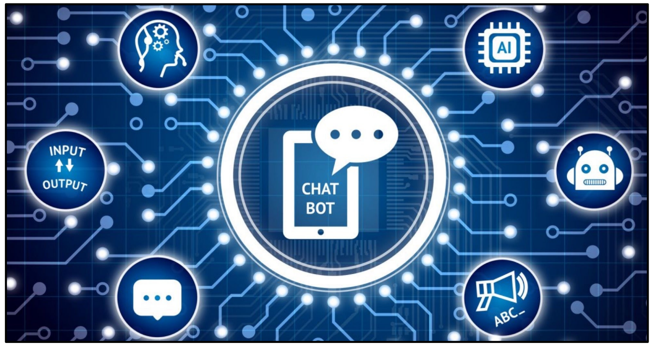
Scalable, Robust, and Hardware-aware Speculative Decoding for Efficient Long Sequence Generation

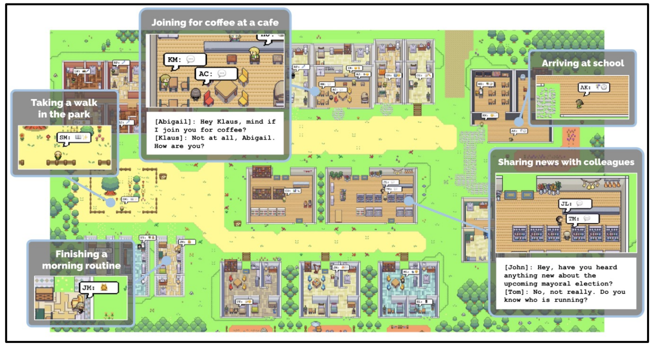
Beidi Chen (CMU)

Sequoia: Scalable, Robust, and Hardware-aware Speculative Decoding. Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, Beidi Chen.
https://github.com/Infini-AI-Lab/Sequoia

TriForce: Rethinking Applicable Speculative Decoding For Long-Context Model Serving. Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, Beidi Chen.

LLMs are Powerful

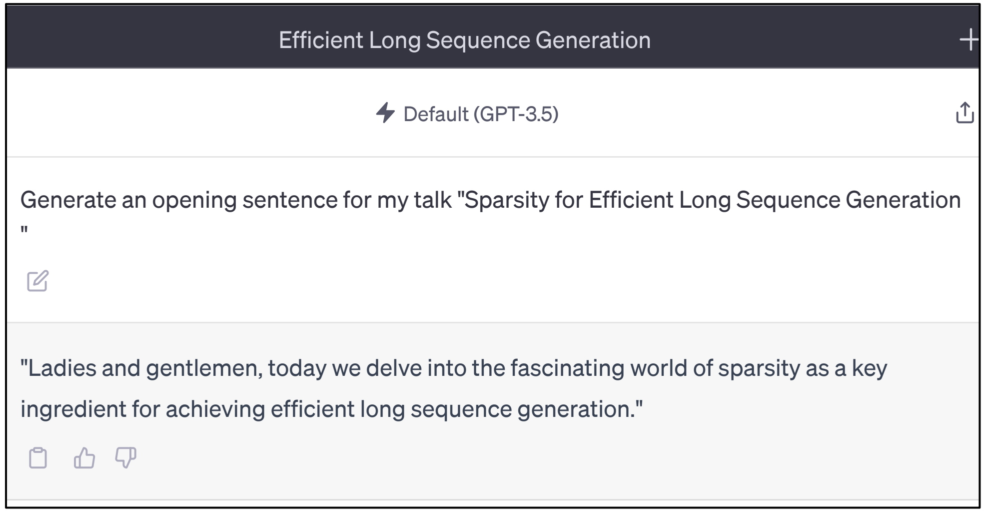


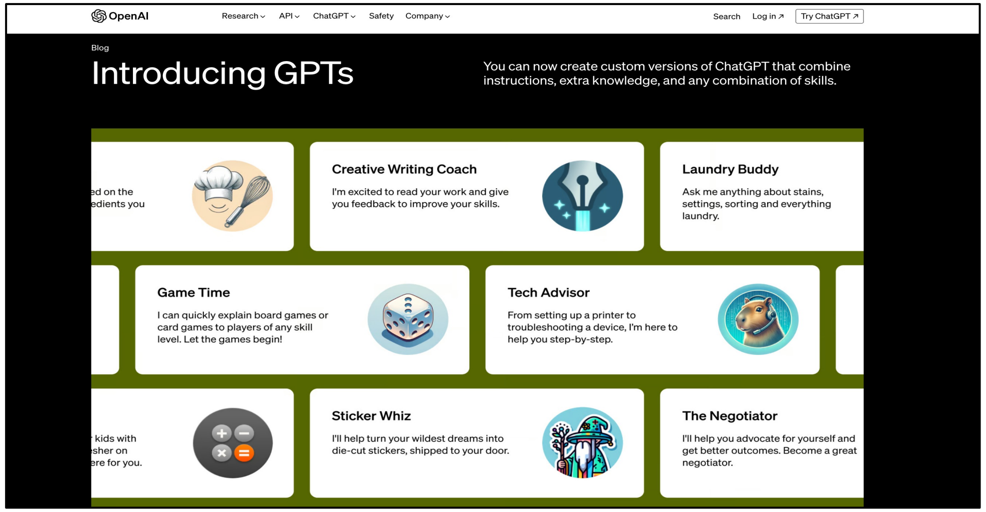


Conversational AI

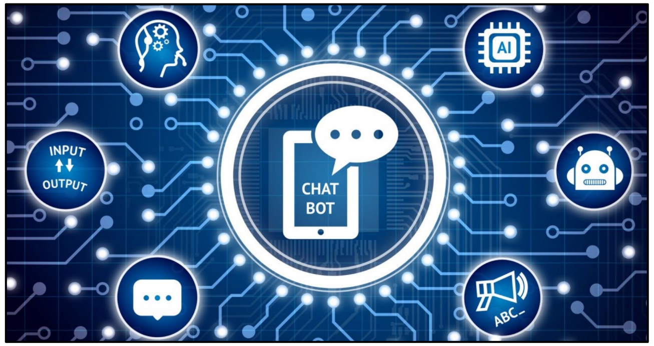
Content Generation

AI Agents



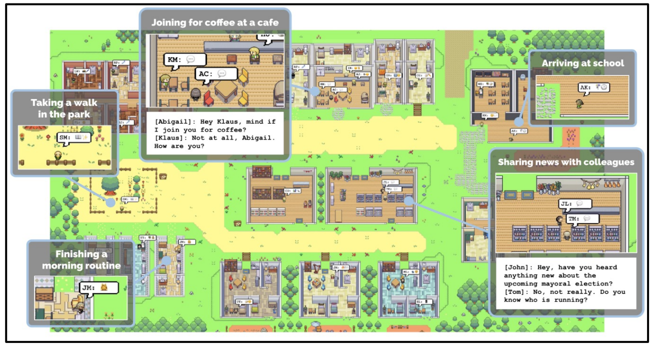


LLMs are Powerful , but Very Expensive to Deploy



Conversational AI

Content Generation

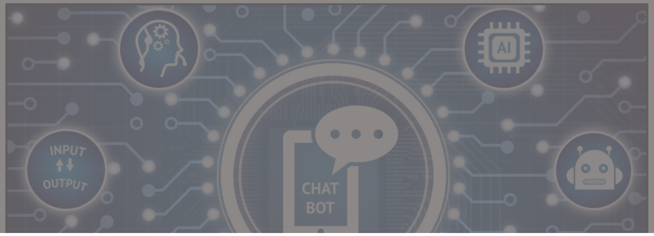


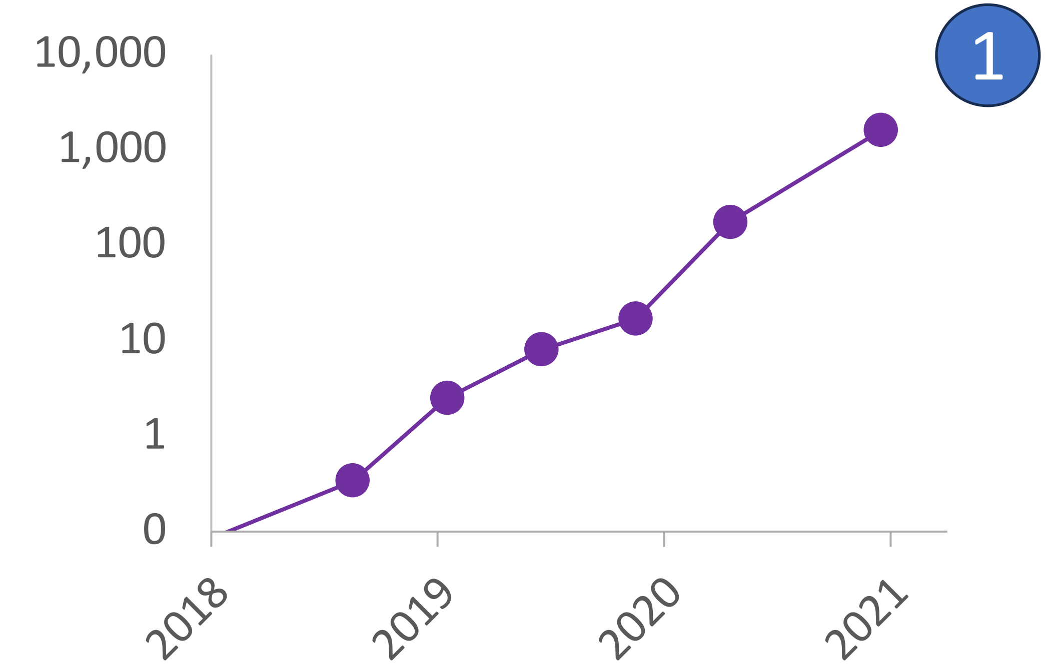
AI Agents

Major Challenges: memory IO (*Pope et al.*)

- large mem, e.g. a Llama2-70B model needs
  - 140 GB for parameters,
  - 160 GB for activation (KV cache),
    even with Multi-Group-Attention (8K seqlen + 64 batch size)
- low parallelizability, e.g. generate 100 tokens -> load model, KV cache 100 times

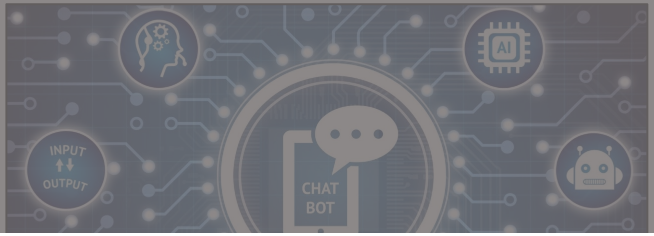
LLMs are Powerful , but Very Expensive to Deploy

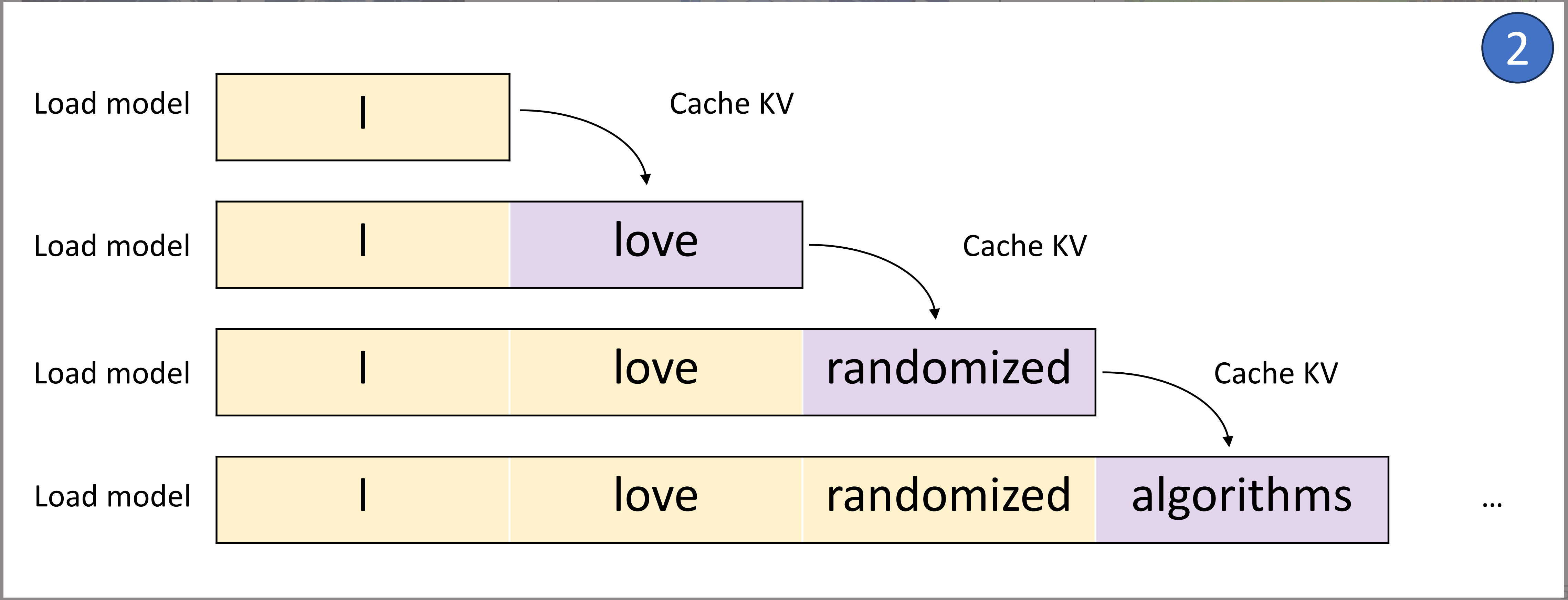




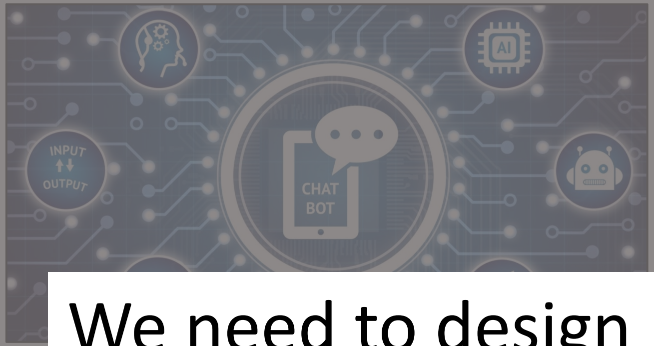
Exponential model size

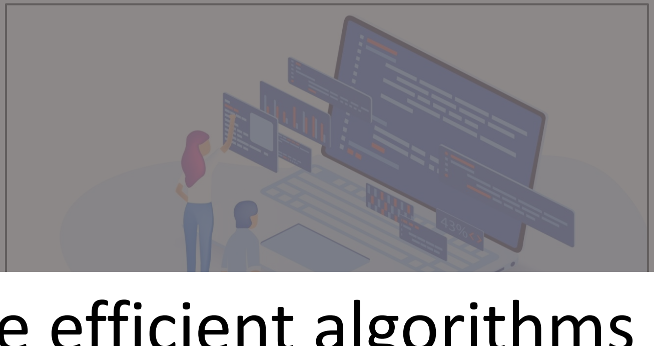
LLMs are Powerful , but Very Expensive to Deploy

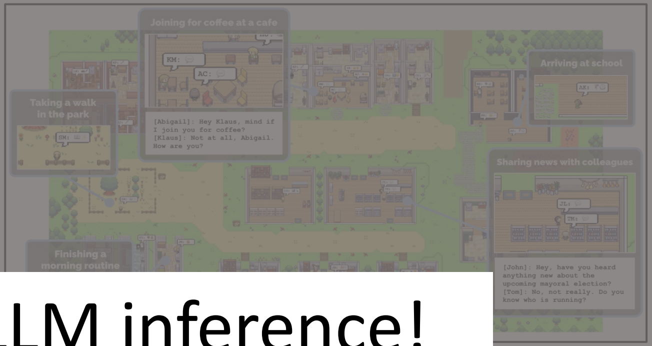




LLMs are Powerful , but Very Expensive to Deploy







We need to design more efficient algorithms for LLM inference!

