



CS 498: Machine Learning System Spring 2025

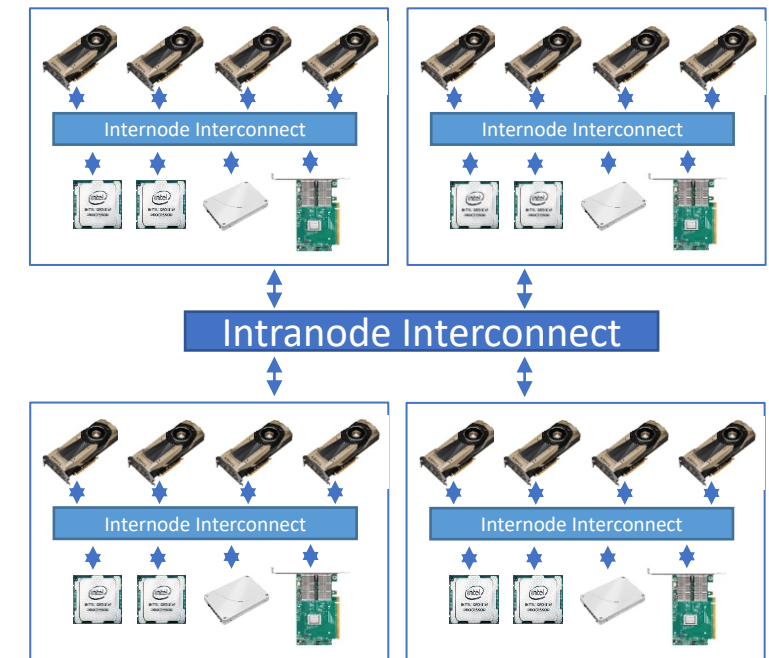
Minjia Zhang

The Grainger College of Engineering

Training with Heterogeneous Memory

- ZeRO-Offload

- Model parallelism (Megatron-LM)
 - Partition the model states vertically across multiple GPUs.
- Pipeline parallelism (PipeDream, Megatron-LM-v2)
 - Partition the model states horizontally across layers.
- ZeRO: Zero Redundancy Optimizer (ZeRO)
 - Split the training batch across multiple GPUs without model states duplication.

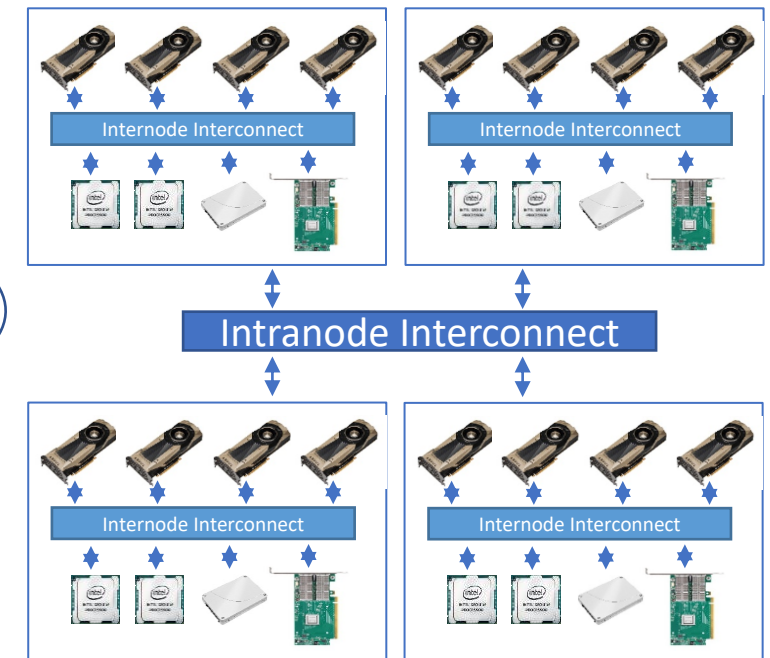


Distributed GPU Cluster

- Model parallelism (Megatron-LM)
 - Partition the model states vertically across multiple GPUs.

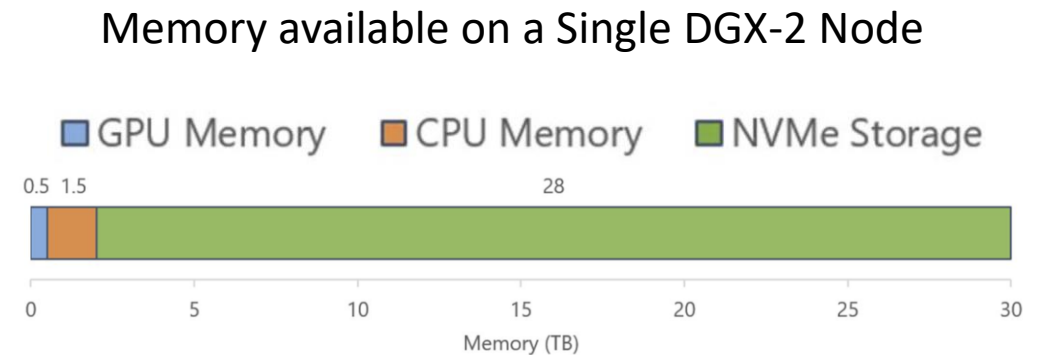
Require multiple GPUs resources

- ZeRO: Zero Redundancy Optimizer (ZeRO, SC'19)
 - Split the training batch across multiple GPUs without model states duplication.

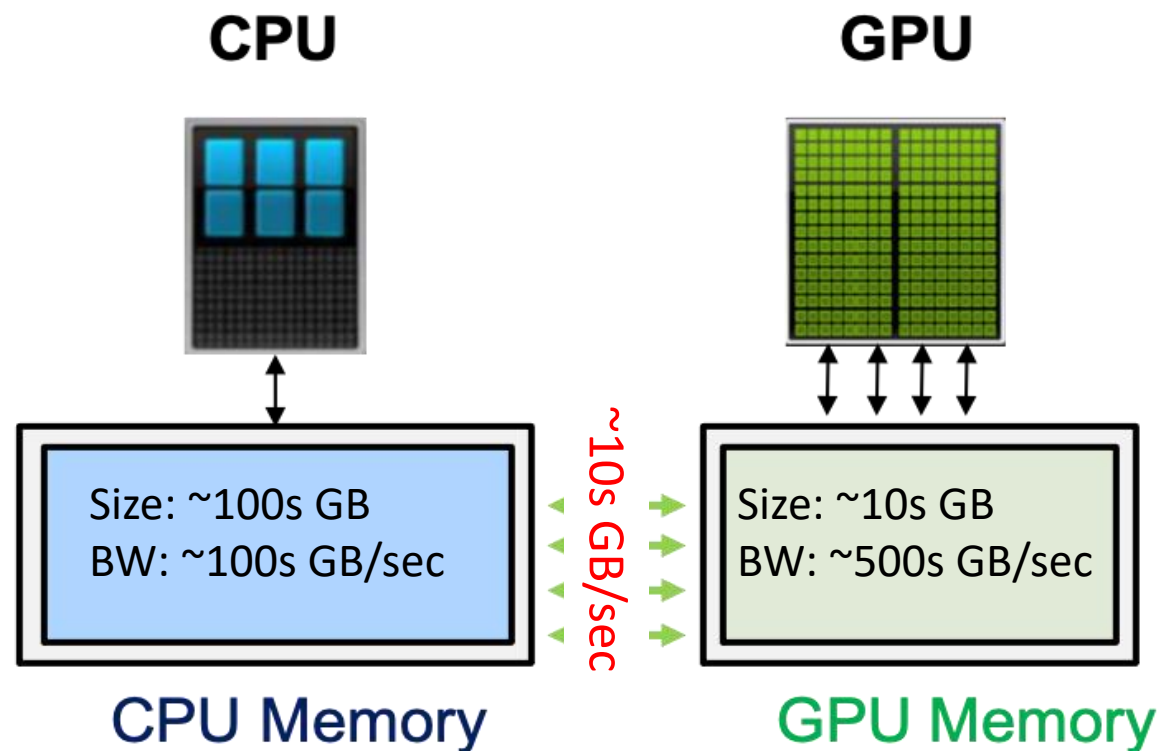


Distributed GPU Cluster

- Modern clusters have heterogeneous memory systems
- GPU memory comprises a small fraction
- Can we extend existing training technology to use CPU/NVMe memory?

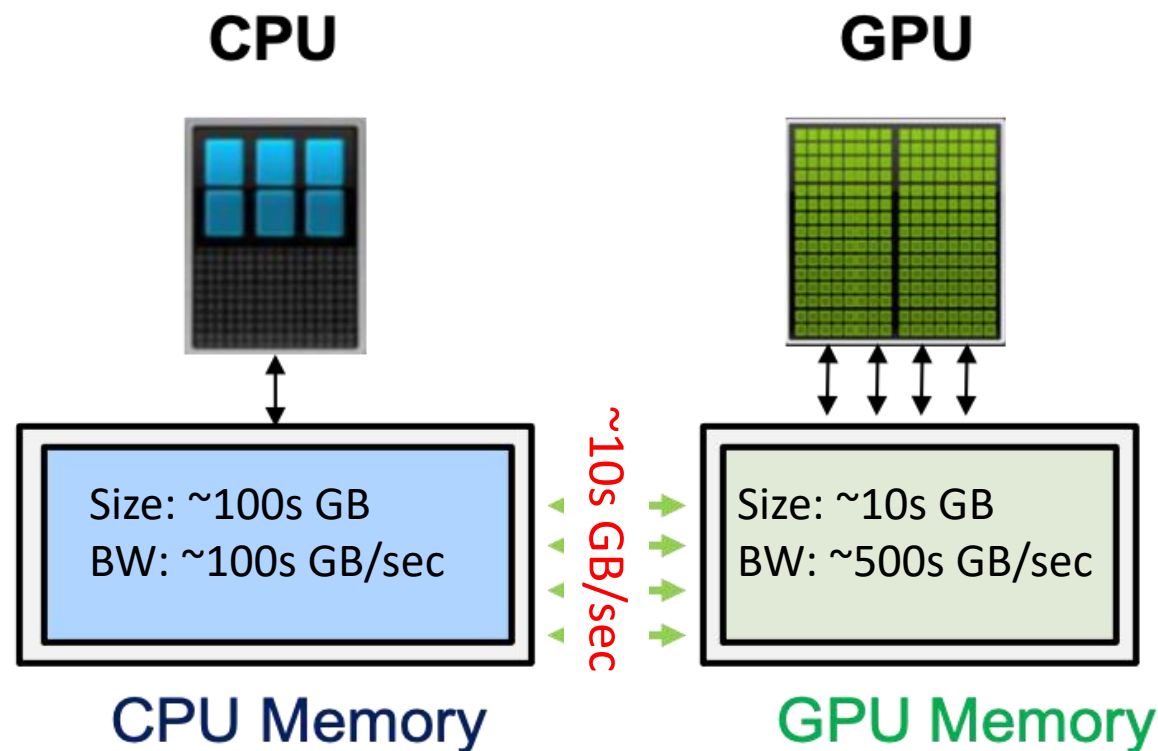


C1: Beyond CPU memory, can we also exploit CPU compute?



C1: Beyond CPU memory, can we also exploit CPU compute?

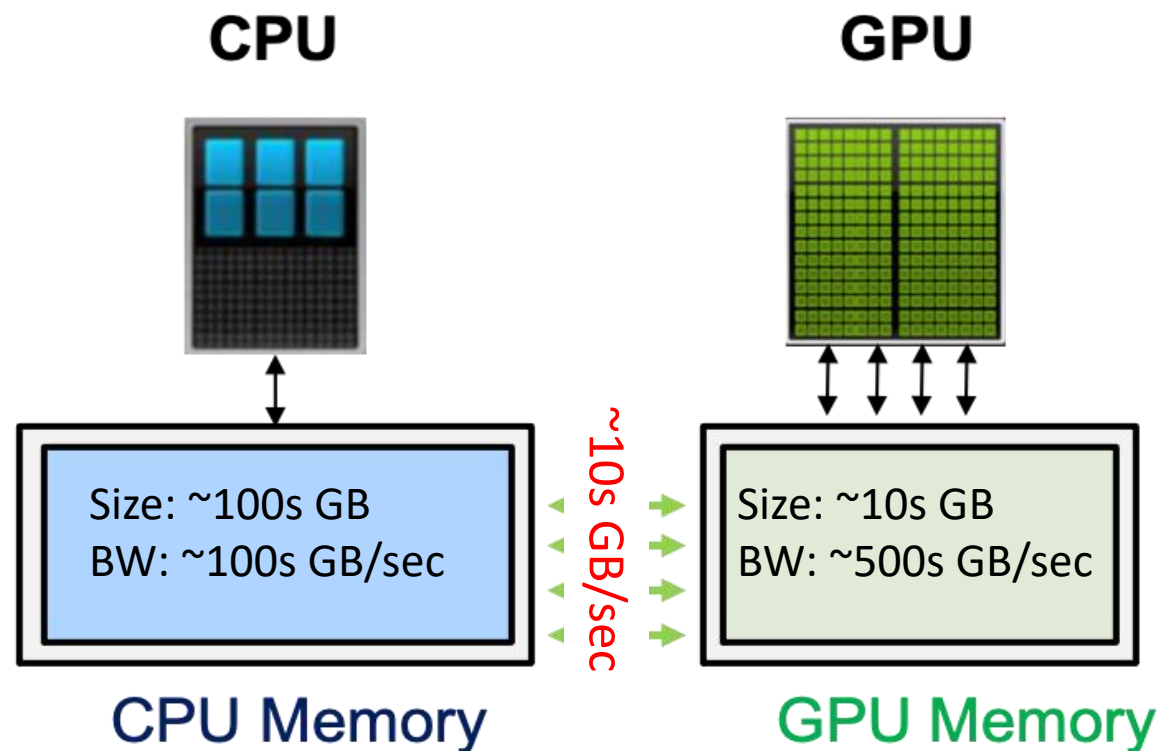
C2: Is CPU $\leftrightarrow$ GPU bandwidth sufficient to achieve comparable compute efficiency to non-offload strategies?



C1: Beyond CPU memory, can we also exploit CPU compute?

C2: Is CPU $\leftrightarrow$ GPU bandwidth sufficient to achieve comparable compute efficiency to non-offload strategies?

C3: Scalability during multi-GPU training?



C1: Beyond CPU memory, can we also exploit CPU compute?

C2: Is CPU $\leftrightarrow$ GPU
to achieve co-
efficiency to

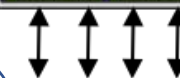
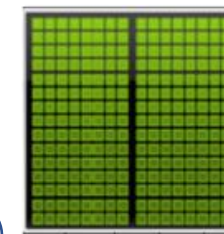
C3: Scalability during multi-GPU training?

Question: How do we effectively leverage CPU and GPU?

