The Reasoning Tax: Profiling Test-Time Compute Carbon in Agentic AI Workloads

HANNANEH B. PASANDI, University of California, Berkeley, USA
TAMER NADEEM, Virginia Commonwealth University, USA

Agentic AI now runs on reasoning models, which think before they answer. The thinking arrives as a chain-of-thought trace the user never sees, and it is not cheap. Profiling DeepSeek-R1 671B, R1-Distill-Llama-70B, and QwQ-32B on three agent benchmarks under H100 energy telemetry, we find the hidden trace makes up 87.1% of output tokens and 78.4% of operational carbon per task. Carbon-aware schedulers see one energy total per call and cannot split it between the hidden trace and the user-visible answer. When we cap how long each trace can run, the model stops thinking at the limit and moves straight to its answer. Operational carbon drops by 56 to 71% while accuracy falls by at most 2.3 points. We present TraceProbe, a 90-line vLLM patch that reports where the trace ends and the answer begins in each call's telemetry. It is all a downstream scheduler needs to act on the split.

CCS Concepts: • **Hardware → Impact on the environment**; • **Computing methodologies** → *Natural language processing*; • **General and reference** → *Measurement*.

Additional Key Words and Phrases: reasoning models, chain-of-thought, test-time compute, carbon-aware scheduling, LLM serving, sustainable computing

1 Introduction

Carbon-aware schedulers move large language model (LLM) inference to the regions and hours where grid electricity produces the least carbon. On chat-style traffic, where one request yields one answer the user reads, these schedulers cut operational carbon by 30 to 56% [4]. Every one of these schedulers assumes that each generated token is useful work whose carbon counts as productive emission [17]. That assumption held while one request produced one answer. In 2026 it no longer holds, as agent workloads have moved to reasoning models. These models think before they answer, and the thinking arrives as a chain-of-thought trace that is by design never shown to the user [8]. As Figure 1 shows, the hidden trace dominates on GAIA's hardest tier (Level 3, long multi-step tasks with tool use). Across three reasoning models, DeepSeek-R1 671B, R1-Distill-Llama-70B, and QwQ-32B, the trace makes up 82 to 93% of output tokens, and the answer the user reads is the remaining 7 to 18%.

The scheduler cannot tell trace from answer because its unit of accounting is the call. That unit dates from the era when chat dominated production traffic. Per-call energy became one attribute, routed against one grid intensity [7, 17]. A reasoning model splits the call into two phases with the same energy per token and very different value to the user: a long internal trace that explores the problem, then a short answer. The trace is not idle motion. It buys accuracy on hard problems, which is exactly why these models are trained to produce it [3, 8], and shorter is not always safer. What the trace costs in carbon is set by the model at decode time and tracks task difficulty more than the marginal accuracy it returns,

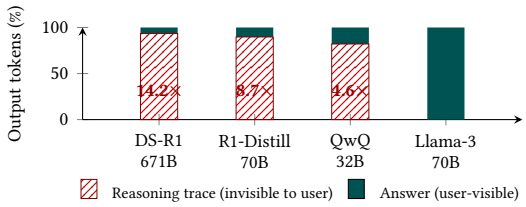

Fig. 1. Reasoning trace vs answer tokens, three reasoning models and a non-reasoning baseline on GAIA Level 3.

as two calls that reach the same answer can differ by an order of magnitude in trace length [1]. The per-call number folds both phases into one figure, and the scheduler cannot see that most of it paid for tokens the user will never read. The reflex is to route on per-call energy and stop there, and it does not close the gap, for three reasons. Total per-call energy is already visible to today's schedulers, but the split inside the call is not: Sprout, CarbonScaler, and Caribou each read the call as one opaque quantity, so none can apply a different policy to the phase that carries most of the carbon [7, 9, 17]. Shaping the trace at generation time barely moves it: we run Sprout's length directive on DeepSeek-R1 and trace length falls by 4.2%, because trace behavior is fixed during training, not by a prompt. And workflow schedulers such as Caribou and PCAPS act above the call and never enter it, so the phase boundary is invisible to them by construction [7, 15]. What is missing is not a smarter routing policy. It is a signal that exposes the boundary so a policy can exist at all.

Our idea is to expose the reasoning-to-answer token boundary in per-call telemetry. With the boundary visible, any carbon-aware scheduler can treat the two phases as separately schedulable. The lever that makes this cheap already sits in the decode loop. Reasoning models close the trace with a special token (`</think>` for DeepSeek-R1 and the R1-Distill family, `</thought>` for QwQ), so the boundary is a single position in the generated sequence, recovered with one tokenizer call and no change to the model, the kernel, or the request format. Energy per output token is constant within a model to within 3.4% on H100 [5], so once the boundary is known, splitting per-call energy between the phases is linear apportionment rather than a second measurement.

