
Sustainable Control of Geo-Distributed Datacenters by Distilling Numerical Experts into Adaptive LLM Agents

Antonio Guillen-Perez* Ashwin Ramesh Babu* Sahand Ghorbanpour*
Avishek Naug Vineet Gundecha Sifat Muhammad Abdullah
Ricardo Luna Gutierrez Soumyendu Sarkar*†

Hewlett Packard Enterprise

{antonio.guillen, ashwin.ramesh-babu, sahand.ghorbanpour, avishek.naug,
vineet.gundecha, sifat-muhammad.abdullah, rluna, soumyendu.sarkar}@hpe.com

Abstract

The sustainable control of geo-distributed datacenters is a critical systems challenge, defined by large-scale, dynamic, and uncertain operating conditions. While specialized numerical experts, such as those from Reinforcement Learning (RL) or Model Predictive Control (MPC), can be trained to find optimal control policies, their practical deployment is blocked by fundamental systems-level flaws: they are brittle, failing to scale with the system; opaque, preventing operator trust; and rigid, unable to adapt to new runtime objectives. This paper introduces a novel framework that directly addresses these issues by **distilling the policy of a numerical expert into an adaptive LLM agent**. Our method transforms the expert’s opaque logic into a transparent, interactive, and agentic workflow. To validate this approach, we distill a state-of-the-art RL policy for carbon-aware workload orchestration. Evaluated in a high-fidelity simulation, our resulting LLM agent demonstrates the capabilities essential for real-world systems deployment. It solves the scalability problem, successfully managing topologies more than three times larger than the expert’s training environment. It enables true runtime adaptability, altering its strategy in minutes in response to complex operator commands that would require days of costly retraining for the original expert. By making powerful optimizers manageable and resilient, our work offers a practical pathway to the sustainable control of large-scale computer systems.

1 Introduction

The sustainable control of geo-distributed datacenters is a defining systems challenge of the modern AI era. As energy consumption from large-scale models grows unsustainably (1; 2), the need for intelligent, carbon-aware control systems becomes critical. This requires orchestrating computation across a heterogeneous, globe-spanning infrastructure to capitalize on real-time differences in renewable energy and grid carbon intensity (3).

This complex control problem has been a target for powerful numerical optimization techniques, with expert controllers developed using Reinforcement Learning (RL), Model Predictive Control (MPC), and others (4; 5; 6). However, these numerical experts are fundamentally brittle, suffering from a **Scalability Failure**: a policy tuned for an N-node system fails when the infrastructure scales. They

*These authors contributed equally.

†Corresponding author.

are also opaque "**Black Boxes**" that prevent the operator trust and auditability essential for managing high-stakes systems. Most critically, they exhibit a profound **Adaptability Failure**, as their fixed numerical inputs prevent them from responding to new operator goals or dynamic system changes without a costly full retraining, a crippling limitation for resilient runtime optimization.

This paper introduces a general framework that solves these issues by **distilling the policy of a numerical expert into an Adaptive LLM Agent**. Our method transforms a rigid policy into an interactive, agentic workflow, creating a new class of controller that achieves the properties essential for deployment: system scalability, runtime adaptability, and operational manageability.

2 Related Work

The application of ML to optimize computer systems for sustainability has evolved from single-node power management to the complex orchestration of geo-distributed resources (7). While numerical experts like RL have been used for datacenter scheduling (8; 9; 10; 11; 12), their outputs are typically brittle and opaque policies that fail in production. A distinct trend applies Large Language Models (LLMs) to systems challenges, often casting them as high-level system designers (13; 14), zero-shot agents (15), or as 'teachers' in policy distillation to guide smaller agents (16; 17; 18; 19). Our work introduces a crucial inversion of this paradigm. Instead of distilling knowledge *from* a generalist LLM, we distill the policy *from* a specialized numerical expert *into* an LLM. This transforms the opaque expert into an agentic copilot, equipping it with the scalability, manageability, and adaptive runtime optimization that were absent in the original expert and are the focus of our contribution.

3 Method: Distilling a Numerical Expert into an Adaptive LLM Agent

Our method transforms any specialized numerical expert, such as an advanced numerical heuristic, an MPC controller, or an RL agent, into a flexible, language-based agentic copilot using the four-stage distillation framework shown in Figure 1.

The General Distillation Framework. The framework first uses a pre-existing numerical expert as an oracle to **generate a dataset** of state-action trajectories from a high-fidelity environment. The critical translational step is to **textualize** this raw numerical data. This is a crucial, human-in-the-loop design step that creates a semantic, language-based API for the control task. For instance, a numerical state vector representing datacenter loads and carbon intensities is translated into a human-readable string: "Cluster Status: DC-1 {CPU Avail: 80%, CI: 194 g/kWh, ... }". The expert's numerical action vector is similarly translated into a target RESPONSE string. Finally, this textualized dataset is used to **fine-tune a base LLM**, teaching it to map from a linguistic description of a system state to the optimal control action, effectively instilling it with the expert's specialized knowledge.

Case Study: Distillation of a Reinforcement Learning Expert. To validate this framework, we conduct a case study using a state-of-the-art hierarchical RL agent as our numerical expert. Trained with Proximal Policy Optimization (PPO) (20), this expert learns to control the system by decomposing the problem into the two distinct roles shown in Figure 2. This hierarchical structure mirrors real-world operations, separating global strategic planning (the Manager) from local, time-sensitive execution (the Worker). To preserve this expert strategy, we apply our framework to

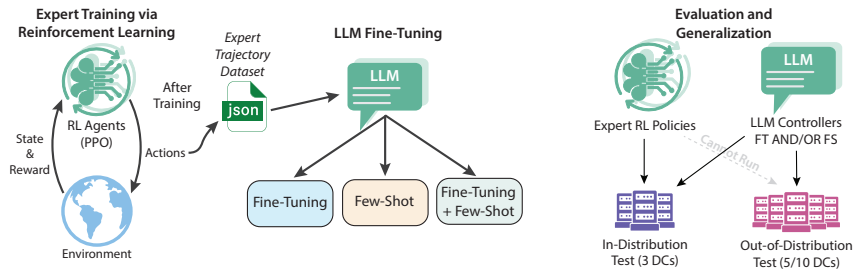


Figure 1: The proposed distillation framework. (1) Any expert numerical controller is used as an oracle. (2) It generates a dataset of state-action trajectories. (3) This numerical data is textualized into structured prompts. (4) An LLM is then fine-tuned on this data to create the final agentic controller.

