Quest: Query-Aware Sparsity for Efficient Long-Context LLM Inference

Jiaming Tang*, Yilong Zhao*, Kan Zhu, Guangxuan Xiao, Baris Kasikci, Song Han

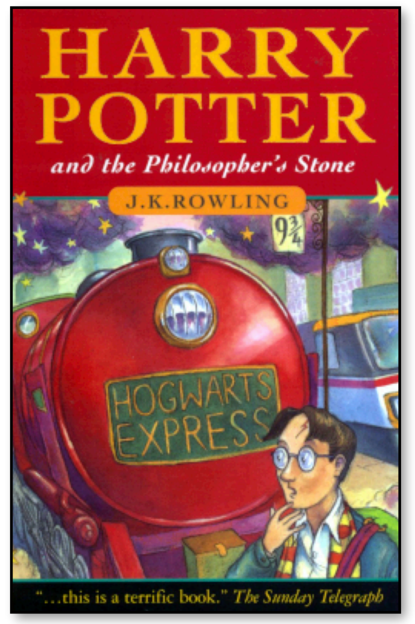
Massachusetts Institute of Technology
University of Washington
Shanghai Jiao Tong University
NVIDIA

Motivation

Why Long Context LLMs?

What inputs can LLMs handle with different context lengths?

A short blog

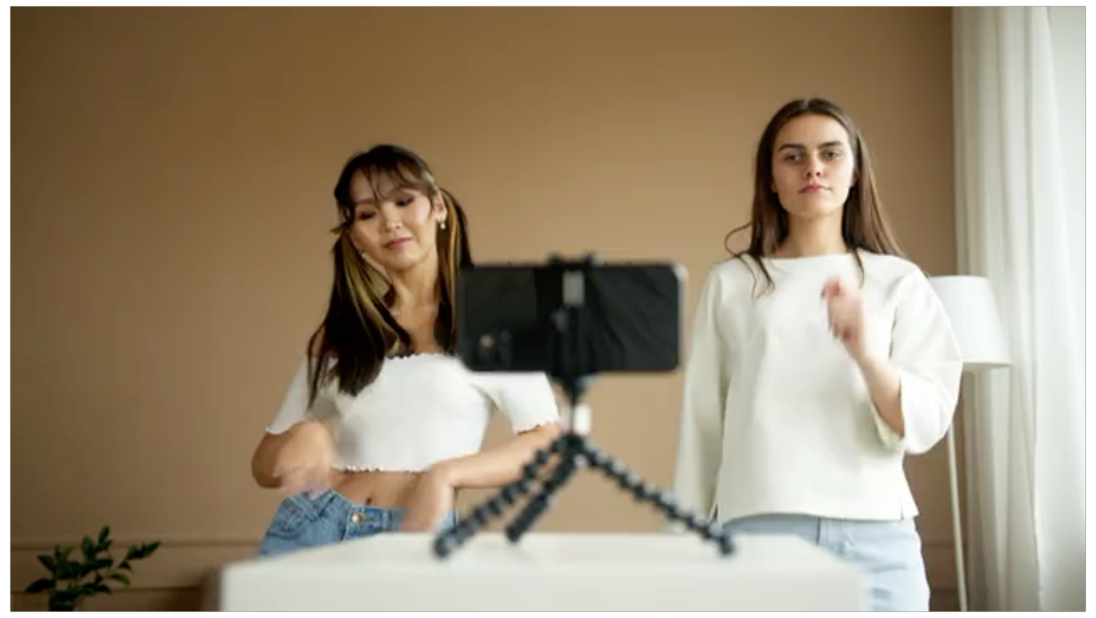


A Harry Potter book

The whole series of Harry Potter books

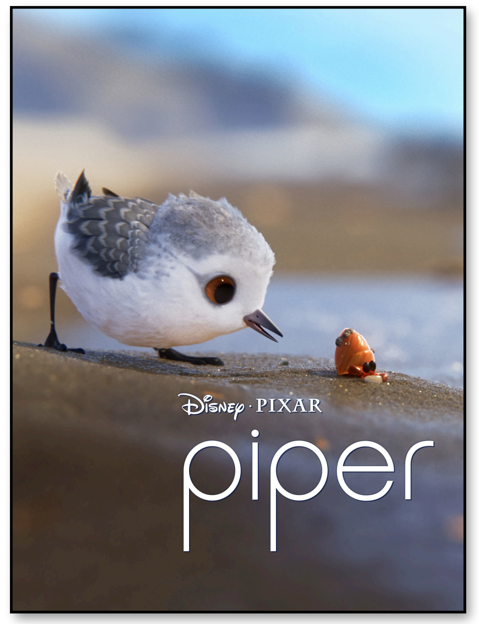
8,000 tokens | 128,000 tokens | 1,000,000 tokens — Context Length

What videos can multi-modal LLMs process (given a rate of 200 tokens per frame per second)?



