Learning on the GPUs the World Owns: A Fleet Contract for Learned Systems on Commodity Hardware

Hannah B. Pasandi
University of California, Berkeley
Berkeley, CA, USA
h.pasandi@berkeley.edu

Mohammad Hosseini
Shahid Beheshti University
Tehran, Iran

Sina Darabi
Barcelona Supercomputing Center
Barcelona, Spain
sinad1367@gmail.com

Abstract

Learned schedulers, admission policies, and placement models are following training and serving workloads onto the GPUs people already own: volunteer swarms, marketplace fleets, and single-box agents built from RTX-class parts. What blocks trusting them there is not model quality but a hardware assumption gap. The best documented production run sets the floor. Llama 3 training on 16,384 ECC-protected H100s still took an unexpected interruption roughly every three hours and caught six silent corruptions. Consumer fleets sit below that floor on every clause: throttling that moves feature distributions, non-ECC memory, deep heterogeneity, and nobody on call. Four stated-model experiments price the consequences, including an interaction the community has not priced: int4 quantization multiplies the damage of a single bit flip 4.4× over fp16, precisely on the fleets that can least afford it. We propose FleetContract, a three-clause admission contract: a measured capability report, a per-feature stability envelope with refusal, and a mandatory stratified canary. Simulators and reference contract checks: https://github.com/hanabhp/fleet-contrast.

CCS Concepts: • **Computer systems organization → Reliability**.

Keywords: consumer GPUs, learned systems, reliability, deployment, silent data corruption, ML for systems

ACM Reference Format:
Hannah B. Pasandi, Mohammad Hosseini, and Sina Darabi. 2026. Learning on the GPUs the World Owns: A Fleet Contract for Learned Systems on Commodity Hardware. In *Practical Adoption Challenges of ML for Systems (PACMI '26), September 29–October 2, 2026, Prague, Czech Republic.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3843967.3844681

PACMI '26, Prague, Czech Republic

ACM ISBN 979-8-4007-2998-0/26/09
https://doi.org/10.1145/3843967.3844681

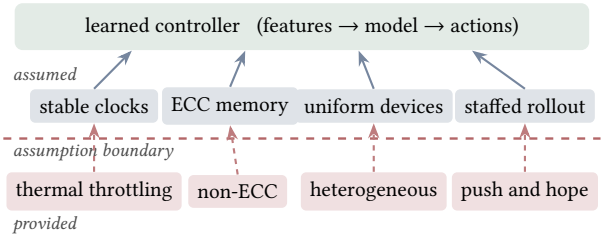

Figure 1. Four assumptions the datacenter quietly provided; the consumer fleet breaks all four, and the controller cannot notice.

1 Introduction

Meta's Llama 3 report is the cleanest ledger we have of what large-scale training costs in reliability: 54 days on 16,384 H100s, 466 job interruptions, 419 of them unexpected, roughly one every three hours, 78% traced to hardware, and six silent corruptions caught only because every part had ECC [17]. That is the pampered regime; now move the workload where it is heading. Petals serves a large model across donated GeForce cards on home cooling and home power [3], and Han's AgenticOS keynote framed the direction plainly: run training and inference on the GPUs people already own [8]. Schedulers, placement models, and admission policies [14, 18, 20], built for the first regime, now aim at the second. Nobody has written down what they lose in the move.

What breaks is not the model. It is the platform contract underneath it. A learned controller trained on datacenter telemetry inherits four assumptions it never states: feature distributions hold because cooling holds clocks, computation is correct because ECC scrubs it, devices are interchangeable because procurement binned them, and rollout is safe because someone staffed canaries and dashboards. Figure 1 names the four. A consumer fleet voids every one,[1] and because the assumptions were never written down, the controller cannot notice when they go.

