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 1.5–2.5× speedup over vanilla LRM inference while improving accuracy by 1.0–9.9%. Compared to speculative decoding without SpecReason, their combination yields an additional 19.4–44.2% latency reduction. We open-source SpecReason at `https://github.com/ruipeterpan/specreason`.

1 Introduction

Inference-time compute has unlocked a new axis in scaling AI progress. The recent advancements of Large Reasoning Models (LRMs) like OpenAI o1/o3 Jaech et al. [2024], ope [2025] and DeepSeek R1 Guo et al. [2025] have demonstrated state-of-the-art accuracy on many complex workloads. Although these LRMs share the same backbone model architecture as traditional Large Language Models (LLMs), their behavior is different: LRMs first "think" by outputting (internal) thinking tokens that break tasks down into composable reasoning steps in a long chain of thought (CoT) Wei et al. [2022] before ultimately outputting tokens that summarize the overall thinking process. Despite their promise, the high quality these models achieve comes at the cost of high inference latency, which is primarily driven by the long reasoning sequences LRMs generate. The problem is further exacerbated by the autoregressive nature of LLMs, which causes latency to scale 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 from typical LLMs and are prohibitively slow for many interactive applications, ultimately hurting users' quality of experience Fu et al. [2024b].

Preprint.

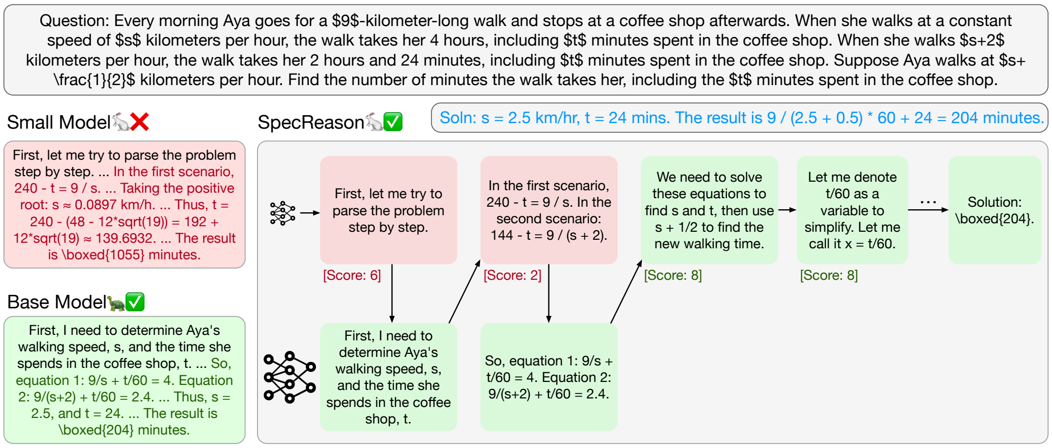
Question: Every morning Aya goes for a 9-kilometer-long walk and stops at a coffee shop afterwards. When she walks at a constant speed of s kilometers per hour, the walk takes her 4 hours, including t minutes spent in the coffee shop. When she walks $s+2$ kilometers per hour, the walk takes her 2 hours and 24 minutes, including t minutes spent in the coffee shop. Suppose Aya walks at $s+ \frac{1}{2}$ kilometers per hour. Find the number of minutes the walk takes her, including the t minutes spent in the coffee shop.
Small Model ❌
First, let me try to parse the problem step by step. ... In the first scenario, 240 - t = 9 / s. ... Taking the positive root: s ≈ 0.0897 km/h. ... Thus, t = 240 - (48 - 12*sqrt(19)) = 192 + 12*sqrt(19) ≈ 139.6932. ... The result is \boxed{1055} minutes.
Base Model ✅
First, I need to determine Aya's walking speed, s, and the time she spends in the coffee shop, t. ... So, equation 1: 9/s + t/60 = 4. Equation 2: 9/(s+2) + t/60 = 2.4. ... Thus, s = 2.5, and t = 24. ... The result is \boxed{204} minutes.
SpecReason ✅
Soln: s = 2.5 km/hr, t = 24 mins. The result is 9 / (2.5 + 0.5) * 60 + 24 = 204 minutes.
First, let me try to parse the problem step by step.
[Score: 6]
First, I need to determine Aya's walking speed, s, and the time she spends in the coffee shop, t.
In the first scenario, 240 - t = 9 / s. In the second scenario: 144 - t = 9 / (s + 2).
[Score: 2]
So, equation 1: 9/s + t/60 = 4. Equation 2: 9/(s+2) + t/60 = 2.4.
We need to solve these equations to find s and t, then use s + 1/2 to find the new walking time.
[Score: 8]
Let me denote t/60 as a variable to simplify. Let me call it x = t/60.
[Score: 8]
...
Solution: \boxed{204}.

