Pool into the Power Manager Exclusive Pool. This mechanism supports the model where cores that are not expected to be pinned to applications can be configured to run in a low power model.

Note, per-core power management works in association with CPU pinning deployment semantics. The Kubernetes Resource Orchestration for 4th Gen Intel® Xeon® Scalable Processors[8] technology guide provides a more detailed discussion on this and other resource allocation considerations.

2.4 Observability

One of the fundamental premises of improving on a current state is the ability to measure it. This is particularly true for activities aiming to support more sustainable computing at the edge & cloud.

The observability in focus in the context of this paper is on telemetry aspects related to power consumption at the full system (node) level as well as at the application level. There are several tools and techniques to measure power at both of these levels. This paper does not aim to identify what might be the best approach, merely use some of the tools and approaches consistently throughout the experiments and

[4]https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

[5]https://www.kernel.org/doc/html/v5.15/admin-guide/pm/intel_pstate.html

[6]https://github.com/intel/kubernetes-power-manager

[7]https://github.com/intel/power-optimization-library

[8]https://networkbuilders.intel.com/solutionslibrary/kubernetes-resource-orchestration-for-4th-gen-intel-xeon-scalable-processors-technology-guide

focus on the differential impact that the proposed approach has on power consumed in the environment.

For infrastructure, processor-specific and in-depth data collection, tools such as telegraf[9] are available. Telegraf has a plugin model that supports data collection from many sources, including from Powerstat[10]. Telegraf can subsequently be used to distribute telemetry data from the node to data sinks.

Prometheus[11] is a Time Series Database with systems monitoring and alerting capabilities. Prometheus can be configured to pull data from telemetry collection tools such as Telegraf via a Telegraf Prometheus exporter plugin.

Grafana[12] is a monitoring and data visualization tool. When the above tools are combined with Grafana, a pipeline can be established to assess and visualize the node-level power consumption.

In addition to node-level power consumption, understanding the per-application power consumption can add to the understanding of how effective power efficient strategies are being from the application perspective. Scaphandre[13] is one tool that is focused on per-app energy consumption metrics.

3 INTENT-DRIVEN ORCHESTRATION

To switch to intent-driven systems, Intel®'s IDO extension for the Kubernetes control plane is key, enabling it to understand and process application-level intents through Custom Resource Definitions (CRDs) as discussed in (Metsch et al., 2023). By enabling Kubernetes CRDs, the intents can be defined as objects through the Kubernetes API. This allows application owners to specify their intents through a set of objectives rather than low-level resource requirements, simplifying the interface and enhancing user-friendliness.

By allowing users to express their *Intent* with a set of objectives and associated Key Performance Indicators (KPIs), the IDO extensions automate the translation of these intents into actionable configurations, facilitating more efficient power management. The *KPIProfile* enable users to define how the orchestration stack can observe their application's behavior. Fig. 1 shows the object model that details how intents, objectives and KPIs relate. Objectives in the

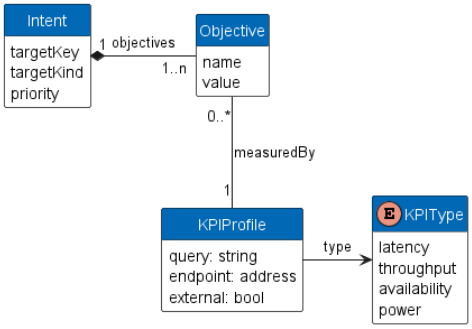


Figure 1: Object model detailing relationship between intents, objectives and KPIs.

IDO model are unit-less objects categorized by a type such as latency, throughput, availability, and power, enabling their specification across a wide range of applications, such as - but not limited to - AI applications, video processing, micro-services, web services, and High-Throughput Computing (HTC).

The Kubernetes extension integrates a planning component within Kubernetes, this planner is responsible for continuously determining the necessary configurations and setup needed to meet the specified intents. It hence determine *what & how* to optimally manage the overall system. It works in tandem with the scheduler, which manages *when & where* to place applications. By leveraging the insights provided by the planner, the scheduler can make more informed decisions, optimizing cluster level resource allocation and workload distribution. This collaboration between the planner and scheduler ensures that the system setup aligns with applications owners and resource provider's objectives, trading of both performance and efficiency.

