The story of training the Llama-3 model

A Meta Labs Production

Scaling Llama 3 Training with Efficient Parallelism Strategies, Weiwei Chu, Xinfeng Xie, Jiecao Yu, Jie Wang, et al, ISCA-2025

Training Setup

405B

Model Parameters

16K

NVIDIA H100 GPUs

16M

Tokens per training step

~1K

Maximum TFLOPs for BF16 that can be achieved

Capability Computing challenge

Applications

Short Context

Long Context

Multimodal

Short Context

8192

Context Length
Tokens in a single sequence

16M

Tokens per training step

2048

Batches of 8K tokens in one sequence

405B

Model Parameters

Distribute Model and Data across 16K GPUs

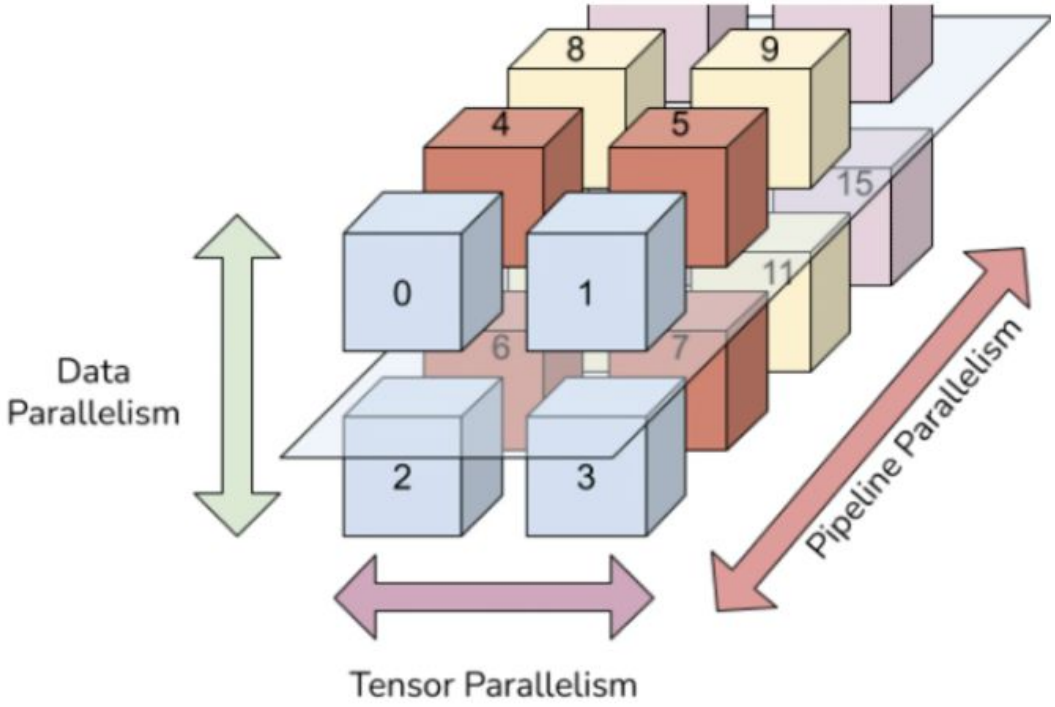
Short Context

Data Parallelism

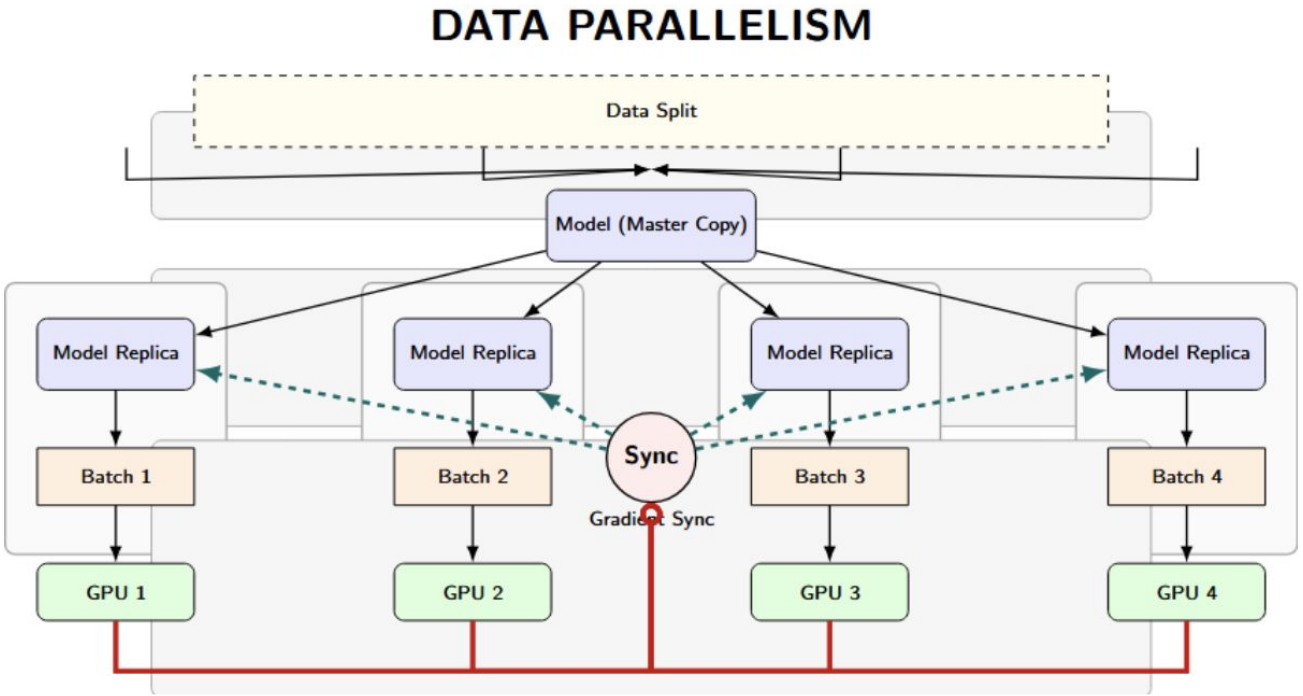
Pipeline Parallelism

Tensor Parallelism

Num of GPU = PP x TP x DP

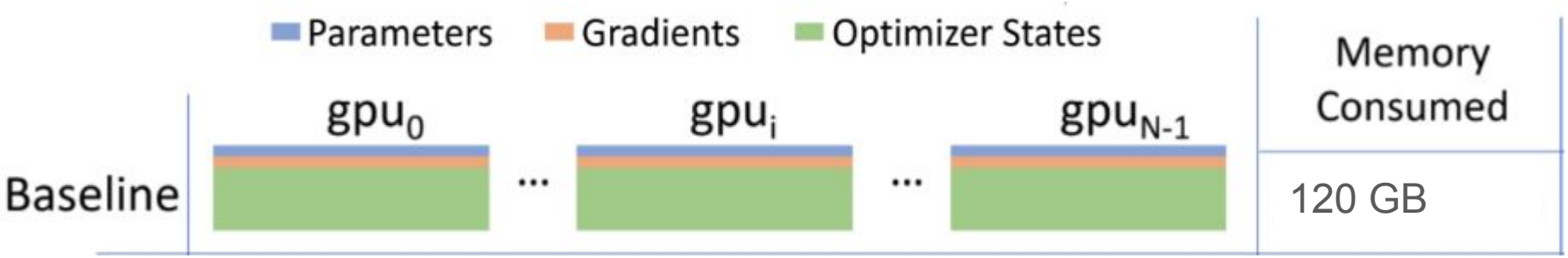


Distributed Data Parallelism



Gradient sync of weights at the end of every training step
Full computation on sharded data