A short video vlog

~ 30 seconds



A short film – Piper

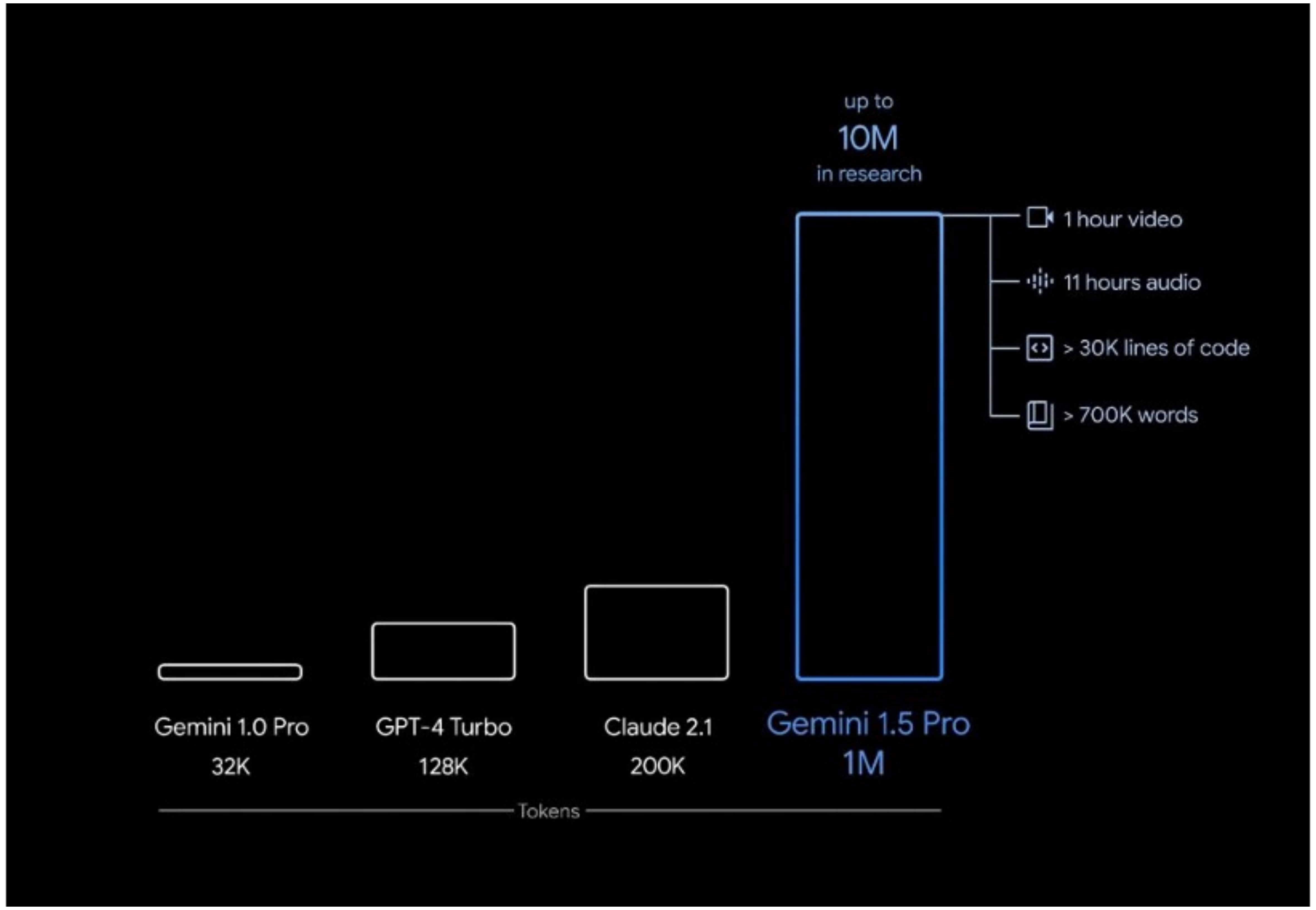
~ 10 minutes

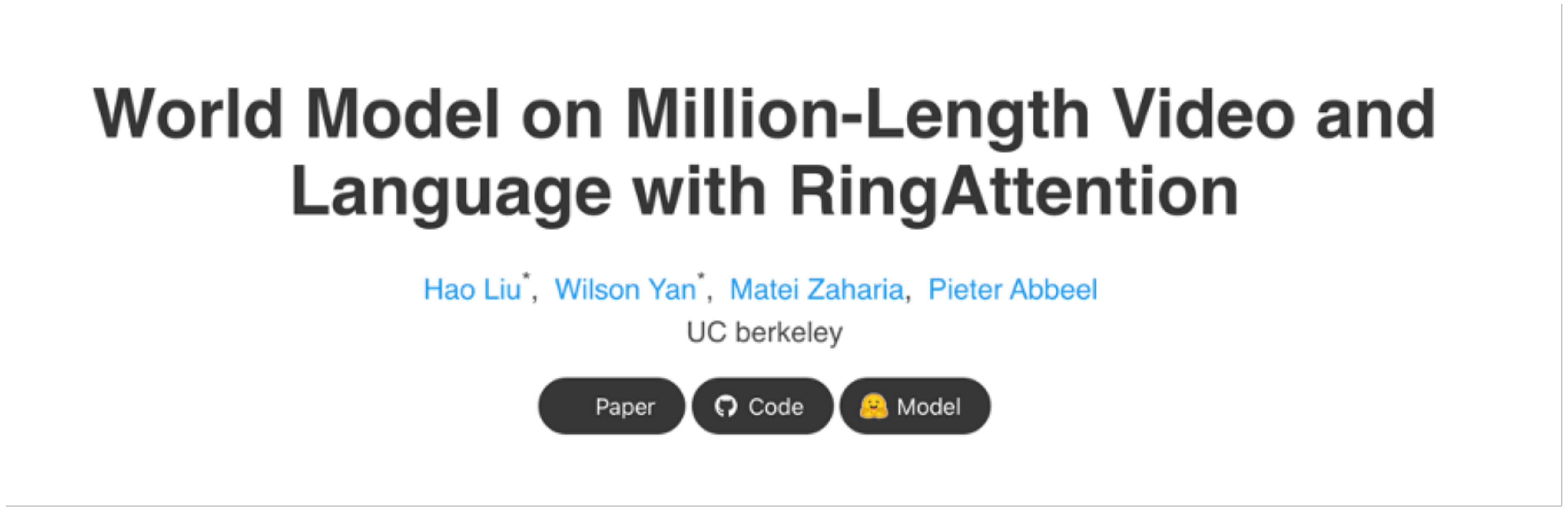
A film

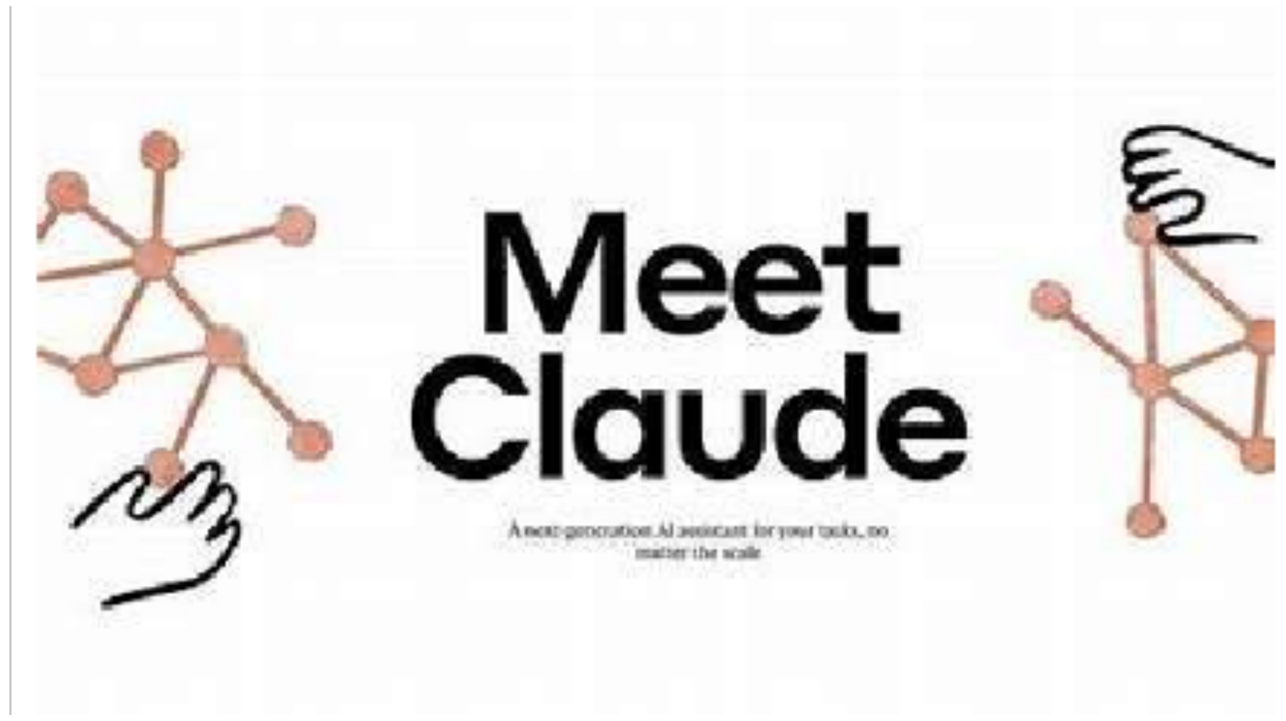
1 - 2 hours

Motivation

- As the demand for long-context large language models (LLMs) increases, models with context windows of up to 128k or even 1M tokens are becoming increasingly prevalent.