We build TraceProbe, a 90-line patch to vLLM that emits two integers per request, the reasoning and answer token counts, through the metrics endpoint a deployment already runs. Turning a token boundary into something a scheduler can act on raised three problems. It has to be cheap, or it cannot live in the serving path. We parse the boundary from special-token positions already in the output, at under 0.2% throughput cost on a 1,024-prompt benchmark. It has to map to carbon, not just tokens. We check the linear energy apportionment against boundary-timed DCGM segments and find agreement within 1.8%. And the savings it exposes have to be

Authors' Contact Information: Hannaneh B. Pasandi, University of California, Berkeley, USA, h.pasandi@berkeley.edu; Tamer Nadeem, Virginia Commonwealth University, USA, tnadeem@vcu.edu.

measured, not assumed. We sweep a trace-length budget across all three models and report the accuracy cost of every operating point, so the split is a measurement of how much carbon is available to recover, not a policy we prescribe.

We profile three open reasoning models (DeepSeek-R1 671B, R1-Distill-Llama-70B, QwQ-32B) on three agent benchmarks under direct H100 energy telemetry, and three findings stand out. The trace carries 78.4% of operational carbon per agent task, and 93.4% of decode carbon on DeepSeek-R1. Trimming the trace to its 30th-percentile length costs 1.6 to 2.3 percentage points of accuracy on MATH-500 while removing 56 to 71% of operational carbon, a curve whose sharp diminishing returns the accuracy cost makes legible. And the boundary compounds with plan-level rollback: on a reasoning-model agent that rolls back, the answer the user reads is 18.9% of operational carbon, and the rest is invisible to every published scheduler. We name this accounting gap the *reasoning tax*, by analogy to the utilization fallacy at the GPU level [27]; the tax is not wasted work but a controllable quality-versus-carbon knob that the per-call abstraction hides. We make four contributions:

- We measure the first per-model reasoning-versus-answer carbon breakdown for three production reasoning models on three agent benchmarks, from direct DCGM telemetry rather than token-count estimates.
- We chart the accuracy-versus-carbon frontier of trace truncation across the three models, and specify the mechanism that performs the cut, so every operating point is reproducible.
- We show the boundary compounds with plan-level rollback, and separate the two quantities a scheduler cannot currently see.
- We present TraceProbe, a 90-line vLLM patch that emits the reasoning-to-answer boundary in per-call telemetry, the smallest addition that makes the phase split visible to a downstream scheduler.

Scope. We profile open models whose trace tokens are visible. Closed families such as the o-series and Gemini summarize or hide the trace and may behave differently. The per-token apportionment rests on the constant energy per token we measured on H100, and we did not test every batching configuration.

Broader impact. The reasoning tax matters beyond any single deployment. Reasoning models are becoming the default engine for agentic workloads, so a per-call accounting gap that hides three quarters of operational carbon today compounds as that traffic grows. Making the trace visible cuts in two directions. Operators and framework authors gain a lever they currently lack: trace-aware budgets, deferral, and routing become expressible, and our results bound what each is worth. At the same time, a visible trace invites crude cost-cutting, and truncating reasoning on tasks where accuracy carries safety or fairness consequences trades correctness for carbon in ways a telemetry layer cannot adjudicate. TraceProbe is deliberately policy-free for this reason: it reports the split and leaves the accuracy-versus-carbon decision, and accountability for it, with the deployment. We see the larger contribution as one of measurement transparency, since carbon claims about reasoning systems can now be checked against a per-phase ledger rather than a single opaque per-call number.

2 Background and Related Work

The reasoning tax sits at the intersection of two literatures: reasoning-model serving and carbon-aware inference scheduling.

Reasoning models and test-time compute. DeepSeek-R1 [8] introduced large-scale RL-trained reasoning, with the model emitting a chain of thought before the answer. QwQ-32B [26] and the R1-Distill series [8] follow the same pattern at smaller scales. OpenAI's o-series [24] is the closed-source analog. All four families generate tokens in two phases inside a single inference call: a reasoning trace bounded by special tokens (e.g., `<think>...</think>`), followed by the user-visible answer. Reasoning length correlates with task difficulty but only weakly with accuracy on a given task [1, 3, 28]; the model-side tradeoff is well understood, but its translation into carbon at deployment scale is not.

