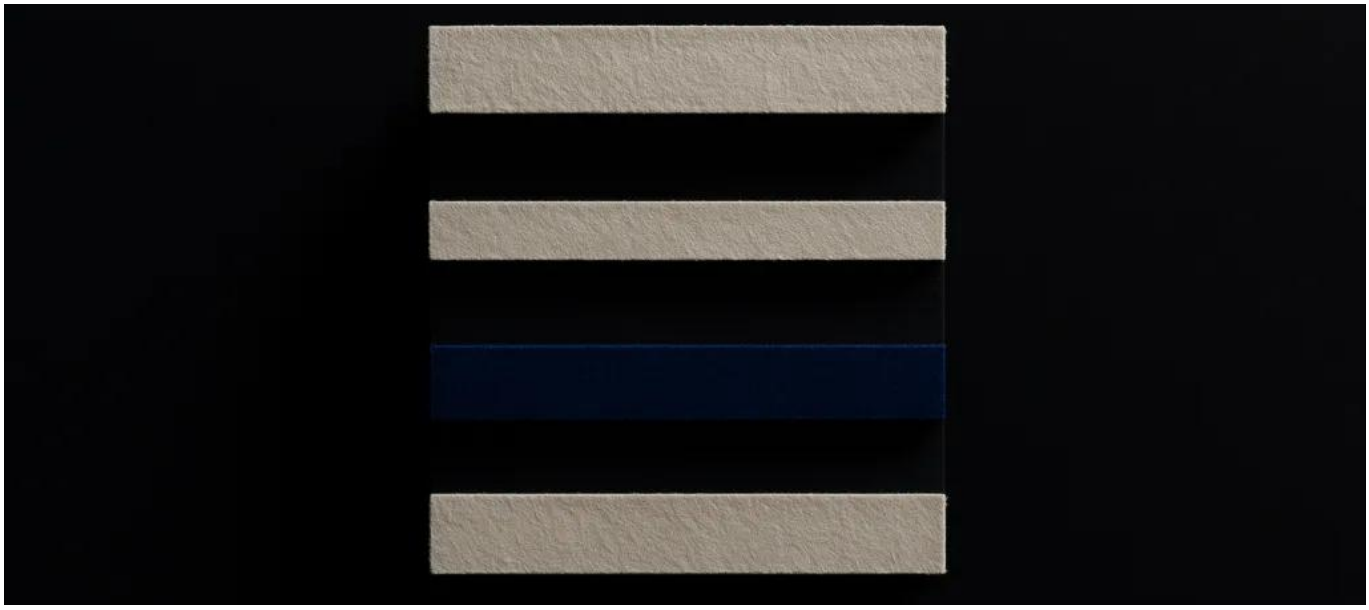

# New Research: Agentic Reasoning Burns Up to 136.5× More GPU Energy Per Query Than a Single-Turn LLM Call

348.41 Wh. That is what one Reflexion agent on Llama-3.1-Instruct 70B spent answering a single HotpotQA question, 136.5 times the energy of the same model doing one-shot inference.

A KAIST group led by Minsoo Rhu (Jiin Kim, Byeongjun Shin, Jinha Chung, Minsoo Rhu) measured what agentic test-time scaling costs at the infrastructure layer. Their paper, The Cost of Dynamic Reasoning (arXiv:2506.04301, v1 submitted 4 June 2025, record updated 2026-05-20), was accepted at HPCA-32, the 2026 IEEE International Symposium on High Performance Computer Architecture. Agent frameworks consume 62.1× to 136.5× the GPU energy per query of single-turn inference, inflate response time by up to 153.7×, and leave GPUs idle for as much as 54.5% of execution time.

## The paper instruments hardware, not benchmark scores or token bills

The paper calls itself “the first comprehensive system-level analysis of AI agents, quantifying their resource usage, latency behavior, energy consumption, and test-time scaling strategies.” That framing is worth taking seriously. Most of what the industry has published about agents is accuracy on a benchmark, or token cost from a vendor bill. This is hardware-level instrumentation: energy per query, latency distributions, and utilization of the GPU while the agent does something other than generate tokens.

The test subjects are two well-known agent designs. Reflexion runs the model, evaluates its own output, and retries with self-criticism in context. LATS wraps tree search around the model, expanding and evaluating multiple reasoning branches before committing. Both are test-time scaling in the plain sense: spend more compute at inference to get a better answer, without touching the weights. The baseline for comparison is a ShareGPT single-turn workload, meaning one prompt, one response, no tools.

The evaluation harness is AgentBench, released on GitHub by the VIA Research group. HotpotQA is the benchmark behind the energy numbers, which matters. HotpotQA is multi-hop question answering, so it is exactly the kind of task where an agent has a legitimate reason to loop, retrieve, and reconsider. These are not numbers from a pathological case chosen to embarrass agents.

## Two agent frameworks differ by 2.2× on the electricity bill

**348.41 Wh**
GPU energy per query, Reflexion
Llama-3.1-Instruct 70B on HotpotQA; 136.5× the ShareGPT single-turn baseline

**158.48 Wh**
GPU energy per query, LATS
same 70B setup; 62.1× the single-turn baseline

**54.5%**
of execution time with GPUs idle
agents blocked on tools, retrieval, or code execution

Take the energy figures first. The interesting part is the spread between two reasonable agent designs on identical hardware and an identical benchmark. Reflexion costs 2.2 times what LATS costs per query. Two frameworks that a team might pick between on a Tuesday

afternoon, based on an accuracy table, differ by more than a factor of two on operating cost. Nobody chooses between them on that basis today, because until this paper nobody had the number.

