Distributed Training of Large-Scale LLMs on GPUs

Shoeybi et al., "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism", 2019
Narayanan et al., "Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM", SC 2021

Aditya Kamat | 04/29

Core Problem

- Training larger transformer models improve the SOA in NLP applications.
- Large models however require a large amount of VRAM to be trained on GPUs.
- Top of the line hardware at the time would not be sufficient to train larger models with the parallelism methods that were common.

8.3B

GPT-2 (2019)

175B

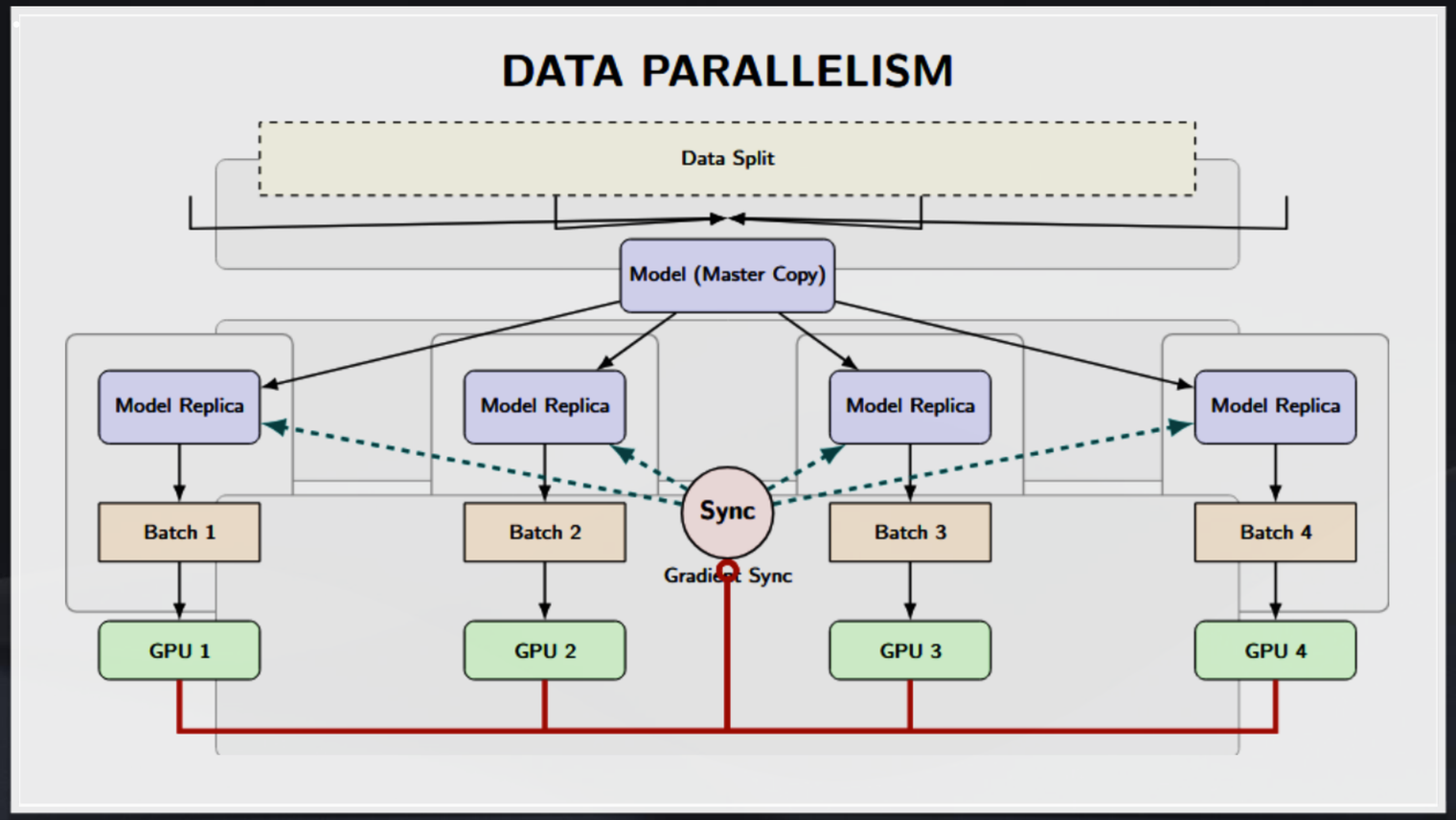
GPT-3 (2020)

1.0T

Kimi K2 (2025)

Existing Parallelism Methods

Data Parallelism



Existing Parallelism Methods

Inter Layer Pipeline Parallelism

- **Naive Approach**: Split layers across multiple GPUs and use them in a pipelined order.