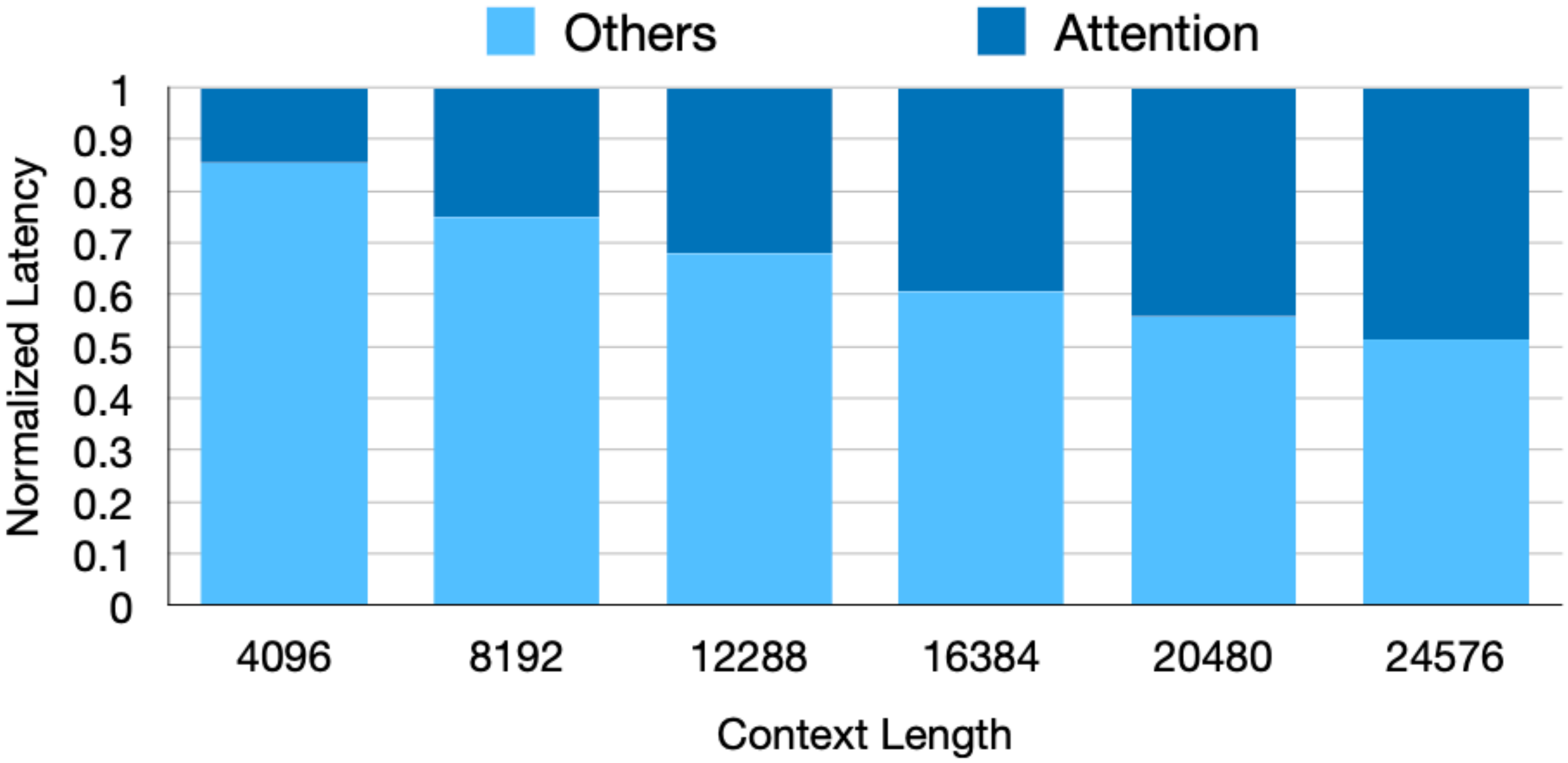




The Problem of Long Context: Large KV Cache

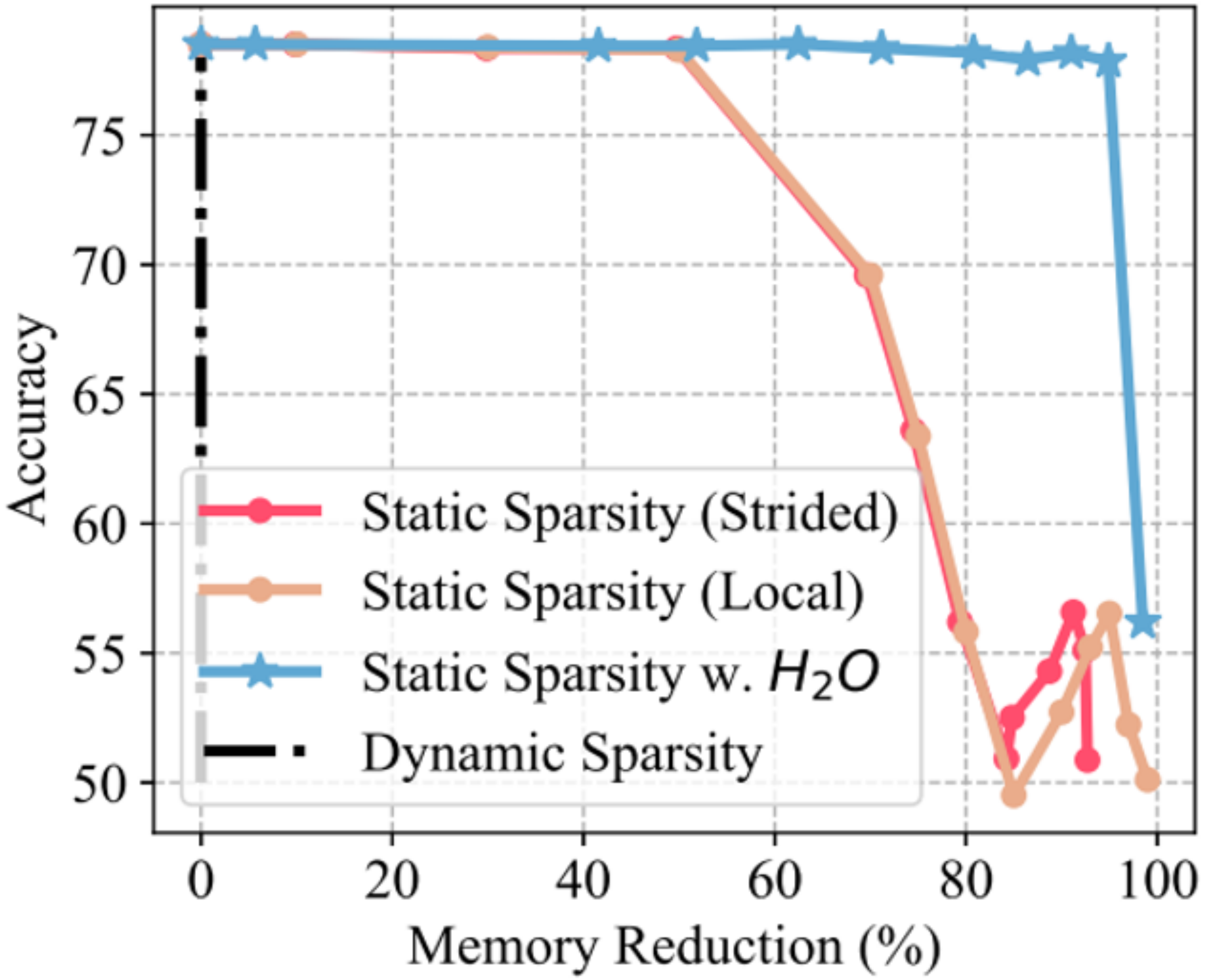
Large KV cache slows down long context inference

