ZeRO: Memory Optimizations Toward Training Trillion Parameter Models

Samyam Rajbhandari, Je Rasley, Olatunji Ruwase, Yuxiong He

Presented by: Hunter Richards

January 31, 2025

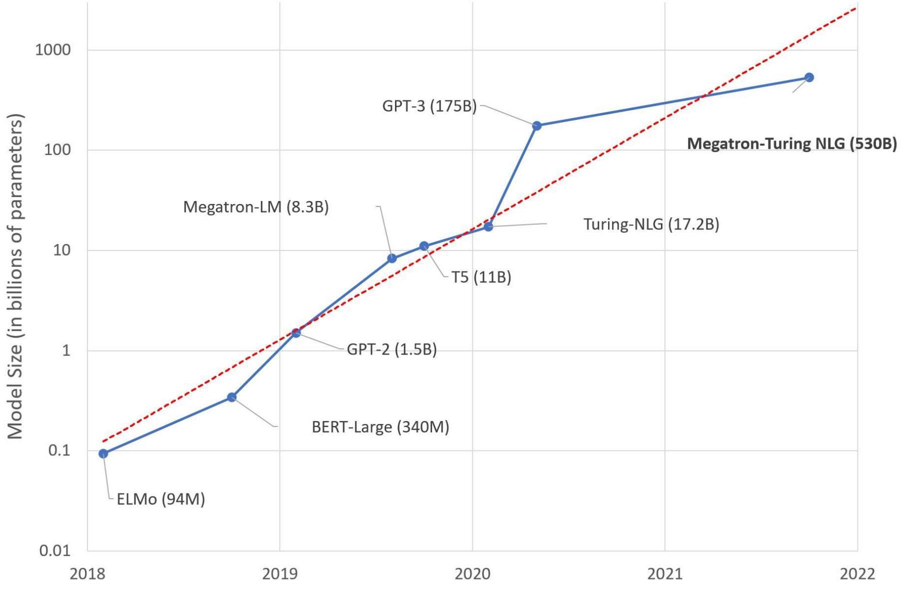
What is Zero Redundancy Optimizer (ZeRO)?

From the DeepSpeed tutorial:

> ZeRO is a powerful set of memory optimization techniques that enable effective training of large models with trillions of parameters, ... Compared to the alternative model parallelism approaches for training large models, a key appeal of ZeRO is that no model code modifications are required.

Motivation

Scaling laws dictate that bigger models consistently improve performance:



https://huggingface.co/blog/large-language-models

Motivation

Many factors contribute to the memory footprint occupied during training:

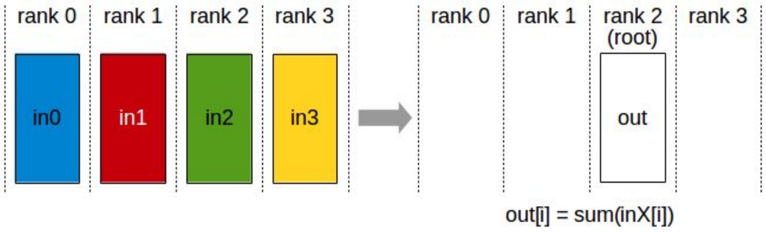
- Parameters
- Gradients
- Optimizer states (e.g., first and second moment estimates in Adam)
- Activations
- Temporary buffers
- Memory fragmentation

Trillion-parameters + Adam + FP16 requires ~16TB for parameters, gradients and optimizer states:

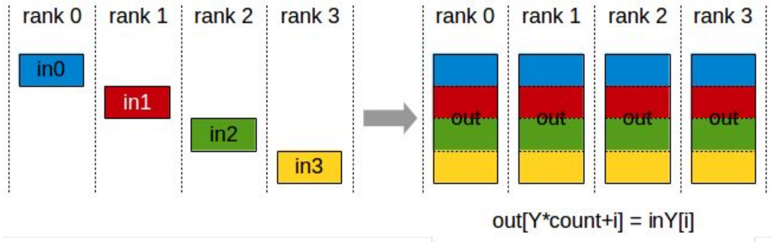
- What do we do when we reach the physical limits of a single GPU/TPU?
- We scale to many GPUs/TPUs

Preliminaries: Collective Operations

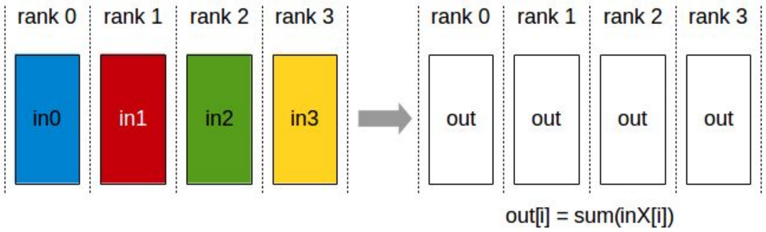
Reduce



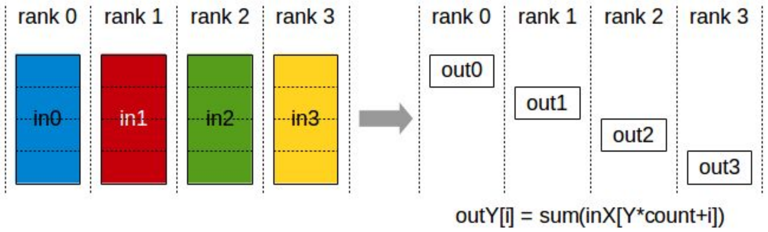
All Gather



All Reduce



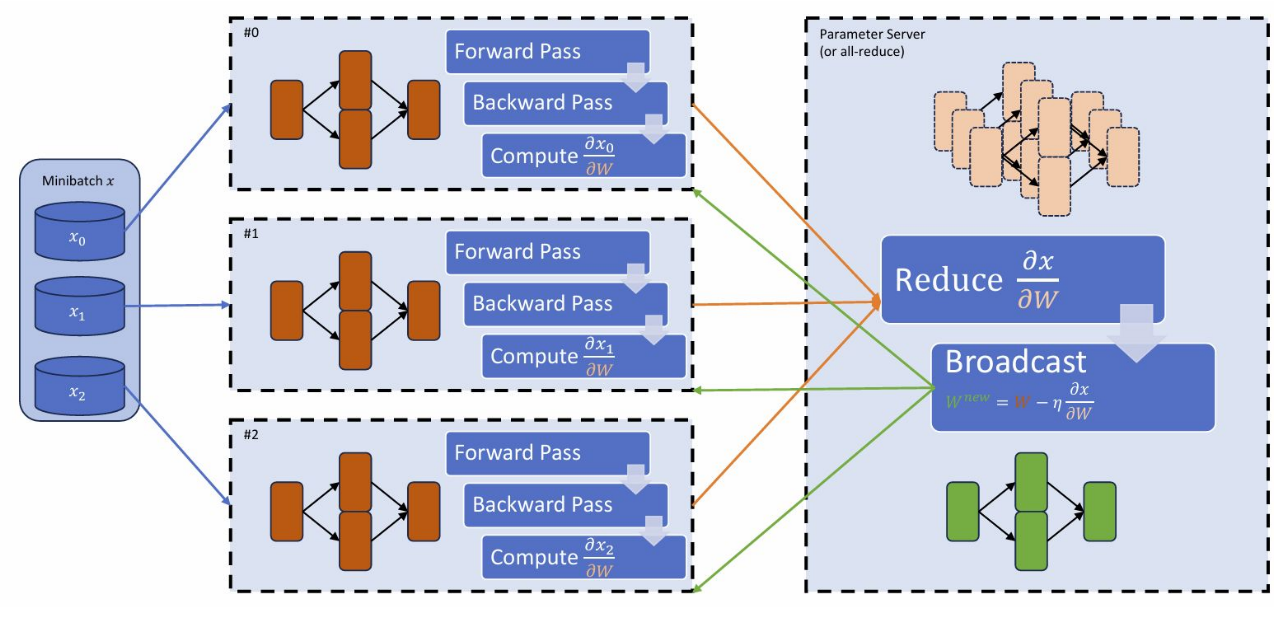
Reduce Scatter



https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/usage/collectives.html

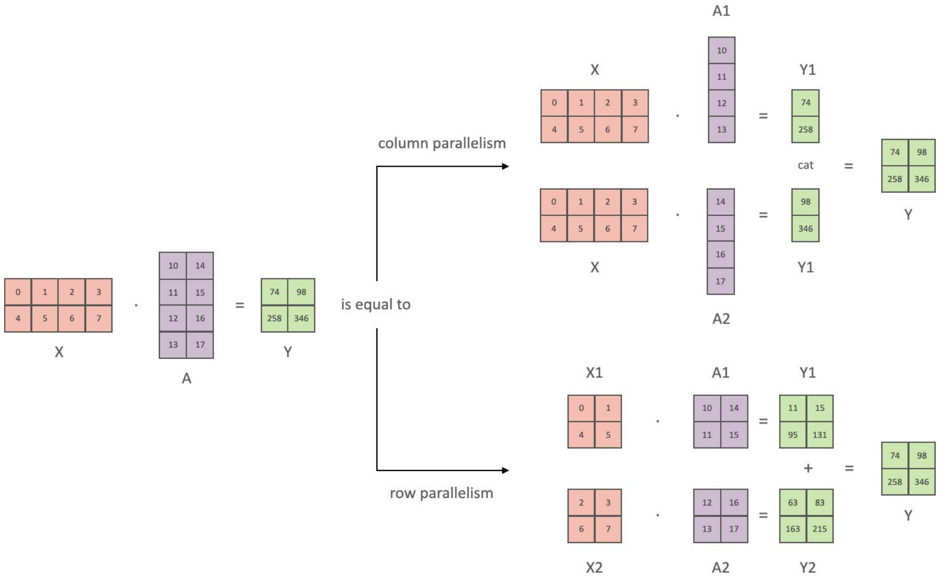
Preliminaries: Data Parallelism (DP)

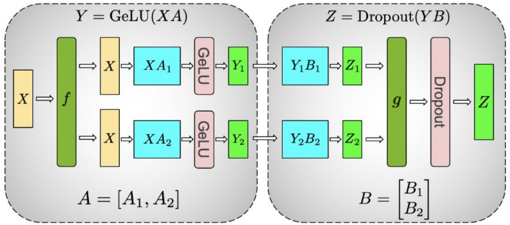
Most simple form of parallelism:



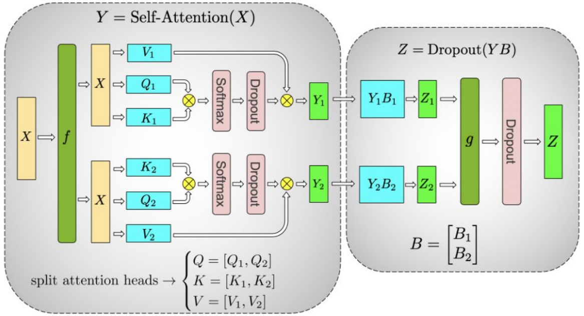
Preliminaries: Model Parallelism (MP)

Also known as **horizontal** model parallelism, or tensor parallelism:





(a) MLP



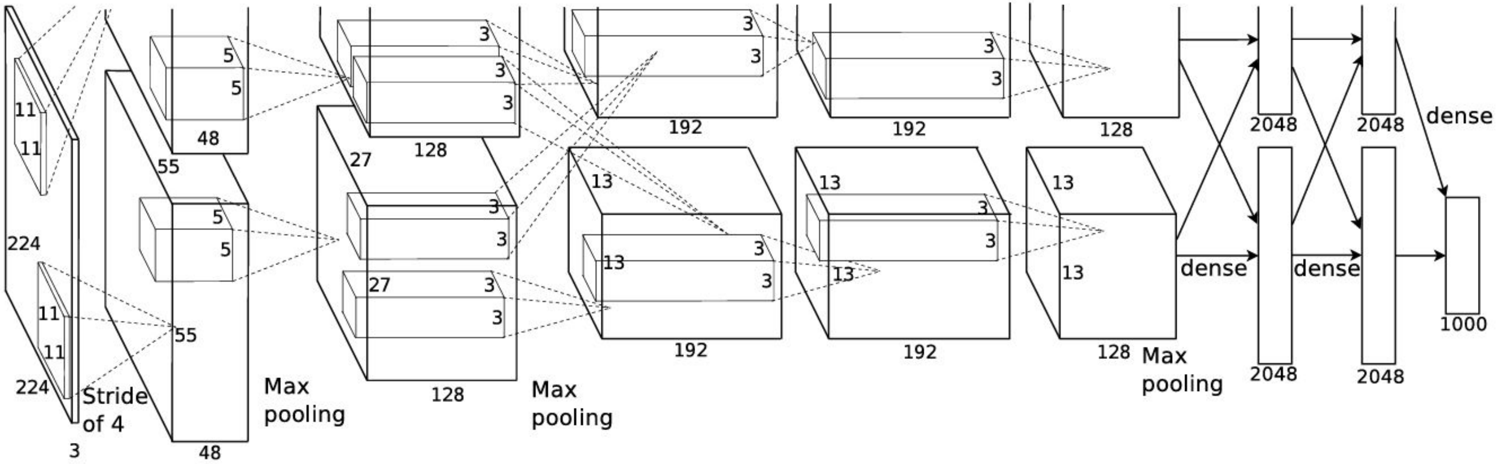
(b) Self attention

https://huggingface.co/docs/transformers/v4.48.1/perf_train_gpu_many

Mohammad Shoeybi et al. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. arXiv, 2019.

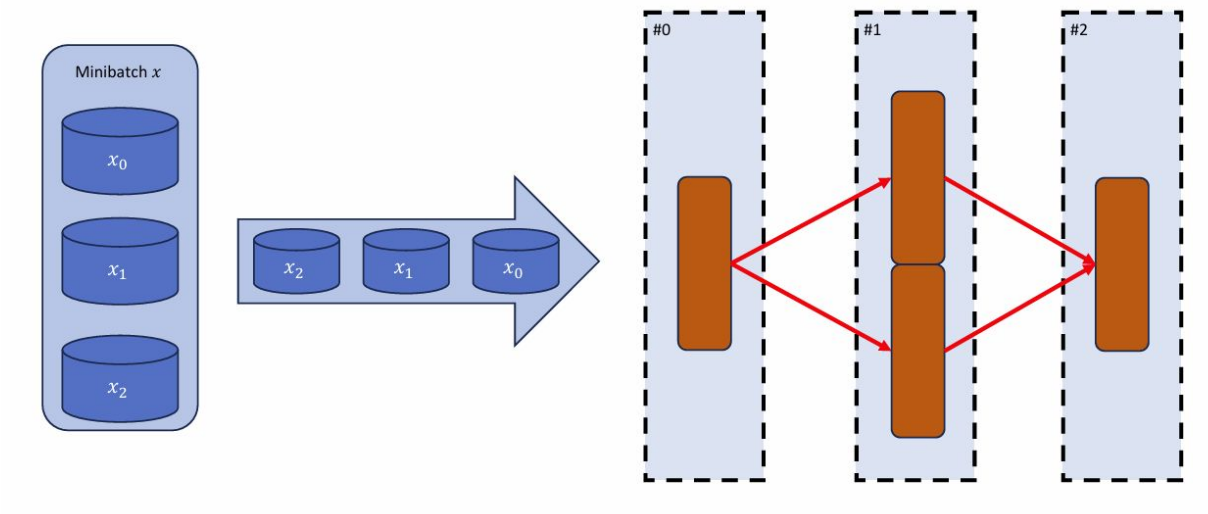
Preliminaries: Model Parallelism (MP)

Original architecture of AlexNet, a historical example of MP:



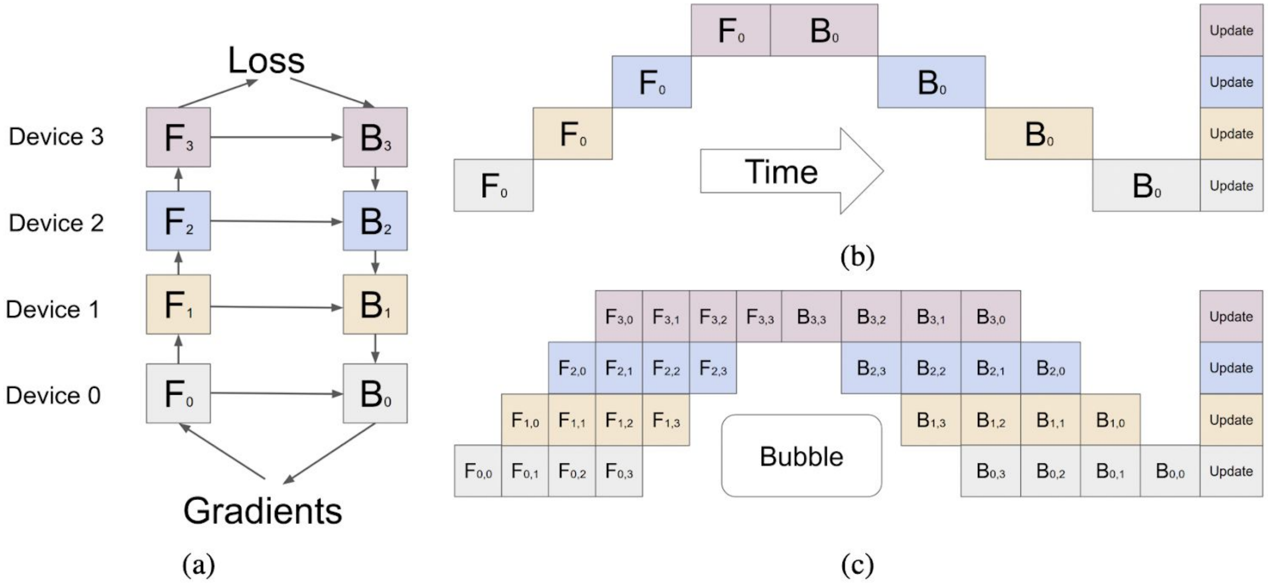
Preliminaries: Pipeline Parallelism (PP)

