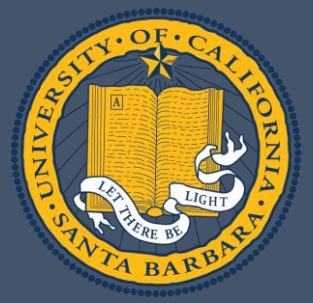
MLSys 2026

SkipKV: Selective Skipping of KV Generation and Storage for Efficient Inference with Large Reasoning Models

Jiayi Tian [1,2] Seyedarmin Azizi[3] Yequan Zhao[1] Erfan Baghaei Potraghloo[3] Sean McPherson[2]
Sharath Nittur Sridhar[2] Zhengyang Wang[1] Zheng Zhang[1] Massoud Pedram[3] Souvik Kundu[2]

[1] University of California, Santa Barbara *[2] Intel Labs USA* *[3] University of Southern California*



Ninth Annual Conference on Machine Learning and Systems (MLSys 2026), May 20, 2026, Bellevue, WA

Reasoning models are bottlenecked by their KV cache

- Large Reasoning Models (DeepSeek-R1, etc.) generate very long CoT traces.
- Token count grows linearly → KV cache memory dominates inference cost.
- Example: **R1-Distill-Llama-8B** can generate **32K+ tokens** on one complex math problem; at **batch size 10,** KV cache becomes **~2.5× larger than the model weights.**

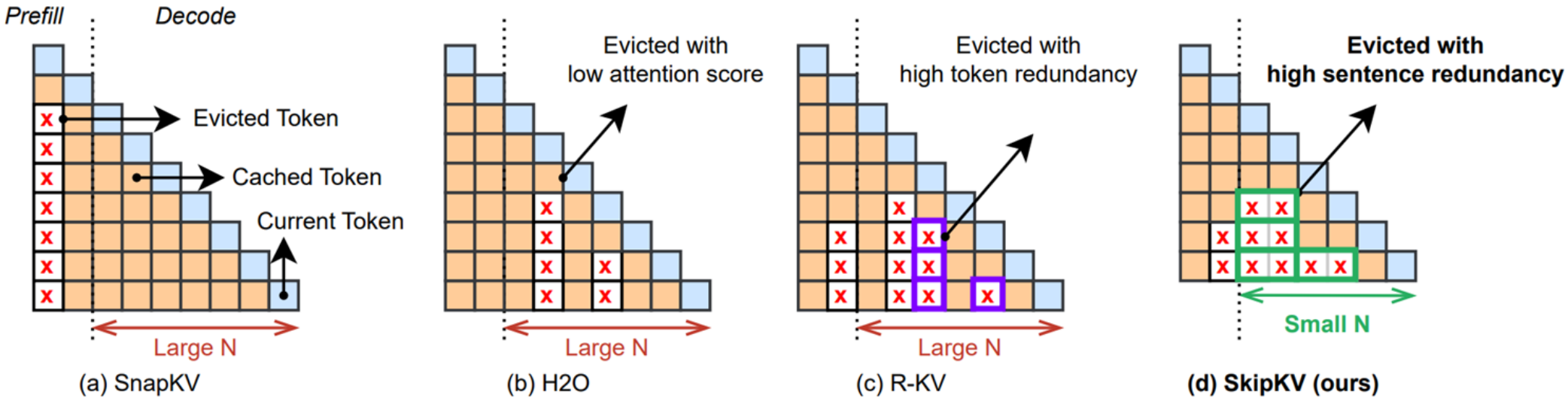

Reasoning models are bottlenecked by their KV cache

- Large Reasoning Models (DeepSeek-R1, etc.) generate very long CoT traces.
- Token count grows linearly → KV cache memory dominates inference cost.
- Example: **R1-Distill-Llama-8B** can generate **32K+ tokens** on one complex math problem; at **batch size 10,** KV cache becomes **~2.5× larger than the model weights.**

KV cache compression is essential for deploying reasoning models at scale — both for memory headroom and decoding throughput.


Failure of Token-level Eviction Strategies

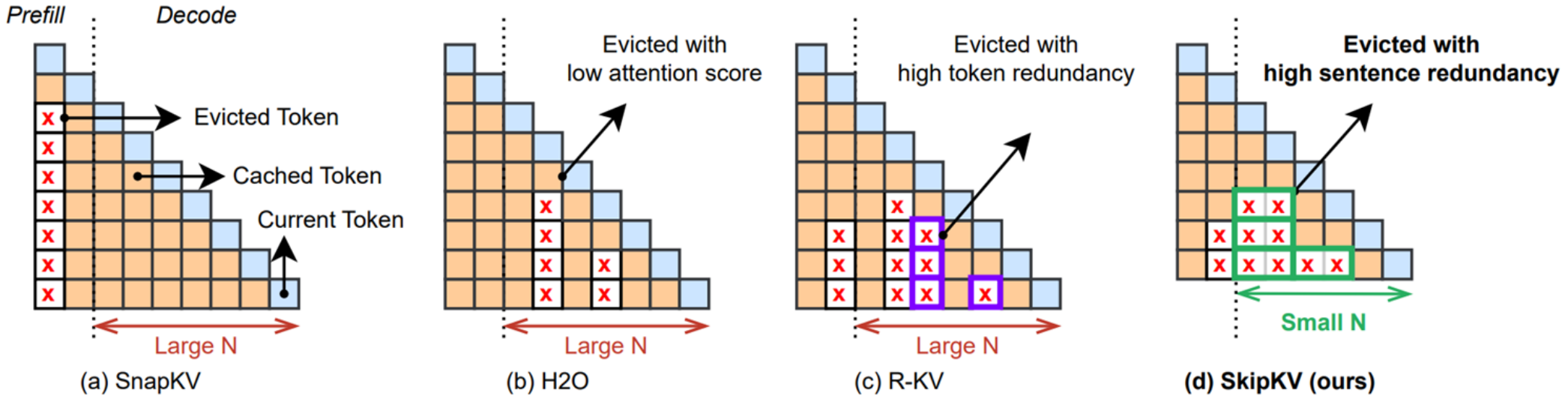
- **SnapKV:** prefill-only eviction → fail on long CoT generations
- **H2O:** attention-magnitude eviction → blind to semantic structure
- **R-KV:** token-level redundancy eviction → fragment semantic units



Failure of Token-level Eviction Strategies

MLSys

- **SnapKV:** prefill-only eviction → fail on long CoT generations
- **H2O:** attention-magnitude eviction → blind to semantic structure
- **R-KV:** token-level redundancy eviction → fragment semantic units

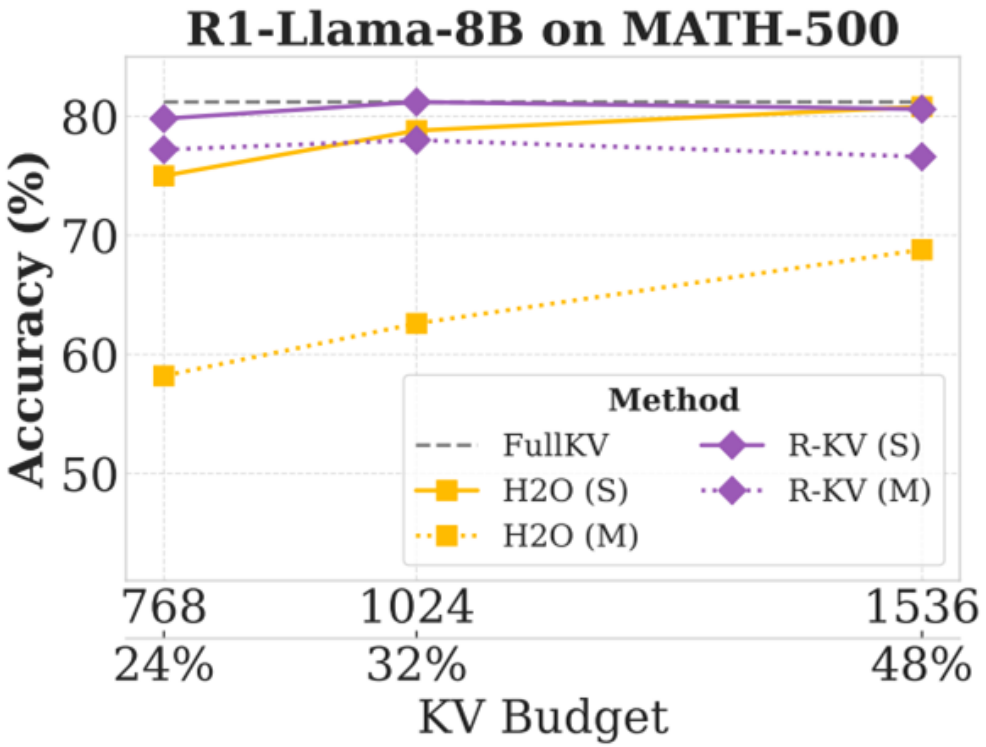


SkipKV: Consider sentences redundancy and property → keep coherent semantic units and generate concise outputs

