SGLang DeepSeek Model Optimizations

Ke Bao @ LMSYS Org

Content

1. DeepSeek MLA Optimizations

2. Data Parallelism Attention

3. DeepSeek-V3 Support & Optimizations

➔ 1. DeepSeek MLA Optimizations

MLA Introduction

MLA (**M**ulti-head **L**atent **A**ttention)[1] is an innovative attention architecture introduced by the DeepSeek-AI team, aimed at **improving inference efficiency**.

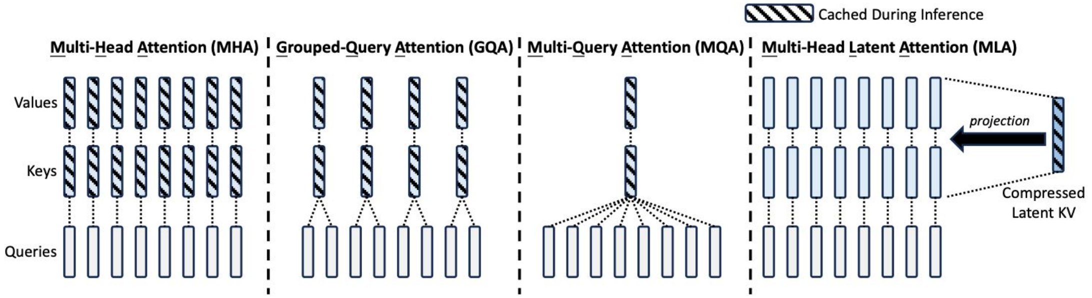
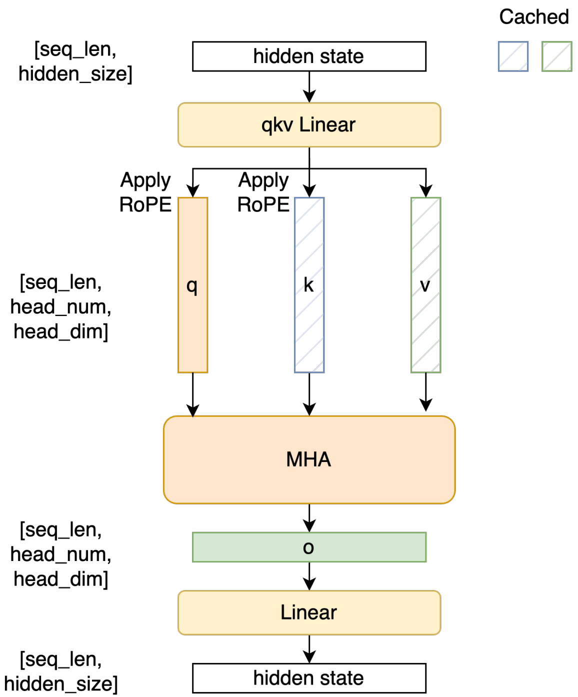


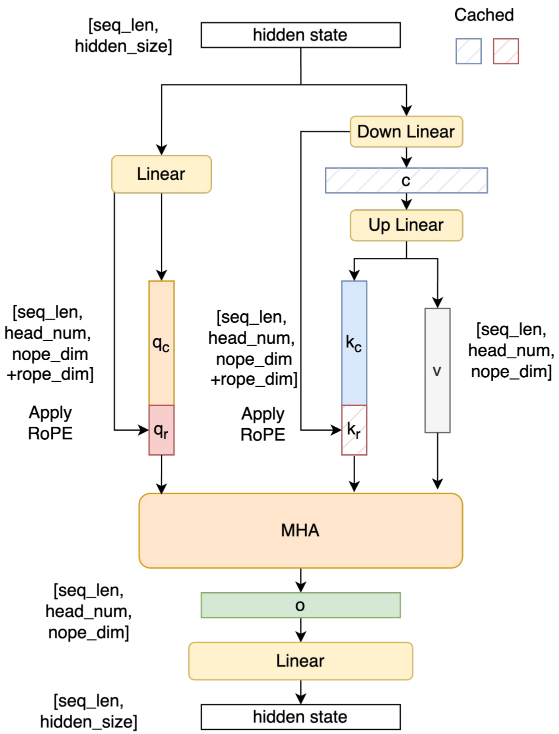
Figure 3 | Simplified illustration of Multi-Head Attention (MHA), Grouped-Query Attention (GQA), Multi-Query Attention (MQA), and Multi-head Latent Attention (MLA). Through jointly compressing the keys and values into a latent vector, MLA significantly reduces the KV cache during inference.

