Memory Offloading

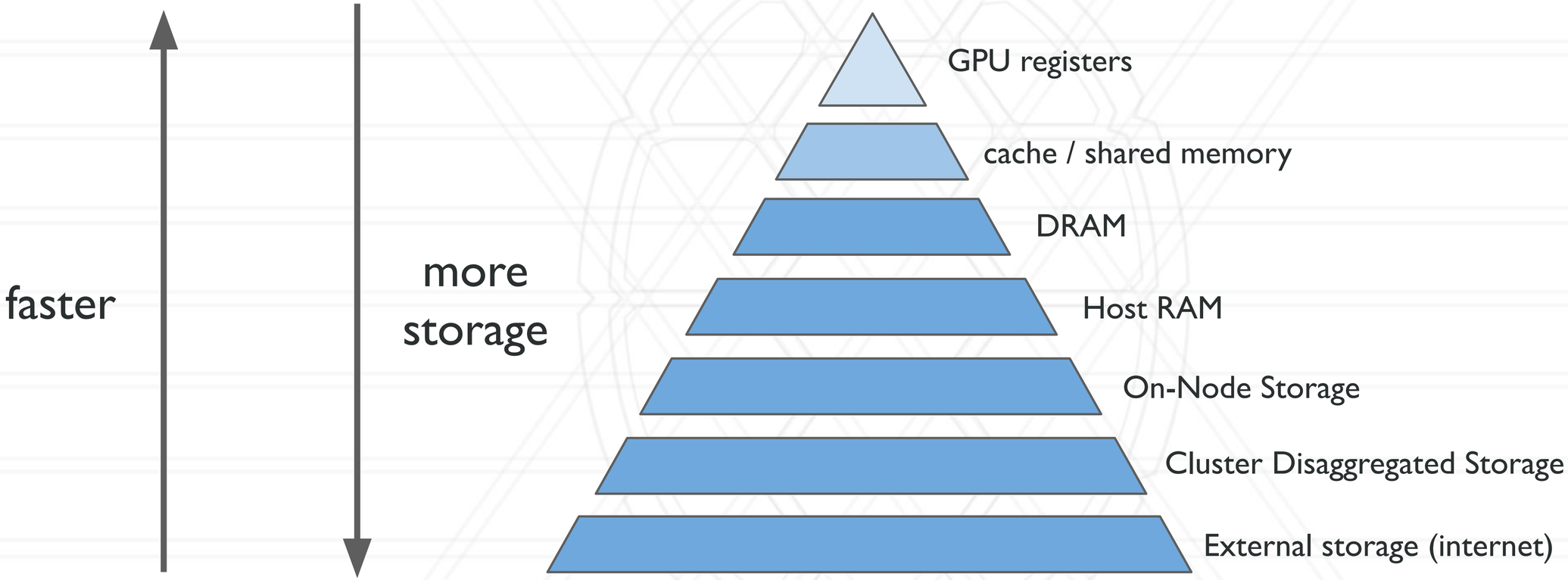
Abhinav Bhatele, Daniel Nichols

UNIVERSITY OF MARYLAND

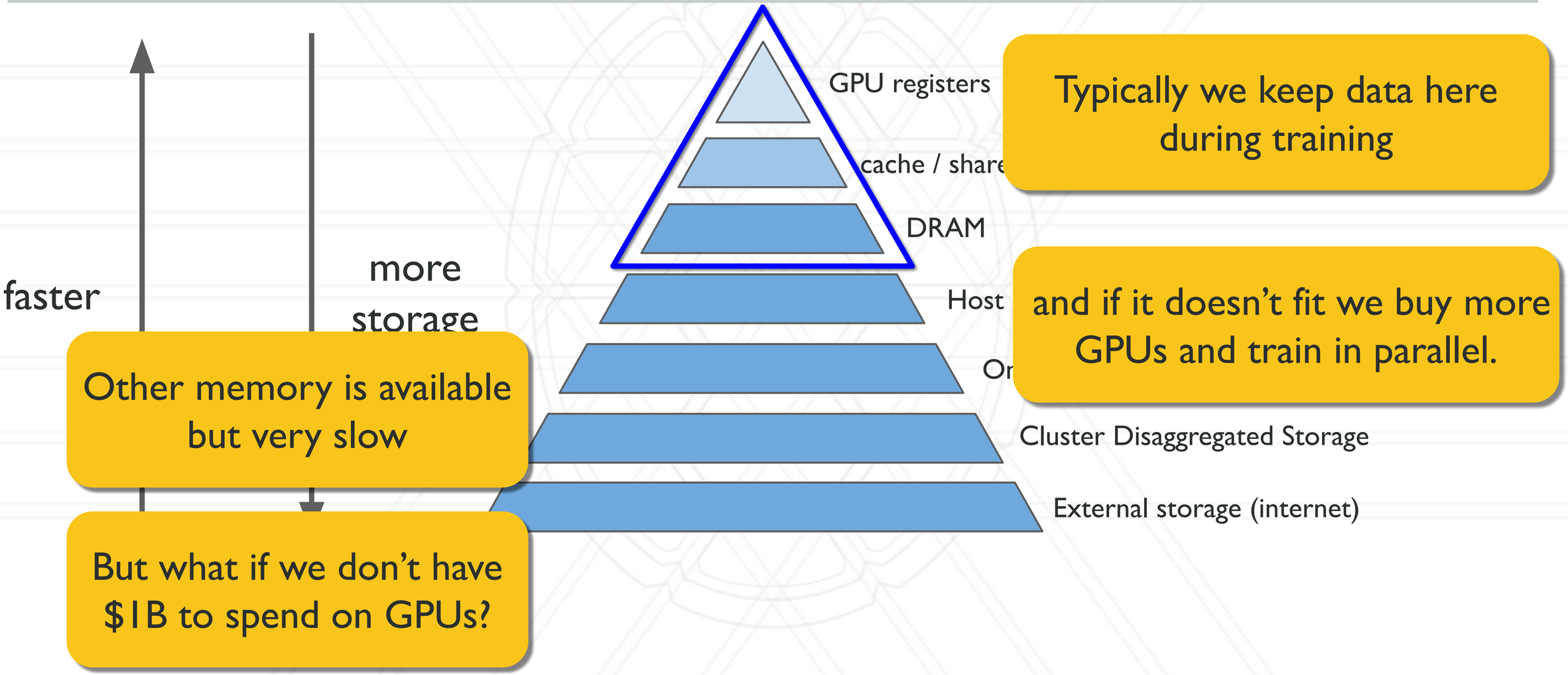
Announcements

- Interim report for the project is due on April 17
- Midterm is on April 10
- Assignment 2 grades are out

Memory Hierarchy

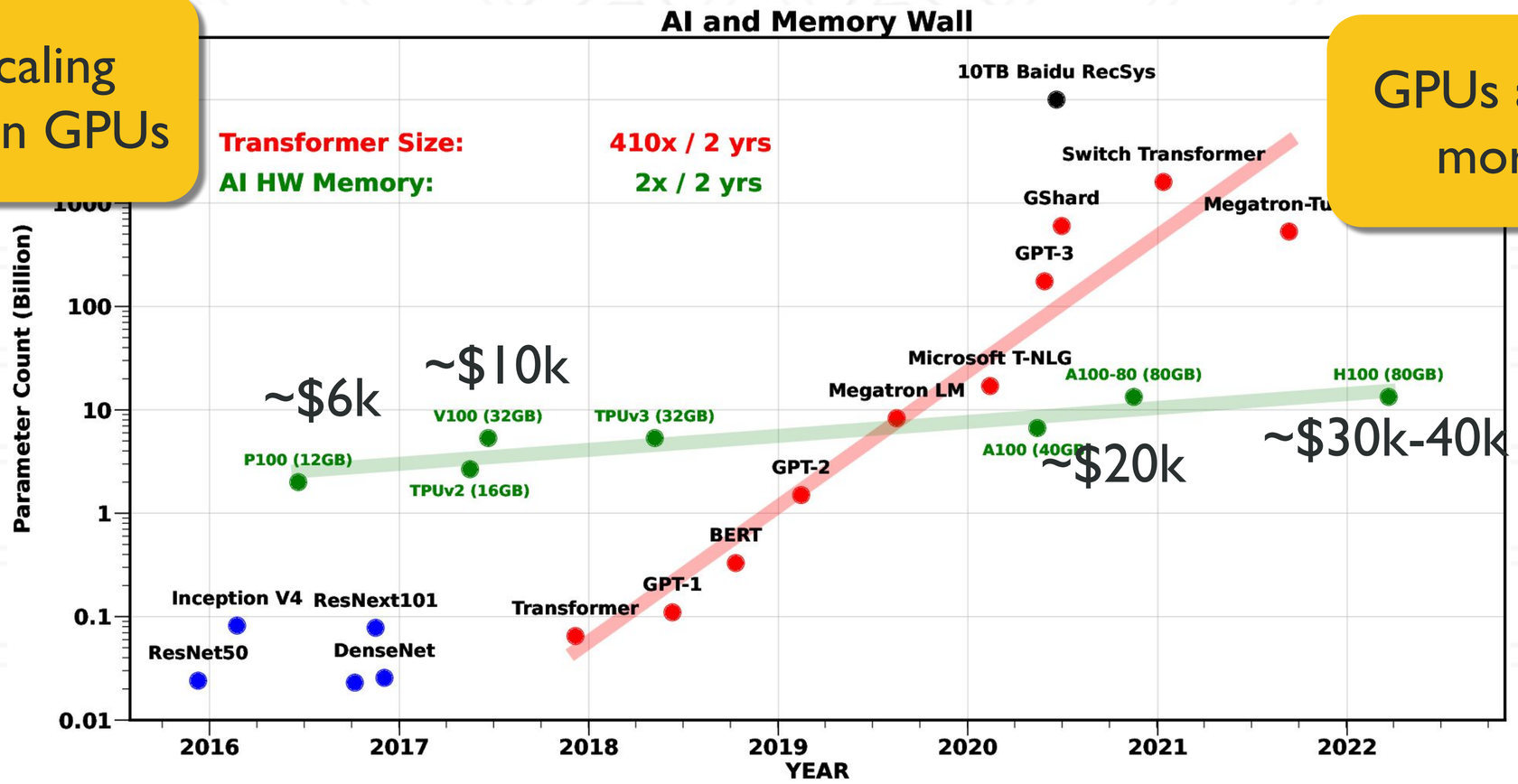


Memory Hierarchy



Problems with Memory Scaling

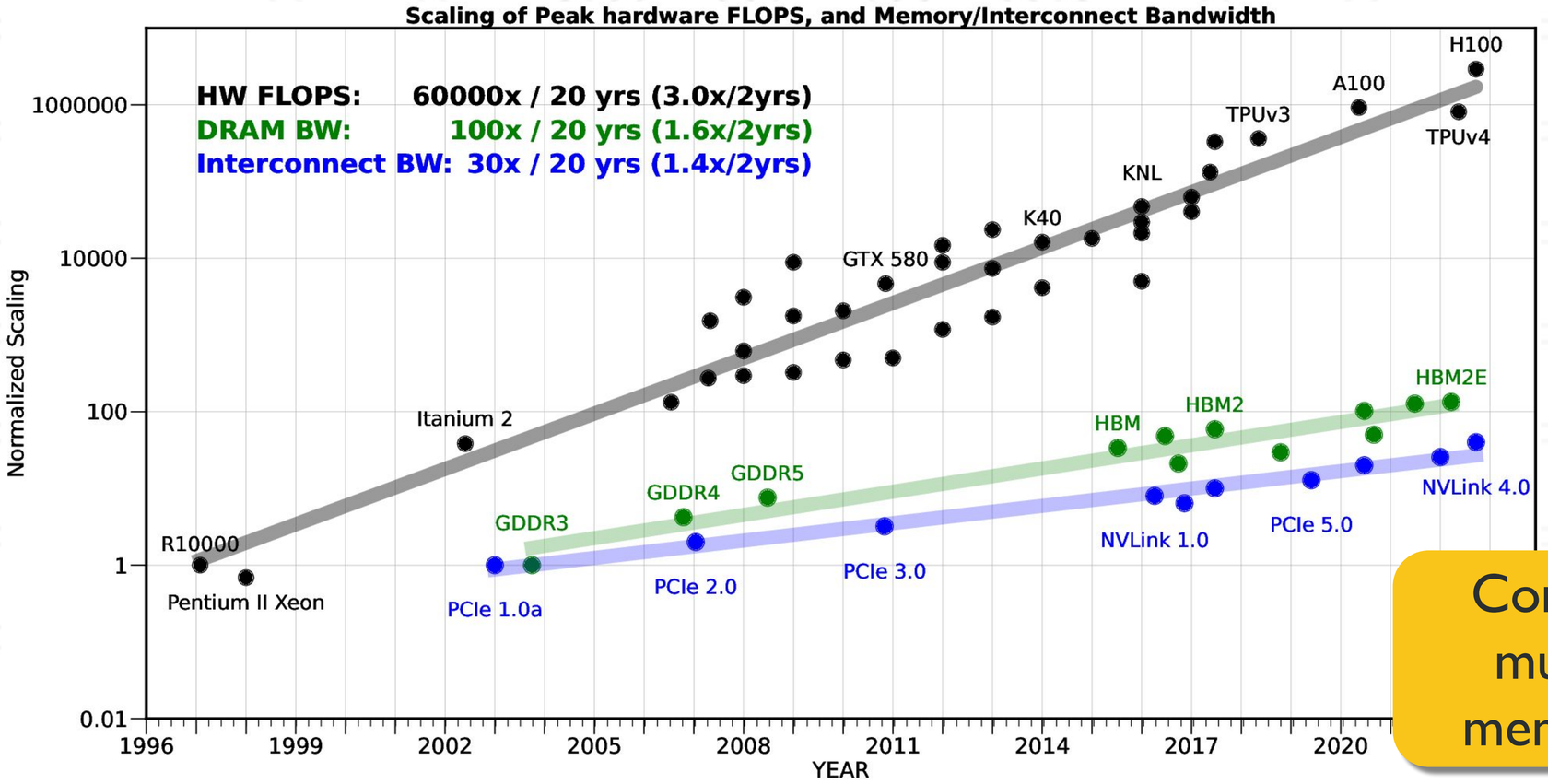
Models are scaling much faster than GPUs



GPUs are also getting more expensive

A. Gholami, et. al., "AI and Memory Wall"

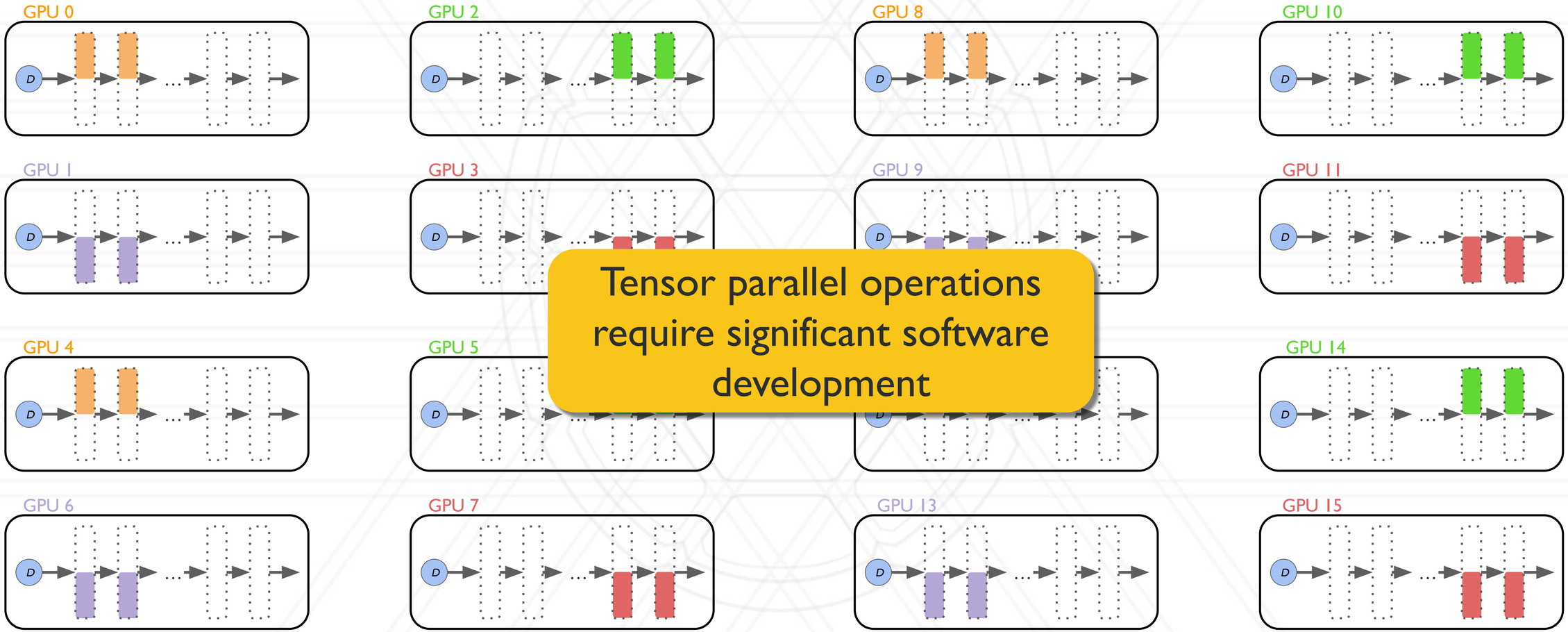
Problems with Memory Scaling



Compute is scaling much faster than memory bandwidth

A. Gholami, et. al., "AI and Memory Wall"

Other Problems with Using More GPUs

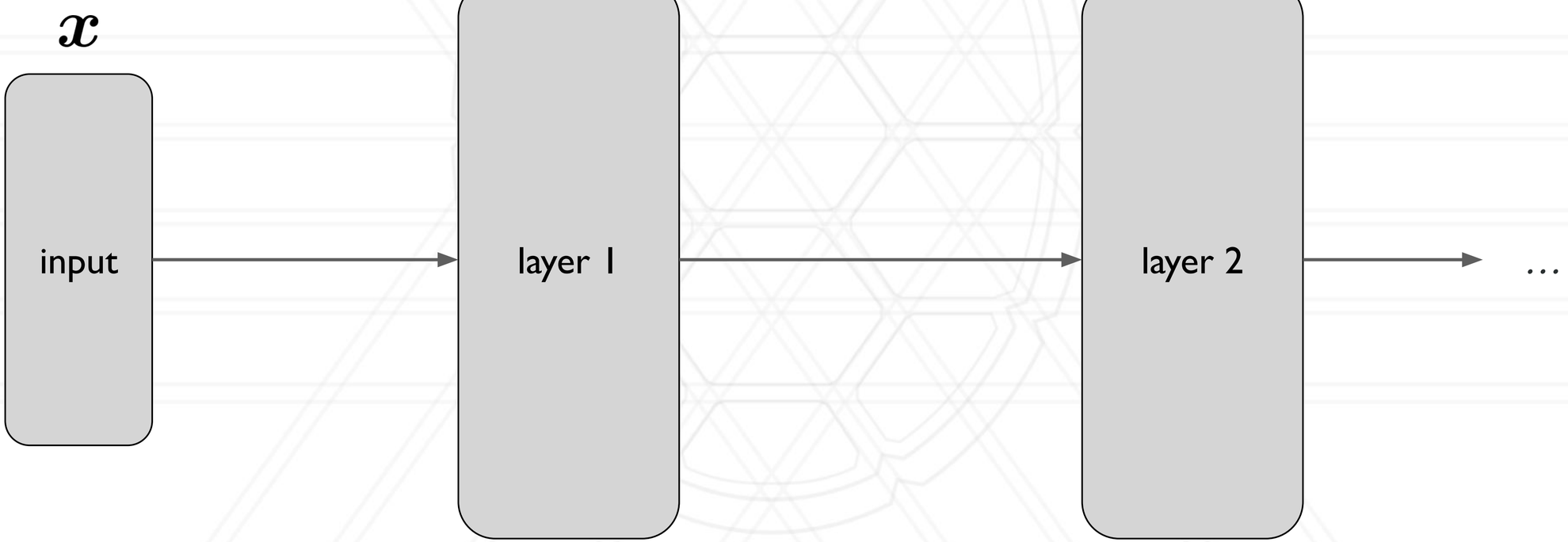


Memory Offloading

- Can we migrate memory to the CPU or disk to save on GPU memory?
- Will be slow, but provide previously unattainable functionality
- Sometimes this is ok
  - Fine-tuning on small datasets
  - Batched inference
  - Reads/writes can be completely masked

What's using the memory?

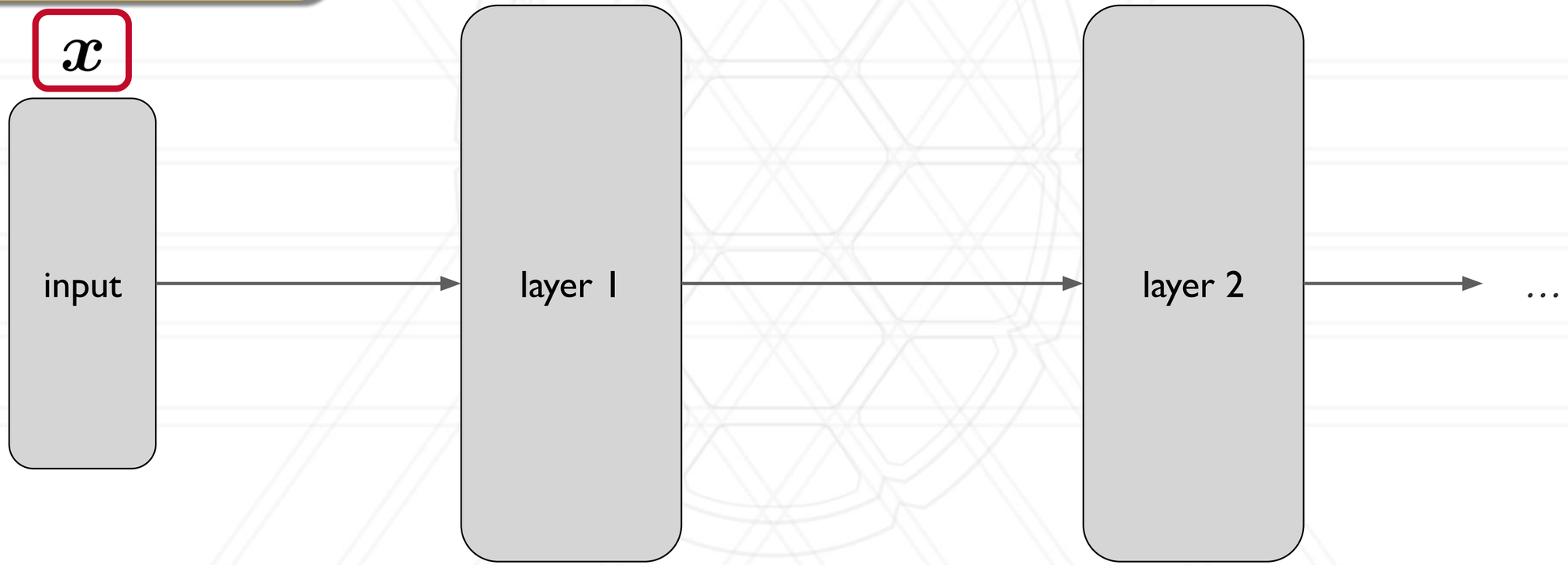
$$h^{(1)} = \sigma\left(\Theta^{(1)T} x\right) \qquad h^{(2)} = \sigma\left(\Theta^{(2)T} h^{(1)}\right)$$



What's using the memory?

