

OCTOBER 21<sup>ST</sup>, 2025

# A Deep Dive into AI at Scale - Part 2

VÄINÖ HATANPÄÄ

Assistant Computer Scientist  
ALCF

# Recap: Data Parallelism

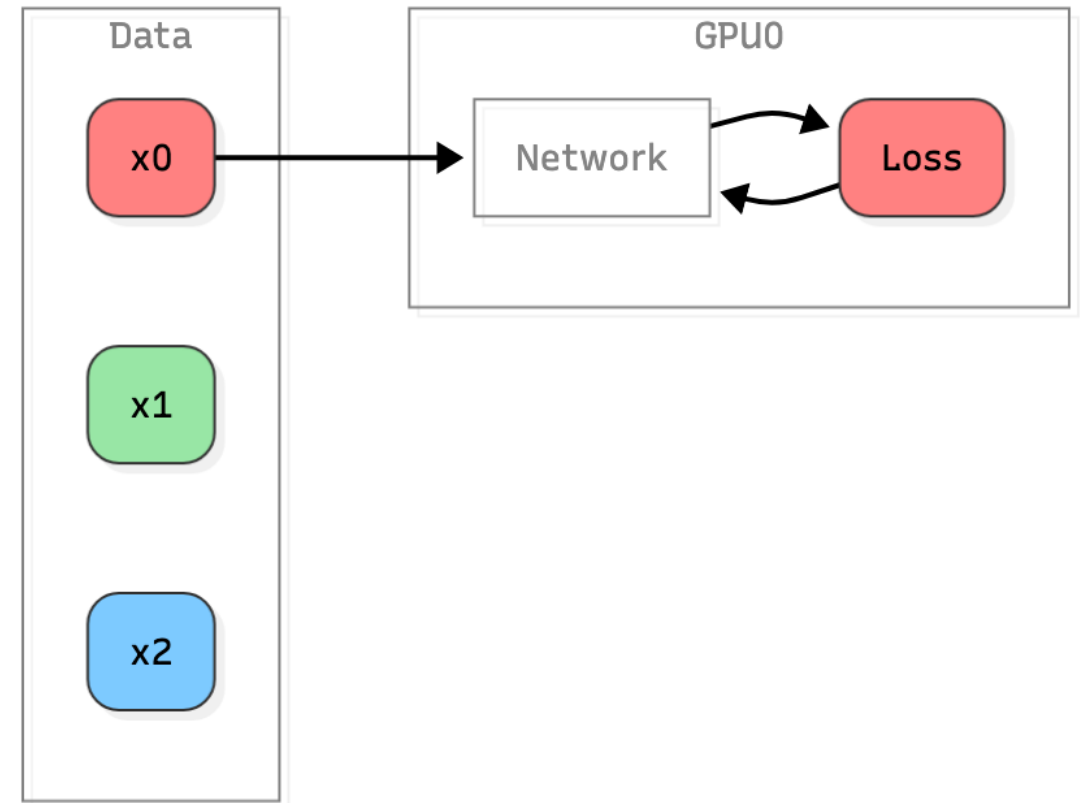
- Each GPU has an identical copy of the model
- Each GPU has unique data sample

Why?

- Training on a single GPU can be slow

Drawbacks:

- More complex to implement
- Requires communication



Diagrams by Sam Foreman

# Recap: Data Parallelism

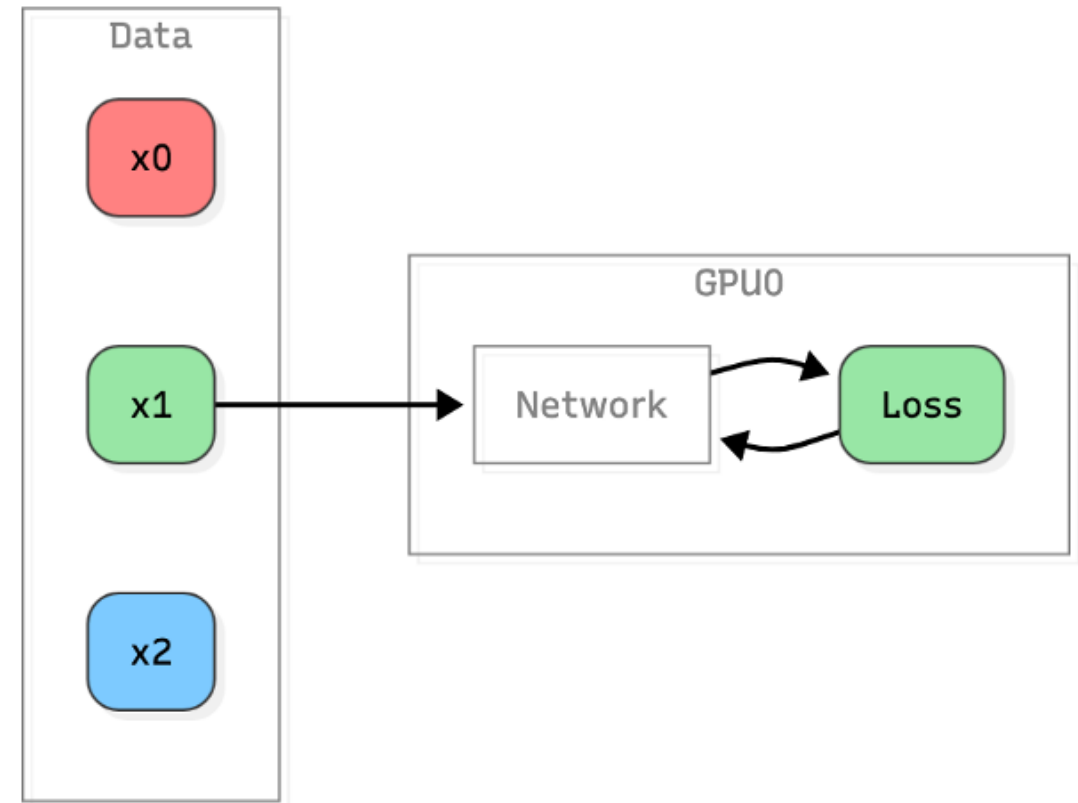
- Each GPU has an identical copy of the model
- Each GPU has unique data sample

Why?

- Training on a single GPU can be slow

Drawbacks:

- More complex to implement
- Requires communication



Diagrams by Sam Foreman

# Recap: Data Parallelism

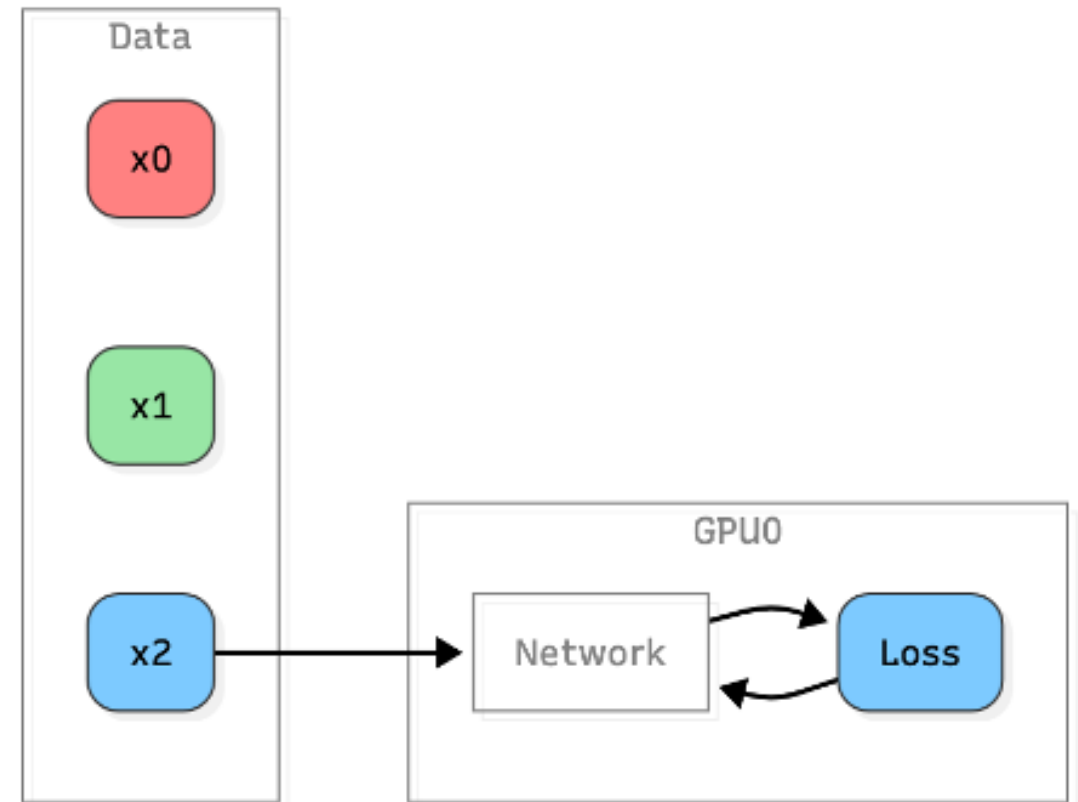
- Each GPU has an identical copy of the model
- Each GPU has unique data sample