Failure of Token-level Eviction Strategies

MLSys

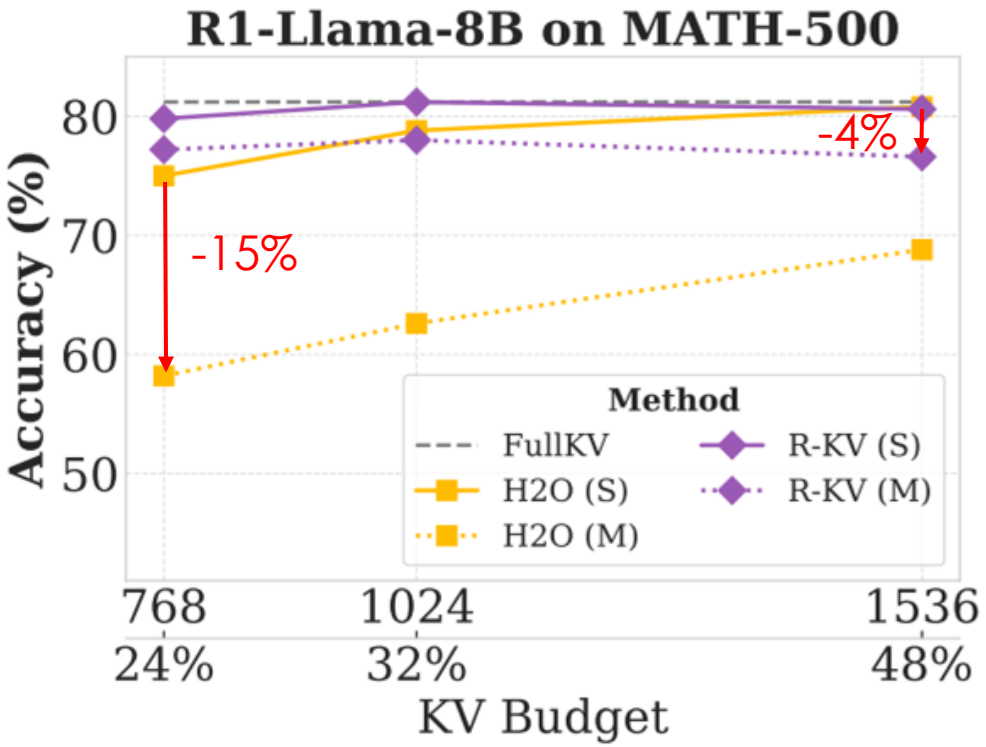
- ***Observation 1:*** *Accuracy drop on multi-batch decoding*



Failure of Token-level Eviction Strategies

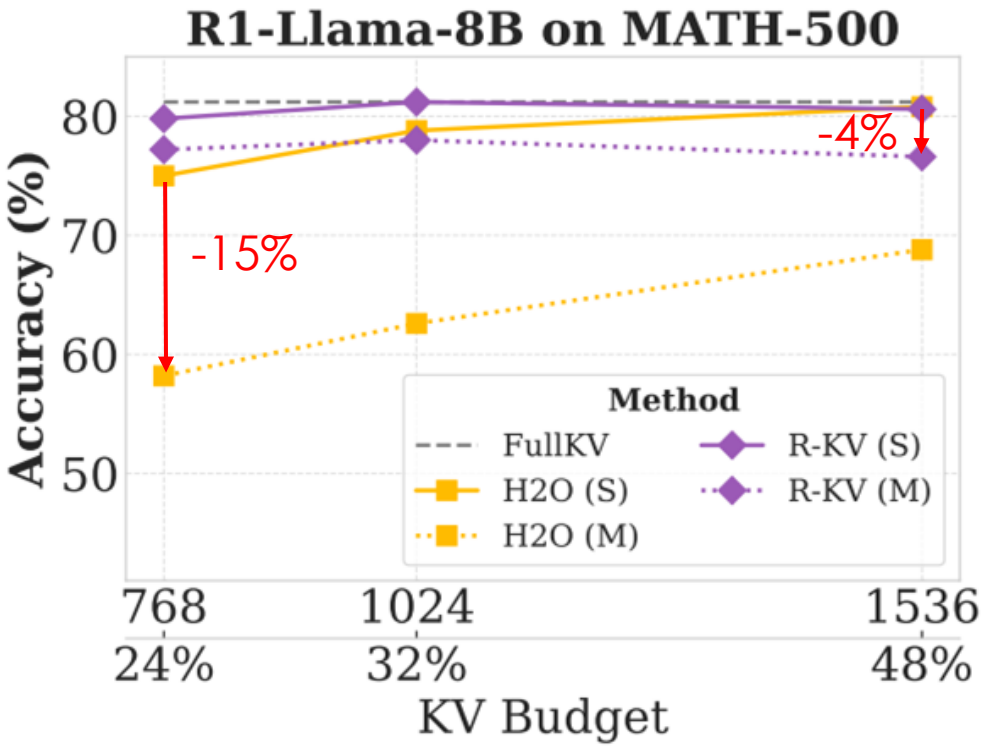
MLSys

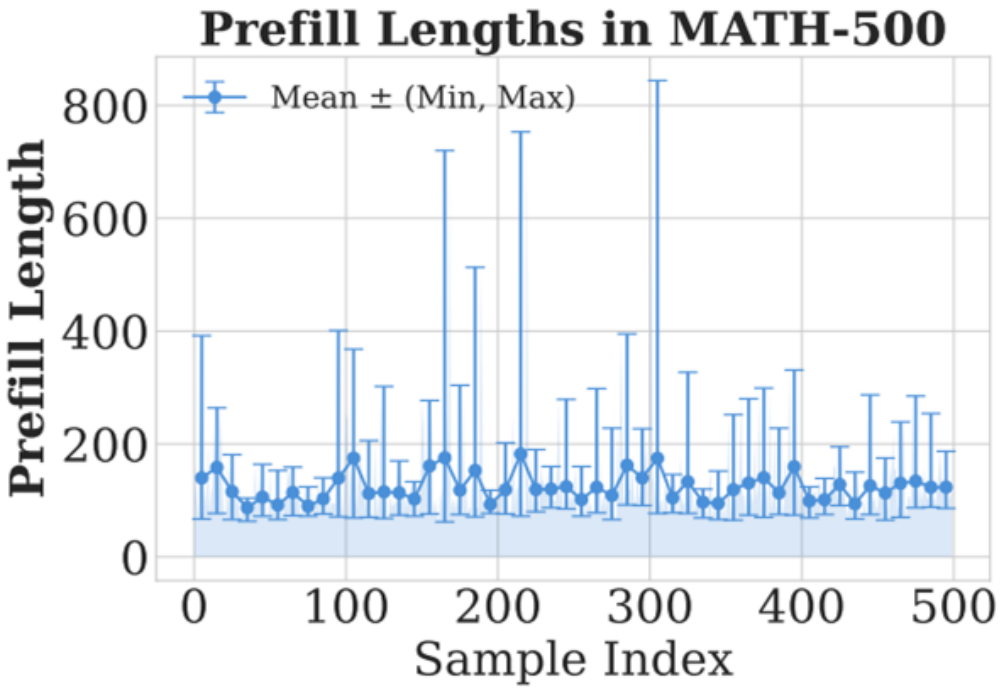
- ***Observation 1:*** *Accuracy drop on multi-batch decoding*



Failure of Token-level Eviction Strategies

- ***Observation 1:*** *Accuracy drop on multi-batch decoding*
- **Explanation 1:** Padding tokens reduce the effective KV budget

