PERFORMANCE AND ENERGY IMPLICATIONS OF DESIGNING EFFICIENT AGENTIC WORKFLOWS: PITFALLS AND OPPORTUNITIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic systems extend traditional LLM inference by coordinating specialized agents with external tools and data sources to execute increasingly complex, multi-step tasks. However, translating these capabilities into cost-effective real-world deployments remains an open challenge. Agentic design choices (including call frequency, tool integration, and orchestration) can substantially change latency and energy while maintaining similar accuracy. In our evaluation, iso-accurate configurations differ by as much as 238s and 1,013× in latency, or 177.8kJ and 1,153× in energy per task. In this paper, we construct 1,656 agentic systems across a diverse set of challenging, state-of-the-art agentic benchmarks and characterize the performance and energy implications of key agentic design decisions from a systems perspective. Through this characterization, we identify opportunities to improve the efficiency of agentic systems that can be used to guide more efficient deployments.

1 INTRODUCTION

Agents are deployed across a broad range of domains to execute increasingly complex, multi-stage reasoning tasks (e.g., coding, scientific research, financial services, healthcare ; Huang et al. 2025; Shen et al. 2025b). A recent industry survey found that 59% of agentic systems coordinate multiple models and that 53% of systems use structured multi-step workflows with more than autonomous steps (Pan et al., 2025). Compared with standalone LLM engines, such complex agentic workflows can incur orders of magnitude higher computational costs, latency, and energy (Oviedo et al., 2026; Kim et al., 2026b; Wang et al., 2025a). With deployments scales ranging from tens of thousands to over one million daily users, the complexity of agentic workloads raises key questions about how to design and deploy agentic systems that have high performance, low cost, and low energy footprint.

In order to improve task accuracy and effectiveness, developers of LLM-driven systems primarily rely on scaling models (e.g., higher parameter count, more FLOPs), higher reasoning effort, and higher output token budgets; unfortunately, this also comes at the expense of higher latency and energy costs. However, this abstraction falls short for evaluating the efficiency of agentic systems, where execution is shaped by the structure of the workflow itself, and the cost of a single LLM call is only a component of the broader system. For instance, an agentic system handling a single request may include multi-turn LLM calls, tool use, orchestration, and parallel execution (Luo et al., 2025), as illustrated in Figure 1 (left). As a result, latency and energy costs depend *heavily* on how multiple models and external tools are integrated or orchestrated together. In this paper, we highlight key design considerations in agentic systems (e.g., workflow structure, number of turns, external tool configuration, model sizes) and show how they can yield up to 1,013.2× and 1152.6× differences in latency and energy at iso-accuracy, respectively. Figure 1 (right) illustrates this variation with the Pareto frontiers of five state-of-the-art agentic systems across four challenging coding, long-context, and agentic tasks, revealing substantial differences in the latency and energy costs of systems despite comparable accuracy. Thus, the cost of a target accuracy is not only by the underlying model used but also by the broader, end-to-end agentic workflow structure.

Prior work on the energy cost of LLM inference (Jin et al., 2025b; Fernandez et al., 2025; Oviedo et al., 2026), along with industry standardized leaderboards and benchmarks such as the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

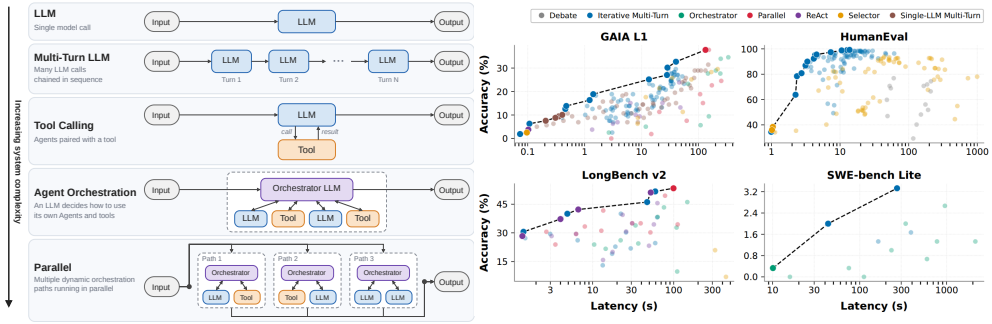


Figure 1: **Left:** We illustrate the bottom-up design space explored in this work, progressively adding agentic workflow complexity from multi-turn coordination to tool use, dynamic orchestration, and parallelization. **Right:** The resulting design space spans 1,000+ agentic configurations across different models, tools, and execution paths. We evaluate a representative set of these configurations across diverse datasets and agentic workflows, revealing up to 1,013.2× variation in latency at comparable accuracy. For a more detailed Pareto-frontier analysis, see Appendix A.

ML.ENERGY Benchmark (Chung et al., 2026), the AI Energy Score from Hugging Face and Hugging Face and Salesforce (2025), and MLPerf Inference (Reddi et al., 2020), has studied single monolithic model serving; none of these prior efforts study the cost and efficiency of multi-stage agentic workflows. Although recent work (Kim et al., 2026a; Raj et al., 2025b; Pan et al., 2025; Wang et al., 2025a) has quantified the cost of given and deployed agentic systems, to our knowledge none have examined the *latency and energy cost* of each design decision under *iso-accuracy* constraints. Characterizing the efficiency of agentic workflows at iso-accuracy presents key challenges: decisions which affect system efficiency (e.g., call budgets, model selection, agent parallelism) are often tightly coupled with task performance. Additionally, agentic workloads often rely on dynamic execution pathways where LLMs are utilized to determine the number of agent calls and tool uses—resulting in heavy-tailed execution costs and highly variable, unpredictable latency and energy. Furthermore, the end-to-end system efficiency is highly dependent on the deployment platform, with optimal configurations varying across serving constraints and hardware platforms.

In this paper, we characterize agentic systems bottom-up, adding one design axis at a time: multi-turn coordination, tool use, dynamic orchestration, and parallelism. We evaluate 1,656 configurations across seven agentic systems (Multi-Turn AgentChat, ReAct, RAG, MagenticOne, SelectorGroupChat, Multi-Agent Debate, and Parallel MagenticOne) on nine state-of-the-art benchmark suites. Our evaluation covers different scale Qwen3 and Gemma3 models across two commodity GPU platforms. For each axis, we identify a pitfall in the design choice that raises latency or energy without commensurate accuracy and an opportunity that can be used to recover the accuracy at lower cost. We believe these principles should guide the design and implementation of efficient agentic workflows across research and practice. Across the agent archetypes presented in Figure 1, we find:

- Achieving Pareto-optimal accuracy–performance tradeoffs requires carefully co-designing model size with agentic workflow structure for given tasks. For instance, we find cascading progressively larger models in a multi-stage pipeline can achieve the same accuracy as multi-turn LLM agents at 4.8× lower cost on GAIA L1 (Section 3).
- Tool use can substantially improve agentic capabilities; however, realizing these benefits without exacerbating costs is challenging due to the large design space and unintuitive tool-agent interactions. For example, we show naively implementing RAG with retrieval over a large vector database fetching more than 5 documents, can incur 9× higher cost than a standalone LLM at iso-accuracy; carefully sizing retrieval and LLM stages can reduce cost of RAG by up to 10× (latency) / 14× (energy) at iso-accuracy (Section 4).
- Complex agentic workflows with dynamic orchestration can exhibit uneven workload distribution across parallel task streams. We find that individual agents can account for up to 85% of end-to-end latency, leading to substantial GPU idle time (e.g., 33.3% of an agent path's lifetime) that further exacerbate serving costs and inefficiency for agentic workflows (Section 5).
- While user-facing services can optimize for tail latency (p90 or p99 latency) or the average case (p50); we find that no single agentic system configuration can be served optimally across

both regimes. For instance, in iso-accurate configurations, configurations that minimize p50 are sub-optimal in the tail scenarios by up to 253% for p90 and 1100% for p99 latency (Section 6).

2 METHODOLOGY

We evaluate seven representative agentic systems spanning the four archetypal design decisions we explore in the paper: multi-turn coordination, tool calling, dynamic agent orchestration, and parallelization. For serving and benchmarking, we use the AutoGen and AgentBench frameworks (Wu et al., 2023; Liu et al., 2023) as our primary evaluation methodology.

- **Multi-Turn AgentChat (Wu et al., 2023)**: A baseline LLM agent that performs inference and makes multiple potential sequential LLM calls across the same and/or different models.
- **ReAct (Yao et al., 2022)**: A single LLM that iteratively reasons and invokes tools: web search (DuckDuckGo), URL fetch, code execution (subprocess), and file read/list.
- **Retrieval-Augmented Generation (RAG; Lewis et al., 2020)**: A single LLM augmented with external knowledge retrieved from a document datastore.
- **MagenticOne (Fourney et al., 2024)**: A multi-agent system with an orchestrator LLM that plans and delegates tasks to specialized agents for coding, terminal execution, file system access, and web browsing.
- **SelectorGroupChat**: A multi-agent system that uses an LLM-based selector to determine which agent should act next (Wu et al., 2023), following the MagenticOne (Fourney et al., 2024) agent configurations.
- **Multi-Agent Debate (Du et al., 2024)**: Multiple agents independently generate and critique solutions through iterative rounds of discussion before producing a final answer.
- **Parallel MagenticOne Systems**: Multiple MagenticOne (Fourney et al., 2024) agentic systems executing in parallel, with a final orchestrator aggregating their outputs into a single answer.

Please see Appendix B for detailed agentic system structure descriptions and their configurations.

Datasets. We evaluate our agentic systems across a diverse set of representative benchmarks spanning question answering (TriviaQA; Joshi et al., 2017), long-context reasoning (LongBench v2; Bai et al., 2025), code generation (HumanEval; Chen et al., 2021), general-purpose task completion (GAIA; Mialon et al., 2024), and software engineering (SWE-bench; Jimenez et al., 2024). In addition, we construct datastores using the production-representative CompactDS and Sphere datasets (Lyu et al., 2025; Piktus et al., 2021).

Models and Serving. We serve our models using vLLM (Kwon et al., 2023) or SGLang (Zheng et al., 2024), which provide high-throughput, memory-efficient execution via optimized attention and KV-cache management. We primarily evaluate the Qwen3 (Yang et al., 2025) and Gemma3 (Team et al., 2025) model families, spanning reasoning and non-reasoning models from 0.6B to 27B parameters. The debate agent additionally draws on Llama 3.1 (Grattafiori et al., 2024), Nemotron-H (Blakeman et al., 2025), Phi-4 (Abdin et al., 2024), and Ministral (Liu et al., 2026).

Hardware. Experiments are conducted on a heterogeneous set of hardware with CPU-based retrieval running on Intel Xeon Gold 6442Y, Intel Xeon Platinum 8380, and Intel Xeon Platinum 8275CL, and model inference deployed using NVIDIA A6000 Ada and L4 GPUs.

