Systems Design For Efficient MoE-based LLM Inference

Pavan Miriyala, Haris Javaid

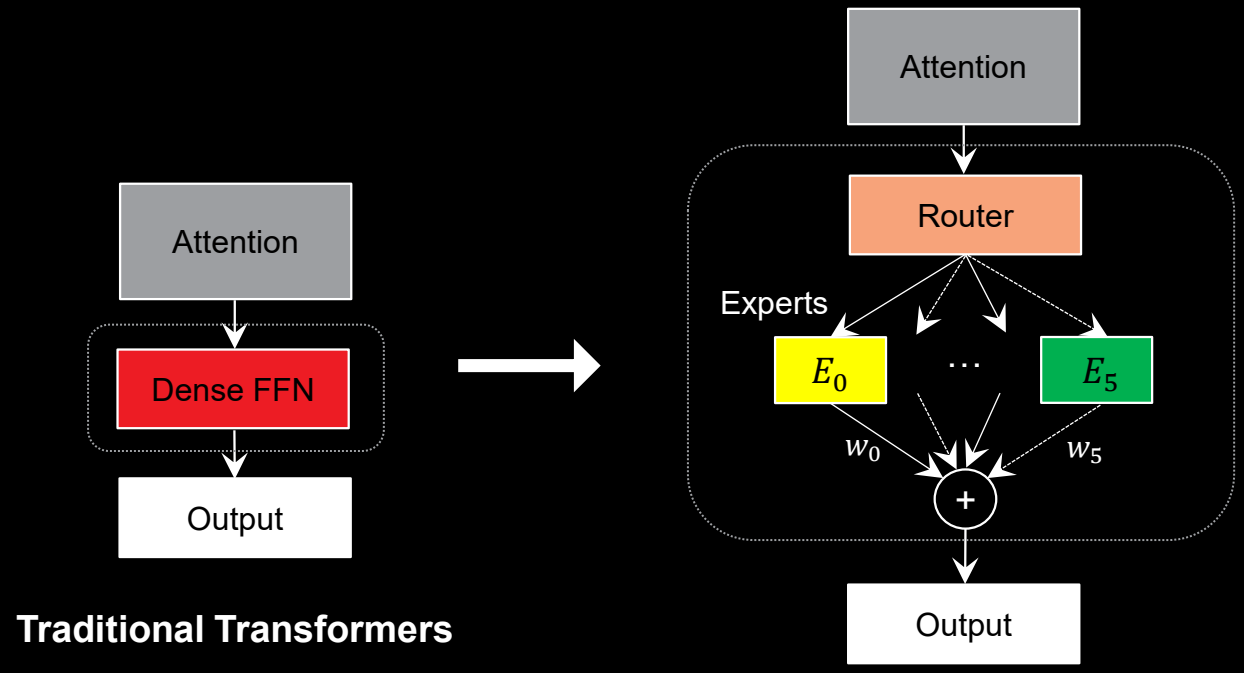
AMD Research and Advanced Development (RAD), Singapore.

AMD
together we advance_

Contents

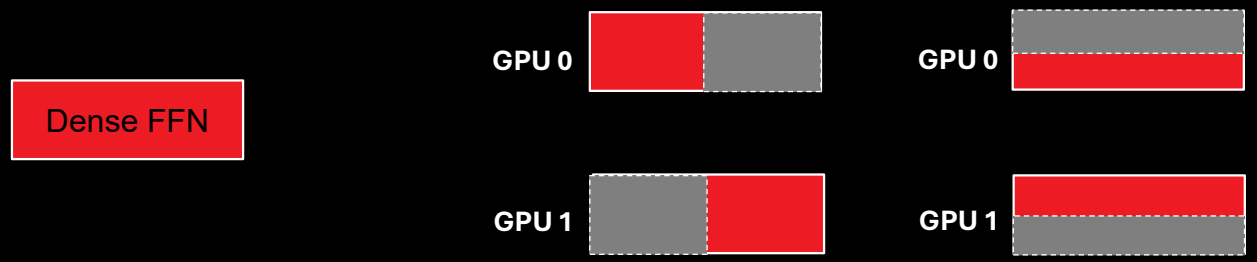
Contents

Mixture-of-Experts (MoE) Architecture

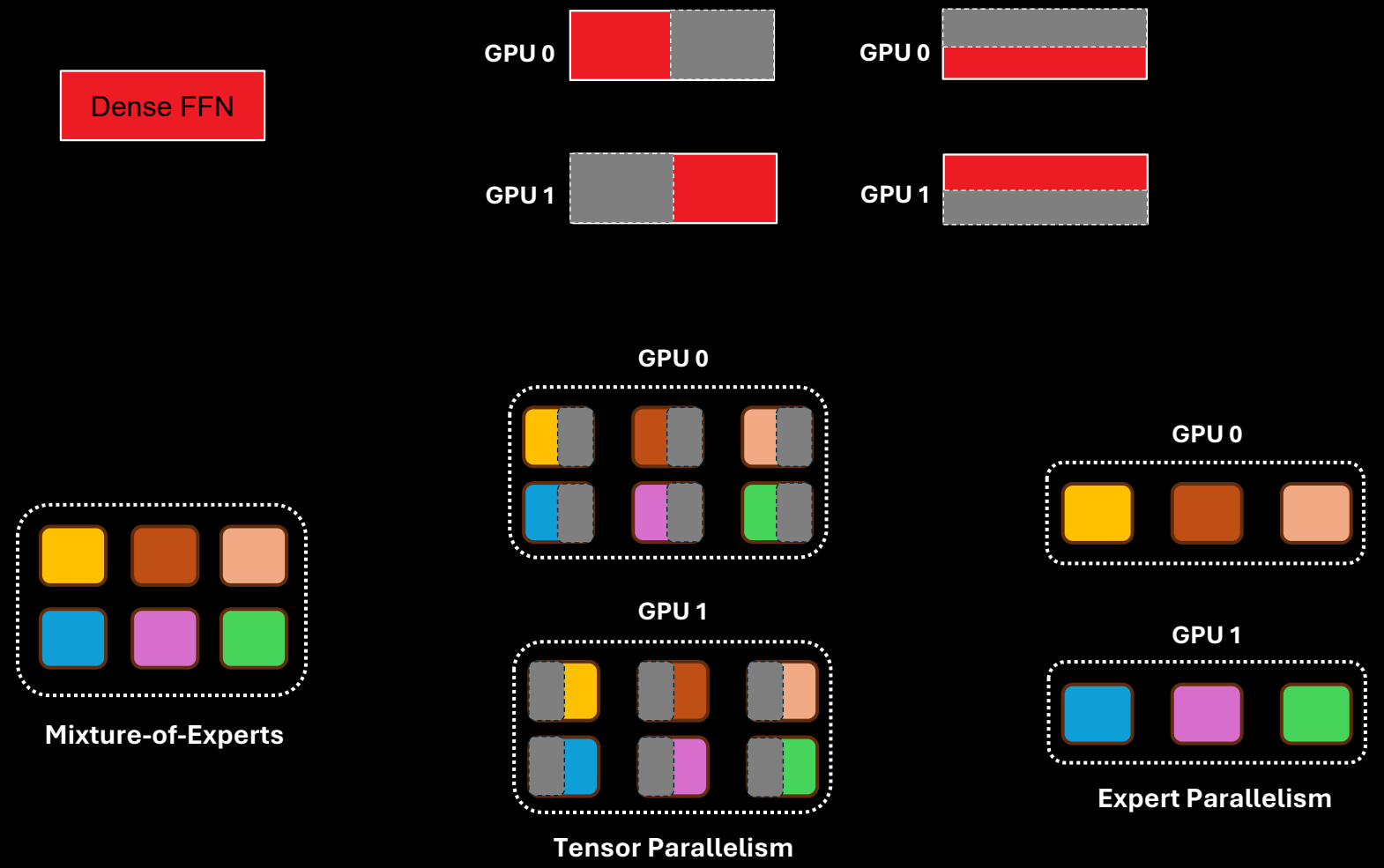


Ashish Vaswani, et. al., Attention Is All You Need, https://arxiv.org/abs/1706.03762, 2017.

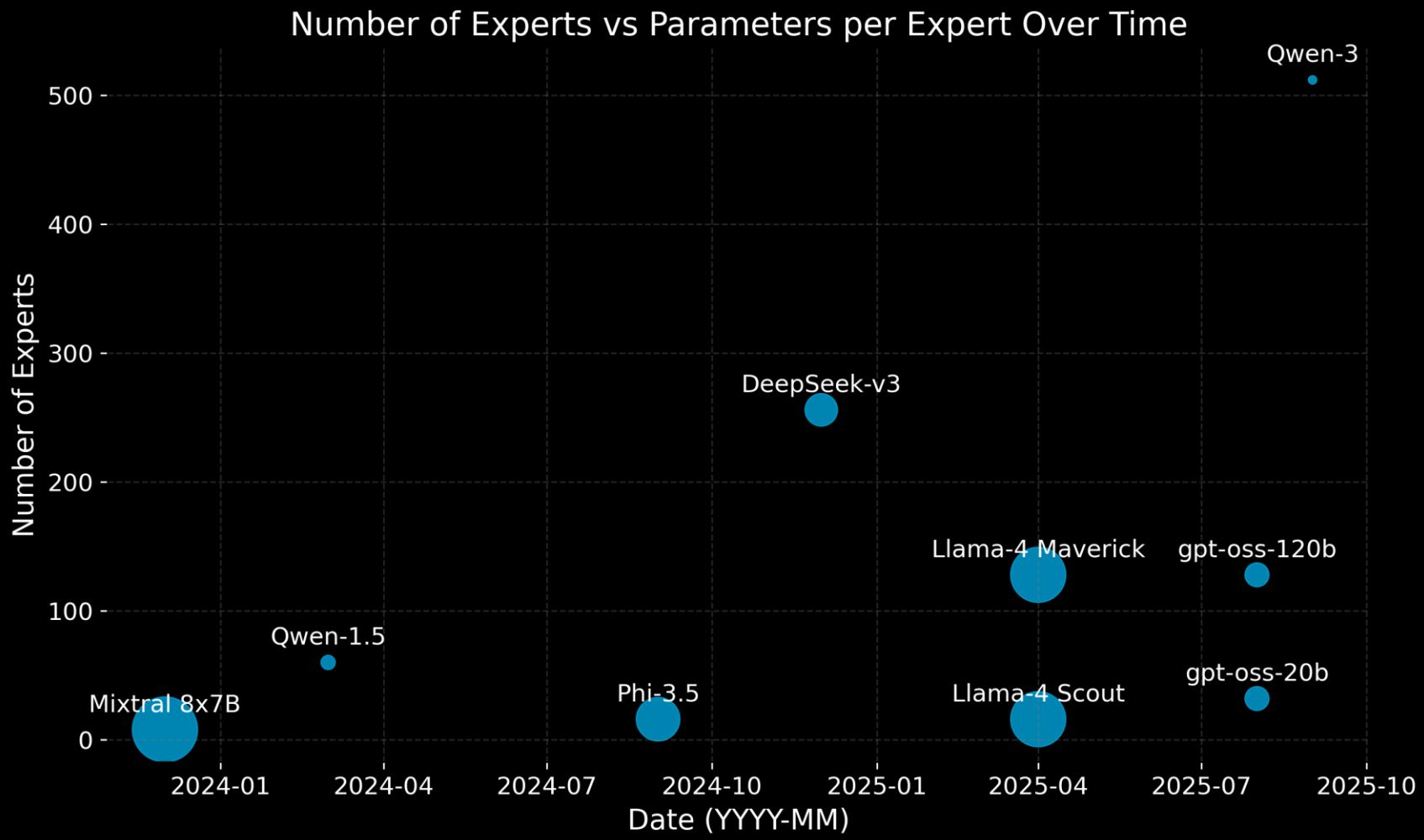
Tensor Parallelism



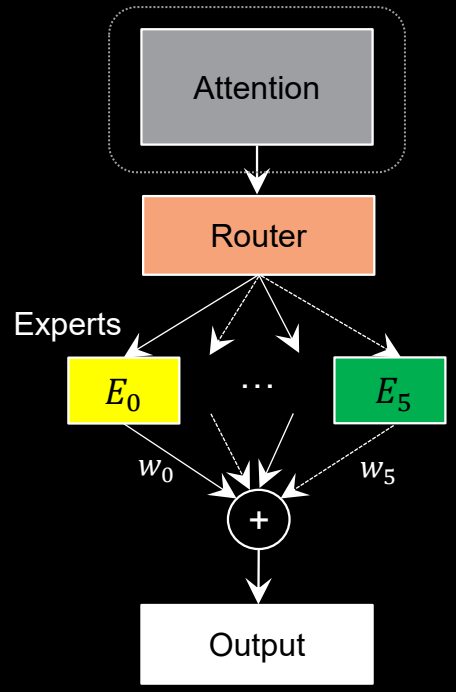
Expert Parallelism: A New Parallelization Technique Has Emerged with MoEs



Trends in MoE Architectures

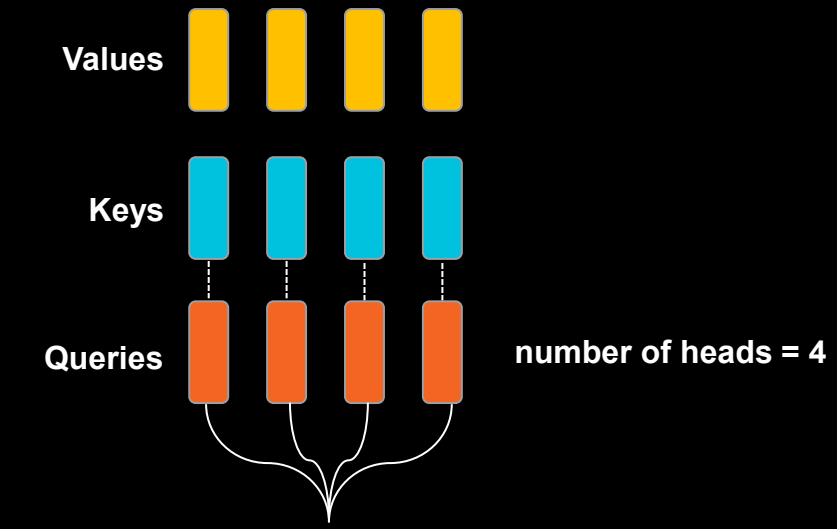


Attention Layer



Mixture-of-Experts (MoE) Transformers

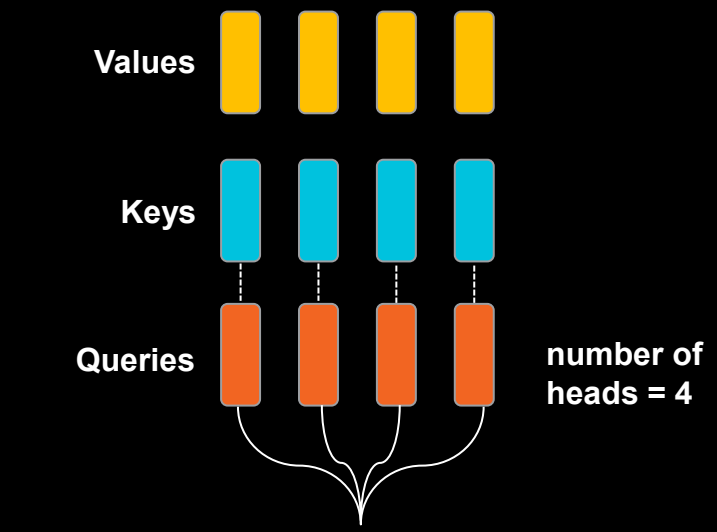
Multi-Head Attention (MHA)

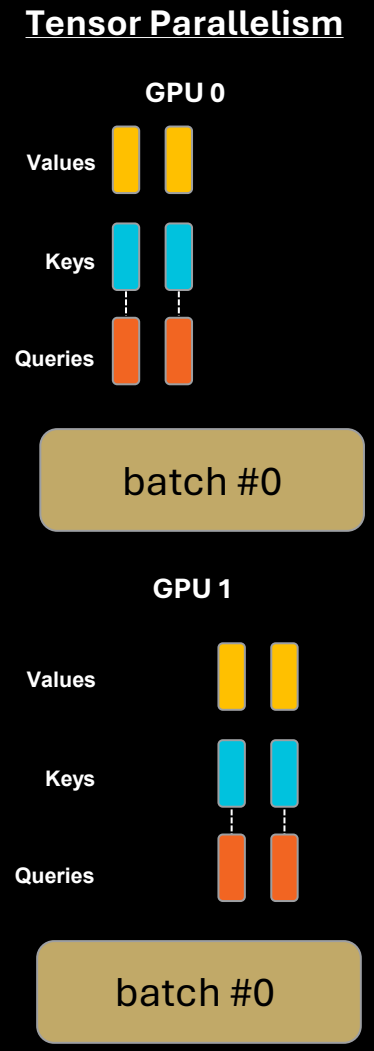


Prompt: The quick brown fox jumps over

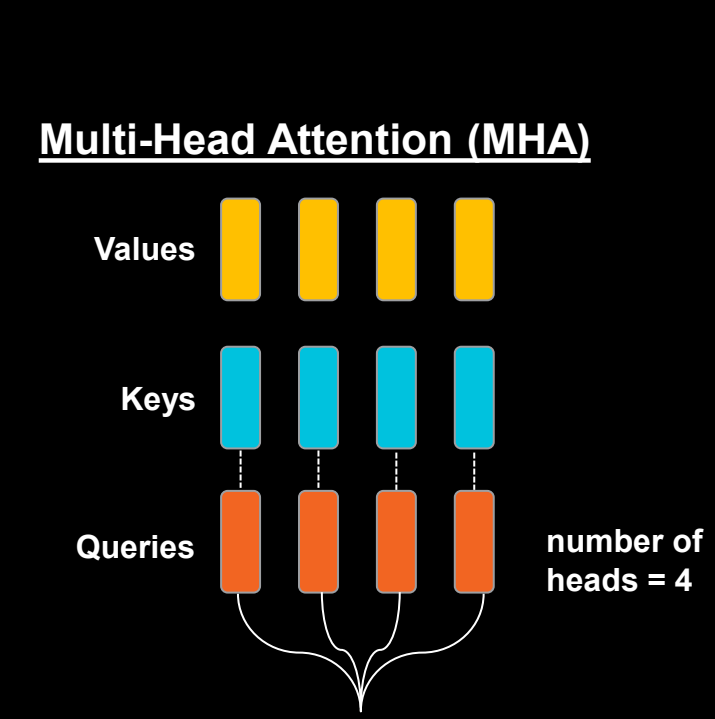
Ashish Vaswani, et. al., Attention Is All You Need, https://arxiv.org/abs/1706.03762, 2017.