fine-tune two separate, specialized LLMs from open-source models (e.g., Llama 3.2, Qwen 3). A **Manager LLM** is trained on the geographical routing data, and a **Worker LLM** is trained on the temporal decision data. To find the most effective distillation method, we systematically evaluate a hierarchy of LLM adaptation techniques: (1) **Few-Shot Prompting** on base models as a baseline; (2) **Parameter-Efficient Fine-Tuning (PEFT)** with Low-Rank Adaptation (LoRA) (21) to create the specialized agents; and (3) a **hybrid PEFT + Few-Shot** approach. This rigorous evaluation allows us to identify the optimal LLM-based reasoning controller. The full technical details of the RL agent’s formulation (Appendix A.1) and the LLM fine-tuning process (Appendix B) are provided in the appendix.

4 Results: Validating Scalability, Adaptability, and Manageability

Evaluation Environment. We evaluate our controller in a high-fidelity simulation (22) using real-world data, including Alibaba workload traces (23) across several system scales (see Appendix C.1 for full details).

4.1 Solving the Scalability Failure

First, we validate that our distilled LLM can match the expert’s performance on the 3-node system used for training. Table 1a shows our LLM finds a superior policy trade-off, matching the expert’s carbon reduction while improving the SLA violation rate and dramatically reducing the worst-case task delay by over 65%. Interestingly, the distilled LLM is not just a clone but a more balanced controller than the expert it learned from. We hypothesize that the distillation process, combined with the LLM’s pre-trained understanding of sequences and constraints, naturally smooths the expert’s aggressive, reward-maximizing behavior into a more practical and less pathological policy. This emergent benefit of distillation warrants further investigation

The primary breakthrough is that the fixed-input RL expert **cannot operate** on larger systems. In contrast, our LLM agent scales seamlessly. Table 1b shows its performance advantage over a heuristic baseline grows with system complexity, proving it has learned a robust, generalizable control strategy and has solved the scalability failure. The full, unabridged results tables and a detailed behavioral analysis of the controller policies can be found in Appendix C.

4.2 From Black Box to Intelligent Copilot: Quantifying the Adaptability Gap

The primary barrier to deploying advanced optimizers is their lack of trust and adaptability. Our framework addresses these by transforming the expert into an intelligent copilot.

Quantifying the Adaptability Gap: One-Time Training vs. Runtime Adaptation. A rigid controller is a brittle one. To quantify the LLM’s adaptability, we must distinguish between the *one-time, offline knowledge acquisition* (the initial expert training) and the *recurring, online runtime*

Table 1: Performance comparison. Left (a): In-distribution results on the 3 Geo-Distributed Datacenters. Right (b): Out-of-distribution generalization results on larger, unseen clusters (5/10 Geo-Distributed Datacenters).

(a) 3-Datacenter (In-Distribution)					(b) Generalization (Out-of-Distribution)			
Controller	Carbon (kg) ↓	Cost (\$) ↓	SLA (%) ↓	Delay (min) ↓	Scenario	Controller	Carbon (kg) ↓	SLA (%) ↓
Heuristic	24,637	10,131	1.33	112	5-DCs	Heuristic	62,777	1.48
RL Expert	22,912	8,974	1.56	1,095		Our LLM	59,558	1.08
Our LLM	22,511	8,817	1.15	355	10-DCs	Heuristic	131,830	1.95
						Our LLM	125,141	1.19

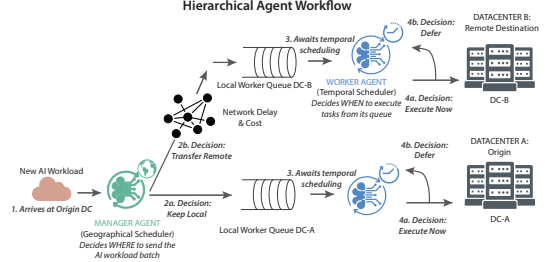


Figure 2: The hierarchical workflow of the RL expert, separating global (Manager) and local (Worker) control.

Table 2: Adaptation Cost and Performance: The LLM controller adapts in minutes by re-using the initial one-time training investment. An equivalent strategic shift for the RL expert requires a new, prohibitively costly investment for every change.

Controller	Adaptation Method	Recurring Cost	Energy Cost (\$)	Carbon (kg)
LLM Agent (Default)	N/A	N/A	8,817	22,511
LLM Agent (Adapted)	Natural Language	Minutes	7,671	25,930
RL Expert (Original)	<i>Cannot Adapt</i>	N/A	8,974	22,912
RL Expert (Retrained)	<i>Full Retraining</i>	<i>48+ GPU Hours</i>	7,615	26,850

adaptation (responding to new operator goals). Our framework treats the initial ~48 GPU-hour training of the RL expert as a one-time investment to create a rich knowledge base of optimal system behavior.

We then tested the recurring adaptation cost by issuing a complex command: "Minimize energy cost, but under the hard constraint that you may not send workloads to the California node due to a grid stability alert." Table 2 shows the result. The LLM agent adapts its policy in **minutes** by re-using the initial knowledge base. In contrast, the RL approach requires a new, costly ~48 GPU-hour investment for *every single strategic change*. The LLM's adaptation is not only orders of magnitude faster but also demonstrates a paradigm shift in runtime manageability, moving from a fixed policy to a reusable, adaptable reasoning engine. This shift in paradigm introduces a modest and quantifiable systems cost; a full analysis of the LLM agent's resource footprint and inference latency is provided in Appendix C.6.

4.3 Solving the Manageability and Adaptability Failures

The most profound advantage of our framework is its ability to transform the expert into an intelligent copilot, solving the core systems problems of trust and adaptability.

Enabling Manageability through Transparency. A system that cannot explain itself cannot be trusted. To solve the "black box" problem, our LLM copilot provides auditable, data-driven justifications for its control actions.

