SpecReason: Fast and Accurate Inference-Time Compute via Speculative Reasoning

Rui Pan[§] **Yinwei Dai**[§] **Zhihao Zhang**[†] **Gabriele Oliaro**[†]
Zhihao Jia[†] **Ravi Netravali**[§]
[§]Princeton University [†]Carnegie Mellon University
{ruipan,yinweid}@princeton.edu, {zhihaoz3,goliaro}@cs.cmu.edu,
zhihao@cmu.edu, rnetravali@cs.princeton.edu

Abstract

Recent advances in inference-time compute have significantly improved performance on complex tasks by generating long chains of thought (CoTs) using Large Reasoning Models (LRMs). However, this improved accuracy comes at the cost of high inference latency due to the length of generated reasoning sequences and the autoregressive nature of decoding. Our key insight in tackling these overheads is that LRM inference, and the reasoning that it embeds, is highly tolerant of approximations: complex tasks are typically broken down into simpler steps, each of which brings utility based on the semantic insight it provides for downstream steps rather than the exact tokens it generates. Accordingly, we introduce SpecReason, a system that automatically accelerates LRM inference by using a lightweight model to (speculatively) carry out simpler intermediate reasoning steps and reserving the costly base model only to assess (and potentially correct) the speculated outputs. Importantly, SpecReason's focus on exploiting the semantic flexibility of thinking tokens in preserving final-answer accuracy is complementary to prior speculation techniques, most notably speculative decoding, which demands token-level equivalence at each step. Across a variety of reasoning benchmarks, SpecReason achieves $1.4 - 3.0\times$ speedup over vanilla LRM inference while improving accuracy by $0.4 - 9.0\%$. Compared to speculative decoding without SpecReason, their combination yields an additional $8.8 - 58.0\%$ latency reduction. We open-source SpecReason at `https://github.com/ruipeterpan/specreason`.

1 Introduction

Inference-time compute has unlocked a new axis for scaling AI capabilities. Recent advancements in Large Reasoning Models (LRMs) such as OpenAI o1/o3 [Jaech et al., 2024, ope, 2025] and DeepSeek R1 [Guo et al., 2025] have demonstrated state-of-the-art performance across a wide range of complex tasks. Although these LRMs share the architectural backbones as traditional large language models (LLMs), their inference behavior differs significantly: LRMs first "think" by generating internal *thinking* tokens—tokens that decompose a task into a sequence of composable reasoning steps via a long chain-of-thought (CoT) [Wei et al., 2022] before producing the final tokens that summarize the reasoning process.

Despite their promise, LRMs incur substantial inference latency due to the length of the reasoning sequences they generate. This challenge is primarily driven by the autoregressive nature of LLMs, where decoding time scales linearly with sequence length. As a result, final output generation can routinely take minutes, if not hours, to answer a single query; such delays far exceed those of typical LLMs and are prohibitively slow for many interactive applications, ultimately degrading user experience [Fu et al., 2024b].

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: NeurIPS 2025 Workshop on Efficient Reasoning.

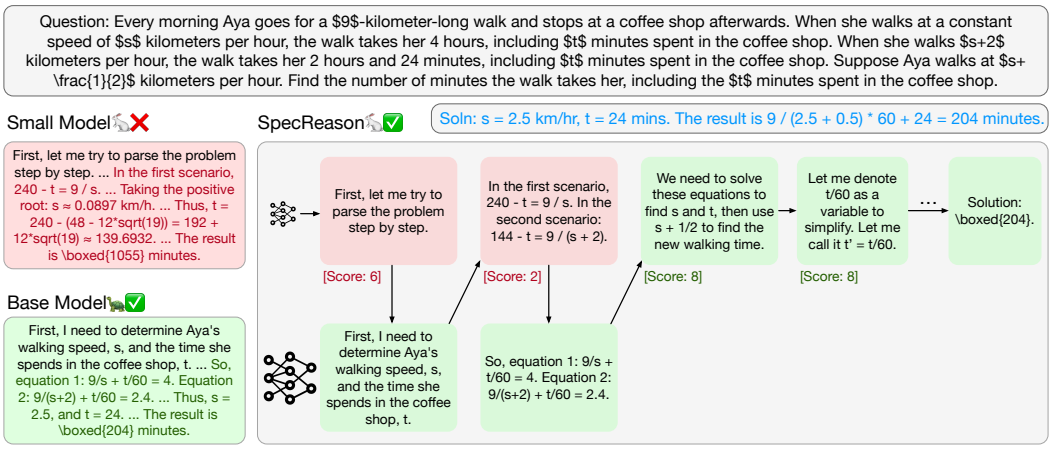


Figure 1: SpecReason leverages a smaller reasoning model to speculate individual reasoning steps, deferring to the base model only for assessment (and optionally as a fallback), enabling faster yet accurate reasoning. For illustration, we show a math question as an example; our evaluation includes more general reasoning workloads.

Our approach to tackling reasoning delays—**without compromising accuracy**—is rooted in two fundamental properties of LRMs: (1) LRMs tackle difficult tasks by generating long CoTs that decompose them into many simpler, sequential steps. For example, in mathematical problem solving, a few key reasoning steps require complex long-term planning and have a major influence on downstream reasoning, while most subsequent steps simply execute the plan through straightforward calculations or case analyses (Fig. 1); (2) The utility of an individual reasoning step hinges less on the exact wording of the thinking tokens but more on the *semantic insight* it provides. That is, as long as a step contributes meaningfully to advancing the CoT, it remains effective—even if phrased imprecisely or differently (Fig. 2). Moreover, LRMs possess self-reflection capabilities that enable them to revise or correct occasional missteps from earlier steps.

