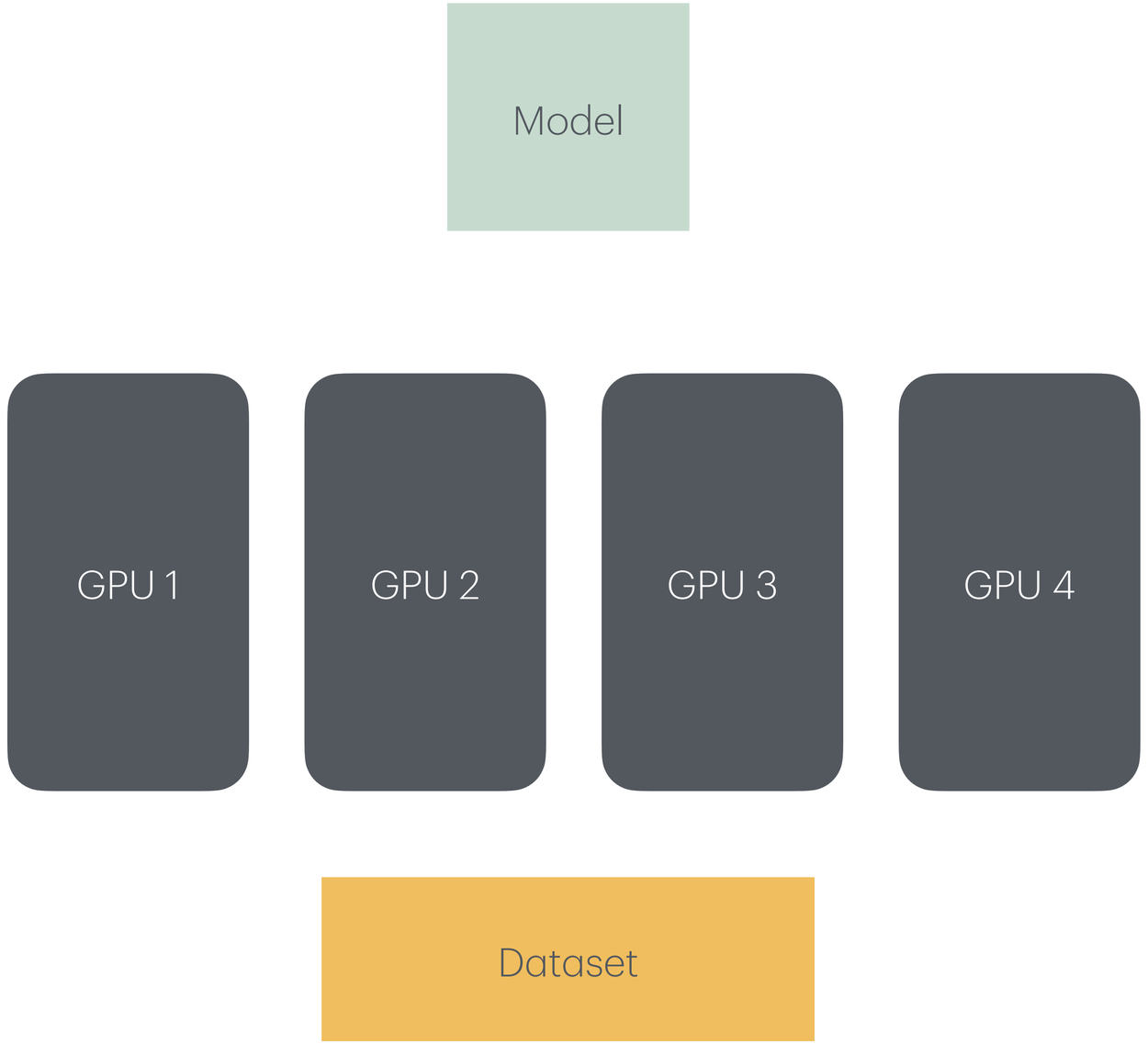
Zero redundancy training

Philipp Krähenbühl, UT Austin

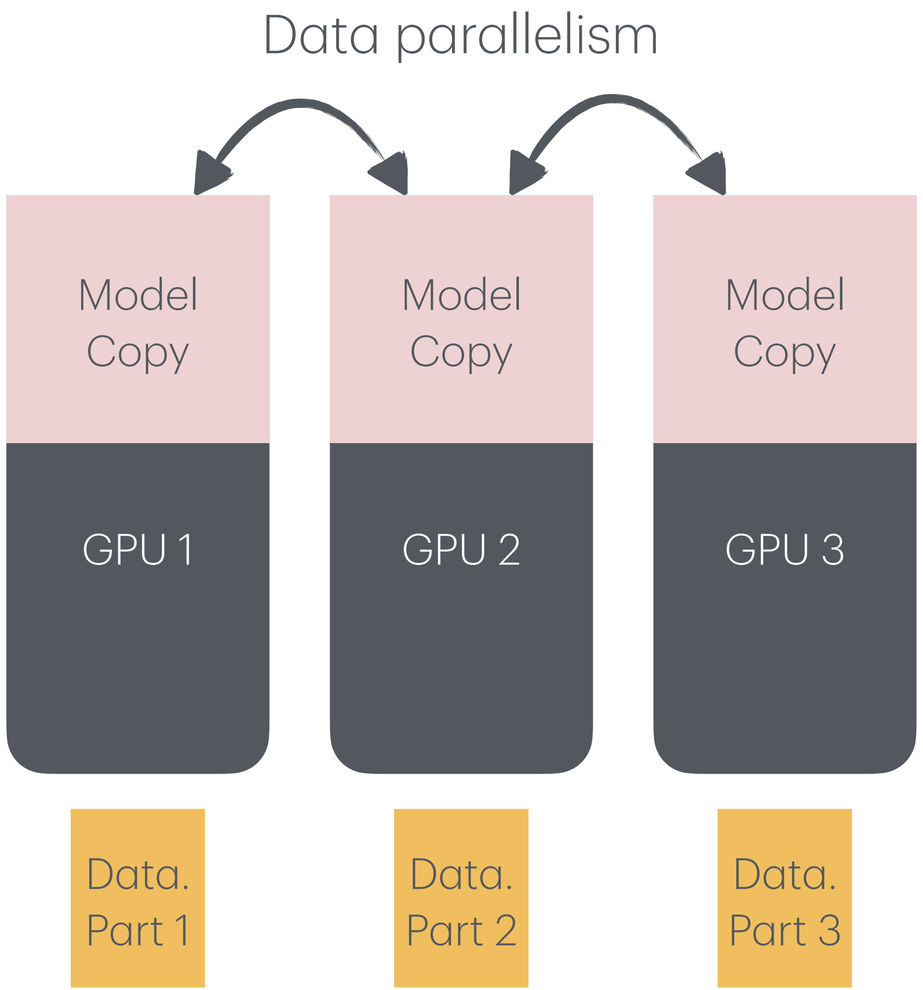
Training on multiple GPUs

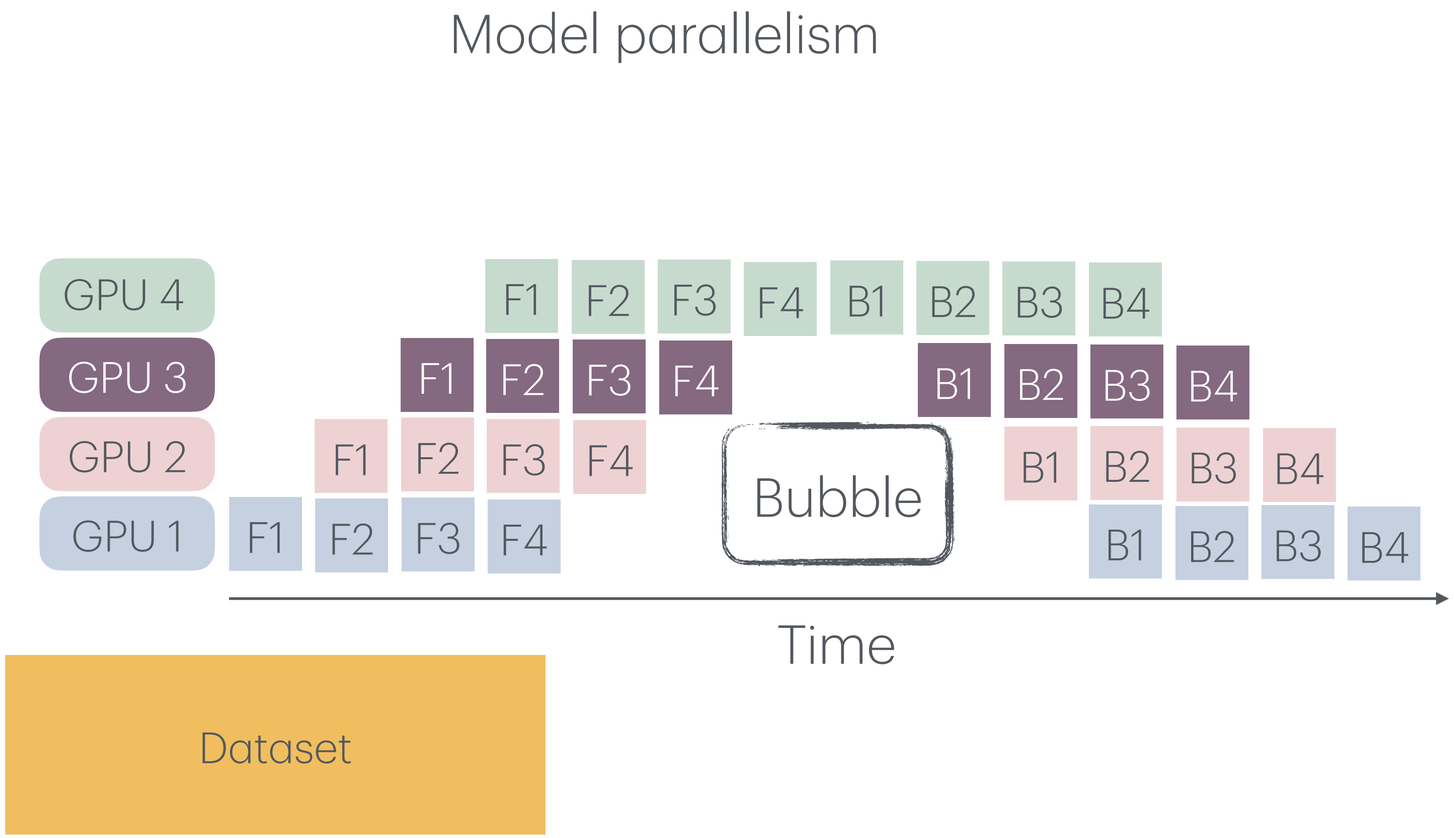
Setting

- One model
  - One set of weights
- Large dataset
- Multiple GPUs



