Distributed Training

Arvind Krishnamurthy

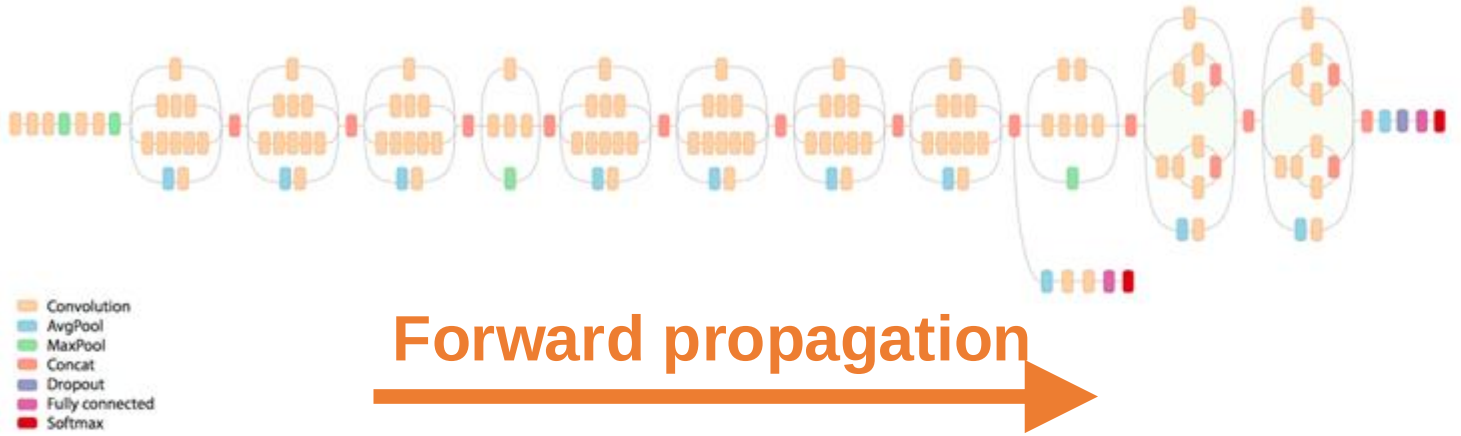
Material adapted from slides by Tianqi Chen & Zhihao Jia (CMU), Minjia Zhang (UIUC)

Stochastic Gradient Descent (SGD)

Train ML models through many iterations of 3 stages

1. **Forward propagation**: apply model to a batch of input samples and run calculation through operators to produce a prediction
2. **Backward propagation**: run the model in reverse to produce a gradient for each trainable weight
3. **Weight update**: use the accumulated gradients to update model weights

Model inputs



Model prediction

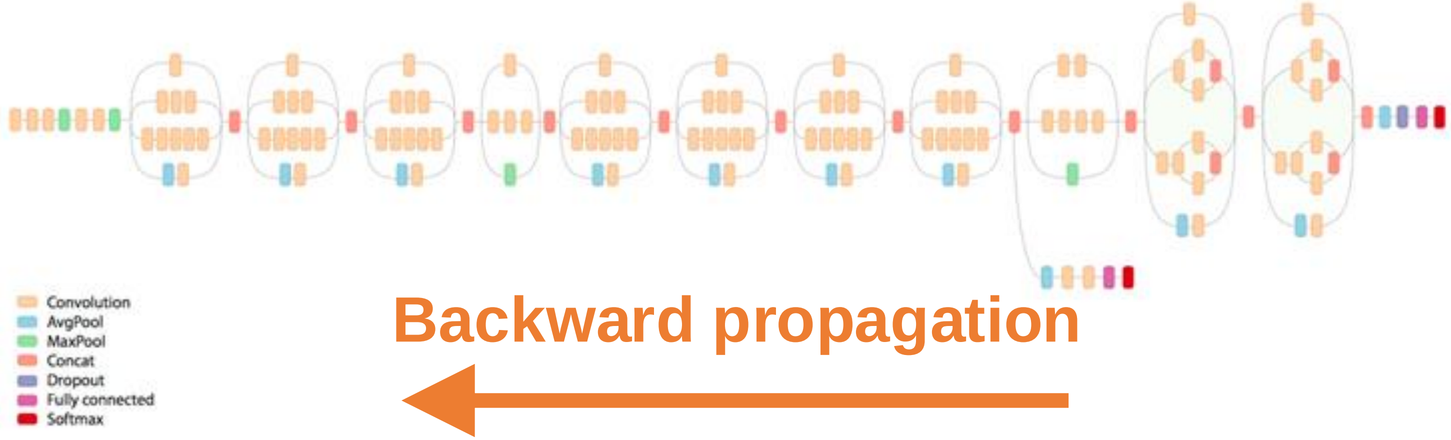
Forward propagation

Stochastic Gradient Descent (SGD)