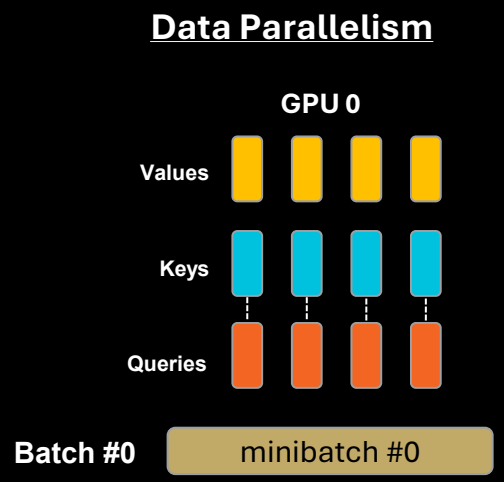
Multi-Head Attention Parallelization: Tensor Parallelism

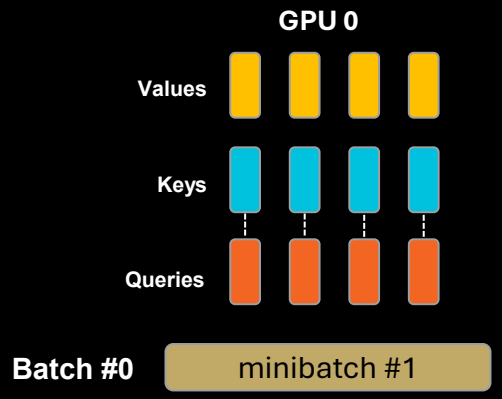




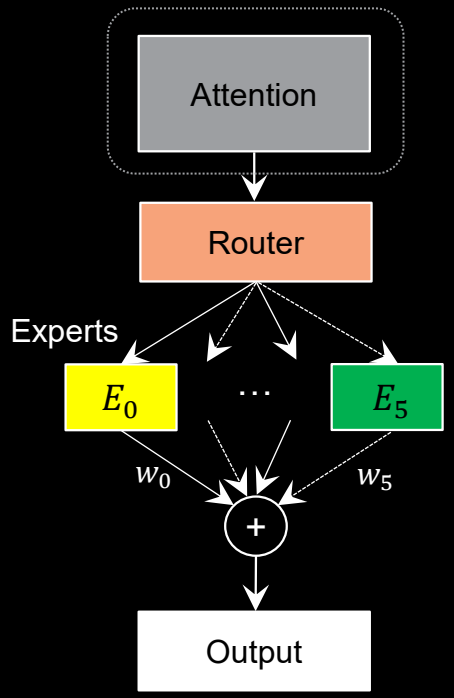
Multi-Head Attention Parallelization: Data Parallelism





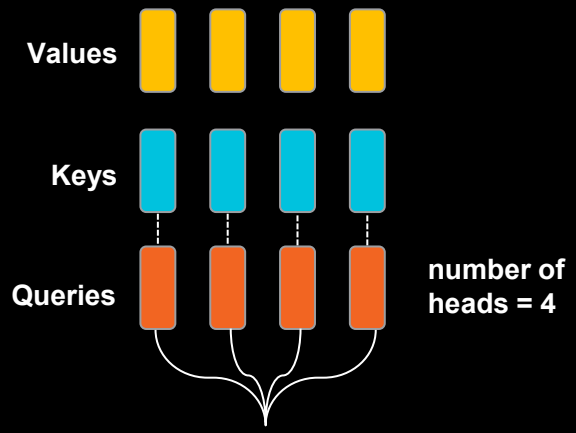


Trends in Attention Layers



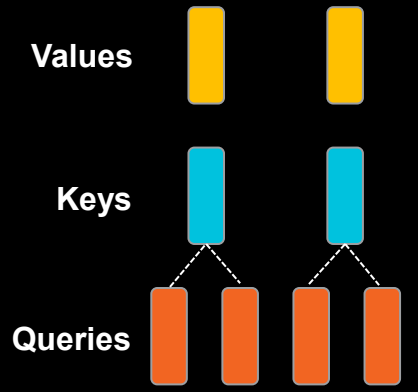
Mixture-of-Experts (MoE) Transformers

Multi-Head Attention (MHA)



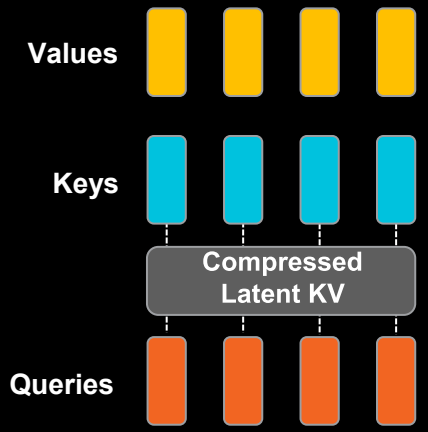
Prompt: The quick brown fox jumps over

Group-Query Attention (GQA)



Example: Mixtral 8x7B

Multi-Head Latent Attention (MLA)

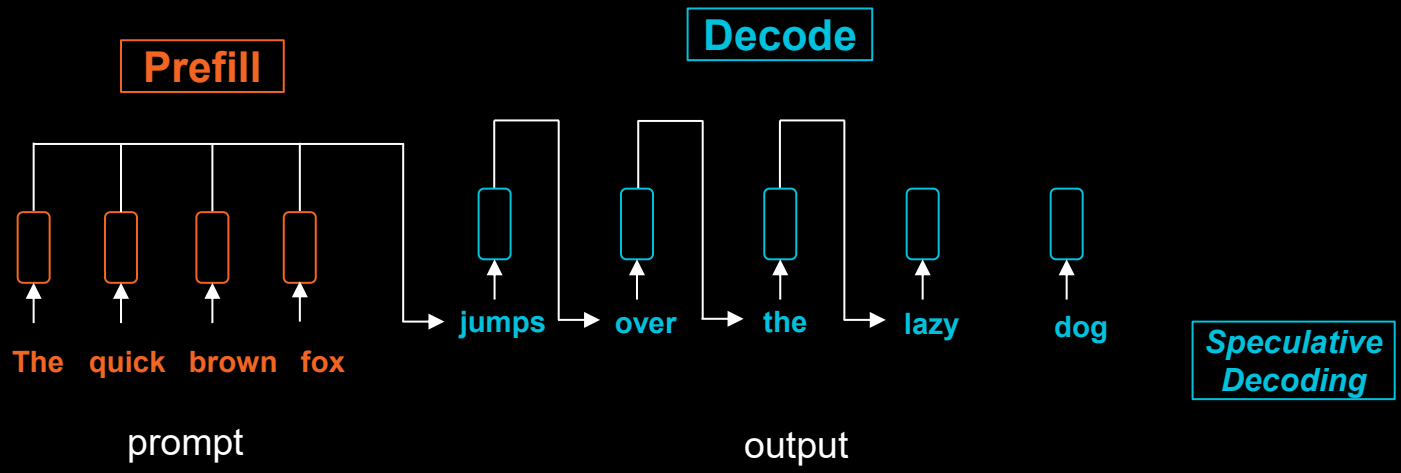


Example: Deepseek-V3

Albert Q. Jiang, et. al., Mixtral of Experts, https://arxiv.org/abs/2401.04088, 2024.
DeepSeek-AI, DeepSeek-V3 Technical Report, https://arxiv.org/abs/2412.19437, 2024.

Contents

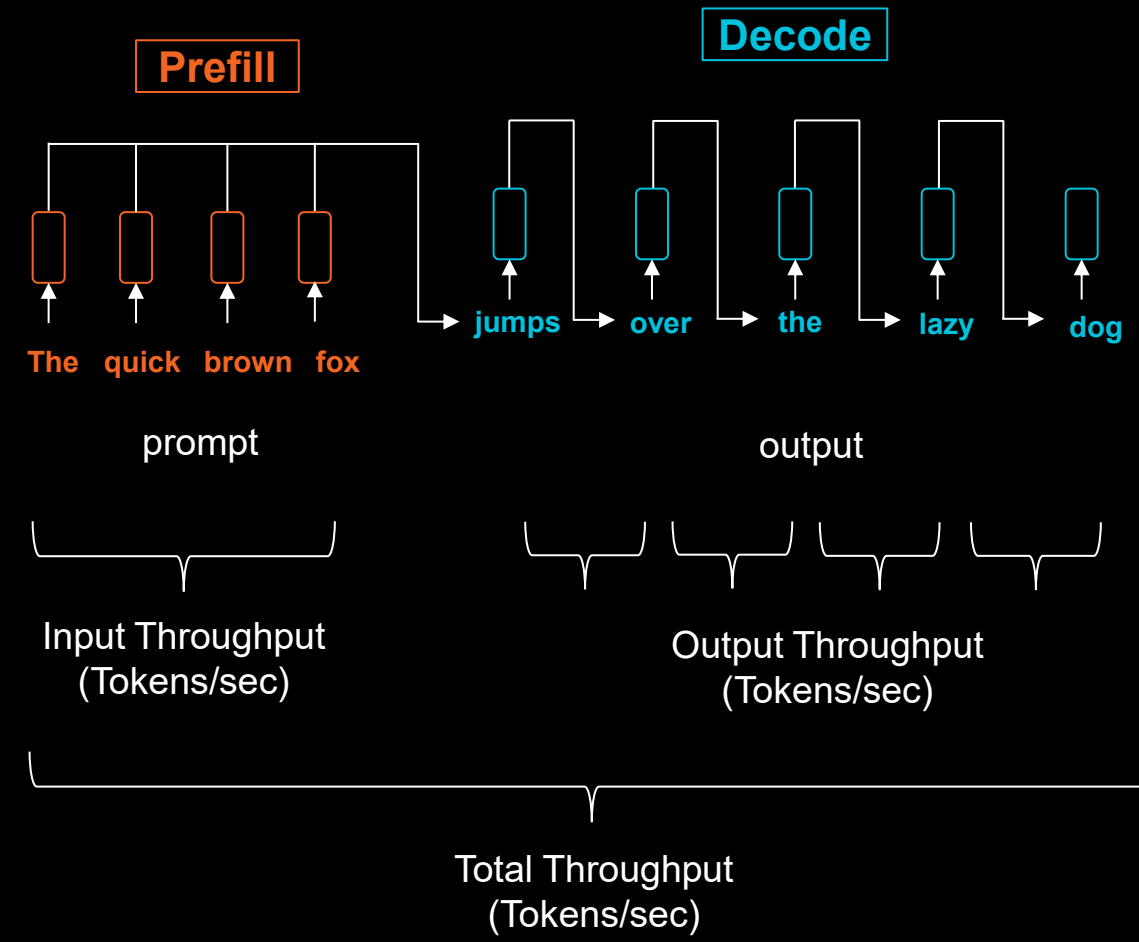
Large Language Model (LLM) Inference



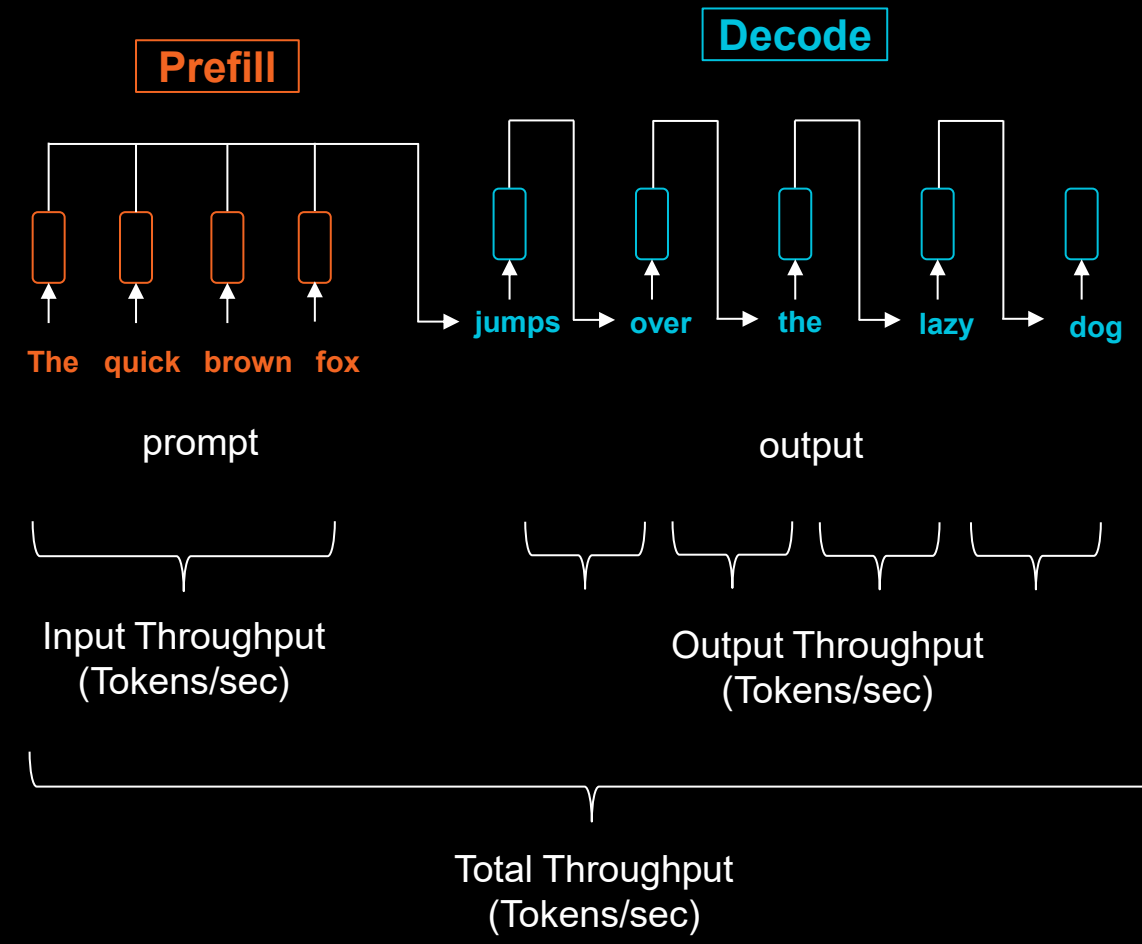
KV Vectors
- *Computation (Cache fill) during prefill stages*
- *Memory access (Cache read) during decode stages*

KV Vectors
- *Memory access (KV cache filled during prefill (orange blocks))*
- *Computation (Cache fill) during decode stages*
- *Memory access (KV cache filled during decode stages (previous blue blocks))*