Decision Justification: Answering "Why?"

SITUATION: The controller routes workloads from Node 1 and Node 2 to Node 3. **LLM EXPLANATION:** "The goal is to minimize carbon. **Node 3 has a lower carbon intensity (112.0 g/kWh)** than Node 1 (194.0 g/kWh) and Node 2 (120.0 g/kWh). By routing to Node 3, we reduce the carbon footprint while ensuring it has sufficient resources."

Enabling Resilience through Runtime Adaptability. A rigid controller is a brittle one. Our copilot enables system resilience by adapting its strategy in real-time to new operator commands, a capability essential for managing unforeseen events.

Planning for an Unforeseen Event

OPERATOR COMMAND: "A grid emergency requires Node 2 to reduce power by 50% for one hour. Formulate a plan." **LLM STRATEGY SHIFT:** "Acknowledged. Implementing 'evacuation' protocol for Node 2. (1) **New Routing Policy:** Node 2 is now an invalid destination for all new workloads. (2) **Local Action:** The Worker agent at Node 2 will defer all non-critical tasks."

Appendix D provides a comprehensive benchmark of these copilot capabilities, including extended examples of conceptual understanding, counterfactual reasoning, and strategic adaptation.

5 Conclusion

This work introduced a general framework for the **sustainable control of geo-distributed datacenters by distilling numerical experts into Adaptive LLM Agents**. Our method transforms brittle, opaque, and rigid numerical policies into deployable, agentic copilots. Our case study, distilling an RL agent, demonstrated a controller that solves the fundamental systems-level flaws of its predecessor by achieving true **system scalability**, providing transparent **manageability** through explanations, and enabling **runtime adaptability** in minutes, not days.

By bridging the gap between theoretical optimization and the demands of real-world systems deployment, this work offers a method to make our most powerful optimizers reliable and manageable. This accelerates the development of the intelligent, agentic systems needed for the sustainable control of large-scale computer systems and provides a tangible pathway for climate action.

References

- [1] E. Strubell, A. Ganesh, and A. McCallum, “Energy and Policy Considerations for Deep Learning in NLP,” *arXiv*, June 2019.
- [2] Z. Ding, J. Wang, Y. Song, X. Zheng, G. He, X. Chen, T. Zhang, W.-J. Lee, and J. Song, “Tracking the carbon footprint of global generative artificial intelligence,” *Innovation*, vol. 6, p. 100866, May 2025.
- [3] D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, Z. Jiang, S. Zhong, and X. Hu, “Data-centric Artificial Intelligence: A Survey,” *ACM Comput. Surv.*, vol. 57, pp. 1–42, Jan. 2025.
- [4] X. Guo, Y. Che, Z. Zheng, and J. Sun, “Multi-timescale optimization scheduling of interconnected data centers based on model predictive control,” *Front. Energy*, vol. 18, pp. 28–41, Feb. 2024.
- [5] Z. Hu, B. Li, and J. Luo, “Flutter: Scheduling tasks closer to data across geo-distributed datacenters,” in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pp. 10–14, IEEE.
- [6] S. Sarkar, A. Naug, A. Guillen, V. Gundecha, R. L. Gutiérrez, S. Ghorbanpour, S. Mousavi, A. R. Babu, D. Rengarajan, and C. Bash, “Hierarchical Multi-Agent Framework for Carbon-Efficient Liquid-Cooled Data Center Clusters,” *AAAI*, vol. 39, pp. 29694–29696, Apr. 2025.
- [7] A. James and D. Schien, “A Low Carbon Kubernetes Scheduler,” *ICT for Sustainability*, 2019.
- [8] A. Naug, A. Guillen, R. Luna, V. Gundecha, C. Bash, S. Ghorbanpour, S. Mousavi, A. R. Babu, D. Markovikj, L. D. Kashyap, D. Rengarajan, and S. Sarkar, “SustainDC: benchmarking for sustainable data center control,” in *Guide Proceedings*, vol. 37, pp. 100630–100669, Curran Associates Inc., Dec. 2024.
- [9] S. Sarkar, A. Naug, R. Luna, A. Guillen, V. Gundecha, S. Ghorbanpour, S. Mousavi, D. Markovikj, and A. Ramesh Babu, “Carbon footprint reduction for sustainable data centers in real-time,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 22322–22330, Mar. 2024.
- [10] S. Sarkar, A. Naug, R. Luna Gutierrez, A. Guillen, V. Gundecha, A. Ramesh Babu, and C. Bash, “Real-time carbon footprint minimization in sustainable data centers with reinforcement learning,” in *NeurIPS 2023 Workshop on Tackling Climate Change with Machine Learning*, 2023.
- [11] S. Sarkar, A. Naug, A. Guillen, R. Luna, V. Gundecha, A. Ramesh Babu, and S. Mousavi, “Sustainability of data center digital twins with reinforcement learning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 23832–23834, Mar. 2024.
- [12] S. Sarkar, A. Naug, A. Guillen, R. L. Gutierrez, S. Ghorbanpour, S. Mousavi, A. R. Babu, and V. Gundecha, “Concurrent carbon footprint reduction (c2fr) reinforcement learning approach for sustainable data center digital twin,” in *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, pp. 1–8, 2023.
- [13] X. Guo, D. Keivan, U. Syed, L. Qin, H. Zhang, G. Dullerud, P. Seiler, and B. Hu, “ControlAgent: Automating Control System Design via Novel Integration of LLM Agents and Domain Expertise,” *arXiv*, Oct. 2024.
- [14] R. Zahedifar, S. A. Mirghasemi, M. S. Baghshah, and A. Taheri, “LLM-Agent-Controller: A Universal Multi-Agent Large Language Model System as a Control Engineer,” *arXiv*, May 2025.
- [15] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, “ReAct: Synergizing Reasoning and Acting in Language Models,” *arXiv*, Oct. 2022.
- [16] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, “Policy Distillation,” *arXiv*, Nov. 2015.
- [17] J. Liu, C. Xu, P. Hang, J. Sun, W. Zhan, M. Tomizuka, and M. Ding, “Language-Driven Policy Distillation for Cooperative Driving in Multi-Agent Reinforcement Learning,” *arXiv*, Oct. 2024.

