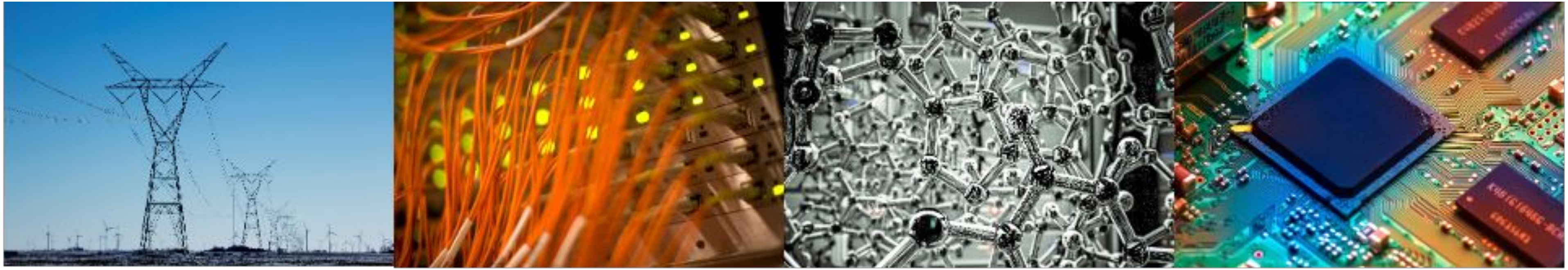
ZeRO++: Extremely Efficient Collective Communication for Giant Model Training

G. Wang et al., arXiv'23

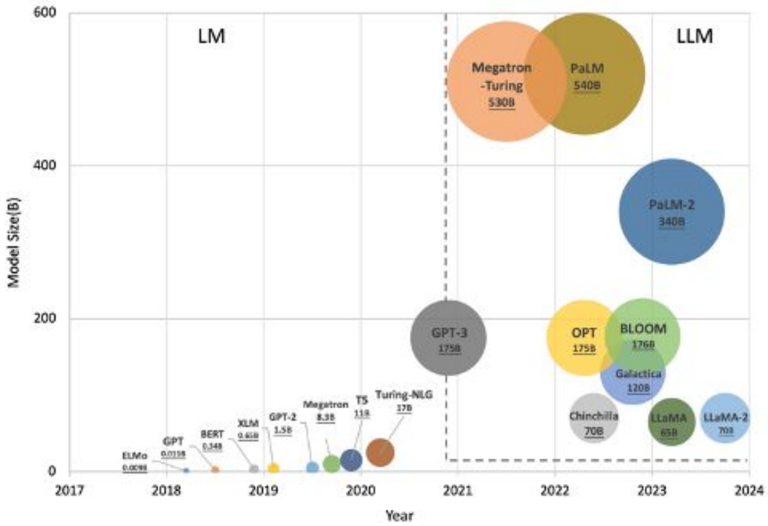


Presenters: Noelle Crawford, Hyungyo Kim

ILLINOIS
Electrical & Computer Engineering
COLLEGE OF ENGINEERING

Distributed Training of Large Models

Distributed training is now a default due to the large model and training data size



[K. He et al., Proceedings of IEEE'21]

3D Parallelism

- ❑ data + pipeline + tensor parallelism
- ❑ achieves excellent compute/memory efficiency
- ❑ But, requires code refactoring