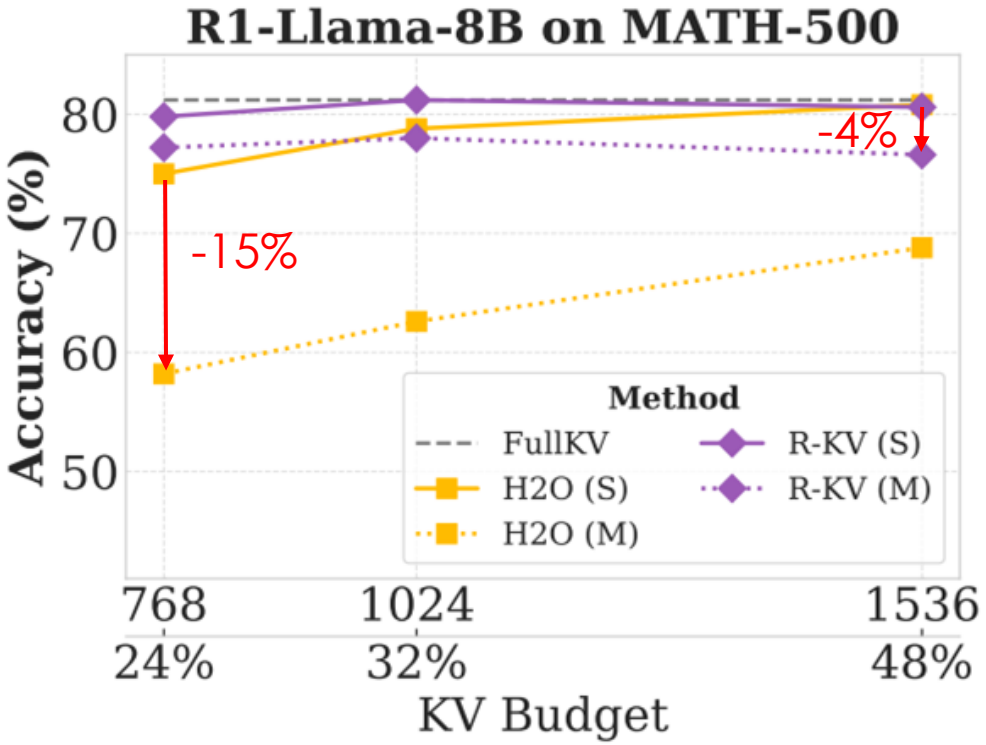


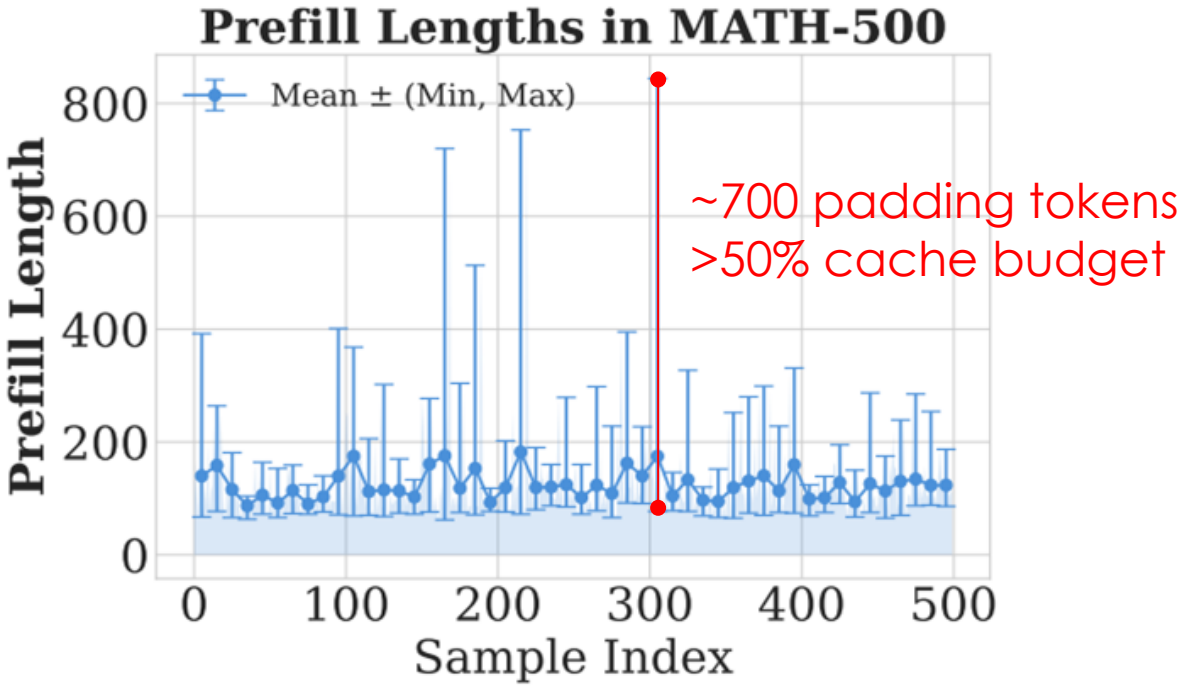


Failure of Token-level Eviction Strategies

MLSys

- ***Observation 1:*** *Accuracy drop on multi-batch decoding*
- **Explanation 1:** Padding tokens reduce the effective KV budget

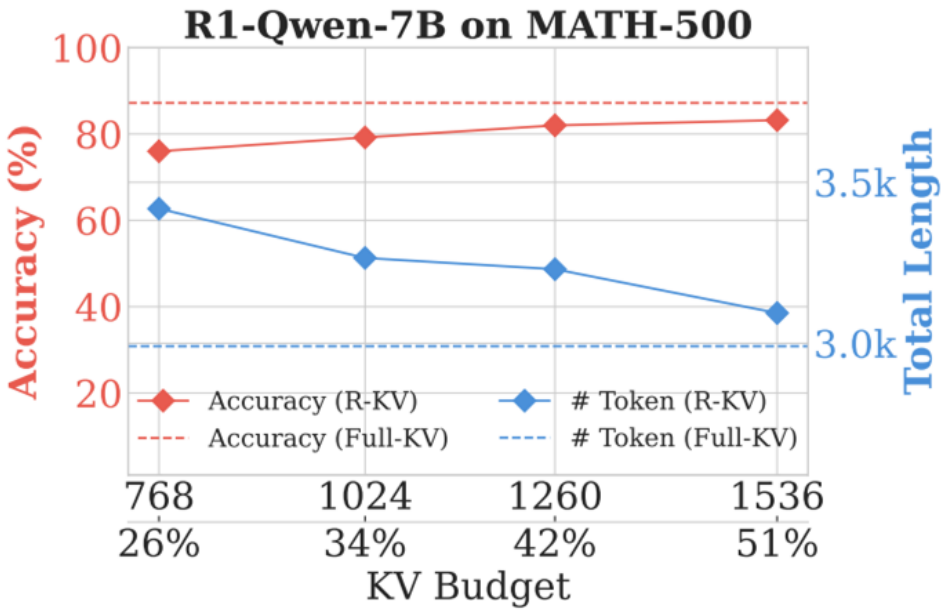




Failure of Token-level Eviction Strategies

MLSys

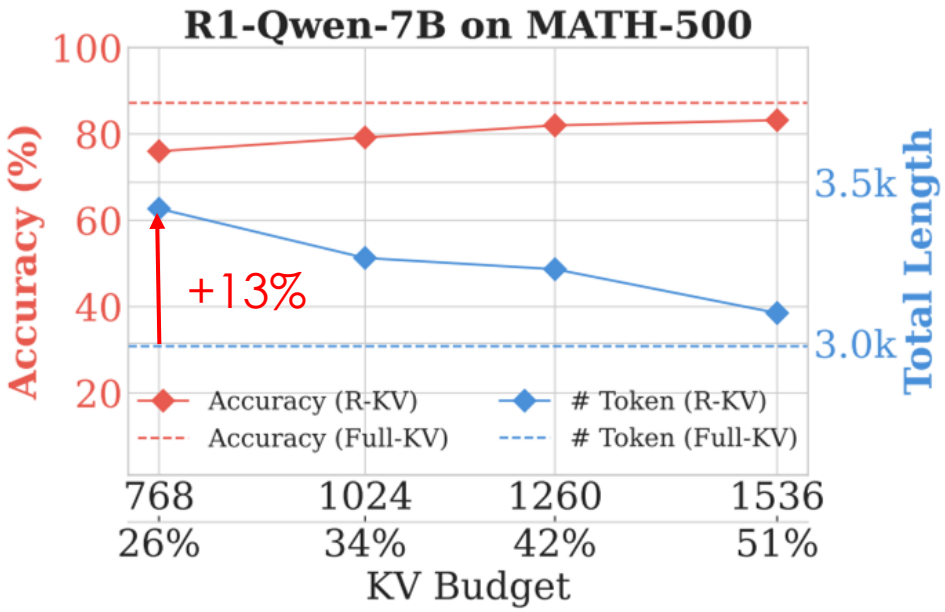
- ***Observation 2:*** *Generation token length increase*



Failure of Token-level Eviction Strategies

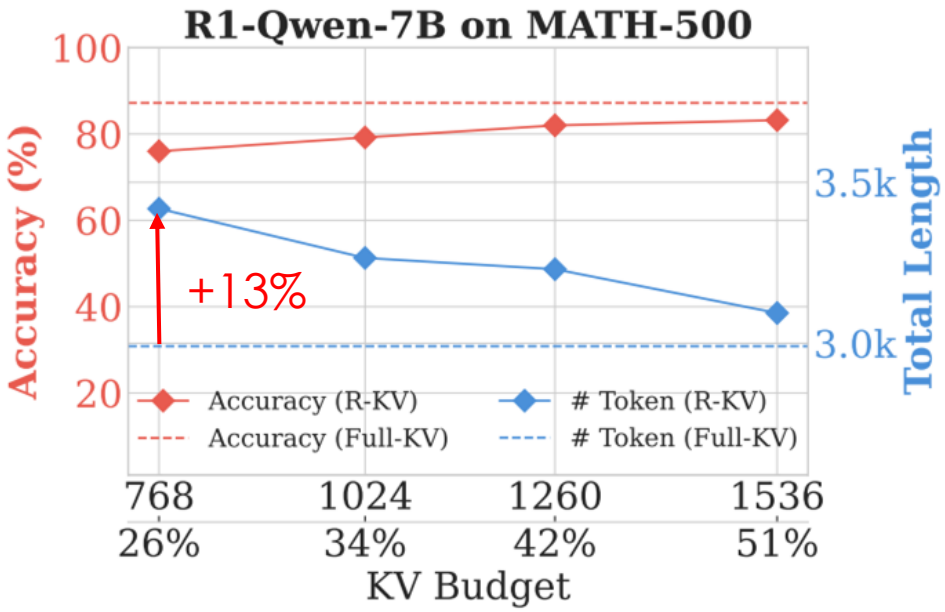
MLSys

- ***Observation 2:*** *Generation token length increase*



Failure of Token-level Eviction Strategies

- ***Observation 2:*** *Generation token length increase*
- **Explanation 2:** Evicting similar tokens disrupts semantic units in computation steps, causing repeated self-validation.



<|User|>Answer the following questions. You should think step-by-step and put your final answer within \boxed{}.
Question: Evaluate $(1+2i)6-3i$.<|Assistant|>
[Re-validate for 8 times and generate 1517 tokens in total]
... So, the result should be 6 + 9i.
Wait, let me double-check... 6 + 9i. That seems correct.