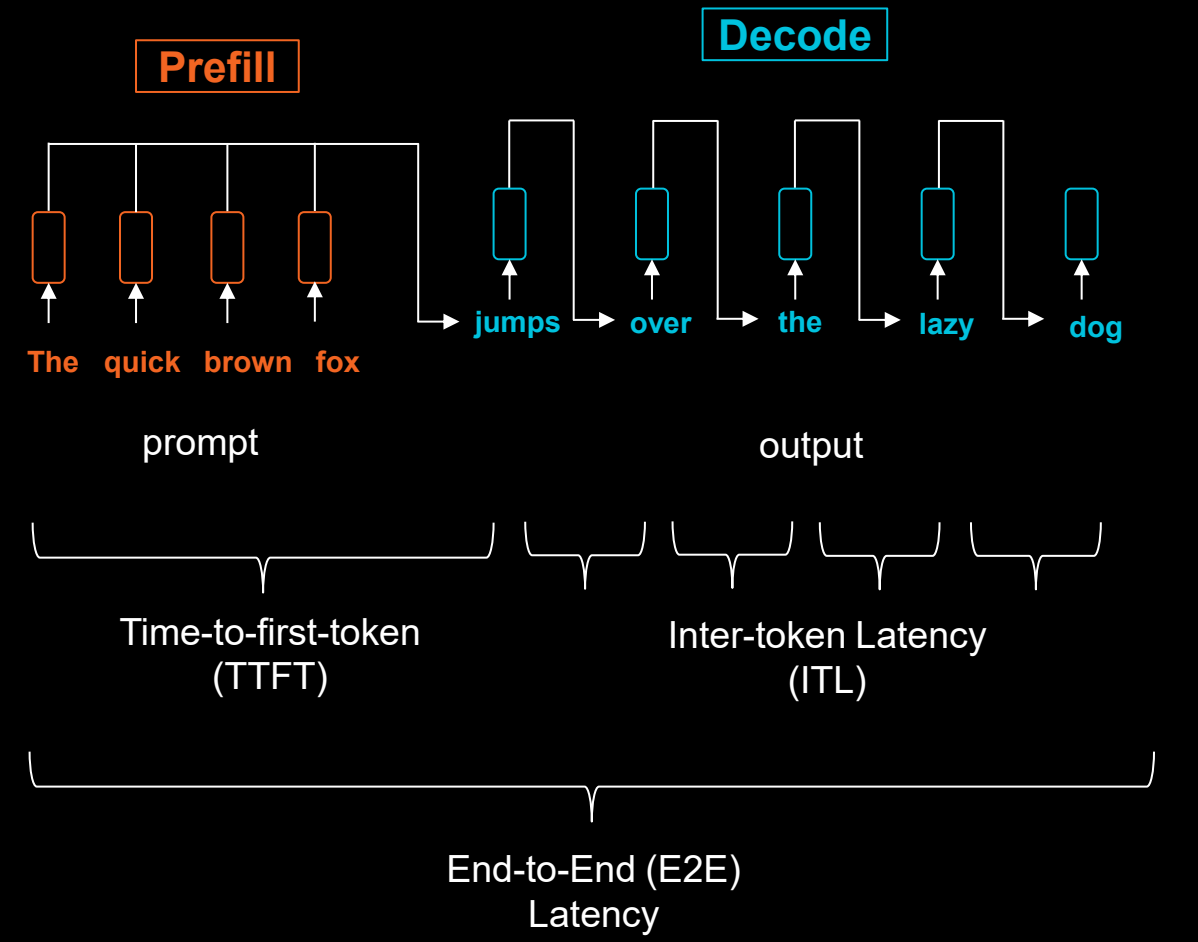
LLM Performance Metrics (Throughput)



Increased Throughput Leads To Reduced Costs



LLM Performance Metrics (Latency)



Contents

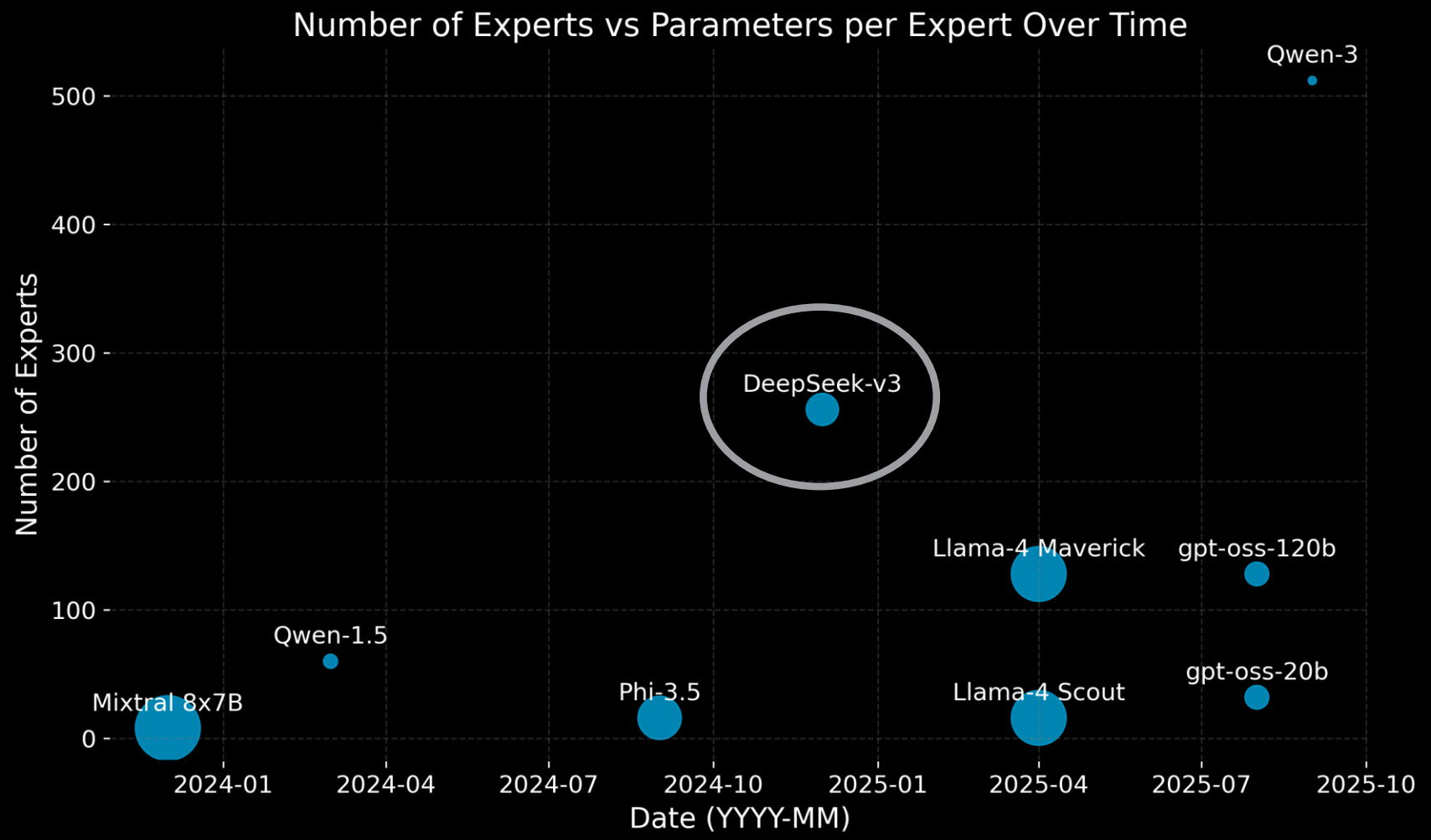
Hands-on Tutorial [SGLang, vLLM]

SGLang: https://docs.sglang.ai/
vLLM: https://github.com/vllm-project/vllm

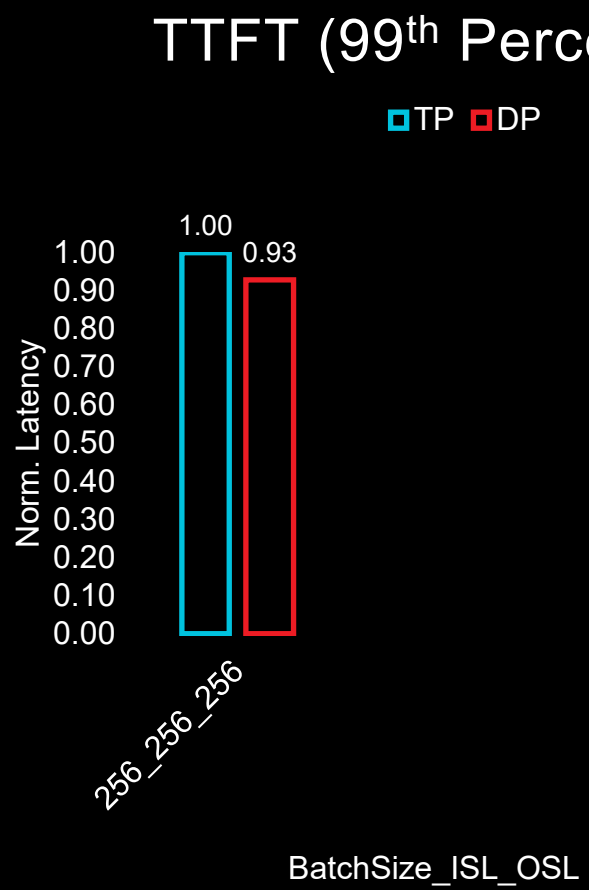
Example Workloads

BatchSize_ISL_OSL	Example Workload
256_256_256	Conversation/translation/paraphrase questions
4096_256_256	Higher batch size Conversation/translation/paraphrase questions
256_256_4096	Heavy reasoning tasks
256_4096_256	Summarization tasks

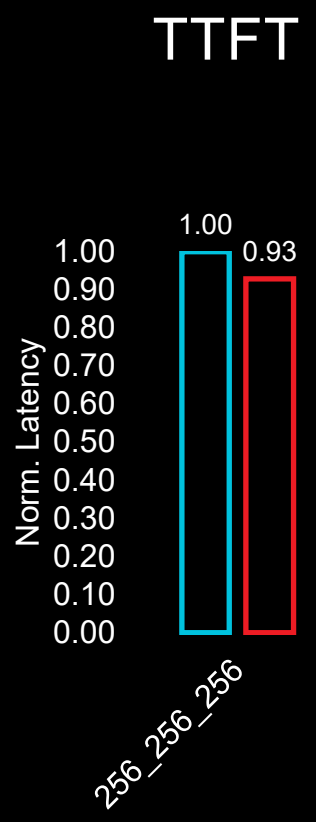
Trends in MoE Architectures

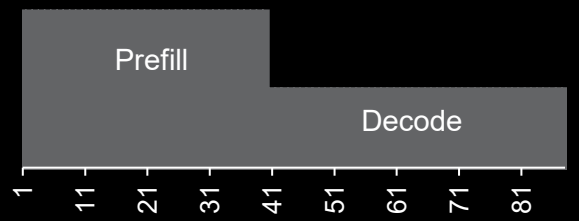


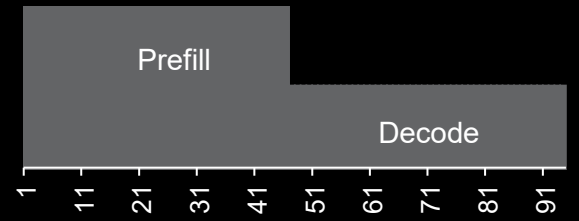
Tensor Parallelism (TP) vs. Data Parallelism (DP) Attention [DeepSeek-V3]



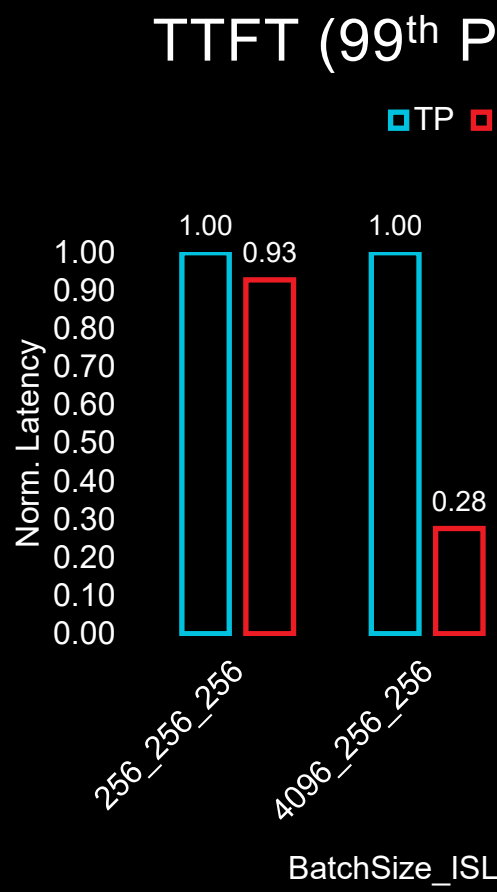
TP vs. DP Attention [DeepSeek-V3]

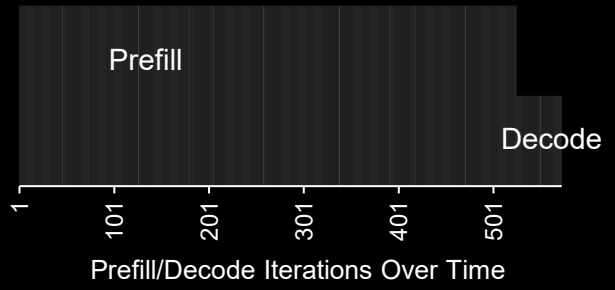






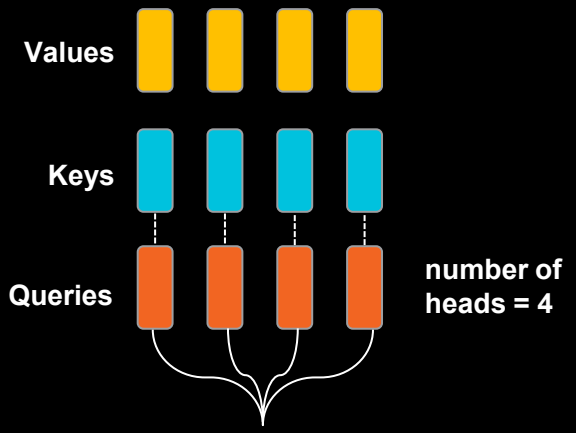
TP vs. DP Attention [DeepSeek-V3]





TP vs. DP Attention [DeepSeek-V3]

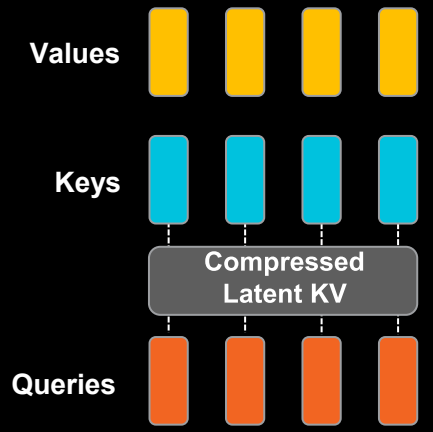
Multi-Head Attention (MHA)



Prompt: The quick brown fox jumps over

l: num of attention layers
b: batch size
s: context length
n: num of heads
h: head dimension
c_kv: compressed latent vector

Multi-Head Latent Attention (MLA)



Example: Deepseek-V3

Memory requirements: l x b x s x n x h x 2

With TP:

l x b x s x (n / TP) x h x 2

With DP:

l x (b / DP) x s x n x h x 2

Memory requirements: l x b x s x c_kv

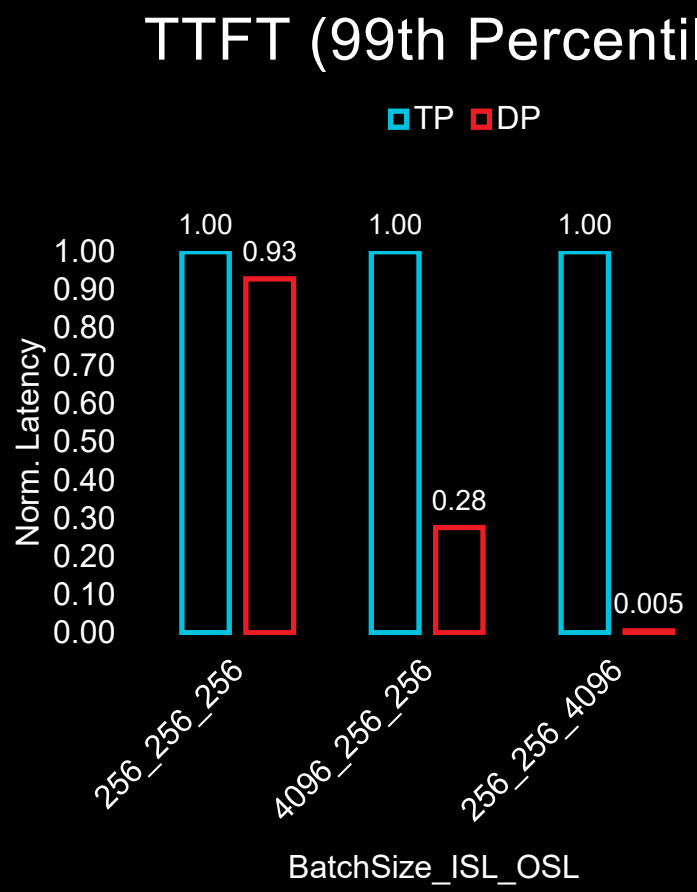
With TP:

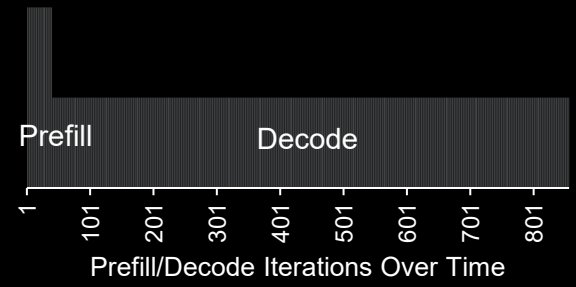
l x b x s x c_kv

With DP:

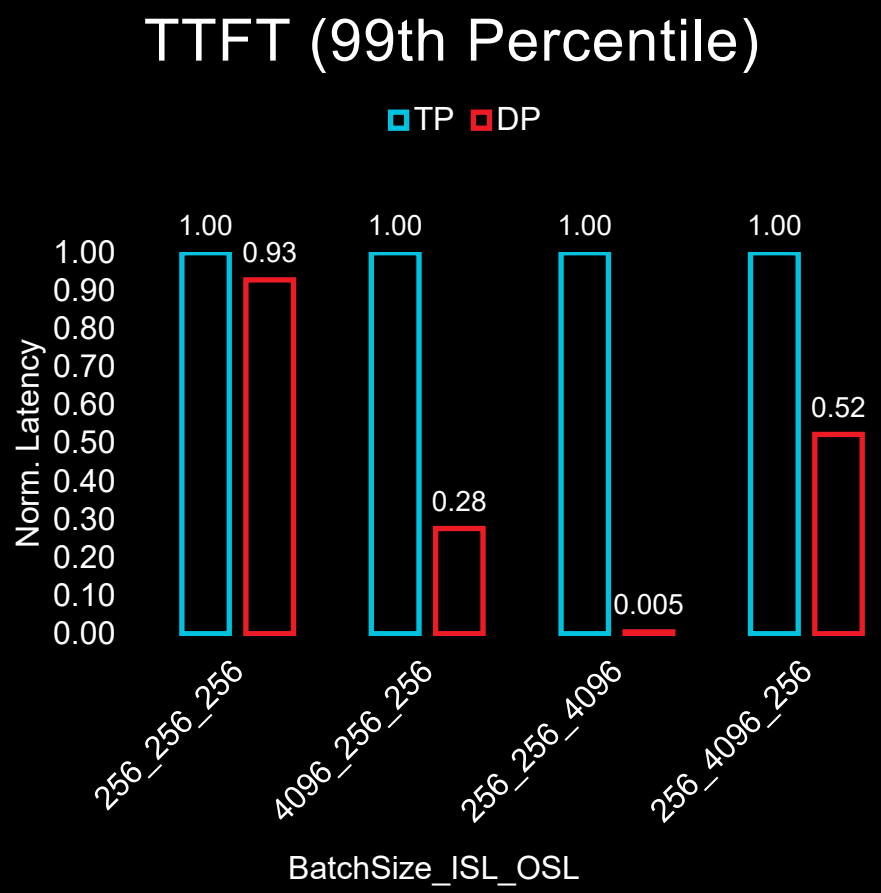
l x (b / DP) x s x c_kv

Tensor Parallelism (TP) vs. Data Parallelism (DP) Attention [DeepSeek-V3]

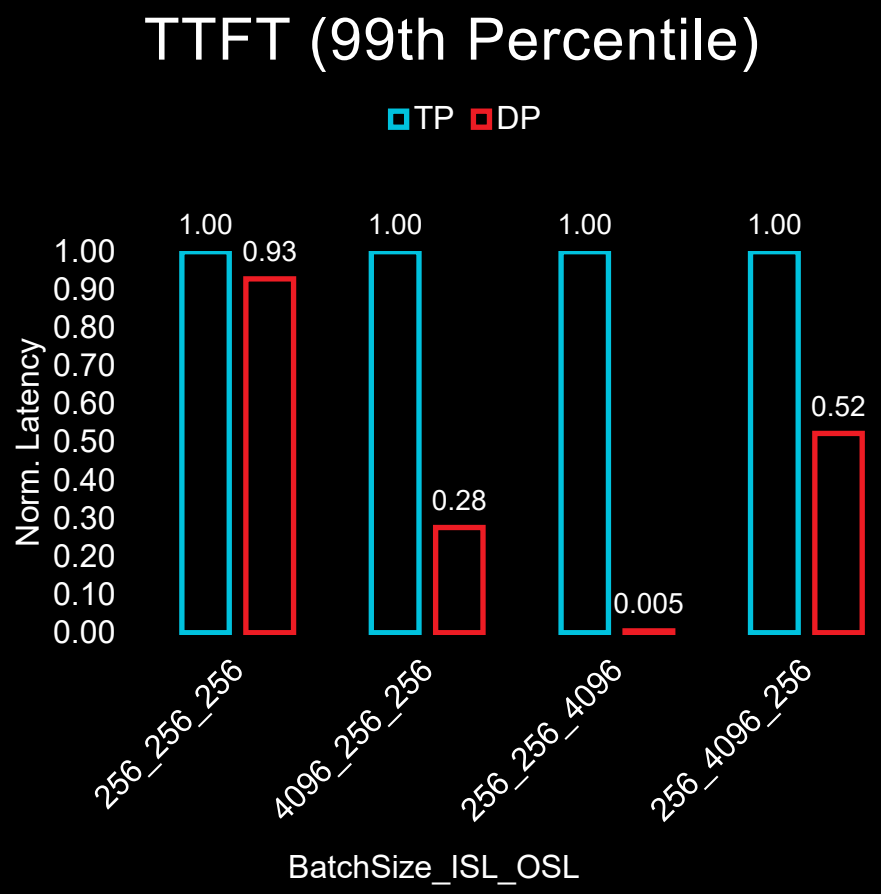


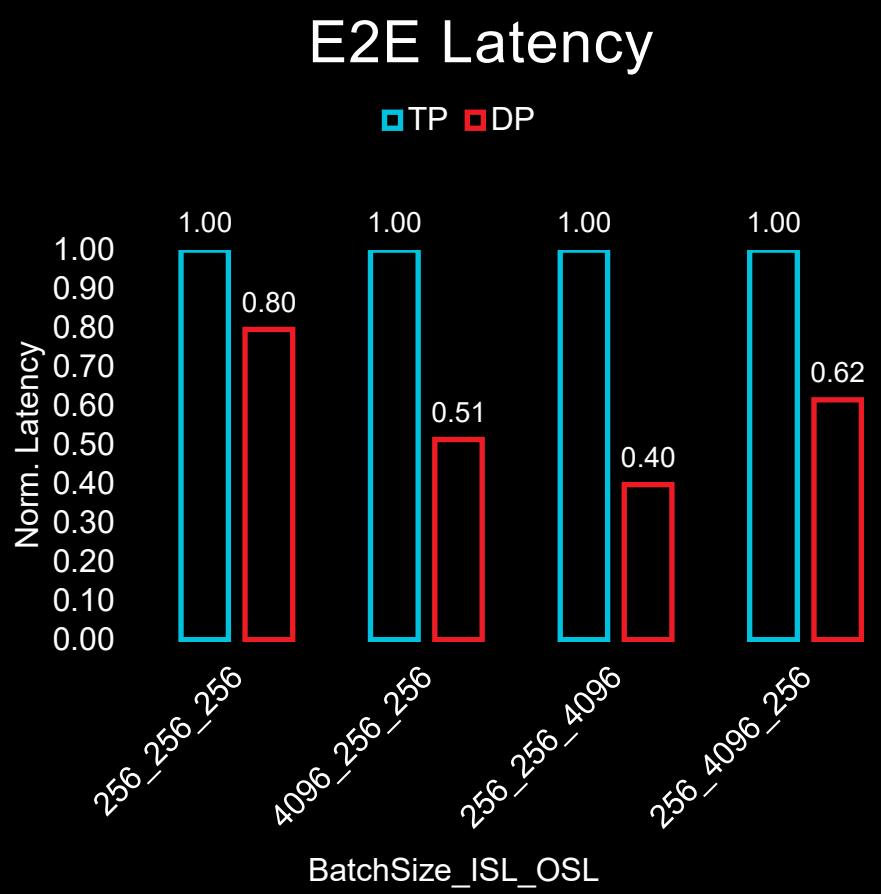


TP vs. DP Attention [DeepSeek-V3]



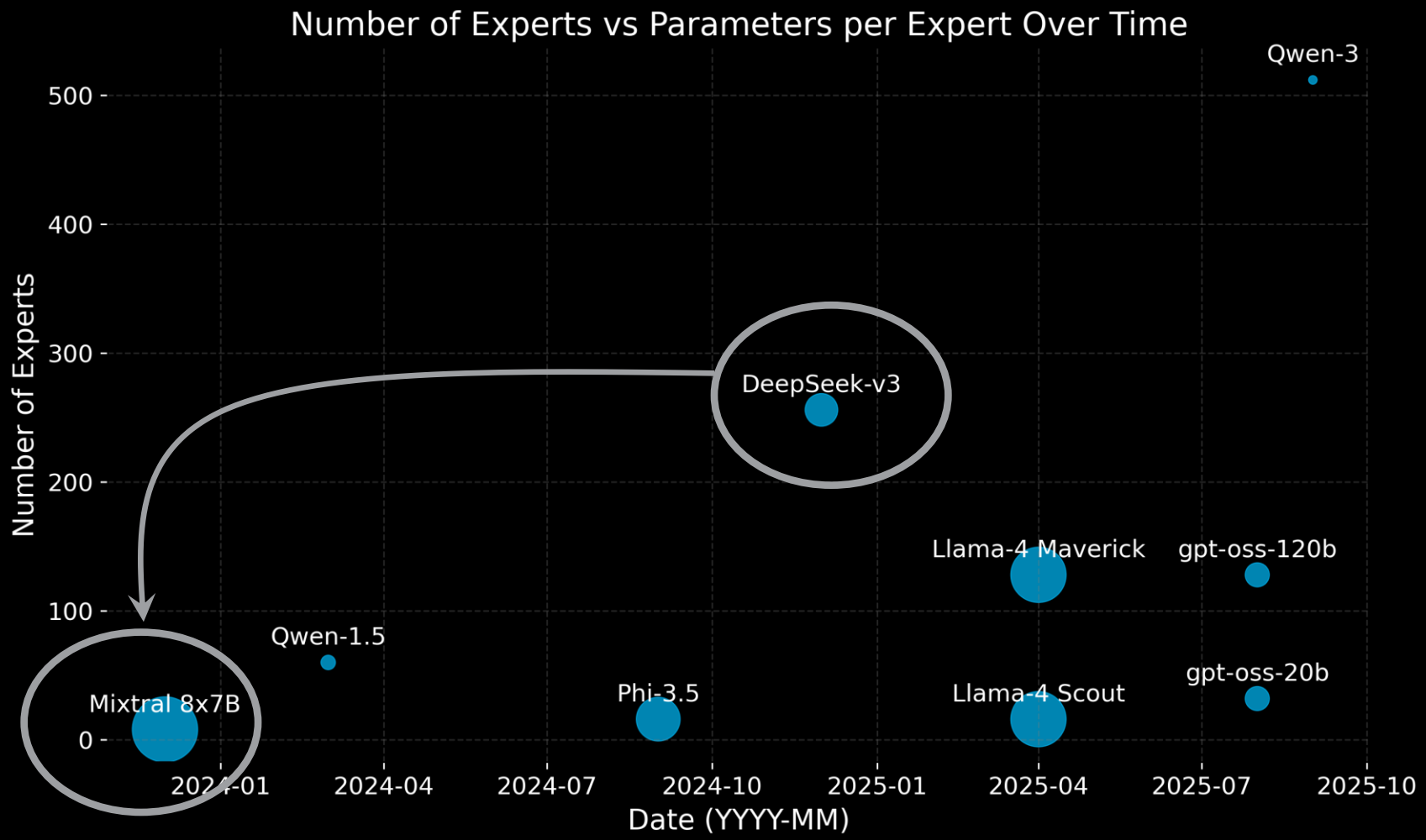
TP vs. DP Attention [DeepSeek-V3]