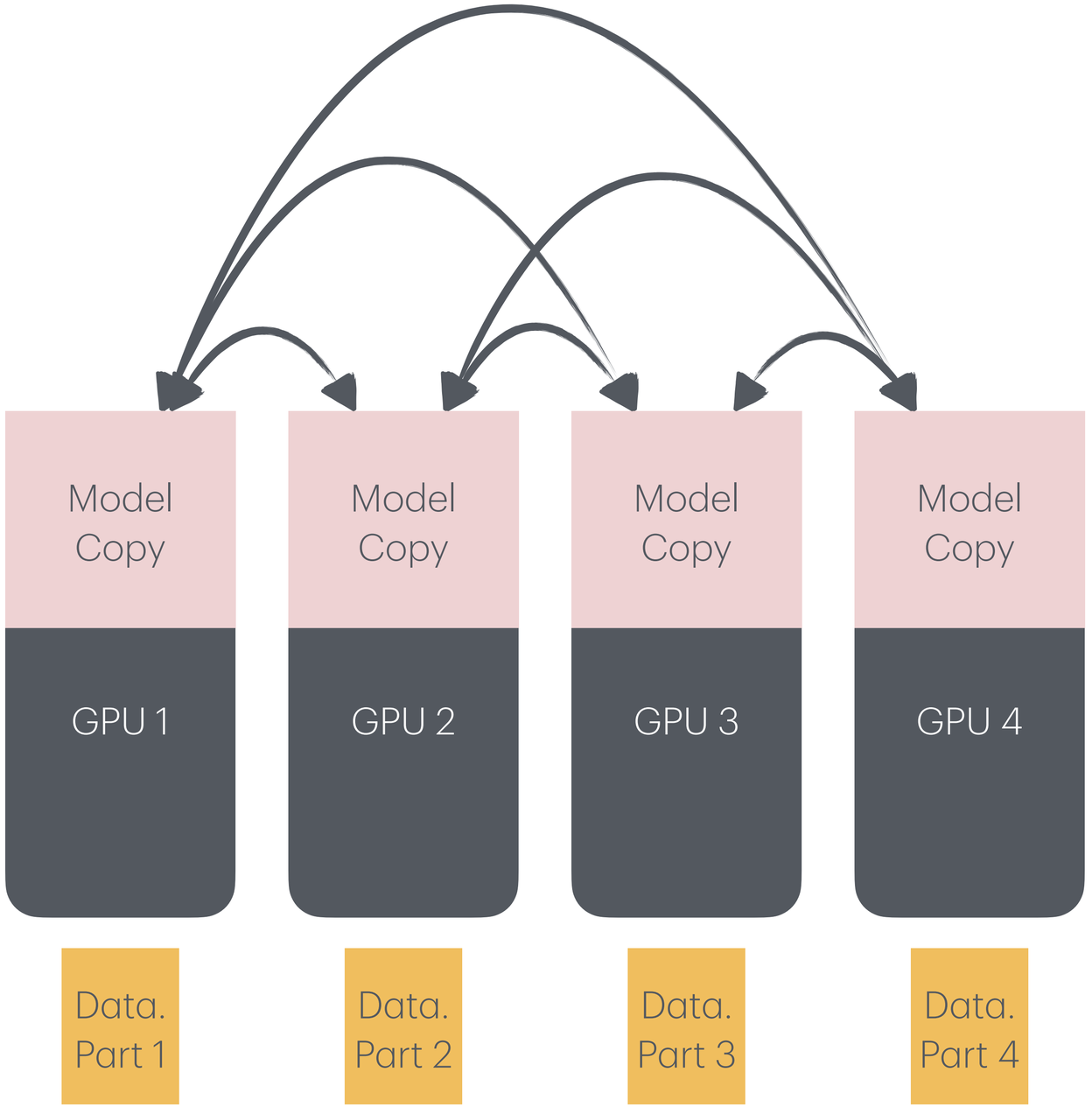
Training on multiple GPUs





Zero redundancy

- Recall: Memory use
  - Model weights (2-4N bytes)
  - Gradients (2-4N bytes)
  - Optimizer State (6-8N bytes)
- Optimizer state only used once per step
  - Why not distribute this across GPUs