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 is rooted in two fundamental properties of LRMs: (1) LRMs approach difficult tasks by generating long CoTs that decompose them into many simpler steps. For example, when solving math problems, a few key reasoning steps require complex long-term planning and greatly influence downstream tasks, while most subsequent steps simply execute the plan through calculations or case analysis (Fig. 1); (2) The utility of a reasoning step hinges less on the exact thinking tokens but more on the semantic insight it contributes to the overall reasoning process – insights that advance the CoT even if the precise wording varies or is imperfect (Fig. 2). Moreover, the self-reflection capabilities of LRMs allow them to correct (occasional) missteps from prior reasoning steps. **Taken together, these properties make the decoding of thinking tokens – the primary overhead in LRM inference – inherently more *approximation tolerant* than typical LLM serving in that a large fraction of intermediate reasoning steps are aligned with the capabilities of, and tolerant of potential errors introduced by, lightweight reasoning models**, as our results in Fig. 3 demonstrate.

Building on this observation, we propose **SpecReason** to accelerate LRM inference while preserving accuracy by carefully offloading these selected, easier intermediate reasoning steps to be *speculated* by a smaller model. SpecReason uses a smaller reasoning model to produce individual reasoning steps, and only uses the slower but more capable base model to efficiently assess the outputs of the small model (§4.1) and guide reasoning onto the appropriate trajectory (Fig. 1). We find, as prior works have with other tasks Song et al. [2025], that base models can be prompted to act as critic models and judge the utility of intermediate steps well (Fig. 6).

Speculative reasoning vs. speculative decoding. While SpecReason is conceptually similar to speculative decoding Leviathan et al. [2023], an approach that relies on a small draft model to speculate future tokens, there exist key differences. Most notably, speculative decoding is an *exact* optimization that targets token-level equivalence between the small model and the base model, i.e., focusing on typical LLM serving where all generated tokens are part of the final model output being assessed. In contrast, SpecReason relaxes this stringent requirement and specifically aims to leverage the approximation tolerance that reasoning affords for thinking tokens, i.e., outputs for intermediate reasoning steps. Indeed, we find that semantic-level similarity for thinking tokens (Fig. 2) is enough to preserve final-output accuracy (Fig. 3), which opens up the door for great latency savings in LRM inference. More importantly, because speculative reasoning and speculative decoding operate at different levels, we show that they are *complementary optimizations* (§4.2) and can be combined in a hierarchical speculation scheme to achieve even larger latency reductions.

We evaluate SpecReason on a wide range of reasoning workloads that cover common tasks of varying complexity aim [2025], Hendrycks et al. [2021], Rein et al. [2024]. Overall, we find that SpecReason reduces end-to-end latency by $1.5 - 2.5\times$ compared to vanilla LRM inference while improving accuracy by $1.0 - 9.9\%$. Moreover, it is can be combined with speculative decoding to provide a $19.4 - 44.2\%$ improvement over speculative decoding.

