Carnegie Mellon University

Efficient Inference Methods

Large Language Models: Methods and Applications

Daphne Ippolito and Chenyan Xiong

slido

Please download and install the Slido app on all computers you use

Audience Q&A

ⓘ Start presenting to display the audience questions on this slide.

Learning Objectives

Learn the general concepts of efficient inference methods for LLM serving

Learn to build speculative decoding systems and potentially conduct research on model-based efficiency

Learn the basics of paged attention and flash attention

Outline

Overview

Model level efficiency: Speculative Decoding

Memory management efficiency: Paged Attention

System level optimization: Flash Attention

Serving LLMs

Serving large models live was costly and slow.

- Lots of OpenAI's cost are on serving
- E.g. if one model instance require 8 A100 then it's $10+ per hour

Serving LLMs

Serving large models live was costly and slow.

- Lots of OpenAI's cost are on serving
- E.g. if one model instance require 8 A100 then it's $10+ per hour

Huge challenge for the feasibility of LLMs being a real business

- Missing the economies of scale
- Even losing money per token sometimes
- Restricting to cloud-backed scenarios

Serving LLMs

Serving large models live was costly and slow.

- Lots of OpenAI's cost are on serving
- E.g. if one model instance require 8 A100 then it's $10+ per hour

Huge challenge for the feasibility of LLMs being a real business

- Missing the economies of scale
- Even losing money per token sometimes
- Restricting to cloud-backed scenarios

Unfortunately, the power of LLMs comes from scale

Serving LLMs

Serving large models live was costly and slow.

- Lots of OpenAI's cost are on serving
- E.g. if one model instance require 8 A100 then it's $10+ per hour

Huge challenge for the feasibility of LLMs being a real business

- Missing the economies of scale
- Even losing money per token sometimes
- Restricting to cloud-backed scenarios

Unfortunately, the power of LLMs comes from scale

Are we on a bubble bursting trajectory?

Model gets larger →Cost per user more expensive → No one makes money (except Nvidia) →VC gets inpatient (or broke) → bubble burst

Serving LLMs

Guido Appenzeller • 2nd

Investing in Infra & AI at a16z. Previously CTO @ Intel & VMwar...

2mo •

+ Follow

Generative AI is becoming crazy cheap. The cost of LLM inference has come down by 100x over 2 years (~$50 to $0.50 per 1M tokens) and having an LLM listen to everything you say for the entire year costs about $3.50.

Serving LLMs

Guido Appenzeller • 2nd

+ Follow •••

Investing in Infra & AI at a16z. Previously CTO @ Intel & VMwar...

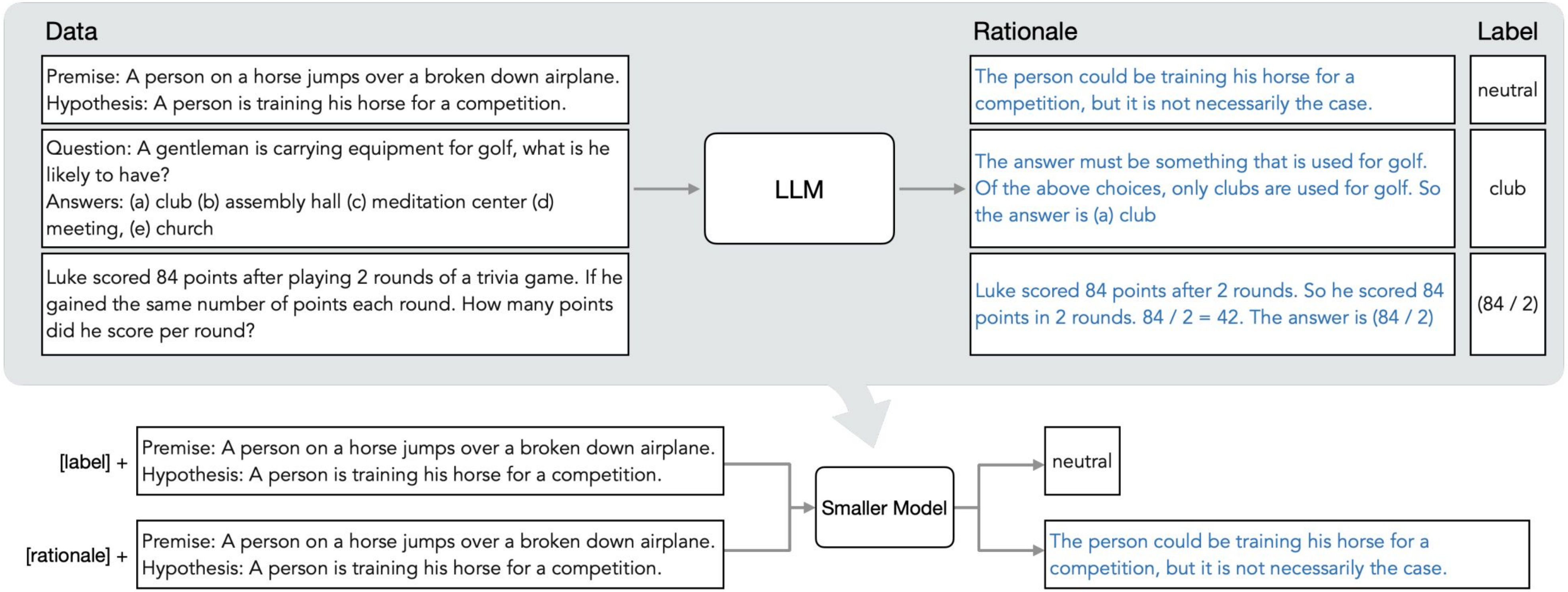
2mo •

Generative AI is becoming crazy cheap. The cost of LLM inference has come down by 100x over 2 years (~$50 to $0.50 per 1M tokens) and having an LLM listen to everything you say for the entire year costs about $3.50.

What got us here?

Trading Capability for Efficiency

Distillation: train smaller student models from the big teacher

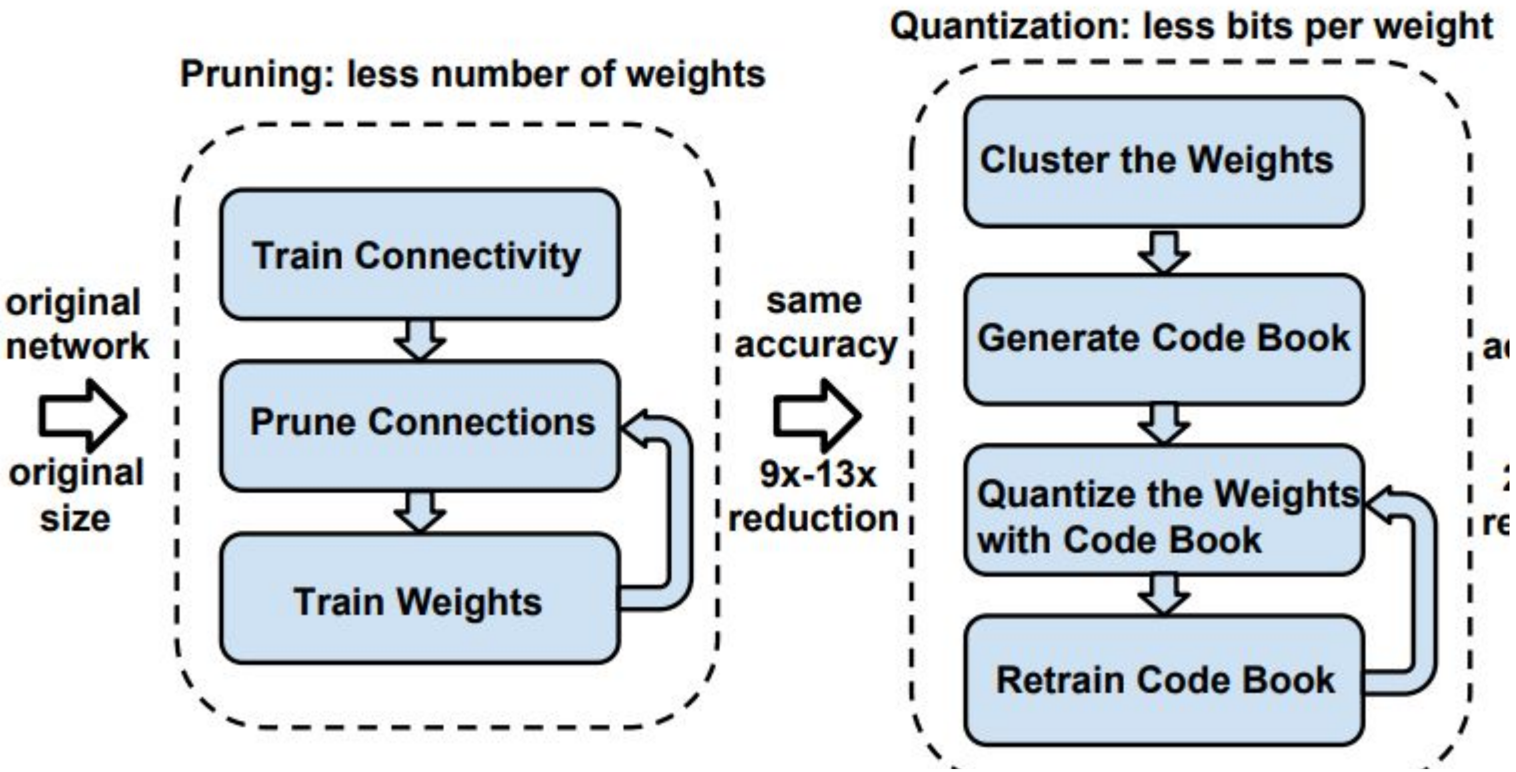


Distillation Step-by-Step [1]

[1] Hsieh et al. 2023. Distilling Step-by-Step! Outperforming Larger Language Models with Less Training Data and Smaller Model Sizes

Trading Capability for Efficiency

Pruning and Quantization: Delete and shrink parameters to cheaper formats

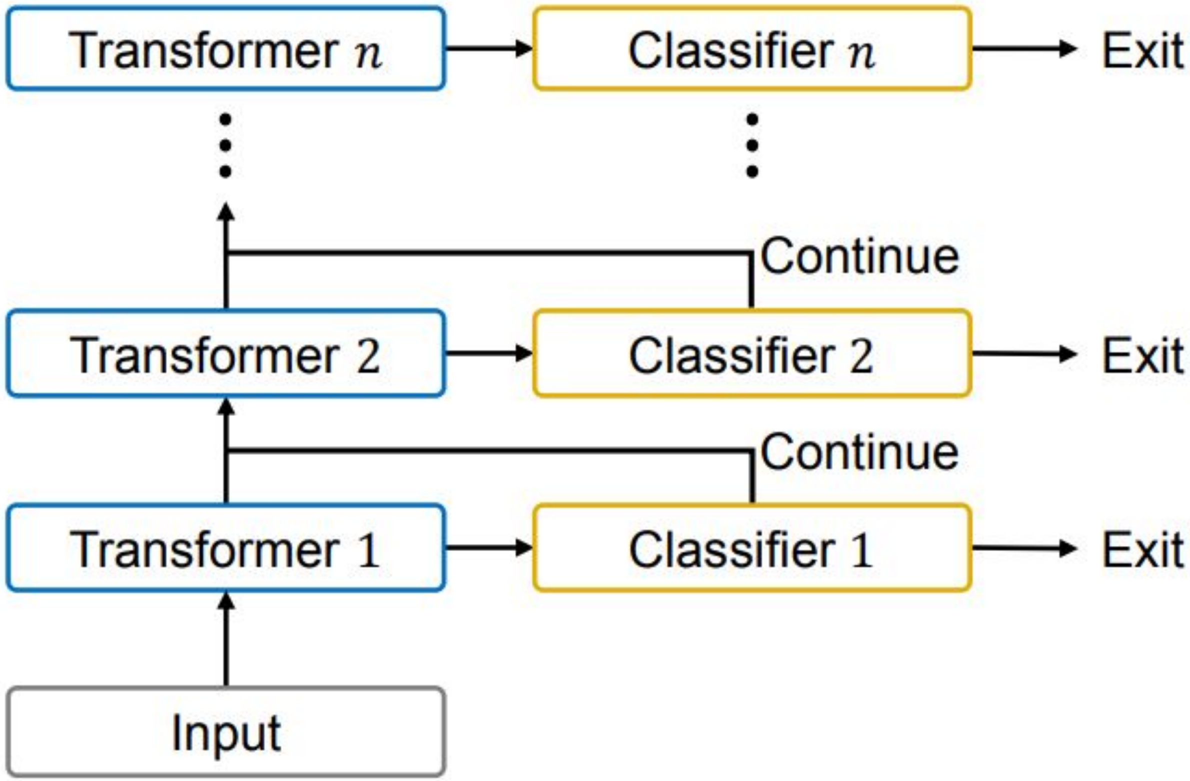


Pruning and Quantization of Neural Networks [2]

[2] Han et al. 2016. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding

Trading Capability for Efficiency

Early Exit: Skip some transformer layers or use earlier layer's predictions

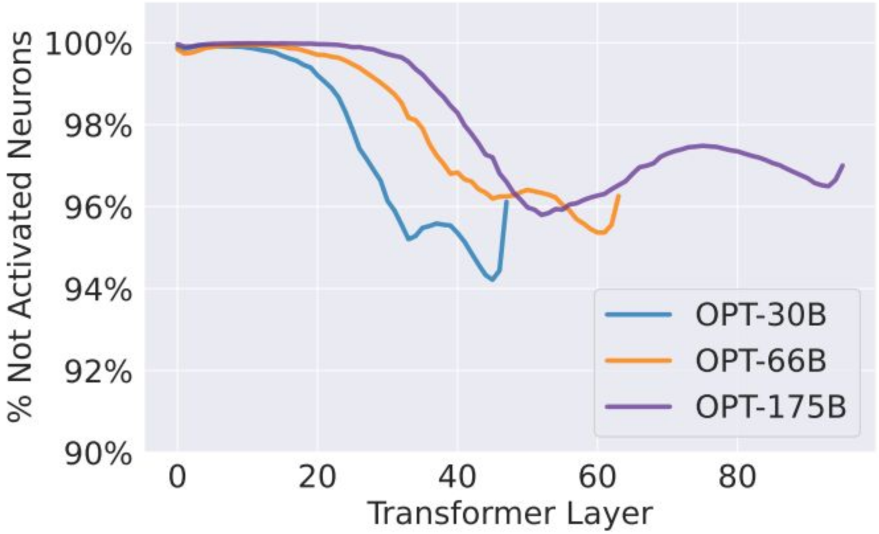


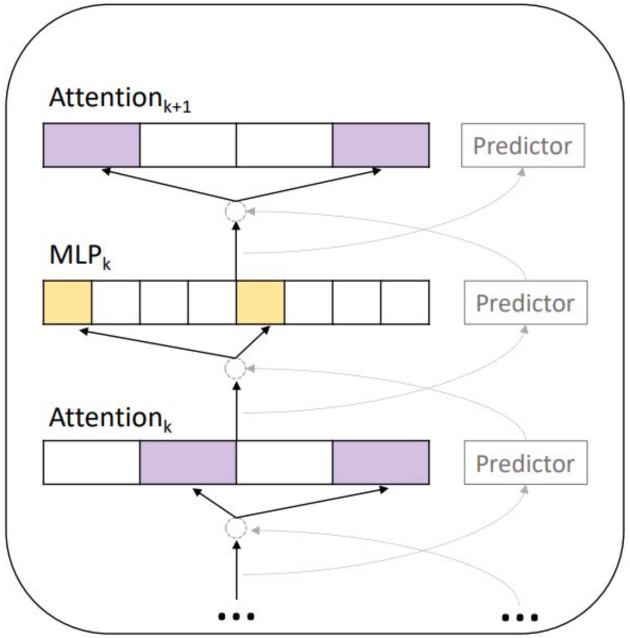
Early Exit in Classification [3]

[3] Xin et al. 2020. Early Exiting BERT for Efficient Document Ranking

Trading Capability for Efficiency

Sparsity: Skip certain neurons/blocks if predicted sparse likely





Early Exit in Classification [4]

Trading Capability for Efficiency

All above methods make the LLM "smaller"

- Inevitably lose some model "capability"

Though on benchmarks the loss is small, but there is no guarantee in real world scenarios

- Zero-shot
- New usage
- Edge cases
- Complicated cases
- Etc….

slido

Please download and install the
Slido app on all computers you use

Audience Q&A

ⓘ Start presenting to display the audience questions on this slide.