Alternatively,... Let me write it out again:
... Then, subtracting 3i: 6 + 12i - 3i = 6 + (12i - 3i) = 6 + 9i.

Just to be thorough, maybe I can represent this in another way. ...
Alternatively, I can think of it as 6*(1 + 2i) - 3i.
Another way to approach this is ...
So, 6 + 12i - 3i = 6 + (12 - 3)i = 6 + 9i. Yep, that's consistent.
I think I'm confident that the result is 6 + 9i. Let me...6 + 9i. That seems reasonable.

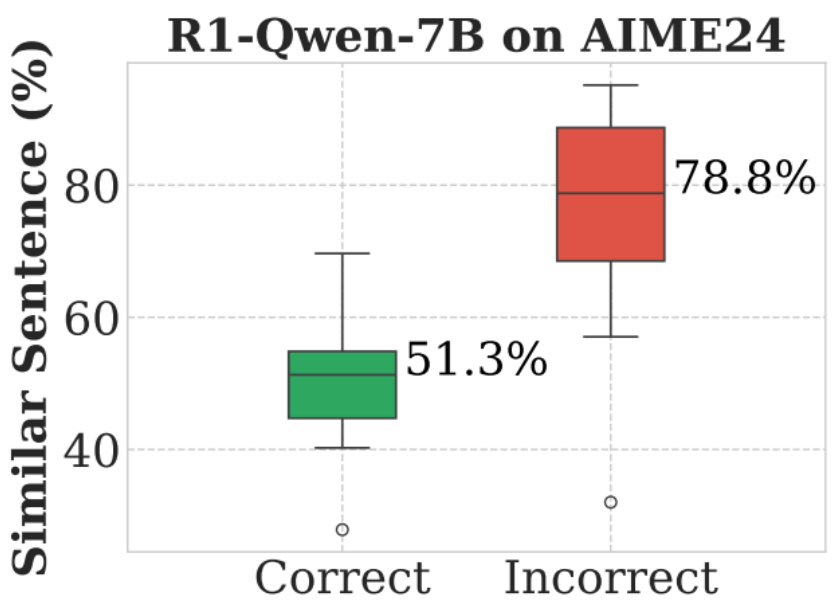
Alternatively, ... (6, 9). So, that's another way to see it.
Just to recap:
... - 6 + 12i - 3i = 6 + (12i - 3i) = 6 + 9i

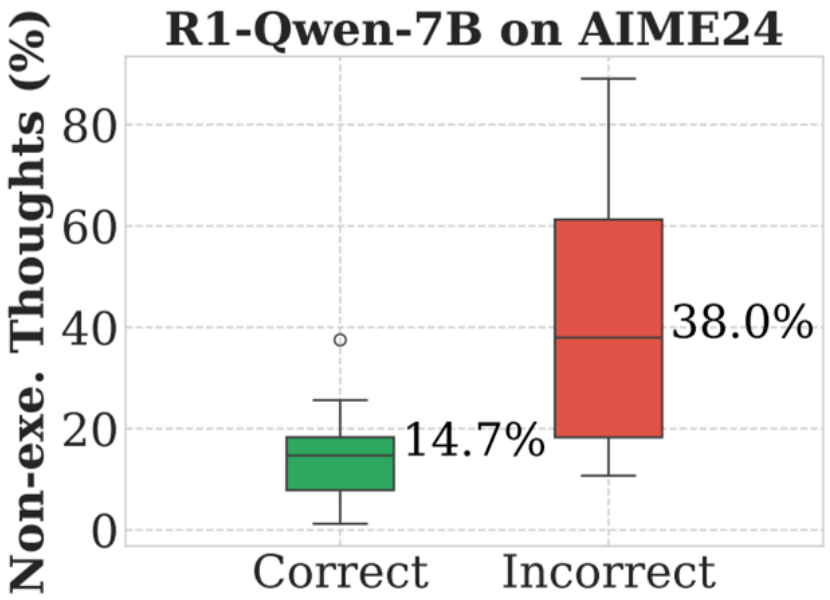
Just to make sure, let me compute it again step by step:
... - 6 + 12i - 3i = 6 + (12i - 3i) = 6 + 9i.

Yep, same result. I think that's solid.
I guess another way ...
So, after all that, I'm pretty confident that the answer is 6 + 9i.
Final Answer
The result of evaluating $(1+2i)6 - 3i$ is \boxed{6 + 9i}.

Motivation of Sentence-level Eviction

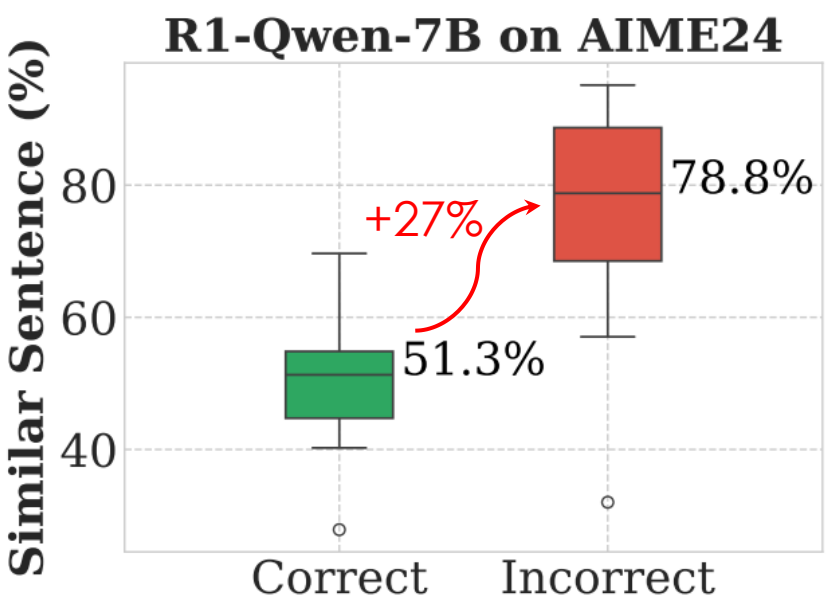
- ***Observation 3:*** *Wrong reasoning traces have more similar and non-execution sentences*

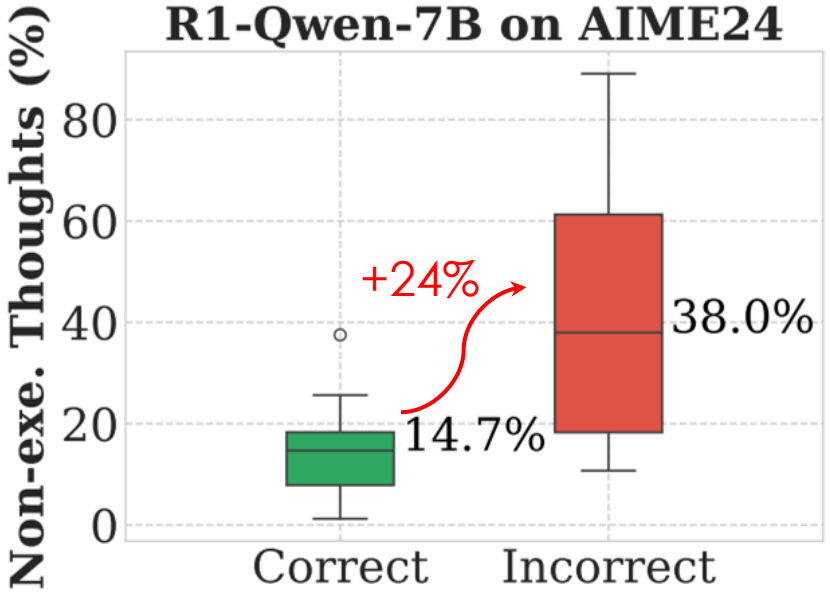




Motivation of Sentence-level Eviction

- ***Observation 3:*** *Wrong reasoning traces have more similar and non-execution sentences*