To make informed decisions, the planner utilizes an A* planning algorithm which leverages a set of heuristics and utility functions that evaluate different actions based on their potential impact on the specified KPIs. These utility functions allow the planner to incorporate the intentions of resource providers into its decision-making. The planner can support various actions, including vertical and horizontal scaling, and the configuration of platform features such as Intel® Resource Director Technology (Intel® RDT) [14] and Intel® Speed Select Technology (Intel® SST)[15]. This ensures that the system can be dynamically adjusted to meet a set of performance and power management goals leveraging platform features under the hood, catering for the needs of cloud-native applica-

[9]https://github.com/influxdata/telegraf
[10]https://manpages.org/powerstat/8
[11]https://prometheus.io/
[12]https://grafana.com/
[13]https://github.com/hubblo-org/scaphandre

[14]https://www.intel.com/content/www/us/en/architecture-and-technology/resource-director-technology.html
[15]https://www.intel.com/content/www/us/en/support/articles/000095518/processors.html

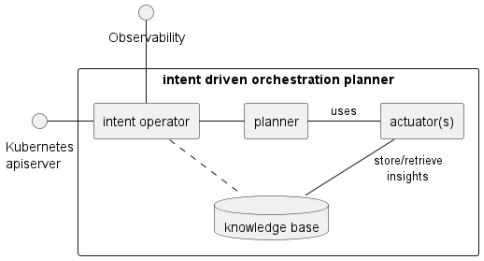


Figure 2: System overview of Kubernetes extensions enabling intent-driven orchestration.

tions. The A* planning algorithm may be replaced by other algorithms and is a subject for further study.

A key feature of the IDO model is its pluggable architecture, which employs plugins known as actuators as shown in Fig. 2. Each actuator is responsible for a specific function within the system and can use either static lookup tables or more complex AI/ML models to determine the best orchestration actions based on current conditions and user intents. This flexible architecture allows the system to adapt to varying workloads and objectives, continually optimizing for performance and power consumption. The actuators' ability to dynamically adjust configurations based on near-real-time data ensures that the system remains efficient and responsive to changing demands. This system operates in the orchestration time-domain, i.e. the nature of how quickly the control loop may be closed is a function of the orchestration software that underpins the system. Timeliness should be considered with seconds/minutes/hours granularity.

For addressing efficient power management, a new actuator has been developed. This actuator can adjust power settings based on power and performance objectives, optimizing power usage while balancing it with application performance. Additionally, it can be configured to prioritize power or carbon reduction intents, aligning with broader environmental goals. Next to this power management actuator a CPU rightsizing actuator has been used in tandem. The following sub-sections describe the key models used to enable the concepts introduced in this paper.

3.1 Dynamic CPU Rightsizing

Dynamic CPU allocation rightsizing is a key in energy management, aiming to dynamically adjust CPU resources allocated to applications. The primary goal is to optimize energy usage while trading performance by reducing the resources allocated at runtime. This approach contrasts with static allocation methods, which often lead to resource overprovisioning and consequently energy wastage. By tailoring CPU allocation to current application needs, dynamic rightsizing conserves energy while avoiding resource overuse. Maximizing the utilization of available resources and ensuring that those resources associated with an application are setup according to the intents.

To enable the dynamic resource allocations, a curve fitting model that predicts resource requirements-based performance metrics is used. The equation 1 showcases the model used for latency-related metrics. Whereby the goal is to learn the values of the parameters p_0, p_1, p_2 that describe the characteristics of the application in the current context. This model, applicable to a wide range of applications across various domains and allows the planner to anticipate CPU needs and adjust resources accordingly. Our approach leverages pre-training, on-the-fly learning, and proactive adaptation to continuously refine the model. Pre-training with historical data establishes a baseline understanding of application behaviors, while on-the-fly learning incorporates near-real-time performance data to ensure the model remains accurate as applications evolve.

$$latency = p_0 * e^{-p_1 * cpu_{units}} + p_2 \quad (1)$$

Despite the gains of these techniques, the training process is lightweight, ensuring that the energy savings from dynamic CPU rightsizing are not offset by the costs of model maintenance. By dynamically adjusting CPU resources based on predictive modeling and adaptive learning, our approach achieves energy savings while maintaining application performance as will be discussed in section 4.1.

3.2 Dynamic Power Management

Associating the most efficient power profile with an application is crucial for effective energy management. The power profile selection must consider both the application's intents and the current system status, including the available energy supply. This approach facilitates optimal energy use, addressing both economic and environmental concerns. For instance, during periods of high sustainably-sourced energy availability, a performance-oriented power profile might be appropriate, whereas during constrained sustainably-sourced energy availability, a power-saving profile can be adopted to conserve energy (Intel Network Builders, 2024).

To facilitate this dynamic power management, a *RandomForestRegressor* model as described in (Geurts et al., 2006) is used. This model is trained to learn the relationships between selecting different performance power profiles, allocated resources, as

well as the power consumption associated with each profile. By leveraging historical data coming from the observability stack, the model learns how various workload and system conditions impact power usage and performance. Based on this model the planner can make informed decisions about which power profile to apply, optimizing both performance and power efficiency. The use of a *RandomForestRegressor* is particularly advantageous due to its robustness and ability to provide insights.