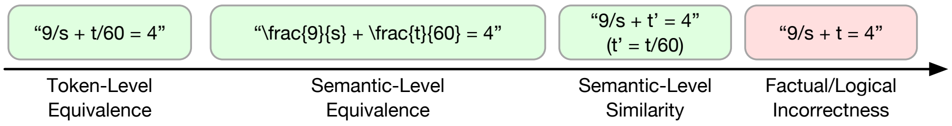

Figure 2: The spectrum of approximations of one example reasoning step (equation 1 in Fig. 1).

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 and 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 Brown et al. [2024] for scaling inference-time compute.

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] 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], 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.

3 Motivation

In this work, we argue that reasoning workloads run on LRMs exhibit unique opportunities for latency reduction thanks to their tolerance to approximations, which sets them apart from traditional generation tasks on LLMs. We illustrate these properties using a representative example from the AIME dataset that was selected for ease of presentation.

Intermediate steps are easier than end-to-end reasoning. One key observation is that the difficulty of reasoning tasks is not uniformly hard across the many reasoning steps within a long CoT. As seen in Fig. 1, while the overall task might be challenging for a small model, only a few key reasoning steps (e.g., analyzing the problem, breaking it down via formulations or case analyses, and coming up with a plan to reason about the problem step-by-step) are critical to the overall reasoning progress, whereas many other reasoning steps are comparatively easy. This behavior is by design: LRMs are trained with RL to generate CoTs that decompose hard problems into a sequence of simpler, more tractable reasoning steps. These intermediate steps often include straightforward reasoning, e.g., arithmetic calculations, case analyses, or logical deductions that are much easier to decode compared to synthesizing the full solution directly in LLM inference. This heterogeneity in difficulty and importance allows a lightweight model to efficiently perform these tasks with sufficient accuracy.

Reasoning progress depends on insights, not exact tokens. Another key takeaway in our work is that the utility of a reasoning step lies not in the exact tokens, but more in the insight it conveys that advances overall reasoning progress. 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 shown in Fig. 2, there exists a spectrum of valid wordings for a given step: Many semantically-equivalent or semantically-similar sequences of tokens can serve the same functionality in guiding subsequent reasoning.

Occasional mistakes can be corrected via self-reflection. Furthermore, LRMs exhibit strong self-reflection capabilities. Even if an earlier step contains a factual or logical error, LRMs often correct themselves in subsequent reasoning steps, oftentimes marked by outputting tokens like “Wait” or “Hmm”. Moreover, unlike LLM inference where all output tokens are evaluated for accuracy, only the tokens after the thinking tokens count toward accuracy in LRM inference. Therefore, LRM inference can tolerate occasional mistakes, as the models can often identify these mistakes and steer reasoning back onto the correct path during self-reflection. This fault-tolerant property further suggests the potential benefits of approximations.

In summary, compared to traditional LLM inference, LRM inference is inherently more tolerant of approximations that don’t require token-level equivalence as long as the overall reasoning trajectory is maintained. This property is not limited to a single, linear CoT; rather, it extends to more general forms of inference-time compute scaling paradigms, e.g., tree-based searches.

4 Method

4.1 Speculative Reasoning

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. SpecReason reduces decoding latency by offloading easier or less critical steps – defined as semantically self-contained units like full reasoning steps or sentences – to a smaller, faster model. In decoding each reasoning step, SpecReason proceeds in two stages: (1) the lightweight speculator proposes the next step based on the prior trajectory, and (2) the base model assesses its utility. If accepted, SpecReason moves on to the next step; otherwise, SpecReason falls back to the base model to regenerate the step. While our implementation uses a simple static-threshold verification scheme, the framework supports richer customizations of strategies. We outline key design principles below.

