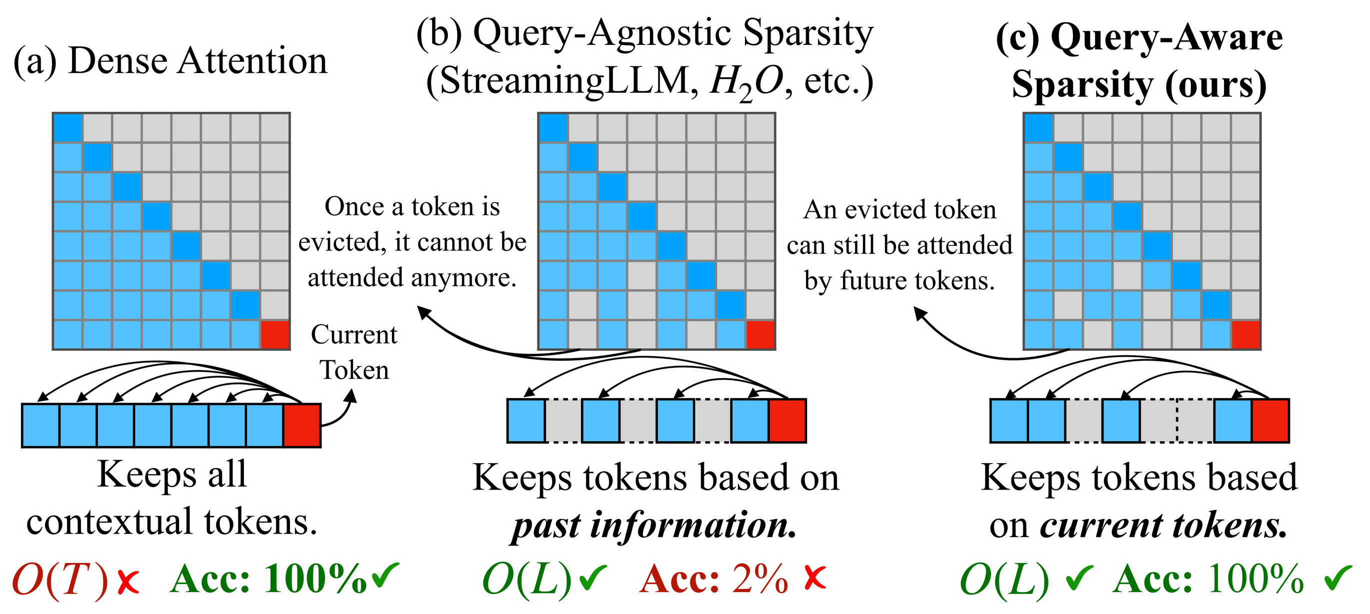
Quest: Query-Aware Sparsity for Efficient Long-Context LLM Inference

Jiaming Tang[1,2*], Yilong Zhao[1,3*], Kan Zhu[3], Guangxuan Xiao[2], Baris Kasikci[3], Song Han[2,4]

[1]SJTU, [2]MIT, [3]UW, [4]NVIDIA https://github.com/mit-han-lab/Quest

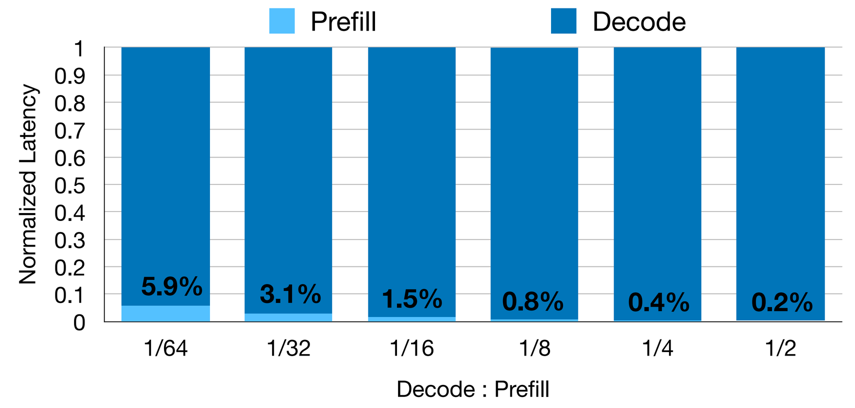
Introduction

- **Long-context** text-generation has gained popular applications.
- However, it poses great **memory pressure** to the inference system.
- In this work, we prose Quest, which exploits **query-aware sparsity** in self-attention operator to boost inference efficiency, with negligible accuracy loss.