Why?

- Training on a single GPU can be slow

Drawbacks:

- More complex to implement
- Requires communication



Diagrams by Sam Foreman

# Recap: Data Parallelism

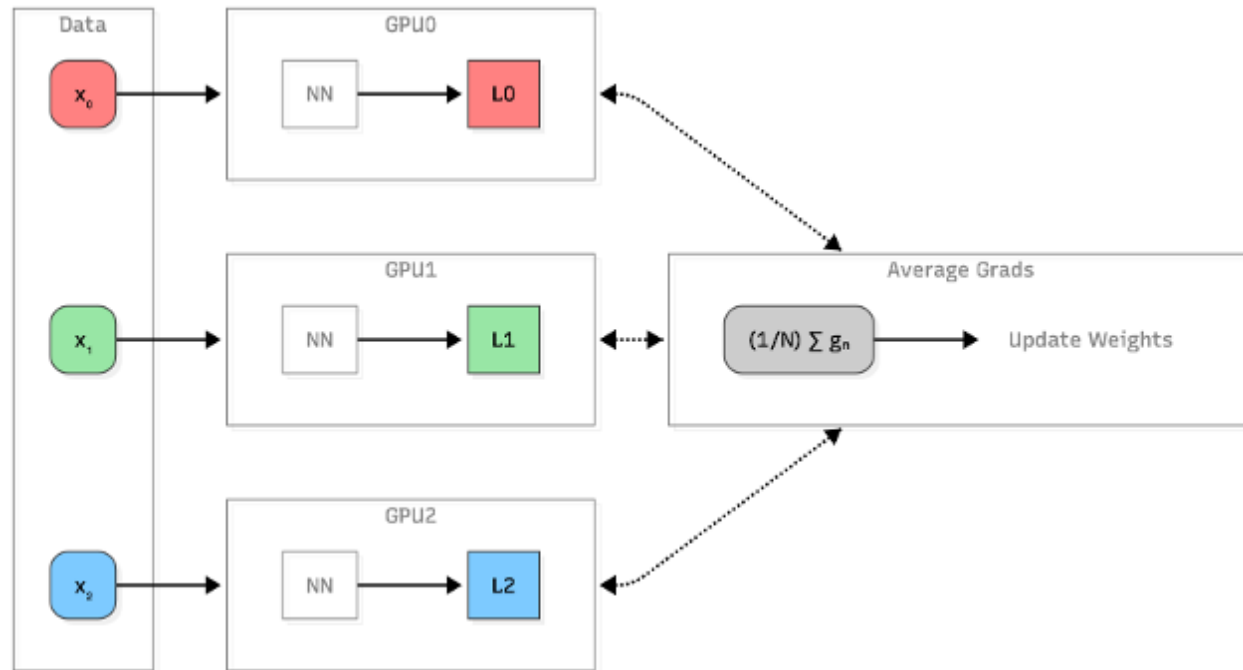
- Each GPU has an identical copy of the model
- Each GPU has unique data sample

Why?