Efficient verification. Because each step requires the base model’s verification, it’s crucial for the verification overhead to remain 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. One simple implementation is prompting the base model to generate a single token “score” (e.g., 0-9) that assesses the utility of the speculated step. The assessment 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. A static acceptance threshold (e.g., score $\geq$ 7) determines whether to accept the speculative step. This threshold can be adjusted to control the aggressiveness of speculation (Fig. 4). If the score falls below the threshold, we discard the speculative step and regenerate it using the base model. We emphasize that this rudimentary static-threshold scheme represents a lower bound on verification quality: more sophisticated alternatives – such as logprobs-based confidence estimates or dynamic thresholds – can be incorporated into SpecReason with no additional cost, and may further improve performance. The key requirement is that verification remains lightweight and step-level.

Implementation details. Due to the lightweight nature of the small model, we colocate the small model and the base model on the same GPU. The memory reserved for KV caches 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. When speculative steps from the small model are rejected, we simply discard the corresponding KV cache entries. Inference is performed sequentially: the small model and base model take turns, avoiding compute kernel 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. 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

5.1 Setup

Models. In our main results, we use QwQ-32B qwq [2025] as the base model and a distilled version of DeepSeek-R1, R1-1.5B Guo et al. [2025], as the small model – both of which are fine-tuned on Qwen-2.5 Yang et al. [2024] and embed the capability of reasoning with long CoTs. 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 §5.4.

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 model and the base model as the latency baseline and accuracy baseline, respectively. Aside from SpecReason, we also run speculative decoding ("SpecDecode") with R1-1.5B 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.

5.2 Main Results

We compare SpecReason against baseline methods in Fig. 3. Across the three datasets—MATH, AIME, and GPQA—SpecReason achieves a 2.5×, 1.9×, and 1.5× reduction in latency, respectively, compared to vanilla inference with the base model. Alongside these gains, it also yields modest accuracy improvements of 1.0%, 3.1%, and 9.9% compared to the base model. We 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.

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 44.2%, 33.8%, and 19.4% over speculative decoding alone. The most significant performance gains for SpecReason occur on the MATH dataset, where both models achieve relatively high accuracies and the capability gap between the small and

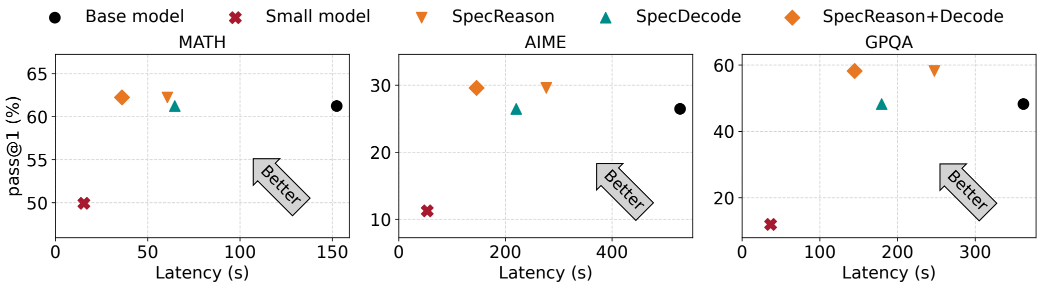

Figure 3: Comparison of the accuracy and latency of different schemes. SpecReason significantly reduces latency while improving accuracy over vanilla inference, and is also complementary with speculative decoding.

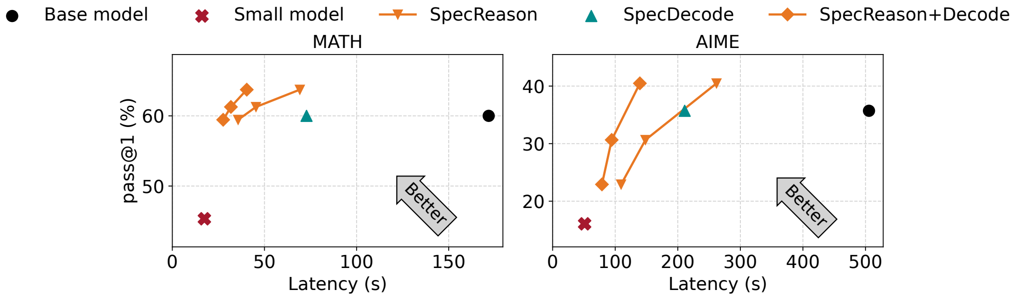

Figure 4: SpecReason allows trading off latency for accuracy via adjusting the acceptance threshold (from left to right, the thresholds are: 3, 5, and 7 out of 9).