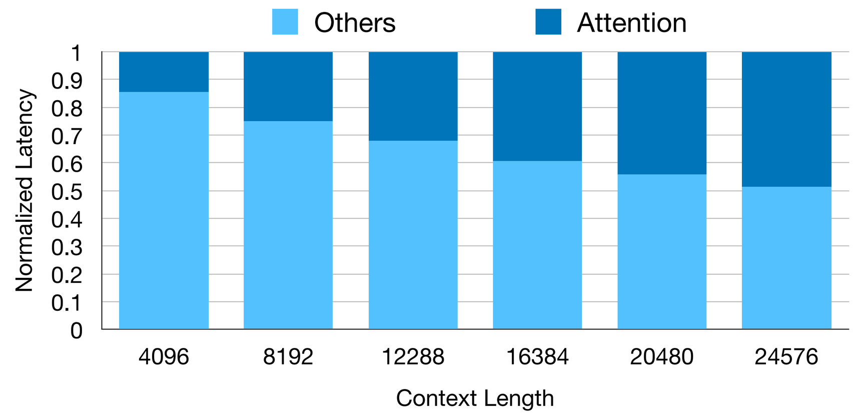
Attention in Decode Phase is Costly

- **Decode phase** consumes great portion of time compared to prefill phase, due to the auto-regressive inference of LLMs.



Time ratio under 1K prefill length with various decode length

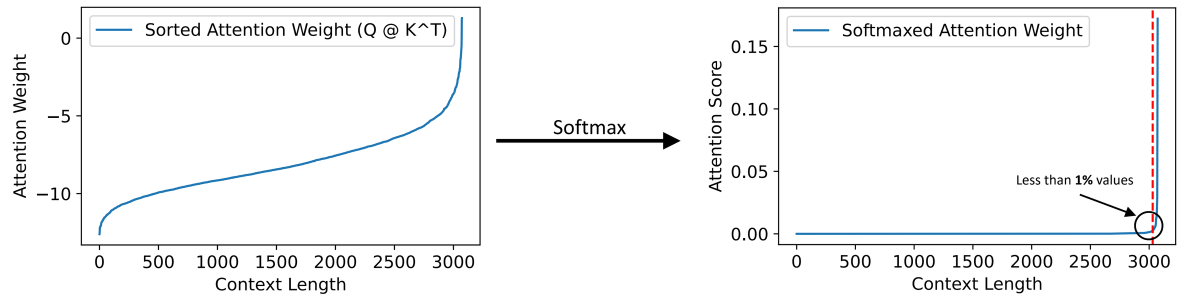
- **Attention** operator needs read entire KV-Cache at each iteration, which increases linearly with the context length.



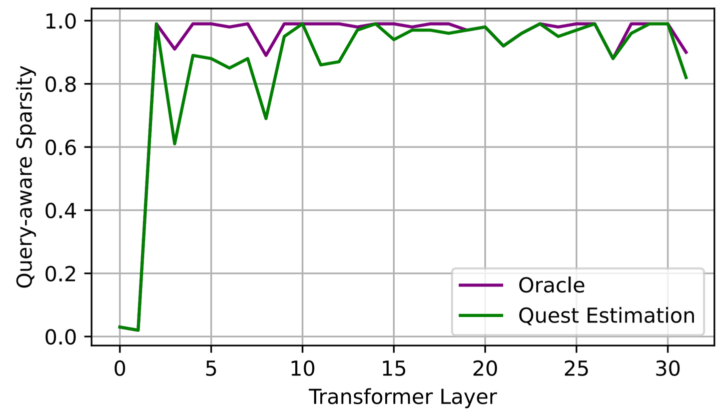
Inference Time Breakdown

Finding 1: Attention is Sparse

- During attention calculation, only small portion of tokens has much **larger magnitude of attention scores** than others.



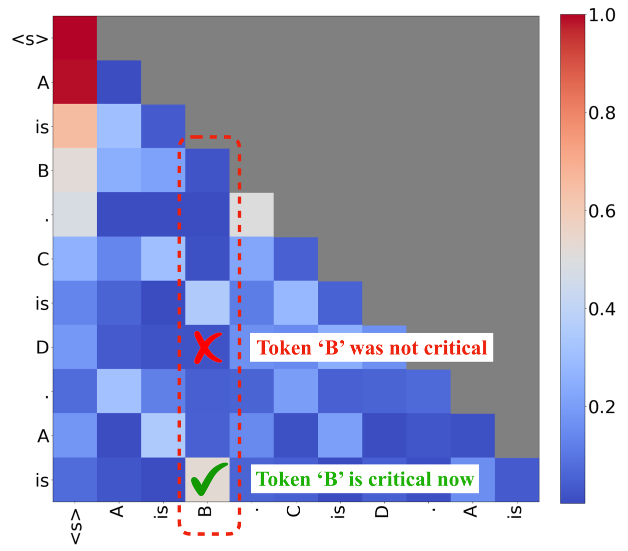
- Therefore, attention output can be effectively approximated by only a small portion of **critical tokens**, which is called **sparsity**.