Why is this not already solved by monitoring, robust training, or an ops playbook? Monitoring observes a controller after admission; every failure in this paper happens silently inside an admitted controller, so the watch starts one step

[1]Workstation RTX parts (A and Ada series) can switch on ECC; the GeForce cards that fill volunteer pools and GPU marketplaces cannot.

too late. Robust training hardens a model against noise it has seen; it cannot decide whether one specific 3060 with one specific thermal history should run the model at all. And the rollout discipline that decided adoption in prior deployments [9, 11] assumes an operations team these fleets have none of. The two literatures that could meet here do not: democratization optimizes throughput and cost on commodity parts, reliability work documents the datacenter, and neither writes the contract a learned controller needs on the first, built from the lessons of the second.

Our position fits in one sentence: *a learned controller on a commodity fleet must earn admission per device against a measured contract, and must refuse loudly when the measurement fails.* The lever that makes this cheap already exists: every clause the datacenter provided implicitly is checkable explicitly, with a short benchmark, an interval test, and a sampling rule, each a few lines of runtime logic below the controller.

- We name the implicit platform contract of deployed learned systems and show that a consumer fleet voids all four clauses at once (§2).
- We quantify the gap with four stated-model experiments: throttling drives a predictor from 2.4% to 21.7% error; encoding separates a 9-point reward loss from a 58-point collapse under identical bit flips; a 1% canary admits up to 13.6% fleet harm where a 5% stratified slice stays under 2.9%; int4's shared scales raise per-parameter corruption exposure 24% over fp16 (§4).
- We specify FleetContract, a measured capability report, a per-feature stability envelope that refuses to a fallback heuristic, and a mandatory stratified canary, and we argue why these three clauses and not others (§5); we release the simulators and reference checks, and state the on-fleet predictions that would falsify each experiment (§7).

Broader impact. The contract decides who gets protected when democratized AI meets the hardware people can actually buy. Emerging-region services run int4 on the oldest cards because that is where unit economics close [2], and the energy and carbon of a silently wrong rollout land on whichever fleet serves it, a bill test-time compute is already inflating [22]. Refusal semantics are consumer protection here: a device owner learns their card was refused and why, instead of serving degraded answers into someone else's product. The risk cuts the other way too: gatekeeping that excludes the hardware democratization meant to reach, which is why envelope width is a policy knob, not a fixed number.

Scope and boundary conditions. Every number here comes from a simulation with a stated model; we claim mechanisms and orderings, not fleet forecasts or deployment, and §7 lists the on-hardware predictions that would falsify each experiment. The contract targets the platform assumptions of learned controllers, not an adversarial device owner, energy budgets, or weight hardening, and §5 says why those stay out. Two boundary cases: on a workstation fleet with ECC enabled, clause 2 still pays because throttling and heterogeneity remain, and on a single box the canary clause is vacuous while the report and envelope still apply, so the contract degrades gracefully to N=1.

Table 1. The implicit platform contract, and its status on a consumer fleet.

Assumption	Datacenter	Consumer fleet
Feature distributions stationary	Cooling holds clocks; drift is slow [16]	Throttling moves features within minutes
Computation is correct	ECC; SDC hunted at scale [4, 10]	Non-ECC; flips land in weights and features
Devices interchangeable	Homogeneous SKUs, binned	VRAM, thermals, drivers all vary
Rollout is staffed	Canaries, dashboards, on-call [11]	Push and hope

2 What Learned Systems Assume

The deployed ML-for-systems literature, read as a platform specification, yields Table 1; the assumptions are rarely written down, they arrive with the training data. Each is individually reasonable in a datacenter and individually false on a consumer fleet, and the failures compound: a drifted feature on a corrupted model, rolled out with no canary, is exactly the silent, unattributable degradation PACMI retrospectives keep reporting [16, 19].

Is the regime real or extrapolated? Both, and it pays to say which is which. Real today: Petals routes requests across a public, heterogeneous swarm of donated cards [3]; training across multi-cluster consumer GPUs is a SIGCOMM result, not a forecast [15]; and the agentic stack is placing schedulers and resource controllers on single consumer machines by design [26, 27]. Extrapolated: fleets of tens of thousands of such devices running learned admission end to end. Most academic consumer-GPU setups today are a dozen cards in a lab, a fair objection, and our answer is that the contract prices per device, not per fleet: the report and envelope pay at N=12 exactly as at N=10,000, and the canary clause switches on only when the fleet does.