ZeRO

- ❑ No model code refactoring
- ❑ Good throughput scalability

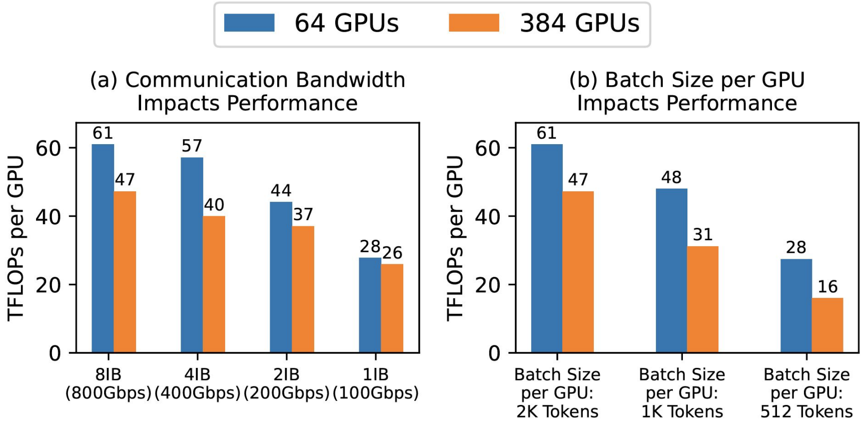
(Recap) ZeRO Stage 3

Divides model parameters, gradients, and optimizer states across all GPUs

- Close to linear reduction in memory footprint

Communication can be the bottleneck for

- (case 1) systems with slow interconnects
- (case 2) small per-GPU batch size
    - Communication cost is amortized across tokens in a single GPUs batch
    - Global batch size limited due to convergence



[Interconnect BW, batch size vs Throughput]

ZeRO Communication Overhead

Communication requirements

- Forward All-Gather: Model Parameters (M)
- Backward All-Gather: Model Parameters (M)
- Backward Reduce-Scatter: Gradients (M)

→ Total communication volume: 3M

- Occurs both on fast intra-node interconnect and slow inter-node interconnect
- Bottlenecked by inter-node communication

ZeRO++: 3M → 0.75M Comm. Cost Reduction

Prior Works on Communication Reduction

Quantization

- If done naively, can severely impact model accuracy
- Advanced techniques: outlier filtering, block-based quantization

ZeRO-3 Optimizations

- Trades on-device memory *when available* for communication (MiCS)
- Replicate model states across sub-groups
- Similar to hpZ (coming soon) but with more replication

Gradient Compression

- Extreme gradient compression (1-bit ADAM, LAMB) assume each GPU has full optimizer states
- Not directly applicable to ZeRO-3

ZeRO++ Design Overview

Quantization → **qwZ**: Block quantization of weights

- If done naively, can severely impact model accuracy
- Advanced techniques: outlier filtering, block-based quantization

ZeRO-3 Optimizations → **hpZ**: Hierarchical partitioning strategy

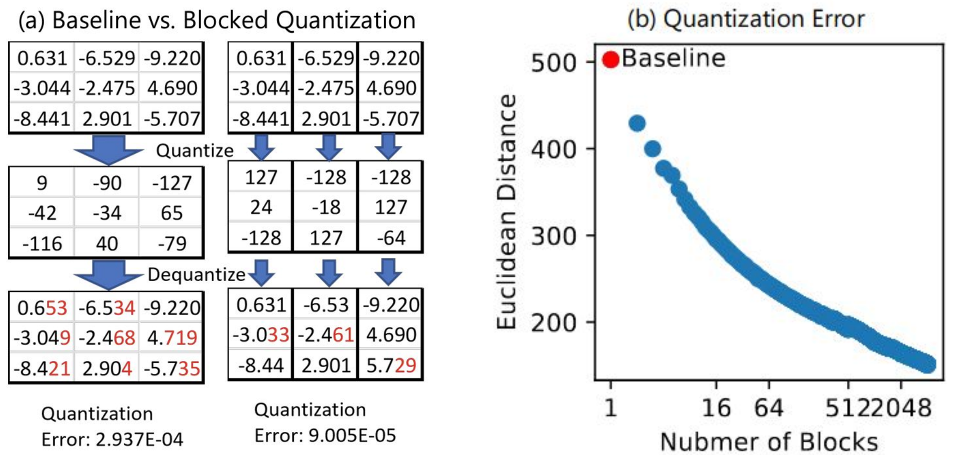
- Trades on-device memory *when available* for communication (MiCS)
- Replicate model states across sub-groups
- Similar to hpZ (coming soon) but with more replication

Gradient Compression → **qgZ**: Quantization of gradients

- Extreme gradient compression (1-bit ADAM, LAMB) assume each GPU has full optimizer states
- Not directly applicable to ZeRO-3

