ENERGY-OPTIMAL TEST-TIME SCALING: LATENCY VS. POWER TRADEOFFS IN SEQUENTIAL AND PARALLEL SCALING

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern LM systems frequently leverage the scaling of test-time computation, via both longer model outputs for reasoning chains and repeated parallel samples from a model, to improve task performance. While both sequential and parallel scaling increase task performance, the costs of this additional test-time computation are poorly quantified, with existing test-time scaling laws relying on efficiency proxies such as FLOPs rather than real-world costs such as latency and energy use. In this work, we develop an analytic task performance model for parallel and sequential test-time scaling and conduct an empirical efficiency benchmark of efficiency costs for a wide range of test-time computing configurations. We demonstrate that energy use scales with test-time compute distinctly from latency and FLOPs, with energy-optimal scaling utilizing longer reasoning chains rather than increased repeated samples. Accordingly, we identify energy-optimal test-time configurations for the Qwen3 family of models and find that energy-optimal selections reduce inference energy use by up to 60.1% and 55.8% as compared to latency-optimal and FLOPs-optimal selections on AIME26, respectively.

1 INTRODUCTION

The energy requirements of computing are growing, with multiple projections estimating that developments in AI will result in datacenters consuming more than 10% of US national energy demand by 2030 (Electric Power Research Institute (EPRI), 2024; Shehabi et al., 2024). In particular, inference workloads are expected to account for the vast majority of energy demand due to increased adoption by the general public and increases in per-query energy requirements. A primary driver of the increases in per-query energy use is a shift towards test-time computing, in which additional computation is allocated during inference to obtain higher task performance. State-of-the-art reasoning systems frequently generate thousands of tokens in response to each individual query, consuming more than 13x the energy of traditional queries (Oviedo et al.). Likewise, modern math and code systems rely on multiple samples drawn in parallel for increased diversity, often relying on upwards of 16x generations for a single query (Olmo et al., 2025; Du et al., 2026).

Existing test-time scaling laws (Brown et al.; Wu et al., 2024; Muennighoff et al., 2025; Yang et al., 2025; Snell et al., 2025) model task performance as a function of computational cost based on proxy metrics such as a generation budget in tokens or floating point operations (FLOPs). In recent work, Sadhukhan et al. (2025) demonstrated that FLOPs underestimates the costs of smaller language models by failing to account for the memory access costs of decoding. However, the scaling patterns of energy use remain distinct from latency. GPU power draw varies substantially across model and serving configurations, differing with model size, batch size, active parameters, and hardware platform (Patel et al., 2024; Chen et al., 2026). Other recent work has sought to evaluate the energy cost of language models with test-time computing approaches, but these works do not account for variability in serving configurations and often neglect to include task performance or accuracy (Oviedo et al.; Jin et al., 2025b; Fernandez et al., 2025).

If GPU power draw remained constant across serving configurations and test-time strategies, latency and energy would exhibit identical scaling behaviors, as energy would simply be latency scaled by GPU power draw. However, inference frequently occurs in a transient power regime, where power

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.

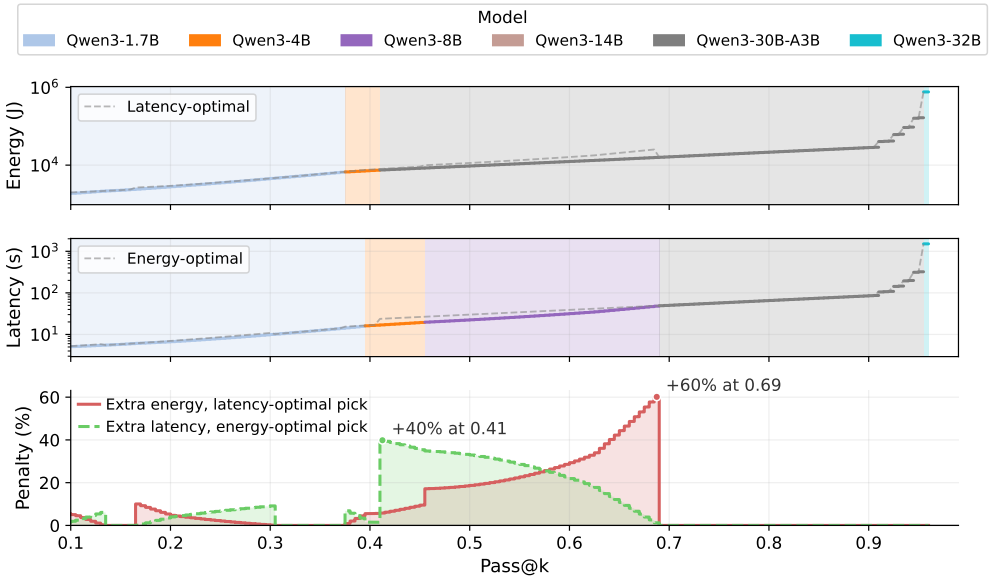


Figure 1: **Optimal model selection by latency optimality (top) and energy optimality (middle).** Energy-optimal model selection utilizes MoEs and smaller models more frequently, and yield *up to 55.8% and 60.1% reductions in energy use* relative to FLOPs-optimal and latency-optimal configurations (bottom).

draw varies by serving configuration. Such scenarios include when long reasoning traces or input contexts limit the maximum concurrent sequences due to memory requirements of growing KV caches, when models are being served with strict latency targets, low traffic, or when a model is sharded widely across several GPUs. To address these scenarios, we seek to answer: *How does the energy-optimal choice of reasoning length and number of samples differ from latency-optimal and FLOPs-optimal choices?*

We conduct an empirical study of energy-optimal test-time scaling. We benchmark task performance across a wide range of configurations with variable output sequence length and parallel repeated samples. We then develop a joint analytic model of task performance as a function of both sequential and parallel test-time compute. Our model predicts pass@k coverage with $R^2 > 0.95$ across the Qwen3 family of models on math and code reasoning tasks. Additionally, we measure the latency, steady-state power draw, and energy for each test-time configuration on Nvidia A100, H100-NVL, and H100-SXM GPUs. By combining the task performance predictive model with empirical efficiency measurements, we estimate the accuracy and cost for each model across iso-accuracy levels.

We find that energy-optimal test-time scaling frequently diverges from FLOPs- and latency-optimal scaling, and that the divergence follows directly from the power scaling of GPUs. Because increasing the number of parallel samples raises instantaneous power draw while increasing reasoning length does not, energy-optimal configurations favor longer reasoning chains at lower degrees of parallelism. Where latency-optimal scaling utilizes larger dense models that saturate the hardware, energy-optimal scaling more frequently benefits from smaller models and sparse mixture-of-experts models, whose low power draw at small batch sizes makes long generations comparatively cheap. On the Qwen3 family, energy-optimal configurations reduce inference energy by up to 60.1% relative to latency-optimal configurations and by 55.8% relative to FLOPs-optimal configurations on AIME 26. As inference moves toward larger servers, longer contexts, and multi-turn agentic workloads that constrain batch sizes, we expect energy-optimal scaling to become more distinct from the proxies currently used to guide test-time compute allocation.

Our contributions are:

1. A *joint sequential-parallel scaling model* for task accuracy ($R^2 > 0.95$ on Qwen3, six tasks) that yields iso-accuracy contours over test-time configurations.
2. An *accuracy-latency-energy benchmark* across reasoning budgets, sample counts, model architectures, and GPU platforms.
3. A *characterization of energy-optimal test-time scaling* and identification of model and serving configurations which reduce energy use via longer sequence lengths and reduced parallelism.

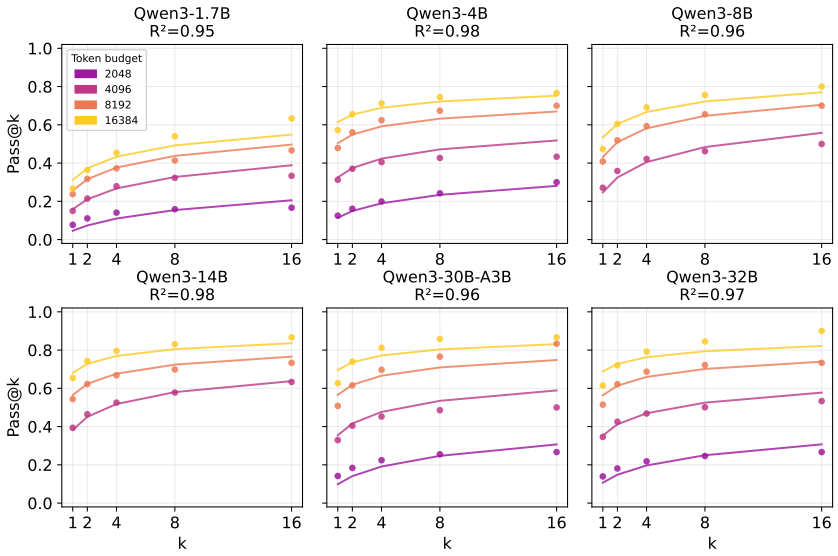


Figure 2: **Joint Task Performance Scaling Model for Sequential and Parallel Test-Time Scaling.** Across Qwen3 model sizes, scaling of both output reasoning tokens and number of repeated samples improves pass@k coverage. Analytically, the joint scaling model defined in Eqn 3 observes high $R^2 > 0.95$ correlation predicting pass@k coverage on AIME26.

