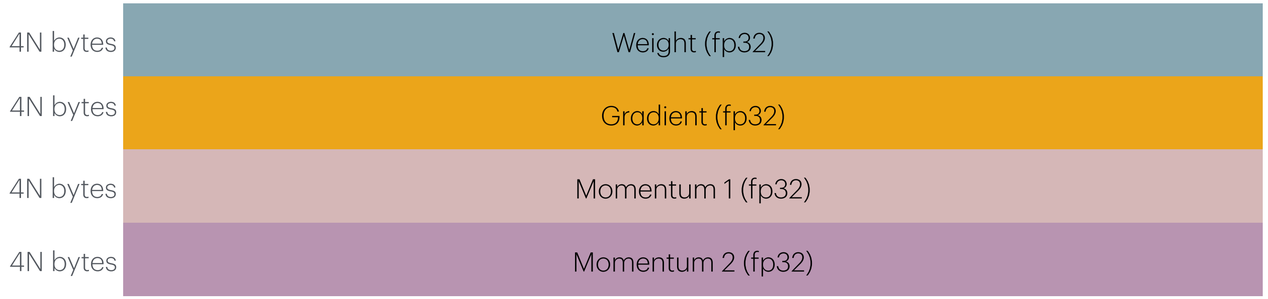
Distributed Training

Philipp Krähenbühl, UT Austin

Training large models

Memory requirements

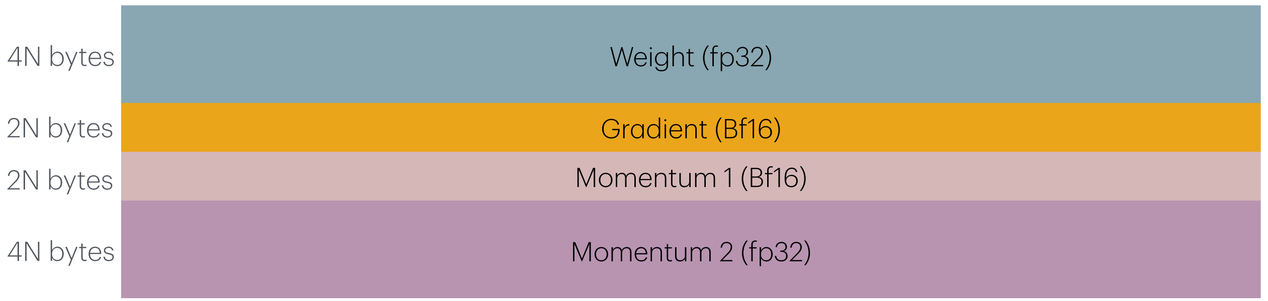
- Without optimization:
  - Model parameters: N
  - Weights: N floats
  - Gradients: N floats
  - Momentum: N floats
  - 2nd momentum (ADAM): N floats
- 16N bytes without counting activations



Training large models

Memory requirements

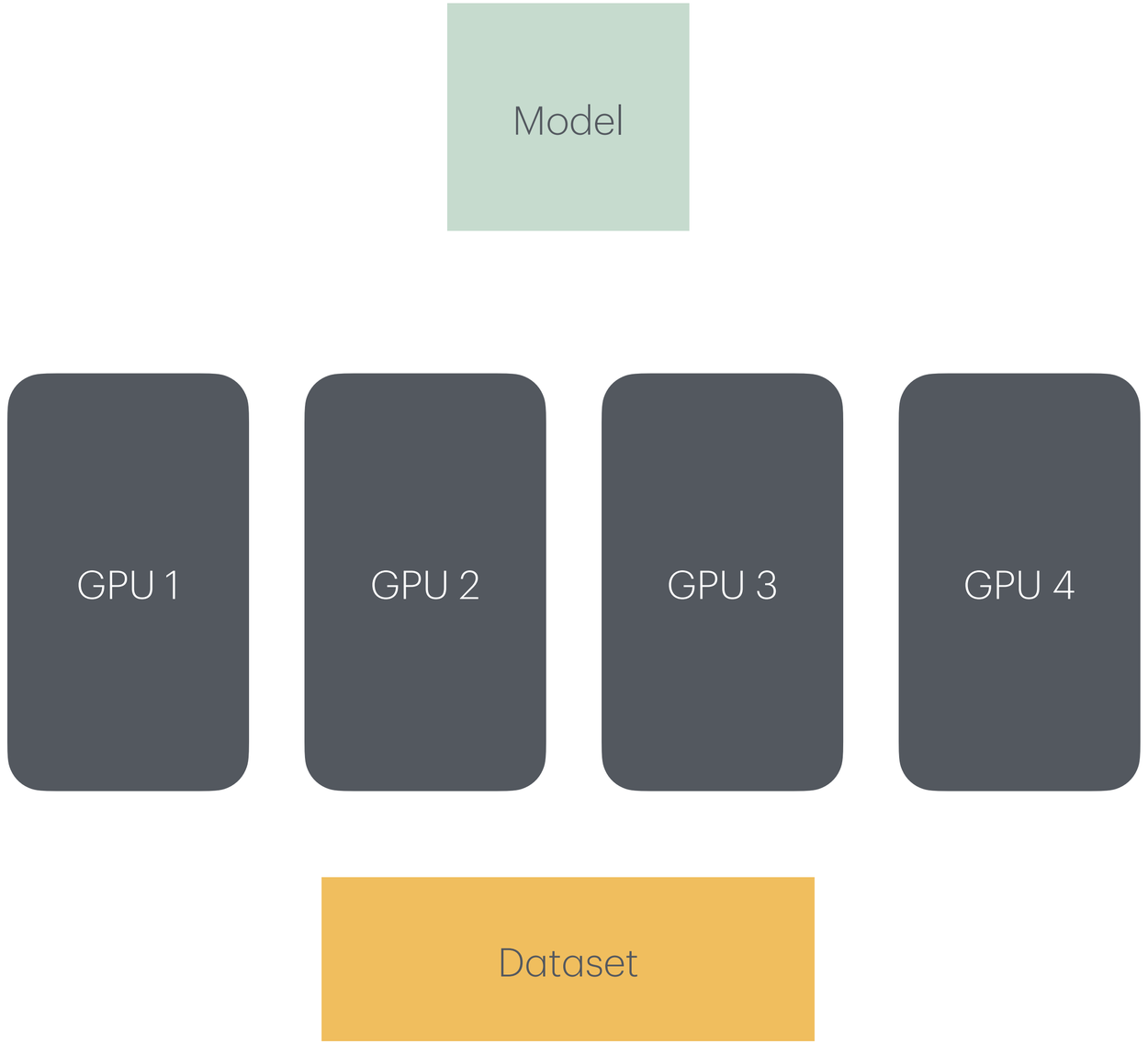
- Mixed precision
  - Model parameters: N
  - Weights: N floats
  - Gradients: N bfloat16
  - Momentum: N bfloat16
  - 2nd momentum (ADAM): N floats
- 12N bytes without counting activations