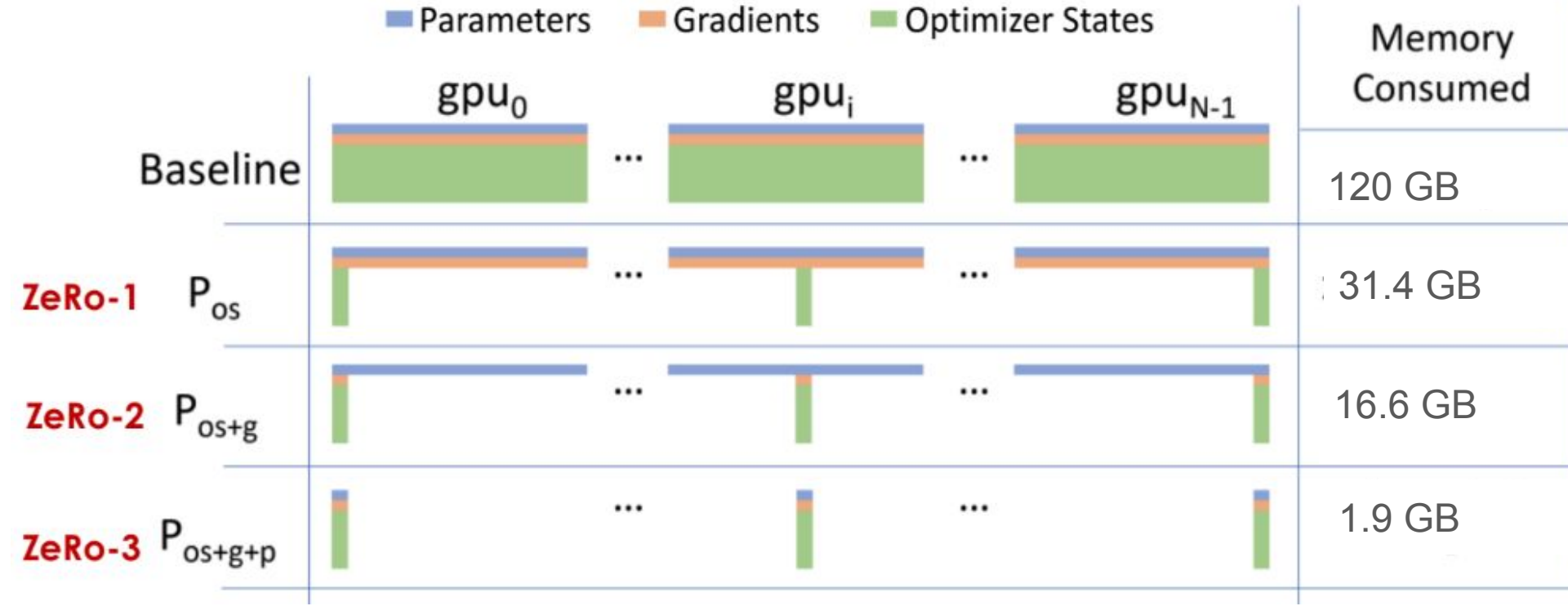
Data Parallelism Extended/Redefined - Fully Sharded Data Parallelism (FSDP)



Data Parallelism Extended/Redefined - FSDP

ZerO : Zero Redundance Optimizer - popularized by DeepSeek

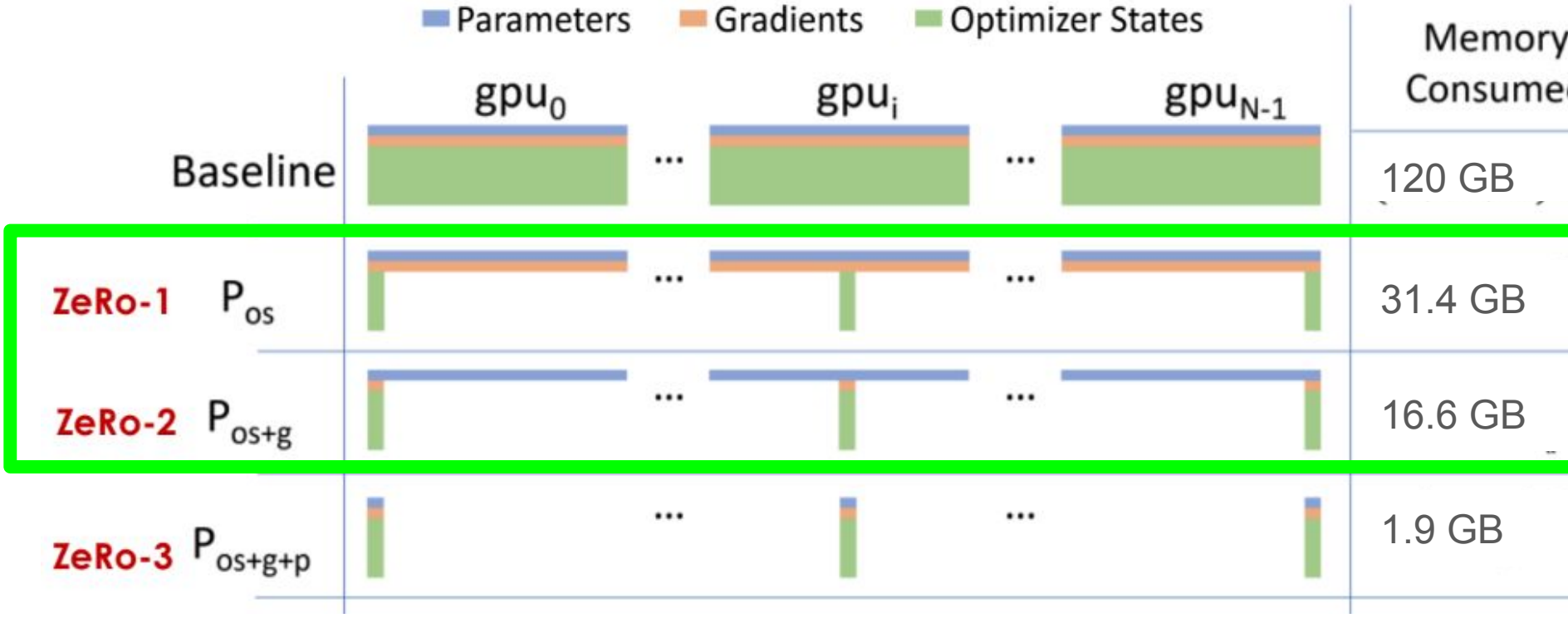
- shard optimizer states, gradients and models



Data Parallelism Extended/Redefined

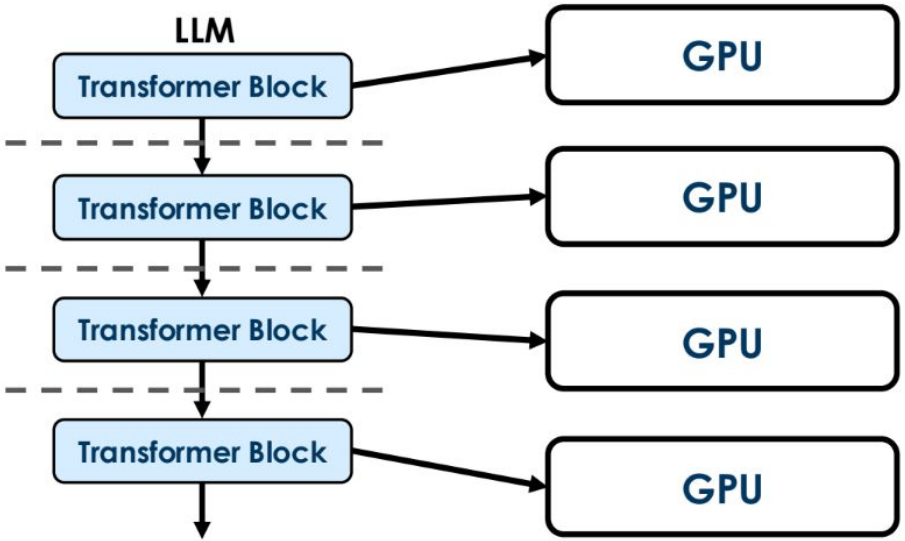
ZerO : Zero Redundance Optimizer - DeepSeek

