分布式训练系列

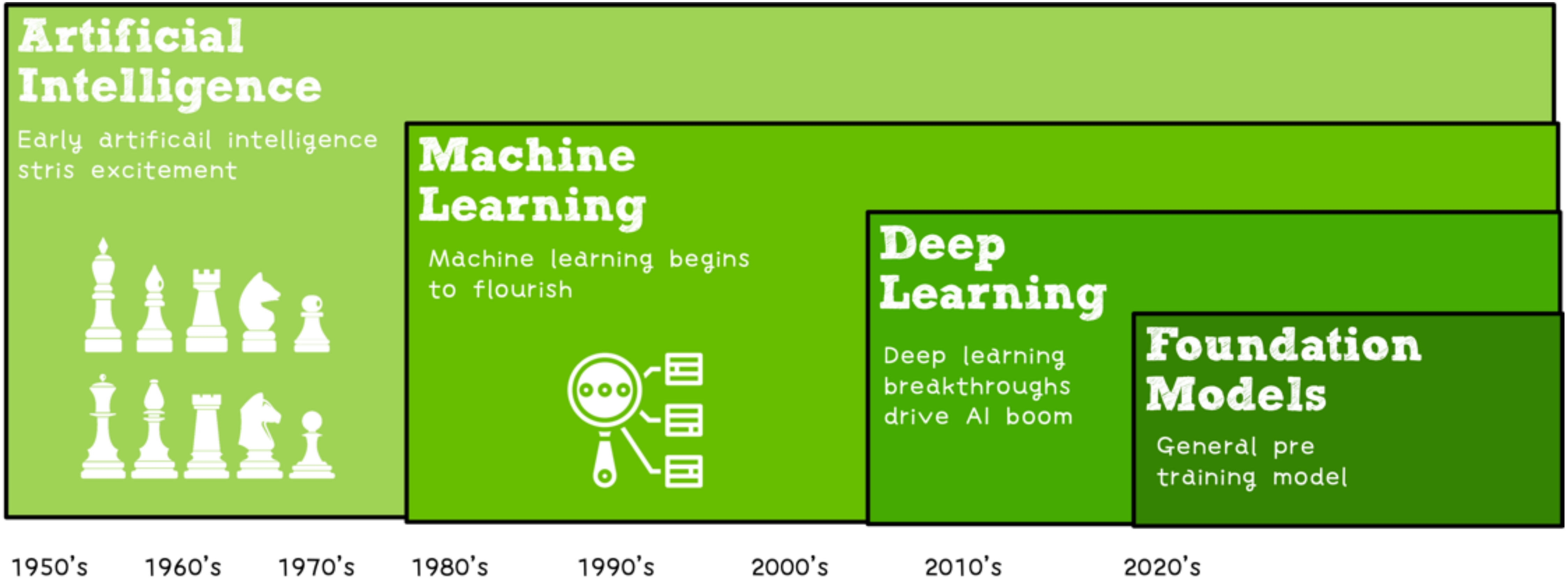
数据并行

ZOMI

BUILDING A BETTER CONNECTED WORLD

Ascend & MindSpore

www.hiascend.com
www.mindspore.cn


Artificial Intelligence
Early artificail intelligence stris excitement
Machine Learning
Machine learning begins to flourish
Deep Learning
Deep learning breakthroughs drive AI boom
Foundation Models
General pre training model
1950's
1960's
1970's
1980's
1990's
2000's
2010's
2020's