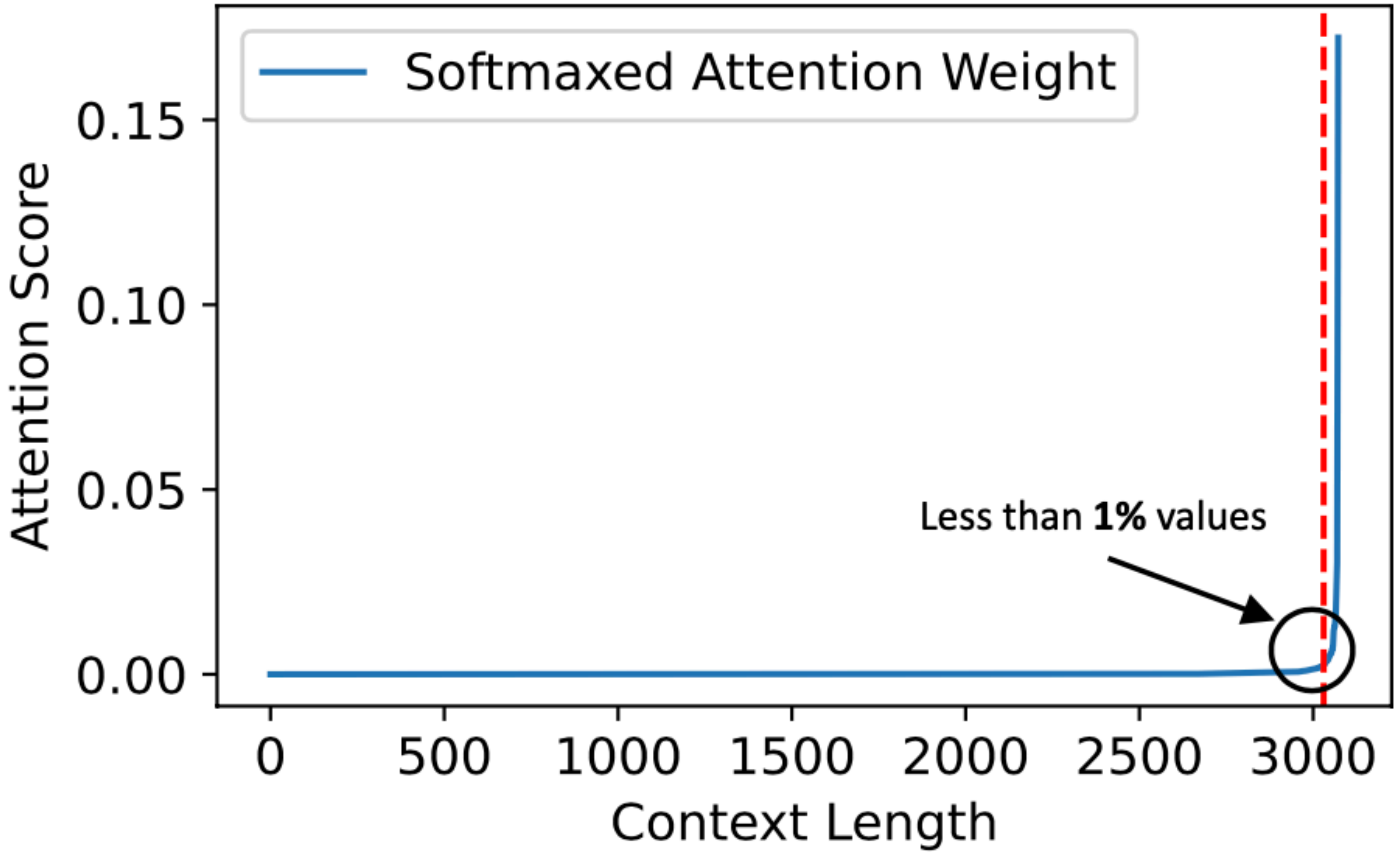
- However, long-context LLM inference is challenging since the **inference speed decreases significantly as the sequence length grows**.
- This slowdown is primarily caused by **loading a large KV cache during attention**.



The Sparsity in Attention Mechanism

- Previous research has highlighted the **inherent sparsity in attention mechanism**.
- Due to this property of self-attention, a small portion of tokens in the KV cache, called critical tokens, can accumulate sufficient attention scores, capturing the most important inter-token relationships.

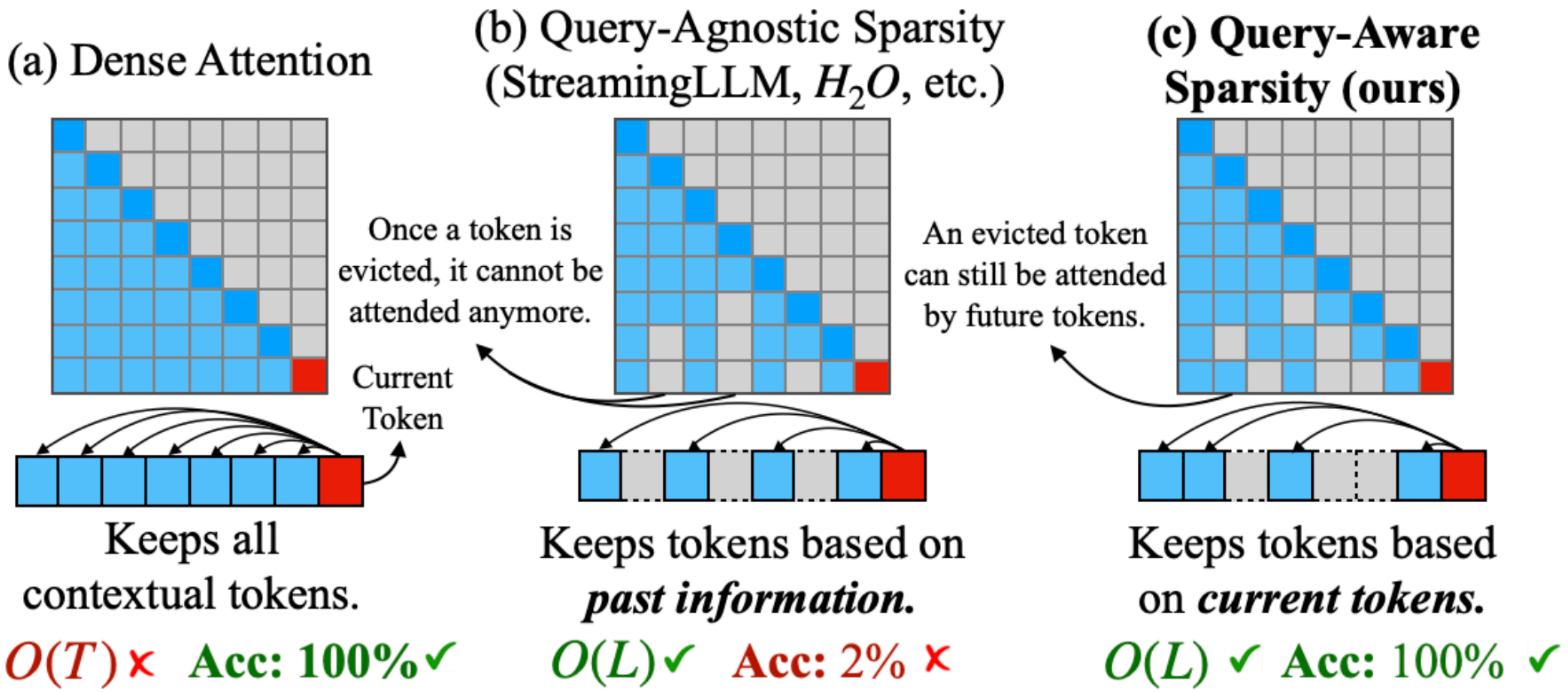




H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. Zhang et al.

