

deepspeed

Model Scale

- 10+ Trillion parameters

Speed

- Fast & scalable training

Usability

- Few lines of code changes

Accelerated inference

- Up to 12x faster & cheaper

DeepSpeed and Trillion-parameter LLMs: Can synergy of ~~MPI~~ and NCCL improve scalability and efficiency?

MVAPICH

MUG '24










Ammar Ahmad Awan

(on behalf of several DeepSpeed team members at Microsoft and Snowflake)

<https://github.com/microsoft/DeepSpeed>

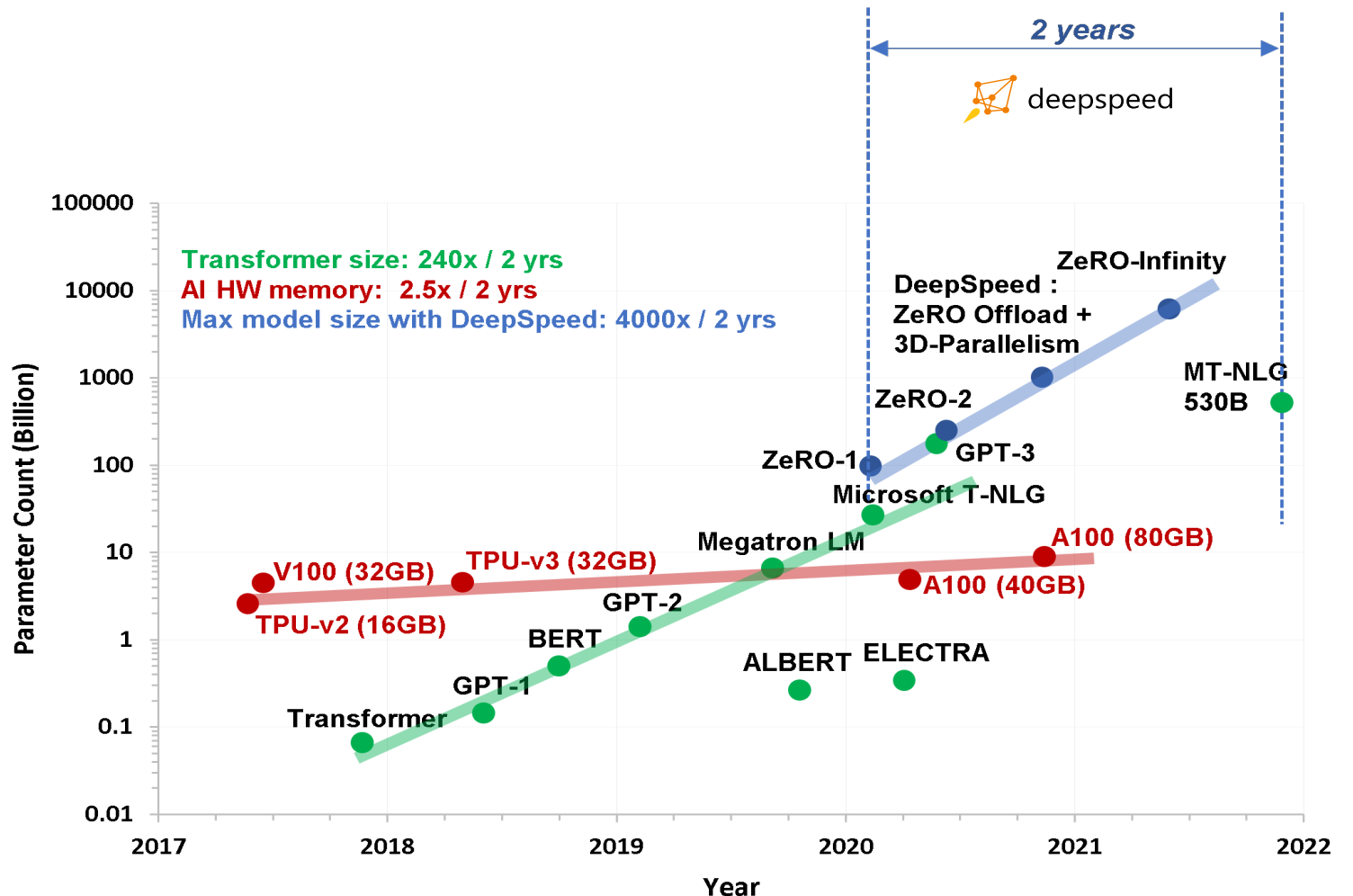
DeepSpeed Training: Reshaping the LLM Training Landscape

DeepSpeed Powered Massive Models:

- METRO-LM (5.4B) 
- Microsoft-Turing NLG (17B) 
- GPT Neo-X (20B) 
- AlexaTM (20B) 
- YaLM (100B) 
- GLM (130B) 
- BLOOM: Big Science (176B) 
- Jurassic-1 (178B) 
- Megatron-Turing NLG (530B) 
- ...

Key training technologies:

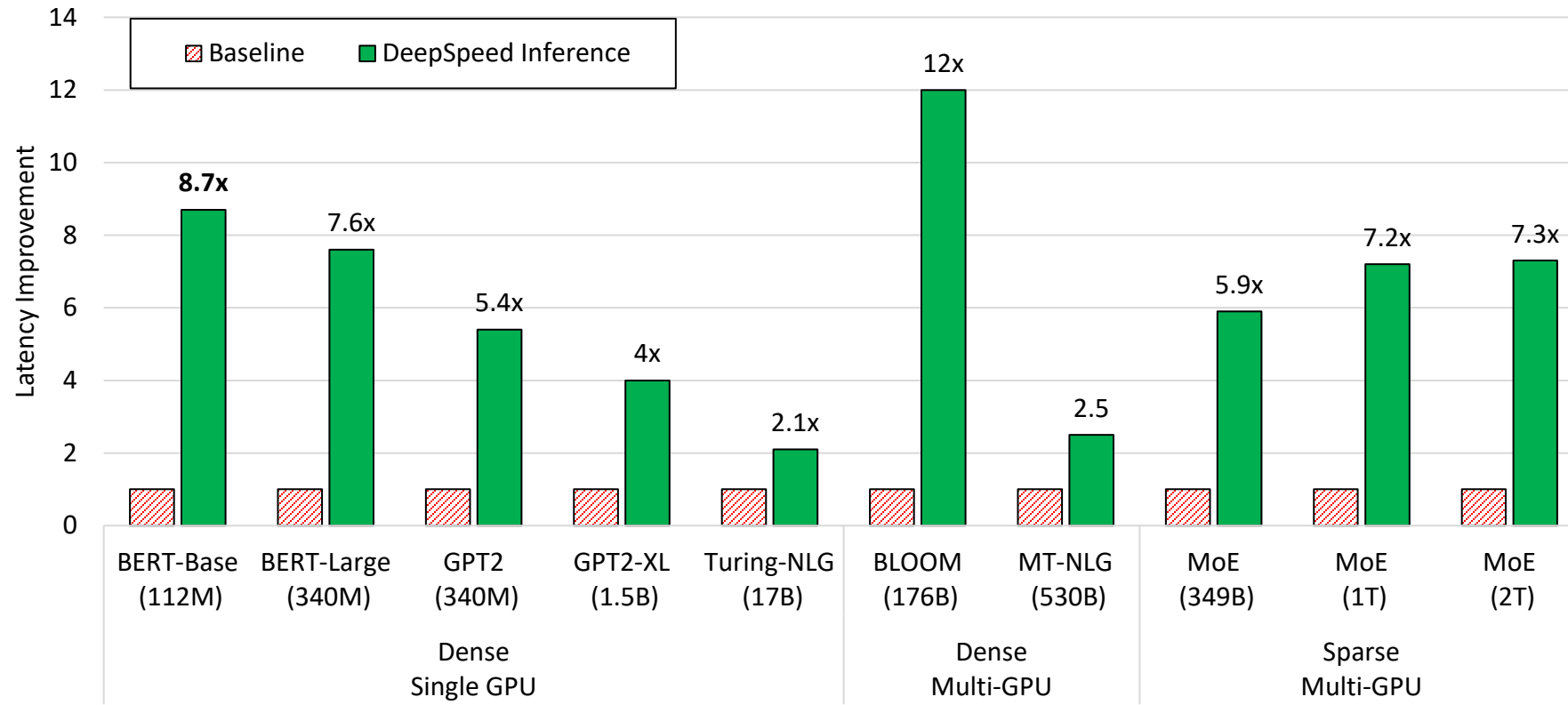
- Zero Redundancy Optimizer (ZeRO)
- ZeRO-Infinity
- 3D parallelism
- Memory and compute efficient MoE training
- Optimized CUDA/ROCm/CPU kernels
- Gradient compression 1-bit
Adam/LAMB, 0/1 Adam
- Sparse Attention
- Mixture of quantization
- Progressive layer dropping
- Curriculum learning
- ...



System capability to efficiently train models with *trillions of parameters*

DeepSpeed Inference: Accelerated serving for LLMs and SLMs

- ❑ Many-GPU Dense transformer optimizations – *powering large and very large models like Megatron-Turing 530B*
- ❑ Massive Scale Sparse Model Inference– *a trillion parameter MoE model inference under 25ms*
- ❑ ZeRO-Inference → *40x bigger model inference on single-GPU device*



DeepSpeed Inference: SoTA latency and throughput across the large model inference landscape

DeepSpeed OSS Impact

- 10x YoY growth of DeepSpeed usage
- 6 Million installs since release
- 356+ unique contributors
- 1.6k+ public packages have hard dependencies on DeepSpeed
 - Open-source frameworks
 - Hugging Face, PyTorch-Lightning, EleutherAI, MosaicML, etc.
 - External companies
 - Meta AI (FAIR), AstraZeneca, Fidelity, Salesforce, Intel, Bloomberg, Tencent, SAP, etc.
 - National Labs
 - Oak Ridge, Argonne, Lawrence Livermore, etc.



DeepSpeed is simple to use and extend!

```
# construct torch.nn.Module
model = MyModel()

# wrap w. DeepSpeed engine
engine, *_ = deepspeed.initialize(
    model=model,
    config=ds_config)

# training-loop w.r.t. engine
for batch in data_loader:
    loss = engine(batch)
    engine.backward(loss)
    engine.step()
```

```
ds_config = {
    "optimizer": {
        "type": "Adam",
        "params": {"lr": 0.001}
    },
    "zero": {
        "stage": 3,
        "offload_optimizer": {
            "device": "[cpu|nvme]"
        },
        "offload_param": {
            "device": "[cpu|nvme]"
        }
    }
}
```

+

•

○

MCR-DL

Quentin Anthony, Ammar Awan, Jeff Rasley, Yuxiong He, Aamir Shafi, Mustafa Abduljabbar, Hari Subramoni, and Dhabaleswar K. (DK) Panda - [IPDPS '23](#).

