Energy-Aware Agentic Orchestration for Multi-Site Service Deployment

Rodrigo Moreira[1], Alisson Chaves[2, 3], Larissa F. Rodrigues Moreira[1], Tereza Carvalho[4],
Tiago Barros[3], Daniel Corujo[3], Rui Aguiar[3], David Gomez-Barquero[5], Flávio de Oliveira Silva[6]
[1]Federal University of Viçosa (UFV), Minas Gerais, Brazil
[2]Federal University of Uberlândia (UFU), Minas Gerais, Brazil
[3]Instituto de Telecomunicações and Universidade de Aveiro (UA), Aveiro, Portugal
[4]University of São Paulo (USP), São Paulo, Brazil
[5]Universitat Politecnica de València (UPV) Valencia, Spain
[6]University of Minho (UMinho), Braga, Portugal
Emails: {rodrigo, larissa.f.rodrigues}@ufv.br, alisson@ufu.br, terezacarvalho@usp.br,
{aoc.alisson, tiagobarros, dcorujo, ruilaa}@av.it.pt, dagobar@iteam.upv.es, flavio@di.uminho.pt

***Abstract*—Energy-aware orchestration across geographically distributed infrastructures is increasingly important as data centers absorb growing Artificial Intelligence (AI) workloads. However, existing approaches often exclude the energy cost of automated decision-making from system-level sustainability assessments. This paper presents an energy-aware agentic orchestration framework that treats Large Language Model (LLM) reasoning as a measurable component of the control loop. We compare single-shot tool calling, ReAct, and plan-and-execute strategies on a multi-site Kubernetes testbed spanning two European data centers and instrumented at layers ranging from power distribution units to per-pod energy attribution. Using ten natural-language orchestration intents with varying decision complexity, we show that reasoning strategy and intent complexity strongly influence graphics processing unit (GPU) energy consumption during orchestration decisions. ReAct and plan-and-execute generated executable tool calls across all evaluated intents but required multiple LLM inferences, whereas the single-shot configuration was less costly but non-actionable. Prefix caching reduced mean ReAct GPU energy consumption by 29.4%. For the evaluated lightweight workload, the infrastructure-energy changes observed within the one-second measurement window were insufficient to offset the measured GPU decision energy. These findings demonstrate the importance of accounting for reasoning overhead when assessing the sustainability of AI-native orchestration and motivate energy-budgeted selection of agent strategies.**

***Keywords*—Green networking, Energy-aware orchestration, Agentic AI, Large language models, Kubernetes, GPU energy**

I. Introduction

Digital services have become the connective tissue of contemporary economies, sustaining domains as diverse as public administration, industrial automation, and consumer entertainment. As these services become more interactive and data-intensive, providers increasingly distribute their workloads across multiple geographic sites, making multi-site container orchestration the operational substrate of modern cloud and edge platforms. According to the International Energy Agency (IEA), data centers already account for close to 1.5% of global electricity demand and are projected to more than double their consumption by 2030, a trajectory amplified by the rapid uptake of accelerator-bound artificial intelligence workloads. Deciding when, where, and how to deploy containerized services across geographically distributed infrastructures without compromising sustainability has thus emerged as a central, unresolved operational challenge [1], [2], [3], [4].

Two decades of work in cluster scheduling, autoscaling, and energy-aware placement have produced rich heuristic, optimization, and learning-based controllers, and modern Kubernetes ecosystems already expose layered telemetry from power distribution units down to per-pod attribution [5], [6], [7]. More recently, the operations community has begun to investigate Large Language Model (LLM) agents as a new orchestration interface, translating natural-language intents into sequences of monitoring and actuation calls through patterns such as single-shot tool calling, ReAct, and plan-and-execute. The promise is substantial: operators could request the most energy-efficient site for a new workload in plain language, get causal explanations for sudden power spikes, or trigger graceful migrations during congestion events, thereby indirectly benefiting end users with greener, more reliable services.

However, existing studies evaluate these agents primarily by task accuracy and operational effectiveness, while generally treating the energy consumed by the decision-making process as negligible relative to the managed infrastructure. This assumption may not hold when iterative reasoning executes repeatedly on accelerator-based platforms. In energy-aware orchestration, excluding this control-plane overhead can overestimate the sustainability benefits attributed to an automated decision. Determining whether, and under which conditions, the operational value of LLM-driven orchestration justifies its additional energy cost therefore remains an important open question [8], [9].

To address this gap, this paper proposes an energy-aware

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Accepted author manuscript for publication in the Proceedings of the 2026 IEEE Future Networks World Forum (FNWF 2026), Lisbon, Portugal, October 26–28, 2026.

agentic framework for orchestrating containerized services across geographically distributed Kubernetes clusters, in which the energy consumed by the agent itself is treated as a first-class, measurable quantity alongside the energy consumed by the infrastructure it influences. The framework couples a reasoning agent with a typed tool interface that exposes fine-grained Power Distribution Unit (PDU), Running Average Power Limit (RAPL), and per-pod telemetry as well as life-cycle actuation across sites, and instantiates the same pipeline under three reasoning strategies evaluated on a benchmark of ten natural-language intents stratified by decision complexity. We hypothesize that the sustainability of agentic orchestration depends on the interaction between intent complexity, reasoning strategy, and workload magnitude: more adaptive strategies improve actionability at the cost of additional inference energy, while lightweight workloads may fail to amortize this reasoning cost.