3 SCALING MODEL SIZE ALONE IS NOT ENOUGH: AGENTIC STRUCTURE SETS THE COST OF ACCURACY

A natural first step to improving the accuracy of an LLM is: scaling the model size or extending chain-of-thought reasoning (Wei et al., 2022) by repeatedly invoking a model with its previous output appended as additional context (Shinn et al., 2023; Chen et al., 2024a;b). Although both approaches require minimal orchestration logic, we find this is sub-optimal when extending to more sophisticated agentic workflows. In this section, we evaluate two example but distinct types of multi-turn agentic workflows: (1) repeatedly invoking an LLM and (2) cascading progressively larger models. We show that even when both agentic workflow structures achieve commensurate accuracy on the GAIA L1 task set, they exhibit orders of magnitude different performance and energy costs.

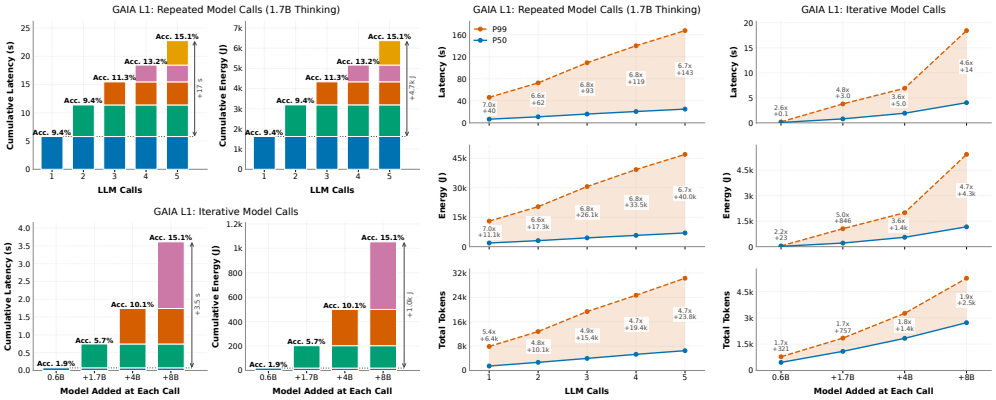


Figure 2: **Left**: repeating a 1.7B reasoning Qwen model five times raises latency and energy 3.9× (5.8s → 22.8s; 1.6kJ → 6.4kJ) as accuracy rises from 9.4% to 15.1%. A 4-call cascade adding a larger model each call (0.6B → 8B) reaches the same 15.1% at 3.5s and 1kJ, 4.8× lower. **Right**: the P50 (solid) to P99 (dashed) latency gap widens with calls, reaching 143s for repeated calls (a 6.7× gap) while only being 14s for iterative model calls (4.6×)

Pitfall: *Scaling model size or call count while oblivious to workflow structure lands off the accuracy–latency and accuracy–energy Pareto frontier and widens latency tails.*

Figure 2 compares two structure-oblivious ways to reach 15.1% accuracy on GAIA L1: repeating a 1.7B thinking model up to five times (top), and cascading from a 0.6B to an 8B non-thinking model, one size per call (bottom). Repeatedly invoking the same model appended with output of each prior turn costs an additional 17s of latency and 4.7kJ of energy over a single call improves accuracy by 5.7%; on the other hand, the cascading architecture improves accuracy by 13.2% for 3.5s and 1.03kJ. At iso-accuracy, the single-model multi-turn workflow is 4.9× slower and 4.6× more energy-intensive than the multi-model cascade.

Structure also determines tail-effects of agentic services. Latency-sensitive, user-facing deployments provision for tail latency (P99), not the median, so a workflow whose P99 drifts far from its P50 must be over-provisioned to meet its target. Figure 2 (right) shows that multi-turn workflows compound per-call variability, widening the P50–P99 gap with every turn. For the repeated 1.7B thinking model, the gap grows from 40 s after one call to 143 s after five, with P99 reaching 167 s against a P50 of 25 s (6.7×). The cascade widens far less: after four calls its P99 is 18.44 s against a P50 of 4 s (4.6×), a 14.4 s gap that is 9.9× narrower than the multi-turn workflow's.

Opportunity: *Co-designing structure with model size opens counter-intuitive trade-offs, such as larger models being cheaper: cascading up to an 8B model matches five calls of a 1.7B model at 6.3× lower latency and 6.1× lower energy.*

Traditional approaches to agentic system design often assume that increasing model capability improves accuracy at the expense of higher latency and energy per inference (Kaplan et al., 2020; Hoffmann et al., 2022; Luccioni et al., 2024). Compared with naively repeating calls to a smaller model, progressively introducing more capable models provides greater opportunities to improve the final answer while requiring fewer total model calls to reach a target accuracy. Two effects drive this gap. First, queries answered correctly terminate early rather than proceeding through every round, with the cascaded system making 3.7 LLM calls on average versus 4.4 for the naive. Second, the cascade requires fewer tokens to correctly answer queries, 12.6× per task (278 vs. 3,497). In contrast, repeatedly invoking a smaller model requires more queries to progress through additional rounds: the naive configuration requires 6 rounds to reach the same accuracy achieved by the cascade, which starts from a lower 1.9% accuracy but gains accuracy more efficiently across fewer turns.

This structure dependence is not specific to GAIA L1. Figure 7 (Appendix A) overlays Pareto frontiers across all nine benchmarks and seven agentic systems, and the frontier is traced by a different structure in each accuracy regime: RAG dominates the knowledge-intensive tasks (TriviaQA, NQ, MMLU) at sub-second latency, multi-turn and selector workflows trace the frontier for code generation (HumanEval) and long-context reasoning, and orchestrated and parallel systems appear on the frontier only for the hardest agentic tasks (GAIA L2/L3, SWE-bench Lite). No single structure is Pareto-optimal across tasks, so the choice of workflow must be made per task rather than fixed once.

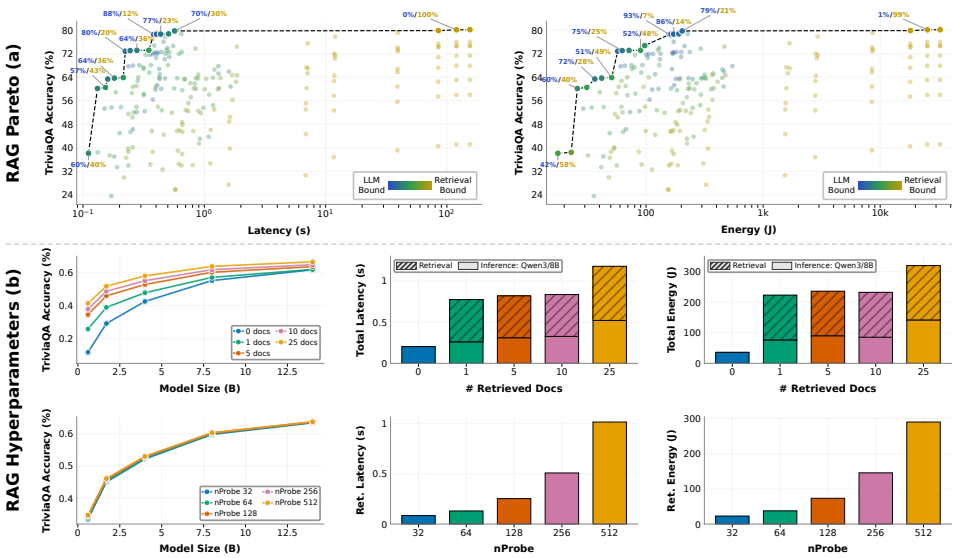


Figure 3: **(a)** RAG's Pareto frontier is mostly LLM-bound for the 38%→80% acccuracy regime for a 5× latency/energy cost. Only at the tail does it flip tool-call-bound: retrieval makes up ≈100% of the cost, where 150–250× more latency/energy buys just 0.1–0.4 more accuracy points. **(b)** Identifying performance efficient points is challenging due to the large design space of RAG hyperparameters, e.g., nProbe 32→512 costs 12× more for +0.5% accuracy, while increasing docs retrieved from 0→25 yields +8.6% accuracy at lower cost.

4 Tools Are Not Free: Tool Hyperparameters Must Be Co-Designed with the LLM

Agentic systems often implement myriad tools (Yao et al., 2024; Xie et al., 2024; Patil et al., 2025), each of which can introduce its own set of configuration choices and opportunities for optimization. As an example, in this section, we study the design space introduced by retrieval engines, typically found in retrieval-augmented generation (RAG) (Lewis et al., 2020). RAG is a widely used and well-established form of tool integration that exposes multiple independently configurable dimensions. Retrieval is one instance of a broader pattern. For instance, code interpreters expose sandbox and timeout settings; computer use environments simulate file system access; and web search and browsers expose page-fetch depth and rendering options that impact time and quality.

Pitfall: *Tools are not free: retrieval, not the LLM, is the bottleneck in 68% of RAG configurations, and tool hyperparameters expand the design space in ways that require domain expertise to search.*

However, emerging literature details scenarios for code generation and validation, web search, and multi-modal input processing where tools can pose non-trivial overheads to end-to-end agentic workflows (Yang et al., 2026; Raj et al., 2025a). Similarly, although prior work develops RAG agents, it often underestimates the full extent of the overheads introduced by these systems (Jiang et al., 2025; Jin et al., 2025a; Lyu et al., 2025; Huang et al., 2023; Jiang et al., 2024). However, Figure 3 shows that this assumption does not always hold. In fact, of the configurations we evaluate, 141/208 are retrieval-bound (where retrieval makes up more than 50% of the total latency or energy cost), making retrieval rather than LLM inference the primary performance bottleneck. This is due to two primary overheads: (1) the time and energy spent to retrieve documents from large (TB-scale) datastores can be high (Shen et al., 2025a; Jiang et al., 2025), and (2) the prefill cost to process additional documents in-context can exacerbate inference overheads.

Finding effective configurations for tools, such as the RAG configurations explored in our study, introduces a challenging design-space exploration problem. LLM inference already exposes a large configuration space spanning model size, context length, quantization, offloading, and speculative decoding. Tools such as RAG further expand this space with additional parameters, including search depth, number of retrieved documents, datastore size, and embedding quantization (see Appendix D for details). These parameters interact in non-obvious ways with the LLM, making the combined space difficult to navigate. For example, increasing $nProbe$ (Search Effort) from 32 to 512 increases retrieval latency and energy by roughly 12× (0.08s → 1.01s; 23J → 290J for Qwen3-8B) for only a

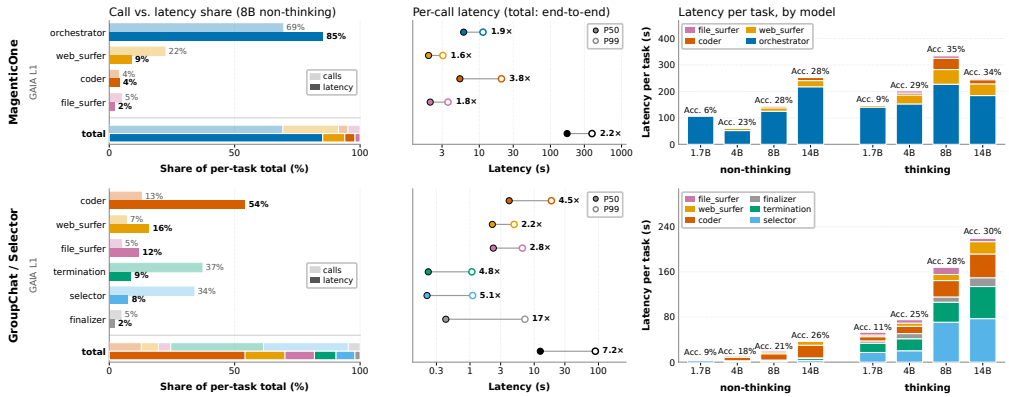


Figure 4: **Left**: MagenticOne is orchestrator-dominated (69% calls, 85% latency), while GroupChat/Selector shifts latency to tools like the coder (54%) despite selector/termination agents accounting for 71% of calls. **Middle**: Task-level latency variance compounds into the end-to-end tails, stretching GroupChat/Selector from 12.4s (P50) to 89.4s (P99) (7.2×) versus MagenticOne's 171.7s (P50) to 384s (P99) (2.2×). **Right**: When an LLM orchestrate its own execution performance is less predictable, where a 1.7B model orchestration system performing worse than a 4B system.