2 PRELIMINARIES

Broadly, test-time computing methods can be categorized into approaches that scale inference computation sequentially or in parallel. A representative technique of sequential scaling is *chain-of-thought reasoning* (CoT; Wei et al. 2022), in which a model generates additional tokens prior to a final response to reason through problems, resulting in longer output sequence lengths (e.g. DeepSeek-R1, OpenAI o1, *inter alia*). Alternatively, test-time computation can be scaled in parallel with methods which utilize repeated sampling to generate multiple candidate answers (Brown et al.) (e.g. majority vote, best-of-*n* sampling, tree-of-thoughts, etc.).

To understand the joint scaling behavior of both task performance and energy use with test-time computing resources, we first consider the scaling behaviors of task performance and efficiency costs individually.

2.1 TASK PERFORMANCE SCALING WITH TEST-TIME COMPUTE

Previous investigations of chain-of-thought reasoning scaling identified that task performance observes a log-linear relationship with the number of reasoning tokens Muennighoff et al. (2025); Zhang et al. (2026). Concretely, given a reasoning budget of L tokens and modeled parameters $m, n \in \mathbb{R}$ the task performance Acc for a given model can be modeled as:

$$Acc(L) \approx m \log(L) + n \tag{1}$$

Similarly, parallel test-time scaling, in which the number of repeated samples is increased, can be applied to the same tasks. Coverage performance (i.e., pass@k) observes a power-law relationship with the number of samples (Brown et al.; Schaeffer et al., 2025). Concretely, given k parallel samples and modeled parameters $a, b \in \mathbb{R}$, coverage (i.e. pass@k) performance can be modeled as:

$$\text{Coverage}(k) \approx A_{max} \exp(ak^{-b}) \tag{2}$$

where A_{max} is the maximum attainable accuracy on the task by a model, a corresponds to the likelihood of a single example being correct, and b parametrizes the improvement per-repeated sample.

To develop a model of the task performance dependent on both sequential output tokens and parallel samples, we couple the log-linear scaling of accuracy with chain-of-thought length to the power law scaling of pass@k coverage. This yields the joint task performance scaling model in Equation 3 with details in Appendix B:

$$\text{Coverage}(k, L) \approx A_{max} \left(\tilde{m} \cdot \log(L) + \tilde{n}\right)^{k^{-b}} \tag{3}$$

2.2 Compute Efficiency Scaling with Test-Time Compute

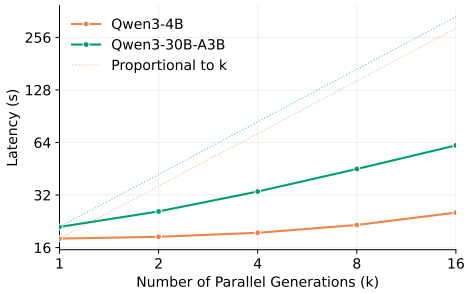
Similarly, as the number of tokens generated via repeated sampling and reasoning increases, the computation and real-world efficiency cost indicators observe comparable increases as well.

FLOPs Scaling. Analytically, floating-point operations (FLOPs) for decoding are counted by:

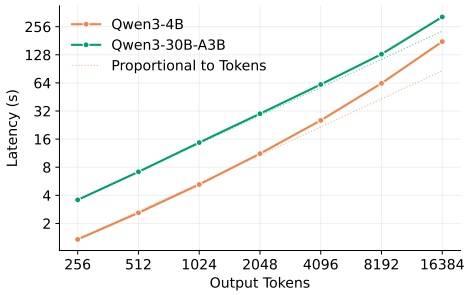
$$\text{FLOPs} = k\left(2PL_{\text{out}} + r(2L_{in} + L_{out})L_{out}D\right) \tag{4}$$

where P is the number of active model parameters, r is the ratio of grouped-query attention to vanilla full attention layers, and D is the key-value cache per token (Brown et al.). Accordingly, the number of FLOPs increases with both the number of parallel sequences, k, and with increased number of output tokens for reasoning L_{out}.

Latency Scaling. With increasing sequence length, per-sample latency grows linearly with the number of generated tokens due to the autoregressive nature of decoding and increasing size of KV-cache with sequence length. By contrast, increasing parallel samples increases latency sub-linearly as the increased batch sizes enable requests to better utilize parallel accelerator hardware.



(a) Latency scales sub-linearly with the number of parallel generations in the transient power regime.



(b) Latency scales linearly with output tokens.

Figure 3: Latency Scaling for Increasing Batch Size (left) and Sequence Length (right).

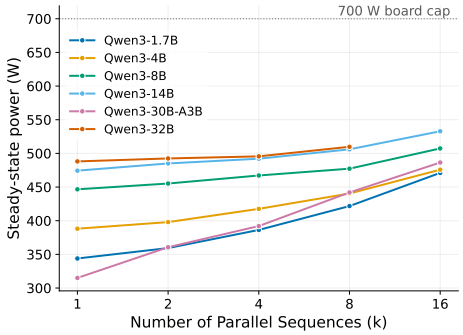
Power and Energy Scaling. Existing coarse-grained accounting of GPU energy use often relies on approximations on hardware-rated thermal design power (TDP). However, GPU power draw often operates below the TDP during inference. Autoregressive decoding is memory-bound which results in under-utilization of tensor core compute units (Patel et al., 2024). Increasing both model and batch size can improve hardware utilization, with increased tensor core activity and HBM bandwidth utilization (see Appx. F); this likewise increases instantaneous power draw (Chen et al., 2026).

For workloads which saturate a GPU, energy use is simply calculated by scaling latency by the hardware's TDP. However, for workloads in the *transient power regime* where power draw scales with the size of the workload, latency alone is insufficient for predicting energy use.

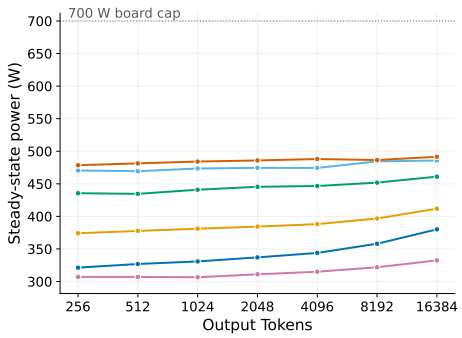
2.3 Experimental Setup

Based on the test-time scaling principles discussed in Section 2, we conduct an empirical study of the scaling behavior of sequential and parallel test-time scaling as applied to reasoning tasks. Specifically, we examine the scaling properties of the Qwen3 family of language models on math and code reasoning tasks as an application area commonly known to exhibit scaling in task performance via both sequential and parallel scaling. We examine the scaling performance on a set of reasoning-intensive tasks: AIME 26, Math500, and LiveCodeBench (Hendrycks et al.; Jain et al., 2025). We provide 5-shot demonstrations for each example as specified in Gu et al. (2025a). Additional results for tasks which benefit less from CoT reasoning sequential scaling are available in Appendix E.

Evaluating Accuracy. To examine the parallel scaling properties, we evaluate the pass@k coverage when sampling between 1 to 16 samples per example. Likewise, to investigate the sequential scaling performance, we enforce maximum thinking budgets which induce answer generation after exceeding a specified thinking token budget ranging from 256 up to 16,384 thinking tokens; with



(a) Steady-state power draw increases with **number of parallel generations**.



(b) Steady state power is largely constant across **output sequence lengths**.

Figure 4: With few parallel generations, the steady-state power draw for models with few active parameters can be less than 50% of rated TDP.

the exact ranges determined by task. For sequences which exceed the maximum thinking budget, we inject a termination sequence via budget forcing (Muennighoff et al., 2025).

As a measure of the allowed test-time computation, we report both the maximum allowed thinking budget and the mean generated number of tokens. The maximum number of tokens corresponds to the hard-cap thinking budget, whereas the mean generated tokens corresponds to the actual number of tokens generated by the model.

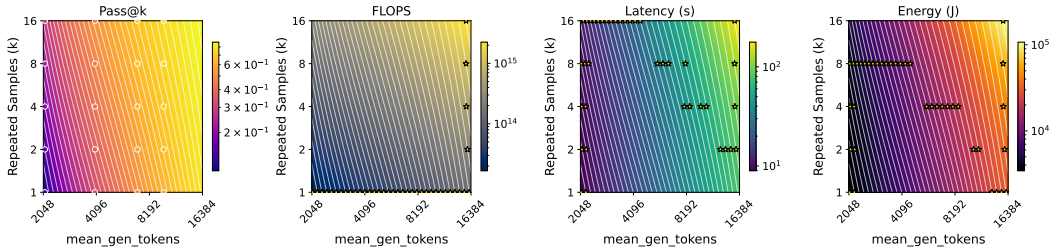
Across all tasks, we find that the joint scaling model specified in Equation 3 is able to model the joint scaling of parallel and sequentially scaling test-time computation, with $R^2 > 0.95$ across models and tasks. Additionally, we find that the joint-scaling model for test-time compute and coverage can be applied by scaling either maximum token budget or mean generated tokens. As observed in Figure 2, both parallel and sequential test-time compute yield improvements to AIME26 task performance; with results for TTC scaling on the remaining tasks in Appendix E. Notably, the degree of marginal improvement per additional sample or thinking token differs across each model and task.