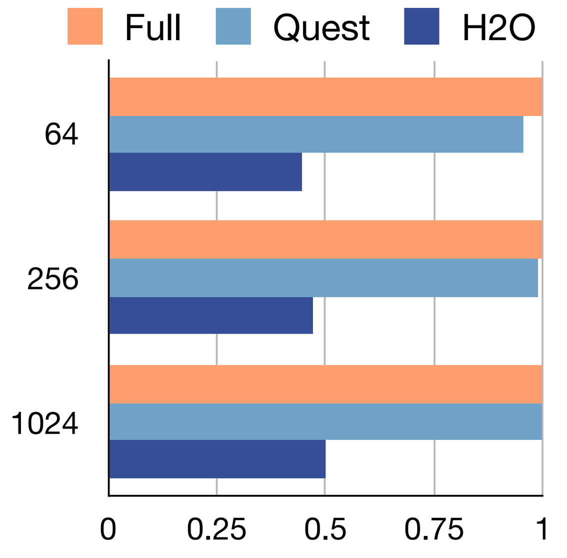
Sparsity of maintaining PG19 Perplexity

Finding 2: Sparsity Depends on Query

- We argue that **critical tokens depend on the input query.** For e.g., summary task will attend on different paragraphs, sequentially.
- Therefore, **query-agnostic** sparsity (like H2O) will prune tokens which will be critical in future.