- Training on a single GPU can be slow

Drawbacks:

- More complex to implement
- Requires communication



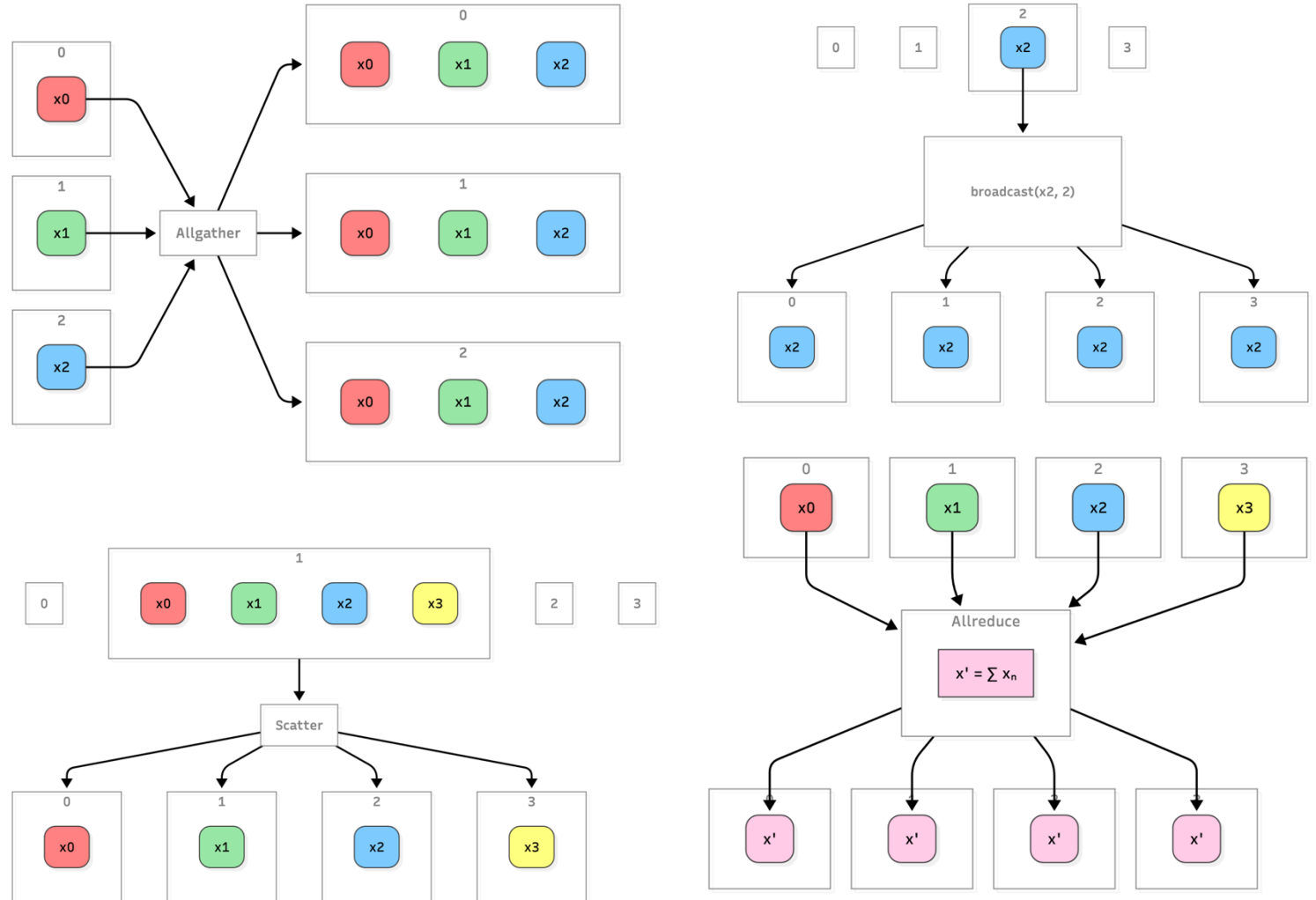
Diagrams by Sam Foreman

# Beyond data parallelism

- Data parallelism is very helpful when the model and the data fits into the memory of a single GPU and can be trained reasonably fast
- In the case of large language models (LLMs), the model sizes can be billions of parameters, requiring significant amount of memory
- Model parallelism can help!