Train ML models through many iterations of 3 stages

1. **Forward propagation**: apply model to a batch of input samples and run calculation through operators to produce a prediction
2. **Backward propagation**: run the model in reverse to produce a gradient for each trainable weight
3. **Weight update**: use the accumulated gradients to update model weights

Model inputs



Model prediction

Backward propagation

Stochastic Gradient Descent (SGD)

Train ML models through many iterations of 3 stages

1. **Forward propagation**: apply model to a batch of input samples and run calculation through operators to produce a prediction
2. **Backward propagation**: run the model in reverse to produce a gradient for each trainable weight
3. **Weight update**: use the accumulated gradients to update model weights

$$w_i := w_i - \gamma \nabla L(w_i) = w_i - \frac{\gamma}{n} \sum_{j=1}^{n} \nabla L_j(w_i)$$

Key Concepts

ML training state can be classified into the following:

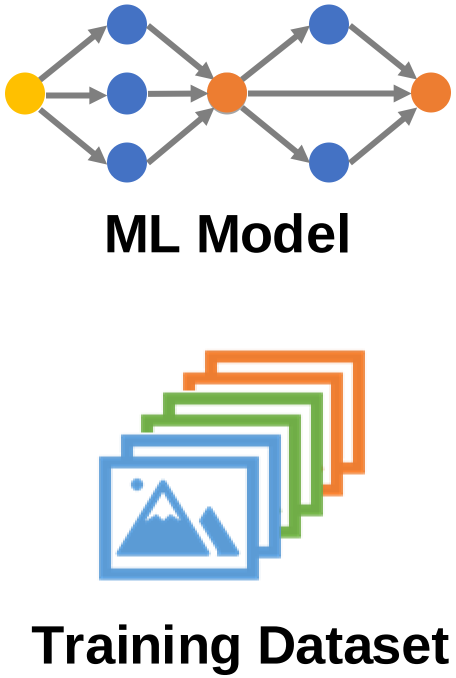
- **Model parameters:** eventually becomes the released model
- **Gradients:** how the loss function varies with small changes to parameters
- **Activations:** intermediate results from the forward phase required for back-propagating gradients
- **Optimizer state:** maintains information about how parameters change over time

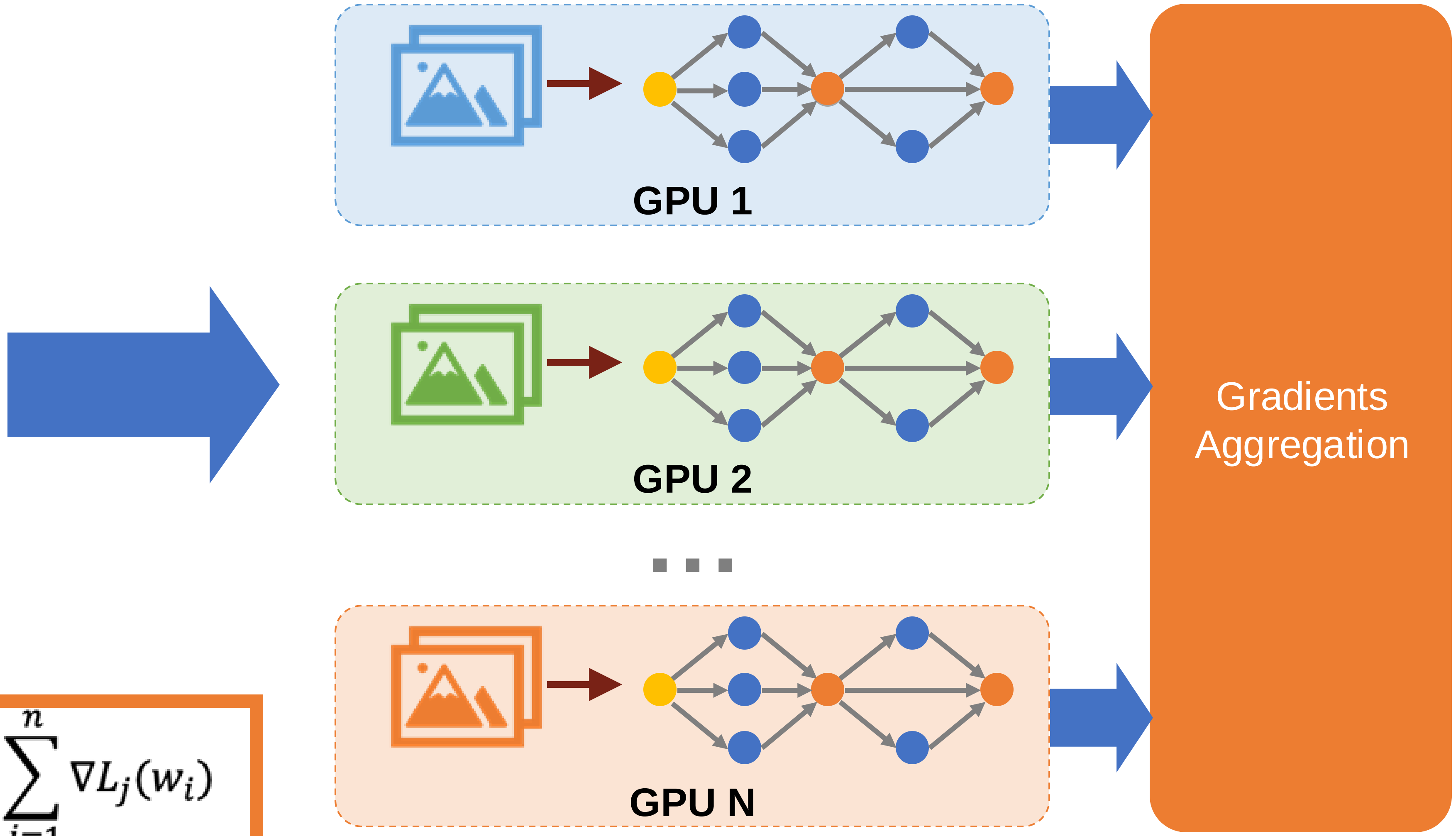
How can we parallelize ML training?