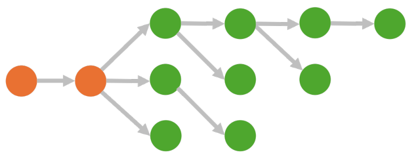
AI Agents

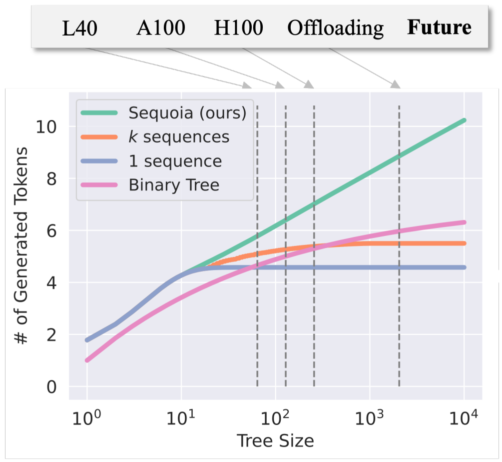
Major bottleneck: memory IO (*Pope et al.*)

- large mem, e.g. a Llama2-70B model needs
  - 140 GB for weights,
  - 160 GB for KV cache even with MGA (8K seqlen + 64 batch size)
- low parallelizability, e.g. generate 100 tokens -> load model, KV cache 100 times

LLMs are Powerful , but Very Expensive to Deploy

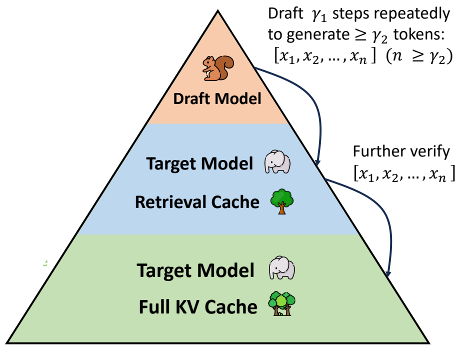
Sequoia (*new* 🔥)

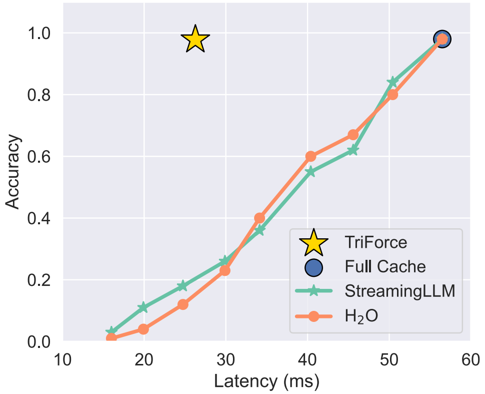




- Serve a Llama2-70B on a single RTX-4090 with 0.57s / token latency, 9× faster than DeepSpeed-Zero Offloading
- Serve a Llama2-7B, Llama2-13B, and Vicuna-33B on an A100 by 4.04×, 3.73×, and 2.27×

TriForce (*coming soon* 🔥)

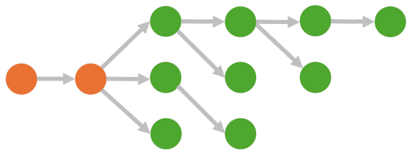


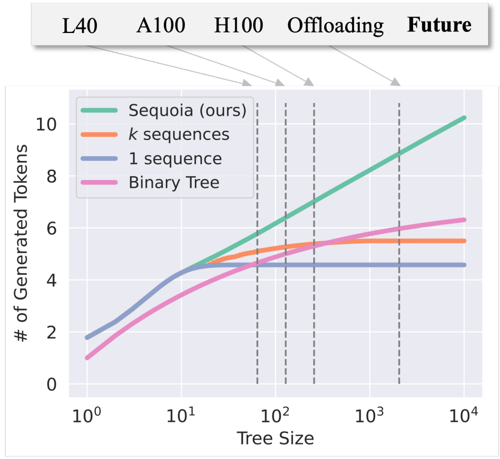


- Serve a Llama2-7B-128K (78GB mem) on a single RTX-4090 with 0.3s / token latency, 8× faster than DeepSpeed-Zero Offloading
- 2.3× speedup on a single A100 GPU

LLMs are Powerful , but Very Expensive to Deploy

Sequoia (*new* 🔥)

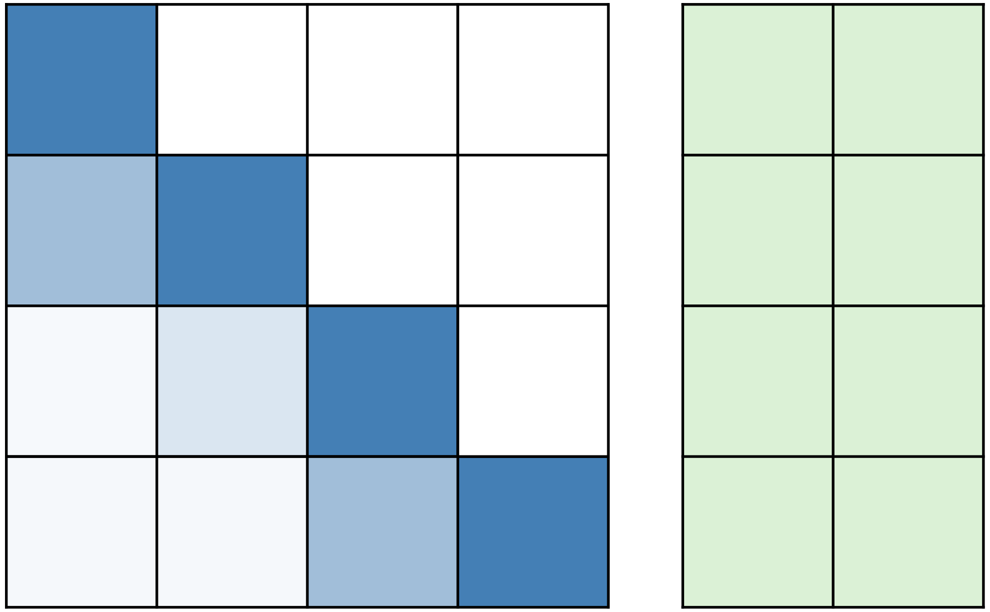




- Serve a Llama2-70B on a single RTX-4090 with 0.57s / token latency, 9× faster than DeepSpeed-Zero Offloading
- Serve a Llama2-7B, Llama2-13B, and Vicuna-33B on an A100 by 4.04×, 3.73×, and 2.27×

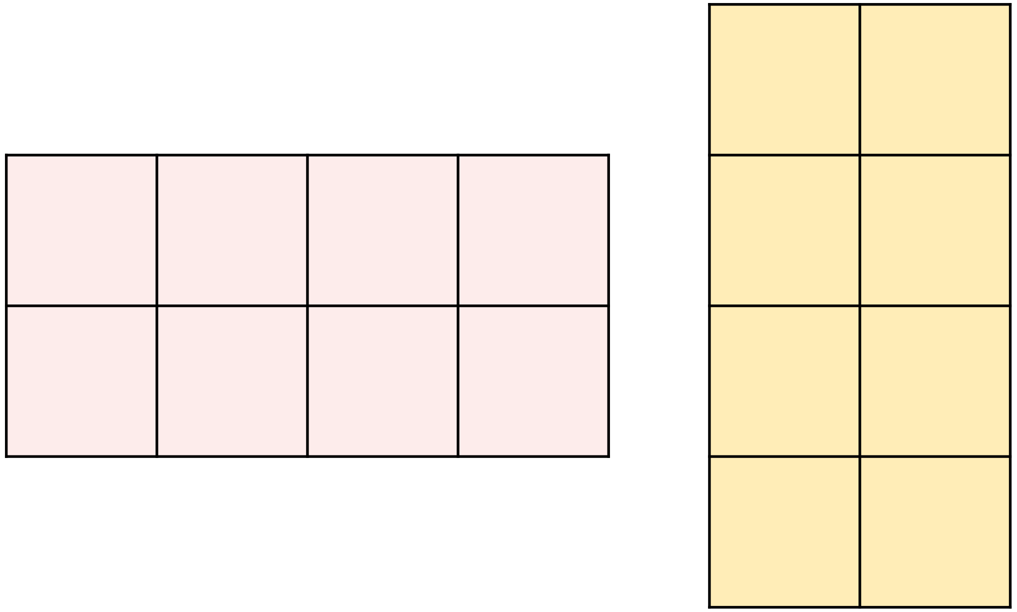
Background: Transformer Architecture

Attention



$\{W_q, W_k, W_v, W_o\} \in R^{d\text{x}d}$

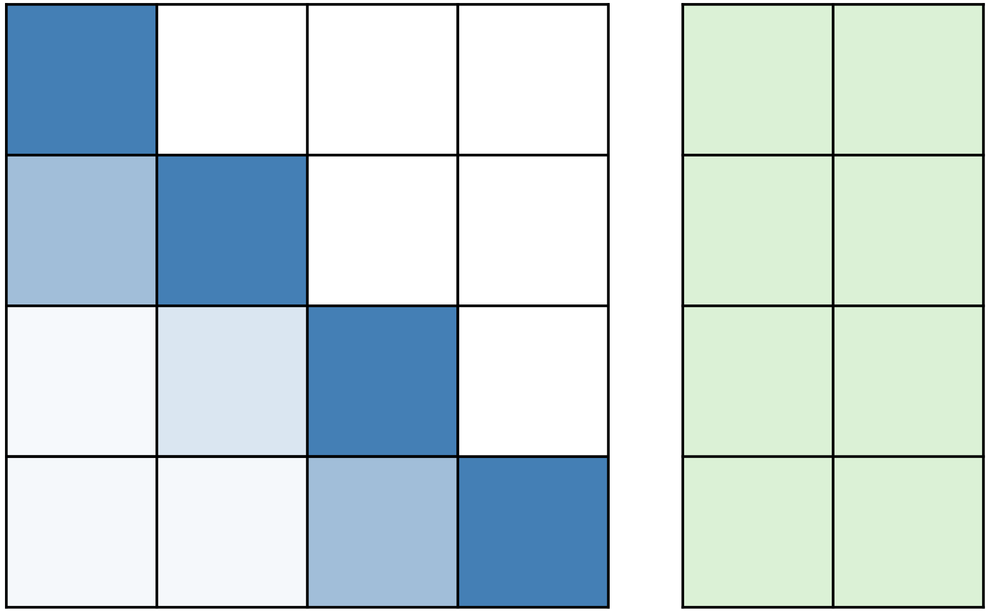
MLP



$\{W_1, W_2\} \in R^{d\text{x}4d}$

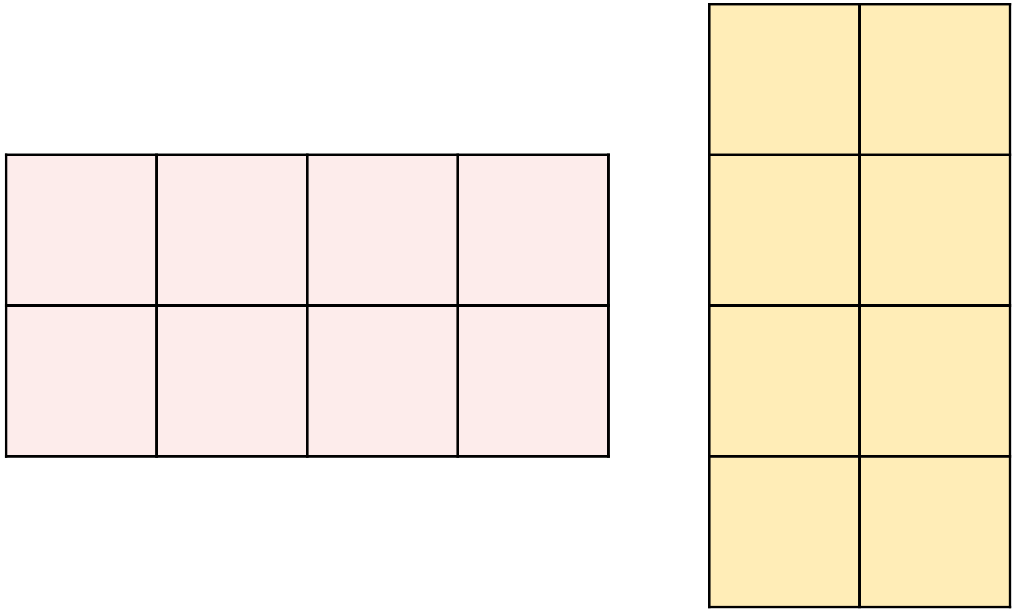
Background: Transformer Architecture

Attention



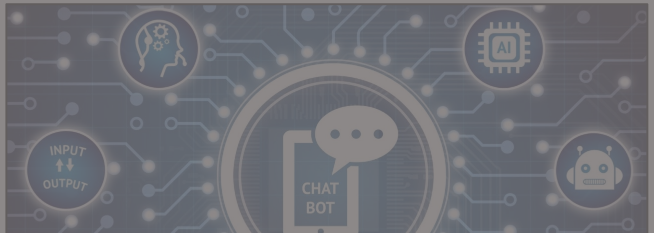
$A = \text{softmax}(QK^T)$ V

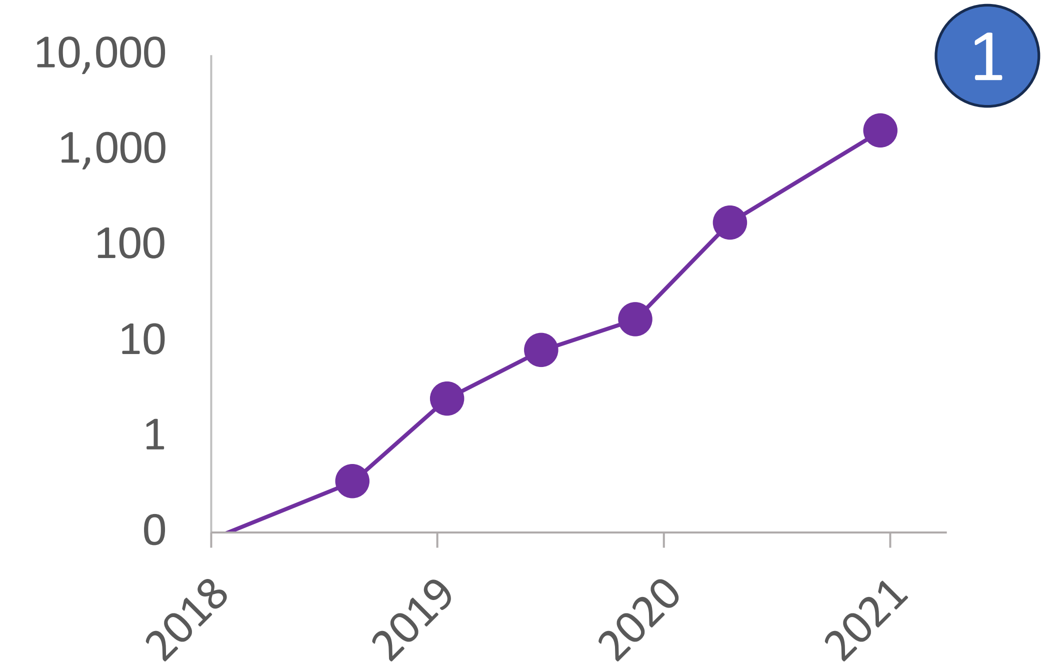
MLP



W_1 W_2

LLMs are Powerful , but Very Expensive to Deploy

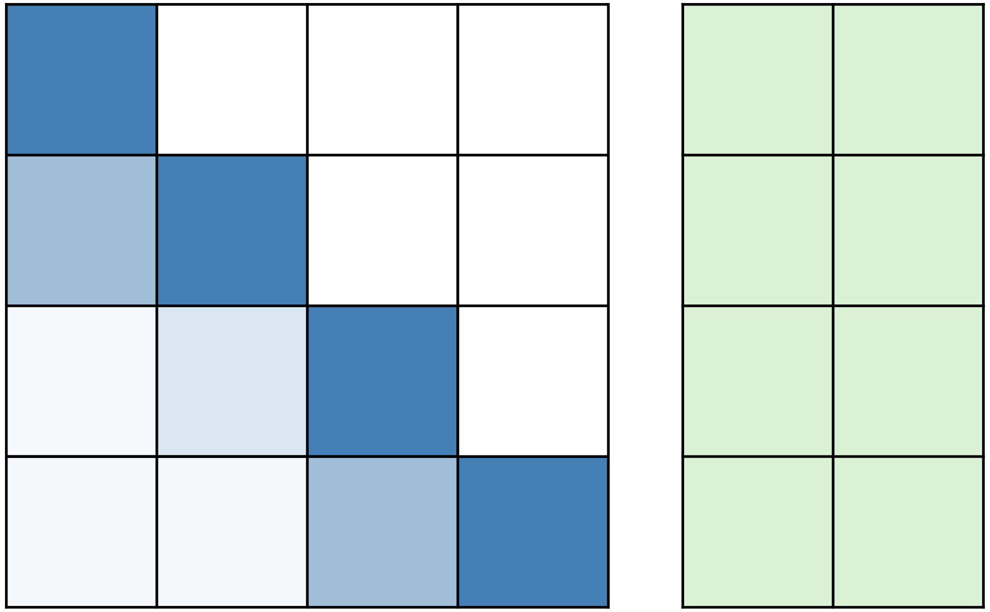




Exponential model size

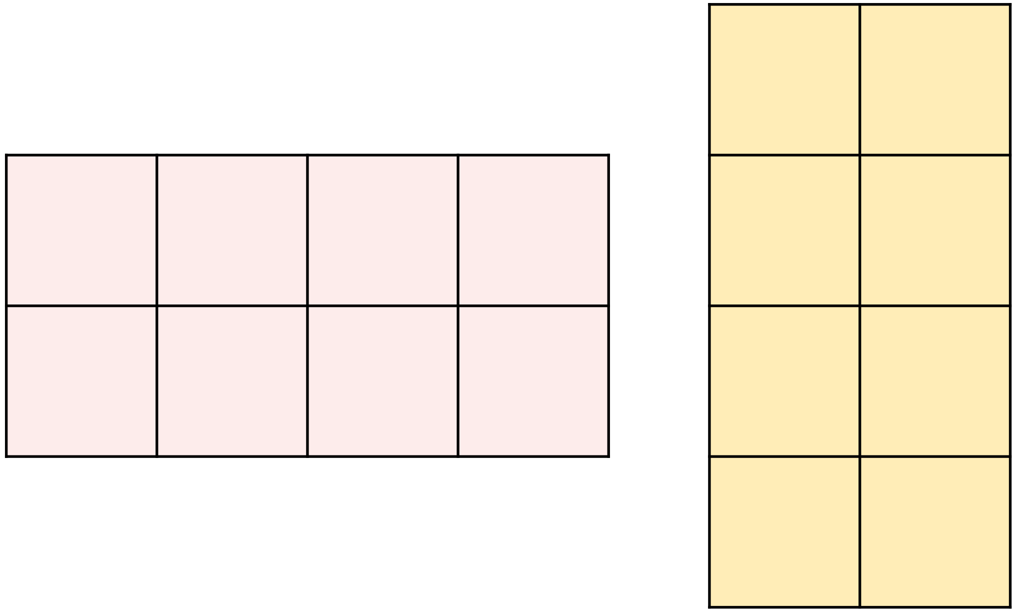
Background: Transformer Architecture

Attention



$$\{W_q, W_k, W_v, W_o\} \in R^{d\mathrm{x}d}$$

MLP

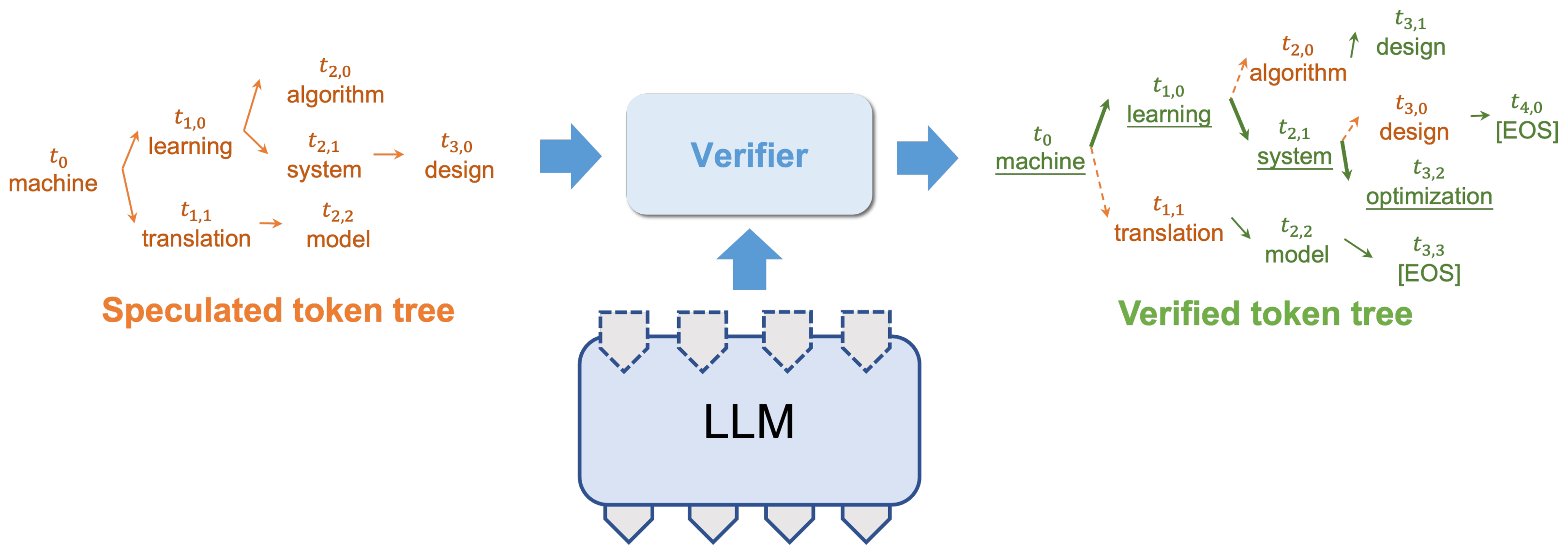


$$\{W_1, W_2\} \in R^{d\mathrm{x}4d}$$

Existing Approaches and Challenges

The idea of speculative decoding to accelerate LLM inference while preserving the LLM's output distribution has been widely studied!