Taken together, these properties make the decoding of thinking tokens—the dominant source of inference latency in LRMs—inherently more *approximation tolerant* than typical LLM decoding. A large fraction of intermediate reasoning steps can be effectively handled by lightweight reasoning models, which both align with the nature of these steps and can tolerate minor inaccuracies. As shown in Fig. 3, this opens the door to significantly faster inference without sacrificing output quality.

Building on these insights, we propose **SpecReason**, a system for accelerating LRM inference by selectively offloading easier intermediate steps to be *speculated* by a smaller model without compromising final output accuracy. SpecReason employs a lightweight reasoning model to generate individual reasoning steps, while reserving the slower but more capable base model to efficiently verify these speculated steps (§4.1) and guide the reasoning process along the correct trajectory (Fig. 1). Consistent with prior findings [Song et al., 2025], we observe that base models can be prompted to act as critic models—assessing the utility of intermediate steps and accepting or rejecting them as needed (Fig. 7).

Speculative reasoning vs. speculative decoding. While SpecReason is conceptually related to speculative decoding [Leviathan et al., 2023], which accelerates LLM inference by using a smaller draft model to predict future tokens, there are key distinctions between the two. Most notably, speculative decoding is an *exact* optimization: it relies on *token*-level equivalence between the small and base models, i.e., focusing on typical LLM serving where all generated tokens are part of the final model output being assessed. In contrast, SpecReason explicitly leverages the *approximation tolerance* inherent in reasoning: it targets *thinking tokens*—intermediate steps in the reasoning process—where semantic alignment, rather than token-level equivalence, is sufficient. This relaxation enables substantial latency savings during LRM inference, as semantically similar intermediate steps (Fig. 2) are often adequate to preserve end-task accuracy (Fig. 3). In many cases, SpecReason even *improves* final accuracy over the base model by generating fewer unnecessary tokens (Fig. 4). To further address the high inference cost of LRMs, SpecReason also exposes a user-configurable knob that allows trading off accuracy for latency by adjusting the tolerance level

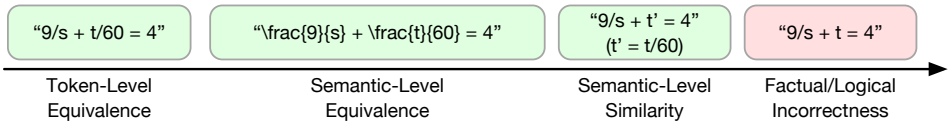


Figure 2: The spectrum of approximations of one example reasoning step (equation 1 in Fig. 1). SpecReason can control the exactness of reasoning approximations by adjusting its acceptance threshold to navigate through the accuracy-latency tradeoff space (§5.3).

for speculative approximations. Finally and most importantly, because speculative reasoning and speculative decoding operate at different levels, we show that they are *complementary* techniques (§4.2), and when combined in a hierarchical speculation framework, achieve even greater reductions in inference latency.

We evaluate SpecReason across a wide range of reasoning workloads spanning tasks of varying complexity [aim, 2025, Hendrycks et al., 2021, Rein et al., 2024]. Overall, SpecReason reduces end-to-end inference latency by $1.4 - 3.0\times$ compared to vanilla LRM inference while improving accuracy by $0.4 - 9.0\%$. Moreover, SpecReason can be *combined* with speculative decoding to provide an additional $8.8 - 58.0\%$ improvement over speculative decoding alone.

2 Background

Inference-time scaling. LRMs introduce a structured problem-solving approach that breaks down complex problems into multiple simpler reasoning steps, commonly referred to as a long chain of thought (CoT) [Wei et al., 2022]. This enables the model to generate intermediate reasoning steps before progressing further, reflect, and backtrack to correct errors if needed. LRMs that output long CoTs have been a popular approach to scale inference-time compute [Guo et al., 2025, Jaech et al., 2024, ope, 2025], and there also exist other schemes like Tree of Thoughts [Yao et al., 2023], process-reward-model-guided tree search [Lightman et al., 2023, Qi et al., 2024, Guan et al., 2025], and repeated sampling for scaling inference-time compute [Brown et al., 2024].

Speculative decoding. Speculation has long been a classic concept in the literature of computer architecture [Burton, 1985]. Due to the memory-bound nature of LLM decoding, recent work has also leveraged the technique of speculation to accelerate the decoding phase [Stern et al., 2018, Leviathan et al., 2023, Yan et al., 2024] of LLM inference. The speculative decoding process alternates between speculation and verification steps to ensure correctness while achieving speed-ups. The speculation phase usually consists of either a standalone draft model [Leviathan et al., 2023, Miao et al., 2024], a trainable module on top of the base model [Cai et al., 2024, Li et al., 2025], a tree-based token cache [Oliaro et al., 2024, Luo et al., 2024, Zhao et al., 2024], an n-gram lookup table [Fu et al., 2024a], or a retrieval-based data store [He et al., 2023] to make efficient but less accurate speculations. The verification process, on the other hand, is a base model chunked-prefill over the speculation results, which usually consists of either a single sequence of tokens as in Leviathan et al. [2023] or tree-like structures to further boost the accuracy of speculation [Miao et al., 2024, Cai et al., 2024, Li et al., 2025, Chen et al., 2024]. The verification process then accepts the longest matched sequences on the token level from the speculation results and repeats the process. As a result, the speculation length is usually conservative to maintain an optimal trade-off between the speculation overhead and accuracy.