Also known as **vertical** model parallelism:



Preliminaries: Pipeline Parallelism (PP)

Mini-batches reduce the time idle, but still result in a pipeline bubble:



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

Limitations of DP, MP and PP

Data parallelism (DP):

- Doesn’t actually solve the memory issue
- Model parameters and all state are **replicated**

Model parallelism (MP):

- Certain layers force a **collective operation** (e.g., batch norm.)
- Requires extra care when designing the model architecture for the hardware and network

Pipeline parallelism (PP):

- Pipeline bubble results in idle resources
- Load imbalance caused by non-uniform execution time on each model partition

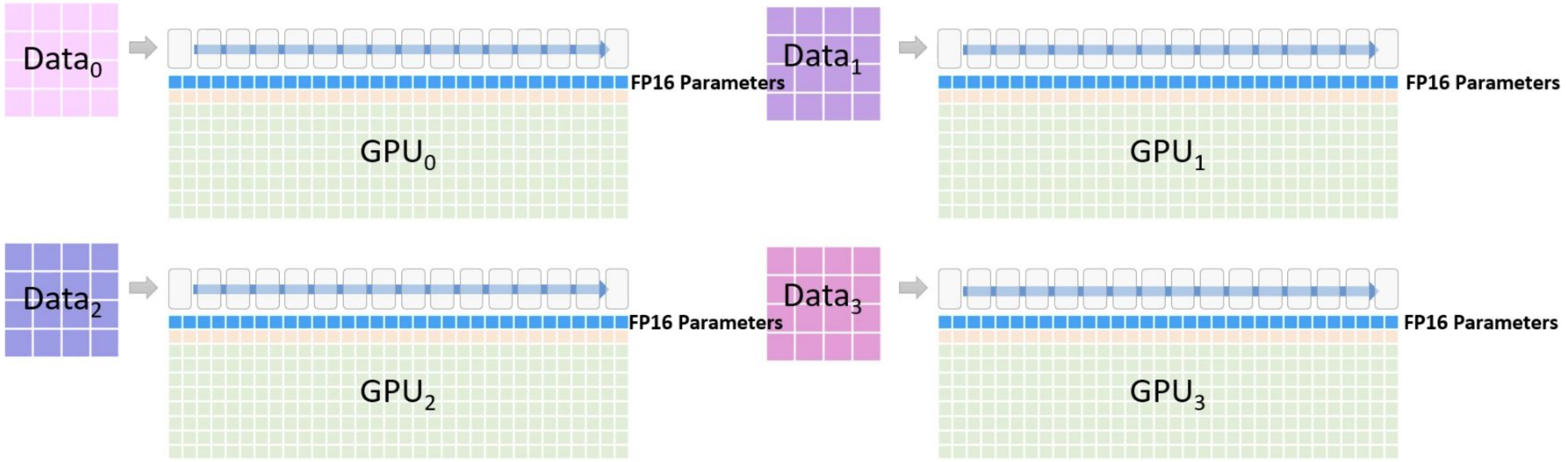
ZeRO

Cells under a transformer layer represent its GPU memory:

https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

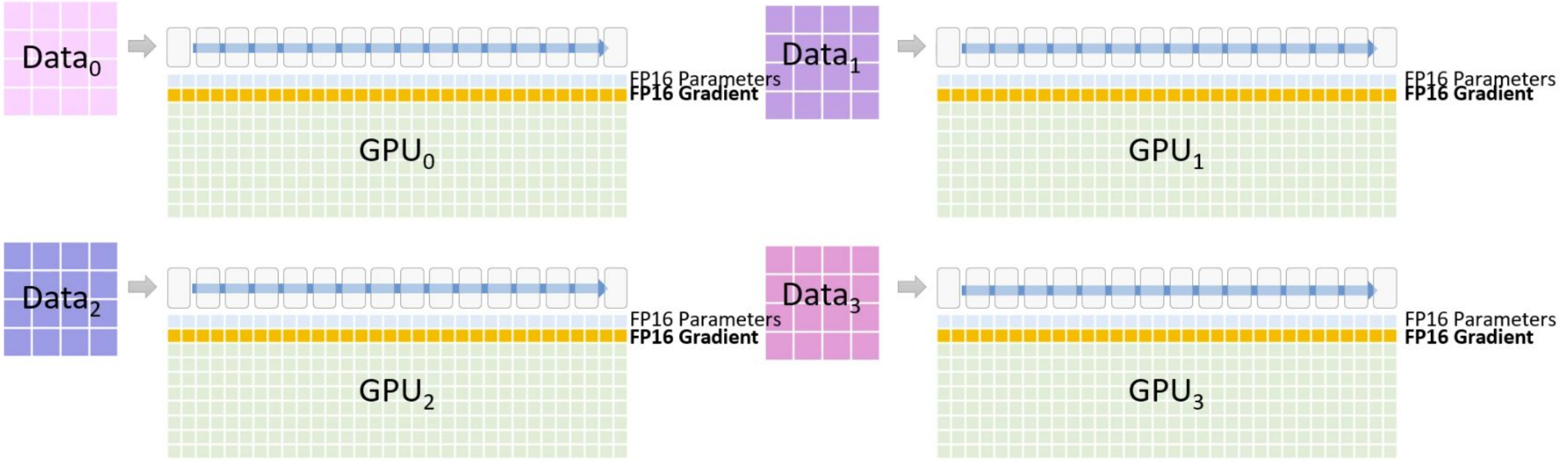
Blue cells represent model **parameters**:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

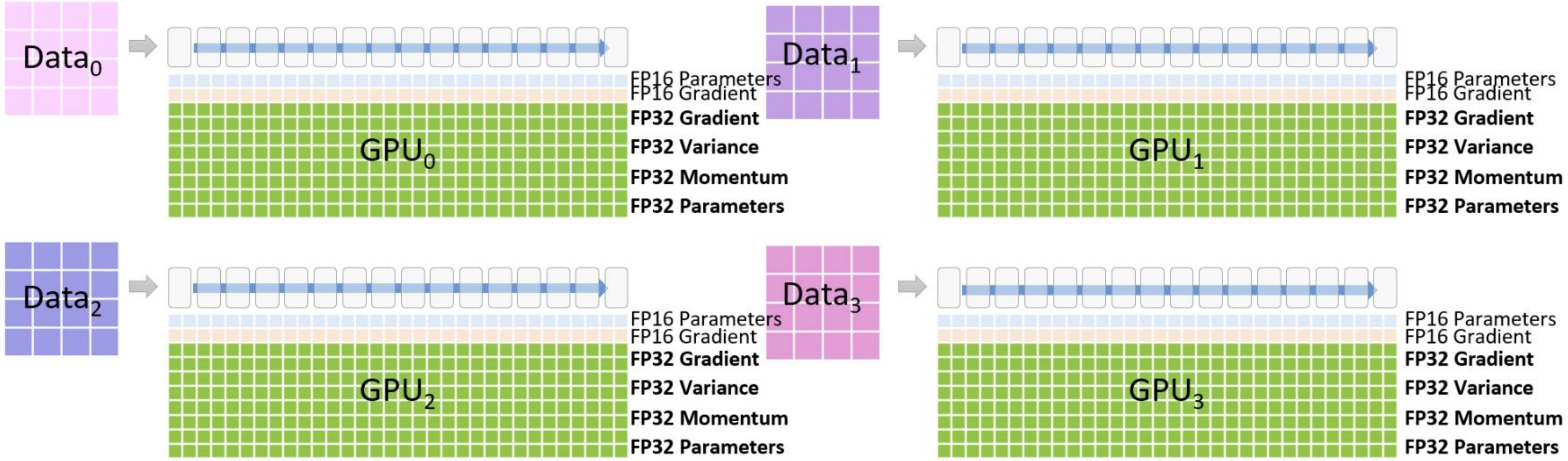
Orange cells represent **gradients**:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

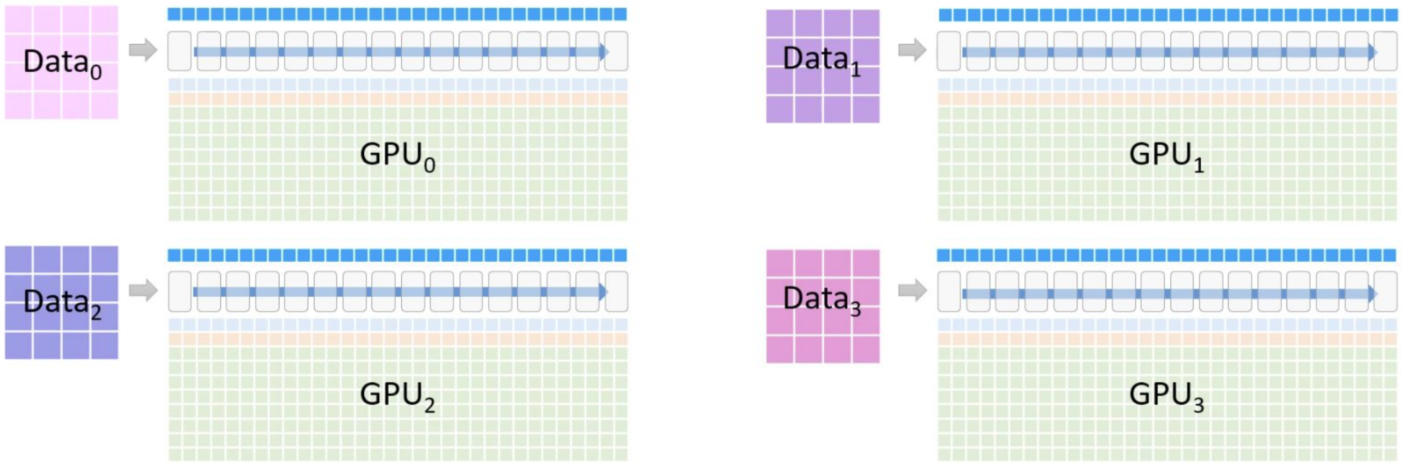
ZeRO

Green cells represent **optimizer state** (including temp. space for updated parameters):



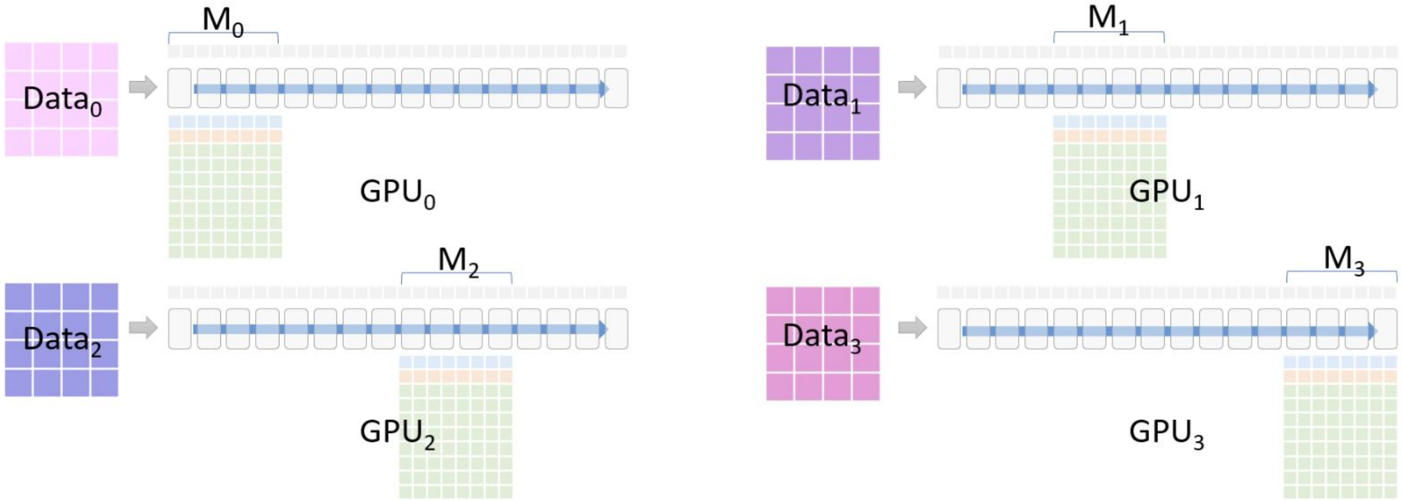
ZeRO

Blue cells along the top represent **activations**:



ZeRO

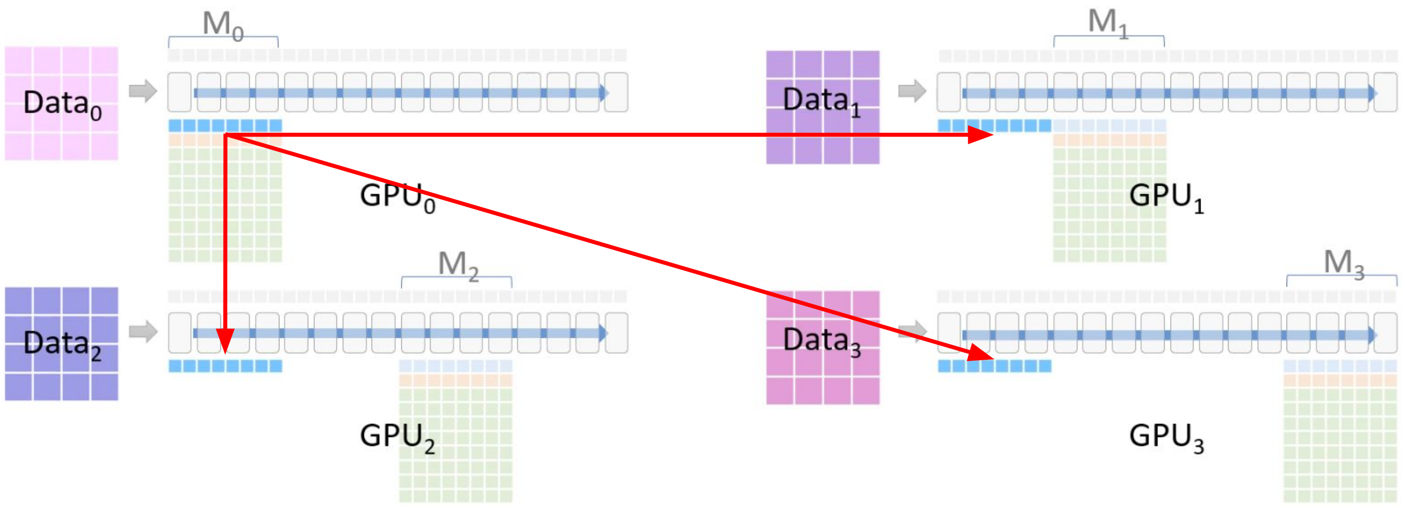
Model is **vertically partitioned** across each device:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