3 The Floor Is Datacenter-Shaped

The strongest argument that the consumer fleet is a different regime is how hard the datacenter regime already is, on the numbers already on the table from §1. Kokolis *et al.* generalize the picture across 150 million A100 GPU-hours: failure rates that make time between failures a first-order design constraint, plus lemon nodes that fail far above the median [13]. MegaScale needed deep observability just to attribute stragglers at 10,000-GPU scale [12], and even then clouds validate components proactively, because gray failures pass health checks and surface as user-visible degradation [25]. Hold the picture: ECC memory, one SKU, redundant cooling, a staffed control room, still an interruption every three hours, a validation layer in front of every job, every mitigation the

consumer fleet lacks. The floor is not a baseline the fleet starts from; it is a ceiling the fleet cannot reach.

4 The Gap, Measured

Four stated-model experiments quantify what the broken clauses do. Each simulation is transparent so the mechanism is inspectable; none claims fleet realism, §7 lists what would falsify each on real hardware, and the scripts behind every plotted point ship with the paper.[2]

E1: throttling breaks the feature contract. Throughput and latency predictors are the input every deployed learned scheduler shares, from device placement [20] to cluster scheduling [18], so we stress that input first: we train a predictor on features collected at nominal clocks, then evaluate as thermal throttling deepens, under a linear feature-to-clock model with measured GDDR6 sensitivities.[3] Figure 2a shows error growing from 2.4% at nominal to 21.7% at 40% depth; the controller never fails loudly, it just becomes confidently wrong. A ±8% per-feature stability envelope, clause 2 below, admits the device only to about 15% depth, where error still sits at 7.0%. What breaks is the input distribution; the envelope converts that into refusal instead of silent error.

E2: what corruption breaks depends on the feature encoding, not the flip rate. We inject uniform bit flips at rate λ into the state of two controllers that share a policy and differ only in what they read: a bandit scheduler on normalized, bounded features, and the same policy on thresholded raw counters. Figure 2b sweeps λ from 10^{-5} to 10^{-2} per parameter-step. The bounded variant slides from 97.4% to 88.5% reward because clipping absorbs outliers; the thresholded variant collapses to 38.9%, since one flipped register crosses a threshold and latches a wrong branch. Fragility lives in the encoding, checkable before deployment. Uniform flips are the simplest member of the fault family; real corruption clusters by row and bank and tracks the workload [4, 10]. The claim survives because it is comparative: a correlated burst still latches one wrong branch in the thresholded register, while bounded features average it away, so clustering should widen the gap (a §7 prediction).

E3: canaries do not miniaturize. With device-type failure probability φ, a canary catches a failure only if it contains the failing type. We model a 10,000-device fleet with realistic type skew and measure fleet harm, the traffic share a bad model serves before detection; shipping to everyone scales harm linearly with φ. We first sized the slice at 1% because that is what we would have shipped ourselves; watching it miss the failing family in 93 of 100 draws at moderate φ is what pushed us to a stratified floor. Figure 2c shows the result: the 1% slice admits harm up to 13.6%, while a 5% stratified slice, sized so every device type appears, bounds harm under 2.9% everywhere we sweep. The datacenter never met this failure; its canaries sampled a homogeneous population.

E4: model choice and device choice interact. The experiments above hold the model fixed; the fleet does not get that luxury. Small VRAM forces a choice between quantizing and offloading past device memory [7, 24], and here is the surprising bit: each looks benign alone, and together they are not, in two measurable ways.

Fit decides admission. Let τ be model footprint over device VRAM. Our timing model keeps per-token latency stable while the model is resident ($\tau \leq 0.85$) and adds a host-transfer term, amplified by thermal jitter, once it spills. Figure 3a sweeps admitted fraction against τ at three throttle depths. The moment the model spills, admission collapses, because offloading routes every token through the host path where thermal variance is largest; at 30% depth nothing is admitted at any τ, consistent with E1. On this fleet, a model that fits is an admission decision, not a performance preference.