Existing approaches for reducing latency. Sky-T1-Flash Team [2025] reduces unnecessary thinking tokens by fine-tuning models to curb overthinking, thereby reducing the length of reasoning chains and, consequently, latency. Dynasor-CoT Fu et al. [2024b, 2025] takes a different approach by probing intermediate model confidence and terminating the reasoning process early when the model exhibits sufficient confidence in its current output.

3 Motivation

In this work, we show that reasoning workloads executed by LRMs exhibit unique opportunities for latency reduction due to their inherent tolerance to approximation— setting them apart from

traditional generation tasks in LLMs. We illustrate these properties using a representative example from the AIME dataset, selected for its clarity and ease of exposition.

Intermediate steps are easier than end-to-end reasoning. A key observation in LRM behavior is that reasoning difficulty is not uniform across the steps in a long chain-of-thought (CoT). As shown in Fig. 1, while the overall task might be too challenging for a small model to solve end-to-end, only a few critical steps—such as problem analysis, decomposition through formulations or case analyses, and high-level planning—are critical to the overall reasoning progress. In contrast, many other steps are significantly easier.

This behavior is intentional by design: LRMs are often trained with reinforcement learning to generate CoTs that decompose complex problems into sequences of simpler, more tractable reasoning steps. These intermediate steps often include routine reasoning such as arithmetic calculations, case enumeration, or basic logical deductions—operators that are much easier to decode than synthesizing a full solution directly. This heterogeneity in step difficulty and importance creates an opportunity for lightweight models to handle a substantial portion of the reasoning process both efficiently and accurately.

Reasoning progress depends on insights, not exact tokens. Another key takeaway from our work is that the utility of a reasoning step lies in the semantic contribution it makes to the overall reasoning process, rather than the precise tokens it uses. Unlike tasks like translation in traditional LLM inference, where fidelity to exact combinations of tokens matters more, reasoning CoTs within LRM's thinking tokens care more about the information that advances the reasoning chain. As illustrated in Fig. 2, a spectrum of valid phrasings often exists for a given step: semantically equivalent or similar expressions can convey the same insight and lead to the same downstream reasoning trajectory. This semantic flexibility is a key enabler for approximation-tolerant inference.

Occasional mistakes can be corrected via self-reflection. LRMs exhibit strong self-reflection capabilities, enabling them to recover from earlier reasoning errors. Even when an earlier step contains a factual or logical mistake, the model often revises its trajectory in subsequent steps, marked by tokens like "Wait" or "Hmm". Moreover, unlike LLM inference where *all* output tokens contribute to the final answer, in LRM inference, only the tokens generated *after* the thinking tokens determine the final outcome. Therefore, LRM inference can tolerate occasional mistakes during the reasoning phase, as the model can often identify and correct these mistakes during self-reflection. This inherent fault tolerance further underscores the viability and effectiveness of approximation-based acceleration.

In summary, compared to traditional LLM inference, LRM inference is inherently more tolerant of approximations that do not require token-level equivalence as long as the overall reasoning trajectory is preserved. This property is not limited to a single, linear CoT; rather, it extends naturally to more general inference-time compute scaling paradigms such as tree-based search strategies and other structured reasoning approaches.

4 Method

4.1 Speculative Reasoning

Due to its reliance on autoregressive decoding, LRM inference incurs significantly higher latency than typical LLMs—often to the point of being prohibitively slow for interactive applications and degrading user experience [Fu et al., 2025]. Existing approaches for latency reduction include using a distilled version of the base model [Guo et al., 2025], limiting the number of thinking tokens via a predefined *token budget*, or disabling the reasoning process altogether by omitting the thinking tokens (`<think>` and `</think>`) during generation [qwe, 2025]. However, these approaches impose a harshly trade-off between accuracy for latency: they either limit the model's capacity to reason or apply a lower-quality model uniformly across all reasoning steps. In contrast, SpecReason takes a more fine-grained and adaptive approach. Instead of explicitly restricting output length, it selectively offloads only the easier reasoning steps to a lightweight model, preserving overall reasoning quality while substantially reducing inference latency.

The approximation-tolerant nature of LRM reasoning enables a new form of speculative execution: tentatively carrying out reasoning steps using a lightweight model, assessing their utility with a stronger base model, and selectively accepting them. SpecReason leverages this flexibility to reduce

decoding latency while preserving output quality. To achieve this goal, SpecReason offloads easier or less critical reasoning steps—defined as semantically self-contained units such as complete sentences or logical steps—to a smaller, faster *speculator* model. Each step is decoded in two stages: (1) the lightweight speculator proposes the next reasoning step based on the current context, and (2) the base model evaluates the proposed step for semantic utility. If the step is accepted, SpecReason proceeds to the next step; otherwise, SpecReason falls back to the base model to regenerate the step. While our implementation uses a simple static-threshold mechanism for verification, the framework supports richer, customizable decision strategies. We outline key design principles below.

Navigating the Pareto frontier of the latency-accuracy tradeoff. SpecReason expands the Pareto frontier of the latency-accuracy tradeoff by exposing fine-grained control knobs to navigate through this space. The key knob SpecReason employs is the acceptance threshold: after each speculated reasoning step, the base model is prompted to generate a single-token utility score (e.g., an integer from 0 to 9) indicating the quality of the step. If the utility score is above a static acceptance threshold (e.g., score ≥ 7), the speculated reasoning step is accepted; otherwise, it is discarded and regenerated by the base model.