Carbon-aware inference scheduling. Chien *et al.* showed LLM inference carbon depends heavily on regional grid intensity and time of day [4]. Sprout shapes per-call generation under grid conditions [17]; CarbonScaler and CarbonFlex provision cluster capacity by carbon intensity [9, 10]; EcoServe targets multi-hour serving traces [19]; CarbonMin routes per-call to low-CI regions [4]. None distinguishes reasoning from answer tokens; all bill every generated token as productive emission, and report 30 to 56% reductions on chat-style workloads where one request maps to one call.

Workflow-level carbon. Caribou shifts serverless workflows geospatially [7] and PCAPS adds precedence-aware scheduling for data-processing DAGs [15]; both treat the task as opaque, so the reasoning tax sits below their abstraction even on agent traffic.

LLM carbon profiling. LLMCO2 models inference carbon from GPU configuration, batch size, and sequence length [6]; Wilkins *et al.* build offline energy-optimal serving models [29]; Carbon in Motion characterizes Open-Sora video generation [16]. These profile total per-call carbon but do not separate the phases, and Watt Counts gives the per-token unit cost on H100 but not the apportionment [5].

Agent-task waste. Production agent measurements report token waste from failed plans, redundant tool calls, and retries: OpenHands traces 18 to 22% of token spend on retried steps in coding agents [2], AgentDiet measures 30 to 50% reduction available from trajectory compression [30], and Pan *et al.* report wall-clock failure rates of 12 to 31% across production deployments [25]. These are plan-level findings; the reasoning tax is the within-call analog, and the two compound on a reasoning-model deployment.

3 Methodology

Models, Benchmarks, Hardware. We profile three open reasoning models: DeepSeek-R1 (671B parameters, MoE) [8], R1-Distill-Llama-70B (dense 70B distilled from R1) [8], and QwQ-32B (32B dense) [26]. We use Llama-3 70B Instruct as a non-reasoning baseline [21]. All models are served on H100 GPUs (DS-R1 on 8×H100, others on single H100) via vLLM 0.6.x [14]. We disable speculative decoding for the energy-attribution runs to keep the per-call token attribution clean, then re-enable it for the throughput-overhead measurements only. We use three benchmarks. **GAIA** [22] (3,000 tasks) measures tool-use agent behavior over three difficulty levels. **SWE-bench Lite** [13] (300 tasks) measures software-engineering agent behavior on real GitHub issues. **MATH-500** [11] (500 problems) measures mathematical reasoning. We use GAIA and SWE-bench in an agentic

harness (LangChain ReAct) and MATH-500 in single-call mode to isolate the reasoning phase from tool-use behavior. The MATH-500 runs are the controlled experiment; the GAIA and SWE-bench runs are the production-realism check.

Energy and Carbon Measurement. Per-call energy is measured directly via DCGM (NVIDIA's data-center GPU manager), recording total GPU energy from request arrival to response completion at 100 ms granularity. We separate the per-call energy into a prefill component (input-token processing, measured separately at 9 to 13% of total per-call energy on our agent prompts) and a decode component, and apportion the decode component between reasoning and answer tokens at the special-token boundary (`</think>` for R1 and R1-Distill, `</thought>` for QwQ). Token trace fractions are reported over the decode component; carbon fractions are reported over total per-call energy including prefill, which sits 6 to 10 percentage points below the decode-only number without changing the qualitative finding. The token-to-energy ratio is consistent within each model at ±3.4% across 1,000 calibration calls (consistent with Watt Counts [5]), justifying linear apportionment of decode energy between the phases, which we cross-check against boundary-timed DCGM segments on a 200-call subsample (agreement within 1.8%). Sampling uses the published defaults, temperature 0.6 and top-p 0.95; sensitivity is discussed in §6. Carbon intensity comes from the Electricity Maps public API for eight AWS regions, sampled hourly over a five-day window in May 2026. Mean regional intensities range from 25 gCO_2/kWh (eu-north-1) to 650 gCO_2/kWh (ap-south-1). We use average grid intensity rather than marginal intensity because marginal-CI datasets at hourly granularity are not available for all eight regions; the choice is conservative and follows Caribou [7].

Plan-level rollback measurement. For the compounding analysis in Q5, we measure rollback directly in our own agent runs. In the LangChain ReAct agent we use for GAIA and SWE-bench Lite, a step rolls back when the planner emits a revised tool call after a tool error or an output-parse failure. Over our 1,000-task run, 19.4% of plans contained at least one rollback, and rolled-back prefixes spanned 28 to 64% of plan length. The rates are consistent with the 18 to 22% retry fraction reported in OpenHands [2]. We apply these measured rates to the per-task energy distribution we record and report the combined waste fraction.