- (*Chen et al 2023, Leviathan et al 2023*, *SpecInfer, SpecTr, …*)



*SpecInfer

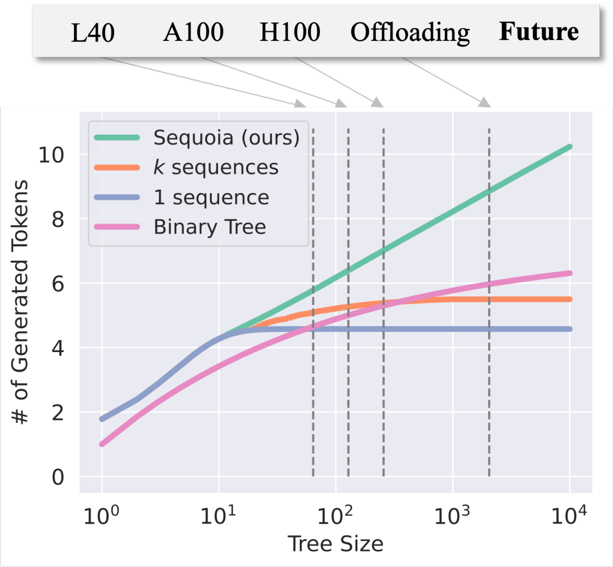
Existing Approaches and Challenges

The idea of speculative decoding to accelerate LLM inference while preserving the LLM's output distribution has been widely studied!

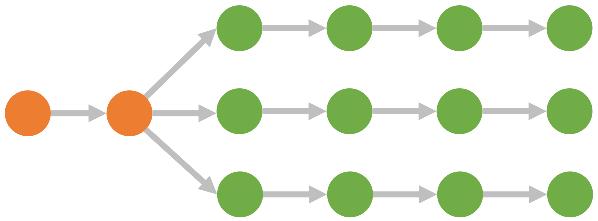
- (*Chen et al 2023, Leviathan et al 2023*, *SpecInfer, SpecTr, …*)

But hard to consistently and drastically speed up LLM Inference

- token tree construction algorithms do not scale with larger speculation budget



Single sequence of tokens



k independent sequences of tokens

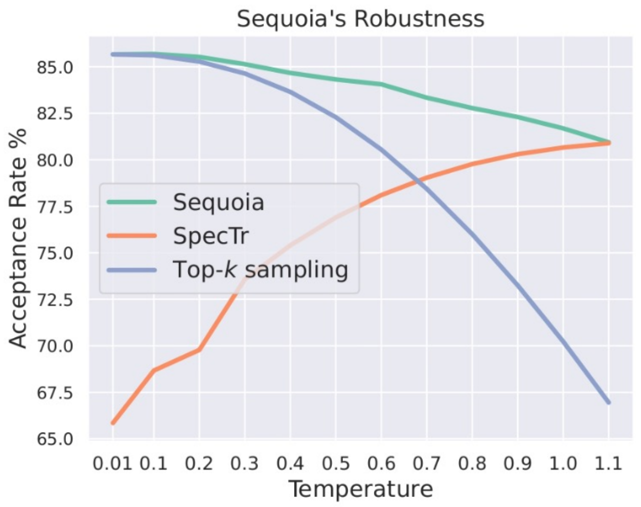
Existing Approaches and Challenges

The idea of speculative decoding to accelerate LLM inference while preserving the LLM's output distribution has been widely studied!

- (*Chen et al 2023, Leviathan et al 2023*, *SpecInfer, SpecTr, …*)

But hard to consistently and drastically speed up LLM Inference

- token tree construction algorithms do not scale with larger speculation budget
- token tree sampling and verification algorithms are not robust across different hyperparameter configuration



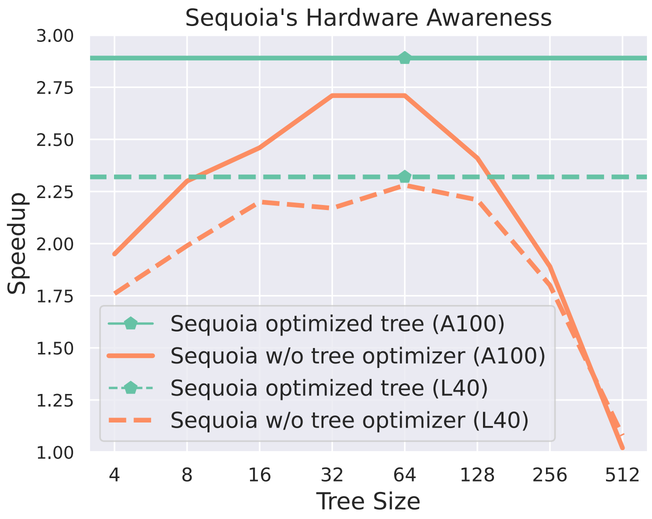
Existing Approaches and Challenges

The idea of speculative decoding to accelerate LLM inference while preserving the LLM's output distribution has been widely studied!

- (*Chen et al 2023, Leviathan et al 2023*, *SpecInfer, SpecTr, …*)

But hard to consistently and drastically speed up LLM Inference

- token tree construction algorithms do not scale with larger speculation budget
- token tree sampling and verification algorithms are not robust across different hyperparameter configuration
- Frameworks are not hardware-aware



Existing Approaches and Challenges

The idea of speculative decoding to accelerate LLM inference while preserving the LLM's output distribution has been widely studied!

- (*Chen et al 2023, Leviathan et al 2023*, *SpecInfer, SpecTr, …*)

But hard to consistently and drastically speed up LLM Inference

- token tree construction algorithms do not scale with larger speculation budget
- token tree sampling and verification algorithms are not robust across different hyperparameter configuration
- Frameworks are not hardware-aware

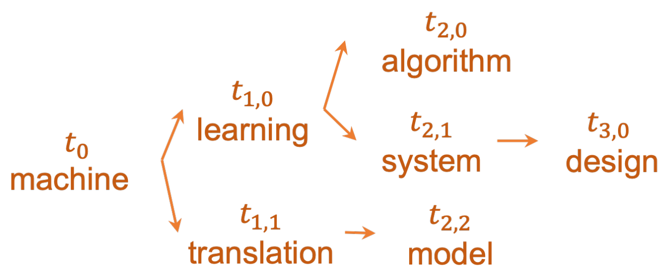
How can we design an optimal tree-based speculative decoding method to maximize speedups on modern hardware? Sequoia!

Scalable: Optimal Tree Construction

Sequoia tree construction algorithm: (1) formulate it as a constrained optimization problem, (2) use dynamic programming to solve this problem optimally and efficiently.

Maximize the expected number of tokens F (T) generated by verifying a token tree T , under a constraint on the size of T.

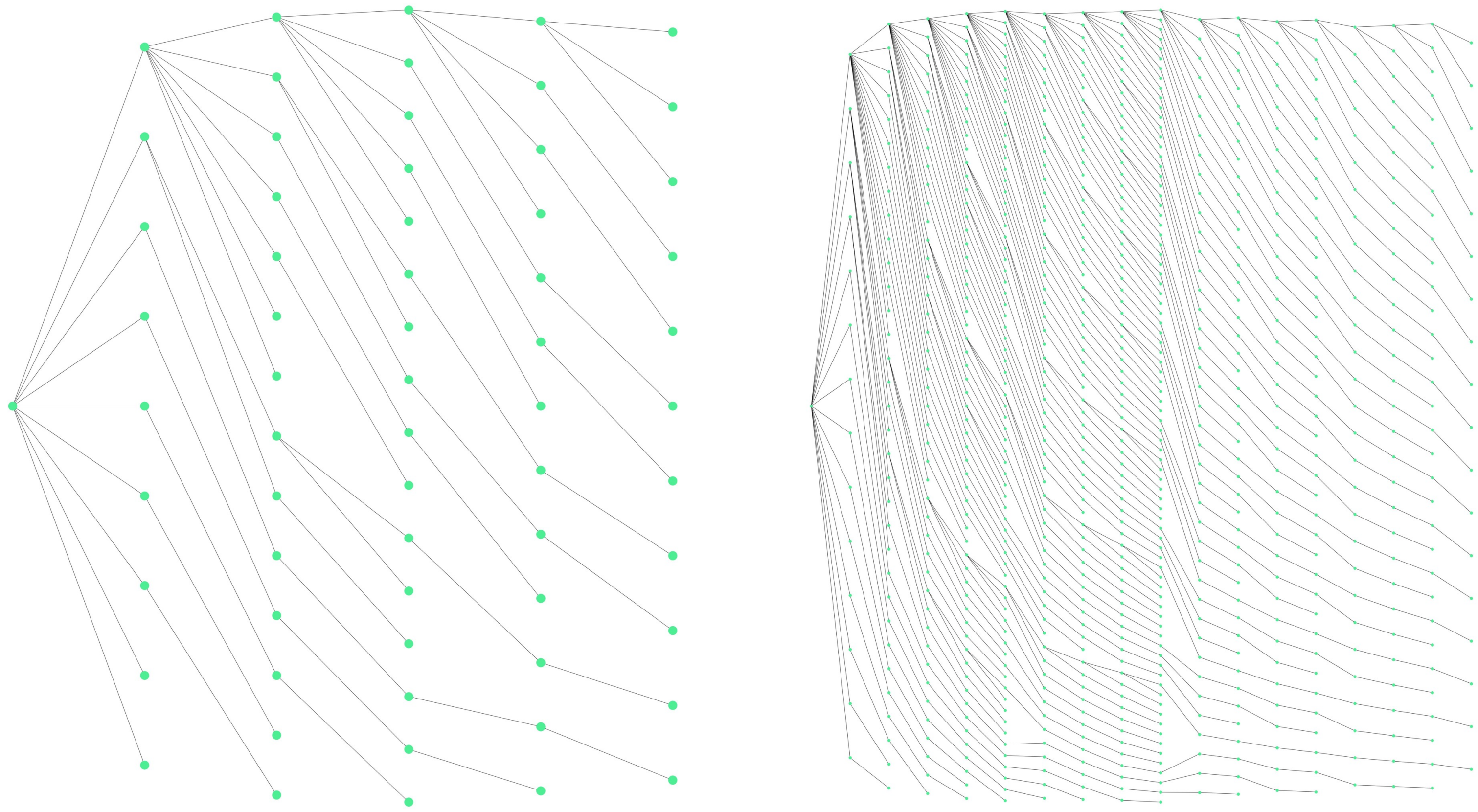
Intuition:
We should not expand all the branches with the same probability because they usually have very different chance being accepted!



Speculated token tree

Scalable: Optimal Tree Construction

Sequoia tree construction algorithm: (1) formulate it as a constrained optimization problem, (2) use dynamic programming to solve this problem optimally and efficiently.

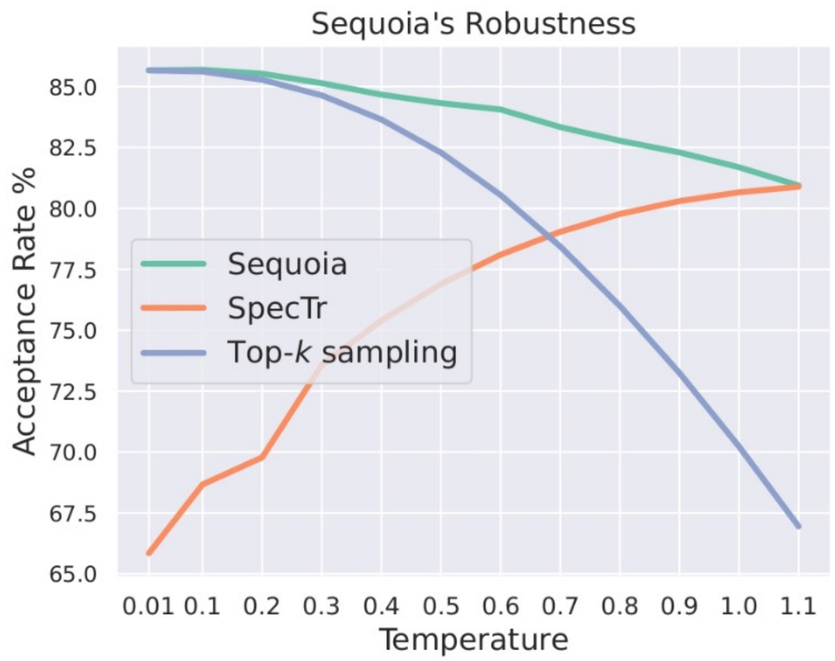


Robust: Sampling without Replacement

Sequoia sampling and verification algorithm: sample without replacement from the same draft model.

Intuition:
(i) Low-temperature, sample with replacement will likely to sample the same token. If being rejected, budgets wasted!

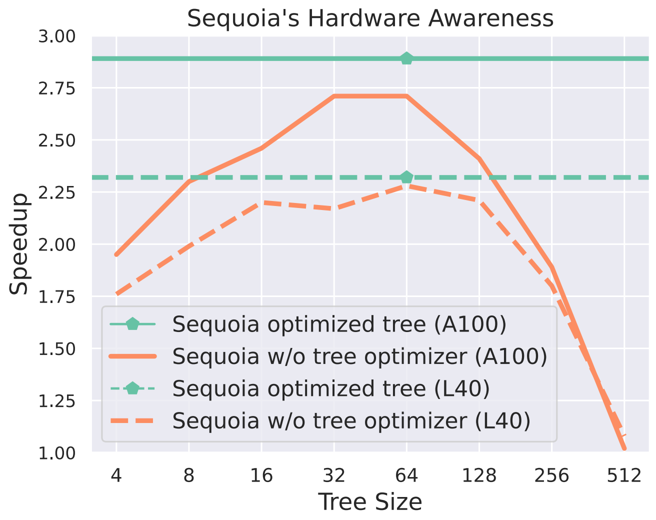
(ii) High-temperature, top-k sampling will have little chance getting exactly the same token as target model.

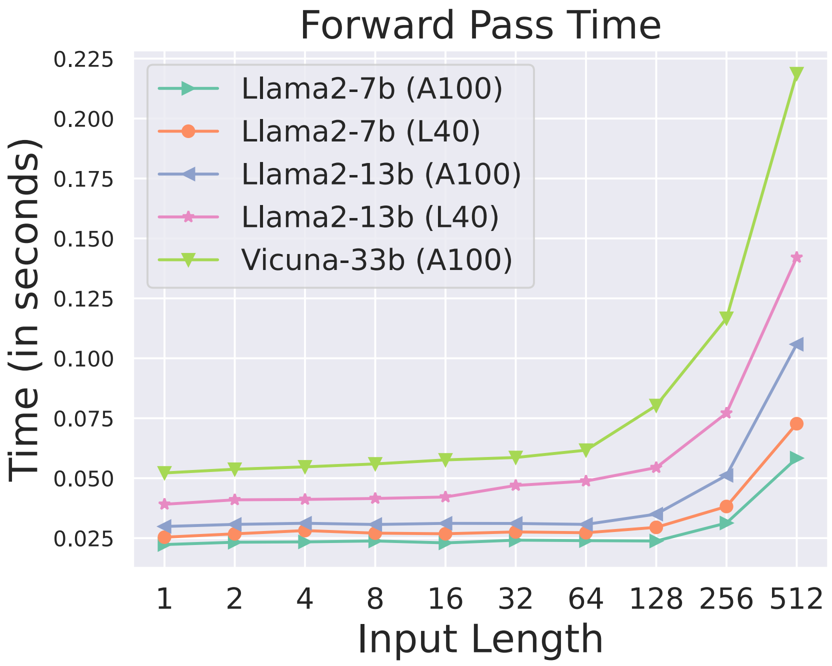


Hardware-aware: Tree Optimizer

Sequoia hardware-aware tree optimizer: search for optimal tree shape and depth.

Intuition: Turning point is different for different model size and hardware.





Sequoia: 9X DeepSpeed-Zero-Inference on RTX4090

GPU	Bandwidth(GB/s)	Target Model	Draft Model	TBT(s)	Baseline(s)
4090	31.5	Llama2-70B	Llama2-7B	0.57	4.54
4090	31.5	Vicuna-33B	TinyVicuna-1B	0.35	1.78
4090	31.5	Llama2-22B	TinyLlama-1.1B	0.17	0.95
4090	31.5	InternLM-20B	InternLM-7B	0.17	0.77
4090	31.5	Llama2-13B	TinyLlama-1.1B	0.09	0.27
2080Ti	15.8	Vicuna-33B	TinyVicuna-1B	0.87	4.81
2080Ti	15.8	Llama2-22B	TinyLlama-1.1B	0.53	3.04
2080Ti	15.8	Llama2-13B	TinyLlama-1.1B	0.34	1.53

Sequoia, a speculative decoding framework that mitigates the gap in the memory hierarchy, adapts to any draft/target pairs and any AI accelerators.