Training on multiple GPUs

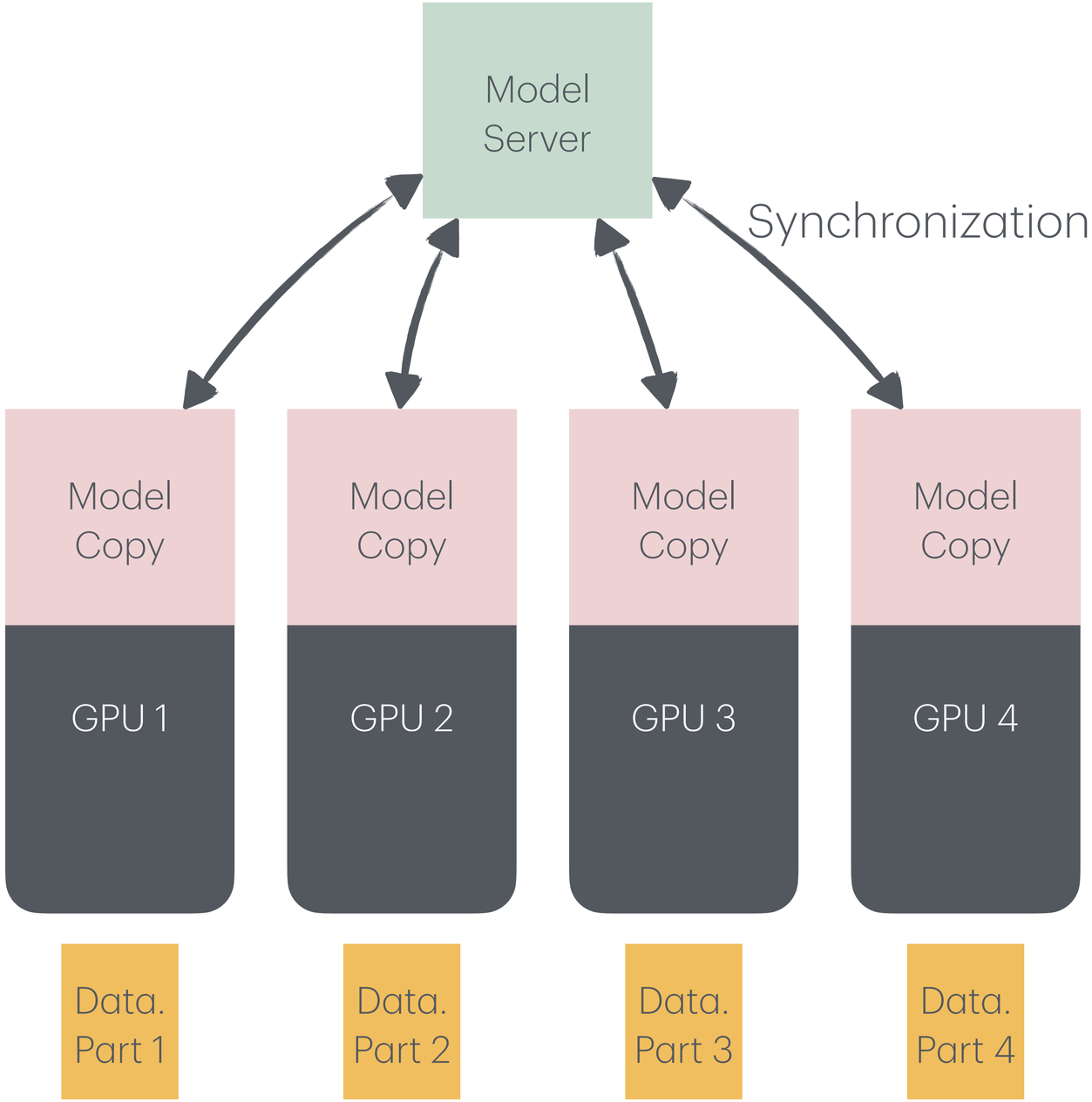
Setting

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



Solution 1: Data parallelism

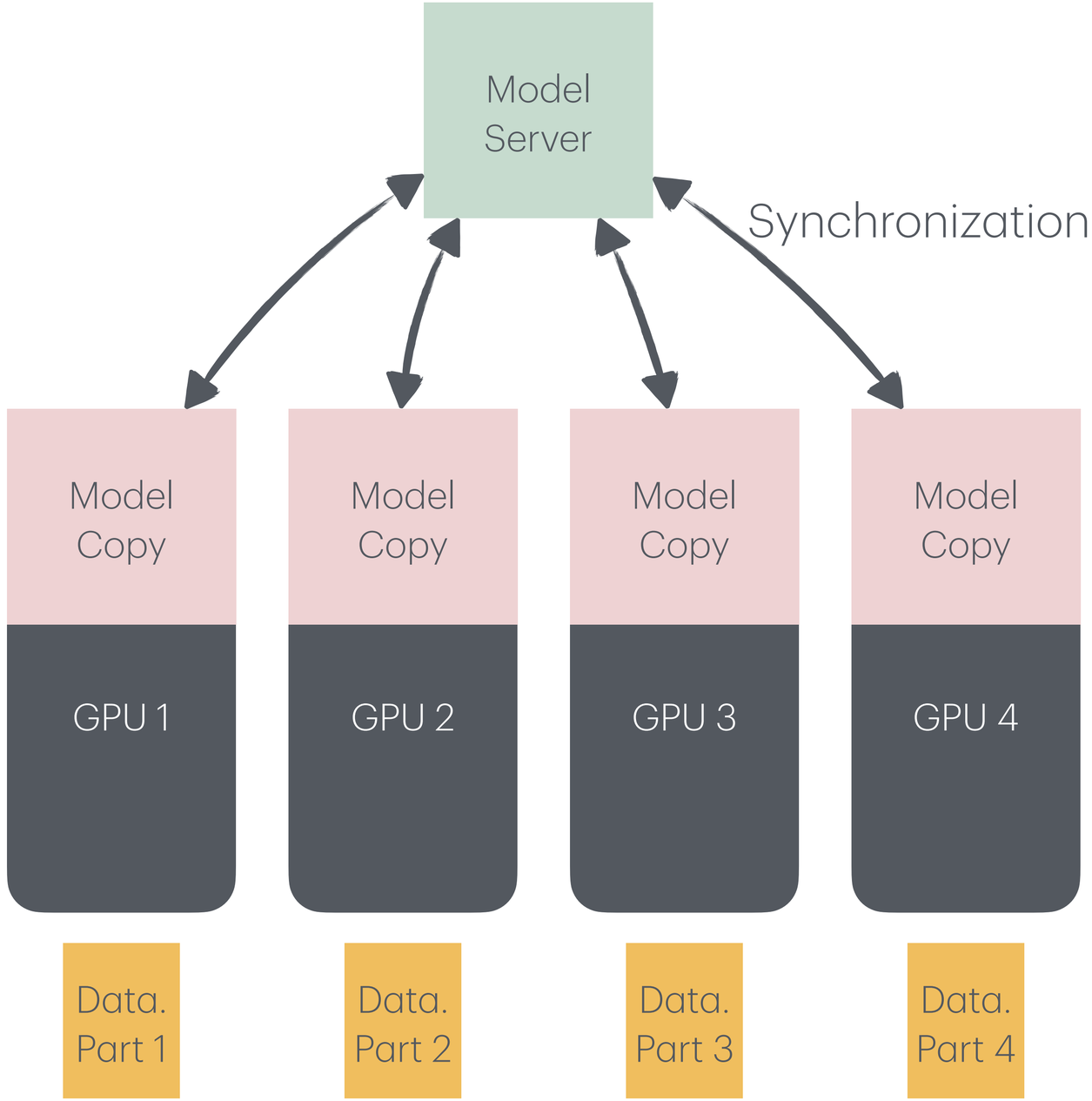
- One model copy per GPU
- Split data among GPUs
- Synchronize model weights
  - Gradients to server
  - New weights come back
- Optimizer on server



[1] Jeff Dean, et al. Large scale distributed deep networks, 2012.

Solution 1: Data parallelism

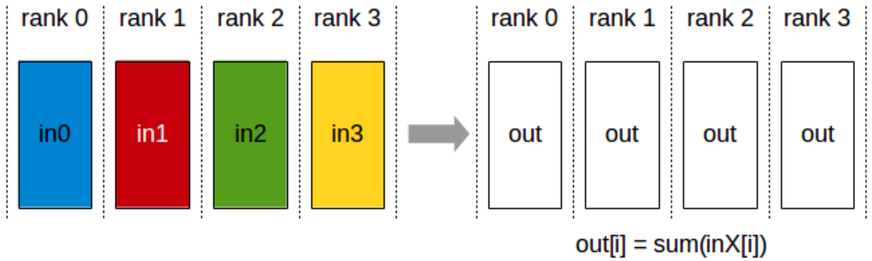
- Advantages
  - Close to $n \times$ speedup for $n \leq 8$ GPUs
  - Simple to implement
- Issues
  - Synchronization is slow for many GPUs
  - Data transfer $2 \times$ #weights
  - Model still needs to fit on GPU