Format decides blast radius. Quantization shrinks the target a flip can hit, but block formats share scale factors, so one flip in a scale corrupts every weight in its block. Under a uniform-flip model, the expected weights corrupted per flip are 1.0 for fp16, 3.0 for int8 at group size 128, and 4.4 for int4 at group size 32 (Figure 3b). Fewer resident bits do mean fewer flips: int4 holds 28% of fp16's bits per parameter. Multiply through and per-parameter exposure still rises 24% moving from fp16 to int4. Each factor is modest; the product is the point. The fleets that most need aggressive quantization, the ones without ECC, are exactly where its blast radius costs most, and no per-format guardrail exists today.

5 The Fleet Contract

The experiments share a shape: each failure was invisible to the controller and cheap to detect one layer below it. FleetContract is that layer, three clauses evaluated per device (Figure 4).

Clause 1: a measured capability report. Before a device runs a learned controller, it runs a short benchmark: sustained clocks under load, memory error screening, transfer bandwidth, latency spread, SuperBench's proactive-validation lesson [25] carried to hardware that never sees a cloud health checker. The report, not the spec sheet, is the device's identity.

Clause 2: a stability envelope with refusal. Every feature declares a validity interval around its training distribution; a device outside it is refused and falls back to the non-learned heuristic, turning the drift PACMI has documented in production [1, 16] into an explicit, loggable event, as E1 and E4a show.

Clause 3: a canary slice that cannot be skipped. No model version reaches the fleet without a stratified slice, sized

[2] https://github.com/hanabhp/fleet-contrast, MIT license; the experiments run on stock Python with no dependencies.

[3] Throttle depth is the fraction of nominal sustained clock lost under load: at 40% depth, a card rated for 2.5 GHz holds 1.5 GHz.

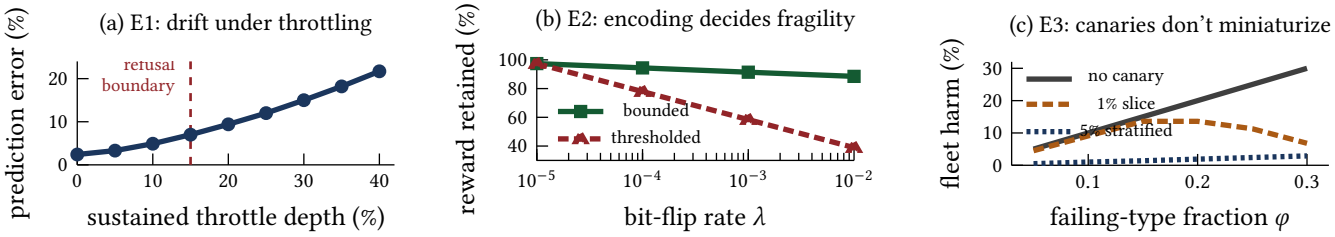

Figure 2. Three broken clauses, in stated-model simulations.

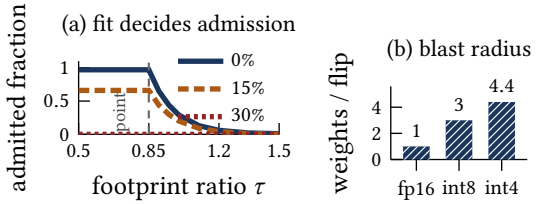

Figure 3. E4: model and hardware interact.

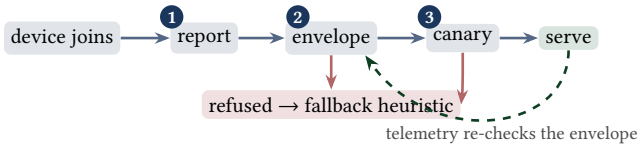

Figure 4. FleetContract: admission is earned per device (1, 2) and per version (3); refusal falls back to the existing heuristic.

so every device type is represented; E3 puts the floor near 5% for realistic skew, importing the rollout discipline from prior PACMI deployments [11] into fleets with no rollout staff, a property of the update path, not a team.

