CSC290/420 Machine Learning Systems for Efficient AI
Training - II

Sreepathi Pai

October 29, 2025

URCS

Outline

A Zoo of Parallelism Techniques

Fault Tolerance

Trying out Training

Outline

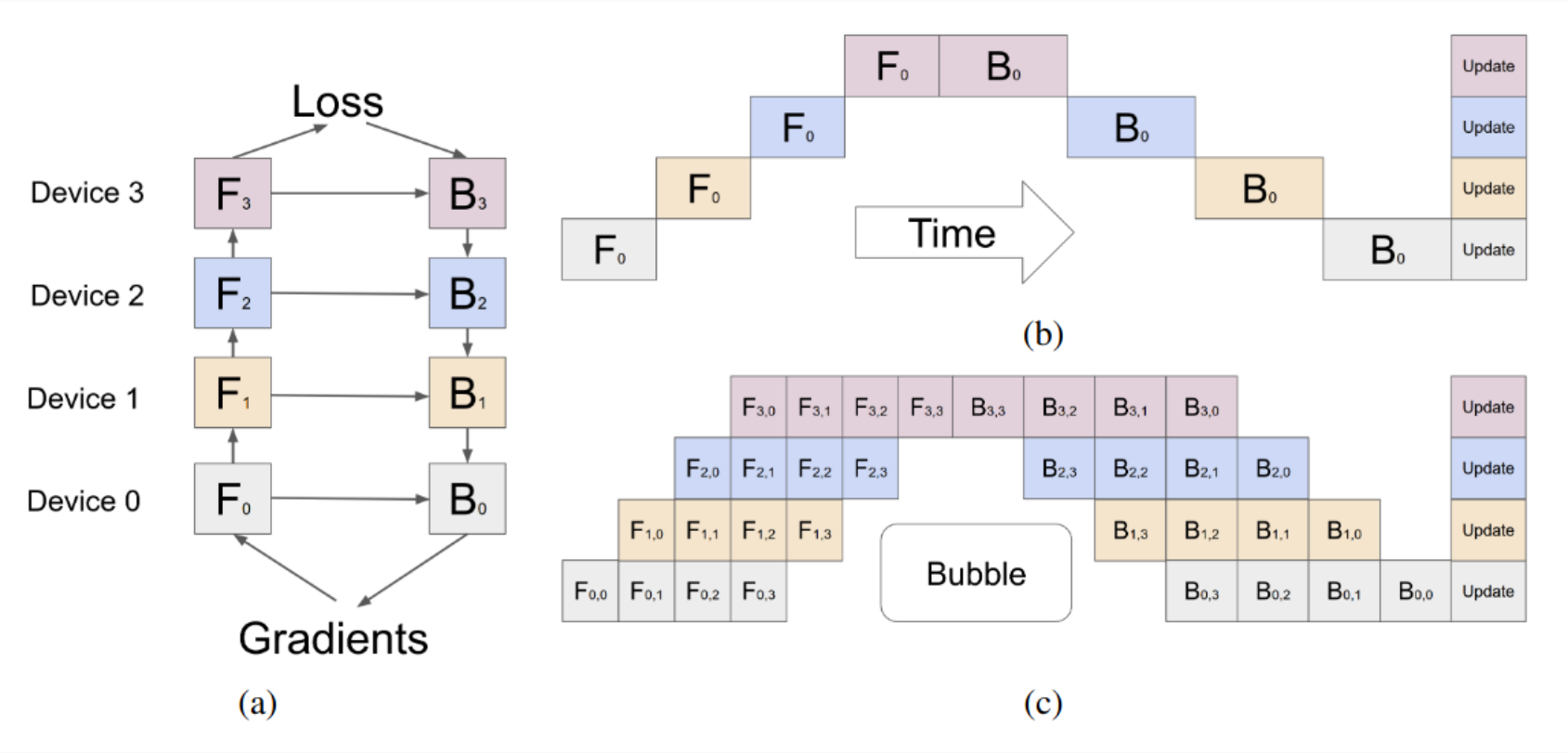
Data Parallelism in Training

- Replicate model to multiple devices
- Partition inputs across multiple devices
- Training:
  - Gather gradients across all devices
  - Average them (Reduce) and update parameters
- Key constraint: Model + Parameters + Gradients + other data must fit in one device