Goals:

- Scale with training data size
- Scale with model size

Data Parallelism





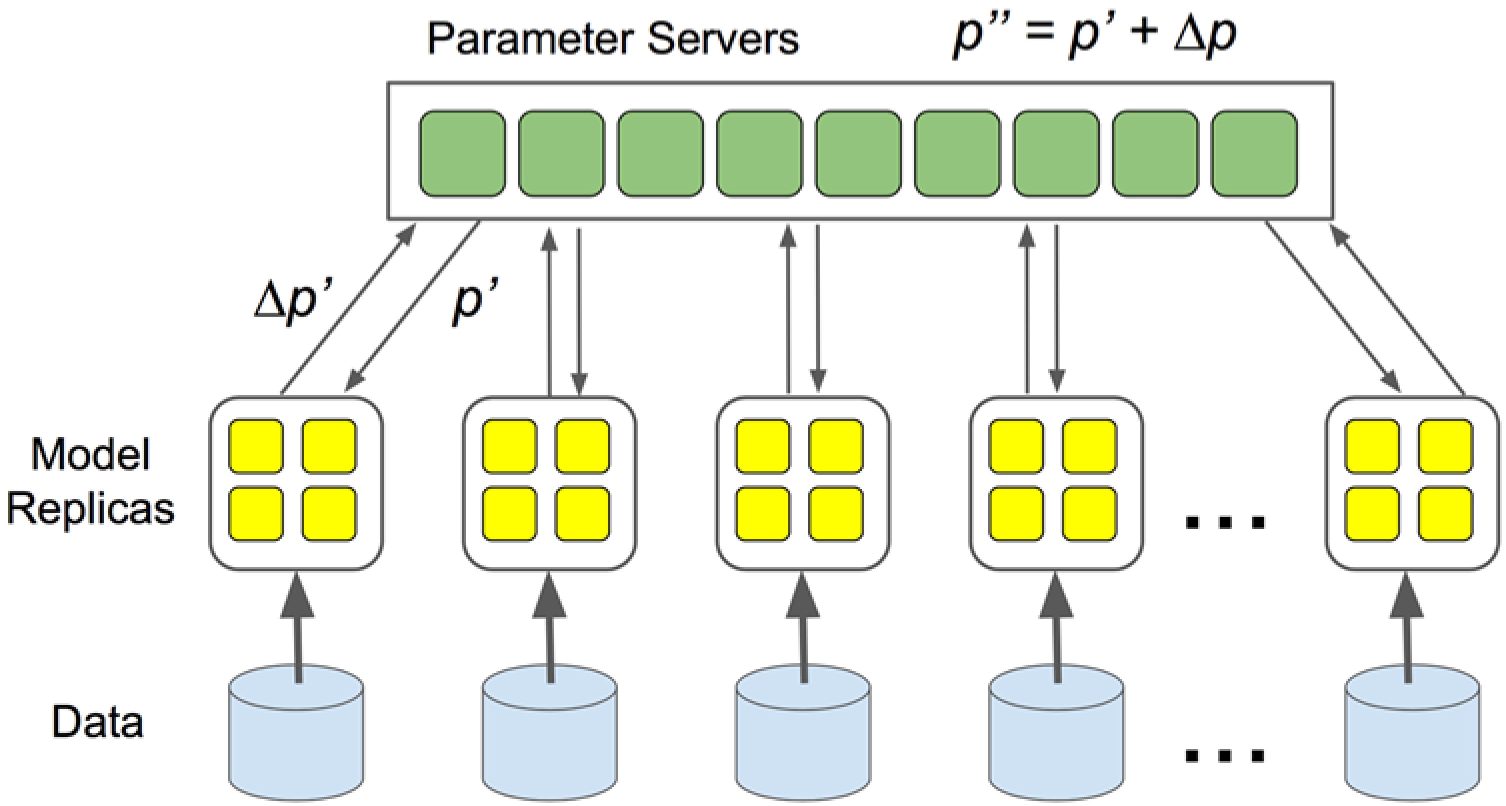
$$w_i := w_i - \gamma \nabla L(w_i) = w_i - \frac{\gamma}{n} \boxed{\sum_{j=1}^{n} \nabla L_j(w_i)}$$