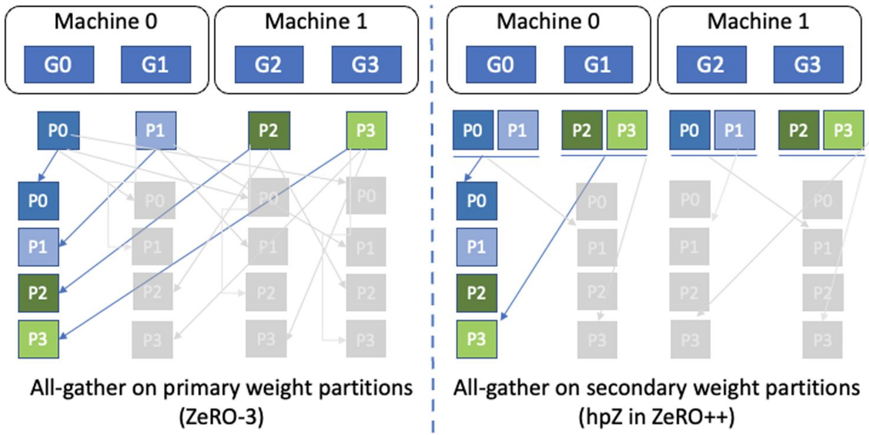
bonus: overlap strategy for compute/comm., custom CUDA kernels

qwZ: Block Quantization of Weights



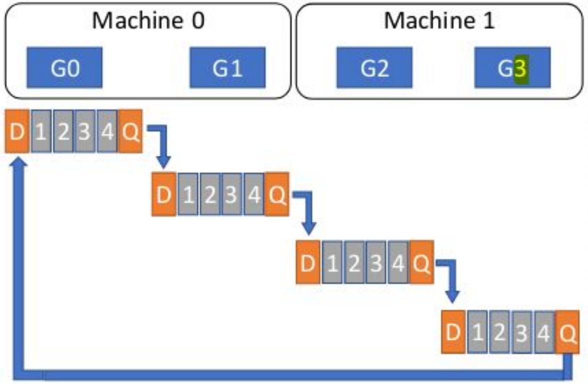
- Reduce weights from FP16 -> INT8
- Quantizing all weights together leads to large decreases in accuracy
- Block quantization introduces a tradeoff between better accuracy (smaller block size) and smaller overhead (larger block size)

hpZ: Hierarchical Partitioning Strategy

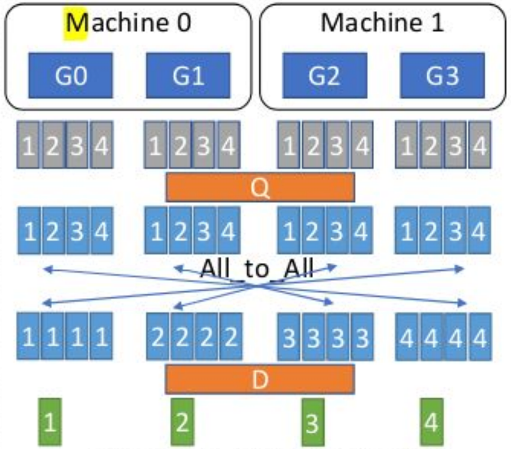


- For current size of models, no need to spread weights among 100s of GPUs
  - “Caching” weights w/in node allow for faster communication during backward pass
- Primary and secondary partitions
  - Primary: across all GPUs, Secondary: w/in node
- Forward pass: primary partition, Backward pass: secondary partition