Pipeline Parallelism in Training

- Partition model across multiple devices
- Transmit data (activations) forwards across these devices
  - And gradients backwards across these devices

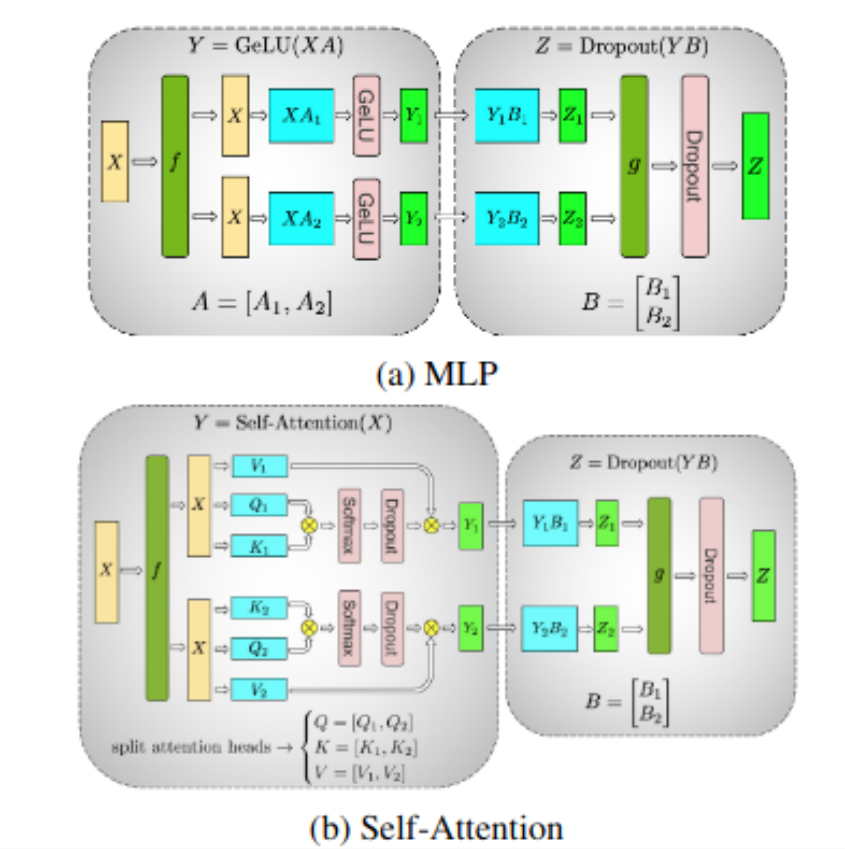
Huang et al., GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism, NIPS 2019



Model Parallelism in Training

- "Intra-layer" Model Parallelism
  - now probably termed tensor parallelism
- Breaks up tensors across multiple devices in the transformer layer