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.

Finally, when comparing SpecReason+Decode with SpecReason, SpecReason+Decode reduces latency by 1.7×, 1.9×, and 1.7×, 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 80.0%, 40.8%, and 44.4%, respectively.

5.3 Controlling the Accuracy-Latency Tradeoff

In Fig. 4, we illustrate how SpecReason enables flexible control over the accuracy-latency tradeoff, using a representative, randomly selected subdataset from the full datasets in §5.4 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, and 7, 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 SpecReason 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 subdataset: as the acceptance threshold increases from 3 to 7, latency grows from 109.4s to 261.9s, and accuracy improves from 22.9% to 40.5%. 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, making aggressive acceptance of its speculative steps more costly in terms of accuracy.

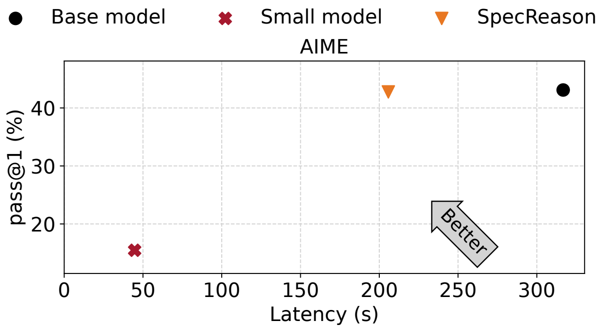

Figure 5: SpecReason's results on the model combination (R1-70B, R1-1.5B).

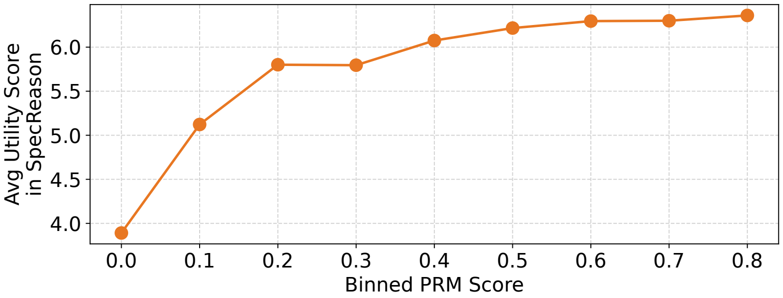

Figure 6: The utility scores in SpecReason closely reflects the quality score judgements from a process reward model. x on the x-axis denotes PRM scores in the range $[x, x + 0.1)$.

5.4 Base Models of Varying Sizes and Architectures

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× 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×) 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× 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× 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).

5.5 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. 6, 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 Related Work

There has been some early work aimed at reducing the inference latency of LRMs. Sky-T1-Flash Team [2025] addresses the problem of overthinking by fine-tuning models to curb unnecessary reasoning, thereby reducing the length of reasoning chains and, consequently, latency. While this approach reduces computation by curtailing reasoning depth, it does not fully leverage the potential of doing approximations on easier intermediate reasoning steps for efficiency.

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. This early-exit strategy helps mitigate latency and is complementary to SpecReason, as both aim to reduce unnecessary reasoning steps.

7 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 1.5–2.5× speedup over traditional LRM inference, with accuracy improvements ranging from 1.0–9.9%. Additionally, when combined with speculative decoding, SpecReason further reduces latency by 19.4–44.2%, 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. 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.

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

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.

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.

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.