DEPTH OVER BREADTH: ENERGY-EFFICIENT SCALING OF MULTI-AGENT LLM SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deploying a multi-agent LLM system (MAS) means choosing how far to scale three axes: per-agent tool-calling depth K, agent count N, and communication rounds R. Each choice buys accuracy and burns energy, yet prior scaling studies price compute in tokens or dollars on closed API models where energy cannot be measured, and tokens misprice it: a decode token costs 335$\times$ a prefill token at short context under our setup. We measure hardware-level GPU energy at every LLM call across four topologies, four agentic benchmarks, and three locally served models (9B to 35B, the larger two on SWE-bench Lite only). We find that test-time scaling axes are not interchangeable: tool-calling depth (K) converts energy into accuracy most efficiently; agent breadth (N) helps less and costs energy linear in the number of agents; and communication rounds (R) buy no measurable accuracy once per-agent iterative depth is held constant (for homogeneous agents), while having steep energy cost.

Based on our findings of this depth-over-breadth ordering, we develop a label-free configuration search-and-deploy policy: binary-search K on a single agent to the deepest budget at which agents still utilize at least half of it, then deploy the target MAS topology at that budget with three agents. Against a two-phase depth-then-breadth scaling policy, this cuts search-and-deploy energy by 59% on average while giving up at most half a point of accuracy on three of four benchmarks; and against running the same label-free rule on the MAS topology instead of a single agent, our policy saves 51% at comparable accuracy.

1 INTRODUCTION

Deploying a multi-agent LLM system (MAS) requires three decisions before it serves a single task: whether to run one at all, which topology to run, and how far to scale it. Each choice affects both the accuracy the system reaches and the energy it burns on every task thereafter, and searching for the right configuration burns energy of its own. A MAS deploys multiple LLM instances in concert, each running its own tool-calling loop, and combines their work through a fixed communication structure (Wu et al., 2024; Hong et al., 2024; Li et al., 2023; Qian et al., 2024a; Kim et al., 2026). It spends far more compute than a single agent along three axes (Figure 1): deeper tool-calling loops (depth K), more agents (N), and more rounds of communication between them (R).

Prior scaling work prices compute in tokens, calls, or dollars (Kim et al., 2026; Qian et al., 2024b; Zhang et al., 2025a), on closed API models where energy cannot be measured. Tokens are an inaccurate stand-in for energy, because MAS change both the volume and the type of tokens generated. Inference runs in two phases with different per-token energy: prefill, which is compute-bound and consumes the prompt, and decode, which is memory-bandwidth-bound and generates the answer at a much higher energy rate (Ruf & Detyniecki, 2026; Niu et al., 2026; Husom et al., 2024). Under our prefix-cached serving, a decode token costs roughly 335 times a prefill token at short context (§ A.1). Communication is prefill-heavy and context-heavy, since it resends and re-reads transcripts, while depth is decode-heavy, since it generates more. Two configurations with the same total token count can therefore have different energy costs, depending on how their tokens split between the two phases. The energy cost of these choices has not been measured, and what each axis buys in accuracy remains disputed, most sharply for added communication (§ 2).

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.

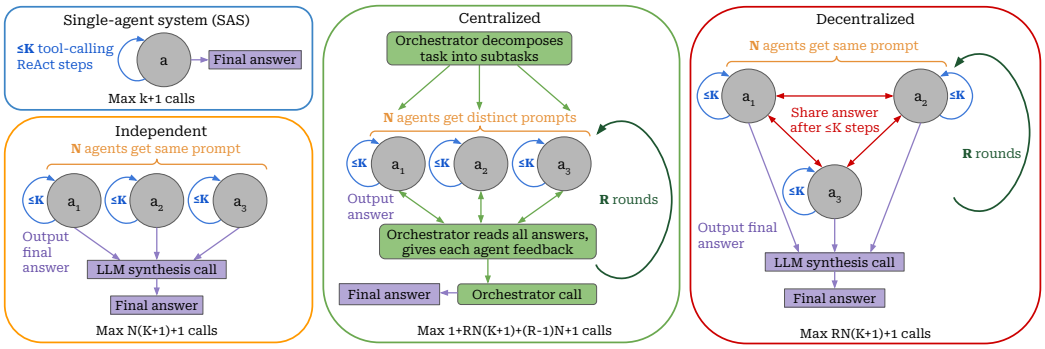


Figure 1: **The four topologies and the three scaling axes.** Each agent runs a ReAct tool-calling loop of at most K steps (depth, blue); N agents run in parallel (breadth; SAS is the N=1 limit; the MAS panels are drawn at the default N=3); Centralized and Decentralized iterate R rounds of communication (red). Maximum LLM calls are shown beneath each panel, and border colors are the topology colors used in all result figures. Swept ranges are in § 3.1; appendix § B.1 states the four protocols in full.

In this work, we measure MAS energy directly, from GPU hardware counters resolved per LLM call. We run experiments spanning four MAS topologies, the three scaling axes, and four agentic benchmarks, on a locally served 9B model and two larger models. We study the relationship between energy and performance; each axis (K, N, R) moves both, and we analyze how efficiently each one converts joules into accuracy. From that analysis, we develop a label-free, energy-efficient search-and-deploy policy that finds a locally optimal MAS configuration.

In sum, we address three research questions around scaling in MAS. **(i) How energy-efficient is each scaling axis (§ 4.1, § 4.2, § 4.4)?** We show that per-agent depth converts energy into accuracy far more efficiently than agent count, whose gains come from ensembling and cost energy linear in the number of agents. **(ii) Do communication rounds improve MAS performance (§ 4.3)?** We show that rounds do not convert energy into accuracy once we hold per-agent iterative depth fixed (for homogeneous agents), while their energy cost climbs steeply. **(iii) What is an efficient search-and-deploy policy for MAS configuration (§ 5)?** We show that the percentage of tool-calling budget used by agents is an effective label-free signal for tuning MAS parameters. We develop a search-and-deploy policy that binary-searches the tool-calling budget on a single agent to the deepest budget agents still half-use; this spends 59% less energy on average than a search that also scales breadth, while giving up at most half a point of accuracy on three of four benchmarks. Against running the same label-free rule on the MAS topology instead of a single agent, it saves 51% energy at comparable accuracy. Supporting energy analyses are in appendix § A.

2 Related Work

Agent-count scaling and ensembling. Kim et al. (2026) measure MAS topologies across nine frontier closed API models and six agentic benchmarks. We adopt four of their topologies and three of their benchmarks, but where they vary model and topology at fixed axes, we fix the model and scale the axes. They find coordination helps weaker models; because they match maximum total iterations but not total tokens, some of the gain attributed to coordination may be the extra tokens coordination spends. Other literature shows that accuracy in agent count follows a saturating curve, whether from coordination (Qian et al., 2024b) or from sampling and voting alone (Li et al., 2024); single agents match MAS at equal token budgets (Tran & Kiela, 2026); and the single-agent analogue is a log-linear coverage law in repeated samples (Brown et al., 2024). These establish that breadth gains are real but come from ensembling, and that redundancy bounds them (§ 4.4). No prior work prices axes in energy.

Reasoning and tool-calling depth. For pure reasoning, longer chains raise accuracy up to a point (Snell et al., 2024; Wu et al., 2026), beyond which extra steps hurt (Gema et al., 2025), and self-correction helps only with an external signal (Huang et al., 2024; Stechly et al., 2025); for tool-using agents, whose depth axis is the number of ReAct steps (Yao et al., 2023), budget-scaling studies find the same diminishing returns (Zhu et al., 2025; McCleary & Ghawaly, 2026; Liu et al., 2025; Sethi,

2026). But these examine single agents and measure tokens or tool calls; we sweep depth inside each fixed MAS topology, price all three axes in energy, and find depth the most efficient.

Communication rounds. Du et al. (2024) introduced multi-round debate and found it improves accuracy; however, under their setup, each round also adds reasoning depth, confounding communication with increased total depth (Chen et al., 2026). Several studies report rounds null or harmful when each configuration runs at whatever compute it naturally consumes, with no budget held equal across conditions (Smit et al., 2024; Kaesberg et al., 2025; Wynn et al., 2025). Choi et al. (2025) separate the components directly, showing pre-debate voting accounts for most of debate's gain, and prove a martingale over belief trajectories explaining why; we extend it to tool-using agents at matched depth in § 4.3. Studies that hold a budget fixed trade agents against rounds at a fixed product, confounding the round effect with agent count (Huang & Caragea, 2026; Huang et al., 2024; Yang et al., 2025; Wunderlich et al., 2026; Wang et al., 2024); in theory, that trade cannot even reduce total computation, since added agents save depth only by spending communication (Rizvi-Martel et al., 2026). No prior work isolates debate rounds at normalized total iterative depth, splitting the same per-agent depth across more rounds. We both fill this gap and price rounds in energy.

Energy measurement and MAS optimization. On the cost side, measurement studies establish that decode tokens can cost orders of magnitude more energy than prefill unbatched (Fernandez et al., 2025; Niu et al., 2026; Ruf & Detyniecki, 2026; Husom et al., 2024). Two studies measure agentic energy itself: SWEnergy (Tripathy et al., 2026) addresses single agents; Librarian (Cho et al., 2026) examines orchestrator-delegation MAS on SWE-bench Verified and finds MAS draw more GPU energy than a single agent even at lower token counts. Neither sweeps a scaling axis. A separate line searches MAS structure (Zhuge et al., 2024; Liu et al., 2023; Zhang et al., 2025b; 2024; 2025a; Zhou et al., 2026), always against tokens, dollars, calls, or latency; none price the search itself in energy. Our approach (§ 5) measures search and deployment cost in energy and tunes scale on an already-fixed topology, so it composes with these methods rather than replacing them.

3 Experimental setup

We design experiments to price each scaling axis in energy. With everything else held fixed (model, tasks, serving stack), we sweep one axis at a time and measure, call by call, what each added unit of depth, breadth, or communication costs in joules and buys in accuracy. The subsections below define the axes, benchmarks, models, and measurement; in § 4, we report the sweeps, and in § 5, we turn their ordering into a deployment policy.

3.1 Scaling axes and topology protocols

Figure 1 illustrates the four topologies and the three axes. We sweep one axis at a time, holding the others at defaults: $N = 3$ and $R = 2$ where the topology has them, and K set so each agent's total step budget $R \times K$ is 20. Depth: $K \in \{1, 2, 3, 5, 7, 10, 15, 20, 30, 50\}$, extended to $\{100, 200\}$ on SWE-bench Lite. Breadth: $N \in \{1, 2, 3, 4, 5\}$, extended to $N = 10$ on SWE-bench Lite and BrowseComp+. Rounds (Centralized and Decentralized only): $R \in \{1, 2, 3, 5, 6, 10, 15, 30\}$ at matched per-agent iterative depth $R \times K = 30$, one ReAct loop of $K = 30/R$ steps per agent per round.

All agents share the same model, system-role framing, and tool APIs; only the communication structure differs. At each step of the ReAct loop (Yao et al., 2023), the model either emits tool calls, whose results are appended before it decides again, or emits none, which is its final answer, so K bounds tool-calling steps exactly. SAS and all orchestrator/synthesizer calls run at temperature 0; Independent agents sample at 0.7, Centralized workers and Decentralized debaters at 0.5 (temperature is a weak lever for final-answer diversity in ensembles (Zhu et al., 2026; Li et al., 2024)).

3.2 Benchmarks and tool environments

All four benchmarks are agentic in the sense of Kim et al. (2026), meaning the task cannot be answered from parametric knowledge in one shot, so the model must act across a multi-step

tool-calling trajectory. Agentic tasks give coordination genuine work to divide, and their trajectories accumulate context, so the steep decode-prefill cost asymmetry compounds against a context that is itself still growing. Every deployment is fully local (datasets, corpora, tools, sandboxes, scoring), as remote retrieval or judging API would inject unmeasured compute and make per-task energy unaccountable. **FanOutQA** (Zhu et al., 2024) is fan-out multi-hop QA over Wikipedia, each question decomposing into independent per-entity lookups, scored by `loose_acc`. **WorkBench** (Styles et al., 2024) is a simulated business workspace (690 tasks over 5 databases, 27 tools), scored `correct` on the gold action set, matching Kim et al. **BrowseComp+** (Chen et al., 2025) poses fact-seeking questions over a fixed 100K-document corpus (830 questions), scored by normalized string match. **SWE-bench Lite** (the 300-instance curated split of Jimenez et al., 2024) takes real GitHub issues and requires a repository patch, scored by the official harness (`real_correct`). K-sweep cells use n = 50 tasks; N- and R-sweep cells use n = 100 (the same 50 plus 50 more), doubling the sample where the scaled range on the independent variable is smaller. All cells of a sweep share one fixed task set. Tool environments and full deployment detail are in appendix § B.

3.3 Models and serving configuration

The primary model is Qwen3.5-9B (dense, BF16, non-thinking), which runs the grid on one exclusively allocated NVIDIA RTX A5000 24 GB (three cells excepted, § B.2). Two secondary models replicate the SWE-bench Lite scaling experiments on an A6000 48 GB: Qwen3.5-35B-A3B (MoE, same family) and Gemma 4-31B (dense, different family); deployment details are in appendix § B.2. All models run on one SGLang server instance, with requests strictly serial and non-streaming at batch size 1, which clean per-call attribution requires (batch-serving experiments in appendix § A.2).

3.4 Energy and token measurement

We read the GPU's hardware energy counter (NVML, millijoule resolution) before and after every LLM call and tool execution; since requests run serially on a GPU we hold exclusively (§ 3.3), the difference between two readings is the energy of exactly one call. We subtract idle power (from a 10 s quiet window after warmup) times wall time, floored at zero (Iyengar et al., 2026), and sum per task to get GPU dynamic energy. Prompt (P) and completion (C) token counts come from the serving engine; $T = P + C$.

4 Scaling experiment results

4.1 K-sweep

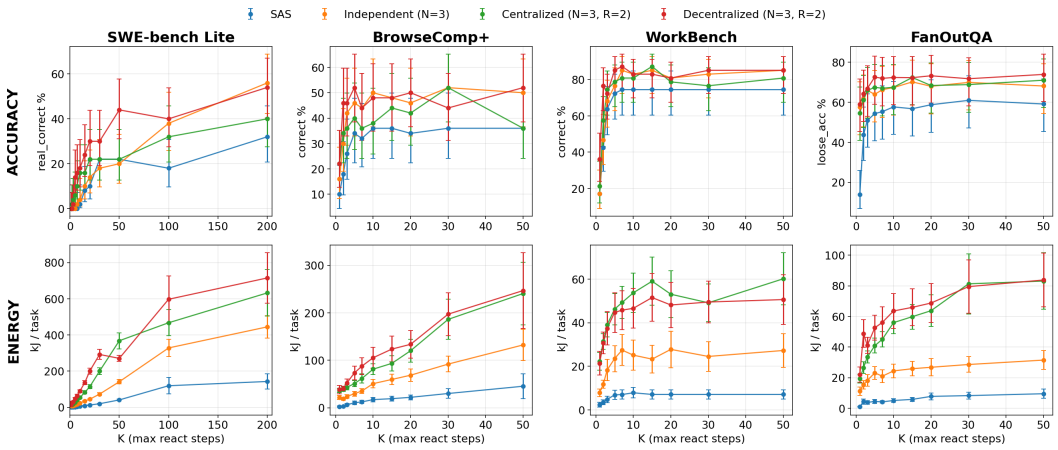


Figure 2: **Accuracy and energy vs. tool-calling steps (K).** K sweeps 1-200 on SWE-bench Lite and 1-50 on other benchmarks; each MAS topology runs $N = 3$. The K-sweep does not normalize total step budget: Centralized and Decentralized run $R = 2$ at every K, so their agents may consume up to $2K$ steps where SAS and Independent consume K (the N- and R-sweeps, Figures 3 and 4, do match budgets). Each point is the mean across the same n=50 tasks. Error bars: 95% Wilson intervals on accuracy, 1.96·SEM on energy.

Accuracy scales strongly with K but saturates quickly on the other three benchmarks (Figure 2); only SWE-bench Lite keeps scaling, and even it flattens past $K = 200$ (SAS extended to $K = 800$, Figure A7). All MAS topologies raise the accuracy ceiling above SAS. Energy growth, too, tapers as K scales: K is only a ceiling, and agents increasingly stop short of it, leaving budget unused (Figure A6). Tool-calling budget utilization varies sharply by benchmark (SAS at $K = 50$: 94% on SWE-bench Lite, 8-53% on the prose benchmarks; Figure A6), which is the signal § 5's policy search uses.