Why these three clauses and not others. Each clause answers one broken row of Table 1: the report answers heterogeneity and screens correctness, the envelope answers stationarity, the canary answers rollout, heterogeneity covered twice on purpose. The contract converts the four silent failures of §4 into loggable refusals; it does not defend against an adversarial device owner, budget energy or thermals, or harden the model itself, complements that belong to other layers, not substitutes for this one [27]. Clauses we cut: a driver pin (inside the report), an accuracy SLO (policy above the contract), continuous retraining (treats drift as data, never safe by default).

The contract is deliberately boring, three checks, a few lines of runtime logic, each producing an auditable record; what changes is the default. Today a learned controller assumes admission and fails silently. Under FleetContract it earns admission and refuses loudly.

6 Who Bears the Mismatch

Position papers should say who is exposed, because the bearers are rarely the people who publish papers; Table 2 does. The pattern is uncomfortable: fleets with the least slack are pushed hardest toward the exposed configurations, aggressive quantization, offloading, aged devices, and are least able to detect or absorb the failure. The contract does not make their hardware better; it makes the exposure visible before users do, with a defined refusal path instead of a silent one.

Table 2. Who bears the mismatch.

Deployment	Forced choice	Dominant exposure	Cl.
Volunteer pool [3]	int4 + offload on mixed cards	blast radius (E4b), hot-node drift (E1)	1+2
Garage training pod [15]	no ECC, consumer parts	silent corruption (E2)	1
Emerging-region service [2]	oldest cards, int4, thin margins	admission collapse (E4a), no rollout staff	2+3
On-prem agent box [26]	resident SLM, tight latency	drift breaks latency model (E1)	2

7 Discussion

Doesn't refusal cost coverage? Yes, and that is the policy knob: the key tension is coverage against admitted error, and E1 makes the trade explicit rather than leaving it implicit at width infinity.

What would falsify this. On real fleets: predictor error should show a knee at documented throttle depths (E1); incident rates in non-ECC deployments should track weight format at fixed model quality (E4b); clustered faults should separate encodings at least as widely as uniform flips do (E2); and fleets canarying below the stratification floor should hear about failures from users, not telemetry (E3). Any of these failing to appear would weaken the position.

8 Related Work

Democratization [3, 5, 7, 15, 23, 24] and *datacenter reliability* [4, 10, 12, 13, 17, 25] are the two stacks §3 bridges. *PACMI* [1, 6, 9, 11, 16, 19] and *learned systems* [14, 18, 20–22, 26, 27] complete the picture.

9 Conclusion

We introduced FleetContract, a three-clause admission contract, a measured capability report, a stability envelope with refusal, and a stratified canary, that lets a learned controller earn admission on a commodity fleet rather than assume it. We showed that throttling, non-ECC memory, heterogeneity, and unstaffed rollout each silently break deployed controllers, and that quantization and device choice interact in ways the field has not priced. The contract turns those silent failures into loggable refusals.

References

[1] Navidreza Asadi, Dalal Ali, Razvan-Mihai Ursu, and Wolfgang Kellerer. 2025. Challenges in Designing Robust RL-Based Autoscalers. In *Proceedings of the 4th Workshop on Practical Adoption Challenges of ML for Systems (PACMI '25)*. Association for Computing Machinery, New York, NY, USA.

[2] Rohail Asim, Ankit Bhardwaj, Arjuna Sathiaseelan, Yasir Zaki, and Lakshmi Subramanian. 2026. Modeling Economic Viability for Scalable AI Deployment in Emerging Regions. In *1st Workshop on Agentic Operating Systems (AgenticOS '26)*.