Trading Capability for Efficiency

All above methods make the LLM "smaller"

- Inevitably lose some model "capability"

Though on benchmarks the loss is small, but there is no guarantee in real world scenarios

- Zero-shot
- New usage
- Edge cases
- Complicated cases
- Etc....

Are there any "free" lunch?

Outline

Overview

Model level efficiency: Speculative Decoding

Memory management efficiency: Paged Attention

System level optimization: Flash Attention

Speculative Decoding

- Observations:
- We have a lot of decent smaller models
- They likely mimic the large model's behavior
  - Same model family, Distilled, or sub model from the large model
- Though generation of a sequence is sequential, scoring the sequence is O(1)

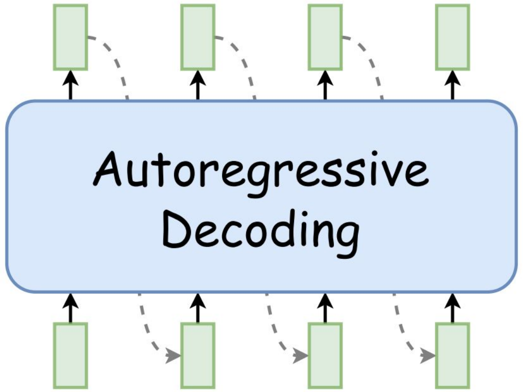
$$O\big(p(x_{n:n+k}|x_{<n})\big) \approx O(p(x_n|x_{<n}))$$

[5] Xia et al. 2024. Unlocking Efficiency in Large Language Model Inference: A Comprehensive Survey of Speculative Decoding

Speculative Decoding

- Observations:
- We have a lot of decent smaller models
- They likely mimic the large model's behavior
    - Same model family, Distilled, or sub model from the large model
- Though generation of a sequence is sequential, scoring the sequence is O(1)

$$O\left(p(x_{n:n+k}|x_{<n})\right) \approx O(p(x_n|x_{<n}))$$

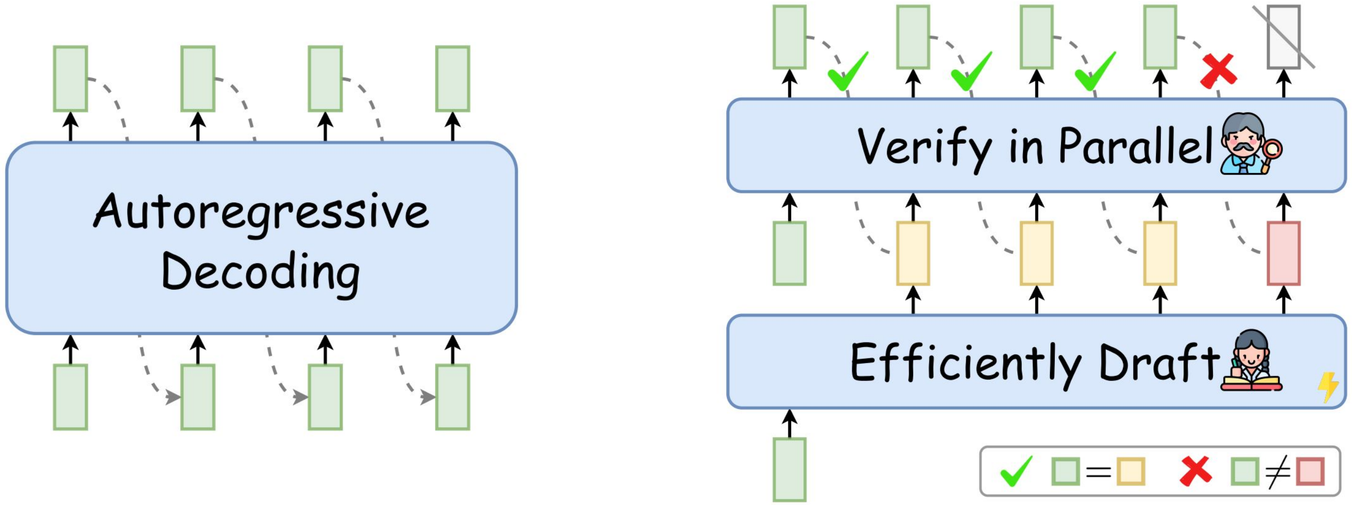


Speculative Decoding

Observations:

- We have a lot of decent smaller models
- They likely mimic the large model's behavior
  - Same model family, Distilled, or sub model from the large model
- Though generation of a sequence is sequential, scoring the sequence is O(1)

$$O\big(p(x_{n:n+k}|x_{<n})\big) \approx O(p(x_n|x_{<n}))$$



Speculative Decoding with Smaller Models to Propose and Large Model to verify [5]

[5] Xia et al. 2024. Unlocking Efficiency in Large Language Model Inference: A Comprehensive Survey of Speculative Decoding

Speculative Decoding

- Rejection Sampling to Recover the Large Model's Distribution p(x) Using a Small Model q(x) [6]

1. Sample $x \sim q(x)$
2. If $q(x) < p(x)$, keep x, finish
3. else reject x and
4. resample $x \sim norm(\max(0, p(x) - q(x)))$

[6] Leviathan et al. 2023. Fast Inference from Transformers via Speculative Decoding

Speculative Decoding

- Rejection Sampling to Recover the Large Model's Distribution p(x) Using a Small Model q(x) [6]

1. Sample $\boldsymbol{x \sim q(x)}$
2. If $\boldsymbol{q(x) < p(x)}$, keep x, finish
3. else reject x and
4. resample $\boldsymbol{x \sim norm(\max(0, p(x) - q(x)))}$

Proved that this is equal to Sample $\boldsymbol{x \sim p(x)}$

- Standard rejection sampling

[6] Leviathan et al. 2023. Fast Inference from Transformers via Speculative Decoding

Speculative Decoding

- Rejection Sampling to Recover the Large Model's Distribution p(x) Using a Small Model q(x) [6]

1. Sample $x \sim q(x)$
2. If $q(x) < p(x)$, keep x, finish
3. else reject x and
4. resample $x \sim norm(\max(0, p(x) - q(x)))$

Proved that this is equal to Sample $x \sim p(x)$

- Standard rejection sampling

To speed up need to speculate

Algorithm 1 SpeculativeDecodingStep

Inputs: $M_p, M_q, prefix$.
▷ Sample γ guesses $x_{1,\ldots,\gamma}$ from M_q autoregressively.
for $i = 1$ **to** γ **do**
  $q_i(x) \leftarrow M_q(prefix + [x_1, \ldots, x_{i-1}])$
  $x_i \sim q_i(x)$
end for
▷ Run M_p in parallel.
$p_1(x), \ldots, p_{\gamma+1}(x) \leftarrow$
  $M_p(prefix), \ldots, M_p(prefix + [x_1, \ldots, x_\gamma])$
▷ Determine the number of accepted guesses n.
$r_1 \sim U(0,1), \ldots, r_\gamma \sim U(0,1)$
$n \leftarrow \min(\{i - 1 \mid 1 \le i \le \gamma, r_i > \frac{p_i(x)}{q_i(x)}\} \cup \{\gamma\})$
▷ Adjust the distribution from M_p if needed.
$p'(x) \leftarrow p_{n+1}(x)$
if $n < \gamma$ **then**
  $p'(x) \leftarrow norm(max(0, p_{n+1}(x) - q_{n+1}(x)))$
end if
▷ Return one token from M_p, and n tokens from M_q.
$t \sim p'(x)$
return $prefix + [x_1, \ldots, x_n, t]$

Speculative Decoding

- Rejection Sampling to Recover the Large Model's Distribution p(x) Using a Small Model q(x) [6]

1. Sample $x \sim q(x)$
2. If $q(x) < p(x)$, keep x, finish
3. else reject x and
4. resample $x \sim norm(\max(0, p(x) - q(x)))$

Proved that this is equal to Sample $x \sim p(x)$

- Standard rejection sampling

To speed up need to speculate

- Draft a sequence from q()

Algorithm 1 SpeculativeDecodingStep

Inputs: $M_p, M_q, prefix$.

$\triangleright$ Sample γ guesses $x_{1,\ldots,\gamma}$ from M_q autoregressively.

Draft γ tokens from small model

for $i = 1$ **to** γ **do**

  $q_i(x) \leftarrow M_q(prefix + [x_1, \ldots, x_{i-1}])$

  $x_i \sim q_i(x)$

end for

$\triangleright$ Run M_p in parallel.

$p_1(x), \ldots, p_{\gamma+1}(x) \leftarrow$

  $M_p(prefix), \ldots, M_p(prefix + [x_1, \ldots, x_\gamma])$

$\triangleright$ Determine the number of accepted guesses n.

$r_1 \sim U(0,1), \ldots, r_\gamma \sim U(0,1)$

$n \leftarrow \min(\{i - 1 \mid 1 \leq i \leq \gamma, r_i > \frac{p_i(x)}{q_i(x)}\} \cup \{\gamma\})$

$\triangleright$ Adjust the distribution from M_p if needed.

$p'(x) \leftarrow p_{n+1}(x)$

if $n < \gamma$ **then**

  $p'(x) \leftarrow norm(max(0, p_{n+1}(x) - q_{n+1}(x)))$

end if

$\triangleright$ Return one token from M_p, and n tokens from M_q.

$t \sim p'(x)$

return $prefix + [x_1, \ldots, x_n, t]$

[6] Leviathan et al. 2023. Fast Inference from Transformers via Speculative Decoding

Speculative Decoding

- Rejection Sampling to Recover the Large Model's Distribution p(x) Using a Small Model q(x) [6]

1. Sample $x \sim q(x)$
2. If $q(x) < p(x)$, keep x, finish
3. else reject x and
4. resample $x \sim norm(\max(0, p(x) - q(x)))$

Proved that this is equal to Sample $x \sim p(x)$

- Standard rejection sampling

To speed up need to speculate

- Draft a sequence from q()
- Evaluate their probability using p()

Algorithm 1 SpeculativeDecodingStep

Inputs: $M_p, M_q, prefix$.

▷ Sample γ guesses $x_{1,\ldots,\gamma}$ from M_q autoregressively.

Draft γ tokens from small model

for $i = 1$ **to** γ **do**

$q_i(x) \leftarrow M_q(prefix + [x_1, \ldots, x_{i-1}])$

$x_i \sim q_i(x)$

end for

Batch evaluate

▷ Run M_p in parallel.

$p_1(x), \ldots, p_{\gamma+1}(x) \leftarrow$

$M_p(prefix), \ldots, M_p(prefix + [x_1, \ldots, x_\gamma])$

▷ Determine the number of accepted guesses n.

$r_1 \sim U(0,1), \ldots, r_\gamma \sim U(0,1)$

$n \leftarrow \min(\{i - 1 \mid 1 \leq i \leq \gamma, r_i > \frac{p_i(x)}{q_i(x)}\} \cup \{\gamma\})$

▷ Adjust the distribution from M_p if needed.

$p'(x) \leftarrow p_{n+1}(x)$

if $n < \gamma$ **then**

$p'(x) \leftarrow norm(max(0, p_{n+1}(x) - q_{n+1}(x)))$

end if

▷ Return one token from M_p, and n tokens from M_q.

$t \sim p'(x)$

return $prefix + [x_1, \ldots, x_n, t]$

[6] Leviathan et al. 2023. Fast Inference from Transformers via Speculative Decoding

Speculative Decoding

- Rejection Sampling to Recover the Large Model's Distribution p(x) Using a Small Model q(x) [6]

1. Sample $x \sim q(x)$
2. If $q(x) < p(x)$, keep x, finish
3. else reject x and
4. resample $x \sim norm(\max(0, p(x) - q(x)))$

Proved that this is equal to Sample $x \sim p(x)$

- Standard rejection sampling

To speed up need to speculate

- Draft a sequence from q()
- Evaluate their probability using p()
- Keep to the first rejected position

Algorithm 1 SpeculativeDecodingStep

Inputs: $M_p, M_q, prefix$.

$\triangleright$ Sample γ guesses $x_{1,\ldots,\gamma}$ from M_q autoregressively.

Draft γ tokens from small model

for $i = 1$ **to** γ **do**

$q_i(x) \leftarrow M_q(prefix + [x_1, \ldots, x_{i-1}])$

$x_i \sim q_i(x)$

end for

Batch evaluate

$\triangleright$ Run M_p in parallel.