Adjusting this threshold allows users to control the *strictness* of speculation (Fig. 5): a higher threshold requires speculated steps to be closer to token-level equivalence on the equivalence spectrum (Fig. 2), improving accuracy but reducing the acceptance rate and thereby increasing latency. Conversely, a lower threshold increases speculation efficiency at the cost of potential accuracy degradation.

An additional knob involves forcing the first n reasoning steps to be decoded by the base model. Since LRMs often use the initial steps to analyze the problem and formulate a high-level plan, assigning these initial steps to the base model can steer the overall reasoning trajectory toward higher quality. We show in Fig. 6 that this knob also allows SpecReason to manage the latency-accuracy tradeoff, though with less impact than the acceptance threshold knob.

While our current implementation uses a simple, discrete threshold-based scoring scheme—offering only a coarse-grained configuration space—it establishes a lower bound on verification quality. Future work can explore more sophisticated strategies, such as logprob-based confidence estimates or dynamic thresholds, to enable finer-grained tradeoffs without incurring additional runtime cost, and may further improve overall performance.

Efficient verification. Because each step requires verification by the base model, it's crucial to keep verification overhead low to avoid compounding latency. Instead of autoregressively decoding or reranking multiple candidate steps, SpecReason evaluates each speculated step in a single *prefill-only* pass of the base model. The verification prompt is templated to reuse most of the CoT prefix, so each verification requires prefilling only $\sim$70 new tokens. Since short-prefill forward passes are memory-bound, the overhead is comparable to decoding just 1–2 tokens, making verification highly efficient in practice.

Implementation details. Since the small model is lightweight, we colocate both the small and base models on the same GPU. The memory reserved for Key-Value caches [Kwon et al., 2023] is statically partitioned between the two models. They do not share any internal model states–only the token IDs of the generated reasoning steps are managed and shared by SpecReason. If a speculative step is rejected, the corresponding KV cache entries are discarded.

Inference is performed sequentially: the small and base models take turns, avoiding kernel-level interference. In future work, we plan to explore pipelining to overlap the small model's decoding with the base model's inference. While this may introduce mild resource contention, it could further reduce end-to-end latency.

4.2 Hierarchical Speculation across Semantic Similarity and Token Equivalence

At a high level, SpecReason's speculative reasoning resembles the philosophy behind traditional speculative decoding, but differs in two important ways. First, speculative decoding guarantees token-level equivalence between draft and verified outputs, making it a form of exact acceleration. In contrast, SpecReason targets semantic-level similarity, accepting steps that carry the same insight even if phrased differently, and exposes knobs to control the exactness of reasoning approximations. Second, speculative decoding is typically applied to output generation tasks (e.g., text continuation or translation), where the fidelity of each token matters. SpecReason, on the other hand, is designed

specifically for internal thinking tokens in reasoning tasks, where intermediate steps are approximate and interchangeable as long as they preserve the logical progression of thought.

Further, because SpecReason and speculative decoding operate at different levels (semantic-level similarity vs. token-level equivalence), these two approaches are complementary and can be combined into a unified, hierarchical system – SpecReason+Decode first applies step-level speculative reasoning to draft and verify reasoning steps. If a step is rejected and regenerated by the base model, standard token-level speculative decoding can be applied during the base model regeneration to further accelerate decoding.

5 Evaluation

The overview of our evaluation results includes:

- **Reducing end-to-end latency.** Because many intermediate steps are easier than end-to-end reasoning, many (up to 80%) of the speculated steps are accepted. SpecReason achieves a $1.4 - 3.0\times$ speedup over vanilla LRM inference. Additionally, when combined with speculative decoding, SpecReason further reduces latency by $8.8 - 58.0\%$ over speculative decoding alone, highlighting the complementary nature of these optimizations.
- **Improving token-budget-aware accuracy.** Beyond latency reduction, SpecReason also improves accuracy over the base model by $0.4 - 9.0\%$ under the same token budget. We empirically find that small, lightweight models typically have shorter output sequence lengths – meaning, they need fewer thinking tokens before deriving an answer. Thus, by accepting many small model's speculated reasoning steps, SpecReason reduces the token consumption compared to the base model's vanilla inference. When the token budget is low – a common setup to curb inference cost and latency – SpecReason helps improve accuracy as the base model would need more tokens to get to an answer (Fig. 4).