4.2 N-SWEEP

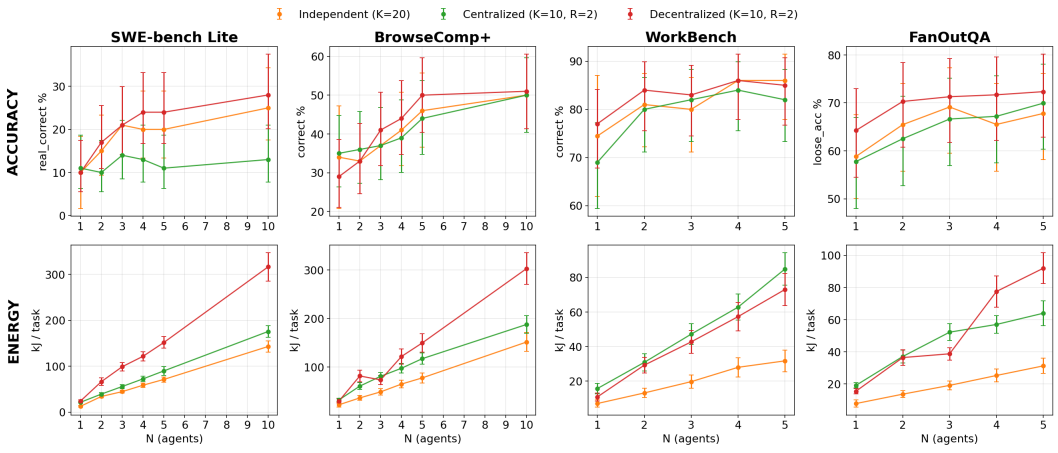


Figure 3: **Accuracy and energy vs. agent count (N).** N sweeps 1-10 on SWE-bench Lite and BrowseComp+ and 1-5 on FanOutQA and WorkBench; each topology runs at its matched per-agent budget (Indep $K = 20$; Cent/Decent $K = 10$, $R = 2$). The $N = 1$ point for Independent is SAS at $K = 20$, and is a true $N = 1$ run at $K = 10$, $R = 2$ for Centralized and Decentralized (orchestration and rounds kept, one worker). Each point is the mean across the same n=100 tasks. Error bars as in Figure 2.

Accuracy improves modestly with N from the ensembling effect (Figure 3), and shows diminishing returns as N scales further. Energy scales roughly linearly with N; compounding prefill cost does not make it superlinear here.

4.3 R-SWEEP

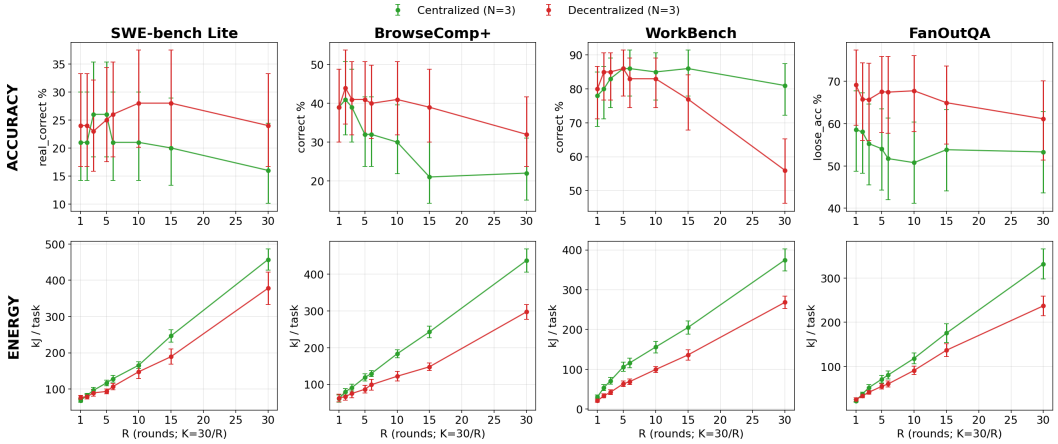


Figure 4: **Accuracy and energy vs. communication rounds (R).** R sweeps 1-30, and total K-budget is fixed at 30 such that per-round $K = 30/R$ (e.g. $R = 5$ has $K = 6$); only Centralized and Decentralized sweep R. Each point is the mean across the same n=100 tasks. Error bars as in Figure 2.

With per-agent iterative depth held at $R \times K = 30$, added rounds buy no accuracy over the range where the serving window is not binding ($R \leq 10$; Figure 4), while their energy climbs steeply throughout; the heavy decline at high R, largest on WorkBench, comes from aggressive truncation

of accumulated debate context, not from communication (numbers in appendix § C.1). Studies reporting that communication rounds aid accuracy allow depth to grow with rounds (§ 2): in an uncapped R-sweep with $K = 10$ held per round so total iterative depth grows with R (Figure A8), we too observe rounds "helping". This is the same pattern those studies report; our results do not conflict with them, but rather examine a different depth-normalized regime.

Why rounds do not help, even with agentic systems. Choi et al. (2025) prove debate among agents who update only on each other's messages is a martingale: expected belief in the correct answer does not move across rounds unless something outside the exchange biases updates toward the truth (Zhu et al., 2026). Their proof covers text-only multiple-choice agents; ours are tool-using, and tool returns might seem like private evidence that should break the martingale. However, since every agent runs the same model on the same tools, each complete run (queries, tool returns, answer) is an independent draw from one shared distribution. Writing ω for an agent's own run and ω' for an independent second run, a peer's message is some $f(\omega')$, and $(\omega,\ \text{peer message}) \overset{d}{=} (\omega,\ f(\omega'))$: the message reads as a summary of one more independent draw of the agent's own search. Conditioning on the channel is self-conditioning at extra cost. The identity is exact for Decentralized peers in round 1 and approximate otherwise (later rounds, Centralized subtasks).

This identity holds only while agents are homogeneous models. Thus, communication could carry real information where agents are heterogeneous in definition, data, or tools, as in layered aggregation over different models (Wang et al., 2025); it could also help where the task outgrows a single agent's context, which is the regime of Kim et al.'s largest coordination wins (Kim et al., 2026). Other studies support this boundary: replacing a peer's message with one from an unrelated task barely changes accuracy when agents hold the same information, but is costly when the receiver depends on the sender's evidence (Cheng et al., 2026; Zhang & Emu, 2026), and splitting a task's evidence across agents makes deliberation help (Li et al., 2026).

4.4 Cross-axis scaling efficiency

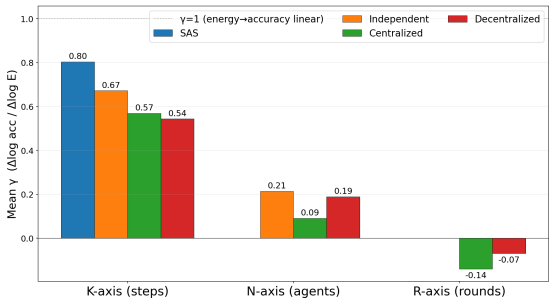


Figure 5: **Accuracy-vs-energy elasticity (γ) of each scaling axis.** Each bar is the accuracy–energy elasticity $\gamma = \Delta\log(\text{accuracy})/\Delta\log(E)$ of one axis for one topology: the slope of a log–log fit over that axis's cells within the common design ($K \leq 50$, $N \leq 5$, $R \leq 30$), fit per benchmark and averaged with equal weight over up to four benchmarks. Here, $\gamma \approx 1$ means energy converts to accuracy roughly linearly; $\gamma < 1$, diminishing; $\gamma < 0$, wasteful. Per-benchmark values: § C.2 ($R \leq 10$: Table A10).

Scaling K is the most energy-efficient axis, N also scales positively but less efficiently, and scaling R buys nothing (Figure 5). From the cross-benchmark mean, every axis shows diminishing returns ($\gamma < 1$).

K is our depth axis: each step conditions on a fresh tool return (§ 4.3), providing external evidence. Moreover, we find that energy grows more slowly than tokens as context grows with K, because marginal steps are mostly re-reading context, so a token-based analysis over-penalizes deep agents (Figure A5). N is our breadth axis, and the ensembling gains from increasing agent count match prior literature (Qian et al., 2024b; Li et al., 2024; Brown et al., 2024). The redundancy limit of Yang et al. (2026), where two genuinely diverse agents match sixteen identical ones, helps explain why our returns on N (γ 0.09-0.21) are weak compared to K (γ 0.54-0.80): identical copies sample one distribution, so each adds less.

4.5 Replication across model scale and architecture

We replicate these scaling experiments on SWE-bench Lite (which shows the strongest scaling with K) for two further models: a 35B mixture-of-experts (Qwen3.5-35B-A3B, ∼3B active per token) and Gemma 4-31B (dense, different family, conventional softmax attention).

The depth-over-breadth ordering and the communication-null hold across all three models on this benchmark. In Figure 6, the K column rises strongly in every topology for both additional models,

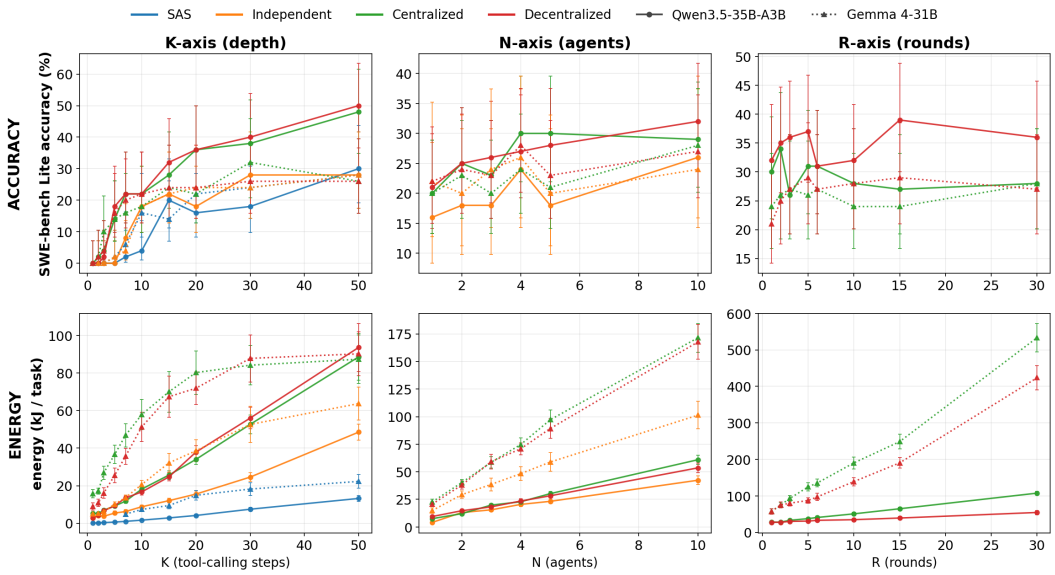


Figure 6: **Scaling on SWE-bench Lite across Qwen3.5-35B-A3B and Gemma 4-31B.** Accuracy and energy on SWE-bench Lite for Qwen3.5-35B-A3B (circle, solid) and Gemma 4-31B (triangle, dotted) on each scaling axis (columns: K, N, R). SAS appears only in the K column; only Centralized and Decentralized sweep R, at matched iterative depth $K \cdot R$=30. Each point is the mean across the same n=50 tasks on the K axis and n=100 tasks on the N and R axes. Error bars as in Figure 2.

the N column scales positively but less strongly, and the R column is flat-to-declining. Expressed as elasticities the same ordering is explicit (Figure A9). On energy, the MoE sits below both dense models in every cell and dominates the cross-model energy-accuracy frontier (Figure A10), dominating Qwen3.5-9B on the shared $K \leq 50$ cells; because the models differ in family and training as well as size and precision, we read this as showing the importance of model choice, not a controlled precision-versus-size result. The per-model energy mechanisms (active-parameter streaming, quantization, early termination) are in appendix § D, along with a check that the cross-model comparison is not card-confounded.

5 Configuration search-and-deploy policies

5.1 Search policies and parameter tuning

Given § 4's ordering (K most efficient, then N, R not at all), we design search policies that scale (i) depth, (ii) breadth, or (iii) depth then breadth. On the depth axis, we stop on utilization, the fraction of the step cap actually consumed: $u = \frac{1}{NR}\sum_{i,r} s_{i,r}/K$, where $s_{i,r}$ is the number of ReAct steps agent i took in round r, averaged over the probe tasks; $u = 1$ means every loop hit the cap. Utilization needs no labels, is low-variance (SE 0.015 to 0.074 at n=25), and falls monotonically in K (Figure A6), so binary search can find the threshold crossing. The labeled alternative (stop when the accuracy gain over the previous rung falls below δ) is never more accurate than τ-stopping and needs ground truth (appendix § E.2, § E.4); we use it only on the breadth axis, where u provides little to no signal (§ E.5).

Policy families. A scales K on a single agent, then deploys the target MAS topology once at the selected K^* with $N = 3$; B scales K on the target MAS topology itself, deploying wherever it stopped; C is a breadth-only control that scales N (adds agents) on the target topology one rung at a time, stopping once two consecutive rungs each gain less than a threshold δ in accuracy (appendix § E.6); D runs A's depth probe, then fixes K^* and runs C's breadth sweep, testing whether it is worth scaling depth-then-breadth (§ 5.2). Centralized and Decentralized deployments use $R = 2$, the minimum allowing communication. Parentheses name the stopping threshold (A(δ) stops A's probe on δ instead), and D(τ=0.5) abbreviates D(τ=0.5, δ^*). A(τ) is our main policy.

Each evaluation splits a benchmark's 50 tasks into 25 probe and 25 held-out tasks (means over 200 splits). Search-and-deploy energy is the upfront energy to get from nothing to a deployed

configuration that has served its first 25 tasks (it amortizes with deployment volume): every probed rung at its measured energy, plus one run of the deployed configuration (for B, C and D, this is the last rung evaluated; for A, this is an added MAS run, which contributes 41-59% of its total energy).

Setting τ, and what binary search saves. As we sweep τ, held-out accuracy rises to a maximum at τ = 0.5 then declines, while search-and-deploy energy falls monotonically in τ (Figure 7). We therefore choose τ = 0.5 as the setpoint (full sweep in Table A13; the secondary models corroborate it, Figure A12). Binary search finds the same K^* as a linear scan in every cell (hence the single accuracy curve in Figure 7) for 29% less probe energy at τ = 0.5, so every τ-stopped policy uses binary search (Algorithm 1).

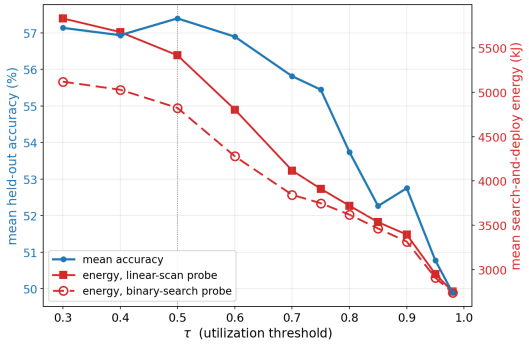


Figure 7: **Accuracy and energy cost across τ (utilization threshold), using linear and binary search.** Mean held-out accuracy (blue, left axis) and mean search-and-deploy energy (right axis) over the twelve (benchmark × target) cells; energy drawn for the linear-scan (square, solid) and binary-search (circle, dotted) probes. τ sweeps the eleven-rung ladder 0.30-0.98 (Table A12). The setpoint sweep uses the common ten-rung ladder ($K \leq 50$) on every benchmark, including SWE-bench Lite.

Algorithm 1 A(τ), utilization-guided search-and-deploy.

Inputs: utilization threshold τ, probe tasks D_{probe}, depth ladder $L = (K_1 < \cdots < K_m)$, target topology T

Phase 1, depth probe on a SINGLE agent (SAS), binary search on L:

lo $\leftarrow$ 1; hi $\leftarrow m$
while lo $<$ hi **do**
  mid $\leftarrow \lceil (\text{lo} + \text{hi})/2 \rceil$; run SAS at $K = L[\text{mid}]$ on D_{probe}
  $u \leftarrow$ mean fraction of the K-cap consumed ▷ label-free signal
  if $u \geq \tau$ **then** lo $\leftarrow$ mid ▷ go deeper if utilization $\geq \tau$
  else hi $\leftarrow$ mid $- 1$ ▷ back off if utilization $< \tau$
end while
$K^* \leftarrow L[\text{lo}]$ ▷ largest rung with $u(K) \geq \tau$ (boundaries: § E.3)

Phase 2, deploy:

run T at ($K = K^*$, $N = 3$; $R = 2$ if T is Centralized or Decentralized) on D_{probe}

5.2 RESULTS

Table 1: A(τ=0.5) **against** B(τ=0.5) **(left) and against two-phase** D(τ=0.5) **(right).** Δ = A's held-out accuracy minus the baseline's (pp); acc improv. = Δ / baseline accuracy; energy saved = 1 − A/baseline energy (positive favors A). Both panels pin τ at the 0.5 setpoint and average over the three MAS topologies (coverage in § E.1); full results for each family at its best-accuracy threshold in Table A15; protocol and per-family results in § E.4–§ E.7.