Our key contributions are: (i) an energy-aware agentic orchestration framework incorporating LLM decision energy into multi-site infrastructure assessment; (ii) the Net Energy Benefit (NEB) metric comparing short-horizon infrastructure savings against accelerator decision energy; (iii) a real two-site Kubernetes testbed instrumented from PDUs down to per-pod attribution and GPU accelerators; and (iv) an empirical evaluation quantifying how intent complexity, reasoning strategy, and prefix caching impact decision cost, latency, and actionability.

Sections II, III, IV, and V present the related work, method, testbed, and results, respectively; Section VI concludes the paper.

II. Related Work

This section reviews energy-aware orchestration approaches across edge, cloud, wireless, and Artificial Intelligence (AI)-serving systems, highlighting how prior work has addressed energy efficiency, and situates our agentic approach within this landscape.

Prior work addresses energy-aware orchestration through reinforcement learning, optimization, forecasting, and heuristic scheduling across MEC dispatch, DNN placement, GPU power prediction, sustainable AI workload management, wireless resource coordination, and carbon-aware cloud orchestration [10], [11], [12], [13], [14], [15]. Related efforts consider edge-AI pruning, green-cloud optimization, multi-tenant DNN inference, and sustainable LLM serving across geographically distributed data centers [16], [17], [18], [19].

Table I compares these approaches across key operational dimensions. Existing methods cover subsets of these metrics but generally treat the controller's decision-making cost as external to the system budget. Our method combines telemetry-informed, tool-enabled multi-site orchestration with explicit measurement of agentic decision overhead.

III. Proposed Method

Figure 1 summarizes the proposed energy-aware agentic framework, which maps natural-language intents to orchestration decisions using current site-energy telemetry.

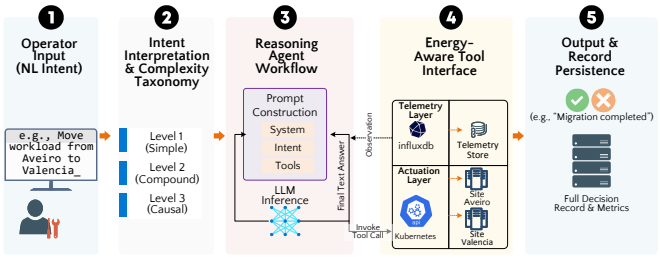


Fig. 1: End-to-end pipeline of the proposed method.

The method comprises four coupled components. The first is an intent interpretation layer that receives the operator's natural-language request and routes it to a reasoning agent. The second is a telemetry access layer, which exposes fine-grained energy measurements of the infrastructure as callable tools. The third is an actuation layer that exposes lifecycle operations on target deployments as callable tools. The fourth is the reasoning agent itself, whose strategy determines how it orchestrates these tools to reach a decision. We describe each component and then explain how they combine into the end-to-end workflow shown in Figure 1.

Each intent is a free-form request that refers to one or more sites and expresses a goal related to energy, performance, or placement. Requests range from simple observations, such as querying a site's current power consumption, to complex optimization goals involving multiple sites and constraints. We categorize intents by complexity level based on the number and nature of operations required to satisfy them. Level 1 intents are satisfiable by a single query or a single actuation. Level 2 intents require a sequence of reads and writes with an explicit dependency between them. Level 3 intents require causal reasoning, hypothetical projection, or multi-constraint optimization across sites. This taxonomy is not a property of the natural language itself but of the decision process that the intent demands. It lets us analyze how different reasoning strategies behave as decision complexity grows.

The agent interacts with the infrastructure exclusively through a typed tool interface. Two families of tools are exposed. The first family grants read access to the infrastructure's energy state. It includes tools to retrieve aggregate site power, per-pod energy-consumption ranked by magnitude, host-level power measured at the power distribution unit, energy time series for trend analysis, and a cross-site comparison that enables evidence-based placement decisions. The second family grants the orchestrator write access. It includes tools to query deployment status, scale deployments, migrate workloads between sites, and operate a deployment in both sites simultaneously.

Each tool is described to the agent using a natural-language signature that specifies its purpose and parameters. The agent chooses which tools to call, in what order, and with what arguments. This separation is important for the analysis that follows. The telemetry tools are side-effect-free and can be called repeatedly at low infrastructural cost. Actuation tools modify the cluster's live state and can only be evaluated by executing them. The energy cost of the agent's reasoning, as

TABLE I: Qualitative comparison of energy-aware orchestration approaches.

Approach	Energy	QoS	Telemetry	Runtime	Multi-site	Tools/API	NL agent	Agent Cost	Real Eval.
Munir, S. et al. [10]	●	○	○	●	○	○	○	○	●
Premsankar, G. et al. [11]	●	●	○	●	○	○	○	○	●
Aguirre-Usandizaga, J. et al. [12]	●	○	●	○	○	○	○	○	●
Kohli, H. et al. [13]	●	●	●	●	○	○	○	○	○
Kadhim, M. et al. [14]	●	●	●	●	○	○	○	○	○
Suryadevara, S. [15]	●	●	●	●	●	●	○	○	●
Zawish, M. et al. [16]	●	○	●	○	○	○	○	○	○
Vance, E. [17]	●	●	○	○	○	○	○	○	○
Maisto, V. et al. [18]	●	●	●	●	○	○	○	○	●
Moore, H. et al. [19]	●	●	○	●	●	○	○	○	●
Our	●	●	●	●	●	●	●	●	●

opposed to the cost of the tools themselves, is therefore a distinct and measurable quantity.

The first is a single-shot tool-use baseline, in which the model receives the intent together with the tool catalog and is expected to emit the required tool calls in a single generation.