- [18] W. Choi, W. K. Kim, M. Yoo, and H. Woo, “Embodied CoT Distillation From LLM To Off-the-shelf Agents,” *arXiv*, Dec. 2024.
- [19] A. Naug, A. Guillen, V. Kumar, S. Greenwood, W. Brewer, S. Ghorbanpour, A. Ramesh Babu, V. Gundecha, R. L. Gutierrez, and S. Sarkar, “Lc-opt: Benchmarking reinforcement learning and agentic ai for end-to-end liquid cooling optimization in data centers,” *arXiv e-prints*, 2025.
- [20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” *arXiv*, July 2017.
- [21] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “LoRA: Low-Rank Adaptation of Large Language Models,” *arXiv*, June 2021.
- [22] A. Guillen, A. Naug, V. Gundecha, S. Ghorbanpour, R. L. Gutierrez, A. R. Babu, M. Salim, S. Banerjee, E. H. O. Essink, D. Fay, and S. Sarkar, “SustainCluster: Benchmarking Dynamic Multi-Objective Geo-Distributed Workload Management for Sustainable Data Center Cluster,” in *Advances in Neural Information Processing Systems 38 (NeurIPS 2025) Datasets and Benchmarks Track*, 2025. Under Review – Details to be updated upon publication.
- [23] Q. Weng, W. Xiao, Y. Yu, W. Wang, C. Wang, J. He, Y. Li, L. Zhang, W. Lin, and Y. Ding, “MLaaS in the wild: Workload analysis and scheduling in large-scale heterogeneous GPU clusters,” in *19th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 22)*, 2022.

Appendix

A Numerical Expert Implementation Details

This section outlines the formulation and training of the hierarchical Reinforcement Learning (RL) agent, which serves as the numerical expert in our case study. This controller serves as the expert policy oracle for our distillation framework. Its design is representative of state-of-the-art numerical optimizers for this domain, and its inherent limitations (particularly its fixed-size state space leading to a scalability failure) motivate our distillation approach.

A.1 Reinforcement Learning Formulation

We formulate the problem as a cooperative multi-agent task and train our hierarchical agents using a Centralized Training with Decentralized Execution (CTDE) paradigm. The system consists of two distinct shared policies: π_{mgr} for all N Manager agents, and π_{wrk} for all N Worker agents. The hierarchical workflow is visualized in Figure 3.

State Space. The observation vectors for the Manager and Worker policies are constructed from the components detailed in Tables 3 and 4. Critically, the Manager’s state space has a dimension dependent on the number of datacenters (N). This fixed-size numerical input is the root cause of its **scalability failure**, as the policy cannot be applied to a system with a different number of nodes without retraining.

Action Space.

- **Manager Action Space:** Each Manager selects a discrete action $a_t^{mgr,i} \in \{0, \dots, N - 1\}$, corresponding to the destination datacenter for its workload batch.
- **Worker Action Space:** Each Worker selects a discrete binary action $a_t^{wrk,i} \in \{0, 1\}$, corresponding to *Defer* or *Execute* the task at the head of its queue.

Reward Function. As this is a fully cooperative task, all agents are trained to optimize a shared global reward signal r_t . The reward is a composite function designed to encourage the desired multi-objective behavior, balancing a positive incentive for task completion (throughput) against penalties for total carbon emissions, SLA violations, and excessive task queuing time across the entire cluster. The function is defined as:

$$r_t = w_{thru} \cdot \text{Throughput}_t - (w_c \cdot \text{Carbon}_t + w_s \cdot \text{SLA_Violations}_t + w_{age} \cdot \text{Task_Aging}_t) \quad (1)$$

where each term is calculated based on the global system state at timestep t , and the weights (w) are hyperparameters that balance the trade-offs.

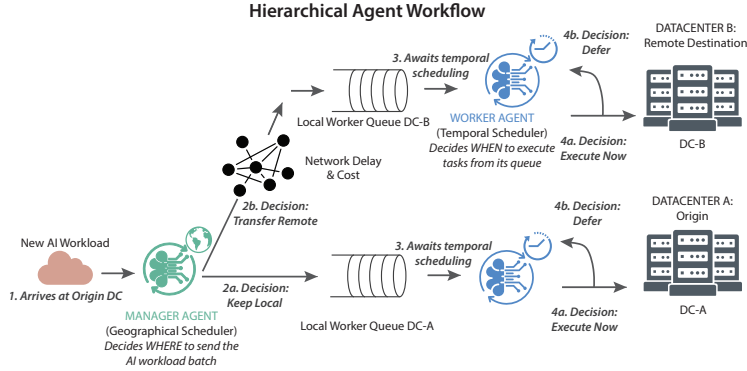


Figure 3: The hierarchical agent workflow of the expert RL policy. The high-level Manager makes a geographical decision, while the low-level Worker makes a temporal decision. This represents the expert’s fixed, two-level control logic.

A.2 Training Hyperparameters

The PPO agents were trained using the Stable Baselines 3 library. The key hyperparameters, selected based on preliminary experiments, are listed in Table 5.

B Distillation Framework Implementation Details

This section provides the specific implementation details for our distillation framework. This includes the design of our novel, language-based control interface and the subsequent fine-tuning of the Large Language Models (LLMs) to use this interface.

B.1 A Language-Based API for Systems Control

A critical step in our framework is the textualization of the numerical state-action data. This process effectively creates a flexible, language-based Application Programming Interface (API) for the datacenter control task. Instead of fixed-size vectors, the controller interacts with a structured text prompt. We designed distinct, structured prompts for the Manager and Worker roles to serve as the two primary "API calls." The exact instruction templates are shown below.