	A(τ=0.5) **vs** B(τ=0.5)			A(τ=0.5) **vs** D(τ=0.5)		
benchmark	Δ (pp)	acc improv.	energy saved	Δ (pp)	acc improv.	energy saved
SWE-bench Lite	+2.7	+6.6%	44%	−0.5	−1.1%	48%
BrowseComp+	−2.1	−4.2%	38%	−8.6	−15.4%	61%
WorkBench	+1.5	+1.8%	55%	+1.7	+2.1%	63%
FanOutQA	−0.4	−0.6%	67%	+0.0	+0.0%	63%
mean	+0.4	+0.9%	51%	−1.9	−3.6%	59%

Against B(τ=0.5) (the same label-free rule run on the MAS topology itself), A finds a roughly equally performant configuration for 38-67% less search-and-deploy energy, 51% on average (Table 1; Figure 8, top row). The gap is wider still against the label-dependent B(δ), which stops on the accuracy gain between rungs: there, A saves 65% energy at slightly higher accuracy (Figure A13). Single-agent utilization is thus a cheap, strong proxy for the MAS stopping point.

Is scaling breadth worth the energy? For each benchmark × topology cell, we grant D the $\delta = \delta^*$ threshold that maximizes its held-out accuracy averaged over the 200 probe splits (deployed

configurations and selected δ^* values in Table A16). This is generous accounting compared to A, whose τ is fixed at 0.5 for every cell; we do this to test whether stacking breadth scaling on top of depth adds any value under lenient conditions.

Compared to A, D trails by 1.7 pp on WorkBench, ties on FanOutQA, buys 0.5 pp on SWE-bench Lite, and buys 8.6 pp on BrowseComp+, the benchmark where N-scaling is strongest relative to K. The energy column is unambiguous: A spends 48-63% less search-and-deploy energy (mean 59%). A also matches or beats the breadth-only control C on the energy frontier on three of four benchmarks, with C achieving higher accuracy only on BrowseComp+ at higher energy cost (§ E.6). Accounting search cost alone, A saves 78% energy over B and 83% over D on average, savings that amortize as deployment volume grows (Figure A11). Although K and N both scale accuracy positively with energy (§ 4.4), on most of our benchmarks, the ensembling gain from breadth is outweighed by scaling depth and does not repay its energy cost (Figure 8, bottom row).

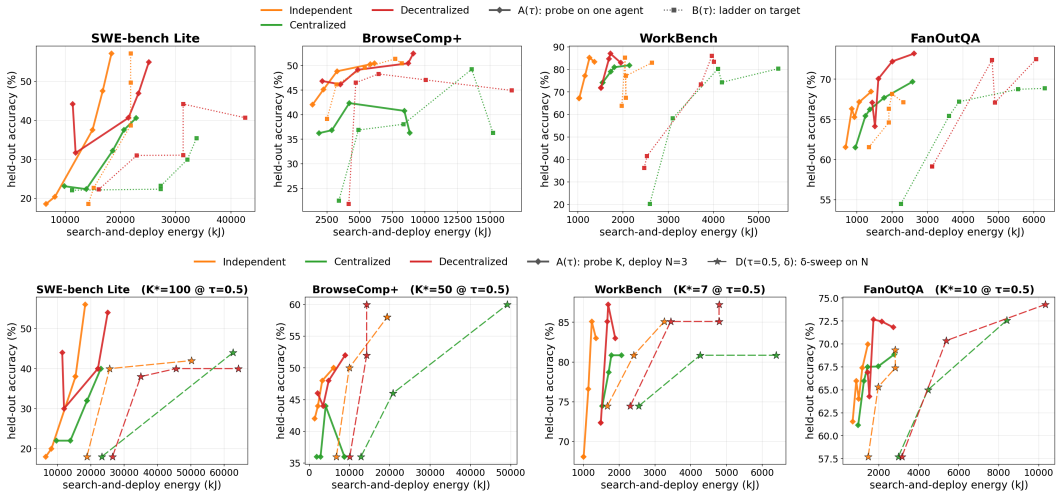


Figure 8: A(τ) **(solid line) against** B(τ) **(top row) and against** D(τ=0.5) **(bottom row): search-and-deploy energy vs held-out accuracy.** A(τ) and B(τ) sweep $\tau \in \{0.30, \ldots, 0.98\}$; D($\tau$=0.5) fixes depth at $K^*(\tau{=}0.5)$ and escalates N over its δ ladder (Table A12). Color = deployed topology; all families use Algorithm 1's binary-search probe. A(τ) reaches B's and D's accuracy range at far lower energy on every benchmark; their highest-accuracy settings exceed A only on BrowseComp+ (B and D) and, for D, on FanOutQA and SWE-bench Lite Centralized.

6 Conclusions and Limitations

We measured GPU energy per LLM call across four MAS topologies, scaling depth (K), agent count (N), and rounds (R) on four agentic benchmarks. Several limitations bound our claims. Hardware counters restrict us to locally served open-weight models and capture GPU dynamic energy only (host-side tool compute is excluded). Every grid measurement runs at batch size 1, as per-call attribution requires, though our structure and workload conclusions survive re-pricing at $B = 64$ (appendix § A.4). Unlike depth, breadth can also be batched, which lowers its wall-clock time; we do not evaluate batched breadth or latency. Our benchmarks must be agentic and locally deployable (no API-calling or open-ended tasks). For our search-and-deploy policy, the τ = 0.5 setpoint is calibrated on the four-benchmark aggregate, but it could shift with tasks far out of the distribution of our benchmarks; broadly speaking, our results provide a general τ threshold guideline and evidence of the efficacy of a label-free utilization threshold.

We find that for MAS, scaling iterative depth (K) converts energy to accuracy most efficiently, breadth (N) buys ensembling gains at near-linear energy cost, and rounds (R) buy nothing at matched per-agent depth (for homogeneous agents). We distill these into a label-free search-and-deploy policy: binary-search K on a single agent to the deepest budget agents still half-use (τ=0.5), then deploy the target topology at that budget with three agents. Against a two-phase depth-then-breadth policy, it cuts search-and-deploy energy by 59% while giving up at most half a point of accuracy on three of four benchmarks; against running the same rule on the MAS topology itself, it saves 51% energy at comparable accuracy.

Ethics Statement

We see no ethical concerns with our work. Our study involves no human subjects, no personally identifying data, and no new data collection. All four benchmarks (FanOutQA, WorkBench, BrowseComp+, SWE-bench Lite) are existing public research datasets used within their intended scope and licenses, and every model is a publicly released open-weight checkpoint. Our work's aim is to reduce the compute and energy that multi-agent systems consume, so its expected effect is to lower rather than raise the environmental cost of deployed systems.

Reproducibility Statement

Every measurement in this paper comes from locally served open-weight models on hardware we control, with no external API calls, so the pipeline is reproducible end to end; code, configurations, and data are available as anonymous downloadable source code. Appendix § B documents the full deployment needed to rebuild the grid: exact model checkpoints and quantizations, SGLang versions and launch flags, the operative context windows and client-side truncation rule, the request path with temperatures and seeds, and the per-benchmark tool environments, datasets, and scoring rules. The energy-measurement protocol (NVML counters, idle-baseline subtraction, per-call attribution) is in § 3.4, the energy-law fit and its specification comparison in appendix § A.1, and the configuration-search protocol (probe/held-out splits, threshold ladders, number of splits) in § 5.1 and appendix § E.

Declaration of LLM Usage

In this work, we used generative AI tools to provide feedback on research methodology and experiments, clean and reformat datasets, and support qualitative and thematic data analysis. We have not used generative AI tools to generate synthetic datasets or formulate mathematical claims. Additionally, we used generative AI tools to write code for generating figures and deploying agents from detailed instructions, summarize existing literature, format references, and edit the research paper to improve readability. We have reviewed all AI-assisted work, verifying all code works as intended and manually reading and checking all literature.

References

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Yongchao Chen, Jiefeng Chen, Rui Meng, Ji Yin, Na Li, Chuchu Fan, Chi Wang, Tomas Pfister, and Jinsung Yoon. TUMIX: Multi-agent test-time scaling with tool-use mixture. In *International Conference on Learning Representations*, pp. 55848–55878, 2026.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-Plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*, 2025.

Jiaming Cheng, Subhransu Das, and Rajiv Ramnath. When does latent communication pay? A causal audit of relayed KV caches in multi-agent LLMs. *arXiv preprint arXiv:2608.04893*, 2026.

Seunghyuk Cho, Sunghyun Choi, Jaeseung Heo, Youngbin Choi, Saemi Moon, MoonJeong Park, and Dongwoo Kim. Long live the librarian! A persistent search sub-agent for energy-efficient multi-agent software engineering systems. *arXiv preprint arXiv:2605.27787*, 2026.

Hyeong Kyu Choi, Jerry Zhu, and Sharon Li. Debate or vote: Which yields better decisions in multi-agent large language models? *Advances in Neural Information Processing Systems*, 38: 101732–101764, 2025.

DeepSeek-AI. DeepSeek-V3/R1 inference system overview. https://github.com/deepseek-ai/open-infra-index/blob/main/202502OpenSourceWeek/day_6_one_more_thing_deepseekV3R1_inference_system_overview.md, February 2025. Open Source Week, Day 6.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*. JMLR.org, 2024.

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.

Aryo Pradipta Gema, Alexander Hägele, Runjin Chen, Andy Arditi, Jacob Goldman-Wetzler, Kit Fraser-Taliente, Henry Sleight, Linda Petrini, Julian Michael, Beatrice Alex, Pasquale Minervini, Yanda Chen, Joe Benton, and Ethan Perez. Inverse scaling in test-time compute. *Transactions on Machine Learning Research*, 2025. URL https://arxiv.org/abs/2507.14417.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *ICLR*, 2024.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, pp. 32808–32824, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/8b4add8b0aa8749d80a34ca5d941c355-Paper-Conference.pdf.

Wei-Chieh Huang and Cornelia Caragea. MADIAVE: Multi-agent debate for implicit attribute value extraction. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 3035–3053, 2026.

Erik Johannes Husom, Arda Goknil, Lwin Khin Shar, and Sagar Sen. The price of prompting: Profiling energy use in large language models inference. *arXiv preprint arXiv:2407.16893*, 2024.

Aniketh Iyengar, Jiaqi Han, Boris Ruf, Vincent Grari, Marcin Detyniecki, and Stefano Ermon. Energy scaling laws for diffusion models: Quantifying compute in image generation. In *The 2026 ACM Conference on Fairness, Accountability, and Transparency*, pp. 8029–8050, 2026.

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*, 2024.

Lars Benedikt Kaesberg, Jonas Becker, Jan Philip Wahle, Terry Ruas, and Bela Gipp. Voting or consensus? Decision-making in multi-agent debate. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11640–11671, 2025.

Yubin Kim, Ken Gu, Chanwoo Park, Chunjong Park, Samuel Schmidgall, A. Ali Heydari, Yao Yan, Zhihan Zhang, Yuchen Zhuang, Yun Liu, Mark Malhotra, Paul Pu Liang, Hae Won Park, Yuzhe Yang, Xuhai Xu, Yilun Du, Shwetak Patel, Tim Althoff, Daniel McDuff, and Xin Liu. Capable language models can outgrow the benefits of collaboration. *Nature Machine Intelligence*, 8(7): 1157–1172, 2026.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In *NeurIPS*, 2023.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=bgzUSZ8aeg.

Yuante Li, Yicheng Tao, Kate Zhang, Taozhi Wang, Gefei Gu, and Yaxin Zhou. Diverse evidence, better forecasts: Multi-agent deliberation under information asymmetry. *arXiv preprint arXiv:2607.01661*, 2026.

Tengxiao Liu, Zifeng Wang, Jin Miao, I-Hung Hsu, Jun Yan, Jiefeng Chen, Rujun Han, Fangyuan Xu, Yanfei Chen, Ke Jiang, Samira Daruki, Yi Liang, William Yang Wang, Tomas Pfister, and Chen-Yu Lee. Budget-aware tool-use enables effective agent scaling (BATS). *arXiv preprint arXiv:2511.17006*, 2025.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic LLM-powered agent network for task-oriented agent collaboration (DyLAN). *arXiv preprint arXiv:2310.02170*, 2023.

Kyle A. McCleary and James M. Ghawaly. Quantifying the accuracy and cost impact of design decisions in budget-constrained agentic LLM search. In *Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026)*, pp. 10291–10300. European Language Resources Association (ELRA), 2026.

Chenxu Niu, Wei Zhang, Jie Li, Yongjian Zhao, Tongyang Wang, Xi Wang, and Yong Chen. TokenPowerBench: Benchmarking the power consumption of LLM inference. In *AAAI*, 2026.

Hiari Pizzini Cavagna, Andrea Proia, Giacomo Madella, Giovanni B. Esposito, Francesco Antici, Daniele Cesarini, Zeynep Kiziltan, and Andrea Bartolini. SweetSpot: An analytical model for predicting energy efficiency of LLM inference. In *Proceedings of the 17th ACM/SPEC International Conference on Performance Engineering (ICPE 26)*, pp. 83–95, 2026.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In *ACL*, 2024a.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling large language model-based multi-agent collaboration (MacNet). *arXiv preprint arXiv:2406.07155*, 2024b.

Michael Rizvi-Martel, Satwik Bhattamishra, Neil Rathi, Guillaume Rabusseau, and Michael Hahn. Benefits and limitations of communication in multi-agent reasoning. In *ICLR*, 2026.

Boris Ruf and Marcin Detyniecki. The cost of context: Profiling the energy footprint of input tokens in large language models. *HotCarbon / SIGENERGY Energy Informatics Review*, 2026. URL `https://hotcarbon.org/assets/2026/paper-17.pdf`.

Khushal Sethi. Don't overthink it: Inter-rollout action agreement as a free adaptive-compute signal for LLM agents. *arXiv preprint arXiv:2604.08369*, 2026.

Andries Smit, Nathan Grinsztajn, Paul Duckworth, Thomas D. Barrett, and Arnu Pretorius. Should we be going MAD? A look at multi-agent debate strategies for LLMs. In *Proceedings of the 41st International Conference on Machine Learning*. JMLR.org, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limitations of large language models on reasoning and planning tasks. In *International Conference on Learning Representations*, 2025.

Olly Styles, Sam Miller, Patricio Cerda-Mardini, Tanaya Guha, Victor Sanchez, and Bertie Vidgen. WorkBench: A benchmark dataset for agents in a realistic workplace setting. *arXiv preprint arXiv:2405.00823*, 2024.

Dat Tran and Douwe Kiela. Single-agent LLMs outperform multi-agent systems on multi-hop reasoning under equal thinking token budgets. *arXiv preprint arXiv:2604.02460*, 2026.

Arihant Tripathy, Ch Pavan Harshit, and Karthik Vaidhyanathan. SWEnergy: An empirical study on energy efficiency in agentic issue resolution frameworks with SLMs. In *Proceedings of the 2026 International Workshop on Agentic Engineering*, pp. 104–111, 2026.

Junlin Wang, Siddhartha Jain, Dejiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. Reasoning in token economies: Budget-aware evaluation of LLM reasoning strategies. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 19916–19939, 2024.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations*, 2025.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang (Eric) Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Ahmed Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *COLM 2024*, August 2024.

Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. When more is less: Understanding chain-of-thought length in LLMs. In *International Conference on Learning Representations*, pp. 127108–127138, 2026.

Florian Valentin Wunderlich, Lars Benedikt Kaesberg, Jan Philip Wahle, Terry Ruas, and Bela Gipp. Multi-agent reasoning improves compute efficiency: Pareto-optimal test-time scaling. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL 2026)*, pp. 1–14, 2026.

Andrea Wynn, Harsh Satija, and Gillian Hadfield. Talk isn't always cheap: Understanding failure modes in multi-agent debate. *arXiv preprint arXiv:2509.05396*, 2025.