CPU



GPU

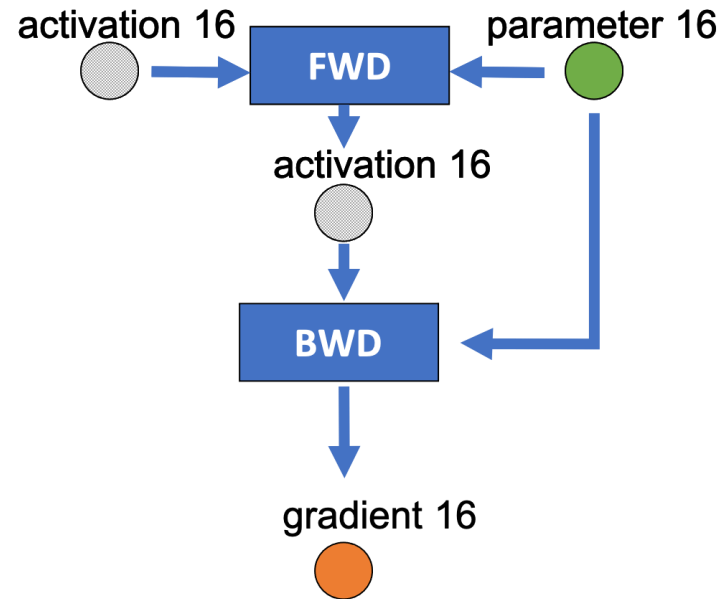


Size: ~10s GB
BW: ~500s GB/sec

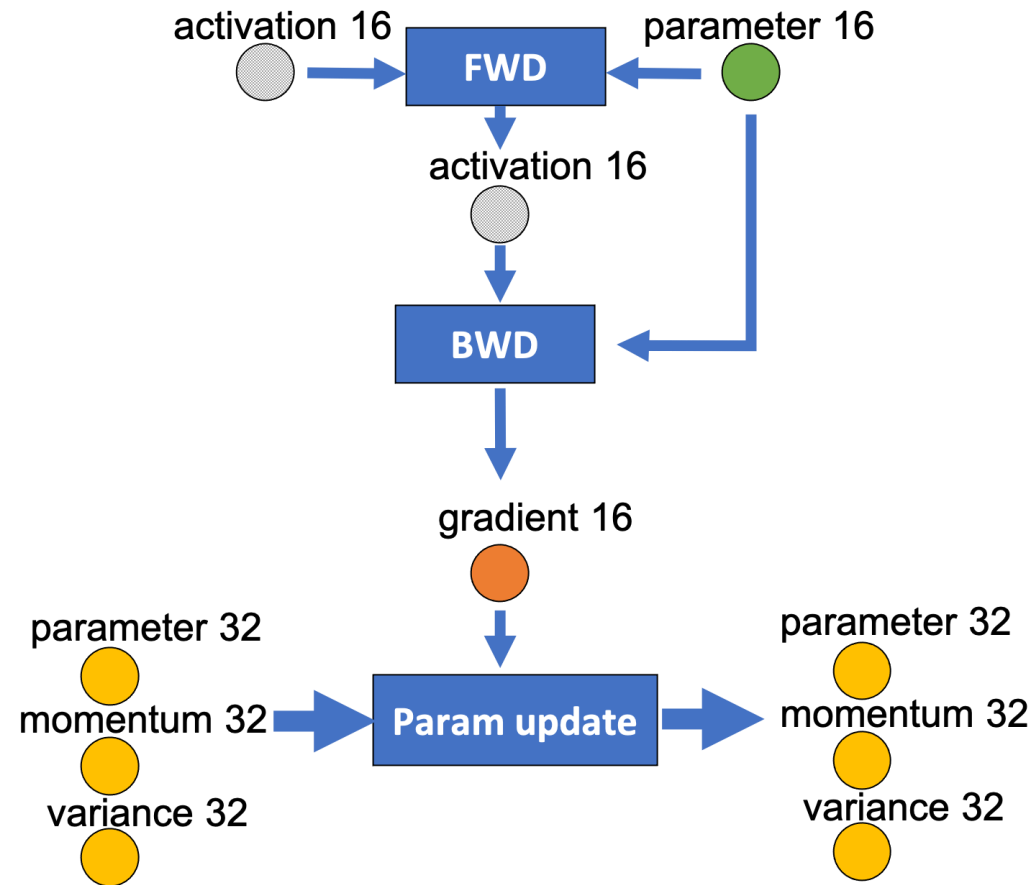
CPU Memory

GPU Memory

GB/sec

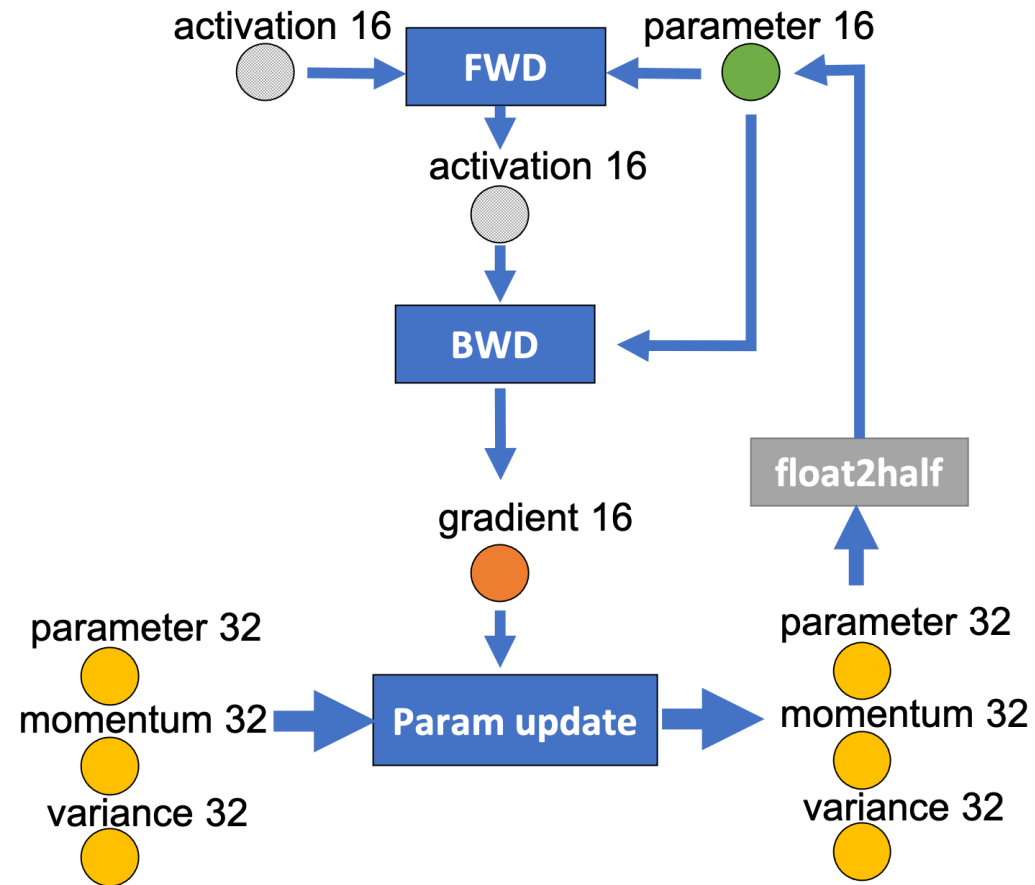


Mixed precision training iteration for a layer.



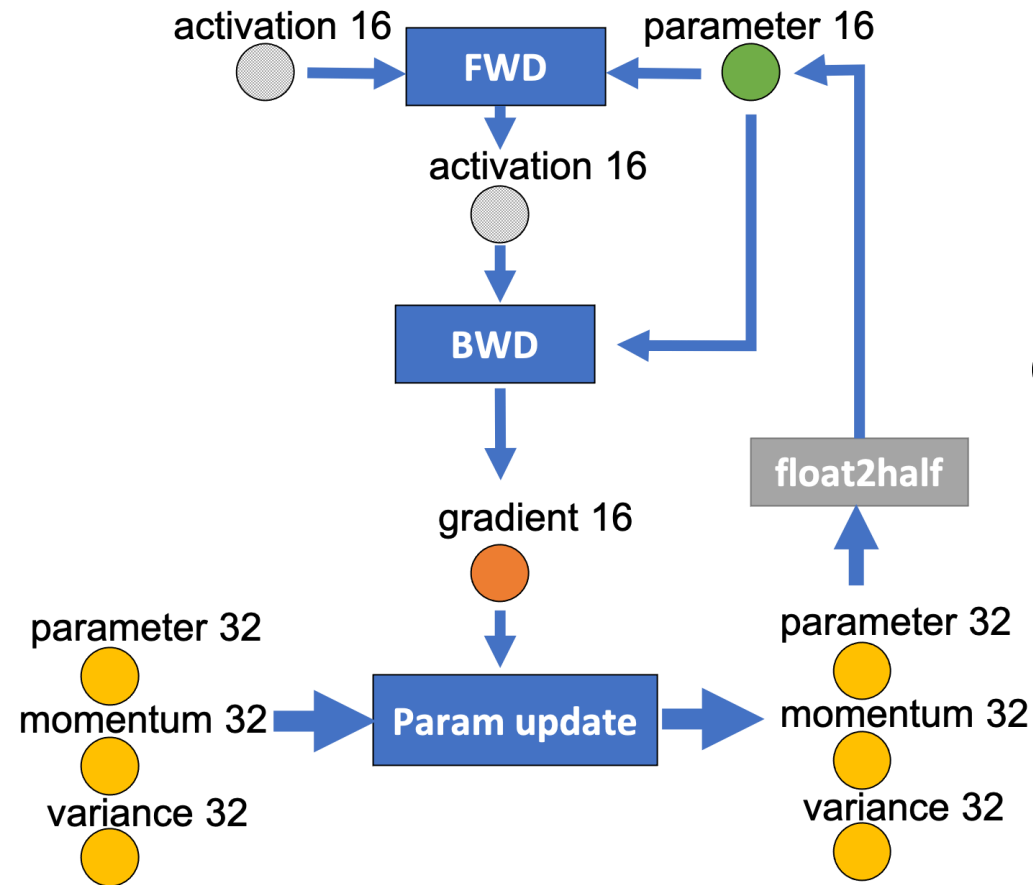
Mixed precision training iteration for a layer.

Mixed-precision Training



Mixed precision training iteration for a layer.

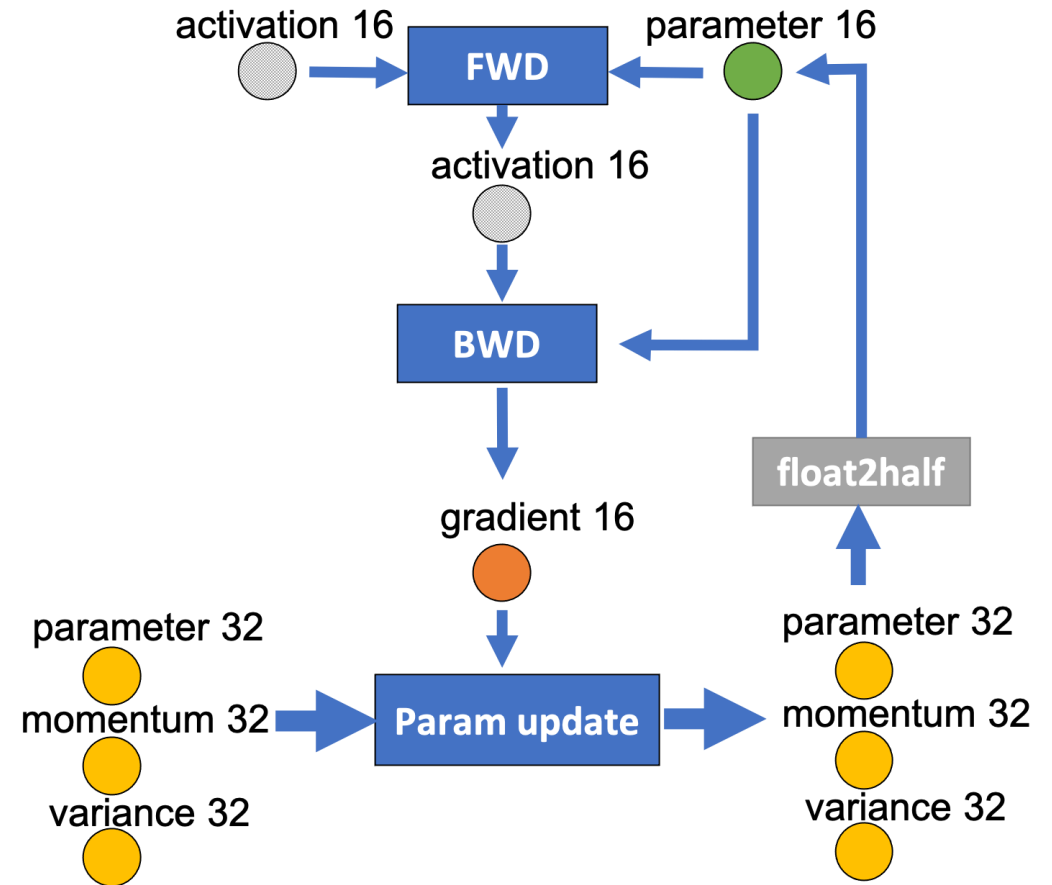
Formulating Heterogeneous Training as Graph Partitioning Problem



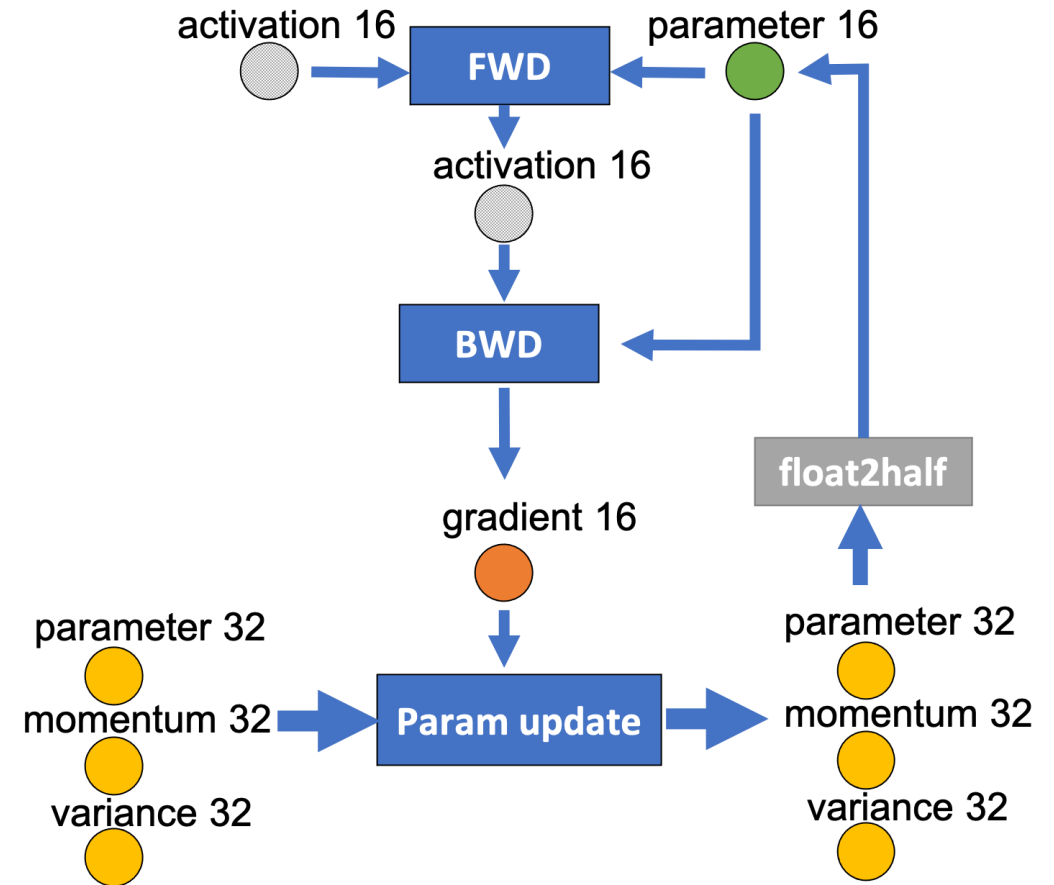
Challenge: how to partition?
NP-hard

Mixed precision training iteration for a layer.