The implementation currently relies on pre-trained models. These models are initially trained in a controlled development environment where experiments with the applications are carried out to determine the effect power profile selections have to gather comprehensive training data. Once the models have achieved satisfactory accuracy and reliability, they are deployed to the production environment. This transfer ensures that the models can operate in various scenarios with minimal adjustments, providing accurate and efficient power management. By selecting power profiles on a per-application basis rather than system-wide, the power management strategy can be tailored to the specific needs of each application. This granularity allows for further energy savings and efficiency gains, as each application can operate under the most suitable power conditions for its intents as will be discussed in section 4.2.

4 EXPERIMENTAL RESULTS

The experiments utilized a system featuring an Intel® Xeon® Processor. These processors are designed for cloud & edge deployments, although the methodologies outlined in this paper are also applicable to other deployment scenarios. The system ran on Ubuntu 22.04 LTS, with Kubernetes 1.27 deployed to manage the extensions described in Section 2. The configured power profiles are detailed in Table 1, chosen based on the CPU's characteristics and power curve behavior. Profiles A and B were selected for their power-saving capabilities, while profiles C and D were chosen for their performance attributes. To enable a power saving mentality all cores are put in Profile A by default. Notably the profiles were selected and configured with the CPUs characteristics in mind by the resources owner, relieving the application owners of the need to have (domain) specific knowledge about their setup & power ratings.

To demonstrate the methodologies, various applications were employed to validate the effectiveness of the features. The following applications were used:

Table 1: Power profiles configurations.

Name	Min	Max	EPP
profile A	800Mhz	1600Mhz	power
profile B	800Mhz	1800Mhz	balance_power
profile C	800Mhz	2400Mhz	balance_ performance
profile D	800Mhz	3500Mhz	performance

- **FaaS (Function-as-a-Service)** This application performs mathematical calculations triggered by HTTP events. Performance-related KPIs were instrumented using the Linkerd[16] service mesh, providing P99, P95, and P50 latencies. The key objective used for these experiments was the P99 latency.
- **OVMS (OpenVINO® Model Server)**[17] This application performs object detection on video frames, representing a typical edge use case aimed at conserving network bandwidth by processing data locally. The application was modified to expose the processing time per video frame as a histogram via a Prometheus client[18]. The key objective for these experiments was the P99 latency.
- **AI Mistral LLM** This application functions as an AI chatbot responding to incoming requests using a Mistral 7B LLM[19]. It uses a Prometheus client to expose the time required to generate a token. The key objective for these experiments was the average token creation time, as the first response token typically takes longer to generate, affecting higher percentile latencies.

The aim of the selected application was to mimic a scenario in which multiple applications from potentially different tenants are run on the node. In this environment it is the goal to optimize the power usage of each node, with the overarching goal to achieve cluster level power optimization. All values were normalized for comparison purposes, ranging from 0 to the maximum observed value of a data series. This normalization facilitates easier data interpretation and demonstrates that the methodology can be applied across a diverse set of use cases and scenarios.

4.1 Dynamic CPU Rightsizing

Fig. 3 illustrates the resulting models for the three applications when applying the methodology described in Section 3.1. While the *FaaS* application can oper-

[16]https://linkerd.io/

[17]https://docs.openvino.ai/2023.3/ovms_what_is_openvino_model_server.html

[18]https://prometheus.io/docs/instrumenting/clientlibs/

[19]https://mistral.ai/news/announcing-mistral-7b/

ate with fewer CPU cores, the *AI Mistral LLM* and *OVMS* applications require a minimum number of cores to function.

The resource allocation curves for each application vary due to their unique implementations and characteristics. However, the planner can utilize these models to efficiently allocate the appropriate number of resources for each application in a specific context.

The common characteristic from the charts in Fig. 3 is that from the perspective of the KPI being assessed, there is a point beyond which adding additional CPU cores provides limited improvements in the KPI, i.e. the curve levels off. This point varies by application but is dynamically determined by the learning algorithms in the IDO implementation. While the underlying curve-fitting model is relatively simple, it accommodates a wide range of applications. Notably, although these curves correspond to a specific power profile, their shapes remain largely consistent across profiles, differing primarily in amplitude.

4.2 Dynamic Power Management

Fig. 4 presents the resulting models after applying the methodologies described in Section 3.2. It highlights the impact of power profile selection on the performance of each application. Where 1.0 is shown in these graphs represents the worst performing latency. The best performing latency is closest to zero. The *Function* and *OVMS* applications require a certain amount of CPU resource to be allocated to achieve lower latency objectives. This is also true for the *AI Mistral LLM* application, however here the selection of the power profile plays a bigger role. To achieve lower latencies at least the profiles C or D need to be selected. This variation is due to the specific implementation and hardware utilization of each application, demonstrating that different applications have unique characteristics that can be efficiently learned. Notably, we found that AI applications are highly flexible and can be efficiently managed within the context of power and energy constraints.

4.3 Enhancing Usability

The following paragraphs describe how the models shown in section 4.1 and 4.2 can be used by the IDO planner as described in section 3. Overall, this demonstrates the ease of use of platform features for the application owners – as they only work with their intents and associated targeted objectives – while providing the resource providers with additional context to efficiently manage their compute platform.