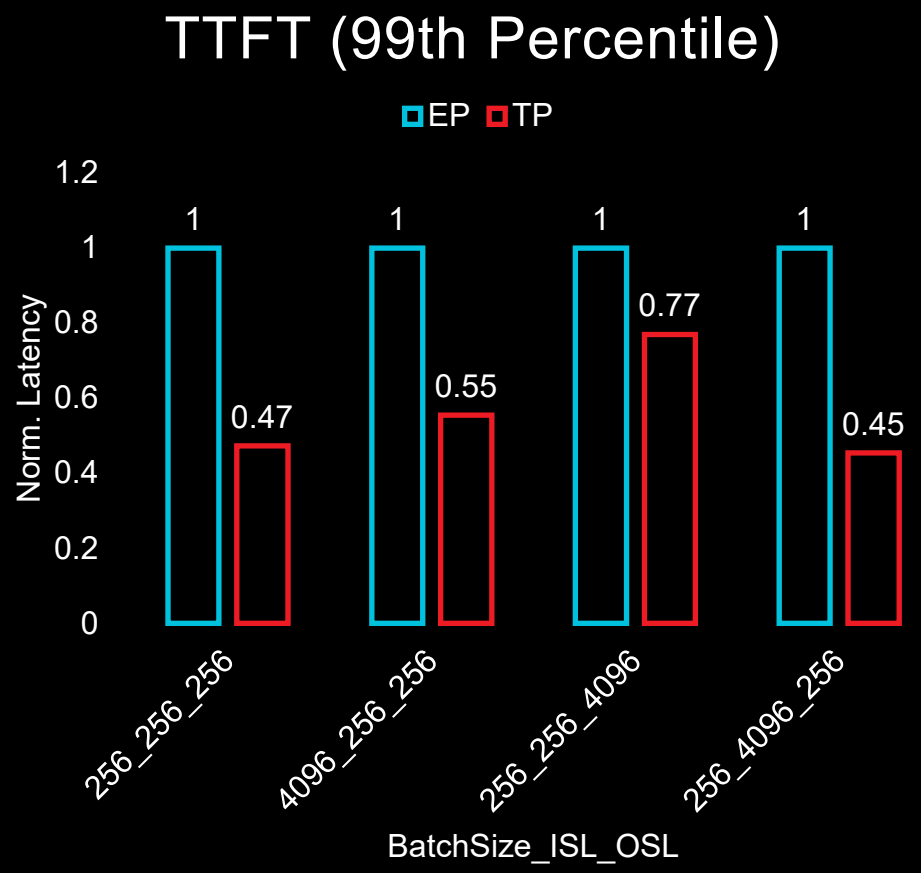


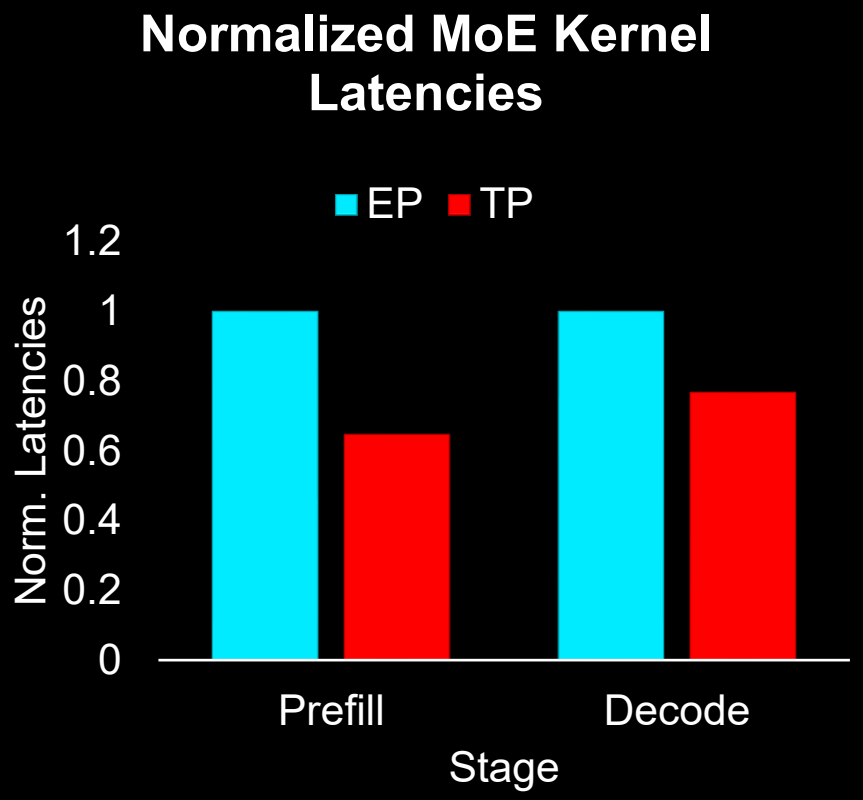
DP has lower latencies in all the cases

Trends in MoE Architectures

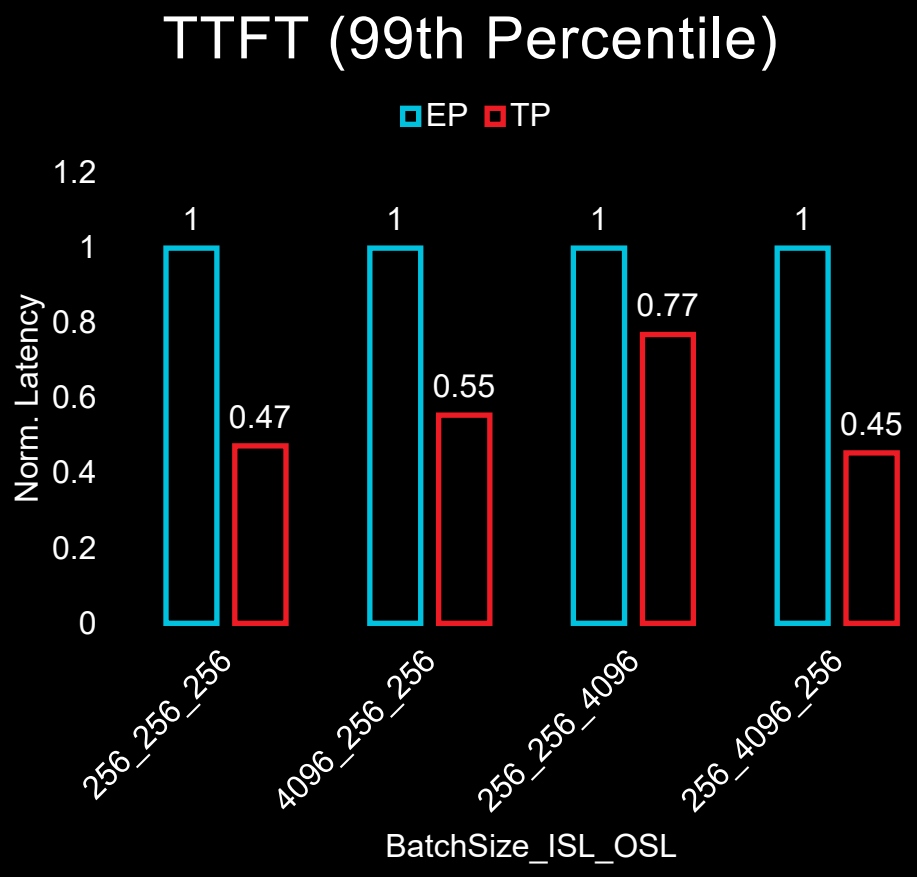


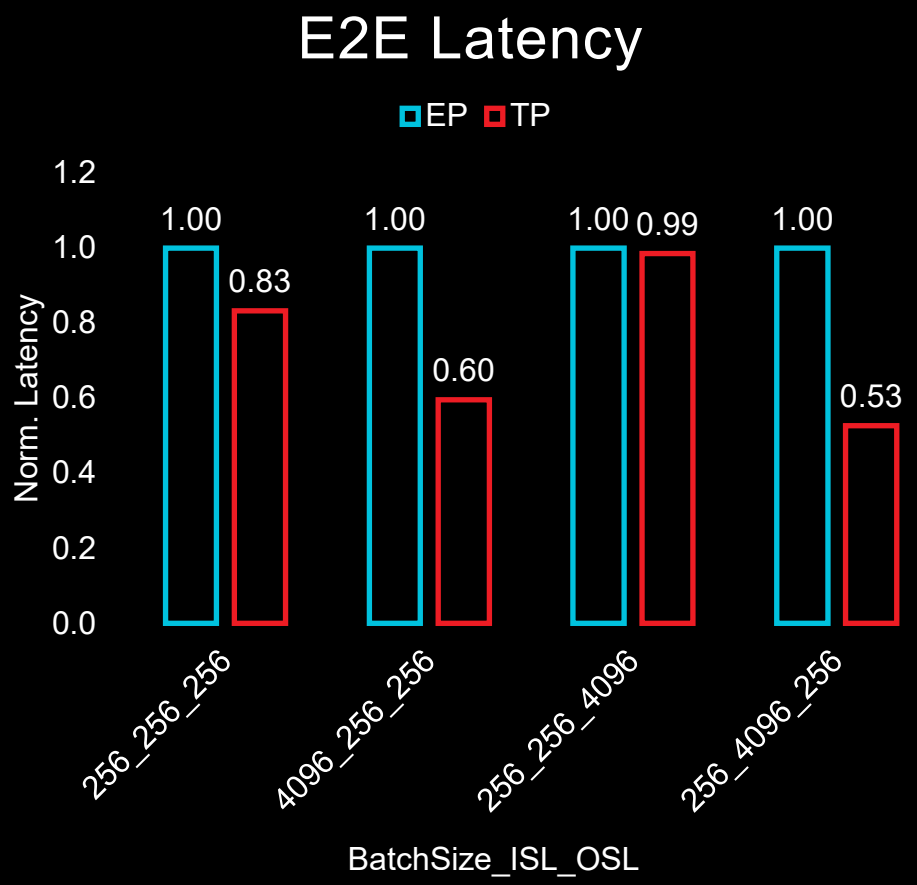
Tensor Parallelism (TP) vs. Expert Parallelism (EP) [Mixtral 8x7B]





Tensor Parallelism (TP) vs. Expert Parallelism (EP) [Mixtral 8x7B]

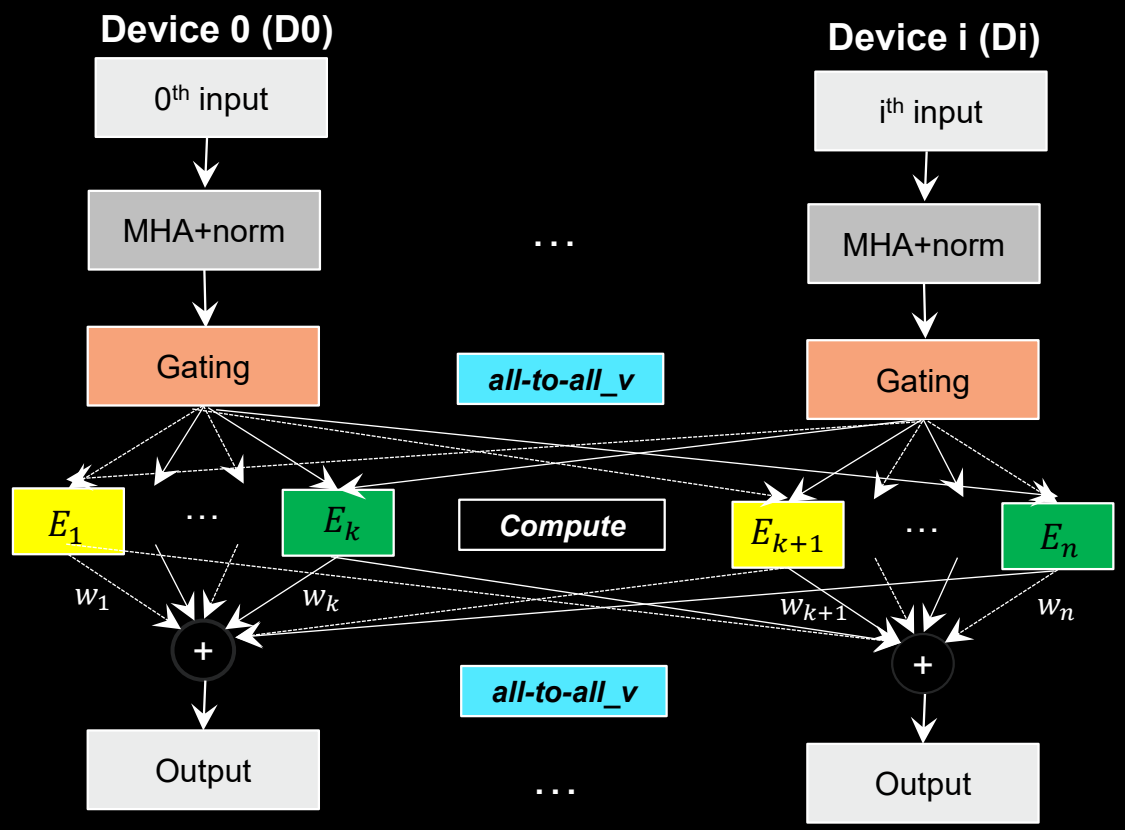




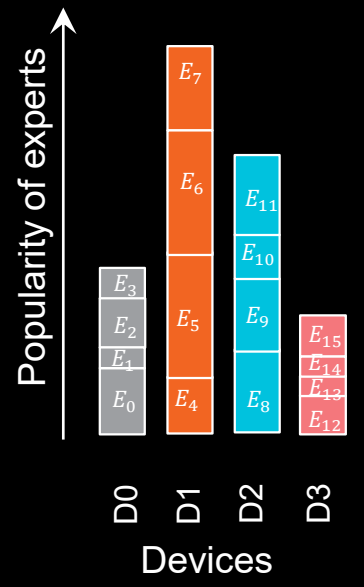
TP has lower latencies in all the cases

Contents

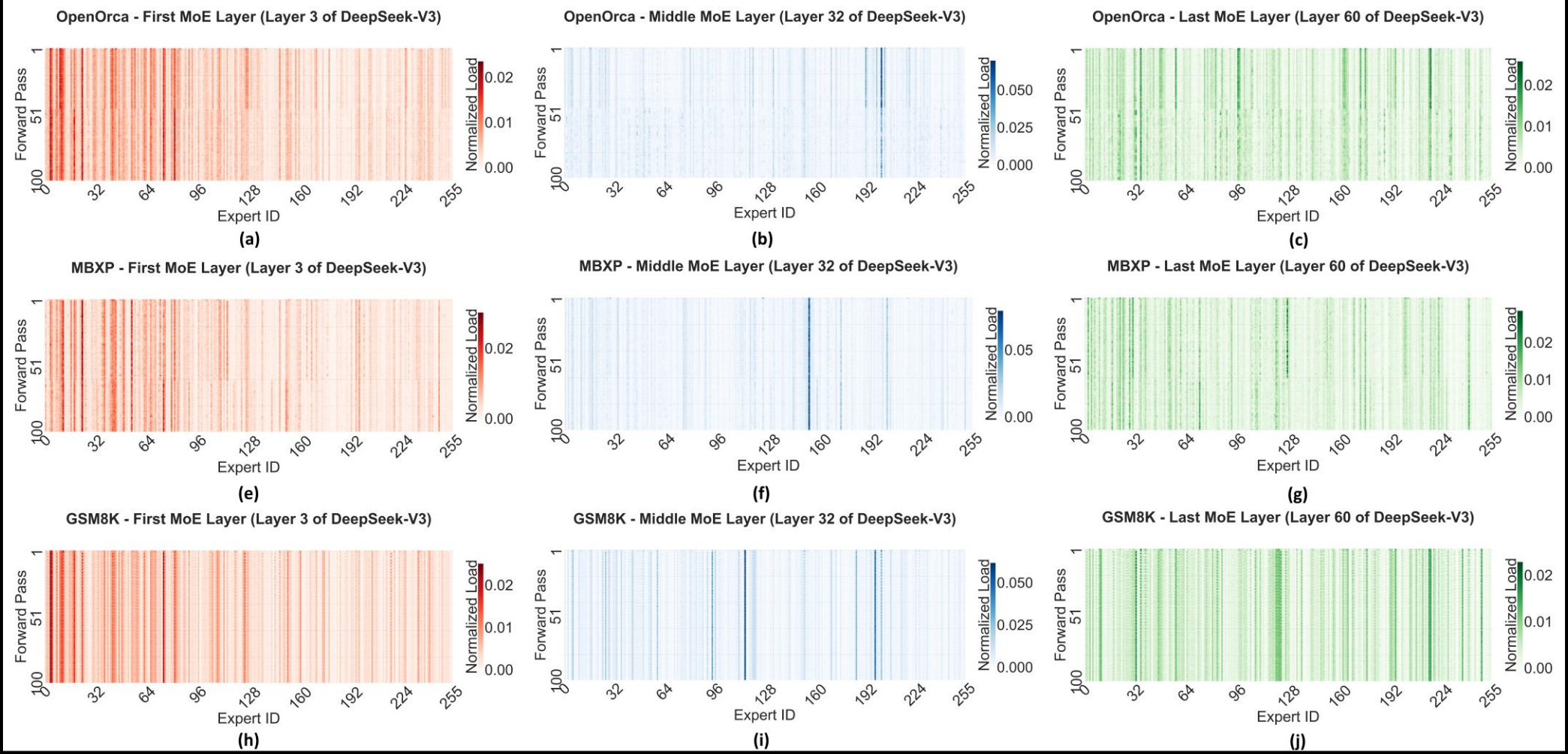
MoE Variable-length Computations & Communications



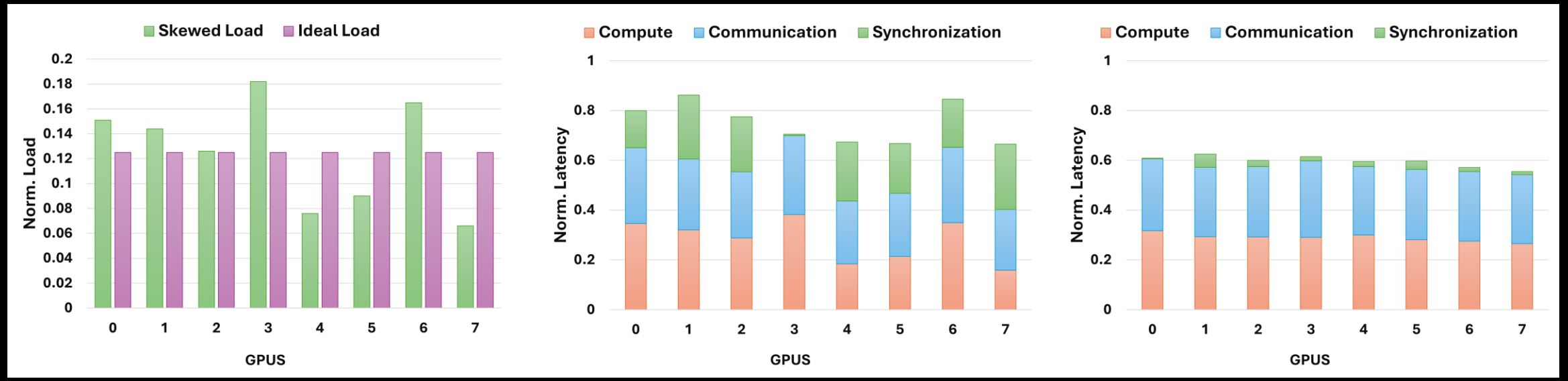
Main Challenges:
Variable-length Computation & Communication Kernels