Manager Action Prompt (Geographical Scheduling)
<p>You are an intelligent workload scheduler for a cluster of geo-distributed Data Centers. Based on the aggregated workload originating at each datacenter and the current global status of the cluster, your goal is to decide the optimal destination for each workload to minimize carbon emissions, energy costs, and SLA violations. Your output must be N integer numbers, where N is the number of datacenters. Each integer corresponds to the destination datacenter ID for the workload originating at that location. Prioritize minimizing carbon intensity, but respect resource constraints. Do not explain your answer.</p> <p>### INPUT: [Populated with textualized global state, e.g., "Workload at DC-1: CPU: 250, ...", "Cluster Status: - DC-1: CPU Avail: 80%, CI: 194 g/kWh, ...", etc.]</p> <p>### RESPONSE:</p>
Worker Action Prompt (Temporal Scheduling)
<p>You are an expert temporal scheduler for N local data centers. Based on the current pending task and the local status of each DC, decide whether to execute the task now (1) or defer it (0). Your goal is to execute tasks when local conditions (carbon intensity, electricity price) are most favorable, without violating the task's SLA. Your output must be N integers (0 or 1) for Worker 1, Worker 2, ..., Worker N. Be concise.</p> <p>### INPUT: [Populated with textualized local states, e.g., "-- DC-1 Status --", "Pending Task: Urgency: 0.8, ...", "Local State: CPU Avail: 70%, CI Trend: [-10, ...]", etc.]</p> <p>### RESPONSE:</p>

Table 3: Structure of the flattened observation vector for the Manager agent (π_{mgr}).

Component	Dimension	Description
Meta-Task Features	8	Aggregated features of the newly arrived workload batch.
Global Cluster State	$N \times 5$	Concatenated state of all N datacenters (e.g., CPU/GPU load, CI).
Time Features	4	Sine/cosine encoding of the current timestep.
Total Dimension	12 + 5N	For $N = 3$, the dimension is 27.

Table 4: Structure of the observation vector for the Worker agent (π_{work}).

Component	Dimension	Description
Pending Task Features	7	Features of the task at the head of the local queue.
Local Datacenter State	6	Current status of the agent’s own datacenter.
Carbon Intensity Trend	8	Historical and forecasted changes in local grid carbon intensity.
Temperature Trend	8	Historical and forecasted changes in local ambient temperature.
Time Features	4	Sine/cosine encoding of the current timestep.
Total Dimension	33	A fixed-size vector, as it only uses local information.

Table 5: PPO hyperparameters used for training the expert RL policies.

Hyperparameter	Value
Learning Rate	3e-4
N_steps (Rollout Buffer Size)	2048
Batch Size	64
N_epochs	10
Gamma (Discount Factor)	0.99
GAE Lambda	0.95
Clip Range	0.2
Entropy Coefficient	0.01
Value Function Coefficient	0.5

To enable the "intelligent copilot" capabilities, we used a separate prompt for generating explanations. This demonstrates how the language-based API can be extended for tasks beyond direct control, such as system auditability.

Explanation Generation Prompt
<pre> ### CONTEXT: [The full input state from the previous prompt] ### DECISION MADE: [The action generated by the LLM, e.g., "DC1 routes to DC3, DC2 routes to DC3"] ### INSTRUCTION: Provide a concise, data-driven justification for why this set of scheduling decisions is optimal, considering the primary goal of minimizing carbon emissions while respecting performance constraints. Reference specific quantitative data from the context in your explanation. ### JUSTIFICATION: </pre>

B.2 LLM Fine-Tuning Details

The fine-tuning of the Llama 3.2 and Qwen 3 models was performed using the Hugging Face ‘transformers’ and ‘peft’ libraries. To train the specialized controllers, we used Low-Rank Adaptation (LoRA), a Parameter-Efficient Fine-Tuning (PEFT) strategy. This approach is highly efficient from a systems perspective, requiring only **1 hour of GPU time** to create a new adaptive controller, in stark contrast to the **~48 hours** required to train the numerical expert from scratch. The specific LoRA configuration is detailed in Table 6.

C Expanded Experimental Results

This section provides the full, unabridged results from our experimental evaluation. We first detail the system configurations under test, then present the baseline performance and, finally, the crucial system scalability and behavioral analysis.

Table 6: LoRA hyperparameters used for fine-tuning the LLM controllers.

Hyperparameter	Value
Rank (r)	64
Alpha (α)	16
Target Modules	'q_proj', 'v_proj', 'k_proj', 'o_proj'
Dropout	0.05
Learning Rate	2e-4
Batch Size	4
Number of Epochs	3

C.1 System Configurations for Evaluation

To rigorously test policy replication and system scalability, we defined three distinct cluster configurations. The geographical locations were chosen to provide diverse and realistic environmental signals for the controllers to react to.

- **3-Node System (In-Distribution):** This is the configuration used to train the expert RL policy. It consists of three datacenters located in: **Germany, Chile, and California (USA)**. All controllers were first evaluated in this scenario.
- **5-Node System (Out-of-Distribution):** To test scalability, we created a larger cluster by adding two datacenters with distinct profiles: **Australia and Singapore**.
- **10-Node System (Out-of-Distribution):** To further test scalability and robustness, we used a large, globally distributed cluster of ten datacenters: **Germany, Chile, California (USA), Australia, Singapore, New York (USA), Brazil, South Africa, the United Kingdom, and South Korea**.

The expert RL policy, due to its fixed-size numerical input, can only operate on the 3-node system. The distilled LLM controllers were evaluated on all three system scales.

Figure 4 provides a high-level summary of the sustainability and efficiency gains achieved by the key controllers relative to a non-optimizing baseline. It clearly visualizes the performance progression, from simple heuristics to the expert RL agent, culminating in the distilled LLM controllers which achieve the highest overall improvements across carbon, cost, energy, and water usage.

C.2 Baseline Performance on a 3-Node System

Table 7 presents the complete performance comparison of all evaluated controllers on the 3-node system. This data validates the choice of the hybrid PEFT + Few-Shot model (e.g., '*QwenLoRA+FS*') as the optimal LLM controller presented in the main paper. The results show that while the PEFT-only ('LoRA') models replicate the expert's aggressive sustainability optimization, they also inherit its poor latency performance ('Max Task Delay'). The hybrid ('LoRA+FS') models emerge as superior, maintaining top-tier sustainability performance while learning a more balanced and practical policy with dramatically better latency.

C.3 System Scalability Experiments (5- and 10-Node Systems)

Tables 8 and 9 provide the full performance data for the system scalability experiments. These results confirm that the performance advantage of the distilled LLM controller over the heuristic baseline grows as the number of nodes in the distributed system increases, demonstrating robust and effective scalability.