0.5-point accuracy gain. In contrast, increasing retrieved documents from 0 to 25 incurs comparable cost (0.20s → 1.18s; 36J → 319J) but improves accuracy by 8.6 points (55.2% → 63.8%).

Opportunity: *Co-designing tool hyperparameters with LLM size can jointly optimize accuracy and energy: allocating the same retrieval budget to more documents rather than deeper search improves accuracy by 8.6% versus just 0.5%, while effective retrieval allows a smaller 1.7B model to match the accuracy of a 14B model without retrieval at 2.8× lower energy.*

Even along the Pareto frontier, retrieval is never negligible. Figure 3 (a) annotates each frontier point with its LLM/retrieval cost split: retrieval accounts for 12–49% of latency and 7–58% of energy across the frontier, and it stays under 40% of end-to-end cost for only 12 of 15 latency points and 11 of 17 energy points. In this LLM-bound regime, co-designing retrieval depth with model size climbs accuracy from 38% to 79.7% for a 5×/12× cost increase (0.11 s to 0.57 s; 18 J to 204 J), a slope that neither lever achieves alone: scaling the LLM without retrieval leaves accuracy below 73% (achieved by gemma3 12b), and deepening retrieval without scaling the LLM improves accuracy by 1% but increases energy by 12× (Figure 3 (b)). At the tail of the frontier the regime flips: the final 0.4% improvement are reachable only through retrieval, which then accounts for nearly 100% of cost. The LLM and its tools must be co-designed in both regimes, to hold retrieval to a minority of cost where the LLM is the bottleneck, and to decide whether the last accuracy points are worth a retrieval-bound system where it is not.

5 Agentic Execution Is Tail-Heavy, Causing Unpredictable Costs and Hardware Underutilization

Beyond the multi-turn calls and fixed tool pipelines from the previous sections, the next step in agentic capability is letting agents dynamically and autonomously orchestrate tasks: deciding at runtime which tools, knowledge sources, and agents to invoke and in what order (Yao et al., 2022; Schick et al., 2023; Fourney et al., 2024). Orchestration broadens the design space and improves flexibility, but it also makes execution harder to predict and control, since the number and type of calls are now chosen by the model rather than by the designer. In this section we examine two orchestration styles: MagenticOne, in which an orchestrator LLM follows a plan across a fixed set of agents, and SelectorGroupChat, in which a selector LLM chooses the next agent after every step.

Pitfall: *LLM-driven orchestration creates highly skewed agent utilization where a single agent can dominate 54% of execution time; in other scenarios, orchestrators themselves can account for up to 69% of calls and 85% of latency.*

Figure 4 breaks down each system's execution on GAIA L1 by agent, showing the share of calls and latency attributed to each agent (left), the P50-to-P99 spread of each agent's per-call latency (middle), and per-task latency by model size and thinking mode (right). Orchestration produces highly

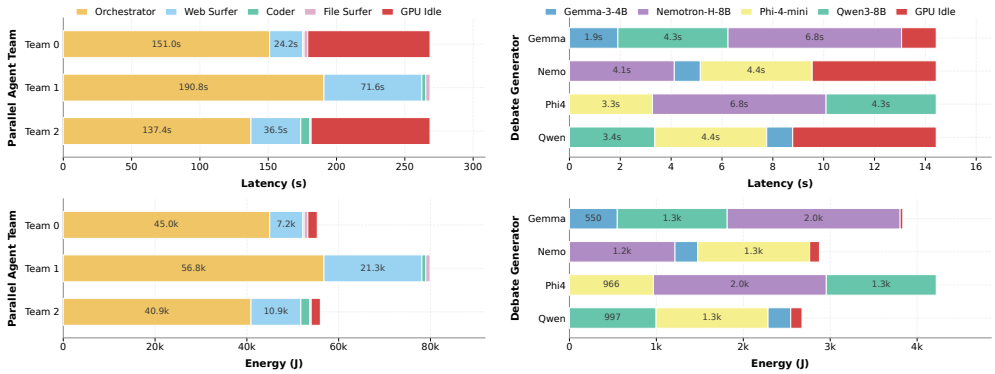


Figure 5: **Left**: Three identical MagenticOne agent teams exhibit divergent completion times due to LLM variability and dynamic self-orchestration, with the slowest and fastest paths differing by up to 89.5,s (33% of the longest path). **Right**: Four debating paths generate and critique results using different models, leading to substantial GPU idle time (completion times range 8.8s–14.4s, so the fastest path idles for up to 5.7s, 39% of its own runtime) from divergent execution times.

uneven execution: MagenticOne's orchestrator accounts for 69% of LLM calls and 85% of end-to-end latency, while SelectorGroupChat's selector and termination agents account for 71% of calls but only 17% of latency. Call frequency, however, does not translate to cost: the SelectorGroupChat coder accounts for only 13% of calls but 54% of end-to-end latency. Individual calls also exhibit substantial tail variability, with *file_surfer* rising from 2 s at P50 to 6.6 s at P99 (2.8×) and *coder* from 5.3 s to 20.5 s (3.8×).

These per-call tails compound across the workflow (as they do in multi-turn LLMs) as the orchestrator decides at runtime which agents to invoke and how many times. Consequently, expected execution cost can differ substantially even across models using the same workflow. For example, we compare a MagneticOne system with 1.7B and 4B non-rasoning system. Although the workflow with 1.7B model implements a cheaper model, it takes 1.7× longer end-to-end than the system with larger 4B model. This is a direct result of the orchestrator invoking the leaner, cheaper model nearly twice as many times (33.9 versus 15.8), leading to 50% more total agent calls (36 vs. 24). Furthermore, the 4B system also generates 52% fewer tokens per call (259 vs. 540), allowing its higher per-token latency to be outweighed by fewer, shorter invocations (Appendix E).

A further layer of complexity is introduced in multi-agent workflows where agents execute independent actions or concurrently iterate on a task in parallel. Parallel execution hides latency by running paths concurrently, at the cost of proportionally higher energy. Because each path inherits the multi-turn and orchestration variability discussed above, paths finish at different times, exposing idle time between the fastest and slowest paths. We examine two parallel systems: multi-agent debate, in which different models generate and critique responses, and parallel MagenticOne, in which three identical instances process the same query.

Pitfall: *Parallel agents suffer from severe load imbalance and GPU under-utilization, with even nominally identical paths wasting up to 39% of critical-path time waiting on slower paths.*

Because parallel execution synchronizes on the slowest path, faster paths remain idle while their GPUs continue consuming power without performing useful computation. Figure 5 shows that parallel execution can nevertheless hide substantial latency, reducing execution time by 3.2× for MAD (45.8s sequential vs. 14.4s parallel) and 2.3× for Parallel MagenticOne (628.8s sequential vs. 268.5s parallel). However, this latency reduction comes with significant idle time: in Parallel MagenticOne, Teams 0 and 2 sit idle for a combined 176.6s while waiting for Team 1's 268.5s critical path, wasting 4.1kJ of energy at a 23W idle power draw (2% additional energy). Similarly, the Gemma, Llama-3.1-8B, and Phi-4-mini paths in MAD collectively idle for 11.9s while waiting for the 14.4s Nemotron-H-8B critical path, adding roughly 270J of idle energy (1.7% additional energy). Despite these relatively small gaps, the parallel agents execute nearly identical workflows. In workflows with heterogeneous models and more divergent execution paths, completion-time differences and idle energy wastage will be substantially larger.

Opportunity: *Given the structural skew in agentic workflows, hardware resources should be right-sized across agents, as opposed to provisioned uniformly. Agents on the critical path should be*

	Latency (s)			Energy (J)		
	P50	P90	P99	P50	P90	P99
1.7nt x2	✓	✓	+939%	✓	✓	+475%
1.7nt x4	✓	✓	+941%	✓	✓	+476%
1.7nt x5	✓	✓	+943%	✓	✓	+477%
Selector: 8B nt	✓	+34%	+79%	✓	+25%	✓
1.7nt-4nt-14nt-14t	✓	+130%	+955%	✓	✓	+521%
0.6nt-1.7nt-4nt-14t	✓	+135%	+923%	✓	+2%	+501%
1.7nt-4nt-4t-14t	✓	+253%	+1.1k%	✓	+54%	+612%
1.7nt-8nt	+49%	+66%	✓	+50%	+67%	✓
8nt-14nt	+103%	+31%	✓	+106%	+31%	✓
Single-Turn: 8B nt	+66%	+69%	✓	+68%	+71%	✓

#	1	2	3	4	5	6
System	1-turn LLM	1-turn LLM	4-turn LLM	4-turn LLM	Orchestrator	Orchestrator
Config	0.6B nt L4	14B nt Ada	0.6B nt: L4 1.7B nt: L4 4B nt: L4 4B t: L4	1.7B nt: L4 4B nt: L4 4B t: L4 14B t: Ada	14B t Ada: all 4	8B t L4: all 4
Acc. (%)	1.9	8.8	25.2	32.7	34.0	34.6
Energy (J)	14.7	136	2.9k	10.9k	74.5k	75.2k

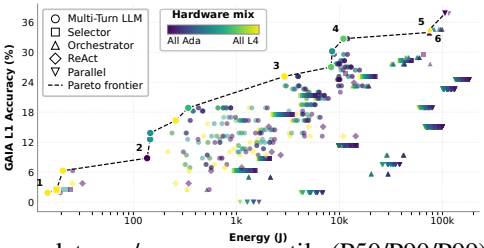


Figure 6: **Left:** Systems on the Pareto frontier at one latency/energy percentile (P50/P90/P99). A ✓ indicates that the configuration is on that percentile's frontier and a % indicates deviation from the nearest frontier point. Setups optimal at P50 can be up to +1.1%/+612% off the P99 latency/energy frontier. **Right:** Hardware-aware placement across A6000 Ada/L4 configurations reduces energy by up to 1.4× versus all-Ada configurations, expanding the space of Pareto-optimal configurations.

optimized for high-performance (latest GPUs, latency-optimized serving), while agents with slack should be run with efficiency-optimized runtimes (batching, smaller models) or cheaper hardware (leaner, older GPUs)

Agentic execution skew is difficult to eliminate, but heterogeneous hardware offers a promising way to mitigate its cost. Rather than treating all inference requests identically, exposing per-agent SLOs would let operators exploit available slack: routing non-critical calls to older or leaner GPUs, applying DVFS or larger batches, and reserving the fastest hardware for critical-path agents. Provisioning frameworks such as Murakkab (Chaudhry et al., 2025) already allocate resources as agents are invoked, and a per-agent SLO would make hardware selection one more knob of agentic orchestration. We evaluate this across A6000 Ada and power-efficient L4 GPUs in Appendix E: assigning the L4 to agents with slack cuts GPU idle time by 80% (142s) at iso-accuracy and iso-energy, with no increase in end-to-end latency (Figure 14 sweeps further configurations). The same mechanism extends the useful life of aging accelerators: off-critical-path agents can absorb older, lower-power GPUs (e.g., NVIDIA L4, NVIDIA RTX PRO 4500 Blackwell) without affecting the workflow's latency. Finally, these SLOs could be exposed to the agents themselves, letting an orchestrator provision resources according to the expected cost and criticality of each action.