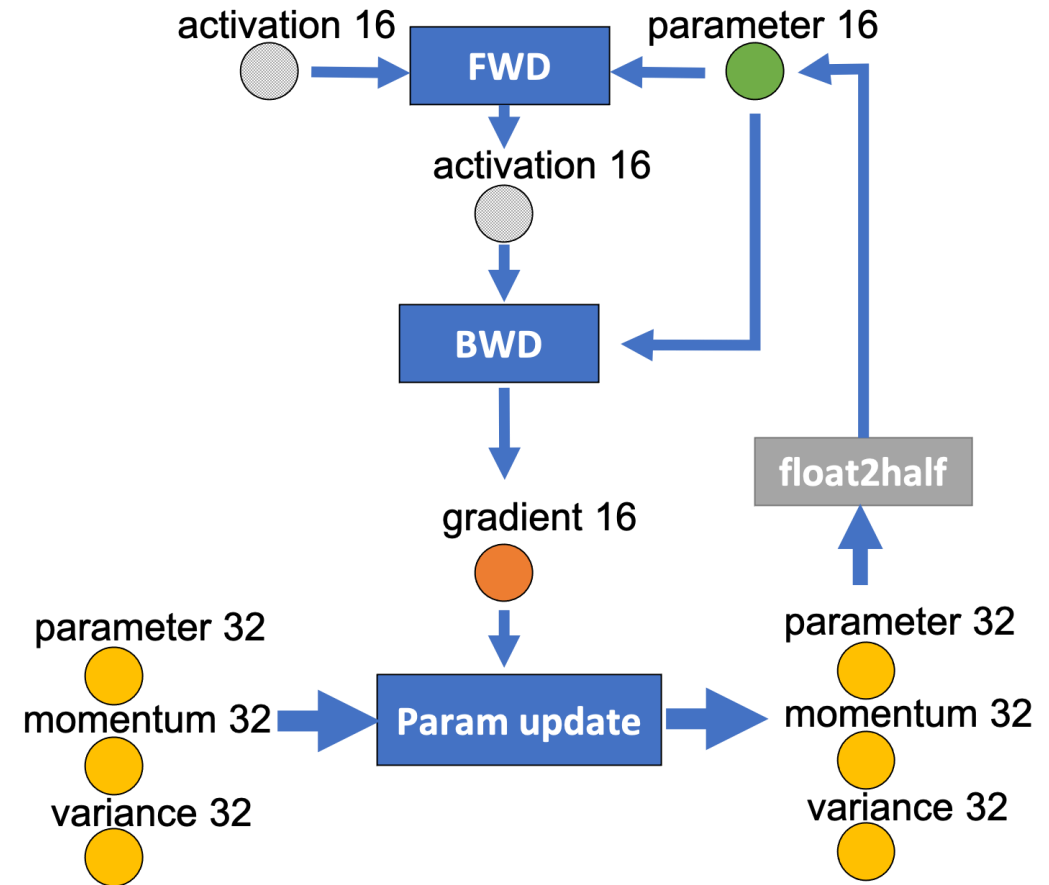
- i. Few computation on CPU

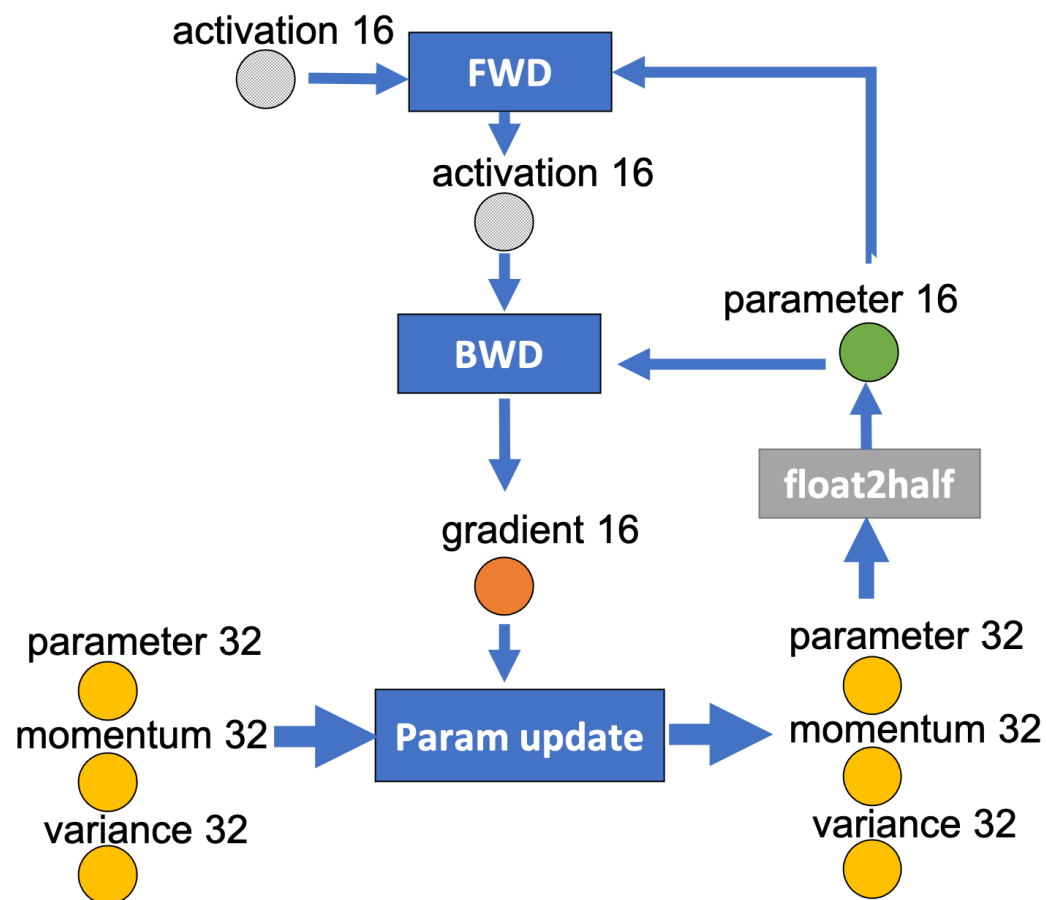


- i. Few computation on CPU
- ii. Minimize of communication volume



- i. Few computation on CPU
- ii. Minimize of communication volume
- iii. Maximize memory saving while achieving minimum communication volume

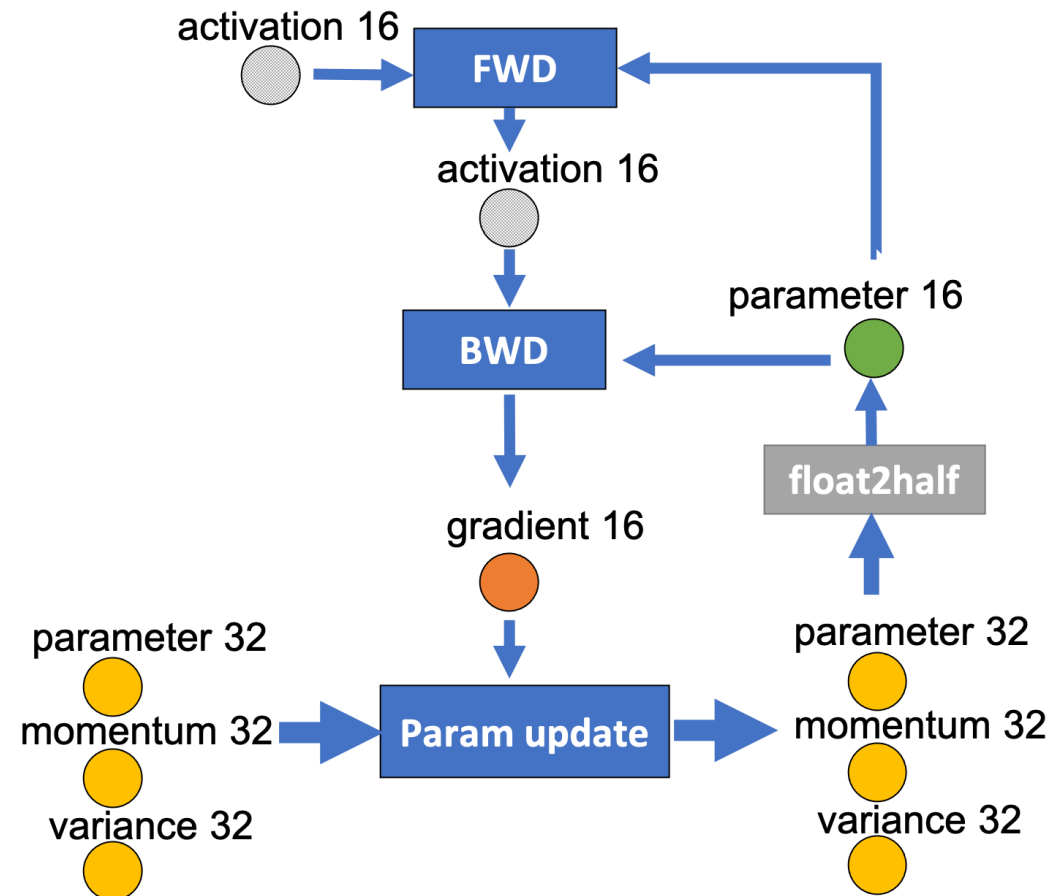




M: model param count
B: batch size

Computational Complexity?

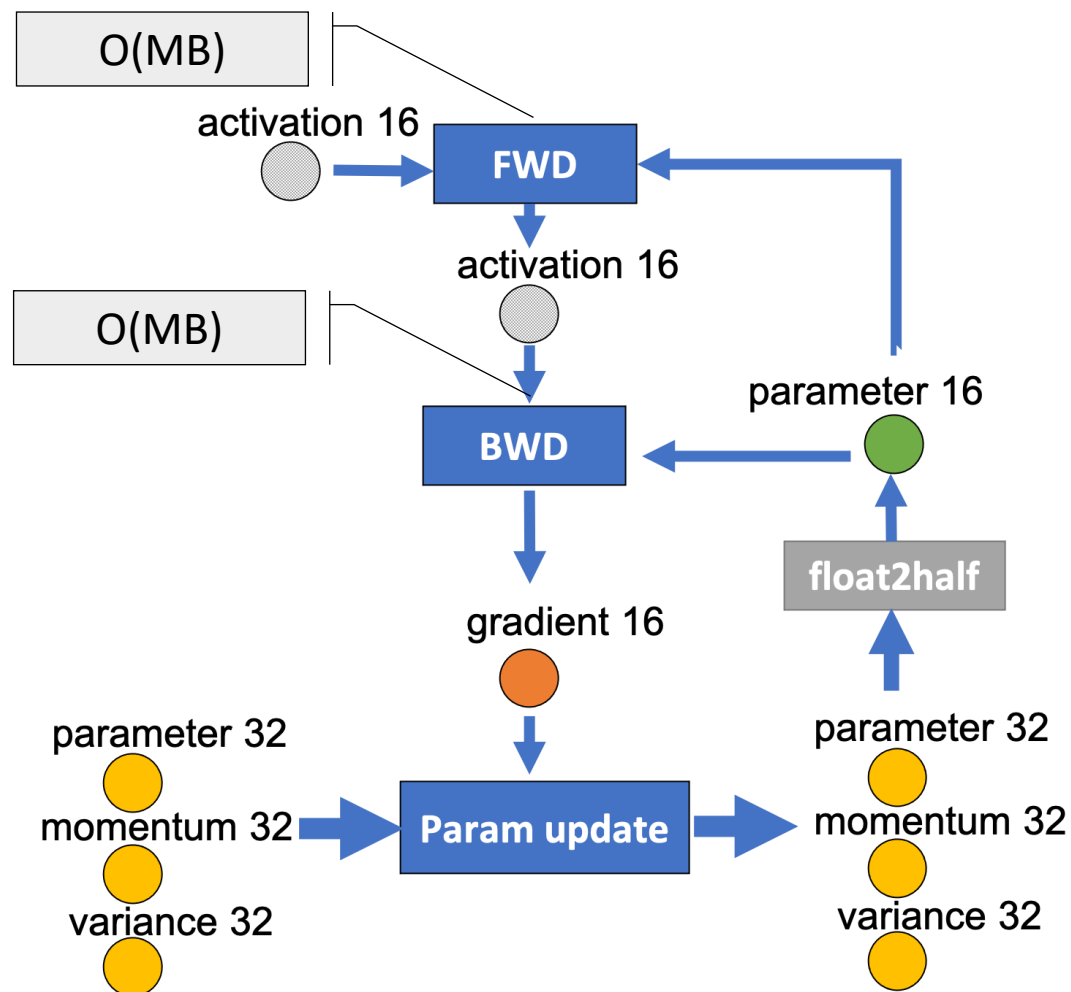
The dataflow of fully connected neural networks with M parameters.



M: model param count
B: batch size

Computational Complexity
 $O(MB)$

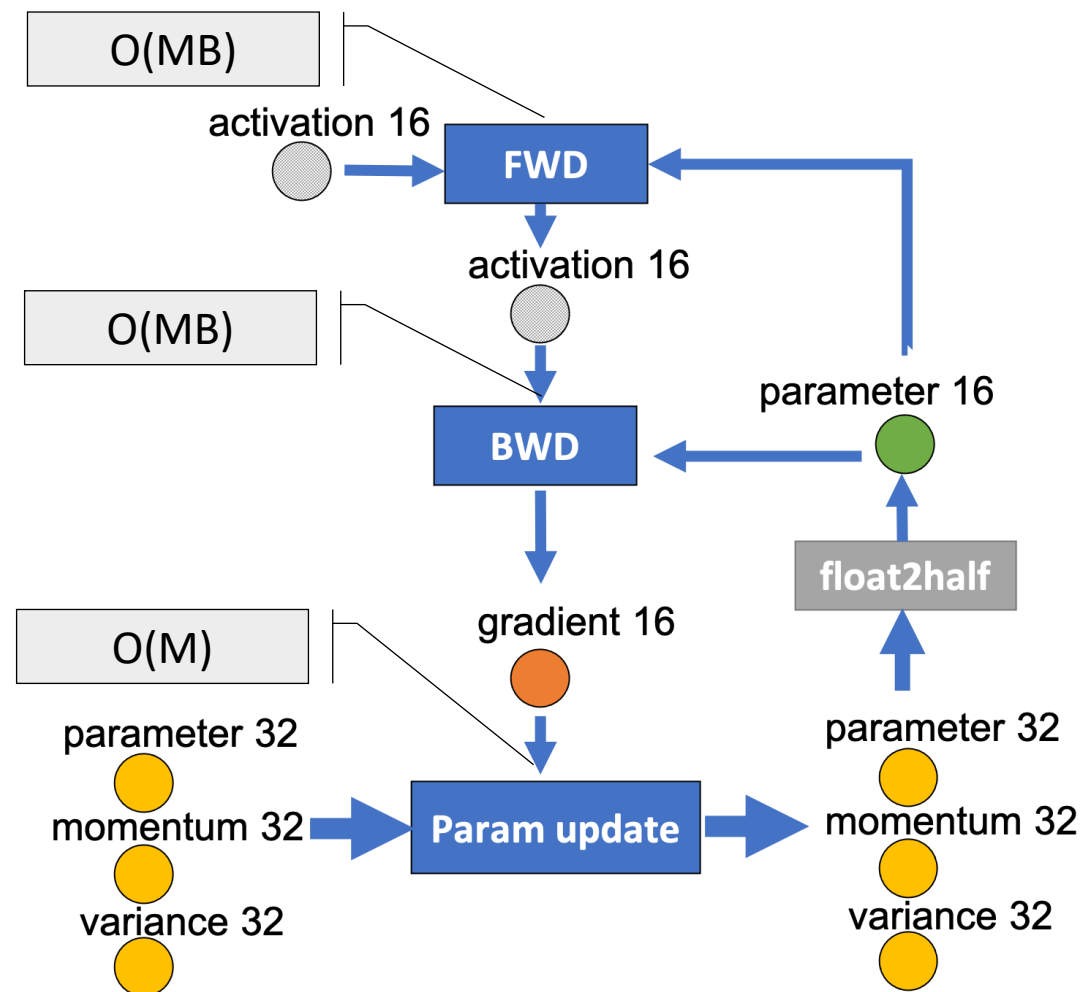
The dataflow of fully connected neural networks with M parameters.