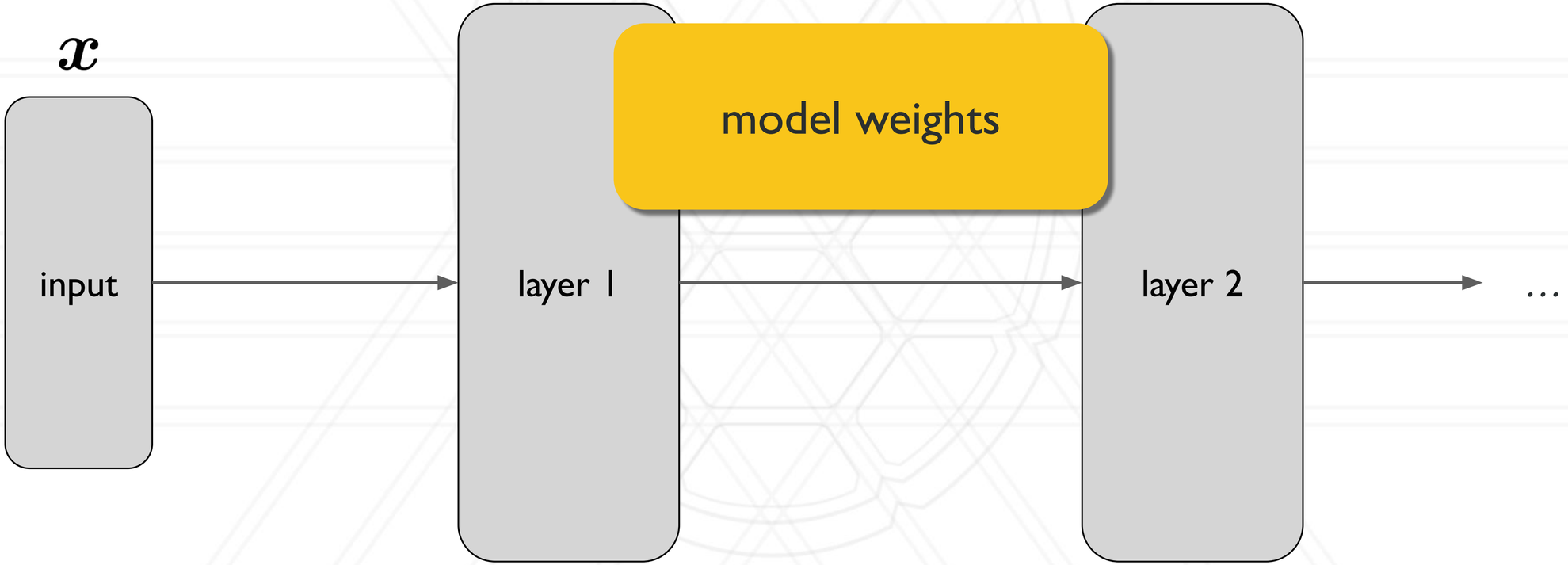
input data – this is generally negligible

$$\boldsymbol{h}^{(1)} = \sigma\left(\boldsymbol{\Theta}^{(1)T}\boldsymbol{x}\right) \qquad \boldsymbol{h}^{(2)} = \sigma\left(\boldsymbol{\Theta}^{(2)T}\boldsymbol{h}^{(1)}\right)$$



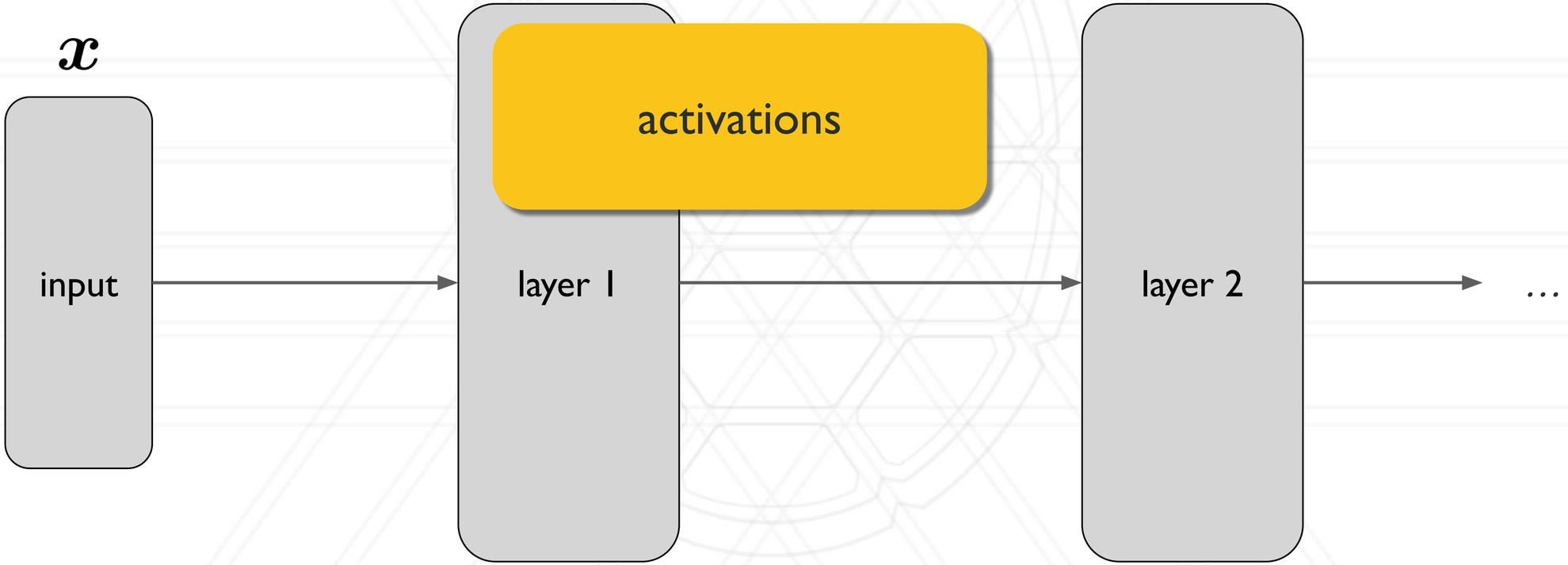
What's using the memory?

$$h^{(1)} = \sigma\left(\Theta^{(1)T} x\right) \qquad h^{(2)} = \sigma\left(\Theta^{(2)T} h^{(1)}\right)$$



What's using the memory?

$$\boxed{\boldsymbol{h}^{(1)}} = \sigma\left(\boldsymbol{\Theta}^{(1)T}\boldsymbol{x}\right) \qquad \boxed{\boldsymbol{h}^{(2)}} = \sigma\left(\boldsymbol{\Theta}^{(2)T}\boldsymbol{h}^{(1)}\right)$$



What's using the memory?

- Activation checkpointing can save us some space
- Memory - performance tradeoff

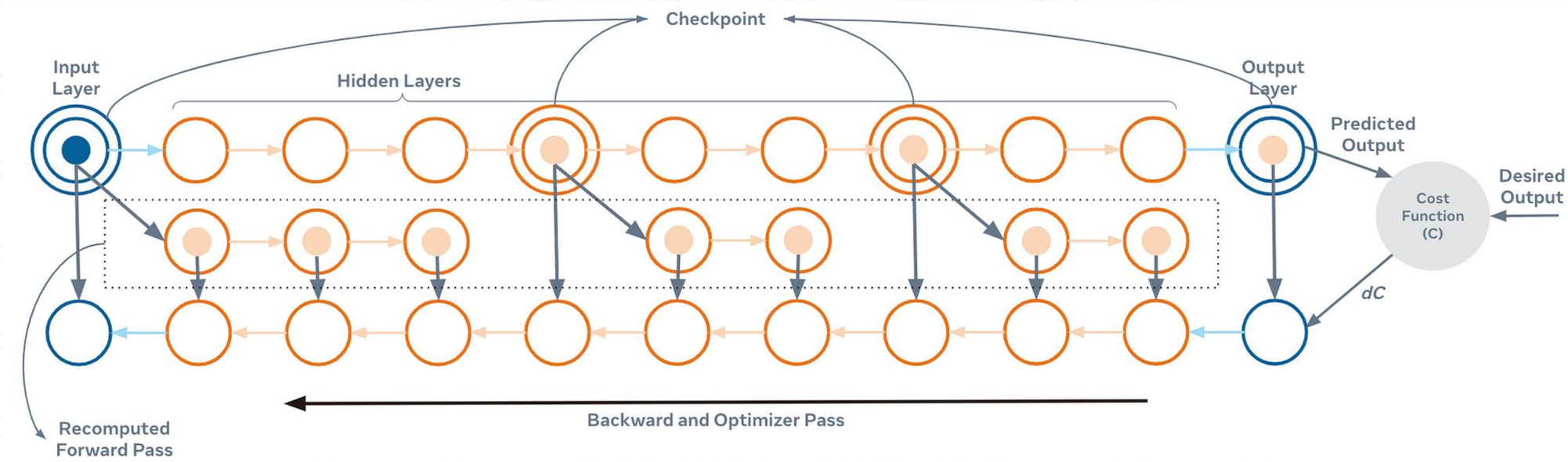
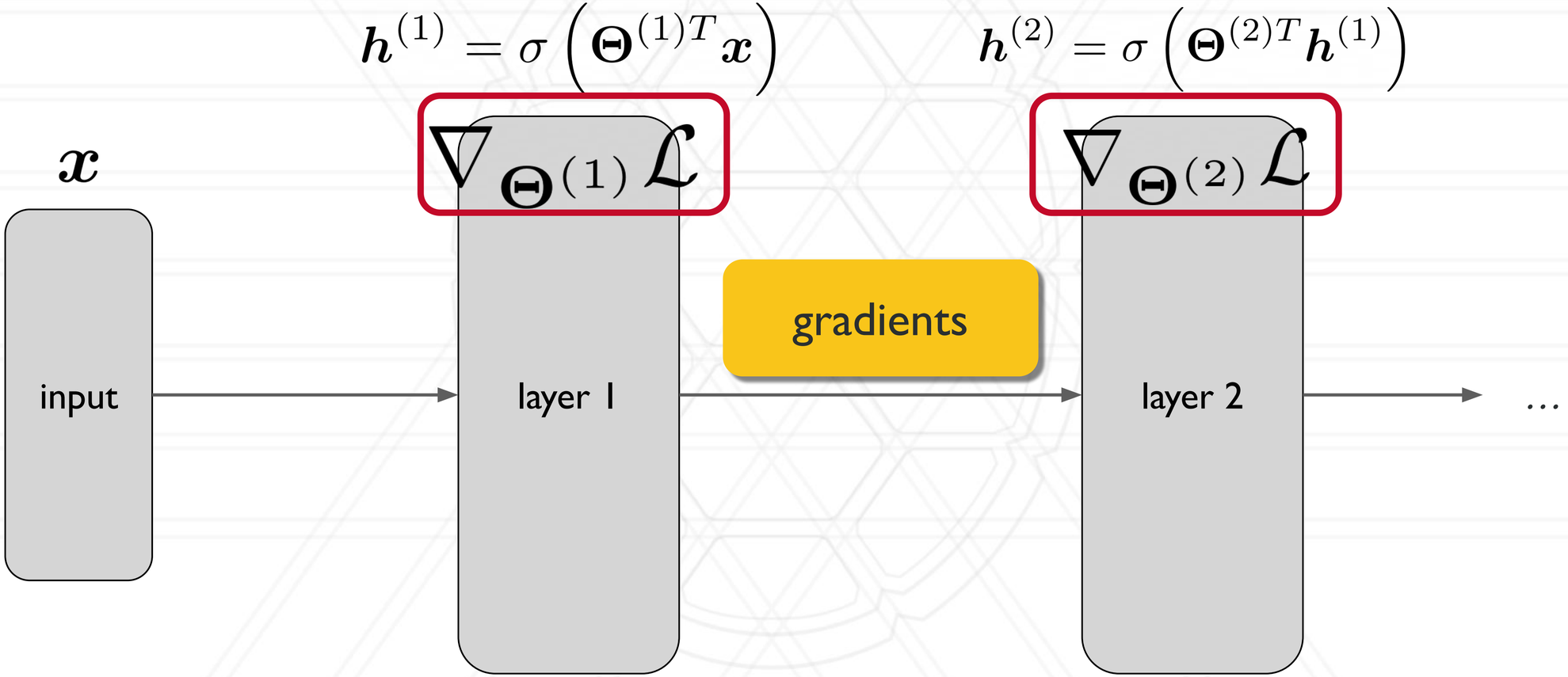
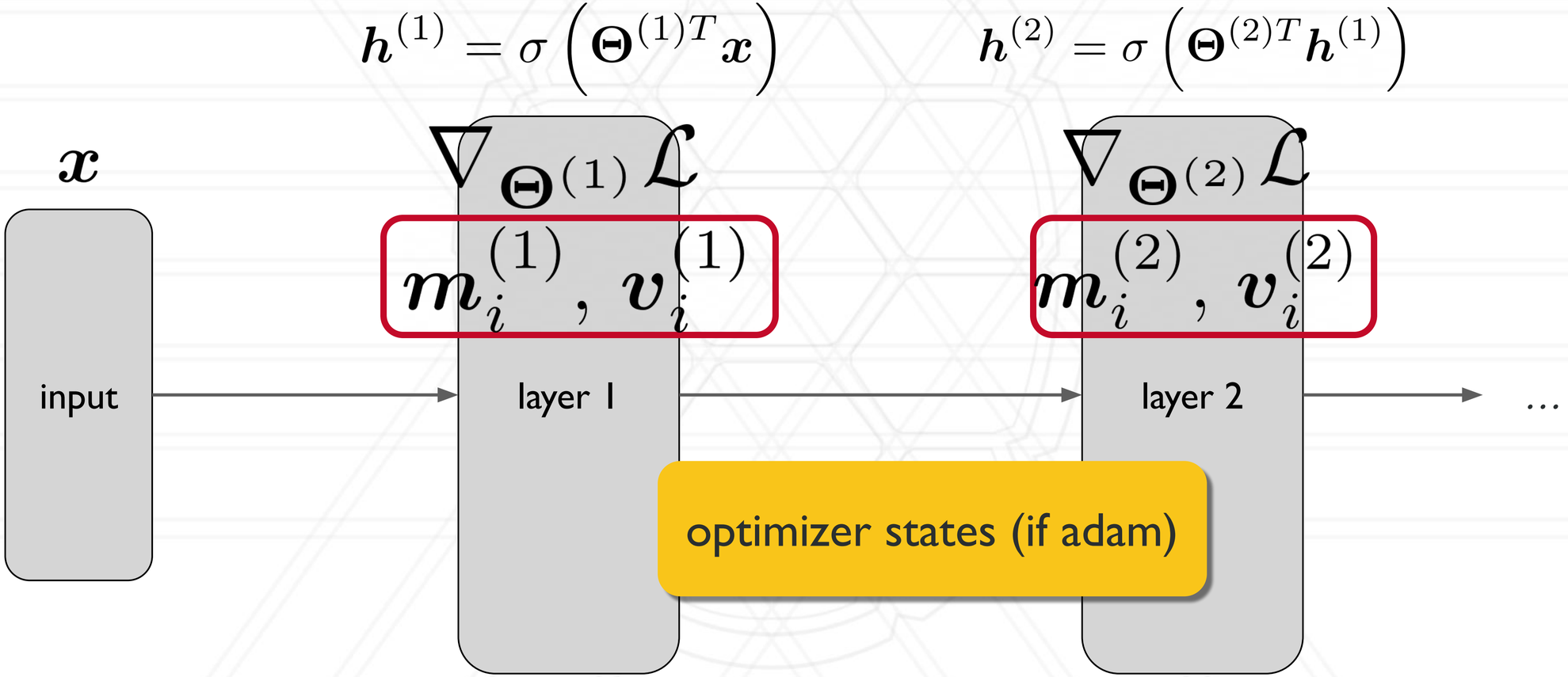


image: https://shivambharuka.medium.com/deep-learning-a-primer-on-distributed-training-part-1-d0ae0054bb1c

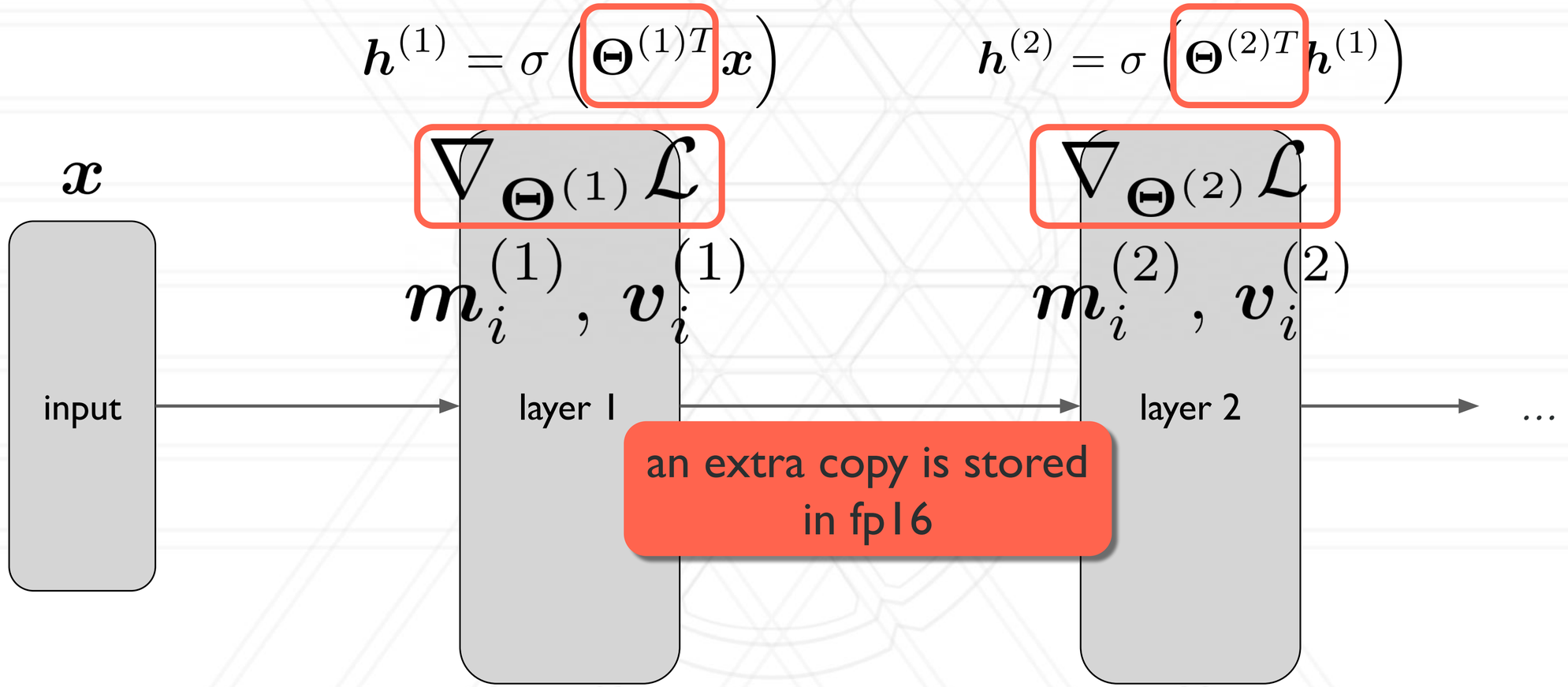
What's using the memory?



What's using the memory?



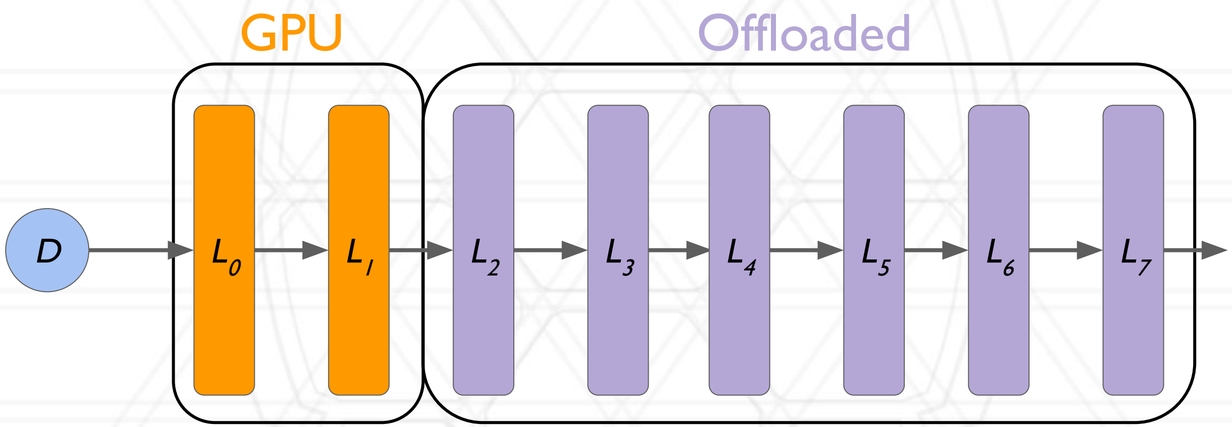
Mixed Precision Training



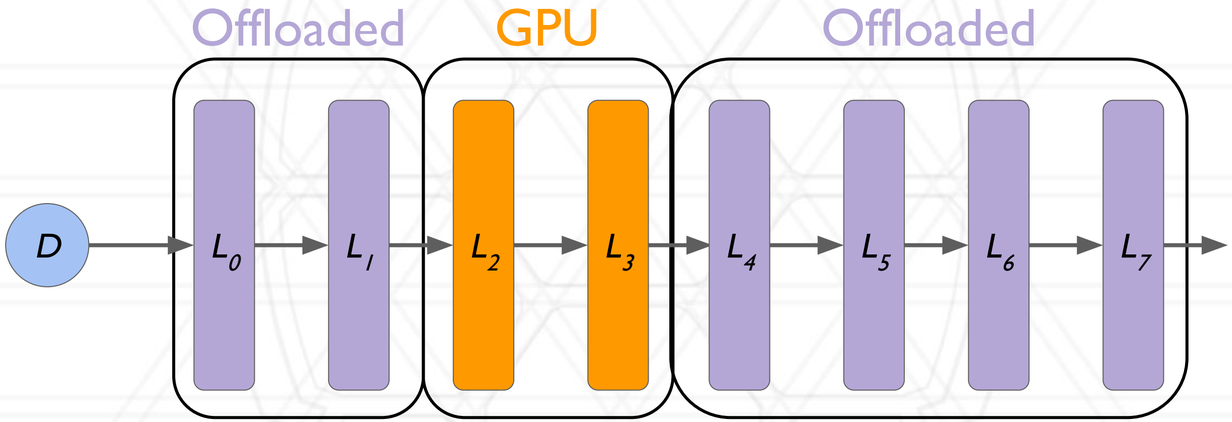
What's using the memory?