Existing Parallelism Methods

Inter Layer Pipeline Parallelism

- **Naive Approach**: Split layers across multiple GPUs and use them in a pipelined order.
  - Problem: High GPU idle time as other than the active GPU all other GPUs remain idle. For p pipeline stages efficiency is only 1/p.

Existing Parallelism Methods

Inter Layer Pipeline Parallelism

- **Naive Approach**: Split layers across multiple GPUs and use them in a pipelined order.
  - Problem: High GPU idle time as other than the active GPU all other GPUs remain idle. For p pipeline stages efficiency is only 1/p.
- **Microbatching (GPipe):** Split a batch of data into micro-batches. Accumulate gradients across all micro-batches and update weights only after all micro-batches of a batch are done.
  - For p pipeline stages and a batch of data divided into m micro-batches, the compute efficiency is $2m/(2m + p - 1)$.

Existing Parallelism Methods

Inter Layer Pipeline Parallelism

- **Naive Approach**: Split layers across multiple GPUs and use them in a pipelined order.
  - Problem: High GPU idle time as other than the active GPU all other GPUs remain idle. For p pipeline stages efficiency is only 1/p.
- **Microbatching (GPipe):** Split a batch of data into micro-batches. Accumulate gradients across all micro-batches and update weights only after all micro-batches of a batch are done.
  - For p pipeline stages and a batch of data divided into m micro-batches, the compute efficiency is $2m/(2m + p - 1)$.
  - Sending activations between the GPUs (stages) adds an overhead.