Evaluating Efficiency. To examine the latency, power, and energy requirements for a given test-time scaling configuration, we evaluate micro-benchmarks on A100, H100-NVL, and H100-SXM GPUs using vLLM with maximum allowable concurrent sequences equal to the number of repeated samples for various pass@k values across variable output token reasoning budget and from 1 to 16 repeated samples. To isolate the resources used during model serving from inference engine warmup, we report the steady state power of each test-time computing configuration as the mean power draw over the range where GPU utilization exceeds 80% for greater than 2 seconds, with the full time-series shown in Appendix F.

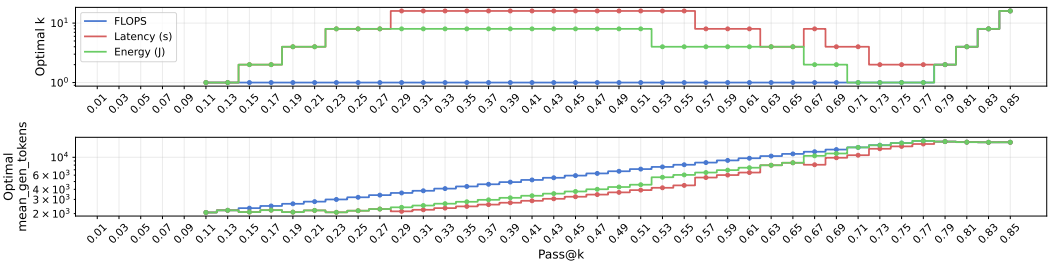
As shown in Figure 4a and consistent with Chen et al. (2026), increasing the number of concurrent requests and parallel samples *increases the power utilization*. Although latency increases with the number of tokens, the power draw is consistent during generation, regardless of the sequence length; shown in Figure 4b.

3 Joint Energy and Accuracy-Optimal Test-Time Scaling

To model the accuracy and efficiency for arbitrary parallel and sequential test-time configurations, we combine the joint performance scaling model with interpolations of empirical efficiency measurements collected in Section 2.



(a) **Accuracy and Efficiency Cost Scaling for Variable TTC Configurations**. FLOPs-, Latency-, and Energy-optimal configurations select differing test-time configurations along the task's iso-accuracy curve. Stars correspond to the cost-optimal configuration for each iso-accuracy level.



(b) **Optimal Test-Time Configurations for Variable Target Accuracy**. Longer reasoning lengths and repeated samples are required to attain higher accuracies, with energy-optimal approaches selecting for lower parallelism and longer outputs.

Figure 5: **Accuracy and Efficiency Tradeoffs for Qwen3 4B on H100-SXM.**

3.1 Iso-Accuracy Scaling

Scaling across both parallel and sequential outputs yields increases in accuracy and efficiency costs. In particular, multiple different test-time configurations can achieve similar accuracies as evident by the iso-accuracy contours in Figure 5a.

As shown in Figure 5b, depending on the target efficiency objective, the optimal test-time computing configuration changes at each iso-accuracy target level. *Energy-optimal* test-time configurations rely on longer sequence lengths, rather than a larger number of repeated samples which would instead increase the instantaneous power draw (as seen in Figure 4a). In contrast, latency-optimal serving configurations utilize high parallelism to offset longer generations which are slow and memory-bound by the decoding process.

Whereas previous work such as Sadhukhan et al. (2025) assumes accelerators are saturated at large batch sizes, the increased model size and input context lengths required by modern test-time computing strategies (or strict latency settings) limit the range of supported batch sizes and query parallelism. As a result, many practical serving configurations operate well below the hardware power cap, where steady-state power draw rises with the number of concurrent sequences (Figure 4a) rather than remaining fixed at TDP.

Previous work has shown that pure FLOPs-based estimates of compute utilization overestimate the efficiency of model decoding due to the memory boundedness nature of decoding. While latency based measurements find that longer sequence decoding with smaller models is less efficient than larger models, latency-optimal configurations do not account for the variability of power draw in the transient power regime at low batch sizes.

Takeaway

1. **FLOPs and token-optimal scaling underestimate the latency and energy costs of generation.** These analytic scaling models do not account for the improved throughput of parallel scaling or the increasing memory access costs associated with the KV caches required for long sequences.

While reduced parallelism decreases instantaneous power draw, the increased latency offsets gains in energy reduction for FLOPs-optimal configurations.

2. **Energy-optimal test-time configurations favor longer sequence lengths and reduced parallelism.** Increasing sequence length has minimal impact on instantaneous power draw and can be used to improve test-time performance with less additional energy costs than parallel scaling.
3. **Energy-optimal scaling diverges from latency-optimality mainly in transient power regimes.** When instantaneous power draw is sensitive to serving setting and test-time configuration, energy does not scale with latency by a constant factor, and energy-optimal scaling can diverge from latency-optimal scaling.

3.2 Variations Across Model Architectures

Model Size. As model size increases, hardware utilization increases so that steady-state power draw approaches the TDP power cap even at single-batch and sample inference; seen in Figure 4a. When power draw is constant, energy use observes identical scaling behavior to latency as power draw does not differ across test-time configurations. In these cases, energy-optimal scaling converges to similar optimal configurations as latency-optimal scaling, as power draw no longer changes with number of samples. By contrast, models with fewer active parameters draw substantially less power at low batch sizes and observe stronger power scaling behavior as the number of parallel samples increases.

Dense vs Mixture-of-Experts Models In contrast to dense models, the mixture-of-expert Qwen3 30B-A3B observes the lowest steady-state power draw at single example inference, with steady-state power draw at 315 W of 700W. The mixture-of-expert's power draw scales with number of parallel generations, approaching the power draw of the dense model with similar number of active parameters (i.e. Qwen3 4B) at larger number of samples (as seen in Figure 4a). As a result, the energy-optimal configurations for MoE frequently utilize single sample inference to avoid the power scaling of increased batched generations.

In Appendix D, we consider the test-time scaling behavior of the Qwen3.5 family of hybrid state-space models. Qwen3.5 uses a gated delta network as a drop-in replacement for attention, reducing the cost of long generations by removing large KV-caches. Accordingly, we find that by reducing the cost of sequential scaling, there are a greater number of viable distinct energy-optimal test-time configurations.

Optimal Model Selection In addition to affecting the optimal test-time configuration for an individual model, energy-optimal scaling leads to distinct model selections when selecting models across architectures. Energy-optimal scaling highlights the benefits of smaller models and mixture-of-experts models as energy-efficient alternatives to larger dense models, as seen in Figure 1. Energy-optimal scaling relies on sparse mixture-of-experts and smaller models more frequently than latency-optimal inference. Concretely, selecting energy-optimal test-time configurations yields up to 60.1% and 55.8% savings in energy as compared to latency-optimal and FLOPs-optimal configurations on AIME26, respectively.

Takeaway

Energy-optimal test-time scaling favors models with lower power intensity. Smaller models and mixture-of-expert models display more scaling behavior between power draw and number of parallel samples; as a result, long generations with these models are notably power- and energy-efficient.

3.3 Variation Across Tasks

Energy-optimal scaling is reliant on task performance positively scaling with number of output reasoning tokens. However, the extent to which this scaling behavior occurs is task dependent.

Task	$\bar{m}$	$\bar{n}$	$\bar{Acc_{Max}}$	$\bar{b}$
AIME26	0.3266 ± 0.0820	-2.3919 ± 0.6033	1.0000 ± 0.0000	0.2495 ± 0.0415
Math 500	0.2200 ± 0.0491	-0.7882 ± 0.4108	0.7054 ± 0.0225	0.2814 ± 0.0507
GPQA Diamond	0.4061 ± 0.4827	-1.5880 ± 1.8625	0.6506 ± 0.3875	0.0466 ± 0.0448
LiveCodeBench	0.0985 ± 0.0254	-0.0254 ± 0.2048	0.6050 ± 0.0143	0.4403 ± 0.1417
HumanEval	0.0417 ± 0.0509	0.5243 ± 0.4411	0.9496 ± 0.0118	0.8524 ± 0.2112
MBPP	0.0264 ± 0.0355	0.6673 ± 0.3147	0.9648 ± 0.0234	0.7523 ± 0.3433

Table 1: **Modeled Coefficients Per-Task.** Coefficients are averaged across Qwen3 models sizes.

Effectiveness of Reasoning. For there to exist joint accuracy-efficiency pareto tradeoffs, task performance must scale with both sequential and parallel test-time compute. The AIME26 and Math500 tasks exhibit the strongest degree of pass@k scaling with both sequential and parallel test-time compute, as evident through relatively larger values for both sequential (m) and parallel (b) scaling coefficients learned via the joint scaling model.

By contrast, the coding tasks (i.e., LiveCodebench, HumanEval, and MBPP) exhibit limited performance gain with additional output tokens and instead observe substantially stronger scaling with repeated sampling. Additionally, GPQA Diamond's performance scales predominantly with sequence length with minimal performance improvement through repeated sampling [1]. Additional results on the scaling behavior of different tasks are provided in Appendix E.