Making Decision Based on Performance Related Objectives. Fig. 7 shows a screenshot of the dashboard for the IDO planner. The table shows the event timestamps, the current state of the application (defined by its objectives), the desired state, and the resulting plan. In this example, the *FaaS* application is under control of the planner, targeting a P99 objective. When the target objective shifts from 100ms to 200ms, the system efficiently frees up resources and selects a more efficient power profile.

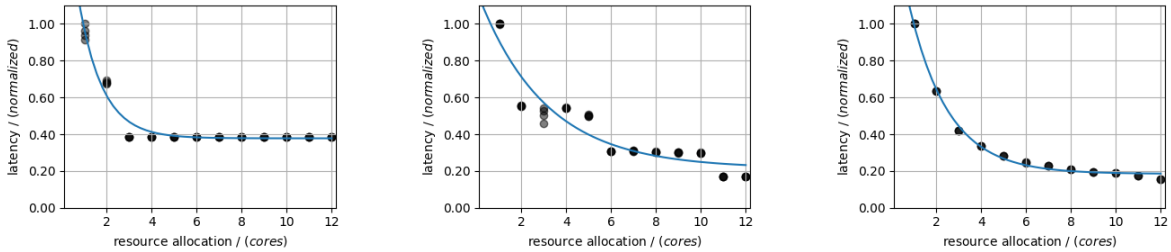
Making Decision Based on Power Related Objectives. Fig. 8 shows a screenshot of the dashboard for the IDO planner when applying power objectives. In this example, the system is purely driven by power objectives. When the target objective shifts from an average of 10W to 5W while running the application, the system selects a more power-efficient profile. This capability is a crucial step toward enabling carbon-based objectives in the future.

4.4 Experimental Gains

Power profile selection impacts not only the performance of individual applications but also the power consumption estimations as shown in the previous section 4.3. Fig. 5 demonstrate how the model described in section 3.2 can also be used predict the power draw for a given application based on its power profile and CPU resource allocation.

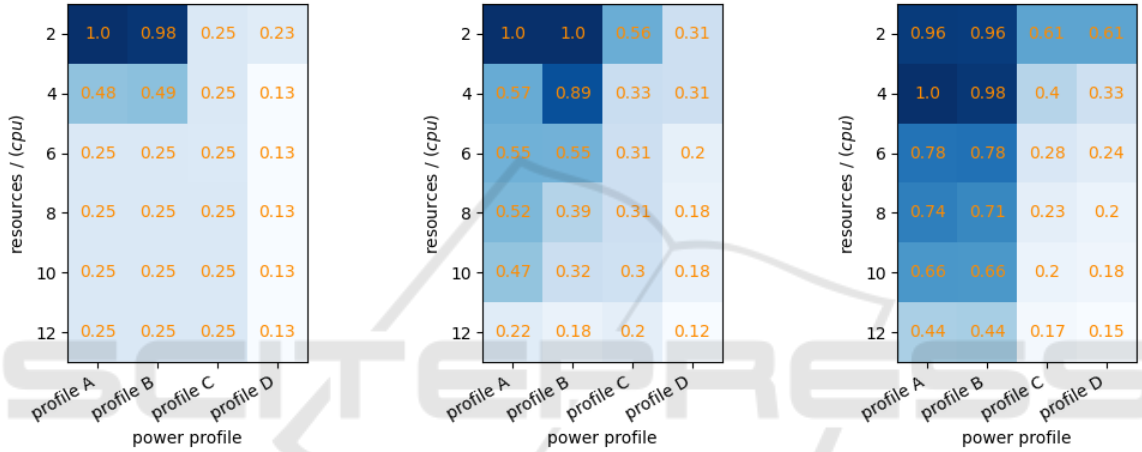
For applications that fully utilize compute resources, high power draw is observed when more cores are allocated, which are allowed to operate at higher frequencies. Next to the core frequencies that can be adjusted using the P-states and C-states, the uncore components (e.g., the memory controller, interconnects, etc.) can also be managed independently. Since uncore components are shared by all applications, their activity significantly impacts each application's overall power draw estimation.

Current power estimation models attribute power usage based primarily on utilization of the cores associated by the processes. However, the power draw of shared components can result in elevated power draw irrespective of which application is responsible, a factor not adequately captured in most power estimation tools. Consequently, further research is required to refine power estimation for improved accuracy. Also, more fine-grained per sub-component frequency scaling methodologies will enable further gains. For applications that fully utilize compute resources and are memory-bound—such as the *AI Mistral LLM* evaluated here (as can be confirmed by analyzing the Instructions per Cycle (IPC) and memory bandwidth us-



(a) Function as a Service App (b) OVMS App (c) AI Mistral LLM App

Figure 3: The CPU rightsizing models illustrate the impact of CPU resource allocations on latency-related objectives for various applications, including: a) P99 tail latency for a FaaS-style deployment, b) P99 compute latency for processing video frames, and c) average token creation time for an AI LLM model. The forecast based on the curve-fitting are shown in blue for the performance profile D.