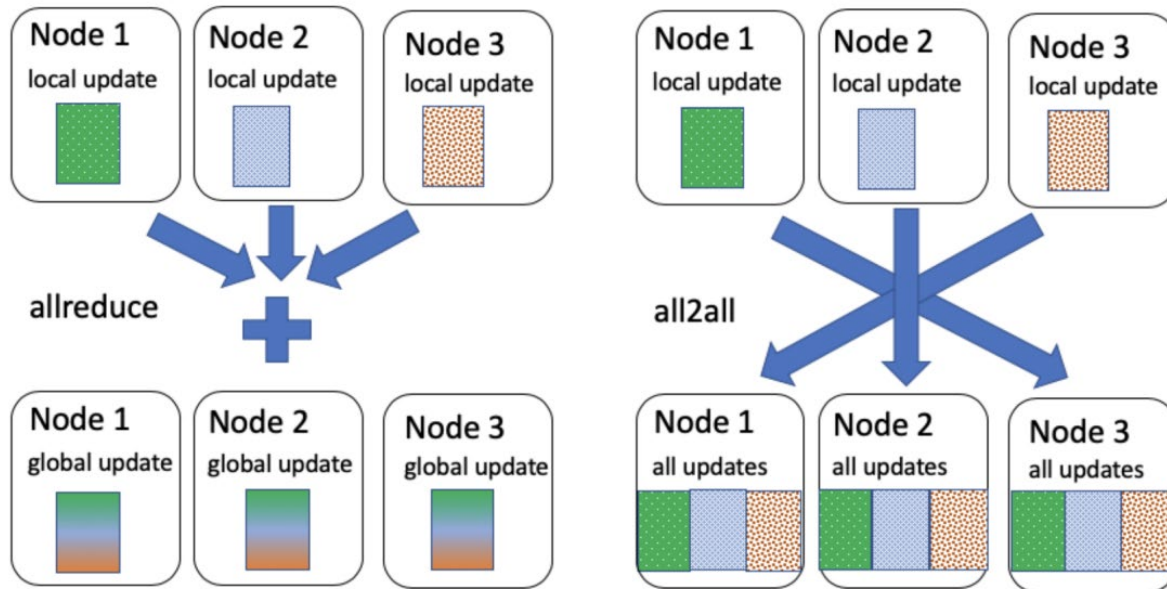
First work that explored the synergy of NCCL and MPI to push the envelope of performance!

Large models need parallelism

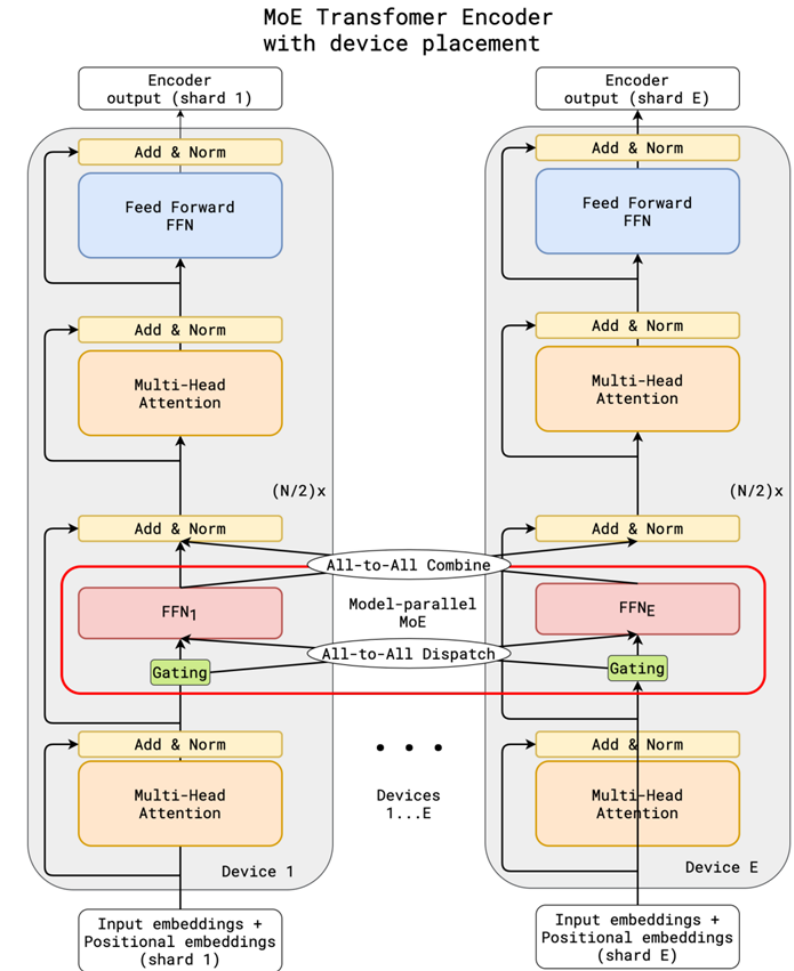
	Max Parameter (in billions)	Max Parallelism	Compute Efficiency	Usability (Model Rewrite)
Data Parallel (DP)	Approx. 1.2	>1000	Very Good	Great
Model Parallel (MP)	Approx. 20	Approx. 16	Good	Needs Model Rewrite
MP + DP	Approx. 20	> 1000	Good	Needs Model Rewrite
Pipeline Parallel (PP)	Approx. 100	Approx. 128	Very Good	Needs Model Rewrite
PP + DP	Approx. 100	> 1000	Very Good	Needs Model Rewrite
MP + PP + DP	> 1000	> 1000	Very Good	Needs Significant Model Rewrite
<i>ZeRO</i>	> 1000	> 1000	Very Good	Great

Communication in Deep Learning models

- DL models use many parallelism schemes
 - A single model replica may span multiple GPUs
 - Thus, requiring many different communication operations



Deep Learning Recommendation Model (DLRM)



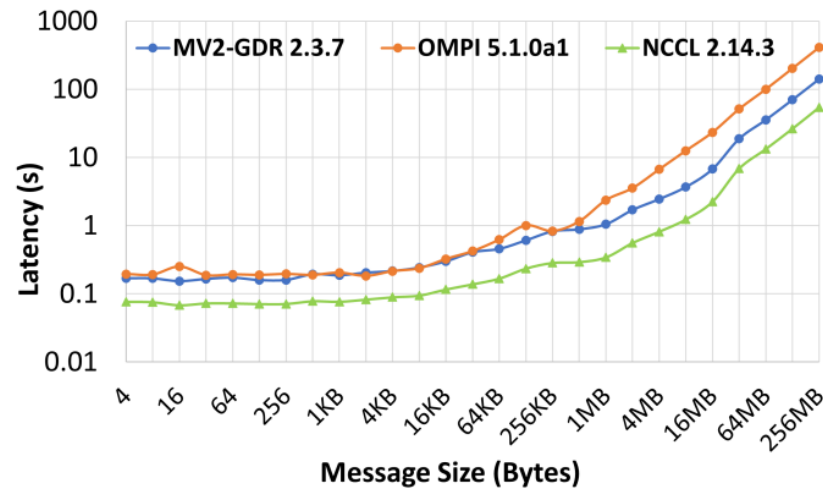
Mixture of Experts (MoE)

Limitations of Existing Solutions

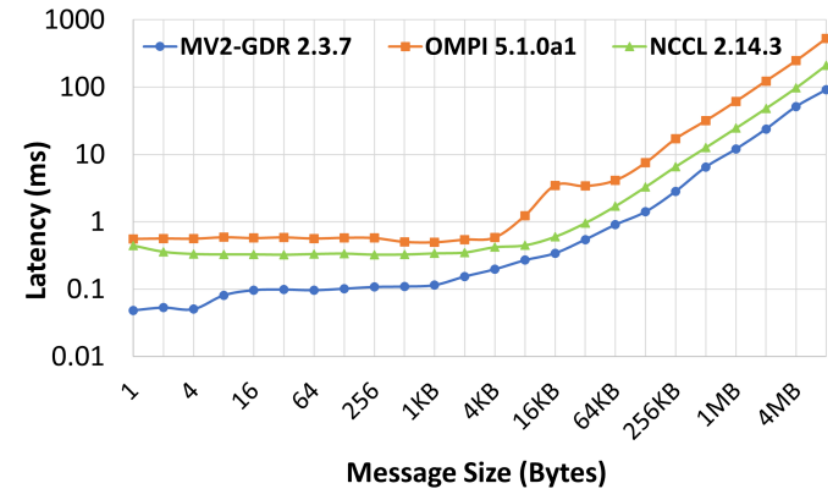
Studies	Features					
	Point-to-Point	Collectives	Vector Collectives	Non-Blocking Operations	Mixed-Backend Communication	Backend as a Class
Horovod	×	✓	×	NCCL Only	Experimental	×
PyTorch Distributed Module	✓	✓	×	NCCL Only	×	✓
LBANN	✓	✓	×	✓	×	×
mpi4py[17]	✓	✓	✓	✓	×	×
Proposed MCR-DL	✓	✓	✓	✓	✓	✓

- Currently, distributed DL frameworks require that the user choose a single communication backend
- Rarely-used communication operations (e.g. IGather) must be custom-implemented as needed

- Consider DS-MoE. Given that communication time is split between AllReduce and AlltoAll, which backend should be used?
 - MVAPICH2-GDR performs best for Alltoall, NCCL performs best for AllReduce



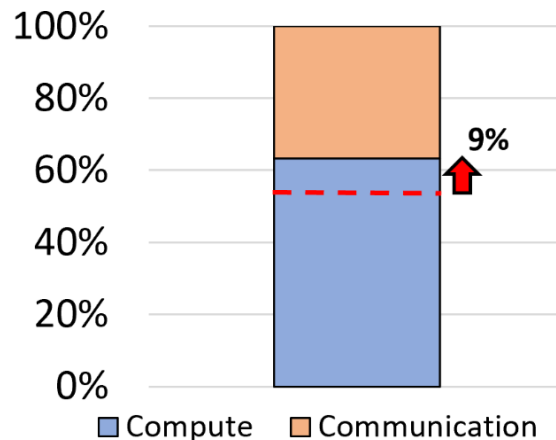
(a) 64 GPUs (16 node 4 ppn) - iAllreduce



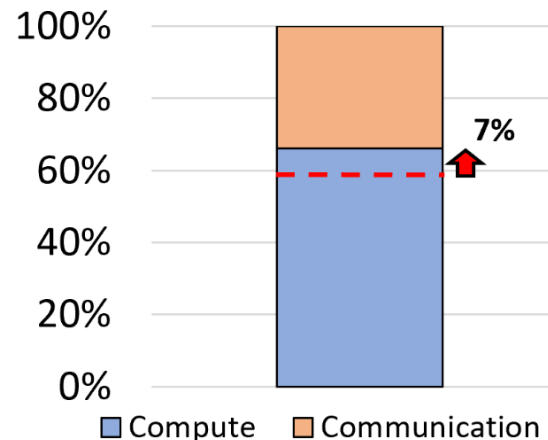
(b) 64 GPUs (16 nodes 4 ppn) - Alltoall

MCR-DL Flagship Results

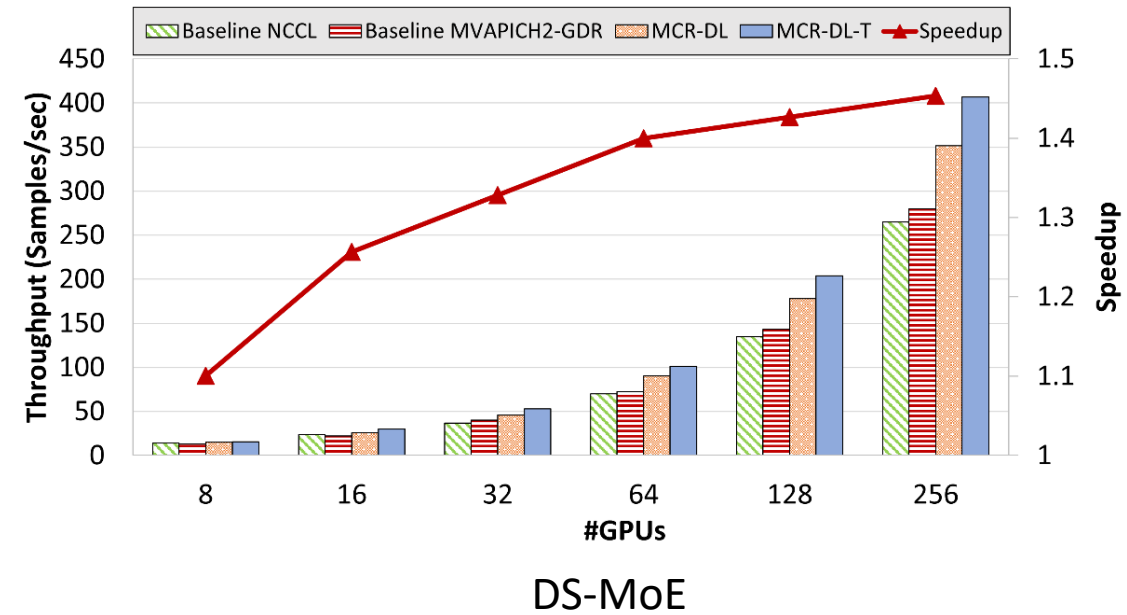
- Benefits for DS-MoE [top] and DLRM [bottom]
- Both primarily use AllReduce and AlltoAll
- Baseline results use a single backend for all operations
- *MCR-DL* coarsely chooses the best backend for each operation based on OMB results, *MCR-DL-T* uses tuning tables to choose the best backend based on the message size
- By using the best communication backends for each individual operation, **MCR-DL-T reduces time spent in communication**