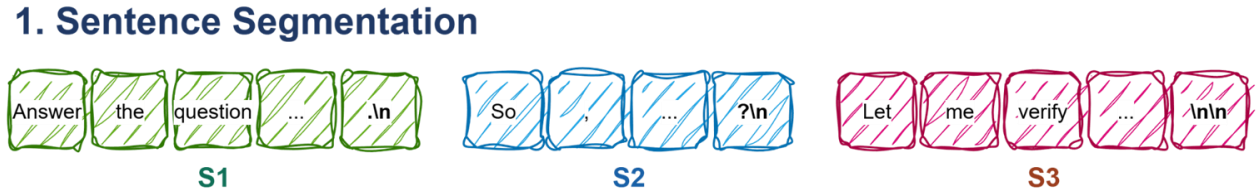


SkipKV: Skip KV Storage

- **Skipping KV storage** via sentence-primary KV cache eviction

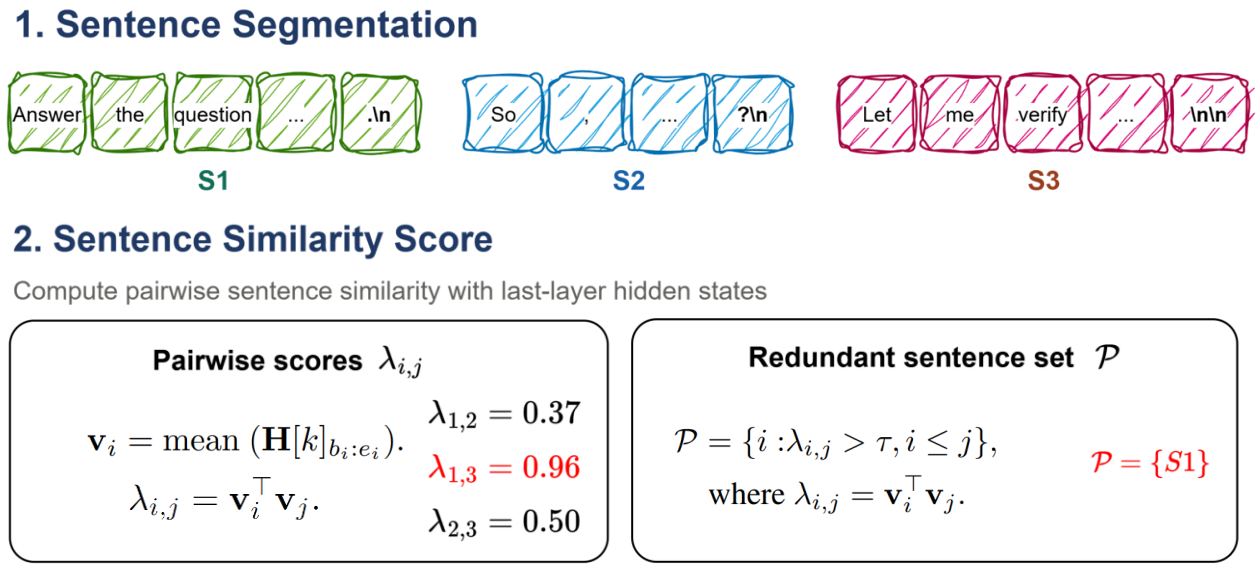
SkipKV: Skip KV Storage

- **Skipping KV storage** via sentence-primary KV cache eviction



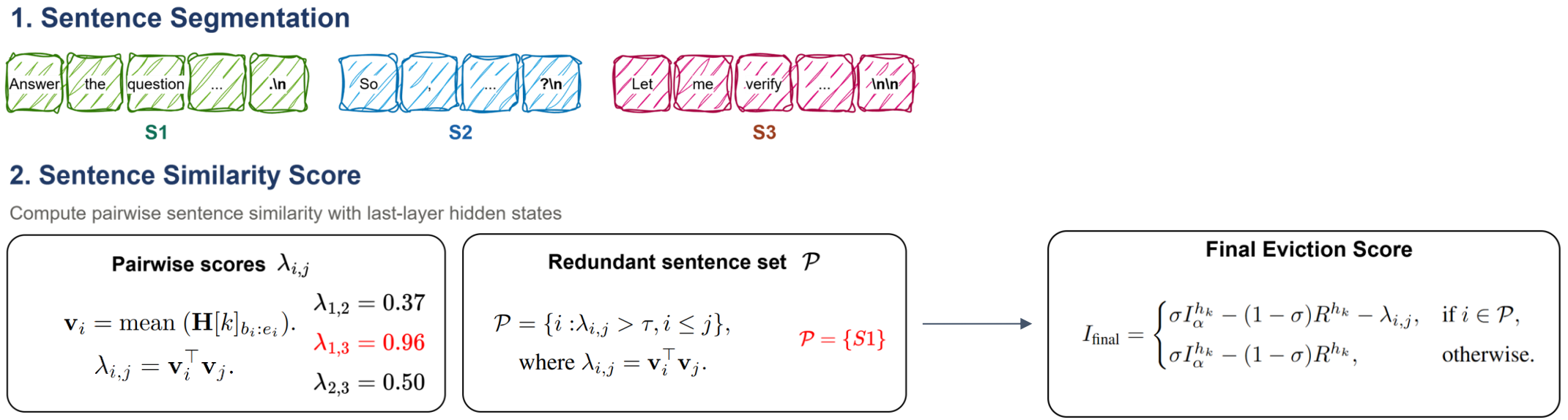
SkipKV: Skip KV Storage

- **Skipping KV storage** via sentence-primary KV cache eviction



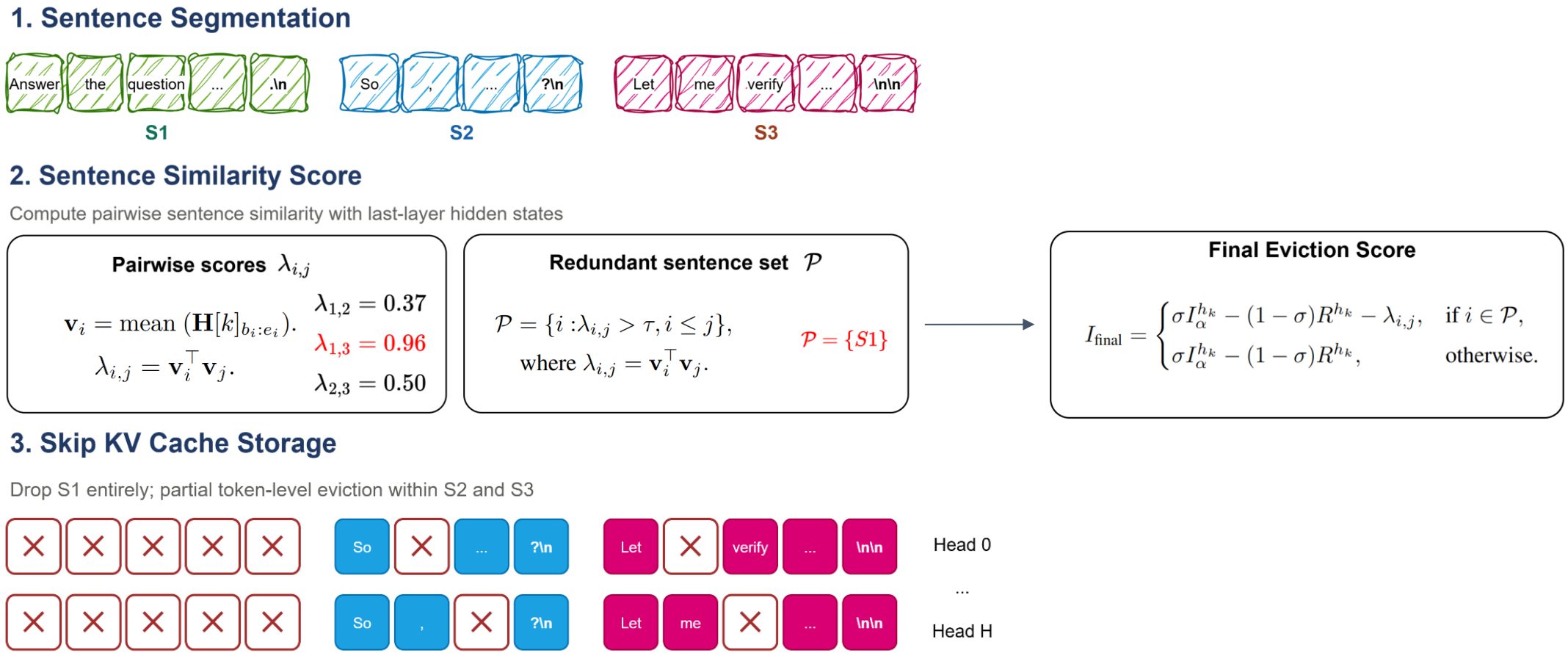
SkipKV: Skip KV Storage

- **Skipping KV storage** via sentence-primary KV cache eviction



SkipKV: Skip KV Storage

- **Skipping KV storage** via sentence-primary KV cache eviction

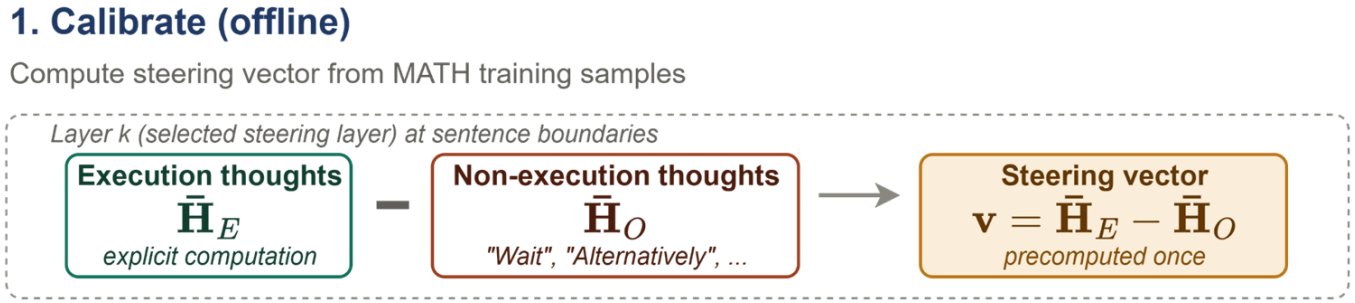


SkipKV: Skip KV Generation

- **Skipping KV generation** via adaptively steering the sentence property in the hidden space