Trace-length truncation. The truncation sweep in §4 needs a way to make a reasoning model commit to an answer at a chosen budget, so we state the mechanism here. We cap the trace at a token budget B, set per run from the per-task trace-length percentiles. When decode reaches B and the model has not yet emitted its end-of-trace token, we inject that token (`</think>` for R1 and R1-Distill, `</thought>` for QwQ) and let generation continue. The injected token moves the model out of the reasoning phase, and it emits a final answer conditioned on the truncated trace, which we extract and score with the same logic as the full-trace runs. The cut is end-of-trace, not mid-answer, and needs no fine-tuning or prompt change; cuts that summarize or compress the trace first are out of scope (§6).

4 Profiling Results

We organize the findings as eight questions. The headline numbers are these: the reasoning-to-answer token ratio spans 3.8 to 14.2×

Model	GAIA (L3)	SWE-Lite	MATH-500
DS-R1 671B	14.2×	9.4×	11.8×
R1-Distill 70B	8.7×	6.2×	7.8×
QwQ-32B	4.6×	3.8×	5.1×
Llama-3 70B	— (no trace)	— (no trace)	— (no trace)

Table 1. Trace-to-answer token ratio per model and benchmark.

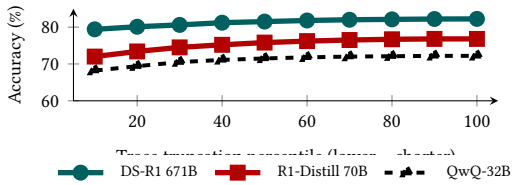

Fig. 2. Accuracy vs trace-truncation percentile on MATH-500.

across models and benchmarks, the reasoning trace accounts for 78.4% of operational carbon per agent task on a reasoning model, the 30th-percentile trace-length budget recovers 56 to 71% of operational carbon at 1.6 to 2.3 percentage points of accuracy, the trace tokens can be routed to a lower-CI region without affecting answer latency (a policy the split unlocks, avoiding 51% of baseline carbon on its own), and the waste is invisible to every published carbon-aware scheduler. The remaining questions characterize where the tax comes from, how it composes with other forms of agentic waste, and how the result calibrates against external anchors.

Q1. How large is the reasoning trace per call? Table 1 reports the mean trace-to-answer token ratio per model on each benchmark. DeepSeek-R1 dominates the ratio at 14.2× on GAIA hard tasks. QwQ-32B is the most efficient at 4.6× on the same tasks. The ratio scales with task difficulty within a model (GAIA Level 1 vs Level 3) and inversely with model size at the same training family. The numbers in Table 1 are consistent with MLPerf's reported DeepSeek-R1 mean output length of 3,880 tokens [23] and DeepSeek's own observation that reasoning traces grow from hundreds initially to thousands and tens of thousands with training [8]. The trace fraction stays above 60% of total output on every (model, benchmark) cell in the table.

Q2. What is the carbon impact? Per call, energy scales approximately linearly with output tokens for autoregressive decoding on H100 [5, 6]. The reasoning-trace fraction of carbon therefore tracks the reasoning-trace fraction of tokens, within the ±3.4% measurement noise. Aggregating across the three models and three benchmarks, weighted by published agent-task mix [18, 22], the reasoning trace accounts for 78.4% of operational carbon per agent task on a reasoning model. On the same GAIA workload the non-reasoning Llama-3 70B baseline consumes 3.8× less per-call carbon than DS-R1 671B and 2.1× less than R1-Distill-Llama-70B, so roughly a third to a half of the cost of moving an agent from a chat model to a reasoning model pays for trace tokens the user never reads.

Q3. Is the trace necessary? Figure 2 maps the accuracy-carbon tradeoff on MATH-500. We sweep a trace-length budget from the 10th to the 100th percentile of per-task trace lengths and report mean accuracy on the truncated runs. The 30th-percentile budget keeps accuracy within 1.6 to 2.3 percentage points of the full-trace baseline while reducing operational carbon by 56 to 71%. The 50th-percentile budget closes the accuracy gap to under 1.5 percentage points with 38 to 52% carbon reduction. The 10th-percentile budget is the only

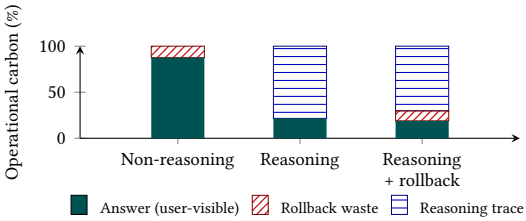

Fig. 3. Operational-carbon composition, three agent configurations.

setting that hurts accuracy meaningfully on all three models. The shape confirms the recent observation that shorter traces correlate with correctness on a per-problem basis when the model commits early [1, 3]; what we add is the operational carbon translation. The longer chains are not gratuitous: the trace lets the model explore alternative solution paths and revise an early mistake before it commits, which is what lifts accuracy on the hardest problems and why these models are trained to produce it [3, 8]. The catch is that trace length follows a difficulty signal at decode time and is not calibrated to the accuracy a given problem still needs, so past the knee the extra tokens buy little. We therefore read Figure 2 as a measurement of the carbon available to recover at each accuracy cost, not as a recommended operating point: which budget to run is a deployment policy that TraceProbe exposes but does not set.