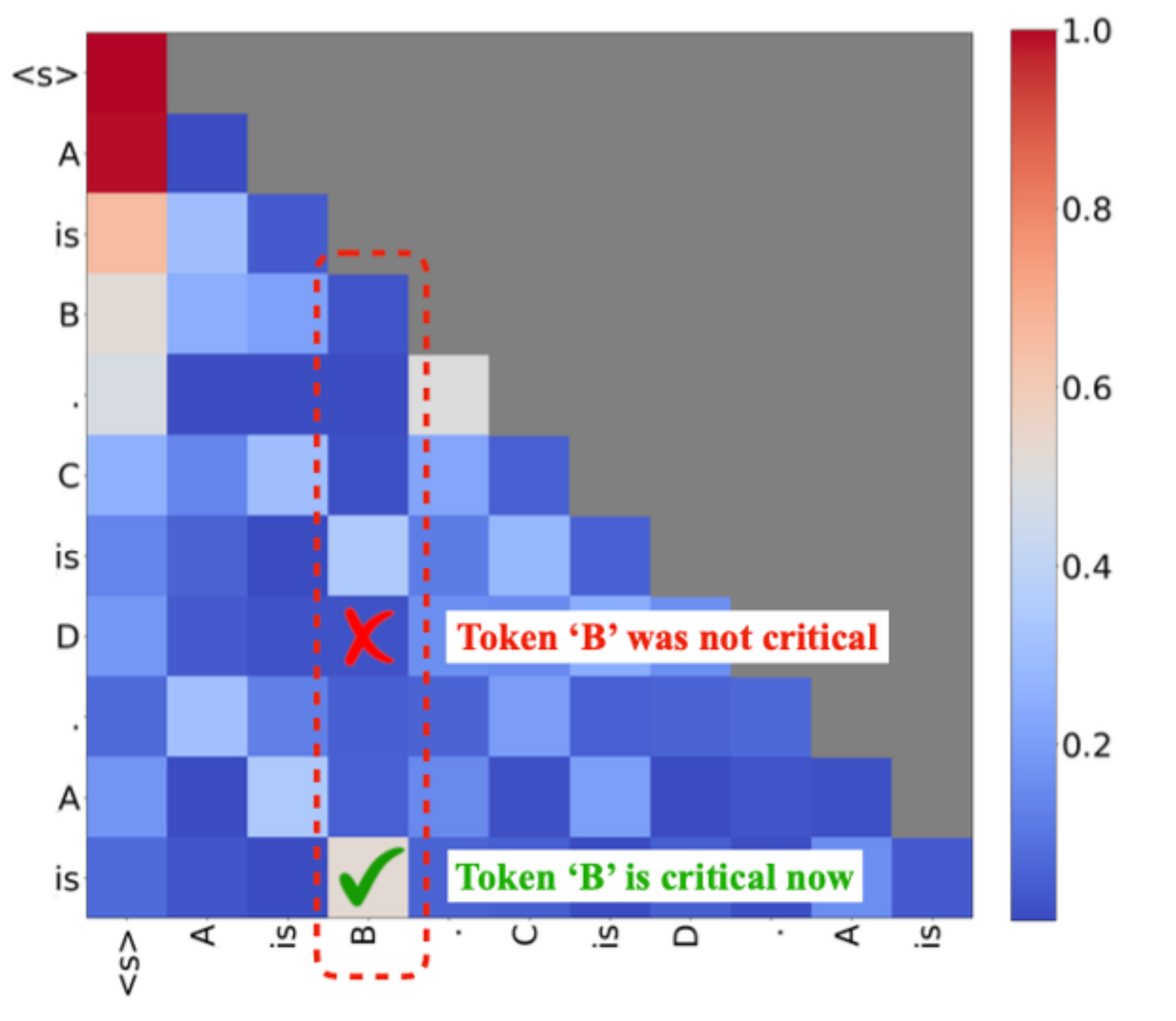
The Limits of Previous Methods

- Many previous efforts have been dedicated to compressing the size of the KV cache to accelerate attention and reduce memory usage.
- These methods decide which parts of the KV cache to discard based on historical information or current states, but **discarded tokens might be important for future tokens**, which may cause the loss of important information.



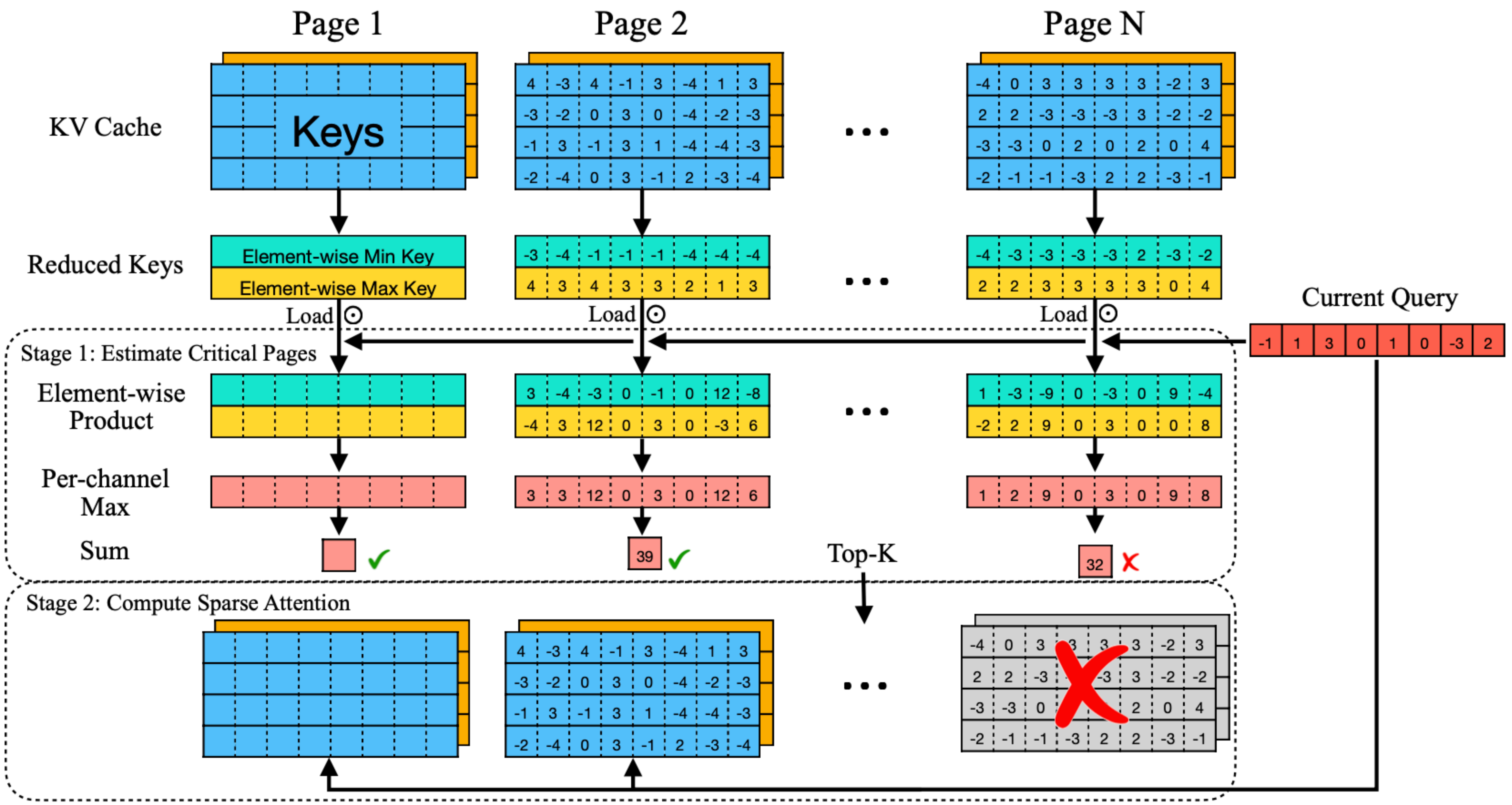
The Limits of Previous Methods

- The **criticality of the tokens is dynamic** and **highly dependent on the query vector Q**.
- Example: the token ‘B’ is critical to the current query ‘is’. Thus, it has a high attention score. However, before the final token ‘is’, ‘B’ is not critical for any previous query and has very low attention scores.