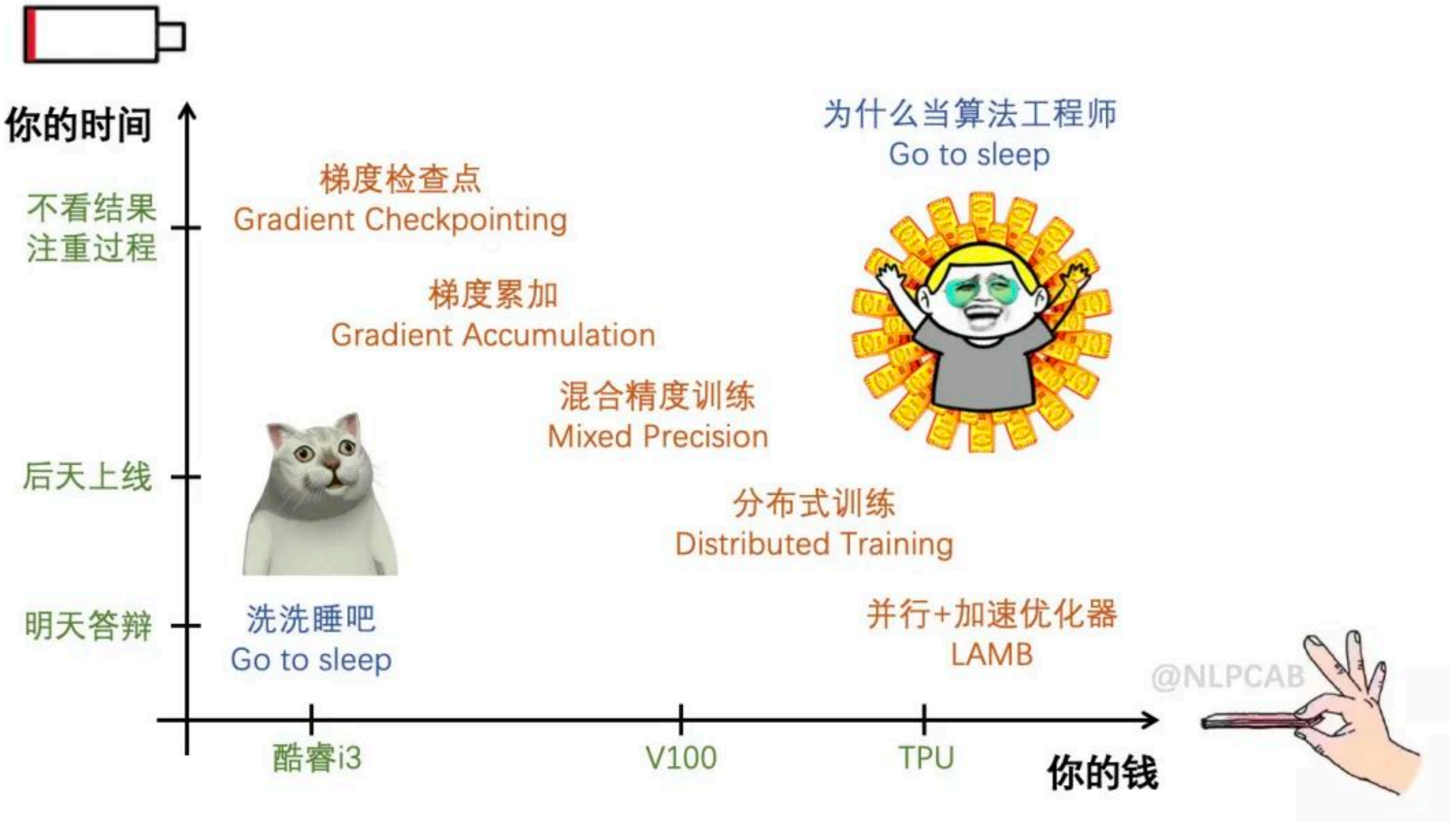
关于本内容

1. 内容背景

- 大规模分布式训练系统：串行到并行 – 并行处理体系 – 深度学习并行训练

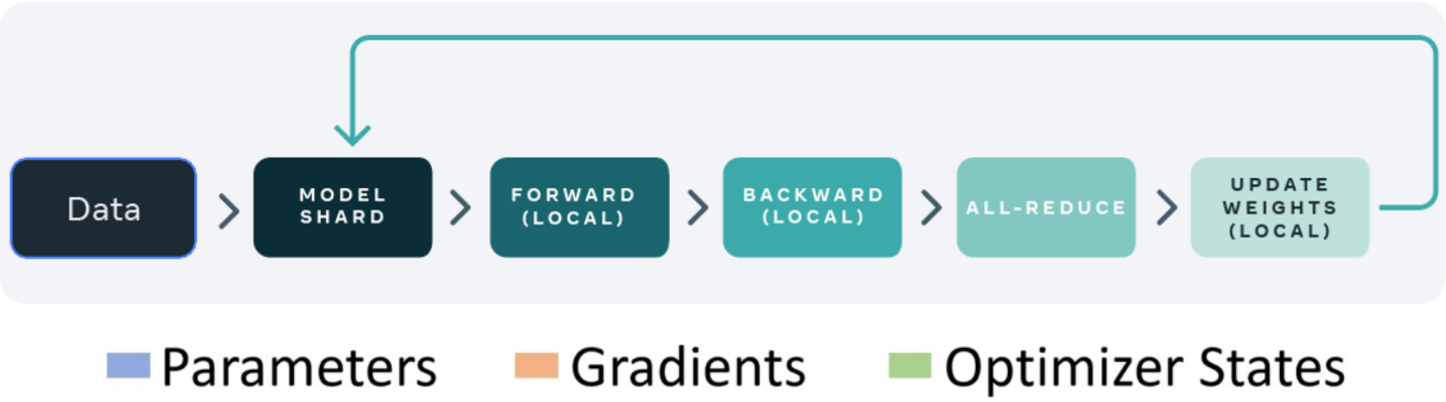
2. 具体内容

- **大模型训练的挑战**：内存墙 – 性能墙 – 效率墙 – 调优墙
- **分布式训练系统**：并行处理硬件架构 – 业界分布式系统分析
- **分布式并行总体架构**：参数服务器模式 – 集合通讯模式
- **通信原语与协调**：通讯协调软硬件 - 通信实现方式 - 通信原语
- **大模型算法结构**：大模型算法发展 – NLP大模型 - CV大模型 – 多模态大模型
- **分布式并行**：数据并行 – 模型并行 – 流水并行 – 混合并行
- **内存和计算优化**：ZeRO内存优化 – 混合精度与计算优化