6 Pareto-Optimal Configurations Do Not Transfer Across Serving Regimes: Favor Robustness Over Peak Efficiency

Interactive, user-facing services provision for tail latency (P90/P99), since the SLO must hold for nearly every request; batch and cost-driven deployments provision for the median (P50), where average throughput per joule matters more than any single request. One would expect a configuration efficient at P50 to remain efficient at P99, with tails growing in proportion to the median. However, the dynamic call counts and per-call variability of Sections 3–5 break this proportionality, with configurations Pareto-optimal at P50 potentially falling far from the frontier at P99.

Pitfall: *Pareto-optimal configurations do not transfer across serving regimes or hardware: a configuration on the P50 frontier can sit over 1,100% off the P99 frontier.*

Figure 6 (left) tabulates, for 10 iso-accurate configurations on GAIA L1, whether each lies on the latency and energy Pareto frontier at P50, P90, and P99 and, if not, how far off it sits; Figure 6 (right) plots the accuracy–energy frontier when each agent may run on either an A6000 Ada or an L4, with the six frontier points annotated by system, model, and GPU assignment. We denote a multi-turn configuration by the model invoked at each turn, so `1.7nt-4nt-4t-14t` calls a 1.7B non-thinking model, then 4B non-thinking, 4B thinking, and 14B thinking, terminating early on correct answers. The frontier is unstable across percentiles: for pareto-optimal configurations at P50, only 40% remain latency-optimal at P90 and 5% at P99; for energy, 47% remain optimal at P90 and 11% at P99 (the table shows a representative subset). The deviations are large. `1.7nt-4nt-4t-14t` lies on both P50 frontiers yet off the P99 frontiers, incurring 1,1k% and 612% latency and energy overhead, respectively, in comparison to iso-accurate Pareto-optimal points on the frontier.

Opportunity: *Rather than optimizing for a single Pareto frontier, target configurations should favor those that remain near-optimal across diverse serving regimes; hardware heterogeneity provides greater flexibility in adapting to varying regimes.*

Attempting to identify a single optimal configuration for a given accuracy level across median and tail serving regimes is infeasible. The breadth of agentic system designs creates ample opportunity to hyperoptimize for a specific regime, often at the expense of performance in other regimes. Instead, we can identify agentic configurations that remain consistently near-optimal across different median and tail serving regimes, even if they are not optimal in any single regime. For example, a four-model cascade using the `4NT-8NT-1.7T-14T` configuration achieves 27.7% accuracy while maintaining an average gap of just 26% from the Pareto frontiers across the P50, P90, and P99 regimes (with individual gaps ranging from 7–52%). In contrast with the configurations shown in Figure 6 (left) are consistently 500–1000% from the Pareto frontier.

Hardware selection is a second lever for robustness. Figure 6 (right) expands each of 124 agentic system designs into every assignment of its agents to an A6000 Ada or an L4, yielding 26,889 hardware configurations. Mixed hardware dominates the energy frontier: 12 of the 15 energy-optimal points place at least one agent on the L4, cutting energy by up to 1.4× (1.3–1.4× at 14–19% accuracy) relative to the all-Ada frontier, whereas the latency frontier is entirely all-Ada (Appendix E). Hardware choice can also bring different architectures to the same point: a 14B thinking orchestrator on four A6000 Adas and an 8B thinking orchestrator on four L4s reach 34.0% and 34.6% accuracy at 74.5 kJ and 75.2 kJ (points 5 and 6). Expanding the hardware space expands the set of near-optimal configurations, giving operators more ways to hold a target accuracy across regimes.

7 Related Work

Characterizations of Agentic System Design and Performance Prior work on agentic systems has largely targeted conventional systems metrics (e.g. throughput, latency, and MFU). Proposed optimizations range from techniques for meeting latency targets (Chaudhry et al., 2025; Asgar et al., 2025) to improved scheduling of agentic workflows (Lin et al., 2024; Abhyankar et al., 2024; Luo et al., 2025; Santhanam et al., 2024; Pagonas et al., 2025). These designs often reduce end-to-end latency but overlook energy consumption and the impact of design choices on task accuracy. Efforts to broaden characterization beyond latency and throughput have mostly been qualitative studies of production agentic deployments (Pan et al., 2025), or have examined CPU efficiency in isolation rather than joint CPU-GPU efficiency (Raj et al., 2025b). More recent work has begun to study the tradeoffs among accuracy, latency, and energy (Kim et al., 2026b), but it covers a narrow set of tasks and a limited design space compared to the modern agentic deployments discussed in Section 2.

Characterizations of Inference Energy Use. As AI energy demands grow, a growing body of work studies the energy efficiency and environmental impact of inference. Evaluations of language model inference have relied either on direct empirical measurement (Samsi et al., 2023; Patel et al., 2024; Luccioni et al., 2024; Fernandez et al., 2025) or on analytic models that estimate energy from assumed utilization and hardware characteristics (De Vries, 2023; Mistral AI, 2025; Rincé & Banse, 2025; Epoch AI, 2025). Recent studies extend these measurements to test-time compute and reasoning, where additional generated tokens can increase per-query energy by up to 13× relative to standard queries (Oviedo et al., 2026; Jin et al., 2025b; Elsworth et al., 2025; Chung et al., 2026). Industry benchmarks such as ML.ENERGY, MLPerf Power, and AI Energy Score have established standardized frameworks for evaluating energy requirements of conventional model serving (Chung et al., 2026; Tschand et al., 2025; Hugging Face and Salesforce, 2025). Unlike traditional LLM inference, agentic workloads incur additional costs from tool calls, dynamic control flow, and repeated invocations of reasoning and non-reasoning models.

8 Conclusion

Agentic systems introduce a fundamentally different systems design space from traditional single-turn LLM inference, changing how systems must be designed as accuracy increasingly interacts with latency, energy, and execution variability. In this paper, we explore the performance and energy implications of key agentic design choices, spanning multi-turn LLM calls, tool use, dynamic orchestration, and parallel execution. Our results highlight the need to treat agentic systems as end-to-end systems, where accuracy, latency, energy, and execution variability are considered jointly rather than optimizing individual agents or components in isolation.

9 AI USAGE DISCLOSURE

In this work, we used generative AI tools for language polishing, experimental environment setup, and figure plotting. The authors reviewed the AI-generated experimental suggestions and made the final decisions on the experimental design and methodology. All AI-assisted text, code, and figures were reviewed by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Reyna Abhyankar, Zijian He, Vikranth Srivatsa, Hao Zhang, and Yiying Zhang. Infercept: Efficient intercept support for augmented large language model inference. *arXiv preprint arXiv:2402.01869*, 2024.

Zain Asgar, Michelle Nguyen, and Sachin Katti. Efficient and scalable agentic ai with heterogeneous systems. *arXiv preprint arXiv:2507.19635*, 2025.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, et al. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, 2025.

Aaron Blakeman, Aarti Basant, Abhinav Khattar, Adithya Renduchintala, Akhiad Bercovich, Aleksander Ficek, Alexis Bjorlin, Ali Taghibakhshi, Amala Sanjay Deshmukh, Ameya Sunil Mahabaleshwarkar, et al. Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models. *arXiv preprint arXiv:2504.03624*, 2025.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pp. 2206–2240. PMLR, 2022.

Gohar Irfan Chaudhry et al. Murakkab: Resource-efficient agentic workflow orchestration in cloud platforms. *arXiv preprint arXiv:2508.18298*, 2025.

Justin Chen, Swarnadeep Saha, and Mohit Bansal. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7066–7085, 2024a.

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, James Zou, Matei Zaharia, and Ion Stoica. Llmselector: Learning to select models in compound ai systems. In *ICML 2025 Workshop on Collaborative and Federated Agentic Workflows*.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024b. ISSN 2835-8856. URL https://openreview.net/forum?id=cSimKw5p6R. Featured Certification.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. 2021.

Jae-Won Chung, Jeff J Ma, Ruofan Wu, Jiachen Liu, Oh Jun Kweon, Yuxuan Xia, Zhiyu Wu, and Mosharaf Chowdhury. The ml. energy benchmark: Toward automated inference energy measurement and optimization. *Advances in Neural Information Processing Systems*, 38, 2026.

Alex De Vries. The growing energy footprint of artificial intelligence. *Joule*, 7(10):2191–2194, 2023.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/du24e.html.

Cooper Elsworth, Keguo Huang, David Patterson, Ian Schneider, Robert Sedivy, Savannah Goodman, Ben Townsend, Parthasarathy Ranganathan, Jeff Dean, Amin Vahdat, et al. Measuring the environmental impact of delivering ai at google scale. *arXiv preprint arXiv:2508.15734*, 2025.

Epoch AI. How much energy does chatgpt use? *Epoch AI: Gradient Updates*, February 2025. URL https://epoch.ai/gradient-updates/how-much-energy-does-chatgpt-use. Accessed: 2026-06-07.

Jared Fernandez, Clara Na, Vashisth Tiwari, Yonatan Bisk, Sasha Luccioni, and Emma Strubell. Energy considerations of large language model inference and efficiency optimizations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32556–32569, 2025.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, et al. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, pp. 3887–3896. PMLR, 2020.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Jie Huang, Wei Ping, Peng Xu, Mohammad Shoeybi, Kevin Chen-Chuan Chang, and Bryan Catanzaro. Raven: In-context learning with retrieval-augmented encoder-decoder language models. *arXiv preprint arXiv:2308.07922*, 2023.

Yuxuan Huang, Yihang Chen, Haozheng Zhang, Kang Li, Huichi Zhou, Meng Fang, Linyi Yang, Xiaoguang Li, Lifeng Shang, Songcen Xu, et al. Deep research agents: A systematic examination and roadmap. *arXiv preprint arXiv:2506.18096*, 2025.

Hugging Face and Salesforce. AI Energy Score: Standardized energy efficiency ratings for AI models. https://huggingface.github.io/AIEnergyScore/, 2025. Accessed: 2026-09-25.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 604–613, 1998.

Wenqi Jiang, Shuai Zhang, Boran Han, Jie Wang, Bernie Wang, and Tim Kraska. Piperag: Fast retrieval-augmented generation via algorithm-system co-design. *arXiv preprint arXiv:2403.05676*, 2024.