# Collectives

- GPUs can communicate with each other with various **collective** operations
- Implemented with a collective communication library such as Nvidia Collective Communications Library (NCCL)

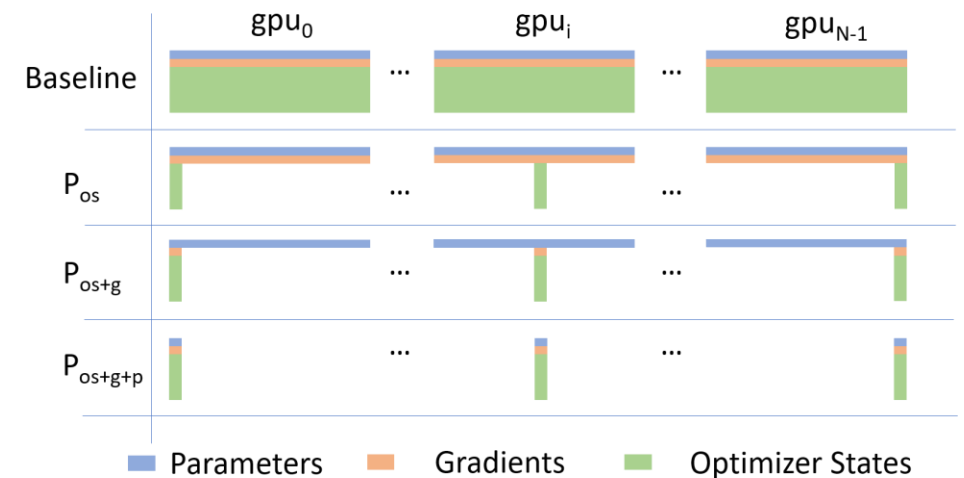


Diagrams by Sam Foreman



# FSDP and ZeRO

- Memory required also for gradients and optimizer states (e.g. momentum, variance)
  - Parameters (BF16): 2 bytes per parameter
  - Gradients (BF16): 2 bytes per parameter
  - Optimizer states (FP32): 3x4=12 bytes per parameter
  - Total of  $\geq 16$  bytes per parameter, so at least 112GB for 7B parameter model, more than most GPUs