Sequoia: 4.04x Speed up for Llama-7B on A100

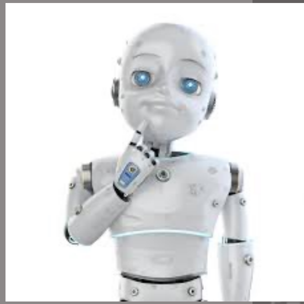
Target LLM	**Draft Model**	**T**	**Dataset**	**Tree Config. (size, depth)**	**Speedup**	**SpecInfer** 5×8	**SpecInfer** 8×8
Llama2-7B	JF68M	0	C4	(128,10)	**4.04** ×**(5.08)**	3.45×(3.96)	3.70×(4.11)
Llama2-7B	JF68M	0.6	C4	(128,7)	**3.18**×**(3.92)**	2.47×(2.97)	2.45×(3.05)
Llama2-7B	JF68M	0	OpenWebText	(128,7)	**3.22**×**(3.86)**	2.79×(3.15)	2.96×(3.24)
Llama2-7B	JF68M	0.6	OpenWebText	(128,6)	**2.71**×**(3.33)**	2.10×(2.54)	2.08×(2.55)
Llama2-7B	JF68M	0	CNN Daily	(128,7)	**3.41**×**(4.05)**	2.95×(3.27)	3.10×(3.37)
Llama2-7B	JF68M	0.6	CNN Daily	(128,6)	**2.83**×**(3.45)**	2.11×(2.58)	2.22×(2.69)
Llama2-13B	JF68M	0	C4	(64,9)	**3.73**×**(4.20)**	3.30×(3.64)	3.10×(3.75)
Llama2-13B	JF68M	0.6	C4	(64,7)	**3.19**×**(3.57)**	2.48×(2.87)	2.42×(3.00)
Llama2-13B	JF68M	0	OpenWebText	(64,7)	**3.18**×**(3.49)**	2.77×(3.05)	2.59×(3.14)
Llama2-13B	JF68M	0.6	OpenWebText	(64,6)	**2.77**×**(3.06)**	2.17×(2.49)	2.01×(2.52)
Llama2-13B	JF68M	0	CNN Daily	(64,7)	**3.33**×**(3.68)**	2.95×(3.22)	2.75×(3.32)
Llama2-13B	JF68M	0.6	CNN Daily	(64,6)	**2.88**×**(3.17)**	2.17×(2.54)	2.09×(2.60)
Llama2-13B	JF160M	0	C4	(64,7)	**3.10**×**(4.69)**	2.74×(4.33)	2.58×(4.42)
Llama2-13B	JF160M	0.6	C4	(64,6)	**2.83**×**(4.06)**	2.07×(3.46)	2.02×(3.53)
Llama2-13B	JF160M	0	OpenWebText	(64,6)	**2.72**×**(3.90)**	2.26×(3.58)	2.15×(3.66)
Llama2-13B	JF160M	0.6	OpenWebText	(64,5)	**2.49**×**(3.38)**	1.80×(2.96)	1.77×(3.07)
Llama2-13B	JF160M	0	CNN Daily	(64,6)	**2.84**×**(4.05)**	2.36×(3.73)	2.25×(3.83)
Llama2-13B	JF160M	0.6	CNN Daily	(64,5)	**2.55**×**(3.47)**	1.79×(2.97)	1.74×(3.03)
Vicuna-33B	SL1.3B	0	C4	(64,6)	**2.27**×**(4.28)**	1.83×(3.86)	1.73×(3.96)
Vicuna-33B	SL1.3B	0.6	C4	(64,6)	**2.19**×**(4.16)**	1.64×(3.53)	1.52×(3.56)
Vicuna-33B	SL1.3B	0	OpenWebText	(64,5)	**2.21**×**(3.93)**	1.75×(3.70)	1.65×(3.79)
Vicuna-33B	SL1.3B	0.6	OpenWebText	(64,5)	**2.13**×**(3.82)**	1.57×(3.36)	1.47×(3.43)
Vicuna-33B	SL1.3B	0	CNN Daily	(64,5)	**2.21**×**(3.93)**	1.75×(3.71)	1.65×(3.79)
Vicuna-33B	SL1.3B	0.6	CNN Daily	(64,5)	**2.16**×**(3.86)**	1.58×(3.40)	1.46×(3.43)

Sequoia demonstrates impressive on-chip performance -- up-to 4.04× speed-up for Llama2-7B on A100.

Demo

(llm) root@5112070c89a1:/Sequoia/tests#
bash run_offloading.sh
Loading checkpoint shards: 100%|█| 15/15
Loading checkpoint shards: 100%|█| 2/2 [
768
Loading data from dataset/mt_bench.jsonl
...
0%| | 0/4 [00:00<?, ?it/s]
[INST]Compose an engaging travel blog po
st about a recent trip to Hawaii, highli
ghting cultural experiences and must-see
attractions.[/INST]
ASSISTANT:
Generating
10.2 tokens/step
(llm) root@5112070c89a1:/Sequoia/tests#
bash run_baseline.sh
Loading checkpoint shards: 100%|█| 15/15
Loading data from dataset/mt_bench.jsonl
...
0%| | 0/4 [00:00<?, ?it/s]
[INST]Compose an engaging travel blog po
st about a recent trip to Hawaii, highli
ghting cultural experiences and must-see
attractions.[/INST]
ASSISTANT:
Generating
1 token/step

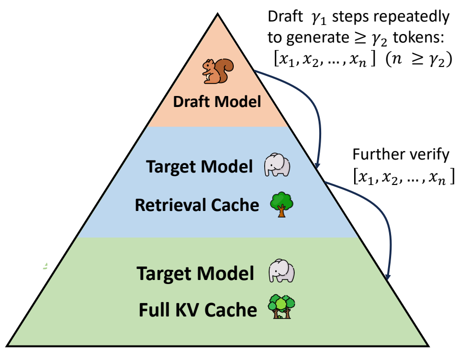
Demo

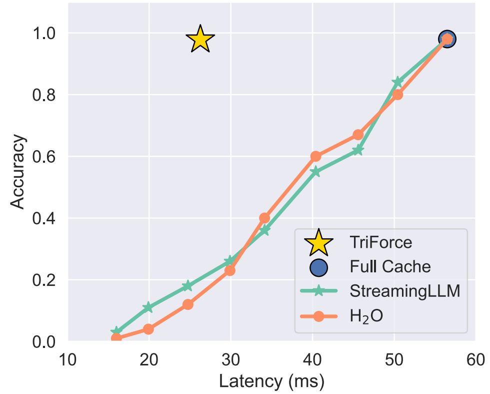


But how about long-context generation?

LLMs are Powerful , but Very Expensive to Deploy

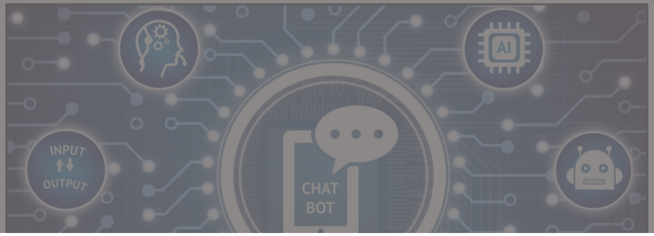
TriForce *(coming soon🔥)*

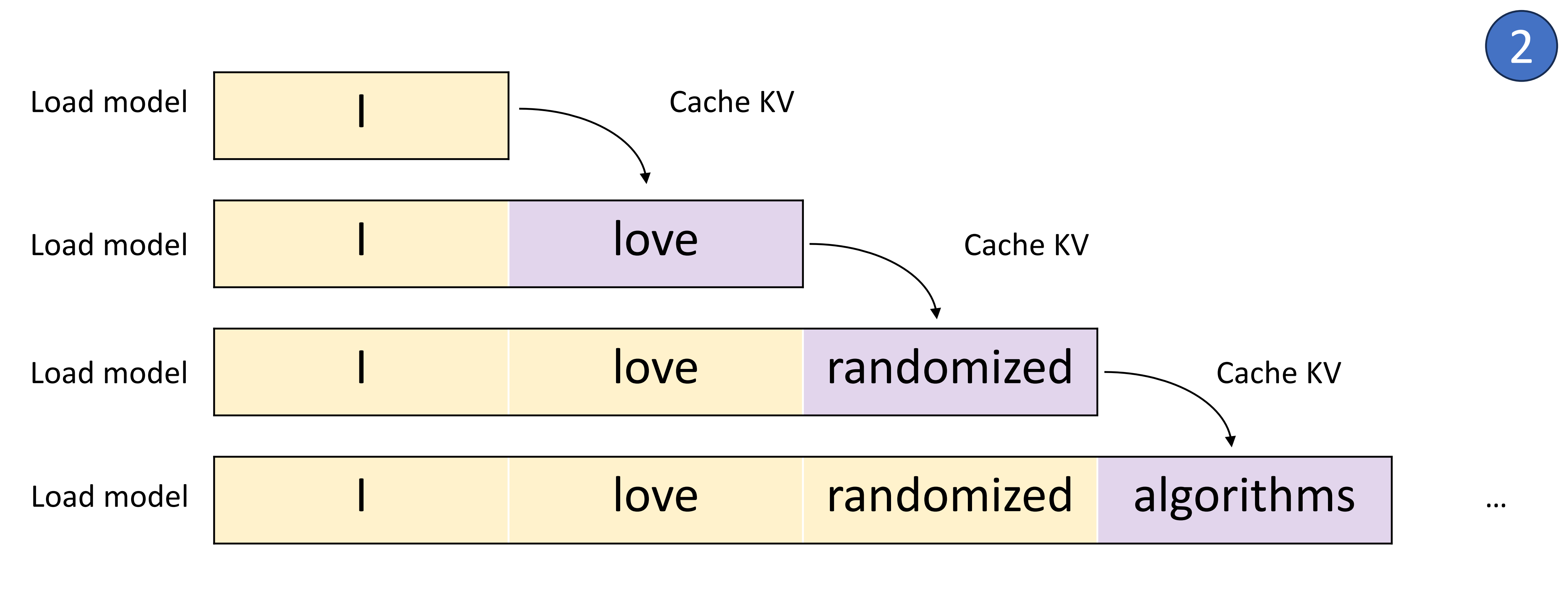




- Serve a Llama2-7B-128K (78GB mem) on a single RTX-4090 with 0.3s / token latency, 8× faster than DeepSpeed-Zero Offloading
- 2.3× speedup on a single A100 GPU

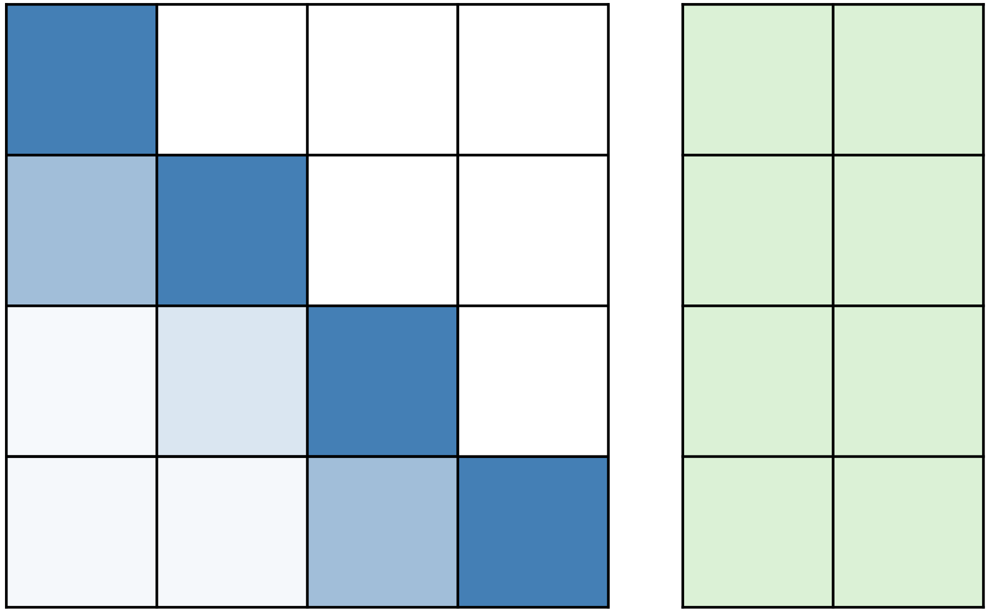
LLMs are Powerful , but Very Expensive to Deploy





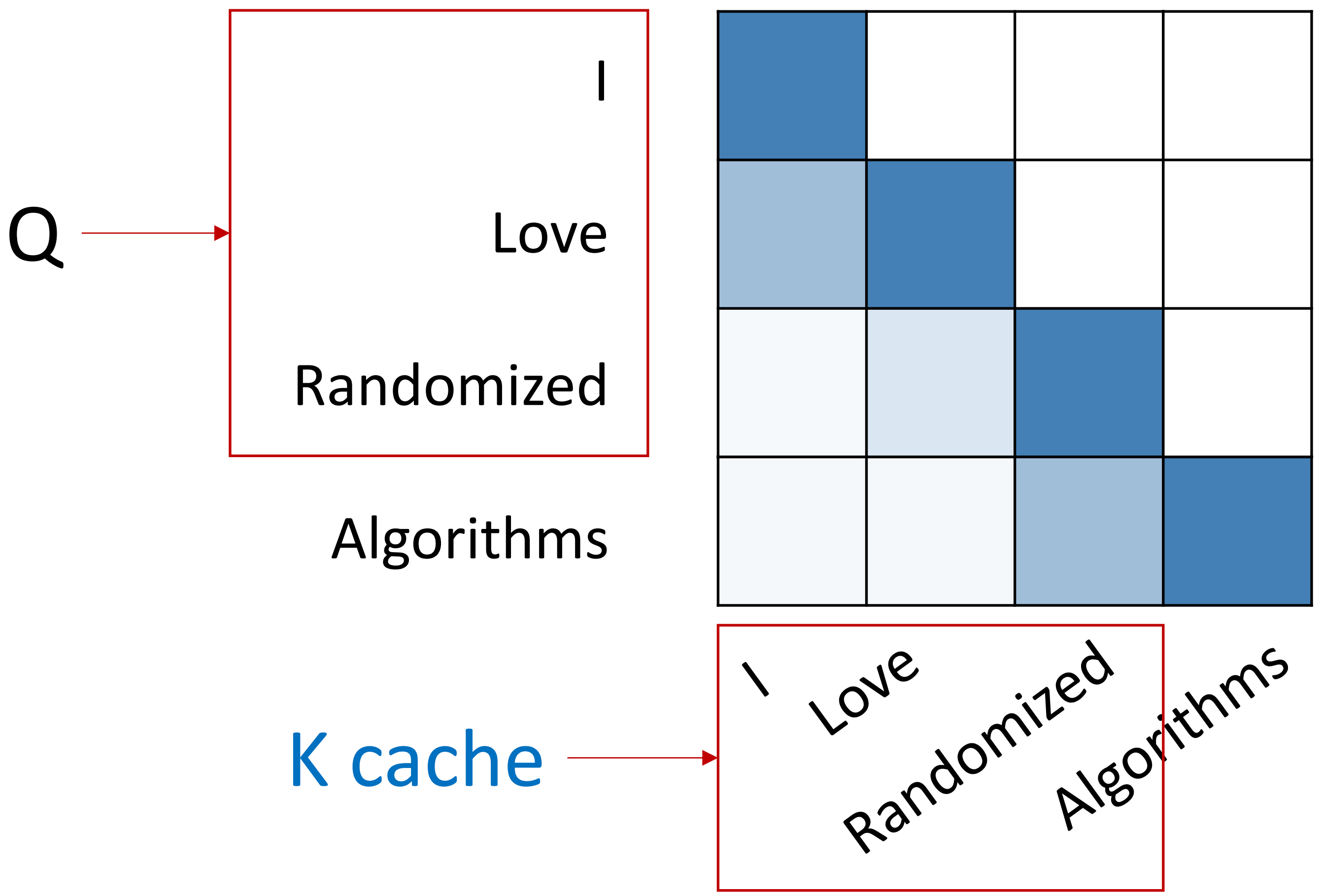
Background: Transformer Architecture

Attention



$A = \mathrm{softmax}(QK^T)$ V

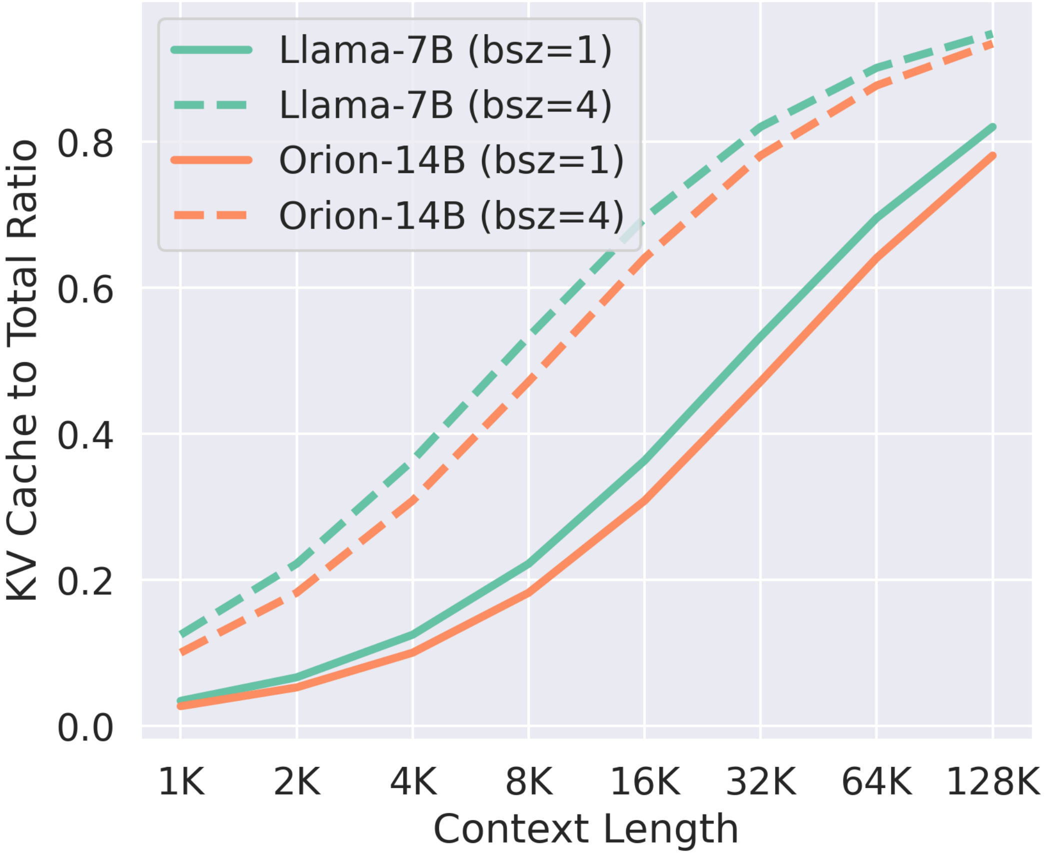
KV Cache Bottleneck



KV states for context or previously generated tokens will be cached to avoid re-computation.

KV cache size scales linearly with sequence length and batch size.