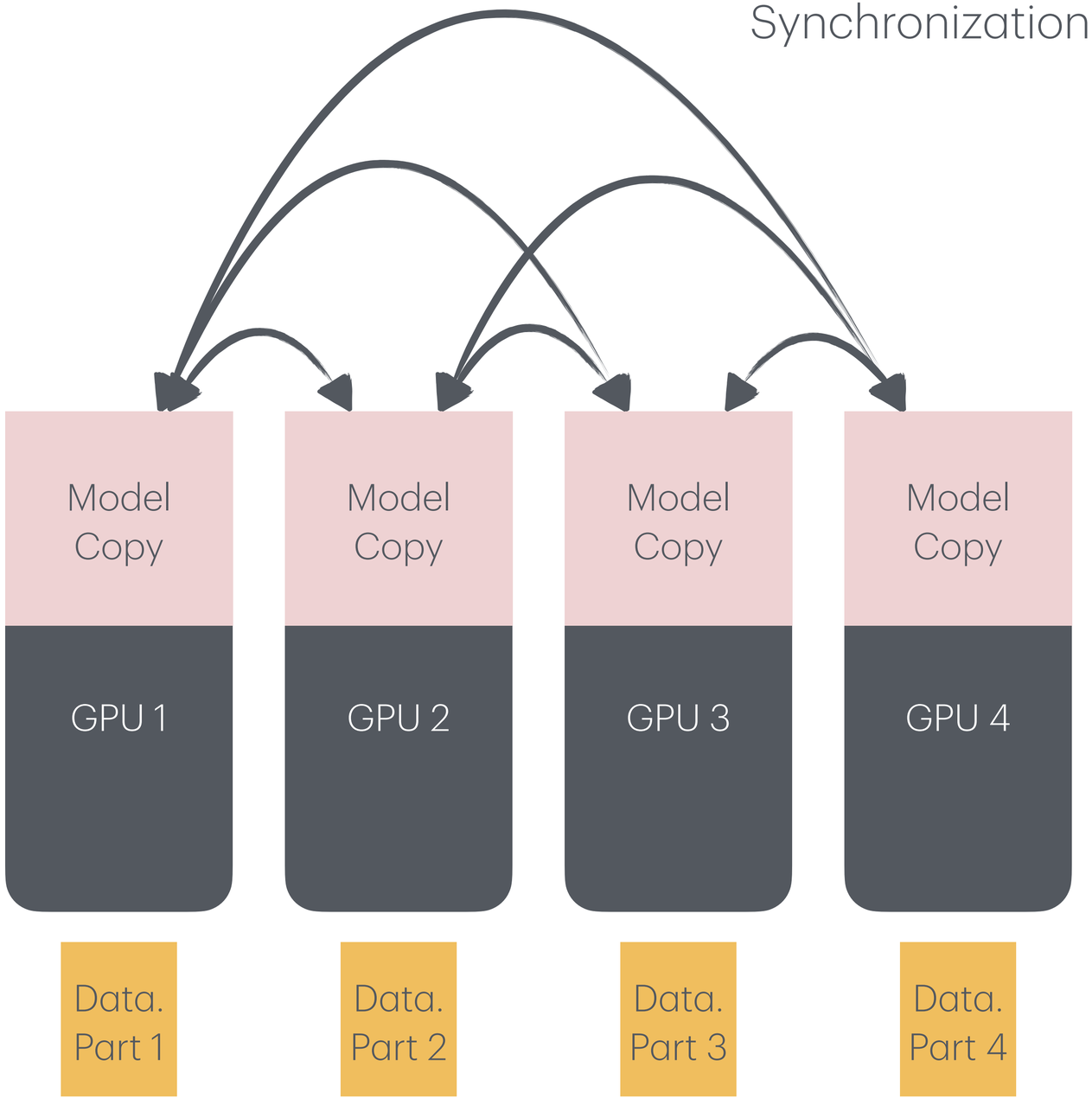


[1] Jeff Dean, et al. Large scale distributed deep networks, 2012.

Solution 1.1: Modern Data parallelism

- One model copy per GPU
- Split data among GPUs
- Only synchronize gradients
  - Each GPU has own optimizer
  - `all-reduce` to accumulate gradient