$p_1(x), \ldots, p_{\gamma+1}(x) \leftarrow$

$M_p(prefix), \ldots, M_p(prefix + [x_1, \ldots, x_\gamma])$

Keep first n passed

$\triangleright$ Determine the number of accepted guesses n.

$r_1 \sim U(0,1), \ldots, r_\gamma \sim U(0,1)$

$n \leftarrow \min(\{i - 1 \mid 1 \leq i \leq \gamma, r_i > \frac{p_i(x)}{q_i(x)}\} \cup \{\gamma\})$

$\triangleright$ Adjust the distribution from M_p if needed.

$p'(x) \leftarrow p_{n+1}(x)$

if $n < \gamma$ **then**

$p'(x) \leftarrow norm(max(0, p_{n+1}(x) - q_{n+1}(x)))$

end if

$\triangleright$ Return one token from M_p, and n tokens from M_q.

$t \sim p'(x)$

return $prefix + [x_1, \ldots, x_n, t]$

[6] Leviathan et al. 2023. Fast Inference from Transformers via Speculative Decoding

Speculative Decoding

- Rejection Sampling to Recover the Large Model's Distribution p(x) Using a Small Model q(x) [6]

1. Sample $x \sim q(x)$
2. If $q(x) < p(x)$, keep x, finish
3. else reject x and
4. resample $x \sim norm(\max(0, p(x) - q(x)))$

Proved that this is equal to Sample $x \sim p(x)$

- Standard rejection sampling

To speed up need to speculate

- Draft a sequence from q()
- Evaluate their probability using p()
- Keep to the first rejected position
- Resample one

Algorithm 1 SpeculativeDecodingStep

Inputs: $M_p, M_q, prefix$.

Draft γ tokens from small model

$\triangleright$ Sample γ guesses $x_{1,\ldots,\gamma}$ from M_q autoregressively.
for $i = 1$ **to** γ **do**
  $q_i(x) \leftarrow M_q(prefix + [x_1, \ldots, x_{i-1}])$
  $x_i \sim q_i(x)$
end for

Batch evaluate

$\triangleright$ Run M_p in parallel.
$p_1(x), \ldots, p_{\gamma+1}(x) \leftarrow$
  $M_p(prefix), \ldots, M_p(prefix + [x_1, \ldots, x_\gamma])$

Keep first n passed

$\triangleright$ Determine the number of accepted guesses n.
$r_1 \sim U(0,1), \ldots, r_\gamma \sim U(0,1)$
$n \leftarrow \min(\{i - 1 \mid 1 \le i \le \gamma, r_i > \frac{p_i(x)}{q_i(x)}\} \cup \{\gamma\})$

Resample one from adjusted probability

$\triangleright$ Adjust the distribution from M_p if needed.
$p'(x) \leftarrow p_{n+1}(x)$
if $n < \gamma$ **then**
  $p'(x) \leftarrow norm(max(0, p_{n+1}(x) - q_{n+1}(x)))$
end if
$\triangleright$ Return one token from M_p, and n tokens from M_q.
$t \sim p'(x)$
return $prefix + [x_1, \ldots, x_n, t]$

Speculative Decoding

- Rejection Sampling to Recover the Large Model's Distribution p(x) Using a Small Model q(x) [6]

1. Sample $x \sim q(x)$
2. If $q(x) < p(x)$, keep x, finish
3. else reject x and
4. resample $x \sim norm(\max(0, p(x) - q(x)))$

Proved that this is equal to Sample $x \sim p(x)$

- Standard rejection sampling

To speed up need to speculate

- Draft a sequence from q()
- Evaluate their probability using p()
- Keep to the first rejected position
- Resample one

Worse case still breaks even

- Sampled one x with one run of p()

Algorithm 1 SpeculativeDecodingStep

Inputs: $M_p, M_q, prefix.$

Draft γ tokens from small model

$\triangleright$ Sample γ guesses $x_{1,\ldots,\gamma}$ from M_q autoregressively.
for $i = 1$ **to** γ **do**
  $q_i(x) \leftarrow M_q(prefix + [x_1, \ldots, x_{i-1}])$
  $x_i \sim q_i(x)$
end for

Batch evaluate

$\triangleright$ Run M_p in parallel.
$p_1(x), \ldots, p_{\gamma+1}(x) \leftarrow$
  $M_p(prefix), \ldots, M_p(prefix + [x_1, \ldots, x_\gamma])$

Keep first n passed

$\triangleright$ Determine the number of accepted guesses n.
$r_1 \sim U(0,1), \ldots, r_\gamma \sim U(0,1)$
$n \leftarrow \min(\{i - 1 \mid 1 \leq i \leq \gamma, r_i > \frac{p_i(x)}{q_i(x)}\} \cup \{\gamma\})$

Resample one from adjusted probability

$\triangleright$ Adjust the distribution from M_p if needed.
$p'(x) \leftarrow p_{n+1}(x)$
if $n < \gamma$ **then**
  $p'(x) \leftarrow norm(max(0, p_{n+1}(x) - q_{n+1}(x)))$
end if
$\triangleright$ Return one token from M_p, and n tokens from M_q.
$t \sim p'(x)$
return $prefix + [x_1, \ldots, x_n, t]$

[6] Leviathan et al. 2023. Fast Inference from Transformers via Speculative Decoding

Speculative Decoding

[START] japan ' s benchmark ~~bond~~ n

[START] japan ' s benchmark nikkei 22 ~~,~~5

[START] japan ' s benchmark nikkei 225 index rose 22 ~~,~~6

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 ~~,~~ points

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 0 1

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 9859

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 ~~,~~ in

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 in ~~tokyo~~ late

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 in late morning trading . [END]

Speculative Decoding with accepted drafts, rejected, and resampled tokens [6]

[6] Leviathan et al. 2023. Fast Inference from Transformers via Speculative Decoding

slido

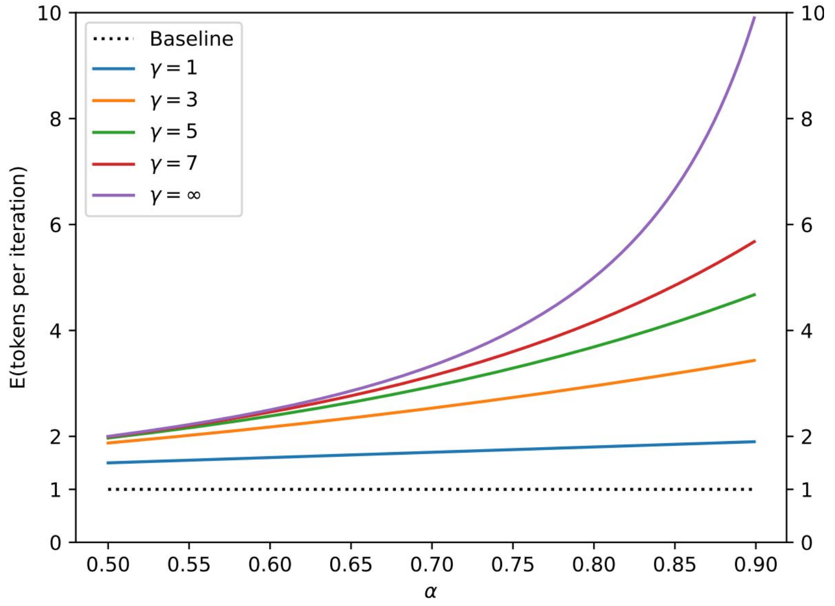
Please download and install the Slido app on all computers you use

Audience Q&A

ⓘ Start presenting to display the audience questions on this slide.

Speculative Decoding: Speed Up

- Two factors determining the speed up:
- Acceptance rate α: the expectation of a drafted token $q(x_t|x_{<t})$ being accepted
  - Stronger and closer q → better α

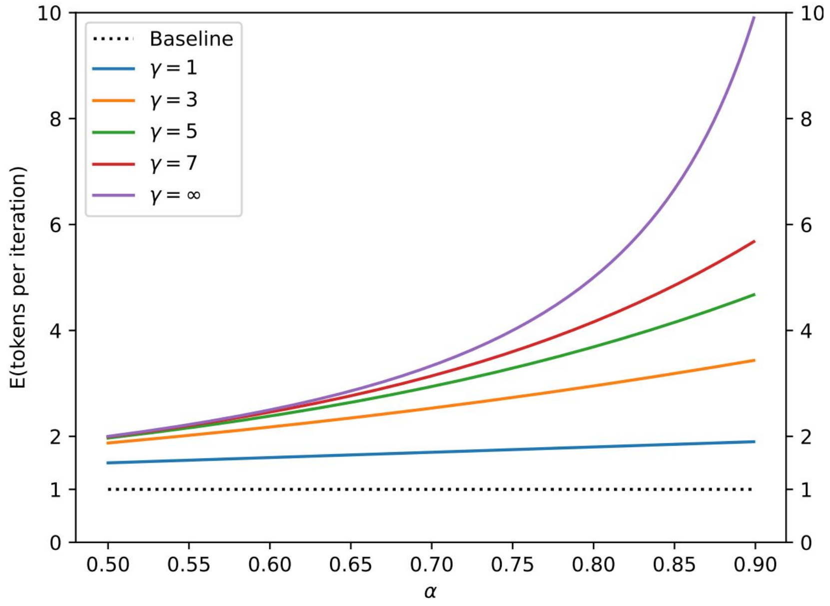


Expected Accepted Token Count with Different α [6]

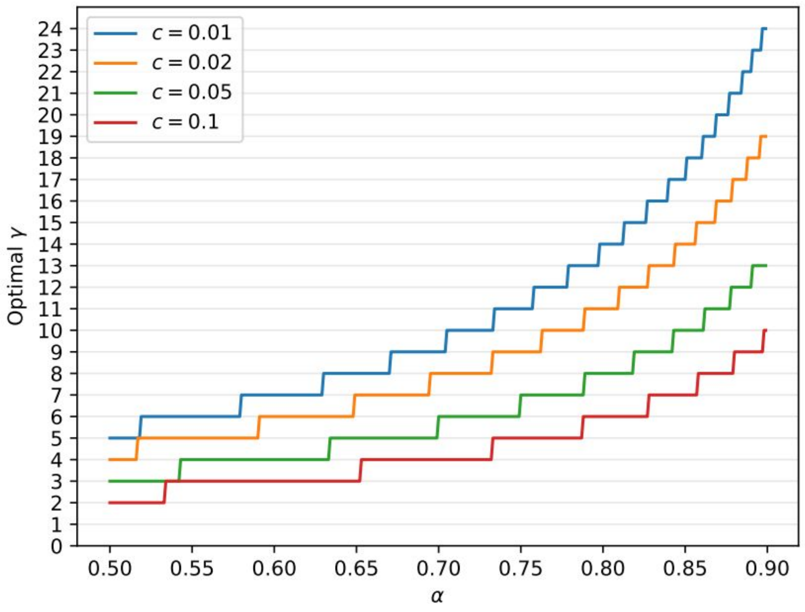
[6] Leviathan et al. 2023. Fast Inference from Transformers via Speculative Decoding

Speculative Decoding: Speed Up

- Two factors determining the speed up:
- Acceptance rate α: the expectation of a drafted token $q(x_t|x_{<t})$ being accepted
  - Stronger and closer q → better α
- Cost coefficient c: time required to run q over to run p $\frac{cost(q)}{cost(p)}$
  - Smaller q leads to better c



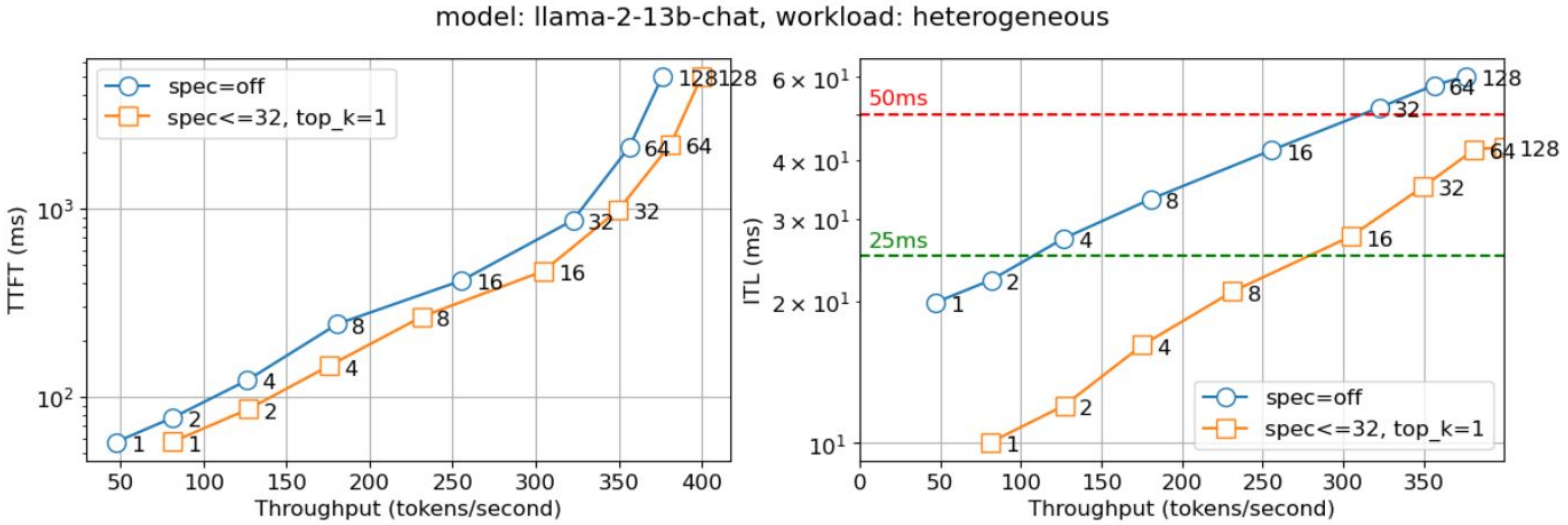
Expected Accepted Token Count with Different α [6]



Optimal Configuration for Different α and c [6]

Speculative Decoding: Performance

Performance gains while guaranteed exactness with rejection sampling



Speed improvement in time to first token (TTFT), inter token latency (ITL) and throughput [7]

Speculative Decoding: Remarks

A commonly deployed technology in various industry systems.