KV Cache Bottleneck

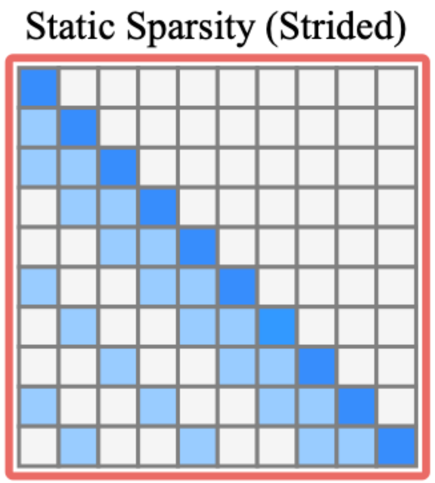


Existing Approaches and Challenges

Naturally, we can limit the cache size like the SW/HW caches. Attention approximation has been widely studied in training long sequences!

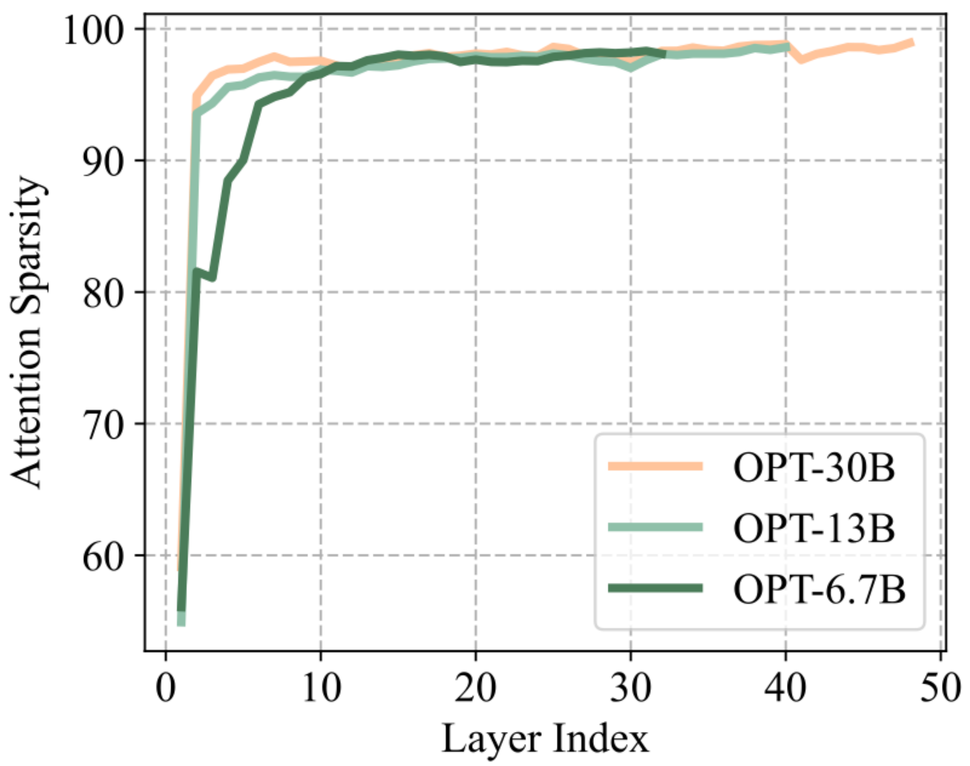
But hard to adapt to generation:

- Reduce quadratic attention but not KV cache size
  - e.g., FlashAttention, Reformer
- Result high cache miss rates and degrade accuracy
  - e.g., Sparse Transformer
- Expensive eviction policy
  - e.g., Gisting Tokens



An ideal cache has a small cache size, a low miss rate, and a low-cost eviction policy.

Sparsity for Smaller Cache Size



Observation: although densely trained, LLMs

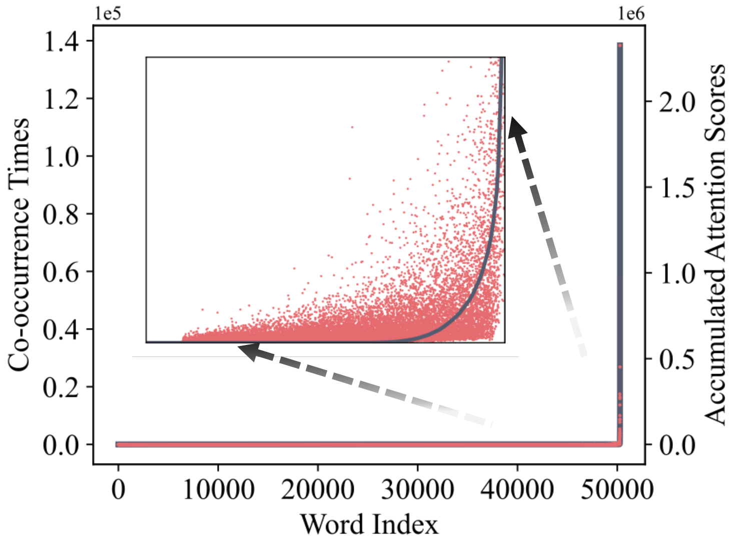
- attention score matrices are highly sparse, with a sparsity over 95% in almost all layers
- leads to 20× potential KV cache reduction
- maintains same accuracy

Attention sparsity widely exists in pre-trained models, e.g. OPT /LLaMA /Bloom/GPT.

H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. *NeurIPS 2023*. Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, Beidi Chen.

Heavy-Hitters for Low Miss Rate

Challenge: how to evict tokens? Once evicted, future tokens can no longer attend to it



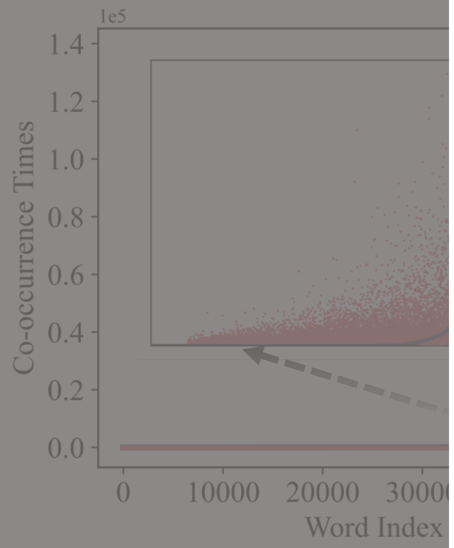
Key Observation: a small set of tokens are important along the generation

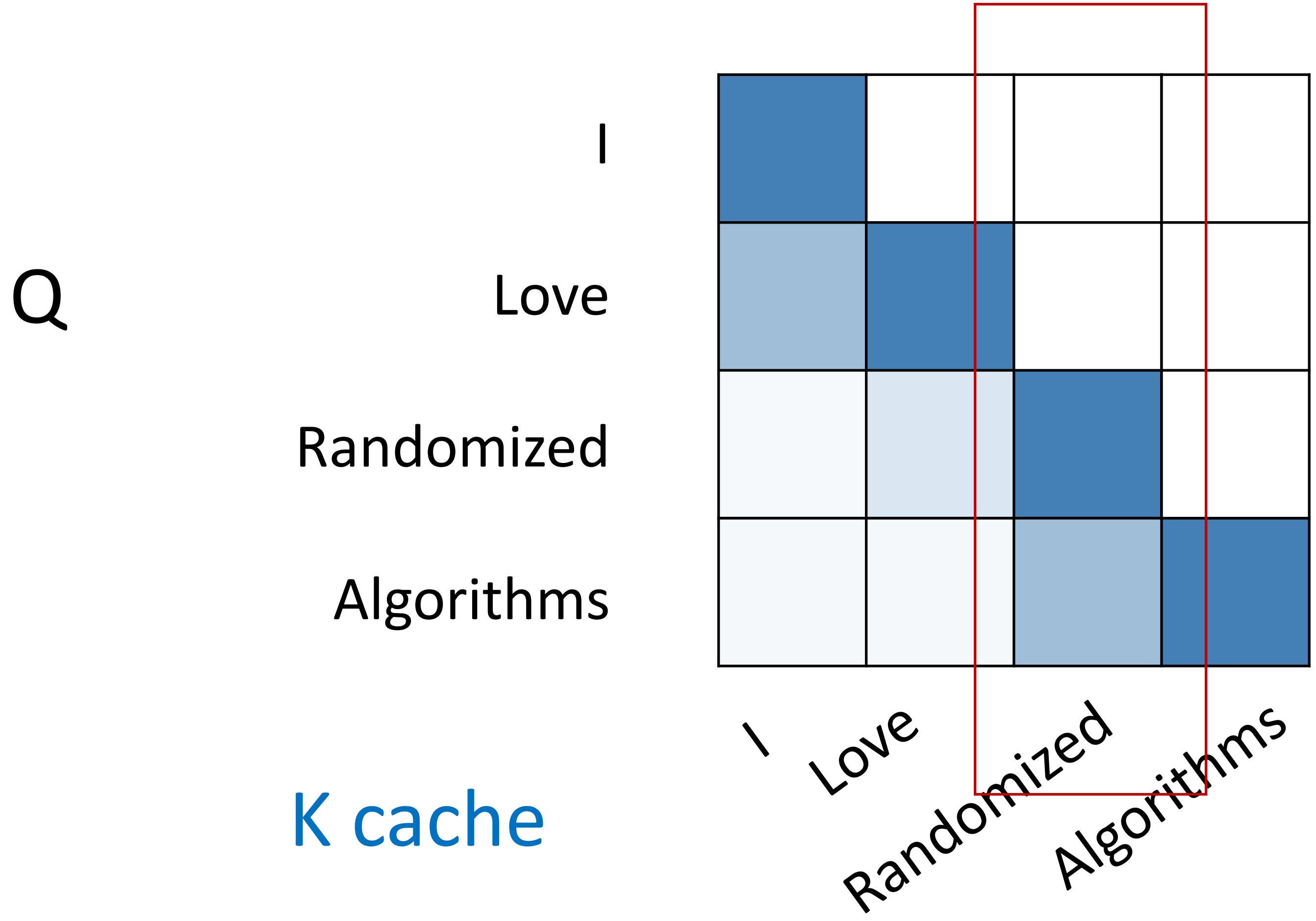
- accumulated attention scores of all the tokens follow a power-law distribution

H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. *NeurIPS 2023*. Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, Beidi Chen.

Heavy-Hitters for Low Miss Rate

Challenge: how to evict to



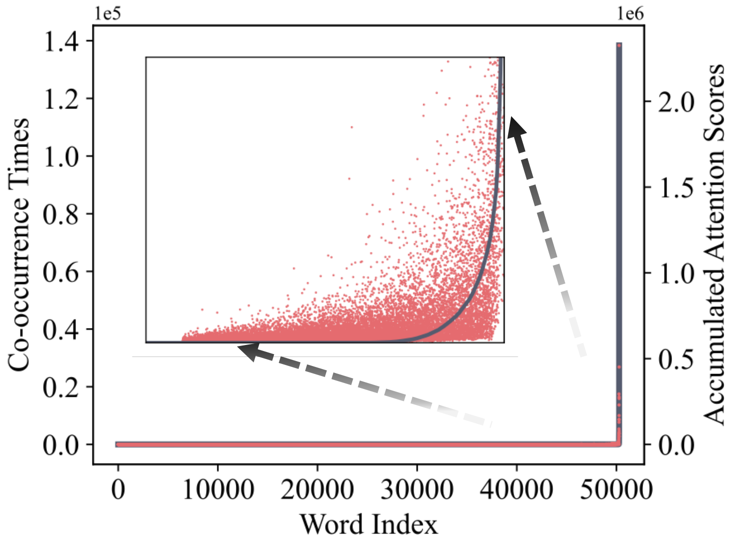


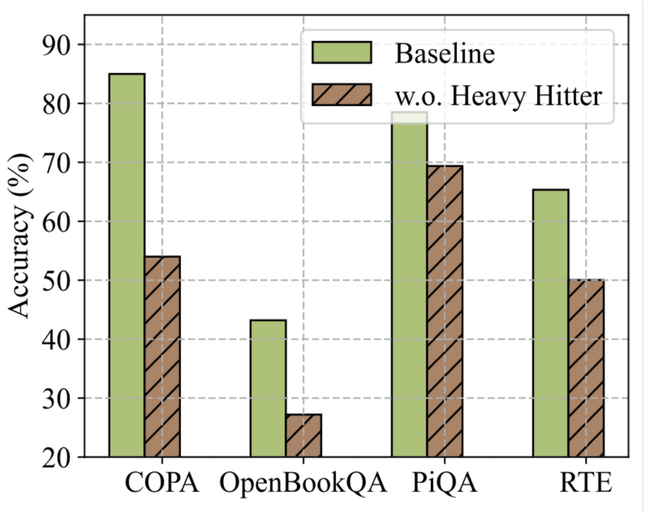
Key Observation: a small s

- accumulated attenti

Heavy-Hitters for Low Miss Rate

Challenge: how to evict tokens? Once evicted, future tokens can no longer attend to it



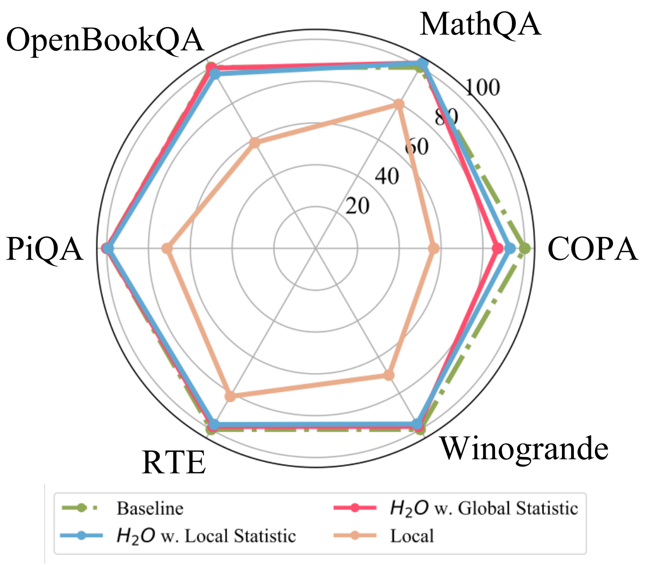


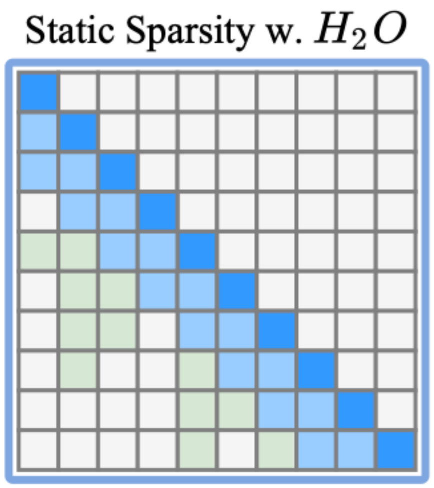
Key Observation: a small set of tokens are important along the generation

- accumulated attention scores of all the tokens follow a power-law distribution
- masking heavy-hitter tokens degrades model quality

Greedy Algorithm for Low-cost Policy

Challenge: how to deploy such algorithm without access to the full attention?

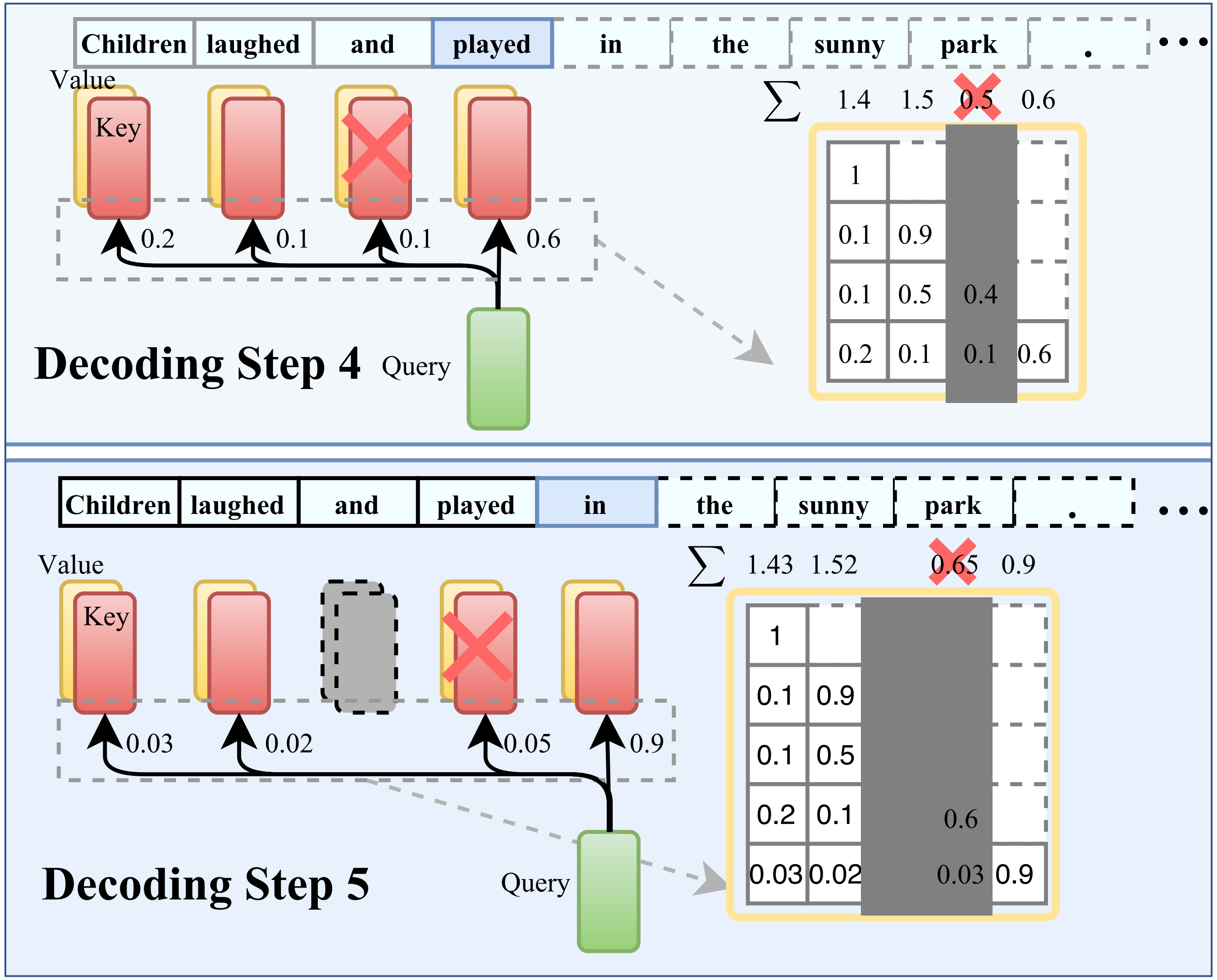




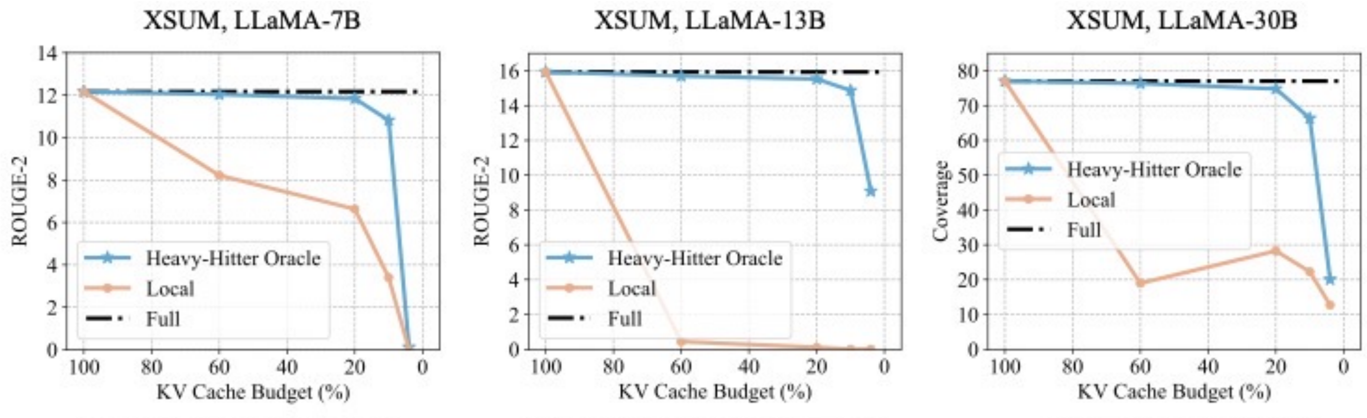
Idea: local greedy algorithm

- sum up the attention scores of the previous tokens every decoding step
- Add local / recent tokens

H_2O: Heavy Hitter Oracle



H_2O: 3-29X Throughput and 1.9X Latency



A100	FlexGen	H_2O
Throughput (token/s)	494	918 (1.9X)
Latency (s)	99	53 (1.9X)

	Hugging Face	Deep Speed	FlexGen	H_2O
Throughput (T4) token/s	0.6	0.6	8.5	18.83 (3-29X)

- compatible with quantization
- generate sentences with fewer repeated words and more creativity

Model Input

In a small, bustling cafe nestled in the heart of a vibrant city, a serendipitous event unfolded, leaving a lasting impression on all who witnessed it. As the patrons sat sipping their coffees and engaging in animated conversations, a talented street musician entered the cafe, carrying a weathered guitar and radiating an aura of creativity.