The second is an iterative strategy based on the ReAct paradigm, in which the agent alternates between reasoning steps and tool invocations. After each tool call, the agent observes the result and decides whether to continue reasoning, call another tool, or terminate with a final answer. This strategy generates multiple inference requests per intent, and the number of requests increases with decision complexity. It offers explicit self-correction at the cost of additional inferences.

The third is a plan-and-execute strategy, in which the agent first emits a complete sequence of tool calls based solely on the intent and the tool catalog, and then executes the sequence without further reasoning. This strategy separates planning from execution. In our implementation, it may require additional model calls for plan normalization or final response synthesis, but it does not perform the observation-reasoning loop used by ReAct.

These strategies occupy distinct positions on the space defined by reasoning overhead and adaptivity. The experimental evaluation in Section V measures where each strategy lands in practice.

Figure 2 details the internal loop that the agent executes when processing an intent. The agent first combines the intent with a system prompt that describes the available tools and the orchestration context. The resulting prompt is submitted to the language model, which produces either a textual response, one or more tool calls, or both. When the model emits tool calls, it dispatches each call to the corresponding telemetry or actuation tool. The tool result is appended to the conversation and, if the strategy permits, fed back to the model for the next reasoning step. The loop terminates when the model emits a final textual answer, when a strategy-specific step budget is exhausted, or when a tool returns a terminal condition.

A key observation is that the single-shot strategy performs only the first inference and terminates after the initial response, regardless of whether it emits valid tool calls.

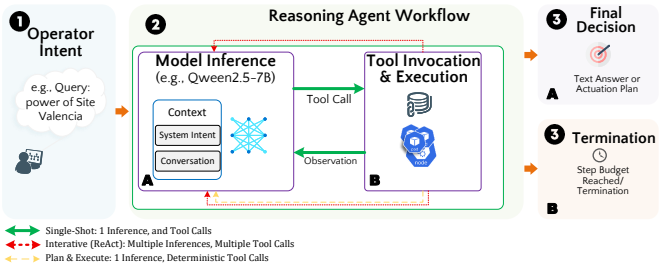


Fig. 2: Agent reasoning loop.

Because the method treats reasoning cost as a measurable quantity, we introduce an explicit attribution scheme that separates three contributions to the total energy associated with an intent. We quantify energy consumption across three layers. We measure inference energy at the compute accelerator hosting the language model. Managed workload energy is measured at the power distribution unit and attributed at each site's software layer. We estimate the energy impact of each agent action from the change in infrastructure energy consumption observed before and after the intervention over a fixed measurement window.

For action intents, the observed short-horizon infrastructure-energy difference is computed as $E_{\text{saved}} = (P_{\text{before}} - P_{\text{after}})\Delta t$, where P_{before} and P_{after} are the mean cluster powers measured over one-second windows immediately before and after the action, respectively, and $\Delta t = 1$ second. We then define the net energy benefit as $NEB = E_{\text{saved}} - E_{\text{GPU}}$, where E_{GPU} is the accelerator energy consumed during the complete decision window. A positive NEB indicates that the observed infrastructure-energy reduction exceeded the decision energy within the evaluated observation window. A negative value indicates that the decision energy was not recovered within that window; it does not establish whether the action could be energetically amortized over a longer operational period.

The end-to-end workflow, illustrated in Figure 1, proceeds as follows. The system receives an intent and pairs it with the system prompt. The agent enters the reasoning loop described above, calling telemetry tools to gather evidence about the

TABLE II: Benchmark of ten natural language intents used to evaluate the proposed method.

Level	ID	Natural Language Intent	Action
L1	L1_01	What is the current power consumption of site Aveiro?	No
	L1_02	Scale the web service deployment to 3 replicas in Valencia.	Yes
	L1_03	Which pod is consuming the most energy in Aveiro right now?	No
L2	L2_04	Reduce energy consumption in Aveiro by 20%.	Yes
	L2_05	Move the workload from Aveiro to Valencia.	Yes
	L2_06	Balance energy consumption evenly across both sites.	Yes
	L2_07	Which site is more energy-efficient per pod right now?	No
L3	L3_08	Energy in Aveiro spiked in the last 5 minutes. What happened?	No
	L3_09	If I deploy 5 more replicas of web service, which site would have lower total energy cost?	No
	L3_10	Optimize overall energy across both sites without reducing total replicas.	Yes

sites' energy state and calling actuation tools when a decision requires changing the live cluster. Throughout this process, the compute accelerator energy is sampled at fixed intervals and integrated over time to obtain the intent's reasoning cost. Before and after any actuation, the monitoring layer records the infrastructure energy state to enable the attribution described above. Once the agent produces its final answer, the NEB is computed for every action-oriented benchmark intent from the paired infrastructure measurements and the corresponding decision energy. Configurations that produced no actuation are retained as non-action baselines, and the full decision record is persisted for analysis.

This workflow is strategy agnostic. The same pipeline serves all three reasoning strategies by varying only the step budget and the prompt instructions provided to the agent. This property is what makes the comparative evaluation in Section V possible: the reasoning strategy is the independent variable, while the rest of the system is held constant.

IV. Experimental Testbed

We evaluate the proposed method on a two-site Kubernetes deployment spanning two European data centers, with the reasoning agent hosted at a third geographic location. Figure 3 depicts the overall topology and the flow of telemetry and orchestration traffic across the three sites. Table II summarizes the benchmark intents used in the evaluation.

The agent and language model run on a workstation in Brazil equipped with an Intel Xeon w3-2435 CPU (8 cores, 30 GiB RAM) and an NVIDIA RTX 4000 Ada GPU (20 GiB VRAM, 130 W TDP). The GPU serves Qwen2.5-7B-Instruct via vLLM 0.12.0, with the agent built on LangChain 1.2.0 and LangGraph 1.0.5.