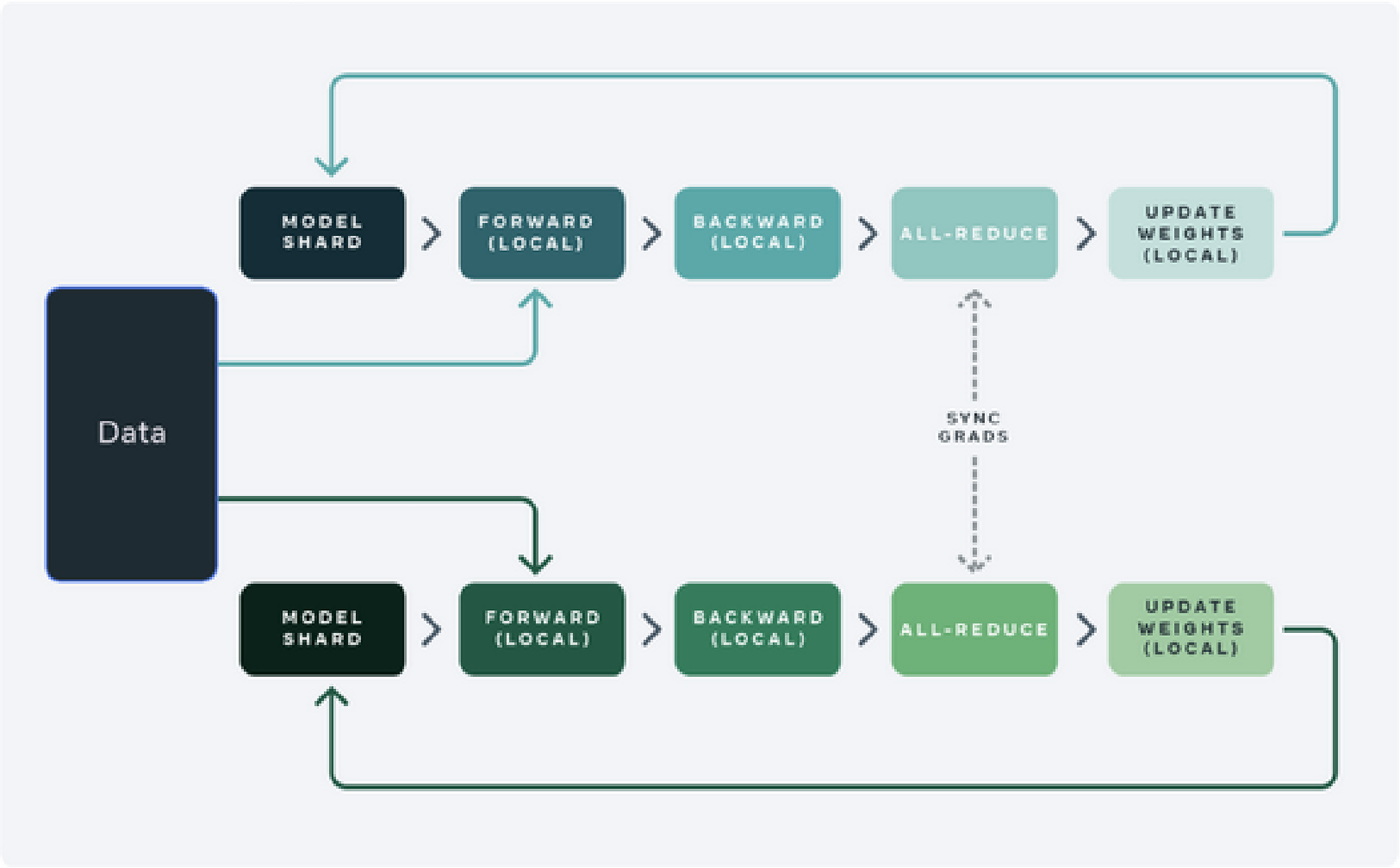
Shoeybi et al., Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism, 2020



Distributed Data Parallel (2019)

- Data Parallel, but over different machines
  - Can be combined with Model Parallel (in PyTorch)