- Model state; not dependent on batch size, sequence length, etc.
  - Weights, gradients, optimizer states
- Residual state; dependent on input data
  - Activations
  - Can be reduced through activation checkpointing

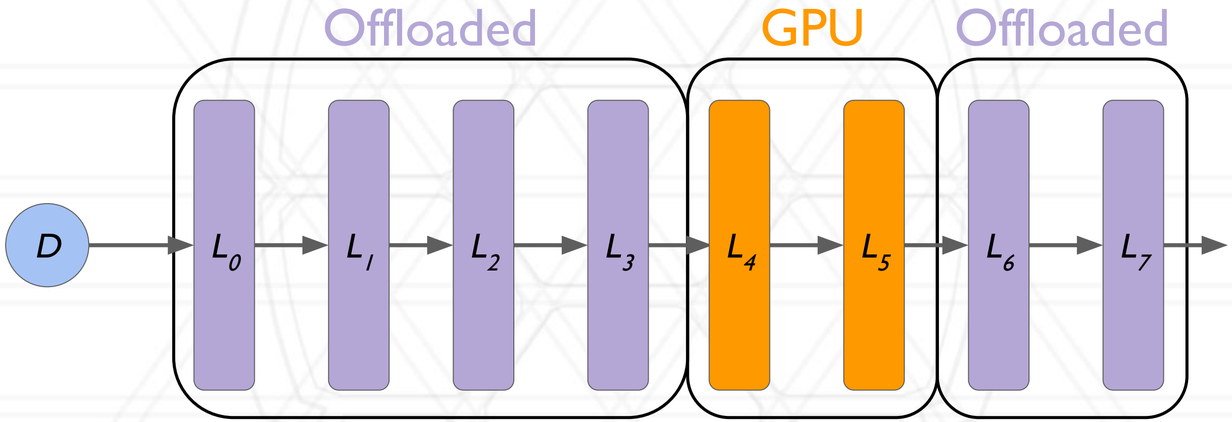
Naive Offloading



Naive Offloading



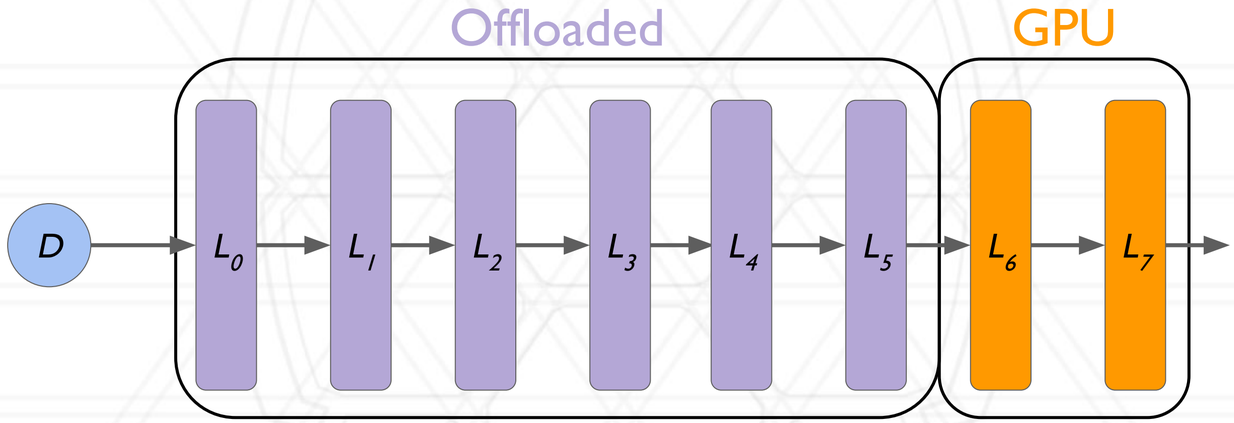
Naive Offloading



Naive Offloading

This is very slow…

How can we improve it?

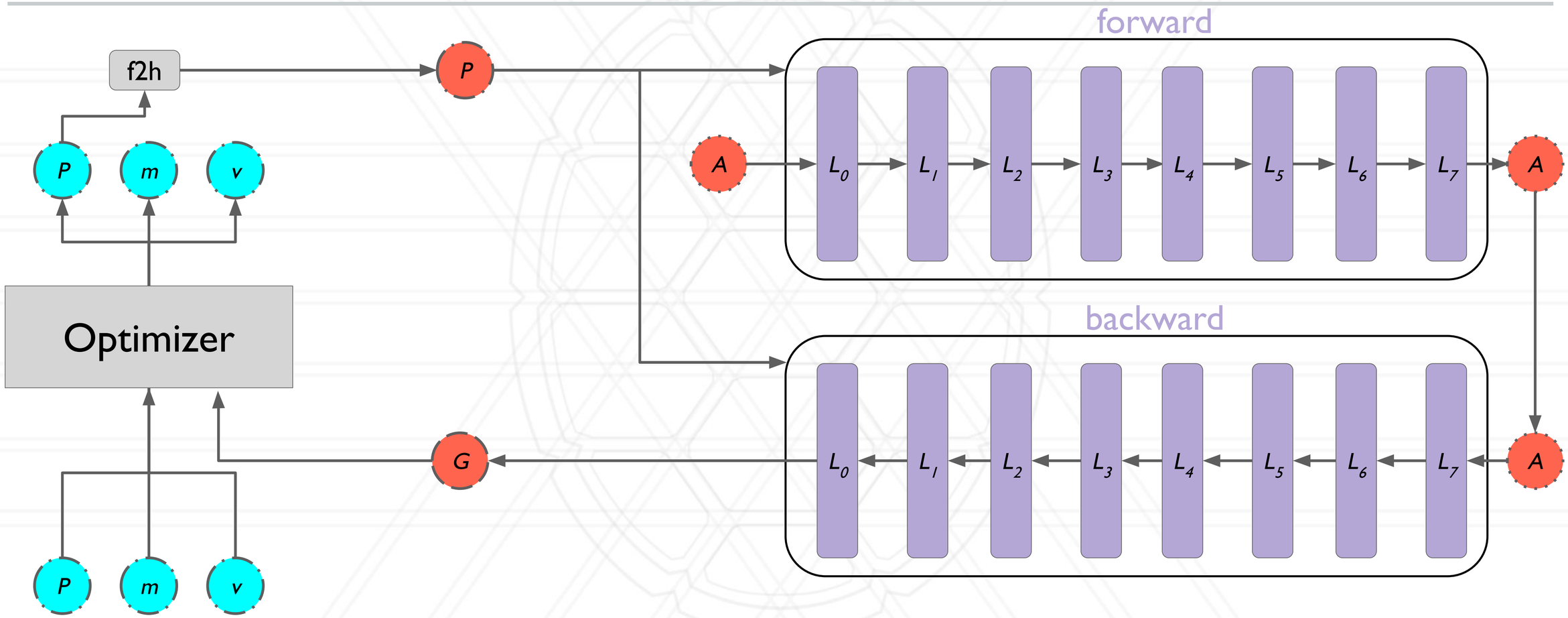


ZeRO-Offload

- An earlier work offloading to CPU memory
- Partition training dataflow graph
  - Put less intensive compute on CPU
  - Minimize CPU-GPU copying

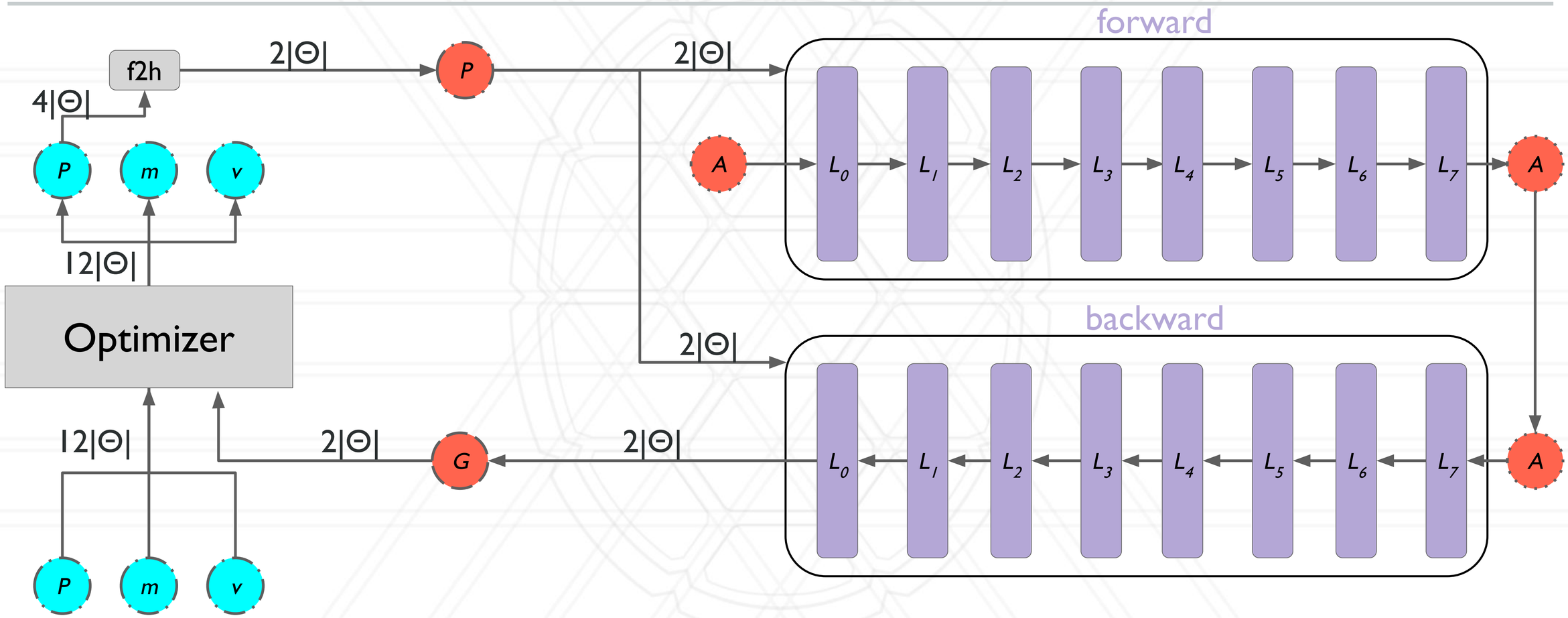
J. Ren, et. al., "ZeRO-Offload: Democratizing Billion-Scale Model Training" 2021

ZeRO-Offload



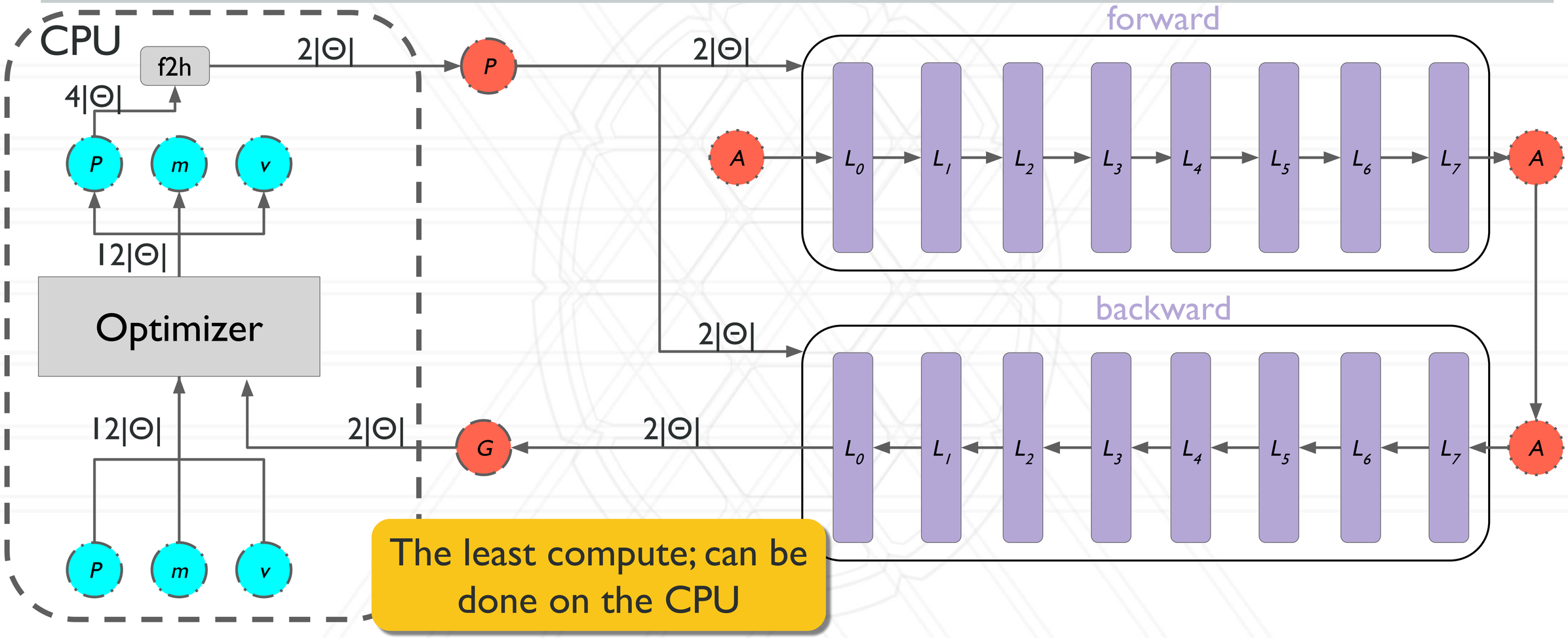
J. Ren, et. al., "ZeRO-Offload: Democratizing Billion-Scale Model Training" 2021

ZeRO-Offload



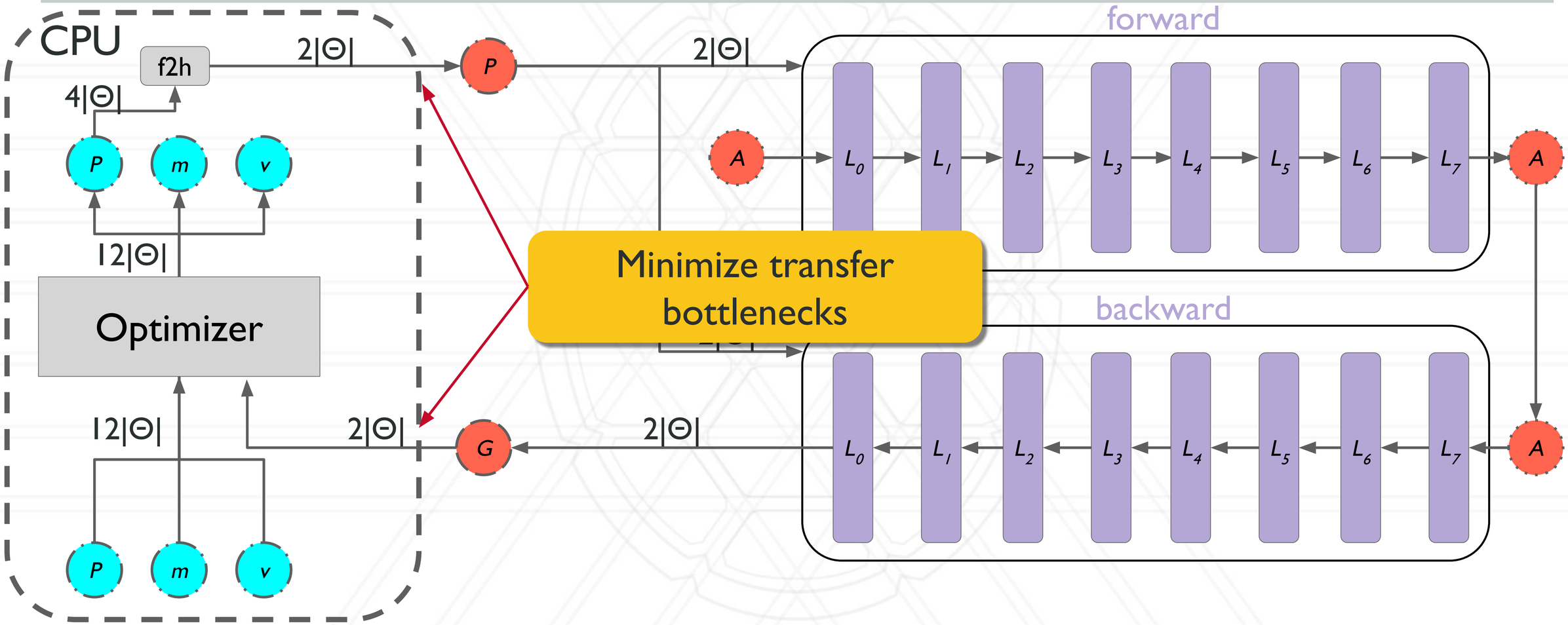
J. Ren, et. al., "ZeRO-Offload: Democratizing Billion-Scale Model Training" 2021

ZeRO-Offload



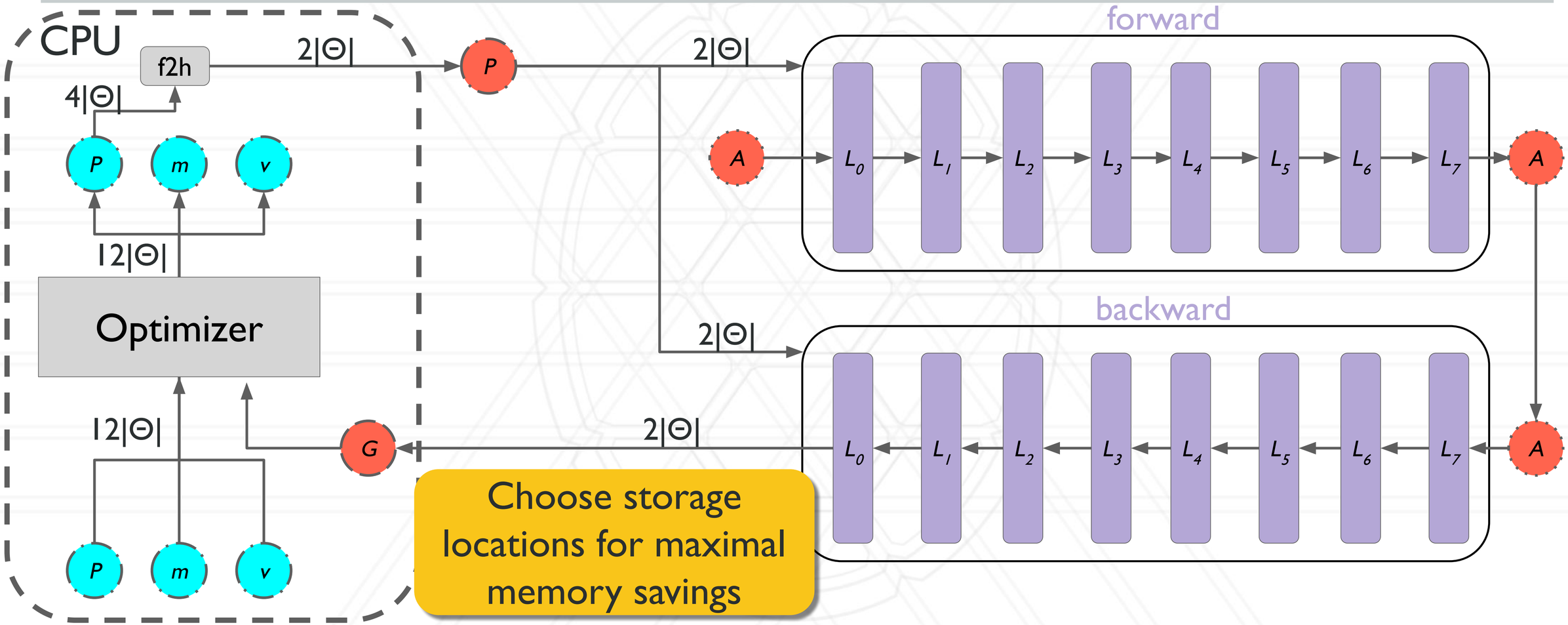
J. Ren, et. al., "ZeRO-Offload: Democratizing Billion-Scale Model Training" 2021

ZeRO-Offload



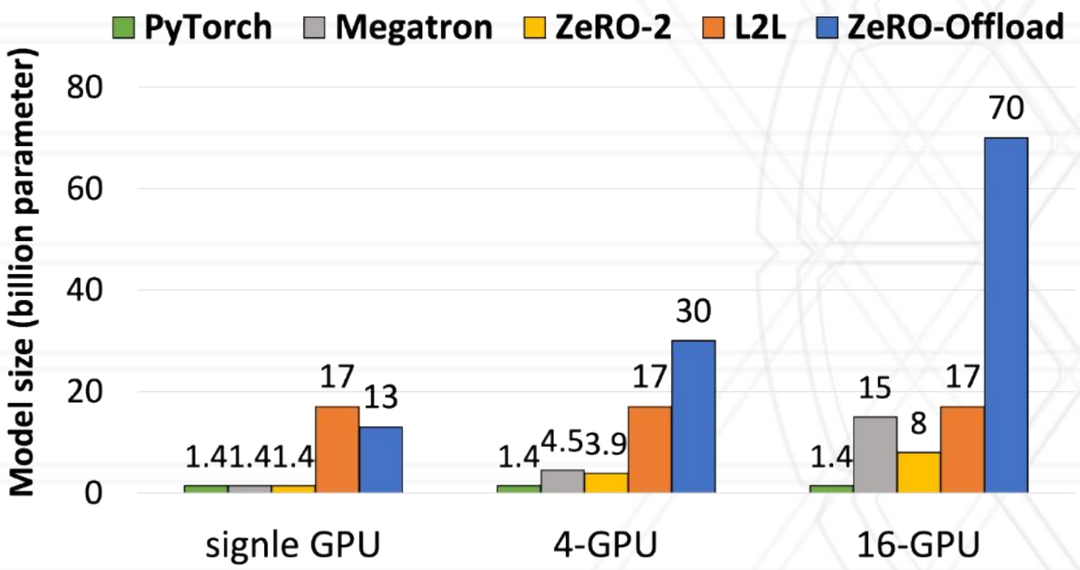
J. Ren, et. al., "ZeRO-Offload: Democratizing Billion-Scale Model Training" 2021

