


HPC Techniques For Efficient LLM Serving

SCA: HPC for Machine Learning

Juan Jose Olivera
November 2025



TurboTransformers: An Efficient GPU Serving System For Transformer Models

Jiarui Fang
Patent Recognition Center, Wechat AI, Tencent Inc
Beijing, China
jiarui.fang@tencent.com

Chengshao Zhao
Patent Recognition Center, Wechat AI, Tencent Inc
Beijing, China
forianshao@tencent.com

Yang Yu
Patent Recognition Center, Wechat AI, Tencent Inc
Beijing, China
josephy@tencent.com

Jie Zhou
Patent Recognition Center, Wechat AI, Tencent Inc
Beijing, China
withzhou@tencent.com

Abstract
The transformer is the most critical algorithm innovation of the Natural Language Processing (NLP) field in recent years. Unlike the Recurrent Neural Network (RNN) models, transformers are able to process on dimensions of sequence lengths in parallel, therefore leads to better accuracy on long sequences. However, efficient deployment of them for online services in data centers equipped with GPUs are not easy. First, more computation introduced by transformer structures makes it more challenging to meet the latency and throughput constraints of serving. Second, NLP tasks take in instances of variable length. The variability of input dimensions brings a

CCS Concepts: Computing methodologies; Concurrent computing methodologies; Natural language generation; Parallel algorithms.

Keywords: Transformers; Deep Learning Runtime; Serving System; GPU

1 Introduction

The recent success of Natural Language Processing (NLP) techniques is enabled largely by the transformer-based Deep Neural Networks (DNNs), such as Seq2Seq [36], BERT [7], GPT2 [25], and XLNet [11], XLBERT [44]. With the success

DeepSpeed Inference: Enabling Efficient Inference of Transformer Models at Unprecedented Scale

Rory Yashini Arinibadi, Sangam Rajbhandari, Minjie Zhang, Anwar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Jeff Rasley, Shaden Smith, Ofirnat R. Raveh, Yuxiong He
Microsoft Corporation
{yashini.rosa,sangam.rajbhandari,minjie.zhang,anwar.ahmad,cheng.li,du.li,elton.zheng,jeff.rasley,shaden.smith,ofirnat.raveh,yuxiong.he}@microsoft.com

Abstract—The past several years have witnessed the success of transformer-based models, and their scale and application scenarios continue to grow aggressively. The current landscape of transformer models is increasingly diverse: the model size varies drastically with the largest being of hundred-billion parameters; the model characteristics differ due to the specific introduced by the Mixture-of-Experts; the target application scenarios can be latency-critical or throughput-oriented; the deployment hardware could be single- or multi-GPU systems with different types of memory and storage, etc. With such increasing diversity and the fast-growing pace of transformer models, designing a highly performant and efficient inference system is extremely challenging.

In this paper, we present DeepSpeed Inference, a comprehensive system solution for transformer model inference to address the above-mentioned challenges. DeepSpeed Inference consists of (1) a multi-GPU inference solution to minimize latency while maximizing the throughput of both dense and sparse transformer models where they fit in aggregate GPU memory; and (2) a heterogeneous inference solution that leverages CPU and NVMe memory in addition to the GPU memory and compute to enable

latency requirements, and thus the batch sizes used are generally small. For small batch sizes, inference latency of a model is lower bounded by the time it takes to load all the model parameters from memory to registers. Meeting the latency requirements of a transformer model inference, therefore, is equivalent to achieving adequate overall memory bandwidths. Maximizing effective memory bandwidth at small batch sizes requires reading memory at near peak memory bandwidth for self-connected (see: heavy) layers which contain the majority of the model weights, while also minimizing kernel launch and data movement overhead of other operators like layernorm and softmax. The GEMM implementations and other kernels designed for training primarily focus on maximizing compute utilization at very large batch sizes and are sub-optimal for inference-critical inference.

In addition, for large models, even the peak memory bandwidth of a single device may not be sufficient to meet inference

FLASHDECODING++: FASTER LARGE LANGUAGE MODEL INFERENCE ON GPUS

Ke Hong¹
Tsinghua University
& Intelligence-AI

Qidi Mao¹
Tsinghua University
& Intelligence-AI

Kangqi Chen¹
Intelligence-AI

Guanhua Dai^{1*}
Shanghai Jiao Tong University
& Intelligence-AI

Xiaohong Li¹
Peking University

Yuhao Dong¹
Tsinghua University

Jiaming Xu¹
Shanghai Jiao Tong University
& Intelligence-AI

Jun Lin¹
Shanghai Jiao Tong University
& Intelligence-AI

Yu Wang¹
Tsinghua University

*daiguohao@sjtu.edu.cn, daiguohao@infimi-ai.com, yu-wang@tsinghua.edu.cn

ABSTRACT

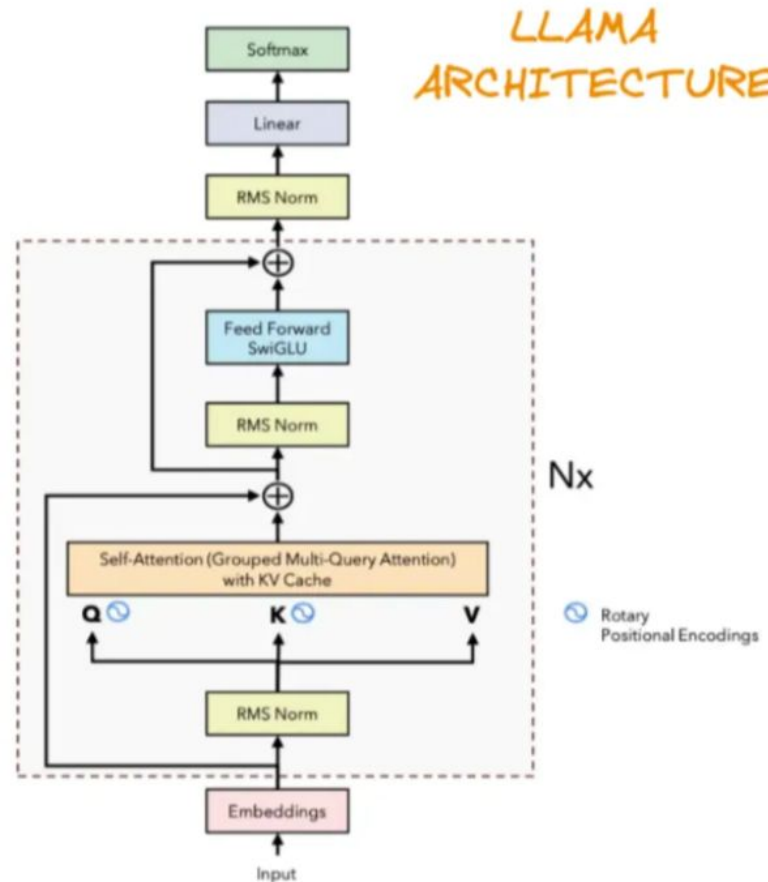
As the Large Language Model (LLM) becomes increasingly important in various domains, the performance of LLM inference is crucial to massive LLM applications. However, the following challenges still remain involved in accelerating LLM inference: (1) Synchronized partial softmax update. The softmax operation requires a synchronized update operation among each partial softmax result, leading to ~30% overheads for the attention computation in LLMs. (2) Under-utilized computation of the GEMM. The shape of matrices performing GEMM in LLM inference is flat, leading to under-utilized computation units. (3) Softmax operation with multiple parallel units.