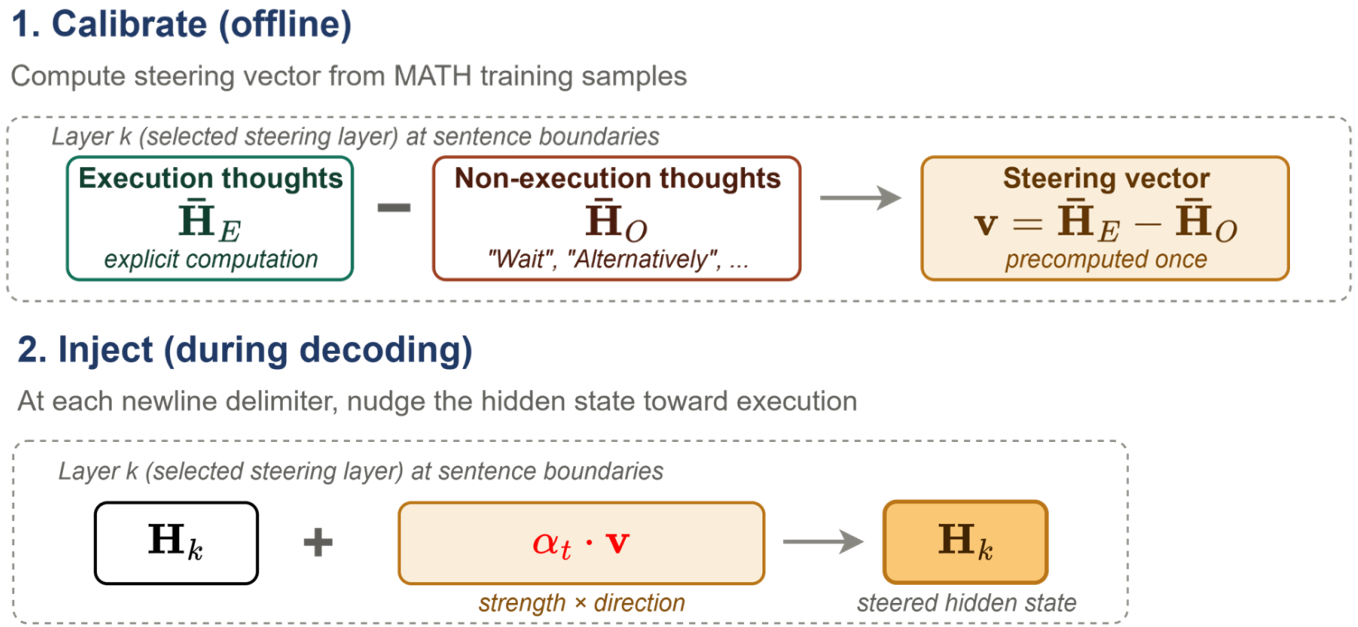
SkipKV: Skip KV Generation

- **Skipping KV generation** via adaptively steering the sentence property in the hidden space



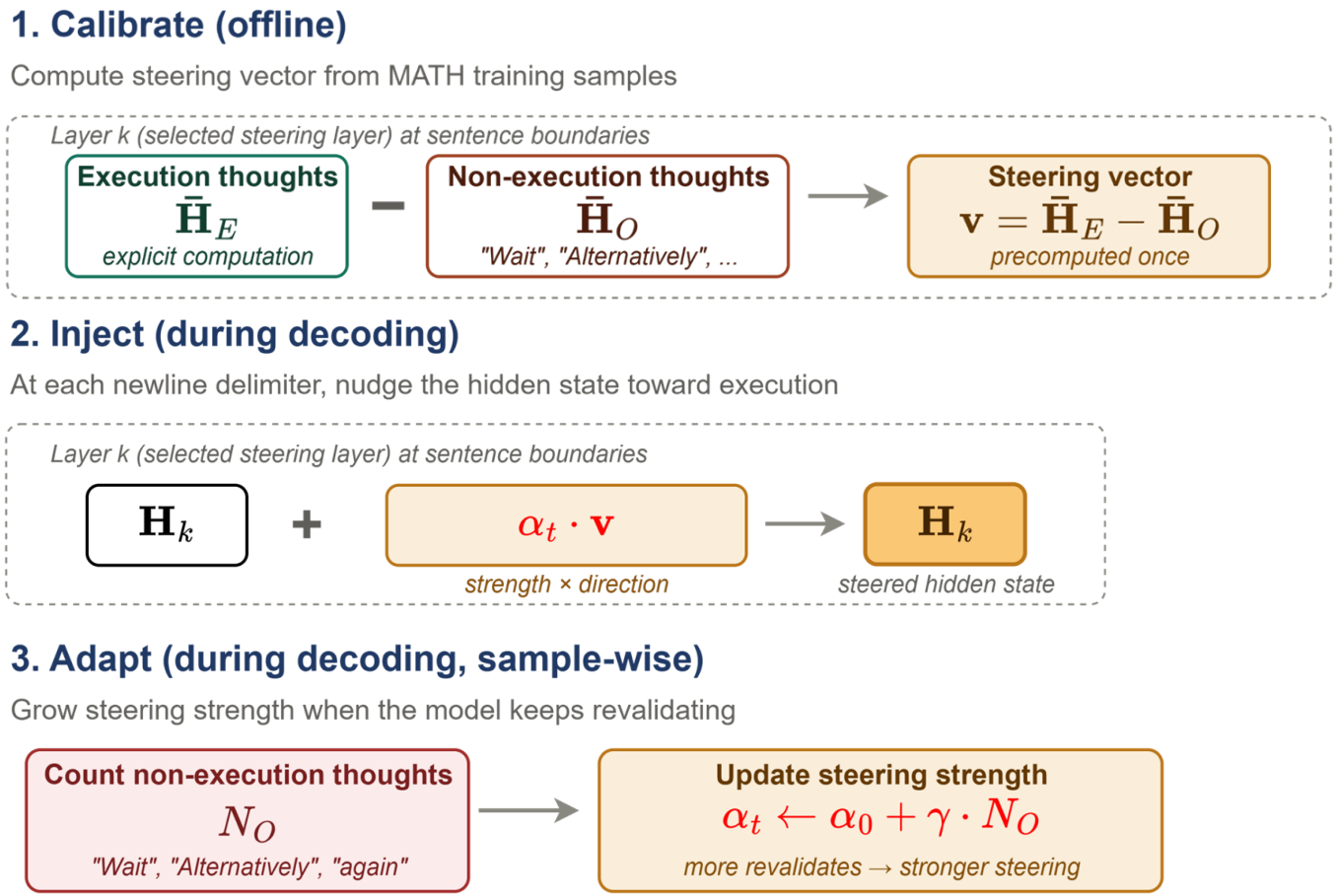
SkipKV: Skip KV Generation

- **Skipping KV generation** via adaptively steering the sentence property in the hidden space



SkipKV: Skip KV Generation

- **Skipping KV generation** via adaptively steering the sentence property in the hidden space

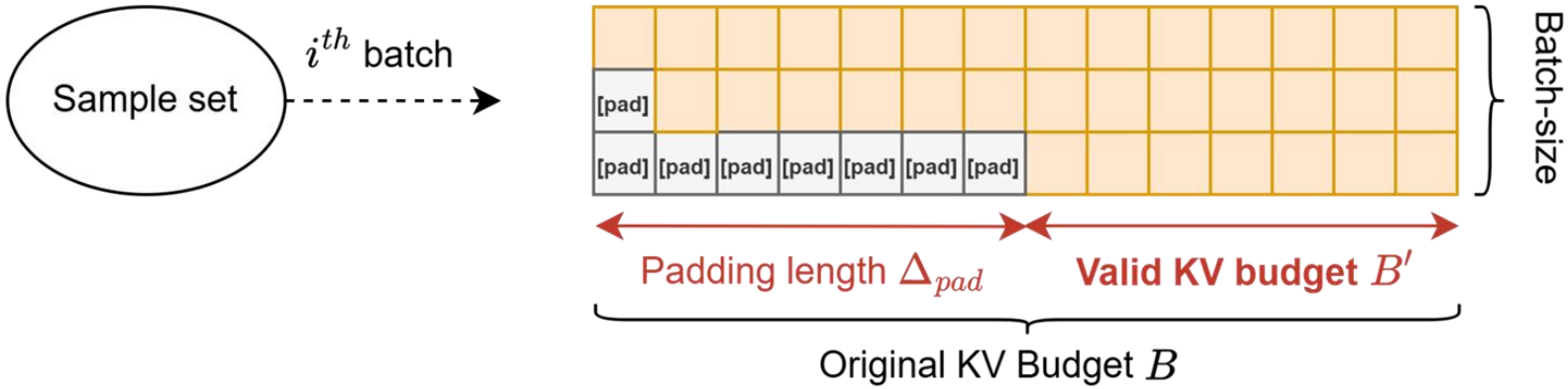


SkipKV: Robust Multi-batch Serving

- **Recover the effective KV cache budget** via reordering samples by prefilling length