Yingxuan Yang, Chengrui Qu, Muning Wen, Laixi Shi, Ying Wen, Weinan Zhang, Adam Wierman, and Shangding Gu. Understanding agent scaling in LLM-based multi-agent systems via diversity. *arXiv preprint arXiv:2602.03794*, 2026.

Yongjin Yang, Euiin Yi, Jongwoo Ko, Kimin Lee, Zhijing Jin, and Se-Young Yun. Revisiting multi-agent debate as test-time scaling: A systematic study of conditional effectiveness. *arXiv preprint arXiv:2505.22960*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for LLM-based multi-agent systems (AgentPrune). *arXiv preprint arXiv:2410.02506*, 2024.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet (MaAS). *arXiv preprint arXiv:2502.04180*, 2025a.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-Designer: Architecting multi-agent communication topologies via graph neural networks. In *Proceedings of the 42nd International Conference on Machine Learning*. JMLR.org, 2025b.

Huixiang Zhang and Mahzabeen Emu. Do latent channels actually communicate? A causal audit of latent multi-agent LLM communication. *arXiv preprint arXiv:2607.26773*, 2026.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Ö. Arık. Multi-agent design: Optimizing agents with better prompts and topologies (MASS). In *ICLR*, 2026.

Andrew Zhu, Alyssa Hwang, Liam Dugan, and Chris Callison-Burch. FanOutQA: A multi-hop, multi-document question answering benchmark for large language models. In *Proceedings of ACL*, 2024.

King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, Changwang Zhang, Chenghua Lin, Jun Wang, Ge Zhang, and Wangchunshu Zhou. Scaling test-time compute for LLM agents. *arXiv preprint arXiv:2506.12928*, 2025.

Xiaochen Zhu, Caiqi Zhang, Yizhou Chi, Tom Stafford, Nigel Collier, and Andreas Vlachos. Demystifying multi-agent debate: The role of confidence and diversity. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 33909–33930, 2026.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*. JMLR.org, 2024.

APPENDIX

In § A, we detail extended analysis of energy: the physical energy law behind our measurements and its validation, how serving occupancy reshapes its coefficients, the token and dollar mispricing of agentic energy, and the decomposition of what drives energy across topology and workload. § B documents deployment: topology protocols, model serving configuration, and the benchmark tool environments. § C provides per-axis detail behind the main scaling results, § D the extended cross-architecture validation, and § E the remaining configuration-search variants and the full per-arm results.

A ENERGY ANALYSIS

Here, we analyze the energy of LLM calls end to end: the physical law (§ A.1), its behavior under batched serving (§ A.2), the mispricing under token and dollar accounting (§ A.3), and the structure-versus-workload decomposition (§ A.4).

A.1 A PHYSICAL ENERGY LAW FOR LLM CALLS

Here, we answer the question: what is the physical shape of an LLM call's energy, and how can it be predicted from token counts alone? We develop a three-term law that prices a call from quantities a deployer already logs, namely how many prompt tokens it read (P), how many completion tokens it generated (C), and how much context each generated token had to re-read. The batching result in § A.2 says which of its terms shrink, the mispricing formula in § A.3 is the law divided by a single token rate, and the cross-architecture test in § D asks whether the same three terms fit other models. Per-call GPU dynamic energy decomposes as

$$E_{\text{call}} \approx \underbrace{e_P\, P}_{\text{prefill}} + \underbrace{e_C\, C}_{\text{decode}} + \underbrace{e_{KV}\left(P\,C + \frac{C^2}{2}\right)}_{\text{context}} \tag{1}$$

where the context term charges each generated token for re-reading the KV cache; fit on all 17,250 primary-model task records, it reaches R^2 = 0.961. **Prefill** ($e_P \cdot P$): the prompt is processed in one parallel pass, almost all of which is linear in P: dense-layer FLOPs are constant per token, weights are read from memory once for the whole pass, and the $O(P^2)$ attention score computation is arithmetic performed on-chip (FlashAttention tiling) rather than off-chip memory traffic, which is what batch-1 energy tracks. **Decode term** ($e_C \cdot C$): each generated token requires its own full forward pass, i.e. one full weight read from off-chip memory per token at batch size 1. **KV-context** ($e_{KV} \cdot (PC + C^2/2)$): generating the j-th completion token re-reads the KV entries of all $P + j - 1$ preceding tokens; summing over C decode steps gives $\sum_{j=1}^{C}(P + j - 1) = PC + C(C-1)/2 \approx PC + C^2/2$ (the $-C/2$ remainder is linear in C and is absorbed by e_C), the same context-read form as SweetSpot's analytical model (Pizzini Cavagna et al., 2026); what we add is a coefficient fit from agentic traces, its physical validation (Appendix A.1.2), and its behavior under batching (§ A.2). Because this term is nonlinear within a call, the task-level regressor must be built call-by-call, $\Sigma KV = \sum_{\text{calls}}(P_i C_i + C_i^2/2)$, not from task-level token totals (which silently degenerates the fit to $e_{KV} \approx 0$; see Appendix A.1.1).

Fit by ordinary least squares with no intercept (through the origin, since zero tokens must cost zero dynamic energy) on the 17,250 primary-model task records, the model reaches $R^2 = 0.961$ with a median absolute percentage error (APE) of 7.6% (Figure A1). The fitted coefficients are $e_P = 0.016$ J per prompt token, $e_C = 5.37$ J per completion token, and $e_{KV} = 1.7 \times 10^{-5}$ J per KV token-pair, so $e_C/e_P \approx 335\times$ at short context under this cached, batch-1 grid (against uncached step-0 prefill, Table A3, the raw hardware ratio is ≈126-132×). We fit at the task level because a task is the unit over which a deployer evaluates cost, and summing a task's calls averages out sub-second timing noise; since the law is linear in the per-call terms once ΣKV is summed call by call, the task-level fit is the same law aggregated.

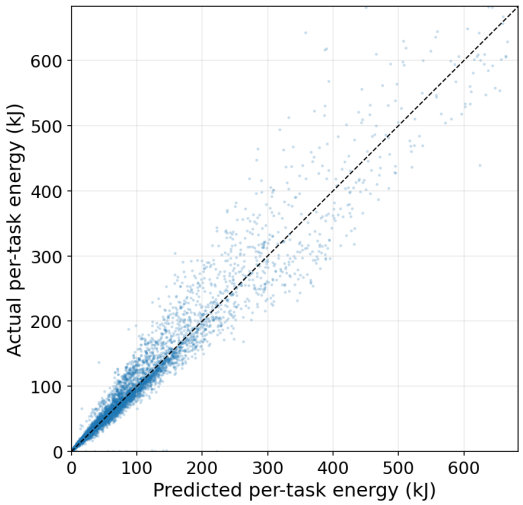


Figure A1: **Task-level energy regression.** Predicted vs. actual per-task GPU dynamic energy under the through-origin model of equation 1, $n = 17{,}250$ tasks.

Table A1 compares equation 1 with two simpler specifications on the same records. Dropping the KV term (S2) barely moves R^2, but a nested F-test ($F = 286$, $p \approx 0$), $\Delta\text{BIC} = -274$, and 5-fold cross-validated median APE (7.6% vs. 9.7%) show it is a statistically significant term that also improves out-of-sample prediction. MAS tasks make up 87% of the records and each carries ∼6× the energy of a single-agent task, so a single-rate total-token model (S3) fit across the grid is dominated by those MAS measurements, fitting MAS tasks to within ∼1.2× but over-predicting single-agent task energy by ∼6×. Token counts misprice whatever differs in composition from the fitted average, and SAS is the extreme case. § A.3 quantifies the same effect cell by cell. Each coefficient is also checked against raw power and speed measurements, independently of the regression fit (Appendix A.1.2).

Table A1: Task-level energy regression, three specifications on the same $n = 17{,}250$ records. Full table, including intercept variants, in Appendix A.1.1.

Specification	**Form**	R^2	**median APE**
S1: three-term, equation 1, through origin	$e_P P + e_C C + e_{KV}\,\Sigma KV$	0.961	7.6%
S2: additive, through origin	$e_P P + e_C C$	0.960	9.7%
S3: single-rate total-token	$a + b\,(P + C)$	0.74	43%

A.1.1 Energy-regression specification comparison

Table A2 lists the candidate forms of the task-level energy regressor, fit on identical canonical records ($n = 17{,}250$ tasks / ∼970K calls); medAPE is the median over tasks of $|\text{predicted} - \text{actual}|/\text{actual}$, which (unlike variance-weighted R^2) gives every task one vote.

Table A2: Task-level energy regressor: candidate forms on the same $n = 17{,}250$ records (a = intercept in J; γ_n multiplies the number of calls per task).

Form	a	e_P	e_C	e_{KV}	R^2	**medAPE**
$e_P P + e_C C + e_{KV}\,\Sigma KV$ **(selected)**	–	0.0160	5.375	1.7e-5	0.9610	**7.6%**
$a + e_P P + e_C C + e_{KV}\,\Sigma KV$	2296	0.0155	5.235	2.2e-5	0.9612	12.0%
$e_P P + e_C C$	–	0.0174	5.651	–	0.9604	9.7%
$a + \gamma_n n_{\text{calls}} + e_P P + e_C C$ ($\gamma_n = -223$)	4177	0.0242	5.896	–	0.9634	12.1%

Nested F-test (correct-KV-origin vs. additive-origin): $F = 286$ (df 1, 17247), $p \approx 0$; $\Delta\text{BIC} = -274$ favoring the KV model despite its extra parameter; 5-fold cross-validated medAPE 7.6%

(KV) vs. 9.7% (additive), so the gain generalizes out of sample. A single-rate total-token model $E = a + b(P + C)$ reaches only $R^2 = 0.74$, 43% median APE. Intercepts are rejected on physical grounds (idle-subtracted energy of zero tokens must be zero) and add no statistical value: adding one to the KV model changes R^2 by 0.0002 and worsens medAPE from 7.6% to 12.0%. The KV regressor must be summed call-by-call ($\Sigma KV = \sum_{\text{calls}}(P_i C_i + C_i^2/2)$); computing it from task-level totals invents cross-terms between different calls' prompts and completions and OLS silently fits $e_{KV} \approx 0$, degenerating to the additive model.

A.1.2 Physical validation of the coefficients

The decode floor: Decode-dominated calls run at 37.3 tok/s at 200 W dynamic power, giving 200/37.3 = 5.4 J/token, which matches e_C directly; the implied memory traffic (768 GB/s ÷ 37.3 tok/s ≈ 21 GB per decode step) matches one BF16 weight read (∼18 GB) plus KV/activation traffic.

Prefill linearity and the caching blend: Splitting prefill-dominated calls ($C \leq 50$) by ReAct-step index at matched prompt size separates uncached from cache-served prefill (Table A3).

Table A3: Prefill energy per prompt token by ReAct-step index and prompt-size decade (uncached step 0 vs. cache-served later steps).

P decade	step 0 (uncached)	steps 1–2	steps 3+
10^3	0.0407 J/tok (n=6,835)	0.0174 (n=22,097)	0.0048 (n=45,985)
10^4	0.0427 J/tok (n=11,283)	0.0033 (n=17,164)	0.0031 (n=93,364)

Step-0 calls, whose prompts are not yet radix-cached, cost a constant ≈0.042 J/token across a 10× range of prompt lengths, confirming linearity in P for uncached prefill; later steps collapse toward ≈0.003 J/token as the prompt becomes dominated by the cached prefix of the trajectory. The fitted e_P is a cache-hit-weighted blend of these two rates.

The KV term is a time tax at constant power: Decode-dominated calls ($C \geq 300$) binned by context size (Table A4).

Table A4: Decode-dominated calls ($C \geq 300$) binned by context size: decode speed, dynamic power, and energy per generated token.

context	n	decode speed	dynamic power	J/token
$\sim 10^2$	11,687	37.7 tok/s	202 W	5.35
$\sim 10^3$	52,558	36.7 tok/s	200 W	5.45
$\sim 10^4$	104,388	33.5 tok/s	199 W	5.95

Power is flat while speed falls, so a long-context token costs more only because it takes longer at the same power: the KV term is a bandwidth-occupancy tax (an access-energy alternative would add joules without adding time). The J/token increase from 10^2 to 10^4 context is 0.60 J, and equation 1 predicts that rise to be 10^4 times its KV coefficient. Equation 1 refit per call rather than per task ($e_P = 0.0154$, $e_C = 4.91$, $e_{KV} = 5.7 \times 10^{-5}$ J on the 970K individual calls, $R^2 = 0.883$) predicts 5.7×10^{-5} J $\times\, 10^4 = 0.57$ J/token, within 5% of the measured 0.60 J/token: the fitted coefficient is the measured slowdown priced at card power. The task-level fit assigns the same cost differently: ΣKV and C are collinear across tasks, so it puts more into e_C (5.37 vs. 4.91 J/token) and less into e_{KV} (1.7 vs. $5.7{\times}10^{-5}$ J); at the grid's typical context ($\bar{L} \approx 11$K) both fits give the same decode-plus-context cost per generated token, 5.56 vs. 5.54 J. The per-call fit therefore attributes the cost between the two terms more cleanly, since context varies within a call at fixed completion length, while the task-level fit predicts task energy better (R^2 0.961 vs. 0.883) and prices the same total.

A.2 Batching amortizes decode, not prefill or context

Here, we measure how each term of the § A.1 energy law responds to serving occupancy. We fit the law under batch-1 serving, which gives clean per-call attribution, but production APIs serve many concurrent sequences per GPU, and because the GPU draws close to full power regardless of how many requests it serves, each token's share of that power falls as more requests are batched together. In this section, we address how the shape of LLM call energy and the resulting token/dollar mispricing changes with serving regime (serial vs. production-style batching).

To measure how each term responds, we run a separate serving experiment on Qwen3.5-9B on both cards (A5000 and A6000). Each trial sends a batch of $B \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ identical-

length requests with unique prefixes, so prefix caching (which affects only prefill) cannot apply and is held out of this comparison on purpose. Three request shapes pin the three coefficients separately: prompt-heavy requests ($P \in \{512, 2048, 8192, 32768\}$, $C = 16$) pin e_P; generation-heavy requests ($P = 64$, $C \in \{256, 1024\}$) pin e_C; and long-context requests ($P \in \{8192, 16384\}$, $C \in \{512, 2048, 4096\}$) pin e_{KV} by holding P fixed while varying C, which breaks the near-collinearity of ΣKV with P and C that a normal request visits. A fourth, context-read set ($P \in \{4096, 16384, 40960\}$, $C = 256$) adds long-prompt points to the same joint fit. Energy is idle-subtracted with the grid's settled baseline (24 W A5000, 32 W A6000).

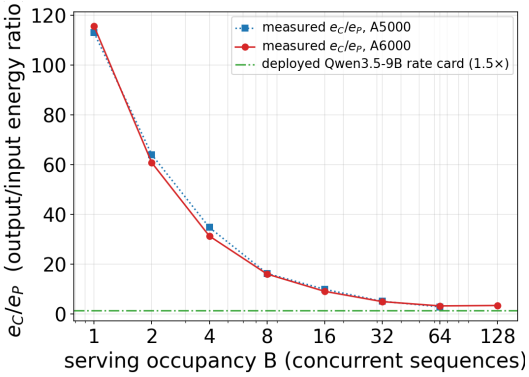


Figure A2: **Energy-law coefficients vs. serving occupancy.** e_C/e_P vs. batch size B, both GPUs. e_P is flat; e_C falls $\sim 1/B$ then floors, so e_C/e_P collapses from ∼115–120 to a floor of ∼3 (2.9–3.3); the A5000 and A6000 trace the same curve. Full per-B coefficient table in Appendix A.2.1.

Prefill does not amortize; decode amortizes, then floors. $e_P(B)$ stays flat (0.037–0.045 J/input-token) at every B: the board runs at its rated thermal design power (TDP) whether it serves one request or many, so uncached prefill energy tracks throughput, and prefill throughput does not depend on B. $e_C(B)$ falls $\sim 1/B$ from 5.0 J/token at $B = 1$ to 0.20 J/token by $B = 32$, then floors (≈ 0.12 at $B = 64$–128): the B concurrent requests share the same weight read, so each byte of weights serves more tokens as B grows, until aggregate decode throughput saturates at ∼2,000 tok/s. The resulting e_C/e_P collapses from ∼115–120× at $B = 1$ to a ∼3× floor (2.9–3.3) at $B = 64$–128 (Figure A2), landing inside the 3–5× output/input band of frontier API rate cards.[1] e_{KV}, however, does not amortize: across B it stays within the same order of magnitude, 0.97–2.5$\times 10^{-5}$ J (Table A5), while e_C falls 40×, because each sequence attends only to its own KV cache, so a token re-reads its full context from memory regardless of how many other sequences share the GPU. At $B = 32$, the decode term has fallen to $e_C \approx 0.20$ J/token while the KV surcharge has not. At the grid's typical context (the median MAS cell re-reads $\bar{L} \approx 11$K tokens per generated token, § A.3), the two terms are therefore within a factor of two ($e_{KV} \cdot \bar{L} \approx 0.11$ J/token), a cost that token counting, blind to context length, cannot price at all.