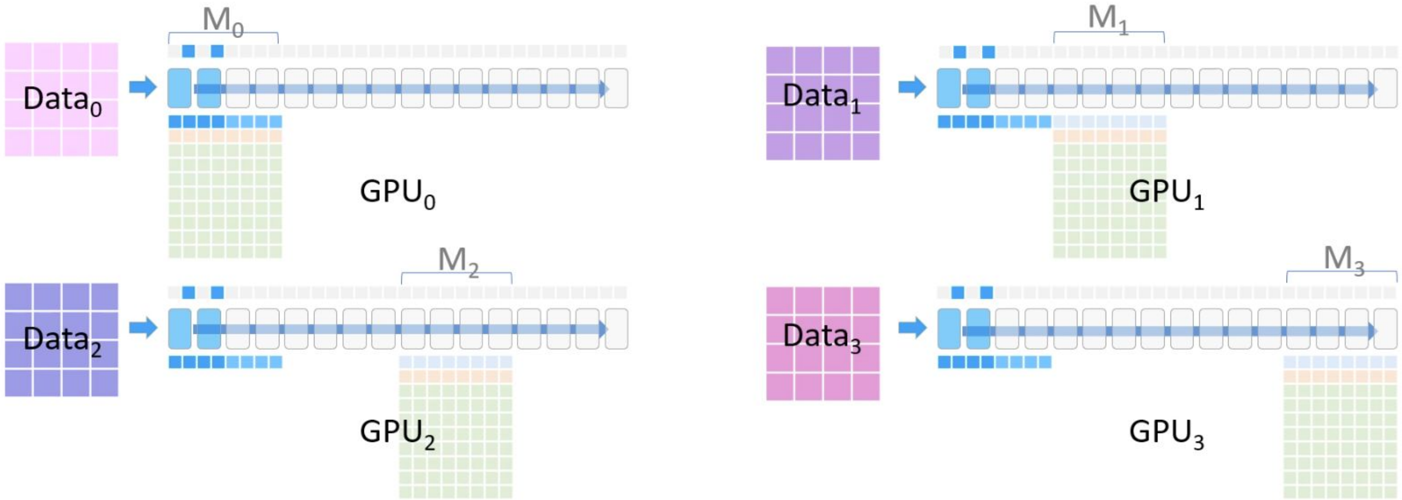
GPU$_0$ **broadcasts** its model partition:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

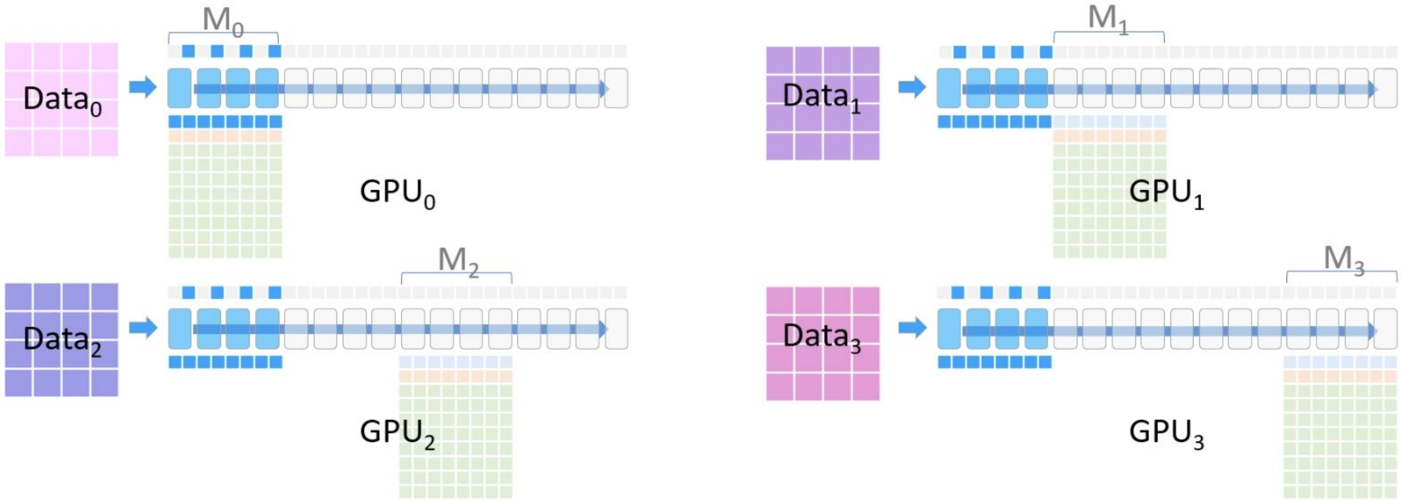
Each device evaluates M_0 on its own data:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

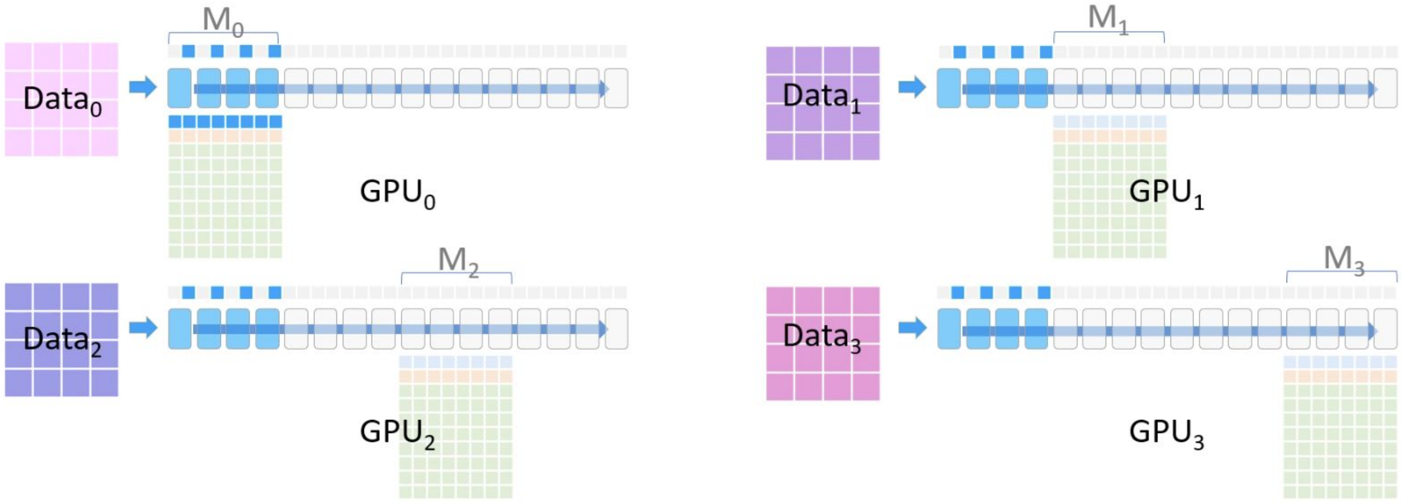
ZeRO

Each device evaluates M_0 on its own data:



ZeRO

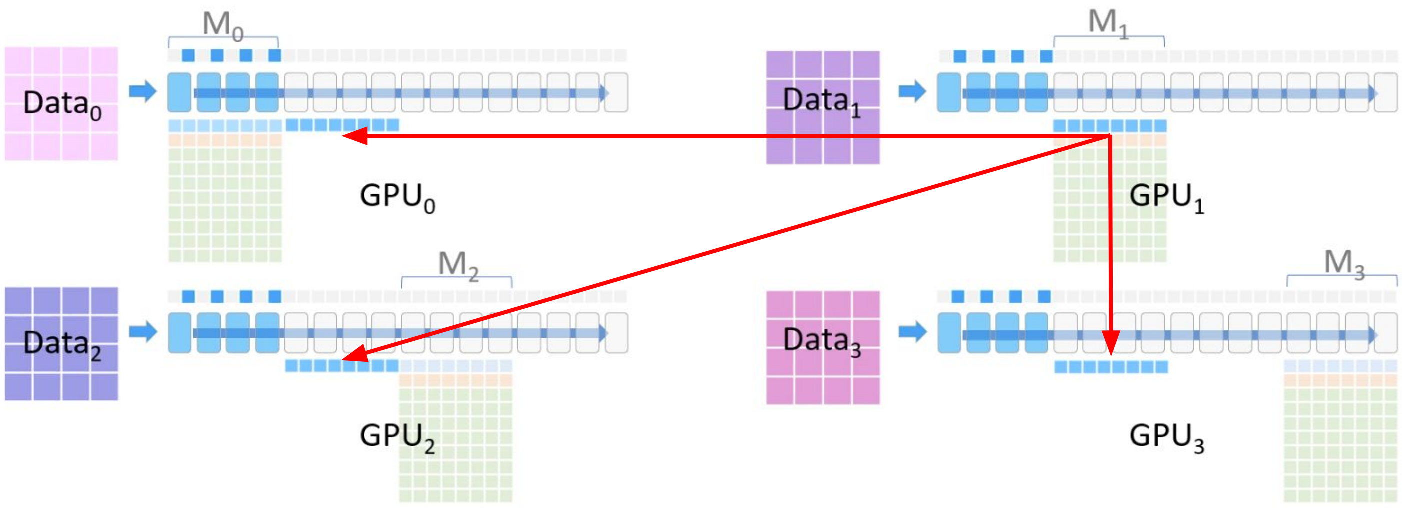
After the forward pass on M_0, all devices except GPU_0 delete M_0:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

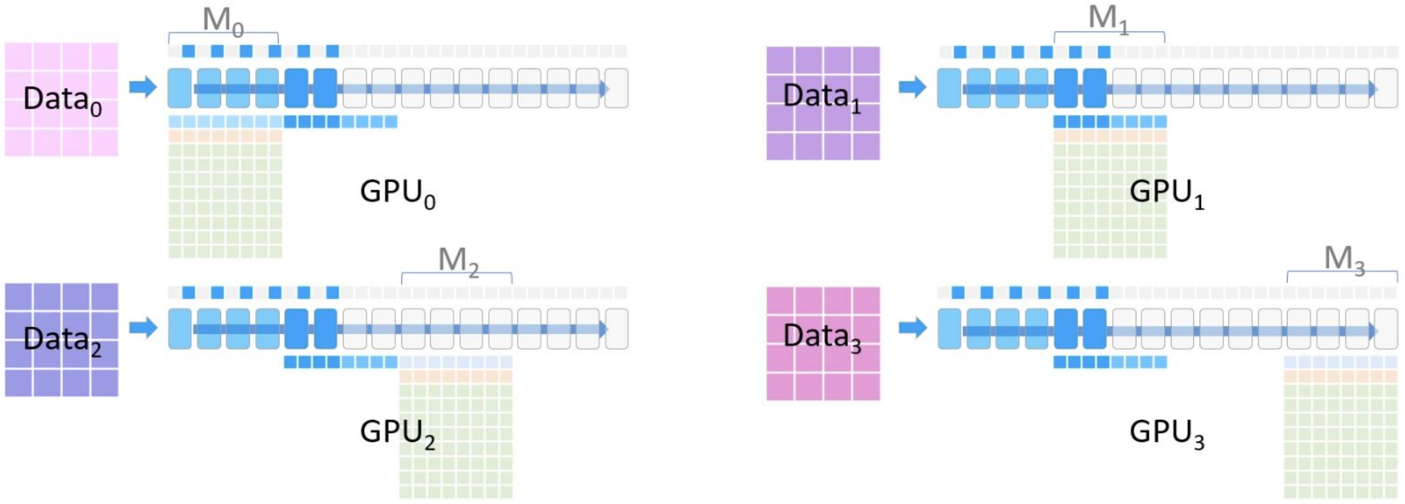
GPU$_1$ **broadcasts** its model partition:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

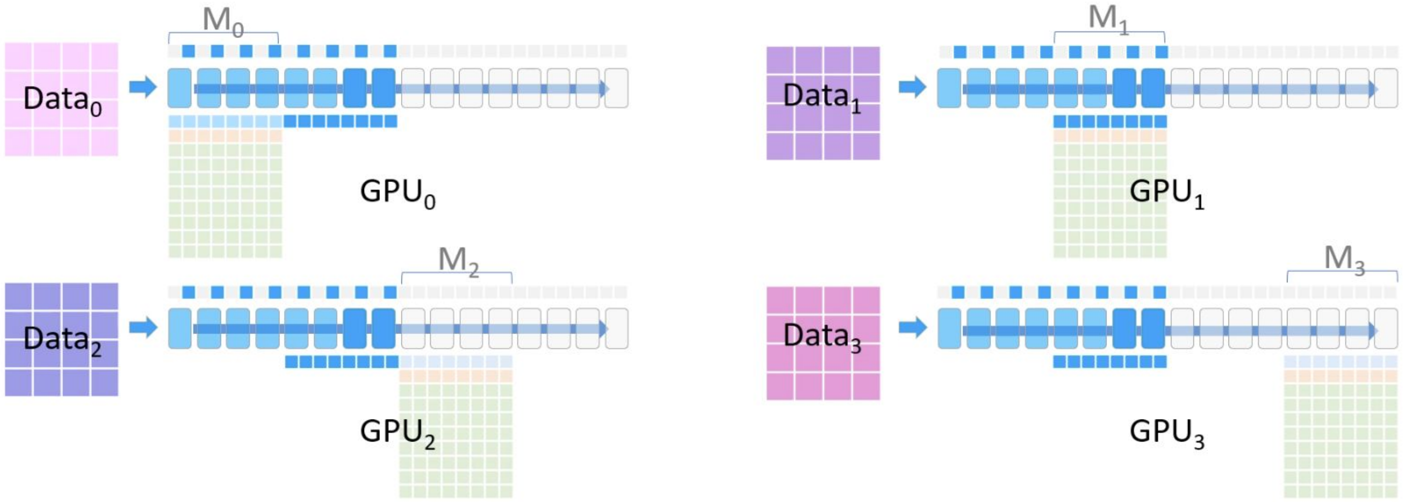
The forward pass continues on M_1:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

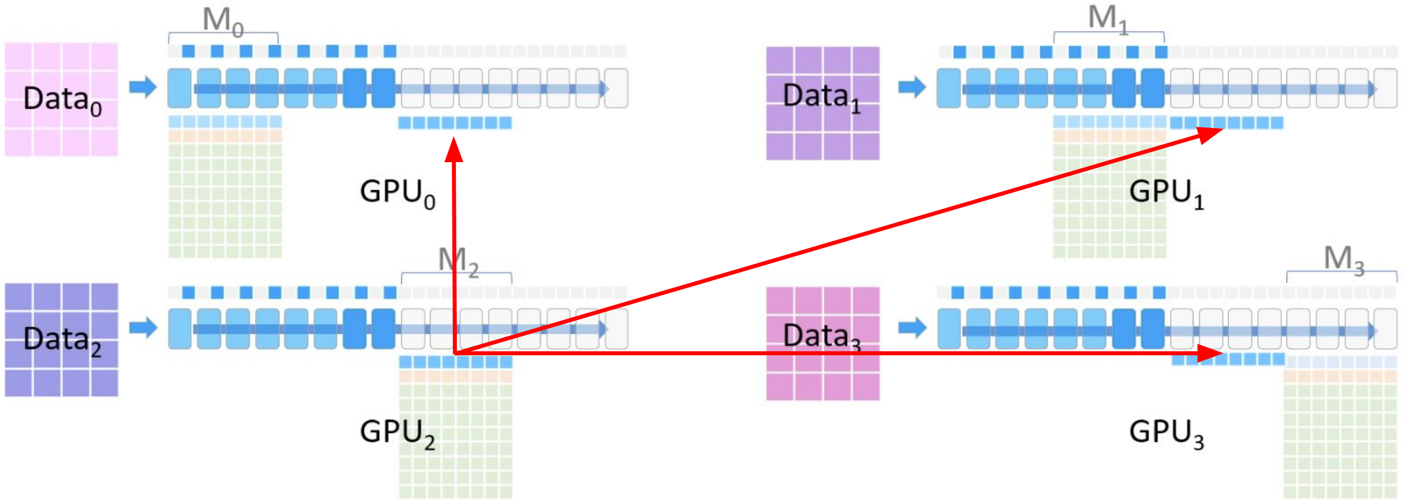
The forward pass continues on M_1:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

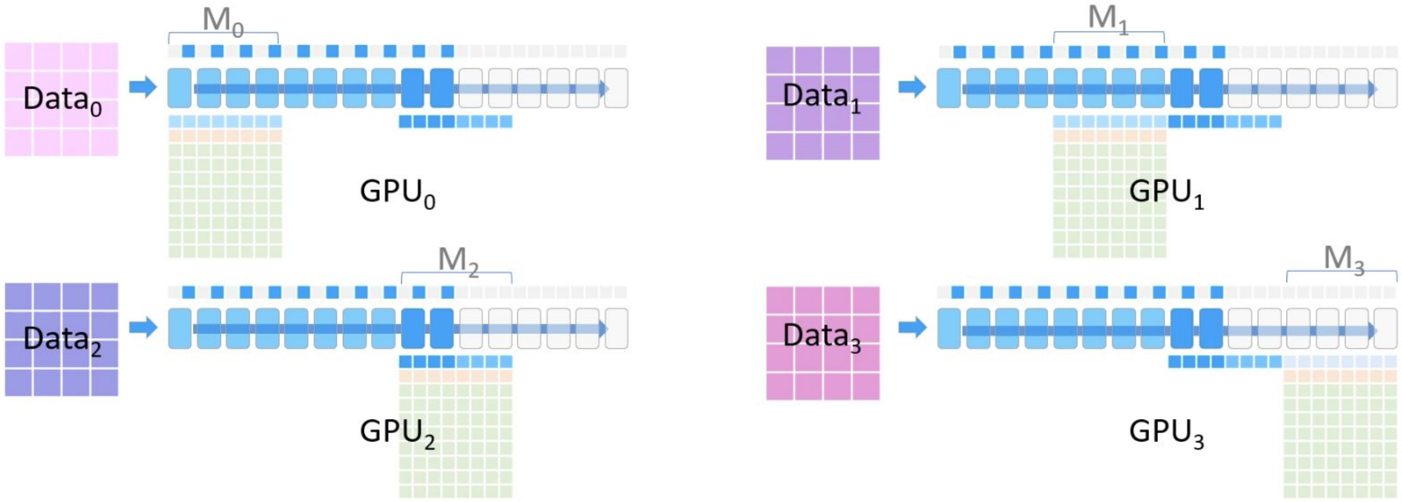
ZeRO

GPU_2 **broadcasts** its model partition:



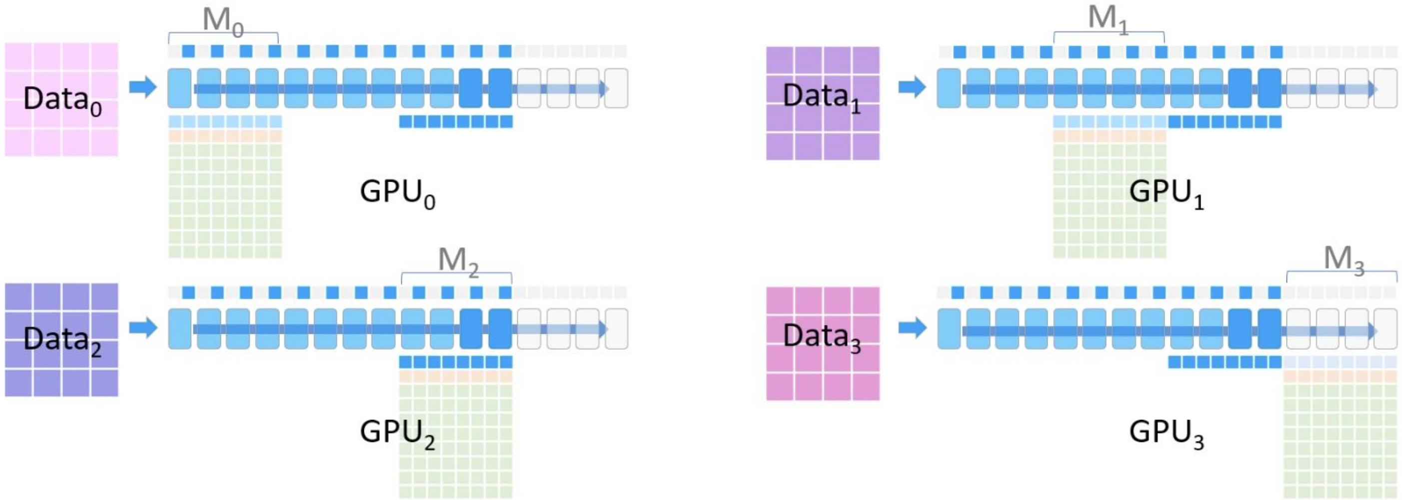
ZeRO

The forward pass continues on M_2:



ZeRO

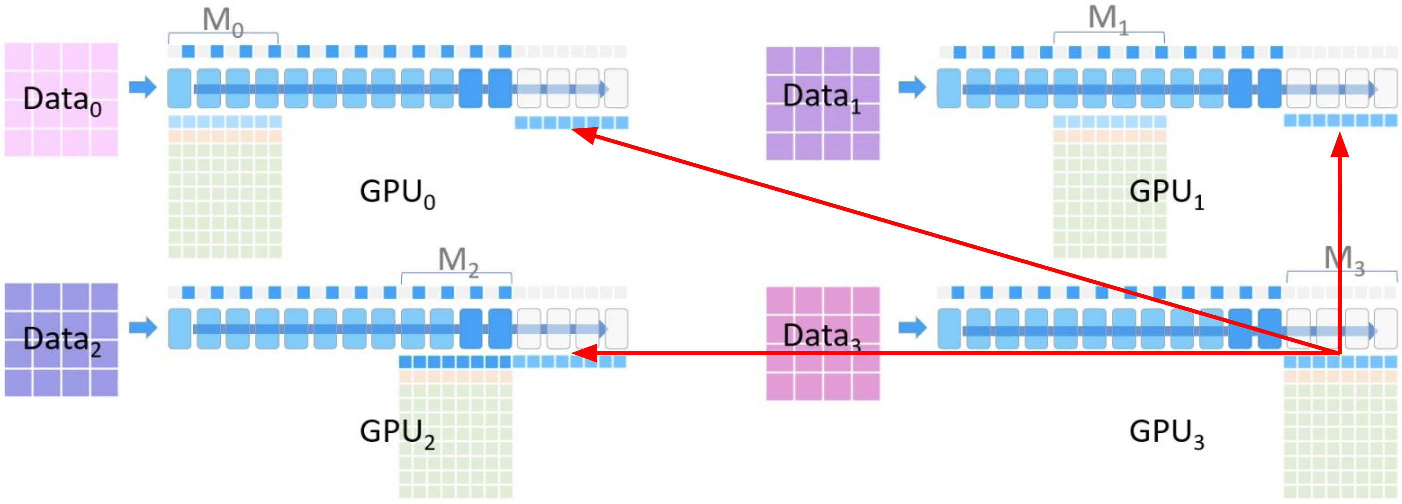
The forward pass continues on M_2:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

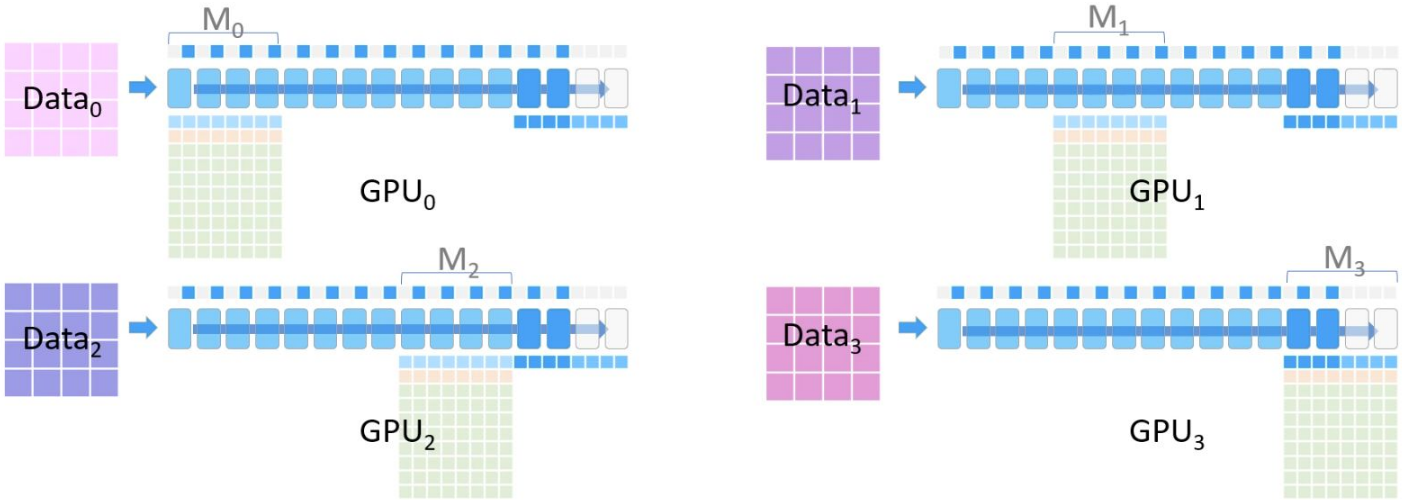
GPU$_3$ **broadcasts** its model partition:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

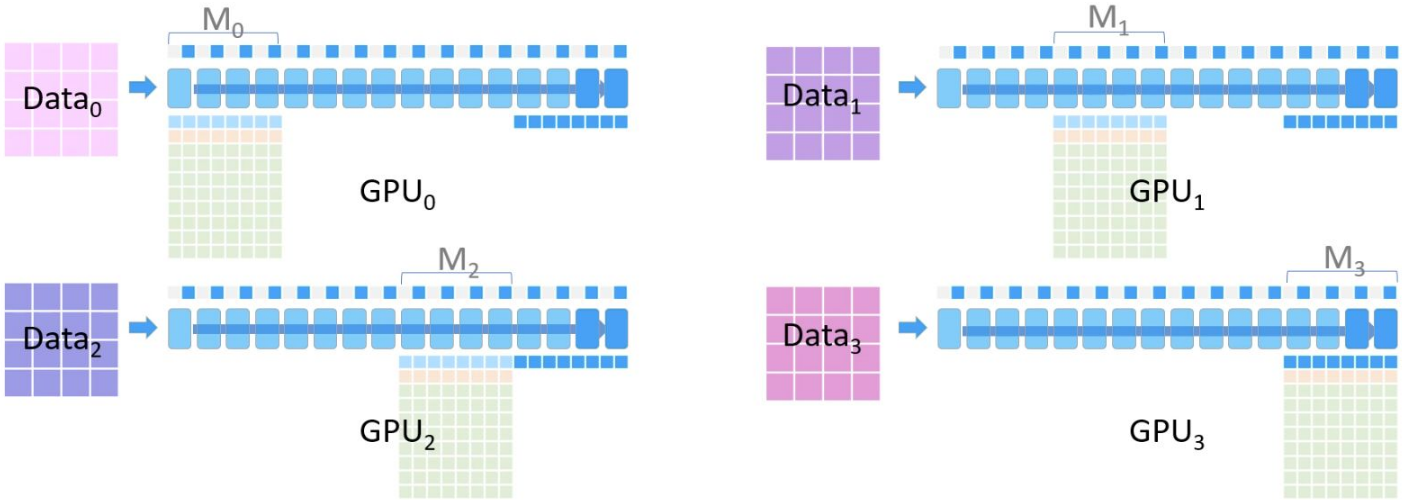
ZeRO

The forward pass continues on M_3:



ZeRO

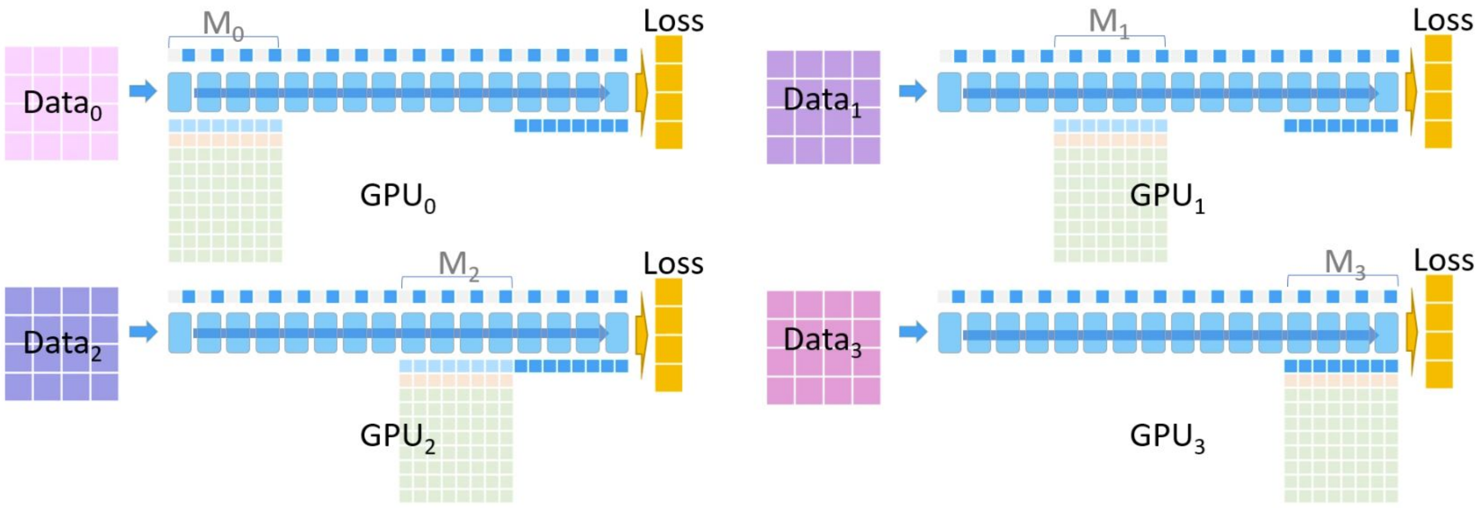
The forward pass continues on M_3:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

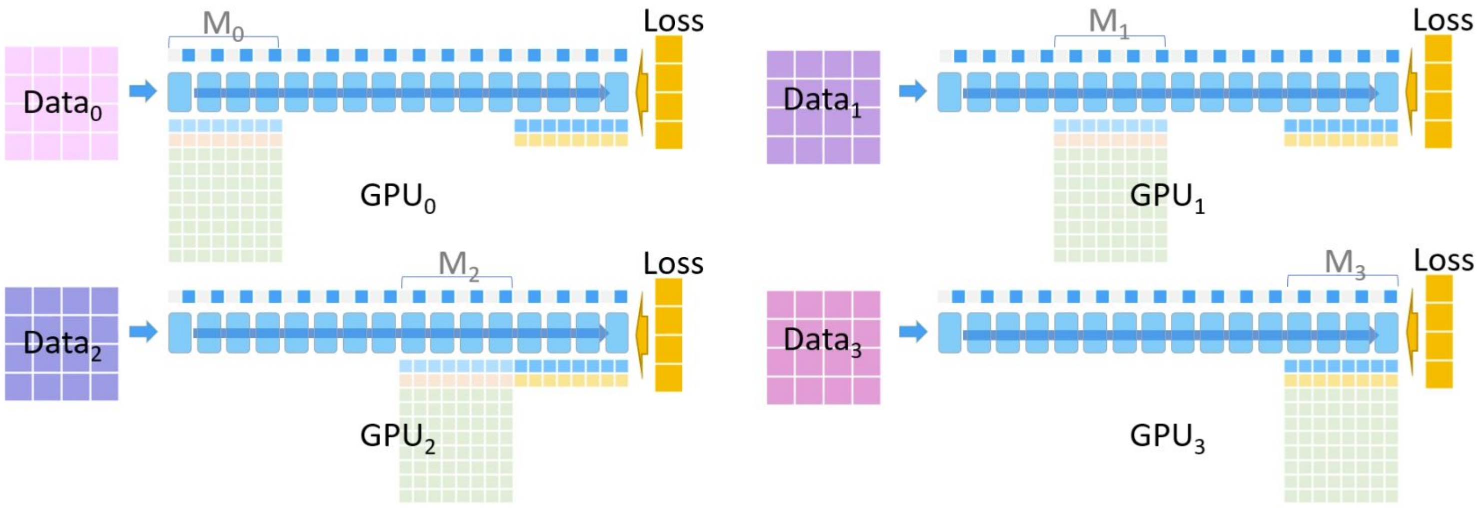
The forward pass is complete. In parallel, each device calculates the loss for its data:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

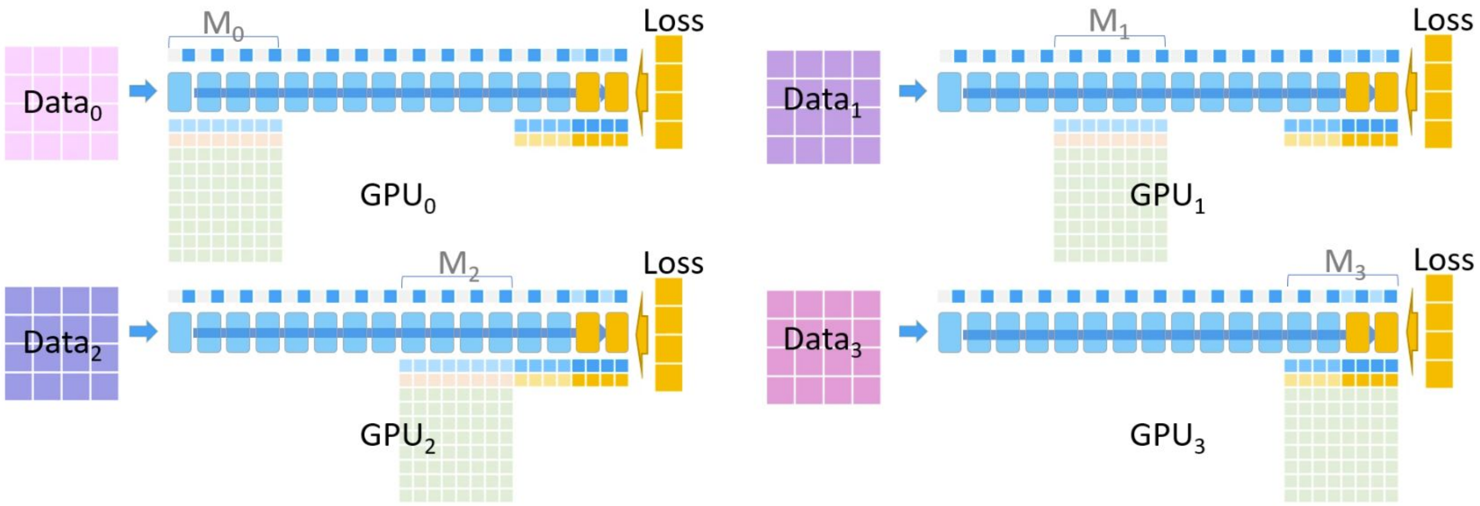
ZeRO

The backward pass starts. All devices allocate space for gradients (shown in pale yellow):