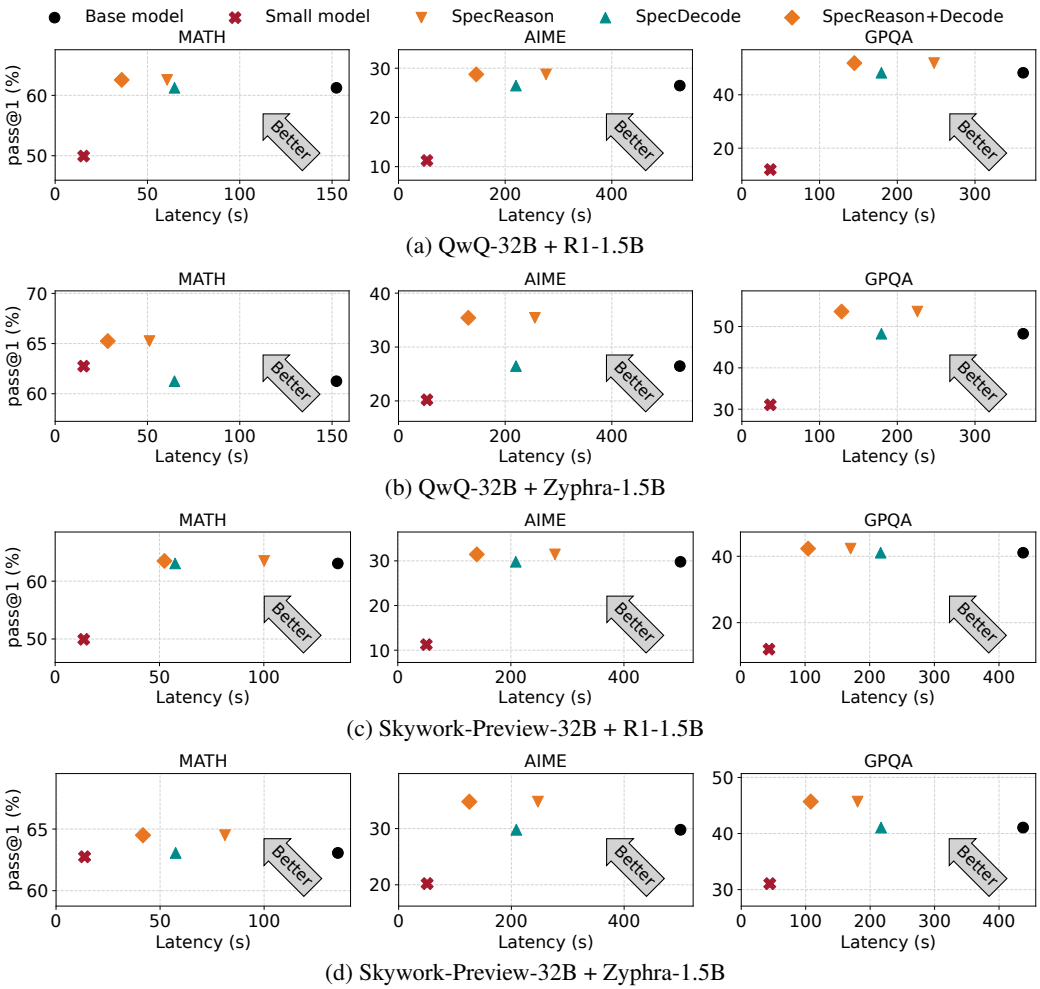
5.1 Setup

Models. In our main results, we use two base models: QwQ-32B [qwq, 2025] and Skywork-OR1-Preview-32B [sky, 2025]. We also use two different small models for speculation: DeepSeek-R1-1.5B [Guo et al., 2025] and Zyphra's ZR1-1.5B [zyp, 2025] – both of which are based on Qwen-2.5 [Yang et al., 2024] and embed the capability of reasoning with long CoTs – and evaluate all four different model combinations. We evaluate an additional base model with a different size and architecture, R1-70B [Guo et al., 2025], a distilled version of DeepSeek-R1 onto Llama3.3-70B [Grattafiori et al., 2024], in §A.1.

Datasets. We evaluate SpecReason on three diverse reasoning benchmarks: AIME [aim, 2025] for high-school competition-level mathematical problems, MATH500 [Hendrycks et al., 2021] for high-school competition-level mathematical problems sampled from AMC 10, AMC 12, and AIME, and GPQA Diamond [Rein et al., 2024] for graduate-level questions in general domains like biology, physics, and chemistry. The accuracy metric we evaluate on is pass@1. Similar to prior work [Guo et al., 2025], we set k=16 when calculating pass@1 – i.e., we generate 16 responses with temperature=0.6 for every query and calculate the average accuracy – and set the token budget to be 8192 tokens to ensure an apples-to-apples comparison between baselines.

Baselines. We run vanilla inference using the small and base models as the latency and accuracy baseline, respectively. Aside from SpecReason, we also run speculative decoding ("SpecDecode") with the smaller model as the draft model, speculating five tokens at a time. To demonstrate SpecReason's compatibility with speculative decoding, we also run a "SpecReason+Decode" baseline that employs the hierarchical speculation described in §4.2.

Hardware. We run our evaluations on two NVIDIA A6000-48GB GPUs. We use vLLM Kwon et al. [2023] 0.8.2 as the underlying inference engine and enable prefix caching. Both models are served with a tensor parallelism degree of two.



(a) QwQ-32B + R1-1.5B

(b) QwQ-32B + Zyphra-1.5B

(c) Skywork-Preview-32B + R1-1.5B

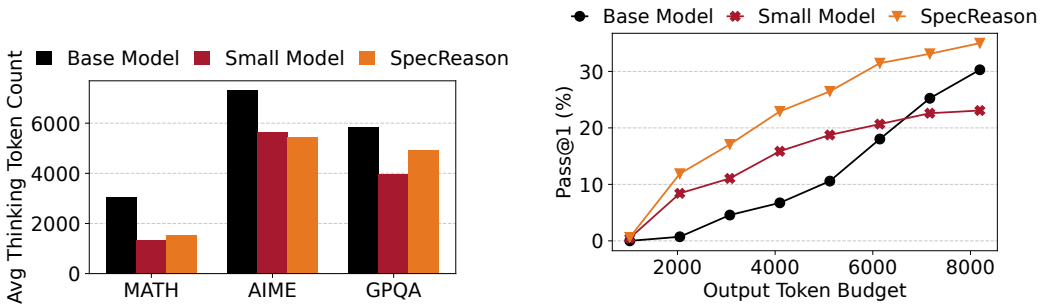
(d) Skywork-Preview-32B + Zyphra-1.5B

Figure 3: Comparison of the accuracy and latency of different schemes on different model combinations. SpecReason significantly reduces latency while improving accuracy over vanilla inference. When combined with speculative decoding, SpecReason outperforms speculative decoding in both latency and accuracy on all datasets and model combinations.