C.4 Analysis of System-Level Behavior

To provide deeper insight into *how* the learned LLM policy differs from a simple heuristic, we visualize their behavior over a 7-day period and the dynamic environmental conditions that the controllers must react to. Figure 5 shows the significant variance in real-time grid carbon intensity

Sustainability & Efficiency Improvement vs. Baseline RBC

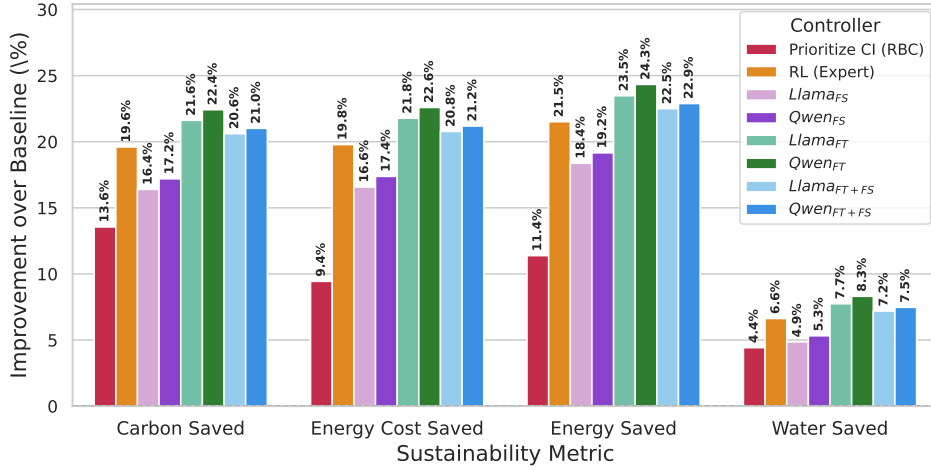


Figure 4: Percentage improvement in sustainability and efficiency metrics for key controllers, relative to the ‘Baseline (RBC)’. Positive values indicate better performance. The fine-tuned LLM controllers achieve the highest reductions across all four metrics, demonstrating the success of the distillation process.

Table 7: Full performance comparison on the 3-node (in-distribution) system. Results are reported as an average per day over a 31-day simulation. For all metrics, a lower value is better.

Controller	Carbon (kg CO ₂) ↓	Energy Cost (\$) ↓	Energy Used (kWh) ↓	Water Used (m ³) ↓	SLA Violated (%) ↓	Max Task Delay (minutes) ↓
Baseline (RBC)	28,499	11,187	140,386	780.0	0.74	95
Prioritize Temperature (RBC)	25,012	9,649	118,485	710.0	1.30	105
Prioritize CI (RBC)	24,637	10,131	124,409	745.5	1.33	112
RL (Expert)	22,912	8,974	110,186	728.3	1.56	1095
Llama _{FS}	23,828	9,333	114,593	742.1	1.48	923
Qwen _{FS}	23,599	9,243	113,492	738.5	1.45	887
Llama _{LoRA}	22,339	8,751	107,431	719.6	1.59	1115
Qwen _{LoRA}	22,110	8,660	106,227	715.2	1.62	1149
Llama _{LoRA} +FS	22,626	8,863	108,809	723.9	1.18	381
Qwen _{LoRA} +FS	22,511	8,817	108,256	721.7	1.15	355

Table 8: Full performance on the 5-node (out-of-distribution) system.

Controller	Carbon (kg CO ₂) ↓	Energy Cost (\$) ↓	SLA Violated (%) ↓	Max Delay (min) ↓
Lowest CI (RBC)	62,777	18,036	1.48	554
Llama _{LoRA} +FS	59,871	17,070	1.11	410
Qwen _{LoRA} +FS	59,558	16,905	1.08	399

Table 9: Full performance on the 10-node (out-of-distribution) system.

Controller	Carbon (kg CO ₂) ↓	Energy Cost (\$) ↓	SLA Violated (%) ↓	Max Delay (min) ↓
Lowest CI (RBC)	131,830	38,778	1.95	727
Llama _{LoRA} +FS	126,557	36,081	1.24	454
Qwen _{LoRA} +FS	125,141	35,540	1.19	424

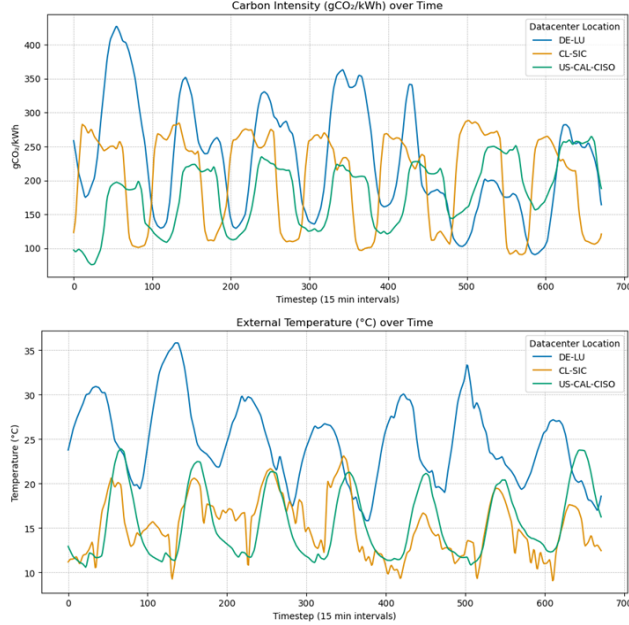


Figure 5: Time-series of the primary environmental drivers for the 3-datacenter evaluation scenario: (Top) Grid Carbon Intensity and (Bottom) External Ambient Temperature. These dynamic signals create the optimization opportunities for the controllers.