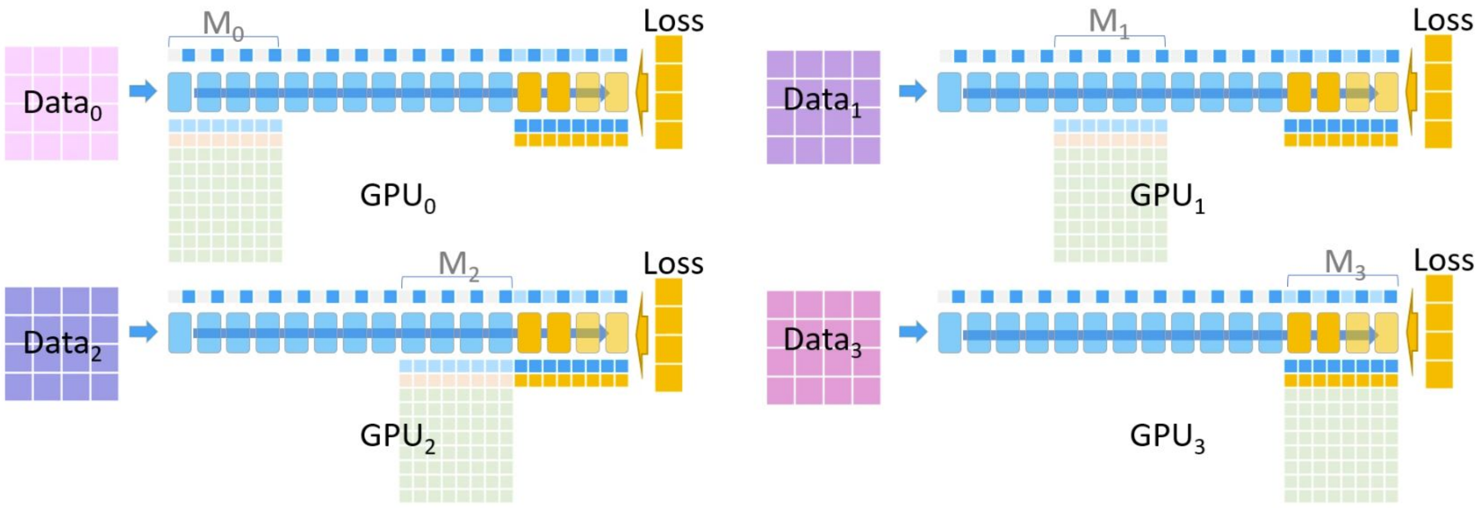
ZeRO

The backward pass proceeds on M_3:



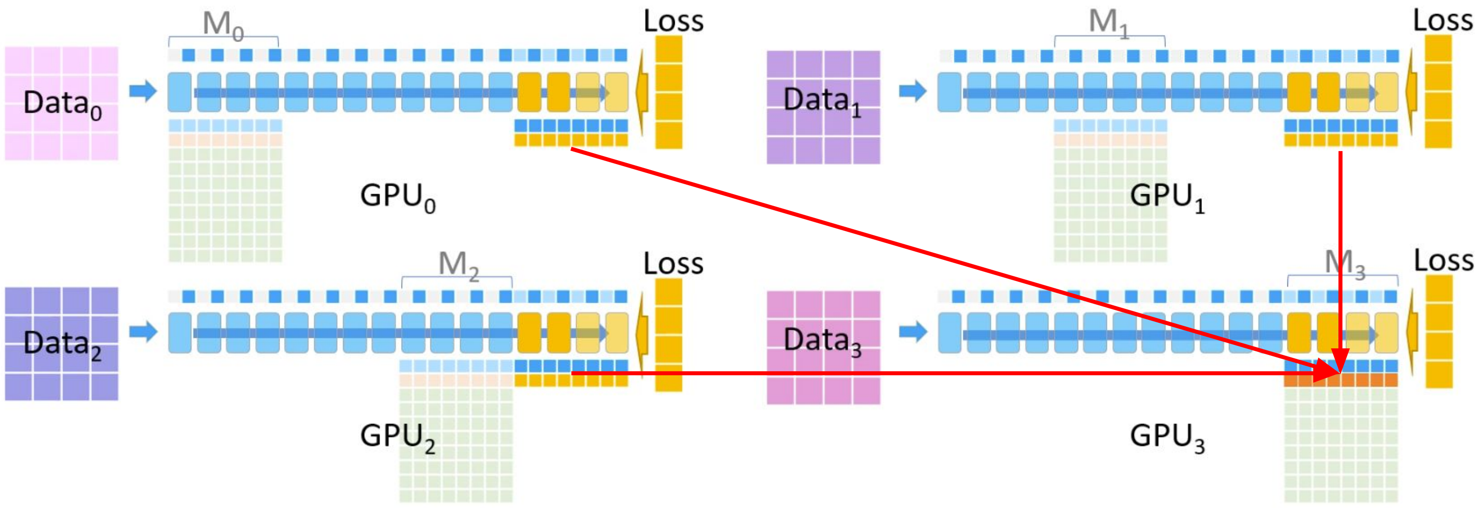
ZeRO

The backward pass proceeds on M_3:



ZeRO

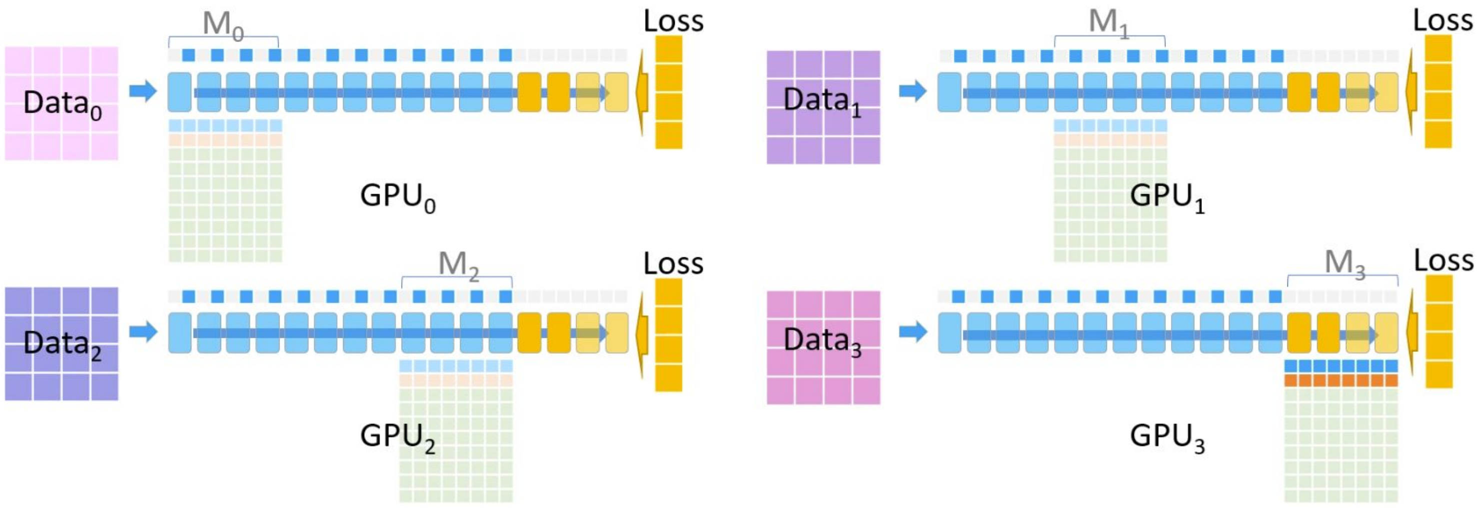
GPU$_3$ performs a **reduce** operation, holding the gradients for M$_3$ across all data:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

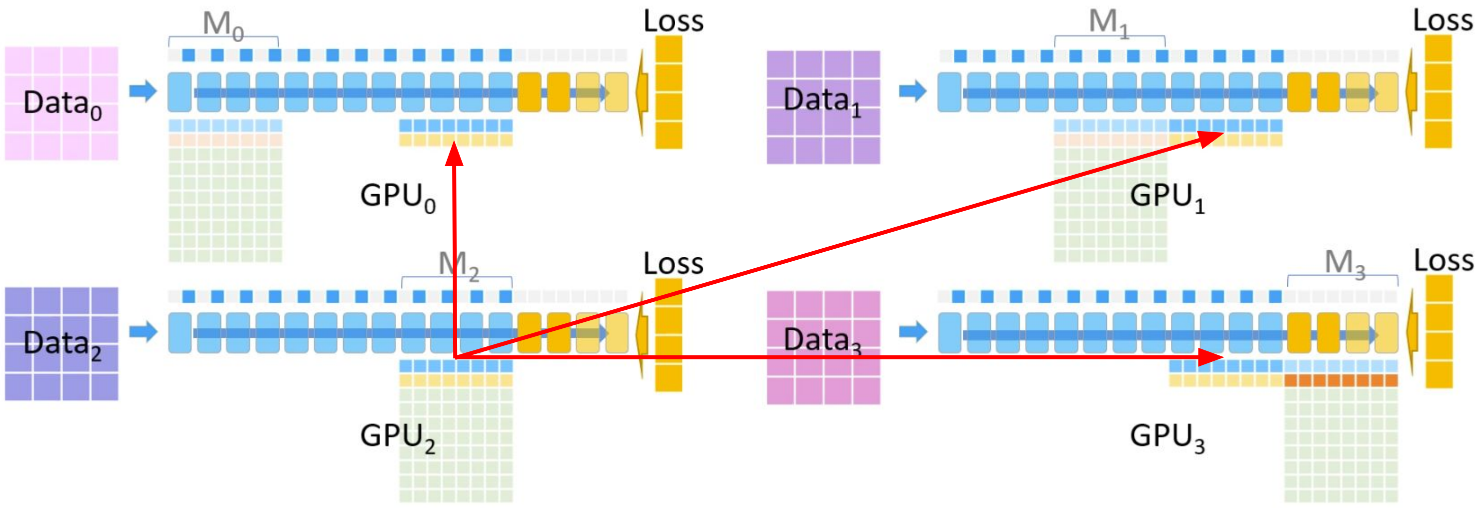
All devices except GPU_3 delete M_3 and its gradients, and all devices delete activations for M_3:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

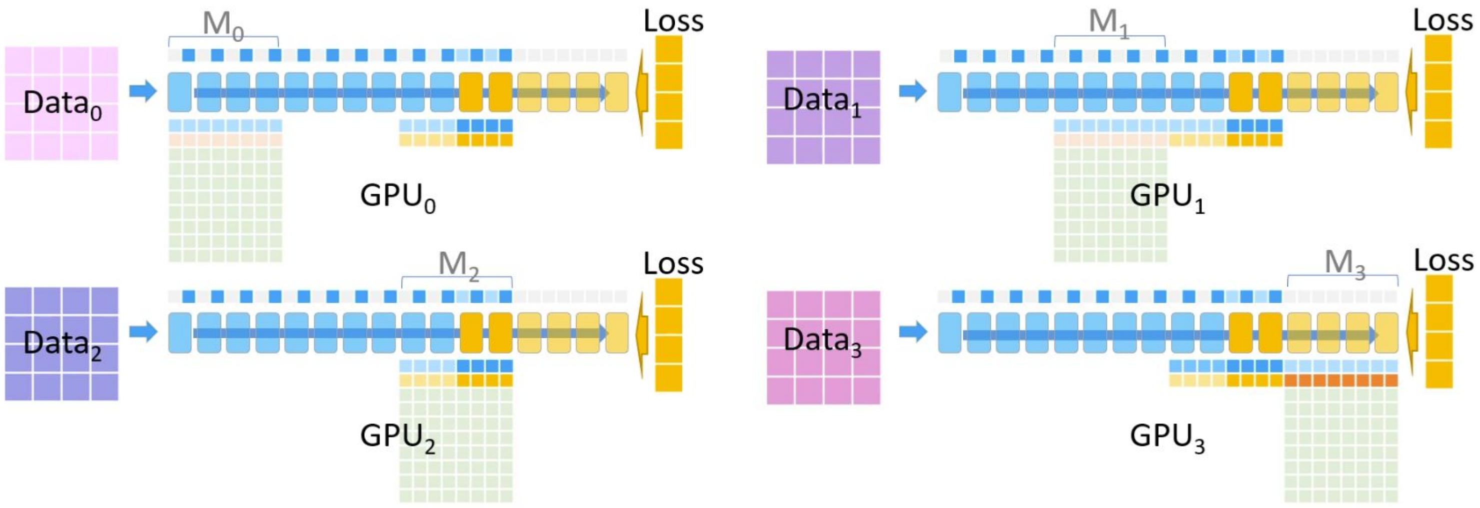
GPU$_2$ **broadcasts** its model partition:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

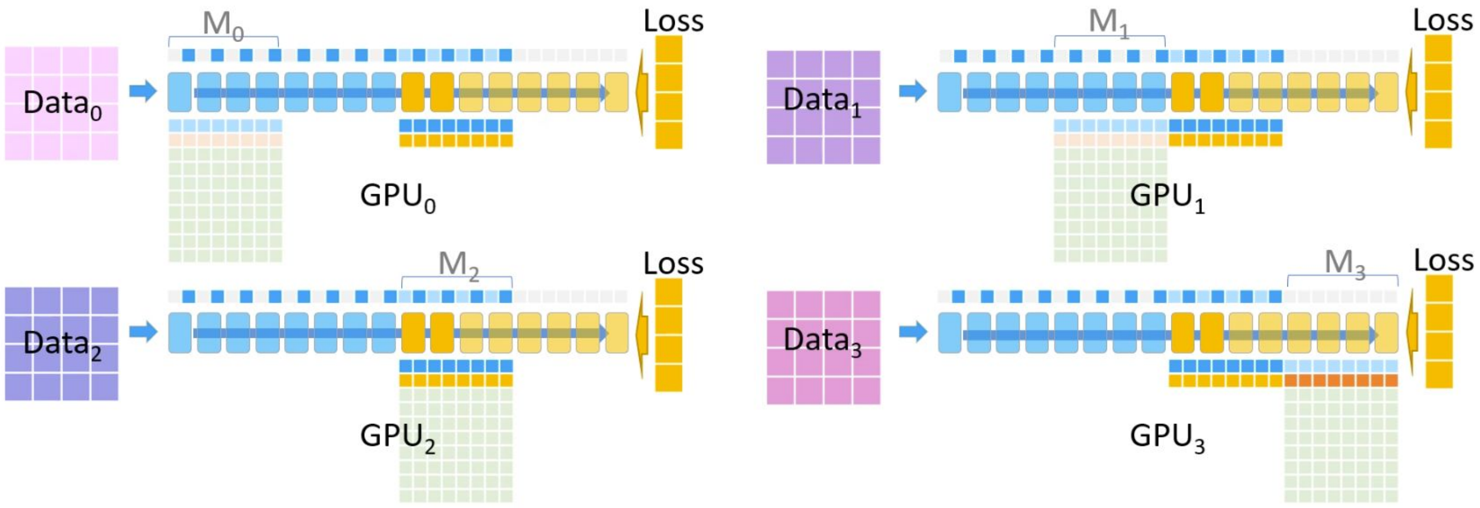
The backward pass continues on M_2:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

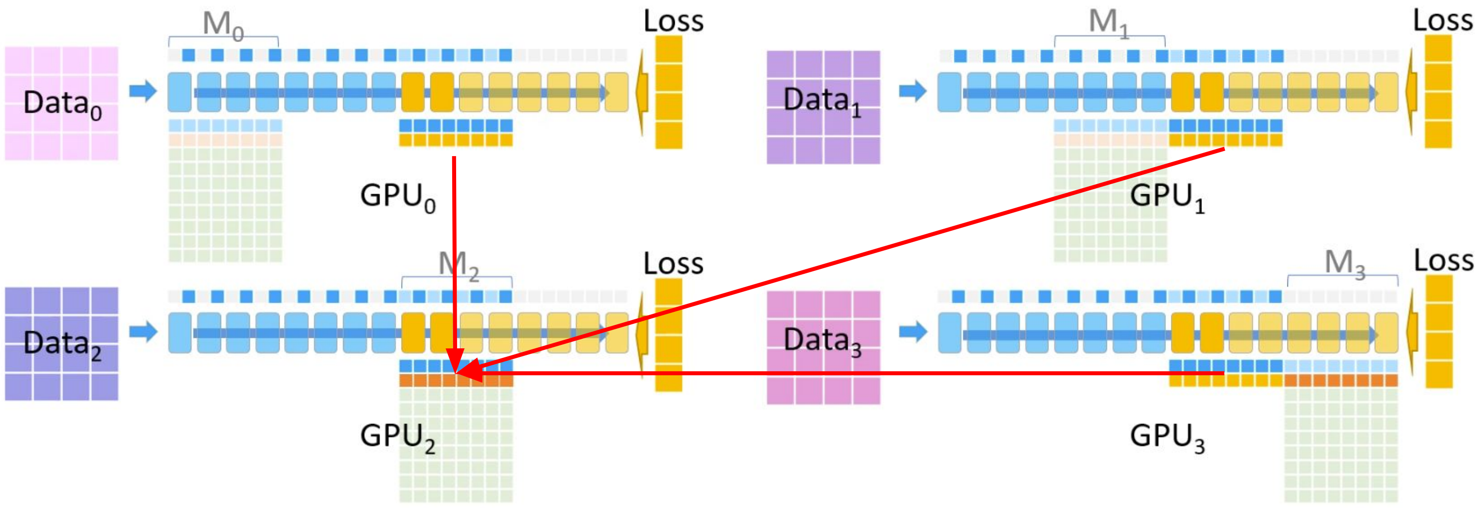
The backward pass continues on M_2:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