Then the idle time. Up to 54.5% of execution goes by with the GPU doing nothing while the agent waits on a search call, a retrieval index, or a sandboxed code run. This is the most actionable line in the paper, and it is structural. An agent loop is a sequence of blocking I/O operations interleaved with bursts of generation, and a GPU reserved for that loop spends half its life warm and unproductive. If you have provisioned dedicated accelerators per agent worker, you are paying full rate for roughly half a machine.

The latency inflation of up to 153.7× against static inference is the number I find least surprising and most dangerous. Unsurprising, because multiplying inference passes multiplies wall clock. Dangerous, because of what the paper says about the shape of the distribution rather than its mean: 95th-percentile latency keeps rising as reasoning steps are added even after mean latency and accuracy have saturated, according to the Moonlight review of the paper. So you tune your agent, watch the average stabilize, declare the configuration good, and ship a system whose tail gets worse with every extra step you allowed it to take. **The mean stops telling you anything long before the tail does.**

That tail behaviour is what turns an internal demo into a support ticket. An agent that averages twelve seconds and occasionally takes four minutes is a different product from one that averages twelve seconds and caps at thirty.

## Why 136.5× is a ceiling for one testbed, not a forecast for yours

The authors are careful here. Their own stated scope limits: the findings cover LLM-based agents with specific designs (Reflexion and LATS) running on specific hardware, so the absolute watt-hour values are testbed-dependent. The contribution is the relative scaling behaviour rather than a universal constant. The KAIST release frames the result as identifying a new energy workload class rather than setting a bound that applies to every agent deployment.

So: two agent frameworks, one model family, one benchmark for the energy figures, one

testbed. The paper does not measure production agents with caching, batching, smaller models for subtasks, or tool calls that hit warm services.

Several things in a real deployment push the ratio down. Batching across concurrent agent sessions reclaims some of that idle GPU. Prompt caching cuts the cost of re-sending the growing context that Reflexion-style self-criticism accumulates, which is precisely where those repeated passes get expensive (I went through the vendor-by-vendor economics of that in my prompt caching cost comparison). Routing easy steps to a smaller model removes 70B passes from the loop entirely. None of these appear in the paper, which is a scoping choice I would have made too: you cannot characterize a workload class and optimize it in the same study.

Other things push the ratio up. A 70B model is mid-sized by current standards. Deeper tool chains mean more blocking calls and more idle. And the paper measures GPU energy, not the retrieval infrastructure, the sandboxes, or the orchestration layer sitting around the model.

My take: the number that will change budgets is 54.5% idle, not 136.5×. Energy ratios can be argued away with batching and better models. Half your accelerator time spent blocked on I/O is an architectural property of the agent loop, and it will survive every model generation until schedulers treat an agent turn as an interruptible workload rather than a reserved session. I expect that to be the next serious systems paper out of this line of work, and I expect the first commercially useful fix to come from inference serving vendors rather than agent framework authors.

## What I would change in an agent design after reading this

Measure energy or GPU-seconds per resolved task, not per query. Per-query cost is close to meaningless for agents, because the number of queries per task is the variable the framework controls. If your dashboard shows tokens and latency but not accelerator time per completed task, you cannot see the trade the paper describes.

Set a step budget and treat it as a product decision. The paper's finding of rapidly diminishing returns means there is a point past which extra reasoning steps buy tail latency and electricity and nothing else. Find that point per task type. It will differ between a multi-

hop research question and a code fix, and the authors' own numbers show it differs between frameworks on the same task.

Instrument the tail before you ship, and alert on p95 rather than mean. Given that p95 keeps climbing after the mean flattens, a mean-based SLO will report green while the experience degrades.

Stop treating framework choice as an accuracy question alone. Reflexion at 348.41 Wh and LATS at 158.48 Wh on the same model and benchmark is a real difference in operating cost. Add energy or GPU-seconds to the selection criteria alongside the benchmark score. This is the same problem I described when scaffolding choices alone moved SWE-bench results by nine points in the harness piece: **the wrapper around the model is doing far more of the work, and the cost, than the model card suggests.**

And separate the loop from the accelerator. If an agent step blocks on retrieval for seconds, the GPU should be serving something else during that window. That is a serving and scheduling change rather than a prompt change, which is why it tends to be nobody's job on an agent team.

## Accuracy still scales with compute, and now the bill is measured

The abstract states both halves plainly: agents "improve accuracy with increased compute" but "suffer from rapidly diminishing returns, widening latency variance, and unsustainable infrastructure costs." The first clause is why every enterprise roadmap has an agent on it. The second clause was previously an intuition and is now measured at HPCA.

What I take from the paper is narrower than the headline ratio and more useful. Agentic reasoning is a distinct workload with a distinct hardware profile: bursty compute, long blocking gaps, and a heavy latency tail. Infrastructure built for single-turn chat serving will run it, expensively and with a bad tail. That is a solvable engineering problem, and the solutions are scheduling and budgeting rather than better prompts.

One honest uncertainty: I do not know how much of the 62.1× to 136.5× range survives a well-tuned production stack, and neither does anyone else yet, because the paper

deliberately did not test that. My guess is that the energy multiple compresses substantially and the idle fraction barely moves. If you are sizing GPU capacity for an agent rollout this quarter and the number you used came from single-turn throughput benchmarks, that is worth a conversation before the hardware order goes out, because the gap between those two models of the workload is where capacity plans go wrong.

WRITTEN BY

Artur Markus

Software engineer in Basel building AI infrastructure and agentic systems, with a background in pharmaceutical automation, GxP compliance, and validation. I help teams turn AI from a chatbot into a reliable operational layer.

Book a meeting →