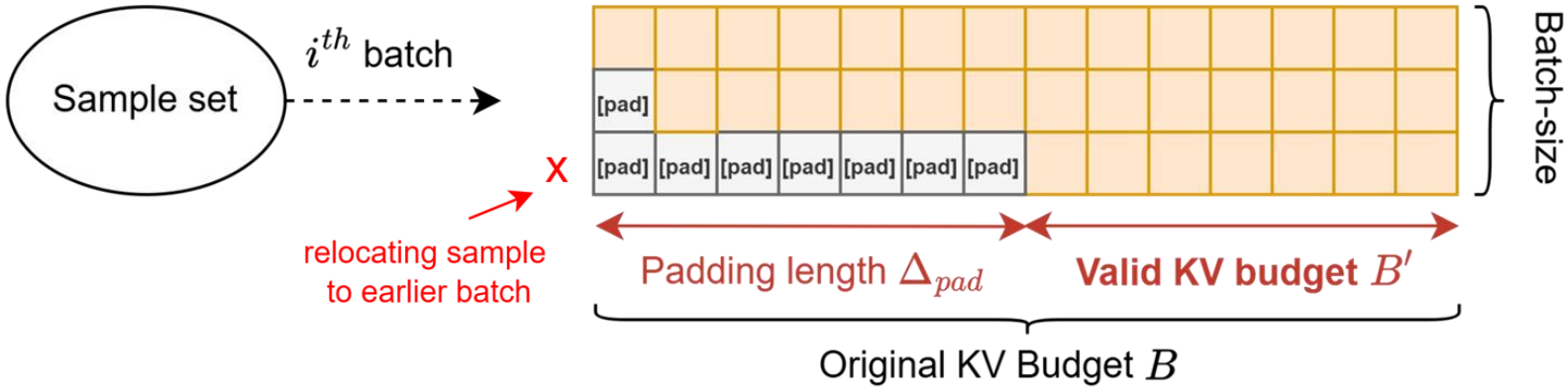

SkipKV: Robust Multi-batch Serving

- **Recover the effective KV cache budget** via reordering samples by prefilling length



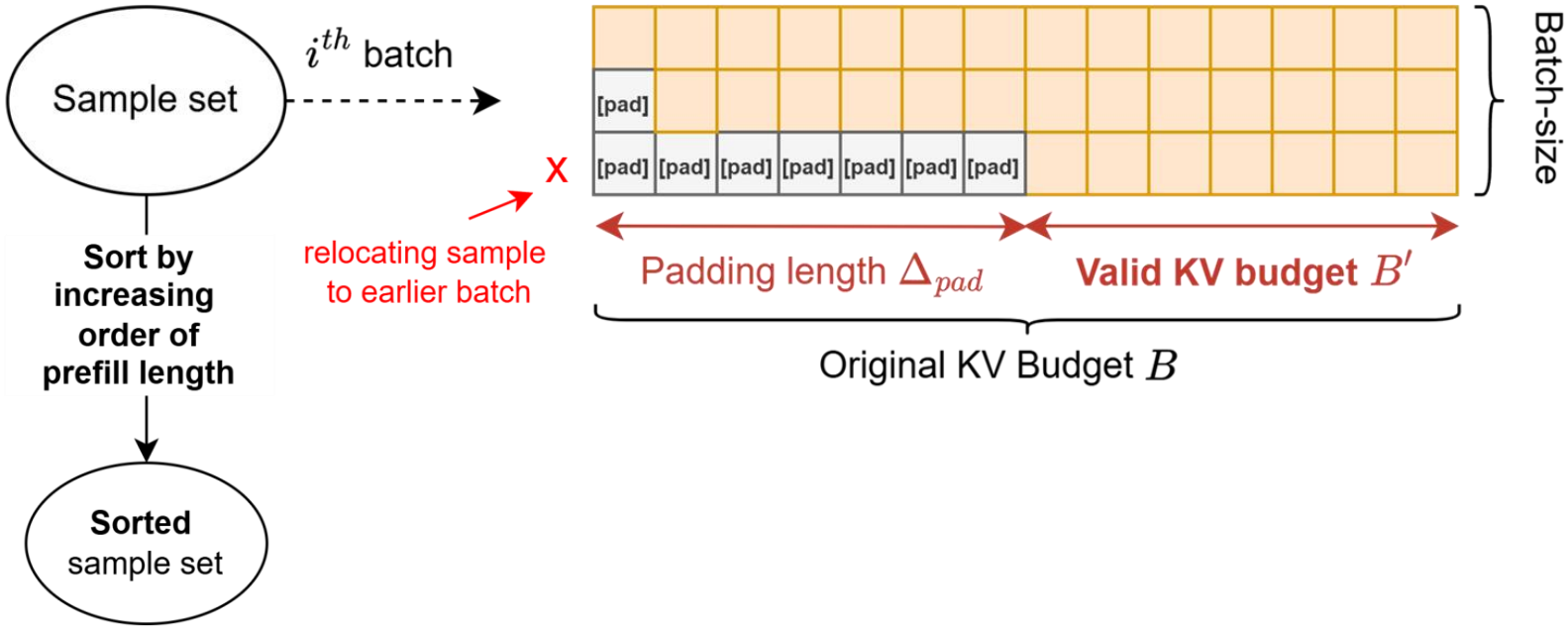
SkipKV: Robust Multi-batch Serving

- **Recover the effective KV cache budget** via reordering samples by prefilling length



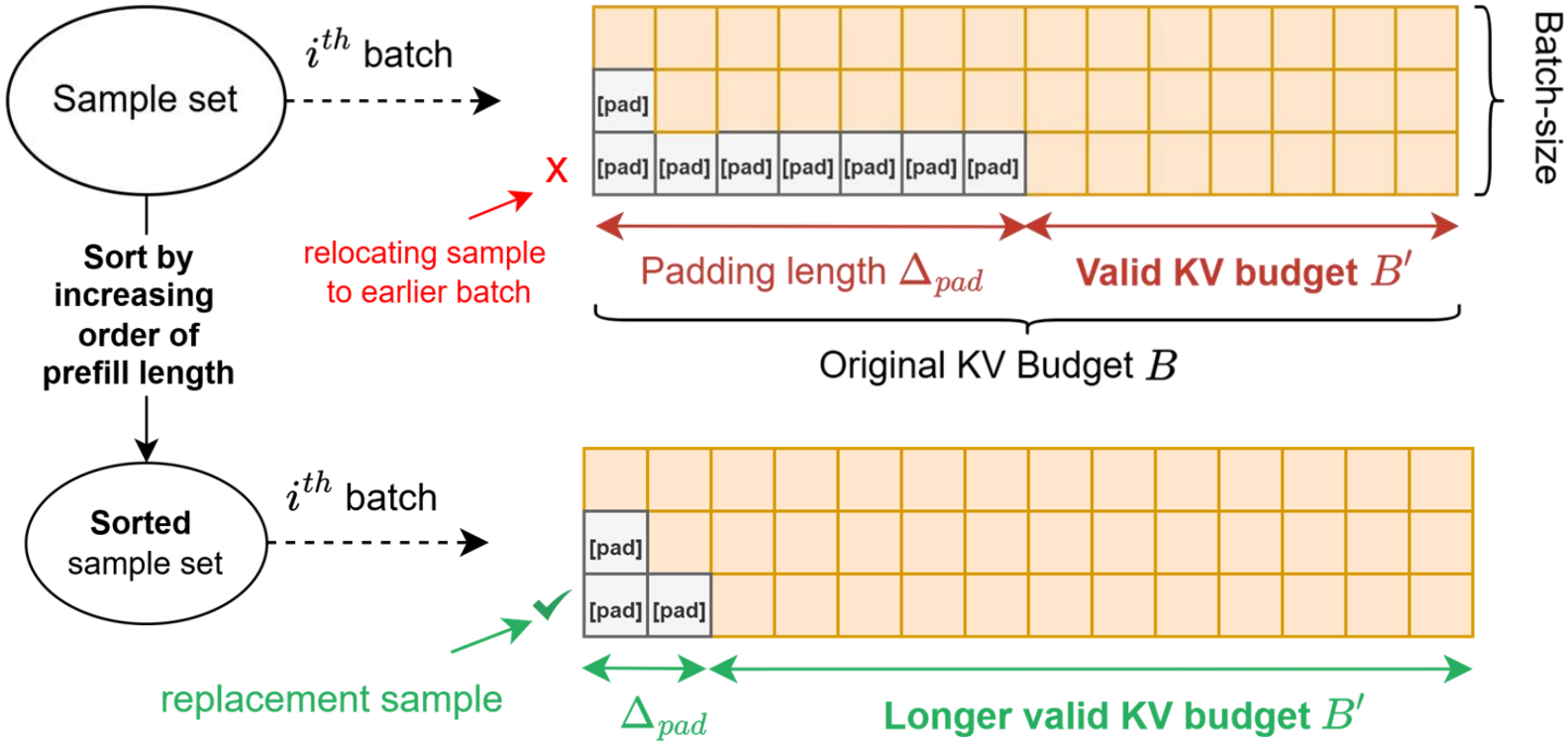
SkipKV: Robust Multi-batch Serving

- **Recover the effective KV cache budget** via reordering samples by prefilling length