Wenqi Jiang, Suvinay Subramanian, Cat Graves, Gustavo Alonso, Amir Yazdanbakhsh, and Vidushi Dadu. Rago: Systematic performance optimization for retrieval-augmented generation serving. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, pp. 974–989, 2025.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Chao Jin, Zili Zhang, Xuanlin Jiang, Fangyue Liu, Shufan Liu, Xuanzhe Liu, and Xin Jin. Ragcache: Efficient knowledge caching for retrieval-augmented generation. *ACM Transactions on Computer Systems*, 44(1):1–27, 2025a.

Yunho Jin, Gu-Yeon Wei, and David Brooks. The energy cost of reasoning: Analyzing energy usage in llms with test-time compute. *arXiv preprint arXiv:2505.14733*, 2025b.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Donghwan Kim, Prakhar Singh, Younghoon Min, Jongryool Kim, Jongse Park, and Kiwan Maeng. Characterization of multi-model agentic ai systems on general tasks via trace-driven simulation. *arXiv preprint arXiv:2606.01725*, 2026a.

Jiin Kim, Byeongjun Shin, Jinha Chung, and Minsoo Rhu. The cost of dynamic reasoning: Demystifying ai agents and test-time scaling from an ai infrastructure perspective. In *2026 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 1–16. IEEE, 2026b.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Chaofan Lin, Zhenhua Han, Chengruidong Zhang, Yuqing Yang, Fan Yang, Chen Chen, and Lili Qiu. Parrot: Efficient serving of {LLM-based} applications with semantic variable. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 929–945, 2024.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.

Yifan Lu, Rixin Liu, Jiayi Yuan, Xingqi Cui, Shenrun Zhang, Hongyi Liu, and Jiarong Xing. Routerarena: An open platform for comprehensive comparison of llm routers. *arXiv preprint arXiv:2510.00202*, 2025.

Sasha Luccioni, Yacine Jernite, and Emma Strubell. Power hungry processing: Watts driving the cost of ai deployment? In *Proceedings of the 2024 ACM conference on fairness, accountability, and transparency*, pp. 85–99, 2024.

Michael Luo et al. Autellix: An efficient serving engine for llm agents as general programs. *arXiv preprint arXiv:2502.13965*, 2025.

Xinxi Lyu, Michael Duan, Rulin Shao, Pang Wei Koh, and Sewon Min. Frustratingly simple retrieval improves challenging, reasoning-intensive benchmarks. *arXiv preprint arXiv:2507.01297*, 2025.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5303–5315, 2023.

Andreas Martin, Hans Friedrich Witschel, Maximilian Mandl, and Mona Stockhecke. Semantic verification in large language model-based retrieval augmented generation. In *Proceedings of the AAAI Symposium Series*, volume 3, pp. 188–192, 2024.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pp. 9025–9049, 2024.

Mistral AI. Our contribution to a global environmental standard for ai, July 2025. URL https://mistral.ai/news/our-contribution-to-a-global-environmental-standard-for-ai/. Accessed: 2026-06-07.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human-generated machine reading comprehension dataset. 2016.

Felipe Oviedo, Fiodar Kazhamiaka, Esha Choukse, Allen Kim, Amy Luers, Melanie Nakagawa, Ricardo Bianchini, and Juan M Lavista Ferres. Energy use of ai inference, efficiency pathways, and test-time scaling. *Joule*, 2026.

Nikos Pagonas, Yeounoh Chung, Kostis Kaffes, and Arvind Krishnamurthy. Cortex: Workflow-aware resource pooling and scheduling for agentic serving. *arXiv preprint arXiv:2510.14126*, 2025.

Melissa Z Pan, Negar Arabzadeh, Riccardo Cogo, Yuxuan Zhu, Alexander Xiong, Lakshya A Agrawal, Huanzhi Mao, Emma Shen, Sid Pallerla, Liana Patel, et al. Measuring agents in production. *arXiv preprint arXiv:2512.04123*, 2025.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Brijesh Warrier, Nithish Mahalingam, and Ricardo Bianchini. Characterizing power management opportunities for llms in the cloud. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 207–222, 2024.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2GmDdhBdDk.

Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Dmytro Okhonko, Samuel Broscheit, Gautier Izacard, Patrick Lewis, Barlas Oğuz, Edouard Grave, Wen-tau Yih, et al. The web is your oyster-knowledge-intensive nlp against a very large web corpus. *arXiv preprint arXiv:2112.09924*, 2021.

Ritik Raj, Souvik Kundu, Ishita Vohra, Hong Wang, and Tushar Krishna. Towards understanding, analyzing, and optimizing agentic ai execution: A cpu-centric perspective. *arXiv preprint arXiv:2511.00739*, 2025a.

Ritik Raj, Hong Wang, and Tushar Krishna. A cpu-centric perspective on agentic ai. *arXiv preprint arXiv:2511.00739*, 2025b.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331, 2023.

Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, et al. Mlperf inference benchmark. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pp. 446–459. IEEE, 2020.

Samuel Rincé and Adrien Banse. Ecologits: Evaluating the environmental impacts of generative ai. *Journal of Open Source Software*, 10(111):7471, 2025. doi: 10.21105/joss.07471. URL https://doi.org/10.21105/joss.07471.

Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc, 2009.

Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE high performance extreme computing conference (HPEC)*, pp. 1–9. IEEE, 2023.

Keshav Santhanam et al. Alto: An efficient network orchestrator for compound ai systems. In *Proceedings of the 4th Workshop on Machine Learning and Systems*, pp. 117–125, 2024.

Kunal Sawarkar, Abhilasha Mangal, and Shivam Raj Solanki. Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. In *2024 IEEE 7th international conference on multimedia information processing and retrieval (MIPR)*, pp. 155–161. IEEE, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Michael Shen, Muhammad Umar, Kiwan Maeng, G Edward Suh, and Udit Gupta. Hermes: Algorithm-system co-design for efficient retrieval-augmented generation at-scale. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, pp. 958–973, 2025a.

Minjie Shen, Yanshu Li, Lulu Chen, Zhichao Fan, Yanhang Li, and Qikai Yang. From mind to machine: The rise of manus ai as a fully autonomous digital agent. *arXiv preprint arXiv:2505.02024*, 2025b.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Wei Song, Zhenya Huang, Cheng Cheng, Weibo Gao, Bihan Xu, GuanHao Zhao, Fei Wang, and Runze Wu. Irt-router: Effective and interpretable multi-llm routing via item response theory. *arXiv preprint arXiv:2506.01048*, 2025.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Arya Tschand, Arun Tejusve Raghunath Rajan, Sachin Idgunji, Anirban Ghosh, Jeremy Holleman, Csaba Kiraly, Pawan Ambalkar, Ritika Borkar, Ramesh Chukka, Trevor Cockrell, et al. Mlperf power: Benchmarking the energy efficiency of machine learning systems from μwatts to mwatts for sustainable ai. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 1201–1216. IEEE, 2025.

Ningning Wang, Xavier Hu, Pai Liu, He Zhu, Yue Hou, Heyuan Huang, Shengyu Zhang, Jian Yang, Jiaheng Liu, Ge Zhang, et al. Efficient agents: Building effective agents while reducing cost. *arXiv preprint arXiv:2508.02694*, 2025a.

Yujing Wang, Hainan Zhang, Liang Pang, Binghui Guo, Hongwei Zheng, and Zhiming Zheng. Maferw: Query rewriting with multi-aspect feedbacks for retrieval-augmented large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25434–25442, 2025b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation, 2023. URL https://arxiv.org/abs/2308.08155.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Shicheng Xu, Liang Pang, Jun Xu, Huawei Shen, and Xueqi Cheng. List-aware reranking-truncation joint model for search and retrieval-augmented generation. In *Proceedings of the ACM Web Conference 2024*, pp. 1330–1340, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jirong Yang, Peizhe Liu, Chaojie Zhang, and Jovan Stojkovic. Architectural implications of agentic ai workflows. *arXiv preprint arXiv:2608.04458*, 2026.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. Cacheblend: Fast large language model serving for rag with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 94–109, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. Rankrag: Unifying context ranking with retrieval-augmented generation in llms. *Advances in Neural Information Processing Systems*, 37:121156–121184, 2024.

Jiarui Zhang, Xiangyu Liu, Yong Hu, Chaoyue Niu, Fan Wu, and Guihai Chen. Ragrouter: Learning to route queries to multiple retrieval-augmented language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

A Appendix: Agentic Design Space Pareto Frontiers

A.1 Detailed Pareto Frontiers

In Figure A we present a full accuracy versus serving cost across all benchmarks and agentic system architectures we tested. We evaluate seven agentic architectures in total: Multi-Turn LLM, which repeatedly invokes the same LLM; Selector, which dynamically selects among agents; Orchestrator, which coordinates multiple agents; ReAct, which interleaves LLM reasoning with tool execution; RAG, which augments LLM inference with retrieval; Debate, which runs multiple agents that critique and refine responses; and Parallel, which executes multiple agentic workflows concurrently. Point colors denote the agentic system, while marker shapes denote the benchmark (GAIA L1–L3, HumanEval, LongBench v2, SWE-bench Lite, TriviaQA, MMLU, and NQ). A list of exactly which benchmarks we tested on which agentic systems can be seen in 1

Table 1: Agentic systems evaluated on each benchmark suite.

System	GAIA	HumanEval	LongBench	SWE-bench	TriviaQA	MMLU	NQ
Multi-Turn LLM	✓	✓	✓	✓			
Selector	✓	✓	✓				
Orchestrator	✓		✓	✓			
ReAct	✓		✓				
Parallel	✓		✓				
Debate		✓	✓				
RAG		✓			✓	✓	✓

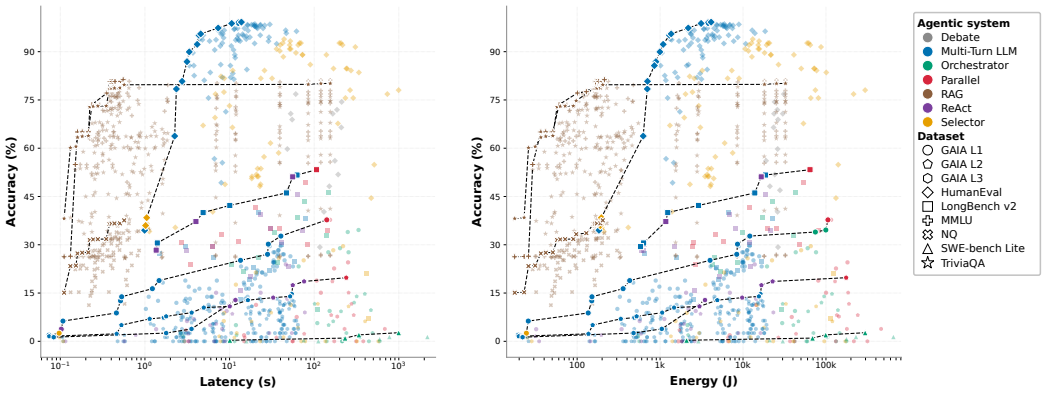


Figure 7: We plot all agentic configurations across all nine benchmark/level combinations (GAIA L1–L3, HumanEval, LongBench v2, SWE-bench Lite, TriviaQA, MMLU, NQ). Across all points, latency spans roughly 0.07 s to 2,180 s (4.5 orders of magnitude) and energy spans roughly 18 J to 651,000 J, while accuracy ranges from 0% to 99.2% depending on dataset. The hardest datasets are the GAIA levels: best achievable accuracy tops out at 37.7% (GAIA L1, Parallel system), 19.8% (GAIA L2, Parallel), and 12.8% (GAIA L3, ReAct), and SWE-bench Lite peaks at just 2.7% (Orchestrator/magenticone). The easiest datasets are HumanEval, which reaches 99.2% accuracy (Multi-Turn LLM, at 13.8 s / 4,126 J), and the RAG datasets, which top out at 80–81% (TriviaQA and MMLU, both via the RAG system).