Agenda

1. **LLM Inference 101**
2. **Performance Metrics**
3. **Motivation and Papers**
4. **(Four) Optimization Techniques**
 - a. **Problem**
 - b. **Insights**
 - c. **Solution**
5. **Conclusions**
6. **Questions**

LLM INFERENCE 101

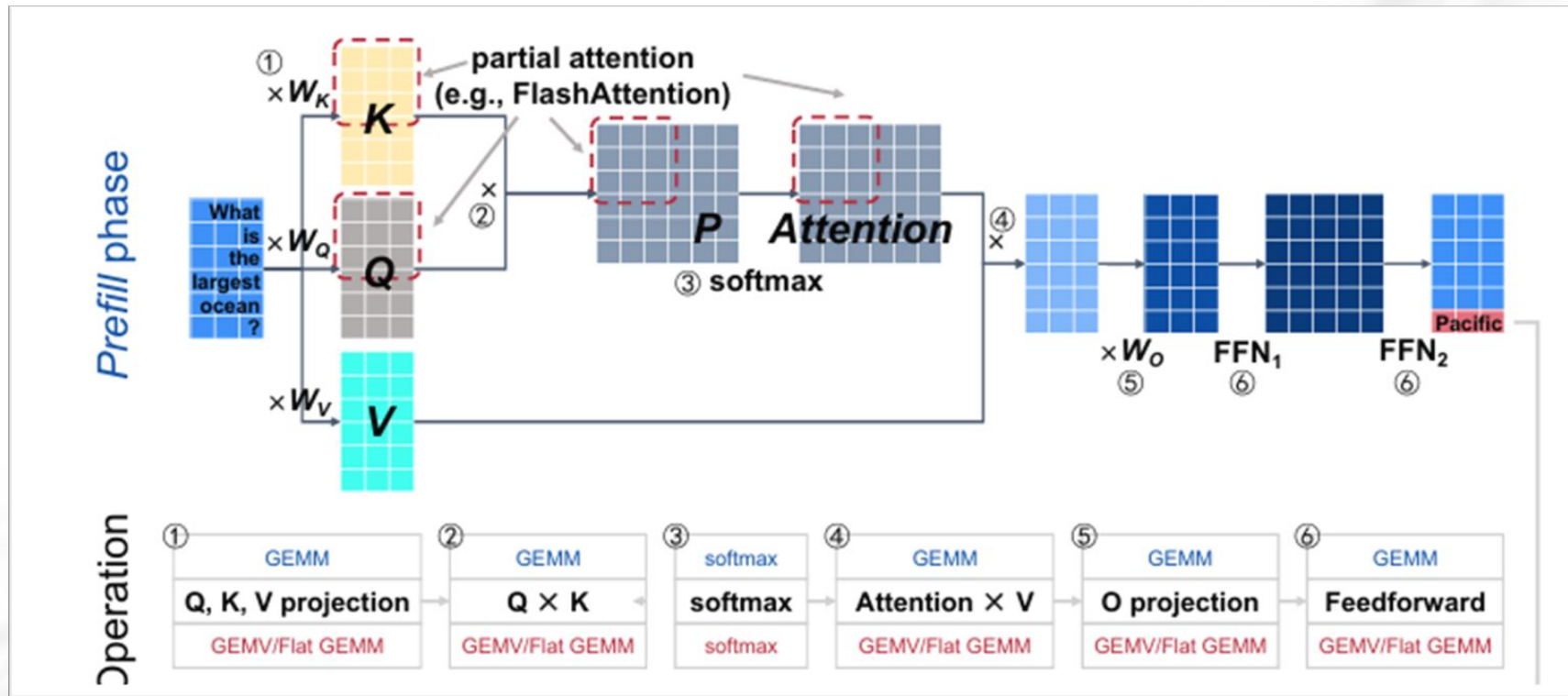
Modern GPT LLMs Architectures



LLM INFERENCE 101

Phase 1 of 2: Prefill

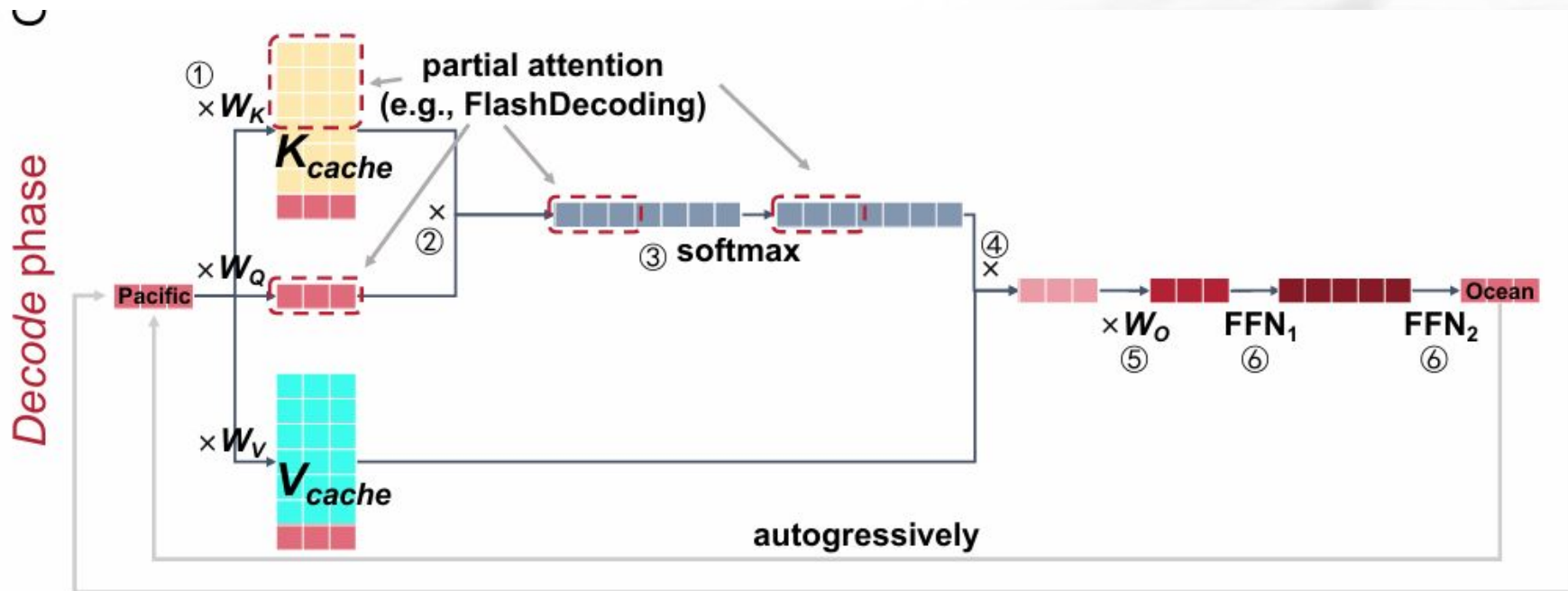
- 4 GEMMs + Softmax
- Big matrices



LLM INFERENCE 101

Phase 2 of 2: Decode

- 4 GEMMs + Softmax
- Slim matrices



Performance Metrics

Latency	End-to-end token generation time [seconds]
Throughput	Amount of tokens generated [tokens/s]
Compute Efficiency	FLOPs achieved against theoretical max [%]
Memory Efficiency	Footprint Size No. mem allocations
Throughput Scalability	Throughput Increase Efficiency [%]