DeepSeek's disclosed 24-hour V3/R1 statistics (DeepSeek-AI, 2025) place one public first-party deployment inside this floor. They imply 84–92 concurrent sequences per H800 GPU (14.8K decode tokens/s per 8-GPU node at their reported 20–22 tokens/s per user), between our B=64 and B=128, where e_C/e_P has already floored; the anchor is order-of-magnitude only, since that 671B MoE runs with expert parallelism across 144-GPU decode units.

A.2.1 Full occupancy coefficients and phase attribution

Joint through-origin fits per (card, B); the A5000 B=64 column is the prefill/short-decode route, since its 24GB pool is memory-limited at that occupancy for realistic response lengths, while the A6000 (48GB) continues to B=128. All rows subtract the settled idle baseline of the grid (24 W A5000, 32 W A6000), replacing the probe's recorded idle window (50–138 W, unsettled), which had depressed the batch-1 decode coefficient to ∼2.7–3.0 J/token; on the corrected baseline the probe's batch-1 coefficients match the grid's (5.0–5.2 vs. 5.37 J/token decode; 0.042–0.045 vs. 0.042 J/token uncached prefill). The two cards agree within 15% at the floor. e_{KV} is identified on both cards at $B \leq 8$ ($\approx 1.0 \times 10^{-5}$; the A5000's KV-separation cells are pool-limited beyond that) and on the A6000 through B=16, each from 30 KV-separation and 15 context-read requests. Beyond that, the A6000 estimate rests on fewer context-varying requests (10 + 5 at B=32, 5 at B=64, none at

[1] List-price output/input ratios from provider pricing pages, July 2026: GPT-4o 4× ($10/$2.50 per million tokens), Claude Sonnet 5× ($15/$3), DeepSeek R1 4× ($2.19/$0.55); Gemini 2.5 Pro is the outlier at 8× ($10/$1.25).

Table A5: Energy-law coefficients vs. serving occupancy B, both GPUs. The long-context request family was not run at B=128, so $e_{KV}(128)$ is identified only from residual context variation in the other request shapes; read it as indicative.

B	1	2	4	8	16	32	64	128
e_P, A5000 (J/input tok)	0.045	0.041	0.039	0.038	0.037	0.037	0.043	n/a
e_P, A6000	0.042	0.042	0.042	0.043	0.039	0.039	0.037	0.037
e_C, A5000 (J/output tok)	5.17	2.67	1.36	0.69	0.37	0.21	0.124	n/a
e_C, A6000	5.03	2.63	1.34	0.68	0.36	0.20	0.121	0.123
e_C/e_P, A5000	115	66	35	18	10	5.6	2.9	n/a
e_C/e_P, A6000	119	62	32	16	9.2	5.0	3.3	3.3
e_{KV}, A6000 (J/KV-tok-read)	11.1e-6	9.7e-6	10.1e-6	10.0e-6	10.4e-6	25.1e-6	13.6e-6	9.9e-6

B=128), which is why the B=32 value (25e-6) sits above its neighbors (10–14e-6): the KV term is the smallest in the law, ∼1e-5 J per token-pair, so a small run-to-run power difference moves it more than the other two. Across the full range, it stays within 0.97–2.5$\times 10^{-5}$ J, the same order of magnitude, while e_C falls 40× over the same B; that contrast is the no-amortization claim.

Phase energy share reverses under batching (all SWE-bench Lite step-budget sweep records, n=2,598; cache-preserving transport with the grid coefficients of § A.1: prefill $e_P P$, decode $e_C\, s(B)\, C$ with $s(64) = 0.024$ from Table A5, KV $e_{KV}\Sigma KV$; Table A6).

Table A6: Phase shares of task energy at serial and production occupancy (SWE-bench Lite step-budget sweep, n=2,598; decode re-priced by $s(64)$ from Table A5).

Regime	**prefill share**	**decode share**	**KV share**
Serial (B=1)	20%	74%	5%
Production floor (B=64)	74%	7%	20%

Decode-dominated energy at batch 1 becomes prefill-dominated at production occupancy, because batching amortizes only decode; the KV share grows from 5% to 20% even though its absolute per-token cost is unchanged, since the decode term shrinks around it.

A.3 Token and dollar mispricing of agentic energy

Here, we quantify the mispricing that single-rate token and dollar accounting produce on the grid. Because prefill and decode carry different energy rates, a single-rate token or dollar accounting (every token priced the same) misprices any cell whose composition differs from the reference average. The plotted cells are the 126 MAS step-budget sweep cells of the grid (4 benchmarks × 3 MAS topologies × the step-budget ladder); the reference set for r adds the SAS sweep cells, and all quantities are per-cell means over tasks. With

$$x = \bar{C}/\bar{P}, \qquad \bar{L} = \overline{\Sigma KV}/\bar{C}, \qquad r = \Sigma E/\Sigma T, \qquad m = \bar{E}/\big((\bar{P}+\bar{C})\, r\big),$$
$$m(x;\bar{L}) = \frac{e_P + \delta(\bar{L})\, x}{(1+x)\, r}, \qquad \delta(\bar{L}) = e_C + e_{KV}\bar{L}, \tag{2}$$

where x is the cell's completion-to-prompt composition, $\bar{L}$ its mean context re-read per generated token, r the single rate over the reference set, and m the mispricing ratio (measured over token-implied energy). Substituting the energy law of equation 1 into m yields the parameter-free prediction of the second line, equation 2. Because r is an average over the reference set, $m = 1$ at the reference composition by construction; the curve therefore predicts a spread, how far cells of different composition depart from that average.

Under serial (batch-1) physics, single-rate accounting fails. It underprices near-pure-decode cells by up to ∼12× and overprices prefill-heavy cells by up to ∼1.6×, a ∼20× spread (Figure A3). Repricing every cell at the B=64 coefficients, which amortize decode and leave prefill and KV-context where they are, collapses the spread to ∼1.4×. Most of the serial mispricing is thus specific to serial serving.

Real APIs are not single-rate; they charge output tokens more than input tokens. Let ρ be that price ratio (output price per token over input price per token), so a two-rate bill is proportional to $P + \rho C$ rather than $P + C$. Under such a bill the composition-driven mispricing cancels when ρ equals the effective asymmetry $\delta(\bar{L})/e_P$, which reduces to e_C/e_P at short context. At the B=64 floor, that asymmetry is ≈ 3, and frontier rate cards (typically 3–5× output over input) sit right inside it, so

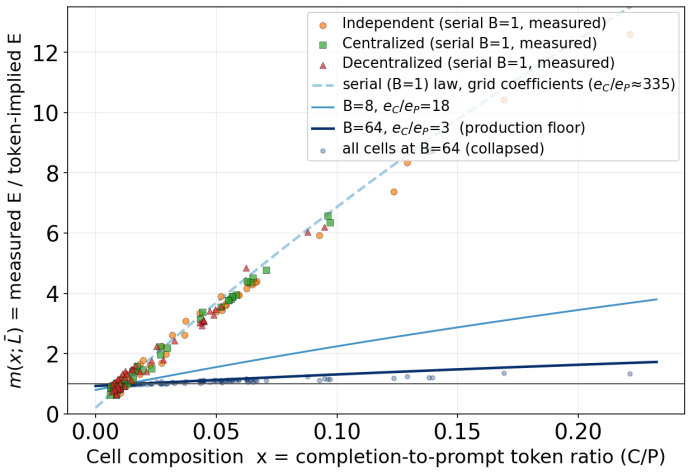


Figure A3: **Single-rate mispricing vs. composition, serial vs. batched.** Each point is one benchmark×topology sweep cell; x = mean completion/mean prompt tokens, m = measured/token-implied energy. Under serial (batch-1) serving, decode-heavy cells are underpriced and prefill-heavy cells overpriced, a ∼20× spread top to bottom (m from 0.63 to 12.6); re-priced at the B=64 coefficients, the spread collapses to ∼1.4× (m from 0.97 to 1.36).

at frontier API scale two-rate pricing tracks energy in both shape and level; a $\rho = 3$ card matched to the coefficients collapses the B=64 spread to $\approx 1.14\times$, close to one-to-one. The rate card of our own primary model is the counter-example. Qwen3.5-9B is listed by public API aggregators at $0.10 per million input tokens and $0.15 per million output tokens (provider range $0.10–0.20 on input), so $\rho \approx 1.5$, below the hardware floor; such a card undercharges completion-heavy workloads relative to their energy. Dollar cost is therefore a faithful energy proxy only when a deployment's price ratio matches its hardware's decode/prefill energy ratio, and that ratio itself depends on serving occupancy (§ A.2) and on context length.

We convert total energy to what it would be under batching, an idealized re-pricing that assumes a memory-unconstrained KV pool (the A5000's own pool could not hold 64 concurrent agentic streams), by plugging in only the amortized decode coefficient e_C from the batching experiments on A5000; only decode amortizes with B; prefill and the context term are flat in B (Table A5), and decode is also the one term caching does not touch (§ A.2). So we keep each cell's own measured prefill and context energy and scale only its decode by $s(B) = e_C(B)/e_C(1)$:

$$E_B = \underbrace{e_P\,P}_{\text{prefill, kept}} + \underbrace{e_C\,s(B)\,C}_{\text{decode, amortized}} + \underbrace{e_{KV}\,\Sigma KV}_{\text{context, kept}}, \qquad s(B) = e_C(B)/e_C(1).$$

A.4 What drives energy: structure vs. workload

Here, we test which factor, agent structure or workload, drives each component of a task's energy. A task's energy is its token count (T) times its energy-per-token (E/T); those two values are driven by different factors. The agent/communication structure (topology) determines how many LLM calls are made and thus sets token count; the workload (benchmark) sets composition, prompt-heavy or completion-heavy, and thus the energy-per-token that equation 1 assigns. To test whether the two factors really separate this way, we run a two-way ANOVA of $\log E$ (total task energy) and $\log(E/T)$ (energy-per-token) on topology and benchmark at a fixed step budget of 10 across the primary-model grid (Table A7). **Structure sets total energy.** Topology explains most of the variance in $\log E$ and benchmark little of it, because structure (agent count, communication rounds) multiplies LLM calls, and every added call brings its own completion tokens, while completion tokens are energetically expensive. **Workload sets the rate.** The ordering reverses for $\log(E/T)$, where benchmark explains the largest systematic share of variance and topology almost none, though most variance is within-cell. Task shape, chiefly tool-return length versus answer length, sets the completion-to-prompt composition that equation 2 prices; retrieval-heavy tasks return large document packs (prompt-heavy, low C/P), tool-heavy tasks return short structured strings (higher C/P). Figure A4 shows the same split directly: topology moves bar height, benchmark moves the prefill/decode proportions.

Both conclusions survive re-pricing at the B=64 batching coefficient with reduced magnitude (Table A7, B=64 rows): batching amortizes only decode, the structure-driven term, so structure's share shrinks, but the qualitative split (structure for volume, workload for rate) is serving-invariant.

Table A7: Two-way ANOVA: share of variance explained (classical η^2, %, Type-II sums of squares), primary-model grid at step budget 10, $n = 1250$ (4 topologies $\times$ 4 benchmarks); all effects $p < .001$. B=64 rows re-price each cell's decode term at the batching coefficient (Appendix A.4.1). Type-II shares on unbalanced cells need not sum to 100%.

Dependent variable	topology	benchmark	interaction	residual
$\log E$ (total energy), B=1	**56.6**	7.7	2.3	32.5
$\log E$ (total energy), B=64	**45.0**	18.9	3.3	30.6
$\log(E/T)$ (energy-per-token), B=1	1.0	**31.0**	3.7	62.5
$\log(E/T)$ (energy-per-token), B=64	1.0	**17.9**	3.3	77.1

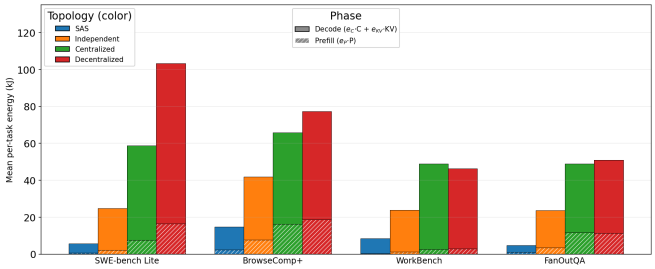


Figure A4: **Prefill/decode energy per topology $\times$ benchmark** (decode solid, prefill hatched; mechanistic split $e_P P$ vs. $e_C C + e_{KV} \Sigma KV$). Bar height (total energy) moves with topology; the hatched fraction (prefill share) moves with benchmark.

A.4.1 Full structural decomposition

Table A7 (§ A.4) reports the two-way ANOVA (classical η^2, Type-II sums of squares, primary-model grid at step budget 10, $n = 1250$; all effects $p < .001$). Energy at B=64 is re-priced via the cache-preserving transport of § A.2: each cell's own measured prefill and KV-context energy is kept, and only its decode term is scaled by $s(64) = e_C(64)/e_C(1)$. Both conclusions survive production batching at reduced magnitude: topology still dominates total energy (2.4$\times$ ratio over benchmark, down from 7.3$\times$ serial) and benchmark still dominates energy-per-token, confirming the structure/workload split of § A.4 holds in both serving regimes. Figure A5 shows the prompt tokens' share of tokens against their share of energy.

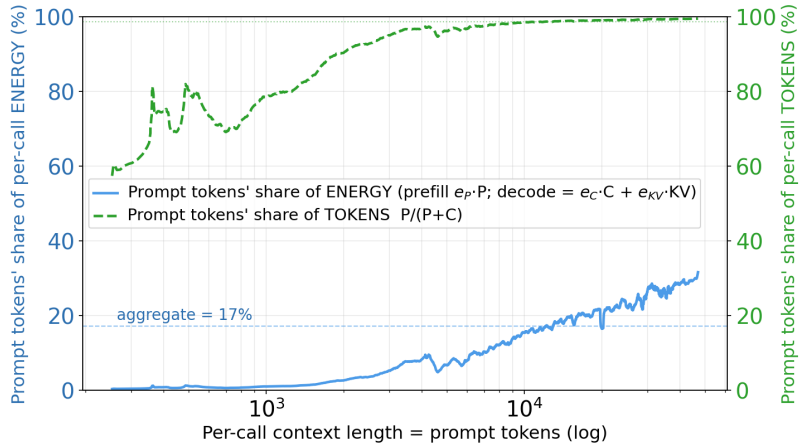


Figure A5: **Prefill vs. decode share of tokens vs. share of energy (serial, batch-1).** Energy is split per call by the § A.1 task-level coefficients (prefill $e_P\,P$; decode $e_C\,C + e_{KV}\,\Sigma KV$). Prompt tokens are most of the tokens but a smaller (growing) share of energy under serial serving: prefill energy share rises from $\sim$1% at short context to $\sim$31% at long context (17% aggregate) while token share rises from $\sim$60% to $\sim$99% (above 90% beyond $\sim$2K context), the token/joule divergence the energy law of § A.1 resolves.

B Model, serving, and benchmark deployment (main paper § 3)

Here, we describe the serving flags, context bounds, and request path of the models we ran, along with the topology protocols and per-benchmark tool environments; the three models and their cards are stated in main text § 3.3.

B.1 TOPOLOGY PROTOCOLS

Here, we describe each topology in full; Figure 1 shows the four visually, and § 3.1 gives the default settings.

SAS: a single agent runs one ReAct loop of at most K tool-calling steps and its answer is final.

Independent: N agents each run the same ReAct loop on the full task, in parallel and isolated at the communication level; no agent receives another's messages (on SWE-bench Lite they share a git worktree, § B.3). A text-only synthesizer then aggregates their answers in a single LLM call with no tool access. Its prompt gives, per agent, a compact summary of the agent's tool trajectory plus its final answer; it is instructed to synthesize the best answer from the findings, not to compare or cross-validate the responses.