B Appendix: Additional Methodology Details

Table 2 summarizes the agentic systems evaluated in this work and the progressively more complex design dimensions they incorporate.

Table 2: Agentic systems spanning progressively more complex design dimensions.

Agentic System	Multi-Turn	Tool Calling	Dynamic Orchestration	Parallelism
Multi-Turn AgentChat	✓	×	×	×
RAG	×	✓	×	×
ReAct	✓	✓	×	×
MagenticOne	✓	✓	✓	×
SelectorGroupChat	✓	✓	✓	×
Multi-Agent Debate	✓	×	×	✓
Parallel MagenticOne	✓	✓	✓	✓

Figure 8 shows the agentic workflows evaluated in our benchmarks.

- **Multi-Agent Debate** Four debaters independently propose answers and then exchange arguments with their neighbors. During three discussion rounds, they critique received arguments and revise their proposals. Once the rounds are complete, a separate judge examines the debaters' final proposals and synthesizes a single answer.
- **Multi-Turn AgentChat** tries a sequence of agents on the same question. After each turn, the benchmark compares the agent's final answer with the reference answer; a correct answer ends the run, while an incorrect answer triggers the next turn.We study two information-sharing variants: independent retries give each turn the original question without earlier answers, whereas shared-context retries also provide earlier visible responses. Hidden reasoning from earlier models does not pass to the next model in the shared-context variant.
- **Parallel Agents** The same question is dispatched concurrently to three independent Magentic-One teams. Each team runs a complete orchestrated workflow with its own conversation and specialist agents, and the teams do not exchange information while solving the task. The implementation waits for all three teams to finish. An aggregator then reviews the candidate answers.
- **ReAct.** A single agent maintains the task context and decides at each step whether it can answer or needs information from an external tool. Available actions include web search, fetching a webpage, reading or listing files, and executing code. When the agent calls a tool, the tool's result becomes an observation in the agent's context; the agent then can make another decision, invoke another tool, or return its final answer.
- **Selector Group Chat.** A selector model examines the shared conversation and chooses which specialist should be assigned next: the coder, terminal executor, file surfer, or web surfer. The interaction ends when a judge decides to terminate or the message limit is reached. A separate finalizer call then reads the conversation and formats the final answer.
- **Magentic-One.** A dedicated orchestrator managed a team containing a coder, terminal executor, file surfer, and web surfer. Rather than using a generic selector to choose the next speaker from the conversation, the orchestrator plans how to approach the work to specialists, reviews their results, and can revise its plan before execution. This process repeats until the task is completed or the limit is reached, after which the workflow produces a final answer.

Parallel Agents launches three independent **Magentic-One** teams and aggregates their outputs. **ReAct** uses a single agent that alternates between the tool calls. **Selector Group Chat** dynamically chooses the next agent and stops at the correct answer acquired, whereas

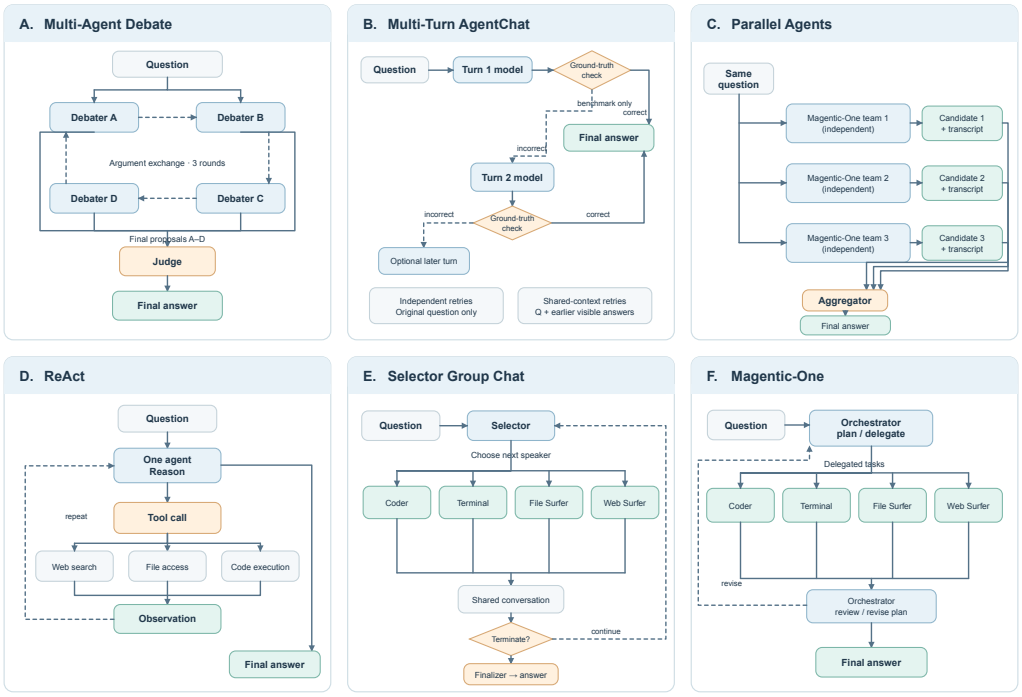


Figure 8: Execution flow of the six agentic design patterns evaluated in this study: (A) Multi-Agent Debate, (B) Multi-Turn AgentChat, (C) Parallel Agents, (D) ReAct, (E) Selector Group Chat, and (F) Magentic-One. Solid arrows denote control flow, while dashed arrows denote iterative or feedback paths. Blue boxes denote agents or models, green boxes denote workers or outputs, and orange boxes denote decisions, aggregation, or tool calls.

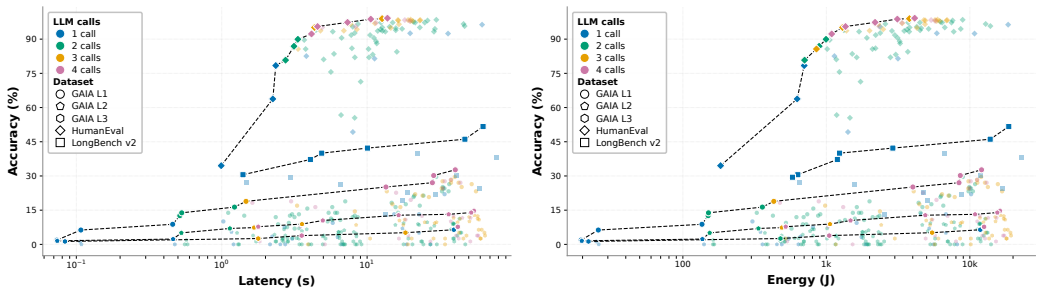


Figure 9: Accuracy versus latency/energy pareto frontiers for 356 Multi-Turn LLM configurations across HumanEval, GAIA L1–L3, and LongBench v2. Cost spans 0.07-78.2 s and 20 J-22.9 kJ, while accuracy varies substantially across datasets. Across the latency frontiers, 16 of the 48 points are single-call, 8 are two-call, 7 are three-call, and 17 are four-call. The energy frontiers have almost the same split (17, 9, 7, and 17 of 50). Single-call points sit at the low-cost, low-accuracy end, except on LongBench v2, where every frontier point is single-call, and 2–3 call cascades fill the middle. Four-call cascades hold the top-accuracy point on every GAIA and HumanEval frontier (32.7% on L1, 14.7% on L2, 7.7% on L3, and 99.2% on HumanEval).

C Appendix: Multi-Turn Pareto Frontier

In Figure C we plot accuracy versus cost for the Multi-Turn LLM system, with latency (left) and energy (right) shown on log-scaled x-axes. Color denotes the number of LLM calls: one-call configurations use a single Qwen3 model, while 2–4-call configurations use a cascade of different models.

D Appendix: Retrieval-Augmented Generation Design Space Exploration

D.1 RAG Design Space and Configuration Parameters

Vector Datastores. Although conceptually straightforward, implementing and effectively using vector databases with large language models often entails a complex design-space exploration challenge, as evidenced by the diverse set of configurable parameters in Table 3. Usage of these databases for RAG systems can traditionally be divided into two stages: an offline stage, in which vectors are carefully organized into data structures optimized for fast and efficient search, and an online stage, where search-time parameters are configured to query the index effectively.

D.1.1 Offline Index Creation

The offline stage centers on constructing a vector index that balances retrieval accuracy, memory footprint, and query efficiency. A primary design choice in this stage is the selection of the index structure itself. State-of-the-art vector index formats include:

- **Inverted File (IVF) indices:** IVF-based indices partition the vector space into a fixed number of coarse clusters, controlled by the number of lists (`nlist`). At query time, search is restricted to a subset of these clusters, substantially reducing the number of distance computations. Adjusting `nlist` changes the granularity of the partitioning, with larger values producing finer clusters. However, adjustments to cluster size introduce trade-offs that must be tuned in coordination with runtime search parameters that trade off retrieval efficiency and accuracy.
- **Hierarchical Navigable Small World (HNSW) graphs:** HNSW indices are multi-layer proximity graphs that enable efficient approximate nearest neighbor search via graph traversal. Construction-time parameters such as `M` (graph edges) and `efConstruction` (nearest neighbors explored) determine the connectivity of the graph by controlling the number of candidate neighbors considered when inserting each vector. Higher values of `M` and

`efConstruction` improve retrieval accuracy at the cost of increased index build time and memory usage.

While HNSW indices offer superior search performance and higher recall, we primarily focus on IVF indices in this work due to the limitations of HNSW at scale. Specifically, HNSW presents substantial challenges regarding memory overhead and dynamic data management. Although the vectors within an HNSW index can be compressed via quantization techniques such as product quantization (PQ), optimized product quantization (OPQ), or scalar quantization (SQ), the graph-based links that connect the vectors remain uncompressed. This creates a disproportionate memory footprint.

For example, for a 100-million-document index using SQ8 quantization, the links account for approximately 92 GB of the total volume, while the vectors themselves occupy only approximately 72 GB. In contrast, an IVF index of the same scale and quantization remains significantly leaner at approximately 73 GB. Since both index types suffer order-of-magnitude performance degradation when queried from disk, the memory overhead of HNSW makes such indices impractical at trillion-token scale. Finally, while HNSW supports additive growth, the removal of vectors is computationally intensive due to the need to preserve graph connectivity. IVF indices do not share these structural rigidities, making them more practical for large-scale, dynamic deployments.