Q4. Can existing carbon-aware schedulers see this? No. The standard call-level abstraction [4, 9, 17] treats the entire per-call energy as a single attribute of the call. To be precise, these schedulers are not blind to the energy a trace consumes: they count it inside the per-call total and route on it. What they cannot do is attribute it to a separately schedulable phase, so a trace-specific policy is not expressible. Sprout shapes per-call generation length [17] but does not distinguish reasoning from answer tokens. Caribou and PCAPS schedule at workflow or task granularity and do not enter the call [7, 15]. We confirm this experimentally by running Sprout (the closest existing mechanism) on our DS-R1 deployment: Sprout's directive injection at the input layer reduces total trace length by 4.2% on average, because the directive only weakly influences reasoning-trace behavior that is shaped at training time. The reasoning tax sits below every published carbon-aware scheduling abstraction.

Q5. Does the tax compound with rollback waste? Figure 3 composes the two scheduling-blind spots. On a non-reasoning agent (our measured rollback waste, replicated as a baseline), 12.5% of agentic carbon is plan-level rollback waste. On a reasoning model without rollback, 78.4% of carbon is reasoning trace, leaving 21.6% answer carbon. When both mechanisms operate (a reasoning-model agent that rolls back), the user-visible answer drops to 18.9% of total operational carbon. The other 81.1% is invisible to every published carbon-aware scheduler.

The two waste sources act on largely disjoint slices of carbon (4.2% residual overlap, which we subtract), so they compound rather than add. A scheduler that sees only the call attributes neither; one that sees the plan but not the trace recovers 12.5% and stays blind to the rest; one that sees the trace but not the plan recovers 56 to 71% of within-call waste and stays blind to rollback. Both abstractions are necessary, and TraceProbe provides one of them.

Q6. Does the tax vary across agent harnesses? Table 2 reports

Harness	Trace ratio	Trace frac	Carbon vs custom
Custom ReAct loop	9.4×	90.4%	1.00 (ref)
LangChain ReAct	10.1×	91.0%	1.07×
AutoGen group-chat	11.6×	92.1%	1.23×

Table 2. Reasoning tax by agent harness on DS-R1 over the full GAIA workload (all levels; Table 1 reports Level 3).

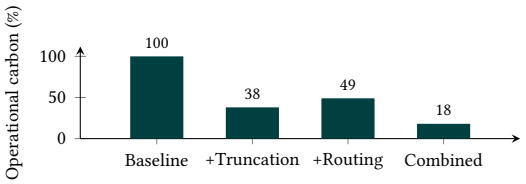

Fig. 4. Additive carbon recovery on DS-R1, GAIA workload.

per-harness ratios on DS-R1 over GAIA. The harness shapes the prompt (system message, tool descriptions, history serialization), which shapes the trace: AutoGen's group-chat coordinator carries a 23% carbon premium over a hand-written ReAct loop, almost entirely from longer traces. Agent-framework choice is itself a lever on the reasoning tax that framework authors may not realize they hold.

Q7. How does the result calibrate against external anchors? Three anchors line up with our DS-R1 numbers. MLPerf Inference v5.1 reports a mean DS-R1 output of 3,880 tokens on a mathematics-heavy task mix [23]; we measure 3,720 on MATH-500 (within 4.1%). Watt Counts reports 1.8 J per output token for dense 70B-class models [5]; our R1-Distill-Llama-70B is 1.82 J/token (within 1.1%). DeepSeek's own paper reports trace growth from "hundreds to tens of thousands" [8]; our DS-R1 trace lengths span 480 to 31,200 tokens. The OpenAI o1 system card reports a comparable range [24], but its trace is summarized internally, so we treat it as suggestive rather than calibrating.