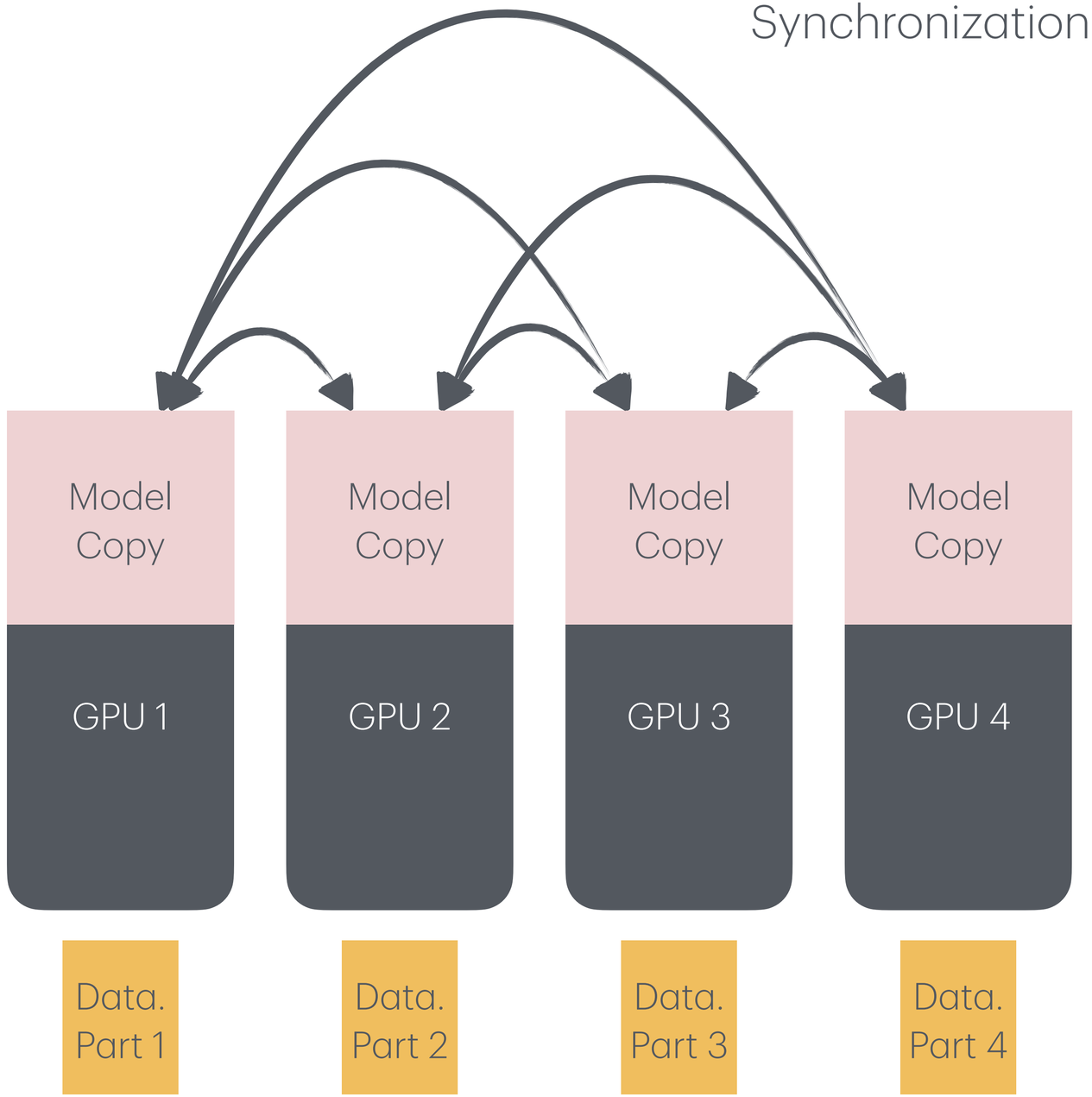




[1] NVIDIA, https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/usage/collectives.html#allreduce

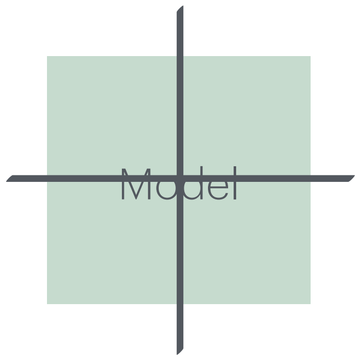
Solution 1.1: Data parallelism

- Advantages
  - Close to $n \times$ speedup for $n \leq 8$ GPUs
  - Simple to implement
  - Data transfer $1 \times$ #weights
- Issues
  - Model still needs to fit on GPU

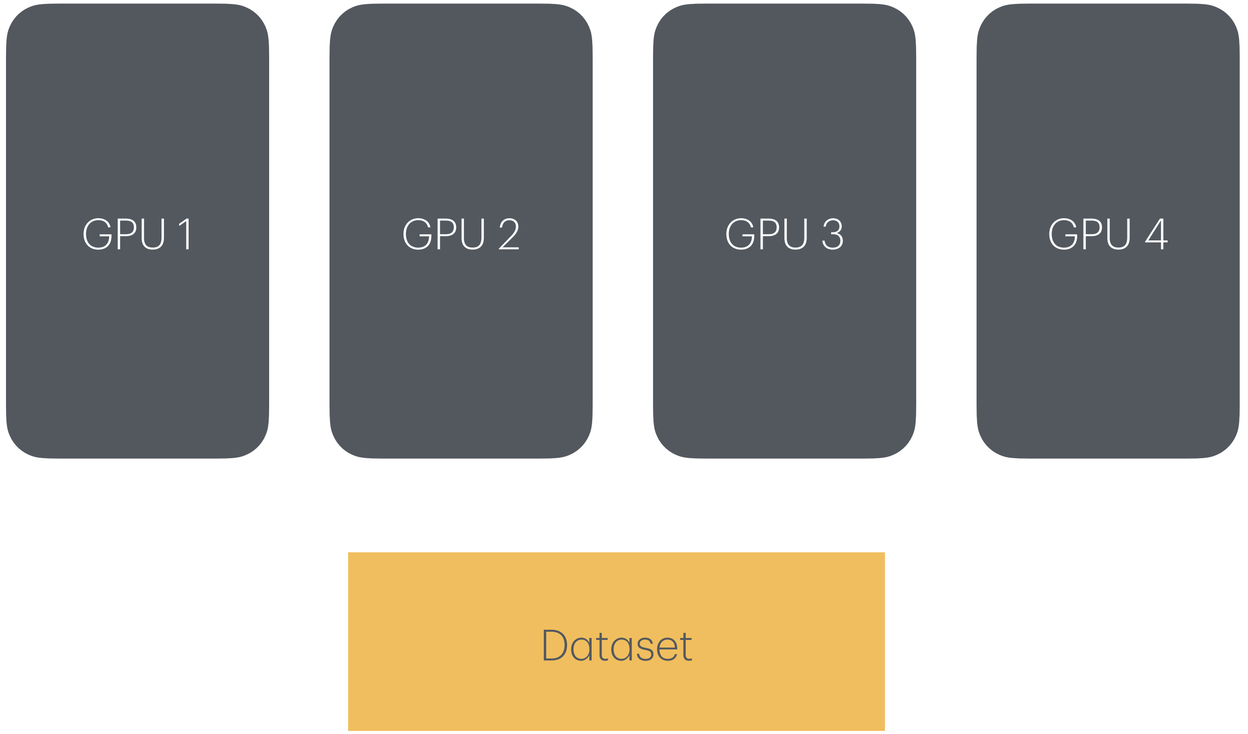


[1] NVIDIA, https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/usage/collectives.html#allreduce