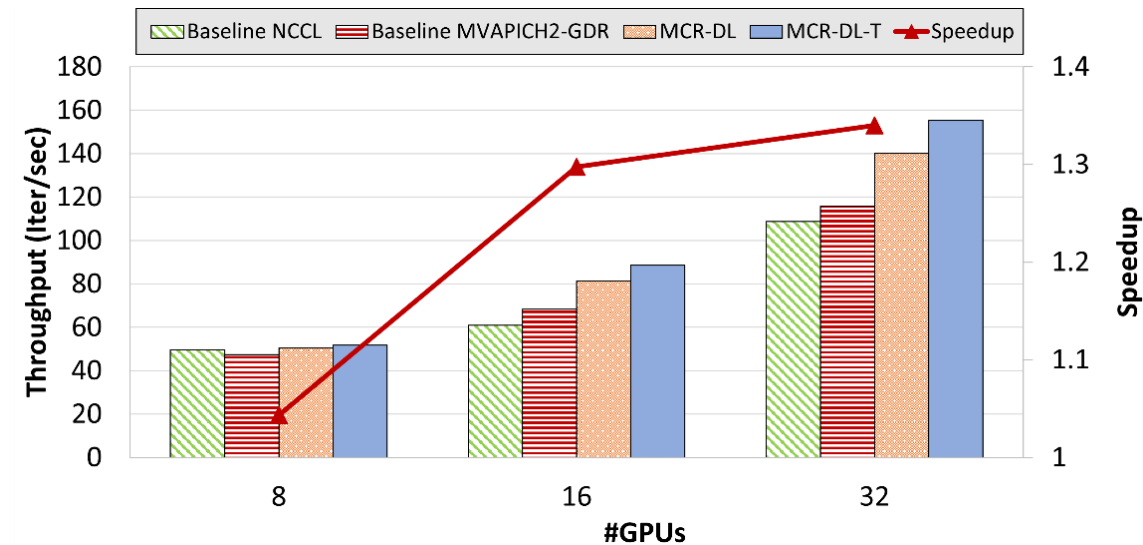
(a) DS-MoE



(b) DLRM



DS-MoE



DLRM

+

•

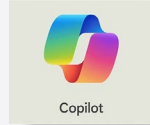
○

DeepSpeed-Ulysses

Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Reza Yazdani Aminabadi, Leon Song, Samyam Rajbhandari, Yuxiong He – [ArXiv '23, PODC '24](#)

System Optimizations for Enabling Training of Extreme Long Sequence Transformer Models

Motivation: Long sequence problems are all around us



Long context LLMs, chat apps and models (1K GPT-3 2022 ->128K Phi-3, 1M Gemini 2024)



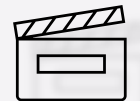
Book (chapter) level summarization



Health care predictive model conditioned on entire patient record



Long-range high-resolution climate modelling

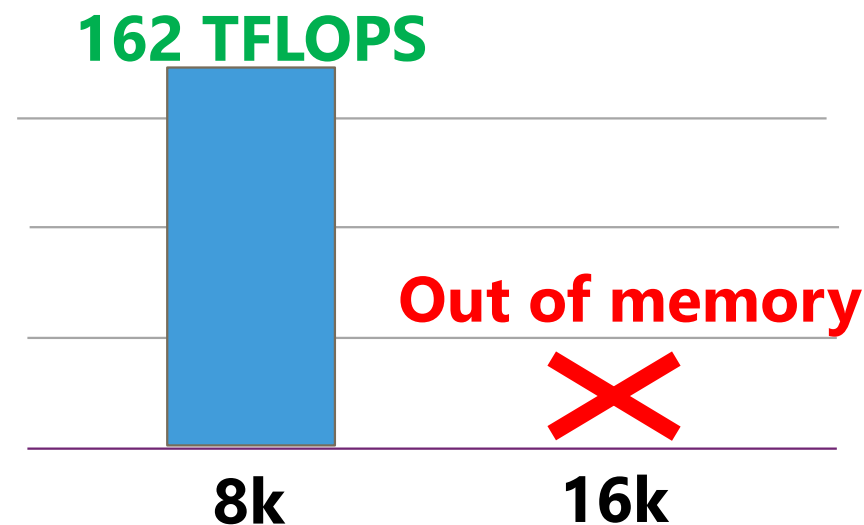


Multimodal AI

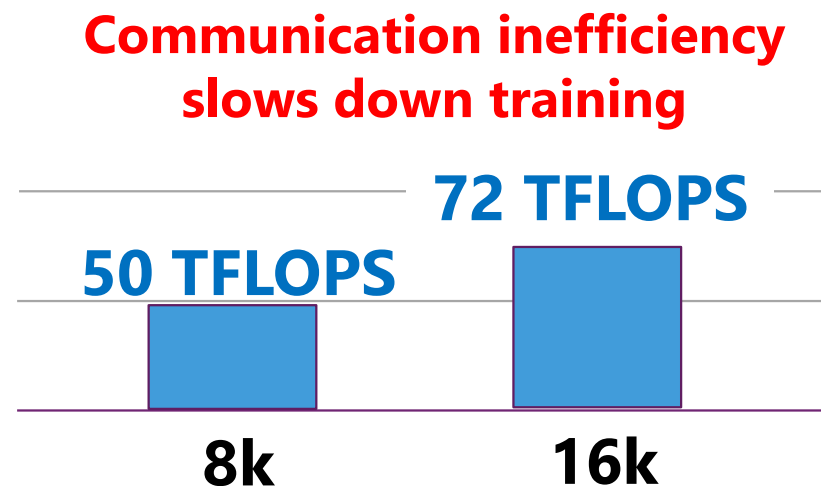
Challenges: Several Inefficiencies

- Memory Inefficiency
 - Existing (data, tensor, pipeline) parallelism approaches cannot address memory demand of extreme long sequences
- Communication Inefficiency
 - Existing sequence parallelism approaches are not effective because of communication inefficiencies.
- Easy of use
 - Existing approaches have limited usability requiring error prone code refactoring

Microsoft DeepSpeed ZeRO-3

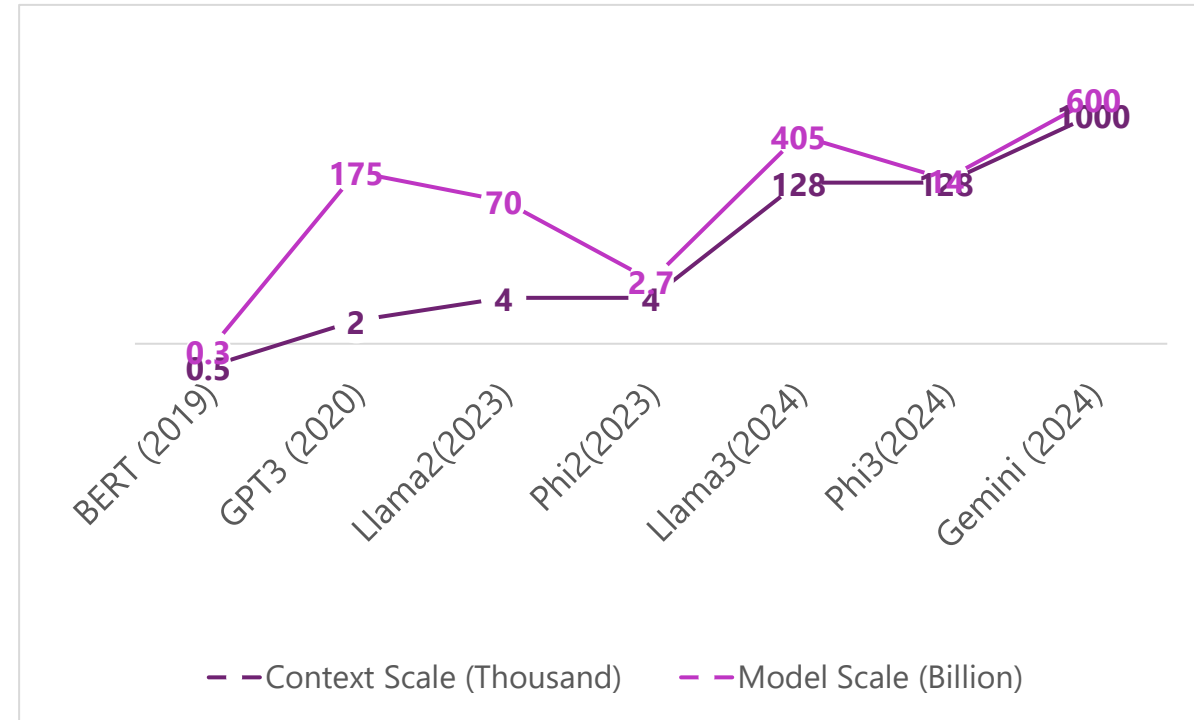



NVIDIA Megatron Sequence Parallelism



DeepSpeed-Ulysses

- DeepSpeed-Ulysses is our technological innovation for long context optimization
 - **Key idea:** partition tensors along **sequence** and **head** dimensions, use optimized alltoall collective for communication
- First long context system optimization to scale to **1M pretraining context length**
- Trains 2.5x faster and 4x longer context length than Megatron-LM



 **DeepSpeed**
@MSFTDeepSpeed

Want to train 1 million token context lengths (all 7 of the Harry Potter books! 📖) on a GPT-like model w. 64 GPUs?