Deepseek-V3 Token-to-Expert Distribution

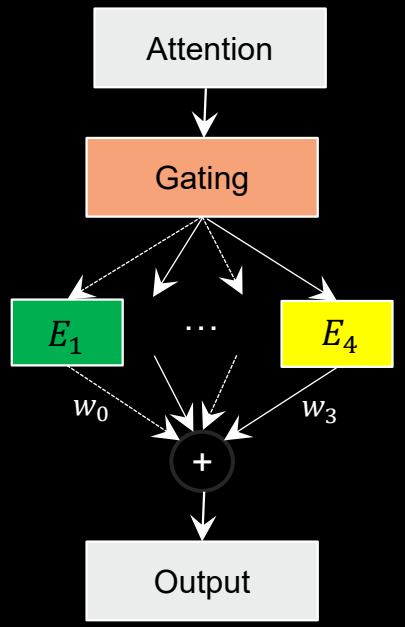


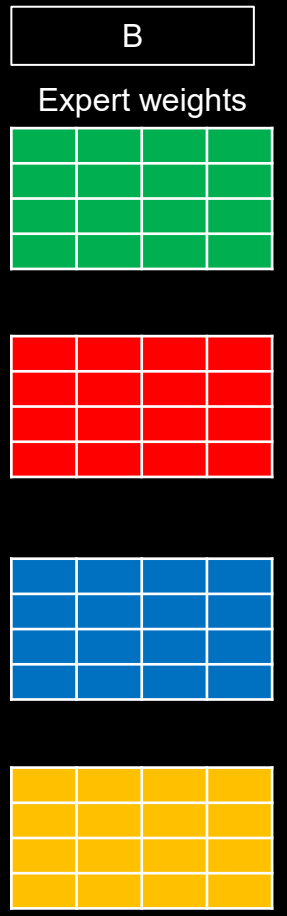
Impact on Performance



33% Performance Degradation Due To Workload Imbalance Across GPUs

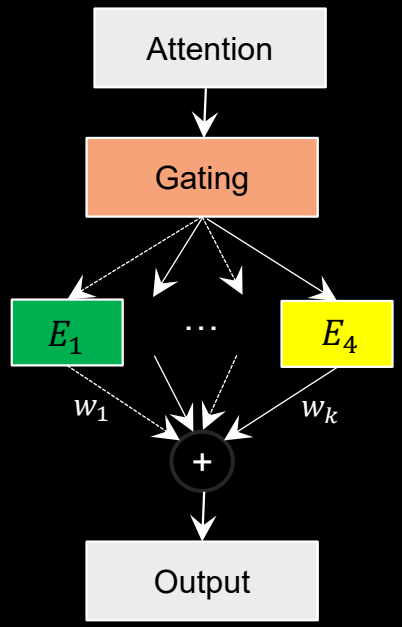
MoE Compute Kernels

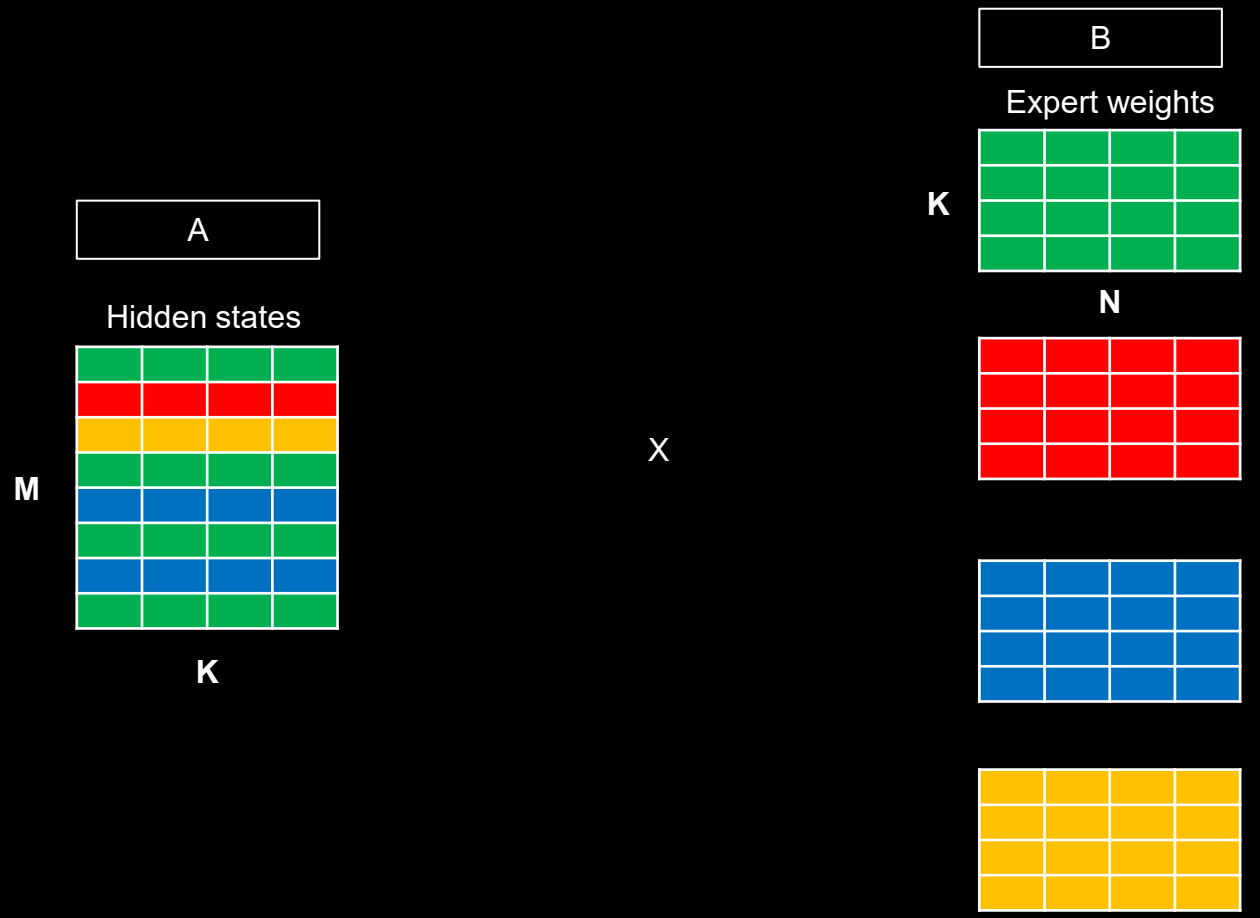




MoE Compute Kernels

Llama 3.1 405 B: (M: batch size, K: 16K, N: 53K)
DeepSeek-v3: (M:batch-size, K: 7K, N: 4K, experts:256, Topk=8)

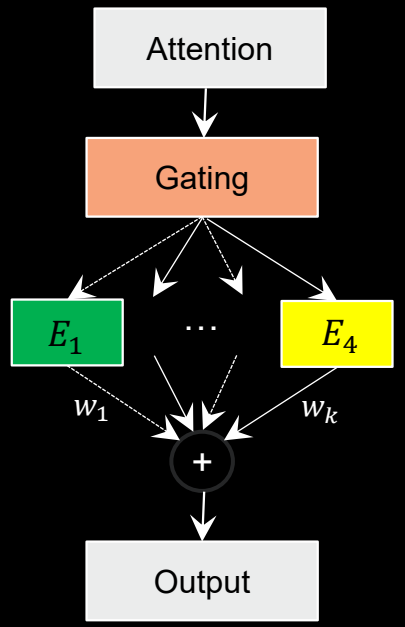




- Poor arithmetic intensity (n-dim is much smaller in MoEs) → memory-bound kernels

MoE Compute Kernels

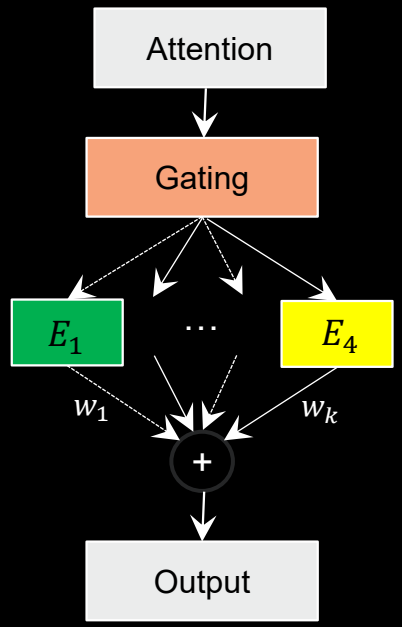
Llama 3.1 405 B: (M: batch size, K: 16K, N: 53K)
DeepSeek-v3: (M:batch-size, K: 7K, N: 4K, experts:256, Topk=8)



- Poor arithmetic intensity (n-dim is much smaller in MoEs) → memory-bound kernels
- Sparse with uncoalesced memory accesses → Needs sorting & re-sorting

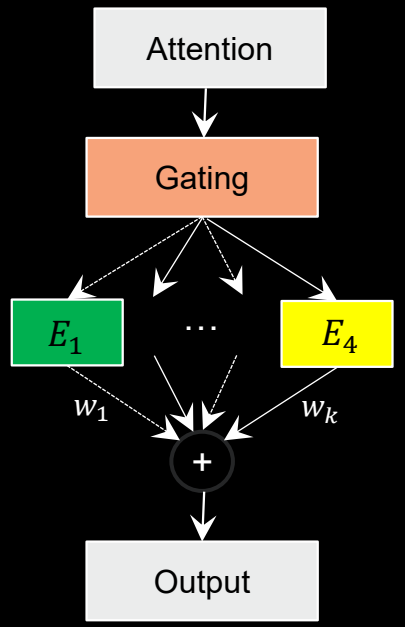
MoE Compute Kernels

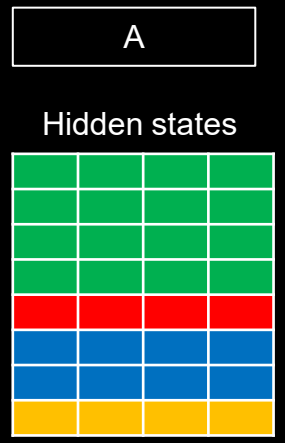
Mixtral 8x7B: (M: batch size, K:4096, N:28672, local_experts:8, Topk=2)
DeepSeek-v3: (M:batch-size, K:7168, N:4096, local_experts:16, Topk=8)

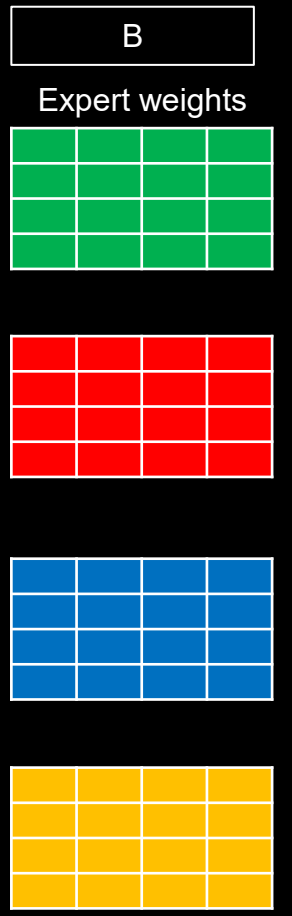


- Poor arithmetic intensity (n-dim is much smaller in MoEs) → memory-bound kernels
- Sparse with uncoalesced memory accesses → Needs sorting & re-sorting
- Uneven distribution of compute to experts → poor resource utilization

Sorting

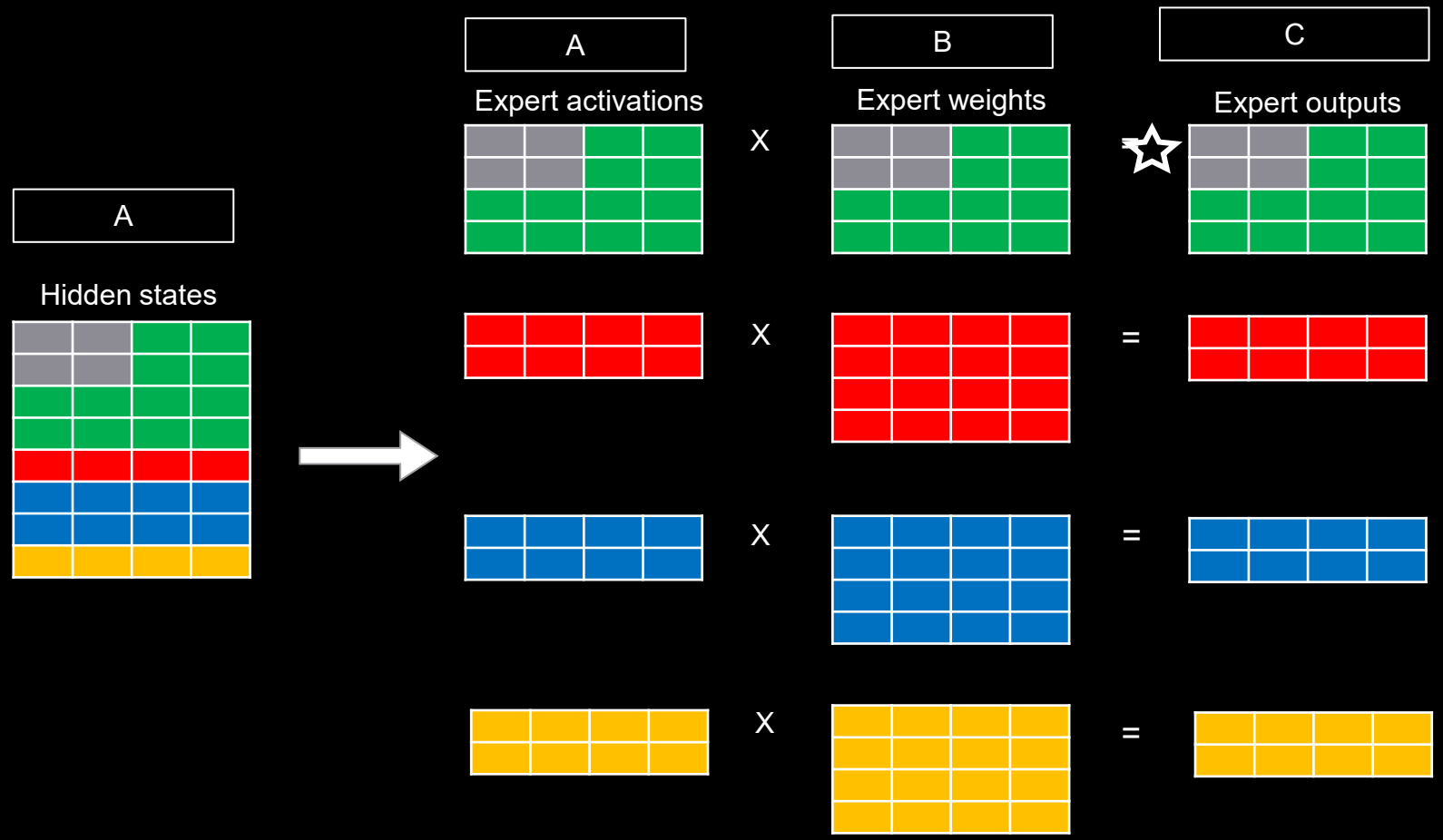






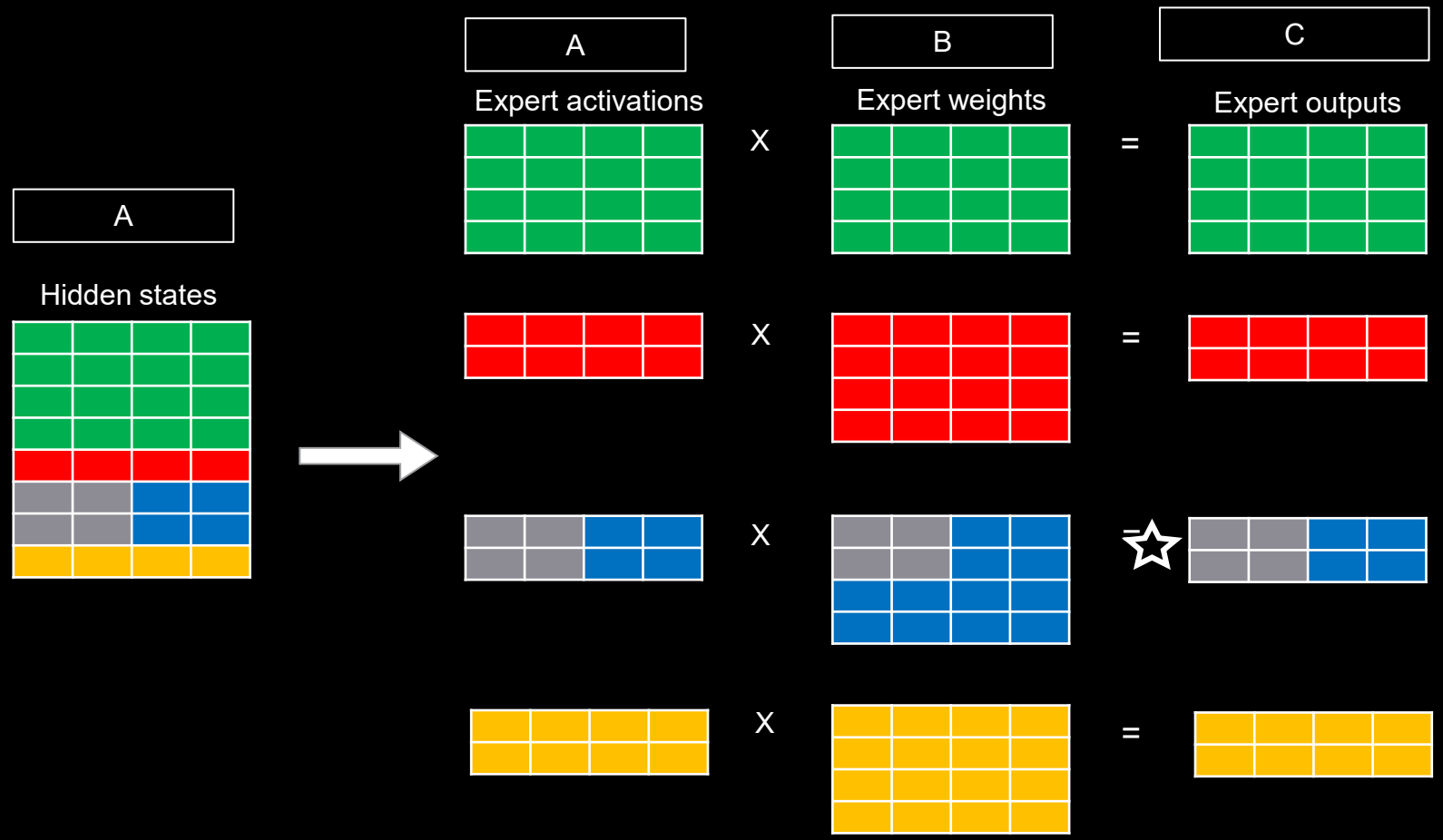
- First step: Sorting

Padding & Preparing Blocks for GEMM Computation



- First step: Sorting
- Second step: Padding & preparing blocks with meta data

Every Block Has Its Own Expert Index



- First step: Sorting
- Second step: Padding & preparing blocks with meta data
- Third step: Compute

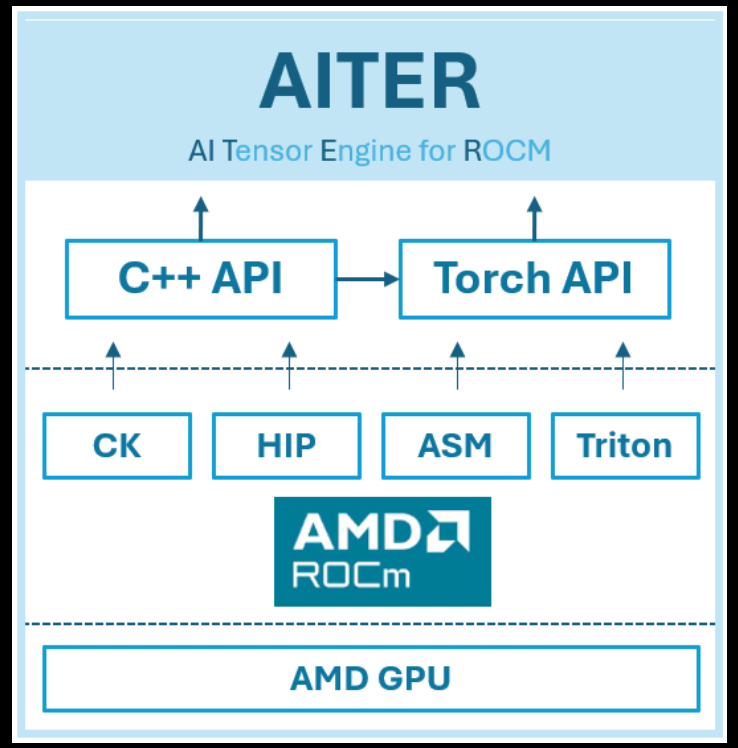
Source: https://docs.vllm.ai/en/latest/design/moe_kernel_features.html#modular-kernel-families