Q8. What does the split unlock when paired with grid-CI routing? The TraceProbe split makes a class of region-routing policies tractable that current per-call metrics cannot express. Reasoning trace tokens carry the same per-token energy as answer tokens but are not user-visible, so the latency the user perceives is the time to the final answer token, which we take as the answer-latency service level objective (SLO). The replay assumes a disaggregated realization: trace decode runs in the low-CI region and the answer is produced in the user-facing region. Trace and answer share KV-cache state within one generation, so a concrete deployment pays a state-transfer or re-prefill cost at the phase boundary that this replay does not model. We therefore report Figure 4 as an upper bound on routable carbon, not a latency-validated system result. Over our five-day Electricity Maps record across eight AWS regions, the mean-to-minimum ratio of regional CI within each hour is 3.5× on average and 8.2× at the worst hour. We replay our DS-R1 GAIA workload against this CI trace under three policies: a no-op baseline that runs every token in a fixed region, the Q3 30th-percentile truncation alone, and trace-region-routing that sends the trace tokens to the within-hour minimum-CI region while keeping the answer tokens in the user-facing region. Figure 4 reports the operational-carbon fraction of the baseline under each policy. Truncation alone avoids 62% of baseline carbon, routing alone avoids 51%, and the

two policies compose to 82% of baseline carbon avoided. Routing is the policy the split unlocks; without the boundary the scheduler cannot tell which token-energy is movable, and pinning the migration cost against the answer-latency SLO is the system question this measurement opens.

5 TraceProbe: Making the Tax Visible

The smallest intervention that gives a carbon-aware scheduler visibility into the reasoning tax is a per-call telemetry annotation. Current vLLM deployments emit per-call total token counts in their Prometheus metrics stream; TraceProbe emits the *split*, which is the boundary that current metrics cannot express. With the split in the metrics layer, three policies become tractable that current per-call metrics cannot support: trace-length budgets keyed to grid carbon intensity, deferral of high-trace requests during high-CI windows, and reasoning-quota brownout that falls back to a non-reasoning model when the grid is dirtiest. TraceProbe is a 90-line patch to vLLM that emits the reasoning-to-answer token boundary in the per-request energy event. The patch adds two integers per request, parsed from the special-token positions in the generated sequence, and exposes them through vLLM's existing OpenAI-compatible metrics endpoint. The emitted record looks like:

```
{request_id, model, reasoning_tokens,
 answer_tokens, prefill_tokens,
 energy_j, region, started_at}
```

The implementation is one function call into the tokenizer per request, with negligible serving-side overhead measured at <0.2% throughput drop on a 1,024-prompt benchmark.

Integration path. TraceProbe does not change the inference kernel, the batching scheduler, or the request format. Existing serving deployments add the patch as a runtime monkey-patch and the metrics flow through their existing Prometheus or OpenTelemetry pipeline. The two new fields are picked up by carbon-accounting downstream of the serving layer (CarbonMin, Sprout, Caribou, or any internal cost-attribution dashboard) without changing the application-facing API. We tested the patch on three reasoning-model deployments and found no regressions in correctness or throughput at SLO.

From a measured split to an ahead-of-time estimate. TraceProbe emits the boundary at call completion, which attributes carbon after the fact. A scheduler that acts before a request runs needs an estimate of the split, and the per-call records TraceProbe accumulates supply that prior. The trace-length distribution per model and task class is queried at admission and refined online as records arrive, turning a measured split into a predicted one. The prediction is the scheduler's to make; TraceProbe provides the ground truth that makes it learnable, which is the piece a per-call total cannot give.

Why expose the trace instead of shaping it. Shaping the trace at generation time needs model-specific prompt engineering and degrades accuracy unpredictably when the trace is cut mid-thought. TraceProbe takes the opposite position. Leave generation unchanged, expose what it is doing, and let the scheduler choose the knob. The intervention is policy-free at the inference layer and policy-rich at the scheduling layer.

6 Discussion

Composing with existing work. TraceProbe is complementary to existing carbon-aware mechanisms: the boundary it emits is consumable by Sprout (to target reasoning and answer length differently), by region routers such as CarbonMin (to route high-trace requests differently), and by Caribou-style workflow schedulers (to budget trace-token quota per stage).

Limitations. We profile open models; the o-series and Gemini hide the trace and may behave differently. The truncation curve measures end-of-trace cuts, and mid-trace pruning could differ. We use average rather than marginal grid intensity [20], and embodied carbon and water are out of scope. The MATH-500 tradeoff may not generalize: we expect creative-writing and conversational tasks to show smaller traces and smaller gains.

Boundary conditions. The 78.4% trace fraction holds on the three open reasoning models we tested under default temperature 0.6 and top-p 0.95. Lower temperatures and tighter sampling shrink trace length [3]; we expect the fraction to fall to roughly 60% at temperature 0.1 on MATH-500, based on a 200-call calibration. The fraction also varies by task family: tool-use agents (GAIA) generate shorter traces than pure mathematical reasoning (MATH-500) because each tool invocation breaks the reasoning into smaller, separately-bounded segments. A 50-call calibration on 5-turn conversational prompts gives a 22% trace fraction, so deployments mixing chat and agentic traffic see a weighted reasoning-tax fraction scaled by the agentic share of tokens.