Announcing DeepSpeed-Ulysses 🚀

This release enables highly efficient and scalable LLM training with extremely long sequence lengths 🤖

[github.com/microsoft/Deep...](https://github.com/microsoft/DeepSpeed-Ulysses)

DeepSpeed-Ulysses: Key Features



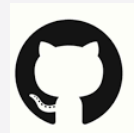
Communication
Efficiency



Memory
Optimization



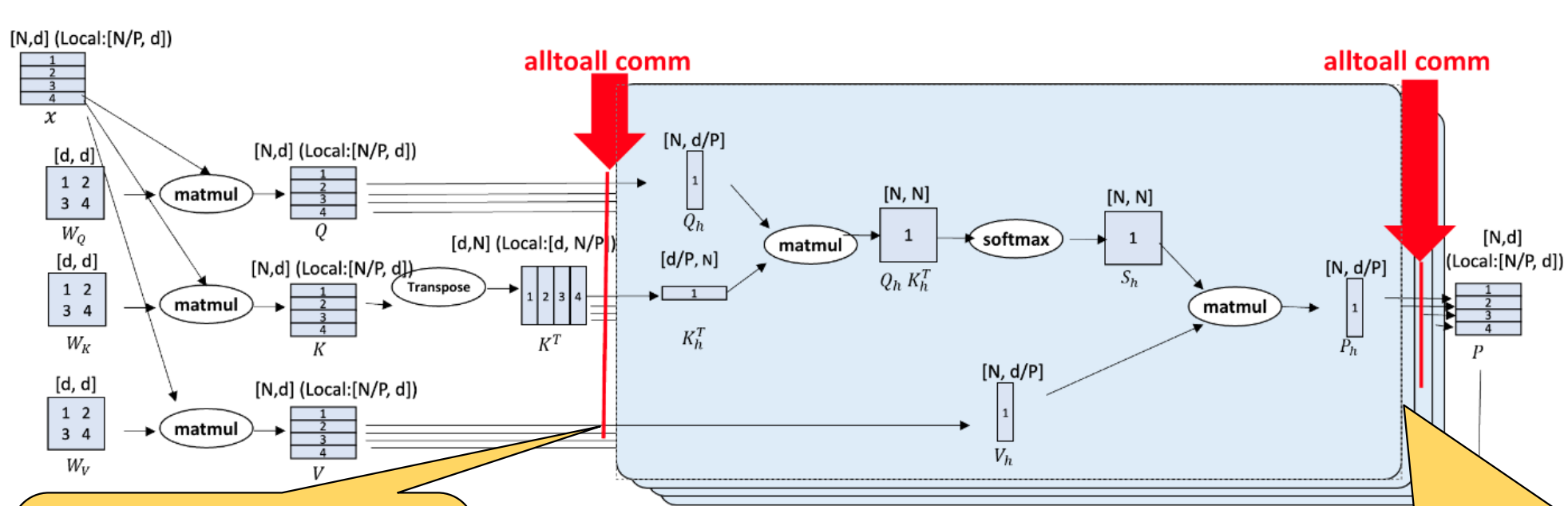
Attention
Agnostic



Ease of Use and
Open Source

DeepSpeed-Ulysses: Core Design

- Partitions individual samples along the sequence dimension
- Employs alltoall communication collective for attention computations



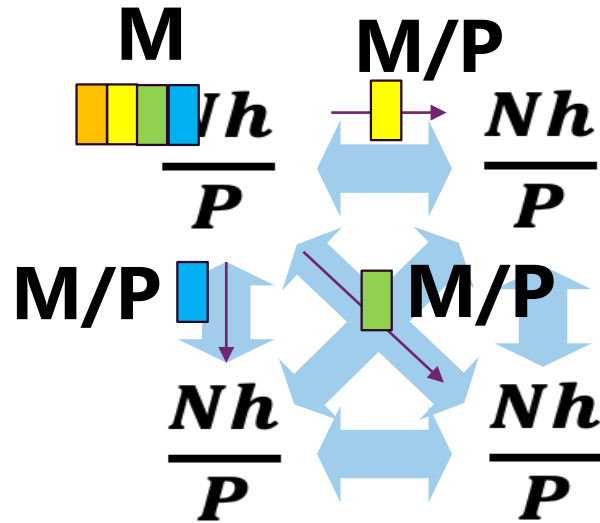
Converts sequence parallelism to head parallelism

N: sequence length
 d: hidden size
 hc: head count
 P: total processor (GPU) count
 Assumes $P = hc = 4$

Converts head parallelism back to sequence parallelism

Communication Volume Analysis

- Key: All-to-all communication overhead is $O(M/P)$



All-to-all on DGX-type network
(each GPU pair has a dedicated link)

Existing work
(Megatron-LM's SP)

$$O(M)$$

($M = 2Nh$)

Before Attention-block

allgather Nh

After Attention-block

reduce-scatter Nh

DeepSpeed-Ulysses

$$O(M/P)$$

($M = 4Nh/P$)

all-to-all $\frac{3Nh}{P}$

all-to-all $\frac{Nh}{P}$

P : GPU count, N : sequence length, h : hidden size

Communication Volume Analysis

- DeepSpeed-Ulysses has less communication overhead

Existing work
(Megatron-LM's SP)

$O(M)$

DeepSpeed-Ulysses

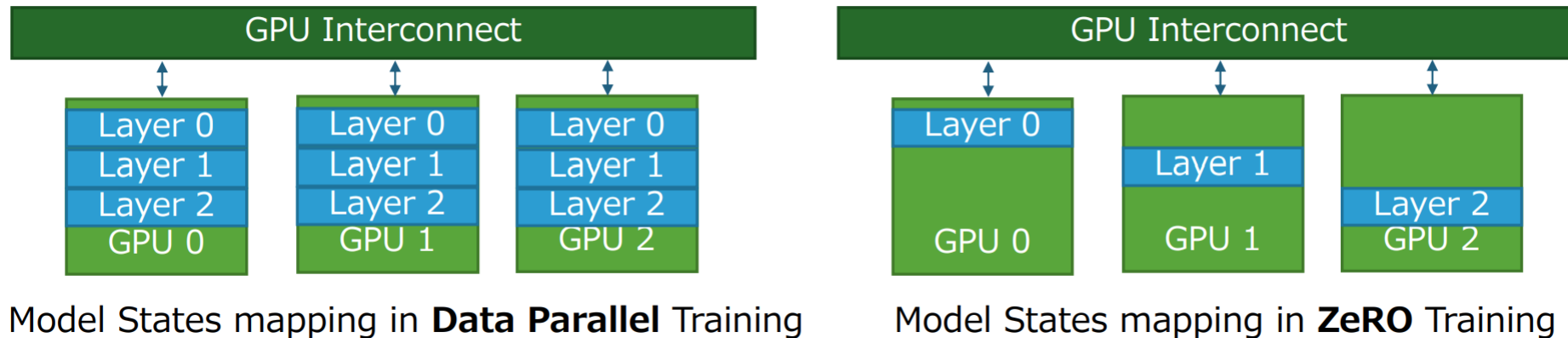
$O(M/P)$

N : message size \sim sequence length, P : GPU count

- **Megatron-LM** increases communication overhead as sequence length increases
- **DeepSpeed-Ulysses** can keep the communication volume consistent by increasing GPUs proportionally to sequence length

Parameter Memory Optimization

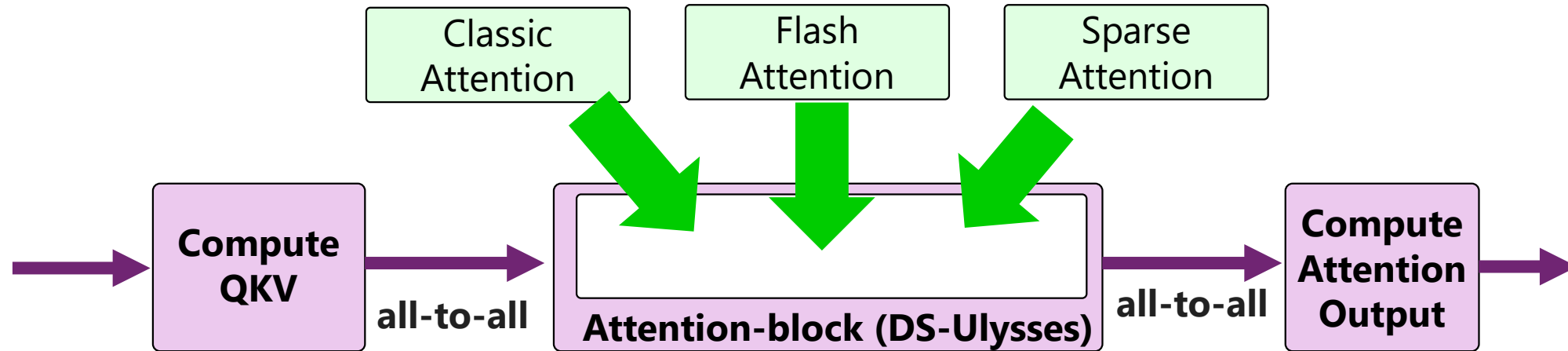
- DeepSpeed-Ulysses leverages DeepSpeed-ZeRO for memory parameter optimization
 - Extends parameter sharding across both data and sequence parallel group
 - Allgather communication collective in forward and backward pass



- Activation partitioning (DeepSpeed-Ulysses) + parameter sharding (DeepSpeed-ZeRO) enables larger batch sizes for higher training throughput

Generality and Easy Use

- Agnostic to attention implementation
- Head parallelism allows for compatibility with different kind of attention



- Require minimal code changes

```
attn = Attention_Ulysses(attn, ...)  
context = attn(q, k, v)
```

**Only wrap the
attention module**

Evaluation : Max Sequence Length

Max sequence length (X1000)