D.1.2 Online Retrieval

Once the indices have been constructed, search-time parameters can be configured to dictate how queries are used to search the pre-built index at runtime. In IVF-based indices, a key parameter is the number of probed clusters (`nprobe`), which determines how many coarse partitions are searched for each query and directly controls the trade-off between recall and latency. Similarly, for HNSW, parameters such as `efSearch` regulate the breadth of graph exploration during query time, influencing both retrieval quality and computational cost.

Online performance is also influenced by a range of system-level factors, including hardware characteristics (e.g., CPU versus GPU execution, memory bandwidth, and cache capacity) as well as software-level configurations, such as query batching strategies and the number of retrieved documents, which determines the amount of external context provided to the language model.

D.1.3 RAG Inference

With the structures for efficient retrieval in place, inference-time retrieved context is augmented to the user query to provide additional context and improve response quality. However, incorporating retrieved documents can significantly increase prompt length, which leads to longer generation times and higher inference costs.

D.1.4 Advanced RAG Techniques

To further improve the inference efficiency of RAG systems, a broad set of techniques have been proposed by prior research to jointly enhance model accuracy and system-level performance. These methods target different stages of the RAG pipeline, spanning retrieval and generation, and in some cases introduce additional components such as verification or re-ranking to improve end-to-end reliability.

- **Retrieval Striding.** Iterative retrieval (Borgeaud et al., 2022) can improve accuracy (Ram et al., 2023), but even a single additional retrieval round often incurs performance costs that outweigh its benefits in practice, despite recent efforts to mitigate overhead via caching (Jin et al., 2025a; Yao et al., 2025) or retrieval-inference pipelining (Jiang et al., 2024).
- **Enhanced RAG Pipelines.** Many RAG systems incorporate auxiliary stages such as query rewriting (Ma et al., 2023; Wang et al., 2025b), document validation (Martin et al., 2024), and reranking (Lyu et al., 2025; Yu et al., 2024; Xu et al., 2024), typically handled by lightweight secondary models to improve robustness at modest additional cost.
- **Sparse and Hybrid Retrieval.** Sparse (Robertson & Zaragoza, 2009) and hybrid retrieval methods (Sawarkar et al., 2024) complement dense embeddings by capturing exact or rare terms, but introduce additional complexity and overhead. Given the stronger ability of

Table 3: **Comprehensive RAG System Configuration Parameters**

Category	Parameter	Configurations
Index	Index Type	FAISS (Douze et al., 2024) (HNSW, IVF, Flat, LSH (Indyk & Motwani, 1998)), ScaNN (Guo et al., 2020), ANNOY
	Construction	nList (IVF), efConstruction (HNSW)
	Index Size	1M, 10M, 100M, 1B+ documents
	Quantization	Scalar Quantization (SQ8, SQ4), Product Quantization (PQ64, PQ256, PQ384)
Dataset	Data Source	Wikipedia, Common Crawl, MS MARCO (Nguyen et al., 2016)
	Total Tokens	1B, 100B, 1T (CompactDS)
Runtime	Batch Size	1, 8, 16, 32, 64
	Search Breadth	nProbe (IVF) $\in \{1, 32, 128\}$; efSearch (HNSW) $\in \{32, 64\}$
	Top-k Retrieval	$k \in \{1, 5, 10, 20, 50, 100\}$
Hardware	CPU Architecture	Intel Xeon Gold/Platinum (Ice Lake, Emerald Rapids), AMD EPYC (Genoa, Zen3), ARM Neoverse
	Parallelism	1–128 threads; hyper-threading (on/off)

dense vector representations to capture semantic similarity in queries, and the additional overheads introduced by hybrid approaches, we focus primarily on dense retrieval in this work.

- **LLM Selection and Routing.** Query-aware LLM routers can improve accuracy by dynamically selecting models (Chen et al.; Lu et al., 2025; Song et al., 2025; Zhang et al.), but these techniques often rely on additional models or expensive comparisons, limiting their practicality for performance-efficient RAG systems.

D.2 Comprehensive RAG Configuration Space

RAG engines encompass a myriad of design-time decisions that directly impact performance, energy, and accuracy tradeoffs and are challenging to alter once the system is deployed. Table 3 summarizes the major configuration parameters considered in RAG systems, spanning index construction, datasets, runtime retrieval, and hardware. Examples of such decisions include how data stores are embedded and indexed, how much context is retrieved, and which language model the system is paired with.

Figure 10(a) highlights the system-level trade-offs associated with different RAG design choices:

- **Retrieved Documents:** Increasing the number of retrieved documents improves accuracy, with gains of 6.3% from zero to one document and a further 5.1% from one to five, but increases end-to-end cost from 0.2s and 35.9J without retrieval to 1.2s and 319J with 25 documents.
- **nProbe:** Increasing search effort from 32 to 512 increases retrieval latency from 0.08s to 1.01s and energy from 22.8J to 290J, while changing downstream accuracy by at most 0.49%.
- **Quantization:** Lower-precision representations can provide substantial efficiency benefits with limited accuracy loss; PQ64 is 7.8× faster and 9.9× more energy efficient than SQ4, despite only a 1.1% accuracy gap among the higher-accuracy configurations.
- **Datastore Size:** Larger datastores improve accuracy with diminishing returns: increasing from 500M to 1B passages improves accuracy by only 0.5%, while increasing latency and energy to 154s and 30kJ, compared with 1.52s and 319J for 10M passages.
- **LLM Size:** The benefit of retrieved information varies with backend LLM size. For example, increasing from Qwen3/0.6B to Qwen3/14B reduces the accuracy gap between PQ64 and the higher-precision configurations from approximately 10% to 2%, while larger models can also substantially improve overall RAG accuracy.

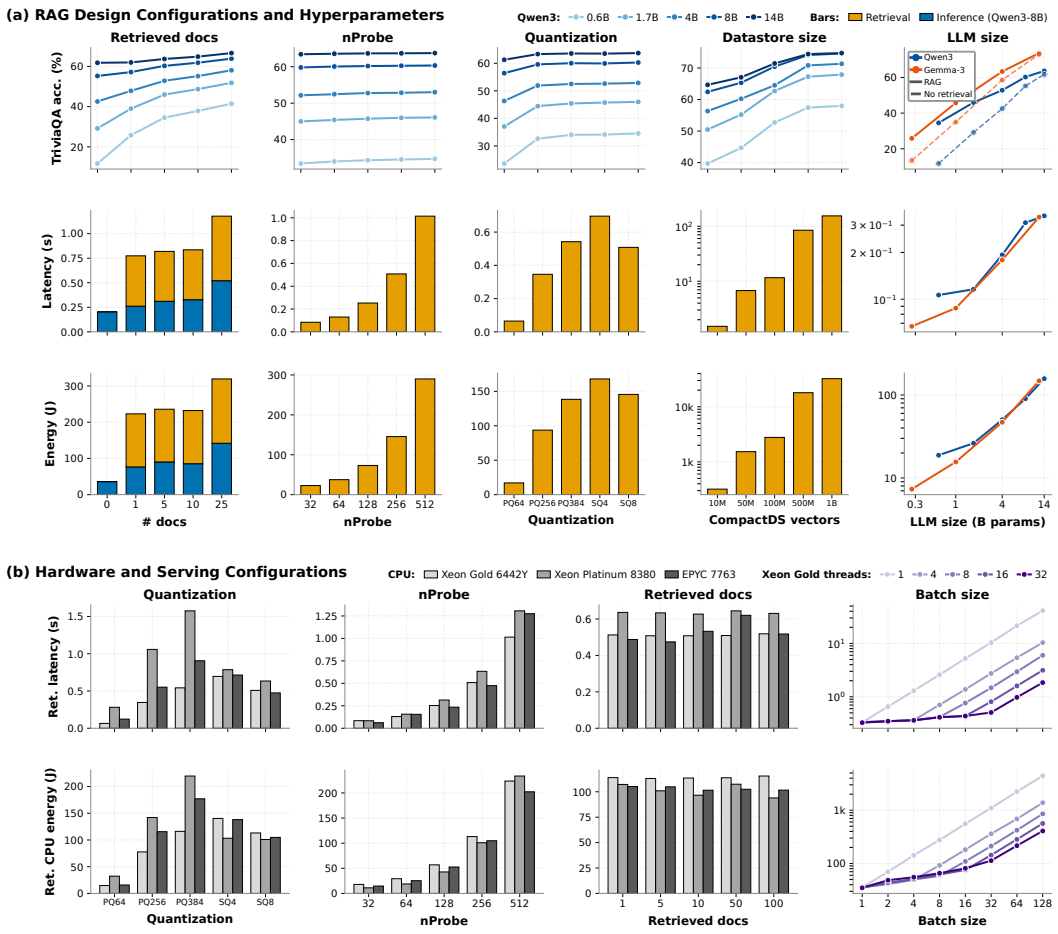


Figure 10: (a) Changing only the quantization shifts retrieval from 17% of end-to-end latency with PQ64 to 69% with SQ4, while varying `nProbe` shifts it from 21% at 32 to 77% at 512, using Qwen3-8B with 5 retrieved documents. These configuration choices produce more than a 10× variation in retrieval cost. (b) the best hardware for a given workload also depends on the hyperparameters: the Xeon Gold is fastest and most energy-efficient for PQ quantization (e.g., 2.9× faster than the Xeon Platinum at PQ384), but the Xeon Platinum uses the least energy for SQ4 and SQ8, and the EPYC is fastest at nProbe 32 while the Xeon Gold is fastest at nProbe 512.

Together, these results demonstrate that RAG performance is shaped by the interaction between retrieval depth, search effort, index representation, datastore scale, and model and workload characteristics, rather than by any single design parameter.

Figure 10(b) shows how RAG performance varies across hardware platforms. The best-performing quantization scheme differs across CPUs: PQ256 achieves 0.34s and 77J on the Intel Gold 6442Y, while SQ8 achieves 0.63s/101J on the Intel Platinum 8380 and 0.47s/105J on the AMD EPYC 7763. Across hardware, the same retrieval workload can incur up to 34% higher latency and 38% higher energy consumption. Batch size and thread count further shift these trade-offs, highlighting that RAG configurations should be optimized jointly with the underlying hardware.

E Appendix: Characterizing Agentic Skew and Hardware-Aware Execution

In this section, we characterize the workload skew in agentic systems and its implications for hardware-aware execution. Agentic workflows devote a substantial fraction of their LLM calls to

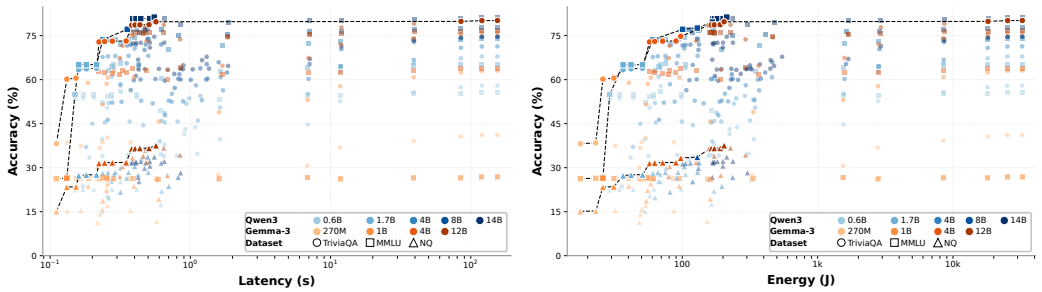


Figure 11: Accuracy Pareto fronts for RAG across reasoning (Qwen3) and non-reasoning (Gemma3) LLMs on TriviaQA, Natural Questions, and MMLU. Across all 432 RAG configurations, retrieval accounts for more than half of total latency and energy in 64% and 65% of configurations, respectively. This trend is particularly pronounced on TriviaQA (68%/68% latency/energy) and MMLU (69%/71%), but less so on NQ (44%/47%). Gemma3 is more retrieval-bound than Qwen3 (69%/73% vs. 61%/60%), as its smaller models reduce the cost of LLM inference.