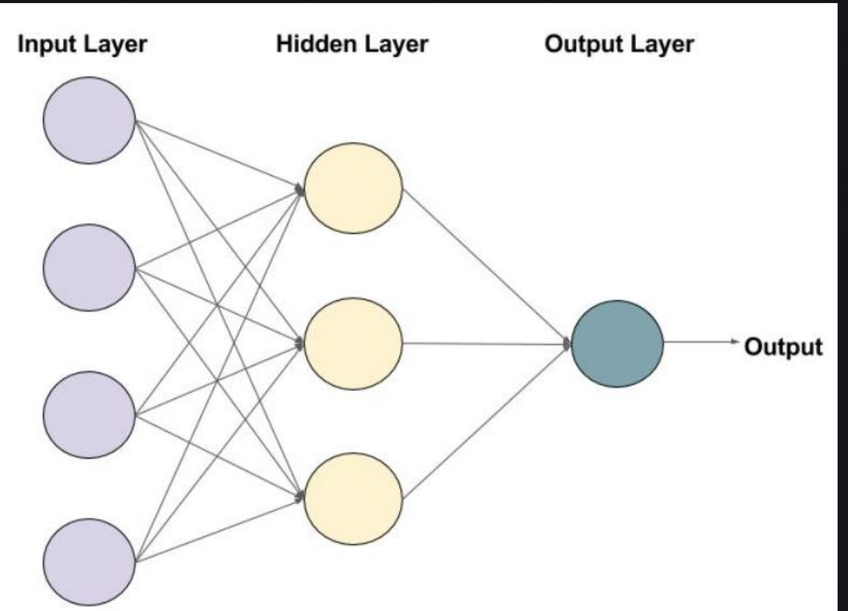
Megatron LM

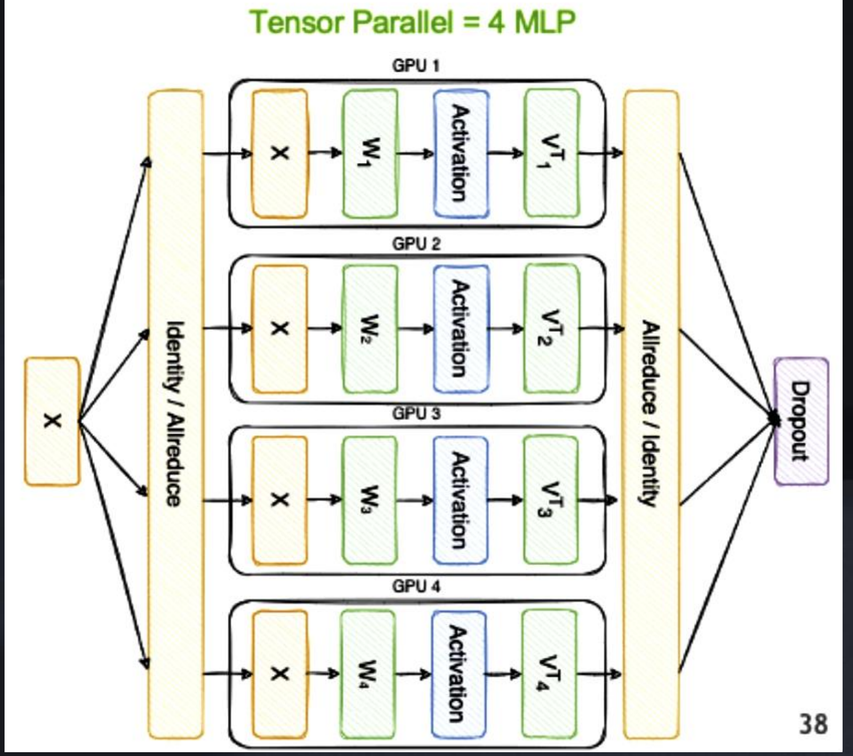
Tensor Parallelism

- The MLP feed forward network in LLMs consists of a Multi-Head Attention and a Feed Forward.
- Multiple matrix multiplications involved.

$$hidden = GeLU(X \times W)$$
$$output = hidden \times V$$

- Split W column wise on multiple GPUs. Each GPU computes its own slice of the hidden layer independently.
- Split V row wise. Each GPU multiplies its partial hidden by its partial V. Final outputs are then summed using an all-reduce function and then projected.

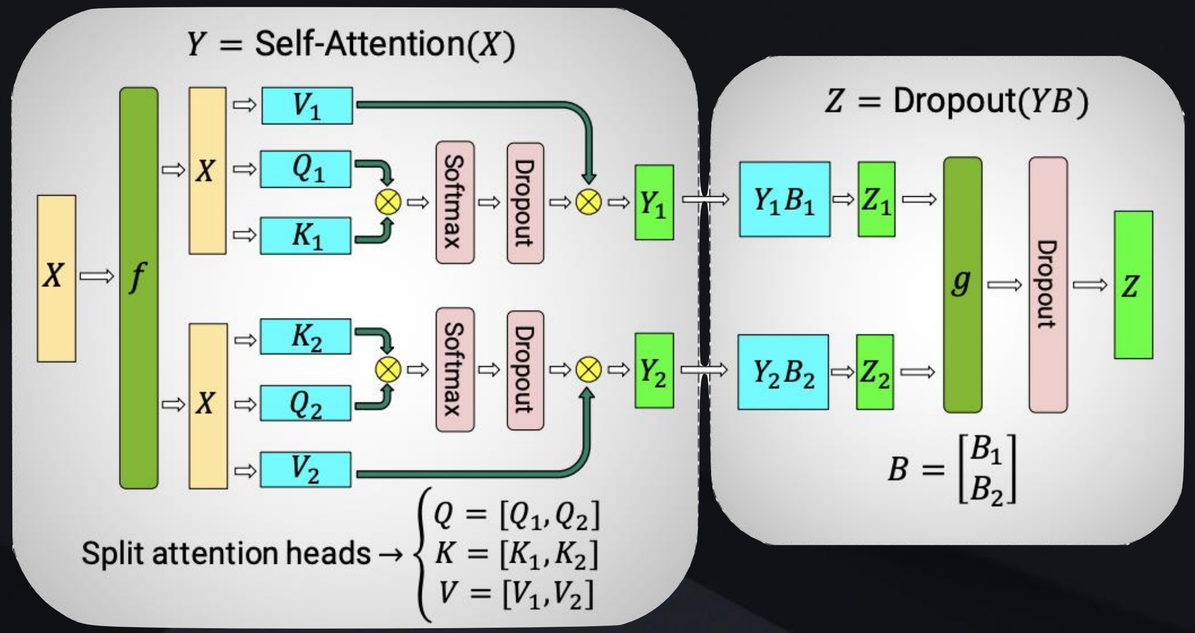




Megatron LM

Splitting Attention

- Each GPU handles a subset of the attention heads.
- Since heads are independent of each other during the Q,K,V computation, there's no communication needed until the outputs are concatenated and projected.