LLaMA-7B Full Cache Output

He began to play, and the patrons were captivated. The musician's performance was so moving that the patrons began to applaud, and the musician was so moved that he began to cry. The patrons were so moved that they began to cry, and the musician was so

LLaMA-7B Local 20% Cache Output

He ((((((((((((((, [)), 1999, 2000, 2001, 1, and, and, and, and, and, and, and, and, and, and, and,

LLaMA-7B H_2O 20% Cache Output

He began to play, and the room was filled with the sound of his music. The patrons of the cafe were enthralled by the music, and the atmosphere was electric. The cafe was packed with people, all of whom were enjoying the music. The musician was a young

Model Input

ling cafe nestled in the heart of a vibrant city, a serendipitous event unfolded, leaving a lasting
all who witnessed it. As the patrons sat sipping their coffees and engaging in animated conversations,
t musician entered the cafe, carrying a weathered guitar and radiating an aura of creativity.

What are these heavy hitters?

He began to play, and the patrons were captivated. The musician's performance was so moving that the patrons began to applaud, and the musician was so moved that he began to cry. The patrons were so moved that they began to cry, and the musician was so

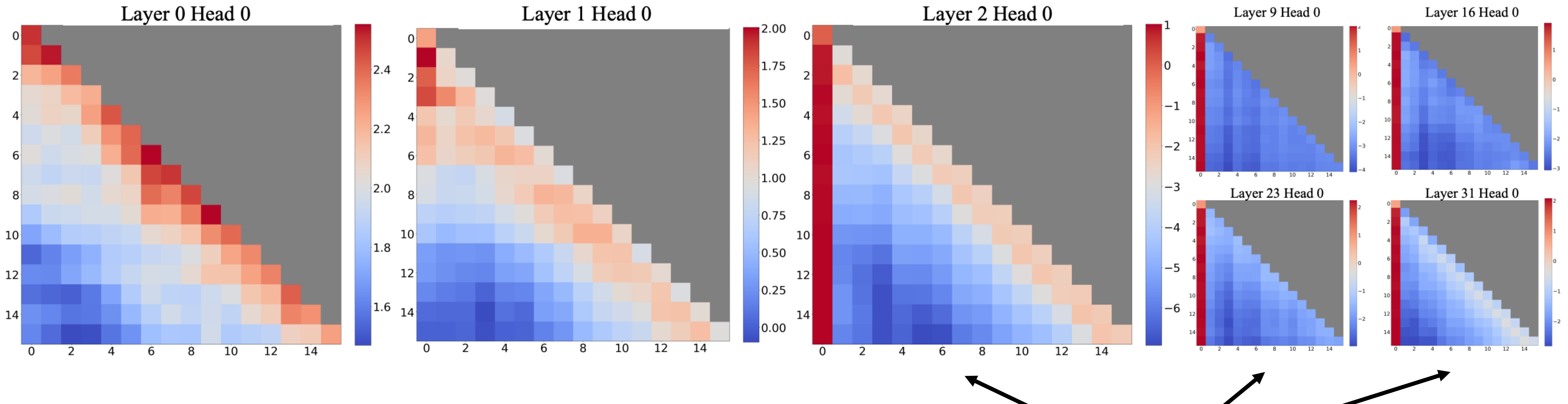
LLaMA-7B Local 20% Cache Output

He ((((((((((((((, [)), 1999, 2000, 2001, 1, and, and, and, and, and, and, and, and, and, and, and,

LLaMA-7B H_2O 20% Cache Output

He began to play, and the room was filled with the sound of his music. The patrons of the cafe were enthralled by the music, and the atmosphere was electric. The cafe was packed with people, all of whom were enjoying the music. The musician was a young

Phenomenon: Attention Sink



Average attention logits in Llama-2-7B over 256 sentences

First few tokens!

- Observation: large attention scores are given to initial tokens, even if they're not semantically significant.
- Attention Sink: Tokens that disproportionately attract attention irrespective of their relevance.

StreamingLLM: Efficient Streaming Language Models with Attention Sinks. Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, Mike Lewis.

Understanding Attention Sinks

- SoftMax operation's role in creating attention sinks — attention scores have to sum up to one for all contextual tokens. (*SoftMax-Off-by-One, Miller et al. 2023*)

$$\text{SoftMax}(x)_i = \frac{e^{x_i}}{e^{x_1} + \sum_{j=2}^{N} e^{x_j}}, \quad x_1 \gg x_j, j \in 2, \ldots, N$$

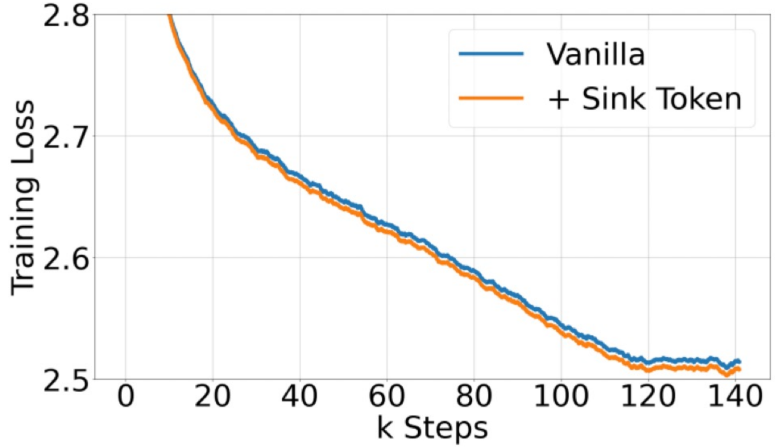
- Initial tokens' advantage in becoming sinks due to their visibility to subsequent tokens, rooted in autoregressive language modeling.

- The model learns a bias towards their absolute position rather than the semantics are crucial.

Llama-2-13B	PPL (↓)
0+1024 (window)	5158.07
4+1024	5.40
4"\n"+1020	5.6

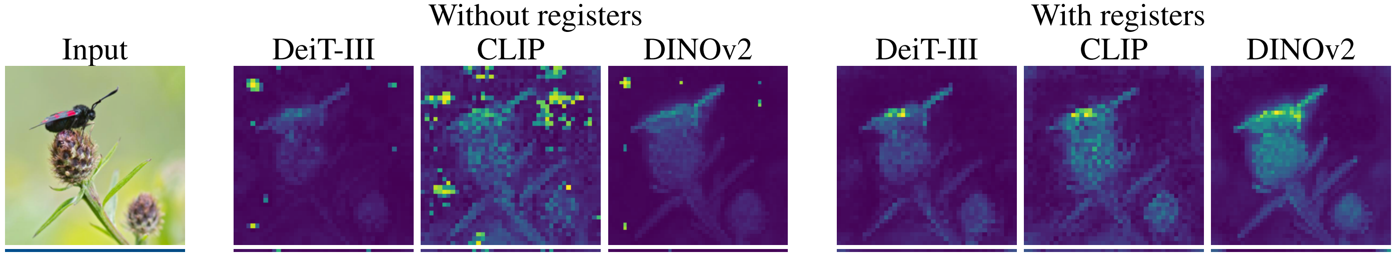
Understanding Attention Sinks

- Pre-train with a Dedicated Attention Sink Token

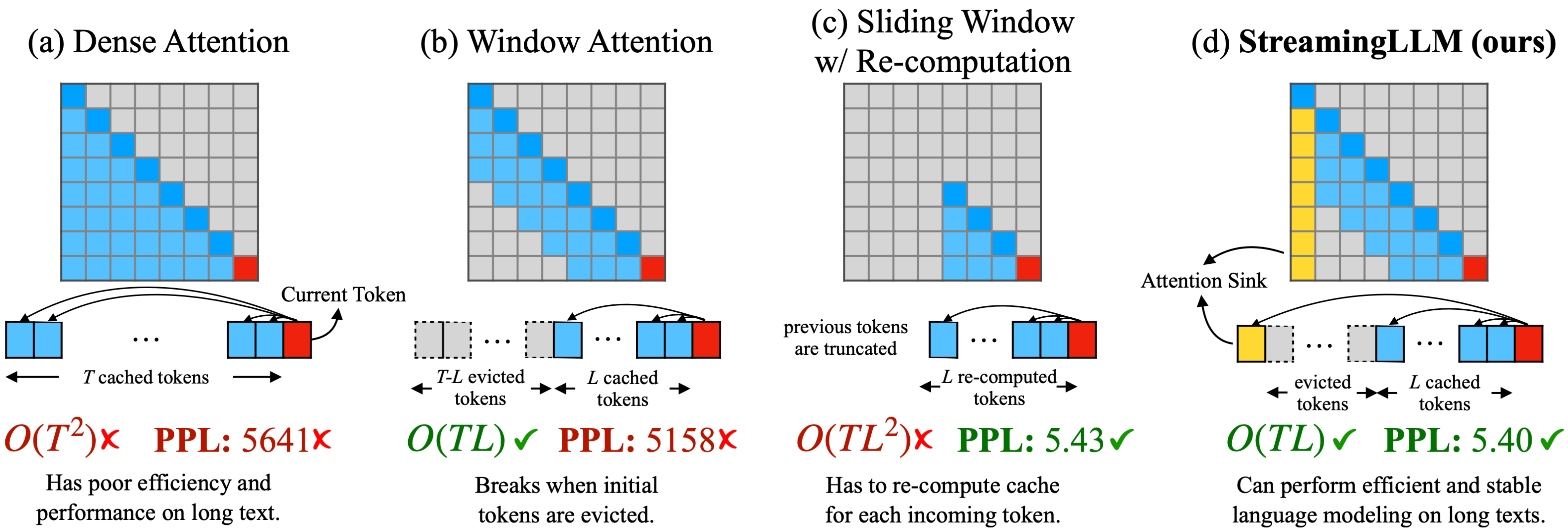


Cache Config	0+1024	1+1023	2+1022	4+1020
Vanilla	27.87	18.49	18.05	18.05
Zero Sink	29214	19.90	18.27	18.01
Learnable Sink	1235	**18.01**	18.01	18.02

- Similar Phenomenon in *Darcet et al. Vision transformers need registers*



StreamingLLM



StreamingLLM: Efficient Streaming Language Models with Attention Sinks. Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, Mike Lewis.

Infinite Streaming Ability

Urgent need for LLMs in streaming applications such as multi-round dialogues, where long interactions are needed.

Key challenge:

- Pre-trained model (e.g., LLaMA) cannot go beyond its pre-trained context window

Train:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**

Test:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**?**	**?**

Opportunity with StreamingLLM:

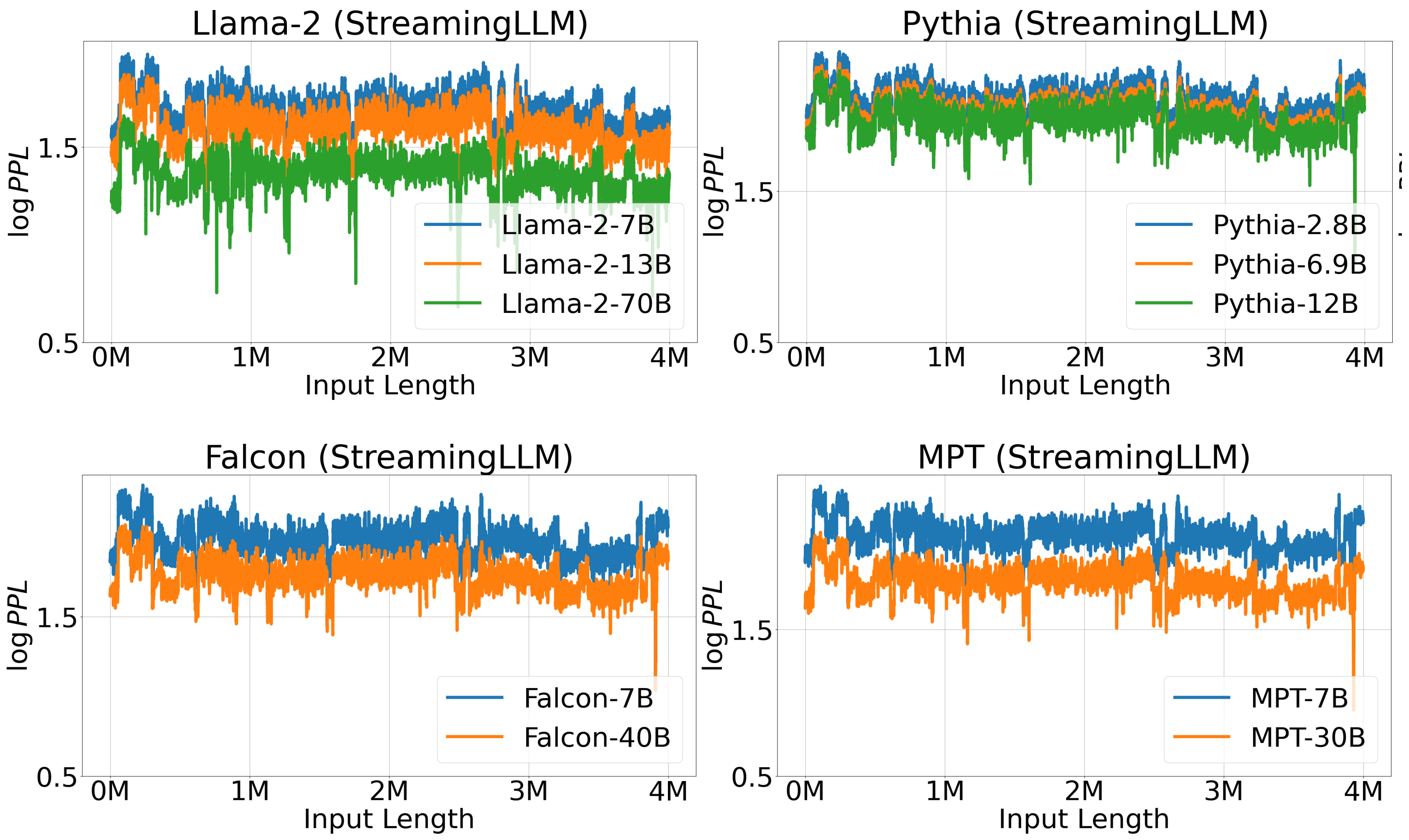
Train:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**

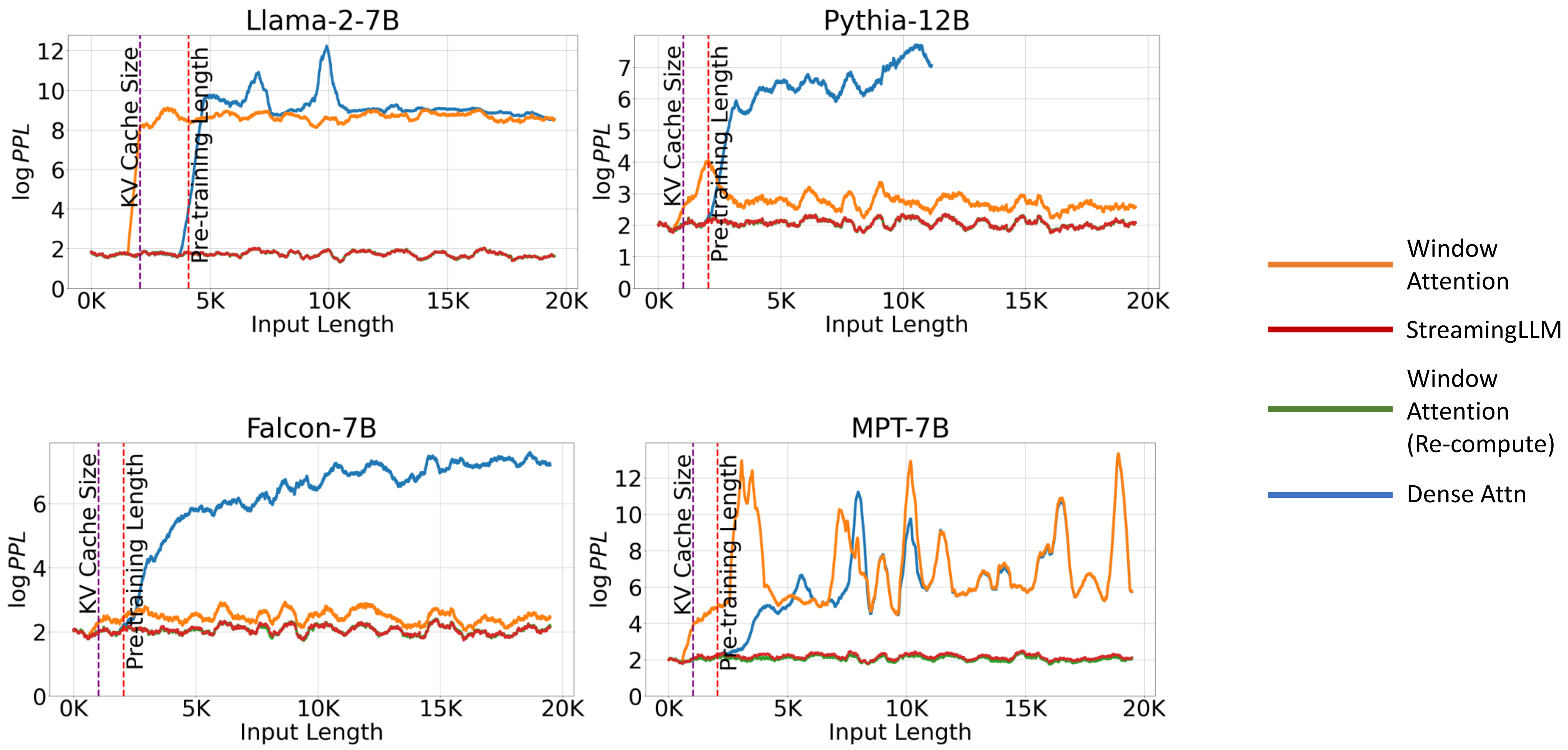
Test:

1	**2**	**3**	**4**	**x**	**x**	**5**	**6**	**7**	**8**

Stably Model up to 4 Million Tokens



22X Faster than Sliding Window Recomputation



Infinite Streaming Ability

Urge LLMs in streaming applications such as multi-round dialogues, where long intera needed.