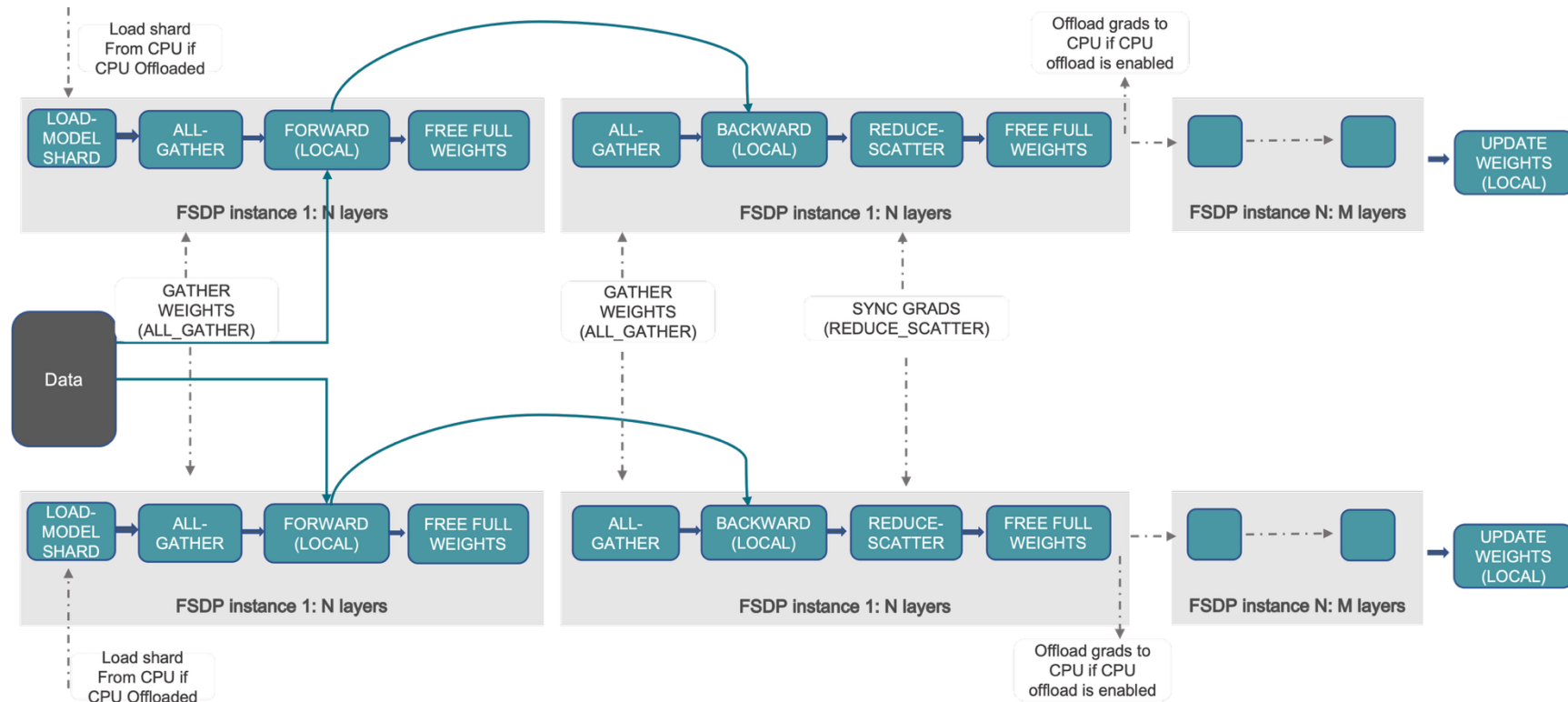


[DeepSpeed ZeRO](#)



# FSDP and ZeRO

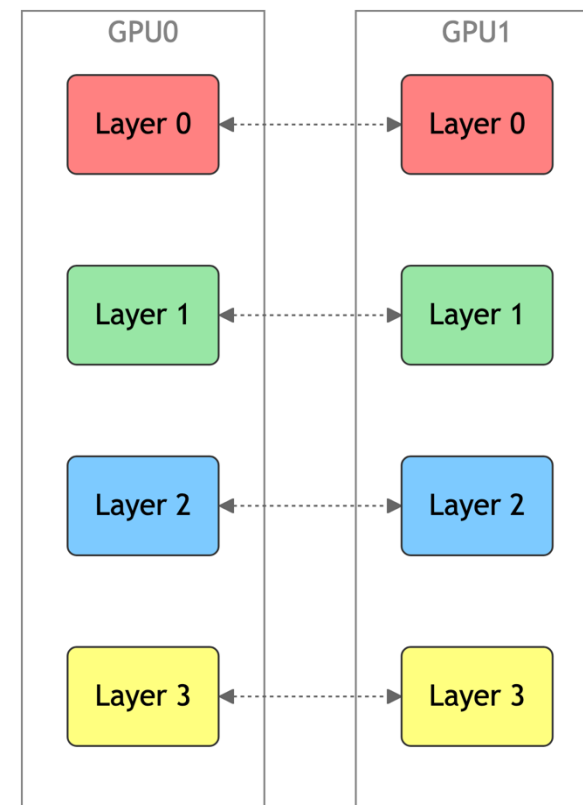
- FSDP distributes parameters, gradients, and optimizer states across data-parallel workers
  - Various settings can be configured and the exact behavior can be adjusted




PyTorch FSDP

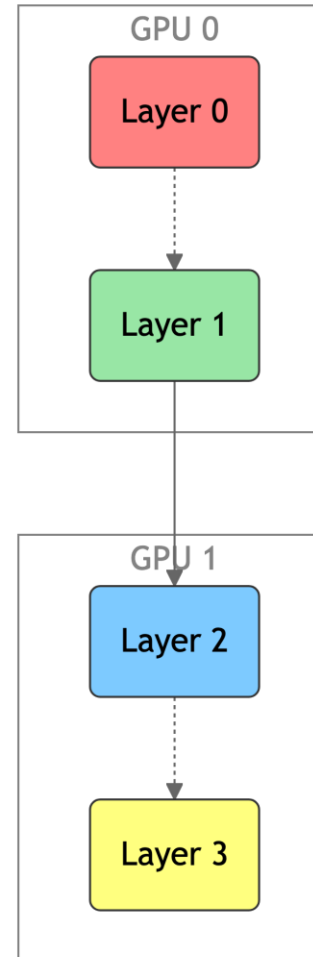
# Tensor Parallelism (TP)