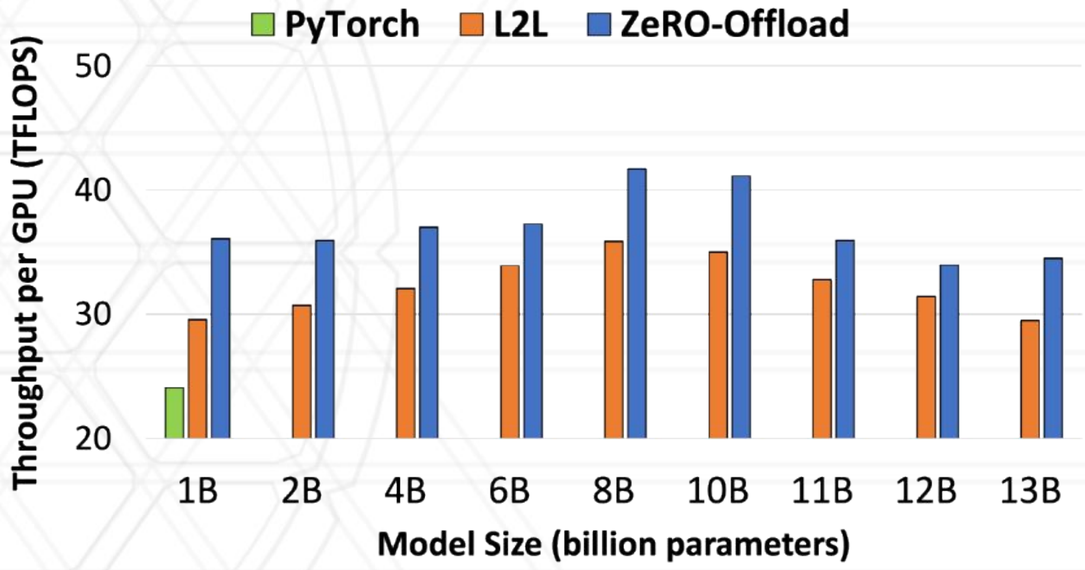
ZeRO-Offload



J. Ren, et. al., "ZeRO-Offload: Democratizing Billion-Scale Model Training" 2021

ZeRO-Offload



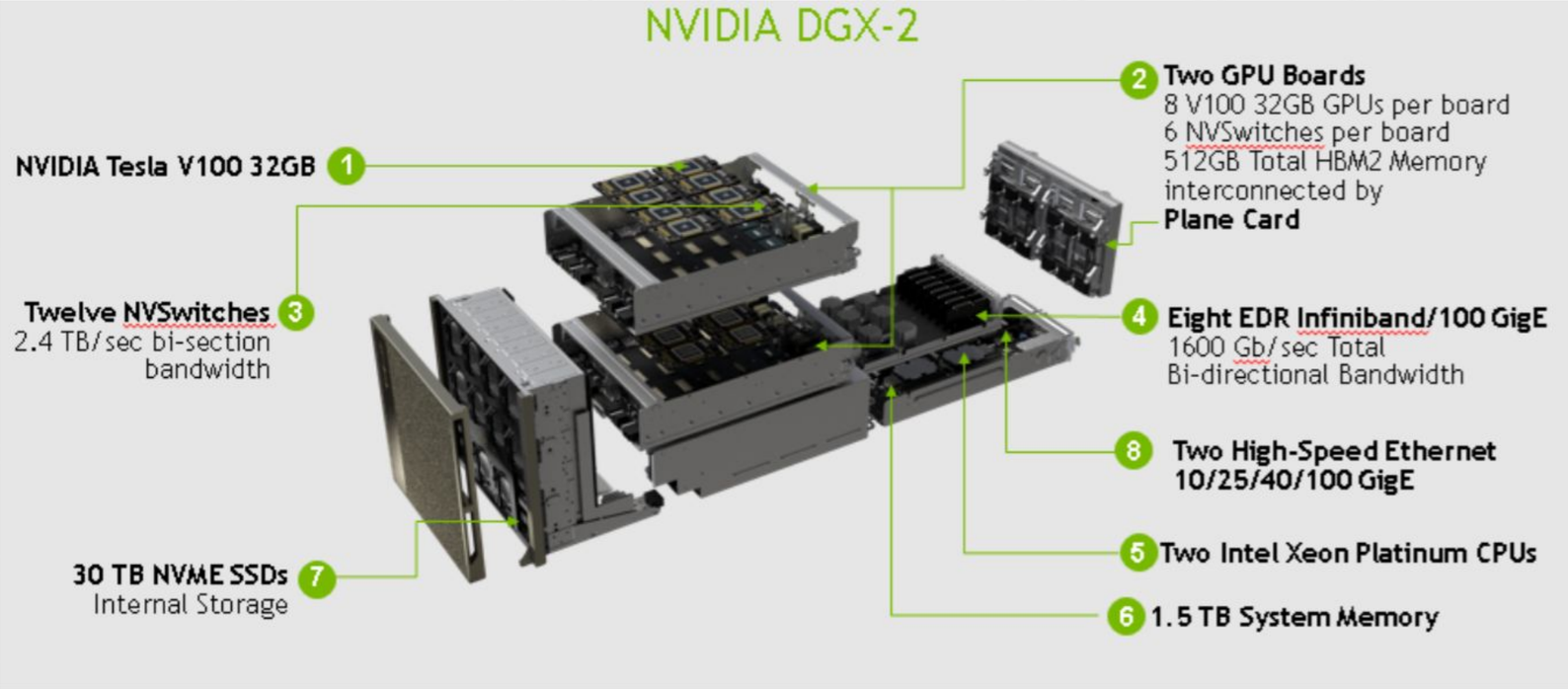


Zero-Infinity

- Modern HPC clusters have lots of memory per node
  - DGX-2
    - 512 GB GPU Memory
    - 1.5 TB CPU Memory
    - 30 TB NVMe Storage
- GPT-3 requires ~2-4 TB to fine-tune
  - Do we need multiple nodes?
  - Could we use all of the available memory to train larger models?

S. Rajbhandari, et. al., "ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning" SC '21

DGX-2



S. Rajbhandari, et. al., "ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning" SC '21

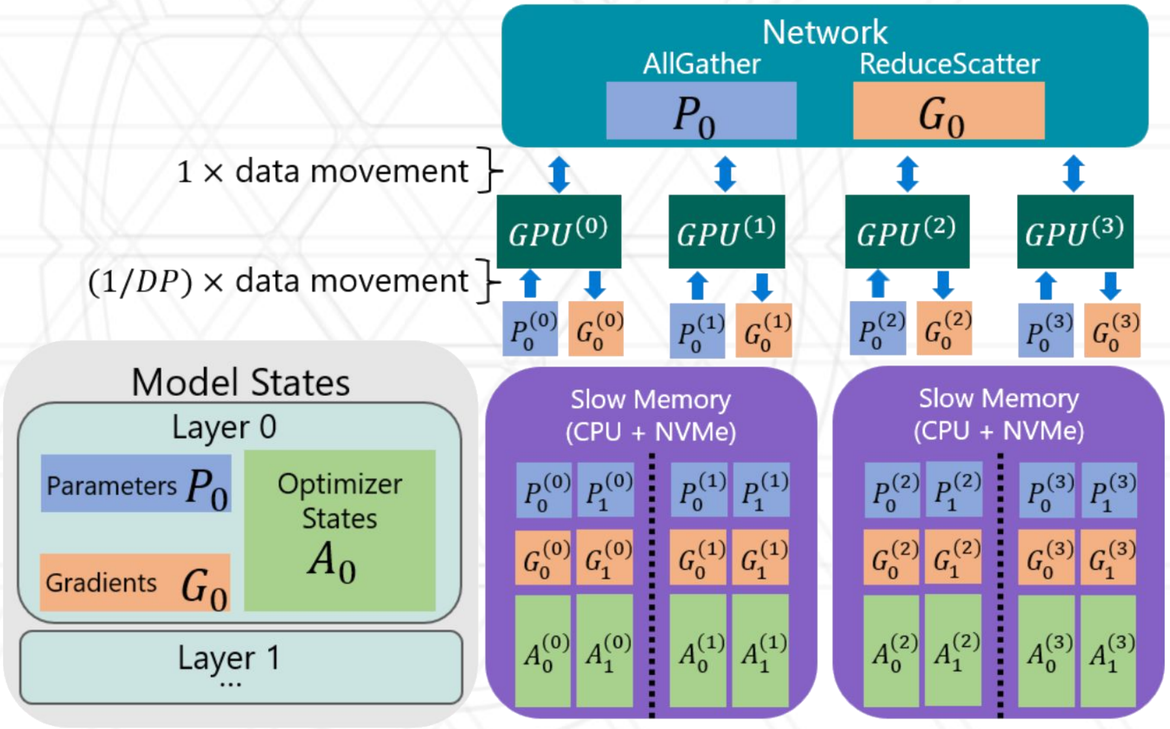
Built on Top of ZeRO

- Deepspeed ZeRO is a precursor to FSDP with 3 stages
  - Stage 1: Split optimizer states
  - Stage 2: Split optimizer states + gradients
  - Stage 3: Split optimizer states + gradients + parameters
- No code refactoring!

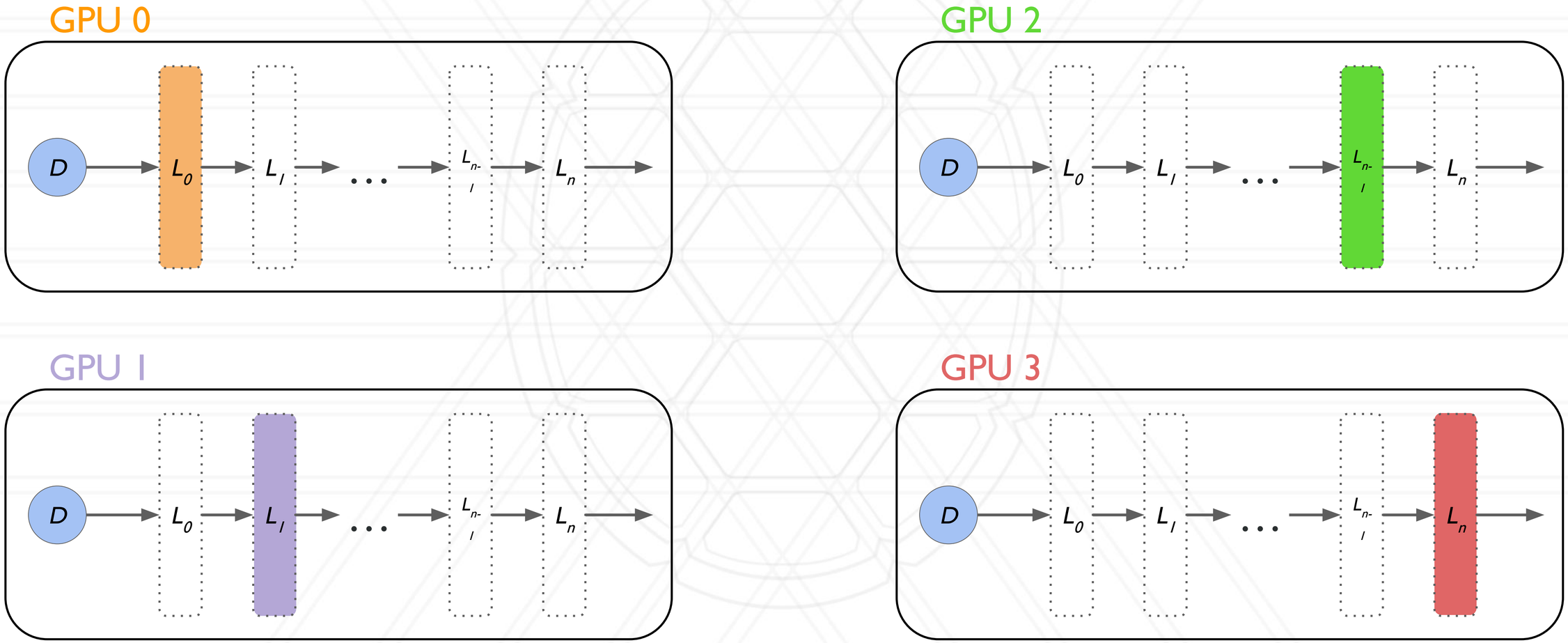
Offloading Parameters in ZeRO

- So can we just offload ZeRO parameters to CPU/NVMe?

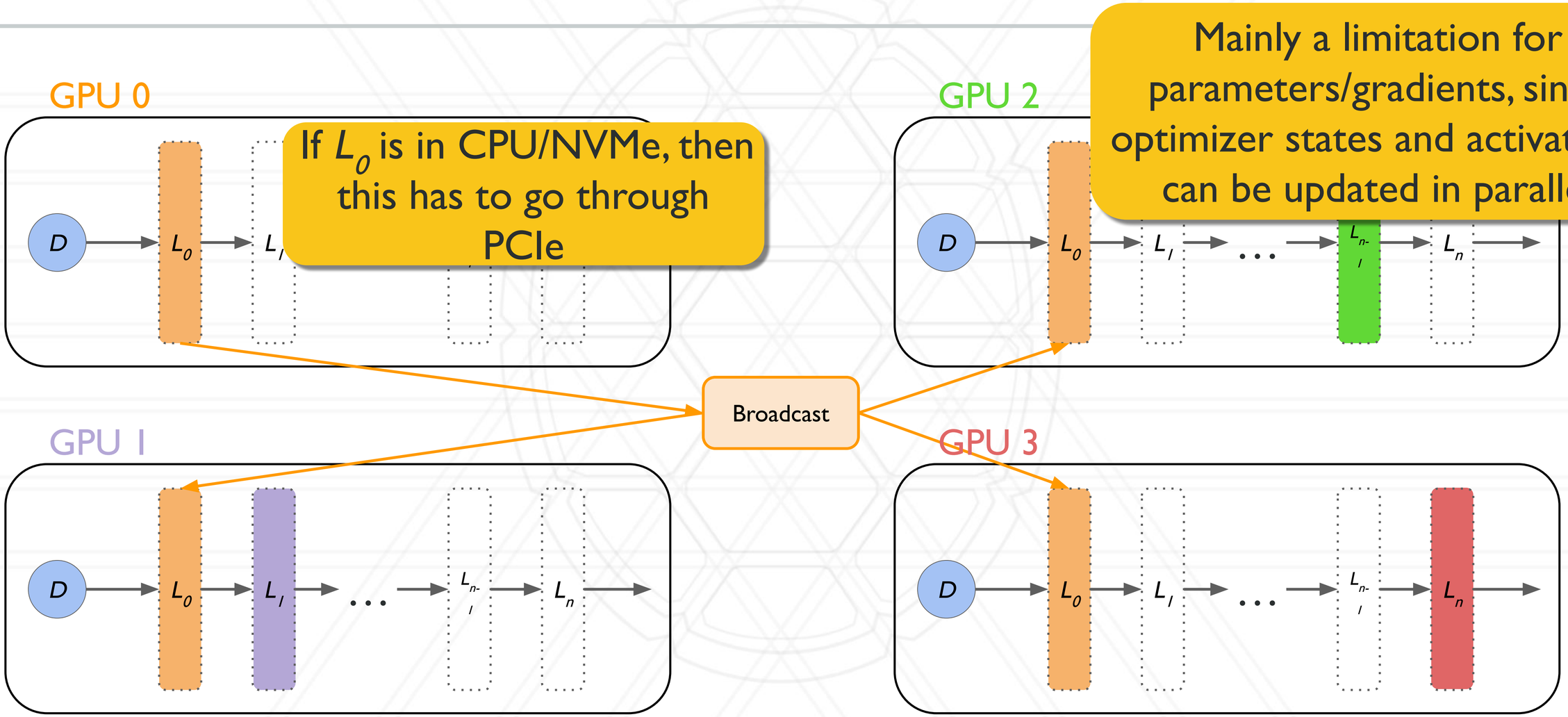
This will be very slow!