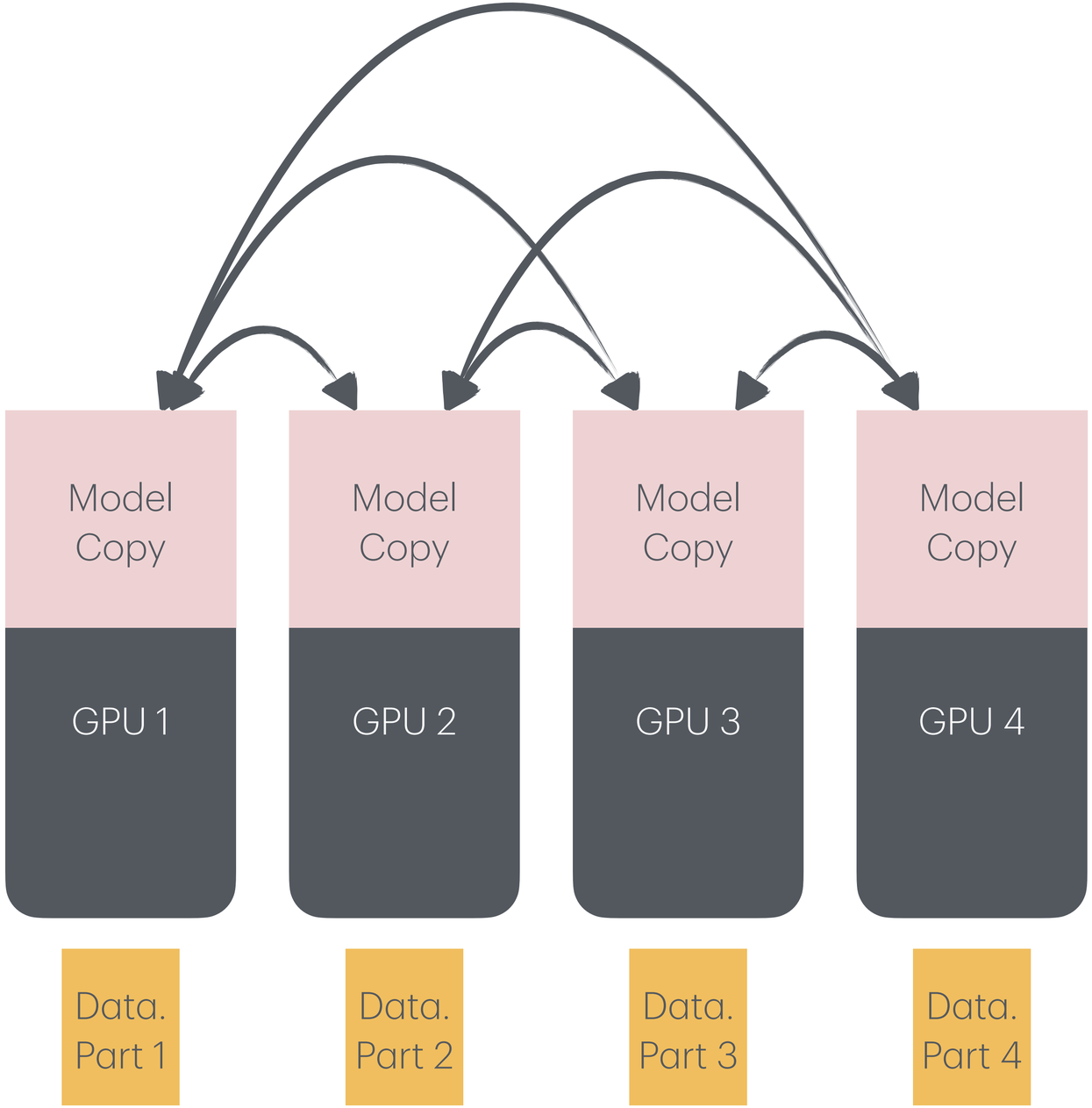


[1] Rajbhandari et al. Zero: Memory optimizations toward training trillion parameter models. 2019

Zero-1

Optimizer State Partitioning

- For M GPUs, each GPU
  - bfloat16 weights (2N bytes)
  - bfloat16 gradients (2N bytes)
  - float32 weights, first, second momentum (3 * 4N/M bytes)
- Forward/backward in bfloat16
- Optimizer step: gather gradients, scatter updated weights

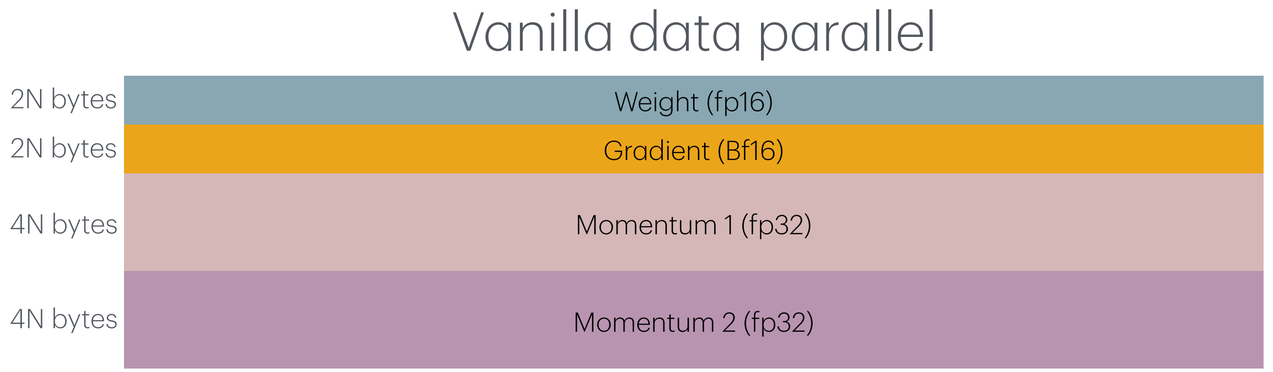


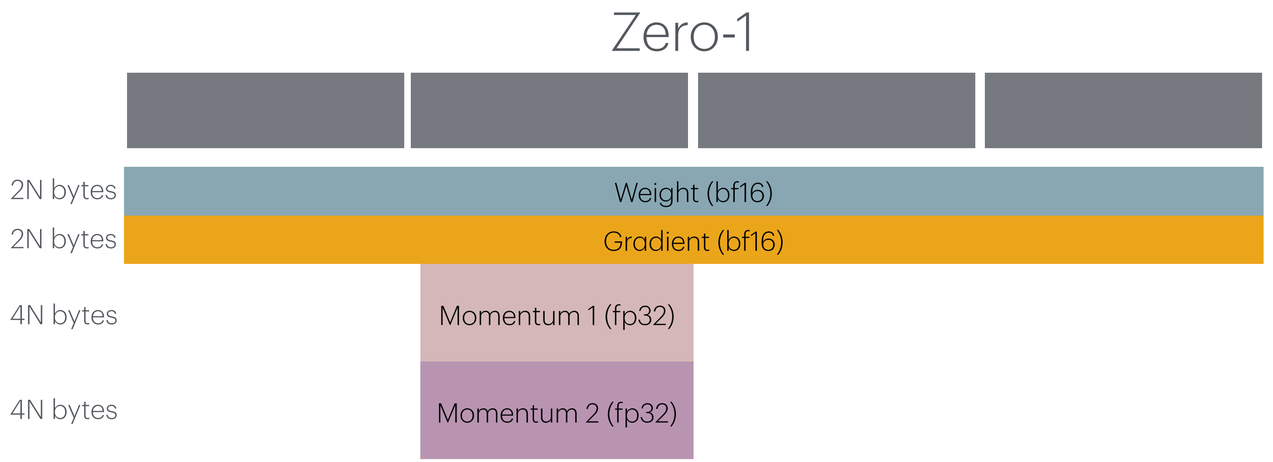
[1] Rajbhandari et al. Zero: Memory optimizations toward training trillion parameter models. 2019