Model Scalability	Model Increase Efficiency [%]

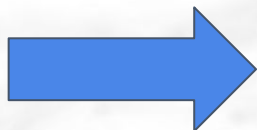
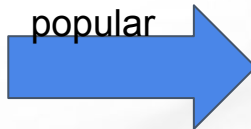
Motivation: Per què?

What is currently out there?
What is being researched?

Motivation Use Case



super
popular



Thus, worth
analyzing



Accelerating LLMs with llama.cpp on NVIDIA RTX Systems

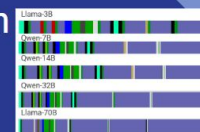


Oct 02, 2024

+17 Like Discuss (0)

LLM Model Size Impact on Application Performance

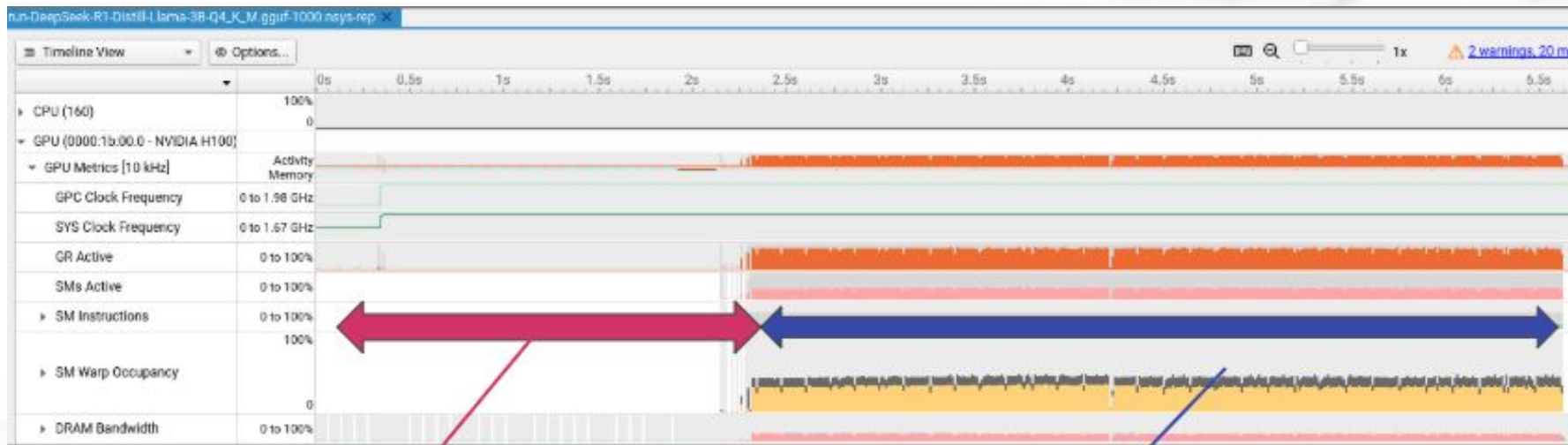
Juan Jose Olivera Loyola



PPTM Final Project - Professor Jesus Jose Labarta
TMIRI-HPC

Motivation Use Case

Overall execution of llama.cpp of 1000 tokens with Llama-3B



Initialization/Model Loading

Real Work: Token Prediction

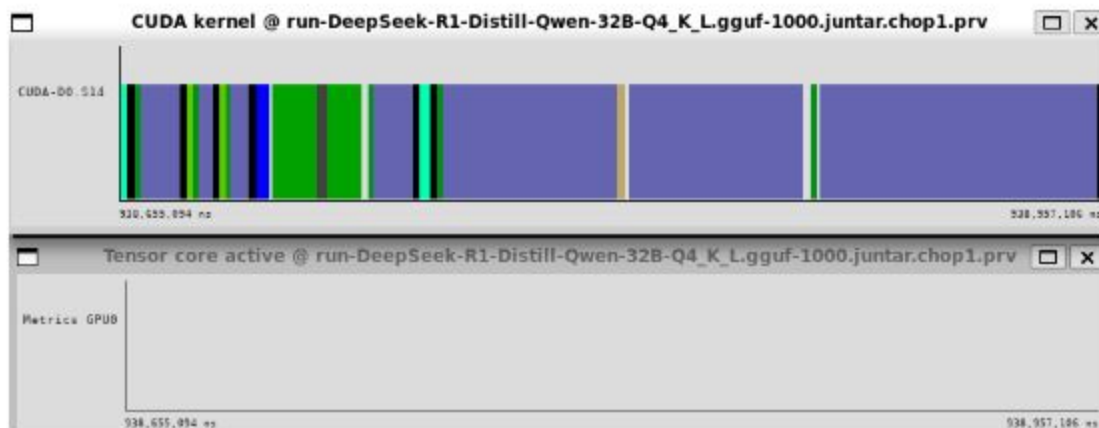
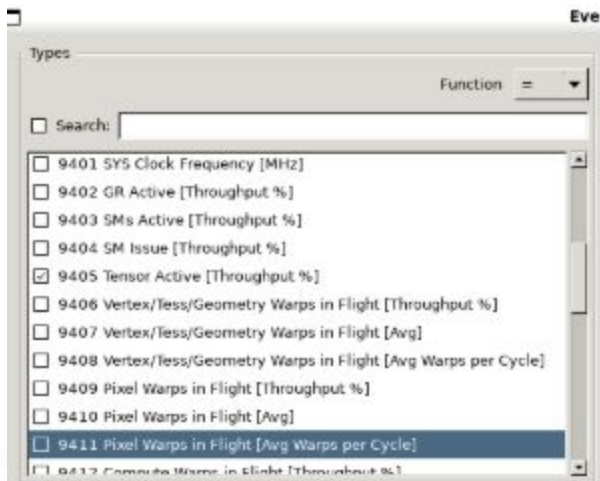
Motivation Use Case