Energy-Optimal Test-Time Scaling is Task Dependent. As accuracy for MBPP and HumanEval saturates within a few hundred reasoning tokens, additional sequential compute provides little benefit ($\tilde{m} < 0.05$) and the iso-accuracy contours run nearly parallel to the token-budget axis. With no sequential lever to exploit, both energy- and latency-optimal selection rely on repeated sampling, and the two objectives coincide regardless of the power savings that longer generations would otherwise offer. By contrast, on reasoning-intensive tasks such as AIME26 and Math500, accuracy scales with both axes, the iso-accuracy contours span a wide range of (k, L) configurations, and energy-optimal selection consistently favors longer reasoning at lower parallelism than the latency-optimal choice. GPQA Diamond represents the opposite extreme: as a multiple-choice task evaluated with majority@k, it gains almost nothing from repeated sampling ($|b| \approx 0.05$), so all three objectives select k=1 and differ only in reasoning length.

TAKEAWAY

Energy-aware allocation of test-time compute must be task-aware. The same model on the same GPU may warrant high parallelism on one task and a single long generation on another, depending on the relative efficacy of each decoding strategy for each task.

3.4 EFFECTS OF HARDWARE PLATFORMS

Hardware Speed and Power Capping. For GPUs with reduced power caps, lower intensity workloads (i.e. smaller models and lesser degrees of concurrent generation) are more easily able to draw the full power available to the device and further increases in parallelism do not raise power draw. As a result, the transient power regime discussed in Section 2.2 is reduced and the energy-optimal configurations are more aligned with the latency-optimal choices. As compared to the H100-SXM, the A100, with a maximum power cap of 300W, and the H100-NVL, with a maximum power cap of 400W, observe only 17% and 53% in excess energy when utilizing FLOPs-optimal test time configurations.

TAKEAWAY

Energy-optimal scaling is dependent on the hardware platform. As each accelerator platforms exhibit different transient power regimes, the effectiveness of lower parallelism at reducing power costs is hardware-dependent.

[1] As the GPQA Diamond is multiple choice, the performance metric is reported with majority@k which observes distinct scaling behavior from pass@k.

4 Related Work

Scaling Patterns in Test-Time Compute Allocation of additional computational resources improves model performance across settings, including mathematical, programmatic, and long-context reasoning. Recent advances in long-context modeling allow state-of-the-art large language models to support increasingly larger sequences. Increasing the length of reasoning traces during generation has been shown to empirically improve the model performance (Muennighoff et al., 2025; Zhu et al., 2025; Jin et al., 2024; Chen et al., 2025). Likewise, parallel sampling has been shown to improve the diversity and coverage of model generations, commonly yielding improvement across math and code tasks Brown et al.; Lightman et al. (2023). Additionally, longer sequences enabled the extension of few-shot in-context learning to the many-shot setting in which models are presented with thousands of example labels and demonstration reasoning traces; with performance improvements across question answering and reasoning tasks (Yue et al., 2025; Li et al., 2024c; Wan et al., 2025; Agarwal et al., 2024; Bertsch et al., 2025; Gu et al., 2025b; Luo et al., 2024).

To increase the efficiency of long reasoning chains, recent work has explored training techniques based on reinforcement learning and supervised finetuning to align language models to output shorter reasoning sequences while maintaining correctness or dynamically adjust output length based on example difficulty (Arora & Zanette, 2025; Aggarwal & Welleck, 2025; Hassid et al., 2025). Additionally, dynamic sequence compression and in-context prompting can be used to condition models to generate shorter sequences (Zhang et al., 2025; Li et al., 2024b). While such work has demonstrated that it is feasible to induce shorter reasoning chains under fixed accuracy constraints, they often rely on token budgets as an indicator of compute efficiency and fail to incorporate real-world measures of latency or energy.

Together, both sequential and parallel scaling observe positive task performance with increasing total tokens, the joint scaling patterns of these two test-time-computing approaches are not well defined across tasks or model architectures. Inference-time scaling laws which relate sequence length, estimated FLOPs, and hardware utilization to performance on reasoning tasks have been developed to characterize the efficiency and resource requirements of TTC workloads (Wu et al., 2024; Yang et al., 2025; Sadhukhan et al., 2025). However, these scaling trends are not grounded in the underlying compute and energy efficiency of the computing hardware; often relying on proxy metrics for compute budget such as token counts and FLOPs.

Power Considerations in LM Inference The emerging prevalence of LLM inference workloads has led to the development of methods which improve serving efficiency as measured with end-to-end system latency and energy efficiency (Li et al., 2024a). As AI energy demands grow, a growing body of work studies the energy efficiency and environmental impact of inference. Evaluations of language model inference have relied on direct empirical measurement (Samsi et al., 2023; Patel et al., 2024; Luccioni et al., 2024; Fernandez et al., 2025), analytic models that estimate energy from assumed utilization and hardware characteristics (De Vries, 2023; Mistral AI, 2025; Rincé & Banse, 2025; Epoch AI, 2025), or the development of standardized benchmarks for energy use (e.g. ML.ENERGY, MLPerf Power, and AI Energy Score; Chung et al., 2026; Tschand et al., 2025; Hugging Face and Salesforce, 2025). Recent studies extend these measurements to test-time compute and reasoning, where additional generated tokens can increase per-query energy by up to 13$\times$ relative to standard queries (Oviedo et al.; Jin et al., 2025a; Elsworth et al., 2025; Chung et al., 2026). However, prior evaluations have focused on the end-to-end energy cost of frontier models, assuming the inference configuration is fixed, without consideration for how accuracy changes with both latency or efficiency. By contrast, we develop a unified approach to evaluate the scaling behaviors of both accuracy and latency.

5 Conclusion

In this work, we examine the accuracy and energy scaling of parallel and sequential test-time computing approaches. We develop a joint model for sequential and parallel test-time scaling and demonstrate that energy exhibits distinct scaling behavior from existing efficiency cost-indicators of FLOPs and latency due to power scaling properties of GPUs. In contrast to latency-optimal inference, we find that energy-optimal test-time configurations often rely on longer sequence lengths and reduced parallelism. Additionally, we find that energy-optimal test time computing benefits from smaller models and sparse mixture of experts models which utilize less power than dense alternatives. We believe that our work will enable researchers and developers to understand the energy requirements of their deployments.

Statement on AI Use

In this work, we used generative AI tools to assist in data visualizations and figures, to provide feedback on human-authored writing, to assist in verification of mathematical claims, and for code assistance. Generative AI was used for feedback on human-proposed frameworks and experimental methodology. However, all decisions were reviewed and made by human authors. We have not used generative AI tools for literature review, data generation, or the remainder of the required subtasks. AI-assisted code, text, and figures were reviewed by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Rishabh Agarwal, Avi Singh, Lei M. Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, John D. Co-Reyes, Eric Chu, Feryal Behbahani, Aleksandra Faust, and Hugo Larochelle. Many-shot in-context learning, 2024. URL https://arxiv.org/abs/2404.11018.

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.04697.

Daman Arora and Andrea Zanette. Training language models to reason efficiently, 2025. URL https://arxiv.org/abs/2502.04463.

Amanda Bertsch, Maor Ivgi, Emily Xiao, Uri Alon, Jonathan Berant, Matthew R. Gormley, and Graham Neubig. In-context learning with long-context models: An in-depth exploration. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 12119–12149, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.605. URL https://aclanthology.org/2025.naacl-long.605/.

Bradley Brown, Jordan Juravsky, Ryan Saul Ehrlich, Ronald Clark, Quoc V Le, Christopher Re, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling.

Huamin Chen, Xunzhuo Liu, Yuhan Liu, Junchen Jiang, Bowei He, and Xue Liu. The 1/w law: An analytical study of context-length routing topology and gpu generation gains for llm inference energy efficiency. *arXiv preprint arXiv:2603.17280*, 2026.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models, 2025. URL https://arxiv.org/abs/2503.09567.

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

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

Wei Du, Shubham Toshniwal, Branislav Kisacanin, Sadegh Mahdavi, Ivan Moshkov, George Armstrong, Stephen Ge, Edgar Minasyan, Feng Chen, and Igor Gitman. Nemotron-math: Efficient long-context distillation of mathematical reasoning from multi-mode supervision. In *3rd AI for Math Workshop: Toward Self-Evolving Scientific Agents*, 2026. URL https://openreview.net/forum?id=FK9LEhmjof.

Electric Power Research Institute (EPRI). Powering intelligence: Analyzing artificial intelligence and data center energy consumption. Technical Report 3002028905, EPRI, Palo Alto, CA, 2024. URL https://www.epri.com/research/products/000000003002028905.

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

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

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

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. Olmes: A standard for language model evaluations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5005–5033, 2025a.

Zhengyao Gu, Henry Peng Zou, Yankai Chen, Aiwei Liu, Weizhi Zhang, and Philip S Yu. Scaling laws for many-shot in-context learning with self-generated annotations. *arXiv preprint arXiv:2503.03062*, 2025b.

Michael Hassid, Gabriel Synnaeve, Yossi Adi, and Roy Schwartz. Don't overthink it. preferring shorter thinking chains for improved llm reasoning, 2025. URL https://arxiv.org/abs/2505.17813.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

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

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. The impact of reasoning step length on large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 1830–1842, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.108. URL https://aclanthology.org/2024.findings-acl.108/.