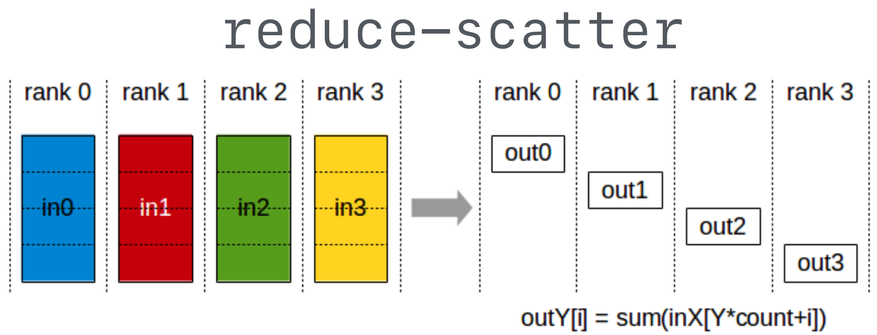
Zero-1

Optimizer State Partitioning

- Reduces memory consumption
  - 12-16 N bytes -> 4N + 12N/M bytes
  - 3-4 × for large enough M
- Requires synchronization through `reduce-scatter`





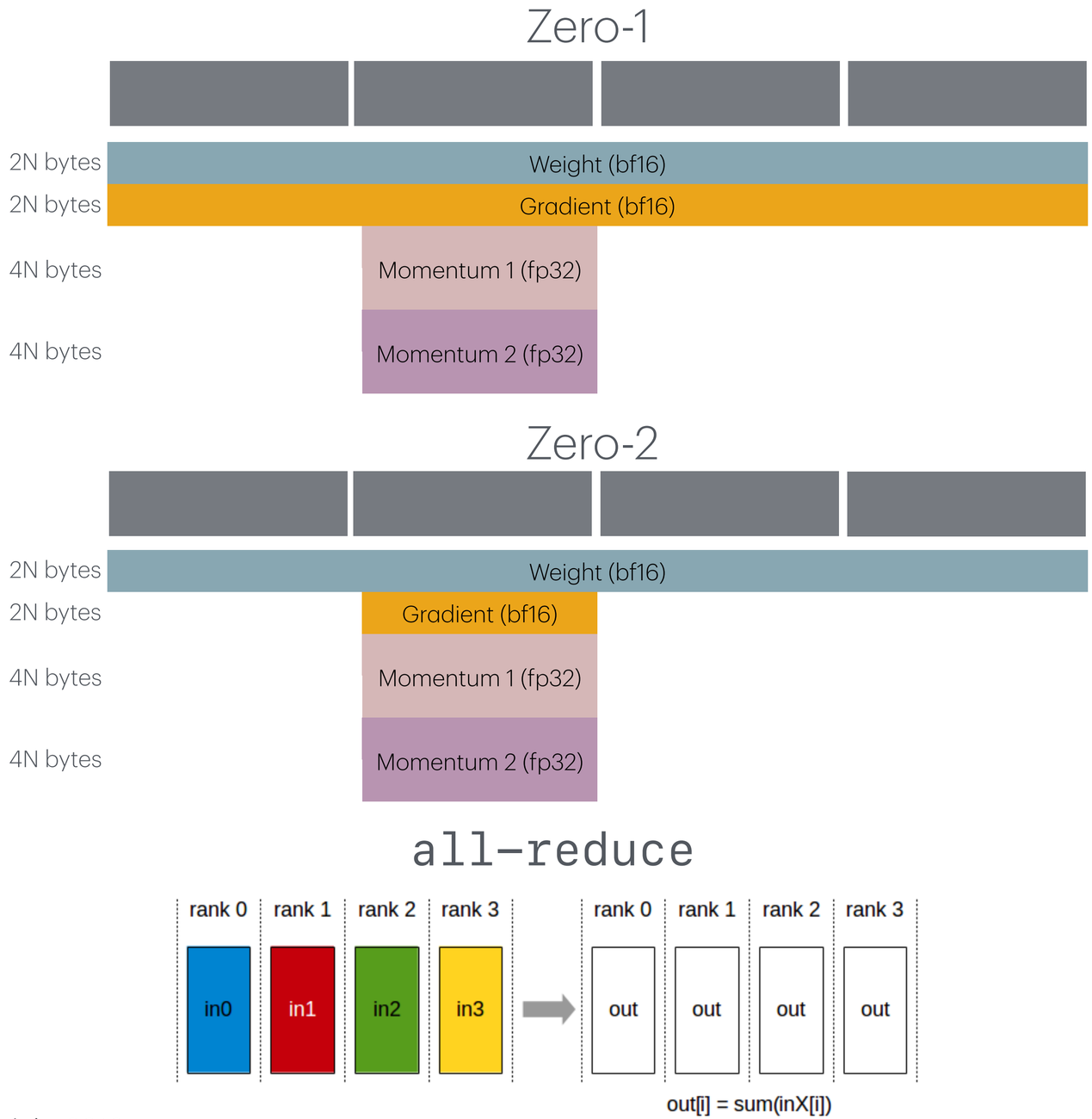


[1] Rajbhandari et al. Zero: Memory optimizations toward training trillion parameter models. 2019
[2] Nvidia: https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/usage/collectives.html#allreduce

Zero-2

Optimizer State Partitioning

- Distributed gradient among GPUs
- In backward
  - Compute gradient
  - `all-reduce`
  - Keep gradient corresponding to GPUs optimizer state