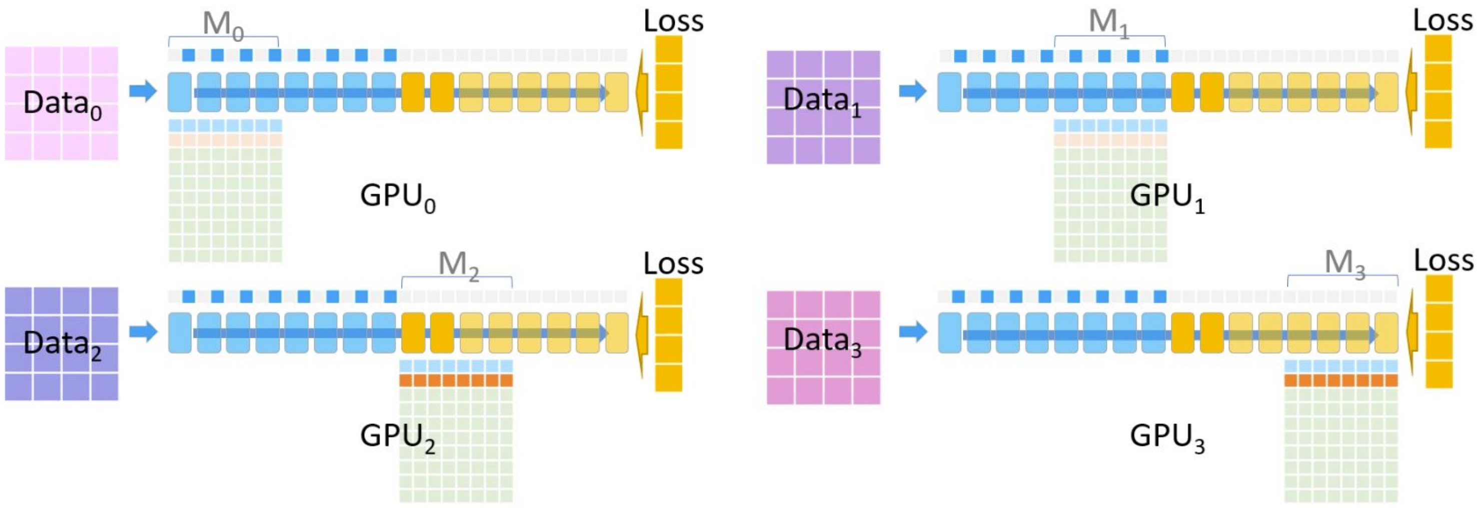
GPU_2 performs a **reduce** operation, holding the gradients for M_2 across all data:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

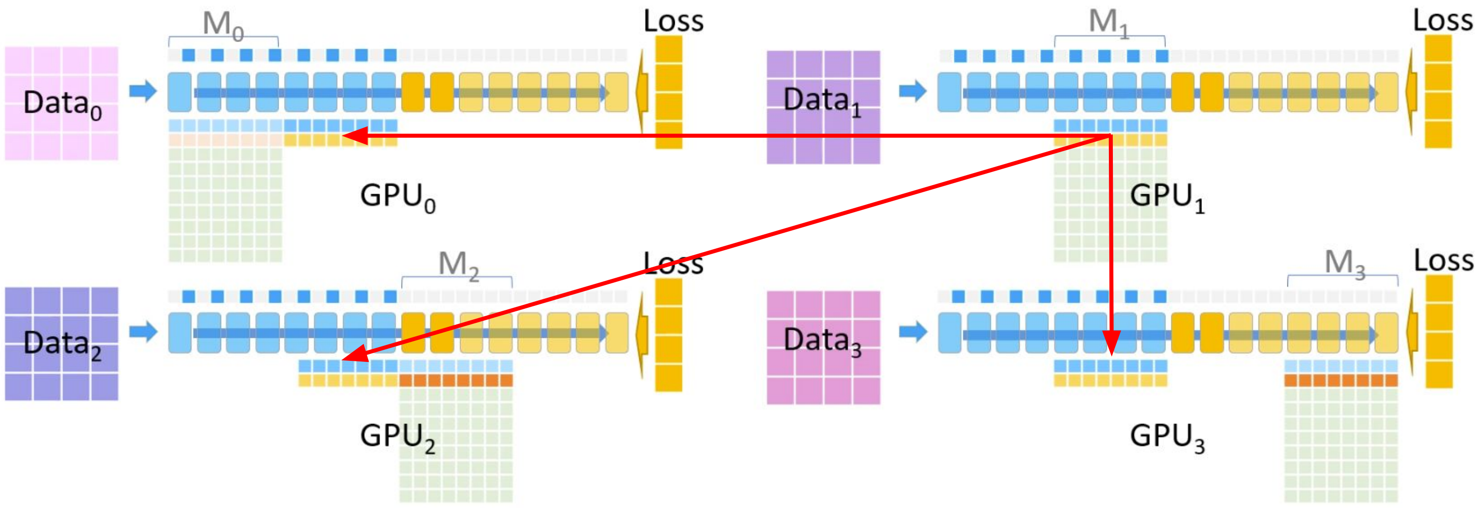
ZeRO

All devices except GPU_2 delete M_2 and its gradients, and all devices delete activations for M_2:



ZeRO

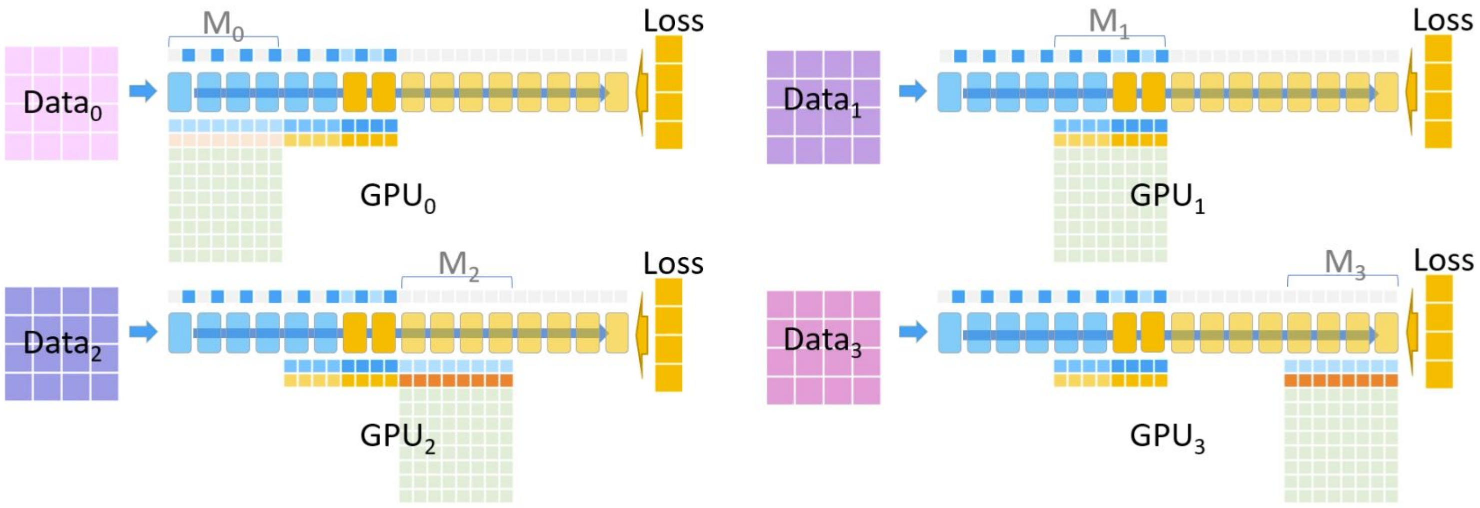
GPU_1 **broadcasts** its model partition:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

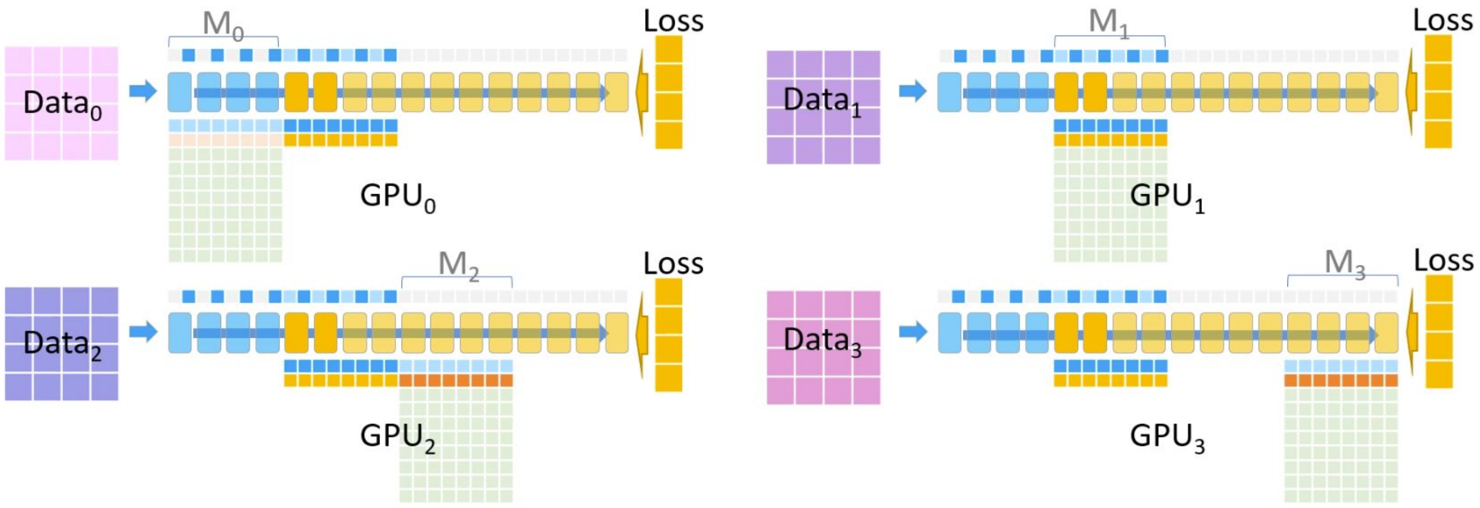
ZeRO

The backward pass continues on M_1:



ZeRO

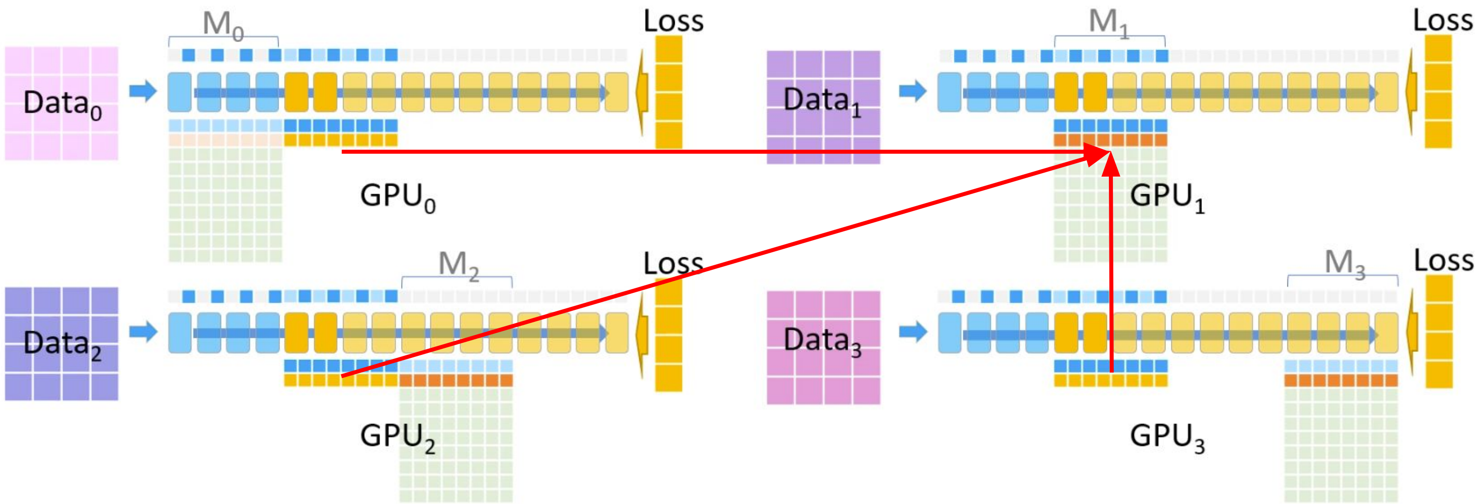
The backward pass continues on M_1:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

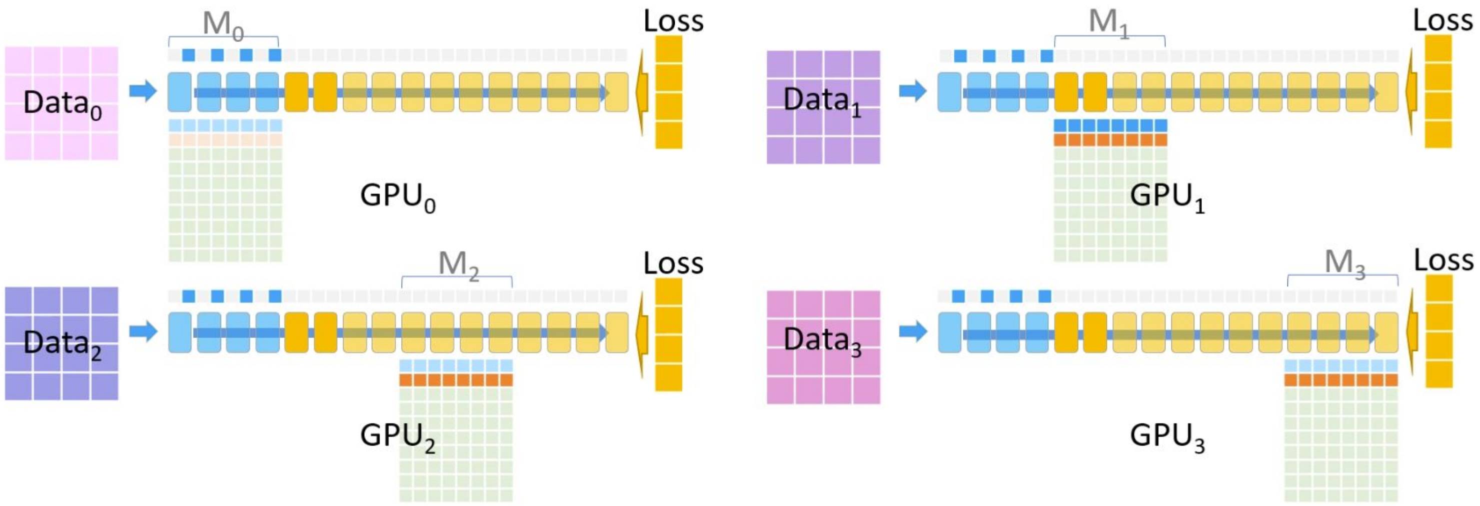
GPU_1 performs a **reduce** operation, holding the gradients for M_1 across all data:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