你的时间
不看结果
注重过程
后天上线
明天答辩
梯度检查点
Gradient Checkpointing
梯度累加
Gradient Accumulation
混合精度训练
Mixed Precision
分布式训练
Distributed Training
并行+加速优化器
LAMB
为什么当算法工程师
Go to sleep
洗洗睡吧
Go to sleep
@NLPCAB
酷睿i3
V100
TPU
你的钱

Data parallelism

1. Data parallelism, DP
2. Distribution Data Parallel, DDP
3. Fully Sharded Data Parallel, FSDP

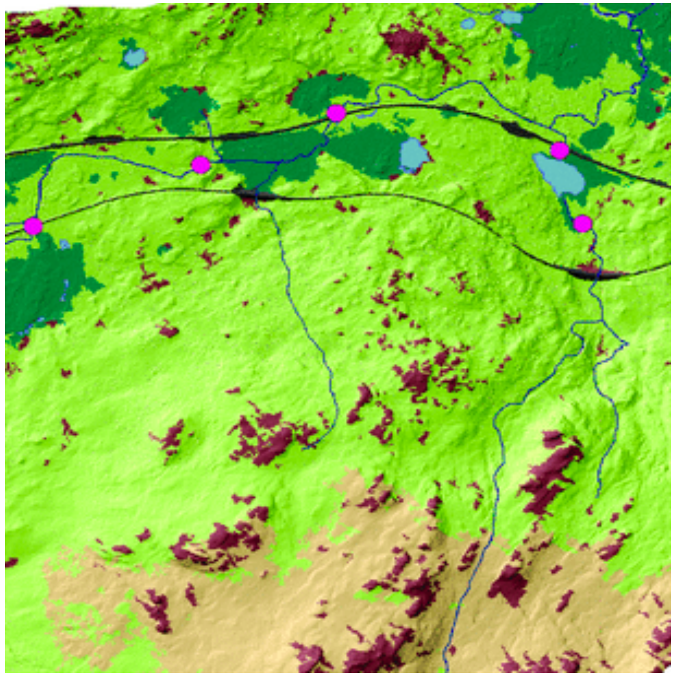
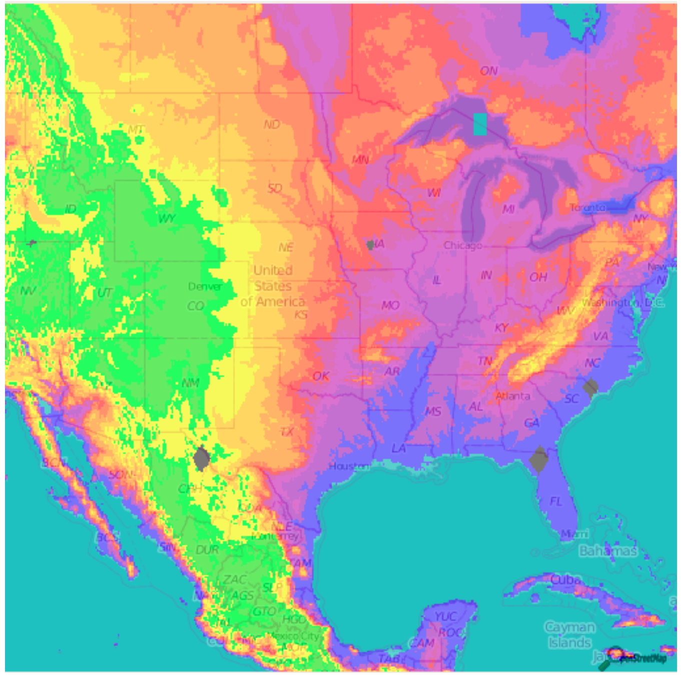