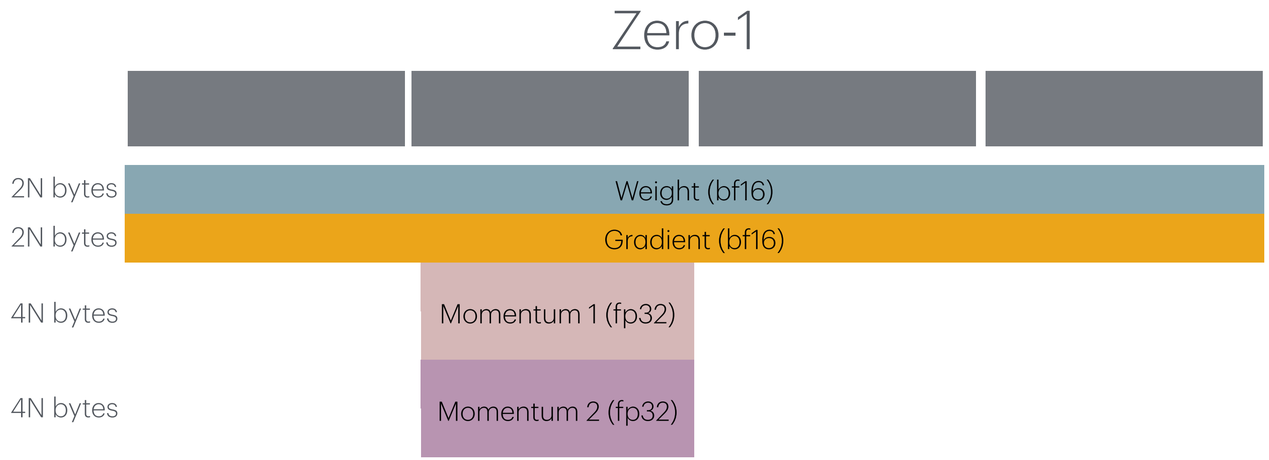


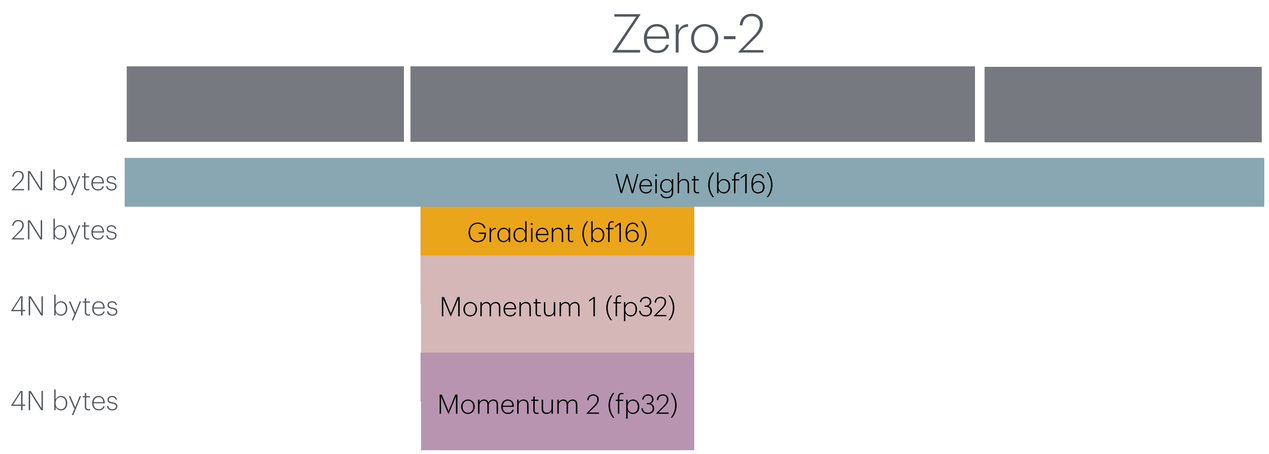
[1] Rajbhandari et al. Zero: Memory optimizations toward training trillion parameter models. 2019
[2] Nvidia: https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/usage/collectives.html#allreduce

Zero-2

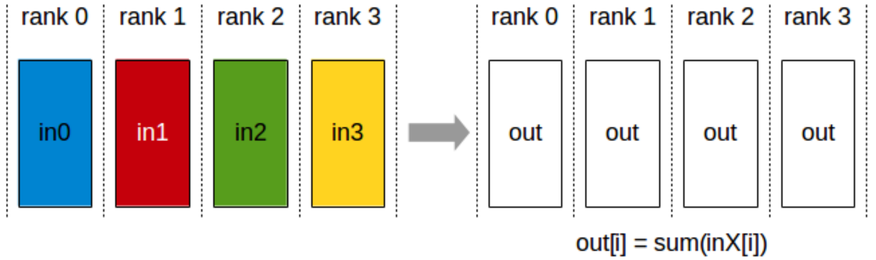
Optimizer State Partitioning

- Reduces memory consumption
  - 12-16 N bytes -> 2N + 16N/M bytes
  - 6-8 × for large enough M
- Requires synchronization through `reduce-scatter, all-reduce`





`all-reduce`

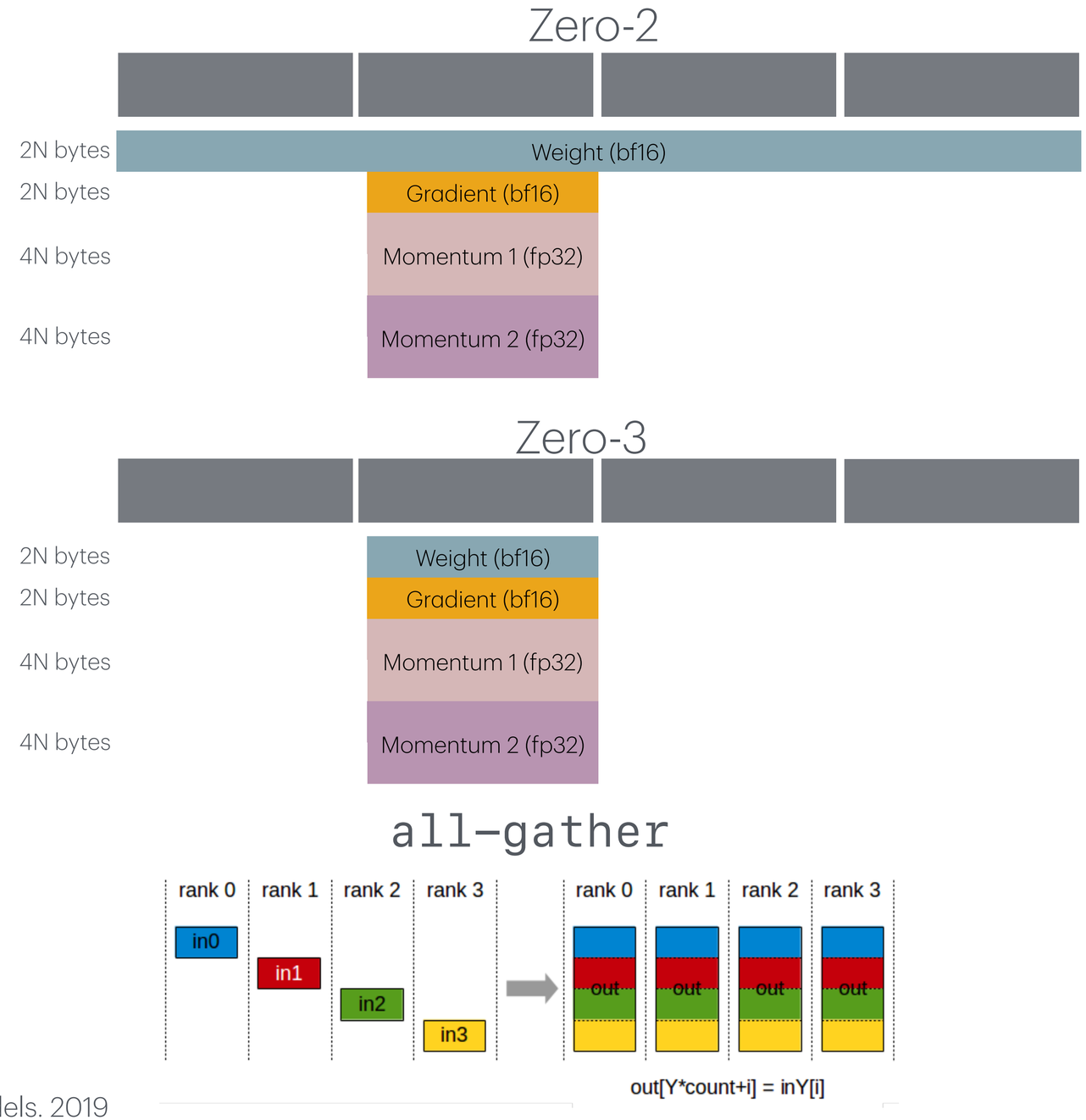


[1] Rajbhandari et al. Zero: Memory optimizations toward training trillion parameter models. 2019
[2] Nvidia: https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/usage/collectives.html#allreduce

Zero-3

Optimizer State Partitioning

- Only store weights that are currently required
- In forward/backward
  - `all-gather`
  - Compute
  - Discard weights
- Use bf16 weights or original fp32