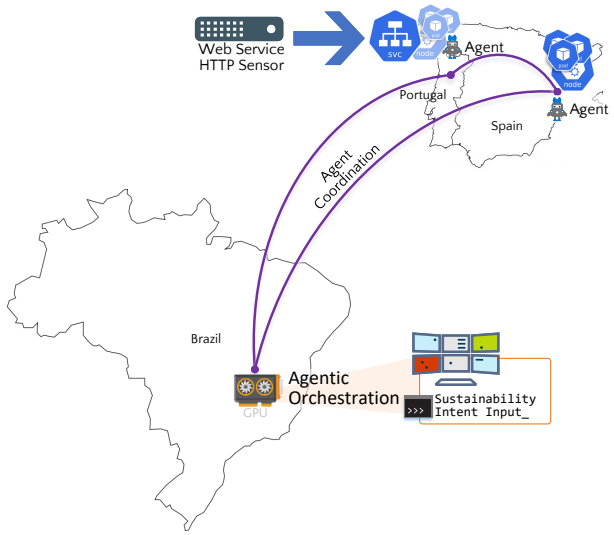


Fig. 3: Three-tier experimental topology.

The infrastructure consists of a single Kubernetes 1.28.15 cluster split across two sites connected by a Virtual Private Network (VPN) tunnel: the primary site at ITAV (Aveiro, Portugal) hosting 3 control plane and 3 worker nodes, and a secondary compute site at UPV (Valencia, Spain) with 3 worker nodes. Control plane responsibilities reside at Aveiro, while Valencia acts as a compute pool.

ITAV nodes run Ubuntu 24.04 on kernel 6.8 with containerd as the container runtime. UPV nodes run Ubuntu 22.04 on kernel 5.15 with the same runtime. The cluster uses Flannel for pod networking over VXLAN. The sites are reachable via private addresses: 10.255.40.0/24 for Aveiro and 172.21.32.0/24 for Valencia, and traffic between them traverses the VPN tunnel that connects the two networks.

The workload under orchestration is a lightweight distributed web application deployed as a Kubernetes Deployment with no explicit resource requests or limits. The agent scales, migrates, and duplicates replicas of this deployment across the two sites. Because the sites host no competing workloads during the experiments, the application's contribution to each worker's power envelope is small relative to the host's baseline consumption. This property is relevant to the interpretation of the NEB reported in Section V.

Each worker node is instrumented to measure energy at three levels, which together provide a layered view of consumption from the physical outlet down to the process. At the physical layer, a PDU measures the bare-metal power drawn by each host. At the operating system layer, Intel RAPL exposes per-socket energy counters for the CPU package and its memory controller. At the container orchestration layer, Kepler runs as a per-node DaemonSet in a dedicated namespace and attributes energy to individual pods and containers by combining RAPL counters with cgroup-level resource accounting. All three data sources are published to a single InfluxDB instance at one-second granularity, and the agent consumes them via the Holosense platform, which exposes the underlying store as a query API. The instrumentation covers both Aveiro and

Valencia workers symmetrically, enabling the agent to compare the two sites on an equivalent basis.

The power draw of the Graphics Processing Unit (GPU) hosting the language model is sampled at 1 Hz through the NVIDIA Management Library, and the resulting power trace is integrated using the trapezoidal rule to estimate the GPU energy consumed per intent. The integration spans the full decision window, including LLM inference, tool execution, and communication delays, and therefore includes the accelerator's power draw during these activities. It does not include the energy consumed by the agent host's CPU, memory, or network interface; consequently, E_{GPU} represents accelerator-side decision energy rather than the total control-plane energy of orchestration.

The evaluation comprises two phases that differ only in their prefix-cache configuration. Phase 1 uses the FP16 model with vLLM's prefix cache enabled, whereas Phase 2 uses the same model and precision with the prefix cache disabled. In each phase, all 10 benchmark intents are evaluated under each of the three reasoning strategies, with five repetitions per intent–strategy combination executed in randomized order. Before every action intent, the cluster is automatically restored to a canonical baseline in which Aveiro hosts one replica and Valencia hosts none, thereby providing a comparable initial placement state across runs. A 5-second cooldown is inserted between consecutive runs to reduce short-term thermal carryover on the GPU.

Every run produces a record that combines four sources of evidence. The GPU energy trace is produced locally on the agent host. The inference-level telemetry, which includes prompt and generation token counts, as well as the prefix cache hit rate, is obtained from vLLM's Prometheus endpoint and differenced over the run window. The cluster energy state, sampled at both the PDU and Kepler layers, is retrieved from Holosense immediately before and after any action intent. The agent's orchestration loop records the reasoning trace, which captures the sequence of model inferences and tool invocations. The four sources are aligned on a common timeline and persisted in a single CSV file per phase for subsequent statistical analysis.

V. Results and Discussion

This section evaluates how the reasoning strategy, intent complexity, prefix caching, and workload magnitude, relative to the cluster baseline, shape the energy cost, operational effectiveness, and net benefit of agentic orchestration.

Energy Cost Across Reasoning Configurations. Figure 4 and Table III compare mean GPU energy across the three reasoning configurations and complexity levels under the cache-enabled condition. For Tool Calling, mean energy increased from 48 J for simple intents to 264 J for complex intents. ReAct increased from 395 J to 3,277 J, corresponding to 8.3× and 12.4× the energy of the non-actionable single-shot baseline, respectively. Plan & Execute ranged from 449 J to 1,476 J and consumed substantially less energy than ReAct for complex intents. These measurements quantify computational cost; the actionability of the configurations is examined separately below.