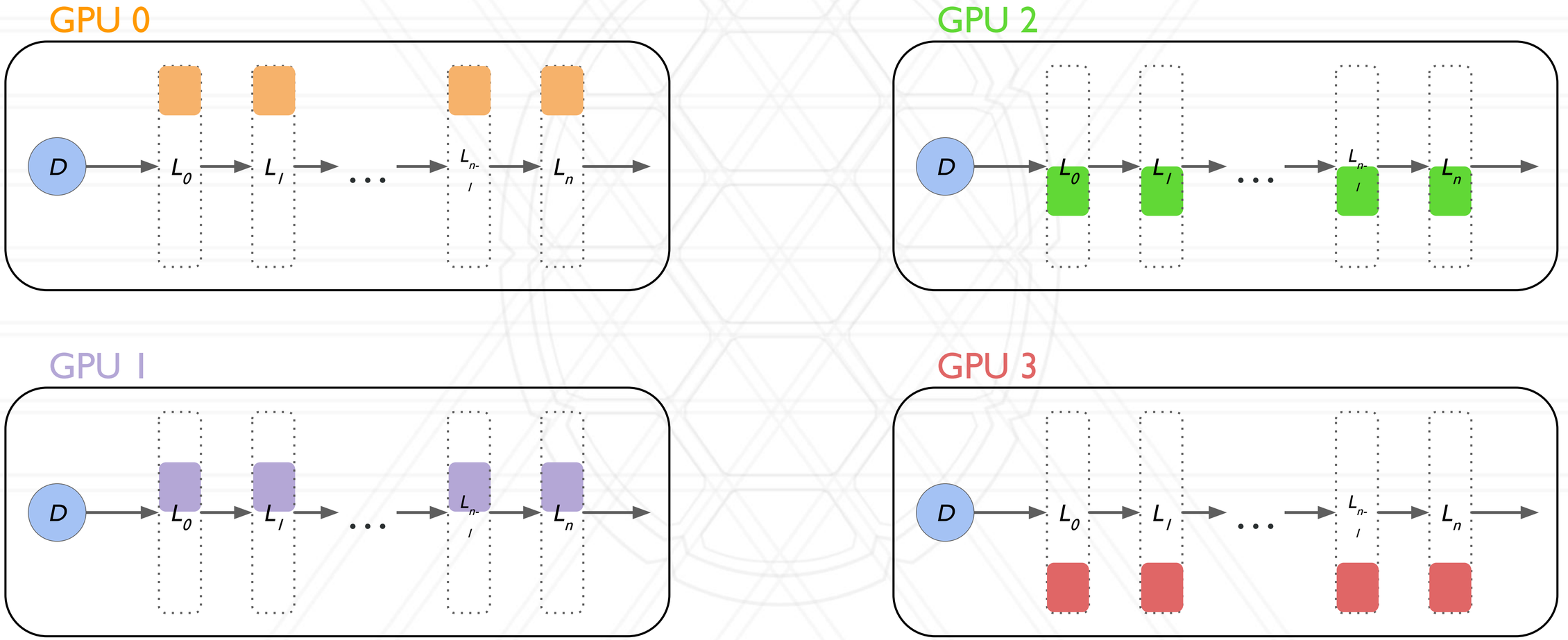
Limited by PCI Bandwidth



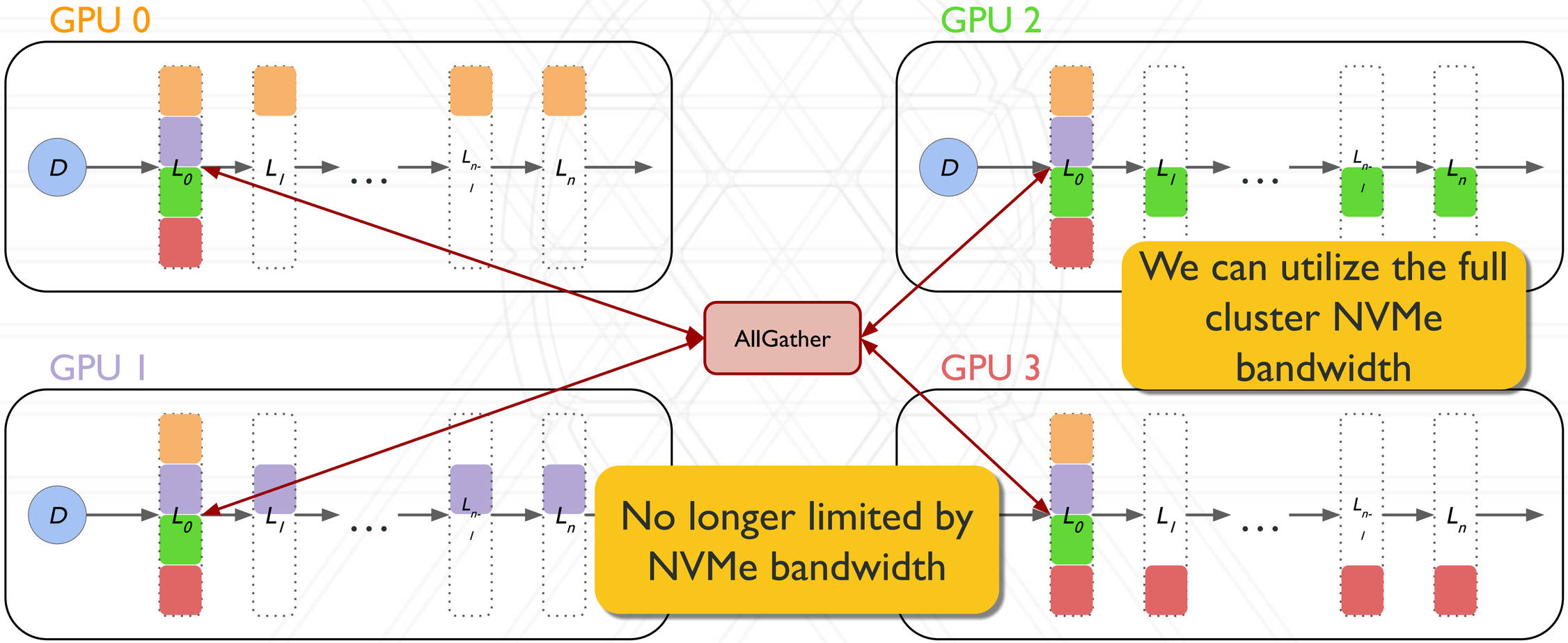
Limited by PCI Bandwidth



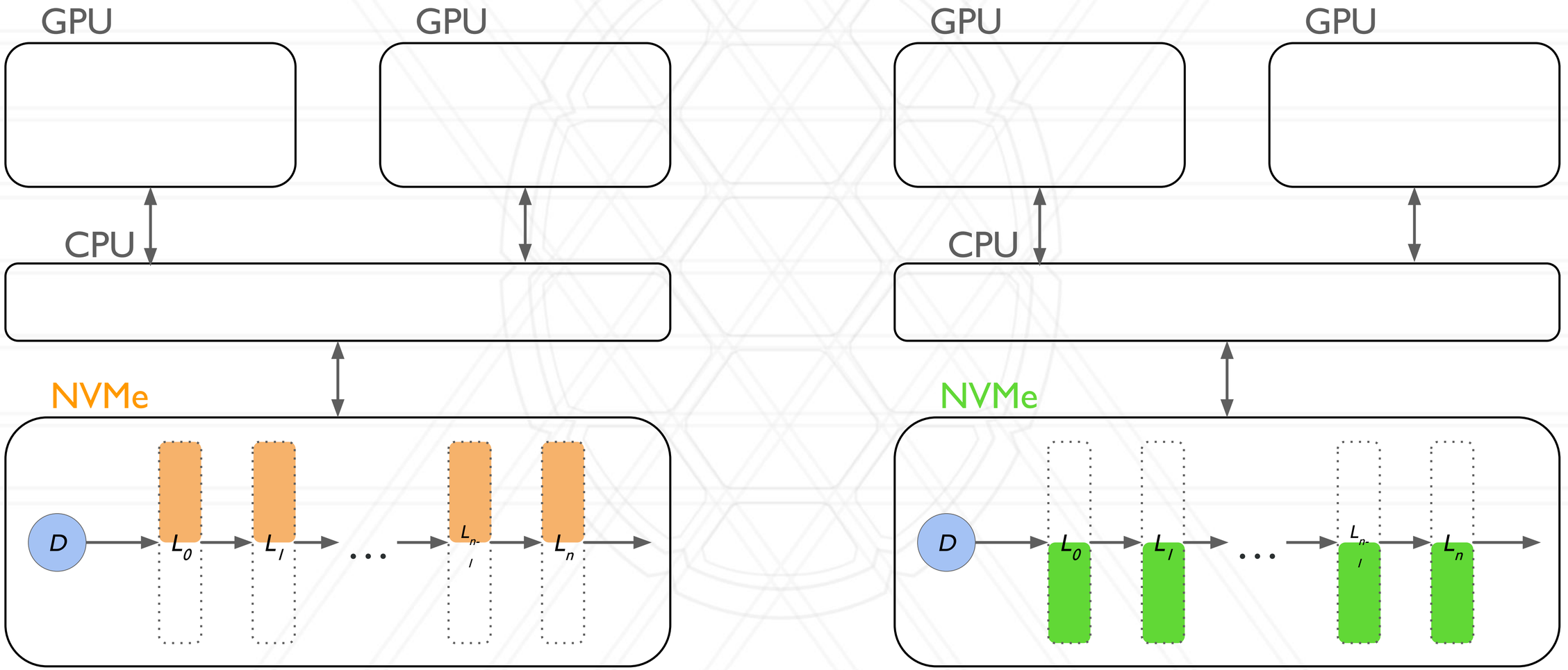
Bandwidth Centric Partitioning



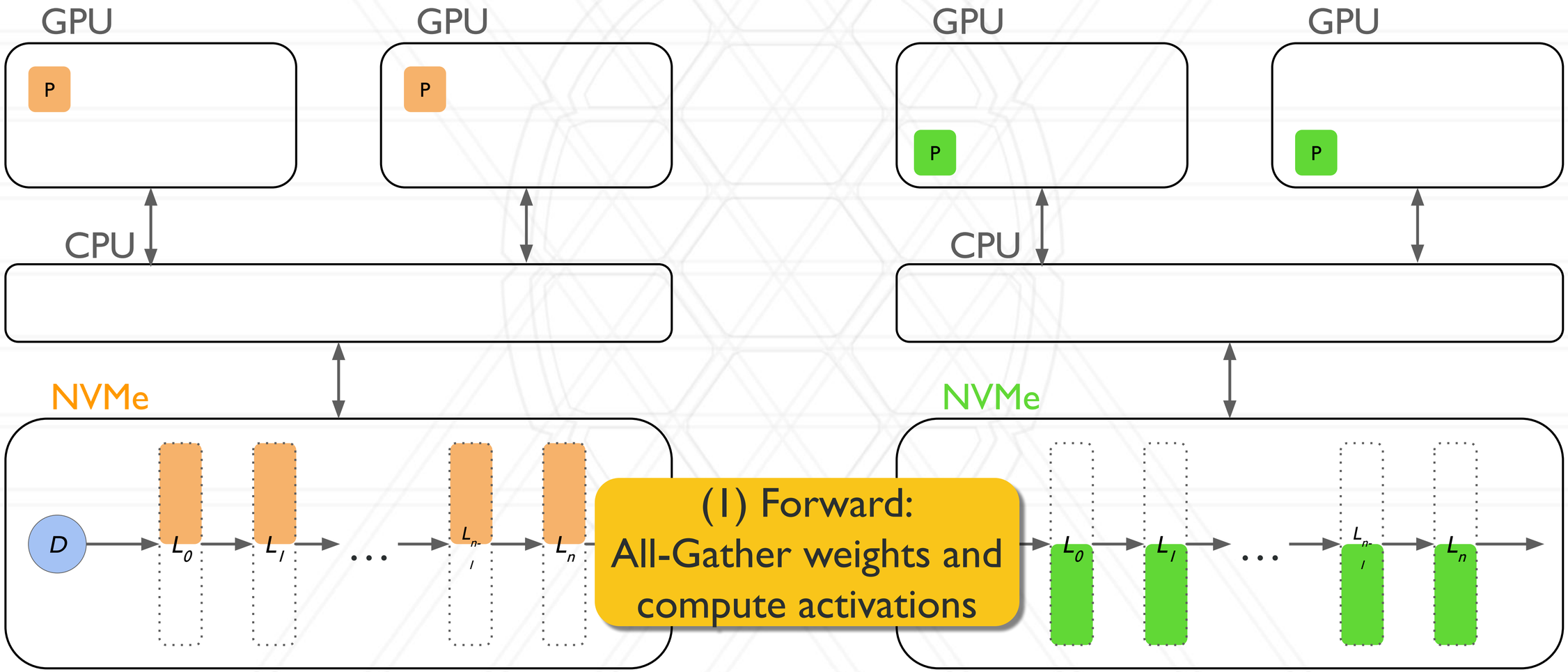
Bandwidth Centric Partitioning



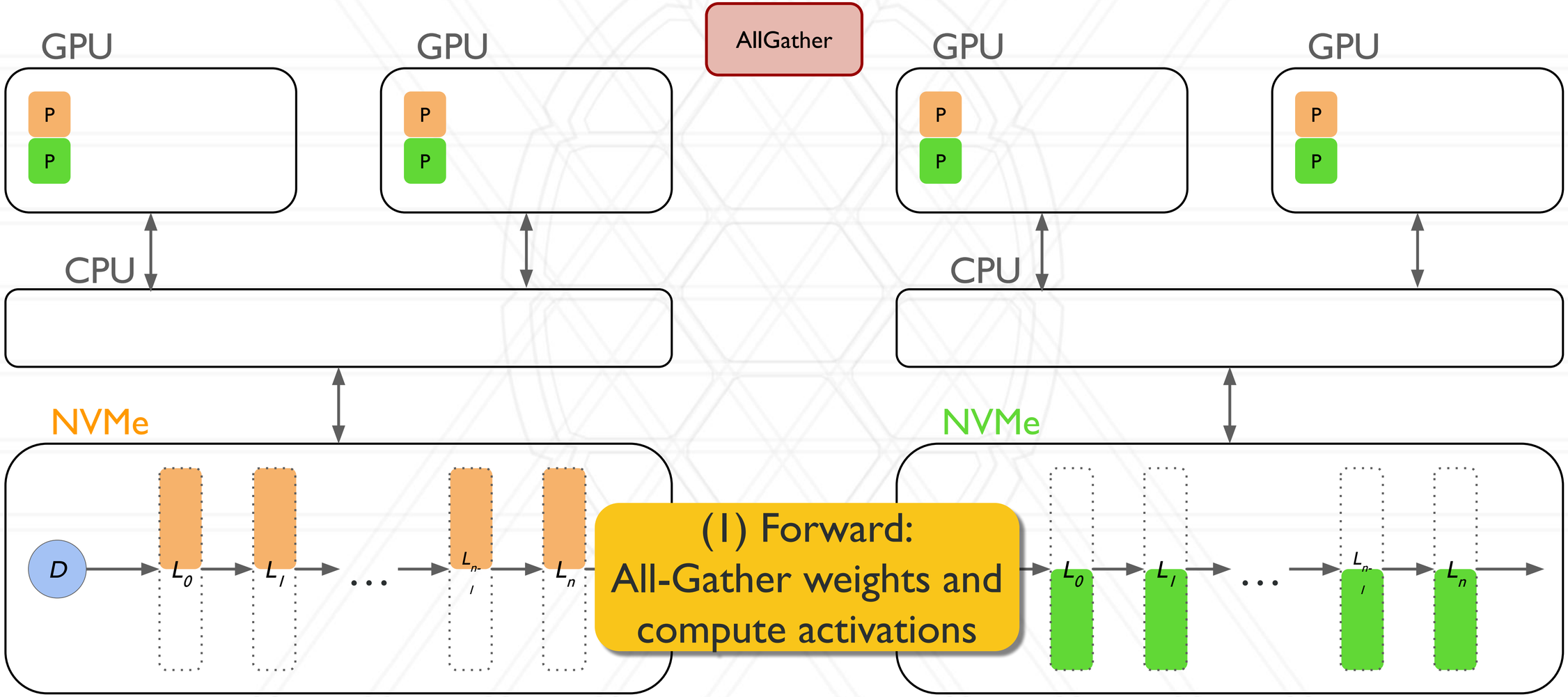
ZeRO Infinity Algorithm



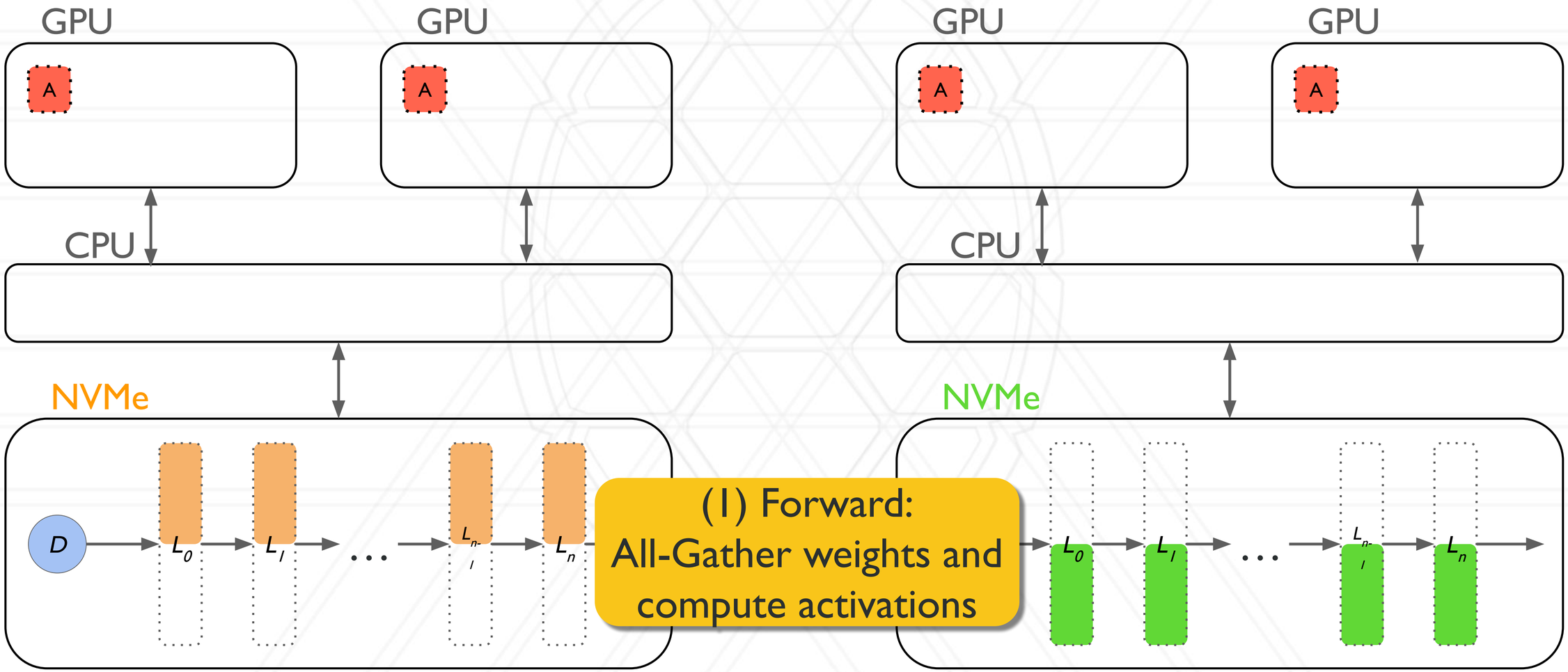
ZeRO Infinity Algorithm



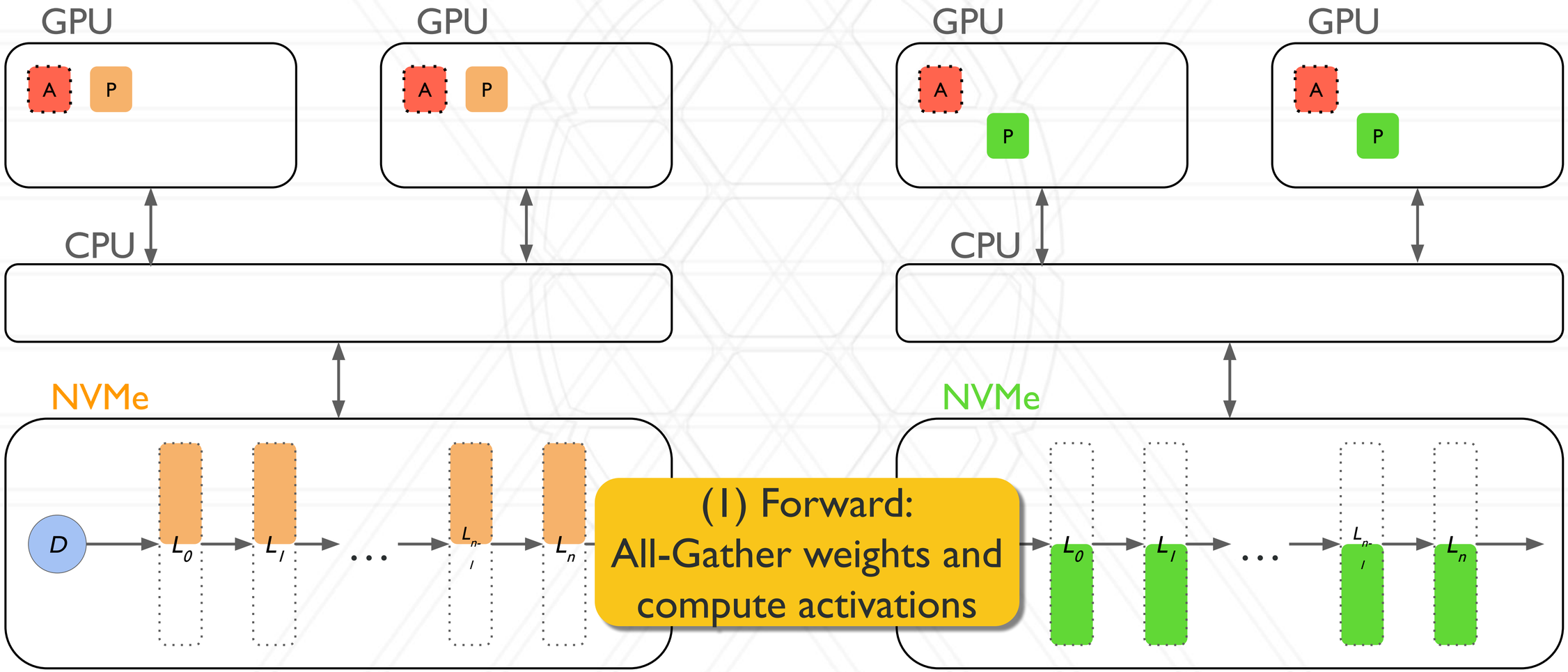
ZeRO Infinity Algorithm



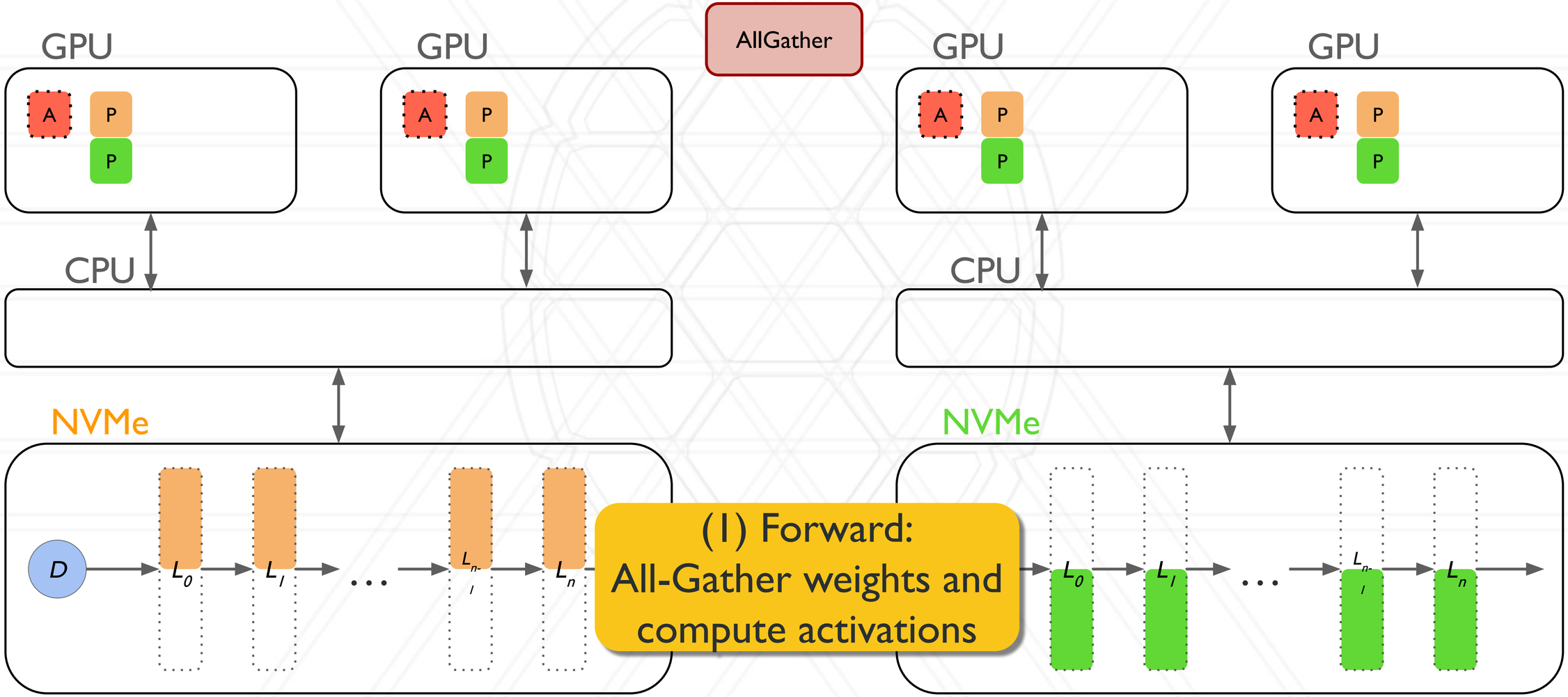
ZeRO Infinity Algorithm



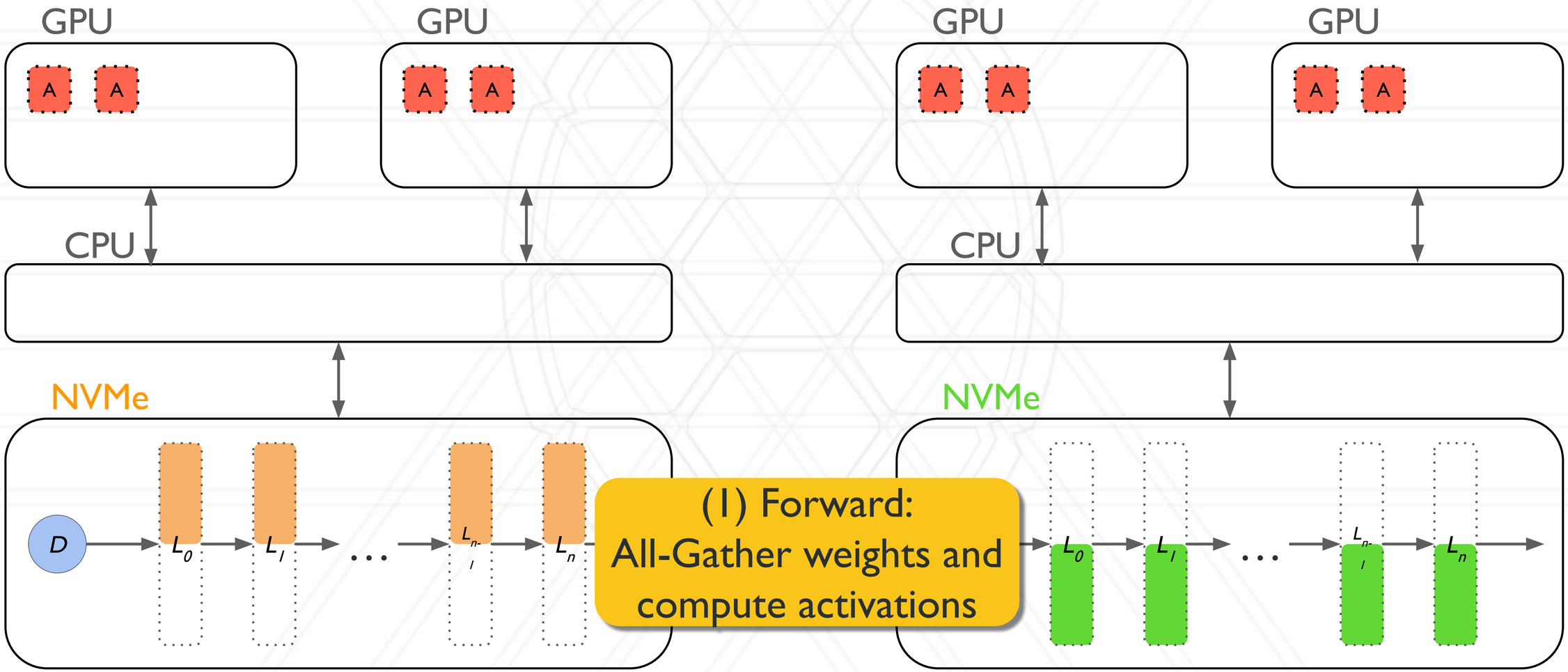
ZeRO Infinity Algorithm



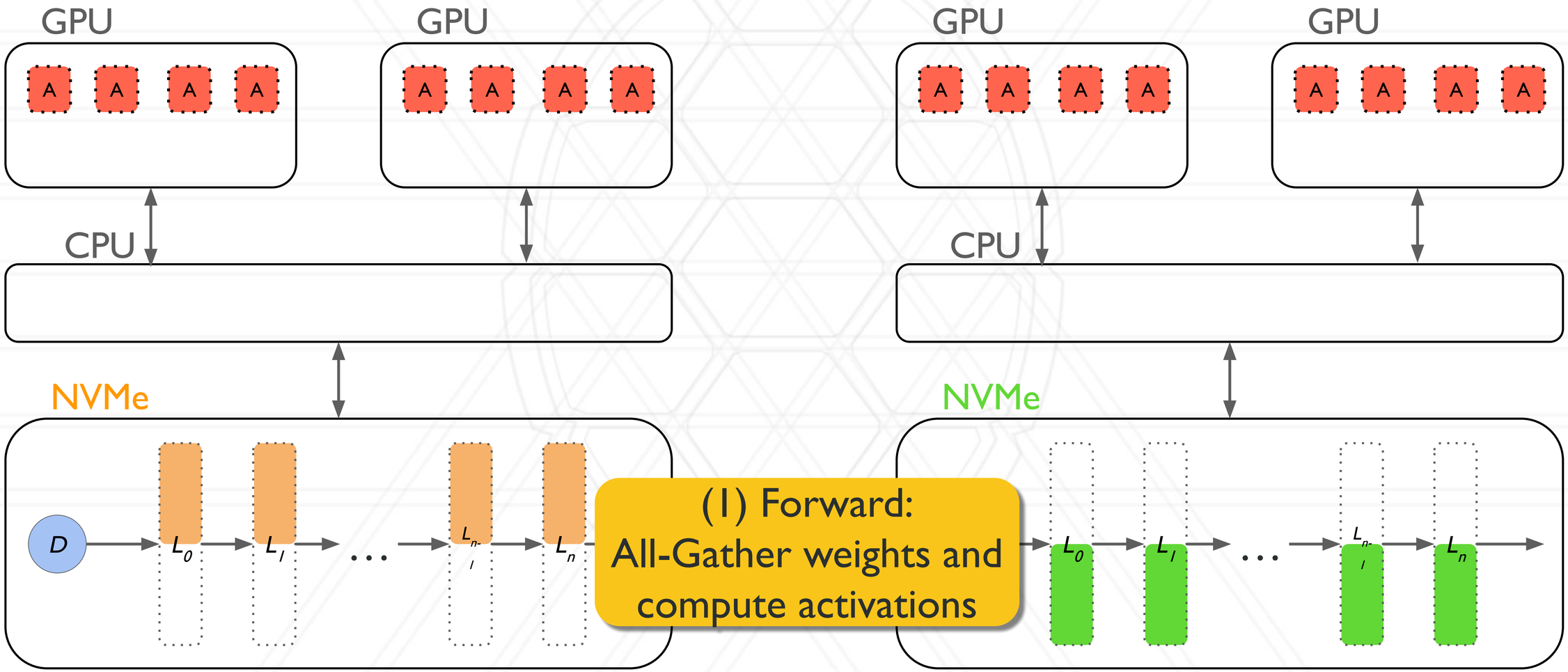
ZeRO Infinity Algorithm