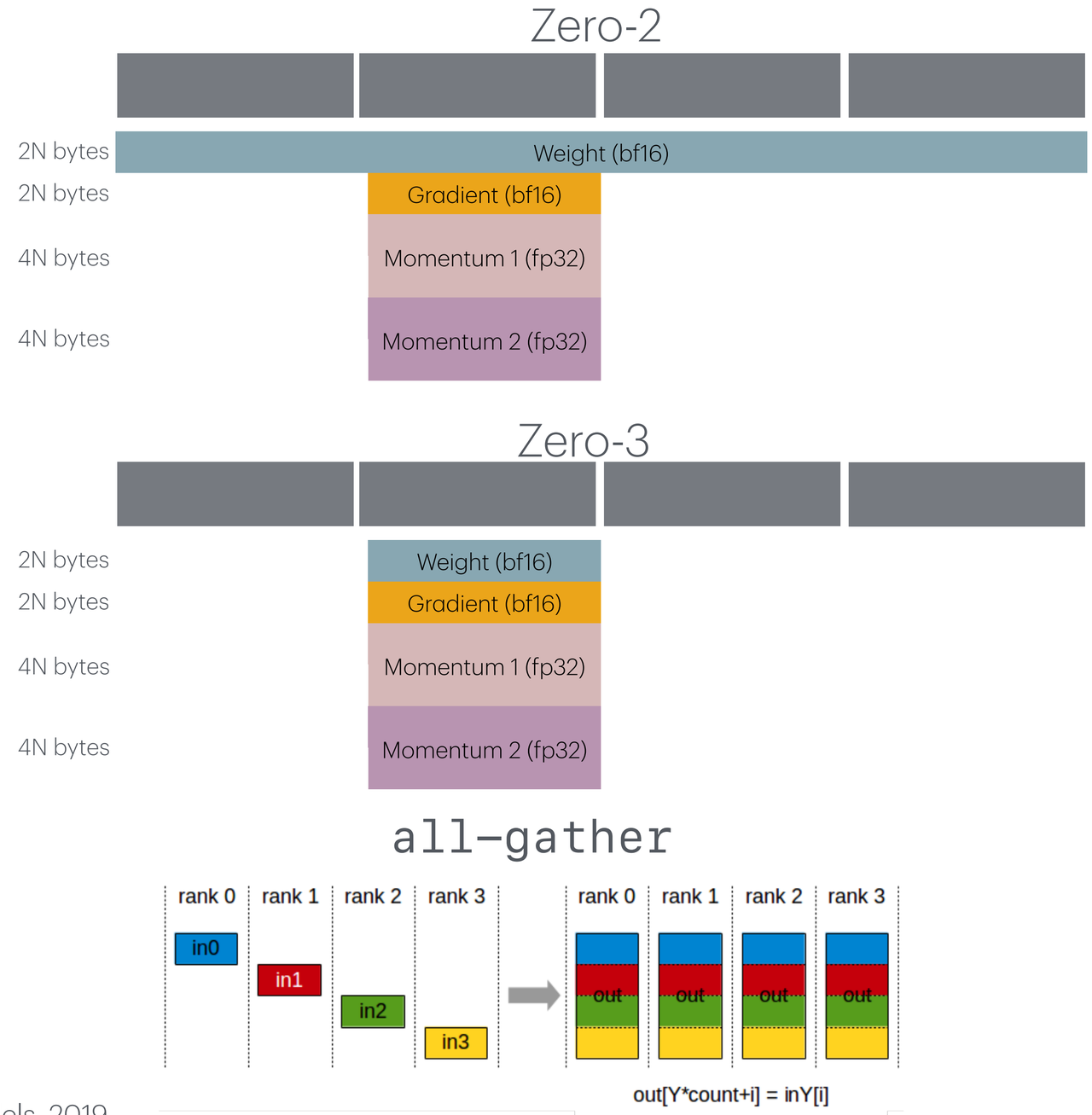


[1] Rajbhandari et al. Zero: Memory optimizations toward training trillion parameter models. 2019
[2] Nvidia: https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/usage/collectives.html#allreduce

Zero-3

Optimizer State Partitioning

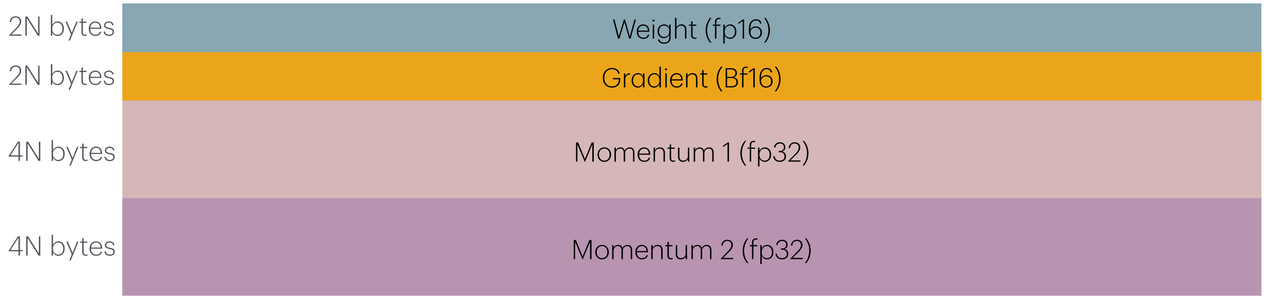
- Reduces memory consumption
  - 12-16 N bytes -> 16N/M bytes
  - 50+ × for large enough M
- Requires synchronization through `reduce-scatter, all-reduce, all-gather` for each layer



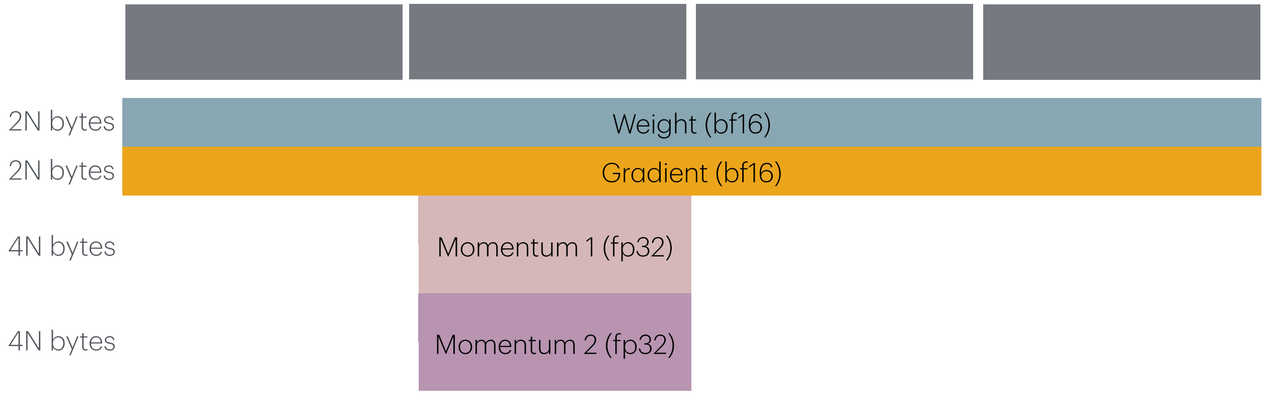
[1] Rajbhandari et al. Zero: Memory optimizations toward training trillion parameter models. 2019
[2] Nvidia: https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/usage/collectives.html#allreduce