[1]DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model (https://arxiv.org/pdf/2405.04434)

Computation Overview



MHA



MLA

Weight Absorption

Change the computation order based on **associative law** of matrix multiplication.

$$q^\top k = [q_{nope}^\top; q_{rope}^\top][k_{nope}; k_{rope}]$$
$$= \boxed{q_{nope}^\top k_{nope}} + q_{rope}^\top k_{rope}$$

① orignal order

$$\underbrace{q_{nope}^\top}_{[n_q, h, d]} \underbrace{(W_{UK} c)}_{[n_k, h, d]}$$

② absorbed order

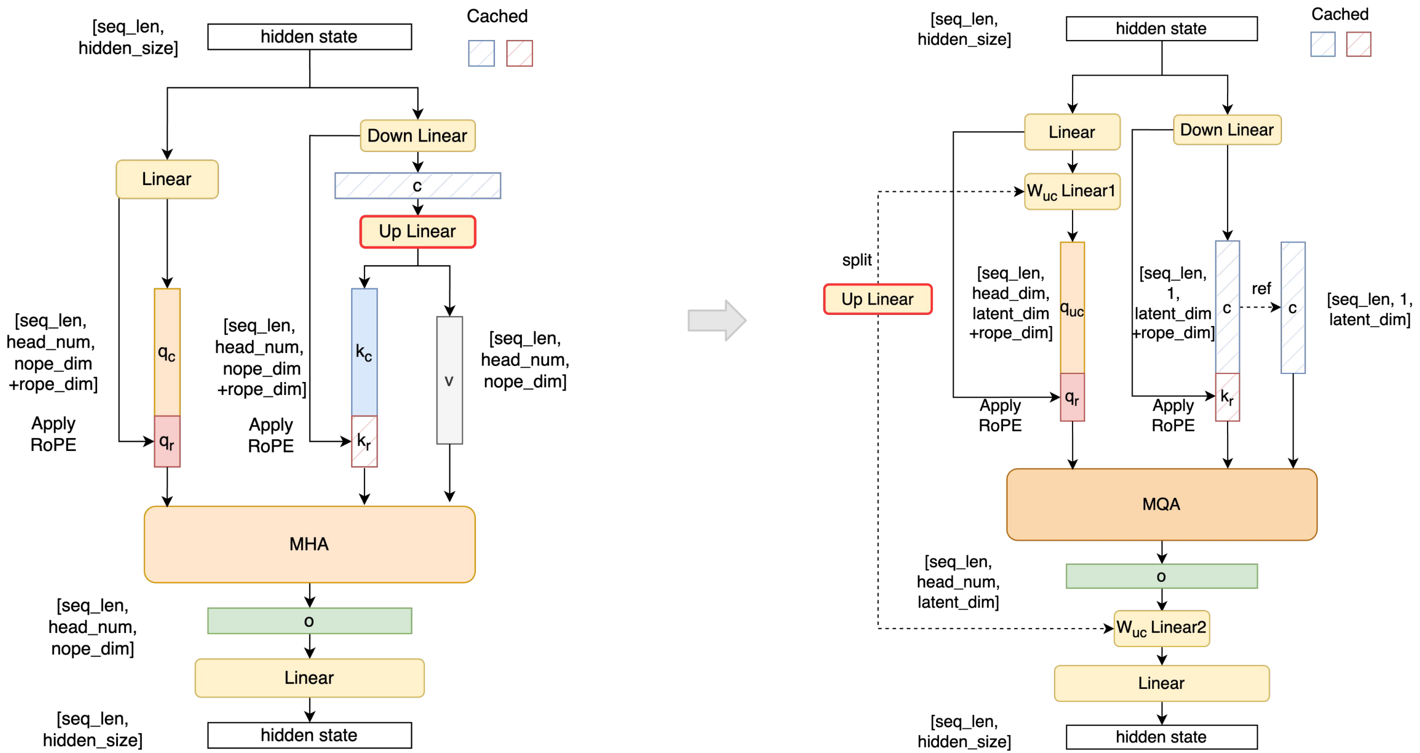
$$\underbrace{(q_{nope}^\top W_{UK})}_{[n_q, h, d_c]} \underbrace{c}_{[n_k, 1, d_c]}$$