Data parallelism

1. Data parallelism, DP
2. Distribution Data Parallel, DDP
3. Fully Sharded Data Parallel, FSDP

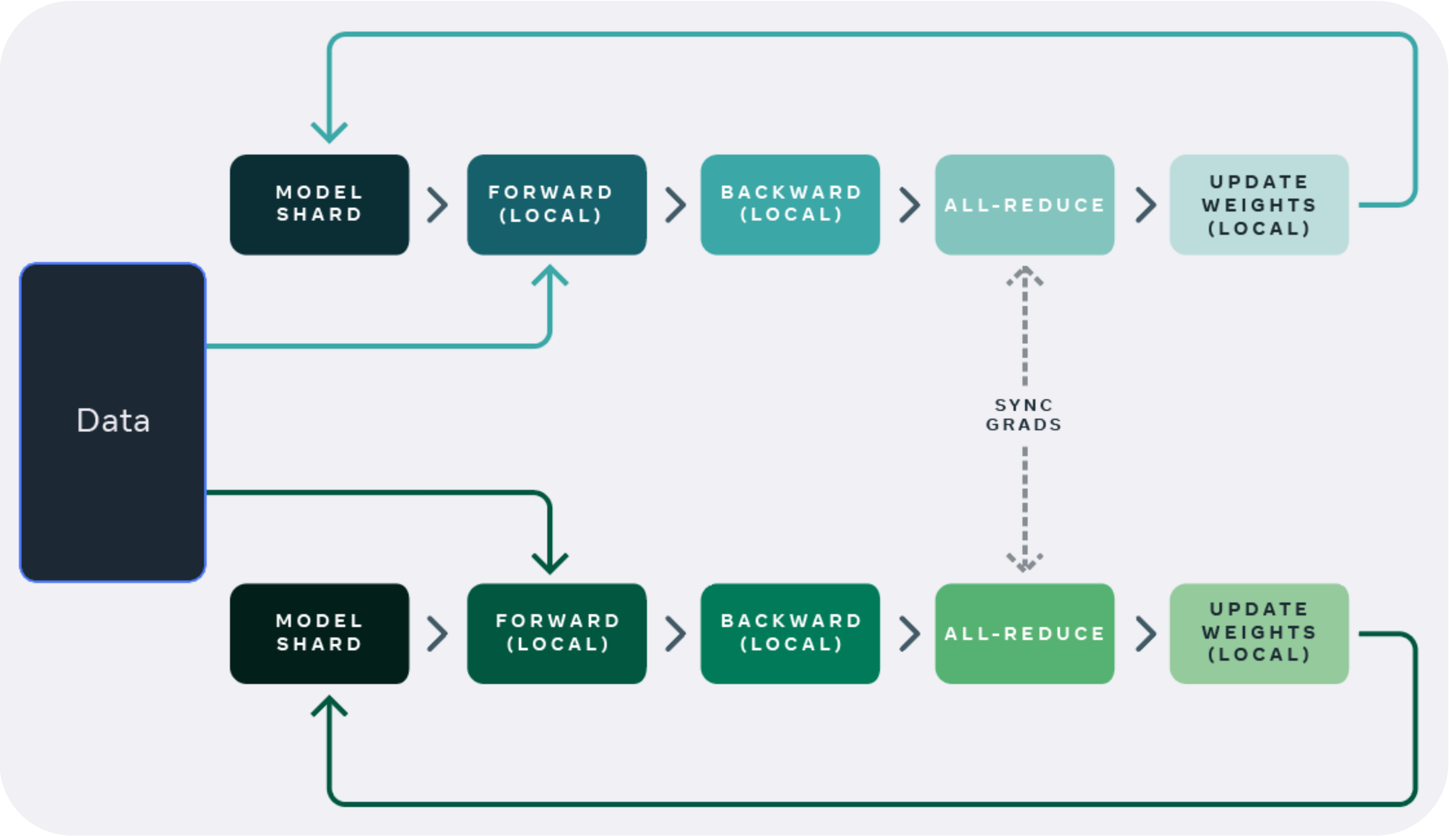
256 x 256 x 3

24599 x 35688 x 256

Data parallelism, DP

- Data Parallel automatically splits training data and sends model jobs to multiple GPUs. After each model completed, Data Parallel will Accumulate Gradients.