Centralized: an orchestrator decomposes the task into N subtasks, one per worker. Each round, every worker runs a ReAct loop of at most K steps on its subtask and keeps its history across rounds. The orchestrator reads the worker reports and either issues individual feedback, one review call per worker (rounds $1 \dots R-1$), or writes the final answer (round R). Workers never communicate directly; all inter-agent information flows through the orchestrator.

Decentralized: N agents work on the full task. Each round, every agent runs a ReAct loop of at most K steps and states its answer; between rounds, each answer is broadcast to all $N-1$ peers and appended to their contexts (final answers only, no tool-call summaries, matching Kim et al.'s channel; round 1 has no peer messages yet). Agents retain their full history across rounds, and after the final round a text-only synthesizer aggregates the last answers. This is essentially R rounds of Independent with full inter-agent communication between rounds.

B.2 MODEL AND SERVING DEPLOYMENT

Here, we document the exact checkpoints, serving flags, and request-path behavior that § 3.3 summarizes. We read all values below from the experiment code (`config.py`, `llm.py`, `run_experiments.py`) and the SLURM launch scripts of the runs they describe.

Model checkpoints: The primary model is `Qwen/Qwen3.5-9B` (dense 9B, hybrid attention), served unquantized in BF16. The two secondary grids use `Qwen/Qwen3.5-35B-A3B-FP8` (35B mixture-of-experts, ∼3B active per token, FP8, launched with `--quantization fp8`) and Gemma 4-31B dense at 4-bit weights (w4a16). The main grid uses Google's quantization-aware-trained checkpoint (`google/gemma-4-31B-it-qat-w4a16`); the pilot used a community post-training AutoRound-AWQ checkpoint of the same format (`mattbucci/gemma-4-31B-it-AutoRound-AWQ`). Both Qwen models run in non-thinking mode (`enable_thinking: false`); Gemma has no thinking toggle and is served with `--reasoning-parser gemma4`.

Hardware: The 9B grid ran on a single exclusively allocated NVIDIA RTX A5000 24 GB per job, except three WorkBench high-memory cells (Decentralized K=2 R=15 N=3, Independent K=20 N=10, Decentralized K=10 R=2 N=10) that overflowed the A5000's ∼49K-token KV pool and ran on an RTX A6000 48 GB with otherwise identical flags. The MoE and Gemma grids ran on the A6000.

Serving: One SGLang server per job (OpenAI-compatible HTTP API on localhost, `--tp 1`), with RadixAttention prefix caching at its enabled default for every grid. SGLang versions: 0.5.10 (Qwen3.5-9B), 0.5.15 (Qwen3.5-35B-A3B), 0.5.14 (Gemma 4-31B; required for the native gemma4 model class and tool-call parser). Launch flags:

- Qwen3.5-9B: `--mem-fraction-static 0.88 --context-length 49152 --tool-call-parser qwen3_coder` (`--context-length 98304` for the SWE-bench Lite SAS and Independent K-sweep)
- Qwen3.5-35B-A3B: `--quantization fp8 --mem-fraction-static 0.90 --context-length 49152 --max-running-requests 4 --cuda-graph-max-bs 4 --tool-call-parser qwen3_coder`
- Gemma 4-31B: `--mem-fraction-static 0.85 --context-length 65536 --tool-call-parser gemma4 --reasoning-parser gemma4`

Caching: Every server runs with RadixAttention prefix caching enabled (SGLang's default), so the repeated prefix of a ReAct trajectory is computed once and served from cache on later calls; measured prefill energy therefore reflects production-style prefix reuse, not full re-computation of the accumulated context at every step.

Context window and truncation: Before every call the client drops the oldest tool-call rounds from the middle of the history (system and task messages always retained) to fit 95% of the context window minus a 4096-token generation reserve, estimating 3 characters per token (conservative); the same window sets the context-scaled tool-return budgets (FanOutQA window/2, BrowseComp+ window/4). The operative window is not the nominal `--context-length` but the KV-cache pool the card can hold after weights, activations and CUDA-graph overhead, which SGLang reports at startup as `max_total_num_tokens`.

The pools are set by KV bytes per token, not GPU capacity. The 9B's hybrid Gated-DeltaNet routes $\sim$75% of layers through constant-size linear attention ($\sim$30 KB/token), yielding a $\sim$49.5K-token pool on the A5000; the bound is stable across the grid (per-call maximum prompt 49,244-49,587 tokens in every long-context cell) and independent of the nominal window. The MoE's GQA keeps its per-token KV comparably small, and its nominal window is deliberately set to 49,152, the pool the A6000 holds, so nominal and operative coincide; the truncation rule therefore targets $\sim$42.6K prompt tokens for both Qwen models, but its character-based estimate is approximate, and observed prompts reach the $\sim$49.5K pool in the longest cells. Gemma's softmax attention (10 global layers; the other 50 sliding-window, § D.2) carries a far larger per-token KV footprint, leaving the 48 GB card only a $\sim$30.3K pool ($\sim$24.7K ceiling): its $K \gtrsim 15$ cells are context-censored, and because a cell at the ceiling stops growing context, Gemma's energy saturates with K in main Figure 6 rather than climbing to the cap (reading consequences in § D.1). The prose cells saturate well below every ceiling; only the depth-scaling SWE-bench Lite cells reach it.

Request path: All requests are serial, non-streaming, batch size 1: one task at a time, and the agents of a topology run sequentially. The client is the OpenAI Python client against the local server (120 s timeout, default transport retries), `max_tokens=4096`, fixed seed (base 42, offset per agent and per round). Context overflow beyond the character-based estimate surfaces as a server-side 400; the runner records it as an overflow call record and retries once at a 0.75$\times$ truncation cap. The plotted N-sweep axes stop at N=5 on WorkBench, as its serving window caps the axis there, and on FanOutQA, whose agent contexts overflow the serving window past that range.

Temperatures: SAS agent 0.0; Independent agents 0.7; Centralized workers and Decentralized debaters 0.5; all orchestrator, review, and synthesis calls 0.0 (`config.py`: SAS_TEMP, INDEPENDENT_TEMP, DEBATE_TEMP, ORCHESTRATOR_TEMP).

Energy measurement: Methodology is in § 3.4. The runner performs 5 warmup calls and a 10 s idle-power measurement per job and writes per-session metadata (GPU name, idle power, CLI args) beside each result file.

B.3 Benchmark deployment documentation

Here, we give the full deployment detail per benchmark that § 3.2 summarizes. Everything below runs locally on the experiment machine (no network calls at run time). The per-benchmark n = 50 fixed tasks below are the K-sweep sets; the N- and R-sweeps extend each to n = 100 by adding 50 more tasks drawn the same way from the same split (§ 3.2), and § 5's probe/held-out splits partition the 50-task K-sweep sets.

FanOutQA (Zhu et al., 2024)

Dataset: FanOutQA (ACL 2024), open-book variant, dev split; n = 50 fixed tasks. Each question has fan-out structure: it decomposes into independent lookups of several Wikipedia entities whose attributes must be combined (e.g. one attribute per entity, aggregated).

Tooling: A single tool, `search(query)`, matching the official open-book protocol (`run_openbook.py`): the query is treated as a Wikipedia article title. On a hit, the article content is chunked (1024 characters per chunk), the chunks are BM25+-ranked against the original question (not the search query), and the top chunks are greedy-packed into the return up to a budget of half the client context window, which makes FanOutQA the longest-tool-return benchmark and the first

to saturate the context window as N or R grows. On a miss, the tool returns similar article titles as suggestions. Wikipedia content is served from a pre-warmed on-disk cache (∼9 GB), so no network access occurs at run time.

Evaluation: Loose accuracy: the fraction of gold answer components found in the model's final text via word-boundary matching after normalization, using the benchmark's own normalizer. (Our runner inlines the official `answer_in_text` logic to bypass a hard BLEURT dependency; when spaCy lemmatization is unavailable it falls back to a lemmatization-free normalizer, which can differ marginally from official scores.)

WorkBench (Styles et al., 2024)

Dataset: WorkBench: 690 tasks over five simulated business databases (calendar, email, analytics, project tasks, CRM) plus multi-domain tasks; n = 50 fixed tasks. Data ships as CSVs and is loaded into pandas DataFrames; each task runs against a fresh in-memory copy, so agent mutations are sandboxed per task.

Tooling: All 27 official tool schemas are exposed (queries and mutations across the five databases). Tool returns are short strings, which is what makes WorkBench the minimal-context-growth benchmark.

Evaluation: Kim et al.'s sequence match (Kim et al., 2026): a task is correct if the gold action set is a subset of the recorded tool calls (gold $\subseteq$ predicted), matched order-insensitively with normalized tool names and argument values. One call recorder is shared by all agents of a task, so in MAS topologies the metric scores the union of every agent's calls (the gold actions may be supplied by any agent or combination of agents). The adapter also implements the benchmark's state-based evaluator (comparing database state after the agent's actions against ground truth); the sequence match is the study's reported metric because exact final-state equality penalizes every exploratory write, and with N agents writing to one shared sandbox that penalty grows with N, so the state metric would measure exploration volume rather than coordination quality.

BrowseComp+ (Chen et al., 2025)

Dataset: BrowseComp+: 830 fact-seeking questions, each identifying a single entity satisfying multiple entangled constraints, against a fixed 100K-document corpus; n = 50 fixed tasks. The dataset ships encrypted with a canary GUID (anti-contamination, not security) and is decrypted at first load; the decrypted copy is cached beside the corpus data on the experiment machine, and the plaintext never enters the code repository.

Tooling: A single `search(query)` tool backed by a local BM25 index over the 100K corpus (built once and pickled); returns are ranked document snippets (512 tokens max each) budgeted at a quarter of the client context window.

Evaluation: Normalized string match against the gold answer: a task is correct if the extracted final answer equals the gold answer or the gold answer appears as a whole word anywhere in the response. The official evaluation uses an LLM judge (GPT-4.1); the string match is the fully-local substitute required by the energy pipeline (no external judge calls). It is strict against paraphrase (paraphrased-but-correct answers score 0) but lenient toward responses that name several candidates, one of them correct.

SWE-bench Lite (Jimenez et al., 2024)

Dataset: SWE-bench Lite (princeton-nlp/SWE-bench_Lite, test split): real GitHub issues to be fixed by a repository patch; n = 50 fixed tasks. Repositories are cloned once; each task checks out the instance's base_commit into a single git worktree shared by all agents of that task (worktree operations serialized with file locks, since concurrent jobs share the clone on networked storage). Agents run sequentially in that shared workspace, and the scored patch is the worktree's git diff at task end (agents call `submit_patch` in only 0-12% of tasks, so the diff fallback dominates); in MAS topologies the scored patch is therefore the union of every agent's file edits, and agents see peers' edits through the filesystem in addition to the communication channel; breadth gains on this benchmark therefore include filesystem-mediated coupling as well as pure ensembling.

Tooling: The Kim et al. tool suite (Kim et al., 2026): `bash` (shell in the workspace), `read_file` (line ranges), `edit_file` (exact-match replace, with a Python syntax lint that auto-reverts breaking edits), `find_file` / `search_dir` (name and grep search), `run_tests` (FAIL_TO_PASS pytest, or Django's runtests for Django instances), and `submit_patch` (captures the git diff, then applies the held-out test patch and runs the FAIL_TO_PASS tests; afterwards further state-changing tool calls are ignored, though the agent loop itself runs on until it emits a final text response or exhausts its step budget). `run_tests` executes in the task's worktree, which does not contain the held-out test patch until `submit_patch` applies it, so it is mostly uninformative.

Evaluation: The official SWE-bench Lite harness, run post hoc: each submitted patch is applied and the instance's FAIL_TO_PASS tests are executed in the instance's Docker image. Records with no patch or a patch the harness cannot process score 0 (they are not dropped), so accuracy shares its denominator with energy and tokens.

Table A8 gives the per-benchmark workload shape at the SAS K=10 cell, which is what places each benchmark on the composition axis of § A.3.

Table A8: **Benchmark workload profiles at the SAS K=10 cell** (n = 50 tasks per benchmark; tokens; tool-return length per step estimated as $P_{next} - P - C$, the prompt growth not explained by the model's own output). WorkBench's tiny structured returns (median 45 tok) are the compound-cost negative control. C/P is the mean of per-task ratios, skewed by a few short tasks (aggregate $\Sigma C/\Sigma P$: FanOutQA 0.010, WorkBench 0.059, BrowseComp+ 0.014, SWE-bench Lite 0.020). BrowseComp+ carries the largest median retrievals and longest contexts; SWE-bench Lite uses its full call budget with the smallest per-call completions.

Benchmark	LLM calls /task	mean prompt tok /call	mean final context (tok)	mean completion tok /call	mean C/P per task	tool return: mean	median	energy (kJ/task)
FanOutQA	5.2	12,819	20,209	124	0.043	6,074	341	5.1
WorkBench	4.3	5,755	6,613	338	0.065	537	45	7.9
BrowseComp+	8.6	17,675	27,106	248	0.018	4,335	2,837	17.1
SWE-bench Lite	11.0	4,222	6,901	84	0.021	560	298	5.4

C SCALING RESULTS DETAIL (MAIN PAPER § 4)

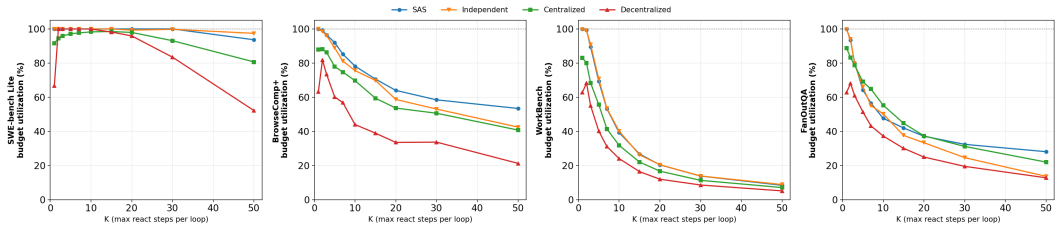


Figure A6: Share of the K budget used (utilization u, § 5.1), per topology and benchmark. Panels share a $K \leq 50$ range; SWE-bench Lite, measured to K=200, stays at ∼20-50% there (not shown). Prose benchmarks decay from ∼100% at K=1 to 5-53% by K=50 as agents converge and stop; SWE-bench Lite stays at ∼100% through $K \approx 25$. This gap underlies § 4.1's split between saturating and still-scaling benchmarks. Decentralized is lowest on prose because debate-round loops often answer without touching tools (one call instead of K+1).

C.1 DEEP-R RUNGS ARE CONTEXT-LIMITED (REFERENCED IN § 4.3)

Debate accumulates context across rounds; the 9B is served in a 49,152-token window with requests truncated at ∼42.6K tokens, so at high R earlier debate history is silently dropped (no run errors). Truncation tracks the accuracy drop on WorkBench Decentralized, whose per-request context grows most with R (mean prompt per call 5.0K to 21.3K tokens from R = 1 to 30):

R	accuracy	tasks with a truncated call
1-10	80-86% (flat)	0-2%
15	77%	6%
30	56%	57%

The decline appears on all four benchmarks (Decentralized, plateau to R = 30: WorkBench 83%→56%, BrowseComp+ 41%→32%, FanOutQA 68%→61%, SWE-bench Lite 28%→24%; n = 100 per cell). The matched-depth null of § 4.3 is therefore read off $R \leq 10$ (truncation ≤2%); the decline at R = 30, and partly at R = 15, is a serving-window artifact, not a property of communication.

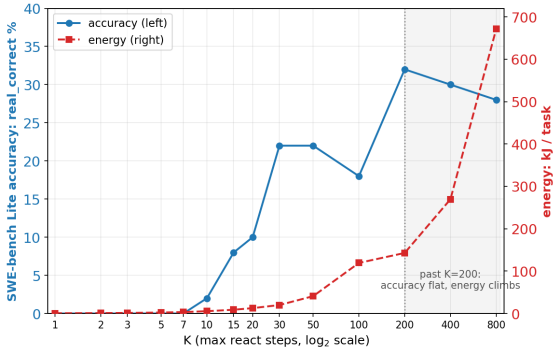


Figure A7: SAS on SWE-bench Lite, accuracy (left) and energy (right), to K=800; K=400 and K=800 are SAS-only runs beyond the K=200 grid ($\log_2$ x-axis, shaded = extended regime). Accuracy rises from 0 through $K \approx 7$ to ∼32% at K=200 (the K=100 dip is n=50 sampling noise, 9-11 no-patch tasks per cell), then flattens at 30% and 28% at K=400 and K=800. Energy climbs monotonically (0.6 to 671 kJ/task, ∼1,100×) as realized ReAct steps keep growing (mean 30 at K=30, 98 at K=200, 276 at K=800). So SWE-bench Lite, which keeps climbing through the grid where the prose benchmarks saturate (§ 4.1), does plateau past K=200; the grid ceiling is not hiding further gains.