1. Partition training data into batches

2. Compute the gradients of each batch on a GPU

3. Aggregate gradients across GPUs

Data Parallelism: Parameter Server

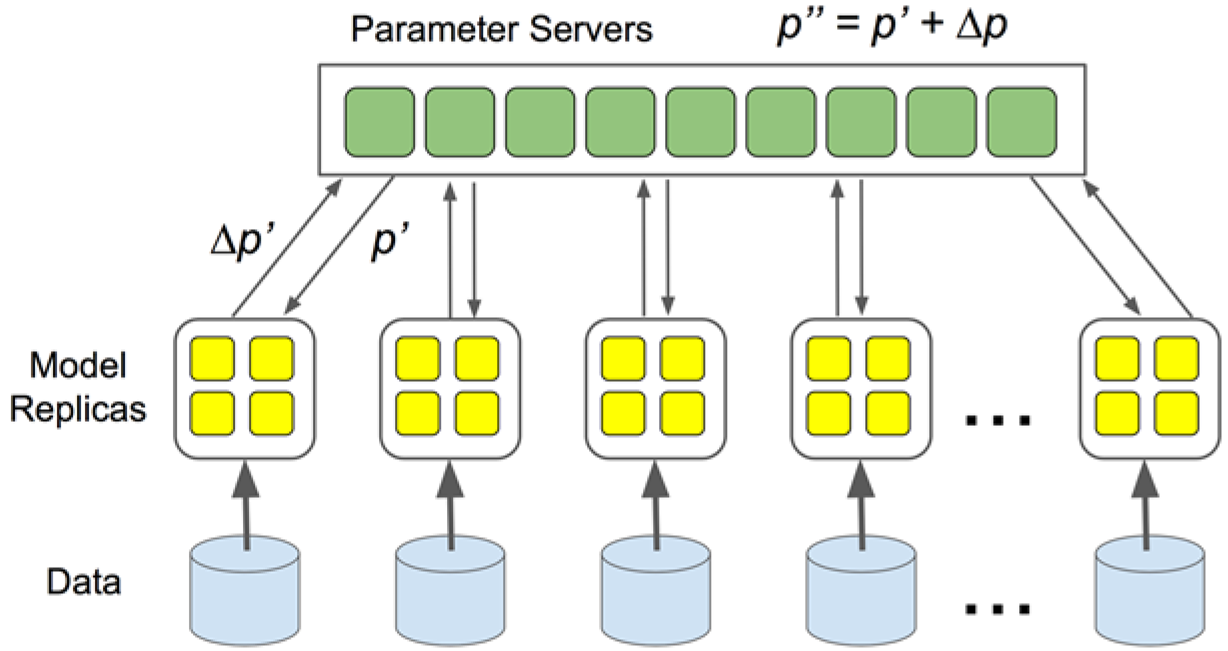


Workers push gradients to parameter servers and pull updated parameters back

Inefficiency of Parameter Server

- **Centralized communication**: all workers communicate with parameter servers for weights update; cannot scale to large numbers of workers

- How can we decentralize communication in DNN training?