But StreamingLLM will forget the middle contents?

Key challenge:

- Pre-trained model (e.g., LLaMA) cannot go beyond its pre-trained context window

Train: 1 2 3 4 5 6 7 8   Test: 1 2 3 4 5 6 7 8 ? ?

Opportunity with StreamingLLM:

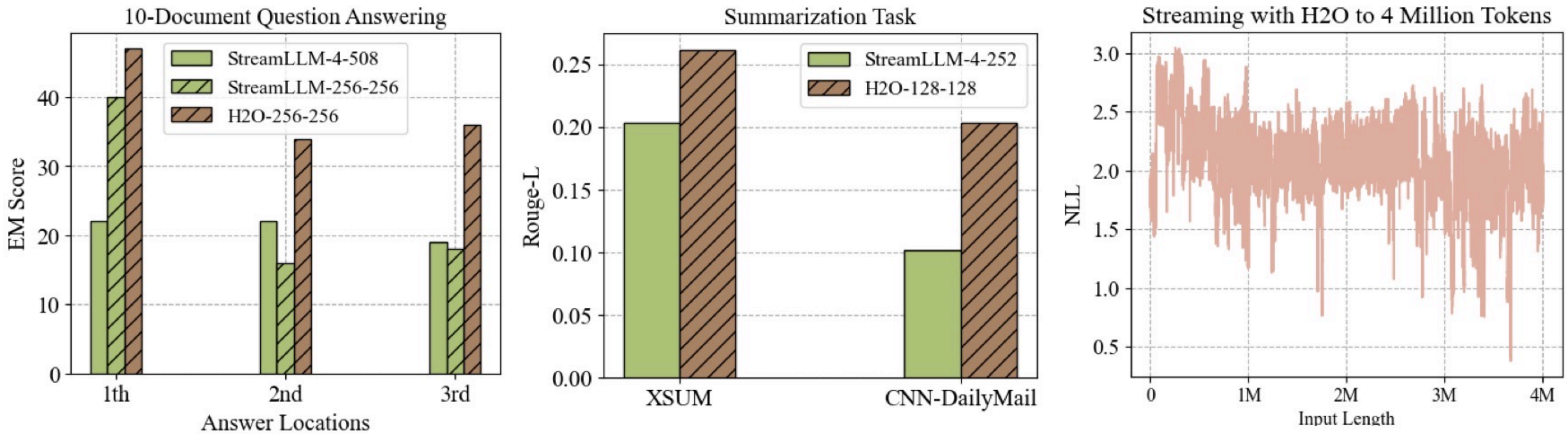
Train: 1 2 3 4 5 6 7 8   Test: 1 2 3 4 x x 5 6 7 8

The perplexity remains stable throughout up to 4 Million Tokens!

StreamingH2O: Infinite Streaming Ability

Similar position squeezing can be deployed on H2O

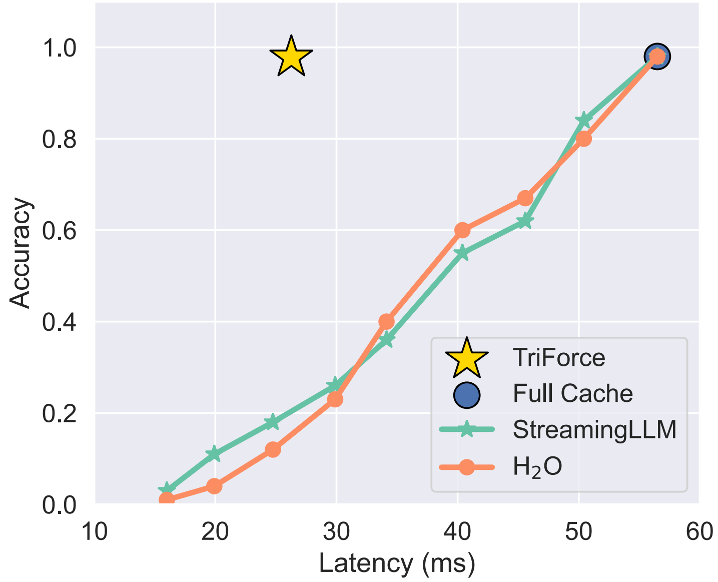
Train:	1	2	3	4	5	6	7	8		
Test:	1	x	2	x	3	4	5	6	7	8



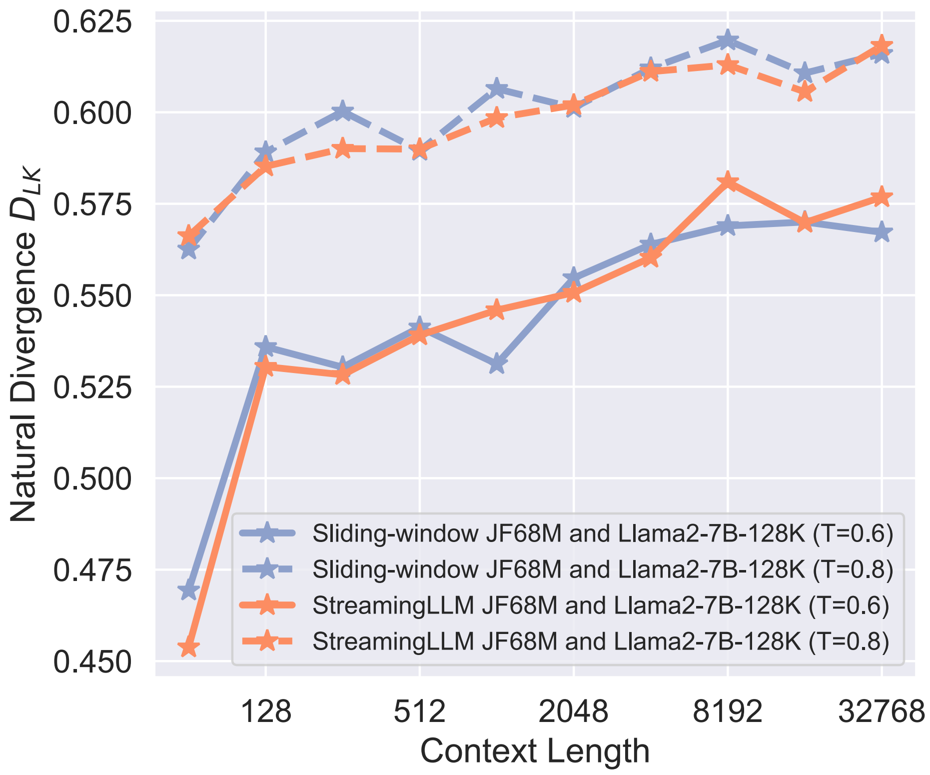
Existing Approaches and Challenges

But it is hard to know what we loose …
What if we compressed very important info?

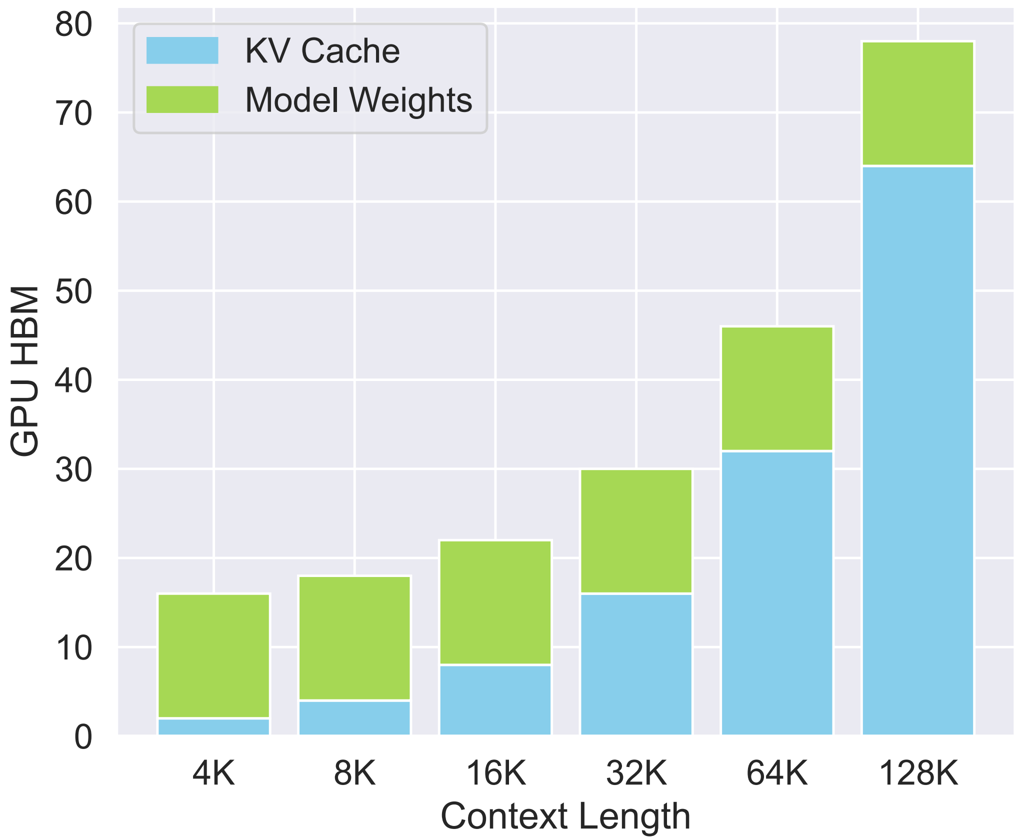
How about speculative decoding?
But training very long-context draft model sounds like a painful job …

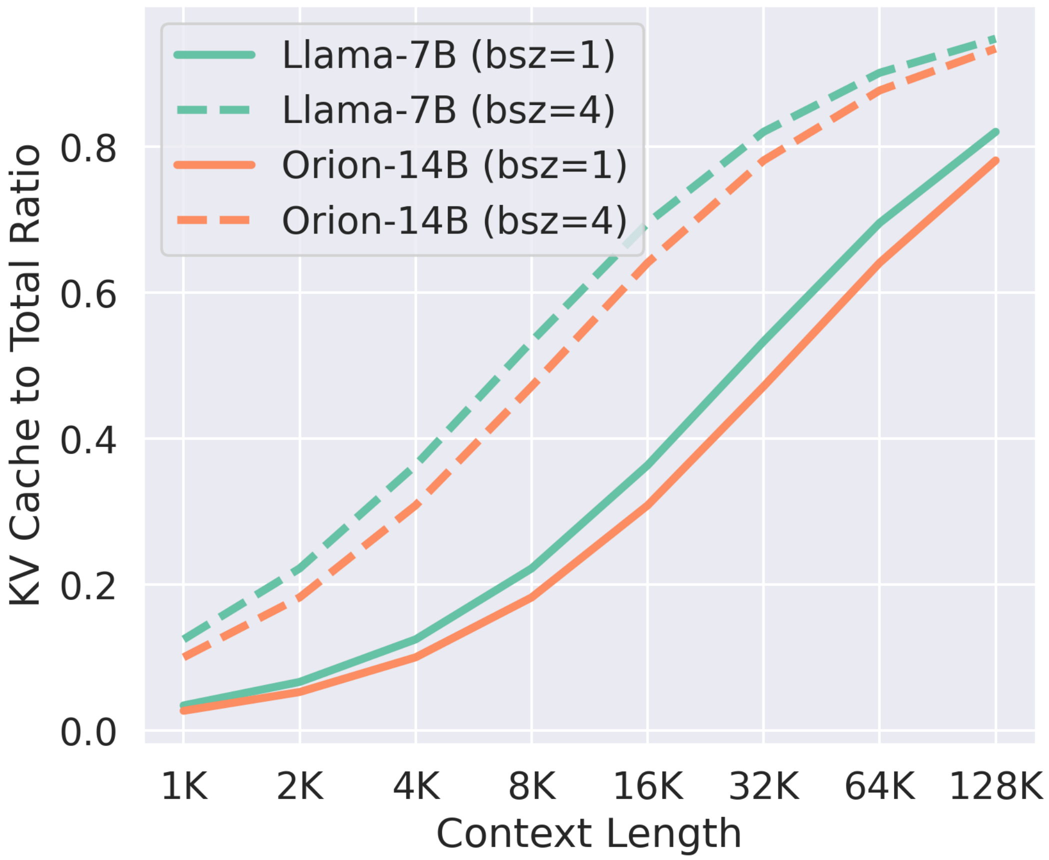


How about KV Compression + Speculative Decoding!

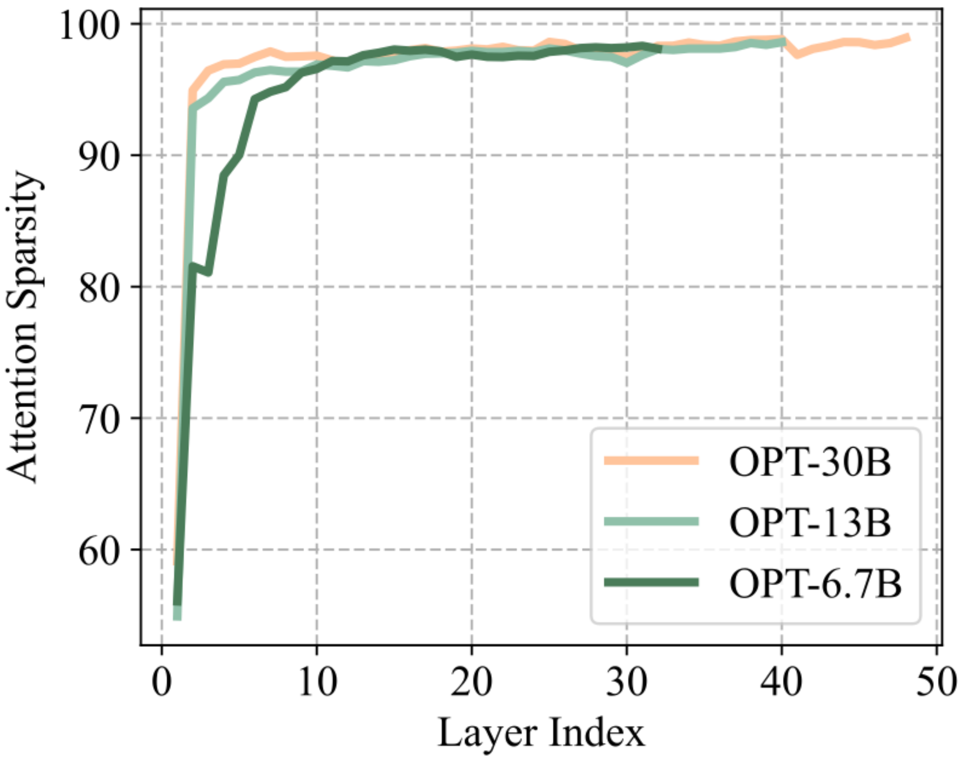


KV Cache Bottleneck





Sparsity for Smaller Cache Size



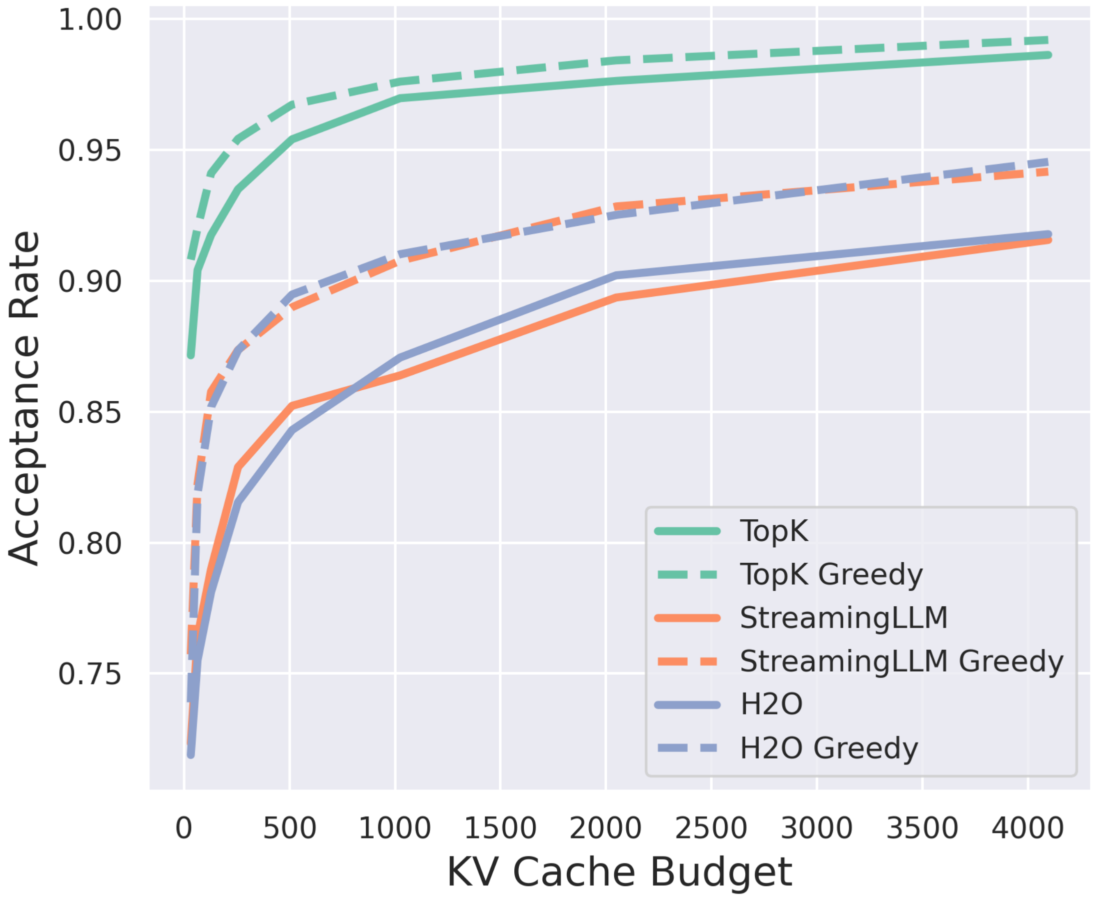
Observation: although densely trained, LLMs