- Each tensor is split up into multiple chunks
- Each shard of the tensor resides on its designated GPU
- During processing each shard gets processed separately (and in parallel) on different GPUs
  - synced at the end of the step
- Challenges:
  - Implementation
  - Communication overhead
  - Reduced compute kernel efficiency
- See: 😊 [Model Parallelism](#) for additional details



# Pipeline Parallelism

- Model is split up **vertically** (layer-level) across multiple GPUs
- Each GPU:
  - has a portion of the full model
  - processes *in parallel* different stages of the pipeline (on a small chunk of the batch)
- See:
  -  [PyTorch / Pipeline Parallelism](#)
  - [DeepSpeed / Pipeline Parallelism](#)



Challenge:  
Scheduling and  
“Pipeline Bubble”



1F1B schedule

# Sequence/Context/Expert parallelism

Model architecture and application specific forms of parallelism

- Sequence parallelism (SP):  
An extension of tensor parallelism (typically for LLMs), where the input sequence is scattered and gathered for specific operations
- Context parallelism:  
Like SP but shards the sequence across all layers, including the attention, implemented with *ring attention*
- Expert parallelism:  
Specific for Mixture of Experts (MoE) models, where different “experts” are placed on different GPUs

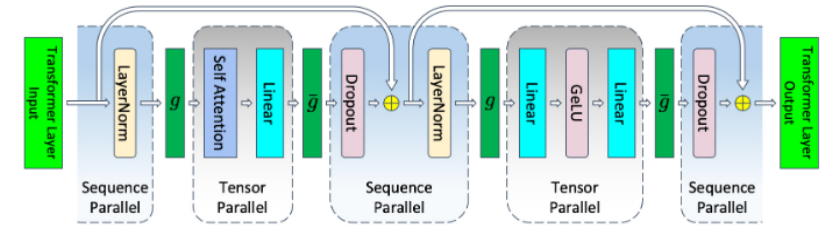
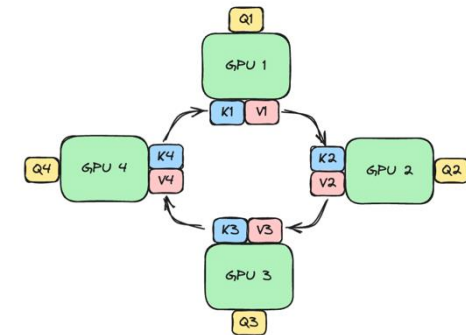


Figure 5: Transformer layer with tensor and sequence parallelism.  $g$  and  $\bar{g}$  are conjugate.  $g$  is all-gather in the forward pass and reduce-scatter in the backward pass.  $\bar{g}$  is reduce-scatter in forward pass and all-gather in backward pass.