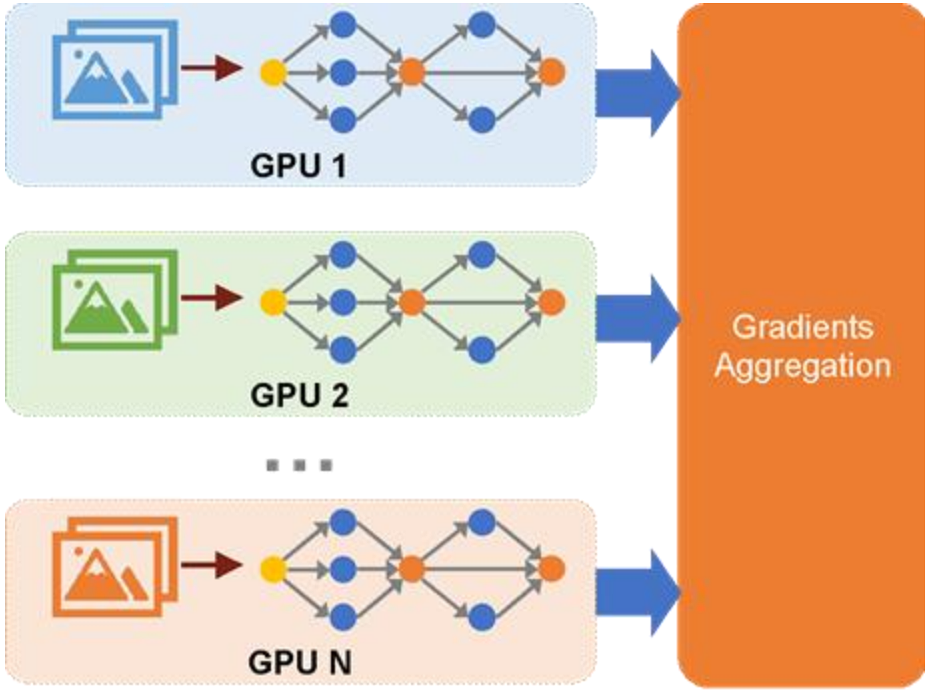


Data Parallelism: AllReduce

- **AllReduce**: perform element-wise reduction across multiple devices

Data Parallelism

Each worker keeps a replica of the entire model and communicates with other workers to synchronize weights updates



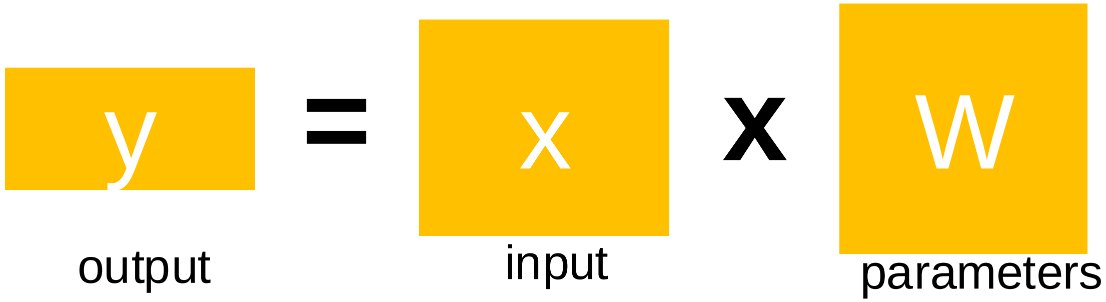
Gradients aggregation methods:

- Parameter Server
- Ring AllReduce
- Tree AllReduce
- Butterfly AllReduce
- Etc.

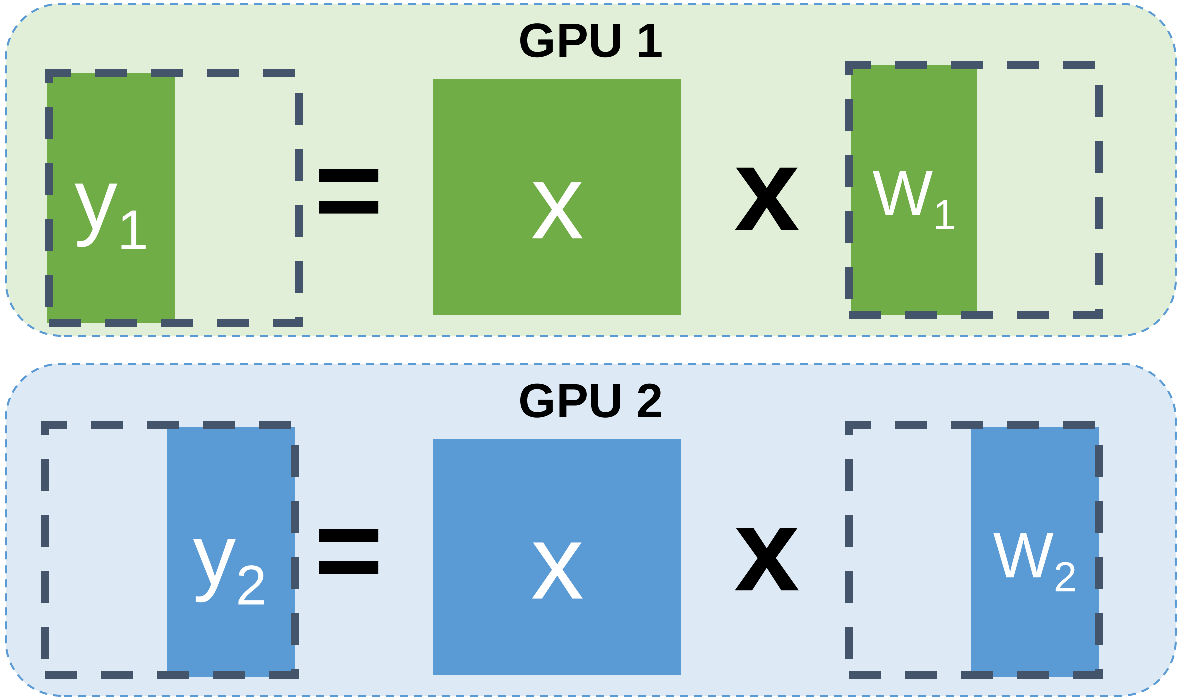
How to parallelize DNN Training?