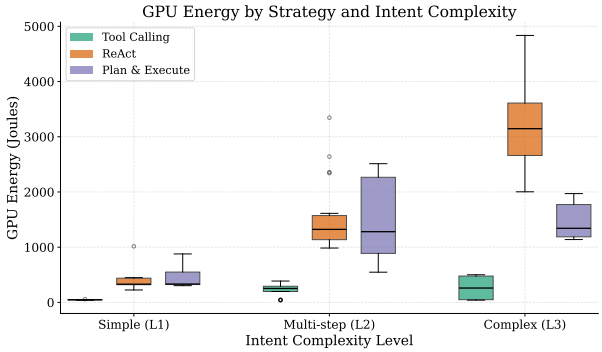


Fig. 4: GPU energy by reasoning configuration and intent complexity with prefix caching enabled. ReAct consumed approximately 8–12× the energy of the non-actionable single-shot Tool Calling baseline.

The observed energy differences relate to the number of LLM inference calls required per intent. Table III shows that Tool Calling consistently performs one inference call regardless of complexity, whereas ReAct increases from 2.0 calls for simple intents to 5.3 for complex intents, and Plan & Execute increases from 2.0 to 3.0 calls. Each additional inference extends the decision window and processes newly generated tokens within an expanding context. Although prefix caching can reuse shared portions of this context, it does not eliminate the energy cost of subsequent inference steps. The number of calls and accumulated context therefore help explain the higher energy consumption of the iterative configurations.

TABLE III: Mean GPU energy (± SD), LLM and tool calls, and latency by reasoning configuration and intent complexity with prefix caching enabled. Five repetitions per intent–configuration pair; n is pooled by complexity level.

Strategy	Level	n	Energy (J)	LLM	Tools	Lat. (s)
Tool Calling	L1	15	48 ± 5	1.0	0.0	1.6
	L2	20	233 ± 124	1.0	0.0	3.5
	L3	15	264 ± 187	1.0	0.0	3.8
ReAct	L1	15	395 ± 183	2.0	1.0	6.6
	L2	20	1,532 ± 643	3.1	2.6	20.9
	L3	15	3,277 ± 817	5.3	5.8	36.8
Plan & Exec	L1	15	449 ± 199	2.0	1.0	6.8
	L2	20	1,450 ± 747	2.5	2.5	19.9
	L3	15	1,476 ± 308	3.0	4.3	16.3

Single-Shot Tool Calling: Low Cost but Non-Actionable. Table III and Figure 5 show that the single-shot configuration produced no tool invocations across the 100 evaluated runs. Under the adopted prompt and Qwen2.5-7B-Instruct configuration, the model generated textual responses instead of structured tool calls. Consequently, this configuration neither retrieved infrastructure telemetry nor executed Kubernetes actions. Its energy measurements therefore represent a low-cost, non-actionable generation baseline rather than an operationally equivalent orchestration strategy. This result is specific to the evaluated model and configuration because tool-use behavior may depend on the model, prompt, chat template,

and structured-output implementation.

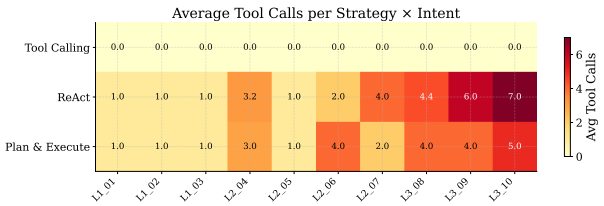


Fig. 5: Average tool calls per strategy and intent.

Figure 6 illustrates the energy–actionability trade-off observed in the experiment. ReAct and Plan & Execute produced executable tool calls across all evaluated complexity levels, corresponding to a 100% tool-call actionability rate, whereas the single-shot configuration produced none. Because the single-shot configuration completed no operational task, its lower energy consumption should not be interpreted as greater orchestration efficiency. In this setup, the two actionable strategies consumed approximately 8–12 times as much GPU energy as the non-actionable baseline, demonstrating that the energy cost of achieving actionability must be accounted for explicitly. The reported rate indicates only whether a strategy produced executable tool calls; it does not assess the semantic correctness or global optimality of the resulting orchestration decision.

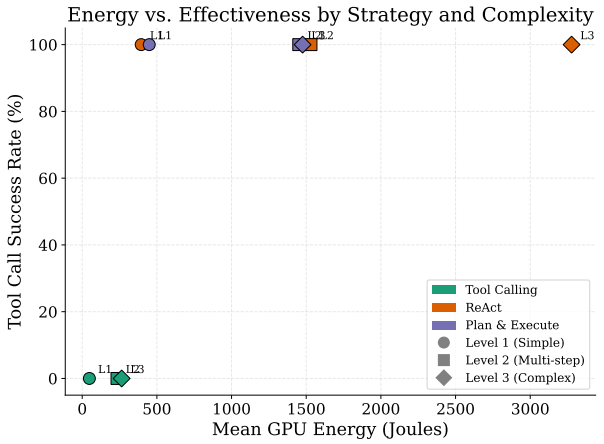


Fig. 6: GPU energy versus tool-call actionability rate.

Statistical Validation. The assumptions of the standard two-way ANOVA were not uniformly satisfied: homogeneity of variance was violated for both ReAct and Tool Calling, and residual normality was additionally violated for Tool Calling. We therefore complemented the ANOVA with Welch's t-test and the non-parametric Kruskal-Wallis and Mann-Whitney U tests, as reported in Table IV. For the ReAct and Tool Calling configurations included in this analysis, intent complexity exhibited the largest reported effect on GPU energy. Table V further shows that complex intents consumed 4.5–8.3 times more energy than simple intents across the evaluated cache conditions.