5.2 Main Results

We compare SpecReason against baseline methods in Fig. 3. Across the four model combinations, SpecReason achieves a 1.5×–2.5×, 1.6×–3.0×, 1.4×–2.5×, 1.7×–2.4× reduction in latency, respectively, compared to vanilla inference with the base model.

Accuracy improvement. Alongside these efficiency gains, SpecReason also yields modest accuracy improvements of 1.3%–3.6%, 4.0%–9.0%, 0.4%–1.7%, and 1.4%–5.0% compared to the base model. The key reason behind this accuracy improvement is the reduction in token consumption required for reasoning. In Fig. 4, we focus on the model combination with the highest overall accuracy improvement, QwQ-32B + Zyphra-1.5B, and compare the average number of thinking tokens needed to derive an answer between the base model, the small model, and SpecReason. As seen in Fig. 4a, the small model is generally less verbose than the base model, and because SpecReason adopts many speculated steps from the small model, its token consumption is also reduced by 1.2×–2.0×. We also focus on the AIME dataset and vary the token budget to study its effect on the difference in accuracy between SpecReason and the base model in Fig. 4b. The effect of token reduction on accuracy is the most significant for tighter output token budgets (16.2% at 4096 tokens) but shrinks as the base model is allowed to generate more thinking tokens (4.7% at 8192 tokens). We also attribute these accuracy gains to SpecReason's explicit judgment and scoring mechanism at each reasoning step, which augments the model's internal self-reflection with more structured assessment.



(a) Output length comparison. SpecReason reduces the token consumption needed to answer queries by adopting speculated steps from small models that are less verbose.

(b) [AIME] Accuracy gap under different token budgets.

Figure 4: [QwQ-32B + Zyphra-1.5B] Intuition behind SpecReason's accuracy improvement. See Fig. 9 in §A for the full set of results.

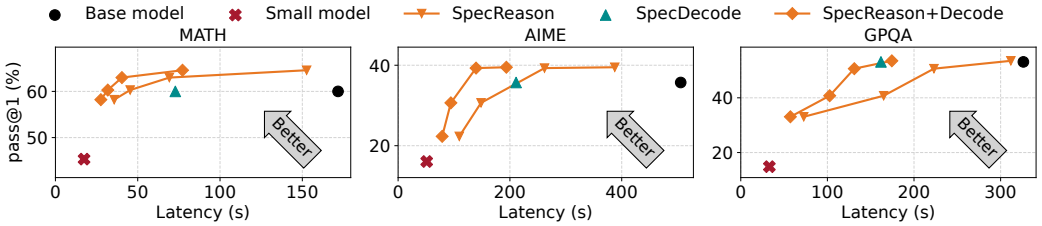


Figure 5: [QwQ-32B + R1-1.5B] SpecReason allows trading off latency for accuracy via adjusting the acceptance threshold (from left to right, the thresholds are: 3, 5, 7, and 9 out of 9).

When compared with speculative decoding, SpecReason lies on the Pareto frontier of the accuracy-latency tradeoff. More importantly, combining SpecReason with speculative decoding (SpecReason+Decode) results in further latency reductions of 19.4%–44.2%, 30.8%–58.0%, 8.8%–52.2%, and 25.1%–51.8% over speculative decoding alone. The most significant performance gains for SpecReason when the base model is QwQ-32B occur on the MATH dataset, where both models achieve relatively high accuracies and the capability gap between the small and base models is the narrowest. This makes intermediate steps easier for the small model to speculate correctly, increasing the acceptance rate of speculated steps and thereby lowering end-to-end latency. In comparison, Skywork-Preview-32B is slightly inferior at instruction following, so SpecReason has to adopt a higher threshold to avoid an accuracy loss, reducing SpecReason's latency wins.

Finally, when comparing SpecReason+Decode with SpecReason, SpecReason+Decode reduces latency by 1.7×–1.9×, 1.7×–1.8×, 1.6×–2.2×, and 1.6×–2.1×, demonstrating the difference in ease of speculation across varying tasks. On these three datasets, the ratio of steps carried out by small models in SpecReason is 38.1%–80.0%, 36.5%–71.3%, 39.3%–70.2%, and 41.4%–66.6%, respectively.

5.3 Controlling the Accuracy-Latency Tradeoff

In Fig. 5, we illustrate how SpecReason enables flexible control over the accuracy-latency tradeoff, using a representative, randomly selected subdataset from the full datasets in §5.2 on QwQ-32B + R1-1.5B for ease of evaluation. During the base model's evaluation of each reasoning step, we vary the acceptance threshold for the utility score between 3, 5, 7, and 9, and report the resulting accuracy and latency.