(a) Function as a Service App (b) OVMS App (c) AI Mistral LLM App

Figure 4: The selection of the right performance profile has an effect on the performance of the application.: a) P99 tail latency for a FaaS-style deployment, b) P99 compute latency for processing video frames, and c) average token creation time for an AI LLM model. Note, a value of 1.0 is the normalized worst (highest) latency reading measured.

age nearing the CPUs' theoretical maximums) – the current power model is effective and allows for intent-driven orchestration using power objectives.

These findings align with those presented in (Pereira et al., 2017) and in general call for more efficient software. While the results are specific to the platform and applications used, an intent-driven model enhances semantic application portability and can learn and adapt to different contexts.

Finally, Fig. 6 shows the experimental gains for the *AI Mistral LLM* application in more detail. In this set of charts, the effects that two of the system-wide CPU frequency scaling governors have when applied to the processor for this application is contrasted to an per-application intent-driven model. The normalization point was chosen as that produced by the default system-wide lower power consuming configuration in the *powersave* governor. For these experiments, the node level utilization was observed to be $\approx$ 35% as multiple applications were deployed.

The effects of the two system-wide configurations are broadly aligned with their monikers. I.e., the *performance* CPU frequency governor delivers the best application KPI which is the lowest average token creation time while consuming the most node-level and application-level power. The *powersave* CPU frequency governor offered a considerably lower power draw by the node and the application, but with a worse (i.e. higher) average token creation time.

The system-wide configuration choice implied by selecting between the two native CPU frequency governors has a significant implication for the resource provider and the application owner. Purchasing a high-performance processor and configuring it to run with the *powersave* CPU frequency governor is shown in the experiment with the AI LLM application to considerably limit the KPI potential. Such a configuration could result in the application owner to require

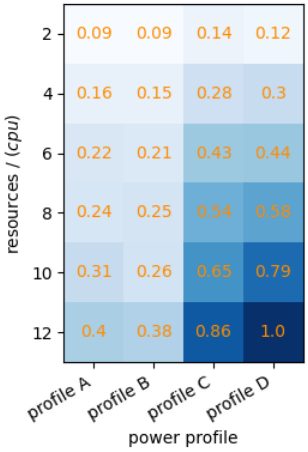


Figure 5: The selection of the right performance profile has an effect on the power estimation per-application - e.g. for an AI LLM model. Note, a value of 1.0 represents the normalized highest power consumption measured.

actions such as invoking earlier scale-out strategies to make up the performance short fall thereby undermining the intention of restricting the power draw of the node. Nonetheless it is a simple, effective mechanism to limit power consumption.

The behavior of the power profiles when applied to the application is broadly aligned with expectations. Profile A offered the worst KPI performance (highest latency) and best power consumption (lowest power draw), with Profile D the opposite. The following points are particularly interesting:

- At one end of the scale a considerably better ($\approx$ 55%) overall power consumption (lowest power draw) could be achieved with Profile A when compared to the system wide *powersave* CPU frequency governor.
- At the other end of the scale an equivalent application KPI, the best average token creation time (lowest time) was achieved with Profile D as with using the *performance* governor while also having a slightly lower ($\approx$ 5%) power draw.
- Profile C offers a $\approx$ 30% performance gain and Profile D a gain of $\approx$ 40% compared to a system-wide *powersave*, while both outperforming the latter. Both profiles offer a better power efficiency, with $\approx$ 5% and $\approx$ 15% power savings respectively. Notably the difference between their performance characteristic is just $\approx$ 5% while the power savings are within the range of $\approx$ 10%, offering options for trading-off efficiency gains.
- Through an IDO model – leveraging an per-application power management capability – similar performance can be achieved, while this model enable showcase power efficiency gains contrasting it to system-wide configuration options across the board.

Similar to the results observed with the *AI Mistral LLM* application, both the *Function* and *OVMS* applications demonstrate potential power savings of up to $\approx$ 50%. This level of efficiency is largely attributed to their smaller CPU footprint, which allows most of the system to remain in a power-saving mode.