Recap

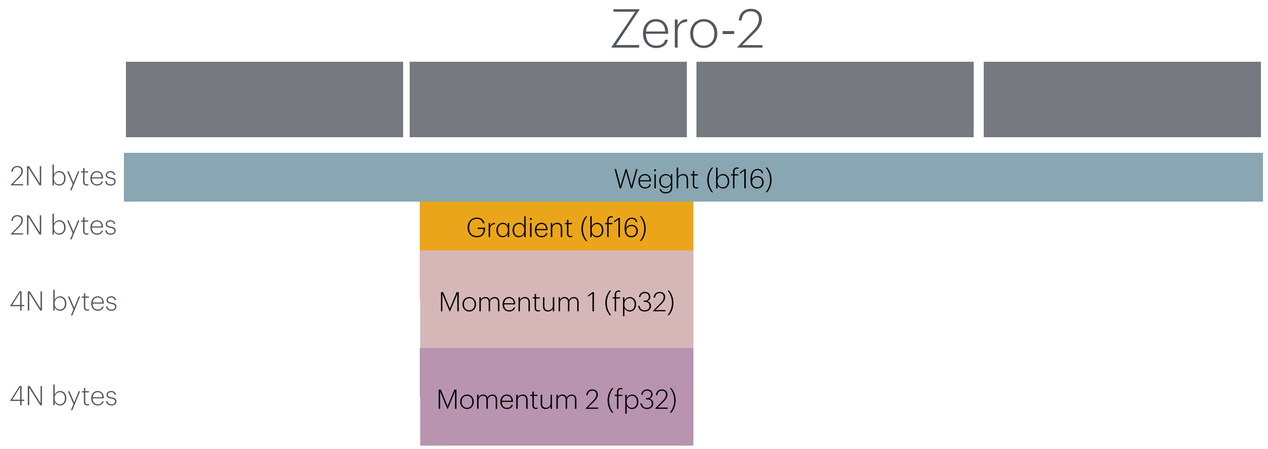
Vanilla data parallel



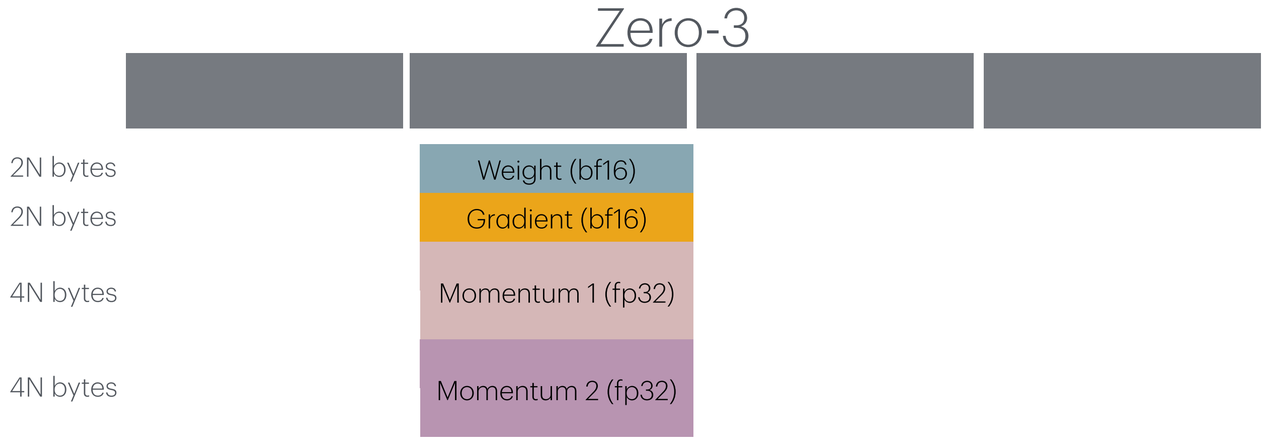
Zero-1



Zero-2

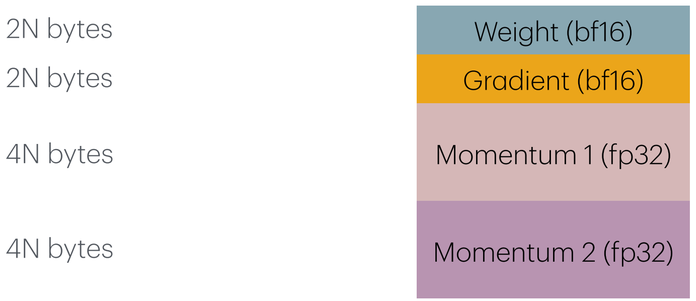


Zero-3

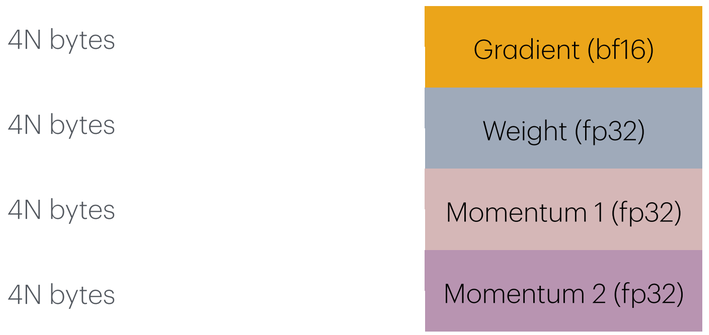


Fully Sharded Data Parallel

Zero-3



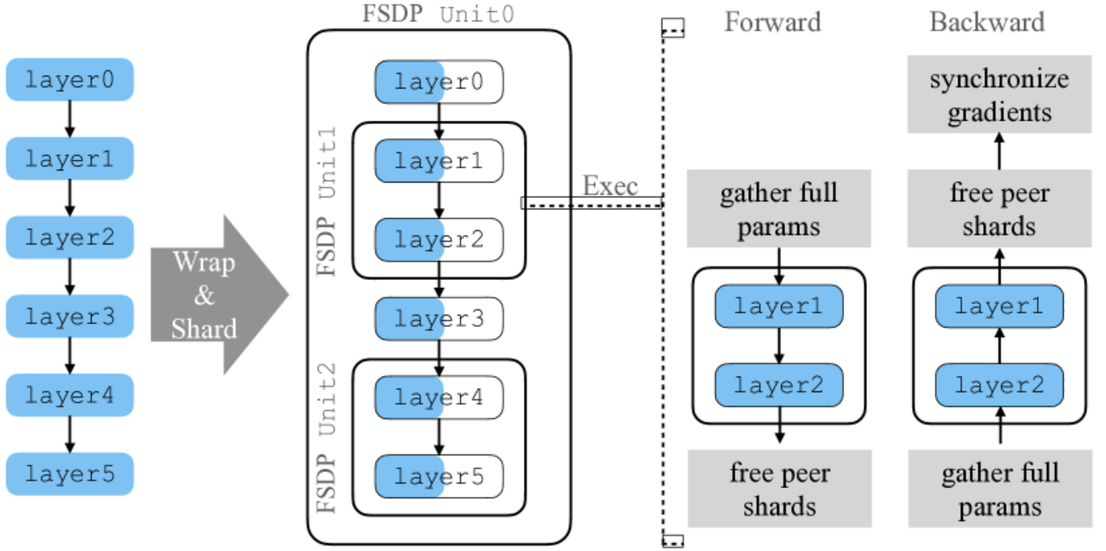
FSDP

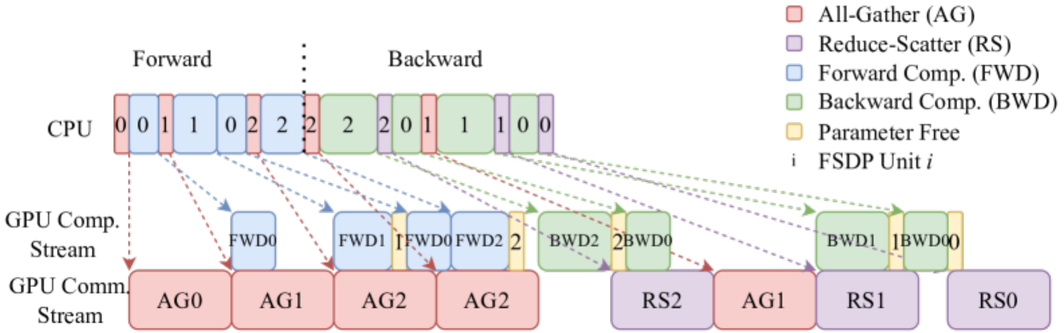


[1] Zhao et al. PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel. 2023

Fully Sharded Data Parallel

- FSDP: Efficient implementation of Zero-3 in PyTorch
  - Synchronize group's (Unit's) of layers
  - Efficient scheduling of communication and computation