1200

1000

800

600

400

200

0

- GPT 1.2B model
- A100-40G

4

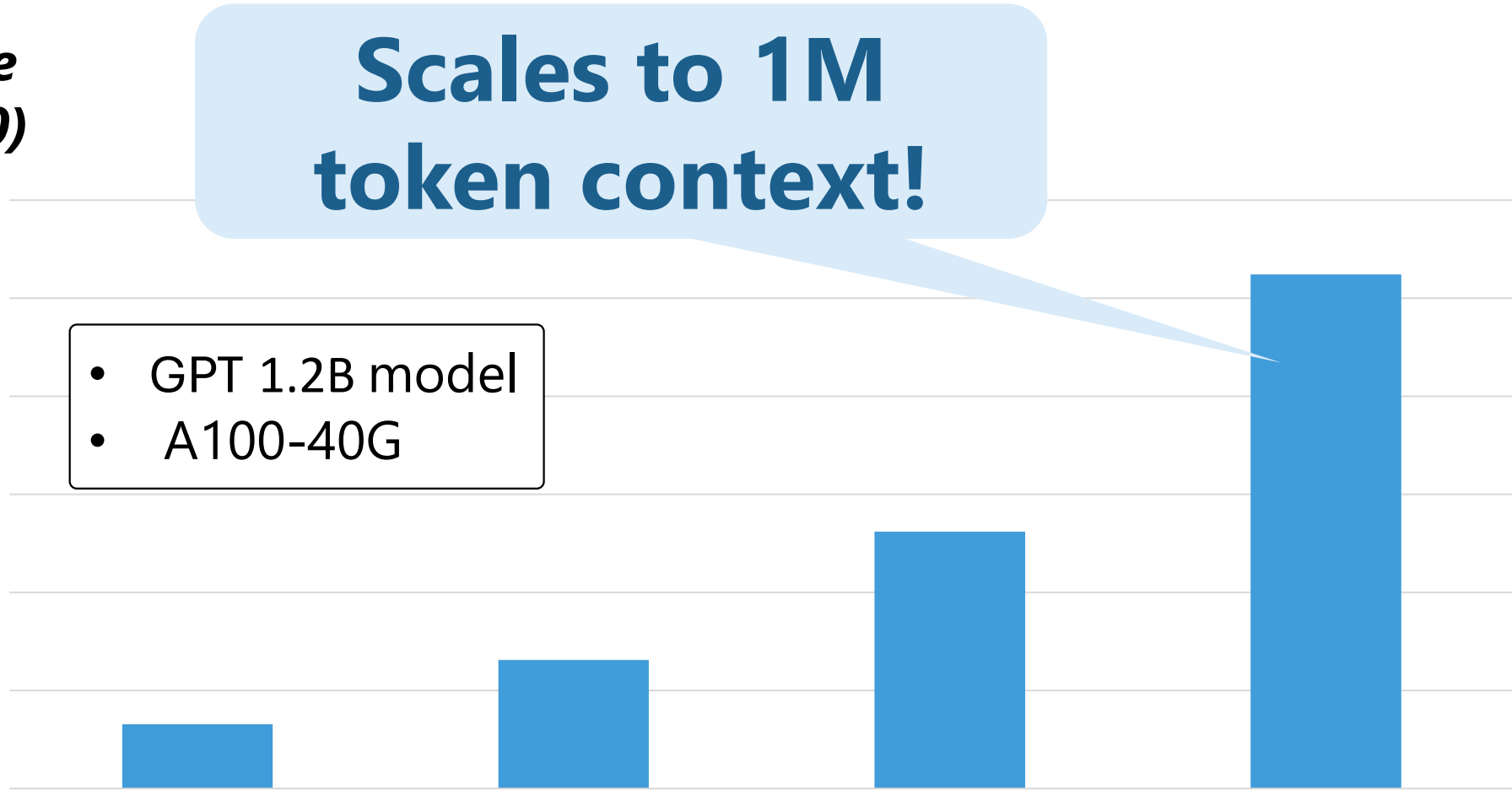
16

32

64

GPU count

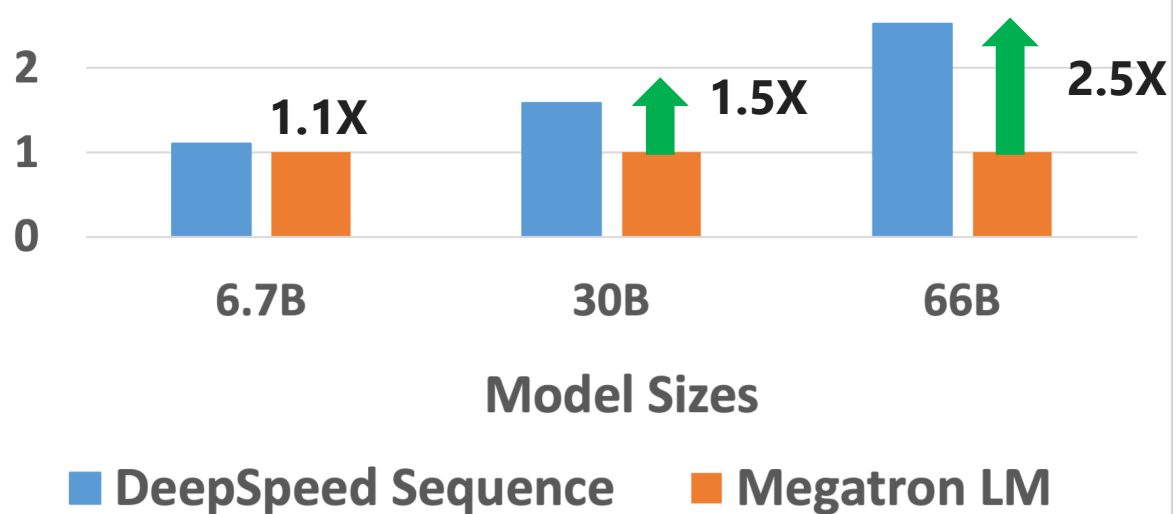
**Scales to 1M
token context!**



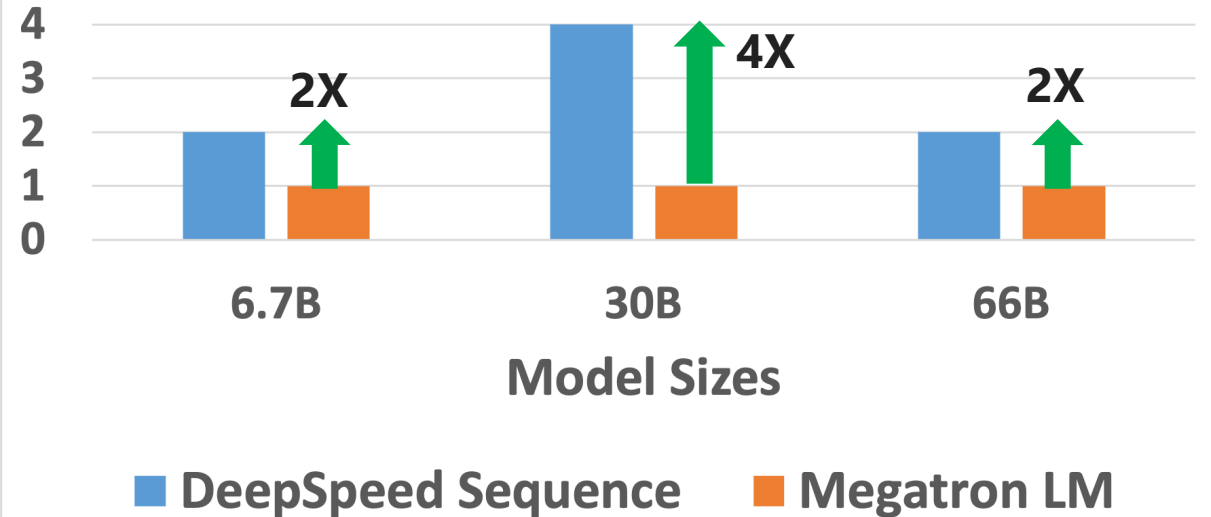
Comparison with Existing Approach

Significantly better throughput for large models

Normalized Throughput



Normalized Maximum Sequence Length



DS sequence parallelism (Ulysses) can train 2.5x faster and 4x longer sequences than SoTA

Ongoing and Future work

- Ongoing work focuses on framework generalization, advanced optimization and democratization
 - HuggingFace integration for post training, finetuning and broad model evaluation
 - Computation-communication overlap
 - 10% improvement over baseline
 - Contribution from Intel
 - Ulysses-Offload (summer intern project)
 - Leverages memory hierarchies
 - Enables training sequences over 2 million tokens for a 13B model using just a single node
 - 4x improvement over baseline
- Explore the synergy of MVAPICH and NCCL?
 - Which student is coming to Microsoft for Internship next? 😊

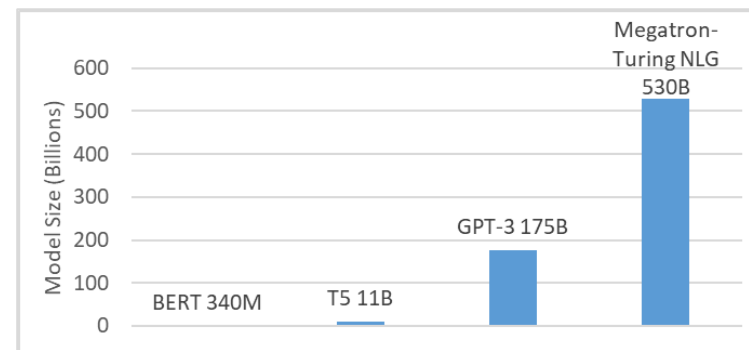
- + • ZeRO++

- Guanhua Wang, Heyang Qin, Sam Ade Jacobs, Xiaoxia Wu, Connor Holmes, Zhewei Yao, Samyam Rajbhandari, Olatunji Ruwase, Feng Yan, Lei Yang, Yuxiong He – ICLR '23

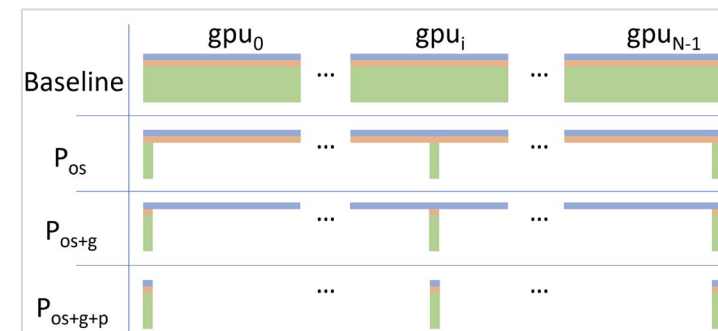
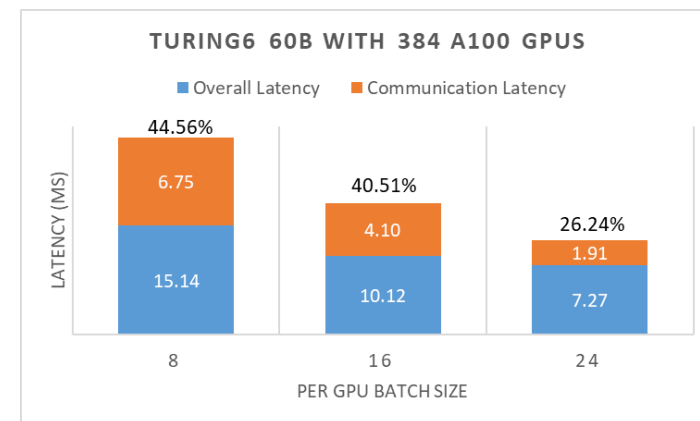
Extremely Efficient Collective Communication for Large Model Training

Motivation

- Large model requires large #GPUs to train
 - Max batch size has a limit
 - Large number of GPU implies smaller batch per GPU
- Communication, a significant overhead
 - Frequent communication with small batch size
- Make efficient large-scale training accessible
 - ZeRO: Easy-to-use large model training
 - Develop techniques to reduce communication volume for better efficiency



Model size grows exponentially



ZeRO: a key enabler of large-scale training

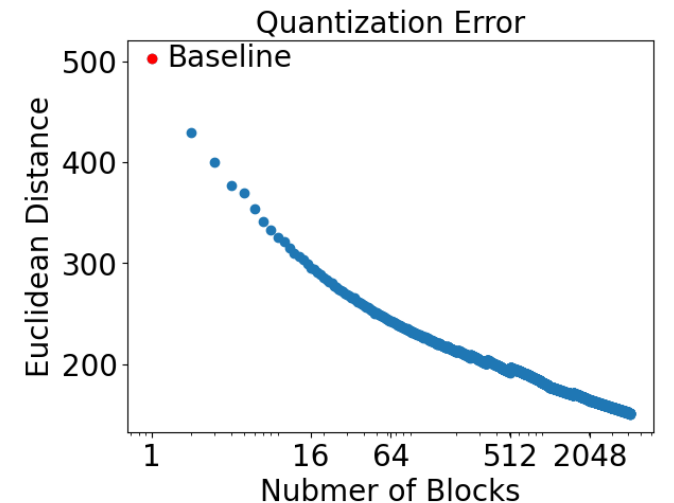
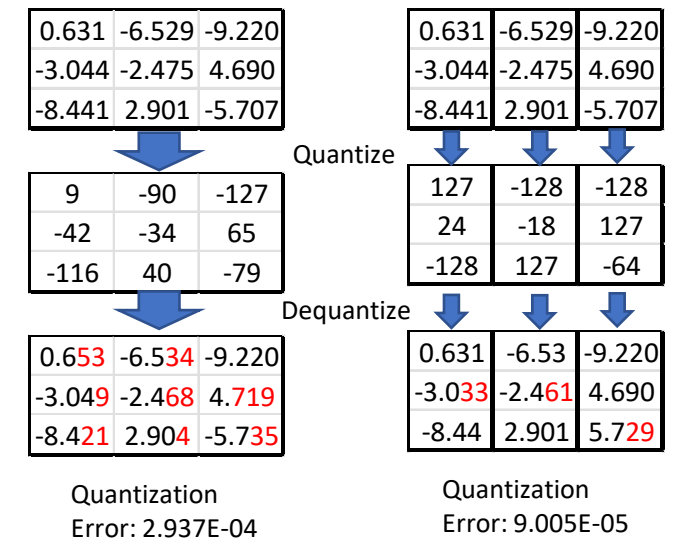
Communication Characterization for ZeRO

- ZeRO partitions model states
 - Utilize aggregate GPU memory
 - Communication collectives to fetch required model states during training
- Communication Volume Breakdown (model size M):
 1. Forward all-gather on weights: M
 2. Backward all-gather on weights: M
 3. Backward reduce-scatter on gradients: M
- ***Total Volume: $3M$***

How to reduce communication volume?