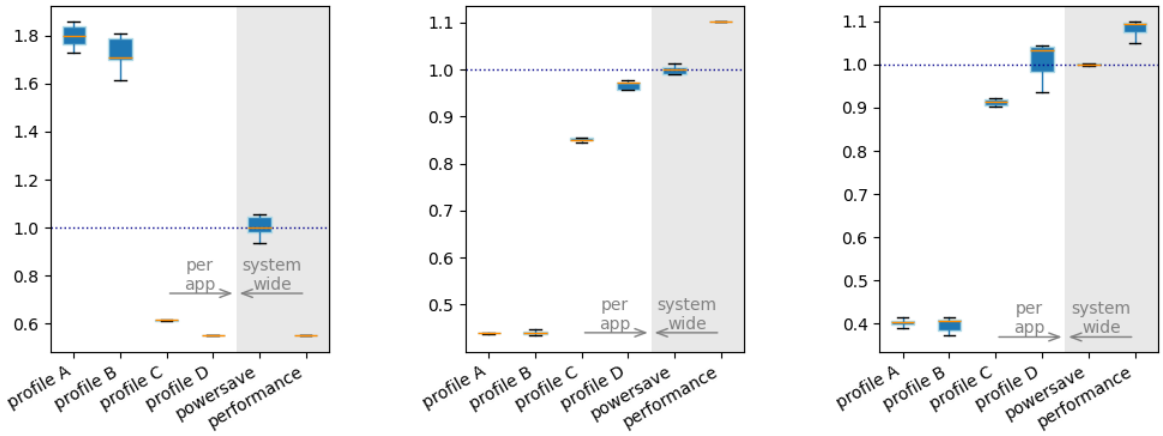
This observation underscores the importance of consolidating applications with compatible power profiles (based on their intents) on servers to optimize power efficiency for resource providers. It is important to note, however, that these results may vary depending on the specific system configuration and applications utilized.

The experiment shows that IDO offers an alternative approach that absolves a system administrator from having to choose between two system-wide settings that either prioritizes power savings while limiting performance or drives performance while forcing a high-power consumption configuration. The dynamic nature by which IDO selects different profiles for the application allows for application intent to drive the minimum power consumed based on respecting a KPI. When the application has a lower KPI requirement or is simply not busy enough to cause it to miss a KPI, IDO can be used to ensure the minimal power needed to meet the KPI is consumed.

5 RELATED WORK

The European Telecommunications Standards Institute (ETSI) has highlighted the necessity for declarative management in evolving telecommunications networks, emphasizing the definition of desired outcomes without specifying declarative resource asks. This approach simplifies network operations, reduces operational costs, and accelerates service deployment, making intent-driven applications crucial for modern management (Cai et al., 2023). This aligns with the goals of the IDO model as presented in this paper.

The increasing demand for compute capacity, driven in part by the rapid growth of AI/ML applications, necessitates rethinking of power and energy management strategies (Lin et al., 2024). The growth in use of these applications may place significant stress on power grids, emphasizing the need for user-friendly energy management solutions. The methodology proposed in this paper is a pivotal piece of this puzzle, enabling users to embrace hardware abstraction while seamlessly utilize platform-level power management features.



(a) Average token creation time (b) Node level power draw (c) Application power estimation

Figure 6: Effect of intent-driven power management on an AI LLM application.

Power capping and frequency management have proven effective in controlling the power draw of AI-intensive applications (Patel et al., 2024). The utilization of platform features has generally been demonstrated to enhance performance and power efficiency across a wide range of applications (Veitch et al., 2024). This paper builds on that foundation by simplifying the adoption of these features. The heterogeneous nature of edge locations adds complexity to managing the cloud-edge continuum, calling for more autonomous management and abstraction. Research has shown that abstracting the complexity of edge infrastructure and using service-level objectives to define service requirements yields significant benefits (Guim et al., 2022). These findings form the foundation for the follow-up work presented here.

Efficient orchestration and resource management has been tackled from multiple angles (Metsch et al., 2015). Previous work has addressed individual problems such as auto-scaling (Roy et al., 2011), placement (Bobroff et al., 2007), and overbooking to maximize utilization (Tomás and Tordsson, 2013). However, these solutions focus primarily on efficient resource usage rather than ease of use for energy efficiency. The work presented is complementary, potentially enhancing the methodology proposed here.

Coordination between compute demand in data centers and (local) power grids capacities has shown to be effective in improving carbon emissions (Lin and Chien, 2023). These techniques require resource providers to plan their capacity ahead and communicate this to the power grid. Efficient planning is crucial, necessitating an understanding of the intentions and priorities of application owners. The intent-driven approach discussed in this paper provides the necessary context to resource providers.

6 CONCLUSIONS

This paper presents an IDO model that simplifies power management for application owners while providing resource providers with essential context for efficient energy and resource optimization. By enabling the configuration of power profiles on a per-application basis based on user intents, this approach offers significant benefits towards greener solutions. Making power-saving modes the default in data centers and at the edge is an essential step forward. Contextual information provided by the intents is critical, as hardware cannot effectively throttle and optimize without it, leaving significant efficiency gains untapped. By accepting minor performance trade-offs can result in substantial power savings.

Our experimental results demonstrate that when intents and their objectives allow for it, power savings of $\approx$ 5-55% for this sample AI application can be achieved compared to baseline settings. We expect these gains to only increase given higher core count systems entering the market. The IDO model facilitates allocating and configuring the necessary resources for an application to meet the intents while maintaining greater power efficiency compared to configuring the system as a whole.