qgZ: Quantization of Gradients



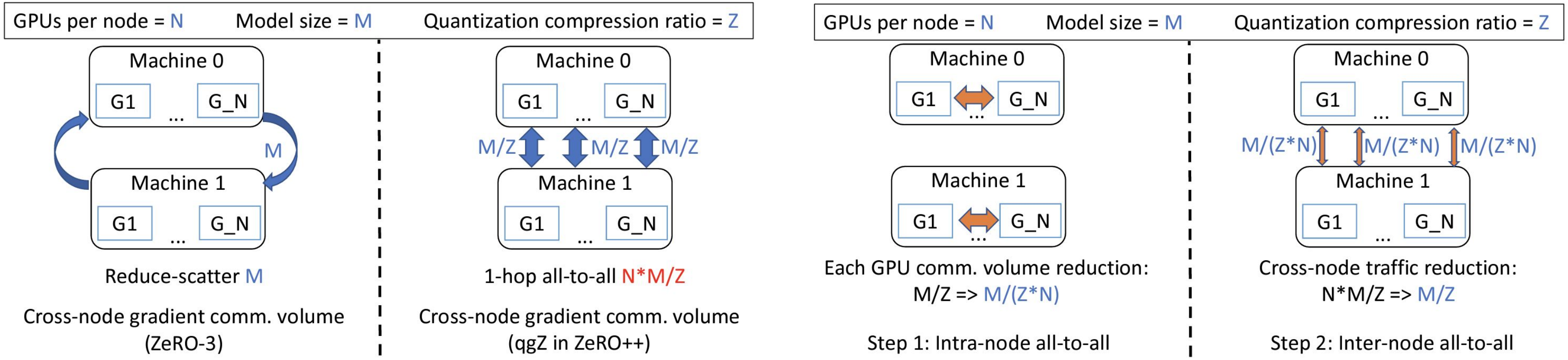
NCCL Ring-based reduce-scatter (ZeRO-3)
# of sequential Q+D == # of GPUs



1-hop all-to-all (qgZ in ZeRO++)
# of sequential Q+D == 1

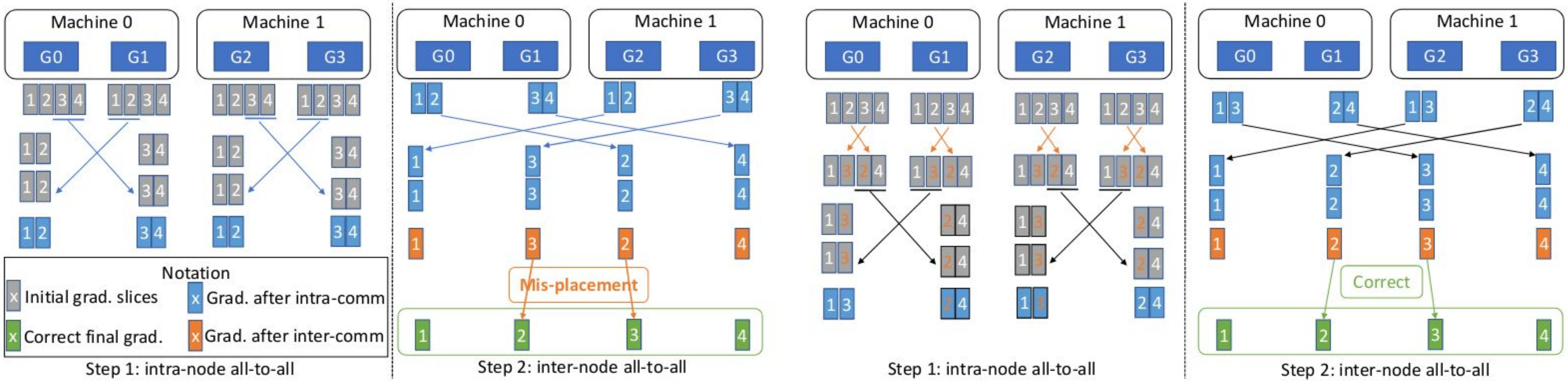
- Ring-based reduce-scatter
  - N repeats of quant./dequant.
  - high communication latency and low accuracy
- Proposed: use all-to-all reduce-scatter
  - Solves problem of repeated Q+D
  - New problem: increase in inter-node communication when using 1-hop

qgZ: Quantization of Gradients (cont.)



- Inter-node comms. volume of all-to-all increase with number of devices (M vs N*M/Z)
- Proposed: Hierarchical 2-hop all-to-all (M/Z)
  - 1st hop: intra-node, 2nd hop: inter-node

qgZ: Quantization of Gradients (cont.)