Strategy-Dependent Effect of Prefix Caching. Figure 7 shows that enabling prefix caching reduced mean ReAct GPU energy by 29.4%, with reductions of 59.5%, 32.8%, and 25.2% for levels L1, L2, and L3, respectively. The pooled ANOVA identified a significant cache effect, whereas Welch's test was marginal under heterogeneous variance; level-specific comparisons remained significant for L1 and L3. For Tool Calling, the observed mean reduction was 9.3% and was not statistically significant. This contrast is consistent with ReAct's repeated inference calls reusing a growing shared prefix, whereas the single-shot configuration provides no accumulated prefix reuse across inference steps.

TABLE IV: Statistical analysis of GPU energy (Joules) across Cache and Complexity factors.

Strategy	Test	Factor	Statistic	p-value (Sig.)	Effect Size
Assumptions: Shapiro-Wilk OK ($p = 0.103$), Levene VIOLATED ($p < 0.001$)					
ReAct	ANOVA (Type II)	Cache	$F = 15.17$	2.6×10^{-4}	$\eta^2 = 0.053$
		Complexity	$F = 211.25$	1.2×10^{-20}	$\eta^2 = 0.744$
		Interaction	$F = 1.48$	0.229 (ns)	$\eta^2 = 0.005$
	Welch t	Cache	$t = -1.81$	0.076 (ns)	$d = -0.48$
		Complexity	$t = -12.99$	$< 10^{-6}$	$d = -3.41$
	Welch t (L1)	Cache	$t = -3.53$	0.003	$d = 1.34$
	Welch t (L3)	Cache	$t = -2.76$	0.011	$d = 1.04$
Assumptions: Shapiro-Wilk VIOLATED ($p < 0.001$), Levene VIOLATED ($p < 0.001$)					
Tool Calling	ANOVA (Type II)	Cache	$F = 0.43$	0.516 (ns)	$\eta^2 = 0.004$
		Complexity	$F = 59.53$	2.3×10^{-10}	$\eta^2 = 0.511$
		Interaction	$F = 0.43$	0.513 (ns)	$\eta^2 = 0.004$
	Kruskal-Wallis	All groups	$H = 29.36$	$< 10^{-5}$	—
	Mann-Whitney U	Cache	$U = 411$	0.569 (ns)	$r = 0.09$
		Complexity	$U = 92$	$< 10^{-6}$	$r = 0.80$
	Welch t (L1)	Cache	$t = 0.11$	0.916 (ns)	$d = -0.04$
	Welch t (L3)	Cache	$t = -0.66$	0.517 (ns)	$d = 0.25$

TABLE V: Pairwise complexity comparisons (Simple vs. Complex) per cache condition.

Strategy	Condition	Simple (J)	Complex (J)	Ratio	p (Welch)
ReAct	Cache ON	395	3,277	8.3×	$< 10^{-6}$
	Cache OFF	975	4,383	4.5×	$< 10^{-6}$
Tool Calling	Cache ON	48	264	5.5×	< 0.001
	Cache OFF	48	304	6.4×	$< 10^{-5}$

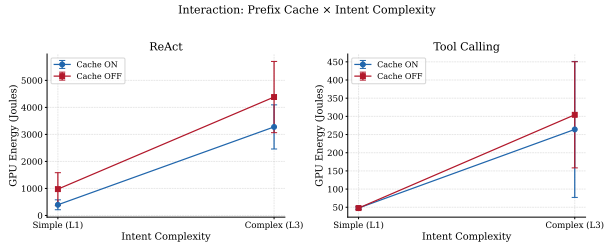


Fig. 7: Effect of prefix caching and intent complexity on GPU energy.

Figure 8 compares per-intent ReAct energy with prefix caching enabled and disabled. The largest absolute reduc-

tion occurred for the complex intent L3_10, where energy decreased from 5,728 J to 4,197 J. This pattern is consistent with key-value (KV) prefix caching: successive LLM calls in a ReAct loop reuse a prefix containing the system prompt, tool definitions, and prior reasoning steps, allowing cached key-value representations to be reused rather than recomputed. Longer reasoning traces provide more opportunities for such reuse, although the resulting energy reduction also depends on context composition and runtime implementation.

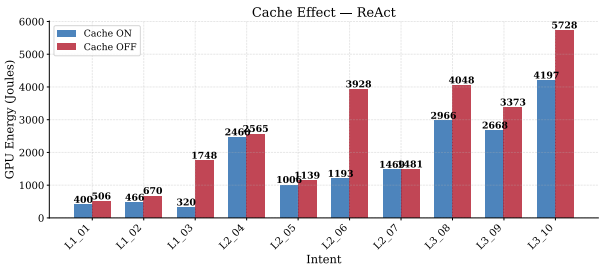


Fig. 8: Per-intent ReAct GPU energy with prefix caching enabled (blue) and disabled (red).

Short-Horizon Infrastructure Impact and NEB. Figure 9 reports the NEB for the five action-oriented benchmark intents. Consistent with the definition in Section III, $NEB = E_{\text{saved}} - E_{\text{GPU}}$, where E_{saved} is the infrastructure-energy difference observed across the one-second pre- and post-action windows and E_{GPU} is the accelerator energy consumed during the decision. The resulting values therefore characterize the short-horizon energy balance of each evaluated configuration, rather than the total savings that an action might accumulate over its operational lifetime.