- Data parallelism
- Model parallelism
  - Tensor model parallelism
  - Pipeline model parallelism

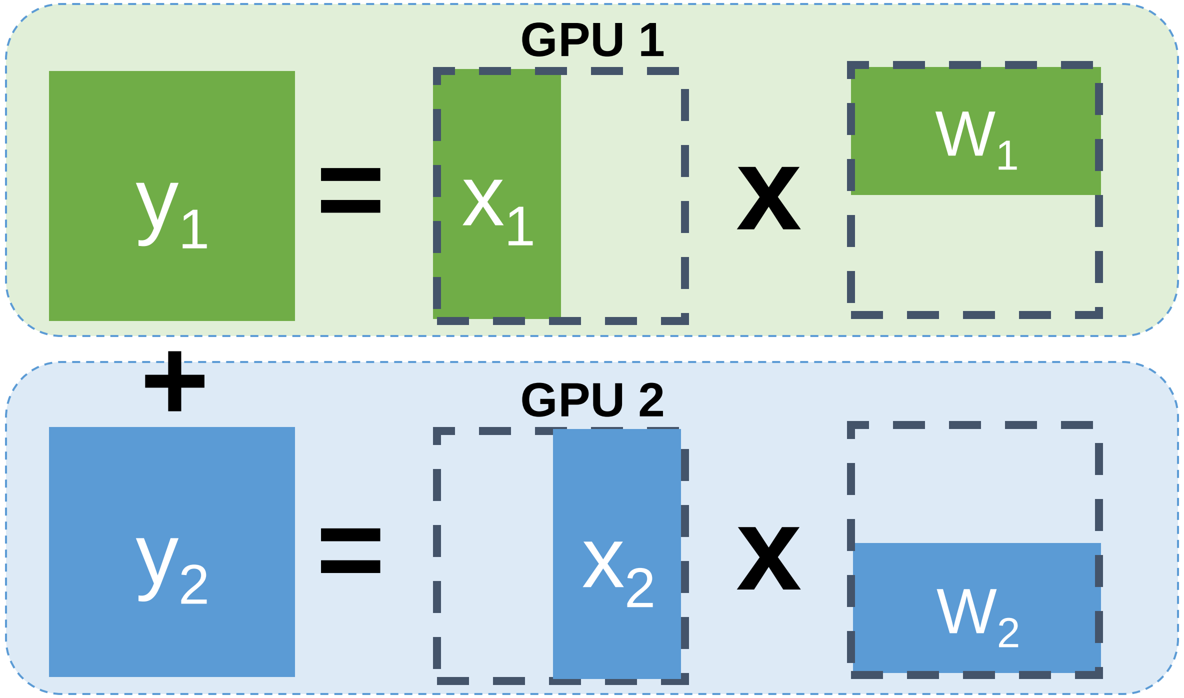
Tensor Model Parallelism



- Partition parameters/gradients ***within*** a layer



Tensor Model Parallelism (partition output)



Tensor Model Parallelism (reduce output)

$$y = y1 + y2$$

Pipeline Parallelism

- Divide a mini-batch into multiple micro-batches
- Pipeline the forward/backward computations across micro-batches
- Generally combined with model parallelism