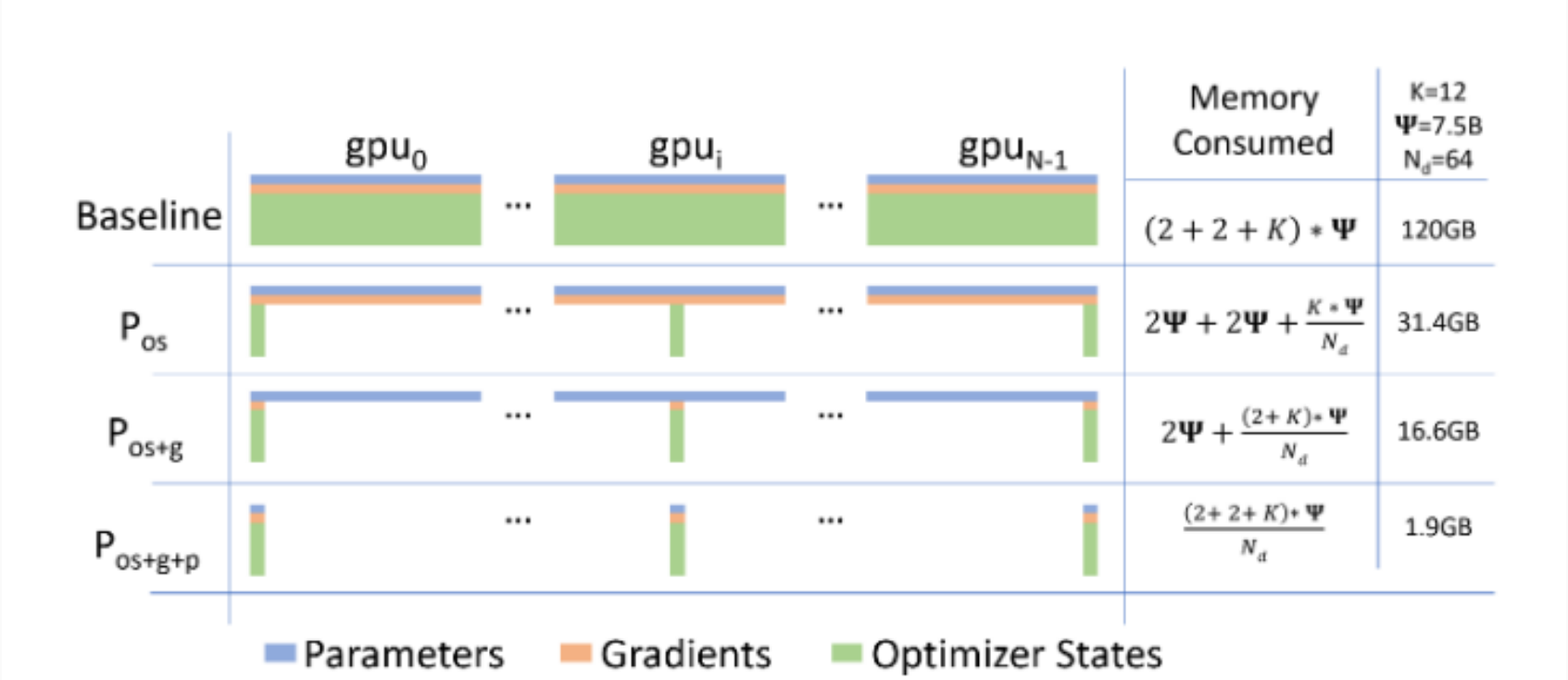
Fully Sharded Data Parallel: faster AI training with fewer GPUs



ZeRO (2020)

- ZeRO asks: where does memory go?
  - Model state: optimizers, gradients, parameters
  - Residual state: activations, etc.
- Do we need full copies of model state on each device?
  - at all times?
  - can throw away other devices portions and rebuild again
  - can also offload to CPU