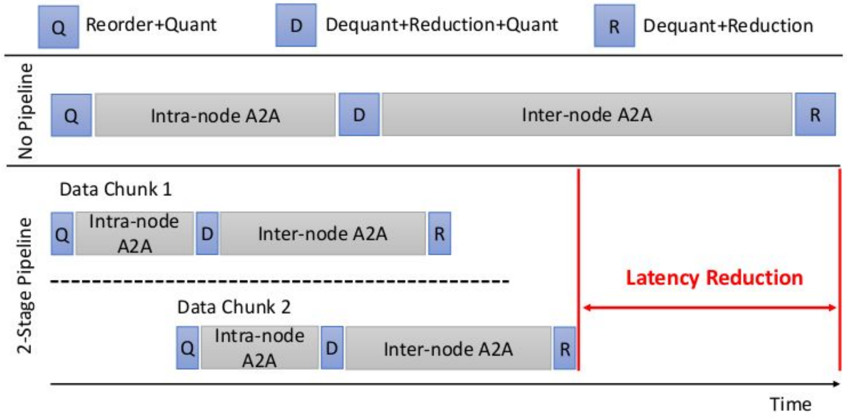


- Gradient misplacement issue
  - Some gradients do not end up on correct machines
- Solve via tensor slice re-ordering before transmission
  - [0, 1, 2, …YX-2, YX-1] -> [0, X, 2X, …(Y-1)X, 1, X+1, (Y-1)X+1, …YX-1]

Additional Optimizations

Overlapping Compute & Comm.

- Non-blocking asynchronous quant.
    - Synchronize quantization stream before used in compute stream
- Pipeline chunks of intra/inter node communication for qgZ



Custom CUDA Kernels

- Able to fully utilize BW due to quantization and good ILP
- Tune size of quantization blocks to minimize traffic with good accuracy
- Fuse tensor reshaping and quantization into a single kernel

ZeRO++ Cross-Node Communication Volume Analysis

Forward All-Gather

- Weights are quantized FP16 -> INT8 reducing comm. from M -> 0.5M

Backward All-Gather

- No cross-node traffic

Backward Reduce-Scatter

- Gradients are quantized FP16 -> INT4 reducing comm. from M -> 0.25M

In total, only 0.75M per training iteration (down from 3M)

Experimental Setup

Hardware

- 24 NVIDIA DGX-2 nodes (16 V100 SXM3 32 GB GPUs each)
- Nodes connected with infiniband, NVLink w/in nodes

Baseline

- ZeRO-3

Model Configurations

- GPT-style transformers
- (micro batch size) 2k tokens per GPU

Results–1: Scalability and Generalizability

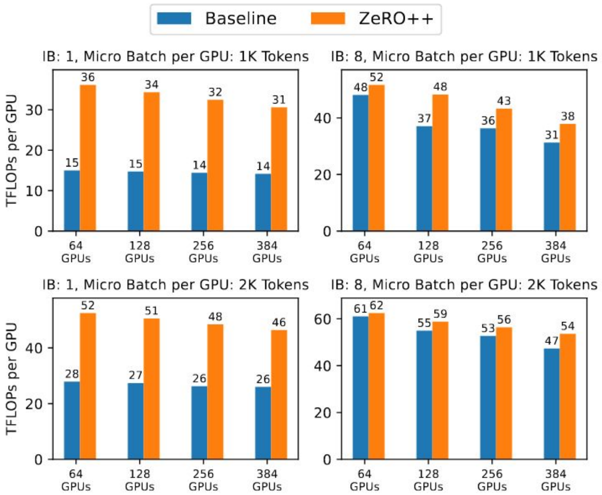


Table 2: End-to-end speedup of ZeRO++ on 384 GPUs with different model sizes

		1 IB Connection			8 IB Connections		
Model Size	Tokens per GPU	Baseline TFLOPs	ZeRO++ TFLOPs	Speedup	Baseline TFLOPs	ZeRO++ TFLOPs	Speedup
138B	2K	19.96	37.90	1.90x	47.55	55.30	1.16x
138B	1K	11.25	21.81	1.94x	34.19	44.38	1.30x
91B	2K	19.99	38.06	1.90x	47.74	56.26	1.18x
91B	1K	11.27	21.93	1.95x	34.49	44.36	1.29x
49B	2K	20.06	38.08	1.90x	48.05	56.24	1.17x
49B	1K	11.27	21.95	1.95x	34.54	44.46	1.29x
18B	2K	25.98	46.40	1.79x	47.31	53.65	1.13x
18B	1K	14.15	30.57	2.16x	31.27	37.87	1.21x