Data parallelism Limitation(I)

Practice

- Simple implementation of code logic.
- Multithreading parallel Controlled by a process, restricted by GIL.

```
device = torch.device("cuda:1,3") ## specify the GPU id's

model = CreateModel()

model= nn.DataParallel(model,device_ids = [1, 3])
model.to(device)
```

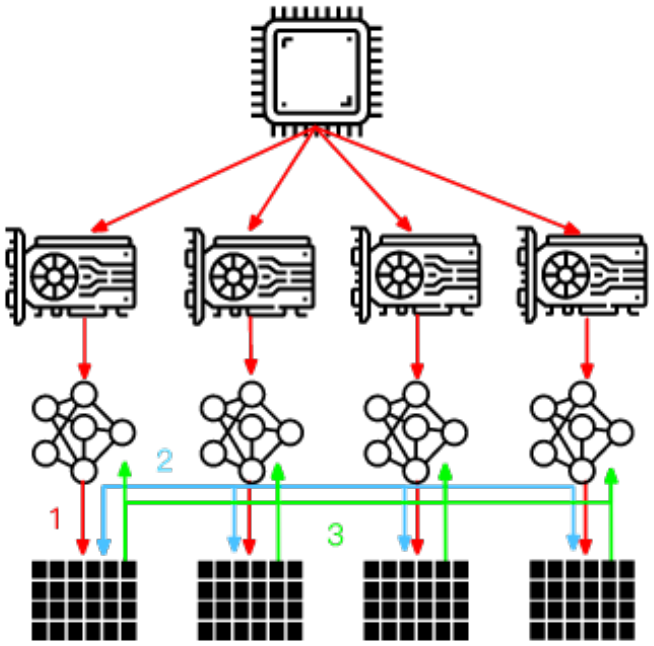
Data parallelism Limitation(II)