- shard optimizer states, gradients and models



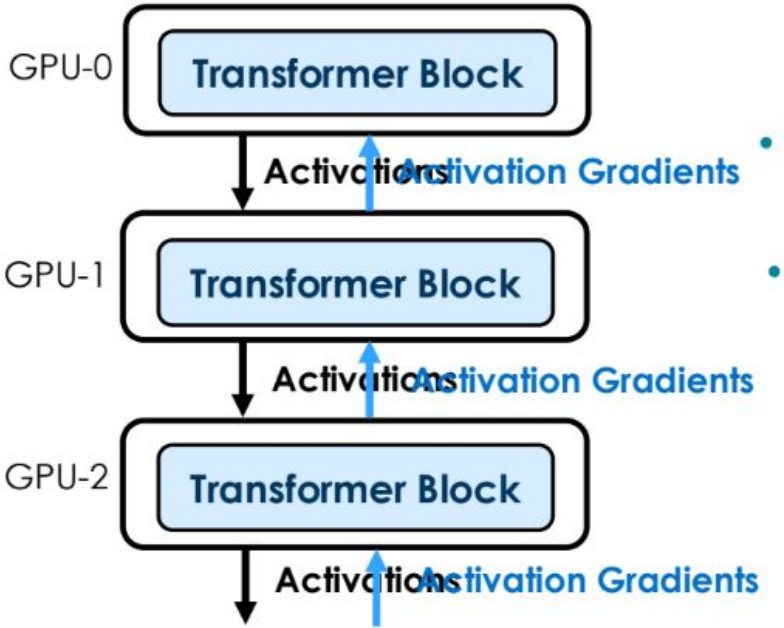
Pipeline Parallelism

Inter-layer Partitioning



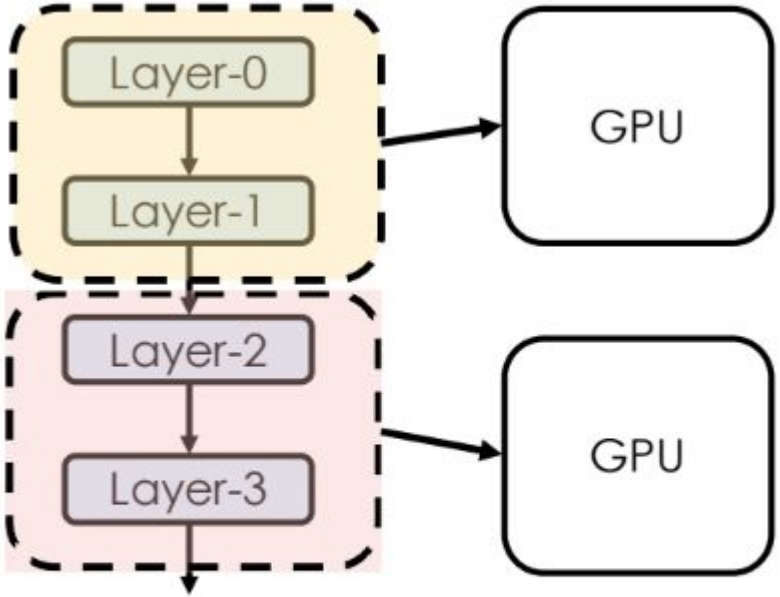
Pipeline Parallelism

Communication Pattern



- P2P communication for activations in the forward pass
- P2P communication for activation gradients in the backward pass

Pipeline Parallelism

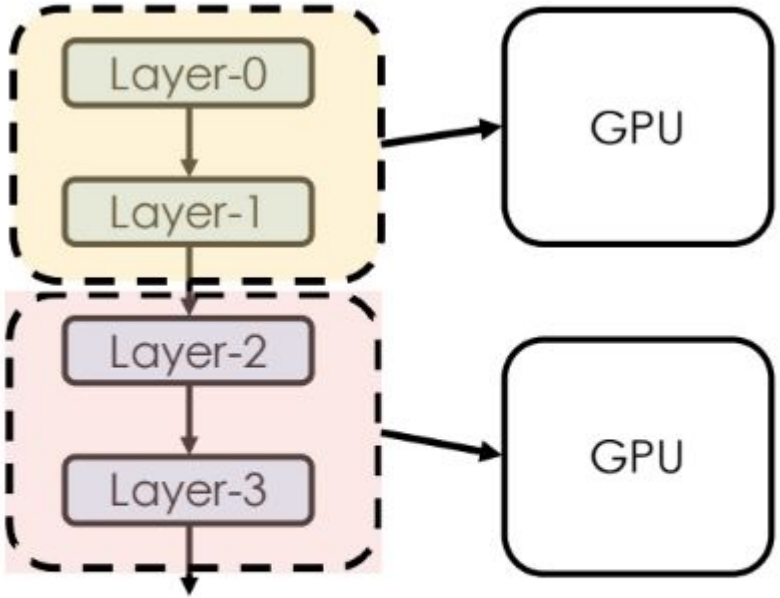


- One micro batch will go through both GPU but only once.

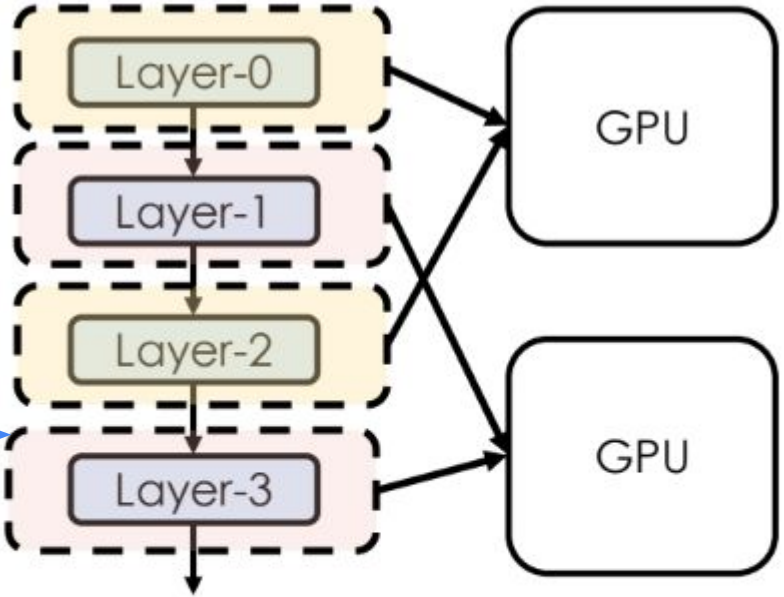
Feasible with 128 vertical layers? Efficient?

Pipeline Parallelism

128 vertically stacked layers



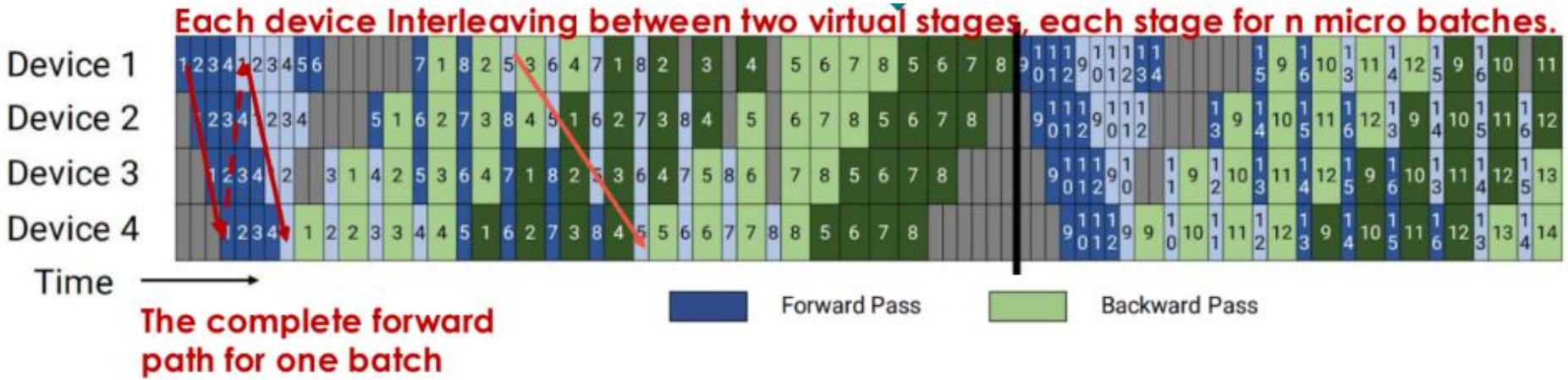
- One micro batch will go through both GPU but only once.