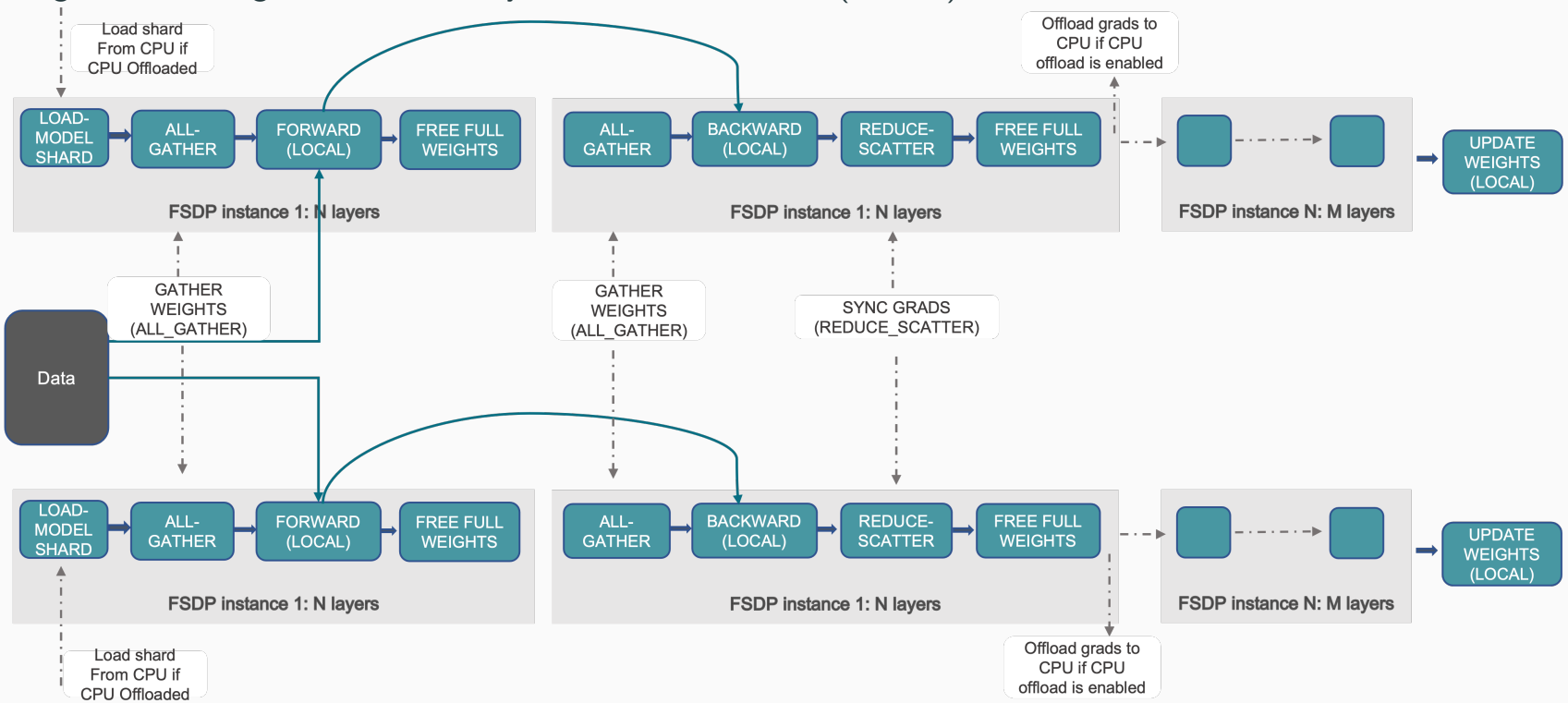
Rajbhandari et al., ZeRO: Memory Optimizations Toward Training Trillion Parameter Models



Fully Sharded Data Parallel

- Shard model state and activations
- Gather everything needed for a layer
- Run layer
- Throw everything away