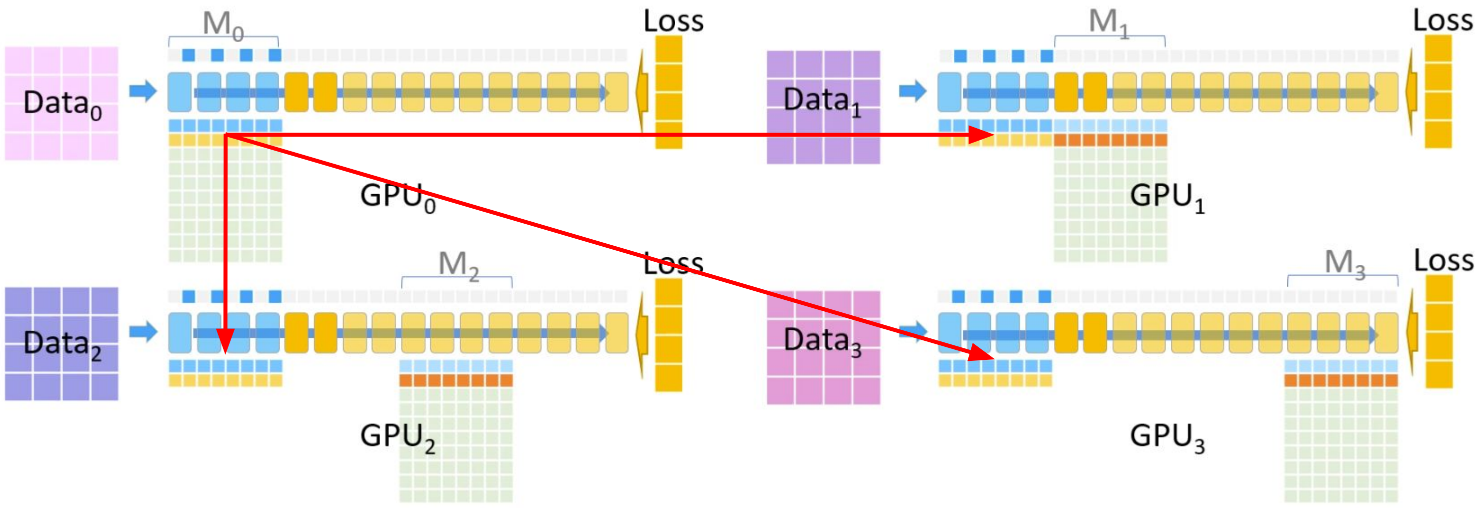
ZeRO

All devices except GPU_1 delete M_1 and its gradients, and all devices delete activations for M_1:



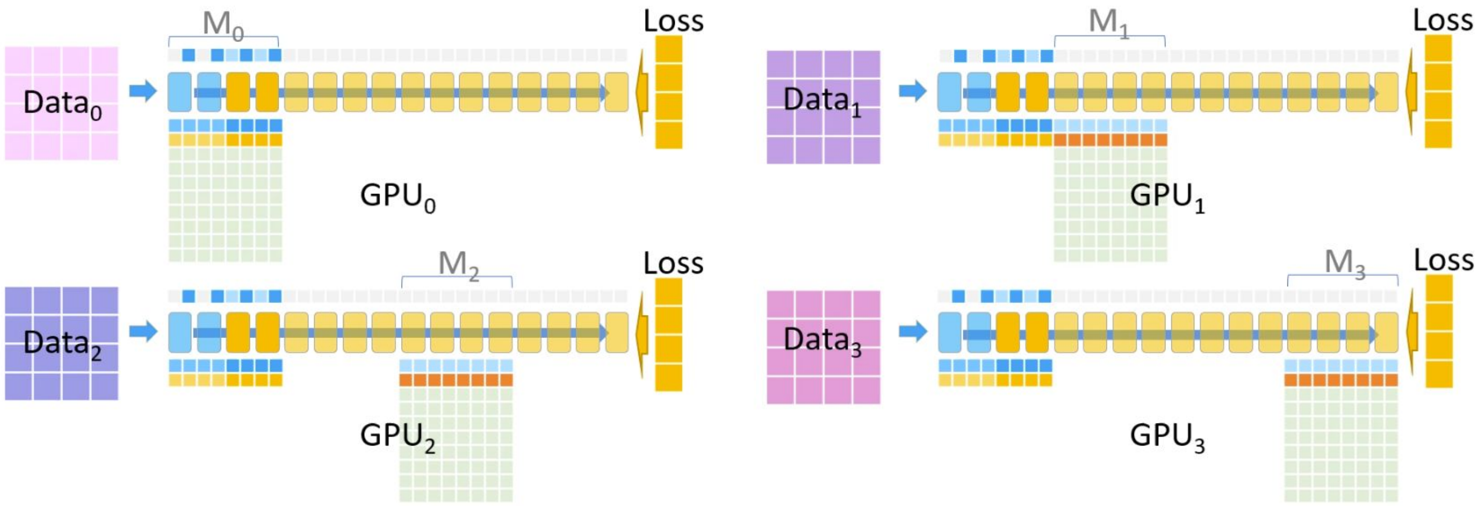
ZeRO

GPU$_0$ **broadcasts** its model partition:



ZeRO

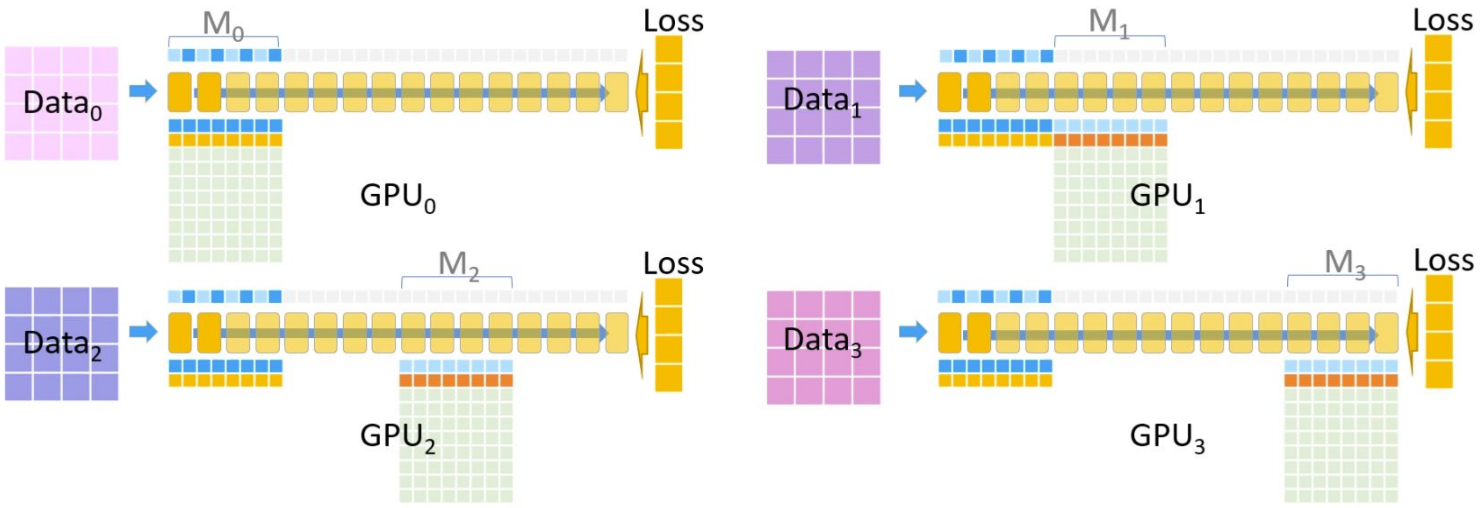
The backward pass continues on M_0:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

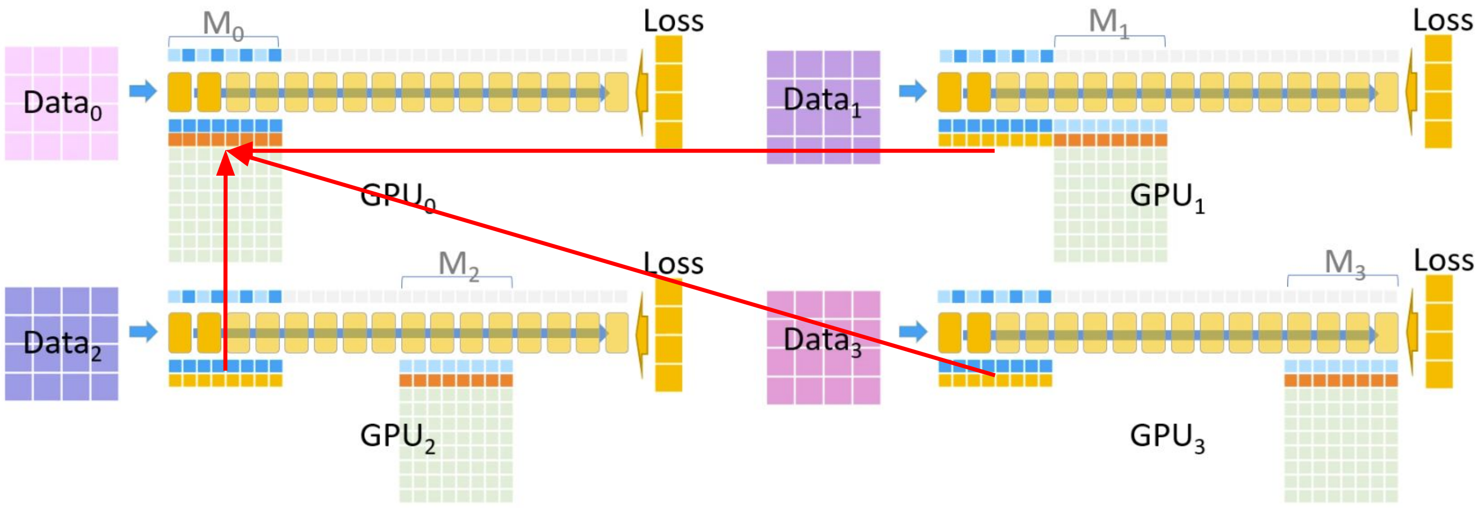
The backward pass continues on M_0:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

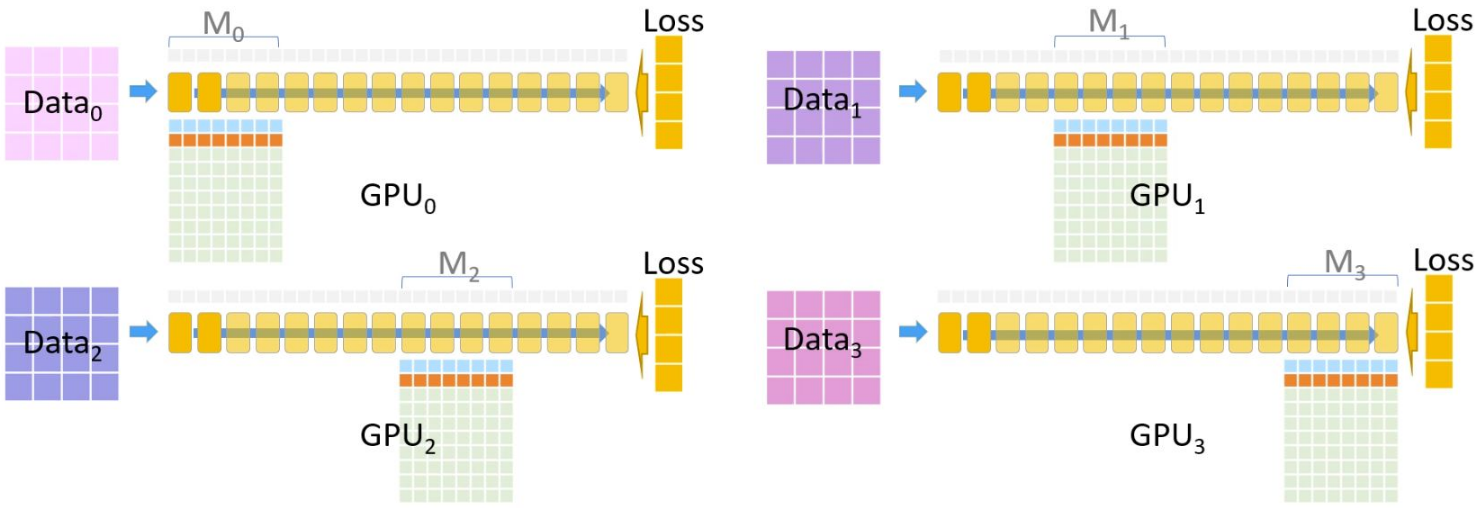
GPU$_0$ performs a **reduce** operation, holding the gradients for M$_0$ across all data:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

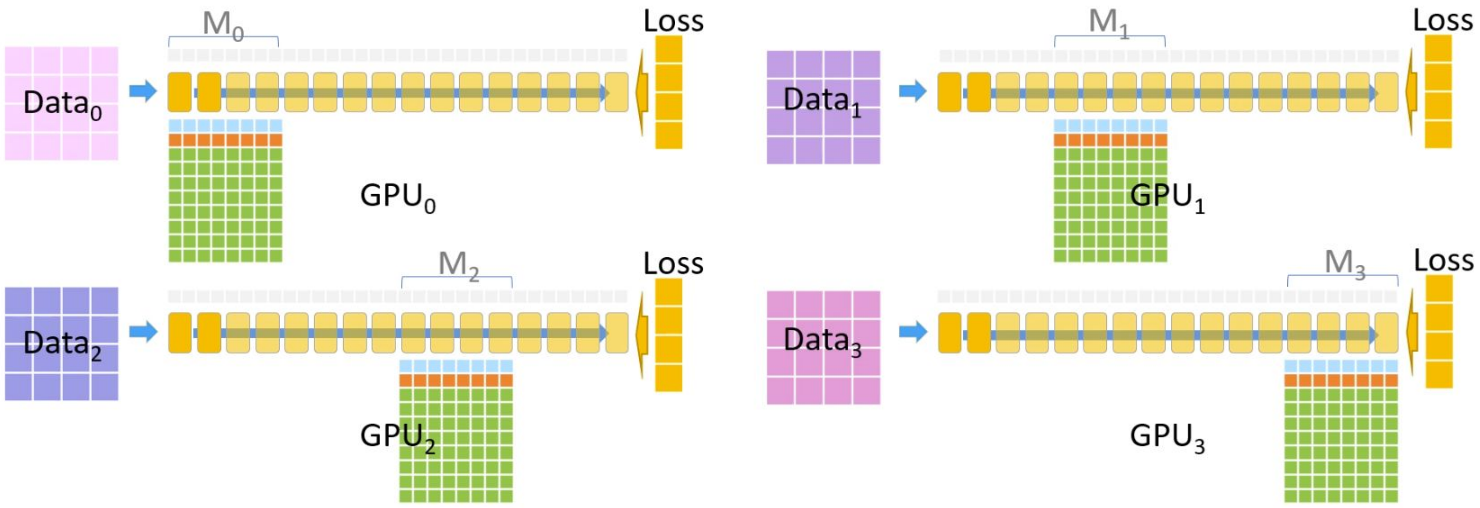
All devices except GPU_0 delete M_0 and its gradients, and all devices delete activations for M_0:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

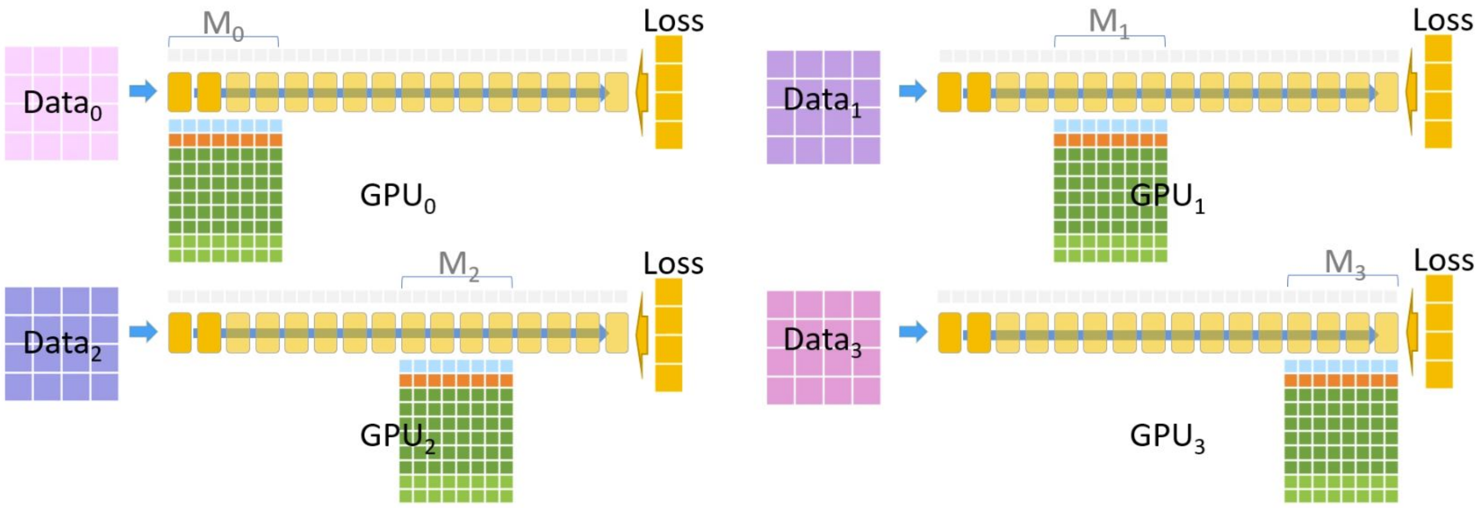
The backwards pass is complete. In parallel, each device performs the optimization step:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