Quest: Using Query-aware Sparsity in Attention

- **Key Idea: preserve all KV cache**, and significantly accelerate inference by reducing the memory movement from the entire KV cache to selected constant K pages.



Quest: Using Query-aware Sparsity in Attention

- Our insight is that in order not to miss critical tokens, we should **select pages containing the tokens with the highest attention weights**.
- However, for an efficient selection of pages, we should **calculate an approximate attention score** following this insight.
- We found that the **upper bound attention weights within a page** can be used to approximate the highest attention score in the page.

When inserting new token to KV cache:
Input: Key vector K, Dimension of hidden states dim, Current maximal vector M_i, Current minimal vector m_i

for $i = 1$ **to** dim **do**
  $M_i = \max(M_i, k_i)$
  $m_i = \min(m_i, k_i)$
end for

When perform self-attention:
Input: Query vector Q, Dimension of hidden states dim, Current maximal vector M_i, Current minimal vector m_i

Initialize $score = 0$.
for $i = 1$ **to** dim **do**
  $score$ += $MAX(q_i * max, q_i * min)$
end for

Quest Performance

Needle-in-a-Haystack

- (i) Results of 10k length passkey retrieval test on LongChat-7b-v1.5-32k.
- (ii) Results of 100k length passkey retrieval test on Yarn-Llama-2-7b-128k.
- Quest can achieve nearly **perfect accuracy** with a KV cache of 64 and 1024 tokens, which is about **1% of the total sequence length**, demonstrating that Quest can effectively preserve the model's ability to handle long-dependency tasks.

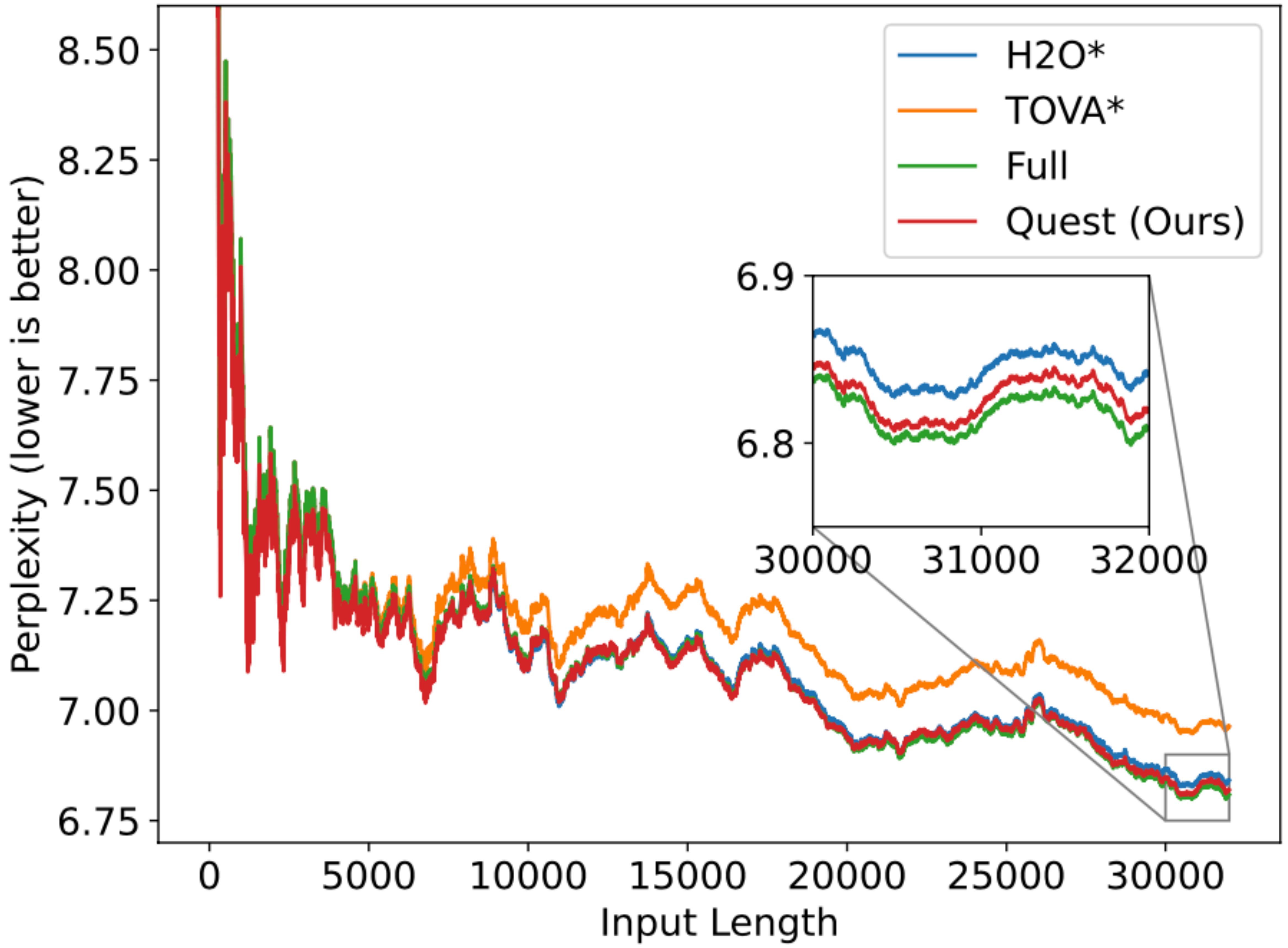
Method / Budget	32	64	128	256	512
H2O	0%	1%	1%	1%	3%
TOVA	0%	1%	1%	3%	8%
StreamingLLM	1%	1%	1%	3%	5%
Quest (ours)	**65%**	**99%**	**99%**	**99%**	**100%**

Method / Budget	256	512	1024	2048	4096
H2O	2%	2%	2%	2%	4%
TOVA	2%	2%	2%	2%	10%
StreamingLLM	1%	1%	1%	2%	4%
Quest (ours)	**88%**	**92%**	**96%**	**100%**	**100%**