- Makes the system more complicated
- But gains of efficiency (huge $$$) without trading effectiveness

Speculative Decoding: Remarks

A commonly deployed technology in various industry systems.

- Makes the system more complicated
- But gains of efficiency (huge $$$) without trading effectiveness

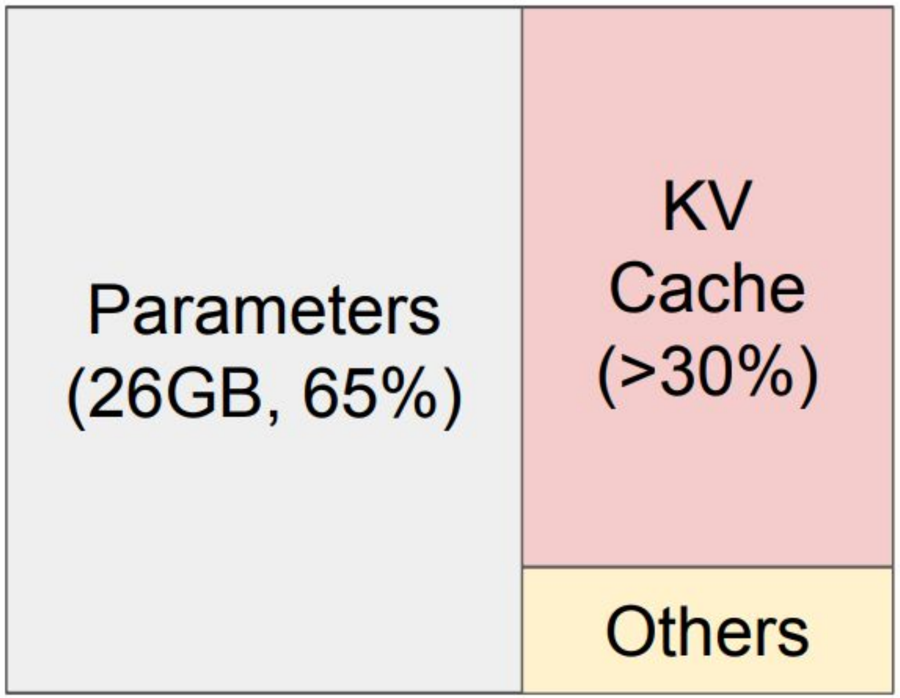
Further ways to improve:

- Better acceptance rate while cheaper drafting model
  - Align drafting model better with target model
- Better infrastructure support
  - MLSys developments

What is the bottleneck in LLM serving?

One GPU serving batched inferences of multiple requests

Model Parameter Storage that is constant, and shared by all inference requests



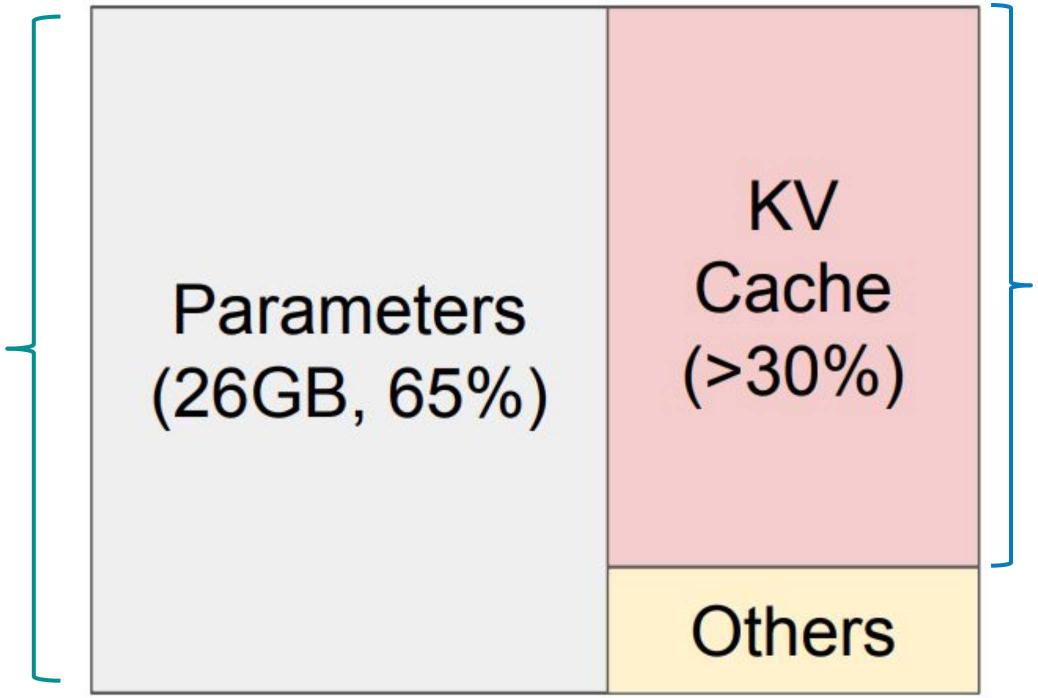
NVIDIA A100 40GB

Key-Value Activations, unique to every request and dynamically grows during inference

What is the bottleneck in LLM serving?

One GPU serving batched inferences of multiple requests

Model Parameter Storage that is constant, and shared by all inference requests



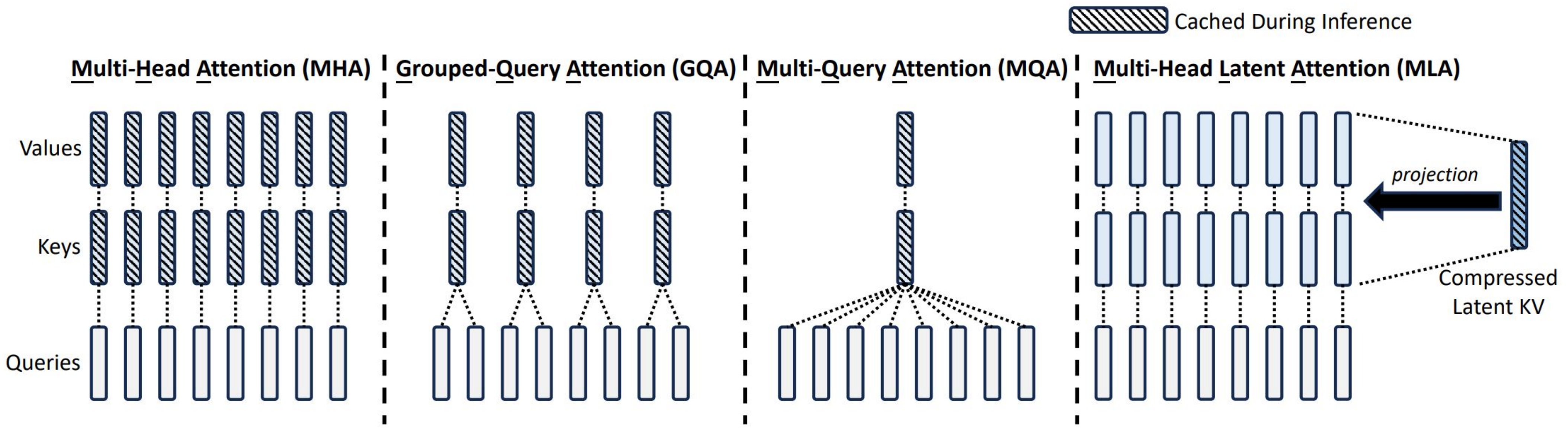
Key-Value Activations, unique to every request and dynamically grows during inference

NVIDIA A100 40GB

Typical LLM service is KV cache memory bound:
GPU memory becomes bottleneck first than other factors like FLOPs

Lossy KV Cache Reduction

Various attention versions with reduced KV cache memory footprint



KV Cache Management is Challenging

- Super dynamic: Grows token by token in our autoregressive generation
- From K,V of $\boldsymbol{x}_{<n}$ to $\boldsymbol{x}_{<n+1}$ after we generated $\boldsymbol{x}_n$

[8] Kwon, et al. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention

KV Cache Management is Challenging

- Super dynamic: Grows token by token in our autoregressive generation
  - From K,V of $\boldsymbol{x}_{<n}$ to $\boldsymbol{x}_{<n+1}$ after we generated $\boldsymbol{x}_n$
- Unpredictable: May end any time before maximum targeted length
  - LLM decides when the sequence ends by generating the <eos> token

KV Cache Management is Challenging

- Super dynamic: Grows token by token in our autoregressive generation
- From K,V of $\boldsymbol{x}_{<n}$ to $\boldsymbol{x}_{<n+1}$ after we generated $\boldsymbol{x}_n$

Unpredictable: May end any time before maximum targeted length

- LLM decides when the sequence ends by generating the <eos> token

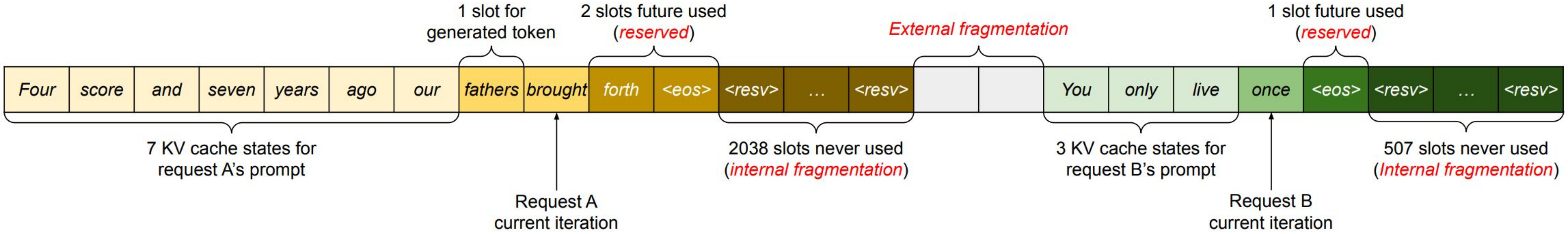
Async in batch: Requests come and responses end at different time

- Interval of requests of the same session also unpredictable

[8] Kwon, et al. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention

KV Cache Management is Challenging

- Super dynamic: Grows token by token in our autoregressive generation
  - From K,V of $x_{<n}$ to $x_{<n+1}$ after we generated x_n
- Unpredictable: May end any time before maximum targeted length
  - LLM decides when the sequence ends by generating the <eos> token
- Async in batch: Requests come and responses end at different time
  - Interval of requests of the same session also unpredictable

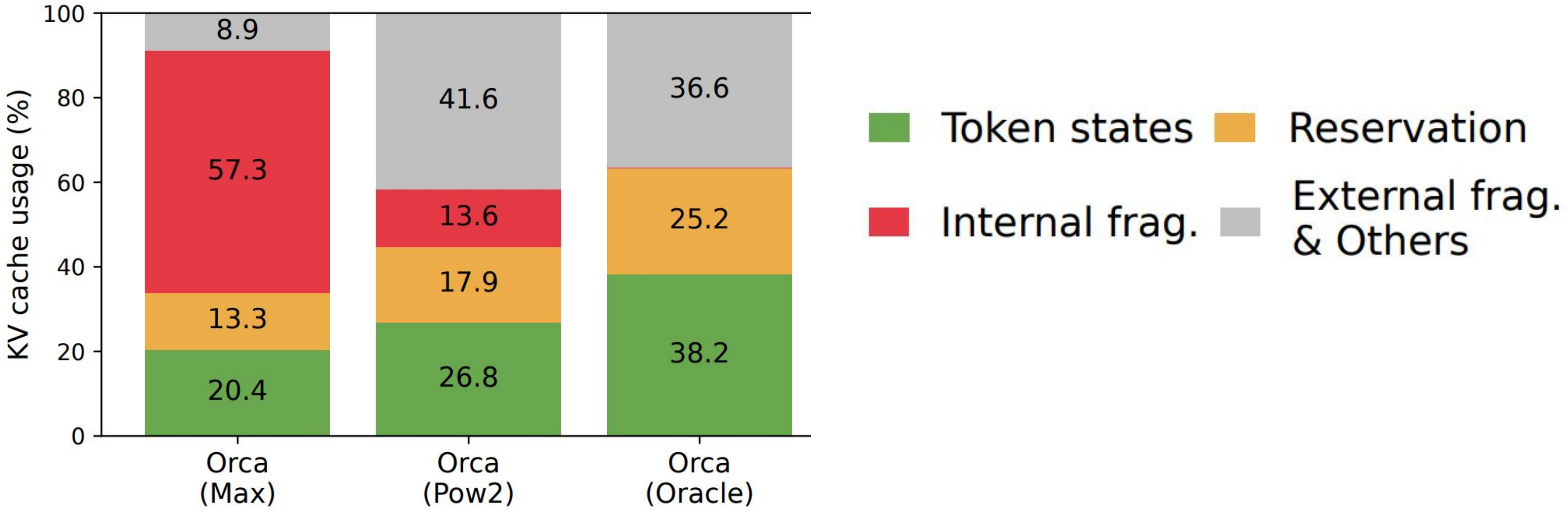


KV Cache Management in Vanilla LLM Serving Systems [8]