Table VI reports a negative NEB for 14 of the 15 strategy–intent combinations under the evaluated one-second attribution horizon. The sole positive value, obtained for Tool Calling on L1_02, cannot be attributed to orchestration because this configuration produced no tool calls and executed no infrastructure action. Instead, it reflects short-term variation between the two one-second cluster-power measurement windows used to compute E_{saved}. The Tool Calling entries should therefore be interpreted as a non-action baseline that illustrates the sensitivity of short-window NEB estimates to background power fluctuations.

Figure 10 shows no consistent reduction in total cluster power between the one-second pre- and post-action observation windows. Because the lightweight web-service workload contributes only a small fraction of each host's overall power draw, any action-induced change is difficult to distinguish from short-term variation in the infrastructure baseline. Consequently, the observed infrastructure-energy differences were generally insufficient to offset the GPU energy consumed by LLM reasoning within the evaluated observation window.

Kubernetes execution records confirmed that the actionable strategies completed the intended operations. ReAct scaled down replicas for L2_04 and migrated the workload for L2_05, while Plan & Execute completed the migration for L2_05. The predominantly negative NEB values should therefore not be interpreted as evidence that these actions failed to execute.

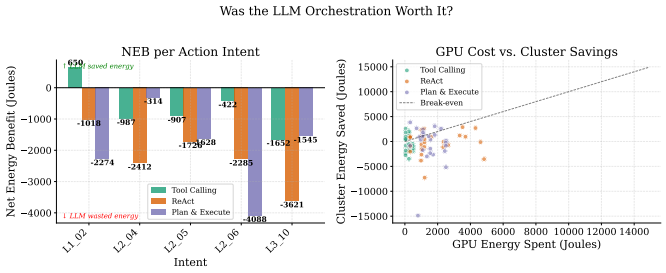


Fig. 9: Short-horizon NEB by action-oriented intent (left) and GPU decision energy versus observed infrastructure-energy difference (right).

TABLE VI: Short-horizon NEB for action-oriented intents with prefix caching enabled.

Strategy	Intent	GPU (J)	Saved (J)	NEB (J)
Tool Calling	L1_02	46	696	**+650**
	L2_04	255	−732	−987
	L2_05	44	−864	−908
	L2_06	254	−168	−422
	L3_10	488	−1,164	−1,652
ReAct	L1_02	466	−552	−1,018
	L2_04	2,460	48	−2,412
	L2_05	1,006	−720	−1,726
	L2_06	1,193	−1,092	−2,285
	L3_10	4,197	576	−3,621
Plan & Exec	L1_02	606	−1,668	−2,274
	L2_04	1,070	756	−314
	L2_05	1,664	36	−1,628
	L2_06	2,492	−1,596	−4,088
	L3_10	1,569	24	−1,545

Rather, under the evaluated lightweight workload and one-second observation windows, their power effects were not consistently distinguishable from short-term variation in the cluster baseline at the PDU level. This result is specific to the evaluated workload and observation horizon and does not rule out a positive NEB for larger workloads or longer operational horizons.

Implications. Three main observations follow from these results. First, LLM reasoning introduces a measurable GPU

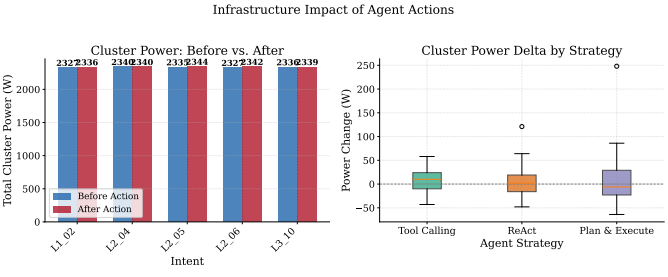


Fig. 10: Cluster PDU power before and after each orchestration attempt (left) and corresponding power difference by reasoning configuration (right).

energy cost ($\sim$3,000 J for complex ReAct intents) that must be explicitly budgeted in sustainable orchestration. Second, prefix caching provides a key optimization for iterative reasoning, reducing mean ReAct GPU energy by 29.4%. Third, negative NEB values reflect our lightweight workload and 1-second observation window; larger workloads or longer observation horizons may amortize decision overhead and achieve net positive energy gains.

VI. Concluding Remarks

Energy-aware orchestration across geographically distributed infrastructures requires accounting for both the energy consumed by managed workloads and the GPU energy cost of automated decision-making. This paper presented an agentic framework that measures these components jointly, introduced the NEB to compare short-horizon infrastructure-energy changes with GPU decision energy, and evaluated three reasoning configurations on a real multi-site Kubernetes testbed. For the evaluated lightweight workload and one-second attribution horizon, the observed infrastructure-energy differences were generally insufficient to offset the energy consumed by LLM reasoning. ReAct and Plan & Execute produced actionable tool calls, whereas the single-shot Tool Calling configuration did not; prefix caching reduced mean ReAct GPU energy by 29.4%. These findings show that the sustainability of agentic orchestration depends jointly on actionability, reasoning strategy, workload magnitude, and the period over which operational savings are evaluated. Future work will investigate larger and more energy-intensive workloads, additional model families, adaptive strategies that match reasoning depth to intent complexity, longer post-action observation periods, and carbon-aware placement based on renewable-energy availability.

Acknowledgment

The authors acknowledge support from FAPEMIG under Grant #APQ00923-24; the FAPESP MCTIC/CGI Research Project SFI2 – Slicing Future Internet Infrastructures under Grant 2018/23097-3; and Fundação para a Ciência e a Tecnologia (FCT) through the R&D Unit Project UID/00319/2025, Centro ALGORITMI (ALGORITMI/UM). The authors also acknowledge support from the 6G-VERSUS project, funded by the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101192633.