% Stream Multiprocessors (SM) Active



Motivation Use Case

NO Tensor Cores Active per Iteration



0% Tensor Cores Utilization

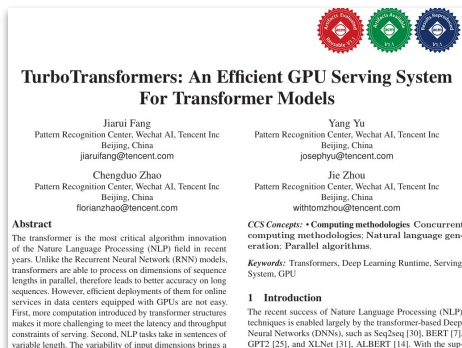
HPC CRIME!!!

Is it really?

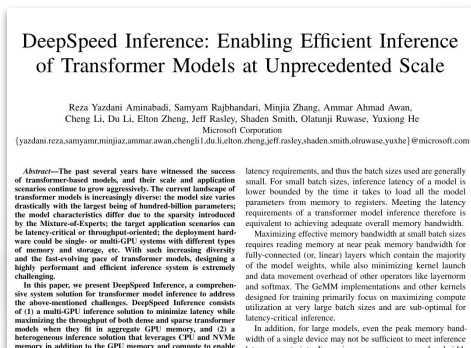
Source: "LLM Model Size Impact on Application Performance"

Motivation & Papers

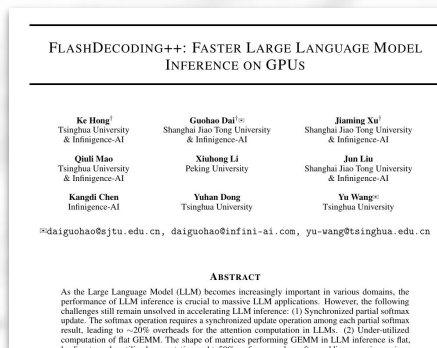
How implementations differ? What HPC techniques can be applied?



TurboTransformers
Feb 2021
Multiple GPUs
Serving/Scheduling



DeepSpeed
Jul 2022
Multiple GPUs
GPUs Parallelism



FlashDecoding++
Nov 2023
Single GPU
Computation Efficiency

T1: Flat GEMM Double Buffering

FlashDecoding++, DeepSpeed

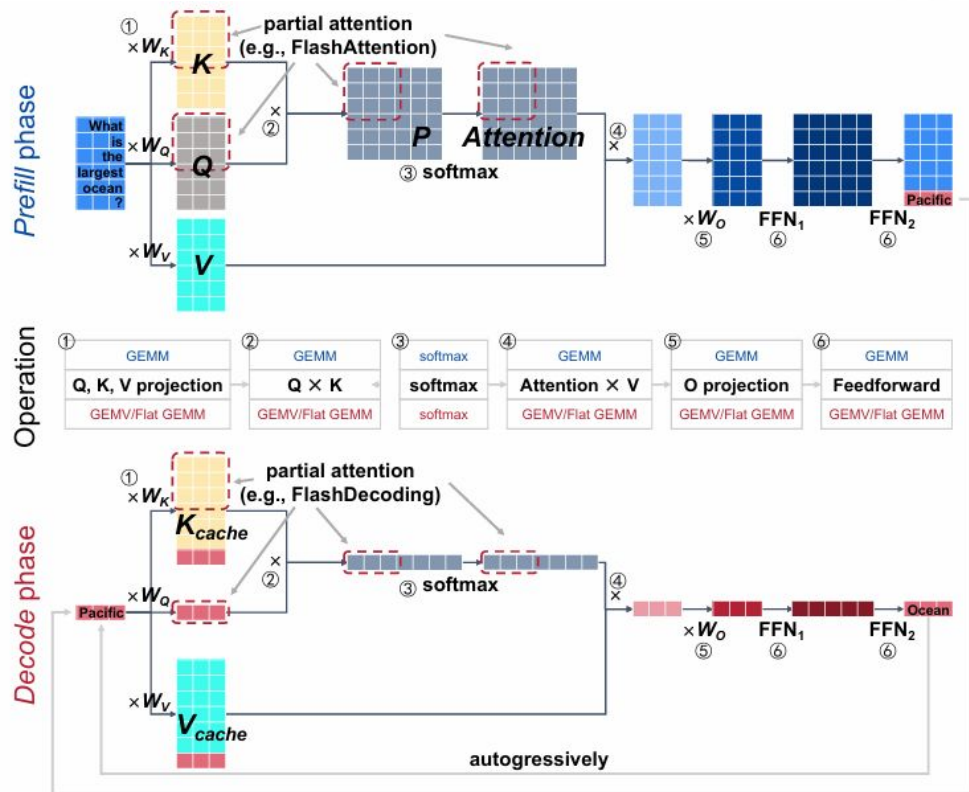
Problem	cuBLAS/cuTLASS TensorCore libraries badly optimized for skinny matrices (few tokens). Small M
Insights	For small N, GEMM is parallelism-bounded (# SMs) For big N, GEMM is memory-bounded.
Solution	Flat GEMM custom implementation with tiling and double buffering Execution of different GEMM implementations based on data nature.