Megatron LM

1F1B Pipeline scheduling

- Backward pass for a micro batch begins as soon as the gradient is ready.

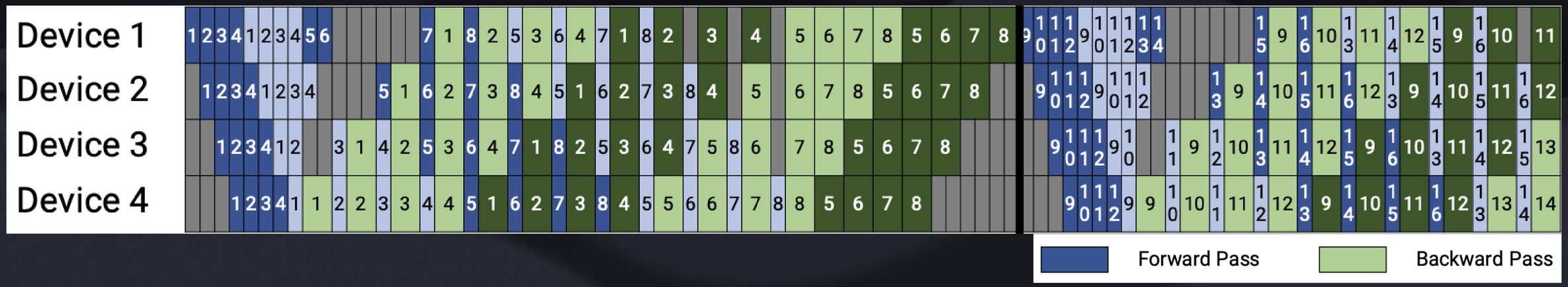
- $Bubble = (p-1)/(m+p-1)$

Megatron LM

Pipeline stage interleaving

- Instead of each GPU owning one contiguous block of layers, each GPU owns multiple non-contiguous blocks of layers.
- Idle stages present in the case of simple 1F1B scheduling can now be filled with the forward passes of a different layer on the same GPU.