On the MATH subdataset, increasing the acceptance threshold from 3 to 7 results in fewer speculative steps from the small model being accepted. This leads to a latency increase from 35.7s to 69.2s, while accuracy improves from 59.4% to 63.7%, due to tighter control over the approximation level of intermediate reasoning steps. Notably, the gap between SpecReason+Decode and SpecRea-

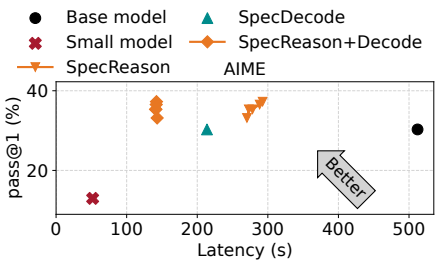


Figure 6: Effect of the alternative knob: forcing the first n steps for base model decoding.

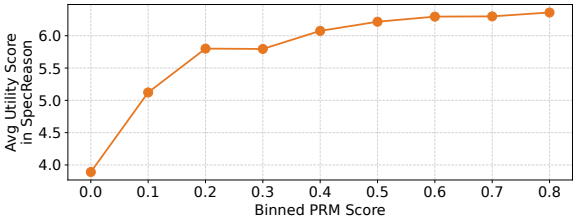


Figure 7: The utility scores in SpecReason closely reflect the quality score judgements from a process reward model. x on the x-axis denotes PRM scores in the range $[x, x + 0.1)$.

son widens from 8.1s to 28.8s, since more reasoning steps are delegated to the base model, and SpecReason+Decode reduces only the base model's decoding time compared to SpecReason.

A similar trend is observed on the AIME and GPQA subdatasets: as the acceptance threshold increases from 3 to 7, latency grows from 109.4s to 261.9s and from 72.7s to 223.0s, and accuracy improves from 22.3% to 39.3% and from 33.1% to 50.7%. However, the accuracy degrades less gracefully as the threshold is relaxed compared to the MATH subdataset. This is because the small model exhibits a larger performance gap relative to the base model on AIME and GPQA, making aggressive acceptance of its speculative steps more costly in terms of accuracy.

In Fig. 6, we also study the effect of the alternative knob, forcing the first n reasoning steps to be decoded by the base model, on the accuracy-latency tradeoff. As we change n from 0 to 10, 20, 30, and 40, SpecReason's accuracy increases from 33.2% to 37.3% while the latency increases from 270.4s to 292.6s, showcasing an alternative approach to improve accuracy with a slight increase in latency.

5.4 Base Model's Judgement Capability

The base model's ability to assess the quality of intermediate reasoning steps is a crucial cornerstone of SpecReason's performance. In this experiment, we compare the scores generated by a process reward model (PRM) – which assigns a reward score to each step within the solution to a math problem – with those given by the QwQ-32B base model on the AIME dataset. Specifically, we use Math-Shepherd [Wang et al., 2023], a PRM trained via reinforcement learning from the Mistral-7B base model on math problems, to score each speculated step produced by the R1-1.5B small model.

In Fig. 7, we bin the reward scores (a float from 0 to 1) into ten bins. Within each bin, we calculate the mean utility score given by the base model in SpecReason. This analysis demonstrates a strong correlation between the base model's and the PRM's assessments, particularly for lower-quality reasoning steps, where both models assign low scores. The results suggest that the base model can effectively approximate the PRM's judgments, making it a viable option for evaluating reasoning step quality in SpecReason.

6 Conclusion

In this work, we introduce SpecReason, a novel approach that accelerates LRM inference by leveraging speculative reasoning. By offloading simpler intermediate reasoning steps to a smaller, lightweight model and reserving the base model for assessment, SpecReason significantly reduces inference latency while maintaining or even improving accuracy. Our results demonstrate that SpecReason achieves a $1.4 - 3.0\times$ speedup over vanilla LRM inference, with accuracy improvements ranging from $0.4 - 9.0\%$. Additionally, when combined with speculative decoding, SpecReason further reduces latency by $8.8 - 58.0\%$, highlighting the complementary nature of these optimizations. We believe this work opens up new angles for efficient LRM inference acceleration, making it especially valuable for scenarios that demand both high accuracy and low latency.

Acknowledgments and Disclosure of Funding

We thank Princeton's Systems for Artificial Intelligence Lab (SAIL) and Princeton Language and Intelligence (PLI) for providing the hardware resources for running experiments. Rui would like to thank Chongyi Zheng for sharing his experience on submitting to NeurIPS. This work was supported by NSF CNS grants 2147909, 2151630, 2140552, 2153449, and 2152313.

References

Aime 2024 dataset card. https://huggingface.co/datasets/HuggingFaceH4/aime_2024, 2025.

Openai o3-mini system card. https://cdn.openai.com/o3-mini-system-card-feb10.pdf, 2025.

Qwen3: Think deeper, act faster. https://qwenlm.github.io/blog/qwen3/, 2025.

Qwq-32b: Embracing the power of reinforcement learning. https://qwenlm.github.io/blog/qwq-32b/, 2025.

Skywork-or1 (open reasoner 1). https://github.com/SkyworkAI/Skywork-OR1, 2025.

Introducing zr1-1.5b, a small but powerful reasoning model for math and code). https://www.zyphra.com/post/introducing-zr1-1-5b-a-small-but-powerful-math-code-reasoning-model, 2025.

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

F Warren Burton. Speculative computation, parallelism, and functional programming. *IEEE Transactions on Computers*, 100(12):1190–1193, 1985.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yu-Hsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable and robust speculative decoding. *Advances in Neural Information Processing Systems*, 37: 129531–129563, 2024.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024a.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Aurick Qiao, and Hao Zhang. Efficiently serving llm reasoning programs with certaindex. *arXiv preprint arXiv:2412.20993*, 2024b.