T1: Flat GEMM Double Buffering

FlashDecoding++, DeepSpeed

PROBLEM

Flat matrices
in decode
phase



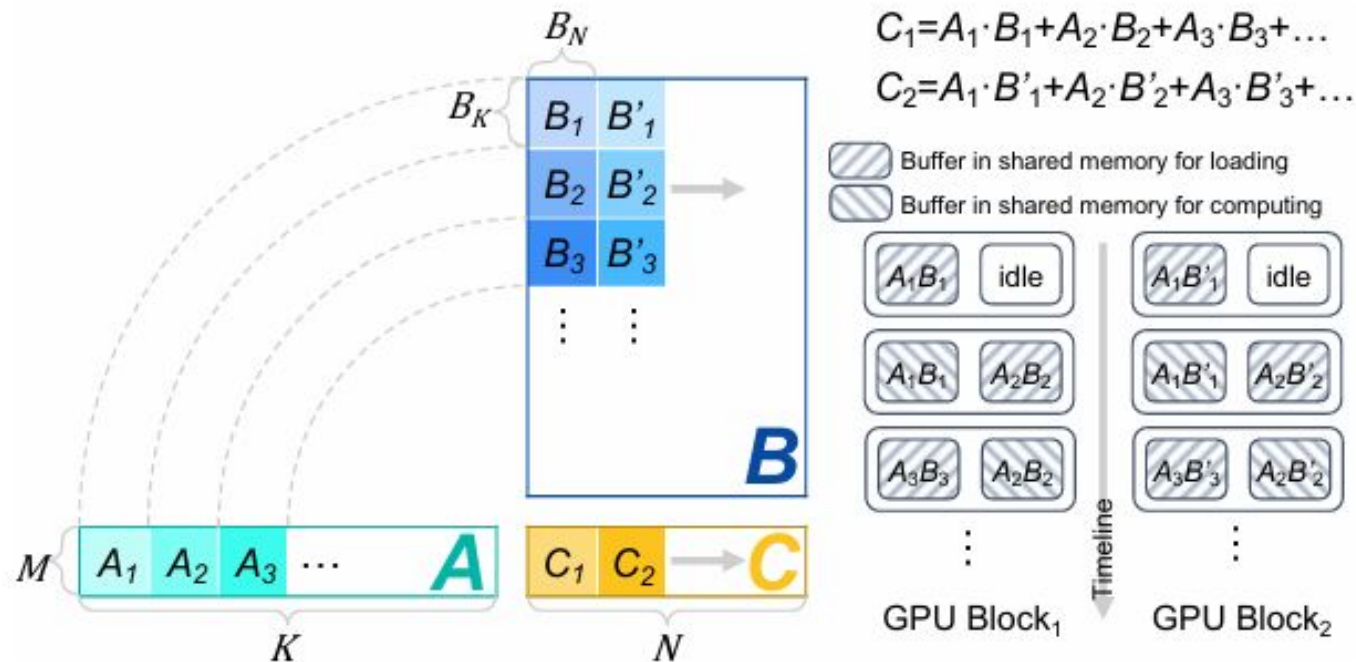
Source: FlashDecoding++: Faster large language model inference on GPU

T1: Flat GEMM Double Buffering

FlashDecoding++, DeepSpeed

S1: Flat GEMM

-Tiles
-Double Buffer



T1: Flat GEMM Double Buffering

FlashDecoding++, DeepSpeed

S2: Different GMMs

- cuBLAS
- FastGEMV
- FlatGEMM

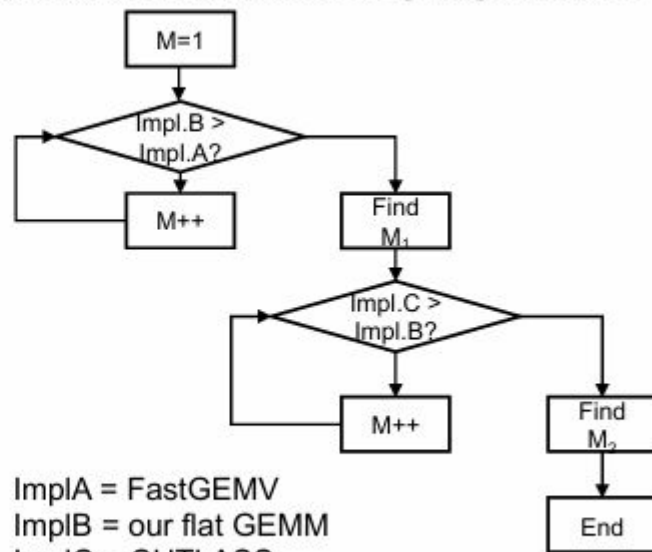
	Operation	M	N	K
Prefill phase	K, Q, V projection	SeqLen*B	HD*3	HD
	O projection	SeqLen*B	HD	HD
	FFN1	SeqLen*B	FD	HD
	FFN2	SeqLen*B	HD	FD
Decode phase	K, Q, V projection	B	HD*3	HD
	O projection	B	HD	HD
	FFN1	B	FD	HD
	FFN2	B	HD	FD

HD: Hidden dimension size
FD: Dimension size after the first FFN
B: Batch size
SeqLen: Input sequence length

Only 4 shapes!

(a) Different shapes of GEMMs in LLM

For a certain LLM, traverse four [N, K] selections



ImplA = FastGEMV
ImplB = our flat GEMM
ImplC = CUTLASS

(b) Decision flow

T1: Flat GEMM Double Buffering

FlashDecoding++, DeepSpeed

T2: Memory Allocation

TurboTransformers

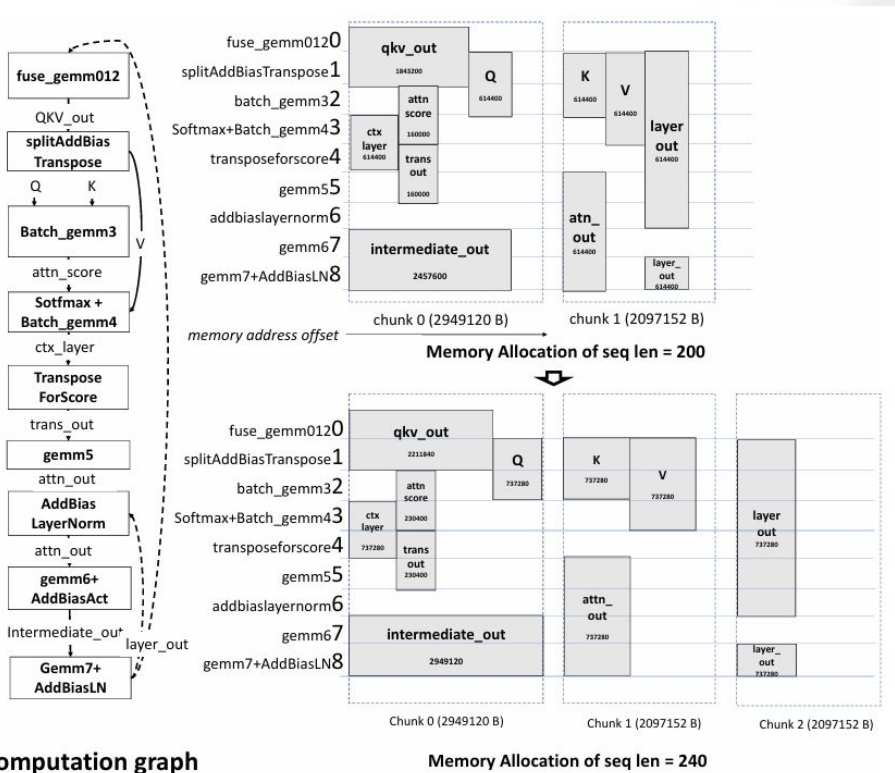
Problem	Few # Allocations → Allocating big memory pool → Big Memory Footprint
Insights	Can allocate and reuse Per-Chunk Computation Graph (CG) → Predict Chunk Life Time
Solution	Sequence-length aware allocator algorithm Finds gaps in memory pool to fit in chunks Computes offsets based on lifetimes from CG to reuse memory