- One micro batch will go through both GPU but now twice.
- GPU are now busy working, just like we have two batches

Pipeline Parallelism

Based largely on Megatron-LM's 1F1B interleaving with virtual stages



Bubble ratio = (PP-1) / (nmb × v)

Nvidia, Stanford, Microsoft, Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM, 2021

Pipeline Parallelism

Megatron way too rigid.

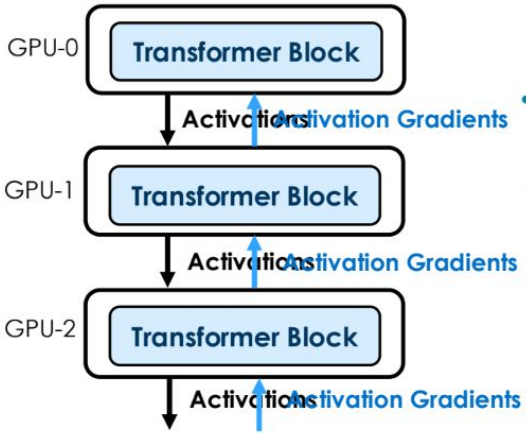
Constraint : batch size to be a multiple of the number of PP ranks for less bubble ratio, Llama 3 changes batch size across training phases

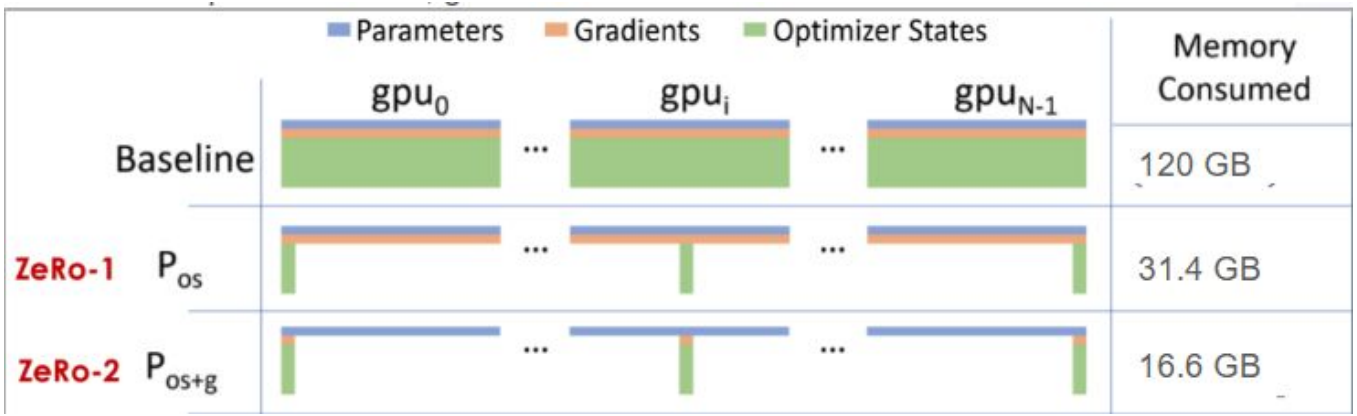
Pipeline Parallelism

Fixes:

1) Flexible PP schedule that supports arbitrary batch size.
   - Extra micro-batches added to overlap point-to-point communication
2) Balanced PP with model co-design
   - First and last layer avoid OOM
3) Co-optimization of PP and FSDP

Co-optimizing PP and FSDP for 3D





ZeRO-2 communication overhead > ZeRO-1 Communication Overhead

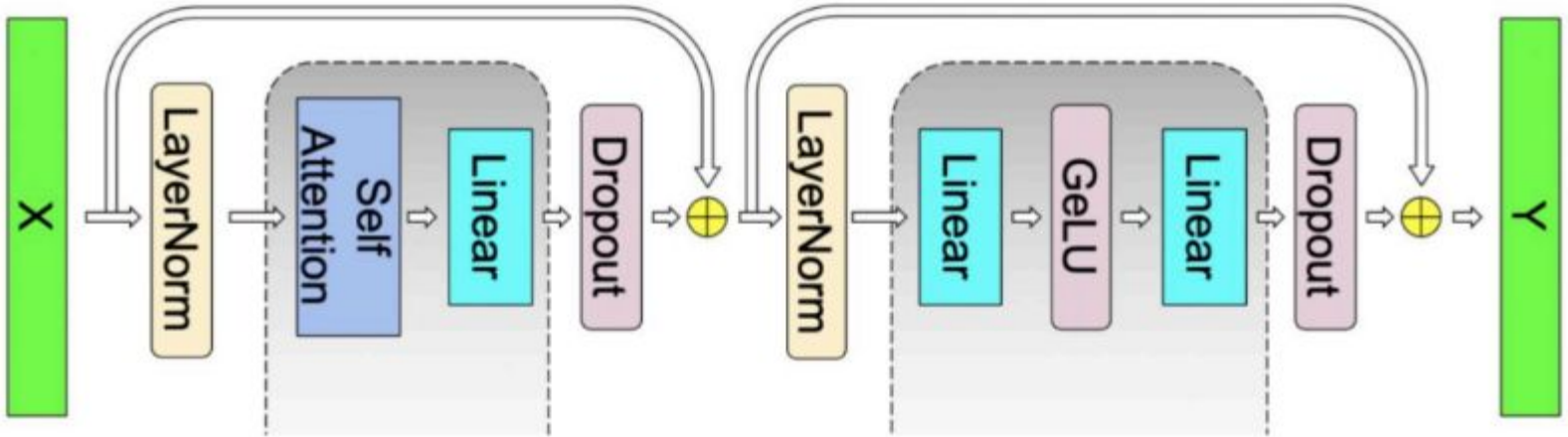
FSDP ZeRO-1 with the 1F1B schedule when $bs >= 2 \times pp$, to hide latency
ZeRO-2 with all-forward-all-backward when$bs < 2 \times pp$, to achieve better performance

Tensor Parallelism

Intra-layer partitioning:

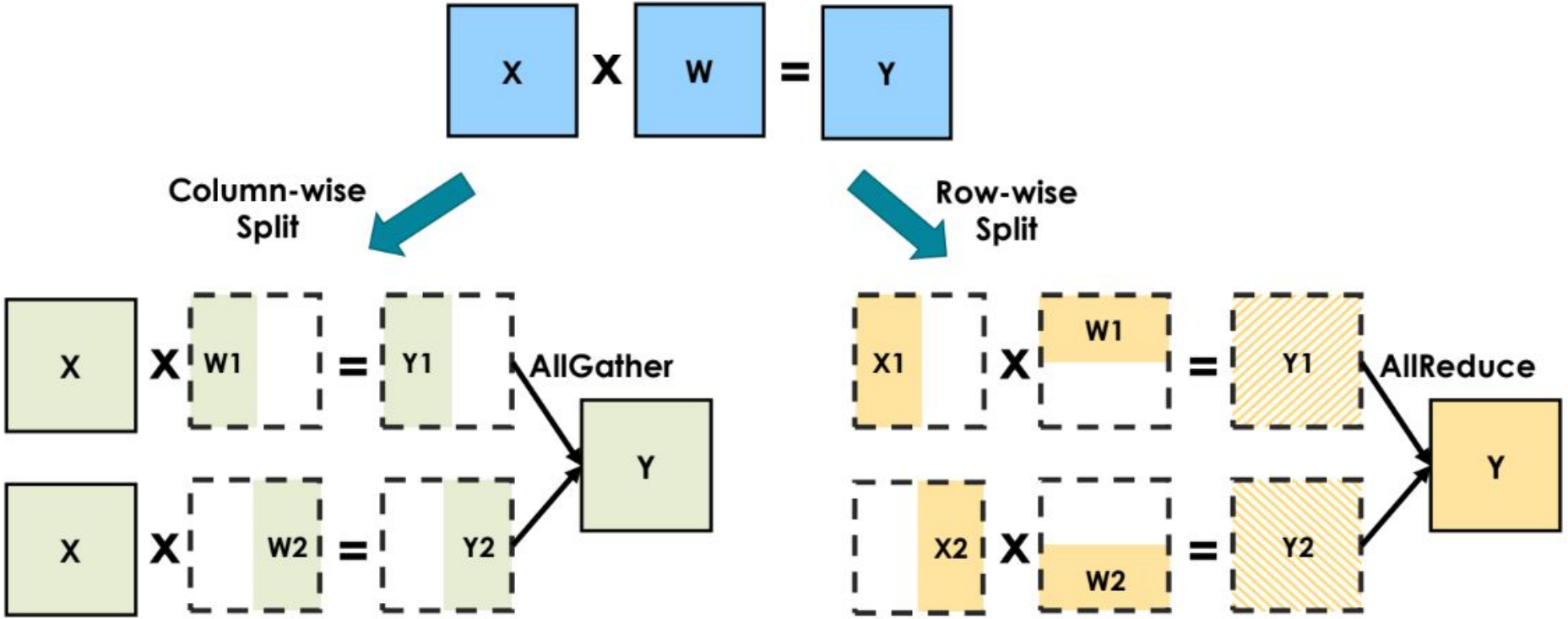
- The vast majority of LLM parameters reside in Linear Projections:
    - Attention: Q, K, V, and Output projections.
    - Linear layer - GEMM

These projections are General Matrix Multiplications (GEMM) (Y=XW).



Tensor Parallelism

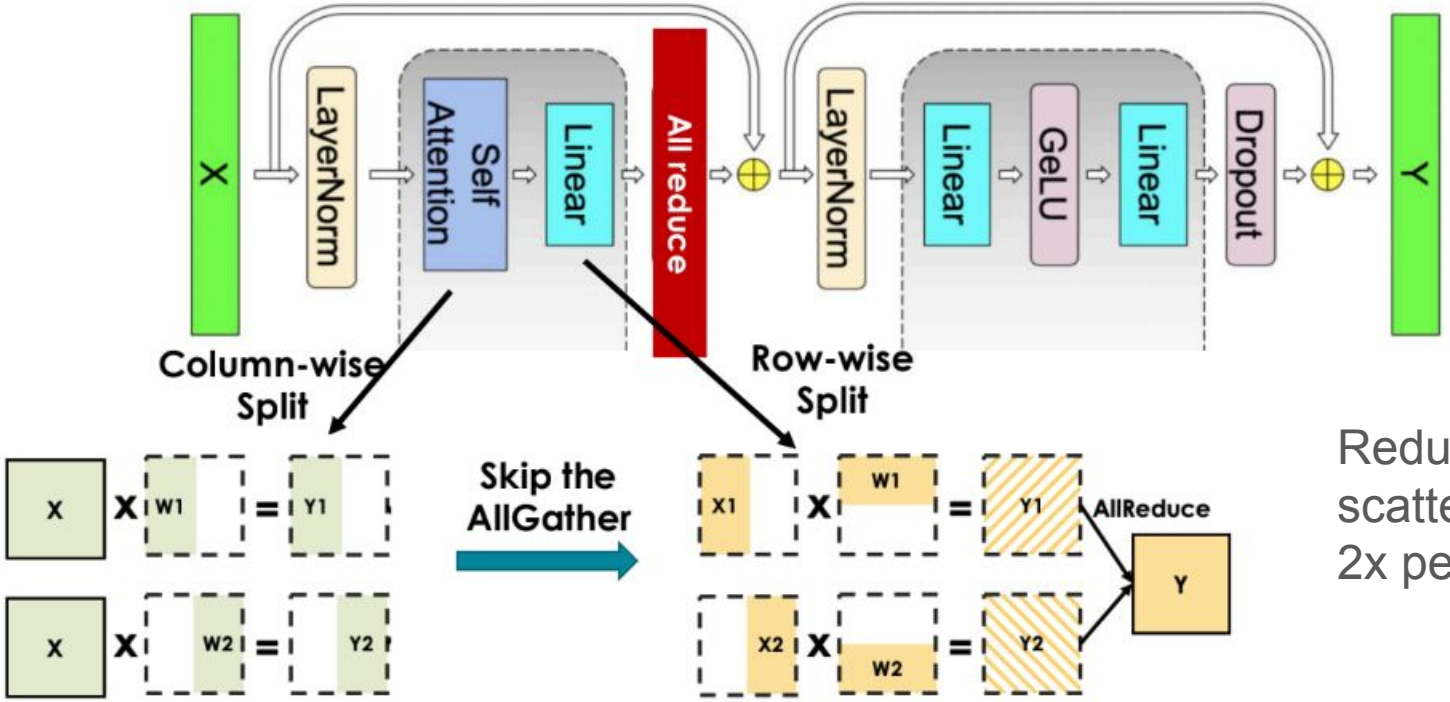
Ways to partition a GEMM:



Tensor Parallelism

Heavy communication : 4x gather/scatter on one block

Interleaving split to minimize communication:



Reduces scatter/gather by 2x per block

Physical Mapping & Configuration

Context Length = 8192 tokens

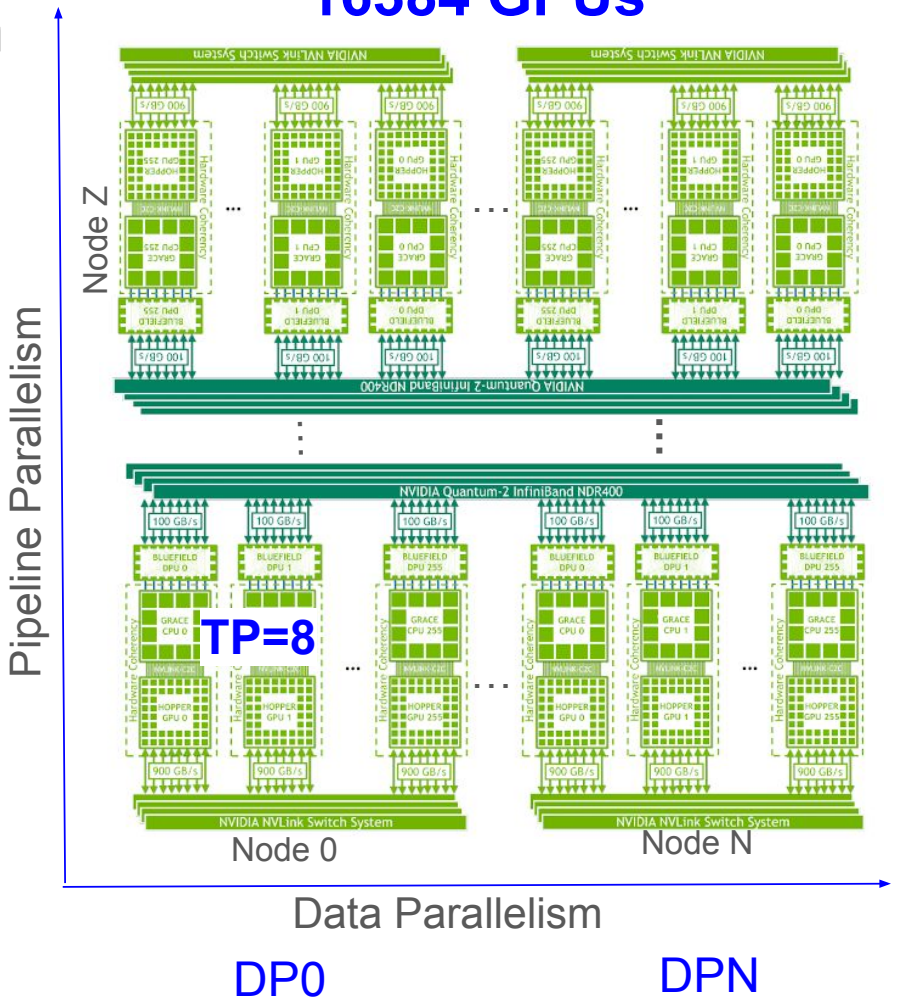
Global token budget = **16M tokens**;