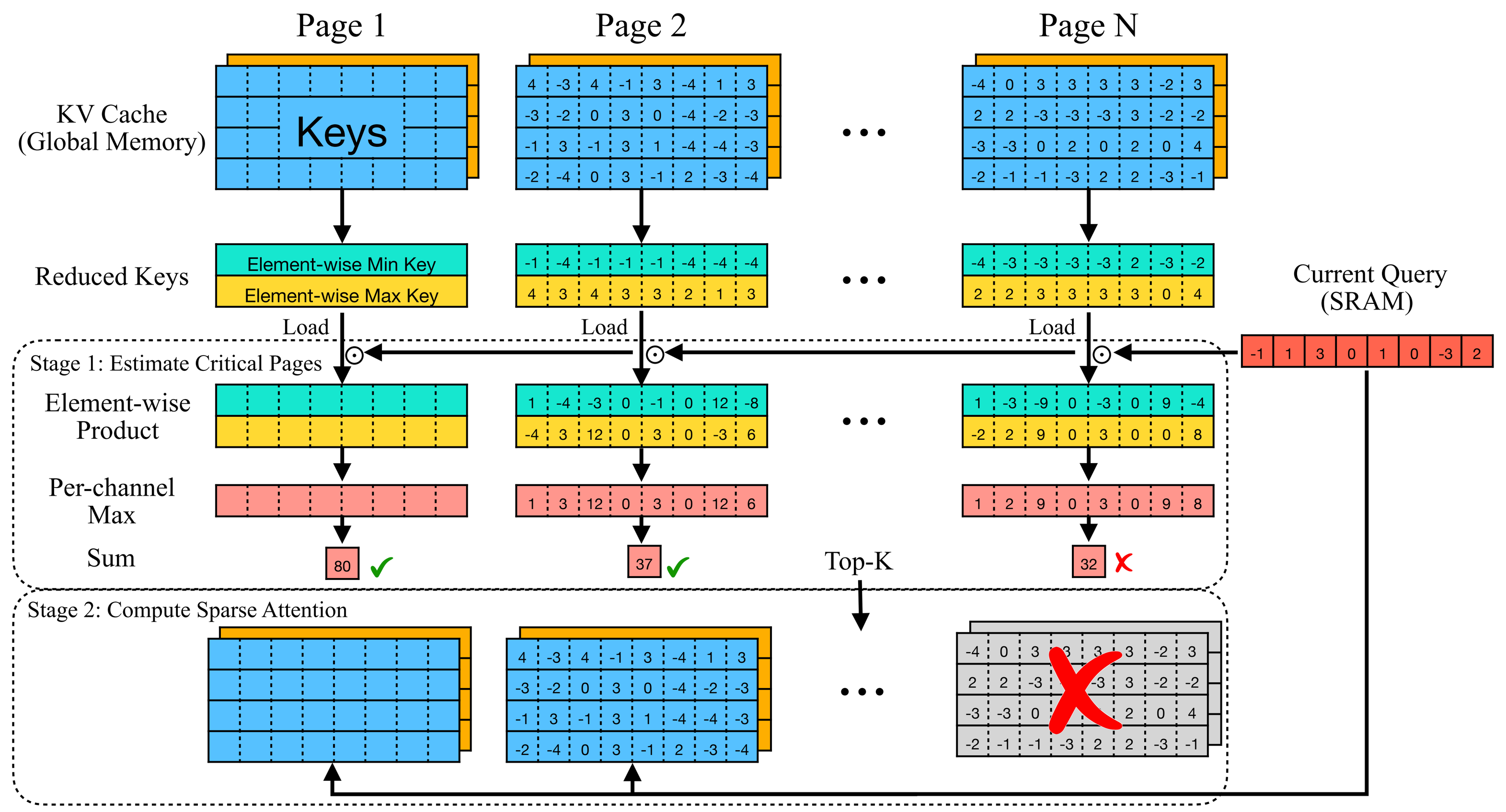
Sampled Attention Map



Top-10 Recall Rate with Various Token Budgets (10K Context Len)

Overview of Quest

- Instead of prune, Quest **dynamically selects critical tokens** via criticality estimation for each query, at the page granularity which is compatible with PageAttention.
- Quest applies sparse attention only **on the selected tokens**, which saves greatly memory movement.



Evaluation & Implementation

- We implemented specialized operators (criticality estimation, Top-K, sparse attention), based on kernel libraries, RAFT and FlashInfer.
- For efficiency evaluation, we run experiments on Ada 6000 with CUDA 12.3.
- For accuracy evaluation, we evaluate on Pass-Key retrieval and common sense tasks from LongBench.

Pass-Key Retrieval

- 10K context length tested on LongChat-7b-v1.5-32k

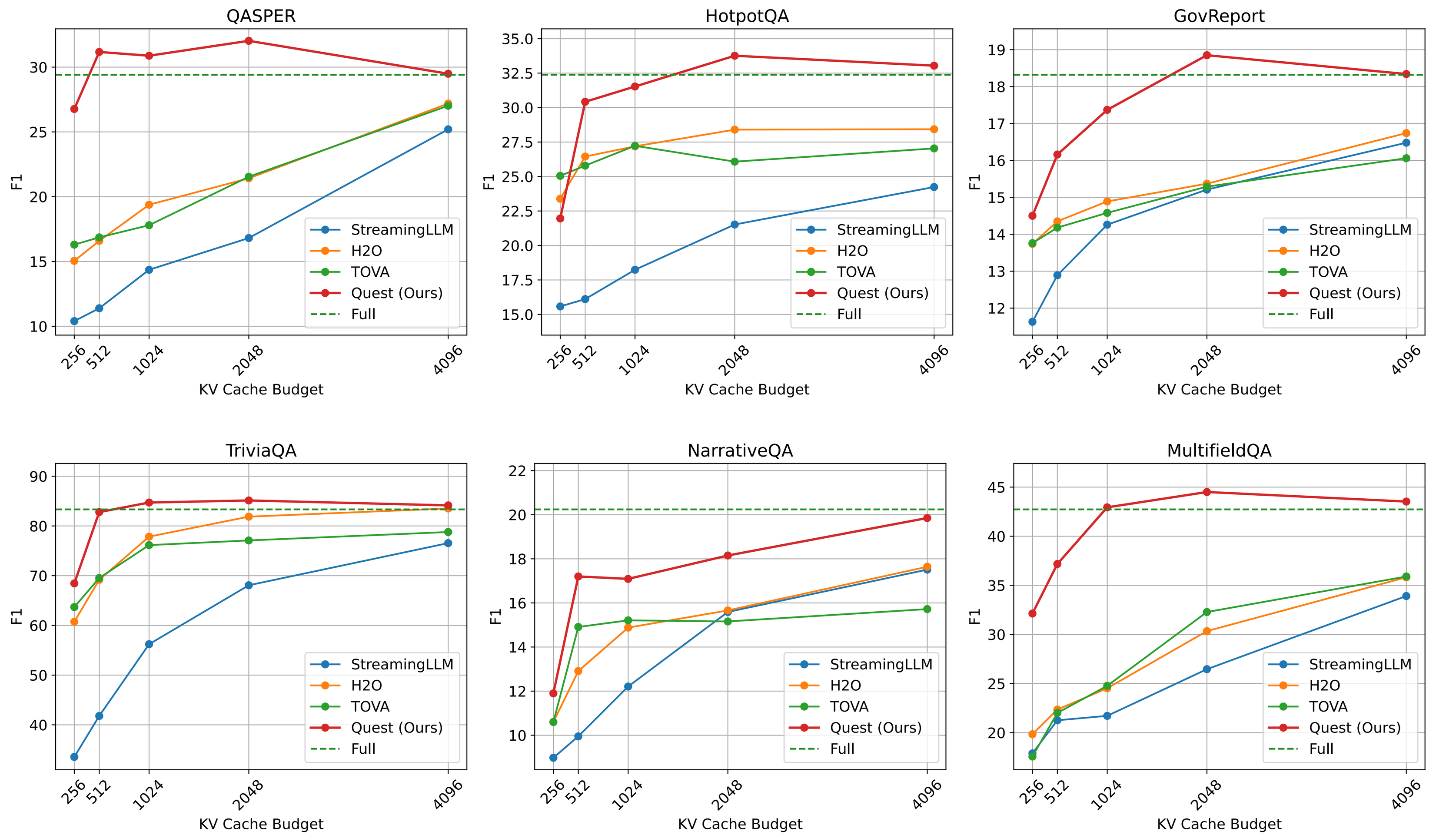
Method / Budget	256	512	1024	2048	4096
H2O	2%	2%	2%	2%	4%
TOVA	2%	2%	2%	2%	10%
StreamingLLM	1%	1%	1%	2%	4%
Quest (ours)	**88%**	**92%**	**96%**	**100%**	**100%**