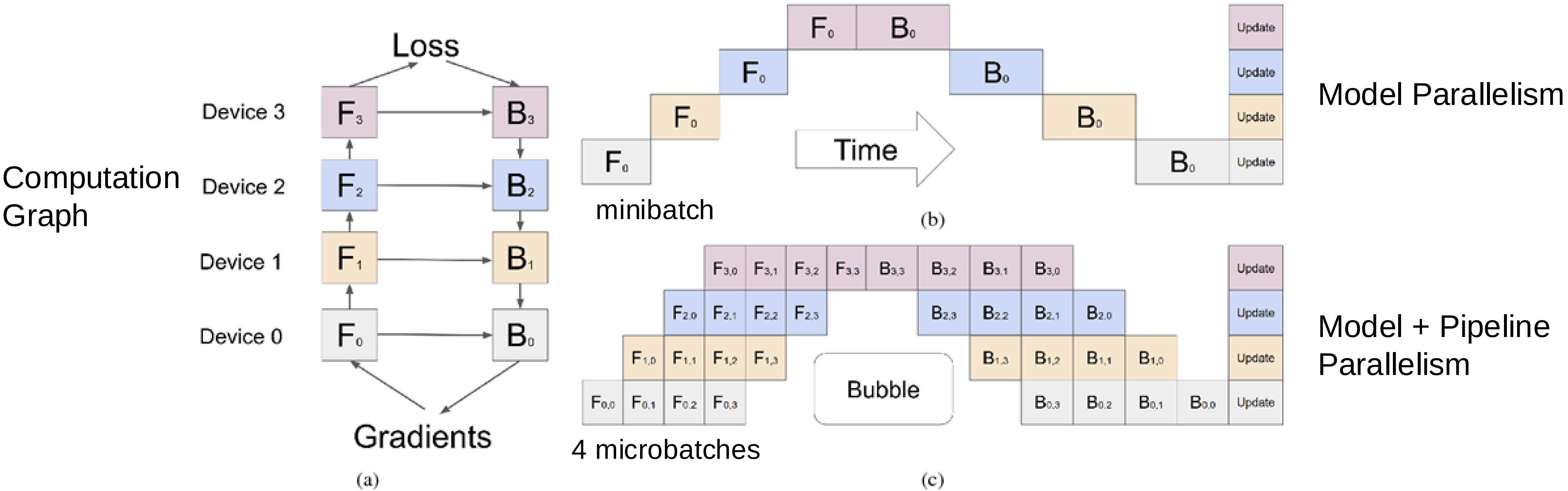
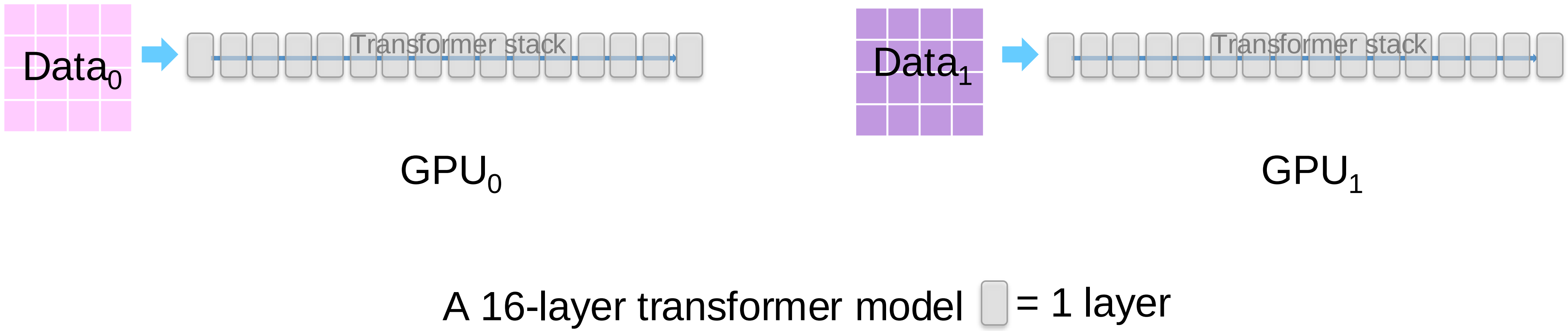
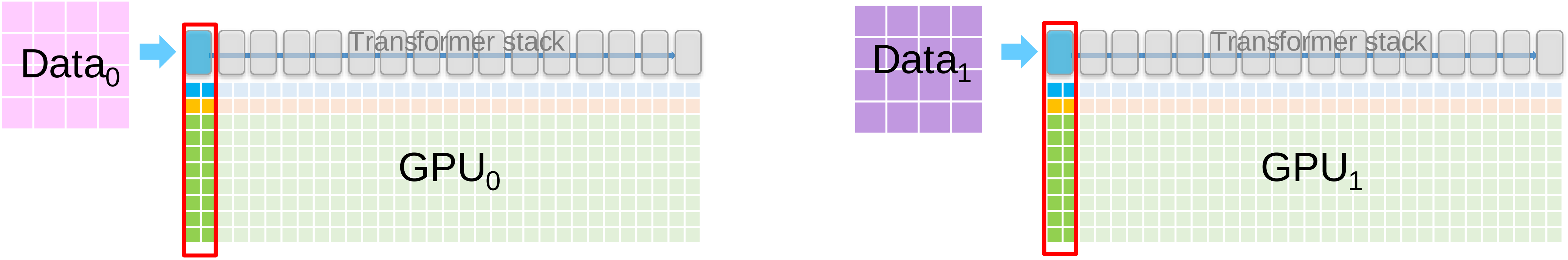