Constraints

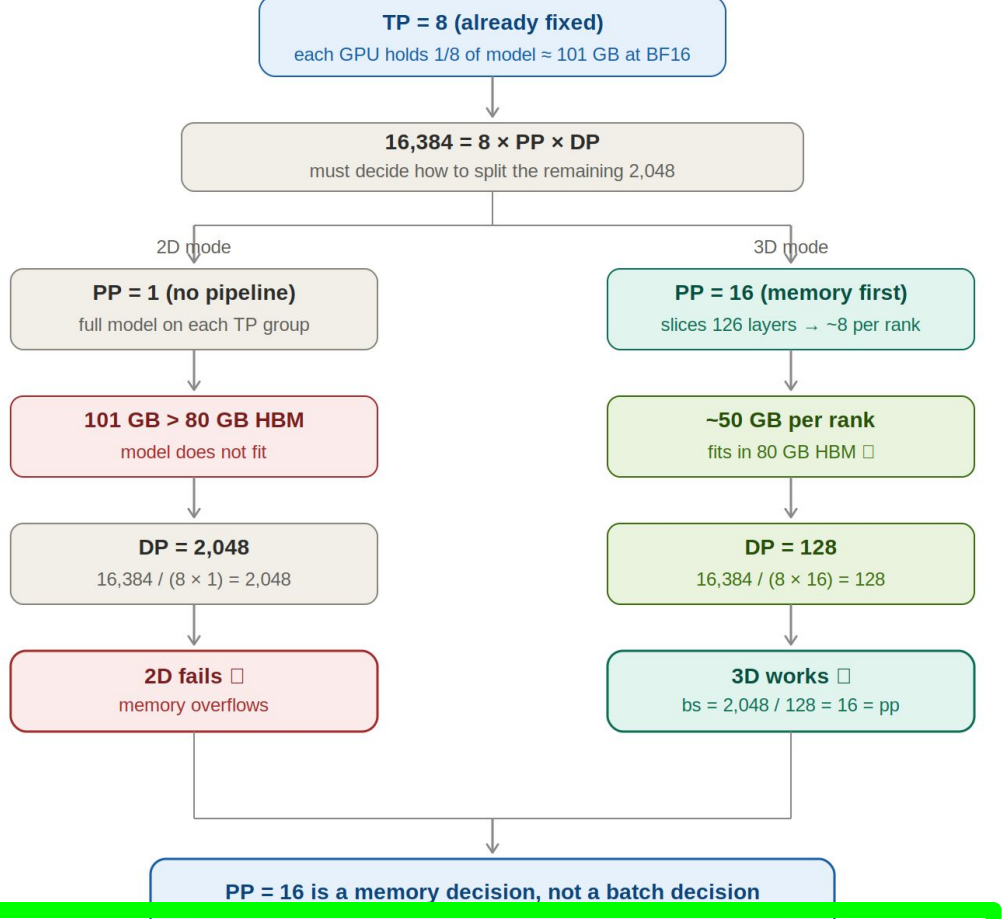
Num of GPU = PP x TP x DP **(spatial)**

bs = global batch size/DP ; **(temporal)**
bs >= PP

Each server node(8 GPUs)
connected by NVLINK



Piecing it together



Context Length	Global Batch Size	TP	CP	PP	DP
8,192	2,048	8	1	16	128

Long Context

Context window size continuously growing in state-of-the-art models:

- Gemini3: 1 million
- ChatGPT-5: 400K
- Grok 4 Fast: 2 million
- Llama 3 : 131K

Long Context - 3D Model breaks

Increasing context Length to 131,072 tokens,

Mem exhausted

PPN

Global token budget = **16M tokens**;