[8] Kwon, et al. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention

KV Cache Management is Challenging

Resulted in Huge waste of GPU memory $$$



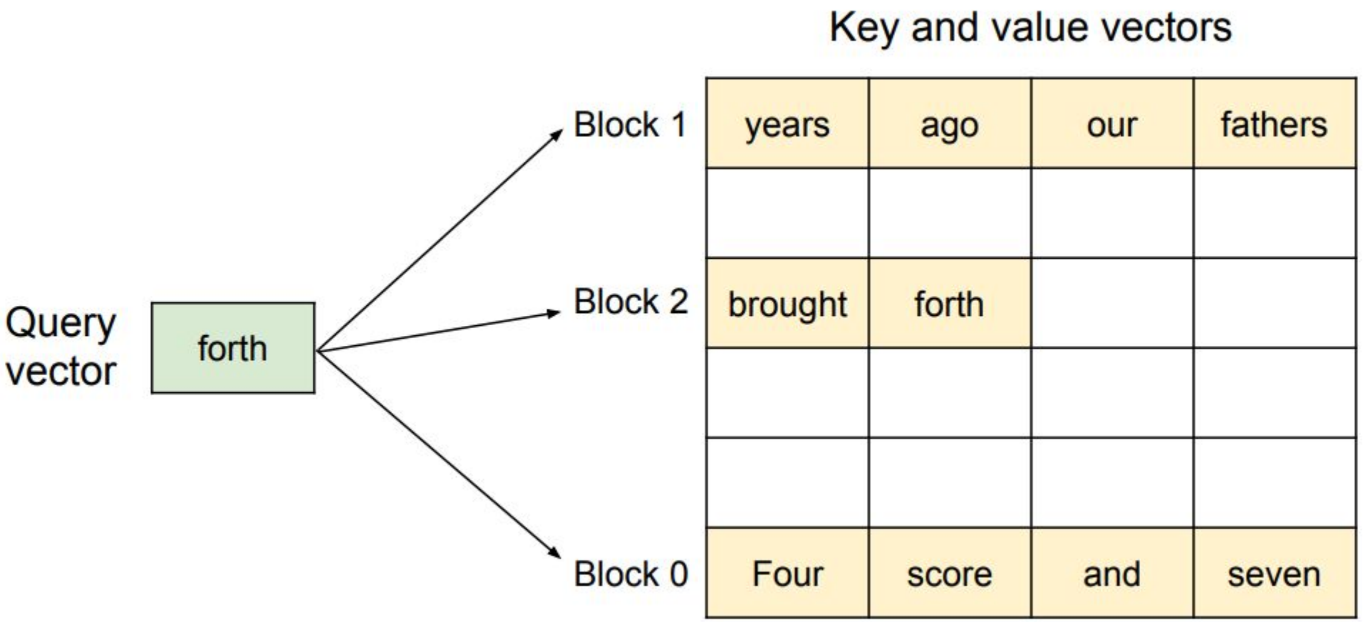
GPU Memory Fragmentations and Wastes in LLM Serving [8]

[8] Kwon, et al. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention

KV Cache Management in vLLM

Splitting KV cache of a sequence into blocks for more flexible allocations [8]

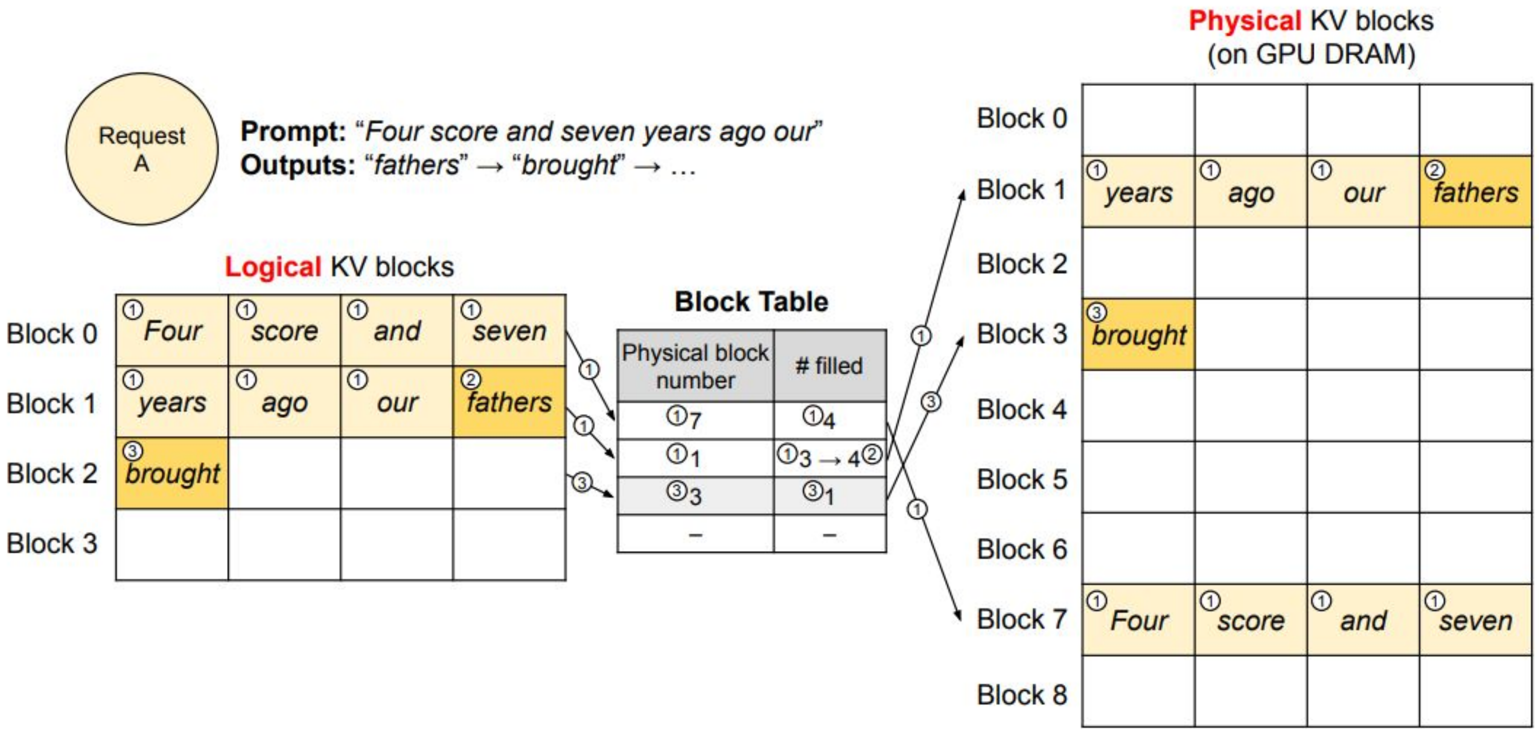
- Classic paging idea in CPU memory management



Splitting Sequence's KV into sub blocks for flexibility [8]

KV Cache Management in vLLM

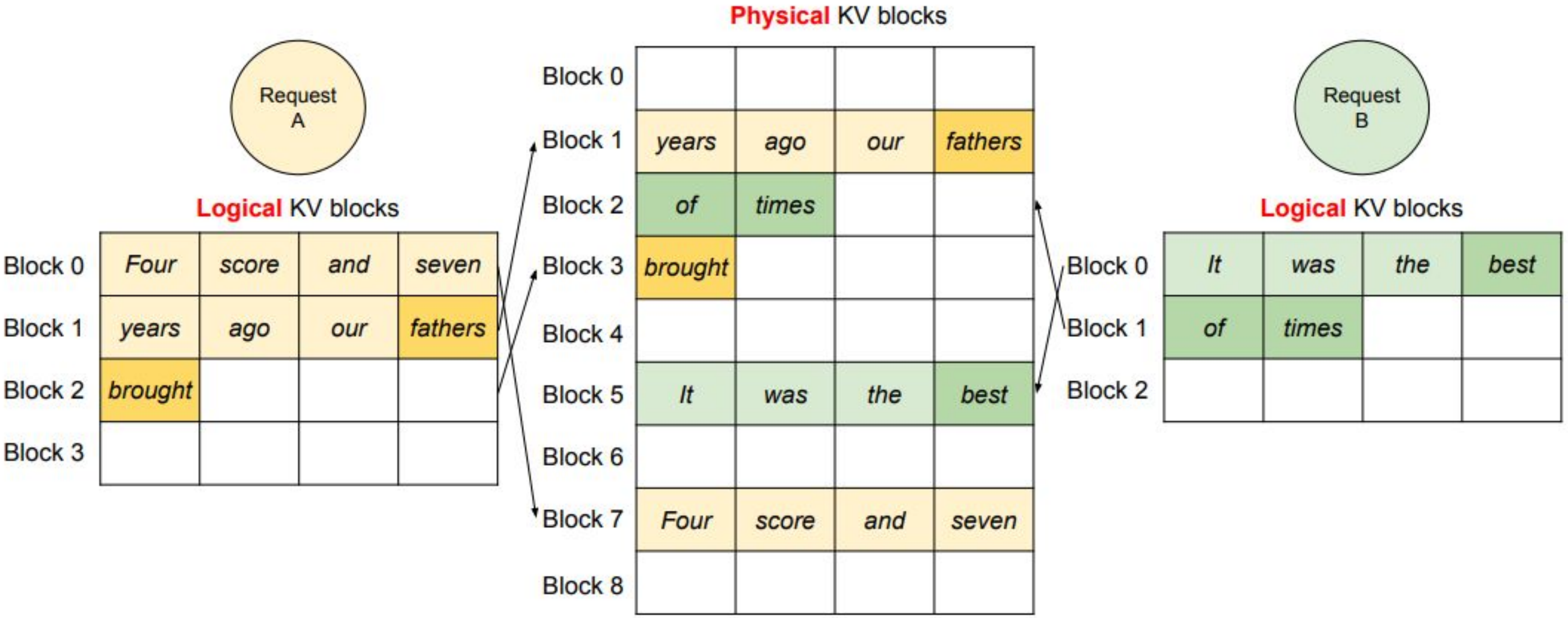
Managing the KV blocks with virtual block tables



Splitting Sequence's KV into sub blocks for flexibility [8]

KV Cache Management in vLLM

More efficient KV cache management at block level

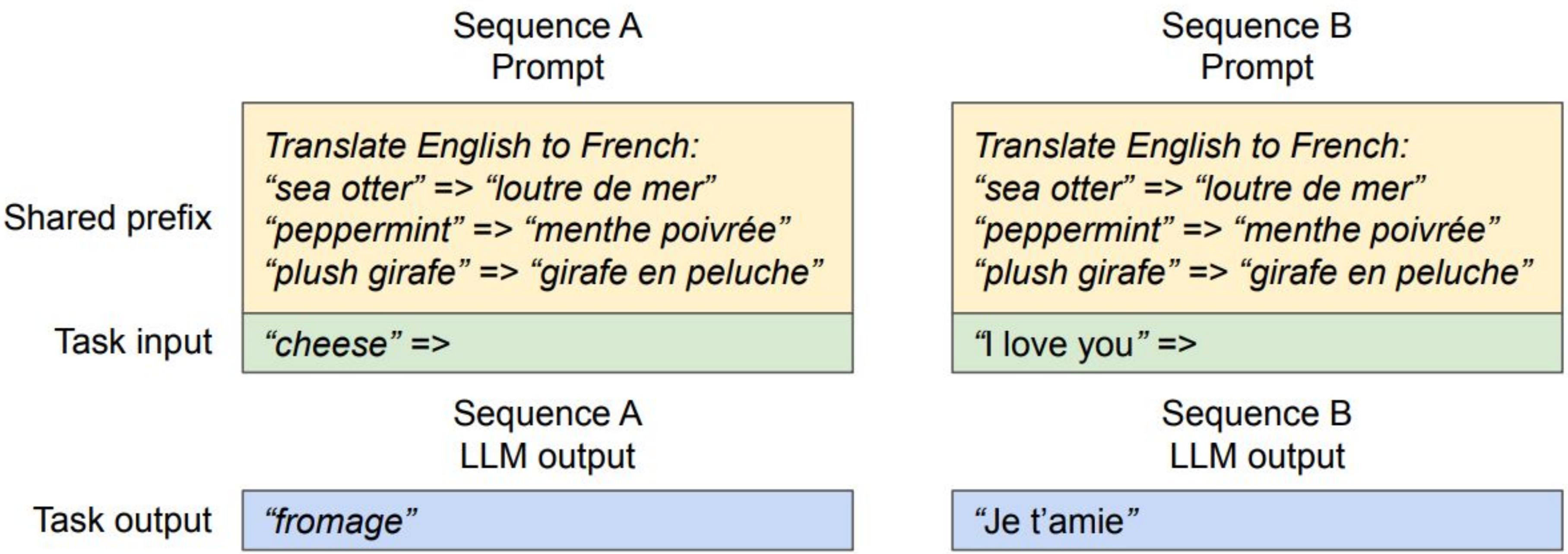


Mixing KV blocks of two requests [8]

[8] Kwon, et al. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention

KV Cache Management in vLLM

Design KV cache block management algorithms for common LLM serving scenarios

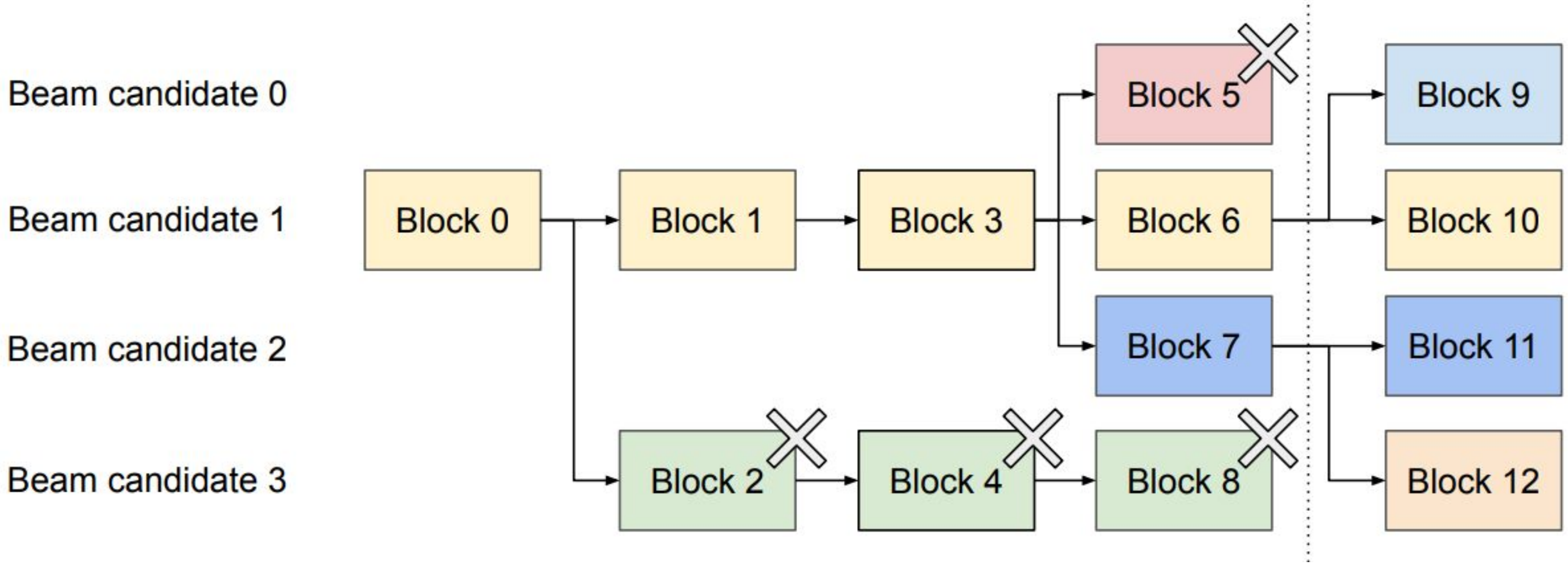


Shared Prompts Using Shared KV Blocks [8]