FLOPs: $(2d_c - 1)hdn_k + (2d - 1)hn_q n_k$ $\quad$ $(2d - 1)hn_q d_c + (2d_c - 1)hn_q n_k$

In DeepSeek-V2,
- $d = 128$
- $d_c = 512$
- $h = 128$

In **decoding stage** (n_q=1), the method ② can take **less computation**.

Weight Absorption Implementation



Original

w/ Weight Absorption

Benefits & Results

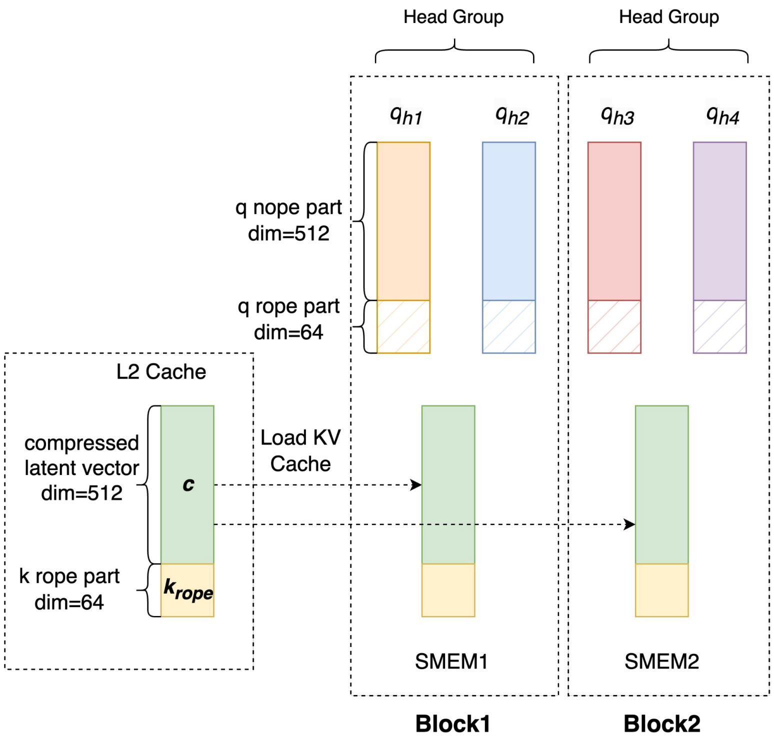
Benefits:

- Reduced overall computation in decoding stage
- Balanced the **computation** and **memory access** in decoding kernel
  - Increased the attention computation intensity
  - Reduced the memory access of KV cache

Results:

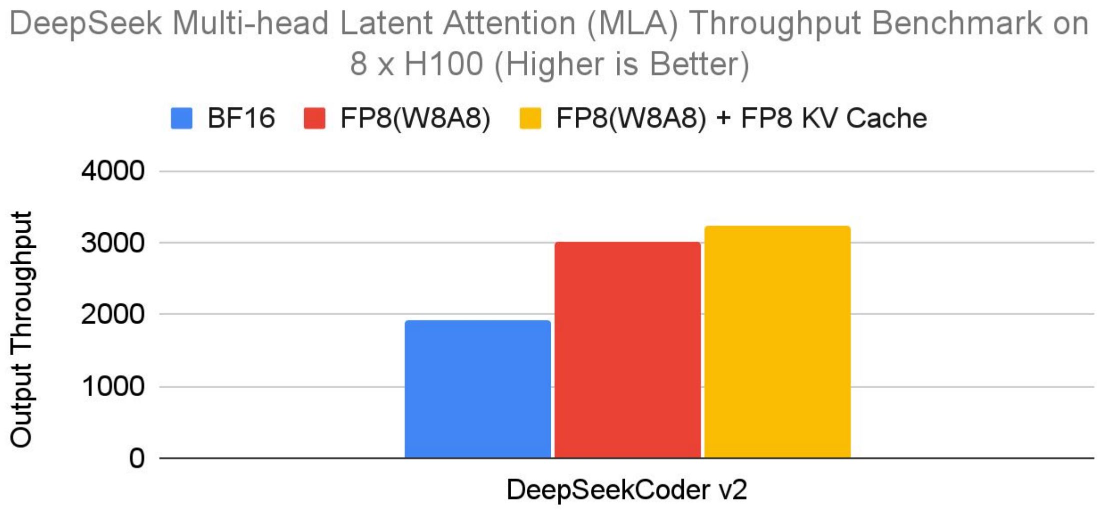
- Achieved **2.4x** throughput improvement for DeepSeek-V2 model.

Triton Decoding Kernel Optimization



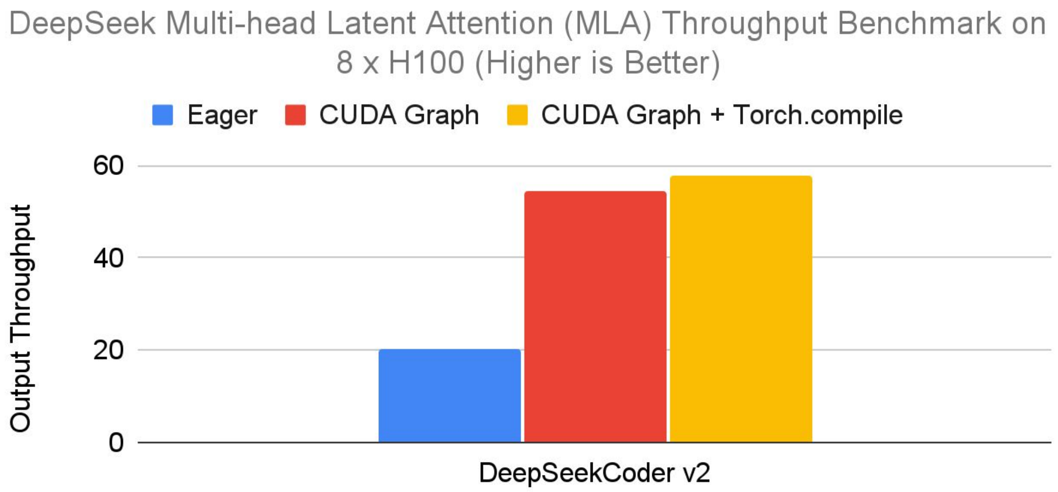
- In the MLA decoding kernel, there is **only one KV head** shared by many query heads.
- We optimized the Triton decoding kernel to reduce **memory access** to the KV cache by **processing multiple query heads within one Triton block**.
- Use **Tensor Core** to do the qk computation.
- Achieved **1.35x** throughput improvement for DeepSeek-V2 and **1.5x** for DeepSeek-V2-Lite.

FP8 Quantization



- Achieved **1.7x** throughput improvement with W8A8 FP8 and KV Cache FP8 quantization.
- Implemented **FP8 Batched MatMul** (BMM) operator to facilitate FP8 inference in MLA with weight absorption.