$$Bubble = (p-1)/(v \times (m+p-1))$$



Megatron LM

Pipeline stage interleaving

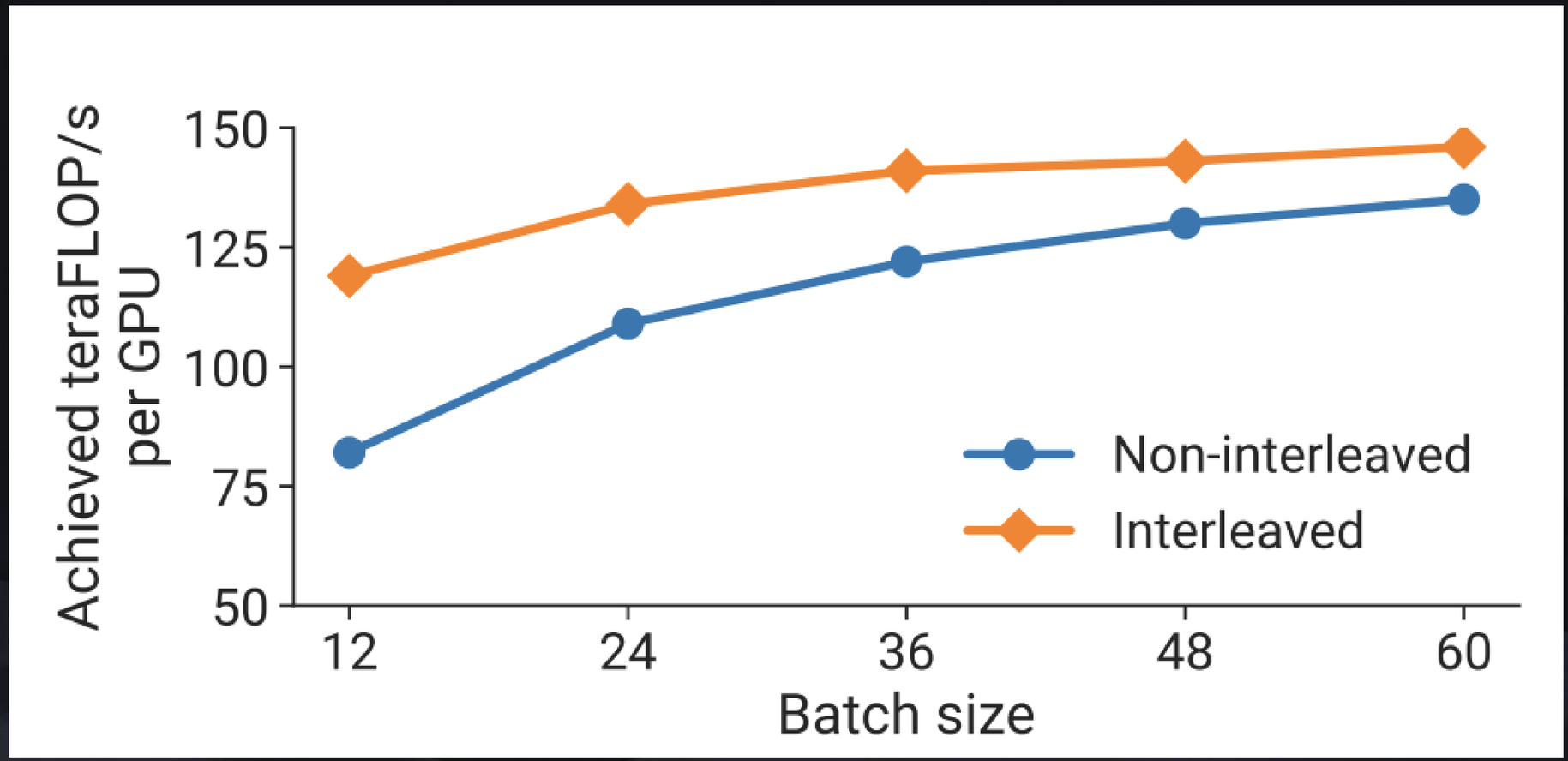
- Instead of each GPU owning one contiguous block of layers, each GPU owns multiple non-contiguous blocks of layers.
- Idle stages present in the case of simple 1F1B scheduling can now be filled with the forward passes of a different layer on the same GPU.

$$Bubble = (p-1)/(v \times (m+p-1))$$

- **Overhead:** Each virtual stage boundary requires sending activations between GPUs — and with v virtual stages per GPU, you now have v times more pipeline boundaries per micro-batch.

Megatron LM

Pipeline Stage Interleaving



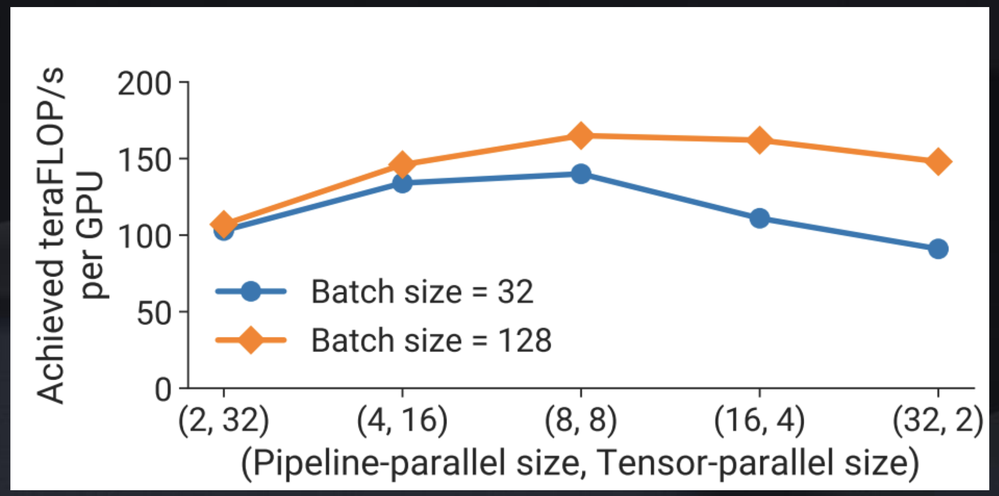
Throughput per GPU of interleaved and non-interleaved schedules for a GPT model (175 billion parameters) on 96 GPUs.