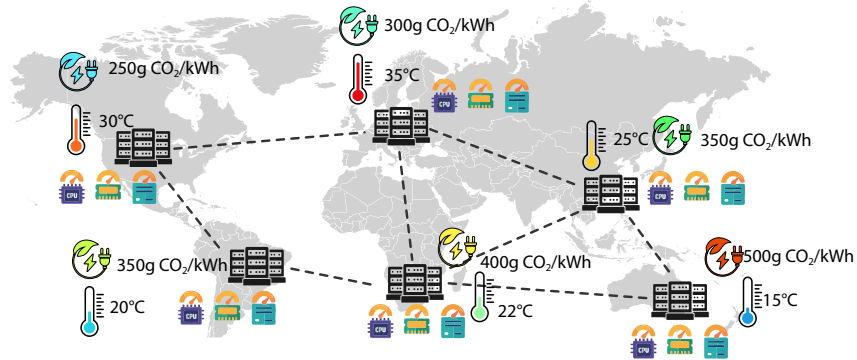


Figure 6: Overview of the geo-distributed datacenter cluster locations simulated, showing varying carbon intensity and other real-world factors.

and ambient temperature across the three datacenter locations, creating clear opportunities for intelligent geographical and temporal scheduling.

Figure 6 shows the geographical distribution of the datacenters in our simulation. The time-series plots reveal that the ‘Lowest CI’ RBC (Figure 7) learns a greedy, "bang-bang" policy, overloading whichever single datacenter is greenest at any moment, leading to highly imbalanced resource utilization. In contrast, the distilled LLM controller (Figure 8) learns a more nuanced and balanced strategy, distributing the load more evenly while still prioritizing low-carbon sites. Figure 9 further illustrates the LLM’s sophisticated temporal scheduling, showing how it strategically builds and drains its local task queues to align execution with periods of favorable environmental conditions.

C.5 Implementation Details

All experiments were conducted on a server equipped with an Intel(R) Xeon(R) Platinum 8470 CPU and two NVIDIA H100 PCIe 80GB GPUs. The software stack was built on Python 3.10 and PyTorch 2.1.

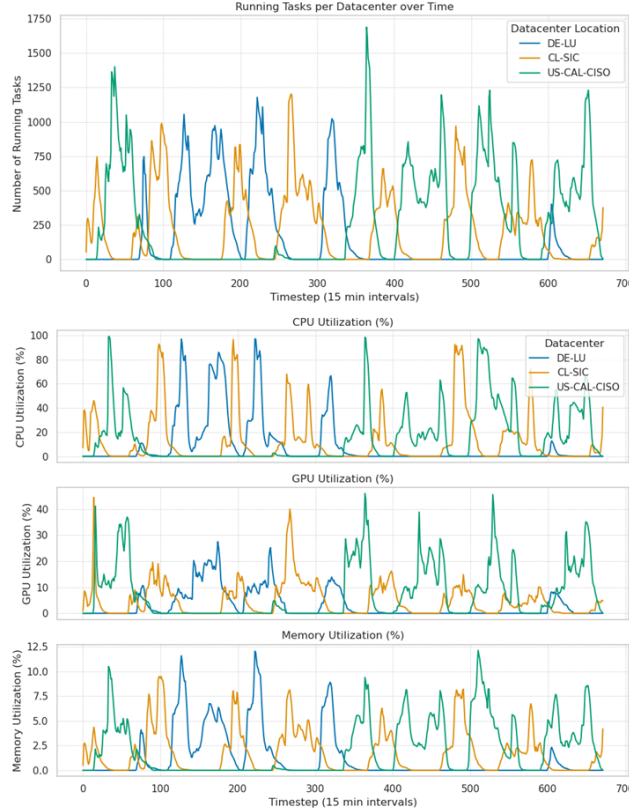


Figure 7: Behavior of the ‘Lowest CI’ Rule-Based Controller. The greedy policy concentrates all tasks in the single greenest datacenter, leading to highly skewed and inefficient resource utilization.

RL Expert Training. The hierarchical PPO agents (π_{mgr} and π_{wrk}) were trained separately using the Stable Baselines 3 library. Each policy was trained for a total of 20 million timesteps, which took approximately 48 hours to complete on a single H100 GPU.

LLM Fine-Tuning. The fine-tuning of the Llama 3.2 8B and Qwen 2 7B models was performed using the Hugging Face Transformers and PEFT libraries. The LoRA fine-tuning process for each LLM controller on the expert-generated dataset was significantly faster than the RL training, completing in approximately one hour per model on a single H100 GPU.

C.6 Runtime Resource Footprint and Latency

A critical aspect of any deployable system is its operational cost. To provide a transparent analysis of the trade-offs introduced by our framework, we measured the systems-level costs of our LLM agent compared to the original numerical expert. The results are summarized in Table 10.

The data shows that the LLM agent has a significantly larger resource footprint and higher inference latency. However, these increased costs represent a deliberate and highly justified engineering trade-off. The resource requirements are well within the capabilities of a single, modern server-class GPU, and for a system with a 15-minute decision cadence (900,000 ms), the sub-second latency of the LLM is operationally negligible. Therefore, the modest, one-time hardware cost is a highly favorable trade-off for gaining the transformative capabilities of runtime adaptability, scalability, and explainability—benefits that are impossible to achieve with the original expert.

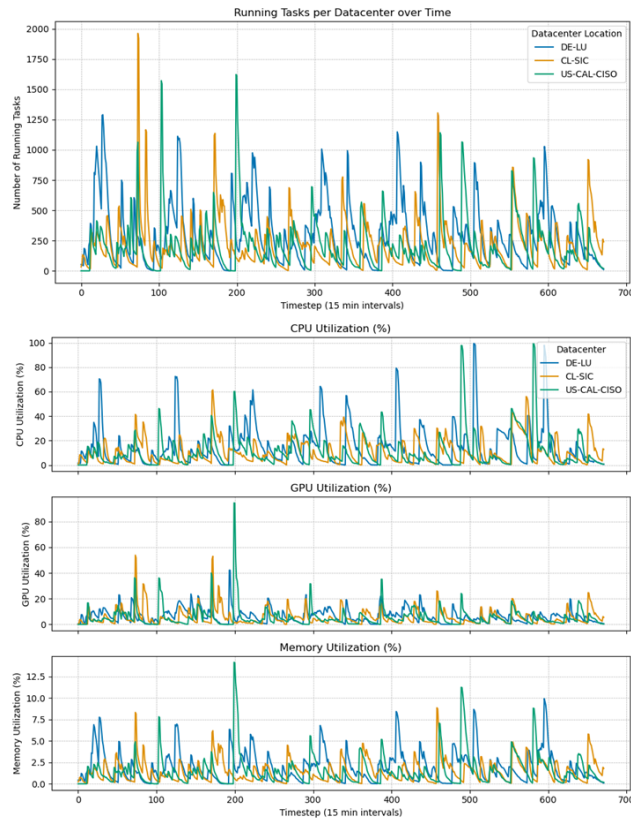


Figure 8: Behavior of the fine-tuned LLM Controller. It learns a more sophisticated, balanced policy, distributing the load more smoothly across the cluster for more efficient resource utilization.