[8] Kwon, et al. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention

KV Cache Management in vLLM

Design KV cache block management algorithms for common LLM serving scenarios

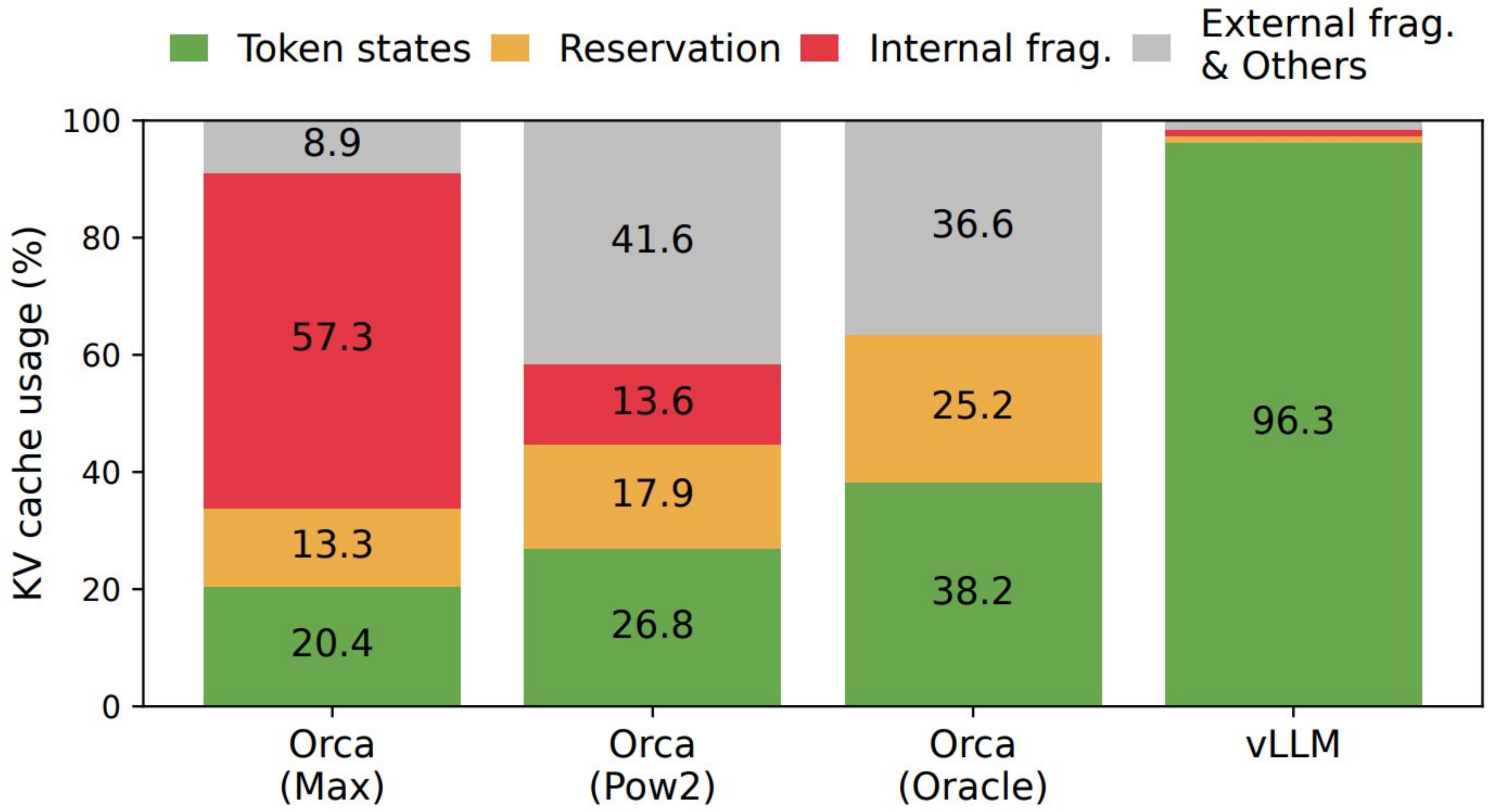


Shared KV Blocks in Beam Search [8]

[8] Kwon, et al. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention

KV Cache Management in vLLM: Performance

No wastes with PagedAttention block management

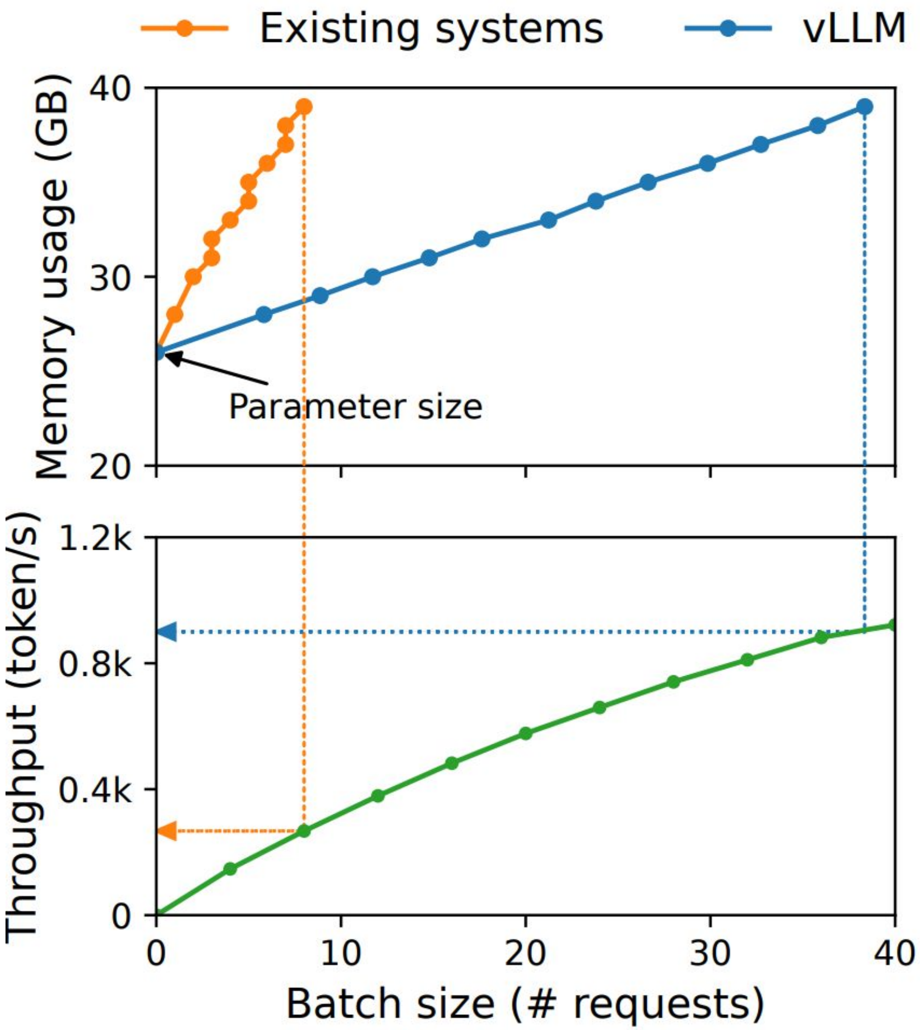


Fragmentation of GPU Memory [8]

[8] Kwon, et al. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention

KV Cache Management in vLLM: Performance

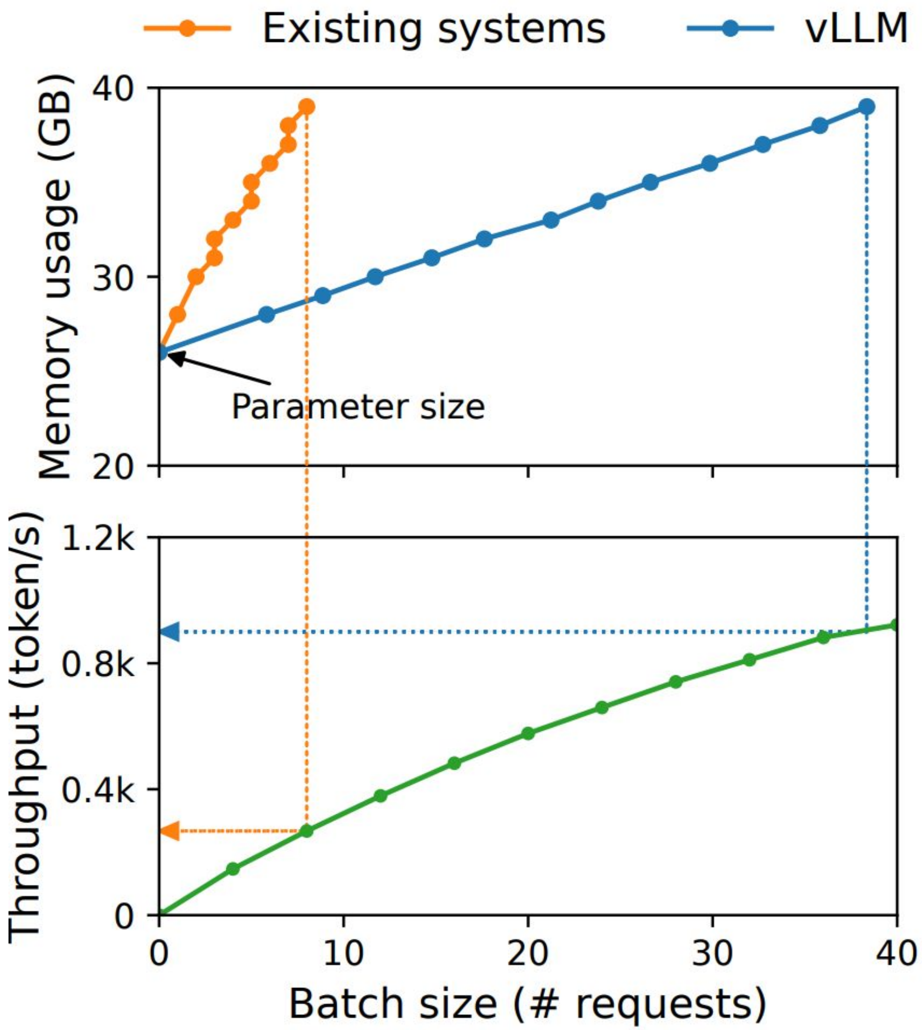
Fits significantly more requests per batch with efficient usage of GPU memory

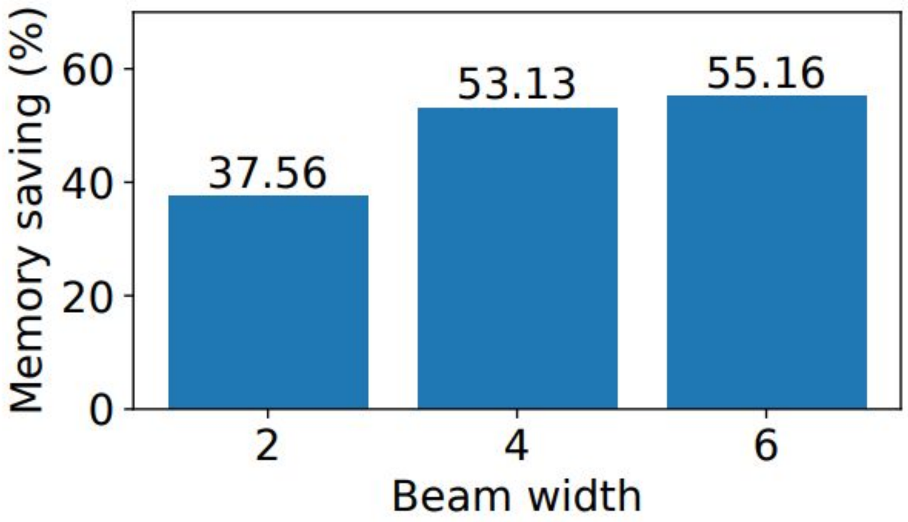


[8] Kwon, et al. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention

KV Cache Management in vLLM: Performance

Fits significantly more requests per batch with efficient usage of GPU memory





vLLM Community Outreach

vLLM x Snowflake Meetup (Wednesday, November 13th, 5:30-8PM PT) at Snowflake HQ, San Mateo

We are excited to announce the last in-person vLLM meetup of the year! Join the vLLM developers and engineers from Snowflake AI Research to chat about the latest LLM inference optimizations and your 2025 vLLM wishlist! Register here and be a part of the event!

Latest News 🔥

- [2024/10] We have just created a developer slack (slack.vllm.ai) focusing on coordinating contributions and discussing features. Please feel free to join us there!
- [2024/10] Ray Summit 2024 held a special track for vLLM! Please find the opening talk slides from the vLLM team here. Learn more from the talks from other vLLM contributors and users!
- [2024/09] We hosted the sixth vLLM meetup with NVIDIA! Please find the meetup slides here.
- [2024/07] We hosted the fifth vLLM meetup with AWS! Please find the meetup slides here.
- [2024/07] In partnership with Meta, vLLM officially supports Llama 3.1 with FP8 quantization and pipeline parallelism! Please check out our blog post here.
- [2024/06] We hosted the fourth vLLM meetup with Cloudflare and BentoML! Please find the meetup slides here.
- [2024/04] We hosted the third vLLM meetup with Roblox! Please find the meetup slides here.
- [2024/01] We hosted the second vLLM meetup with IBM! Please find the meetup slides here.
- [2023/10] We hosted the first vLLM meetup with a16z! Please find the meetup slides here.
- [2023/08] We would like to express our sincere gratitude to Andreessen Horowitz (a16z) for providing a generous grant to support the open-source development and research of vLLM.
- [2023/06] We officially released vLLM! FastChat-vLLM integration has powered LMSYS Vicuna and Chatbot Arena since mid-April. Check out our blog post.

slido

Please download and install the Slido app on all computers you use

Audience Q&A

ⓘ Start presenting to display the audience questions on this slide.