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

Yunho Jin, Gu-Yeon Wei, and David Brooks. The energy cost of reasoning: Analyzing energy usage in llms with test-time compute, 2025b. URL https://arxiv.org/abs/2505.14733.

Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. Llm inference serving: Survey of recent advances and opportunities, 2024a. URL https://arxiv.org/abs/2407.12391.

Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. Sprout: Green generative ai with carbon-efficient llm inference. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 21799–21813, 2024b.

Rui Li, Guoyin Wang, and Jiwei Li. Are human-generated demonstrations necessary for in-context learning?, 2024c. URL https://arxiv.org/abs/2309.14681.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Sasha Luccioni, Yacine Jernite, and Emma Strubell. Power hungry processing: Watts driving the cost of ai deployment? In *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24, pp. 85–99. Association for Computing Machinery, 2024. ISBN 9798400704505. doi: 10.1145/3630106.3658542. URL https://doi.org/10.1145/3630106.3658542.

Man Luo, Xin Xu, Yue Liu, Panupong Pasupat, and Mehran Kazemi. In-context learning with retrieved demonstrations for language models: A survey. *arXiv preprint arXiv:2401.11624*, 2024.

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

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

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

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

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

Ranajoy Sadhukhan, Zhuoming Chen, Haizhong Zheng, Yang Zhou, Emma Strubell, and Beidi Chen. Kinetics: Rethinking test-time scaling laws, 2025. URL https://arxiv.org/abs/2506.05333.

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

Rylan Schaeffer, Joshua Kazdan, John Hughes, Jordan Juravsky, Sara Price, Aengus Lynch, Erik Jones, Robert Kirk, Azalia Mirhoseini, and Sanmi Koyejo. How do large language monkeys get their power (laws)? In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=QqVZ28qems.

Arman Shehabi, Sarah Josephine Smith, Alex Hubbard, Alexander Newkirk, Nuoa Lei, Md AbuBakar Siddik, Billie Holecek, Jonathan G. Koomey, Eric R. Masanet, and Dale A. Sartor. 2024 united states data center energy usage report. Technical report, Lawrence Berkeley National Laboratory, 12 2024. URL https://eta.lbl.gov/publications/2024-lbnl-data-center-energy-usage-report.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025.

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

Xingchen Wan, Han Zhou, Ruoxi Sun, Hootan Nakhost, Ke Jiang, and Sercan Ö. Arık. From few to many: Self-improving many-shot reasoners through iterative optimization and generation, 2025. URL https://arxiv.org/abs/2502.00330.

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

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models, 2024. URL https://arxiv.org/abs/2408.00724.

Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. Towards thinking-optimal scaling of test-time compute for llm reasoning, 2025. URL https://arxiv.org/abs/2502.18080.

Zhenrui Yue, Honglei Zhuang, Aijun Bai, Kai Hui, Rolf Jagerman, Hansi Zeng, Zhen Qin, Dong Wang, Xuanhui Wang, and Michael Bendersky. Inference scaling for long-context retrieval augmented generation, 2025. URL https://arxiv.org/abs/2410.04343.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. *CoRR*, abs/2502.15589, 2025. doi: 10.48550/ARXIV.2502.15589. URL https://doi.org/10.48550/arXiv.2502.15589.

Qianfan Zhang, Tianyu Guo, Xuandi Ren, Jiale Chen, Ming Ding, Ran Xin, and Xia Xiao. Scaling reasoning tokens via rl and parallel thinking: Evidence from competitive programming. *arXiv preprint arXiv:2604.01302*, 2026.

Dawei Zhu, Xiyu Wei, Guangxiang Zhao, Wenhao Wu, Haosheng Zou, Junfeng Ran, Xun Wang, Lin Sun, Xiangzheng Zhang, and Sujian Li. Chain-of-thought matters: Improving long-context language models with reasoning path supervision. *arXiv preprint arXiv:2502.20790*, 2025.

A Implementation Details

A.1 Hardware Configurations

Performance evaluations are conducted on Nvidia H100, H100-NVL, and A100 datacenter GPUs. Accuracy evaluations are conducted using Nvidia L40S GPUs.

GPU	Power	bfloat16 FLOPs	HBM Speed (TB/s)	DRAM Available
H100 SXM	700W	1,979 TFLOPs	3.35 TB/s	80 GB
H100 NVL	400W	1,671 TFLOPs	3.9 TB/s	94 GB
A100	300W	312 TFLOPs	2.0 TB/s	80 GB

Table 2: Comparison of H100 SXM, H100 NVL, and A100 GPU Specifications

A.2 Evaluation Frameworks and Libraries

Pass@k Coverage Evaluations Accuracy evaluations are completed with Eleuther AI's Language Model Evaluation Harness. For each evaluation query, models are provided 5-shot demonstrations sampled 16 times with differing random seeds. Pass@k values for reduced values of k are then reconstructed with an unbiased estimator based on the samples.

Scaling Reasoning Budget with Budget Forcing Output reasoning tokens are scaled between 256 and 16,384 output tokens using budget forcing Muennighoff et al. (2025). For sequences which exceed the maximum allowed token budget, we inject the early-stopping termination sequence as specified in the Qwen3 documentation[2]:

```
"Considering the limited time by the user, I have to give the
solution based on the thinking directly now.</think>".
```

Efficiency Microbenchmarking. Microbenchmarking of the efficiency costs of test-time configurations is conducted using vLLM benchmark, with power and hardware utilization (i.e., Tensor Core Activity and Memory Access Frequency) measured using Nvidia Data Center GPU Manager. For each configuration, requests are prefixed with 768 tokens to correspond to static k-shot demonstrations and 256 random tokens corresponding to the example query, and the maximum number of allowed concurrent sequences is set to the target value of k. The `vllm` engine is then queued with either 4 or 8 waves of k requests, with output length fixed to the token budget. Per-request latency is then approximated by dividing the run latency by the number of waves.

As noted in Section 2.3, we limit the maximum number of parallel sequences to 16, as we observe concurrency throttling for larger number of parallel generations, in which growing KV cache sizes limit the number of supported sequences per GPU.

B Derivation of Joint Sequential-Parallel Scaling Model

Given both the sequential log-linear relationship in sequence length L and the parallel coverage relationship in repeated samples k,

$$Acc(L) = m \log L + n \qquad Cov(k) = A_{Max} e^{a k^{-b}}$$

Both are equivalent at $k = 1$:

$$Acc(L) = A_{Max} e^{a} = Cov(1) \rightarrow a = \log\left(\frac{m \log(L) + n}{A_{max}}\right)$$

Coverage can then be rewritten in terms of both k and L, with $\tilde{m}$ and $\tilde{n}$ renormalized by A_{Max}:

$$Cov(k, L) = A_{Max}\left(\tilde{m} \log(L) + \tilde{n}\right)^{k^{-b}}$$

[2] https://qwen.readthedocs.io/en/latest/getting_started/quickstart.html

C Additional Results on A100 and H100-NVL

Compared to the Nvidia H100-SXM, the A100 and H100-NVL exhibit a reduced transient power regime and saturate power draw for a majority of models even at single batch inference.

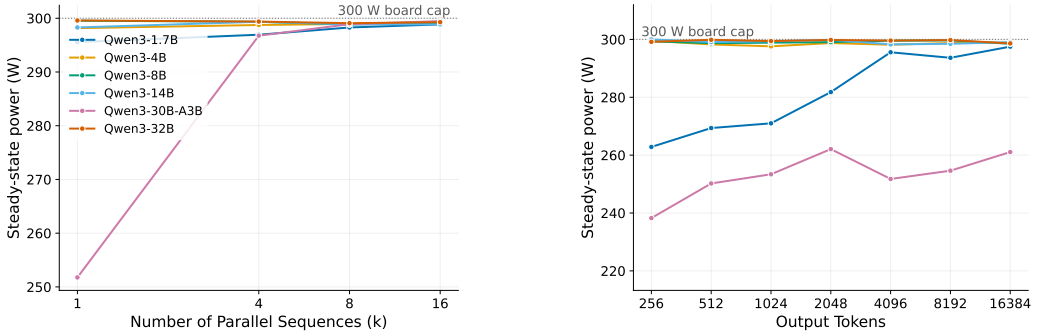


Figure 6: **A100 Steady State Power Draw for Increasing Batch Size and Sequence Length**.

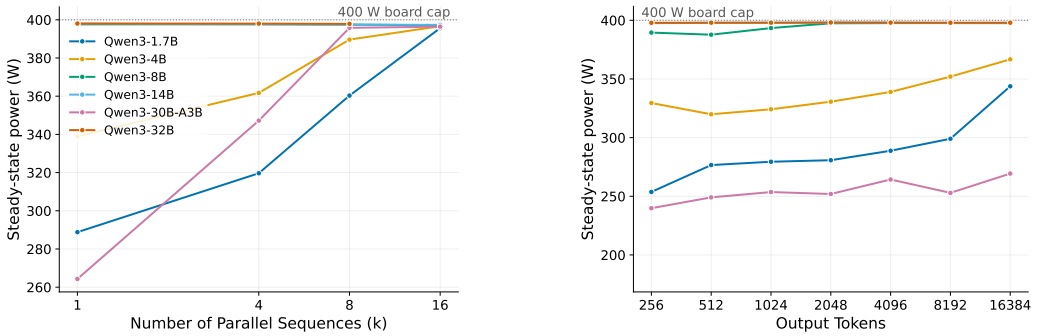


Figure 7: **H100-NVL Steady State Power Draw for Increasing Batch Size and Sequence Length**.