Solution 2: Model parallelism



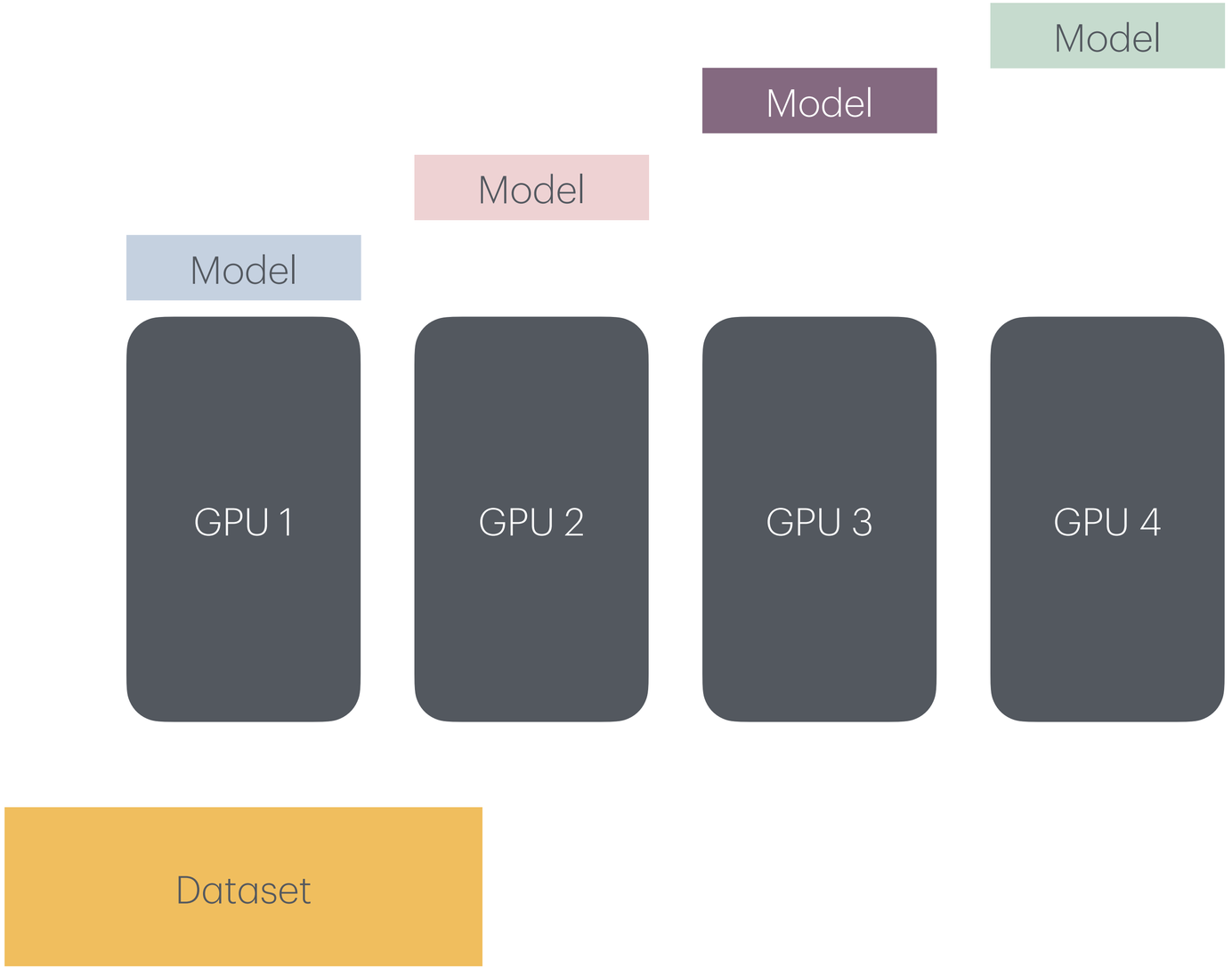
- Split model across GPUs
- Option 1: Split along layers
  - Pipeline parallelism
- Option 2: Split each layer
  - Tensor parallelism
  - Very complex, no longer used



[1] Alex Krizhevsky et al. ImageNet classification with deep convolutional neural networks, 2012

Solution 2: Pipeline parallelism

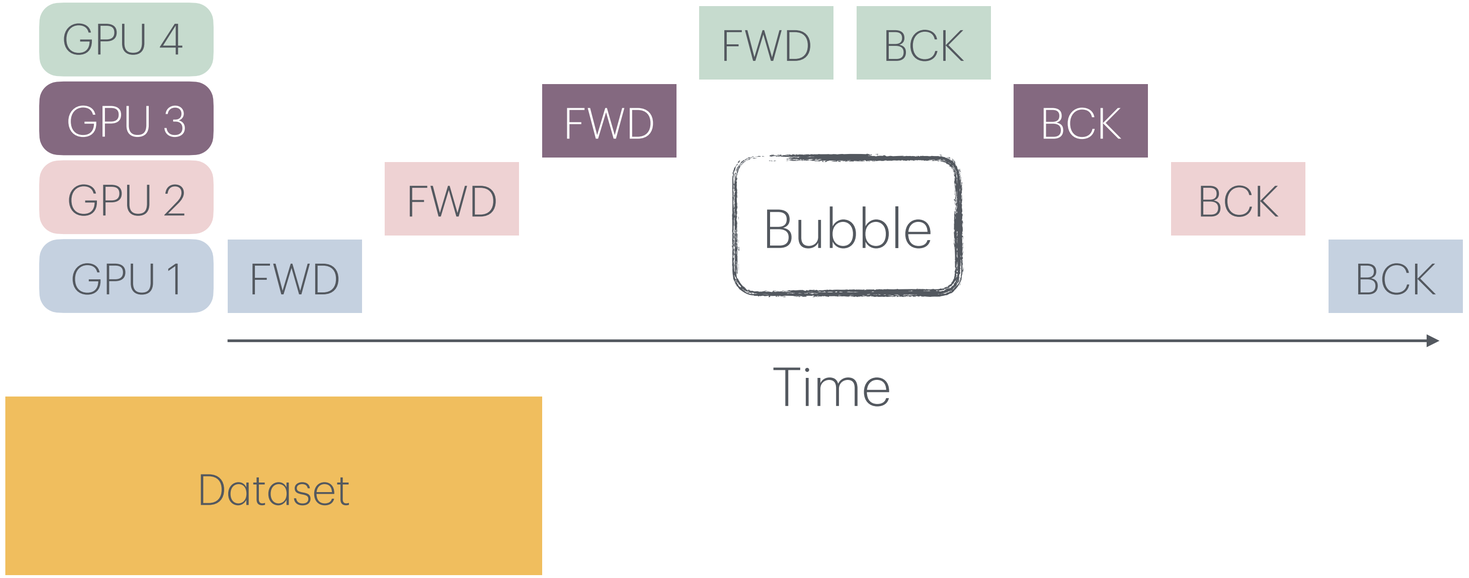
- Each GPU holds a few consecutive layers of a network
- GPU1 hold dataset
- GPUs pass activations on in forward
- GPUs pass gradients on in backward
- How do we split the model?
  - By hand