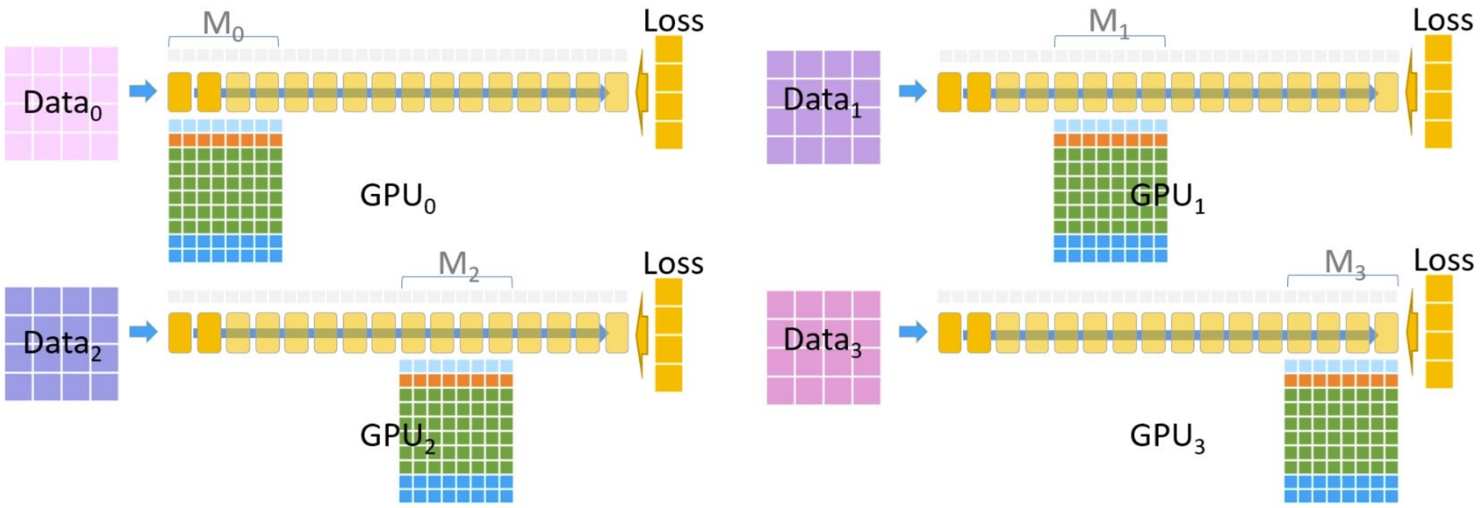
The optimizer runs:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

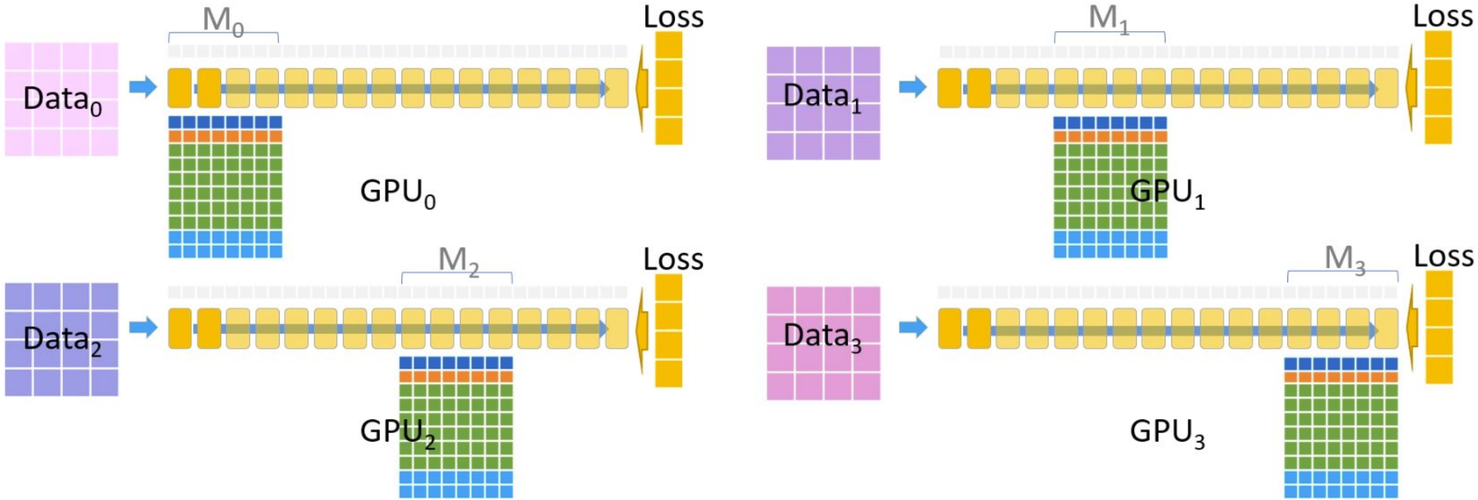
ZeRO

The updated model parameters are output as FP32:



ZeRO

The updated model parameters are converted to FP16 and replace the old parameters:



https://microsoft.com/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

ZeRO

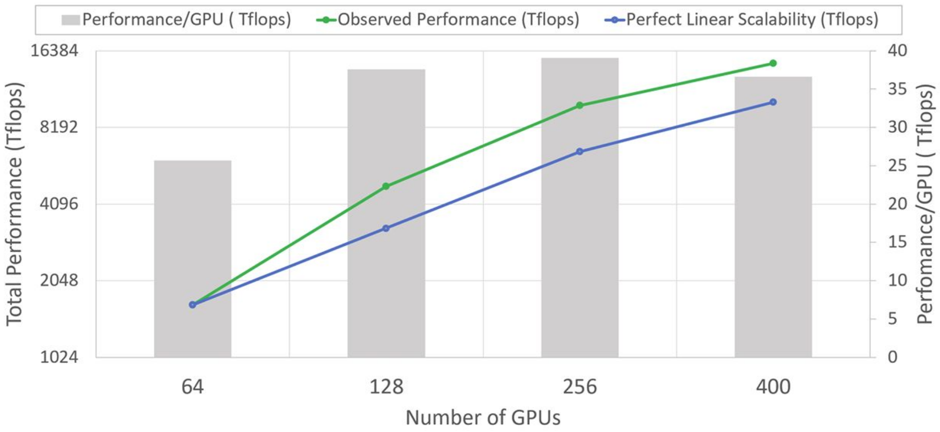
This method vertically partitions the model. Won't this cause the same issues as PP?

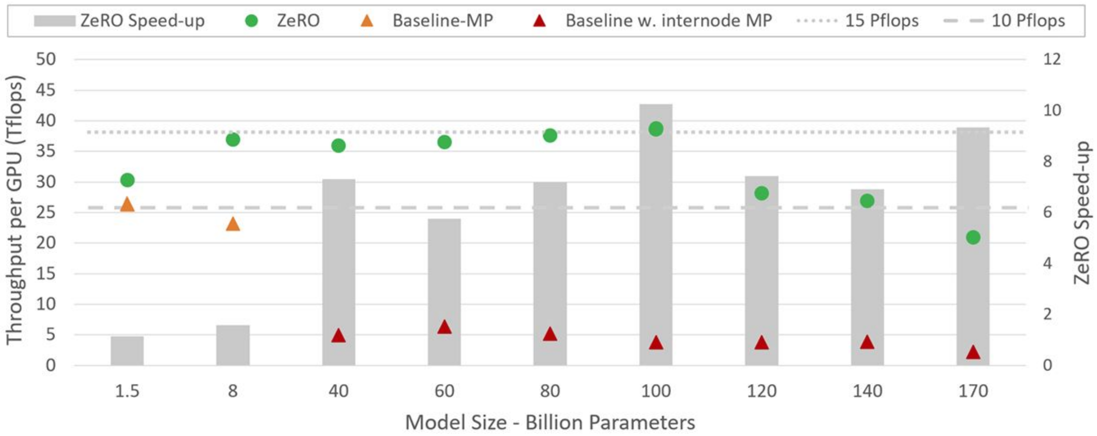
- Model partitions are broadcasted between devices
- Each device actually evaluates the entire model over the course of a forward/backward pass

That seems like a lot of communication. Won't this cause the same issues as MP?

- It is a lot of communication, but…
- MP communicates **activations**
- ZeRO communicates **parameters**
- Sending parameters can be done async. with forward/backward pass calculations

ZeRO

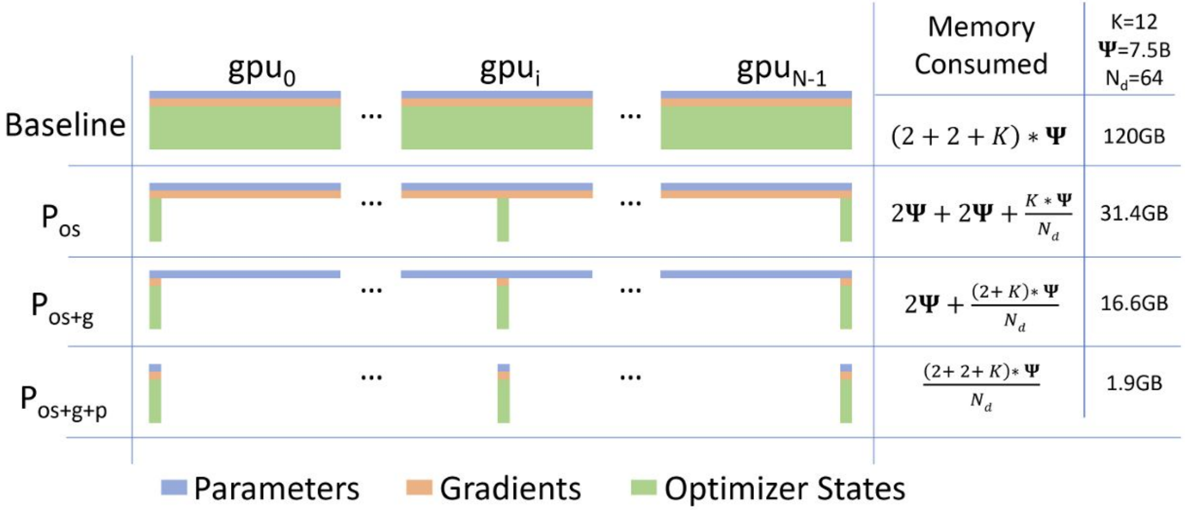

Performance/GPU (Tflops)
Observed Performance (Tflops)
Perfect Linear Scalability (Tflops)
Total Performance (Tflops)
Perfomance/GPU (Tflops)
16384
8192
4096
2048
1024
40
35
30
25
20
15
10
5
0
64
128
256
400
Number of GPUs


ZeRO Speed-up
ZeRO
Baseline-MP
Baseline w. internode MP
15 Pflops
10 Pflops
Throughput per GPU (Tflops)
ZeRO Speed-up
50
45
40
35
30
25
20
15
10
5
0
12
10
8
6
4
2
0
1.5
8
40
60
80
100
120
140
170
Model Size - Billion Parameters

ZeRO

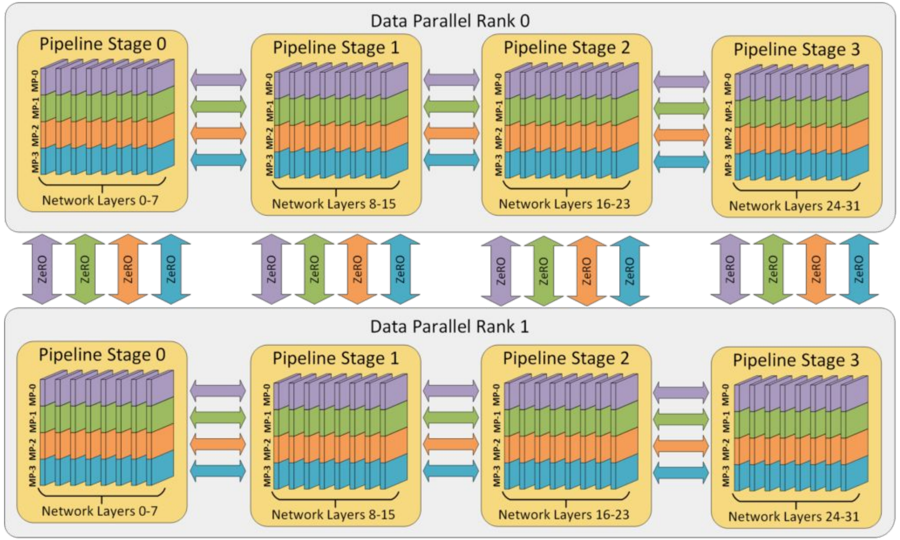
Communication cost can greatly exceed the cost of calculations, reducing the effectiveness of ZeRO.
ZeRO defines three stages of operation, with various degrees of partitioning:

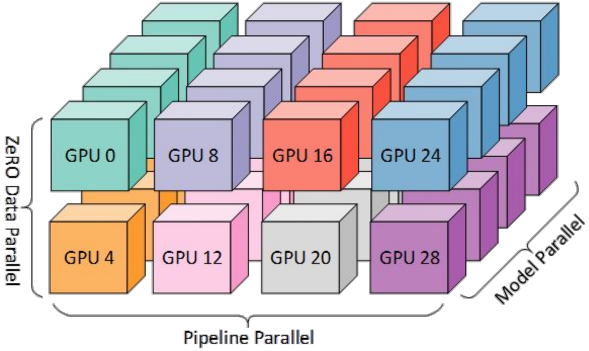
Ψ – model size K – optimizer memory multiplier N_d – DP degree



DeepSpeed: ZeRO + MP + PP

Combines ZeRO stages with other parallelization methods to optimize hardware/network performance:



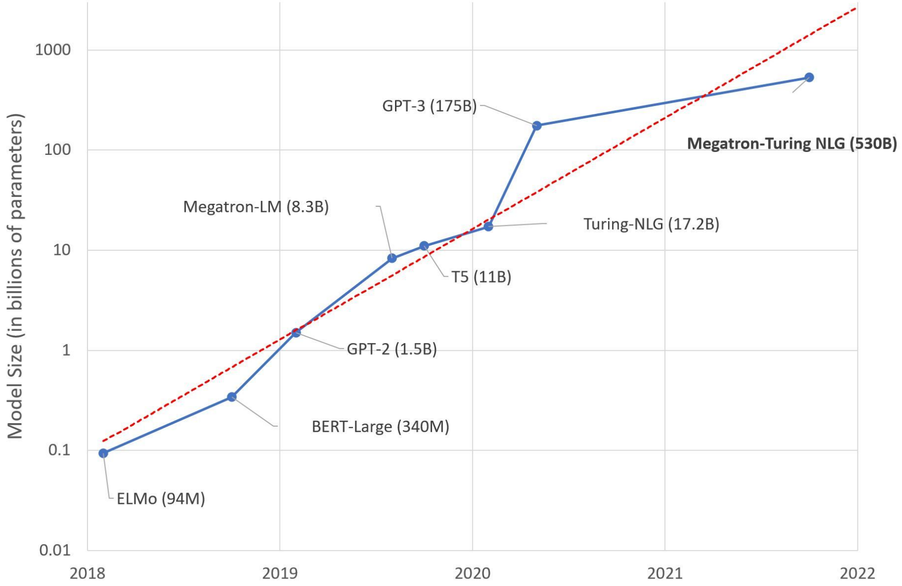


https://microsoft.com/research/blog/deepspeed-extreme-scale-model-training-for-everyone/

https://github.com/microsoft/DeepSpeed

DeepSpeed: ZeRO + MP + PP

Enabled the development of Turing-NLG and Megatron-Turing NLG:



https://microsoft.com/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/

Shaden Smith et al. Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model. arXiv, 2022.

ZeRO-1

Optimization states can be computed in parallel:

- ZeRO stage 1 is almost always a win

ZeRO-1 used in combination with MP and PP to develop many models:

- GPT-NeoX-20B
- Falcon
- DeepSeek-V3

Sid Black et al. GPT-NeoX-20B: An Open-Source Autoregressive Language Model. Association for Computational Linguistics, 2022.

Ebtesam Almazrouei et al. The Falcon Series of Open Language Models. arXiv, 2023.

DeepSeek-AI, Aixin Liu et al. DeepSeek-V3 Technical Report. arXiv, 2024.

Conclusion

Zero Redundancy Optimizer (ZeRO):

- Partitions optimizer states (stage 1), gradients (stage 2) and model parameters (stage 3)
- Communicates parameters and gradients only when they are needed
- Calculates optimizer states without communication
- Effectiveness of ZeRO-2 and ZeRO-3 is reduced by slow device interconnects
- Requires no code changes to model architecture

Thank You!