D Additional Results with Hybrid State-Space Models

We consider the Qwen 3.5 family of hybrid state-space models. As compared to the Qwen3 models which use grouped-query attention across all layers, Qwen3.5 uses a gated delta network as a drop-in replacement for attention which reduces the latency of long-context generation by removing reliance on large KV-caches. Accordingly, we find that reducing the cost of sequential scaling yields a greater number of viable, distinct energy-optimal test-time configurations.

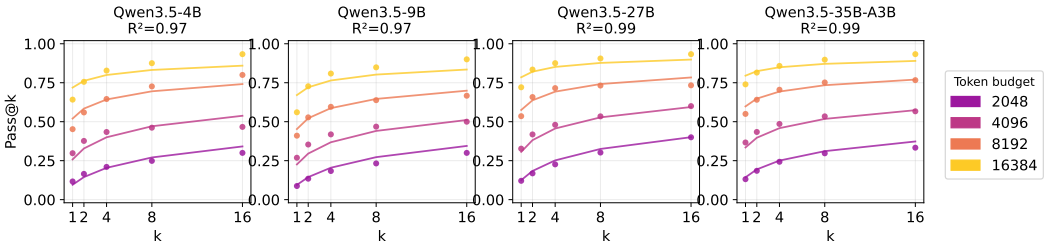
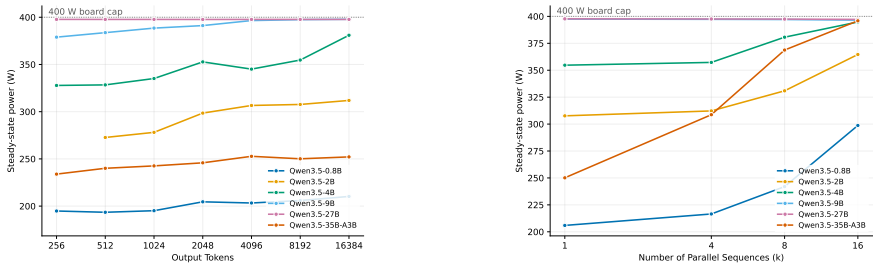
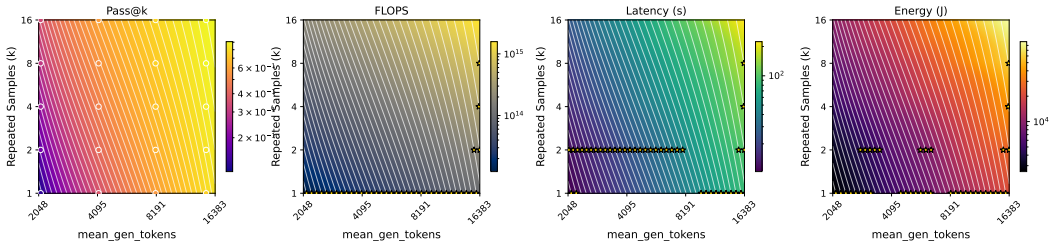


Figure 8: Joint Sequential and Parallel Scaling for Qwen 3.5 on AIME26.

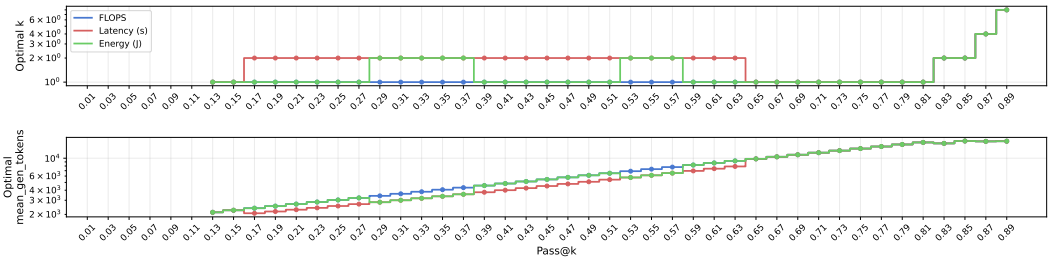


(a) Sequential Power Scaling (b) Parallel Power Scaling.

Figure 9: Power Scaling with Qwen 3.5 Family of Models



(a) **Accuracy and Efficiency Cost Scaling for Variable TTC Configurations**. FLOPs-, Latency-, and Energy-optimal configurations select differing test-time configurations along the task's iso-accuracy curve.



(b) **Optimal Test-Time Configurations For Each Target Accuracy**. Longer reasoning lengths and repeated samples are required to attain higher accuracies, with energy-optimal approaches selecting for lower parallelism and longer outputs.

Figure 10: Qwen 3.5-35BA3B Pareto-Optimal Test-Time Scaling

E Additional Results on Less Reasoning-Intensive Tasks

In addition to AIME26, we find that distinct energy-optimal scaling exists for other math and code tasks. Notably, Math500 and LiveCodeBench both exhibit performance scaling with both sequential and parallel test-time scaling; yielding energy-optimal test-time configurations distinct from latency-optimal configurations.

However, for less reasoning-intensive tasks, such as MBPP, HumanEval, and GSM8k, performance rapidly saturates with the number of sequential output tokens. Tasks which do not benefit from sequential scaling exhibit identical energy-optimal and latency-optimal test-time configurations, with optimal scaling selecting for increased parallelism regardless of energy savings from sequential scaling.

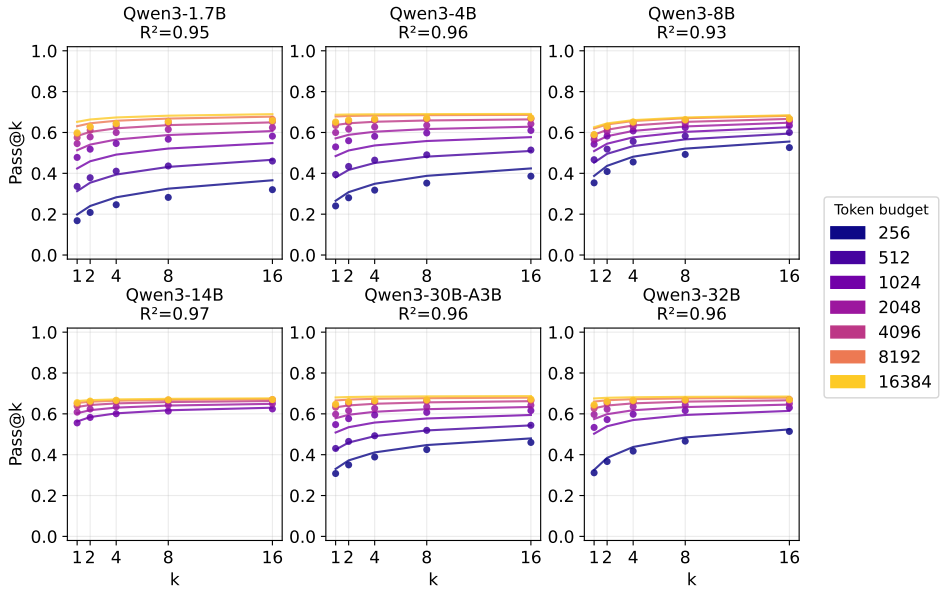


Figure 11: Joint Sequential and Parallel Scaling Coverage Models on Math 500.

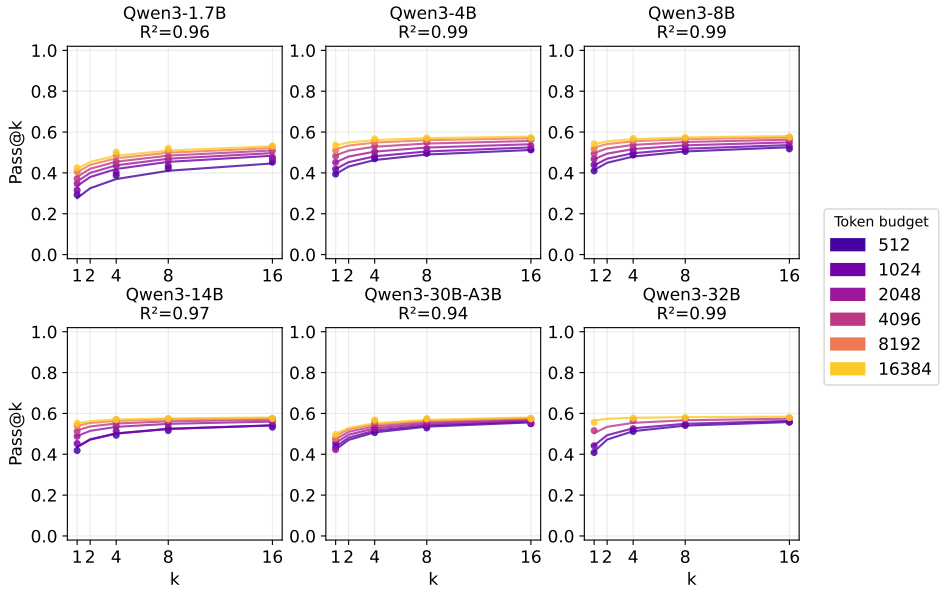


Figure 12: Joint Sequential and Parallel Scaling Coverage Models on LiveCodeBench.