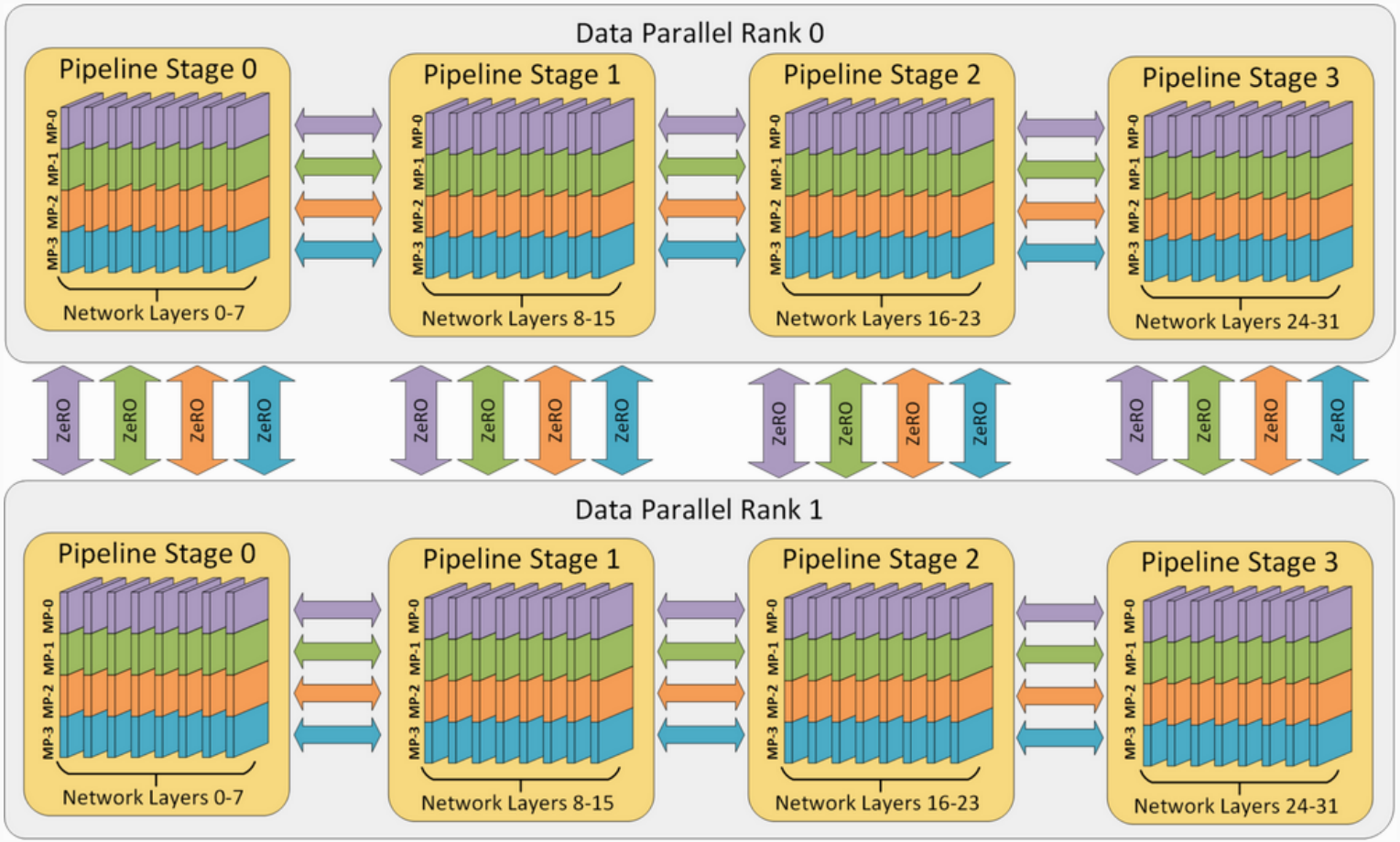
Feng et al., Getting Started with Fully Sharded Data Parallel (FSDP2)



3-D Parallelism

- Pipeline Parallelism + Data Parallelism + Model Parallelism

DeepSpeed.ai, Pipeline Parallelism



Other forms of parallelism

- Sequence Parallelism
  - a form of Tensor Parallelism
- Expert Parallelism
  - a form of Model parallelism, with each Expert on a separate device
- ...