Yichao Fu, Junda Chen, Yonghao Zhuang, Zheyu Fu, Ion Stoica, and Hao Zhang. Reasoning without self-doubt: More efficient chain-of-thought through certainty probing. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025.

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

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025.

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

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D Lee, and Di He. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252*, 2023.

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. *arXiv preprint arXiv:2103.03874*, 2021.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Xianzhen Luo, Yixuan Wang, Qingfu Zhu, Zhiming Zhang, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. Turning trash into treasure: Accelerating inference of large language models with token recycling, 2024. URL `https://arxiv.org/abs/2408.08696`.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pages 932–949, 2024.

Gabriele Oliaro, Zhihao Jia, Daniel Campos, and Aurick Qiao. Suffixdecoding: A model-free approach to speeding up large language model inference, 2024. URL `https://arxiv.org/abs/2411.04975`.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. Mutual reasoning makes smaller llms stronger problem-solvers. *arXiv preprint arXiv:2408.06195*, 2024.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. Prmbench: A fine-grained and challenging benchmark for process-level reward models. *arXiv preprint arXiv:2501.03124*, 2025.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

NovaSky Team. Think less, achieve more: Cut reasoning costs by 50% without sacrificing accuracy. https://novasky-ai.github.io/posts/reduce-overthinking, 2025. Accessed: 2025-01-23.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.

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.

Minghao Yan, Saurabh Agarwal, and Shivaram Venkataraman. Decoding speculative decoding. *arXiv preprint arXiv:2402.01528*, 2024.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Yao Zhao, Zhitian Xie, Chen Liang, Chenyi Zhuang, and Jinjie Gu. Lookahead: An inference acceleration framework for large language model with lossless generation accuracy. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 6344–6355. Association for Computing Machinery, 2024. ISBN 9798400704901. doi: 10.1145/3637528.3671614.

A Appendix

A.1 Base Models of Varying Sizes and Architectures

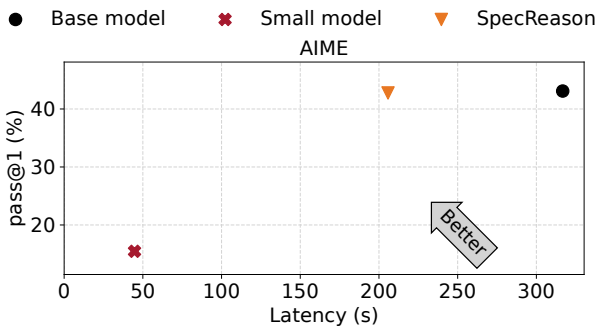


Figure 8: SpecReason's results on the model combination (R1-70B, R1-1.5B).

To demonstrate the generality of SpecReason, we replace the QwQ-32B base model with DeepSeek's R1-70B and evaluate on the same representative subdatasets as in §5.3. Given the size of the R1-70B model, we deploy it across four A100-80GB GPUs using a tensor parallelism degree of 4.

On the AIME subdataset, SpecReason achieves a $1.5\times$ latency reduction compared to vanilla R1-70B inference. This speedup is smaller than the gains observed with the QwQ-32B model in our main results ($1.9\times$) due to two key factors. First, the R1-70B model benefits from both stronger hardware and greater parallelism (4-way TP on A100s), resulting in a $1.5\times$ lower time-per-token (TPT) compared to QwQ-32B (2-way TP on A6000s). In contrast, the smaller model R1-1.5B sees only a modest $1.1\times$ TPT improvement on stronger hardware, which narrows the performance gap between base and small models and thus diminishes latency savings. Second, QwQ-32B is empirically a stronger model – outperforming R1-70B across many reasoning benchmarks qwq [2025] – and this performance gap impacts their respective abilities to assess intermediate steps. To maintain accuracy, we adopt a stricter acceptance threshold when using R1-70B as the base model, which reduces the fraction of steps offloaded to the small model (23.2% compared to 40.8% in the main results).

A.2 Intuition behind Accuracy Improvement

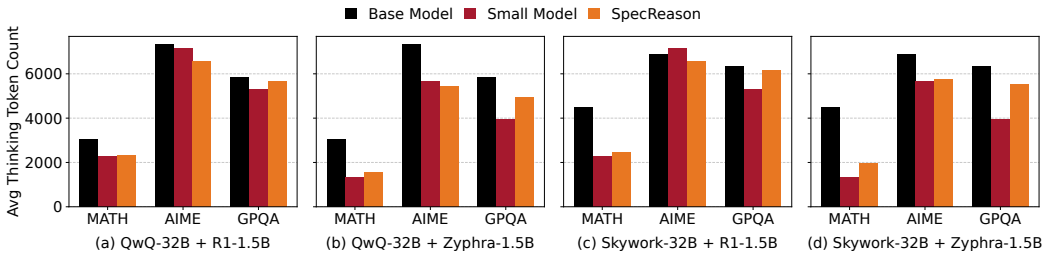


Figure 9: Intuition behind SpecReason's accuracy improvement on all datasets and model combinations.

In Fig. 9, we evaluate the average thinking token count of SpecReason and two vanilla inference baselines on a wide range of datasets and model combinations. We observe that the small model is generally less verbose than the base model, and because SpecReason adopts many speculated steps from the small model, its token consumption is reduced by $1.0 - 1.3\times$, $1.2 - 2.0\times$, $1.0 - 1.8\times$, and $1.1 - 2.3\times$ on the four model combinations, respectively.