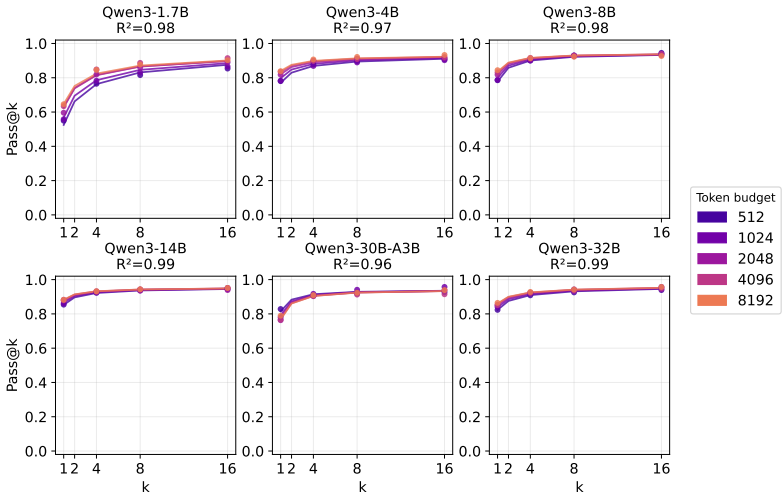


Figure 13: Joint Sequential and Parallel Scaling Coverage Models on HumanEval.

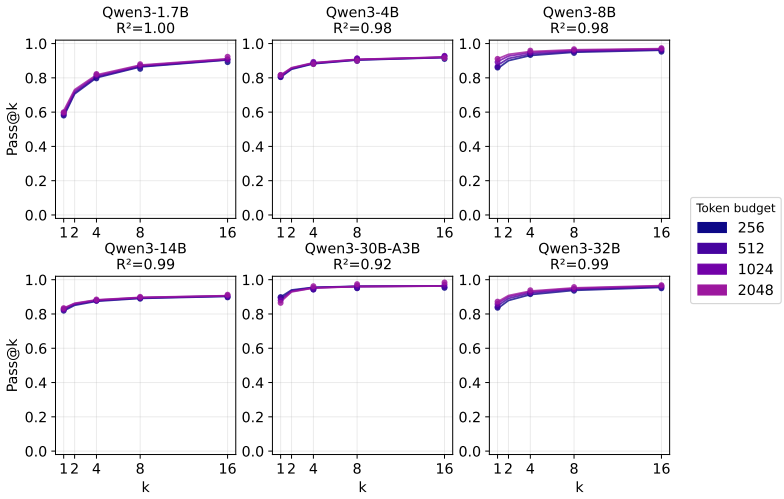


Figure 14: Joint Sequential and Parallel Scaling Coverage Models on MBPP.

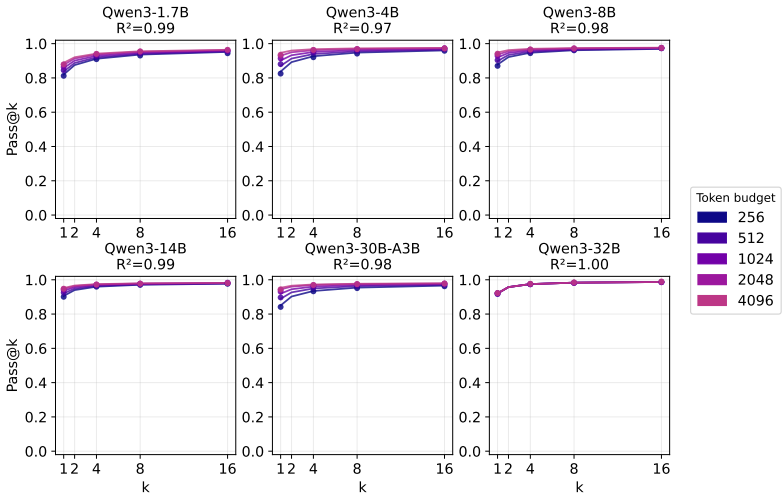


Figure 15: Joint Sequential and Parallel Scaling Coverage Models on GSM8k.

F HARDWARE UTILIZATION

Here we report the hardware utilization of vLLM serving for 8 waves of requests of fixed batch size to each model. GPU power draw remains largely consistent as the GPU processes each batch of requests. Despite high GPU utilization indicating hardware activity, latency-bound operations yield low tensor core and HBM activity, resulting in low power draw and undersaturated hardware at low batch sizes. The Qwen3 30BA3B mixture-of-expert model exhibits both the largest increases in DRAM activity and increases in power as batch size grows.

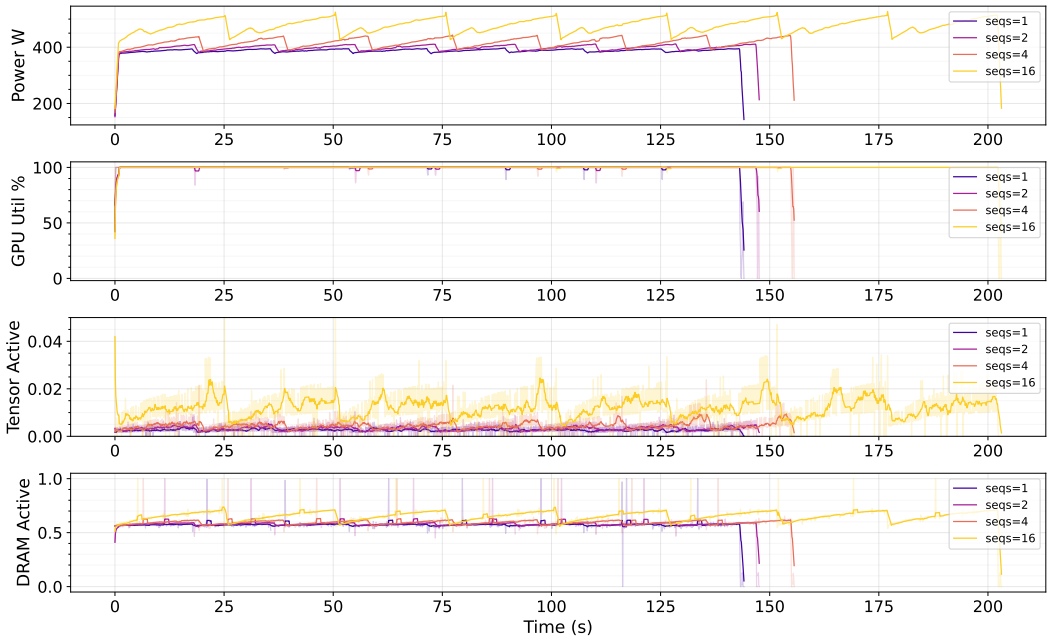


Figure 16: Power Draw and Latency Scale with Increasing Number of Parallel Sequences for Qwen3 4B dense model with 4096 output tokens.

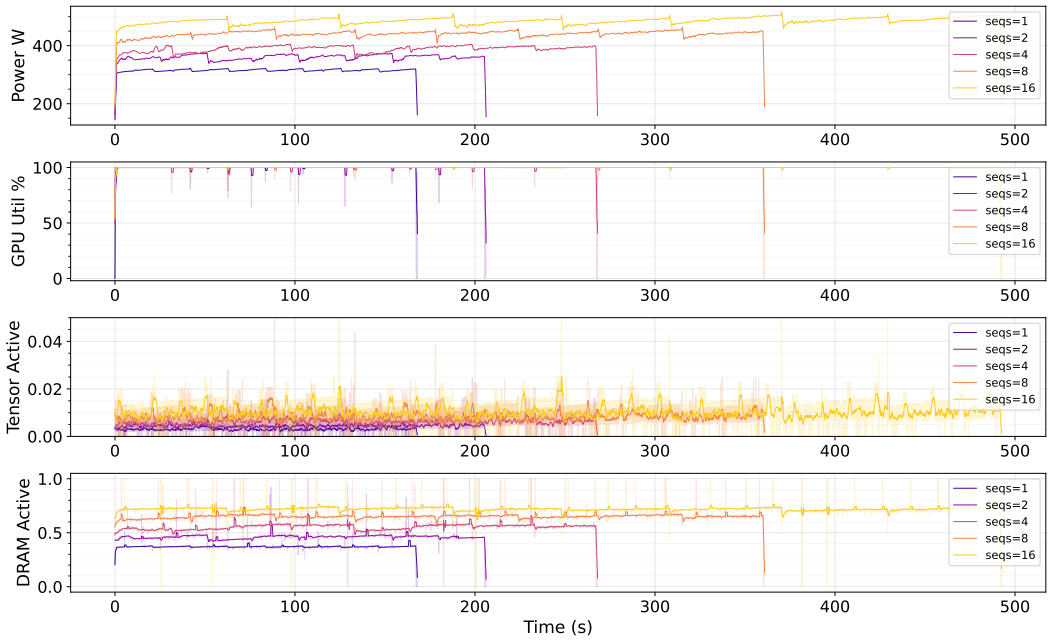


Figure 17: Power Draw and Latency Scale with Increasing Number of Parallel Sequences for Qwen3 30B3A Mixture-of-Experts model with 4096 output tokens