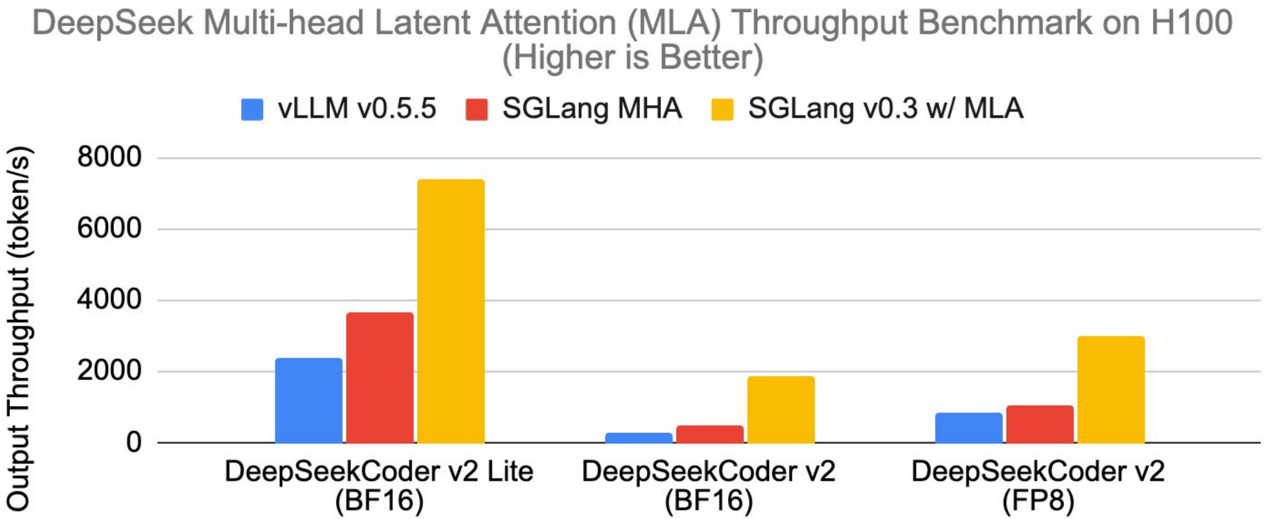
CUDA Graph & Torch Compile



(input_len=output_len=512, batch_size=1)

- MLA & MoE are compatible with CUDA Graph & Torch.compile
- **2.8x** decoding speed acceleration for **batch_size=1**

End2End Benchmark



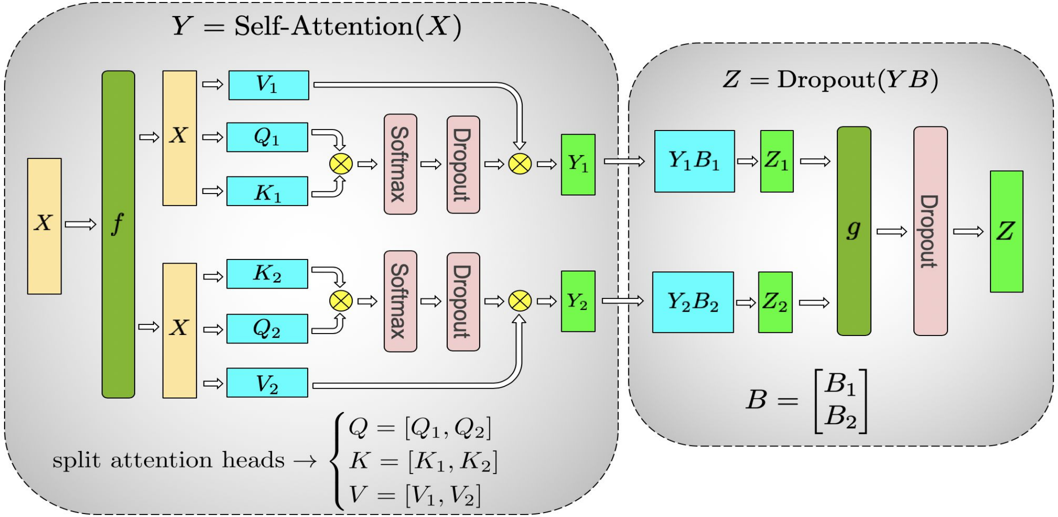
- Overall, we have achieved up to a **3~7x acceleration** in output throughput compared to the previous version.
- Source & Setup: https://lmsys.org/blog/2024-09-04-sglang-v0-3

➔ 2. Data Parallelism Attention

Tensor Parallelism Attention

The most common parallelism strategy for inference is **tensor parallelism** (TP)[1].