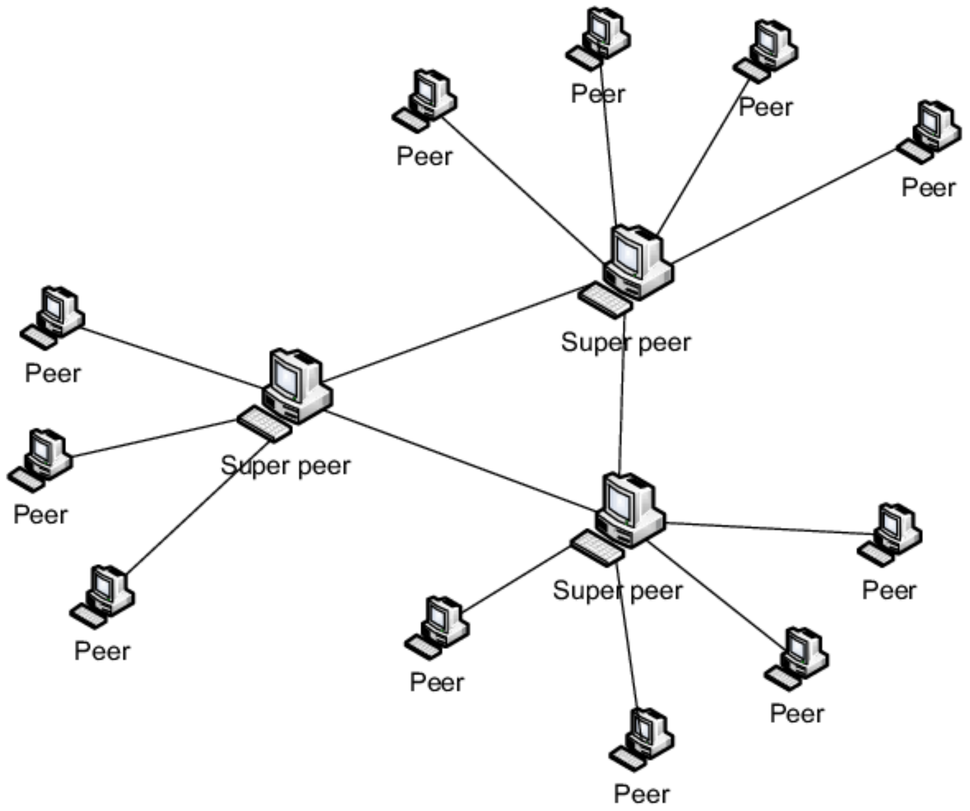




[1] Zhao et al. PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel. 2023

Fully Sharded Data Parallel

- Hybrid Sharding
  - 8-16 GPUs per server
  - Shard within server
  - Regular data parallel (share gradient) between server

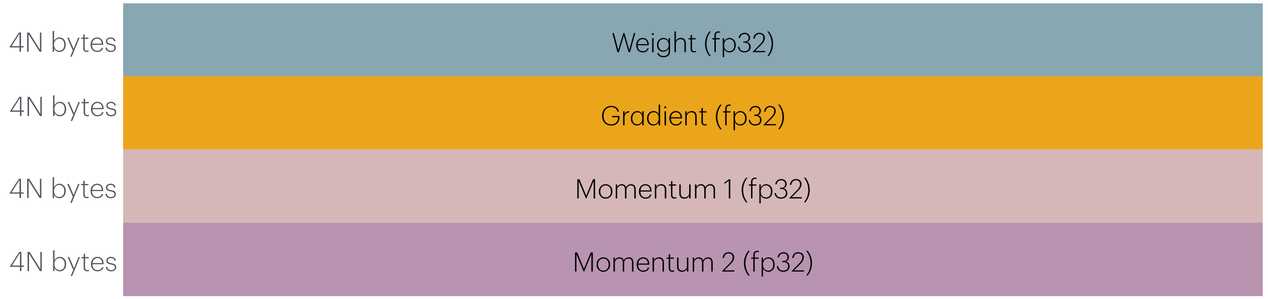


[1] Zhao et al. PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel. 2023
[2] https://www.researchgate.net/figure/Hybrid-peer-to-peer-architecture_fig1_267205698

Training large models

Memory requirements

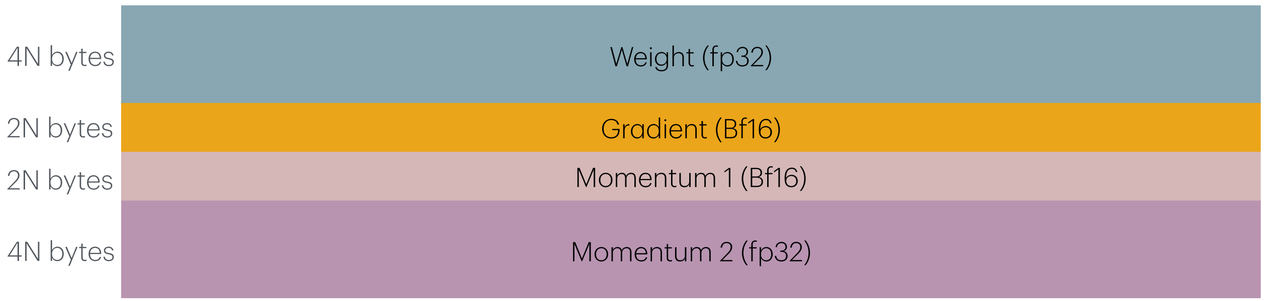
- Without optimization:
  - Model parameters: N
  - Weights: N floats
  - Gradients: N floats
  - Momentum: N floats
  - 2nd momentum (ADAM): N floats
- 16N bytes without counting activations



Training large models

Memory requirements

- Mixed precision
  - Model parameters: N
  - Weights: N floats
  - Gradients: N bfloat16
  - Momentum: N bfloat16
  - 2nd momentum (ADAM): N floats
- 12N bytes without counting activations

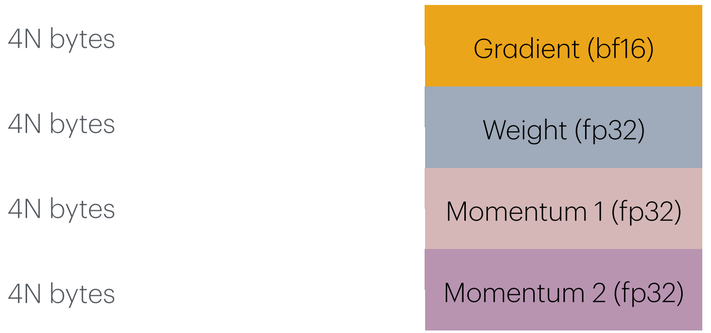


Training large models

Memory requirements

- Zero / FSDP
- 16N / M bytes without counting activations
- For M GPUs
  - Good solution for GPU-rich people

FSDP



References

- [1] Rajbhandari et al. Zero: Memory optimizations toward training trillion parameter models. 2019. (link)

- [2] Zhao et al. PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel. 2023. (link)