M: model param count
B: batch size

Computational Complexity
 $O(MB)$

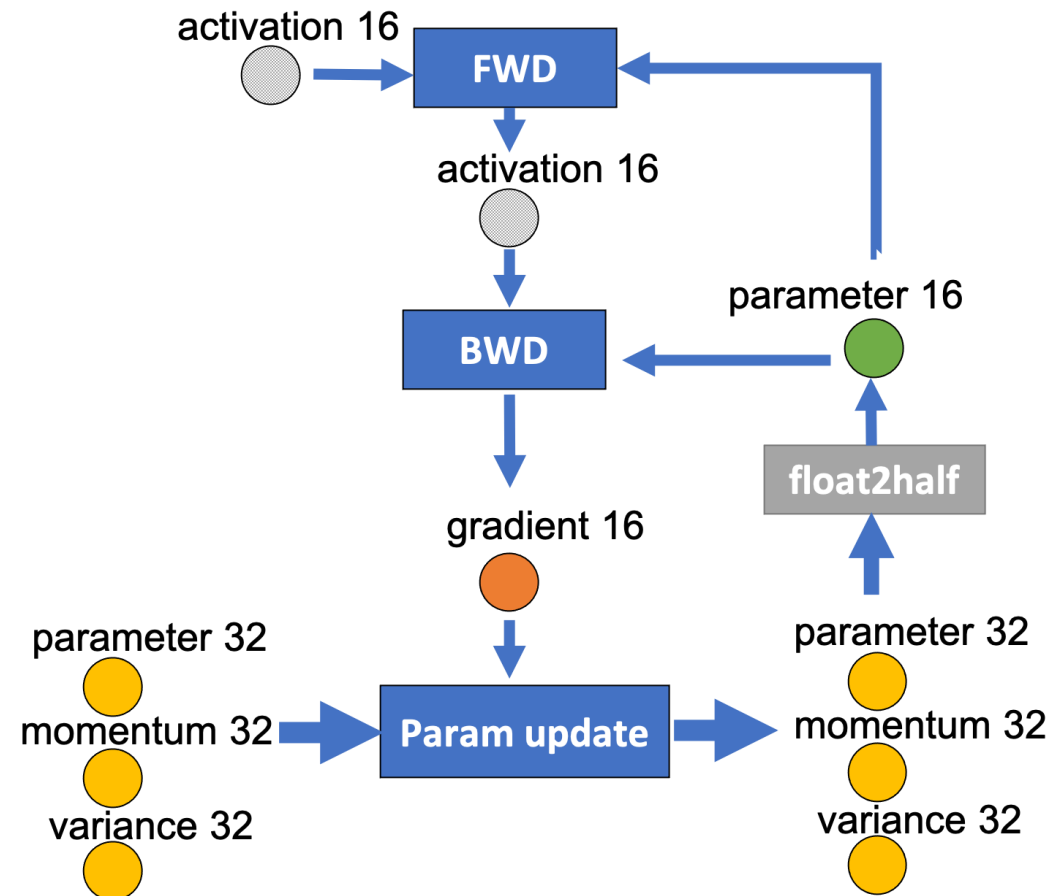
The dataflow of fully connected neural networks with M parameters.



M: model param count
B: batch size

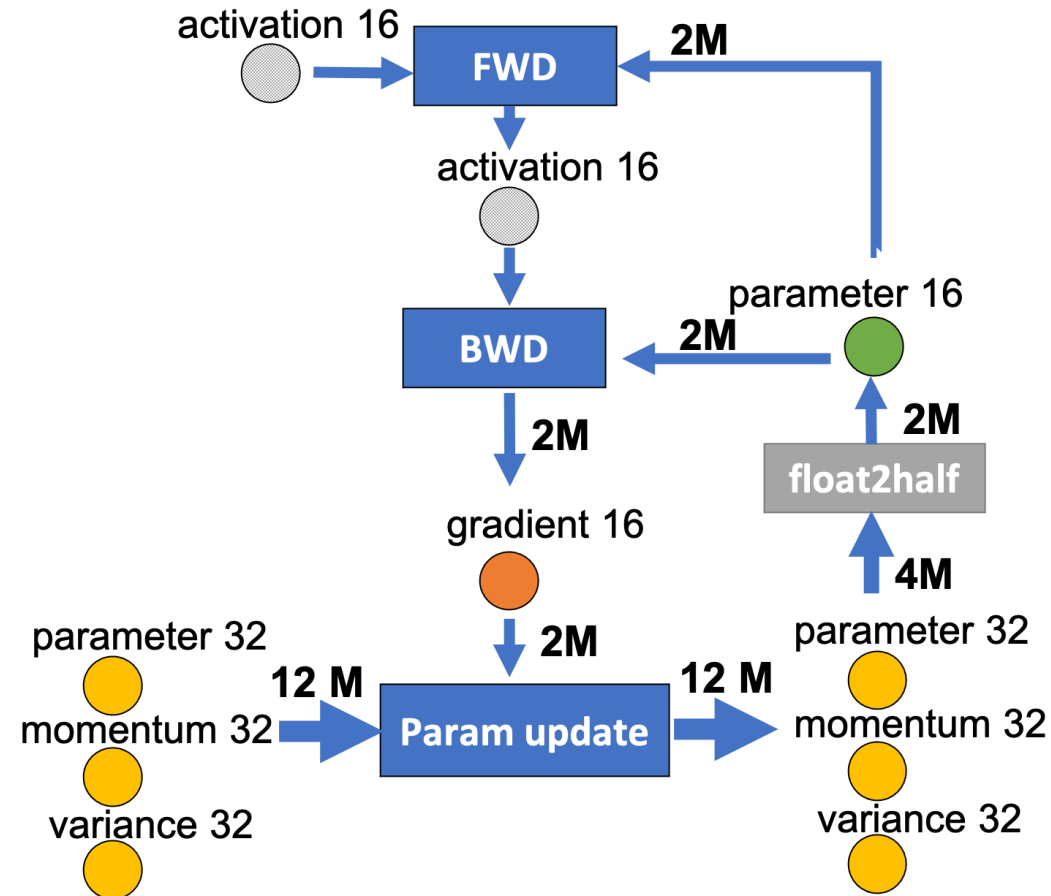
Computational Complexity
 $O(MB)$

The dataflow of fully connected neural networks with M parameters.



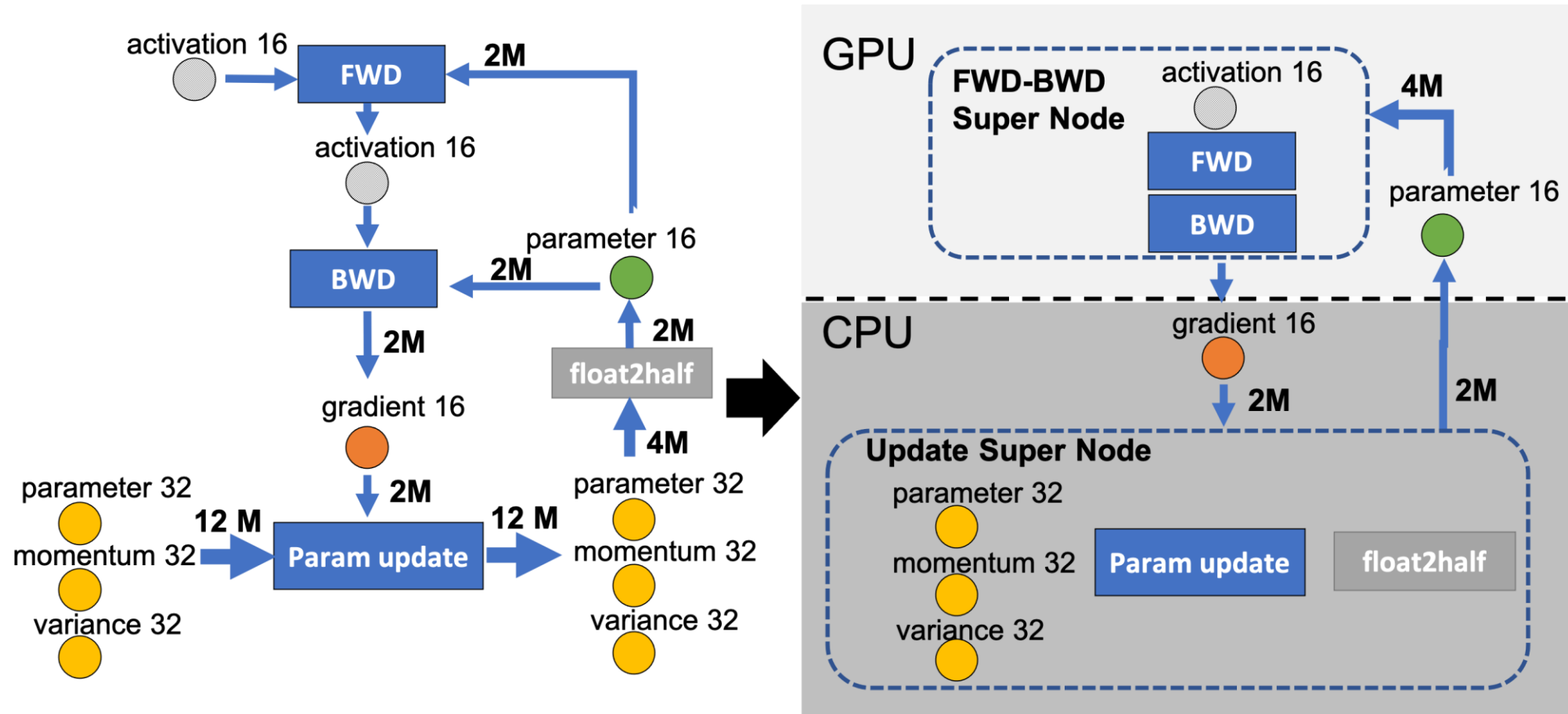
Data transfer volume?

The dataflow of fully connected neural networks with M parameters.

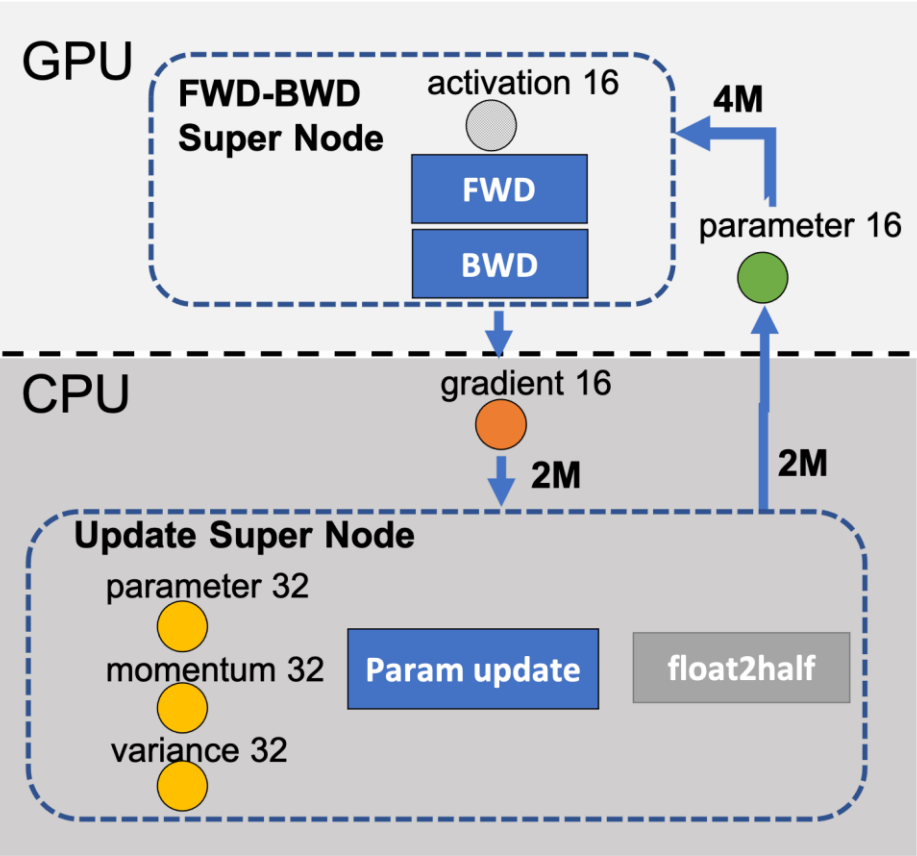


The dataflow of fully connected neural networks with M parameters.