Contents

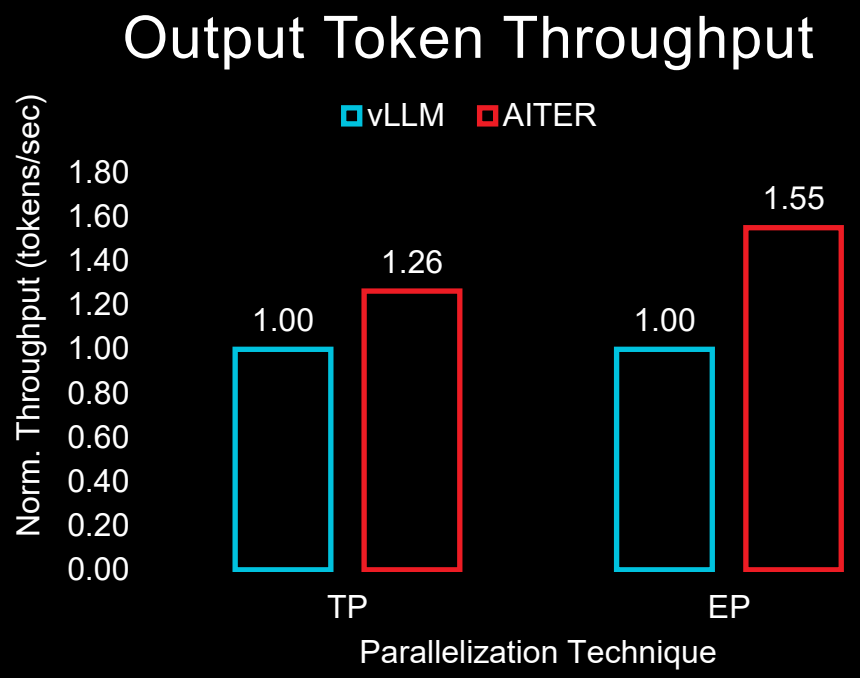
Hands-on Tutorial [SGLang, vLLM]

SGLang: https://docs.sglang.ai/
vLLM: https://github.com/vllm-project/vllm

AMD AITER Kernels



https://github.com/ROCm/aiter



Contents

Summary

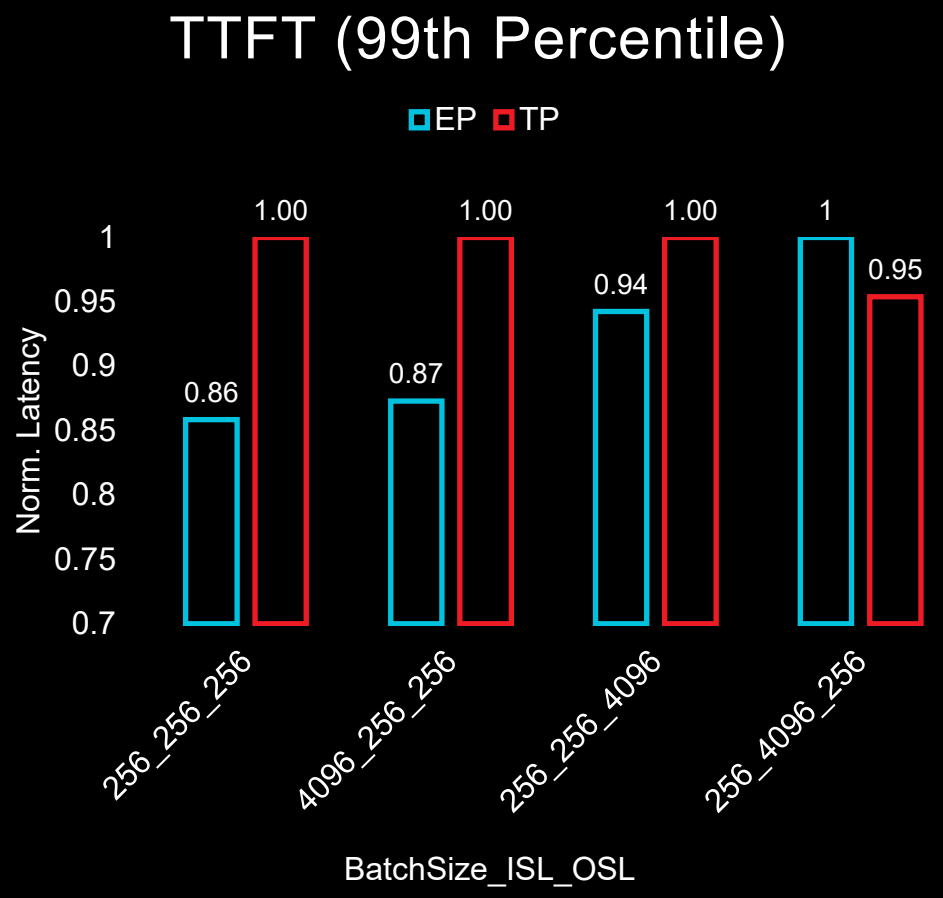
- Presented the trends in attention layer implementations and MoE layer implementations
- Introduced LLM performance metrics
- Discussed the effect of parallelization techniques on performance metrics of DeepSeek-v3 and Mixtral 8x7B model
- Discussed the computational and communication challenges while running MoE inference
- Discussed the impact of AMD AITER kernel library end-to-end performance of LLMs

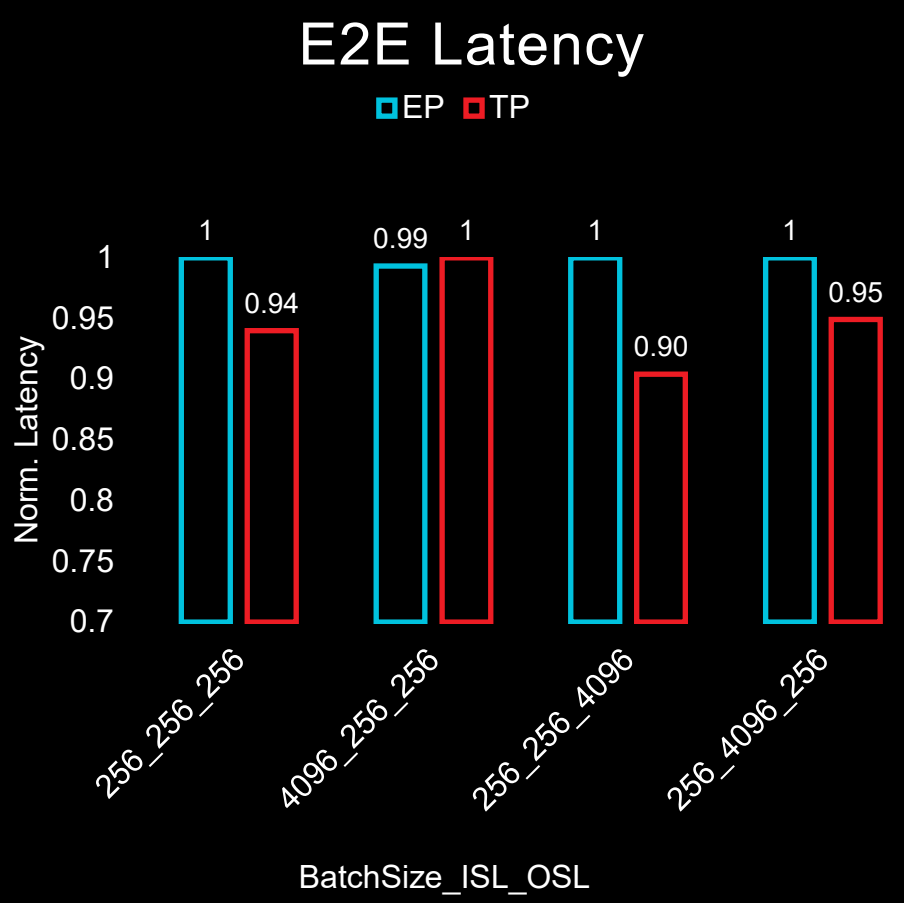
Disclaimer and Attributions

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18


AMD

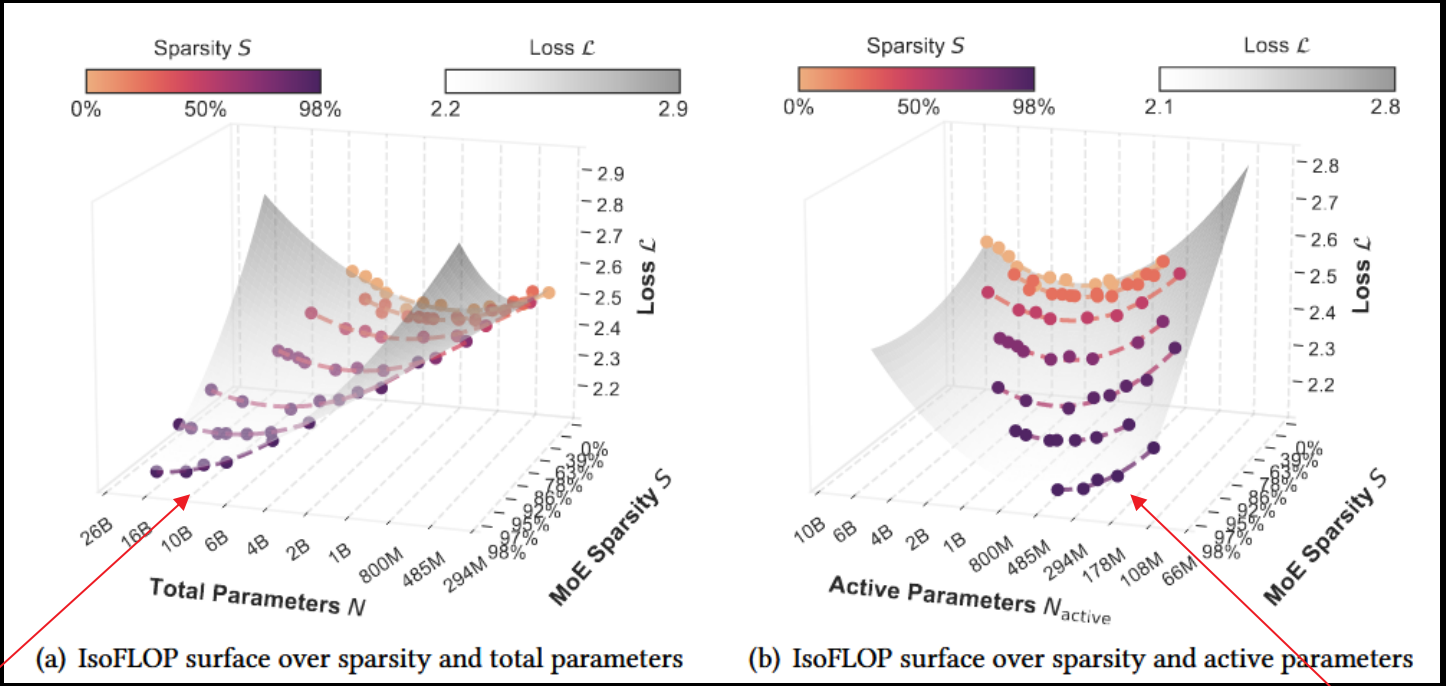
Tensor Parallelism (TP) vs. Expert Parallelism (EP) [Deepseek-V3]





EP is better for lower TTFTs while TP is better for E2E latencies

MoE Scaling Laws



(a) IsoFLOP surface over sparsity and total parameters

(b) IsoFLOP surface over sparsity and active parameters

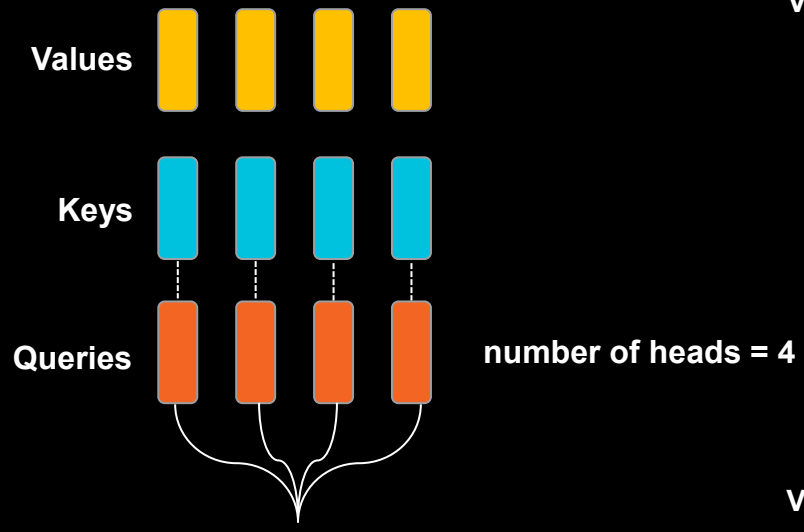
Best pre-training losses: larger models with increased sparsity

Best pre-training losses: fewer (not fewest!) active parameters and increased sparsity

Samira Abnar, et. al., Parameters vs. FLOPs: Scaling Laws for Optimal Sparsity for Mixture-of-Experts Language Models, https://arxiv.org/abs/2501.12370, 2025.

Attention Parallelization Techniques

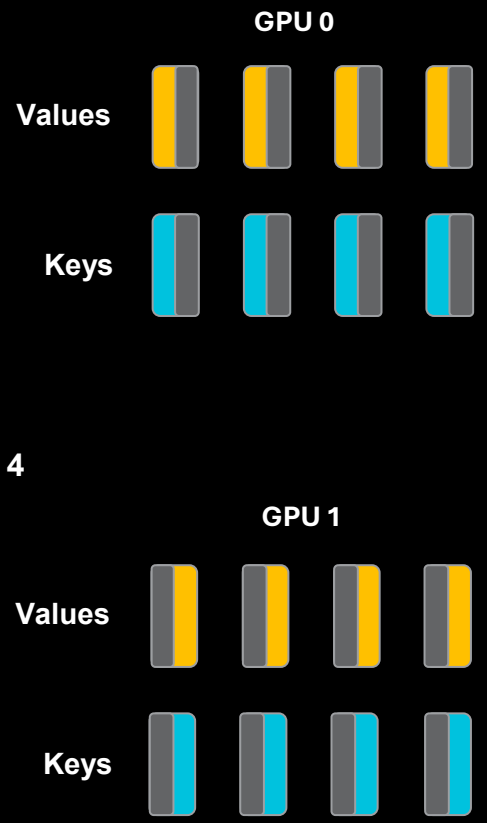
Multi-Head Attention (MHA)