Quest Performance

Super Long Language Modeling

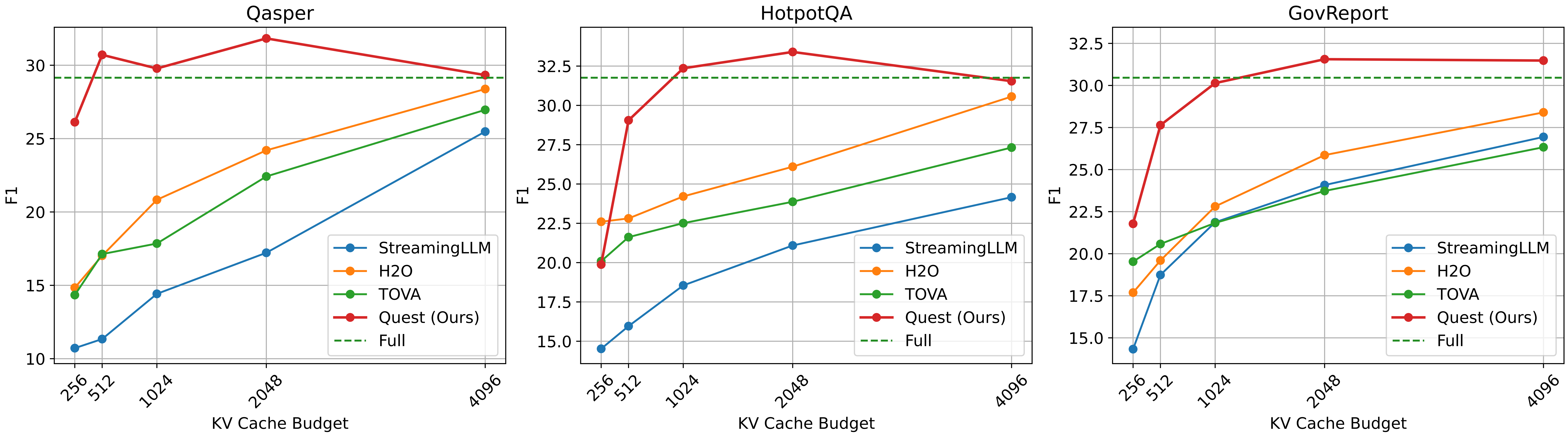
- Language modeling evaluation of Quest on PG19 dataset.
- Quest can **closely match the performance of the full cache model**.



Quest Performance

LongBench

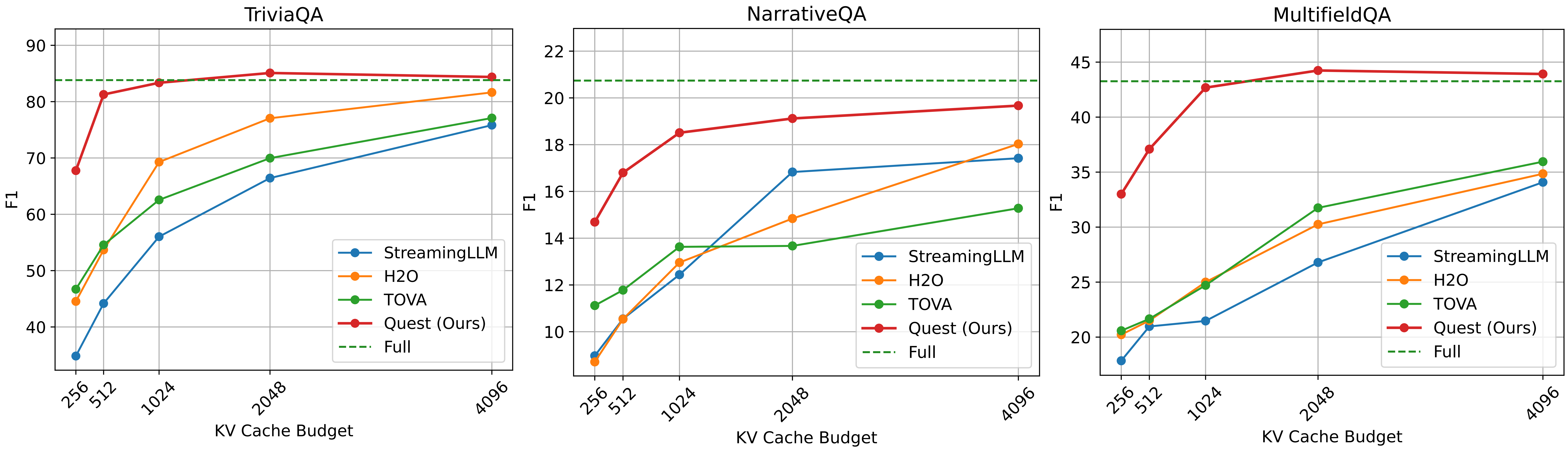
- We evaluate LongChat-7b-v1.5-32k across a wide range of long-context datasets,
- Quest with **a budget of 2k tokens can achieve comparable performance as the model with full KV cache**, while other baselines still exhibit a notable gap from full cache performance even with a larger budget.
- Single-document QA: NarrativeQA, Qasper, MultiFieldQA; multi-document QA: HotpotQA; summarization: GovReport; few-shot learning: TriviaQA.



Quest Performance

LongBench

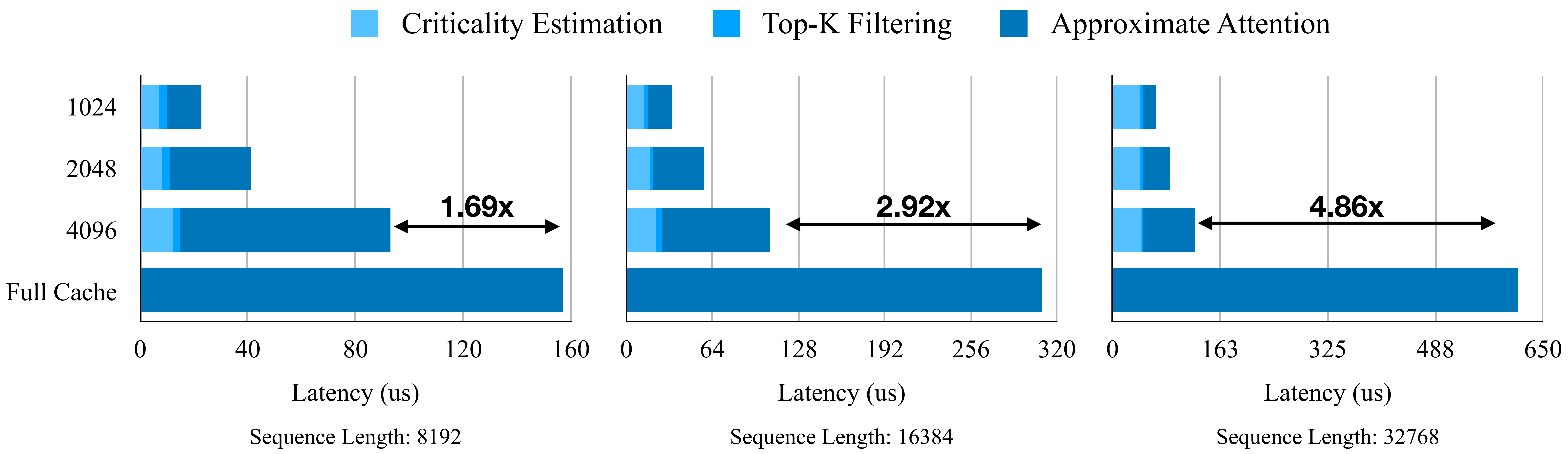
- We evaluate LongChat-7b-v1.5-32k across a wide range of long-context datasets,
- Quest with **a budget of 2k tokens can achieve comparable performance as the model with full KV cache**, while other baselines still exhibit a notable gap from full cache performance even with a larger budget.
- Single-document QA: NarrativeQA, Qasper, MultiFieldQA; multi-document QA: HotpotQA; summarization: GovReport; few-shot learning: TriviaQA.