[3] Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Max Ryabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, and Colin Raffel. 2023. Petals: Collaborative Inference and Fine-tuning of Large Models. *arXiv preprint arXiv:2209.01188* (2023). doi:10.48550/arXiv.2209.01188

[4] Harish Dattatraya Dixit, Sneha Pendharkar, Matt Beadon, Chris Mason, Tejasvi Chakravarthy, Bharath Muthiah, and Sriram Sankar. 2021. Silent Data Corruptions at Scale. *arXiv preprint arXiv:2102.11245* (2021). doi:10.48550/arXiv.2102.11245

[5] Arthur Douillard, Qixuan Feng, Andrei A. Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. 2023. DiLoCo: Distributed Low-Communication Training of Language Models. *arXiv preprint arXiv:2311.08105* (2023). doi:10.48550/arXiv.2311.08105

[6] Abduarraheem Elfandi, Hannah Sagalyn, Ramakrishan Durairajan, and Walter Willinger. 2024. Bootstrapping Trust in ML4Nets Solutions with Hybrid Explainability. In *Proceedings of the 3rd Workshop on Practical Adoption Challenges of ML for Systems (PACMI '24)*. Association for Computing Machinery, New York, NY, USA, 1–5. doi:10.1145/3704742.3704961

[7] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. In *Proceedings of the 11th International Conference on Learning Representations (ICLR '23)*.

[8] Dongsu Han. 2026. Democratizing Deep Learning: Making LLM Training and Inference Accessible with Consumer-grade GPUs. Keynote, 1st Workshop on Agentic Operating Systems (AgenticOS '26).

[9] Maximilian Jakob Heer, Benjamin Ramhorst, and Gustavo Alonso. 2025. Easing the Path to Deployment in ML4Sys through FPGAs. In *Proceedings of the 4th Workshop on Practical Adoption Challenges of ML for Systems (PACMI '25)*. Association for Computing Machinery, New York, NY, USA.

[10] Peter H. Hochschild, Paul Turner, Jeffrey C. Mogul, Rama Govindaraju, Parthasarathy Ranganathan, David E. Culler, and Amin Vahdat. 2021. Cores That Don't Count. In *Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS '21)*. Association for Computing Machinery, New York, NY, USA, 9–16. doi:10.1145/3458336.3465297

[11] Benjamin Hoffman, Alexander Dietmüller, Ayush Mishra, and Laurent Vanbever. 2025. Into the Wild: Real-World Testing for ML-Based ABR. In *Proceedings of the 4th Workshop on Practical Adoption Challenges of ML for Systems (PACMI '25)*. Association for Computing Machinery, New York, NY, USA.

[12] Ziheng Jiang, Haibin Lin, Yinmin Zhong, Qi Huang, Yangrui Chen, Zhi Zhang, Yanghua Peng, Xiang Li, Cong Xie, Shibiao Nong, Yulu Jia, Sun He, Hongmin Chen, Zhihao Bai, Qi Hou, Shipeng Yan, Ding Zhou, Yiyao Sheng, Zhuo Jiang, Haohan Xu, Haoran Wei, Zhang Zhang, Pengfei Nie, Leqi Zou, Sida Zhao, Liang Xiang, Zherui Liu, Zhe Li, Xiaoying Jia, Jianxi Ye, Xin Jin, and Xin Liu. 2024. MegaScale: Scaling Large Language Model Training to More Than 10,000 GPUs. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI '24)*. USENIX Association, 745–760.

[13] Apostolos Kokolis, Michael Kuchnik, John Hoffman, Adithya Kumar, Parth Malani, Faye Ma, Zachary DeVito, Shubho Sengupta, Kalyan Saladi, and Carole-Jean Wu. 2025. Revisiting Reliability in Large-Scale Machine Learning Research Clusters. In *IEEE International Symposium on High-Performance Computer Architecture (HPCA '25)*. IEEE.

[14] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018. The Case for Learned Index Structures. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18)*. Association for Computing Machinery, New York, NY, USA, 489–504. doi:10.1145/3183713.3196909

[15] Hwijoon Lim, Juncheol Ye, Sangeetha Abdu Jyothi, and Dongsu Han. 2024. Accelerating Model Training in Multi-cluster Environments with Consumer-grade GPUs. In *Proceedings of the ACM SIGCOMM 2024 Conference*. Association for Computing Machinery, New York, NY, USA, 707–720. doi:10.1145/3651890.3672228

[16] Shinan Liu, Francesco Bronzino, Paul Schmitt, Arjun Nitin Bhagoji, Nick Feamster, Hector Garcia Crespo, Timothy Coyle, and Brian Ward. 2022. Understanding Model Drift in a Large Cellular Network. In *Workshop on Practical Adoption Challenges of ML for Systems in Industry (PACMI '22), co-located with MLSys 2022*.