Enable large model training by offloading data and compute to CPU



Offloading fp16 gradients and Update Super Node to CPU

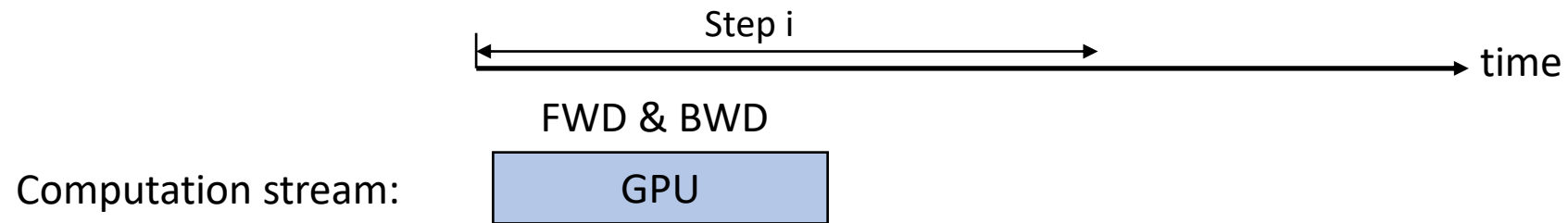


FWD-BWD	param16	gradient16	Update	Memory	Reduction
GPU	GPU	GPU	GPU	16M	1x(baseline)
GPU	GPU	CPU	GPU	14M	1.14x
GPU	GPU	GPU	CPU	4M	4x
GPU	GPU	CPU	CPU	2M	8x

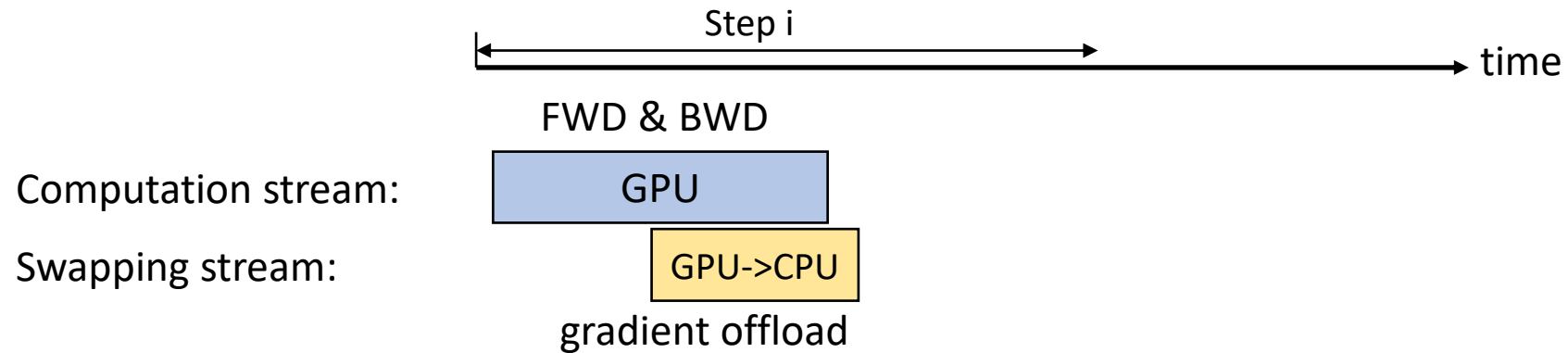
Memory savings for offload strategies that minimize communication volume compared to the baseline.

Challenge: **how** and **when** to offload?

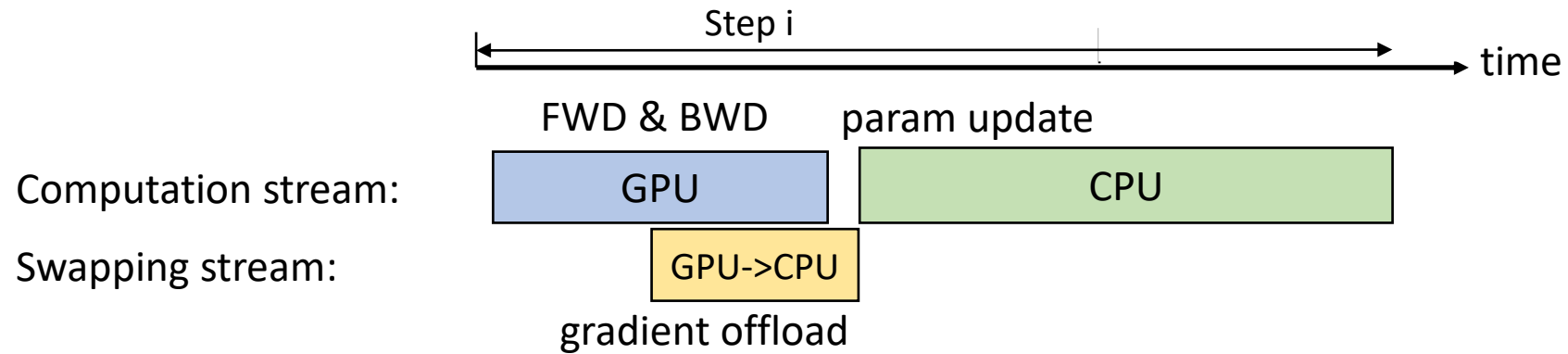
- Would offloading increase total iteration time?
- Would CPU execution become a bottleneck?

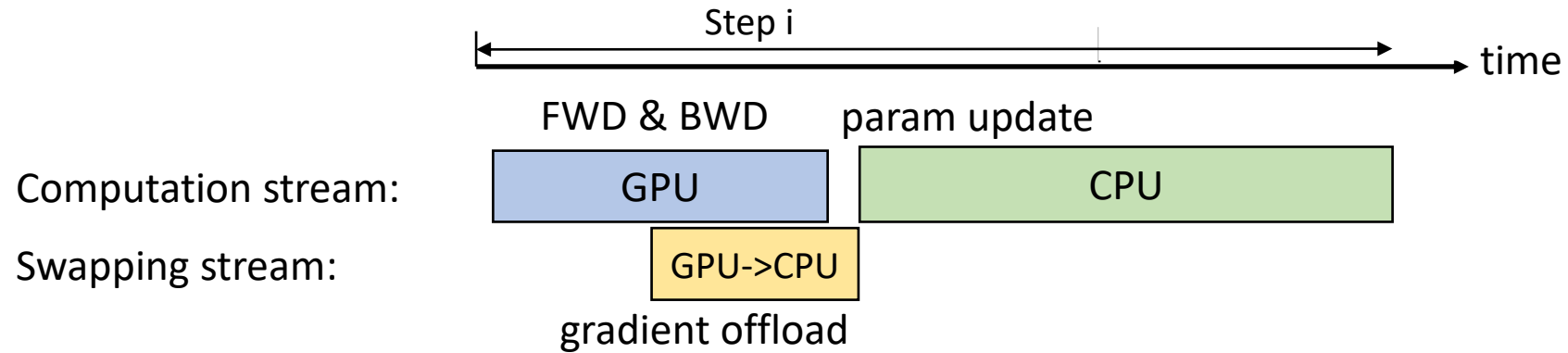


ZeRO-Offload training process on a single GPU.



ZeRO-Offload training process on a single GPU.





Question: Is it possible to reduce CPU execution time?

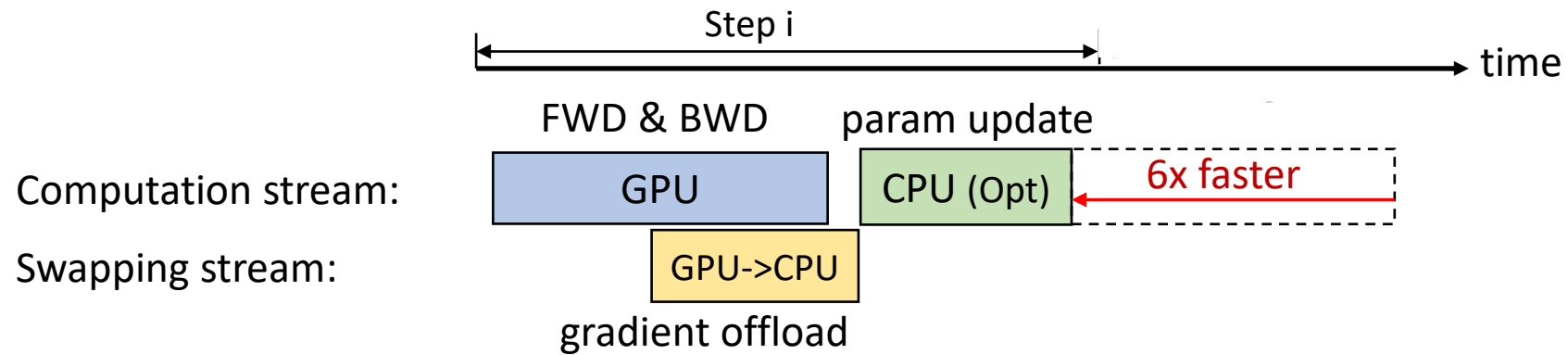
- Highly parallelized CPU optimizer implementation
 - 1) SIMD vector instruction for fully exploiting the hardware parallelism supported on CPU architectures.

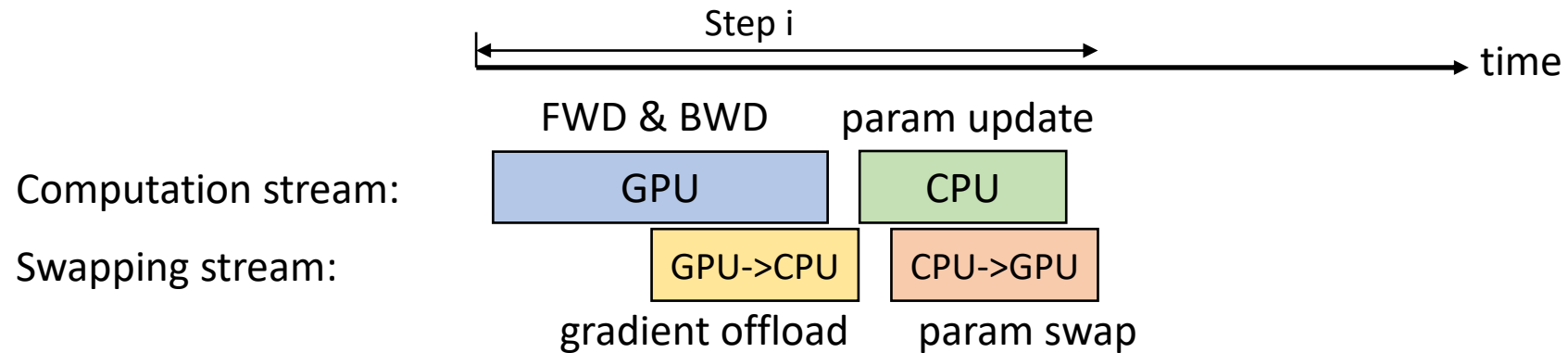
- Highly parallelized CPU optimizer implementation
 - 1) SIMD vector instruction for fully exploiting the hardware parallelism supported on CPU architectures.
 - 2) Loop unrolling to increase instruction level parallelism.

- Highly parallelized CPU optimizer implementation
 - 1) SIMD vector instruction for fully exploiting the hardware parallelism supported on CPU architectures.
 - 2) Loop unrolling to increase instruction level parallelism.
 - 3) OMP multithreading for effective utilization of multiple cores and threads on the CPU in parallel.

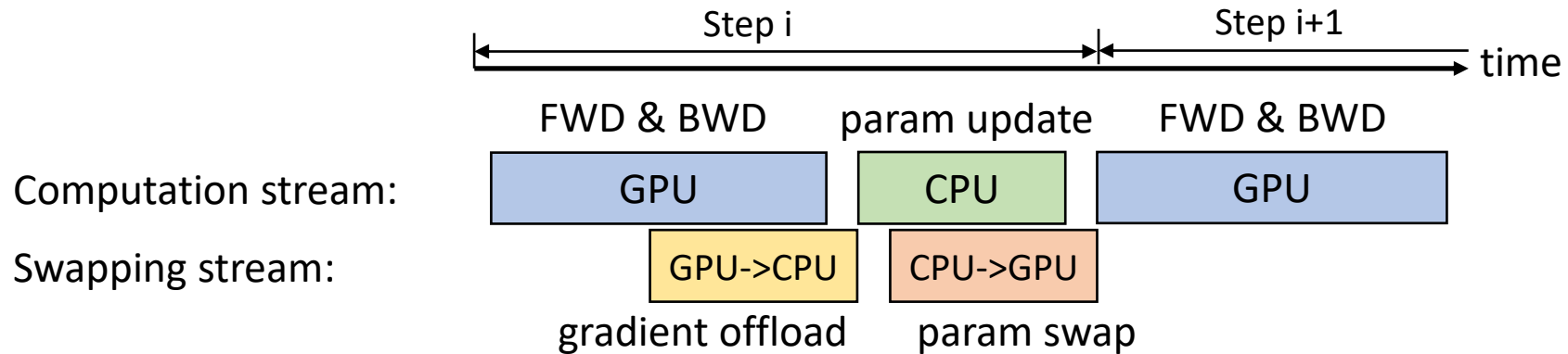
- Highly parallelized CPU optimizer implementation
 - 1) SIMD vector instruction for fully exploiting the hardware parallelism supported on CPU architectures.
 - 2) Loop unrolling to increase instruction level parallelism.
 - 3) OMP multithreading for effective utilization of multiple cores and threads on the CPU in parallel.

	Adam Optimizer: Parameter Update Latency (seconds)		
Model Size (B)	PyTorch-CPU	Optimized-CPUAdam (now in PyTorch)	PyTorch-GPU
1	1.39	0.22	0.10
2	2.75	0.51	0.26
4	5.71	1.03	0.64
8	11.93	2.41	0.87
10	14.00	2.57	1.00

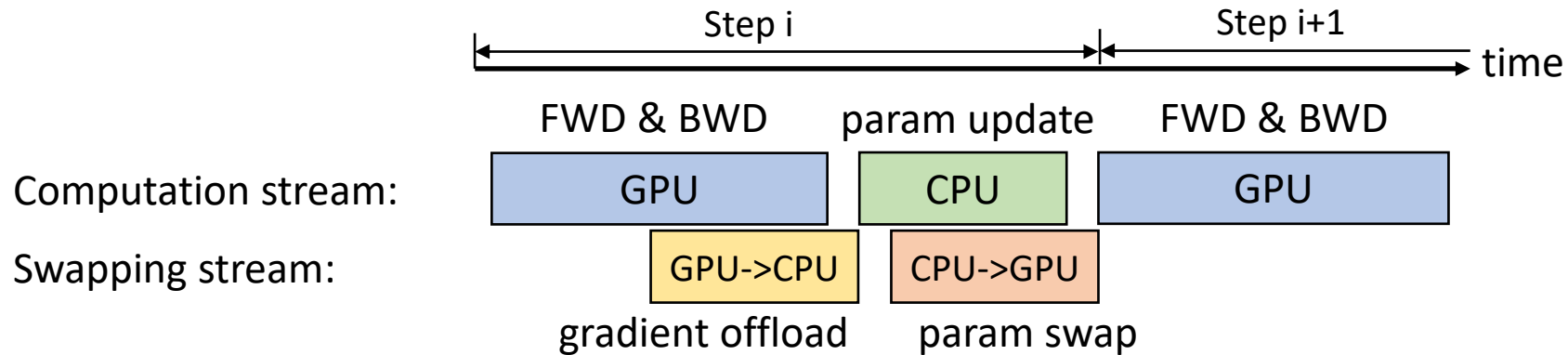




ZeRO-Offload training process on a single GPU.



ZeRO-Offload training process on a single GPU.

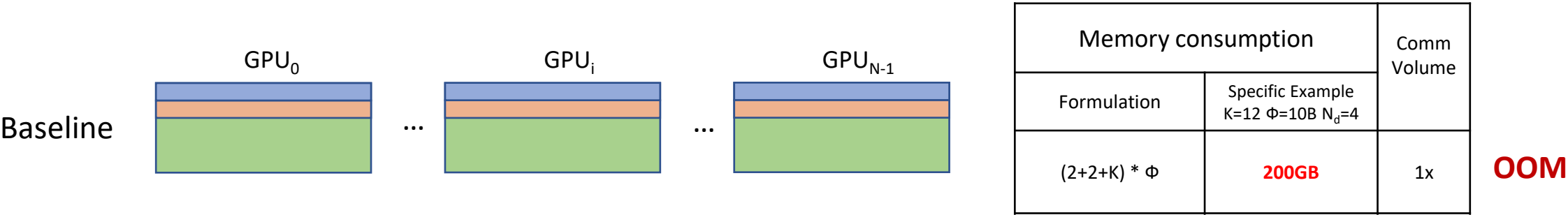


ZeRO-Offload training process on a single GPU.