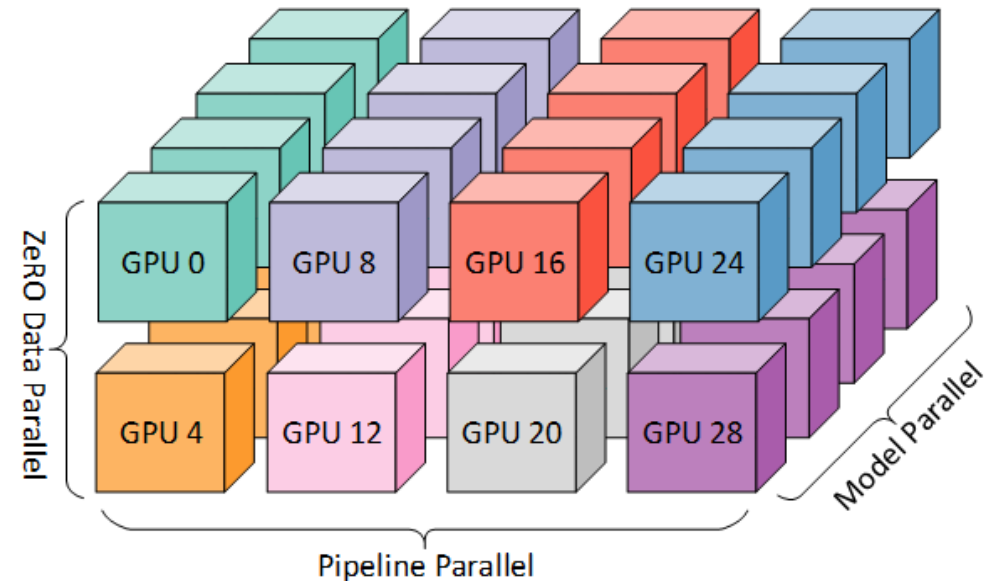
<https://arxiv.org/pdf/2205.05198>



Ring Attention

# 3D parallelism

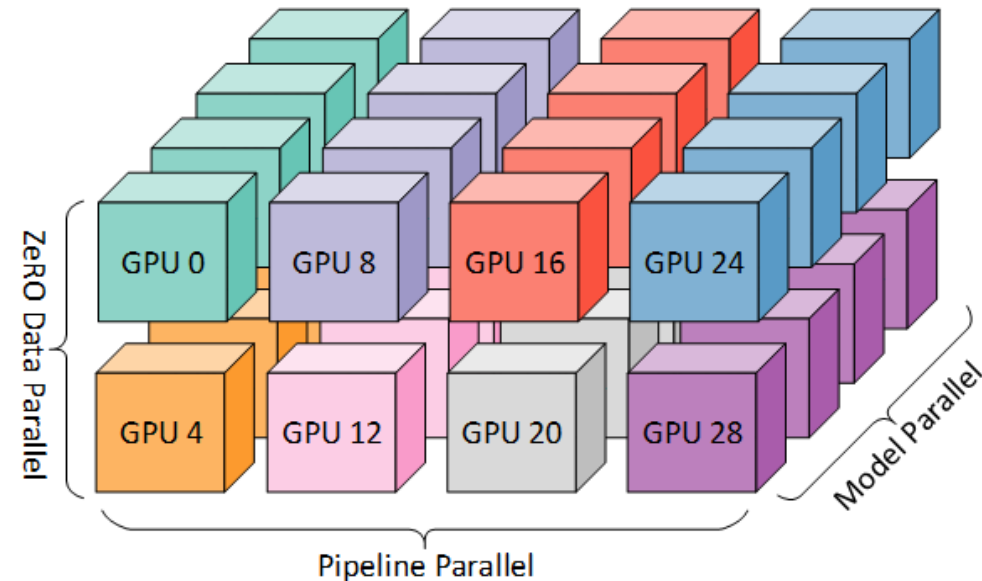
- Combining multiple "orthogonal" forms of the parallelisms
- On the right:
  - Height dimension for data parallelism
  - Width dimension for pipeline parallelism
  - Depth for model parallelism
- Typical to place tensor parallelism ranks close to each other as the communication is frequent and bandwidth intensive



[DeepSpeed 3D-parallelism](#)

# Deciding on parallelism strategy

- Considerations:
  - How many GPUs are available
  - How large is the model
  - How fast is the network connectivity between the GPUs in a node or between nodes
  - What kind of implementations are available
- For example:
  - Simple model, fits into a single GPU -> DP
  - Large LLM -> A lot of GPUs and parallelism required



[DeepSpeed 3D-parallelism](#)

# Hands on and homework

- The hands-on part is an FSDP+TP example where the default configuration does not fit into the memory of a single GPU
- Try to see if enabling tp helps (--tp=4)
- Experiment with the performance impact of TP, by reducing the number of layers (--n-layers=8), and note iteration times for TP of 1,2,4

```
module use /soft/modulefiles
module load conda/2025-09-25
conda activate base
export PATH="/opt/pbs/bin:${PATH}"
export HF_HOME=./.cache
ezpz-launch python3 -m ezpz.examples.fsdp_tp --dataset random
```





**VÄINÖ HATANPÄÄ**  
Assistant Computer Scientist  
ALCF



Argonne National Laboratory is a  
U.S. Department of Energy laboratory  
managed by UChicago Argonne, LLC.



**Argonne Leadership  
Computing Facility**