SkipKV: Robust Multi-batch Serving

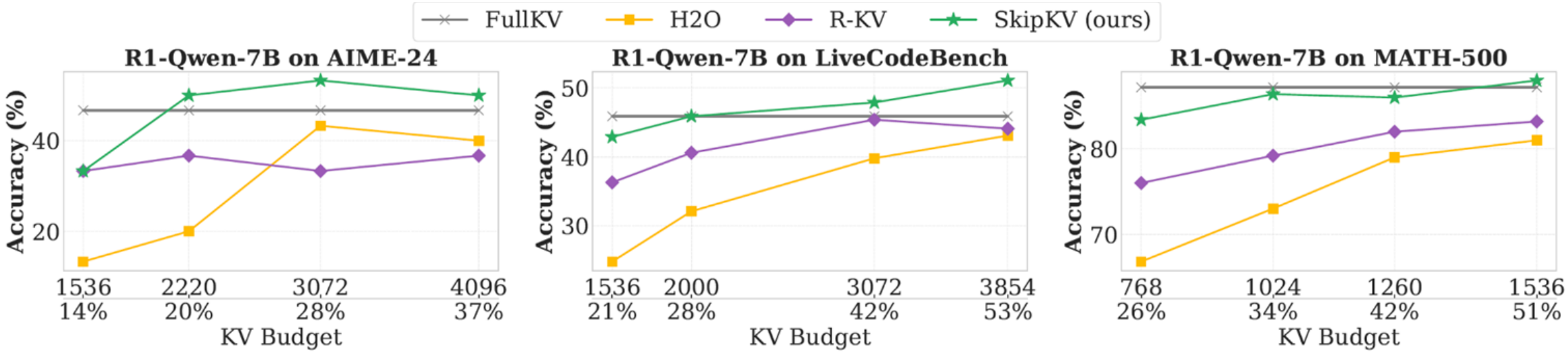
- **Recover the effective KV cache budget** via reordering samples by prefilling length



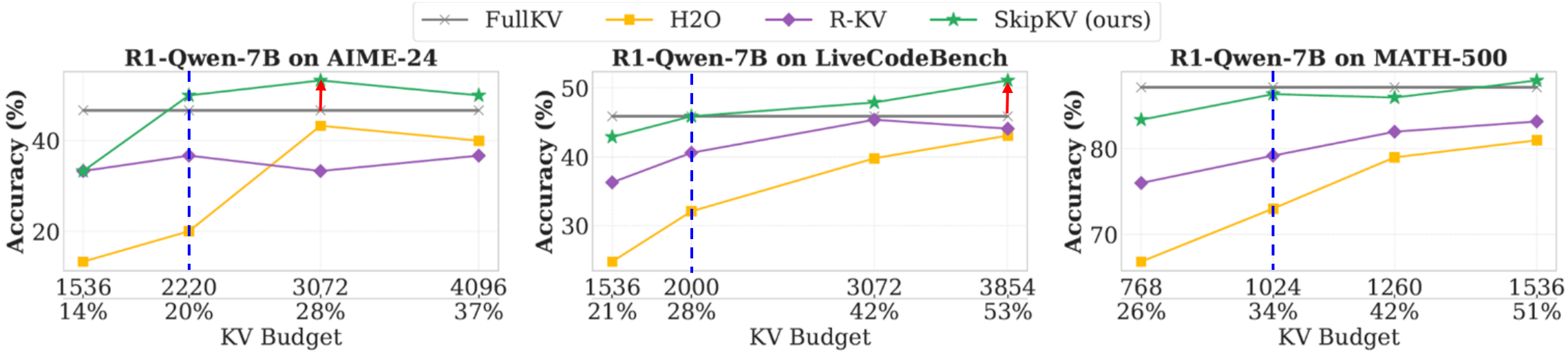
Experimental Setup

- **Setup:** 1 x A100 40GB GPU, Batch-size=10
- **Benchmarks:** GSM8K, MATH500, AIME-24, LiveCodeBench
- **Models:** DeepSeek-R1-Distill-(Qwen-7B, Qwen-14B, Llama-8B)
- **Baselines:** H2O, R-KV, SEAL
- **Metrics:** accuracy, cache memory (GB), generation token length, throughput (samples/min)

Experimental Results: Accuracy vs KV Budget



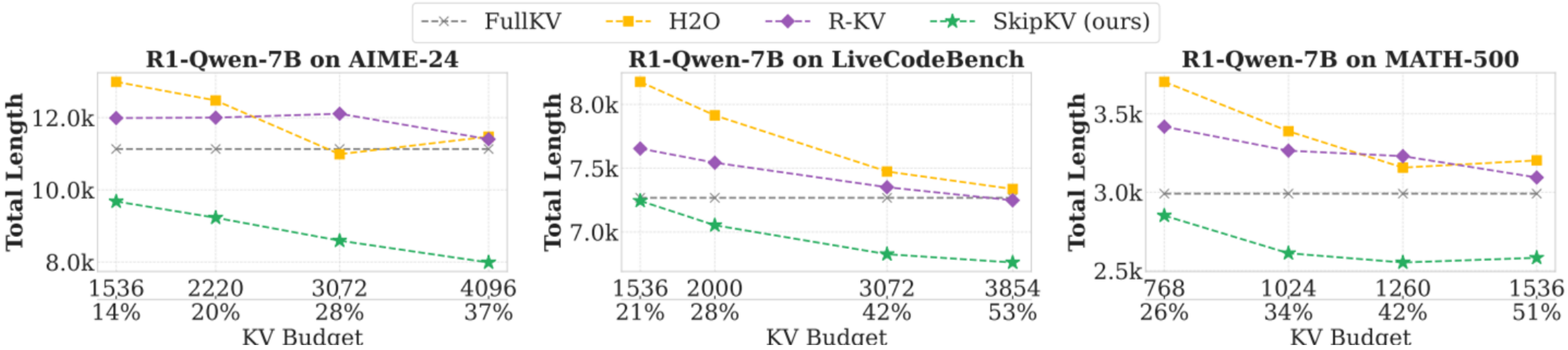
Experimental Results: Accuracy vs KV Budget



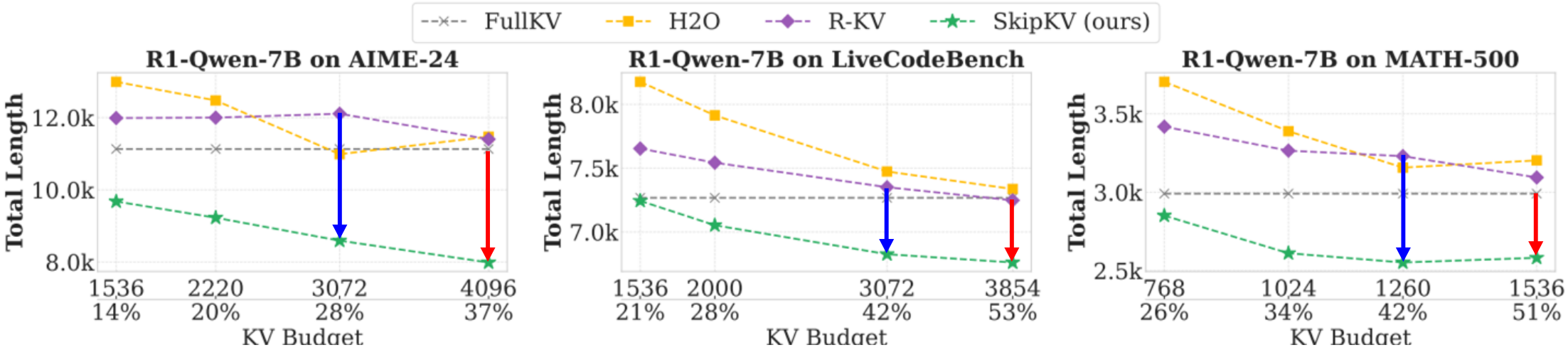
SkipKV vs FullKV:

- **6.7x** less KV cache w/ same accuracy
- **2.5x** less KV cache w/ **+10%** accuracy

Experimental Results: Token Length vs KV Budget



Experimental Results: Token Length vs KV Budget



- SkipKV vs FullKV: -**28%** tokens
- SkipKV vs R-KV: -**48%** tokens

Experimental Results: Throughput

Metric	Batch-size	10	50	100	120	140
Total Latency (min) ↓	FullKV	324	500	OOM	OOM	OOM
	SEAL	178	192	OOM	OOM	OOM
	R-KV	227	96	66	73	70
	SkipKV (ours)	**136**	**58**	**52**	**66**	**68**
Throughput (samples/min) ↑	FullKV	4.07	2.64	OOM	OOM	OOM
	SEAL	7.41	6.87	OOM	OOM	OOM
	R-KV	5.81	13.7	20.0	18.1	18.8
	SkipKV (ours)	**9.70**	**22.7**	**25.4**	**20.0**	**19.4**

Experimental Results: Throughput

Metric	Batch-size	10	50	100	120	140
Total Latency (min) ↓	FullKV	324	500	OOM	OOM	OOM
	SEAL	178	192	OOM	OOM	OOM
	R-KV	227	96	66	73	70
	SkipKV (ours)	**136**	**58**	**52**	**66**	**68**
Throughput (samples/min) ↑	FullKV	4.07	2.64	OOM	OOM	OOM
	SEAL	7.41	6.87	OOM	OOM	OOM
	R-KV	5.81	13.7	20.0	18.1	18.8
	SkipKV (ours)	**9.70**	**22.7**	**25.4**	**20.0**	**19.4**

- SkipKV vs FullKV: **9.6x** higher throughput
- SkipKV vs SEAL: **3.4x** higher throughput
- SkipKV vs R-KV: **1.7x** higher throughput

Takeaways

- **Token-level scores** ignore generation coherence and harms CoT reasoning.
- **Sentence-aware KV eviction and steering** can achieve better accuracy with lower generation token lengths at the same KV budget.
- **Batch grouping** further boost the performance of KV eviction methods via increasing the effective KV budget

Code

Code: https://github.com/TTTTTTris/SkipKV
Email: jiayi_tian@ucsb.edu

Paper