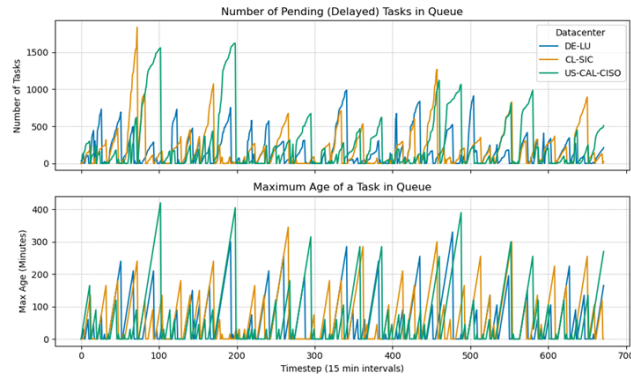


Figure 9: Analysis of the LLM controller’s temporal scheduling. The plots show the number of pending tasks and the maximum task age in each queue, demonstrating how the Worker agent strategically defers tasks to wait for better conditions.

Table 10: System Cost Analysis: A comparison of the runtime resource and latency costs for the deployed controller. The LLM’s higher footprint is a favorable trade-off for its advanced capabilities.

Metric	RL Expert (PPO)	LLM Agent (Qwen 3 8B)
Model Size (on disk)	~15 MB	~16 GB (BF16)
Inference Latency (per decision)	~20 ms	~250 ms
Required GPU VRAM (for inference)	< 1 GB	~18 GB

Table 11: Summary of the qualitative benchmark used to evaluate the advanced capabilities of the distilled LLM copilot, framed as solutions to core systems-level challenges.

Systems Capability Tested	Core Question Answered by the Benchmark
Conceptual Understanding	Does the copilot understand the fundamental principles and trade-offs of the system it is controlling, or is it just pattern-matching numerical inputs?
System Auditability (Justification)	Can the copilot ground its decisions in specific, quantitative system state data, providing a logical and auditable justification for its control actions?
Predictive Planning (Counterfactuals)	Can the copilot reason about hypothetical "what-if" scenarios, demonstrating a deeper understanding of cause-and-effect for system planning and resilience testing?
Runtime Adaptability (New Goals)	Can the copilot dynamically adapt its entire control strategy on-the-fly in response to new, high-level operator goals or hard constraints issued in natural language?

C.7 Reproducibility

To facilitate further research and ensure the reproducibility of our results, the code for the distillation framework, the fine-tuned LLM controllers, and the complete dataset of textualized state-action trajectories could be made publicly available upon publication of this paper.

C.8 Limitations and Future Work

Our evaluation is conducted in a high-fidelity simulation, which, while realistic, does not capture all the complexities of a live production environment. Future work will focus on deploying and validating our LLM agent on a physical hardware testbed to assess its robustness and real-world performance. Furthermore, we plan to investigate more formal methods for measuring the policy fidelity between the numerical expert and the distilled LLM agent, providing deeper insights into the knowledge transfer process.

D The LLM as an Intelligent Copilot for Systems Operators

The main paper claims that our distillation framework transforms the numerical expert into an "intelligent copilot." This section provides the detailed qualitative evidence for this claim, demonstrating the capabilities that are essential for the real-world deployment of automated systems. We designed a comprehensive benchmark to probe the agentic workflow’s capabilities across four key dimensions of system manageability and runtime adaptability, summarized in Table 11.

D.1 System Auditability: Justifying Actions with Real-Time Data

A critical capability for any deployable automated system is the ability to justify its actions, enabling operator trust. The example below shows the copilot reasoning through a complex trade-off, proving it is an auditable agent.

Justification Involving a System Trade-Off

SITUATION: The controller must schedule a large workload (requiring 800 CPU cores). Node 3 has the lowest carbon intensity (30 g/kWh), but Node 2 is chosen instead.

OPERATOR QUERY: "Why did you route the workload to Node 2 (90 g/kWh) instead of the much cleaner Node 3?"

LLM EXPLANATION: "This decision was a necessary trade-off between sustainability and system feasibility. While Node 3 is the greenest option, its **600 available CPU cores** were insufficient for the 800-core workload. Node 2 was the best available option that met the hard resource constraint while still offering a significant carbon reduction compared to executing locally."

D.2 Runtime Adaptability and Predictive Planning

The most transformative capabilities are the copilot's ability to adapt to new operator goals and to reason about hypothetical scenarios for planning purposes. The following examples showcase this flexibility, which is essential for building resilient, human-in-the-loop systems.

Example 1: Predictive Planning via Counterfactual Reasoning

OPERATOR QUERY: "What would be the optimal routing plan if a network issue suddenly doubled the data transfer latency to Node 3?"

LLM 'WHAT-IF' ANALYSIS: "Analyzing hypothetical scenario: latency to Node 3 is doubled. For time-sensitive workloads, routing to Node 3 would no longer be optimal, as the increased latency creates a high risk of SLA violation. The new optimal choice for such tasks would be immediate local execution or routing to the next-best destination with lower latency. The control policy would shift to prioritize performance over sustainability for all urgent tasks."

Example 2: Adapting to a New System-Wide Constraint

OPERATOR COMMAND: "New data residency regulations now forbid workloads originating at our California node (Node 1) from being processed outside of North America."

LLM STRATEGY SHIFT: "Acknowledged. Data residency constraint for Node 1 is now active. For any workload originating at Node 1, all non-North American nodes (e.g., Germany) are now invalid destinations and will be removed from the action space. The routing decision will now be optimized among the remaining compliant nodes only."