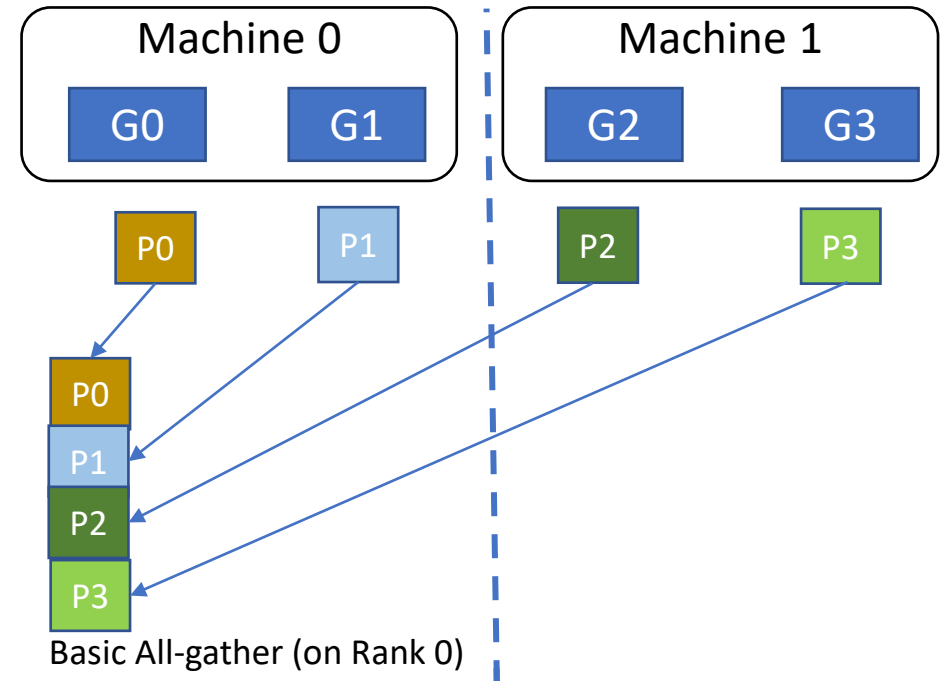
1: Reduce *forward all-gather* Comm

- Communicate 8-bit quantized weights
 - But naïve quantization causes model divergence
- Blocked quantization
 - Reduce quantization granularity to improve precision
 - 3.3x precision improvement in Euclidean distance
 - Optimized kernels for 2.5x faster performance over pytorch
- E2E comm reduction: **3M → 2.5M**
 - Forward all-gather: $M \rightarrow 0.5M$
 - Backward all-gather: M
 - Backward reduce-scatter: M



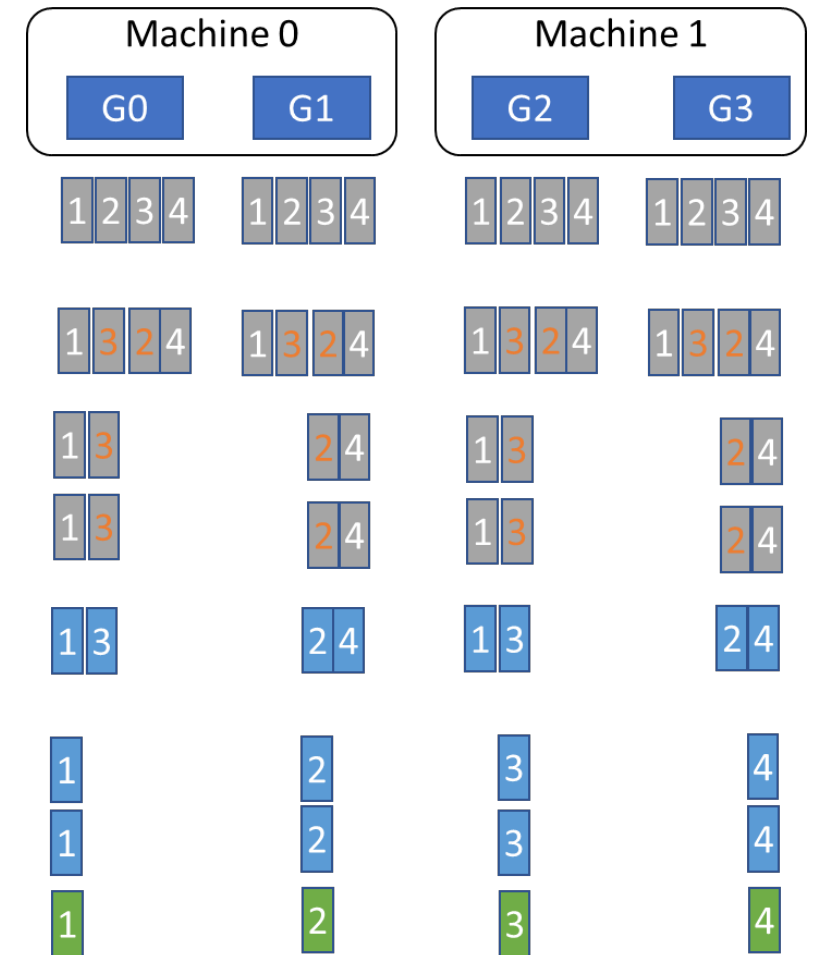
2: Reduce *backward all-gather* Comm

- Heterogeneous partitioning (hpZeRO)
 - Model weights within node, rest across all nodes
 - All-gather happens within node only
 - Trade off between memory and communication
- Eliminates backward all-gather across nodes
- E2E comm reduction: $3M \rightarrow 1.5M$
 - Forward all-gather: $M \rightarrow 0.5M$
 - Backward all-gather: $M \rightarrow 0$
 - Backward reduce-scatter: M



3: Reduce *backward reduce-scatter Comm*

- Can we quantize gradient communication ?
 - Significant precision loss due to reduction operations
- Novel hierarchical All-to-All replacing reduce-scatter
 - Communicate in 4 or 8-bits
 - Reduce in full-precision
- Solves multiple system and algorithmic challenges
 - Details illustration later
- E2E comm reduction: **$3M \rightarrow 0.75M$**
 - Forward all-gather: $M \rightarrow 0.5M$
 - Backward all-gather: $M \rightarrow 0$
 - Backward reduce-scatter: $M \rightarrow 0.25M$



Methodology Summary

Breakdown of ZeRO communication cost (consider a model of size M):

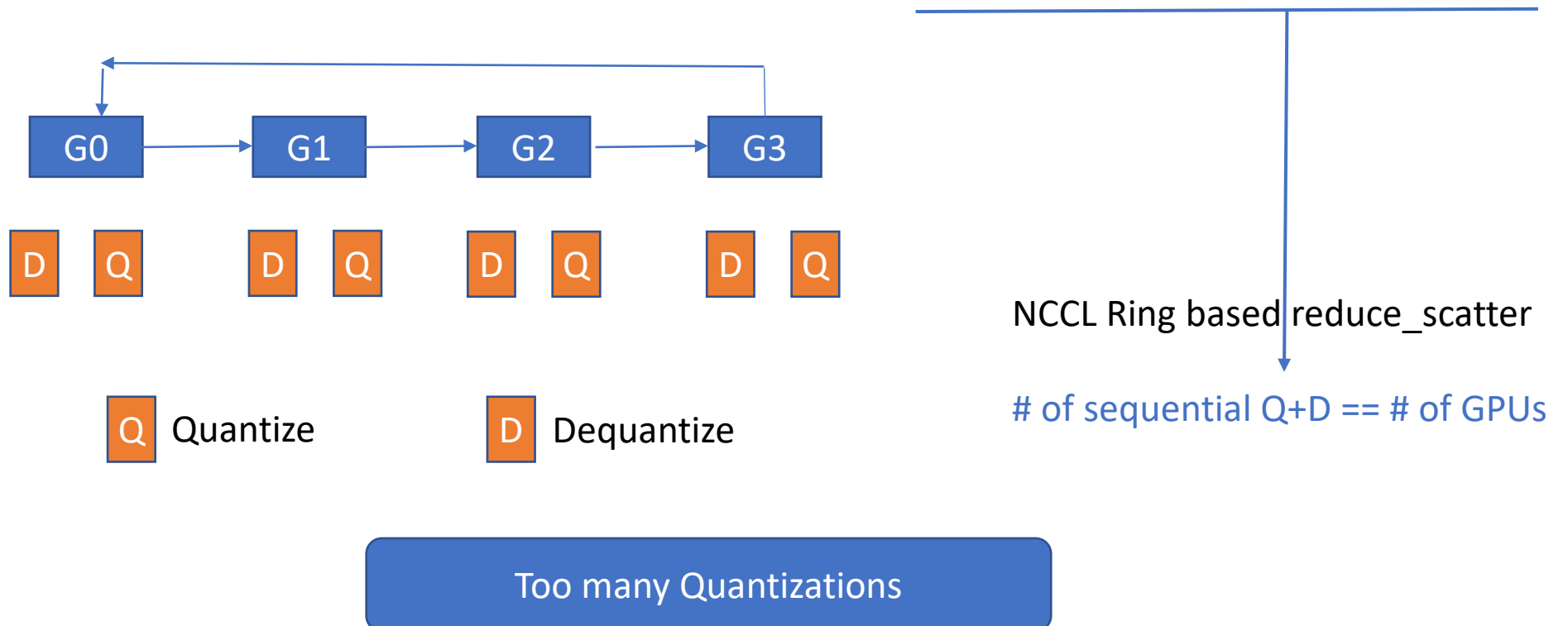
1. Forward all-gather (size M) $\xrightarrow{\text{Accurate \& Efficient Quantization}}$ (size $0.5M$)
2. Backward all-gather (size M) $\xrightarrow{\text{Heterogeneous Partitioning (hpZeRO)}}$ (size 0)
3. Backward reduce-scatter (size M) $\xrightarrow{\text{Novel Quantized Collective}}$ (size $0.25M$)

Overall communication reduction: $3M \rightarrow 0.75M$

System Design for Gradients Communication

Initial Challenges for Quantization on Gradients:

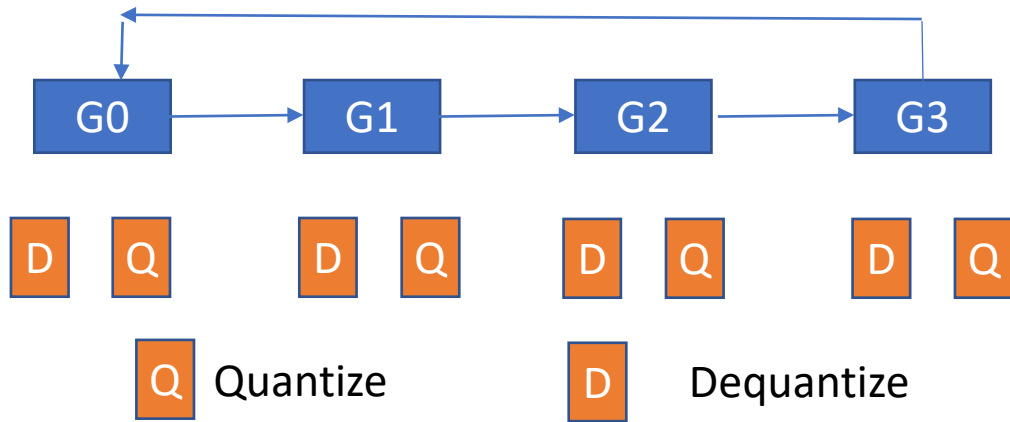
- No existing collectives for quantized gradient communication
- 1-bit Adam optimizer cannot be applied at ZeRO-3.
- Directly apply quantization on reduce_scatter has longer latency & lower precision