- 100K context length tested on Yarn-Llama2-7b-128k

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%**

LongBench Tasks

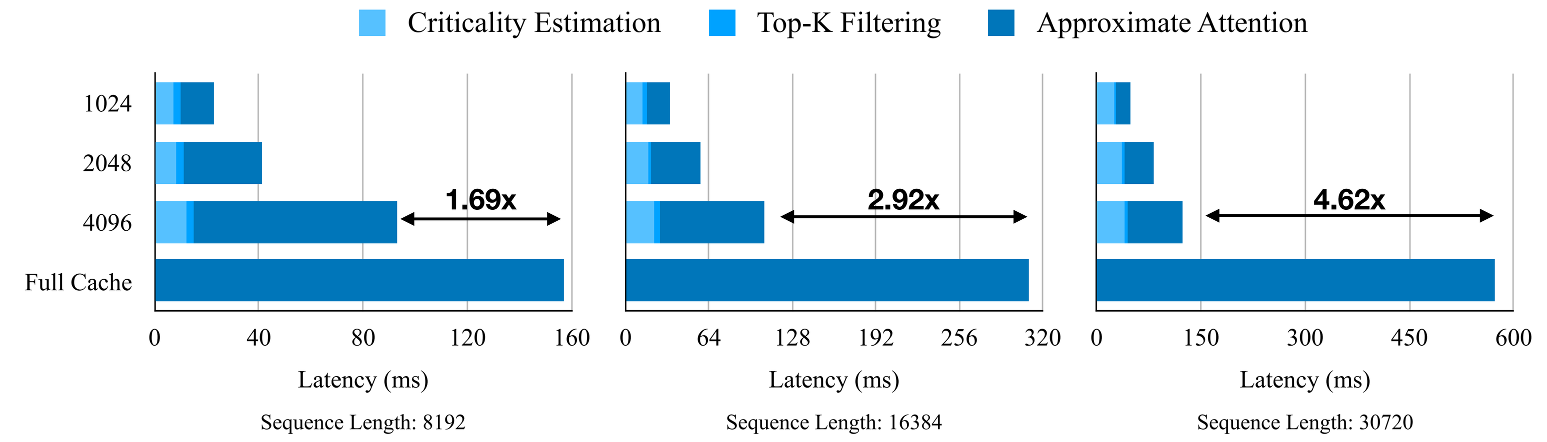
- Quest consistently **outperforms** all baselines.
- Note that Quest achieves full accuracy with **2K budgets** in most cases.



LongChat- 7b-v1.5-32k with various token budgets

End-to-end Efficiency

- Breakdown of Quest's **attention** operator under various context length.



- End-to-end speedup compared to FlashInfer version.