In the attention part, the weights and **attention heads** are split across multiple GPUs.



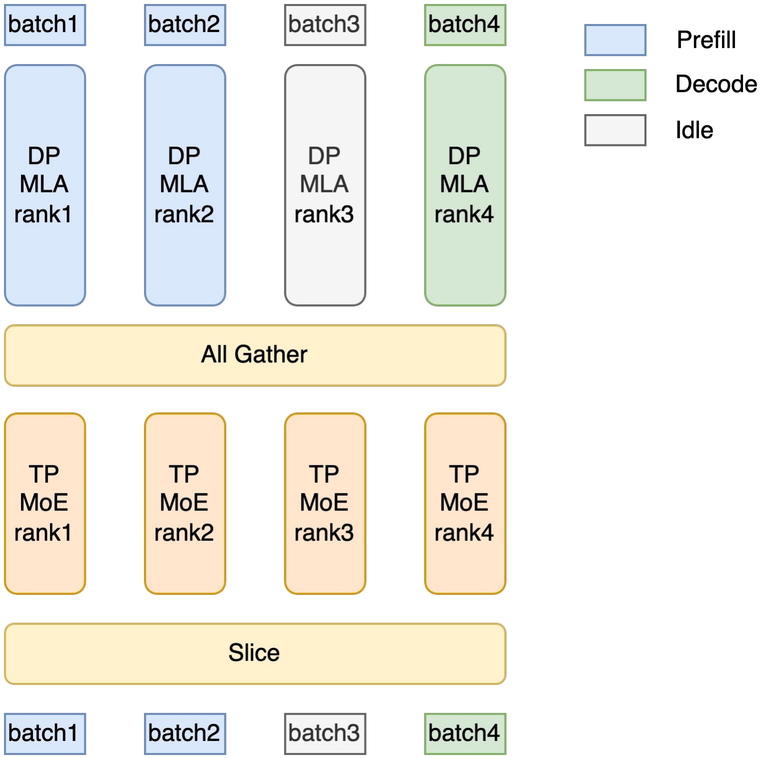
[1]Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism (https://arxiv.org/abs/1909.08053)

Observations

- Tensor parallelism might not be the most efficient strategy for models where the number of **KV heads** < the number of **GPUs available**.
  - For example, DeepSeek models use MLA and only have **one KV head** after weight absorption. If we use TP on 8 GPUs, it will lead to **duplicated KV cache** and unwanted memory usage.

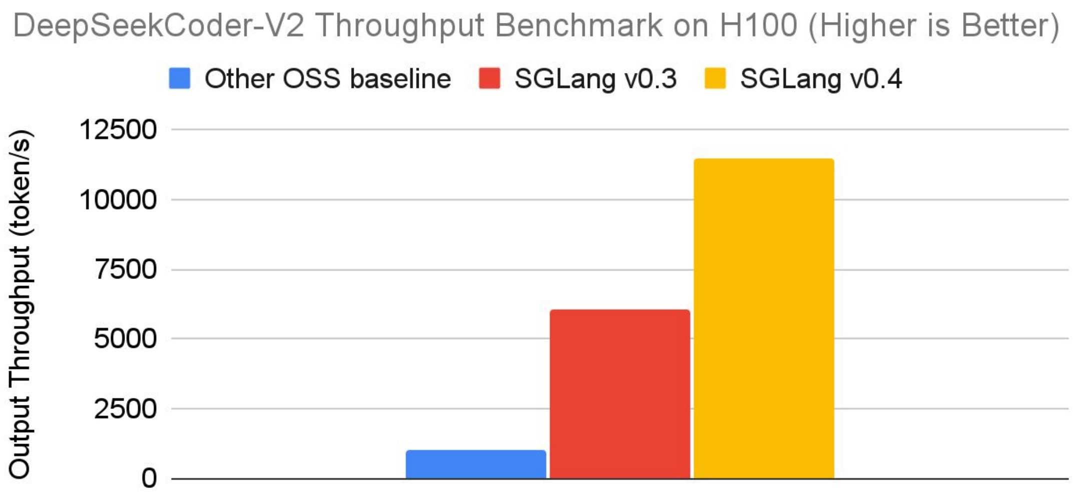
- For many MoE models, the parameters in the attention part take a **small proportion** of the total parameters. (**~3%** for DeepSeek-V2)
  - Allows duplicating the weights and using **data parallelism (DP)** for the attention part.