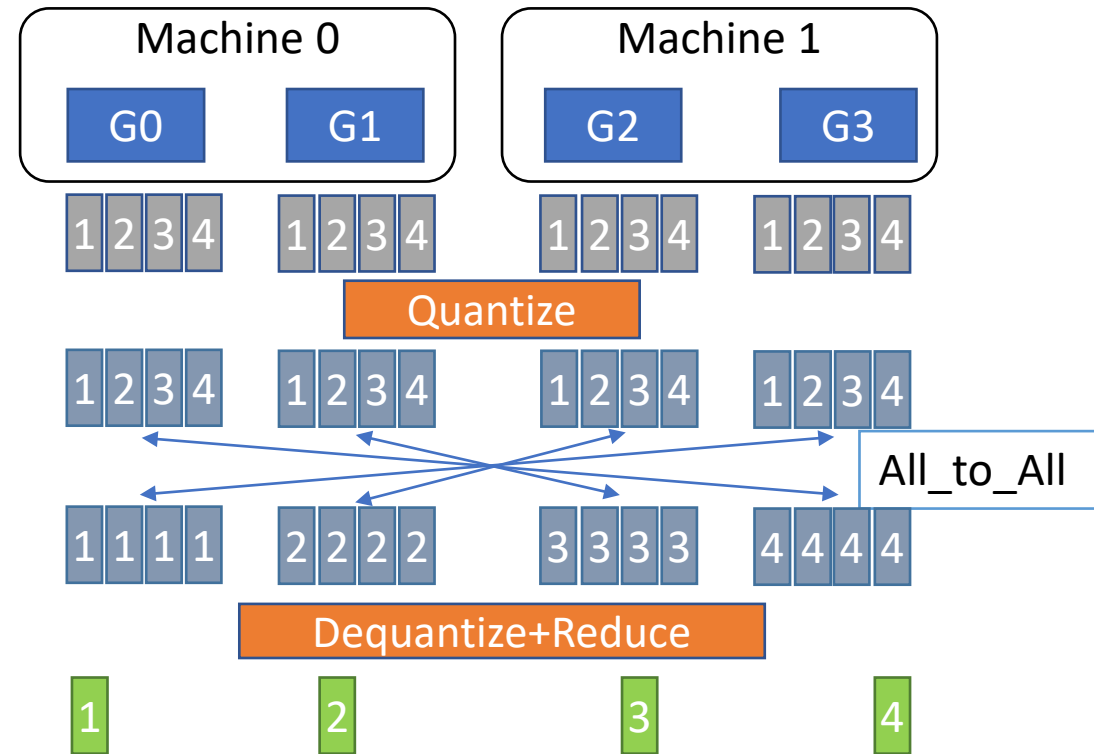
System Design for Gradients Communication

Challenge 1: Too many Quantizations

Solution 1: Replacing ring-based reduce_scatter with 1-hop all_to_all



NCCL Ring-based reduce_scatter
of sequential Q+D == # of GPUs

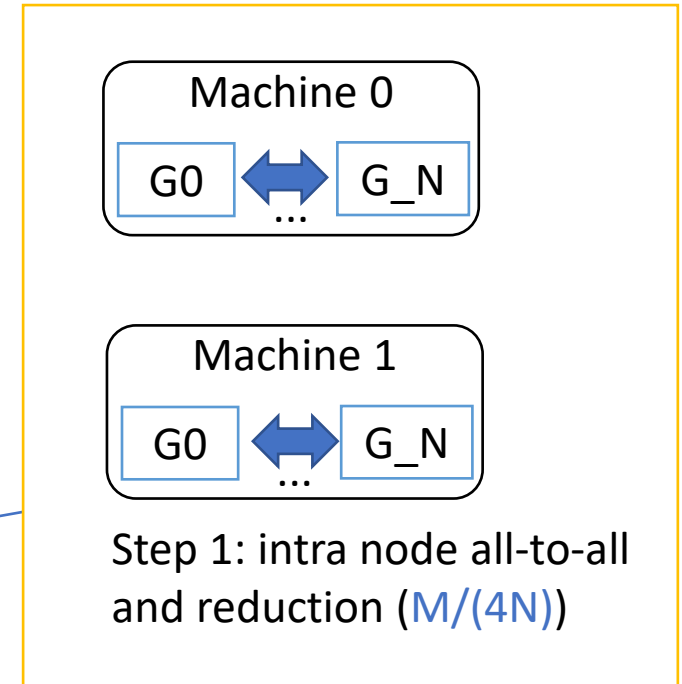
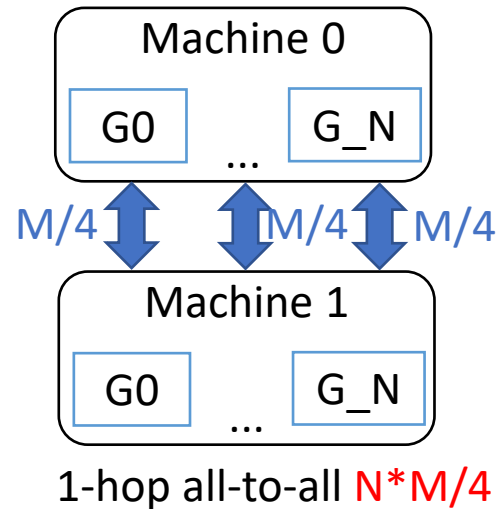
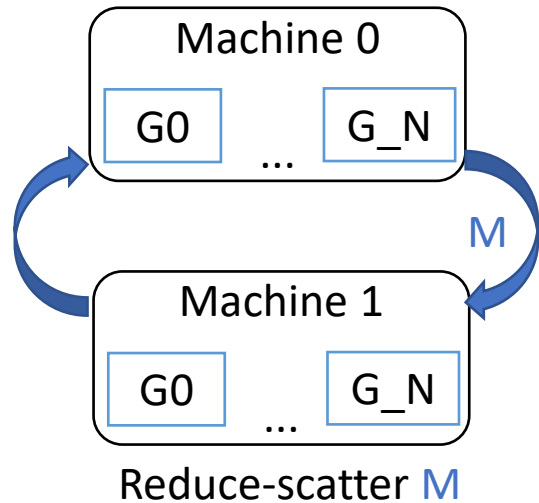


Our 1-hop all_to_all
of sequential Q+D == 1

System Design for Gradients Communication

Challenge 2: Issue with 1-hop all_to_all -> communication volumes blow-up

N gpu per node, model size M

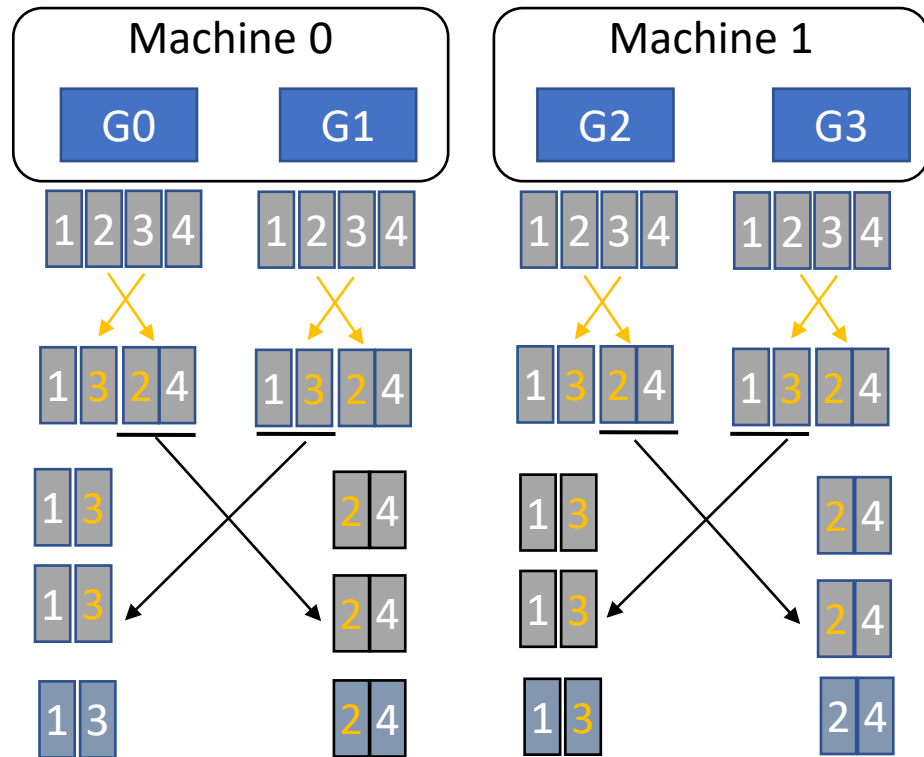


Solution 2: Hierarchical all-to-all

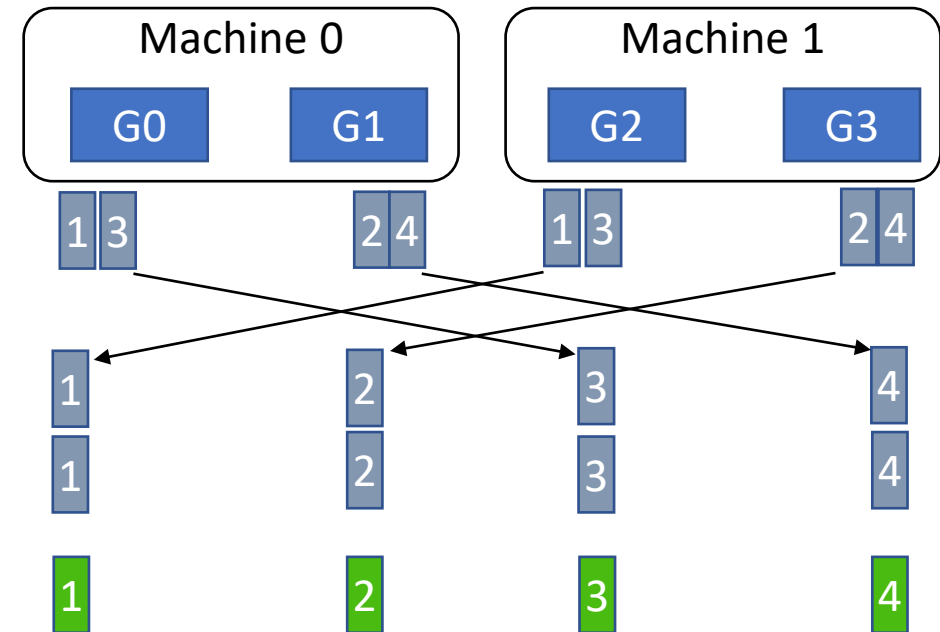
System Design for Gradients Communication

Challenge 3: Hierarchical all-to-all ([Data-misplacement](#))