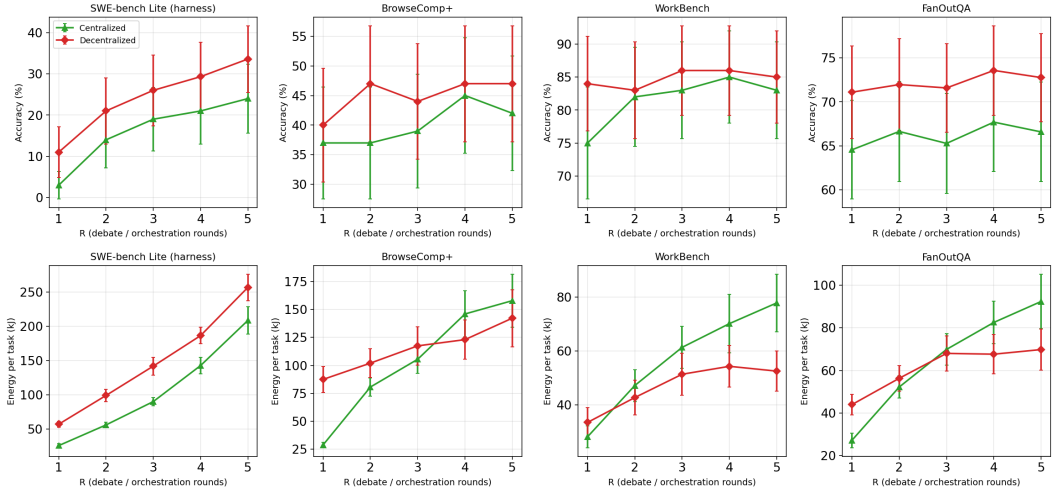


Figure A8: Uncapped R-sweep (§ 4.3): K = 10 per round at N = 3, so per-agent depth grows with R, as in prior round-scaling studies that do not match depth; no budget normalization. Top: accuracy; bottom: energy per task; 95% CIs over non-errored task-runs ($n \geq 100$ per cell). Rounds appear to help on the depth benchmarks (SWE-bench Lite harness 3% to 24% Centralized, 11% to 33% Decentralized; BrowseComp+ 40% to ∼47%) and are flat within CI on FanOutQA and WorkBench, while energy grows with R everywhere (1.6-8× from R = 1 to 5; flattest on prose Decentralized, where consensus early-stopping caps realized rounds). At matched depth (Figure 4) the same axis is null-to-negative, so these gains are purchased by compute, not communication. SWE-bench Lite is harness-scored (unharnessed patch = 0); Decentralized stops early on consensus, so realized rounds fall below R at high R on the prose benchmarks.

C.2 Per-benchmark elasticity detail

Table A9 lists the per-benchmark elasticities behind Figure 5. Each γ is the OLS slope of log(accuracy) on log(mean task energy) over one axis's cells within the common design ($K \leq 50$, $N \leq 5$, $R \leq 30$) for one (benchmark, topology) pair. R-fits use the depth-matched $R \cdot K = 30$ cells; N-fits include each topology's measured $N = 1$ cell where the grid holds one and add the SAS K = 20 cell as a shared matched-budget anchor (the Figure 5 convention; Figure 3 plots the true $N = 1$ runs); zero-accuracy cells are excluded; Figure 5 reports equal-weight row means. ± is one standard error from a paired task-level bootstrap (B = 1,000) that redraws one task sample per benchmark, shared across its cells so task difficulty cancels. The aggregated K-axis means stay positive and both R-axis means negative at 95%; on the truncation-clean $R \leq 10$ range (§ C.1) the aggregated R elasticities are −0.09 [−0.21, 0.02] and +0.09 [−0.03, 0.23] (Centralized/Decentralized), both null (per benchmark in Table A10). The K elasticities are range-sensitive: most of the depth gain comes in the first rungs, and refitting from $K \geq 3$ lowers the topology-mean γ_K to 0.11 (FanOutQA), 0.31 (WorkBench), 0.09 (BrowseComp+) and 0.95 (SWE-bench Lite), which still exceeds γ_N on three

benchmarks but falls below it on BrowseComp+ (0.19). Values are fit from the task-level master records; Figure 5 refits from the per-cell CSVs, shifting the aggregated means by at most 0.05.

Table A9: **Per-benchmark accuracy-energy elasticities γ, with paired-bootstrap standard errors.**

Axis	Topology	FanOutQA	WorkBench	BrowseComp+	SWE-bench Lite
K	SAS	0.64 ± 0.13	1.00 ± 0.29	0.38 ± 0.12	1.14 ± 0.31
K	Independent	0.18 ± 0.05	1.11 ± 0.23	0.34 ± 0.09	1.07 ± 0.26
K	Centralized	0.15 ± 0.04	1.25 ± 0.26	0.20 ± 0.08	0.73 ± 0.16
K	Decentralized	0.17 ± 0.05	0.84 ± 0.21	0.20 ± 0.08	0.98 ± 0.22
N	Independent	0.10 ± 0.05	0.08 ± 0.04	0.24 ± 0.15	0.44 ± 0.28
N	Centralized	0.08 ± 0.04	0.06 ± 0.03	0.12 ± 0.09	0.10 ± 0.19
N	Decentralized	0.08 ± 0.02	0.06 ± 0.02	0.22 ± 0.09	0.41 ± 0.16
R	Centralized	-0.06 ± 0.02	0.01 ± 0.02	-0.38 ± 0.09	-0.17 ± 0.10
R	Decentralized	-0.05 ± 0.02	-0.13 ± 0.03	-0.14 ± 0.07	0.03 ± 0.10

Table A10: R **elasticities on the truncation-free range** $R \leq 10$, with paired-bootstrap standard errors (same protocol as Table A9).

Topology	FanOutQA	WorkBench	BrowseComp+	SWE-bench Lite	mean
Centralized	-0.10 ± 0.03	0.05 ± 0.02	-0.30 ± 0.11	0.01 ± 0.20	-0.09
Decentralized	-0.01 ± 0.02	0.02 ± 0.02	0.08 ± 0.15	0.28 ± 0.22	0.09

D CROSS-ARCHITECTURE VALIDATION

Extended results for Qwen3.5-35B-A3B and Gemma 4-31B; the models are described in § 3.3, checkpoints and serving flags in § B.2.

D.1 CROSS-MODEL SCALING REPLICATION DETAIL

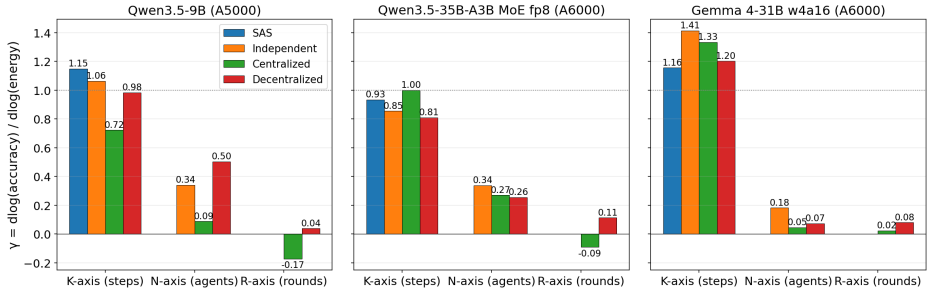


Figure A9: Accuracy-energy elasticity γ by axis and topology, per model (SWE-bench Lite, harness accuracy); the per-model replication of Figure 5. The ordering holds across all three architectures: K (depth) is high everywhere (mean γ ∼0.9-1.3), R (rounds) is near zero (mean -0.07 / +0.01 / +0.05 for 9B / MoE / Gemma; the martingale, replicated), and N (agents) is weakly positive. No error bars: each γ is a single-benchmark fit. Fits use each topology's swept cells only; Table A9 also anchors every N-fit with the SAS $K = 20$ cell, so its 9B N values differ (Independent 0.44 vs 0.34, Centralized 0.10 vs 0.09, Decentralized 0.41 vs 0.50).

On energy, the MoE (∼3B of 35B active, FP8) sits below both dense models in every cell, runs at roughly a third of the 9B's energy per token, and dominates the cross-model energy-accuracy frontier (Figure A10). Quantization acts on the same axis: Gemma has 3.4× the 9B's parameter count but is served at 4-bit (a forward pass reads ∼15.5 GB against the 9B's 18.0 GB at BF16), so its per-token energy is 1.62× the 9B's and its per-task energy on the shared cells averages 0.84×. (Only e_C is strictly comparable across the three fits, § D.2.)

Gemma's energy flattens with K because it stops earlier, not because it is more efficient: at $K = 50$ its single agent (SAS) emits 2,807 completion tokens per task against the 9B's 7,322 and hits the step cap on 20% of tasks against 76%. Most early exits are self-terminations, but 6% are context overflows (none for the 9B), as its larger weights leave a smaller KV pool (§ B.2). The MoE-Gemma gap closes at deep budgets because the MoE's trajectories run 3-4× longer (∼44K against ∼11K completion tokens per task, Decentralized at $K = 50$).

D.2 PER-MODEL ENERGY-LAW REFIT

We refit the through-origin form of equation 1 per model on SWE-bench Lite, the one benchmark all three share; it explains 89–99% of per-task energy variance on each (Table A11). The secondary

models run on the A6000 (too large for the A5000's 24 GB); the § A.2 probe puts the two cards' 9B batch-1 coefficients within 3% (decode) and 7% (prefill), far below the between-model shifts, so the card does not drive the between-model differences. Of the three coefficients, only e_C is strictly comparable across models: e_P absorbs each grid's prefix-cache hit rate and e_{KV} depends on each grid's context range, and both differ with the cell sets (Table A11).

Table A11: Equation 1 fit independently per model on SWE-bench Lite (through origin). Weight bytes/tok is the weight bytes streamed per decode token. Every cell uses the same 50-task subset, so n is 50 times the cell count: 71 cells for the secondary grids, 92 for the 9B, which adds deeper step budgets (100, 200) and N/R sweeps. The 9B row is SWE-bench Lite only for comparability; the pooled four-benchmark fit of § A.1 gives $e_P = 0.016$ and a 335× ratio, because SWE-bench Lite prompts are mostly prefix-cache hits, which lowers e_P. The secondary fits pool July and August A6000 runs, which agree in energy per token within 4%.

Model	**Weight bytes/tok**	e_P	e_C	e_{KV}	e_C/e_P	R^2	**median APE**	n
Qwen3.5-9B	18.0 GB	0.0071	5.58	2.25e-5	784×	0.981	5.6%	4,600
Qwen3.5-35B-A3B	3.0 GB	0.0017	1.59	1.34e-5	943×	0.988	3.3%	3,550
Gemma 4-31B	15.5 GB	0.0144	6.48	3.50e-4	449×	0.889	15.5%	3,550

Each coefficient tracks the architectural feature setting its cost. The MoE streams a sixth of the 9B's weight bytes per token and pays about a third of its decode coefficient; Gemma streams fewer bytes than the 9B yet pays more, consistent with two costs weight bytes miss: its w4a16 kernels dequantize on the fly, and 50 of its 60 layers use sliding-window attention that re-reads a fixed 1,024-token window per generated token. Attention sets the prefill and KV costs: Gemma's 10 global layers carry 16 KV heads against 4 across the 9B's 8 global layers, and its prefill cost is 2× the 9B's, its KV coefficient 16×. Magnitudes differ, but the qualitative conclusions hold on all three models: decode is expensive, KV context does not amortize, and structure and workload act through separate terms. Gemma is the loosest fit (Table A11): equation 1 over-predicts its energy increasingly with context while the other two stay flat, as expected when most of its layers re-read only a fixed window.

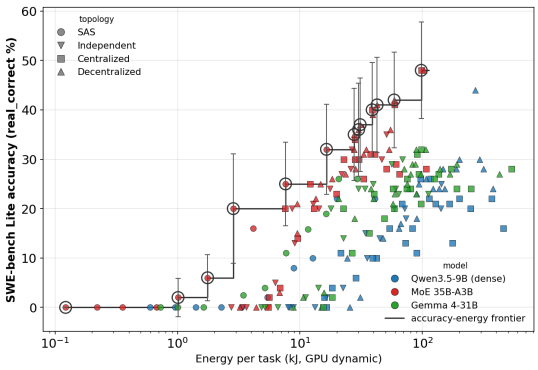


Figure A10: **(cross-model accuracy-energy frontier, SWE-bench Lite).** Every (model × topology × $K/N/R$) cell, per-task GPU dynamic energy (log) vs harness accuracy; color = model, marker = topology, staircase = global frontier (frontier cells ringed, 95% CI on accuracy). Cells span the shared $K \leq 50$ design (the 9B's K = 100/200 cells, up to 56%, are omitted). The MoE 35B-A3B owns all thirteen frontier rungs: it is cheaper at matched accuracy and reaches the highest accuracy (48%), so on these cells the dense 9B and Gemma 4-31B are fully dominated. The frontier re-traces the § 4 order: cheap accuracy from depth (MoE-SAS at low K), then ensembling and coordination only at the high-accuracy tail (MoE Independent/Decentralized).

E Configuration search policies: remaining variants and full results (main paper § 5)

E.1 Threshold ladders, setpoint, breadth coverage, and search accounting

Breadth coverage: Unlike C (fixed anchor depth), D's second phase needs a breadth ladder at whatever depth the probe selects, so we ran the needed $K \times N$ cross product, covering the depths $K^*(\tau{=}0.5)$ the probe lands on (BrowseComp+ 20–50, WorkBench 7, FanOutQA 7–20, SWE-bench Lite 100; the SWE-bench Lite K=100 ladders come from a dedicated fill run, $N \in \{2, 3, 4, 5, 10\}$ on all three topologies at the standard 50 tasks). D is thus deployable on all twelve (benchmark × topology) cells: Figure 8 draws all three topologies on every benchmark, and Table 1's right panel averages all three over all 200 splits (Table A15's D rows keep only splits with a full breadth ladder

at the probed depth). SWE-bench Lite extends the K-ladder to $\{100, 200\}$, and SWE-bench Lite and BrowseComp+ add $N = 10$.

Search-only accounting and amortization: Charged on search alone, A saves 72–87% over B(τ=0.5) (mean 78%) and 81–87% over D(τ=0.5) (mean 83%); the baselines' final rung doubles as their deployment, so this drops only A's separate deployment run. The main text reports the bundled search-and-deploy cost, each arm serving the 25 probe tasks once; Figure A11 shows how A's saving amortizes with deployed tasks.

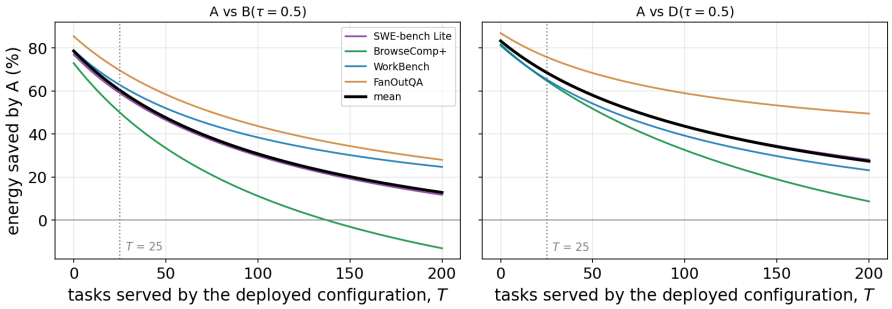


Figure A11: **Energy savings amortize with deployment volume.** A's saving over B(τ=0.5) (left) and two-phase D(τ=0.5) (right) vs tasks T served by the deployed configuration after search (search over the 25 probe tasks), averaged over the three MAS topologies. T=0 is search-only; the dotted line at T=25 is the bundled search-and-deploy quantity.

Table A12: **Stopping-threshold ladders.**

rule	where used	thresholds swept	n
τ	all τ-stopping families	0.30, 0.40, 0.50, 0.60, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95, 0.98	11
δ	all δ-stopping families	0, 0.02, 0.05, 0.08, 0.10, 0.12, 0.15, 0.18, 0.20, 0.22, 0.25	11

Threshold setpoint: Table A13 reports each threshold's **shortfall**, its accuracy gap to the best threshold for the same cell, i.e., the regret from committing to one τ in advance.