Constraints

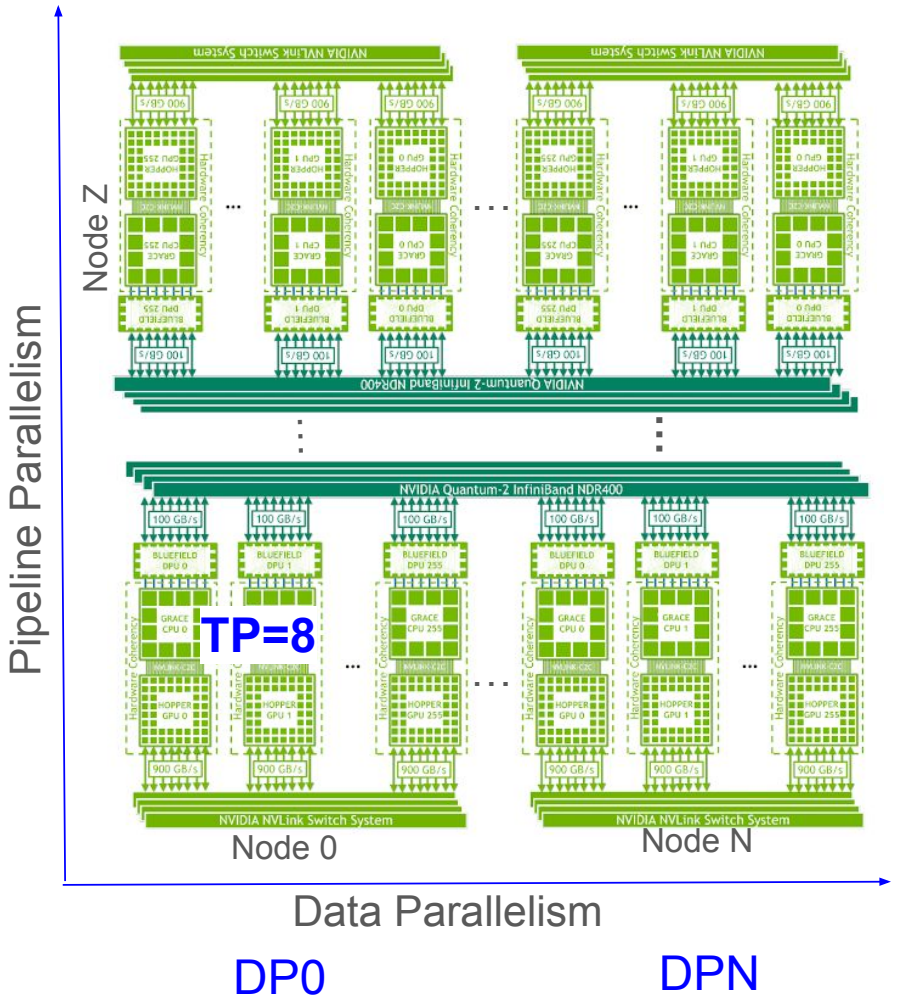
Num of GPU = PP x TP x DP **(spatial)**

bs = global batch size/DP ; **(temporal)**
bs >= PP

TP = 8, fixed. PP = 16, fixed,
num of GPU fixed, therefore DP = 128

PP0

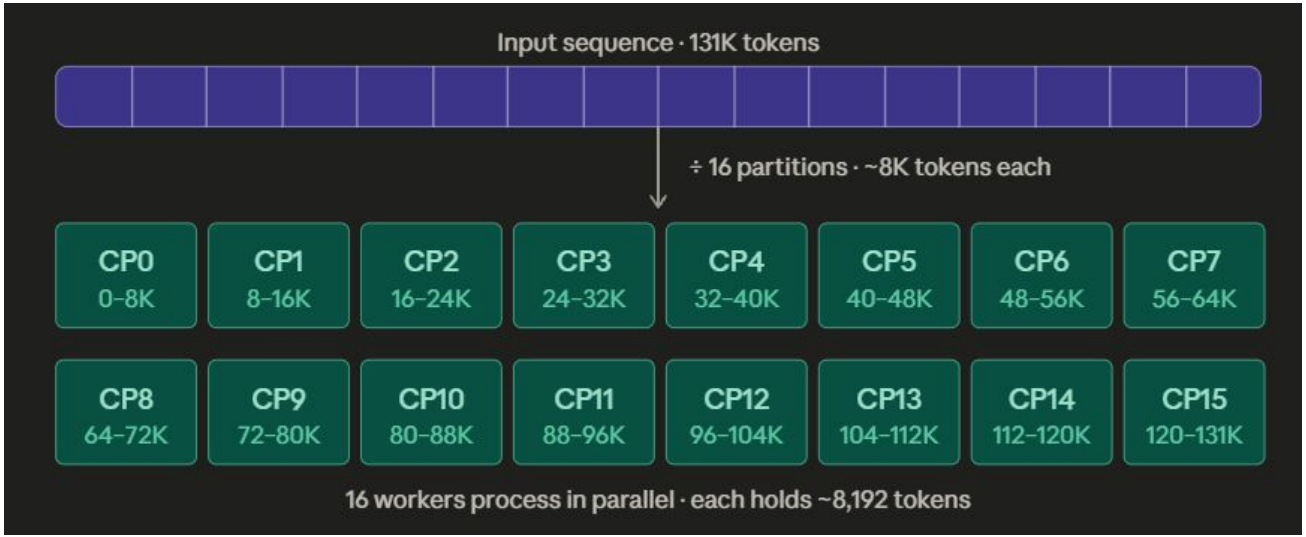
bs=1 and PP won't work. No batches to hide bubbles



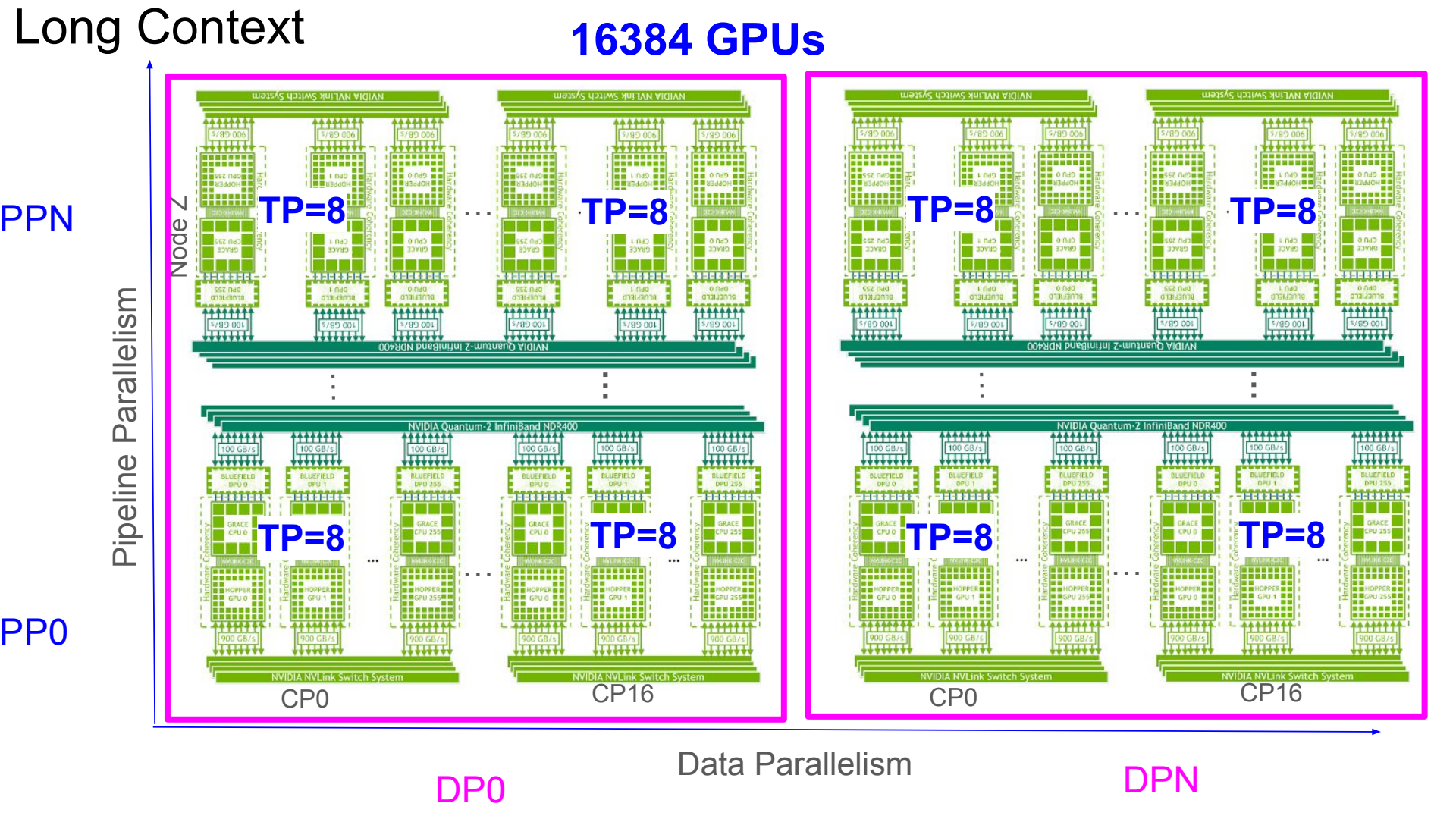
Long Context

New Dimension of parallelism: Context Parallelism(CP)

- splitting input tokens along the sequence length dimension.



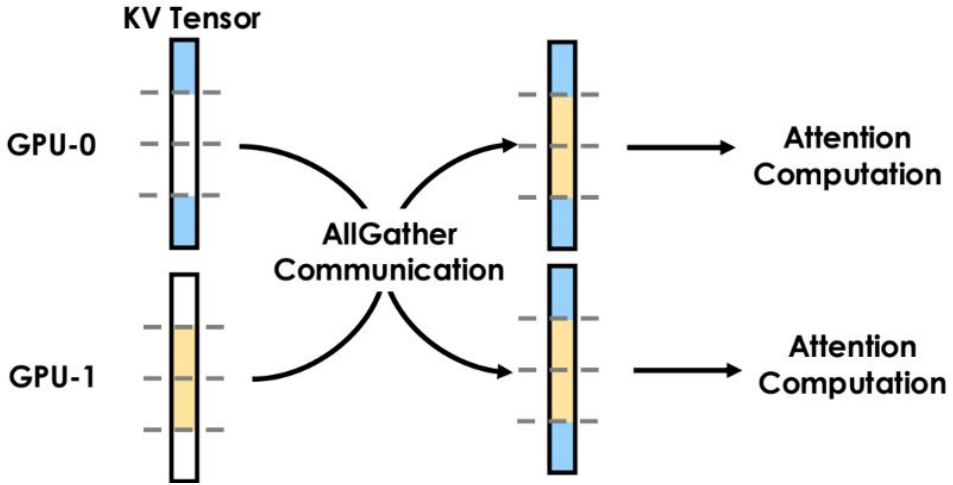
Drawback: Full sequence needed for attention computation


Long Context
16384 GPUs
PPN
PP0
Pipeline Parallelism
Node Z
TP=8
TP=8
TP=8
TP=8
TP=8
TP=8
TP=8
TP=8
NVIDIA Quantum-2 InfiniBand NDR400
NVIDIA NVLink Switch System
CP0
CP16
CP0
CP16
Data Parallelism
DP0
DPN

Long Context

Paper proposes

- **All-gather** based CP attention computation
  - all-gathering key(K) and value(V) tensors before attention computation
- Comm complexity: O(seq) vs Compute: O(seq^2)
- Less overhead and works on irregular masks



Communication-Aware Parallelism Ordering

Innermost

TP

4× collectives/layer, fully exposed, must use fast NVLink

2nd

CP

1× allgather/layer, fully exposed, inter-node but before PP. Needs inter-node communication

3rd

PP

Async P2P, partially hideable, lower sync cost

Outermost

DP

Once per step, fully overlappable with computation

4D Parallelism final configuration for Llama 3 405B

Context Length	Global Batch Size	TP	CP	PP	DP
8,192	2,048	8	1	16	128
131,072	128	8	16	16	8

End-to-End Performance Analysis

400

TFLOPs/GPU

Short Context (8K seq)

380

TFLOPs/GPU

Long Context (131K seq)

5%

Bubble Ratio

when batch = 2×PP stages

12%

Bubble Ratio

when batch = PP stages

Long Context CP Latency Deep Dive (cp = 16, 8K GPUs)

7.64%

CP communication as % of total step time

65.75%

Of that, due to workload imbalance (doc mask)

1.44×

Slowest rank vs fastest rank computation time

2.62%

Max gain from overlapping comm+compute

Multimodal Training

Pre-trained Text model of 128 layers

Pre-trained ViT model of 32 layers

During pre-training,

- self-attention layers are frozen
- image encoder and cross-attention

  Layers trained

Sharding strategy??