Outline

Overview

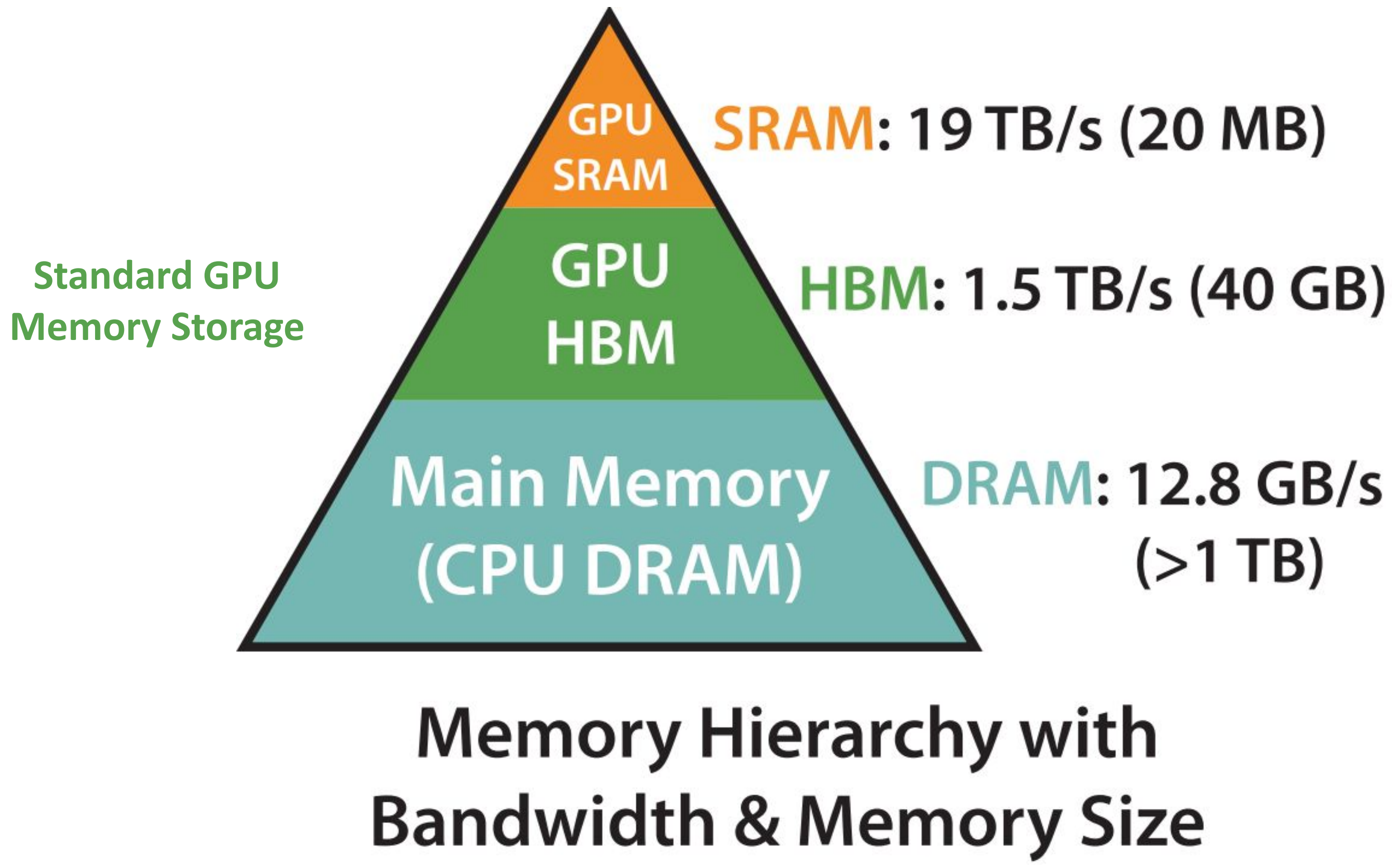
Model level efficiency: Speculative Decoding

Memory management efficiency: Paged Attention

System level optimization: Flash Attention

Full System Optimization

Many other resources available in the computing system



[10] Dao, et al. 2022. Fast and Memory-Efficient Exact Attention with IO-Awareness

Full System Optimization

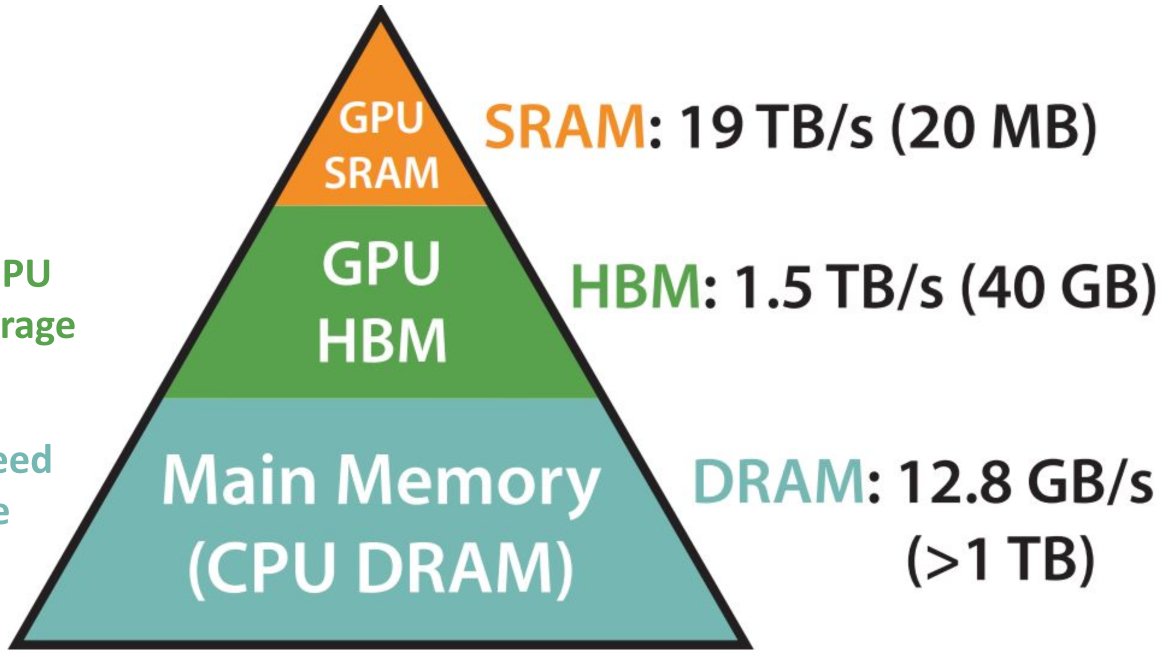
Many other resources available in the computing system

E.g., Gradient Checkpointing

Standard GPU Memory Storage

↓

Trading Speed for Space



Memory Hierarchy with Bandwidth & Memory Size

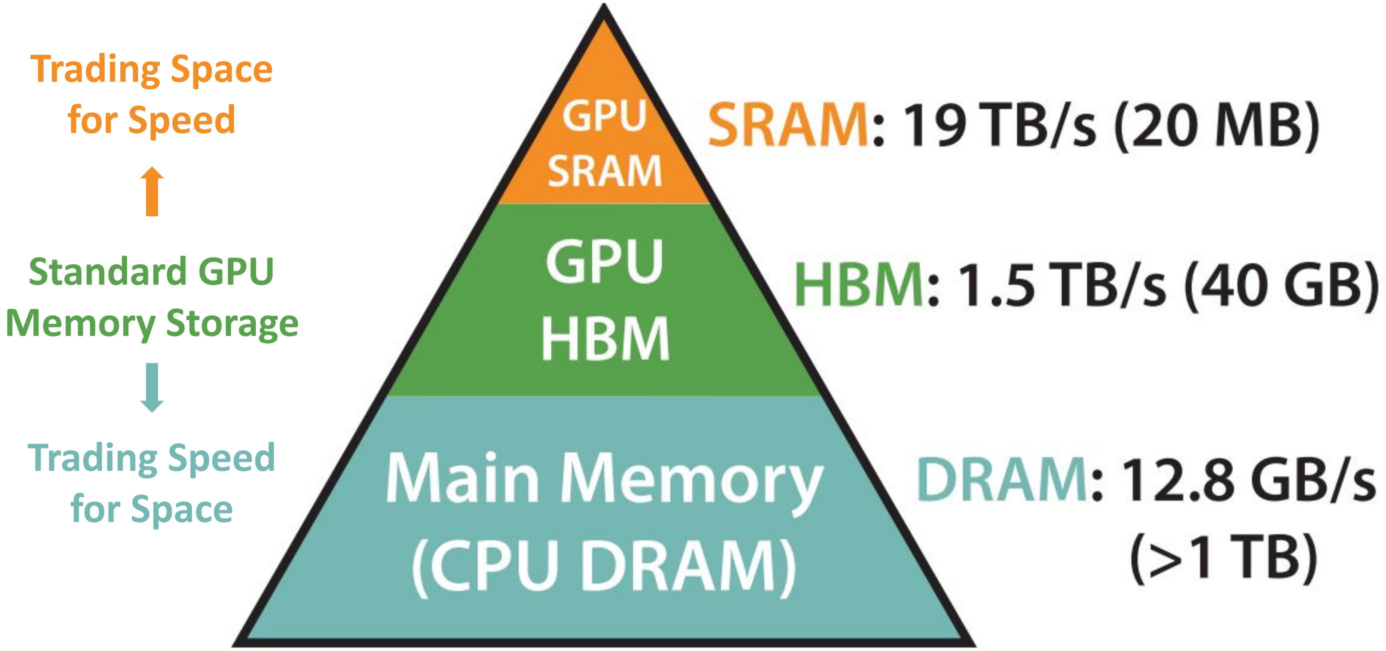
[10] Dao, et al. 2022. Fast and Memory-Efficient Exact Attention with IO-Awareness

Full System Optimization

Many other resources available in the computing system

Next: Flash Attention

E.g., Gradient Checkpointing



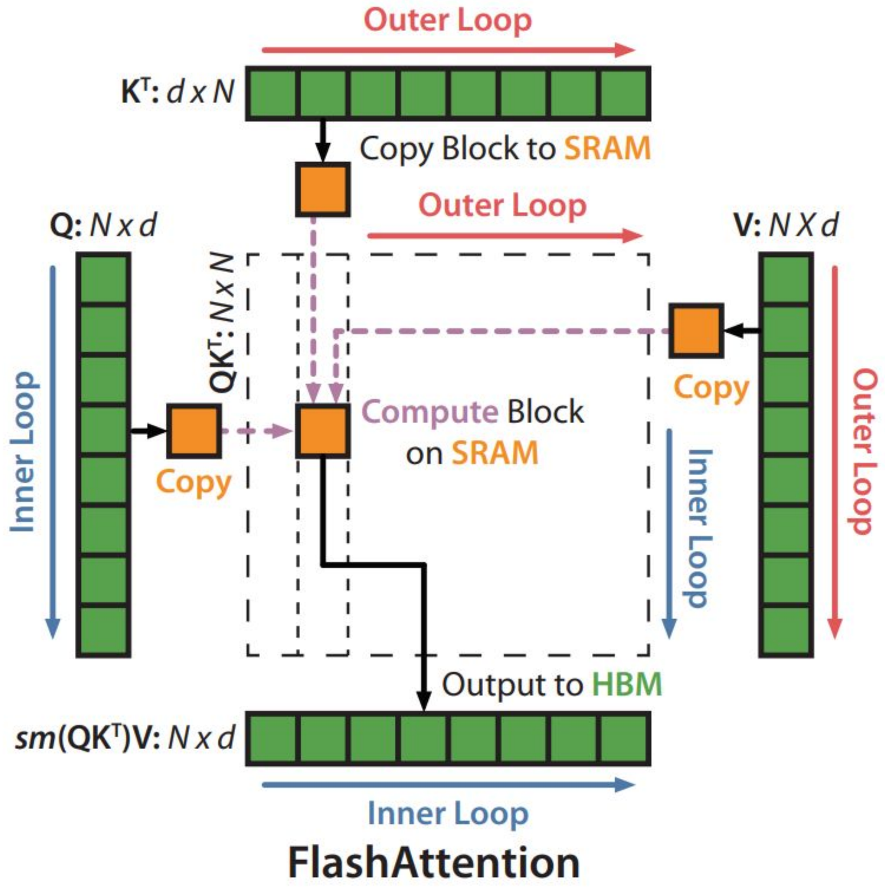
Memory Hierarchy with Bandwidth & Memory Size

[10] Dao, et al. 2022. Fast and Memory-Efficient Exact Attention with IO-Awareness

Flash Attention: Managing SRAM IO Efficiently

Compute Attention as small blocks in fast SRAM

Load K blocks in SRAM and work with all Q



Load V blocks in SRAM for all Q

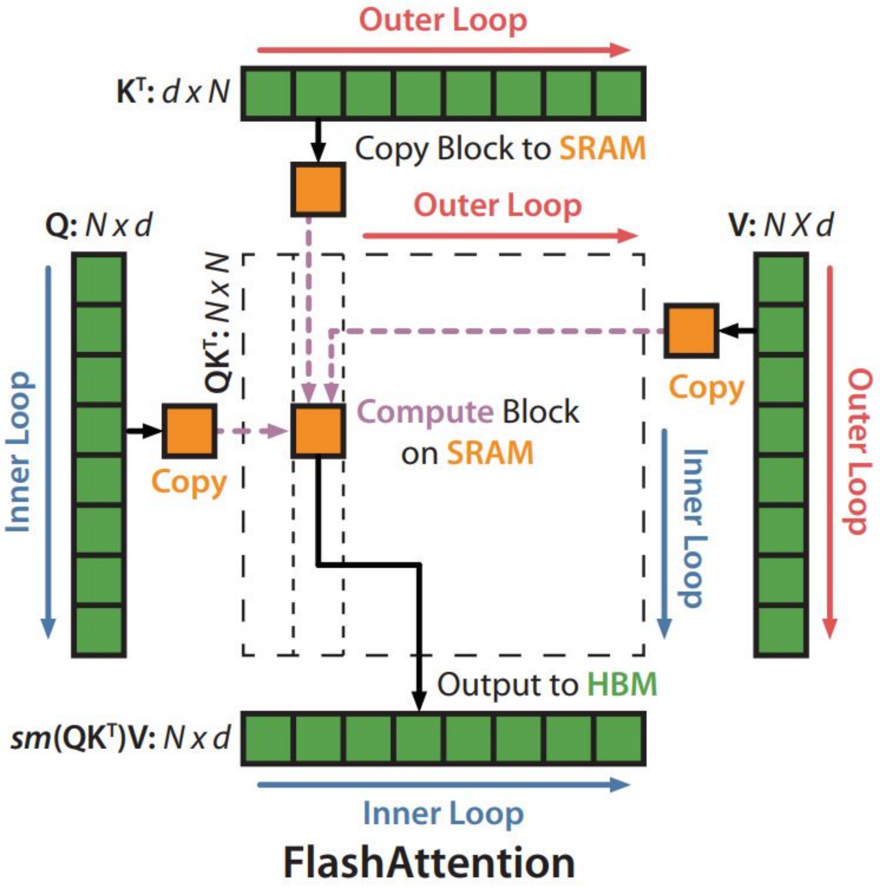
Flash Attention: Managing SRAM IO Efficiently