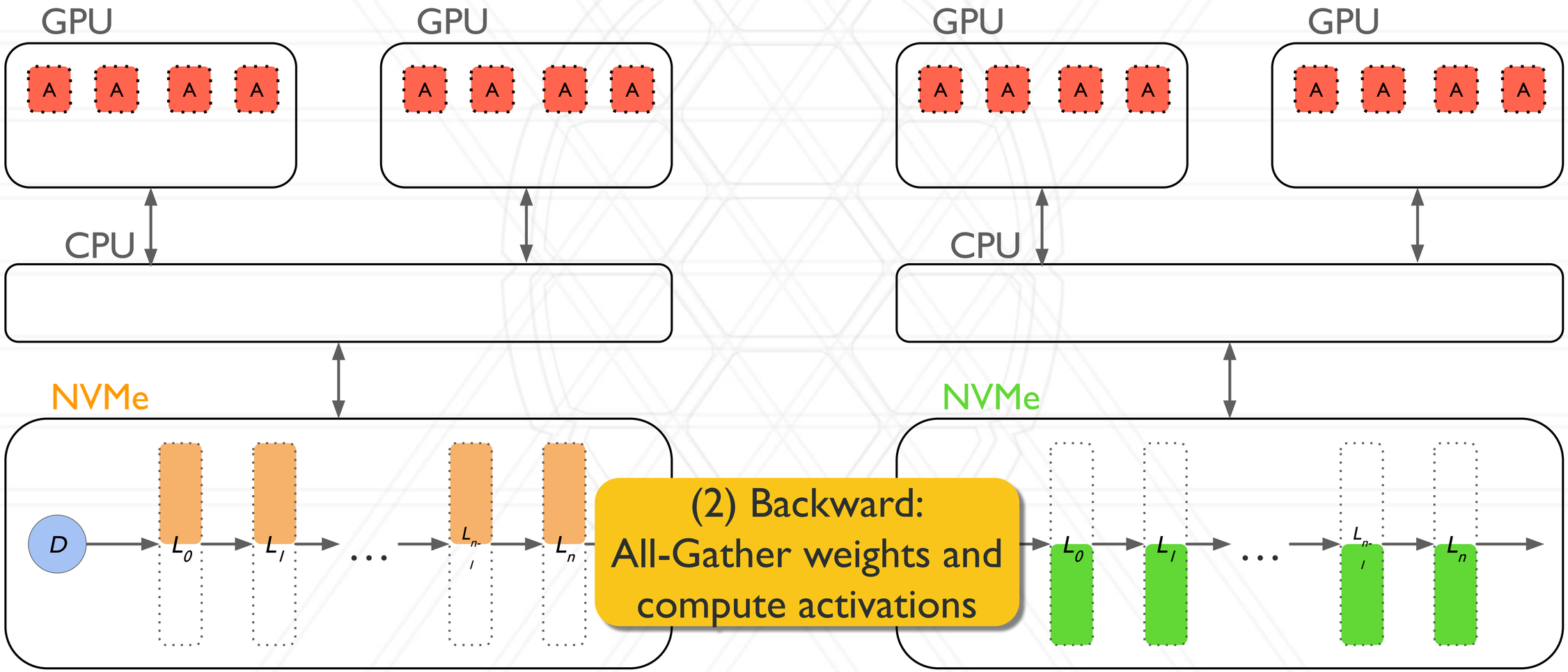
ZeRO Infinity Algorithm



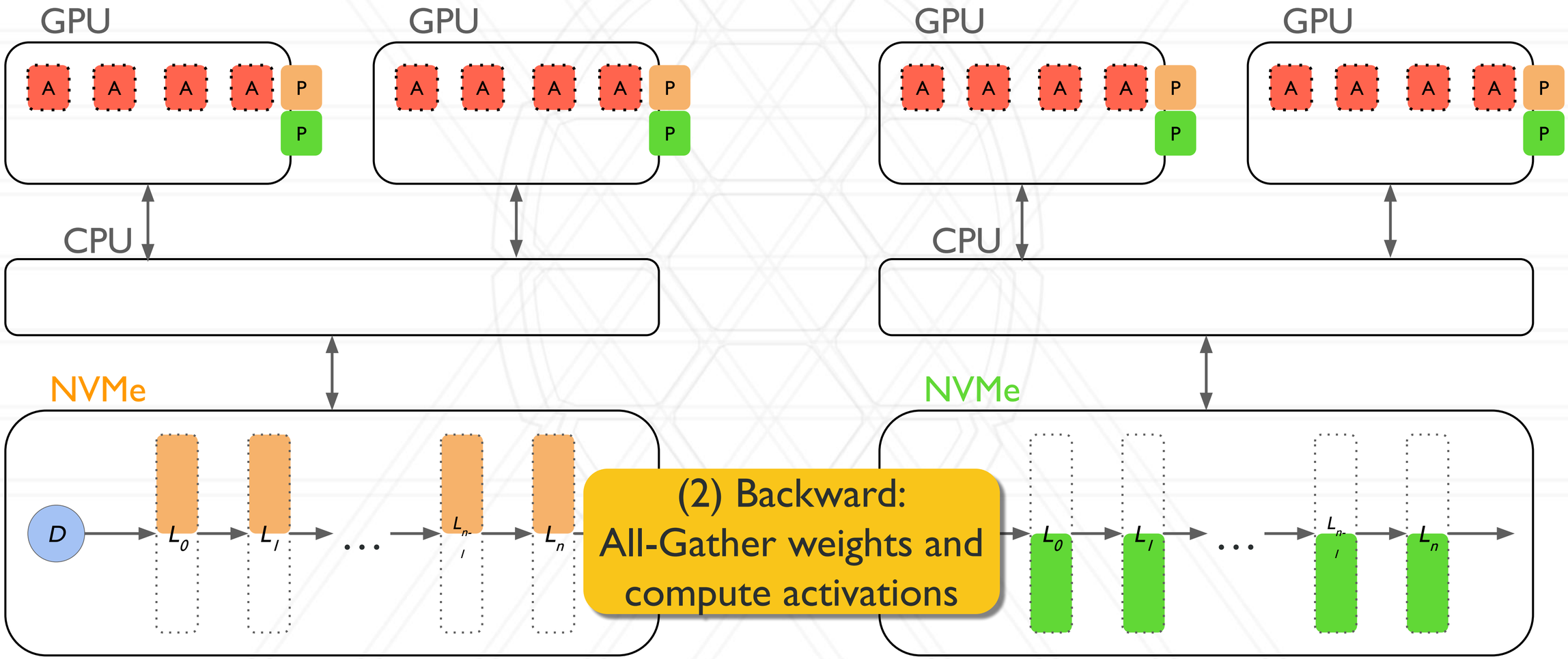
ZeRO Infinity Algorithm



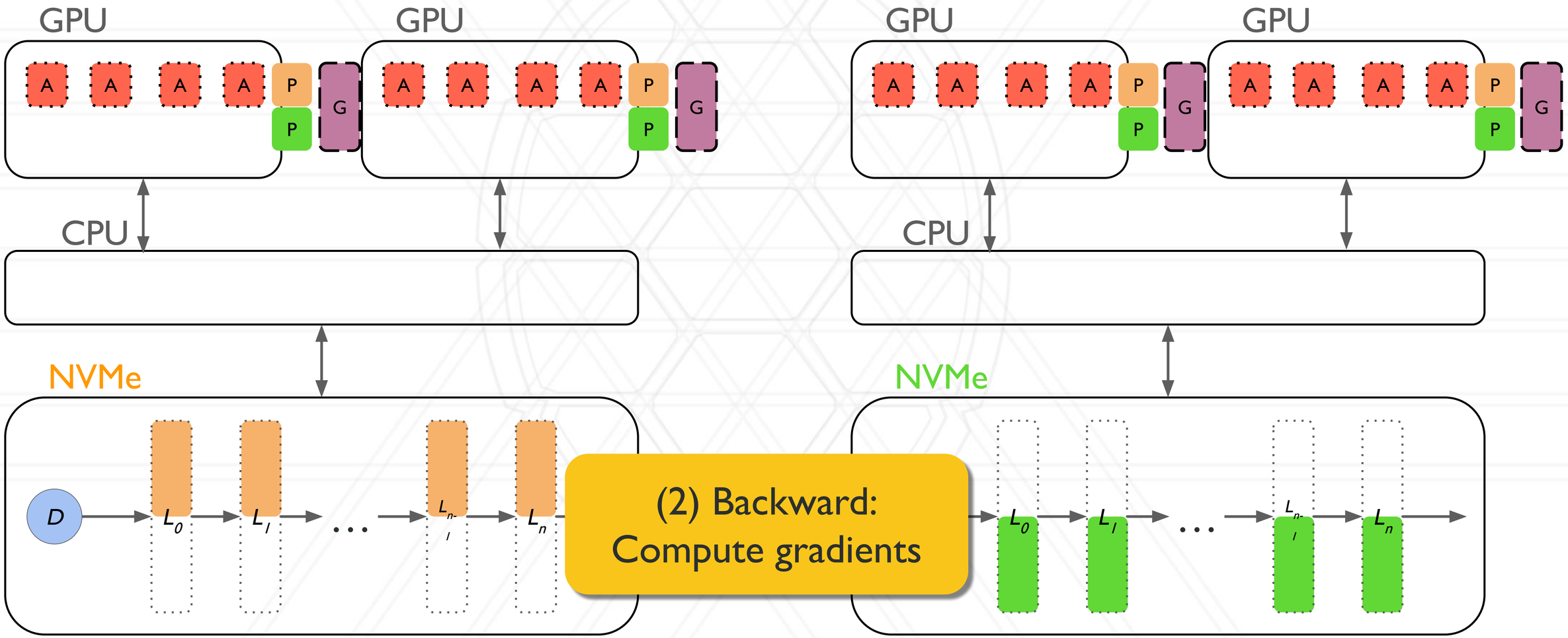
ZeRO Infinity Algorithm



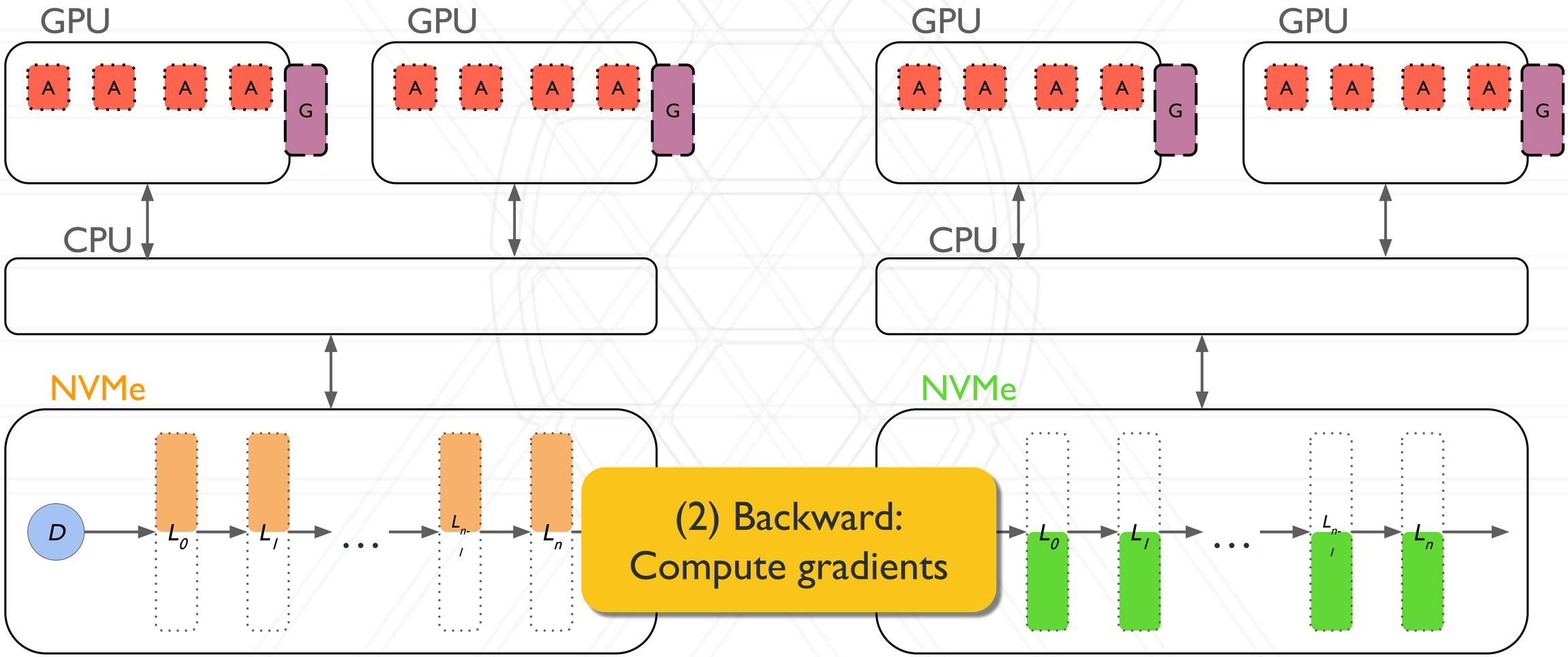
ZeRO Infinity Algorithm



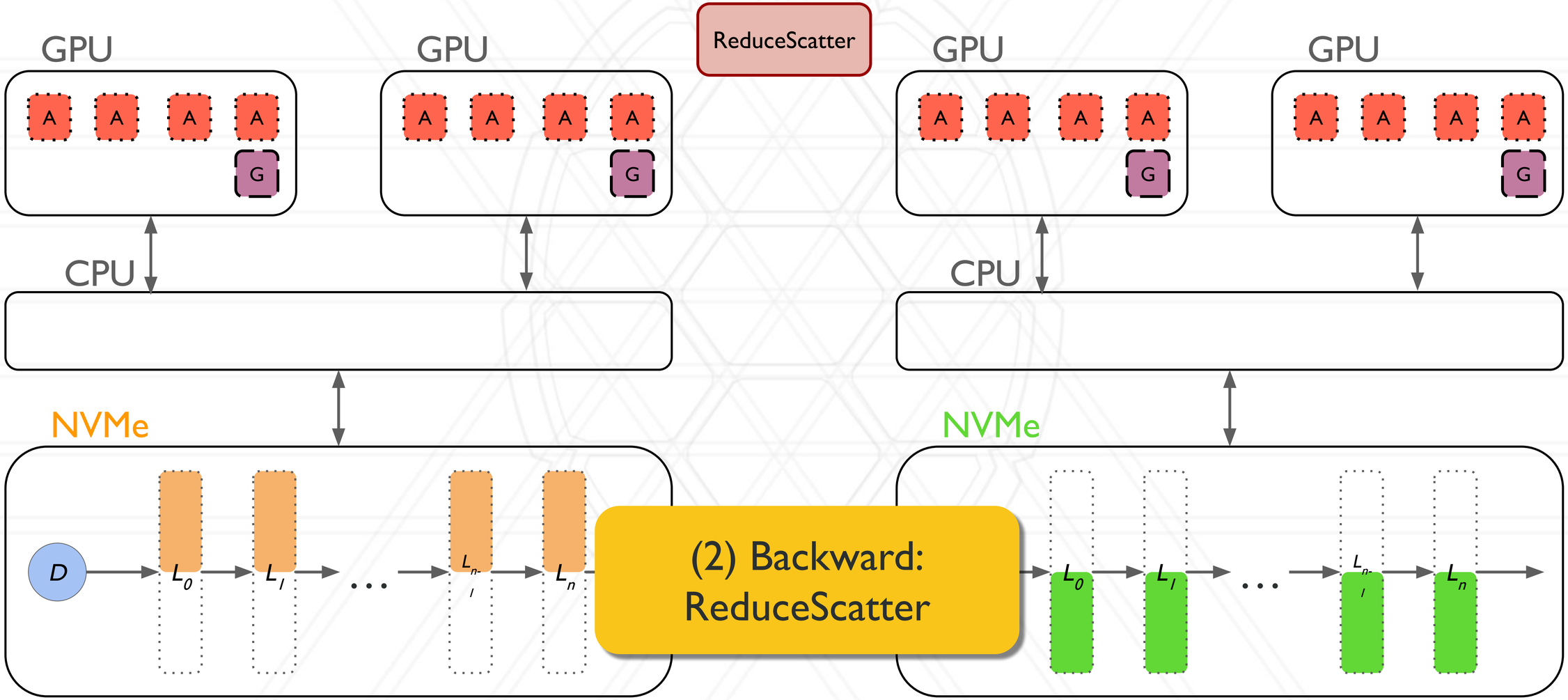
ZeRO Infinity Algorithm



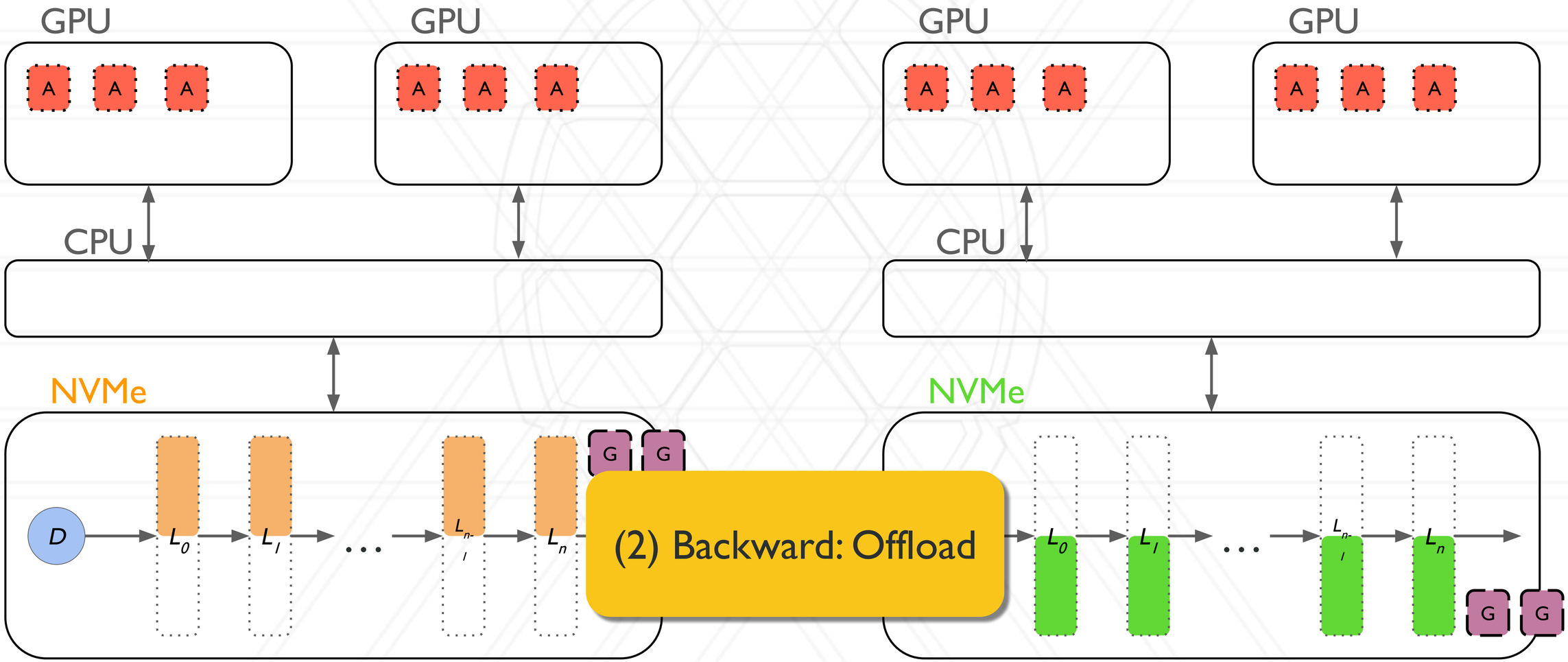
ZeRO Infinity Algorithm



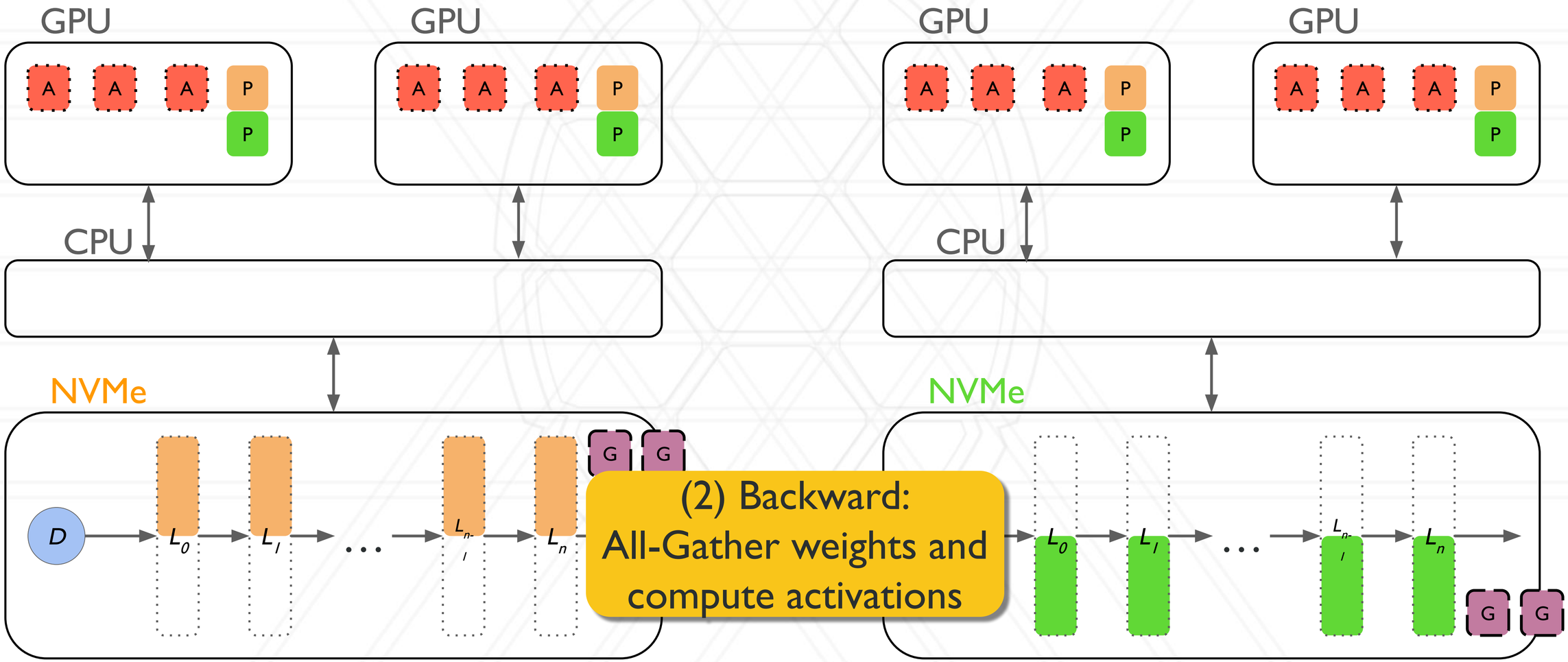
ZeRO Infinity Algorithm



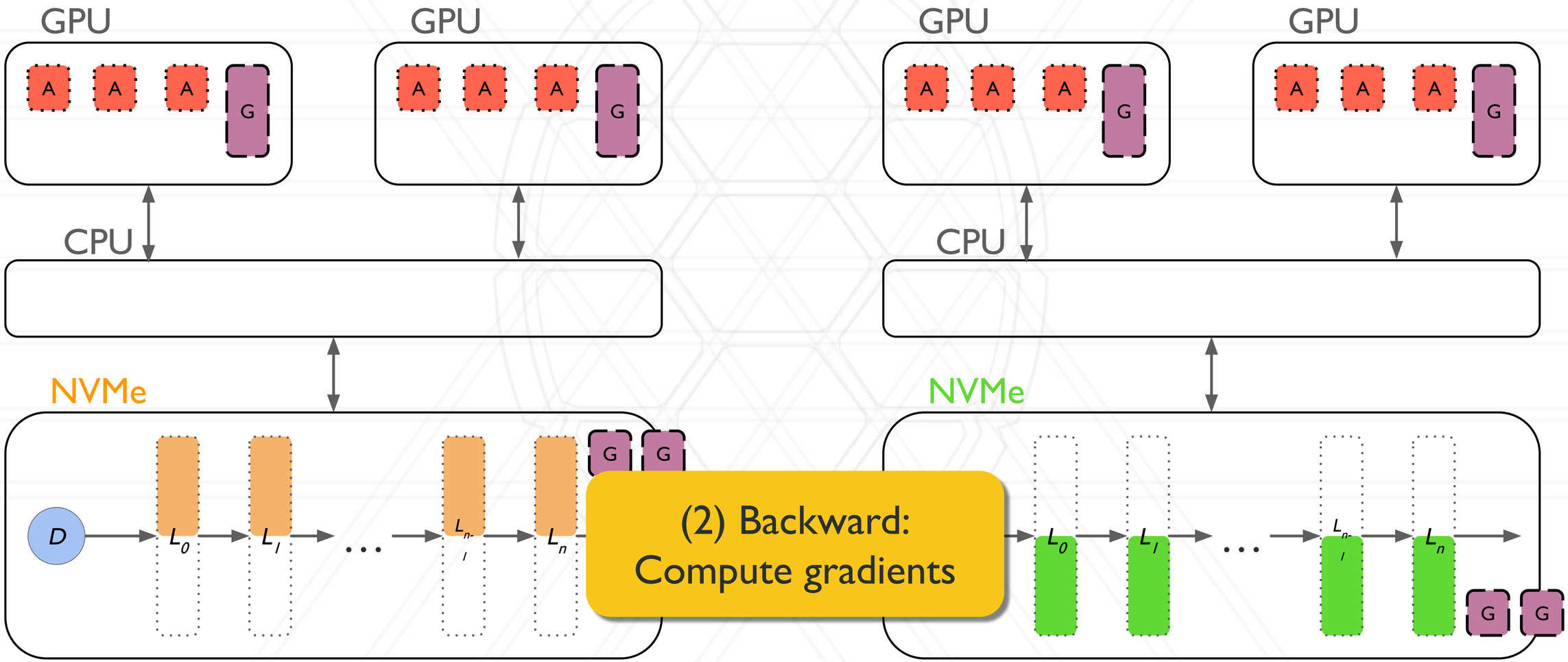
ZeRO Infinity Algorithm



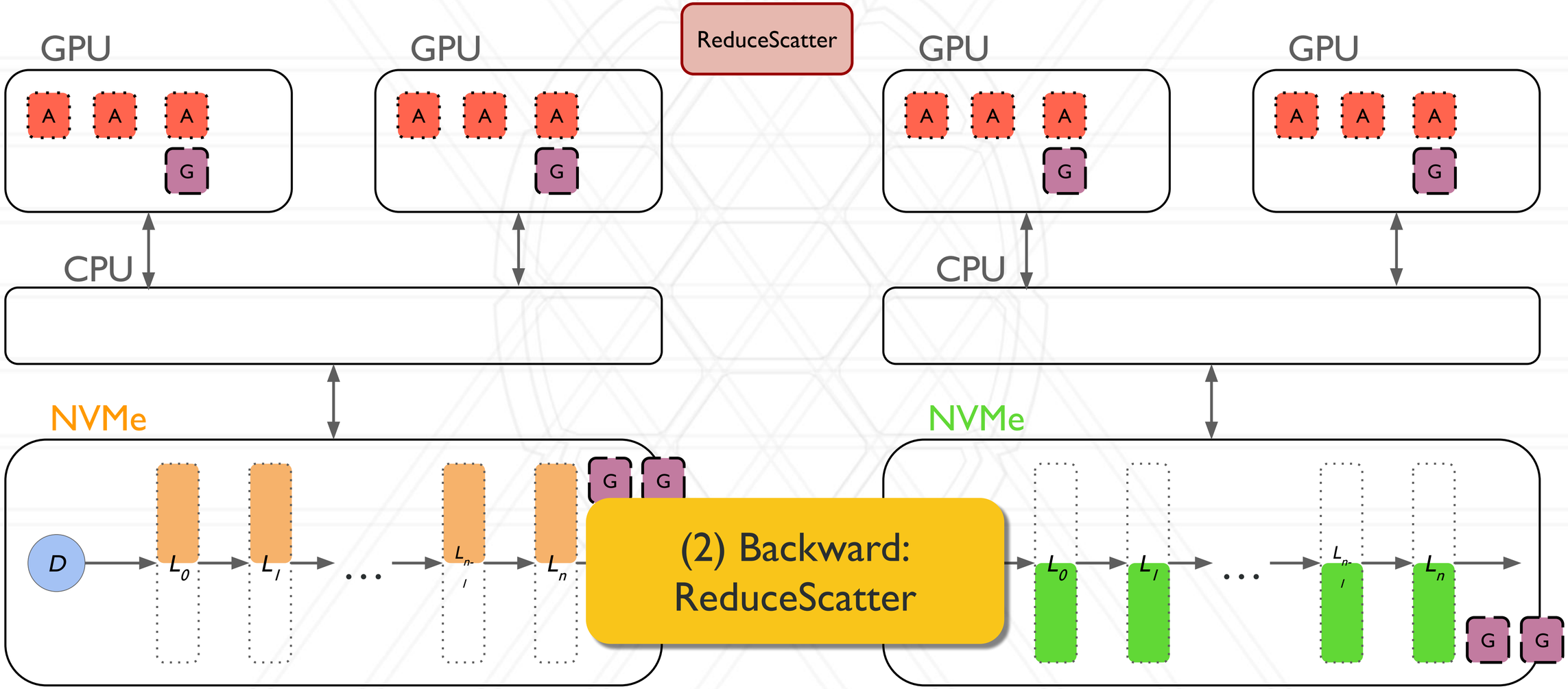
ZeRO Infinity Algorithm



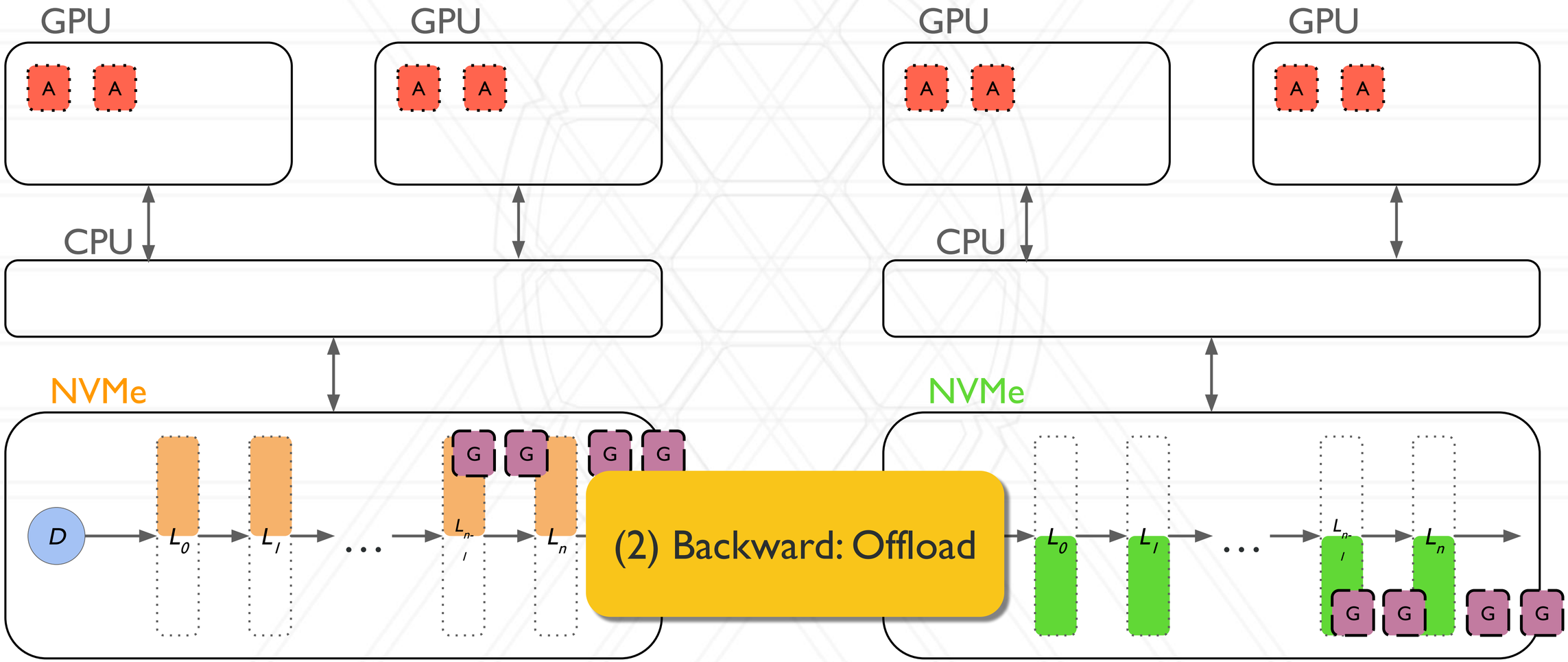
ZeRO Infinity Algorithm



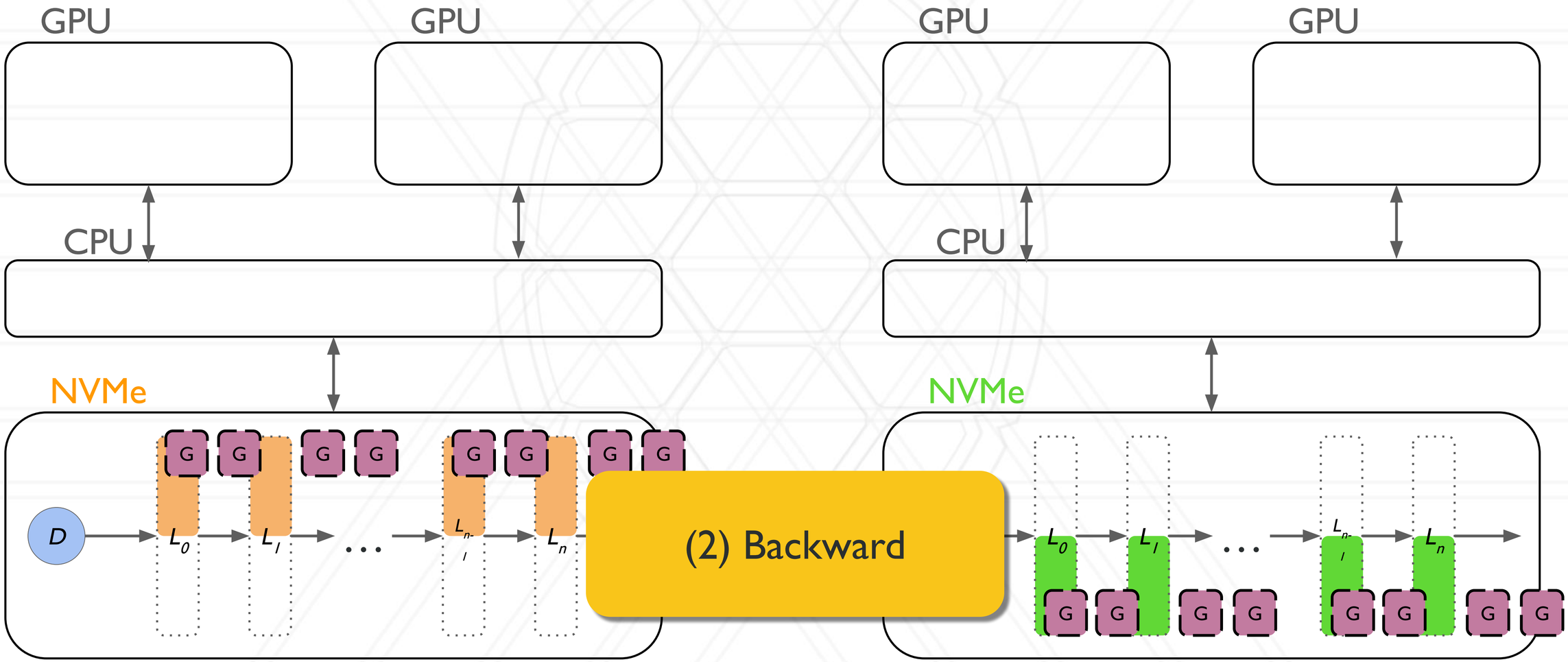