Table A13: **Threshold setpoint sweep: accuracy, shortfall, and search-and-deploy energy by** τ (means over the twelve benchmark $\times$ topology cells, 200 probe splits; linear-scan probe, the solid energy curve of Figure 7). The setpoint sweep uses the common ten-rung ladder ($K \leq 50$) on every benchmark, including SWE-bench Lite.

τ	mean acc	mean shortfall	worst shortfall	mean search-and-deploy kJ	vs τ=0.30
0.30	57.1	1.23	8.06	5,833	-
0.40	56.9	1.43	8.06	5,679	3% cheaper
0.50	**57.4**	**0.97**	**3.58**	**5,417**	**7% cheaper**
0.60	56.9	1.47	3.60	4,806	18% cheaper
0.70	55.8	2.55	8.11	4,118	29% cheaper
0.75	55.4	2.92	8.32	3,911	33% cheaper
0.80	53.7	4.63	13.30	3,721	36% cheaper
0.85	52.3	6.11	18.00	3,535	39% cheaper
0.90	52.8	5.62	18.09	3,395	42% cheaper
0.95	50.8	7.60	18.09	2,950	49% cheaper
0.98	49.9	8.48	18.09	2,751	53% cheaper

τ = 0.5 is the recommended setting: averaged over the twelve cells it is both more accurate and cheaper than every lower threshold. Per cell, it never costs more than τ=0.3 (cheaper in 9, tied in 3) and is more accurate in 3, worse in 6, tied in 3. It also has the smallest shortfall (Table A13), while the high-τ thresholds miss by 13 to 18 points in their worst cell.

On SWE-bench Lite agents consume most of their budget well into large K, so a utilization rule rarely stops early. On Gemma, accuracy falls as τ rises past 0.5 on Independent and Centralized and holds until τ = 0.8 on Decentralized, bracketing the 9B setpoint; the rise past τ = 0.8 on Centralized is an artifact of Gemma's early termination at high K (§ D.1), which makes its Centralized accuracy peak at K = 10. On the MoE, every threshold up to 0.9 selects the same configuration, and held-out accuracy falls steeply beyond. Both models corroborate the 9B finding: τ near 0.5 is at or near the best setting, and accuracy collapses well beyond it.

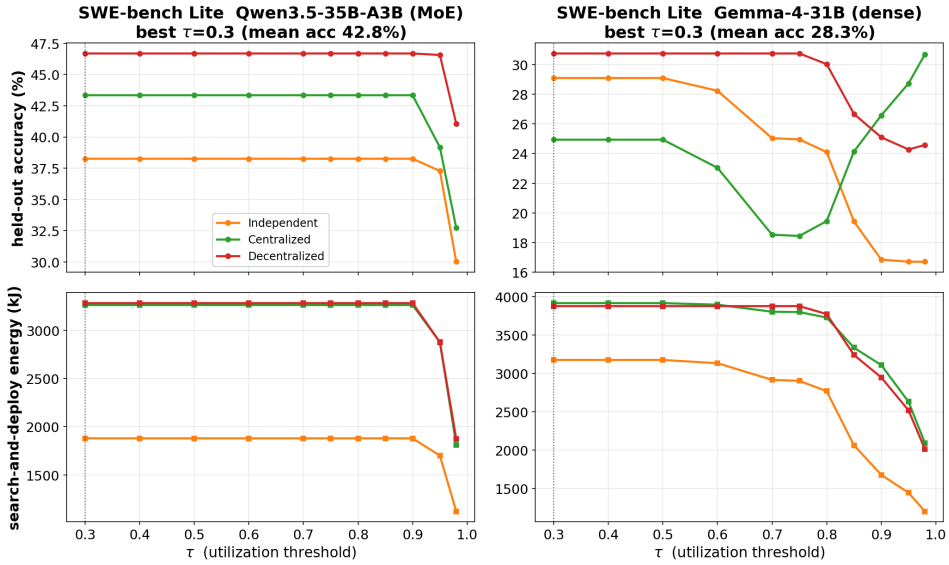


Figure A12: **Threshold setpoint on SWE-bench Lite, secondary models.** Accuracy (top) and search-and-deploy energy (bottom), one column per model, the three MAS targets drawn separately.

E.2 Stopping rule on the depth axis: τ against δ

§ 5.1 stops the depth ladder on utilization; Table A14 instead stops on the accuracy gain between rungs, with ladder and probe fixed, isolating what going label-free costs.

Table A14: $\mathrm{A}(\tau)$ against $\mathrm{A}(\delta)$, the same probe stopped on accuracy instead of utilization, each at its best-accuracy threshold (linear-scan probe); ratios are means of per-topology ratios over the three MAS topologies.

benchmark	**Δ acc ($\tau-\delta$, pp)**	**acc ratio τ/δ**	**energy ratio τ/δ**
FanOutQA	+0.0 pp	1.000×	1.04×
WorkBench	+0.8 pp	1.010×	0.82×
BrowseComp+	+1.9 pp	1.050×	1.31×
SWE-bench Lite	+4.9 pp	1.104×	1.10×

$\mathrm{A}(\tau)$ is slightly more accurate at a small energy premium (1.04× accuracy for 7% more search-and-deploy energy, averaged over the 12 cells), is never less accurate on any benchmark, and gains most on the hardest benchmarks. $\mathrm{A}(\delta)$ also carries the deployment barriers described in § 5.1.

E.3 Finding K^* efficiently: binary search on the depth ladder

Main § 5.1 (Algorithm 1) locates K^* by binary search on the depth ladder: each step runs one SAS probe at the midpoint rung and halves the bracket, so the probe pays for $\sim \log_2 n$ rungs (about four on our ten-rung ladder, and likewise on SWE-bench Lite's twelve-rung ladder) rather than a linear walk. If $u < \tau$ at the lowest rung it deploys that rung; if $u \geq \tau$ at the top rung, the top rung; $u = \tau$ counts as a pass. Because u is monotone, it returns the same K^* as a linear scan, so deployed configuration and held-out accuracy are identical; only probe energy differs. At $\tau = 0.5$ it cuts probe energy by 29% pooled over cells (20–34% per benchmark; the figure quoted in § 5.1), which is 11–15% of total search-and-deploy energy; over the full τ ladder the saving is 3%, since shallow thresholds stop after a few rungs either way.

The saving is largest against the truly naive baseline $\mathrm{B}(\delta)$, which stops on the accuracy gain between rungs (so it needs probe labels) and walks its K-ladder linearly on the deployment topology (Figure A13). Averaged over the three topologies, A's search-and-deploy energy is 0.27×, 0.20×, 0.38× and 0.53× $\mathrm{B}(\delta)$'s on FanOutQA, WorkBench, BrowseComp+ and SWE-bench Lite (mean 0.35× over the twelve cells, a 65% saving), with accuracy if anything slightly higher for A (+0.3 to +1.4 pp per benchmark), because $\mathrm{B}(\delta)$'s label-driven stop is the noisier rule.

E.4 Selection bias of the two stopping rules

The δ rule tends to halt just after a rung that happened to look good on the probe half, biasing that cell's probe-half accuracy upward. The τ rule stops on realized step counts, not the reported metric, so it has no such exposure. In-sample minus held-out accuracy over 200 splits and all thresholds (mean, range over the four benchmarks):

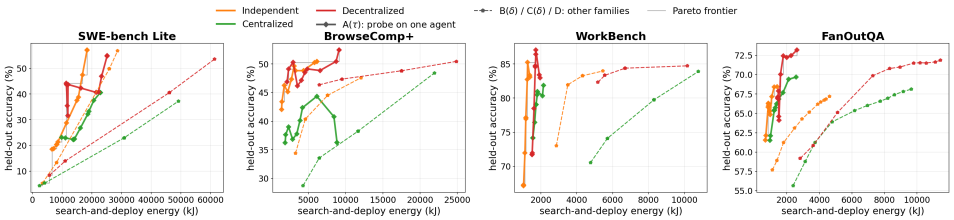


Figure A13: A(τ) **[binary-search probe] against** B(δ) **[linear scan].** Same axes as Figure 8. A binary-searches K^* on a single agent; B(δ) walks its K-ladder on the deployment topology and needs labels. This is the naive baseline behind the saving quoted in § 5.2; the label-free comparison, A against B(τ) with both binary, is Figure 8.

rule	probe signal	deployed config
δ	+3.27 pp (+2.4 to +4.4)	+0.03 pp (-2.2 to +1.8)
τ	-0.33 pp (-1.4 to +0.3)	-0.40 pp (-1.7 to +0.7)

δ's probe-signal bias (about 3 points on every benchmark; none for τ) is why the protocol scores on a disjoint half. It does not reach the reported numbers: the probe noise that drove the stop is only weakly correlated with the deployed configuration's accuracy, and every result here is held-out.

E.5 Why δ and not τ on the breadth axis

Both N-scaling families, C and D's second phase, stop on δ because τ does not transfer to breadth: adding agents does not change any single agent's step cap, so utilization is nearly invariant in N. At each topology's anchor K, the total swing in u across a cell's N ladder is under 0.14 in eleven of the twelve (benchmark $\times$ topology) cells, and as low as 0.001-0.03 on Independent, against a threshold grid spanning 0.30 to 0.98 (FanOutQA Centralized, at 0.261, is the one cell where a τ threshold could land mid-ladder). A threshold on a flat signal is a step function: it clears at N=2 and runs the ladder out, or fails at N=2 and deploys two agents, regardless of whether breadth helps. Best-setting accuracy, Decentralized target:

policy	FanOutQA	WorkBench	BrowseComp+	SWE-bench Lite
D(τ, δ)	71.4	85.0	59.7	39.6
D(τ, τ)	66.5	76.7	37.7	30.5

E.6 The breadth-only baseline: C(δ)

C is the breadth-only control: it tunes only the agent count. Independent is anchored at K=20, Centralized and Decentralized at K=10, the depths of the original N-sweep (§ 4.2). It stops on δ because utilization barely moves in N (§ E.5), and δ often lets it run to the top rung (its most-selected N^* is the top available rung in six of twelve cells).

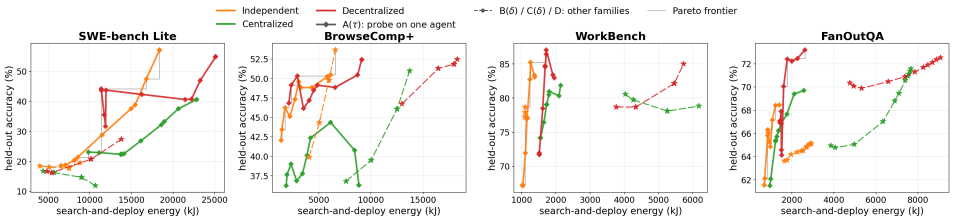


Figure A14: A(τ) **against** C(δ)**, the breadth-only ladder.** Same axes as Figure 8. A(τ): heavy filled diamond, solid line; C(δ): star, long dash; one curve per deployed topology. A uses the binary-search probe; C holds depth at its anchor and walks the agent count on the deployment topology, so the horizontal gap is what a breadth sweep costs against a depth probe. A dominates on WorkBench and SWE-bench Lite, reaching higher accuracy at lower search-and-deploy energy; on SWE-bench Lite the fixed anchor caps C an order of magnitude below where accuracy lives. On BrowseComp+, C(δ) Independent takes the frontier's top rung (a -5.2 pp mean shortfall).

On FanOutQA and WorkBench, A roughly matches C's accuracy at 34% to 73% less energy. On BrowseComp+, the benchmark with the highest γ_N/γ_K (0.6–1.1$\times$ across MAS topologies, Table A9) and the one where scaling breadth past N=3 pays, C is slightly more accurate: the N=5 rung genuinely helps, and C (like D's δ-sweep, which stops only after two consecutive rungs fail the δ test; main § 5.1) walks past the single-task N=4 dip to reach it. On SWE-bench Lite, A buys 2.4$\times$

the accuracy for 2.7× the energy (Table A15), and at strict utilization thresholds improves accuracy at the same energy cost as C (Figure A14).

E.7 FULL SEARCH RESULTS

Table A15 gives every family at both stopping rules, per benchmark, averaged over the deployment topologies; each arm is at its own best-accuracy threshold, accuracy is held-out (choose on 25 probe tasks, score on the disjoint 25, mean over 200 splits), and energy is search-and-deploy over the probe tasks. Probe energies are linear-scan; the binary-search probe of Algorithm 1 deploys the identical configuration at 20–34% lower probe energy (§ E.3), so the τ arms are conservatively priced.

Table A15: **Every search family at its best-accuracy threshold, averaged over deployment topologies, per benchmark.** Held-out accuracy (%) and search-and-deploy energy (kJ); τ-stopped left, δ-stopped right (for two-phase D, the second phase's rule: $\mathrm{D}(\tau,\tau)$ left, $\mathrm{D}(\tau,\delta)$ right; $\mathrm{C}(\tau)$ is reported for completeness since u barely moves in N, § E.5). Unlike Table 1, which pins $\tau = 0.5$, each arm takes its own best threshold (for A on SWE-bench Lite, $\tau = 0.3$, deploying $K = 200$), and D averages only the splits whose depth probe lands on a K with a full N-ladder (under half of the splits in several Centralized cells), so A and D rows are not paired.

		τ-stopped		δ-stopped	
Family	**Benchmark**	**acc (%)**	**energy (kJ)**	**acc (%)**	**energy (kJ)**
A	FanOutQA	70.4	2,795	70.4	2,683
A	WorkBench	84.7	1,984	83.9	2,446
A	BrowseComp+	49.1	8,543	47.2	7,345
A	SWE-bench Lite	50.9	23,901	46.0	21,739
B	FanOutQA	70.5	8,280	69.1	8,570
B	WorkBench	84.1	5,060	84.2	8,800
B	BrowseComp+	49.7	17,926	48.8	19,626
B	SWE-bench Lite	45.6	38,288	49.6	43,837
C	FanOutQA	70.6	5,104	69.7	6,549
C	WorkBench	79.9	3,317	81.4	3,620
C	BrowseComp+	53.7	12,875	52.4	12,846
C	SWE-bench Lite	20.7	11,208	21.2	9,010
D	FanOutQA	68.6	4,156	71.0	6,074
D	WorkBench	80.5	4,305	83.8	5,069
D	BrowseComp+	47.5	20,003	59.7	30,605
D	SWE-bench Lite	28.9	25,475	40.9	48,394

Table A16: **Configurations each family deploys at $\tau = 0.5$.** Modal deployed configuration (depth K, agent count N) over the 200 probe splits; D takes its best-accuracy δ, shown in its column ($\delta = 0$ stops only after two consecutive rungs lower accuracy). A probes depth on a single agent and deploys $N = 3$ at K^*; B runs the same rule on the target topology and often stops at a different depth; D keeps A's K^* and mostly escalates breadth past $N = 3$. On FanOutQA the probe splits almost evenly between $K^* = 7$ (47%) and $K^* = 10$ (44%; $K^* = 10$ on the full 50-task set, as in Figure 8); D's mode sits at $K = 10$ because Centralized has a full breadth ladder only there and Independent's $K = 7$ landings spread over several N.

benchmark	**topology**	$\mathrm{A}(\tau{=}0.5)$	$\mathrm{B}(\tau{=}0.5)$	$\mathrm{D}(\tau{=}0.5,\delta)$
SWE-bench Lite	Ind.	K=100, N=3	K=200, N=3	K=100, N=5, δ=0
	Cent.	K=100, N=3	K=100, N=3	K=100, N=4, δ=0.02
	Dec.	K=100, N=3	K=50, N=3	K=100, N=2, δ=0.1
BrowseComp+	Ind.	K=50, N=3	K=30, N=3	K=50, N=10, δ=0
	Cent.	K=50, N=3	K=30, N=3	K=50, N=10, δ=0
	Dec.	K=50, N=3	K=10, N=3	K=50, N=5, δ=0
WorkBench	Ind.	K=7, N=3	K=7, N=3	K=7, N=5, δ=0
	Cent.	K=7, N=3	K=5, N=3	K=7, N=5, δ=0
	Dec.	K=7, N=3	K=5, N=3	K=7, N=2, δ=0.02
FanOutQA	Ind.	K=7, N=3	K=10, N=3	K=10, N=5, δ=0
	Cent.	K=7, N=3	K=10, N=3	K=10, N=5, δ=0
	Dec.	K=7, N=3	K=5, N=3	K=7, N=3, δ=0.02