Compute Attention as small blocks in fast SRAM

Load K blocks in SRAM and work with all Q

Moving Q block by block to SRAM

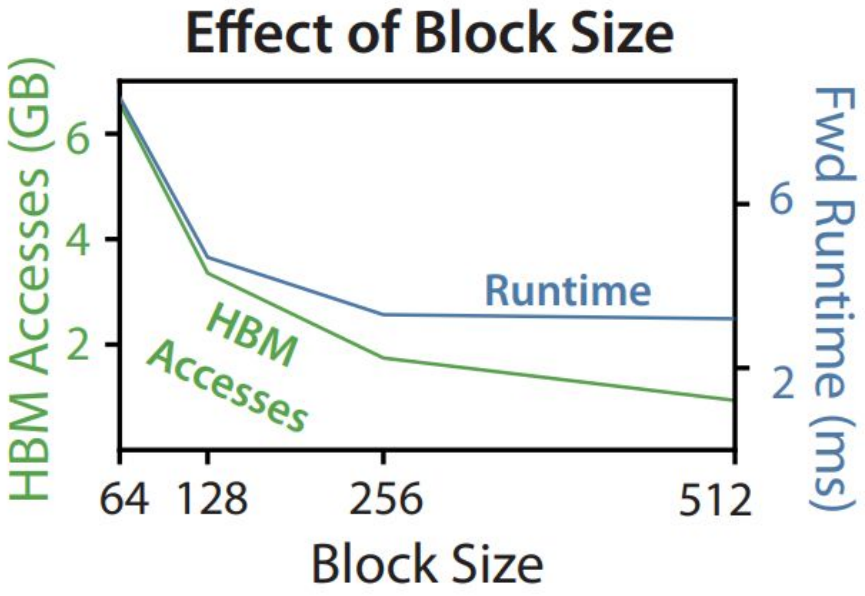
Load V blocks in SRAM for all Q



Flash Attention: Managing SRAM IO Efficiently

Fewer HBM (GPU Memory) IO, faster performance

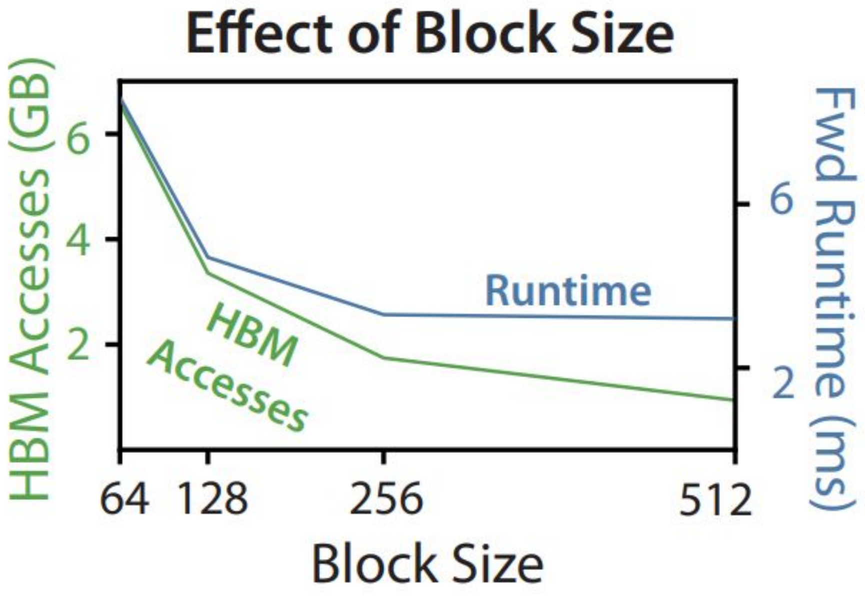
Attention	Standard	FLASHATTENTION
GFLOPs	66.6	75.2
HBM R/W (GB)	40.3	4.4
Runtime (ms)	41.7	7.3



Flash Attention: Managing SRAM IO Efficiently

Fewer HBM (GPU Memory) IO, faster performance

Attention	Standard	FlashAttention
GFLOPs	66.6	75.2
HBM R/W (GB)	40.3	4.4
Runtime (ms)	41.7	7.3



BERT Implementation	Training time (minutes)
Nvidia MLPerf 1.1 [58]	20.0 ± 1.5
FlashAttention (ours)	**17.4** ± 1.4

slido

Please download and install the Slido app on all computers you use

Audience Q&A

ⓘ Start presenting to display the audience questions on this slide.

Remarks

Trading Effectiveness for Efficiency

- Some scenarios do not need too large a model

Model level efficiency: Speculative Decoding

- Utilization the strong agreement of small LM and large LM

Memory management efficiency: Paged Attention

- Addressed the GPU memory manage issue using classic CPU memory management methods
- Designed customized GPU memory management methods specialized to LLM workflows

System level optimization: Flash Attention

- Implemented the caching techniques on GPUs

Remarks

Trading Effectiveness for Efficiency

- Some scenarios do not need too large a model

Model level efficiency: Speculative Decoding

- Utilization the strong agreement of small LM and large LM

Memory management efficiency: Paged Attention

- Addressed the GPU memory manage issue using classic CPU memory management methods
- Designed customized GPU memory management methods specialized to LLM workflows

System level optimization: Flash Attention

- Implemented the caching techniques on GPUs

What got us 100x inference efficiency?

- Fixed problems/lack of optimization on GPU stack
- Customized infrastructure for LLM workflows
- Not necessarily free lunch, more like things left on the table

Remarks

Trading Effectiveness for Efficiency
- Some scenarios do not need too large a model

Model level efficiency: Speculative Decoding
- Utilization the strong agreement of small LM and large LM

Memory management efficiency: Paged Attention
- Addressed the GPU memory manage issue using classic CPU memory management methods
- Designed customized GPU memory management methods specialized to LLM workflows

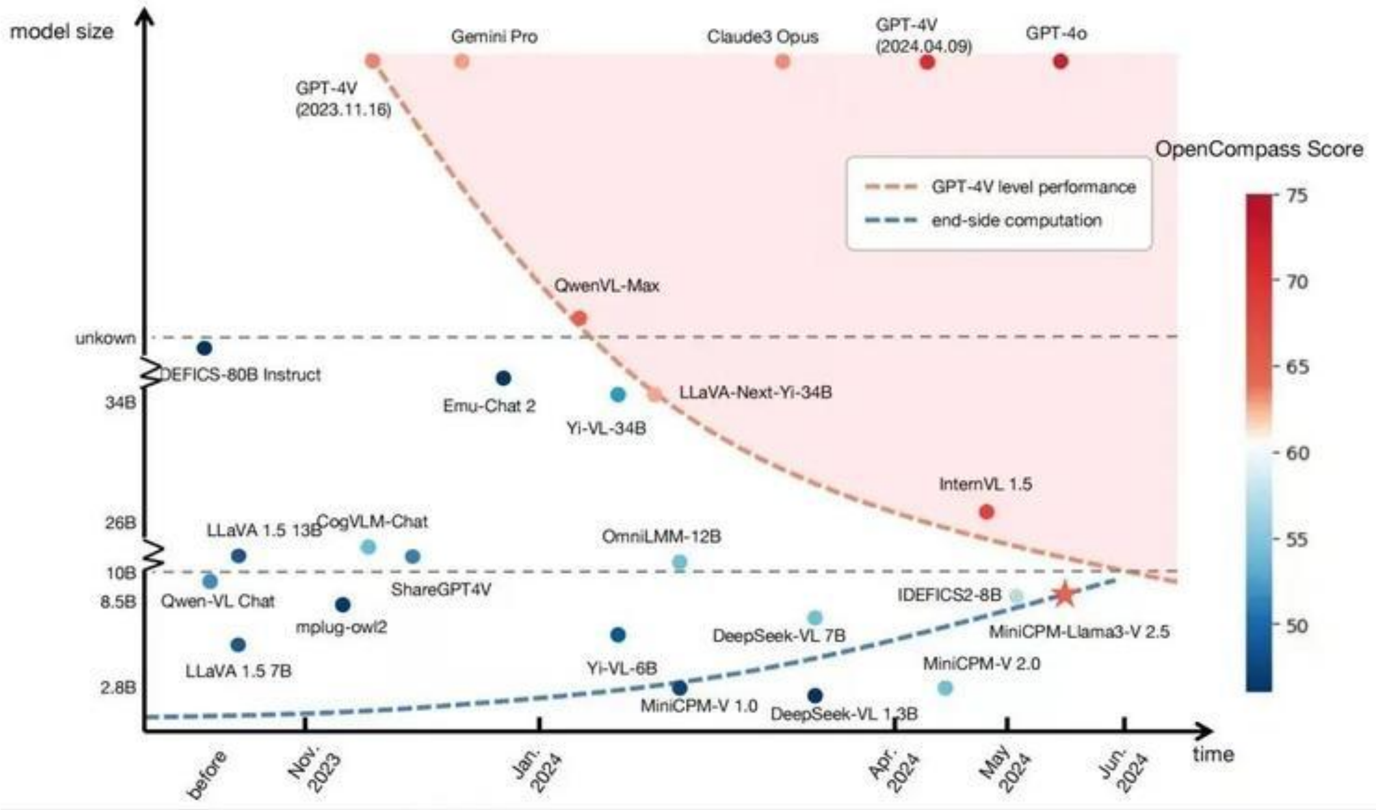
System level optimization: Flash Attention
- Implemented the caching techniques on GPUs

What got us 100x inference efficiency?
- Fixed problems/lack of optimization on GPU stack
- Customized infrastructure for LLM workflows
- Not necessarily free lunch, more like things left on the table

Are these sustainable?

Where is the next 100x speed up?

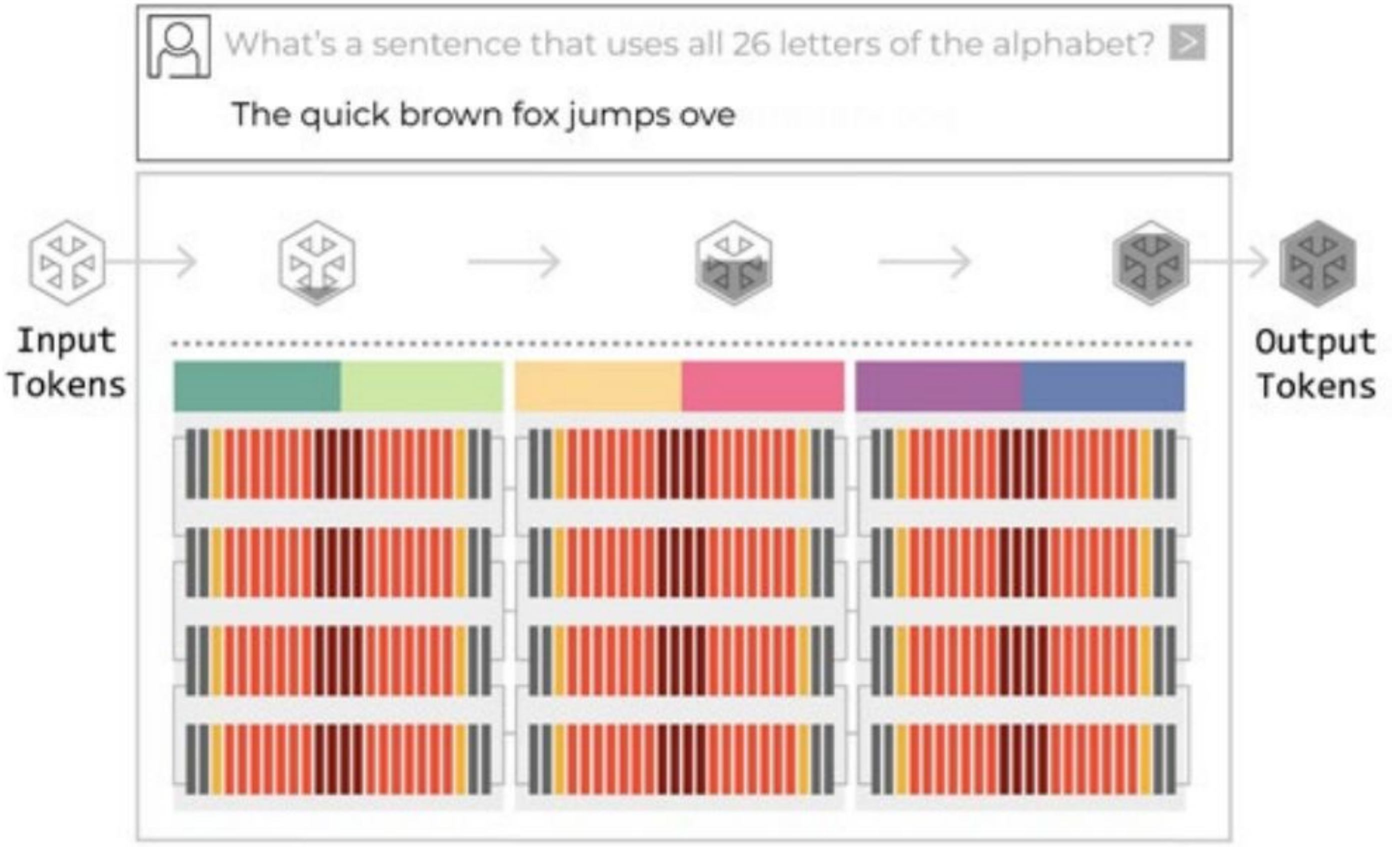
Moore's law of model knowledge density (capability/inference cost)



Model performance at different scale and time [11]

Where is the next 100x speed up?

Hardware super specialized for Transformer LLMs



[12] Grop https://groq.com/docs/

slido

Please download and install the
Slido app on all computers you use

Audience Q&A

ⓘ Start presenting to display the audience questions on this slide.