T2: Memory Allocation

TurboTransformers

S: Seq-Aware Chunk Allocation

- cuBLAS
- FastGEMV
- FlatGEMM



computation graph

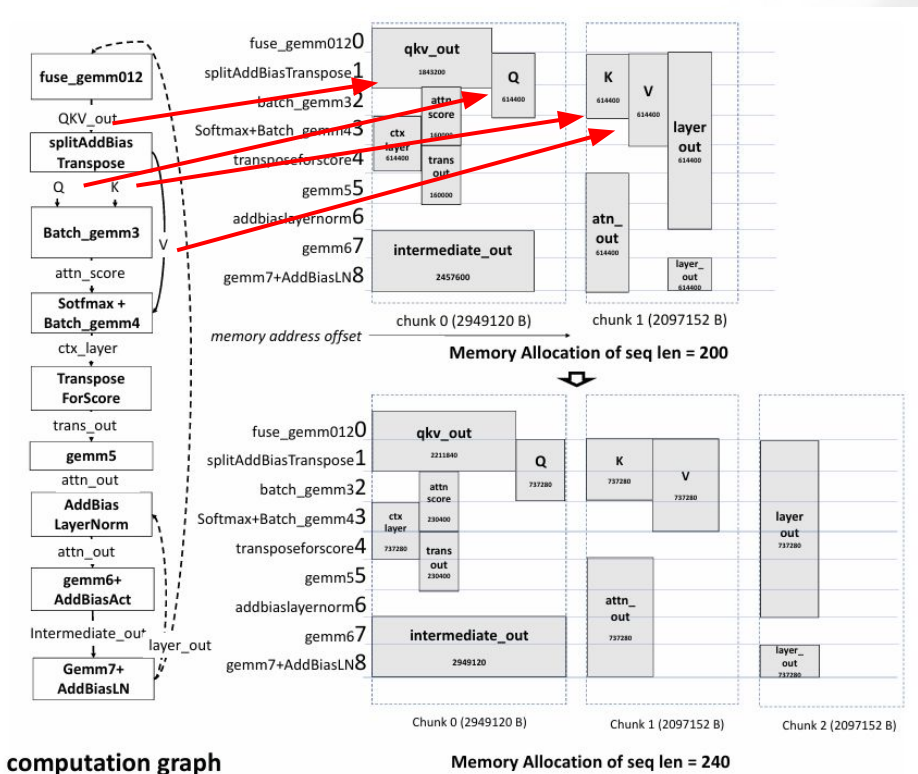
Memory Allocation of seq len = 240

T2: Memory Allocation

TurboTransformers

S: Seq-Aware Chunk Allocation

- cuBLAS
- FastGEMV
- FlatGEMM

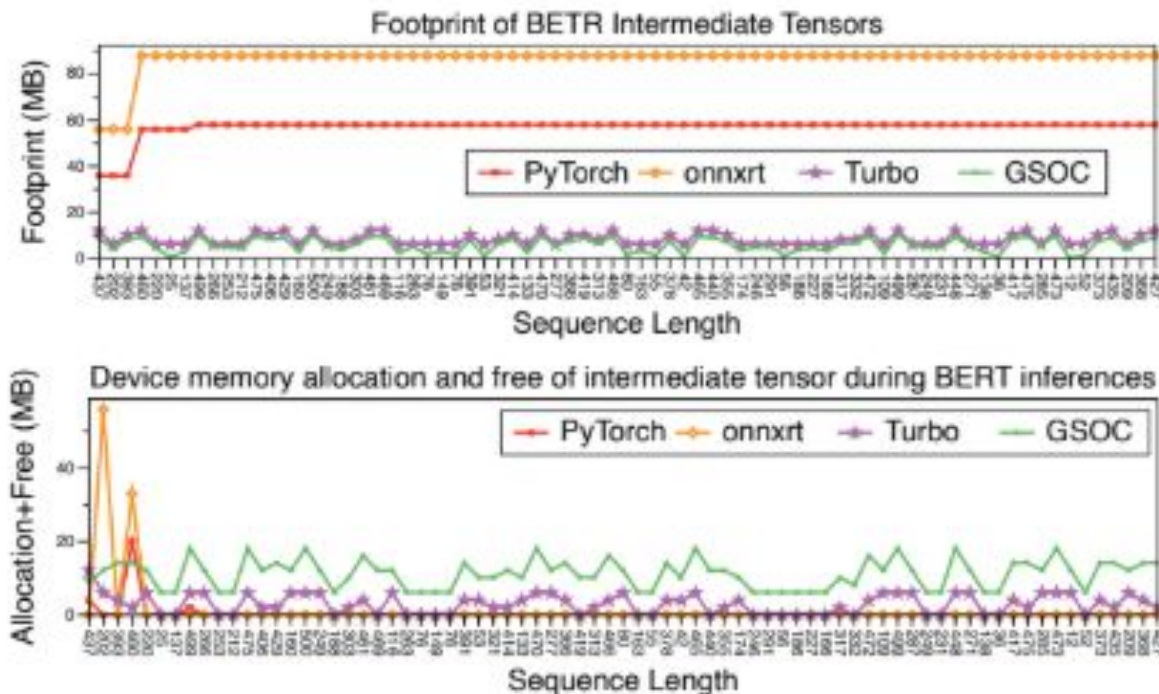


computation graph

Memory Allocation of seq len = 240

T2: Memory Allocation

TurboTransformers



T3: Kernel Fussion

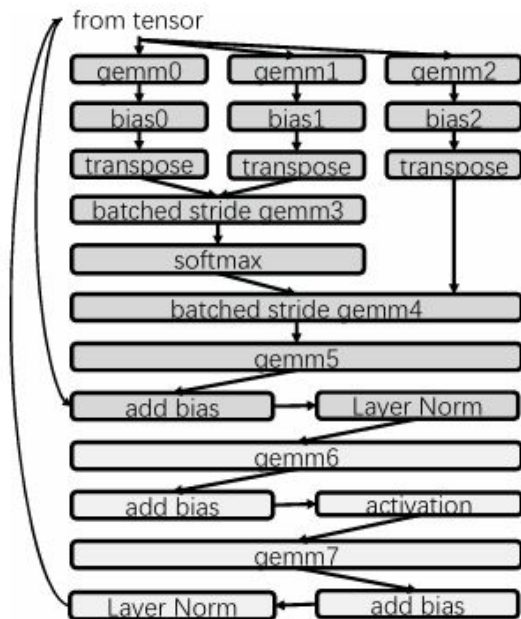
TurboTransformers, DeepSpeed

Problem	Kernel Invocation Adds Latency Overhead Global Memory Data-Transfers Between Kernels
Insights	Model Training Requires Activation Values Model Inference DOES NOT Require Activations
Solution	Fuse many operations (multiple kernels) into same kernel. Deep-Fusion: Tile multiple related kernels into single Tiled Kernel. Use CUDA Graphs