Data Parallelism Attention



- Use **DP** for the **MLA** mechanism to reduce KV cache overhead.
- Each DP worker handles **different types of batches (prefill, decode, idle)** independently.
- The attention-processed data will be **all-gathered** among all workers before the MoE layer, and will be **redistributed back** to each worker after the MoE.

Benchmark



- Benchmark results for **FP8** DeepSeekCoder-V2 model on **8 x H100** 80GB GPUs.
- Achieved **1.9x decoding throughput** improvement compared to SGLang v0.3.
- Source & Setup: https://lmsys.org/blog/2024-12-04-sglang-v0-4

➔ 3. DeepSeek-V3 Support & Optimizations

DeepSeek-V3 Support

We have supported [DeepSeek-V3]() on SGLang from day one.

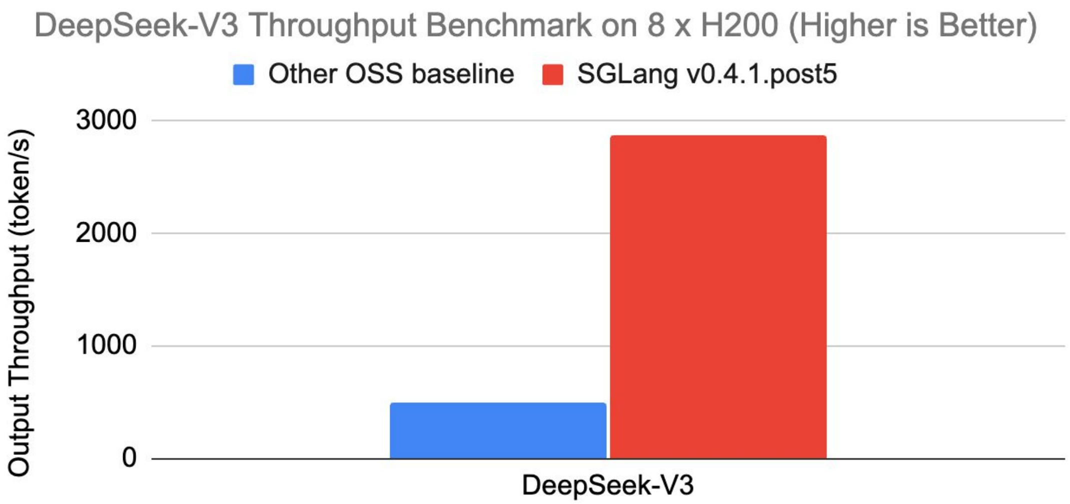
- v0.4.1 release: https://github.com/sgl-project/sglang/releases/tag/v0.4.1
- Usage: https://github.com/sgl-project/sglang/tree/main/benchmark/deepseek_v3

Optimizations for DeepSeek-V2 are effective for DeepSeek-V3 as well.

Further work we have done:

- No-aux MoE gate support
- FP8 block-wise quantization & kernel tuning
- MoE kernel optimizations
- Compatible with CUDA graph
- Multi-node TP inference support

Benchmark



- Large QPS: Achieved **~3000 tok/s** output throughput on ShareGPT dataset.
- Batch size 1: Achieve **37 tok/s** output throughput.

Further Optimizations

- Next-N speculative decoding
- TP + DP Attention
- Multi-node DP Attention
- Implement FP8 GEMM kernel with CUTLASS and CK
- MoE fused topk kernel

Optimization plan:

- https://github.com/sgl-project/sglang/issues/2591

Doc:

- https://sgl-project.github.io/references/deepseek.html

Q & A

Welcome to join our Slack and use SGLang!