[17] Llama Team, AI @ Meta. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783* (2024). doi:10.48550/arXiv.2407.21783

[18] Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. 2019. Learning Scheduling Algorithms for Data Processing Clusters. In *Proceedings of the ACM SIGCOMM 2019 Conference*. Association for Computing Machinery, New York, NY, USA, 270–288. doi:10.1145/3341302.3342080

[19] Shohei Matsuura and Takashi Miyazaki. 2022. Real-World Challenges of ML-Based Database Auto-Tuning. In *Workshop on Practical Adoption Challenges of ML for Systems in Industry (PACMI '22), co-located with MLSys 2022*.

[20] Azalia Mirhoseini, Hieu Pham, Quoc V. Le, Benoit Steiner, Rasmus Larsen, Yuefeng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, and Jeff Dean. 2017. Device Placement Optimization with Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML '17)*. PMLR, 2430–2439.

[21] Melissa Z. Pan, Negar Arabzadeh, Riccardo Cogo, Yuxuan Zhu, Alexander Xiong, Lakshya A. Agrawal, Huanzhi Mao, Emma Shen, Sid Pallerla, Liana Patel, Shu Liu, Tianneng Shi, Xiaoyuan Liu, Jared Quincy Davis, Emmanuele Lacavalla, Alessandro Basile, Shuyi Yang, Paul Castro, Daniel Kang, Koushik Sen, Dawn Song, Joseph E. Gonzalez, Ion Stoica, Matei Zaharia, and Marquita Ellis. 2025. Measuring Agents in Production. *arXiv preprint arXiv:2512.04123* (2025). doi:10.48550/arXiv.2512.04123

[22] Hannaneh B. Pasandi and Tamer Nadeem. 2026. The Reasoning Tax: Profiling Test-Time Compute Carbon in Agentic AI Workloads. *ACM SIGENERGY Energy Informatics Review* 6, 2 (June 2026).

[23] Max Ryabinin, Tim Dettmers, Michael Diskin, and Alexander Borzunov. 2023. SWARM Parallelism: Training Large Models Can Be Surprisingly Communication-Efficient. In *Proceedings of the 40th International Conference on Machine Learning (ICML '23)*. PMLR, 29416–29440.

[24] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU. In *Proceedings of the 40th International Conference on Machine Learning (ICML '23)*. PMLR, 31094–31116.

[25] Yifan Xiong, Yuting Jiang, Ziyue Yang, Lei Qu, Guoshuai Zhao, Shuguang Liu, Dong Zhong, Boris Pinzur, Jie Zhang, Yang Wang, Jithin Jose, Hossein Pourreza, Jeff Baxter, Kushal Datta, Prabhat Ram, Luke Melton, Joe Chau, Peng Cheng, Yongqiang Xiong, and Lidong Zhou. 2024. SuperBench: Improving Cloud AI Infrastructure Reliability with Proactive Validation. In *2024 USENIX Annual Technical Conference (USENIX ATC '24)*. USENIX Association, 835–850.

[26] Yuning Zhang, Yan Yan, Nan Yang, and Dong Yuan. 2026. AgentServe: Algorithm-System Co-Design for Efficient Agentic AI Serving on a Consumer-Grade GPU. *arXiv preprint arXiv:2603.10342* (2026). doi:10.48550/arXiv.2603.10342

[27] Yusheng Zheng, Jiakun Fan, Quanzhi Fu, Yiwei Yang, Wei Zhang, and Andi Quinn. 2026. AgentCgroup: Understanding and Controlling OS Resources of AI Agents. In *1st Workshop on Agentic Operating Systems (AgenticOS '26)*.