The intent-driven model benefits application owners by simplifying the use of power management features, reducing the need for deep domain-specific knowledge. This approach bridges the gap between application owners' needs and what resource providers can offer, as platform features are often not exposed to application owners. Simultaneously, it benefits resource providers by enabling more efficient resource optimization as the intents provide context, resulting in better overall system performance and en-

ergy savings.

Future work is planned to address per-application power estimation, as current solutions lack the desired accuracy. Additionally, we aim to explore carbon objectives built on top of power objectives. One limitation of the current solution is the need for core pinning, which could be replaced with a more flexible model in the future. Better-informed resource allocation can help the scheduler make more efficient decisions based on inputs from the planner, enhancing fleet-wide performance and energy efficiency. For example, consolidating workloads with similar power profile needs can be more efficient, avoiding single application's high-performance demands increase the overall power draw of a node.

By integrating these advancements, the IDO model provides a step towards managing the increasing compute demands driven by AI and ML applications, ensuring that power and energy management strategies remain effective and user-friendly.

ACKNOWLEDGEMENTS

This paper was in-part supported by European Union's Horizon Europe research and innovation programme under grant agreement No. 101135576. The authors would like to sincerely thank Francesc Guim, Denisio Togashi, Karol Weber and Marlow Weston for their contributions to the IDO capability and the methodologies discussed in this paper.

REFERENCES

Bobroff, N., Kochut, A., and Beaty, K. (2007). Dynamic placement of virtual machines for managing sla violations. In *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pages 119–128.

Cai, X., Deng, H., Deng, L., Elsawaf, A., Gao, S., Martin de Nicolas, A., Nakajima, Y., Pieczerak, J., Triay, J., Wang, X., Xie, B., and Zafar, H. (2023). Evolving nfv towards the next decade. https://www.etsi.org/images/files/ETSIWhitePapers/ETSI-WP-54-Evolving_NFV_towards_the_next_decade.pdf. [Accessed 10-07-2024].

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Mach. Learn.*, 63(1):3–42.

Guim, F., Metsch, T., Moustafa, H., Verrall, T., Carrera, D., Cadenelli, N., Chen, J., Doria, D., Ghadie, C., and Prats, R. G. (2022). Autonomous lifecycle management for resource-efficient workload orchestration for green edge computing. *IEEE Transactions on Green Communications and Networking*, 6(1):571–582.

Intel Network Builders (2024). Elastic and energy proportional edge computing infrastructure solution brief. *Intel*.

Lin, L. and Chien, A. A. (2023). Adapting datacenter capacity for greener datacenters and grid. In *Proceedings of the 14th ACM International Conference on Future Energy Systems*, e-Energy '23, page 200–213, New York, NY, USA. Association for Computing Machinery.

Lin, L., Wijayawardana, R., Rao, V., Nguyen, H., GNIBGA, E. W., and Chien, A. A. (2024). Exploding ai power use: an opportunity to rethink grid planning and management. In *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems*, e-Energy '24, page 434–441, New York, NY, USA. Association for Computing Machinery.

Metsch, T., Ibidunmoye, O., Bayon-Molino, V., Butler, J., Hernández-Rodriguez, F., and Elmroth, E. (2015). Apex lake: A framework for enabling smart orchestration. In *Proceedings of the Industrial Track of the 16th International Middleware Conference*, Middleware Industry '15, pages 1–7, New York, NY, USA. Association for Computing Machinery.

Metsch, T., Viktorsson, M., Hoban, A., Vitali, M., Iyer, R., and Elmroth, E. (2023). Intent-driven orchestration: Enforcing service level objectives for cloud native deployments. *SN Computer Science*, 4(3).

Patel, P., Choukse, E., Zhang, C., Goiri, I. n., Warrier, B., Mahalingam, N., and Bianchini, R. (2024). Characterizing power management opportunities for llms in the cloud. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, page 207–222, New York, NY, USA. Association for Computing Machinery.

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J. a. P., and Saraiva, J. a. (2017). Energy efficiency across programming languages: how do energy, time, and memory relate? In *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering*, SLE 2017, page 256–267, New York, NY, USA. Association for Computing Machinery.

Roy, N., Dubey, A., and Gokhale, A. (2011). Efficient autoscaling in the cloud using predictive models for workload forecasting. In *2011 IEEE 4th International Conference on Cloud Computing*, pages 500–507.

Tomás, L. and Tordsson, J. (2013). Improving cloud infrastructure utilization through overbooking. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, CAC '13, pages 1–10, New York, NY, USA. Association for Computing Machinery.

Veitch, P., MacNamara, C., and Browne, J. J. (2024). Uncore frequency tuning for energy efficiency in mixed priority cloud native edge. In *2024 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, pages 925–930.