Efficiency Evaluation

Quest attention time breakdown

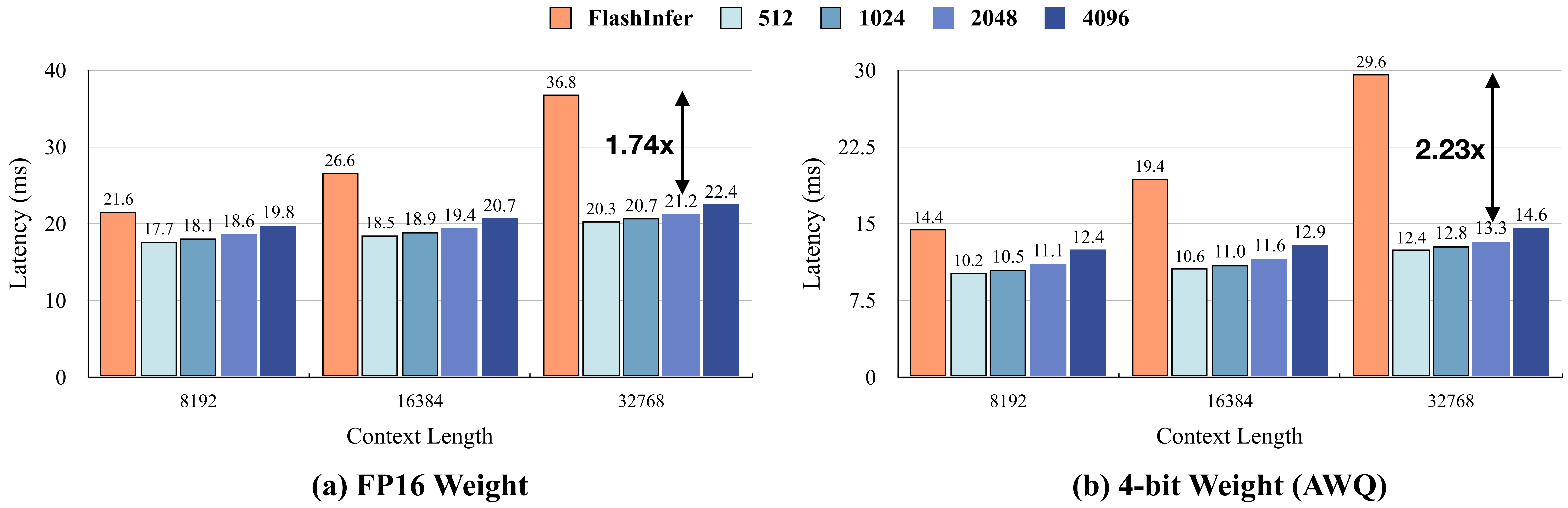
- At all sequence lengths, **Quest significantly outperforms FlashInfer**, as the memory movement is reduced.
- **Quest speeds up self-attention by 7.03×** at sequence length 32k with token budget 2048.



Efficiency Evaluation

End-to-end latency

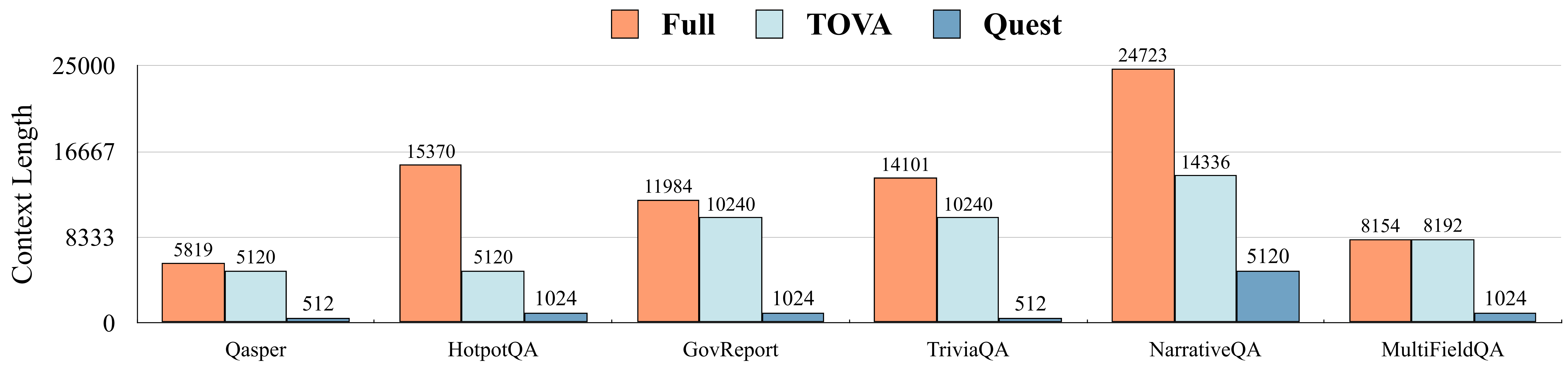
- For all sequence lengths, Quest significantly outperforms FlashInfer. Increasing the sequence lengths only slightly changes the latency of Quest.
- **Quest speedup end-to-end inference by 2.23×** with sequence length 30K, token budget 2048, 4-bit weight quantization.



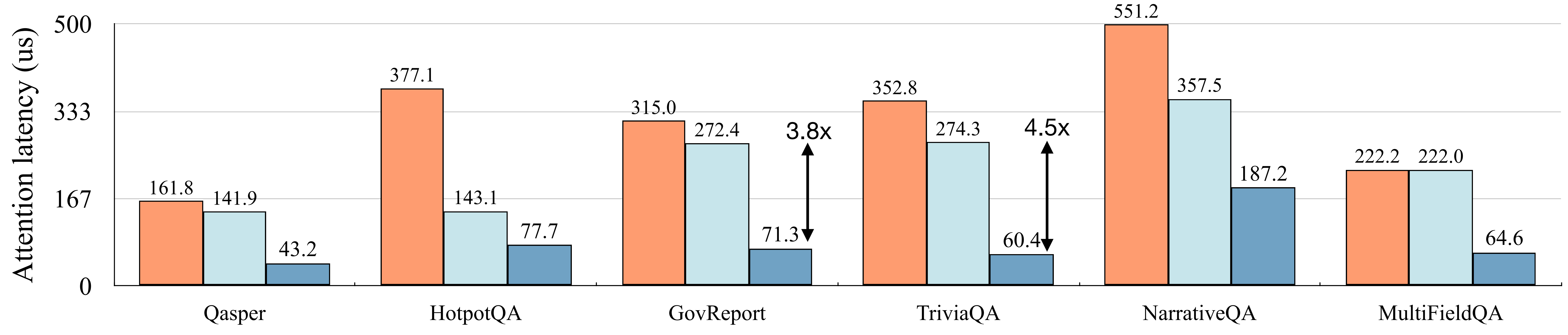
Efficiency Evaluation

Efficiency comparison with baselines

- Baselines need nearly full cache to achieve lossless performance on LongBench benchmarks.
- Therefore, **Quest outperforms the baseline by up to 4.54x with the same lossless accuracy.**



(a) Average context length for lossless accuracy of attention mechanisms



(b) Inference latency of different attention mechanism under benchmarks

Thanks for Listening!

- We propose Quest, an efficient long-context LLM inference framework that **leverages query-aware sparsity in the KV cache** to accelerate the attention mechanism.

- Code: https://github.com/mit-han-lab/Quest
- Paper: https://github.com/mit-han-lab/Quest/blob/main/assets/quest_paper.pdf
- Poster: https://github.com/mit-han-lab/Quest/blob/main/assets/quest_poster.pdf