Open questions. Two threads matter most. First, structured pruning that removes redundant sub-thoughts but keeps the final commit might preserve more accuracy at the same carbon than the end-of-trace cut we measure. Second, the cross-model gap (DS-R1's trace ratio is 2× QwQ-32B's) suggests training-time interventions could re-shape the tax at the source rather than the consumer, a model-side question that needs the measurement target only TraceProbe produces.

Deployment cost. The mechanism holds as the workload evolves: as reasoning models grow toward a projected 80% of agent traffic by 2028, the absolute reasoning-tax carbon grows on top of traffic growth while grid decarbonization leaves the fraction invariant. Against a global LLM-inference base of ~30 TWh/year [12], the recoverable reasoning-tax carbon is on the order of 0.3 to 0.5 $MtCO_2$/year. The deployment cost is one library patch and two integer fields per request.

7 Conclusion

Reasoning models split each call into two phases, and the hidden one dominates. Across DeepSeek-R1, R1-Distill-Llama-70B, and QwQ-32B on three agent benchmarks, we measured the trace at 78.4% of operational carbon per task, a cost no published carbon-aware scheduler can see. We capped traces at the 30th percentile and removed 56 to 71% of that carbon for at most 2.3 points of accuracy. We built TraceProbe, a 90-line vLLM patch that adds the trace and answer token counts to each call's telemetry. Once the boundary is visible, a scheduler can act on it.

References

[1] Aradhye Agarwal, Ayan Sengupta, and Tanmoy Chakraborty. 2025. First Finish Search: Efficient Test-Time Scaling in Large Language Models. *arXiv preprint arXiv:2505.18149* (2025). doi:10.48550/arXiv.2505.18149

[2] Longju Bai, Zhemin Huang, Xingyao Wang, Jiao Sun, Rada Mihalcea, Erik Brynjolfsson, Alex Pentland, and Jiaxin Pei. 2026. How Do Coding Agents Spend Your Money? Analyzing and Predicting Token Consumptions in Agentic Coding Tasks. https://openreview.net/forum?id=1bUeVB3fov

[3] Wei-Lin Chen, Liqian Peng, Tian Tan, Chao Zhao, Blake JianHang Chen, Ziqian Lin, Alec Go, and Yu Meng. 2026. Think Deep, Not Just Long: Measuring LLM Reasoning Effort via Deep-Thinking Tokens. *arXiv preprint arXiv:2602.13517* (2026). doi:10.48550/arXiv.2602.13517

[4] Andrew A. Chien, Liuzixuan Lin, Hai Nguyen, Varsha Rao, Tristan Sharma, and Rajini Wijayawardana. 2023. Reducing the Carbon Impact of Generative AI Inference (today and in 2035). In *Proceedings of the 2nd Workshop on Sustainable Computer Systems (HotCarbon '23)*. Association for Computing Machinery, Boston, MA, USA, Article 11. doi:10.1145/3604930.3605705

[5] Mauricio Fadel Argerich, Jonathan Fürst, and Marta Patiño-Martínez. 2026. Watt Counts: Energy-Aware Benchmark for Sustainable LLM Inference on Heterogeneous GPU Architectures. *arXiv preprint arXiv:2604.09048* (2026). doi:10.48550/arXiv.2604.09048

[6] Zhenxiao Fu, Fan Chen, Shan Zhou, Haitong Li, and Lei Jiang. 2025. LLMCO2: Advancing Accurate Carbon Footprint Prediction for LLM Inferences. *ACM SIGEnergy Energy Informatics Review* 5, 2 (2025), 63–68. doi:10.1145/3757892.3757901

[7] Viktor Gsteiger, Pin Hong (Daniel) Long, Yiran (Jerry) Sun, Parshan Javanrood, and Mohammad Shahrad. 2024. Caribou: Fine-Grained Geospatial Shifting of Serverless Applications for Sustainability. In *Proceedings of the 30th ACM SIGOPS Symposium on Operating Systems Principles (SOSP '24)*. ACM, 403–420. doi:10.1145/3694715.3695954

[8] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[9] Walid A. Hanafy, Qianlin Liang, Noman Bashir, David Irwin, and Prashant Shenoy. 2023. CarbonScaler: Leveraging Cloud Workload Elasticity for Optimizing Carbon-Efficiency. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 7, 3 (2023), Article 57. doi:10.1145/3626788

[10] Walid A. Hanafy, Li Wu, David Irwin, and Prashant Shenoy. 2025. CarbonFlex: Enabling Carbon-aware Provisioning and Scheduling for Cloud Clusters. *arXiv preprint arXiv:2505.18357* (2025). doi:10.48550/arXiv.2505.18357

[11] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. In *Proceedings of the 35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track (NeurIPS '21)*. doi:10.48550/arXiv.2103.03874 The MATH-500 subset used in this paper is the 500-problem evaluation set introduced by Lightman et al., 2023.