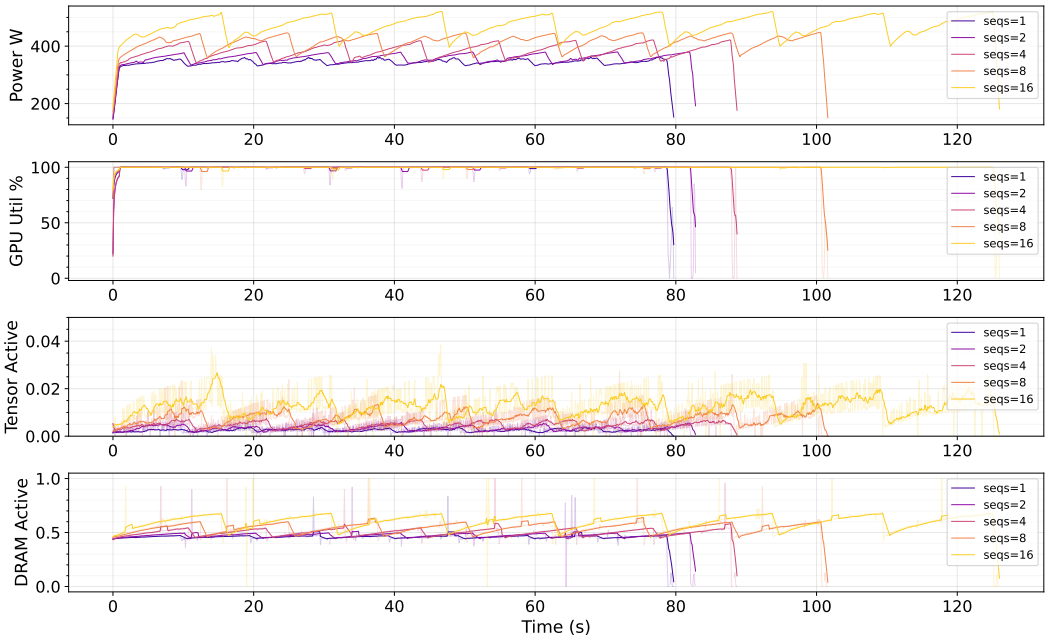


Figure 18: Power Draw and Latency Scale with Increasing Number of Parallel Sequences for Qwen3 1.7B model with 4096 output tokens

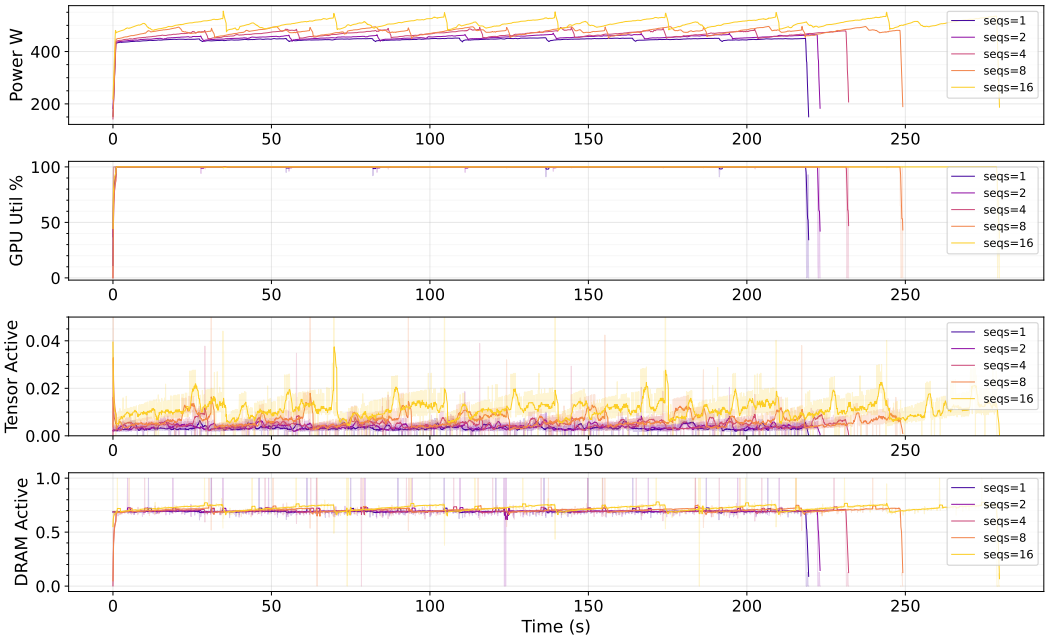


Figure 19: Power Draw and Latency Scale with Increasing Number of Parallel Sequences for Qwen3 8B model with 4096 output tokens

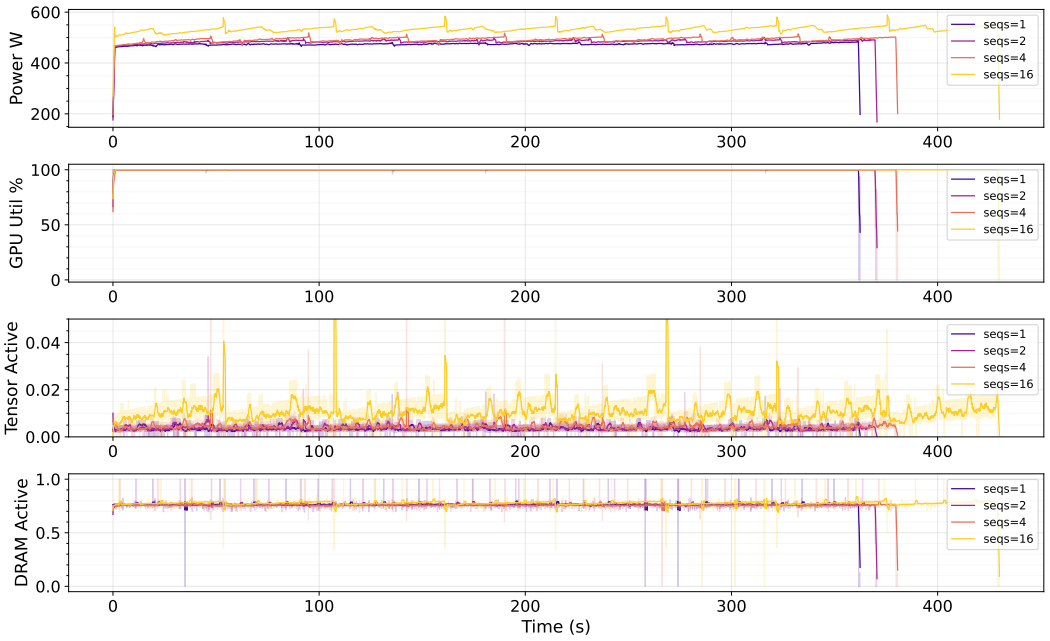


Figure 20: Power Draw and Latency Scale with Increasing Number of Parallel Sequences for Qwen3 14B model with 4096 output tokens

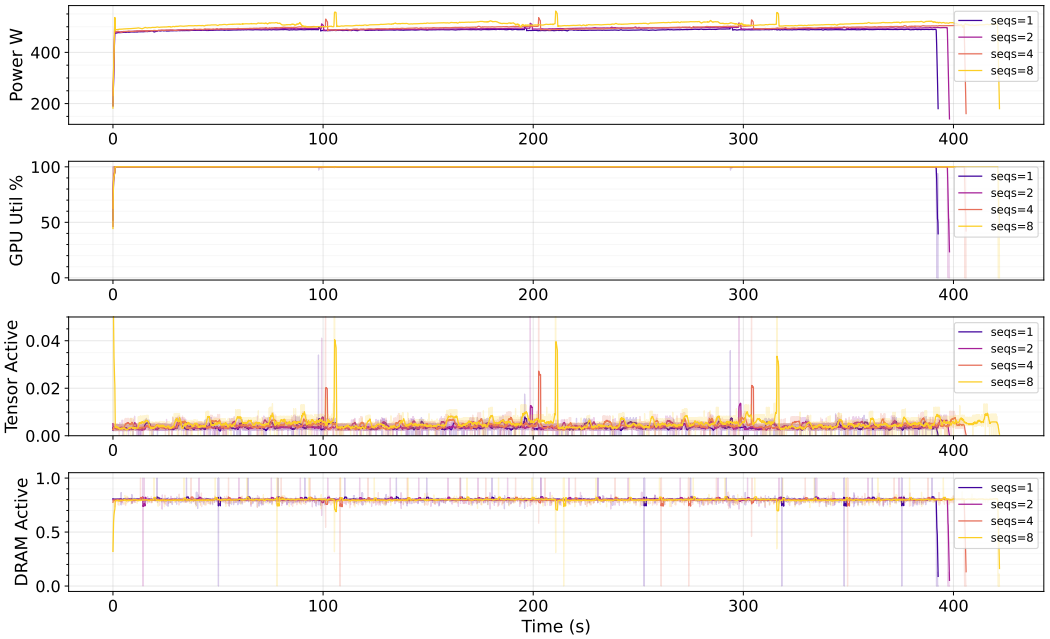


Figure 21: Power Draw and Latency Scale with Increasing Number of Parallel Sequences for Qwen3 32B model with 4096 output tokens