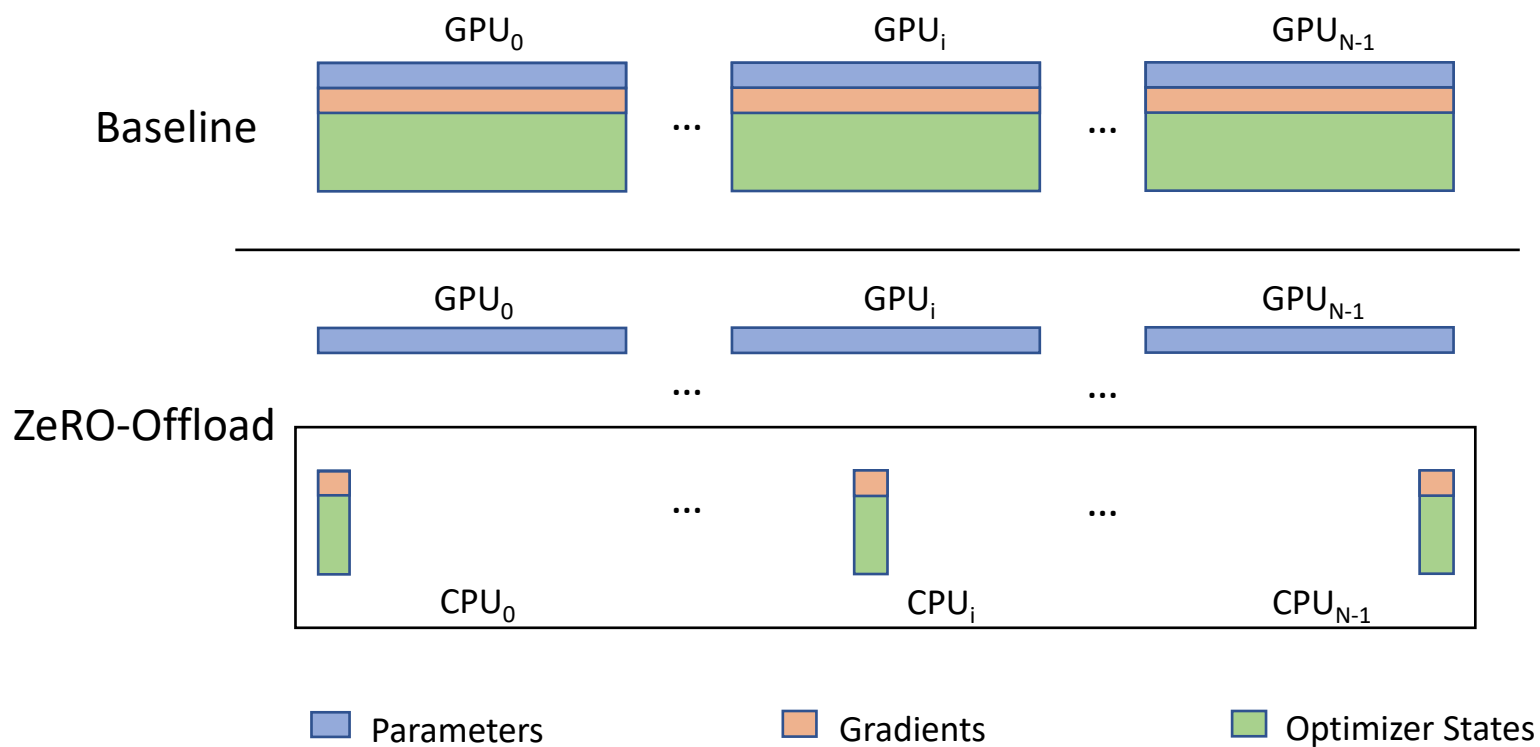
Minimal impact to critical path length by hiding offloading latency and with highly parallelized CPU optimizer

- **Redundancy** across data parallel process



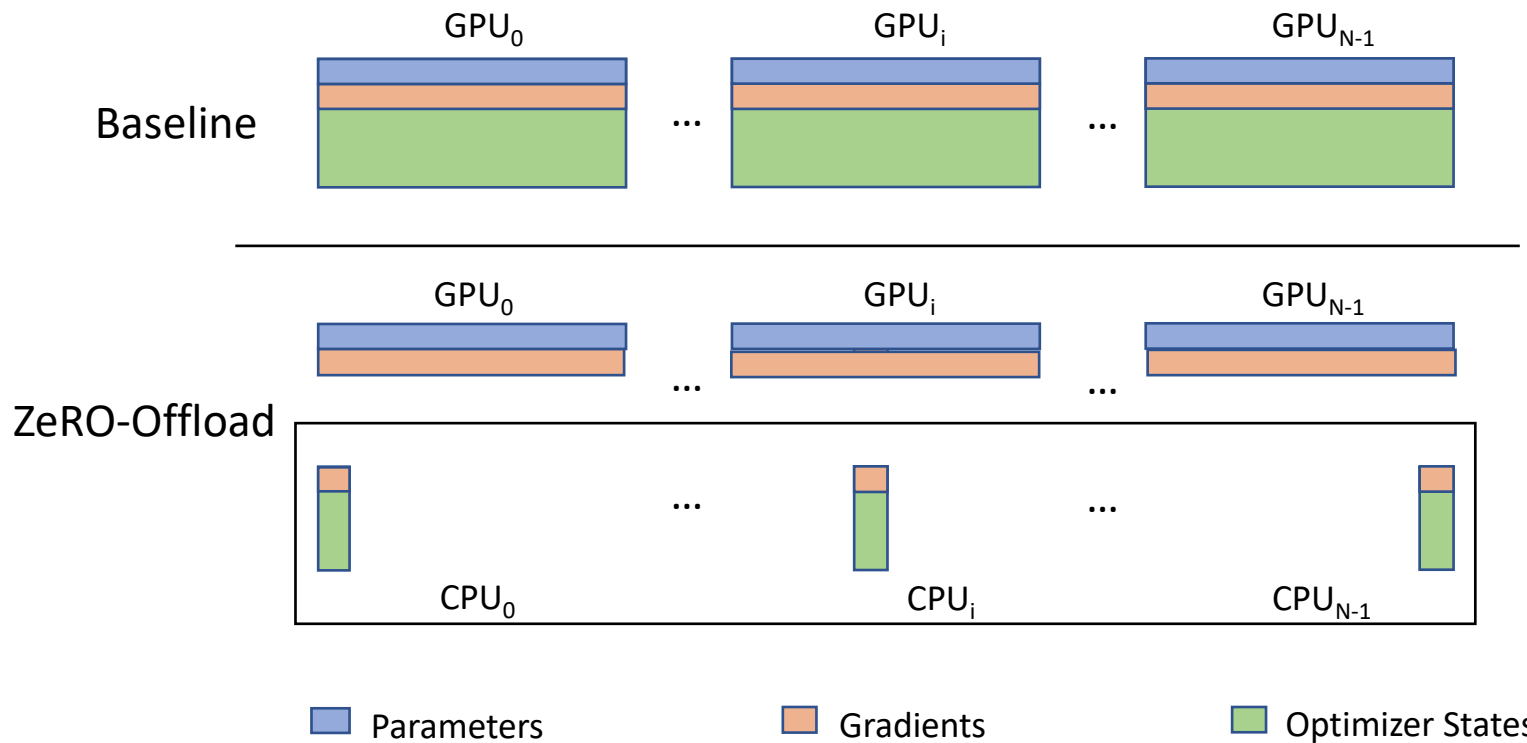
- C1:** Would offloading model states or computation to CPU in combination with data parallelism result in significant replication of CPU memory usage and compute?
- C2:** Would offloading increase communication and limit throughput scalability?

- Partitioning **optimizer states and gradients**



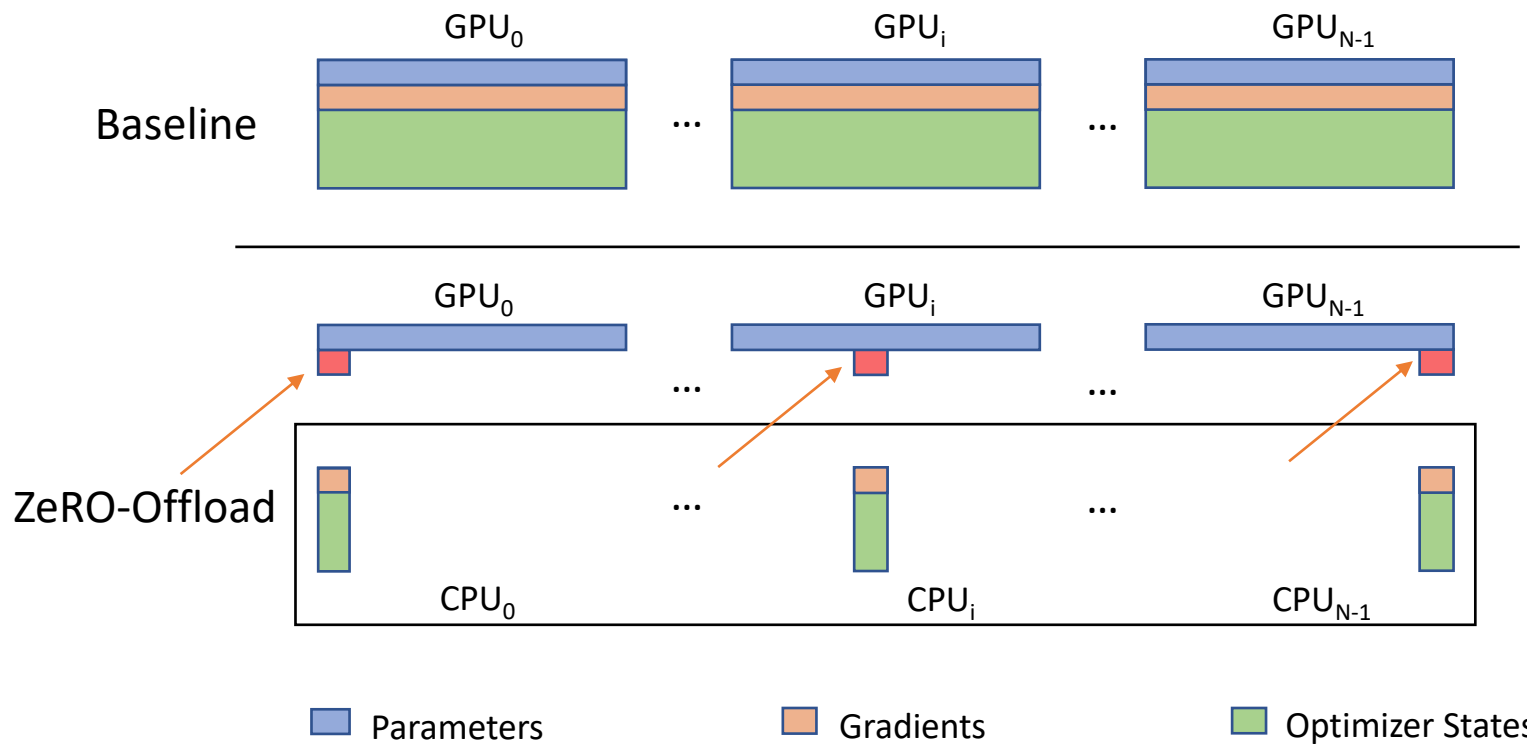
- Partitioning **optimizer states and gradients**