Solution 3: Tensor slices reordering

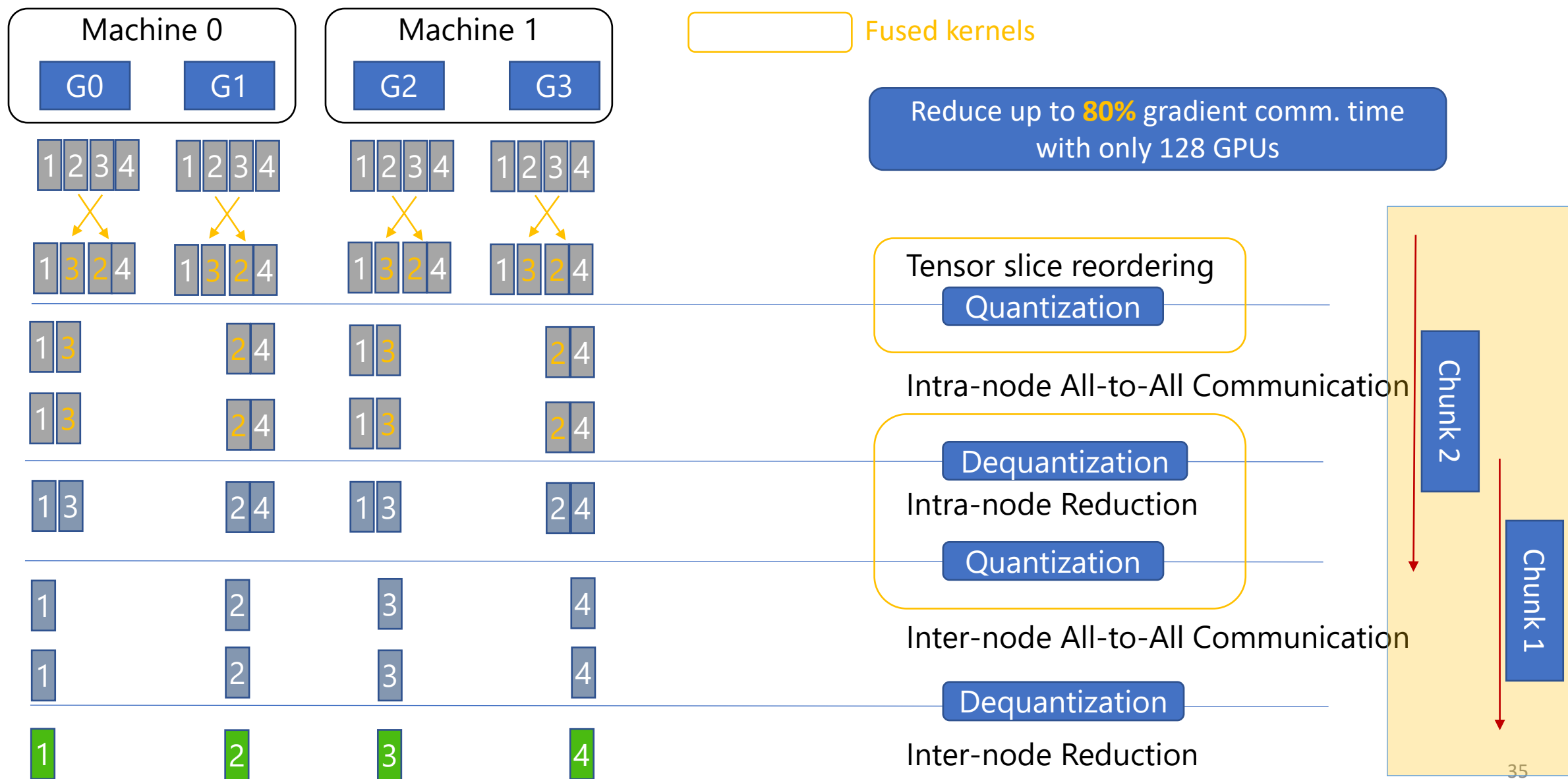


Step 1: intra-node all_to_all



Step 2: inter-node all_to_all

Further Optimization: kernel fusion & overlapping



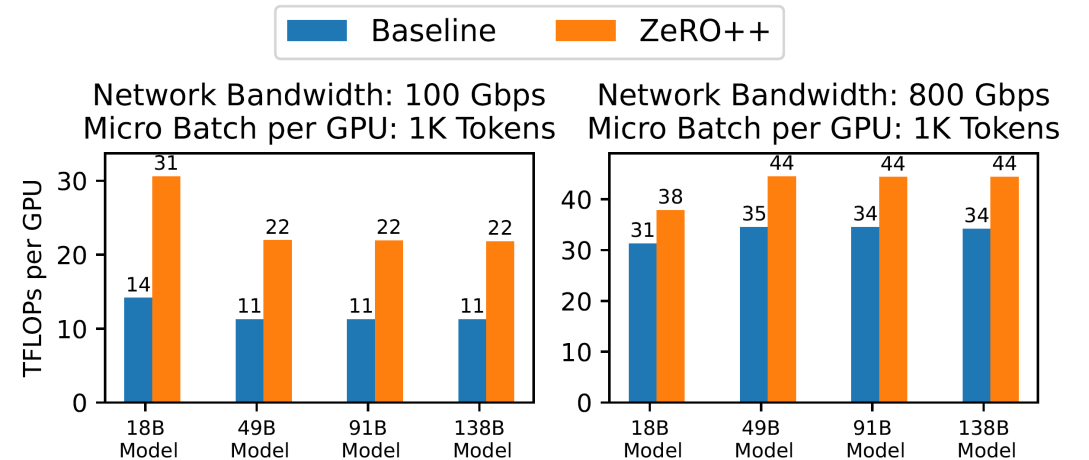
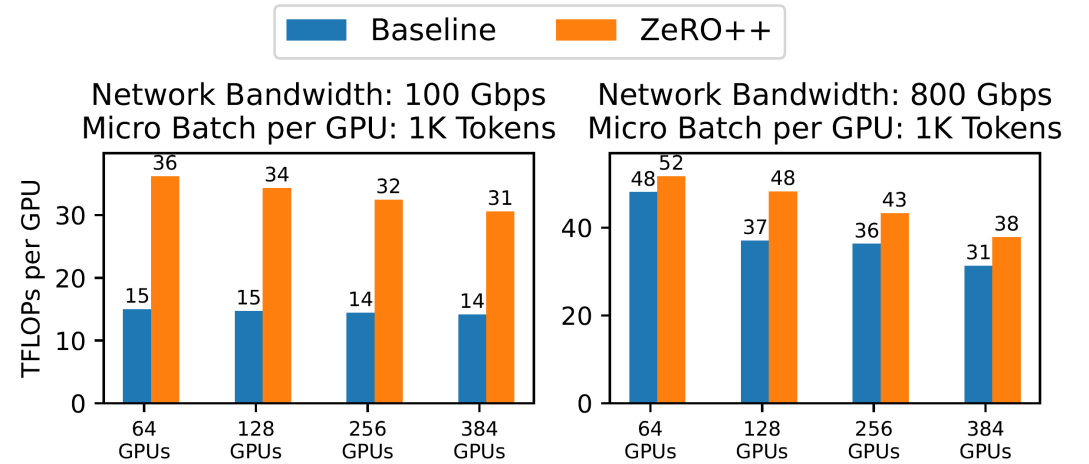
End-to-end Evaluation

- ZeRO++ on different number of GPUs

- ZeRO++ improves throughput on different scalability levels
- 100Gbps: 121% - 140% speedup
- 800Gbps: 8% - 30% speedup

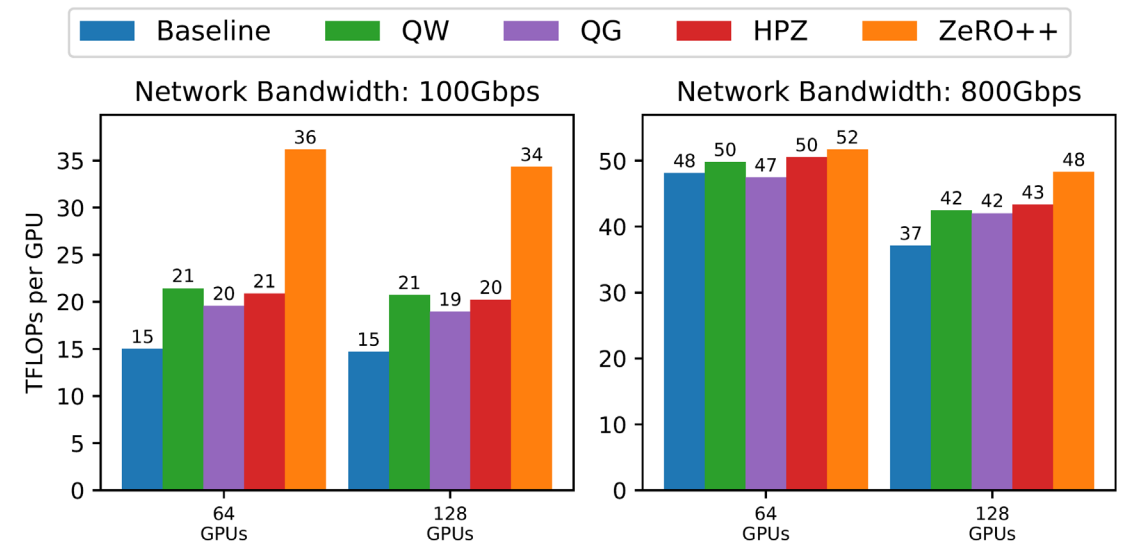
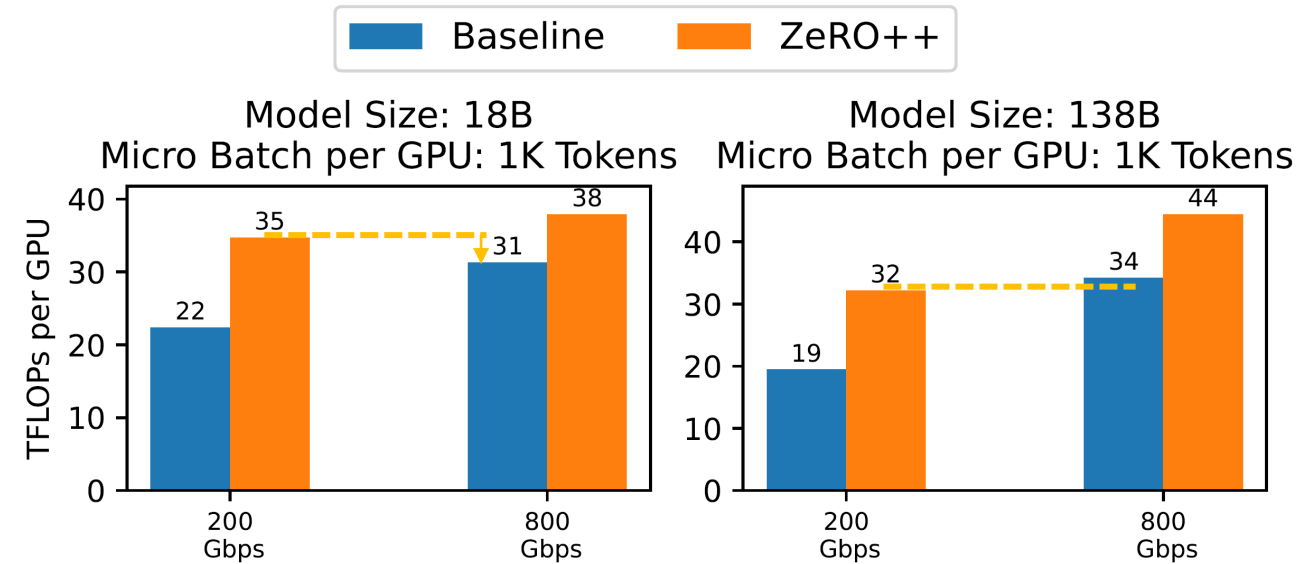
- ZeRO++ on different sizes of models

- ZeRO++ shows consistent speedup regardless of model size
- 100Gbps: over 100% speedup
- 800Gbps: up to 30% speedup
- 10Gbps: up to 300% speedup



End-to-end Evaluation (Cont.)

- ZeRO++ democratizes large scale training
 - ZeRO++ can achieve the same or better throughput with only $\frac{1}{4}$ of bandwidth
 - Confirms our 4x communication reduction
- Ablation study of individual optimizations
 - ZeRO++ achieves a good composition of individual optimizations

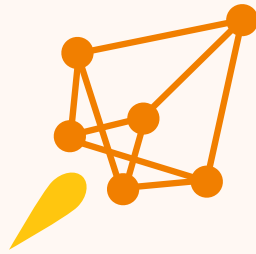


Summary of ZeRO++

- 3 novel communication optimizations on top of ZeRO
 - Reduce communication volume from *3M to 0.75M*
- End-to-end evaluations show 50%-140% speedup on various test scenarios
- Open-sourced as part of DeepSpeed
- LinkedIn reports 2.4x speedup in their training stack by using ZeRO++
- Read more details in the paper: “ZeRO++: expand scalability to power bigger models on more devices by minimizing communication cost” – ICLR ‘23

Exploring new ideas: MVAPICH + NCCL can unlock next level of performance!

- *Impact-first mindset vs. publication-first mindset*
- *Build on top vs. build from scratch*
- *Collaborate not compete! But also, collaborate and compete!*



deepspeed

We are looking for people and organizations to support the open-source DeepSpeed ecosystem

Make your first pull request 😊

<https://github.com/microsoft/DeepSpeed>

www.deepspeed.ai