[1] Huang Yanping, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism, 2019

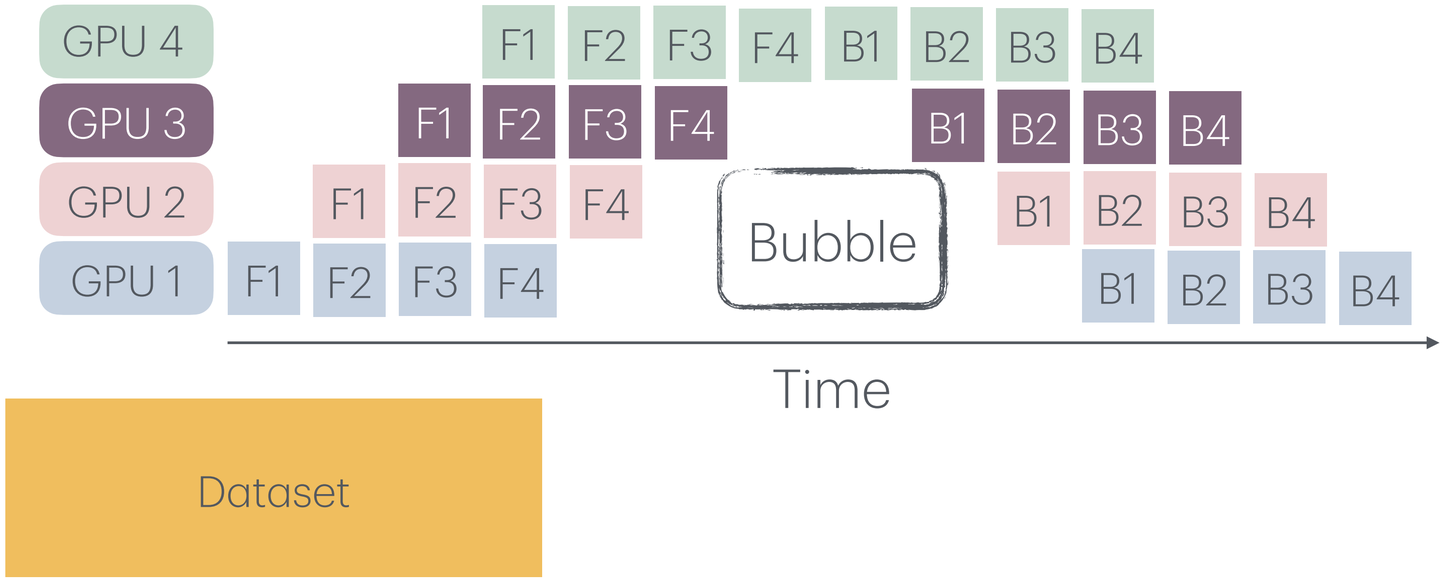
Solution 2: Pipeline parallelism

- Issue: Bubbles
  - Forward / backward one GPU at the time
  - GPU1 idle while others compute result



Solution 2: Pipeline parallelism

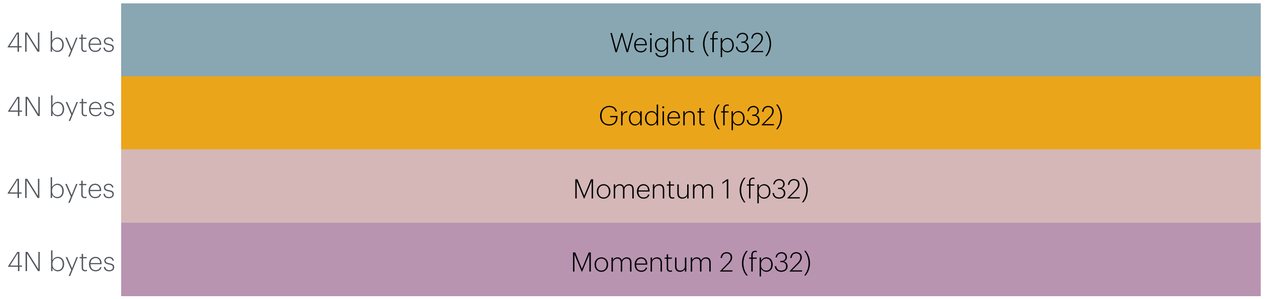
- Solution: Bubbles
  - Micro-batches
  - Split batch into parts
  - Pass activations/gradient of microbatches on
  - Bubble still exists, but is smaller