1. Compute gradients on different data

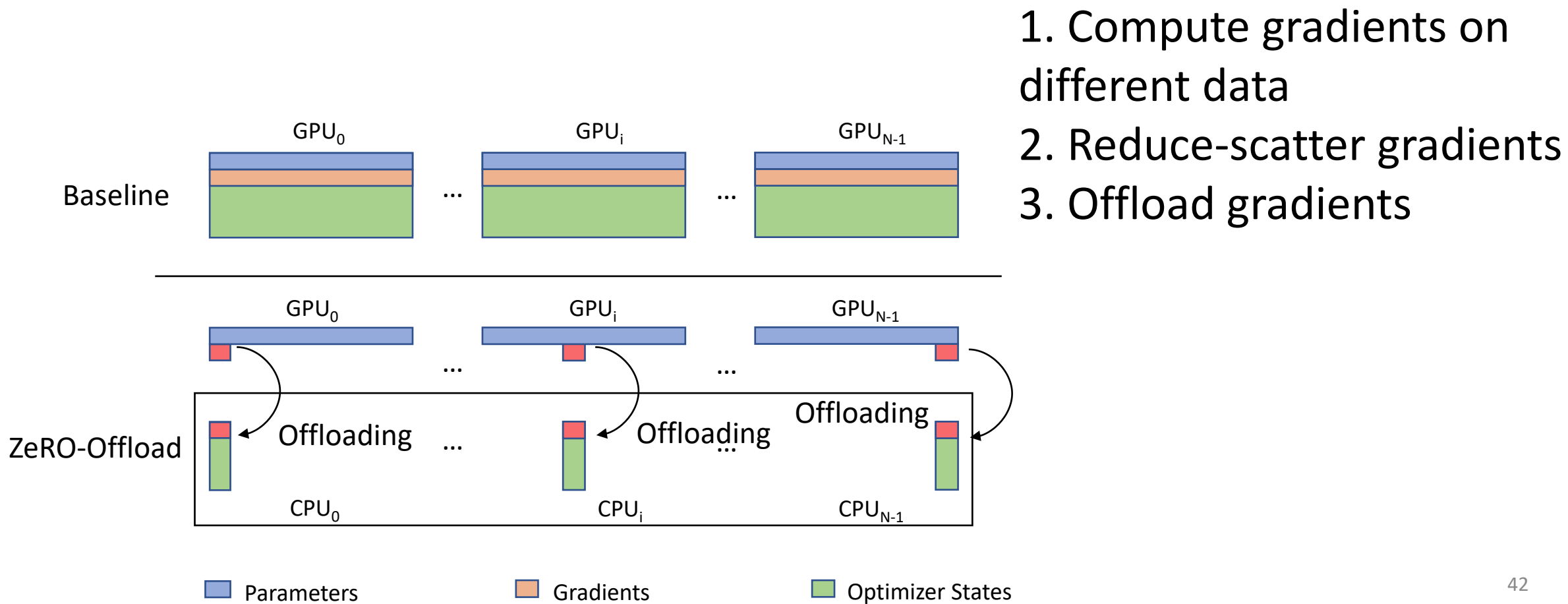


- Partitioning **optimizer states and gradients**

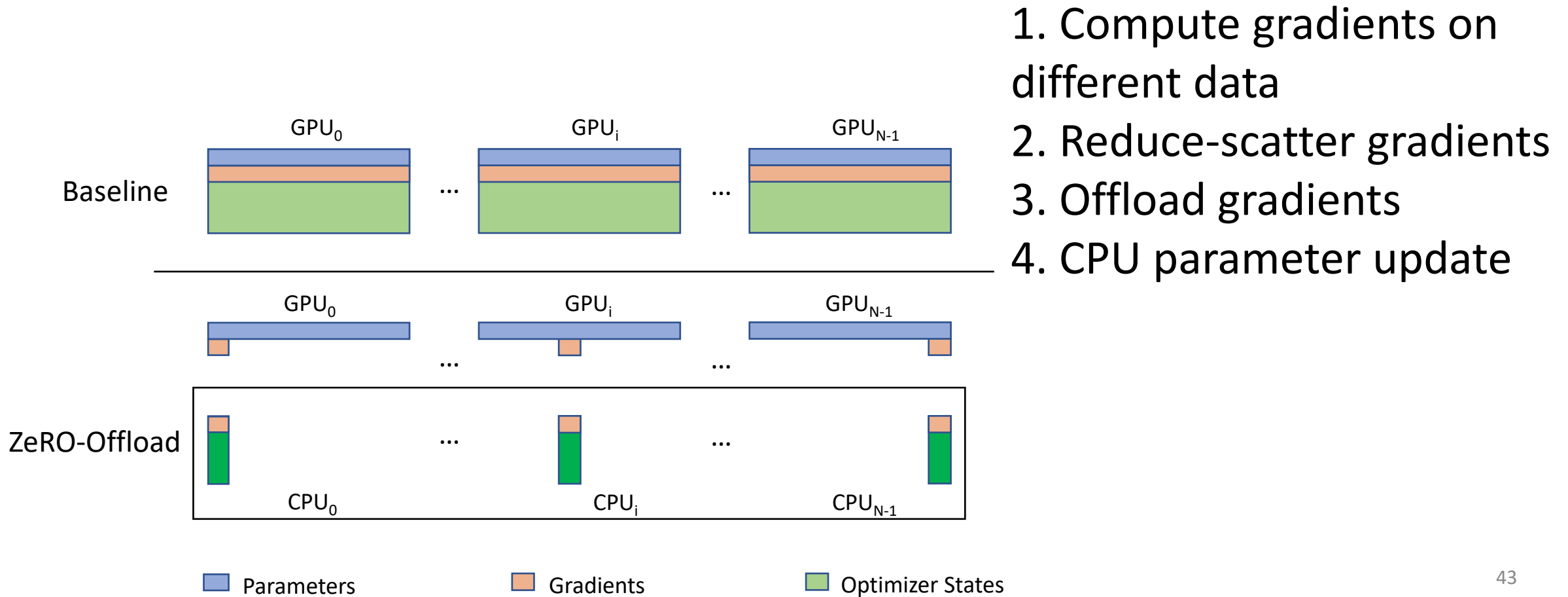
1. Compute gradients on different data
2. Reduce-scatter gradients



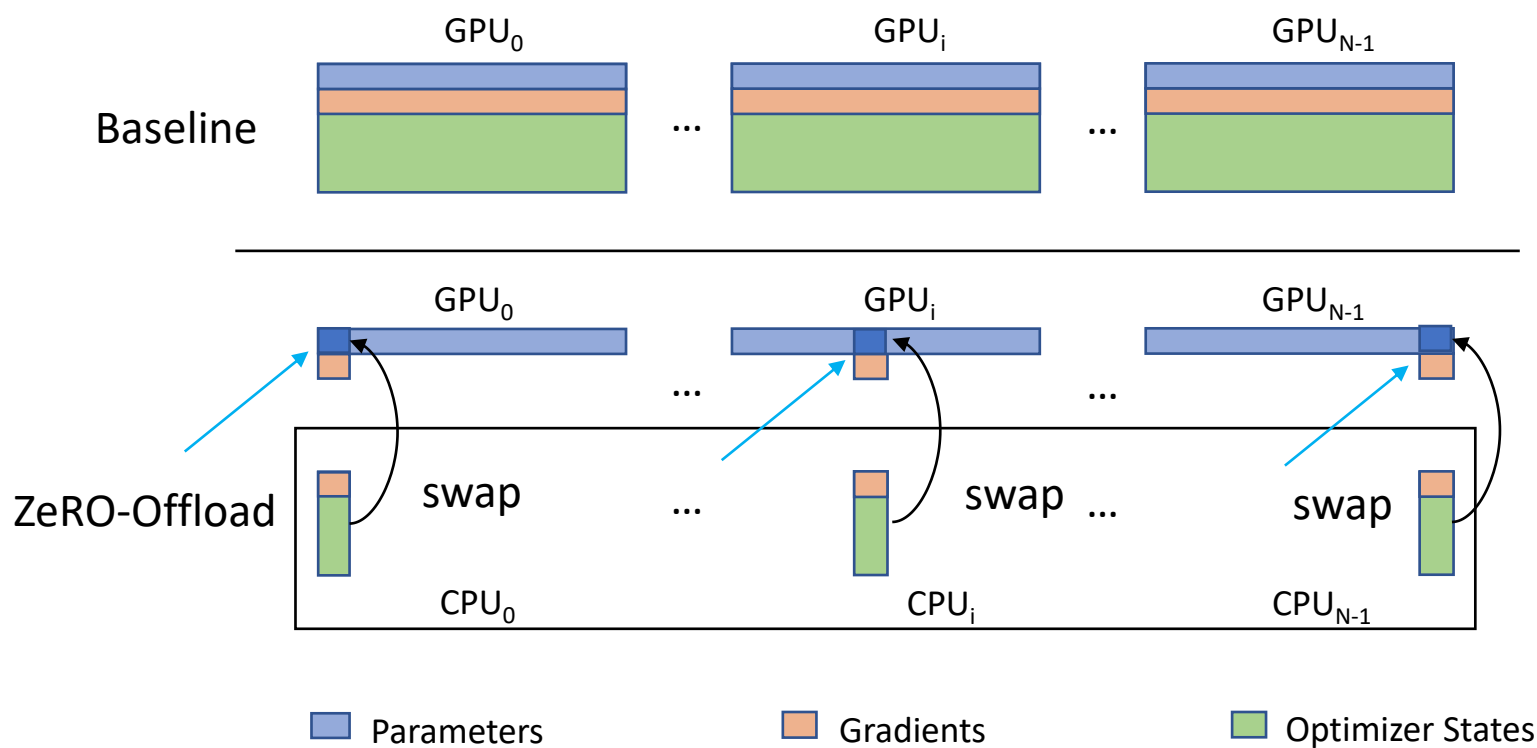
- Partitioning **optimizer states and gradients**



- Partitioning **optimizer states and gradients**

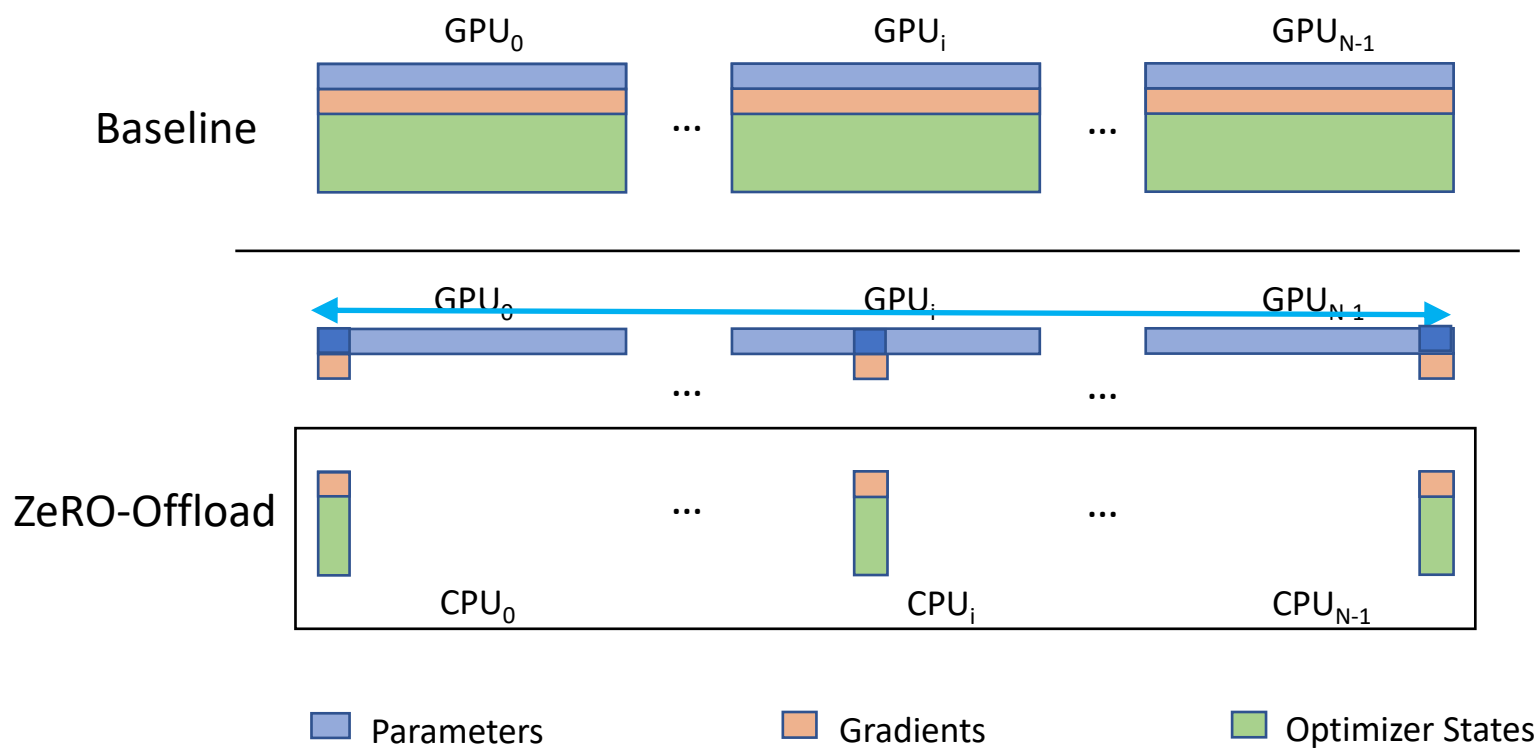


- Partitioning **optimizer states and gradients**



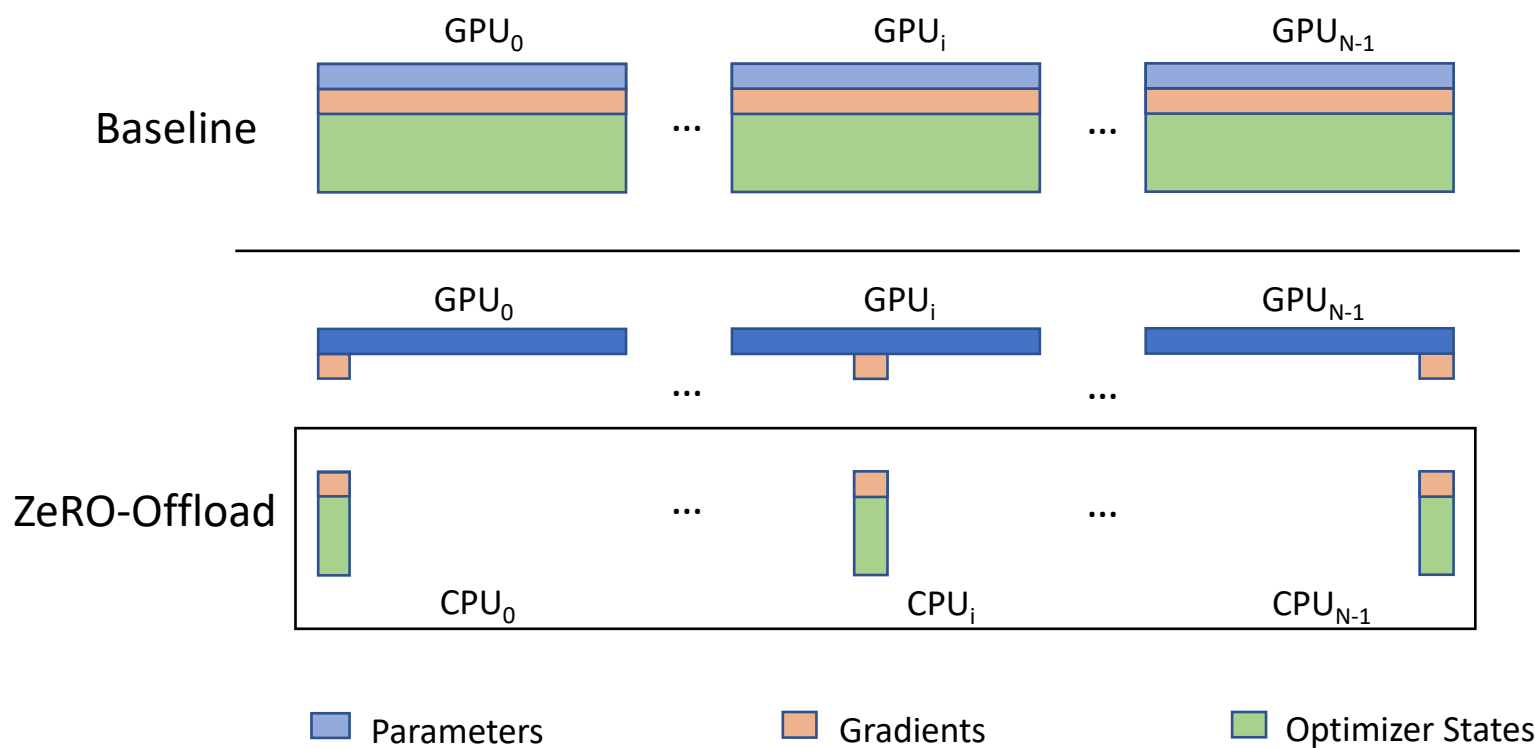
1. Compute gradients on different data
2. Reduce-scatter gradients
3. Offload gradients
4. CPU parameter update
5. Each DP process swaps only the parameter it owns

- Partitioning **optimizer states and gradients**



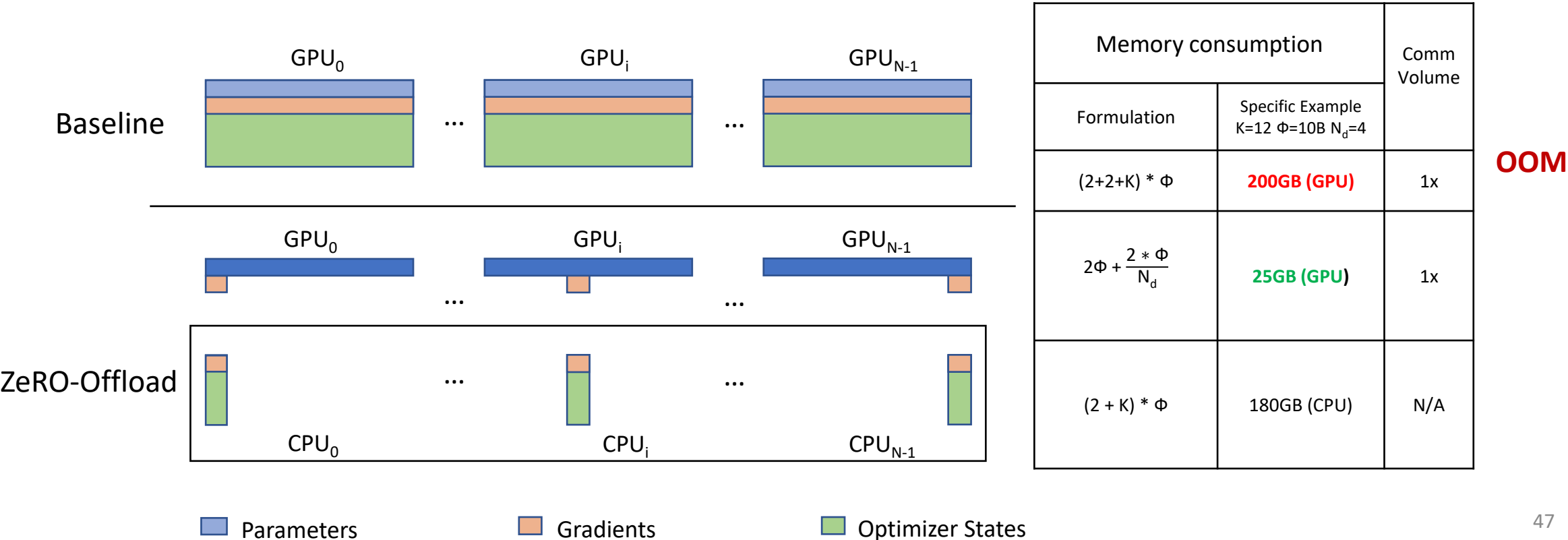
1. Compute gradients on different data
2. Reduce-scatter gradients
3. Offload gradients
4. CPU parameter update
5. Each DP process swaps only the parameter it owns
6. All-gather the parameters