- attention score matrices are highly sparse, with a sparsity over 95% in almost all layers
- leads to 20× potential KV cache reduction
- maintains same accuracy

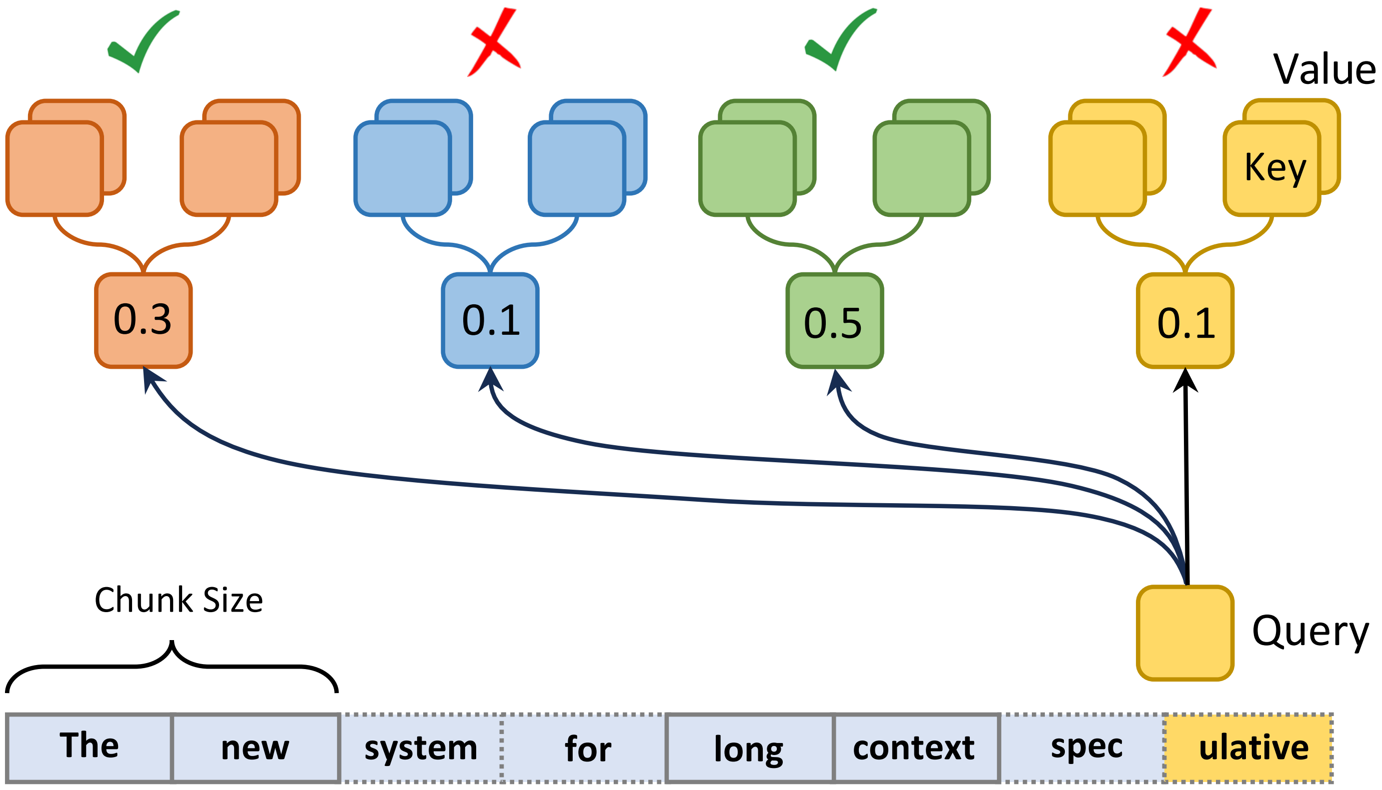
Attention sparsity widely exists in pre-trained models, e.g. OPT /LLaMA /Bloom/GPT.

H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. *NeurIPS 2023*. Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, Beidi Chen.

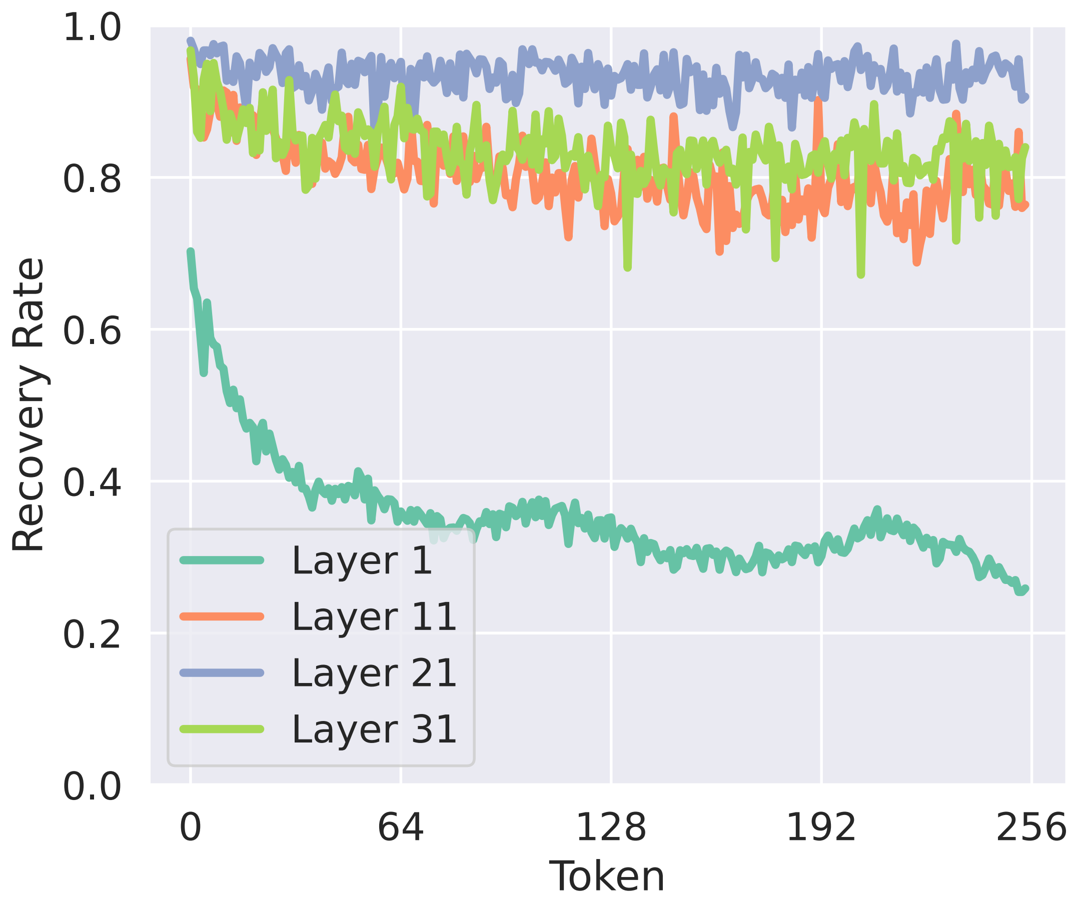
Target Models with Compressed KV as Their Own Drafts



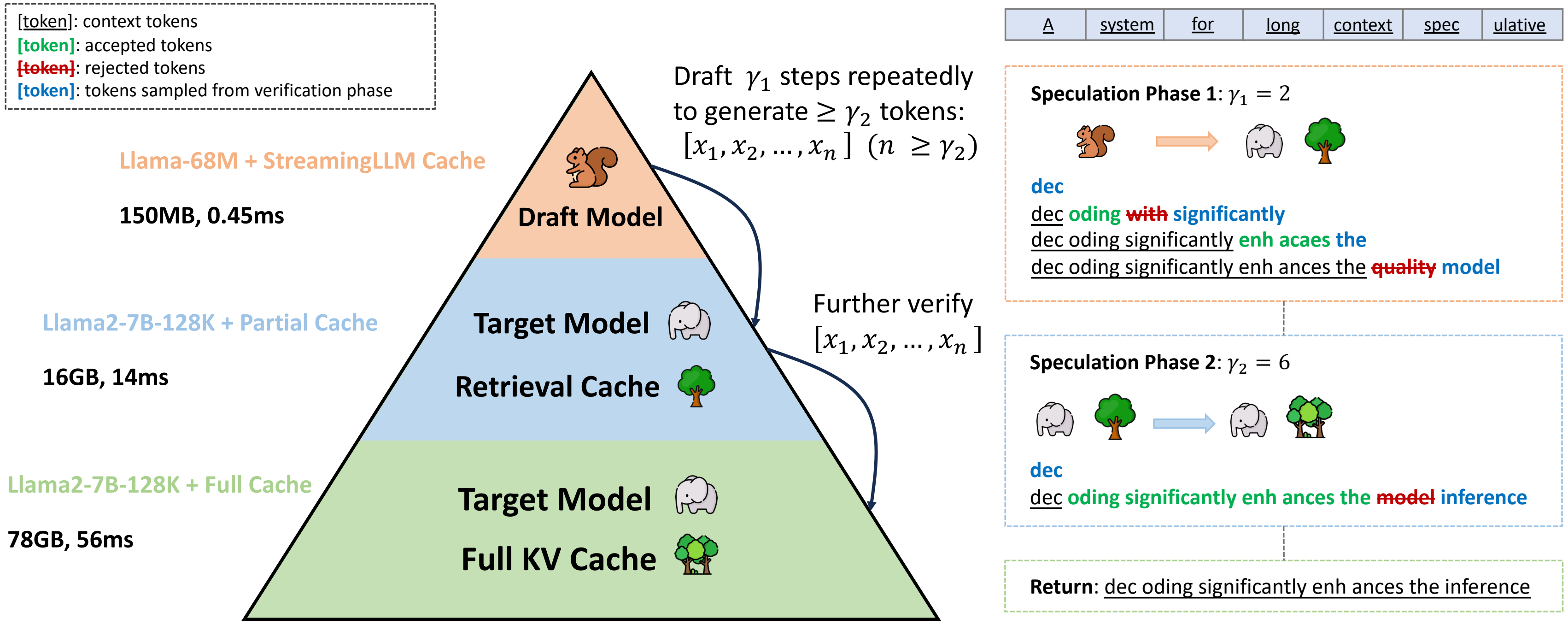
Better KV Compression: Retrieval-based



Contextual Locality for Drafting Efficiency



TriForce: Two Stage Speculation



Draft 68m + Constant KV ------> Draft 7B + Constant KV ------> 7B+128K

Serve Llama2-7B 128K 2.2X on A100

Method	T	Speedup	Naive Policy
TriForce	0.0	**2.31**× (0.9234)	1.56× (0.4649)
TriForce	0.2	**2.25**× (0.9203)	1.54× (0.4452)
TriForce	0.4	**2.20**× (0.9142)	1.47× (0.4256)
TriForce	0.6	**2.19**× (0.9137)	1.42× (0.4036)
TriForce	0.8	**2.08**× (0.8986)	1.34× (0.3131)
TriForce	1.0	**2.08**× (0.9004)	1.29× (0.2872)
TriForce	1.2	**2.02**× (0.8902)	1.27× (0.2664)
Retrieval w/o Hierarchy	0.6	1.80× (0.9126)	-
StreamingLLM w/ Hierarchy	0.6	1.90× (0.8745)	-

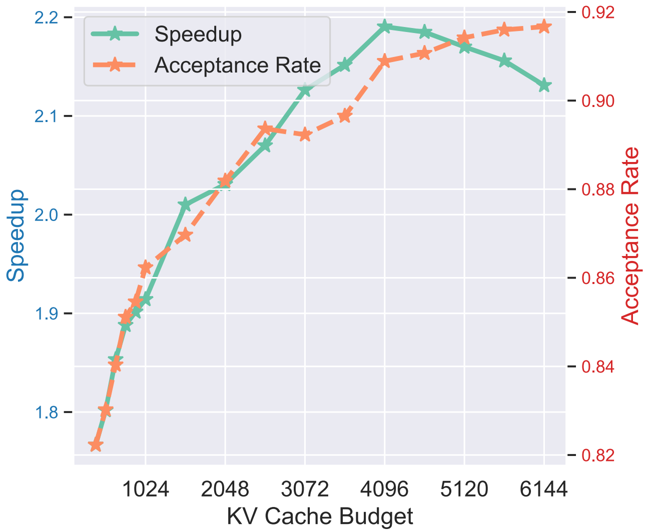
Serve Llama2-7B 0.3s / token Latency on an RTX4090

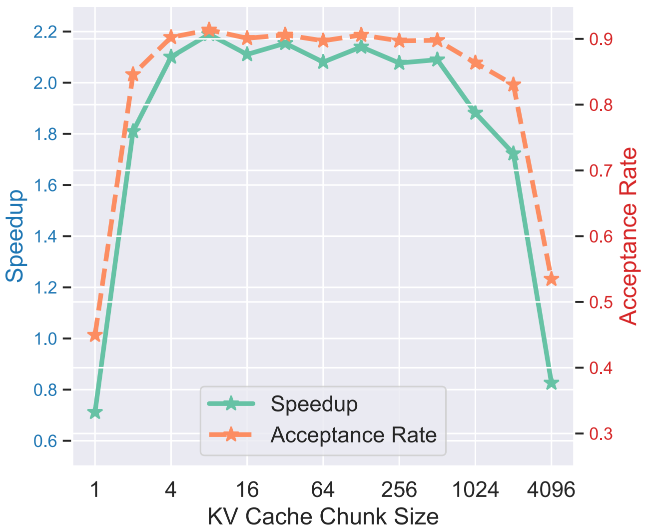
GPUs	Target Model	TriForce (ms)	AR (ms)	Speedup
$2\times$ RTX 4090s	Llama2-7B-128K	108	840	$7.78\times$
$2\times$ RTX 4090s	LWM-Text-Chat-128K	114	840	$7.37\times$
$2\times$ RTX 4090s	Llama2-13B-128K	226	1794	$7.94\times$
$1\times$ RTX 4090	Llama2-7B-128K	312	2434	$7.80\times$
$1\times$ RTX 4090	LWM-Text-Chat-128K	314	2434	$7.75\times$

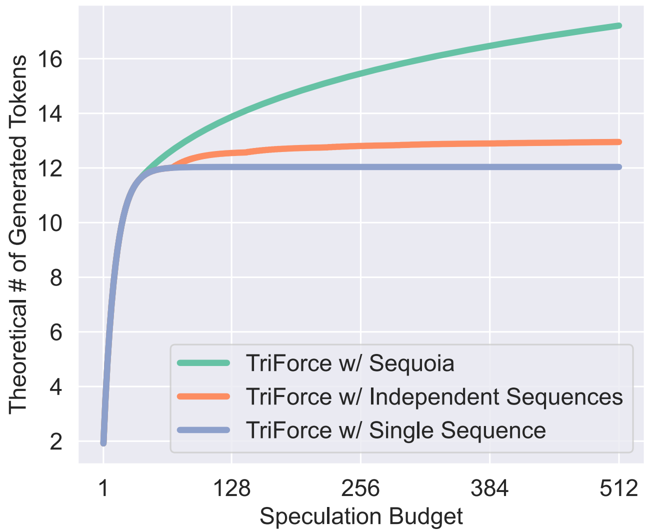
Larger Batch Size

Batch	Budget	T	Speedup	Naive Policy
(2,56K)	(2,1024)	0.0	**1.89**×	1.46×
(2,56K)	(2,1024)	0.6	**1.75**×	1.35×
(6,19K)	(6,768)	0.0	**1.90**×	1.39×
(6,19K)	(6,768)	0.6	**1.76**×	1.28×
(10,12K)	(10,768)	0.0	**1.72**×	1.34×
(10,12K)	(10,768)	0.6	**1.61**×	1.21×

Ablation Studies



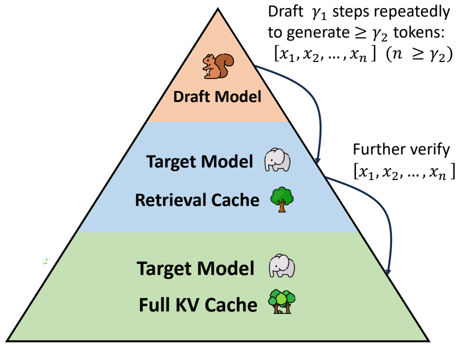


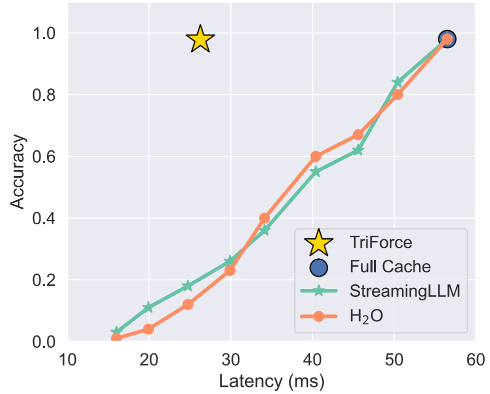


- Optimal KV Cache Budget
- Optimal Chunk Size for Locality
- Compatibility with Sequoia

LLMs are Powerful , but Very Expensive to Deploy

TriForce *(coming soon🔥)*





- Serve a Llama2-7B-128K (78GB mem) on a single RTX-4090 with 0.3s / token latency, 8× faster than DeepSpeed-Zero Offloading
- 2.3× speedup on a single A100 GPU

Thanks You!

Q&A