References

[1] J. Yang, Z. Saad, J. Wu, X. Niu, H. Leung, and S. Drew, "A survey on task scheduling in carbon-aware container orchestration," 2025. [Online]. Available: https://arxiv.org/abs/2508.05949

[2] K. Danach, K. Hamze, H. Harb, and H. Kanj, "Carbon-aware scheduling in cloud computing operations: A multi-objective optimisation approach," *IET Smart Grid*, vol. 9, no. 1, p. e70056, 2026. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/stg2.70056

[3] S. M. P. Andersen, L. C. B. Mumberg, H. Golmohamadi, and M. Albano, "Kubernetes scheduling for green-powered microgrid data centers," in *Energy Informatics*, B. N. Jørgensen, Z. G. Ma, and S. Yussof, Eds. Cham: Springer Nature Switzerland, 2026, pp. 290–306.

[4] R. Moreira, R. Pasquini, J. S. Martins, T. C. Carvalho, and F. de Oliveira Silva, "Ai-driven orchestration at scale: Estimating service metrics on national-wide testbeds," *Future Generation Computer Systems*, vol. 174, p. 107971, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X25002663

[5] B. Kocot, P. Czarnul, and J. Proficz, "Energy-Aware Scheduling for High-Performance Computing Systems: A Survey," *Energies*, vol. 16, no. 2, 2023. [Online]. Available: https://www.mdpi.com/1996-1073/16/2/890

[6] S. N. Agos Jawaddi, A. Ismail, M. S. Sulaiman, and V. Cardellini, "Analyzing Energy-Efficient and Kubernetes-Based Autoscaling of Microservices Using Probabilistic Model Checking," *Journal of Grid Computing*, vol. 23, no. 1, p. 3, Dec 2024. [Online]. Available: https://doi.org/10.1007/s10723-024-09789-9

[7] L. Mohamed, K. I. Hamza, and A. Driss, "A Survey of Energy Efficiency for Kubernetes Clusters," in *2025 International Conference on Intelligent Systems: Theories and Applications (SITA)*, 2025, pp. 1–6.

[8] Y. Lee and E. Park, "Toward Sustainable Agentic AI Systems: A Survey of Architectures and Methodologies," *Sustainable Development*, vol. n/a, no. n/a. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sd.70942

[9] Z. Bellal, L. Lahlou, N. Kara, T. Murphy, T. Phat Nguyen, A. Ahmed, and M. Perez-Jimenez, "Investigating the potential of kepler toward power observability for sustainable cloud computing," *IEEE Transactions on Green Communications and Networking*, vol. 10, pp. 2118–2135, 2026.

[10] M. S. Munir, N. H. Tran, W. Saad, and C. S. Hong, "Multi-Agent Meta-Reinforcement Learning for Self-Powered and Sustainable Edge Computing Systems," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3353–3374, 2021.

[11] G. Premsankar and B. Ghaddar, "Energy-Efficient Service Placement for Latency-Sensitive Applications in Edge Computing," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 17 926–17 937, 2022.

[12] J. Aguirre-Usandizaga, A. I. Torre-Bastida, M. L. D. la puerta, and A. Almeida, "A forecasting-based perception layer for energy-aware resource management in LLM serving deployed on high-performance computing clusters," *Sustainable Computing: Informatics and Systems*, vol. 50, p. 101333, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210537926000430

[13] H. Kohli, A. Hadi, N. Mukhi, M. A. Miah, and K. Bushra Siddiqa, "Energy-aware intelligent computing framework for sustainable ai workloads in next-generation smart systems," vol. 2, p. 34–47, Jan. 2026. [Online]. Available: https://submissions.adroidjournals.com/index.php/ijssic/article/view/201

[14] M. A. Kadhim, A. R. Hamad, and Z. T. Ibraheem, "Advanced AI-Based Energy Efficiency and Network Performance Optimization in Next-Generation Wireless Systems Using Distributed Collective Intelligence," *Journal of Electrical and Computer Engineering*, vol. 2026, no. 1, p. 2275315, 2026. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/jece/2275315

[15] S. S. K. Suryadevara, "AI-Driven Carbon-Aware Orchestration for Sustainable Cloud Deployments," *International Journal of Emerging Research in Engineering and Technology*, vol. 5, no. 2, p. 165–175, Jun. 2024. [Online]. Available: http://ijeret.org/index.php/ijeret/article/view/555

[16] M. Zawish, N. Ashraf, R. I. Ansari, and S. Davy, "Energy-Aware AI-Driven Framework for Edge-Computing-Based IoT Applications," *IEEE Internet of Things Journal*, vol. 10, no. 6, pp. 5013–5023, 2023.

[17] E. Vance, "An intelligent optimization model for energy-efficient cloud computing systems," *Computer Science and Engineering Transactions*, vol. 1, no. 1, Mar. 2026. [Online]. Available: https://csetx.org/index.php/cset/article/view/92

[18] V. Maisto and A. Cilardo, "Compute-Efficient Modelling of Multi-NPU Inference on Edge MPSoCs for Energy-Aware Online Workload Allocation," vol. 23, no. 1, Mar. 2026. [Online]. Available: https://doi.org/10.1145/3799238

[19] H. Moore, S. Qi, N. Hogade, D. Milojicic, C. Bash, and S. Pasricha, "Sustainable Carbon-Aware and Water-Efficient LLM Scheduling in Geo-Distributed Cloud Datacenters," in *Proceedings of the Great Lakes*

Symposium on VLSI 2025, ser. GLSVLSI '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 929–934. [Online]. Available: https://doi.org/10.1145/3716368.3735301