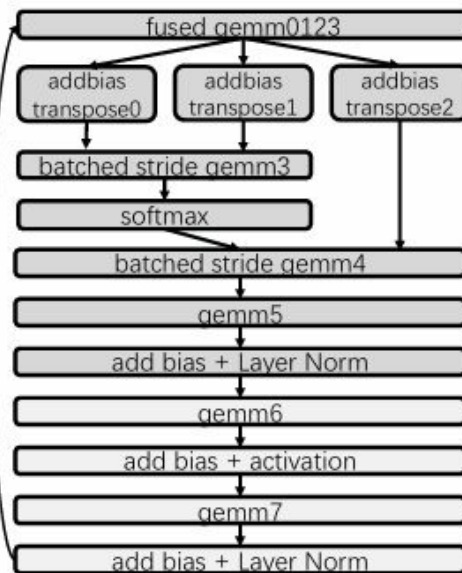
T3: Kernel Fussion

TurboTransformers, DeepSpeed

21 kernels
15 length
compute path



(a) Unfused Transformer Encoder Layer



(b) Fused Transformer Encoder Layer

13 kernels
11 length
compute path

T4: Model Parallelism

DeepSpeed

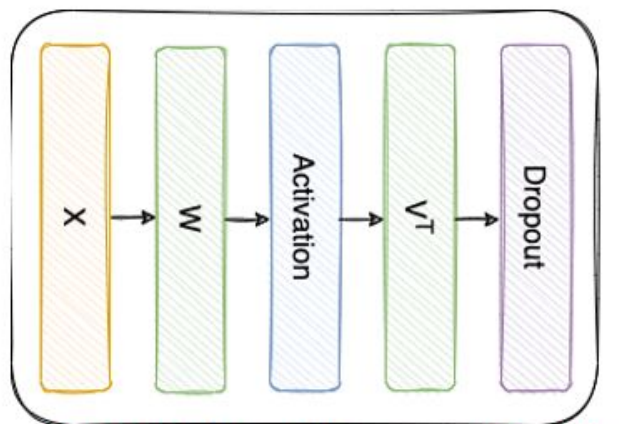
Problem	Big LLM Models (+100B params) do not fit in a single or multiple GPUs
Insights	GPUs connected in same node have high BW. Can use more nodes with lower BW.
Solution	Tensor parallelism (layers horizontal slicing) for GPUs same node Pipeline parallelism (layers vertical slicing) for more nodes.

T4: Model Parallelism

DeepSpeed

Tensor Parallelism (Intra-Layer)

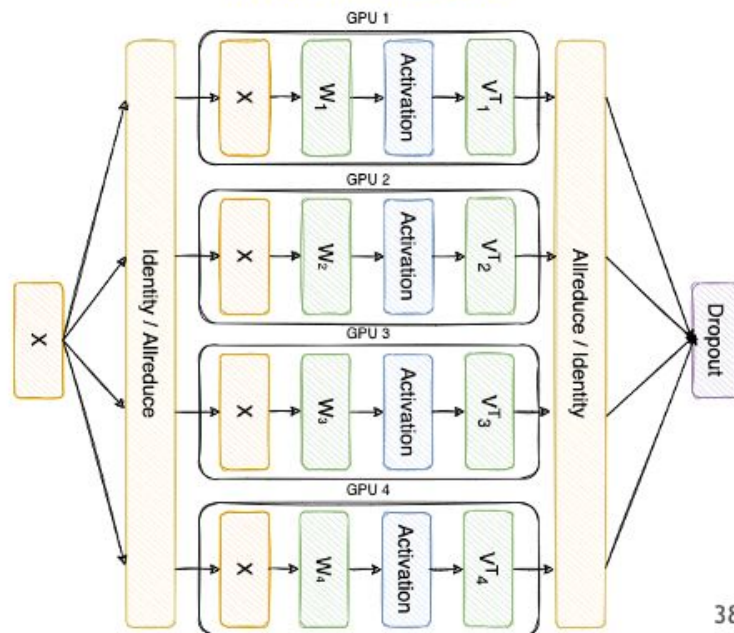
MLP



Split weights W and V^T across multiple GPUs

$$W = [W_1, W_2, W_3, W_4] \quad V = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix}$$