The Next 700 ML Parallelization schemes: Building Blocks

- Partitioning (Sharding)
- Copies (Replication)
- Loading and Unloading
- The first two operations applied to:
  - Operators / Layers
  - Data (Parameters, Optimizer State, Gradients, Activations, etc.)
- Load/Unload are applied to Data based on Liveness
- Communication introduced to:
  - Combine Partitions (Many to One, Many to Many)
  - Transmit Data (One to One)
- Subject to data dependence constraints
  - But can be relaxed to use stale data

Things to watch out for

- Communication volume
  - Can turn compute bound to I/O bound
- Computation intensity
  - In the extreme case, GPUs are underutilized because the computation has become very small
- Look at the timeline of operations and watch for
  - underutilization, idle time
  - excessive synchronization
  - lack of communication/computation overlap

Outline

A Taxonomy of Failures

Failure Symptoms	Failure Domain			Likely Failure Cause
	User Program	System Software	Hardware Infra	
OOM	✓	✗	✗	User Bug
GPU Unavailable	✗	✓	✓	PCIe error, Driver/BIOS, thermals
GPU Memory Errors	✗	✗	✓	Thermal Noise, Cosmic Rays, HBM Defect or Wear
GPU Driver/Firmware Error	✗	✓	✗	Outdated Software, High Load
GPU NVLink Error	✗	✗	✓	Electro/Material Failure, Switch
Infiniband Link	✗	✗	✓	Electro/Material Failure, Switch
Filesystem Mounts	✗	✓	✗	Failed Frontend Network, Drivers in D State, Storage Backend
Main Memory Errors	✗	✗	✓	Circuit Wear, Thermal Noise, Cosmic Rays
Ethlink Errors	✗	✗	✓	Electro/Material Failure, Switch
PCIe Errors	✗	✗	✓	GPU Failure, Poor Electrical Contacts
NCCL Timeout	✓	✓	✓	Userspace Crash, Deadlock, Failed HW
System Services	✓	✓	✓	Userspace Interference, Software Bugs, Network Partition

Kokolis et al., Revisiting Reliability in Large-Scale Machine Learning Research Clusters

What can we do about failures?

- Mean Time to Failure
  - Decreases as number of components increase
- An estimate could help, for example, checkpoint
- Or choose appropriate architecture to maximize *goodput*
  - goodput is useful throughput

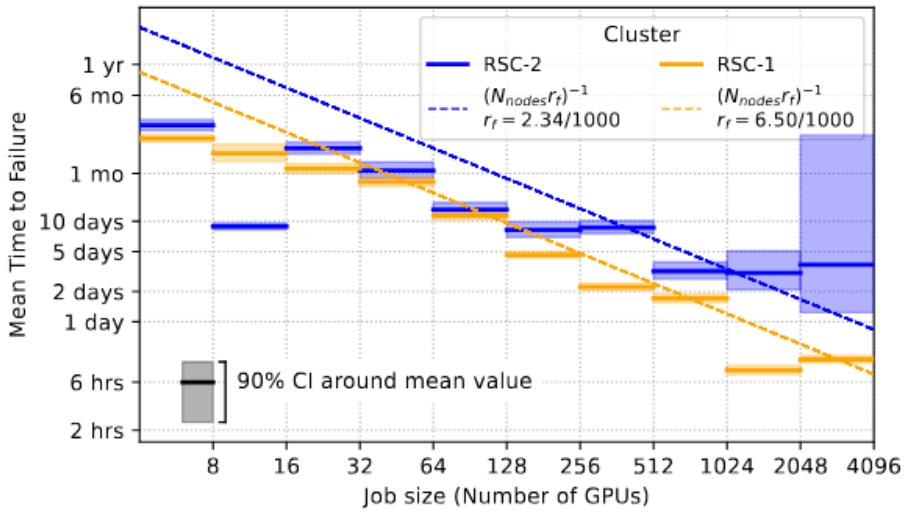


Fig. 7: MTTF analysis by job sizes for `RSC-1` and `RSC-2`, rounded up to the next multiple of 8 GPUs. **CI**: Confidence Interval. MTTF decreases predictably with scale.

Outline

TimeCapsuleLLM

TimeCapsuleLLM

Trained on 1800s text from Project Gutenberg.

nanoGPT

nanoGPT