- Partitioning **optimizer states and gradients**



1. Compute gradients on different data
2. Reduce-scatter gradients
3. Offload gradients
4. CPU parameter update
5. Each DP process swaps only the parameter it owns
6. All-gather the parameters

- Benefits:
 - Single copy of model states in CPU regardless of DP degree
 - No additional communication volume in between CPU and GPU

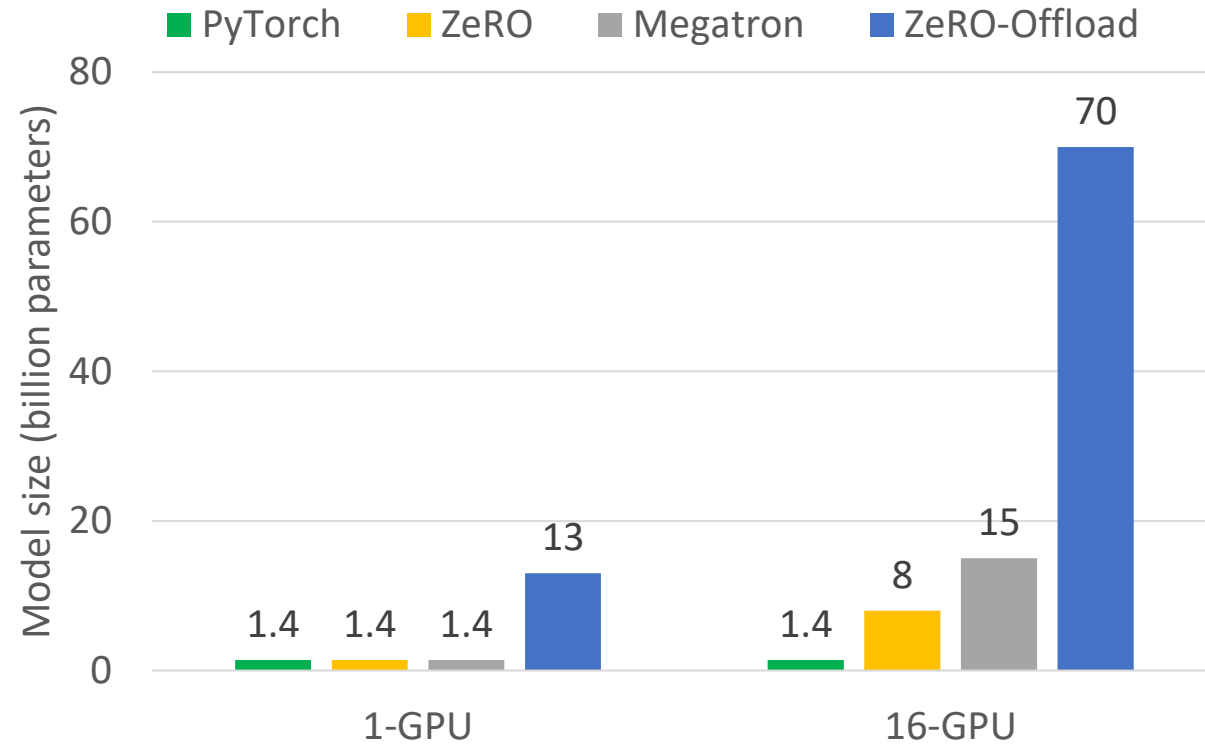


- Testbed

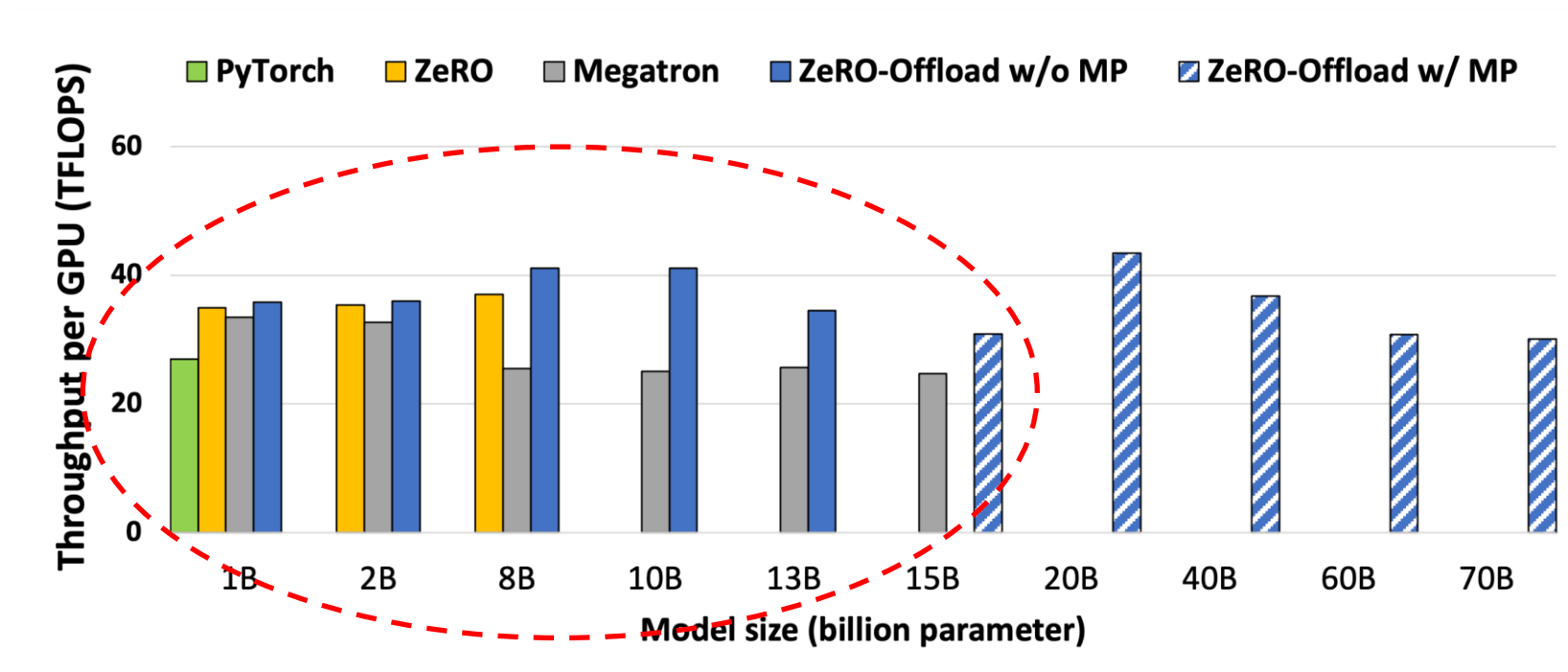
DGX-2 node	
GPU	16 NVIDIA Tesla V100 Tensor Core GPUs
GPU Memory	32GB HBM2 on each GPU
CPU	2 Intel Xeon Platinum 8168 Processors
CPU Memory	1.5TB 2666MHz DDR4
CPU cache	L1, L2, and L3 are 32K, 1M, and 33M, respectively
PCIe	bidirectional 32 GBps PCIe

- Baselines

- 1) Pytorch DDP: distributed data parallelism
- 2) Megatron: model parallelism
- 3) ZeRO: extended data parallelism by eliminating memory redundancies across multiple GPUs

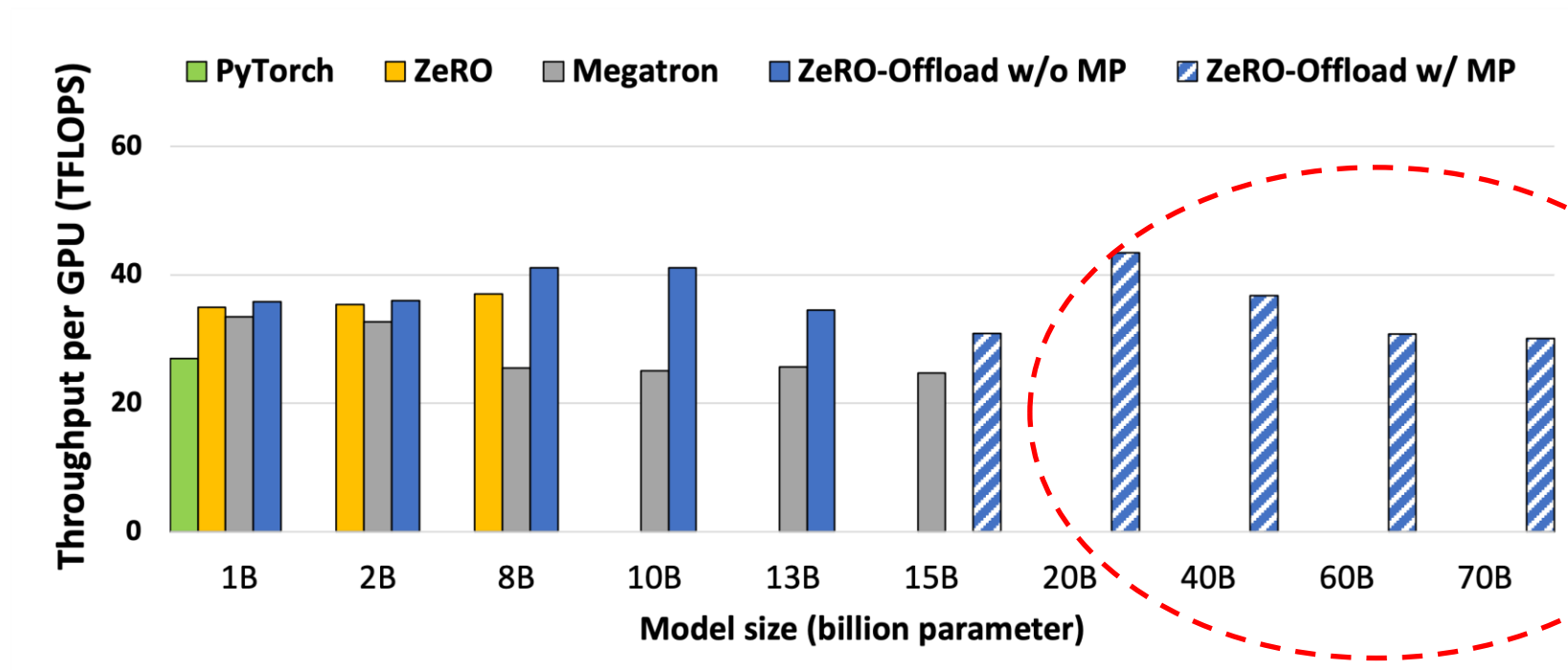


ZeRO-Offload enables 13B model training on a single GPU, and easily enables training of up to 70B parameter with 16 GPUs.



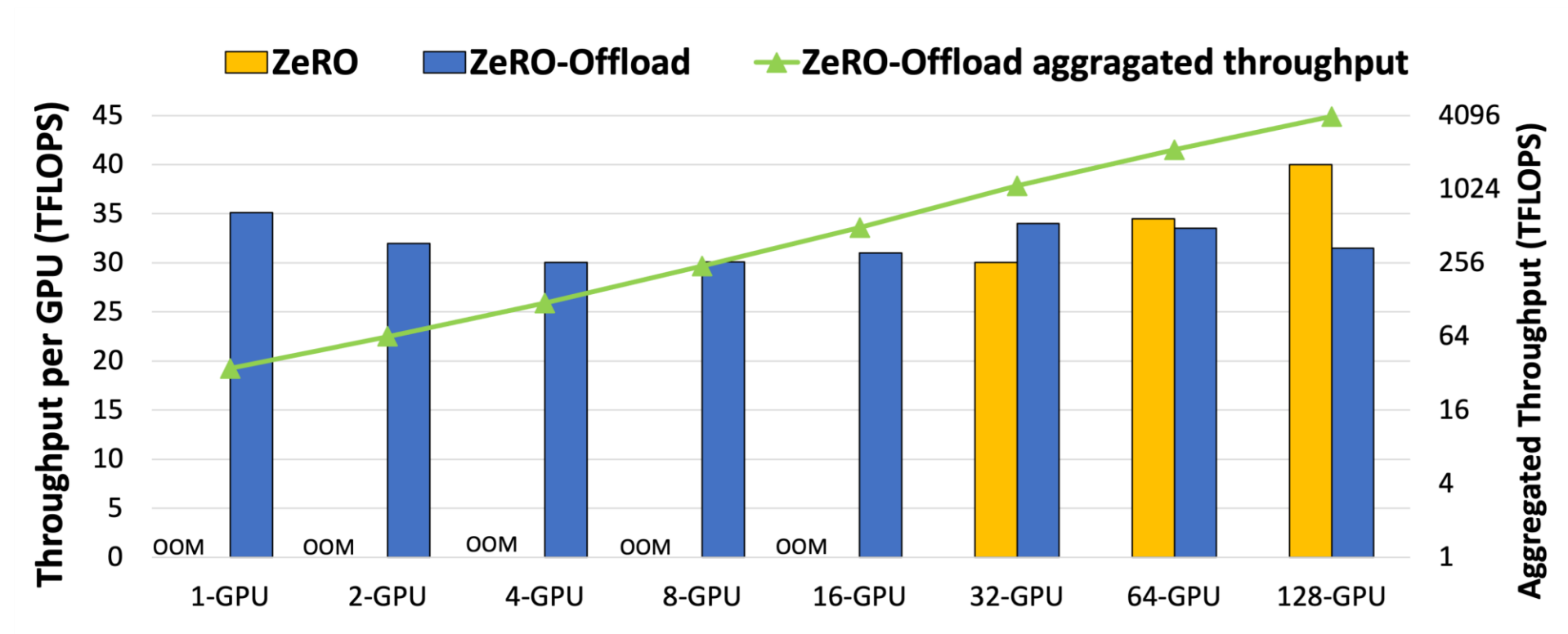
For 1B to 15B models, ZeRO-Offload achieves the highest throughput compared with PyTorch, ZeRO, and Megatron.

Single DGX-2 node (x16 V100-32GB)



Combined with model parallelism, ZeRO-Offload enables training up to 70B parameter models with more than 30 TFLOPS throughput per GPU.

Single DGX-2 node (x16 V100-32GB)



ZeRO-Offload achieves near perfect linear speedup in terms of aggregated throughput running at over 30 TFlops per GPU.

Questions?