ZeRO Infinity Algorithm



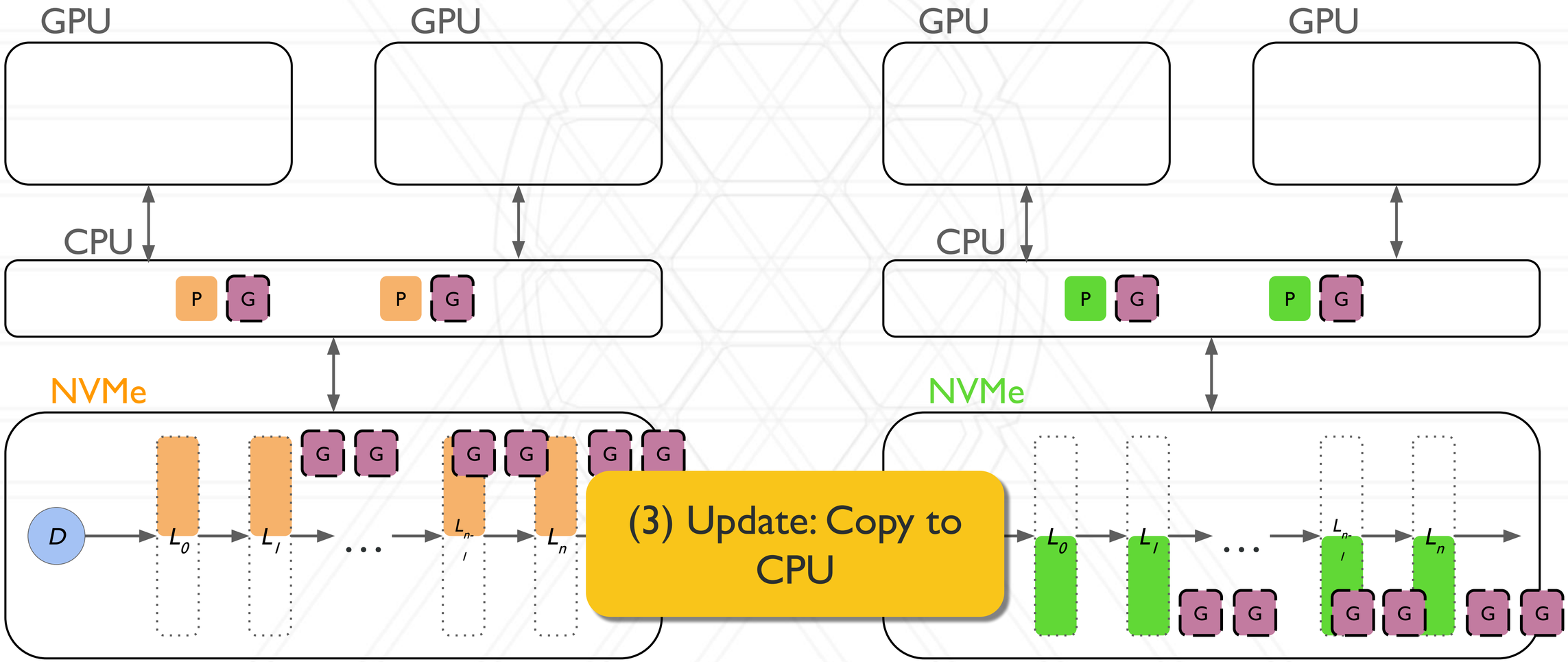
ZeRO Infinity Algorithm



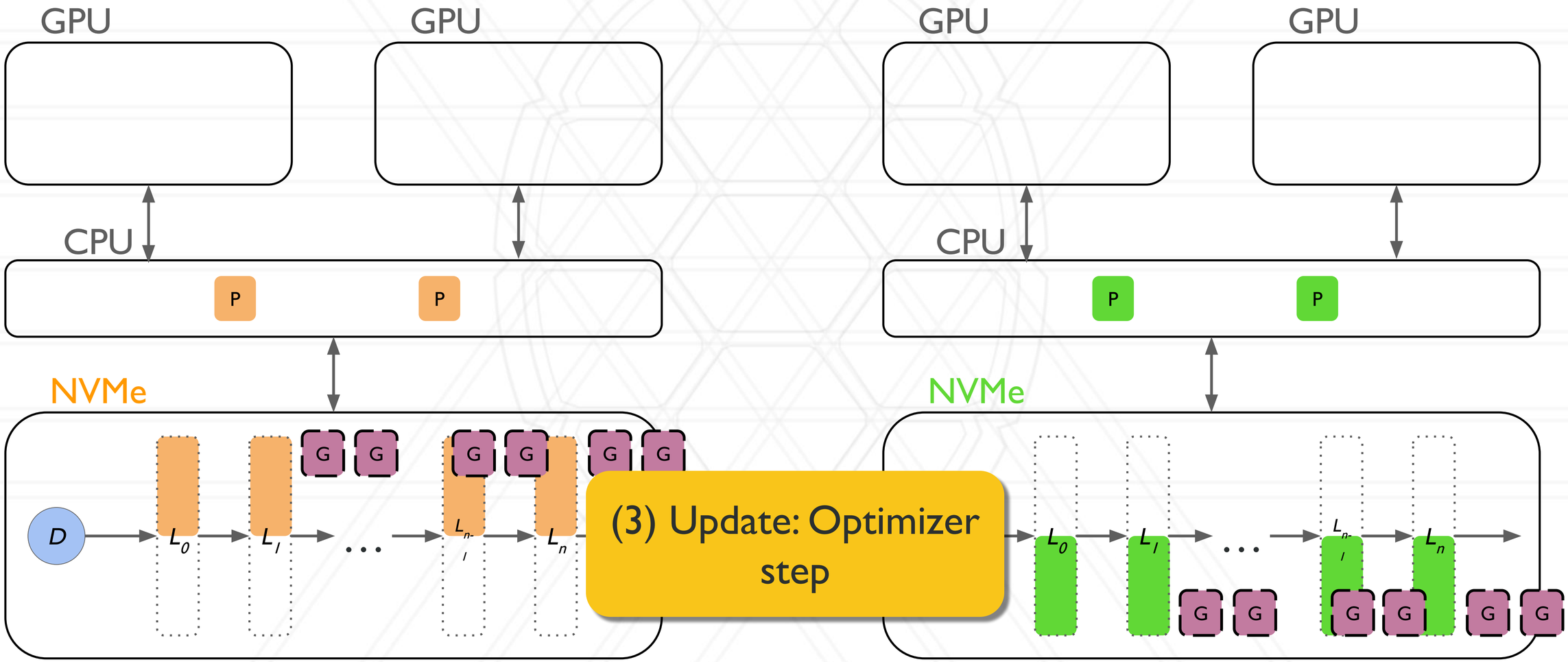
ZeRO Infinity Algorithm



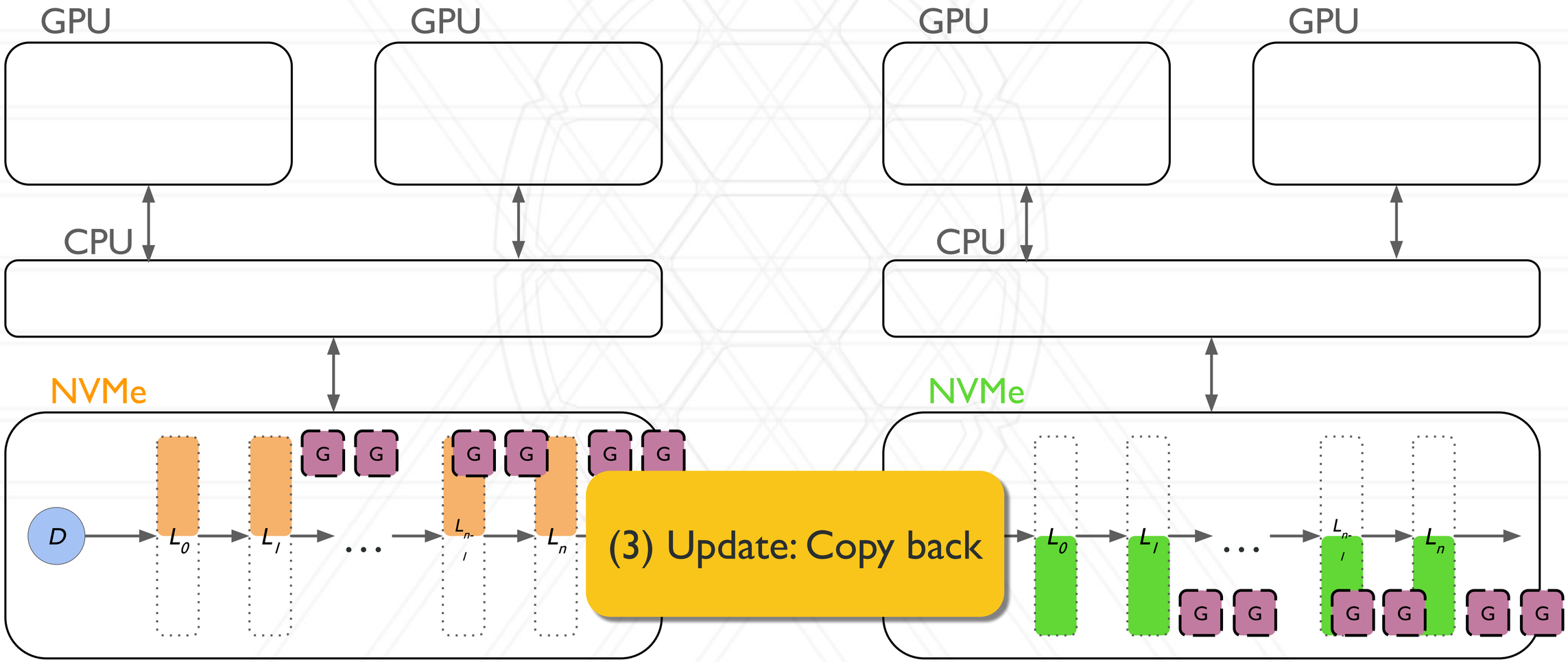
ZeRO Infinity Algorithm



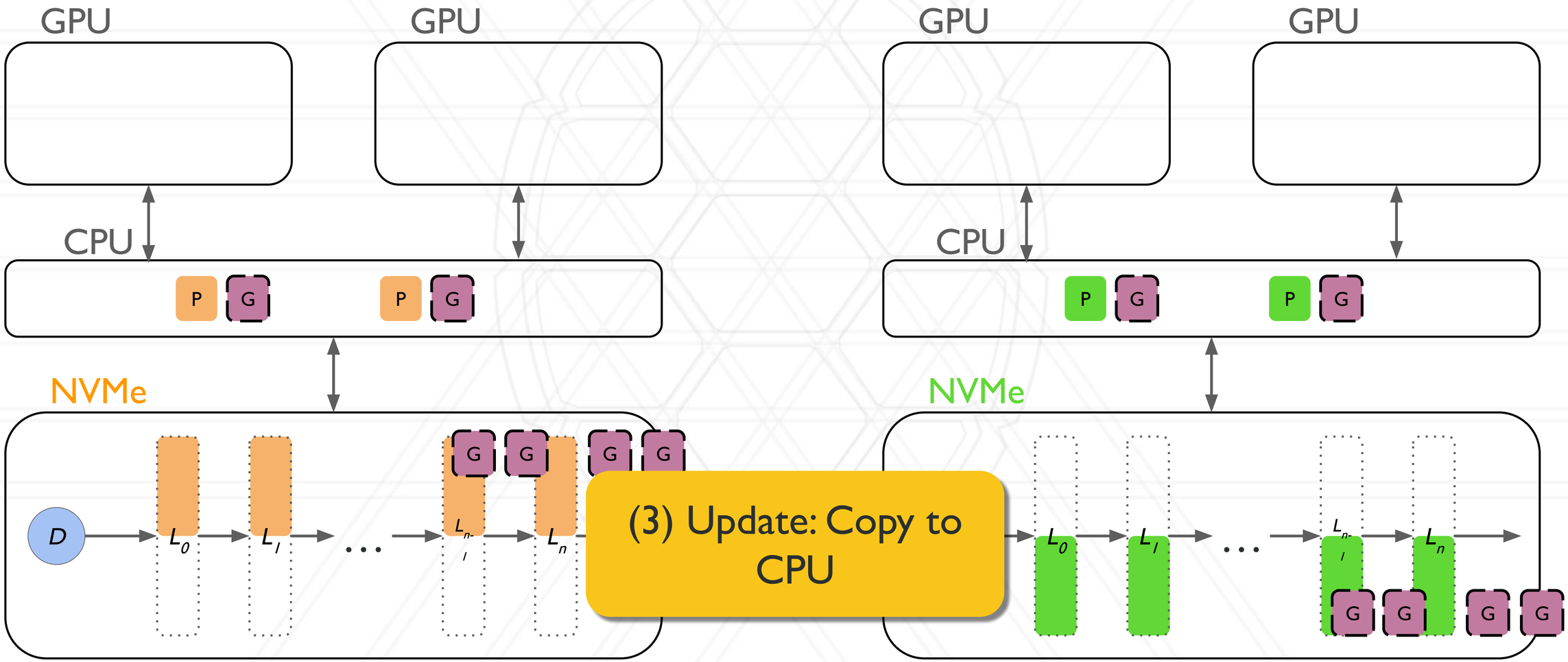
ZeRO Infinity Algorithm



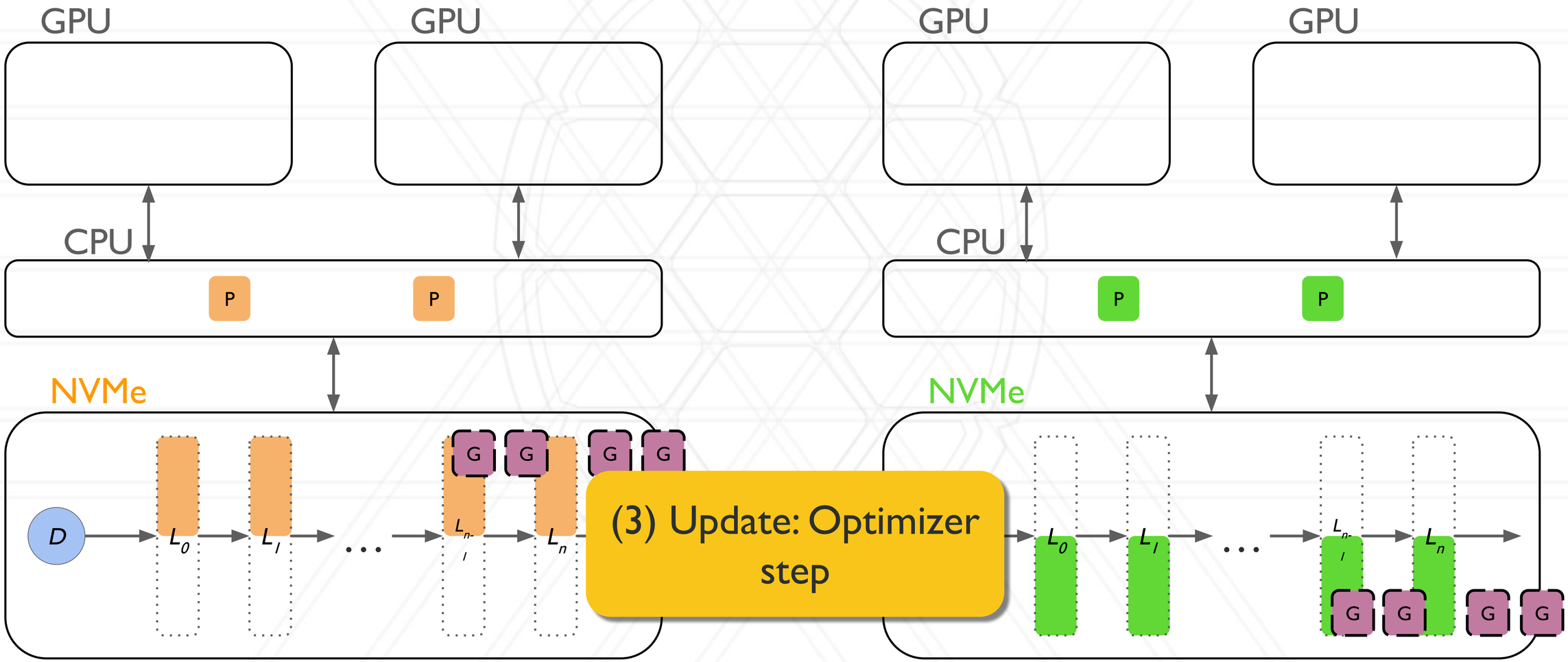
ZeRO Infinity Algorithm



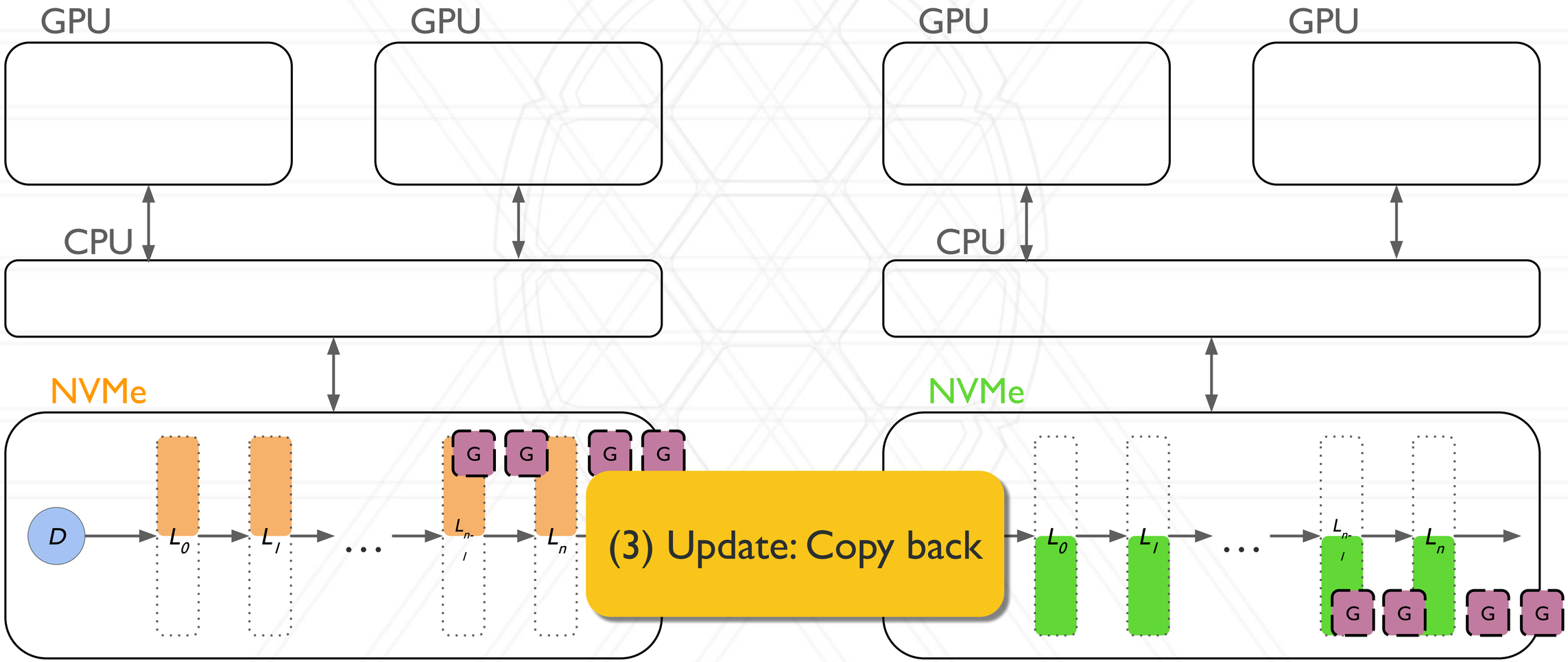
ZeRO Infinity Algorithm



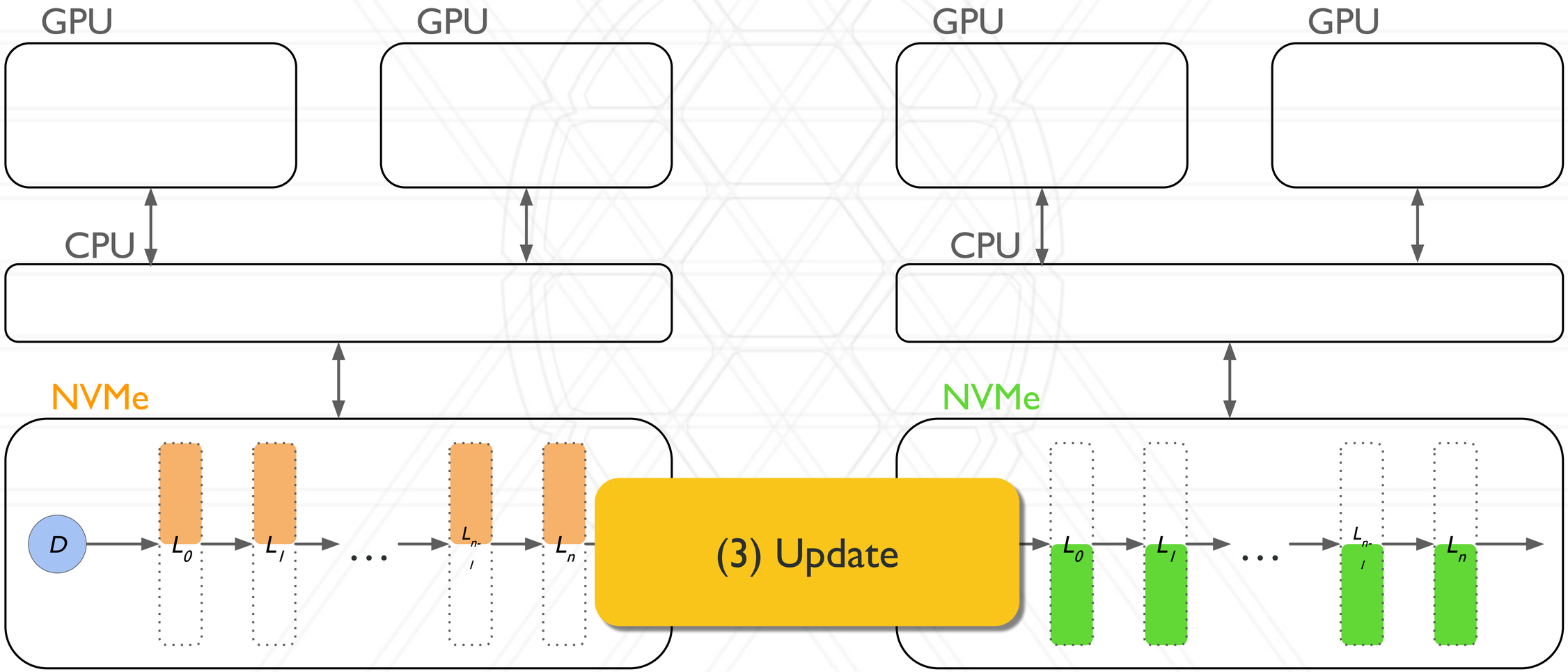
ZeRO Infinity Algorithm



ZeRO Infinity Algorithm



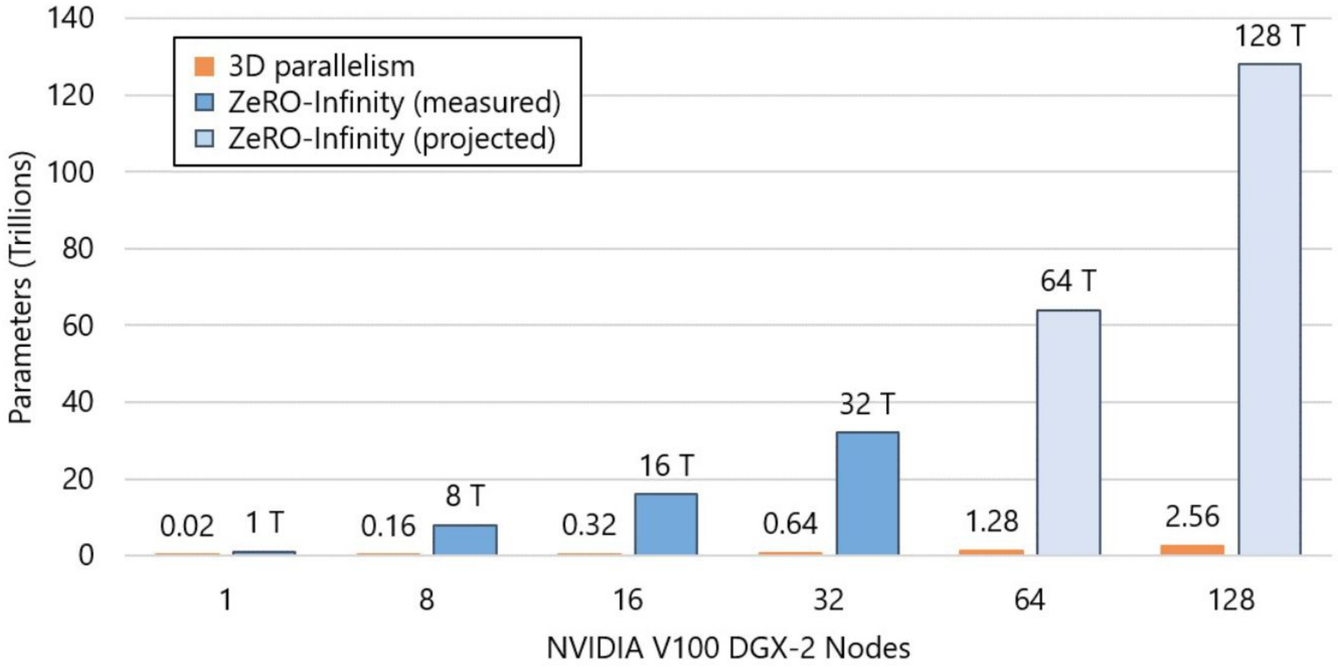
ZeRO Infinity Algorithm



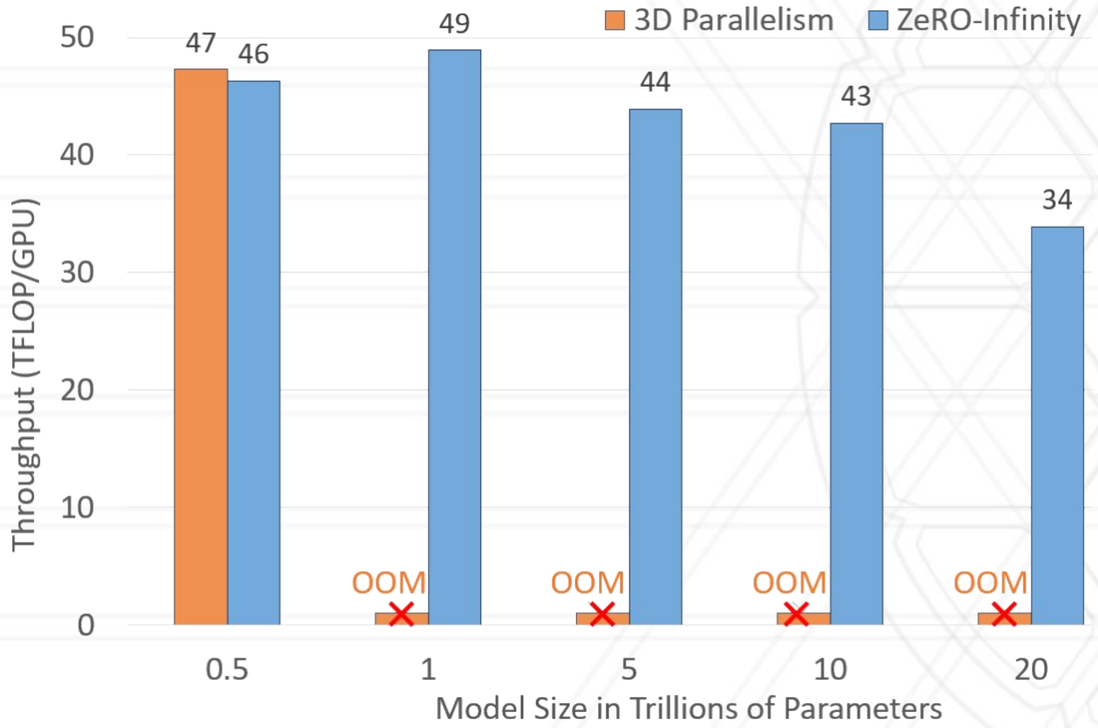
ZeRO-Infinity Optimizations