number of heads = 4

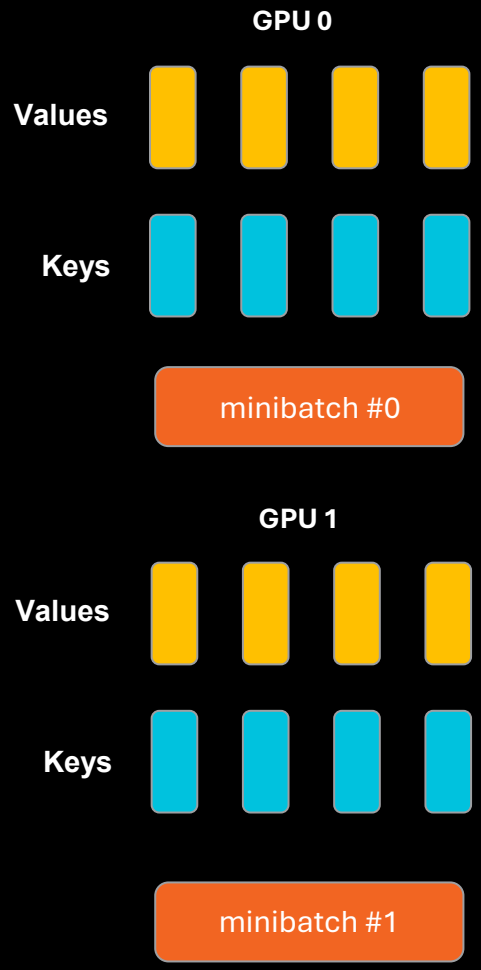
Prompt: The quick brown fox jumps over

Tensor Parallelism

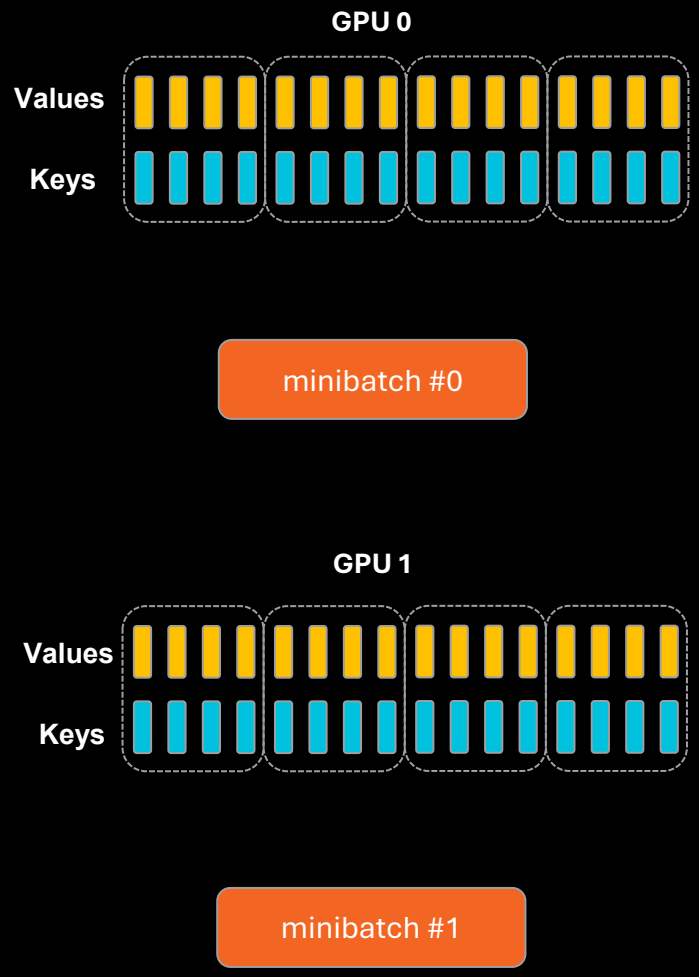


Data Parallel Attention & Multi-Head Attention

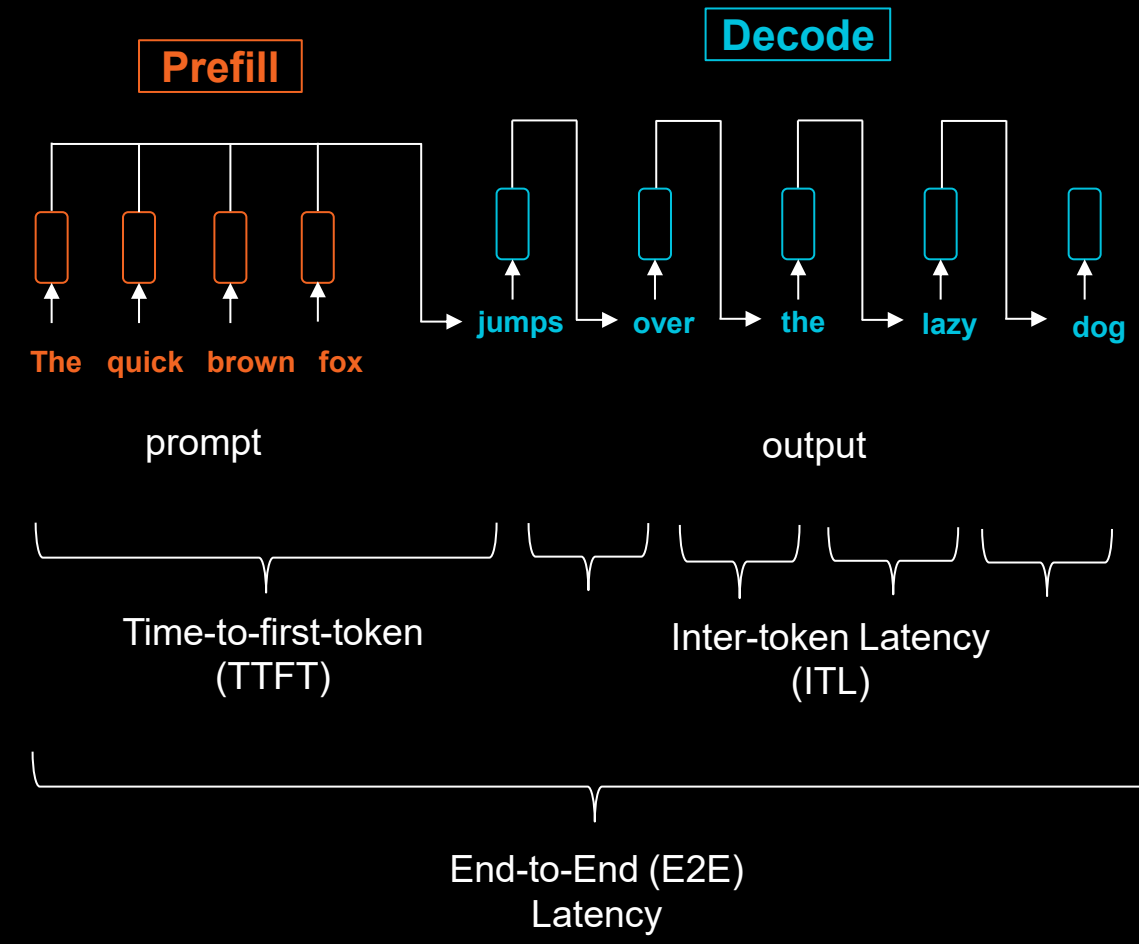
Data Parallel Attention



Data Parallel Multi-Head Attention



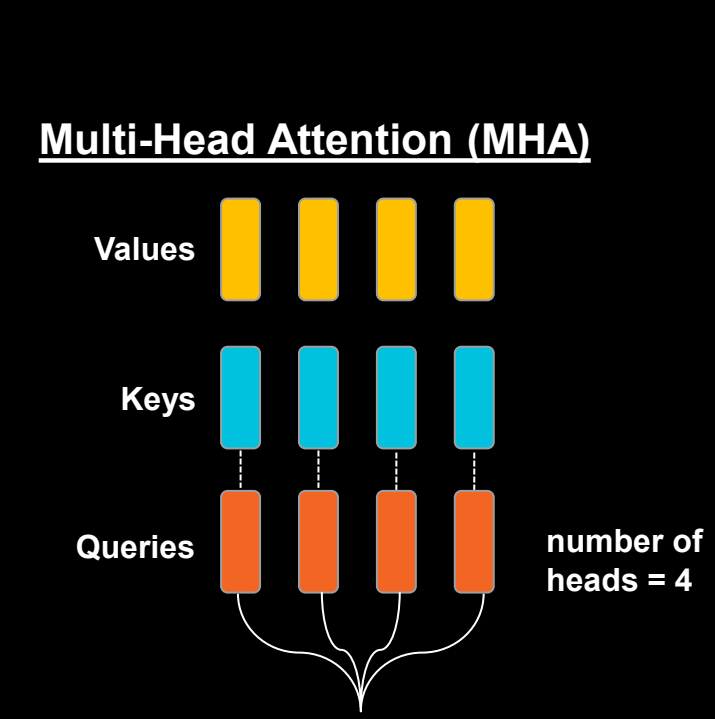
LLM Performance Metrics (Latency)

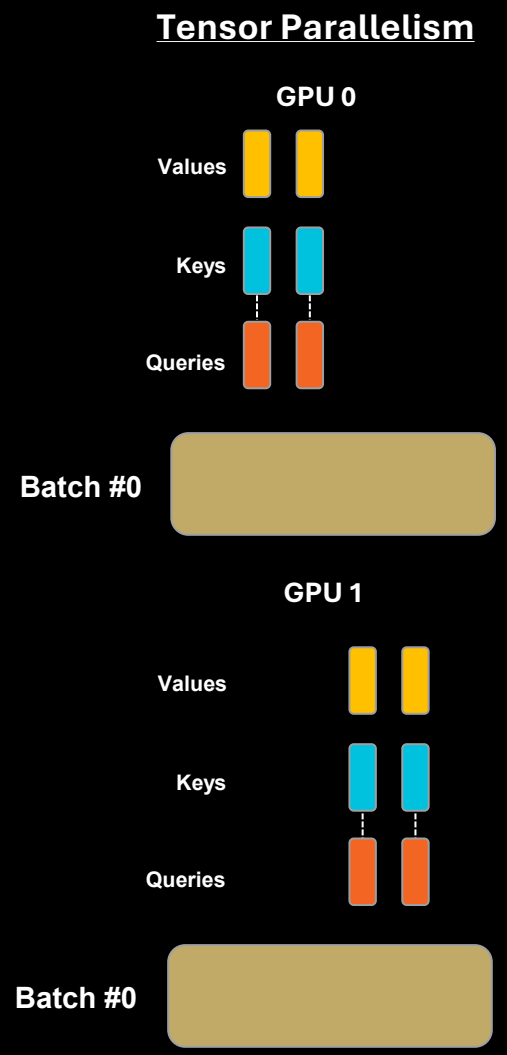


TTFT for each prompt
ITL for each output

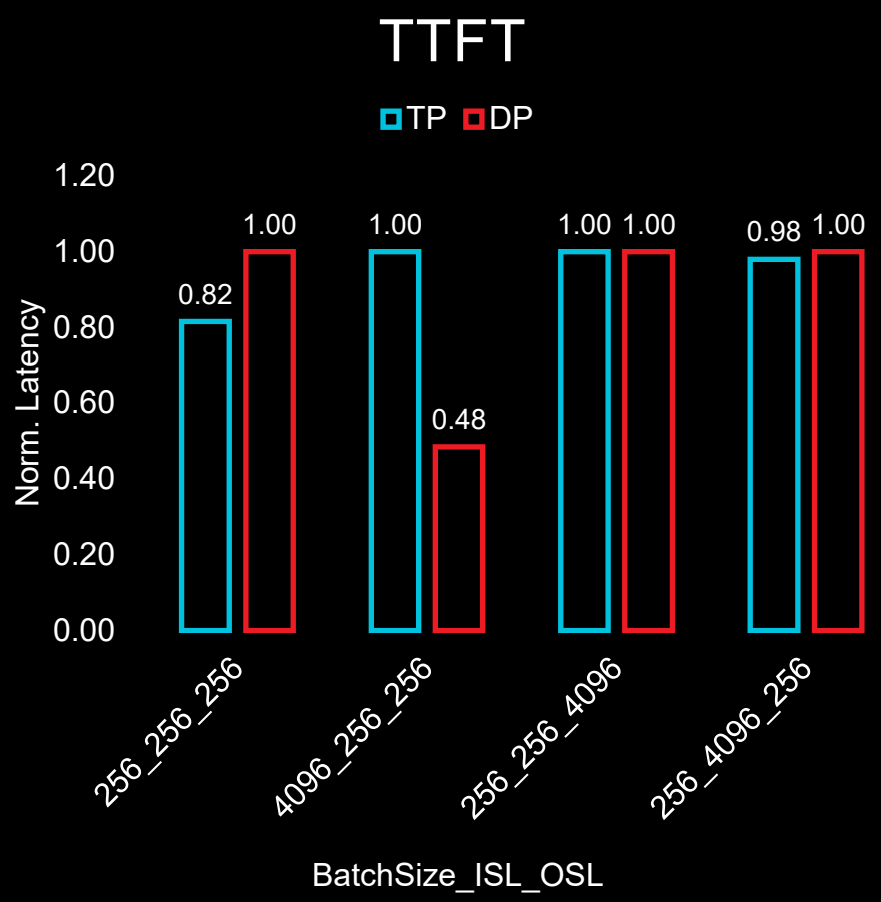
25/50/99 Percentiles for
- TTFT for a batch of prompts
- ITL for a batch of outputs

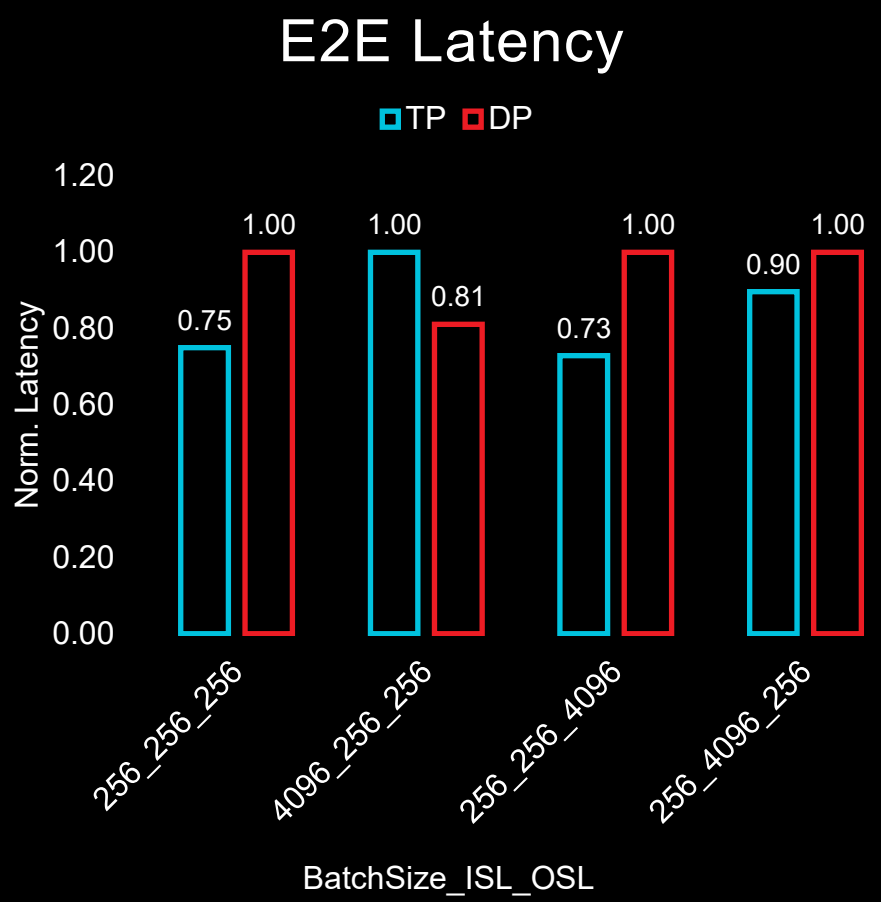
Multi-Head Attention Parallelization: Tensor Parallelism





TP vs. DP Attention [Mixtral 8x7B]





DP outperforms TP for conversation questions with large batch sizes

TP vs. DP Attention [Mixtral 8x7B]

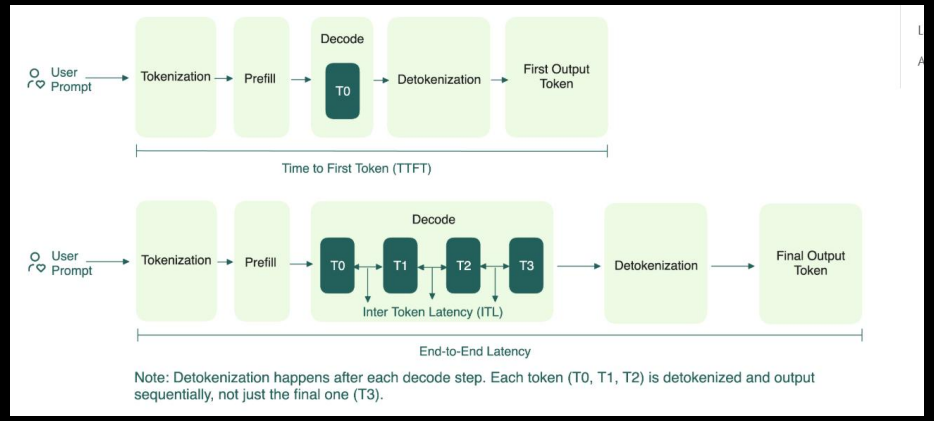
4096_256_256 case: TP vs DP attention

TP attention

```
============ Serving Benchmark Result ============
Successful requests:                     4096
Benchmark duration (s):                  109.82
Total input tokens:                      1043441
Total generated tokens:                  866638
Request throughput (req/s):              37.30
Output token throughput (tok/s):         7891.18
Peak output token throughput (tok/s):    14858.00
Peak concurrent requests:                4096.00
Total Token throughput (tok/s):          17392.25
---------------Time to First Token----------------
Mean TTFT (ms):                          45807.40
Median TTFT (ms):                        43098.17
P99 TTFT (ms):                           99536.77
-----Time per Output Token (excl. 1st token)------
Mean TPOT (ms):                          134.23
Median TPOT (ms):                        118.58
P99 TPOT (ms):                           374.08
---------------Inter-token Latency----------------
Mean ITL (ms):                           109.91
Median ITL (ms):                         71.72
P99 ITL (ms):                            375.25
==================================================
```

DP attention

```
============ Serving Benchmark Result ============
Successful requests:                     4096
Benchmark duration (s):                  89.22
Total input tokens:                      1043441
Total generated tokens:                  869790
Request throughput (req/s):              45.91
Output token throughput (tok/s):         9748.52
Peak output token throughput (tok/s):    24326.00
Peak concurrent requests:                4096.00
Total Token throughput (tok/s):          21443.31
---------------Time to First Token----------------
Mean TTFT (ms):                          26368.95
Median TTFT (ms):                        24975.51
P99 TTFT (ms):                           48227.88
-----Time per Output Token (excl. 1st token)------
Mean TPOT (ms):                          484.27
Median TPOT (ms):                        260.01
P99 TPOT (ms):                           2877.15
---------------Inter-token Latency----------------
Mean ITL (ms):                           246.82
Median ITL (ms):                         158.91
P99 ITL (ms):                            2849.36
==================================================
```



$$\text{Average ITL} = \text{TPOT} = \frac{\text{E2EL} - \text{TTFT}}{\text{Total Output Tokens} - 1}$$

Across multiple requests, however, the difference comes down to how you average:

$$\text{Average ITL} = \frac{\text{Sum of all ITLs across Requests}}{\text{Total Output Tokens across Requests}}$$

In this case, the average ITL is different from the average TPOT since the latter is usually calculated as follows:

$$\text{Average TPOT} = \frac{\text{TPOT}_1 + \text{TPOT}_2 + \cdots + \text{TPOT}_N}{N}$$

Both kernel latencies & scheduling approach are in play here