[12] International Energy Agency. 2023. *Electricity Grids and Secure Energy Transitions*. Technical Report. IEA.

[13] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can Language Models Resolve Real-World GitHub Issues?. In *Proceedings of the 12th International Conference on Learning Representations (ICLR '24)*. https://openreview.net/forum?id=VTF8yNQM66

[14] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP '23)*. Association for Computing Machinery, 611–626. doi:10.1145/3600006.3613165

[15] Adam Lechowicz, Noman Bashir, David Irwin, Mohammad Hajiesmaili, and Prashant Shenoy. 2024. Precedence-Aware Carbon-Aware Scheduling on Heterogeneous Datacenters. In *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems (e-Energy '24)*. Association for Computing Machinery. doi:10.1145/3632775.3661930

[16] Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. 2024. Carbon in Motion: Characterizing Open-Sora Operational Carbon Footprint. In *Proceedings of the 3rd Workshop on Sustainable Computer Systems (HotCarbon '24)*. Association for Computing Machinery. doi:10.1145/3727200.3727224

[17] Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. 2024. Sprout: Green Generative AI with Carbon-Efficient LLM Inference. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP '24)*. Association for Computational Linguistics, 21799–21813. doi:10.18653/v1/2024.emnlp-main.1215

[18] Keyu Li, Junhao Shi, Yang Xiao, Mohan Jiang, Jie Sun, Yunze Wu, Dayuan Fu, Shijie Xia, Xiaojie Cai, Tianze Xu, Weiye Si, Wenjie Li, Dequan Wang, and Pengfei Liu. 2026. AgencyBench: Benchmarking the Frontiers of Autonomous Agents in 1M-Token Real-World Contexts. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (Eds.). Association for Computational Linguistics, San Diego, California, United States, 7422–7440. doi:10.18653/v1/2026.acl-long.337

[19] Yueying Li, Zhanqiu Hu, Esha Choukse, Rodrigo Fonseca, G. Edward Suh, and Udit Gupta. 2025. EcoServe: Designing Carbon-Aware AI Inference Systems. *arXiv preprint arXiv:2502.05043* (2025). doi:10.48550/arXiv.2502.05043

[20] Diptyaroop Maji, Prashant Shenoy, and Ramesh K. Sitaraman. 2022. CarbonCast: Multi-Day Forecasting of Grid Carbon Intensity. In *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '22)*. Association for Computing Machinery, 198–207. doi:10.1145/3563357.3564079

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

[22] Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. GAIA: A Benchmark for General AI Assistants. In *Proceedings of the 12th International Conference on Learning Representations (ICLR '24)*. https://openreview.net/forum?id=fibxvahvs3

[23] MLCommons. 2025. MLCommons Releases New MLPerf Inference v5.1 Benchmark Results. https://mlcommons.org/2025/09/mlperf-inference-v5-1-results/.

[24] OpenAI. 2024. *Learning to Reason with LLMs*. Technical Report. OpenAI.

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

[26] Qwen Team. 2024. QwQ: Reflect Deeply on the Boundaries of the Unknown. https://qwen.ai/blog?id=qwq-32b-preview.

[27] Prasoon Sinha, Dimitrios Liakopoulos, Ruihao Li, and Neeraja J. Yadwadkar. 2025. The Utilization Fallacy and the Real Drivers of Carbon-Efficient Inference Serving. In *Proceedings of the 4th Workshop on Sustainable Computer Systems (HotCarbon '25)*. Association for Computing Machinery. doi:10.1145/3735452.3735598

[28] Libo Wang. 2025. Dynamic Chain-of-Thought: Towards Adaptive Deep Reasoning. *arXiv preprint arXiv:2502.10428* (2025). doi:10.48550/arXiv.2502.10428

[29] Grant Wilkins, Srinivasan Keshav, and Richard Mortier. 2024. Offline Energy-Optimal LLM Serving: Workload-Based Energy Models for LLM Inference on Heterogeneous Systems. In *Proceedings of the 3rd Workshop on Sustainable Computer Systems (HotCarbon '24)*. Association for Computing Machinery. doi:10.1145/3727200.3727217

[30] Yuan-An Xiao, Pengfei Gao, Chao Peng, and Yingfei Xiong. 2026. Reducing Cost of LLM Agents with Trajectory Reduction. *Proceedings of the ACM on Software Engineering* 3, FSE (2026), Article FSE056. doi:10.1145/3797084