- Scales well with # of GPUs
- Higher improvement for lower inter-node BW clusters

- Consistent improvement across different model sizes

Results–2: Ablation Study

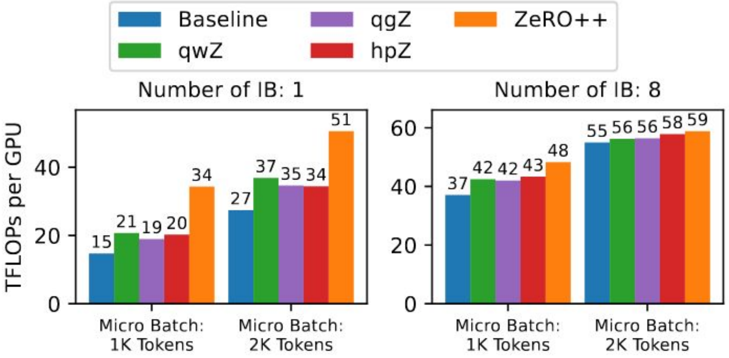


Table 3: End-to-end performance when using ZeRO++ w.\wo. optimized kernels

	Optimized Quantization Kernel	Optimized Fusion Kernel	TFLOPs
Baseline	N/A	N/A	15
ZeRO++	No	No	19.73
ZeRO++	No	Yes	21.6
ZeRO++	Yes	No	31.40
ZeRO++	Yes	Yes	**36.16**

- Each technique results in similar throughput improvements
- 1.3x for low-BW clusters, 1.15x for high-BW clusters

- Custom quant. kernel: 1.67x
- Kernel fusion: 1.15x

Results–3: Comparison to Previous Work

Table 4: hpZ vs MiCS evaluation on a 4 node cluster (16 V100 GPUs per node)

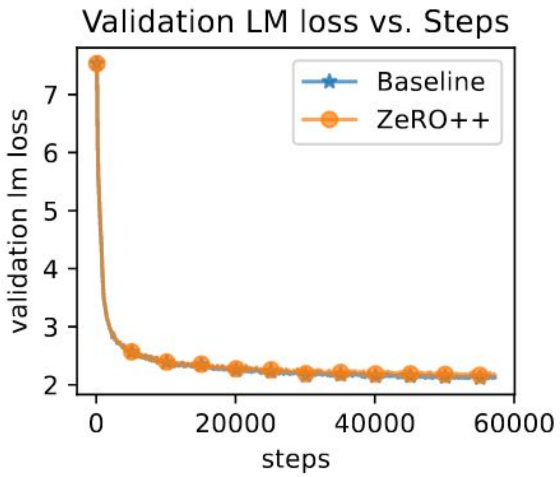
Model Size	Token Size	ZeRO TFLOPs	hpZ TFLOPs	MiCS TFLOPs
7.5B	1K	36.99	38.39	38.96
7.5B	2K	53.3	54.4	52.72
18B	1K	51.47	52.42	OOM
18B	2K	60.94	61.44	OOM

- Allows training of larger model than MiCS, which shards the optimizer states w/in nodes

Results–4: Convergence Validation

Table 5: Validation loss at the end of training (GPT 350M / 30B tokens)

	Evaluation LM loss
Baseline	2.121762
ZeRO++ (hpZ&qwZ&qgZ on)	2.165584
ZeRO++ (hpZ&qwZ on; qgZ on for first 50%)	2.134013
ZeRO++ (hpZ&qwZ on; qgZ off)	2.121653



- Achieves <1% LM loss within that of the baseline
- Very close convergence speed compared to that of the baseline

Our Thoughts

Strengths

- Real world E2E performance improvement with a huge engineering effort
- Practicality: easy to see how this could really be deployed in a datacenter

Weaknesses

- Novelty
  - Authors are simply mixing many pre-existing methods
- Explanation on design choices
  - Why quantize to INT8 / INT4 and not another bit-width? What's the block size for quantization?
- Validation for convergence is weak: evaluated on a small model

Future Directions

- As number of GPUs grows, all-to-all inter-node might become infeasible. How can this be adapted for other network topologies?
- Generalize and analyze the framework to different bit-width for quantization