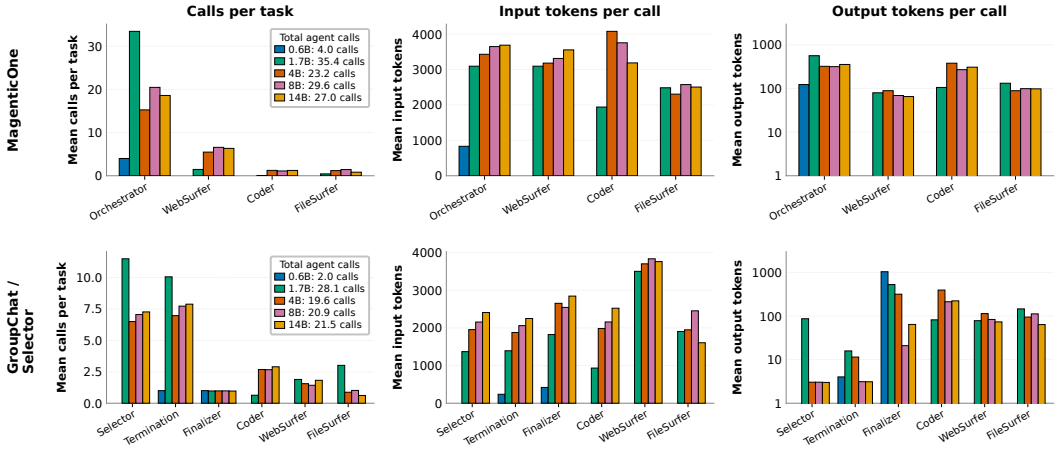


Figure 12: Per-agent workload of MagenticOne and GroupChat/Selector on GAIA Level 1 for Qwen3 0.6B–14B (non-thinking): mean LLM calls per task (legend: total calls per model), and mean input/output tokens per call. Most calls are spent on orchestration: at 8B, the orchestrator accounts for 20.5 of MagenticOne's 29.6 calls, while the Selector and Termination agents account for 14.8 of GroupChat/Selector's 20.9 calls, each producing only $\sim$3 output tokens. Total calls do not increase with model size; for example, MagenticOne averages 35.4 calls with 1.7B but only 29.6 with 8B.

lightweight orchestration, creating significant variation in work across agents. This skew also creates opportunities for heterogeneous hardware: assigning slower but more energy-efficient GPUs to agents with sufficient slack can reduce idle time and energy without extending the critical path. Across the broader configuration space, mixed hardware assignments frequently improve the energy frontier, while latency remains dominated by the faster GPU.

Figure 12 characterizes the sources of workload skew across agent roles, showing that a substantial fraction of LLM calls are concentrated in lightweight orchestration. Figure 13 shows how this skew can be exploited through hardware-aware execution: assigning slower but more energy-efficient L4 GPUs to agents with sufficient slack reduces both synchronization idle time and total energy without extending the critical path. Finally, Figure 14 generalizes this analysis across agentic configurations, showing that heterogeneous hardware assignments frequently appear on the energy Pareto frontier, while latency-optimal configurations remain dominated by the faster A6000 Ada.

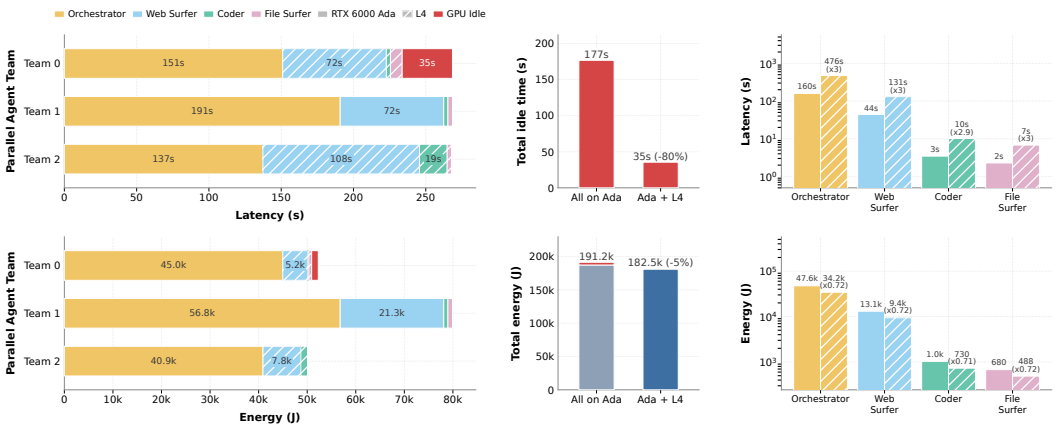
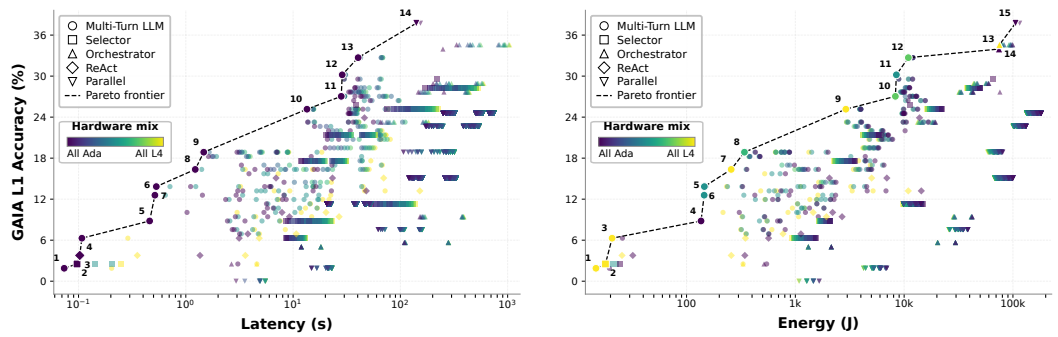


Figure 13: **Left**: Per-team latency and energy balance the across the teams of GAIA L1 parallel Magentic-One by mixing A6000 Ada and L4 GPUs. Solid segments run on Ada, hatched segments on L4, and red marks waiting time and idle energy. Exhaustive search identifies the optimal assignment, moving six agents from Teams 0 and 2 to L4 while keeping the slowest Team 1 entirely on Ada. **Middle**: System-wide idle time drops from 177 s to 35 s (80%), while total energy drops from 191.2 to 182.5 kJ (5%). **Right**: L4 execution is $\approx 3\times$ slower but uses $\approx 0.72\times$ the energy per role.



Latency Pareto frontier

Point	1	2	3	4	5	6	7	8	9	10	11	12	13	14
System	1-turn LLM	Selector	ReAct	1-turn LLM	1-turn LLM	2-turn LLM	2-turn LLM	2-turn LLM	3-turn LLM	4-turn LLM	4-turn LLM	4-turn LLM	4-turn LLM	Parallel
Configuration	0.6B nt Ada	0.6B nt Ada: all 2	0.6B nt Ada	1.7B nt Ada	14B nt Ada	0.6B nt: Ada 14B nt: Ada	1.7B nt: Ada 14B nt: Ada	1.7B nt: Ada 4B nt: Ada	1.7B nt: Ada 4B nt: Ada 14B nt: Ada	0.6B nt: Ada 1.7B nt: Ada 4B nt: Ada 4B t: Ada	0.6B nt: Ada 1.7B nt: Ada 4B nt: Ada 14B t: Ada	1.7B nt: Ada 4B nt: Ada 14B nt: Ada 14B t: Ada	1.7B nt: Ada 4B nt: Ada 4B t: Ada 14B t: Ada	14B nt Ada: all 13
Accuracy (%)	1.9	2.5	3.8	6.3	8.8	12.6	13.8	16.4	18.9	25.2	27.0	30.2	32.7	37.7
Latency (s)	0.07	0.10	0.10	0.11	0.46	0.52	0.53	1.22	1.47	13.5	28.1	28.7	40.4	140.9

Energy Pareto frontier

Point	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
System	1-turn LLM	Selector	1-turn LLM	1-turn LLM	2-turn LLM	2-turn LLM	2-turn LLM	3-turn LLM	4-turn LLM	4-turn LLM	4-turn LLM	4-turn LLM	Orchestrator	Orchestrator	Parallel
Configuration	0.6B nt L4	0.6B nt L4: all 2	1.7B nt L4	14B nt Ada	0.6B nt: L4 14B nt: Ada	1.7B nt: L4 14B nt: Ada	1.7B nt: L4 4B nt: L4	1.7B nt: L4 4B nt: L4 14B nt: Ada	0.6B nt: L4 1.7B nt: L4 4B nt: L4 4B t: L4	0.6B nt: L4 1.7B nt: L4 4B nt: L4 14B t: Ada	1.7B nt: L4 4B nt: L4 14B nt: Ada 14B t: Ada	1.7B nt: L4 4B nt: L4 4B t: L4 14B t: Ada	14B t Ada: all 4	8B t L4: all 4	14B nt Ada: all 13
Accuracy (%)	1.9	2.5	6.3	8.8	12.6	13.8	16.4	18.9	25.2	27.0	30.2	32.7	34.0	34.6	37.7
Energy (J)	14.7	18.0	20.7	136	145	146	257	341	2.9k	8.3k	8.5k	10.9k	74.5k	75.2k	105.7k

Figure 14: Each Mixed GPU configuration is expanded into every assignment of its LLM-calling agents (or turns) to an A6000 Ada or an L4, for 26,889 hardware configuration over 124 agentic system designs. Only the 113 configurations that use Qwen3-0.6B, 1.7B, 4B, or 8B can move agents to the L4, since L4 measurements are complete only for those models. Of the points, 124 are all-Ada, 75 are all-L4, and 26,690 are mixed. Mixed hardware appears on the energy frontier but not the latency frontier: 12 of the 15 energy-optimal points place at least one agent on an L4 (6 all-L4 and 6 mixed), whereas all 14 latency-optimal points run entirely on the Ada because the L4 is about $3\times$ slower. Compared with the all-Ada energy frontier, using the L4 cuts energy by 1.2–1.4$\times$ at roughly half of frontier points (up to 1.4$\times$ at 16% and 25% accuracy) and by 1.3$\times$ at 35% accuracy, while the remaining points improve by only 1.01–1.10$\times$.