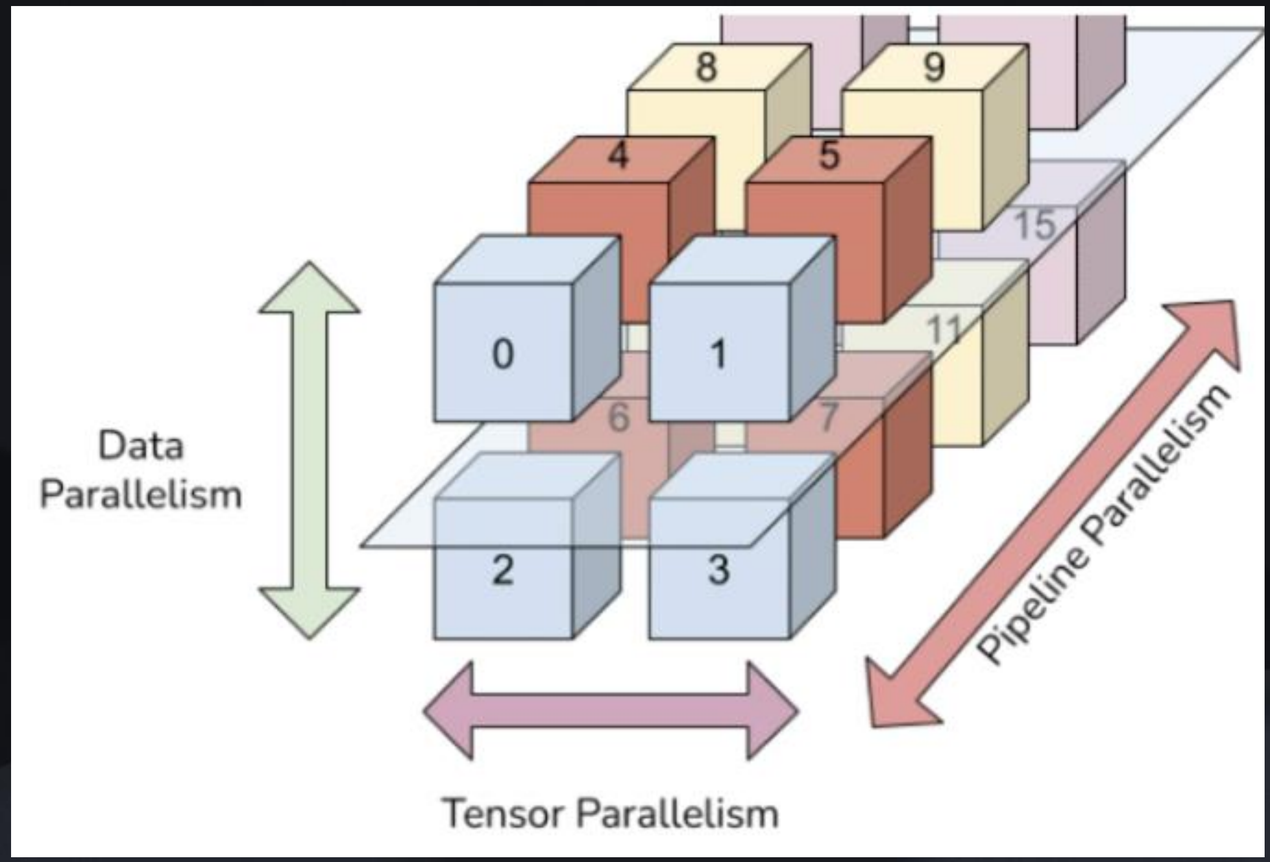
Megatron LM

The Parallelism Cube

- Utilize all three forms of parallelism to maximize throughput.

$$N_{GPUS} = d \times t \times p$$





Thank You