APPENDIX

The Figs. 7 and 8 present screenshots of the dashboard for the IDO planner. The tables in the screenshots display the timestamps, the current state and desired state (as defined by their objectives), and the actions taken. As time progresses, the outcomes of the decisions are reflected in the upper rows of the table.

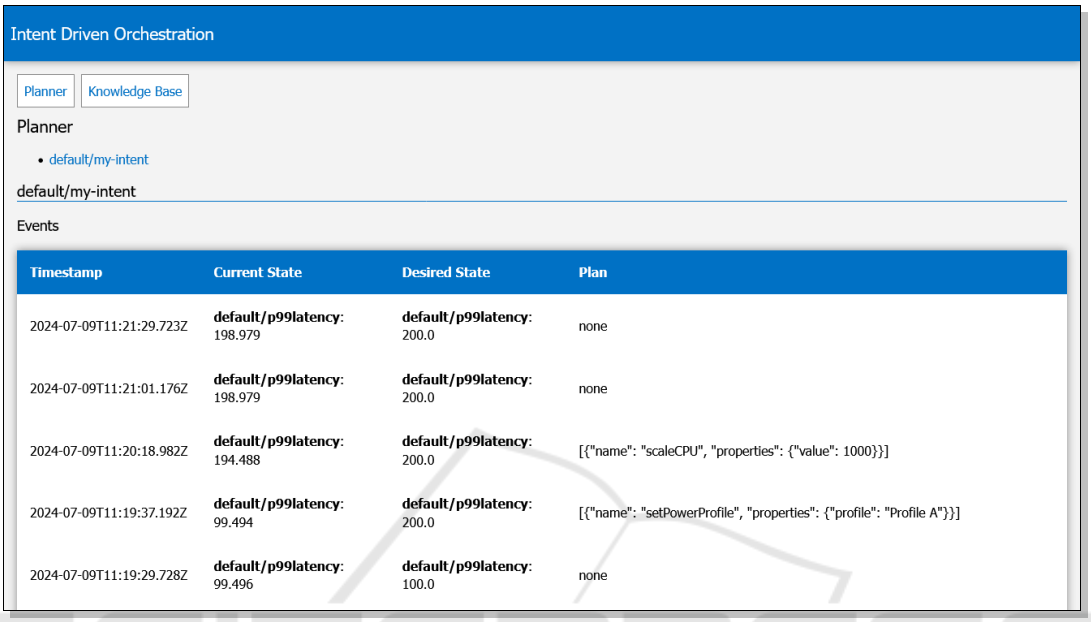


Figure 7: Dashboard showcasing the intent-driven orchestration planner's decisions based on performance related objectives.

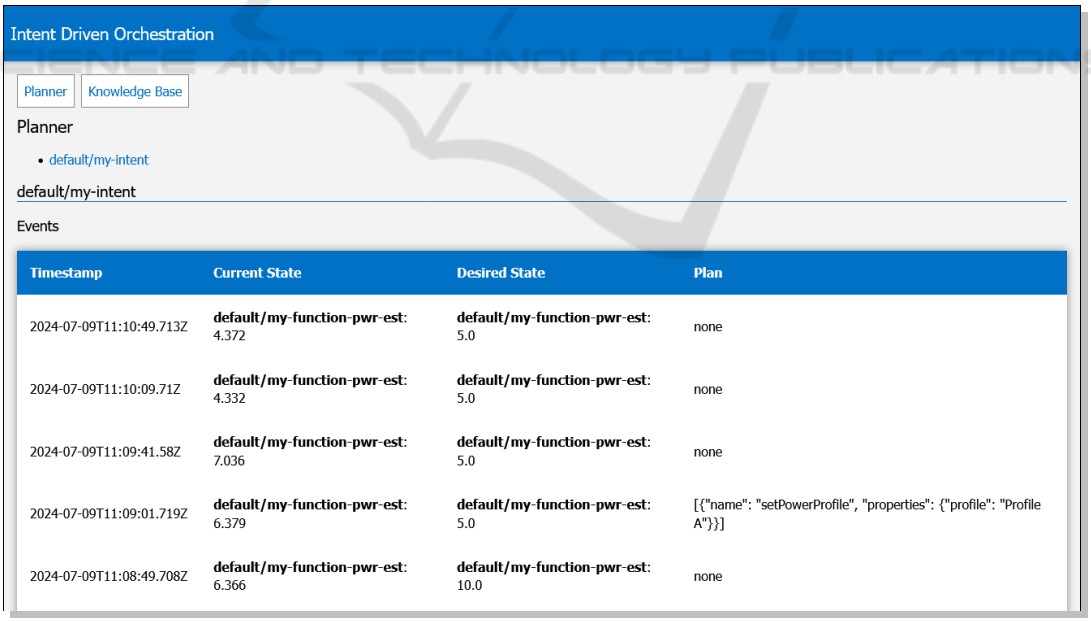


Figure 8: Dashboard showcasing the intent-driven orchestration planner's decisions based on power related objectives.