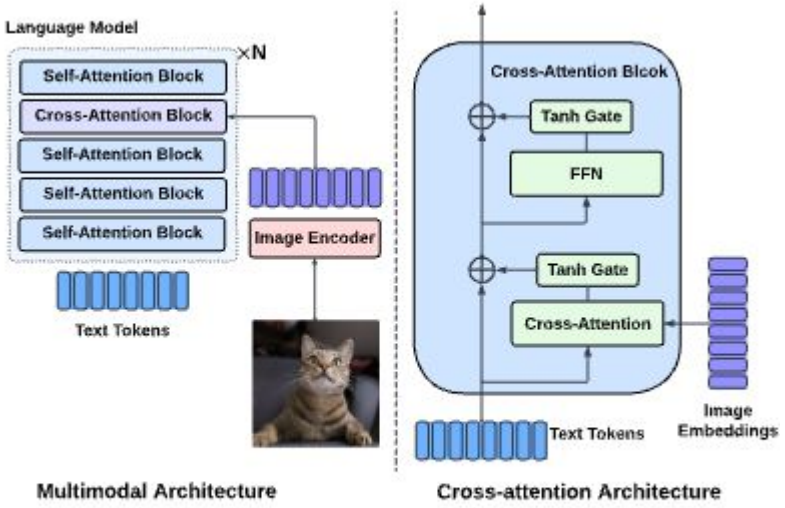


Figure 5: Illustration of Llama 3 multimodal architecture.

Parallelising multimodal training

HYBRID SHARDING

The image encoder is sharded using 2D parallelism (FSDP and TP), while the text model is sharded using 3D parallelism(FSDP, PP, TP)

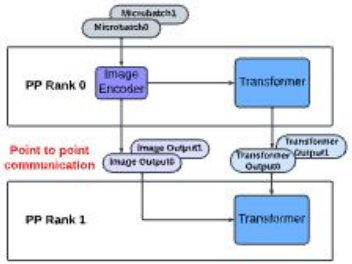
Challenge 1: Sharding of image encoder.

Challenge 2: Workload imbalance of the text model.

Multimodal case study: where to put the image encoder

1 Encoder on first PP rank

Place image encoder on rank 0 alongside text layers. Pass image features down with activations via P2P



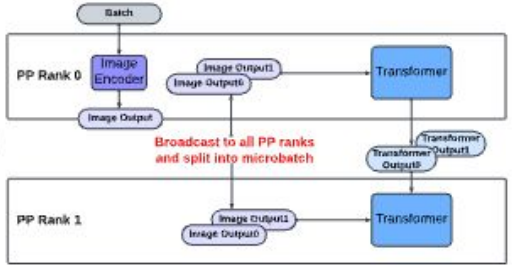
(a) Shard the whole (image+text) model in PP

✓ **Minimal code change**

✗ Worsens PP imbalance; rigid as encoder size changes

2 Pre-process + broadcast

Run image encoder once on rank 0 as a pre-processing stage; broadcast image tokens to all PP ranks; run text PP normally.



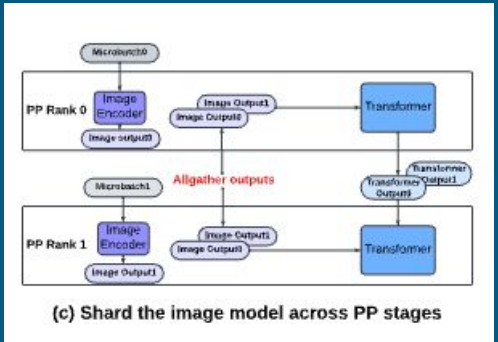
(b) Separate the image and text model, and apply PP only to the text model

✓ **Clean separation of encoder and text model**

✗ Encoder grew from 448→672 px; latency hit 33% of step time

3 Replicate on every PP rank

Replicate image encoder on every PP rank, each processes bs/pp slice in parallel, all-gather outputs before text



(c) Shard the image model across PP stages

✓ **Encoder load drops from 33% → 8% of step time**

✗ More memory per rank for the encoder copy

CHOSEN

Parallelising multimodal training : Challenge 2

The Co-Design Solution: They wrapped multiple self-attention layers and one cross-attention layer together into one single PP virtual stage. The final determined ratio was 4, self-attention layers for every 1 cross-attention layer (4:1)

.

Key Takeaways

01 4D Parallelism Enables Hyperscale

FSDP+TP+PP alone is insufficient for long context. Context Parallelism (CP) is the critical 4th dimension that makes 128K+ training feasible on 16K GPUs.

02 Communication Order Determines Architecture

[TP → CP → PP → DP] ordering is not arbitrary: it maps communication demands to hardware bandwidth tiers (NVLink → InfiniBand → once-per-step).

03 Co-Design Models and Parallelism

Llama 3's 126 layers (not 128) is a direct result of PP load balancing. The 4:1 cross-/self-attention ratio in multimodal came from PP workload constraints.

04 All-Gather CP Beats Ring for Irregular Masks

Document masking makes ring-style attention impractical. All-gather over the smaller K,V tensors (GQA) achieves 95%+ HFU at 128K, outperforming ring at CP=4.

05 Debugging is a First-Class Problem

Top-down rank analysis and FP32 gradient accumulation were critical for production stability. The first failing rank is rarely the actual cause.

06 Real Production story

Not a simulation.

Recommendations for future hardware

01

More HBM capacity

Higher HBM lets you reduce TP from 8⟶4, cutting comms ~50%. They saw +10% end-to-end speedup at 2K-GPU scale.

02

Compute efficiency at small shapes

PP shrinks batch; CP shrinks sequence. GEMMs become small. Hardware can't only optimize for huge shapes.

03

Stronger CPUs

GPU gen-over-gen gains outpace CPU. At small kernels, host-side launch overhead becomes the bottleneck.

04

Deterministic DVFS

Synchronous parallelism = whole cluster runs at speed of slowest GPU. Non-deterministic frequency scaling kills throughput.

05

Hierarchical, oversubscribed networks

Don't pay for full bisection BW at every level. Match BW to actual parallelism comm patterns.

06

Perf-per-Watt over peak Perf

100K-GPU clusters are power-limited, not GPU-limited. The metric that matters is FLOPs per joule.

References

[1] Chu, W., Xie, X., Yu, J., Wang, J., et al. "Scaling Llama 3 Training with Efficient Parallelism Strategies." ISCA '25: Proc. 52nd Annual International Symposium on Computer Architecture, 2025.

[2] Narayanan, D., Shoeybi, M., Casper, J., et al. (NVIDIA, Stanford, Microsoft) "Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM." SC '21, 2021.

[3] Ding, Y. "CSE 291 LLM System Optimization — Lecture 5: Distributed LLM Training: Data Parallelism (ZeRO 1/2/3)." UC San Diego, 2026.

[4] Ding, Y. "CSE 291 LLM System Optimization — Lecture 6: Distributed Training: Pipeline Parallelism & Tensor Parallelism." UC San Diego, 2026.

[5] Ding, Y. "CSE 291 LLM System Optimization — Lecture 7: Distributed Training: Context Parallelism." UC San Diego, 2026.