Tensor Parallel = 4 MLP

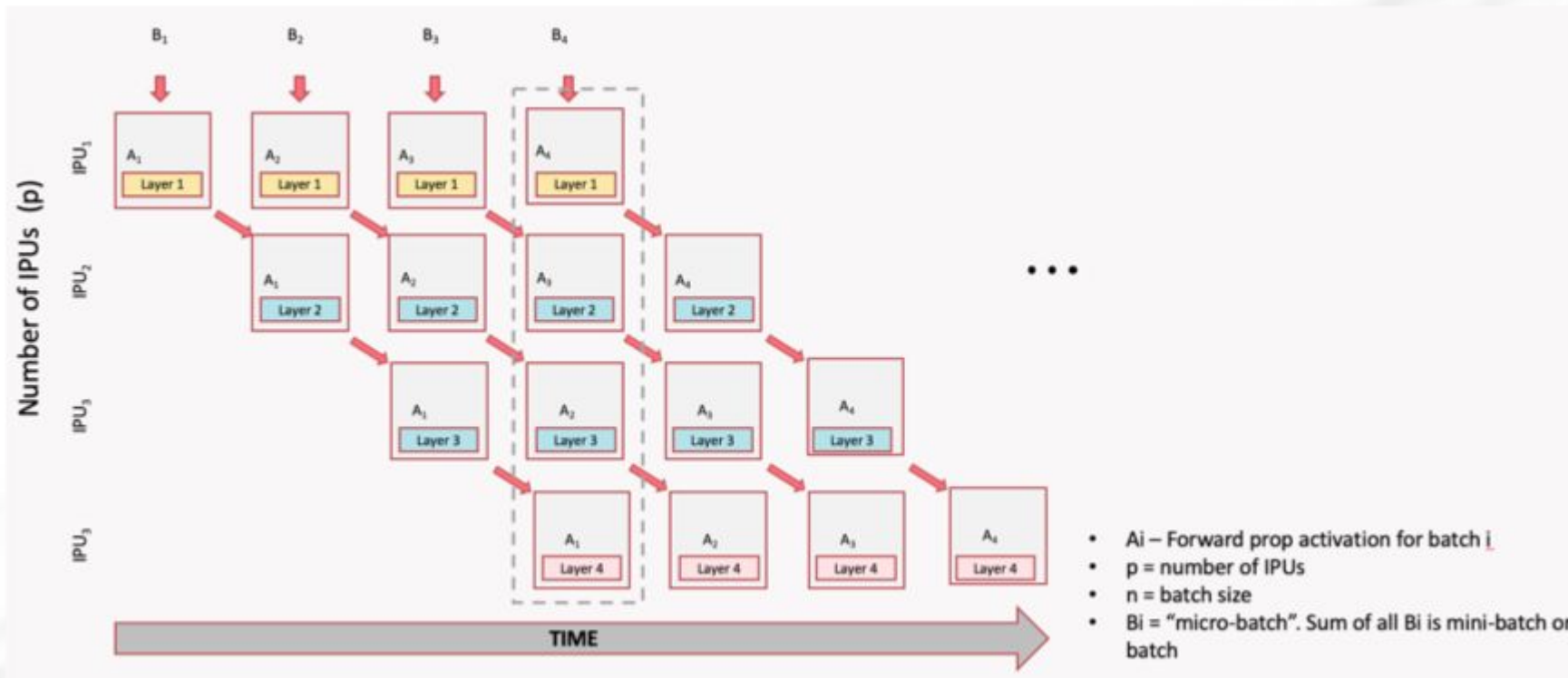


38

T4: Model Parallelism

DeepSpeed

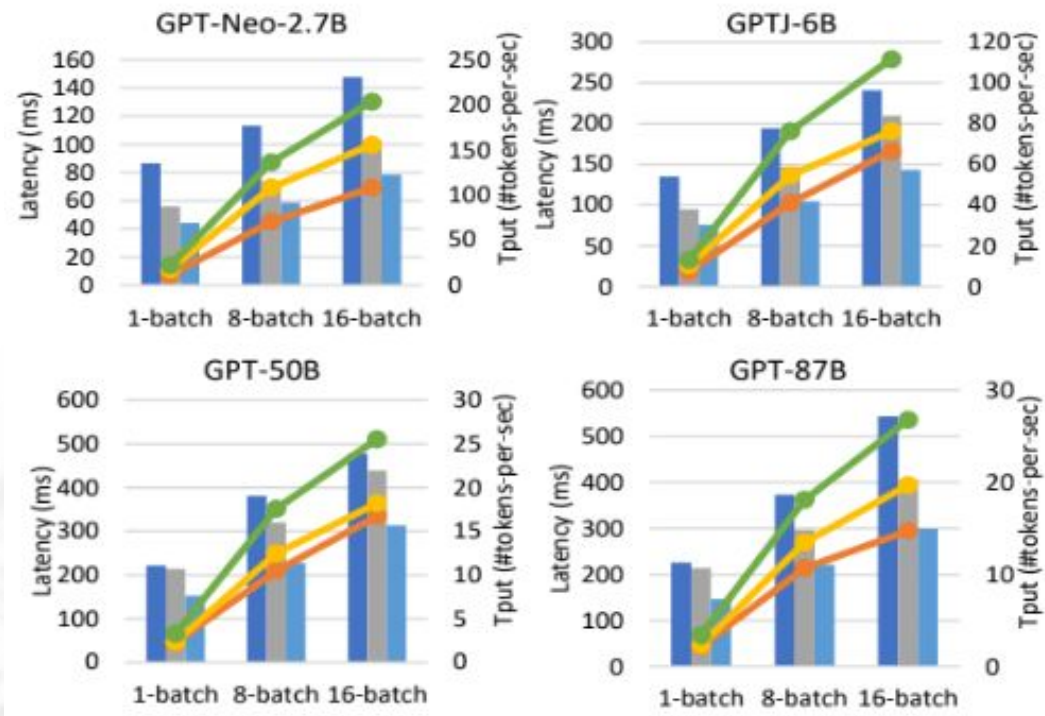
Pipeline Parallelism



T4: Model Parallelism

DeepSpeed

Model Scaling Results



Conclusions

- LLM Inference is a high-impact, compute-heavy task, many constraints → Interesting Problem.
- Multiple Techniques, Different Types.
 - Memory and Compute
 - Single GPU and Multi GPU performance.
- Enormous Effort To Put Everything Together
 - Heterogeneous computing
 - Evolving techniques
 - Variety of models and workflows.
 - Etc

References

- K. Hong et al., “FlashDecoding++: Faster large language model inference on GPUs,” arXiv.org, Nov. 02, 2023. <https://arxiv.org/abs/2311.01282>
- R. Y. Aminabadi et al., “DeepSpeed Inference: Enabling Efficient Inference of Transformer Models at Unprecedented Scale,” arXiv.org, Jun. 30, 2022. <https://arxiv.org/abs/2207.00032>
- J. Fang, Y. Yu, C. Zhao, and J. Zhou, “TurboTransformers: an efficient GPU serving system for transformer models,” arXiv.org, Oct. 09, 2020. <https://arxiv.org/abs/2010.05680>
- J.J.Olivera (2025) “LLM Model Size Impact on Application Performance”. UPC MIRI PPTM Presentation.

THANK YOU
FOR NOT
SLEEPING!

QUESTIONS??