Figure from GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding

Understanding Memory Consumption



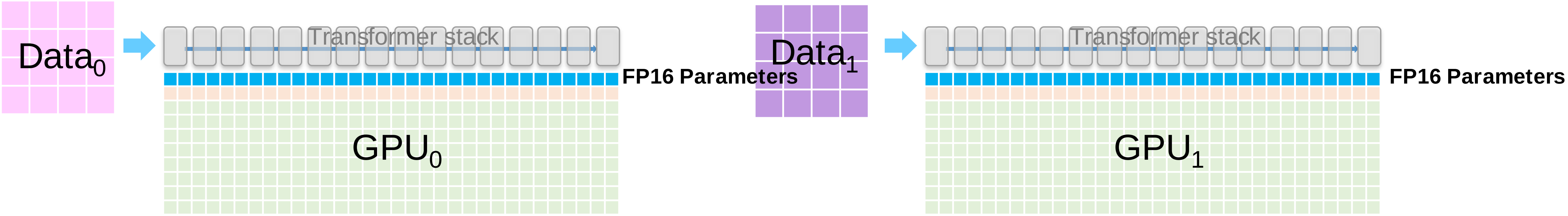
Understanding Memory Consumption



Each cell represents GPU memory used by its corresponding transformer layer

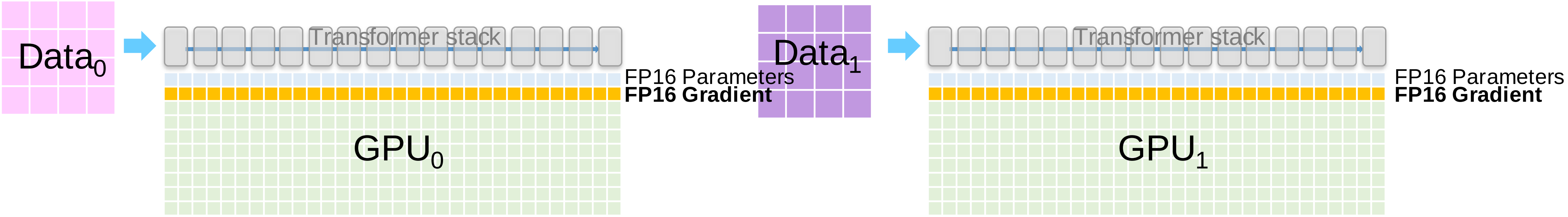
Adapted from Minjia Zhang, DeepSpeed Presentation

Understanding Memory Consumption



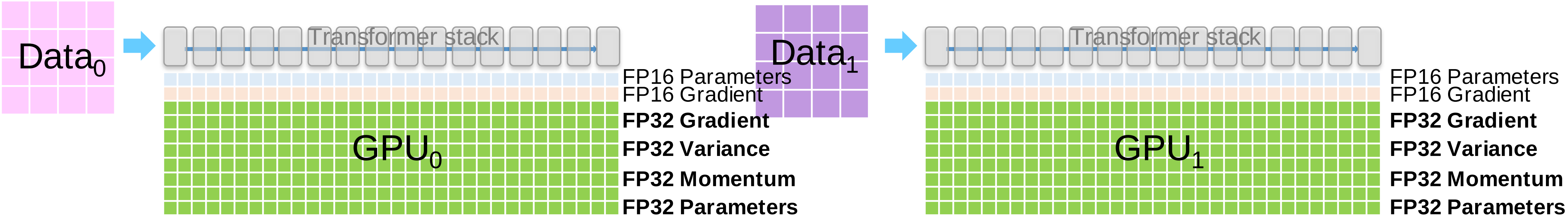
- FP16 parameter

Understanding Memory Consumption



- FP16 parameter
- FP16 Gradients

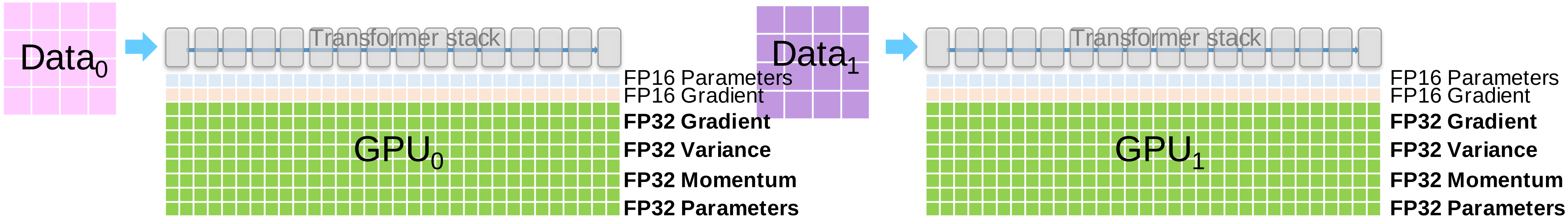
Understanding Memory Consumption



- FP16 parameter
- FP16 Gradients
- FP32 Optimizer States
  - Gradients, Variance, Momentum, Parameters

Adapted from Minjia Zhang, DeepSpeed Presentation

Understanding Memory Consumption



- FP16 parameter : **2M bytes**
- FP16 Gradients : **2M bytes**
- FP32 Optimizer States : **16M bytes**
  - Gradients, Variance, Momentum, Parameters

M = number of parameters in the model

Example 1B parameter model -> 20GB/GPU

Memory consumption doesn’t include:

- Input batch + activations

Adapted from Minjia Zhang, DeepSpeed Presentation

ZeRO-DP: ZeRO powered Data Parallelism

- ZeRO removes the redundancy across data parallel process
- Stage 1: partitioning optimizer states
- Stage 2: partitioning gradients
- Stage 3: partitioning parameters