Training large models

Memory requirements

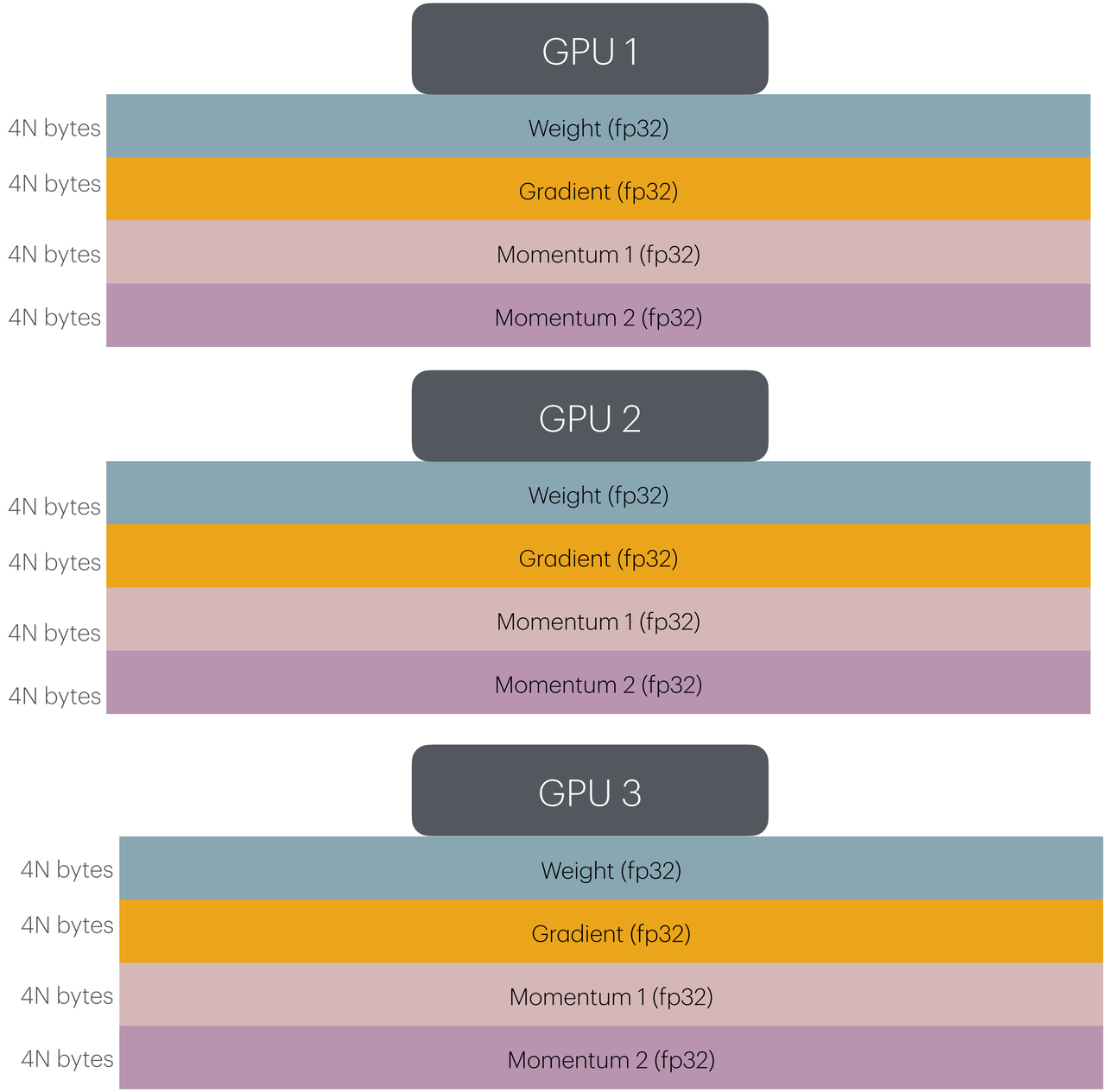
- Data parallelism
  - Model parameters: N
  - Weights: N floats
  - Gradients: N floats
  - Momentum: N floats
  - 2nd momentum (ADAM): N floats
- 16N bytes without counting activations



Training large models

Memory requirements

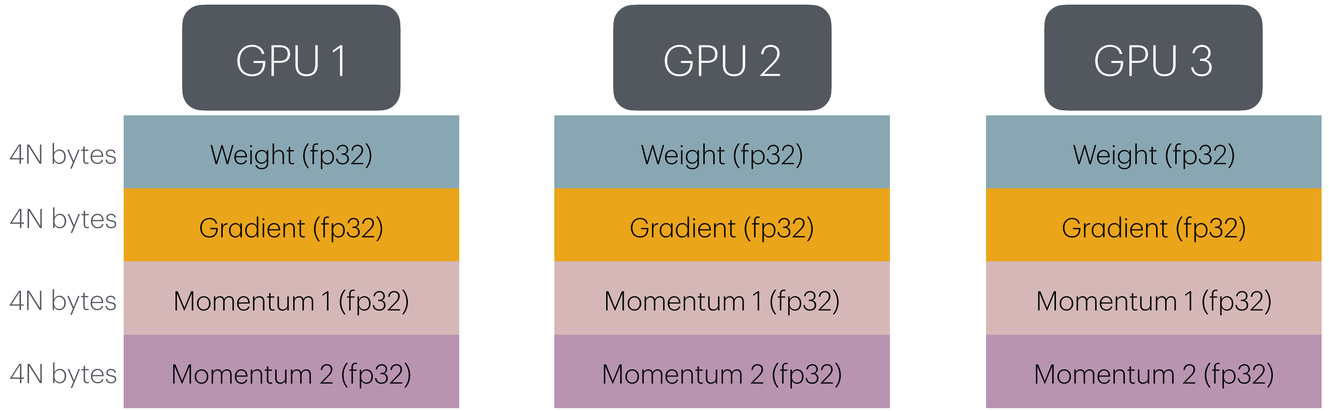
- Data parallelism
  - Model parameters: N
  - Weights: N floats
  - Gradients: N floats
  - Momentum: N floats
  - 2nd momentum (ADAM): N floats
- 16N bytes without counting activations



Training large models

Memory requirements

- Model parallelism
  - Model parameters: N
  - Weights: N floats / #GPU
  - Gradients: N floats / #GPU
  - Momentum: N floats / #GPU
  - 2nd momentum (ADAM): N floats / #GPU
- 16N bytes / #GPU without counting activations



Distributed Training

- Distributed Training
  - Data parallel
    - High memory, high throughput
  - Model parallel
    - Low memory, bubbles -> lower throughput
- Next: Advanced distributed training
  - Lowest memory, high throughput

References

- [1] Llama Team, The Llama 3 Herd of Models, 2024. (link)
- [2] Jeff Dean, et al., Large scale distributed deep networks, 2012. (link)
- [3] Alex Krizhevsky et al,. ImageNet classification with deep convolutional neural networks, 2012. (link)
- [4] Huang Yanping, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism, 2019 (link)