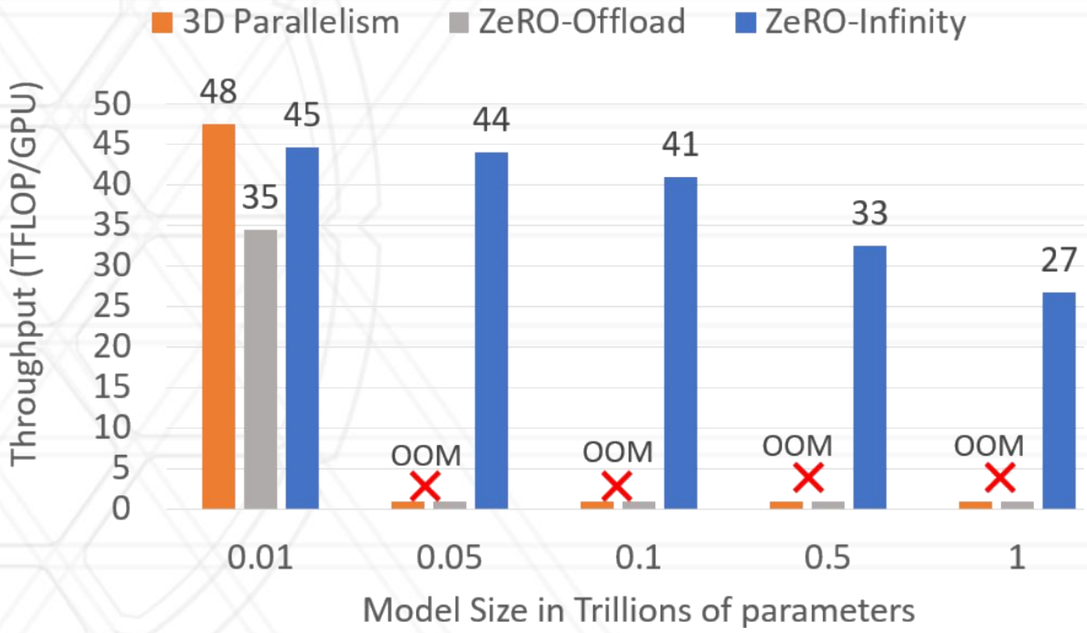
- Overlapped Copying/Communication
  - All data copying is overlapped
  - Next layer data is loading while computing
- Infinity Offload Engine
  - Pinned memory management
  - NVMe memory optimizations

Does it work?


3D parallelism
ZeRO-Infinity (measured)
ZeRO-Infinity (projected)
Parameters (Trillions)
140
120
100
80
60
40
20
0
0.02
1 T
0.16
8 T
0.32
16 T
0.64
32 T
1.28
64 T
2.56
128 T
1
8
16
32
64
128
NVIDIA V100 DGX-2 Nodes

Performance





When to use offloading?

- You need functionality, but do not have access to more hardware
- Approximations/quantizations do not give enough fidelity
- Expensive reads/writes can be completely masked

UNIVERSITY OF
MARYLAND