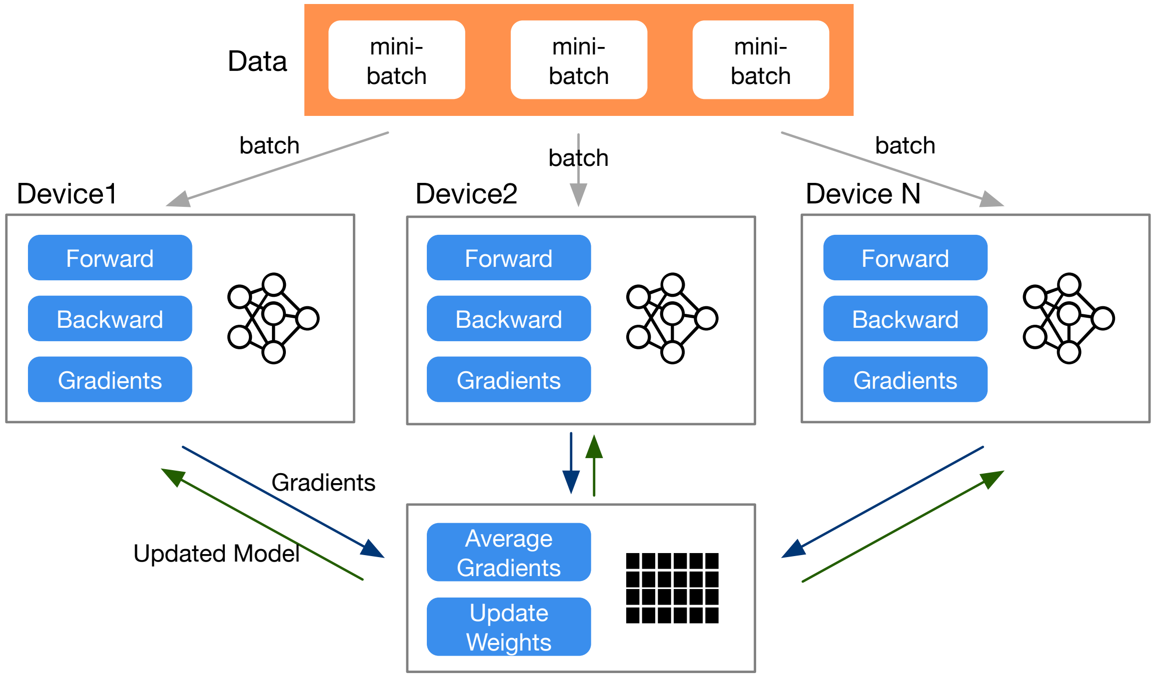
Theoretical

- Models of each machine are independent, clusters has using computing power.
- Each machine maintain self gradient, accumulate gradients using Collective communication.
  1. When Gradient Accumulation?
  2. How Gradient Accumulation efficient?

Gradient Accumulation

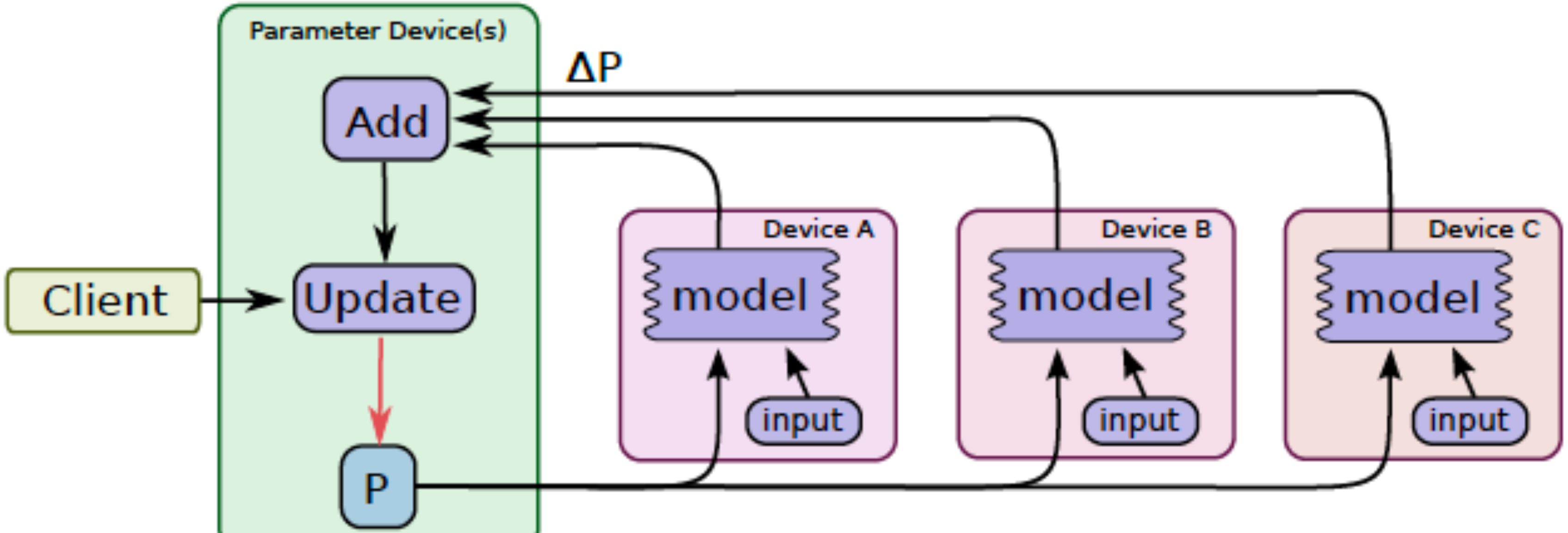
- Each machine maintain self gradient, accumulate gradients using Collective communication.
  1. When Gradient Accumulation?
  2. How Gradient Accumulation efficient?





Gradient Accumulation: Synchronize

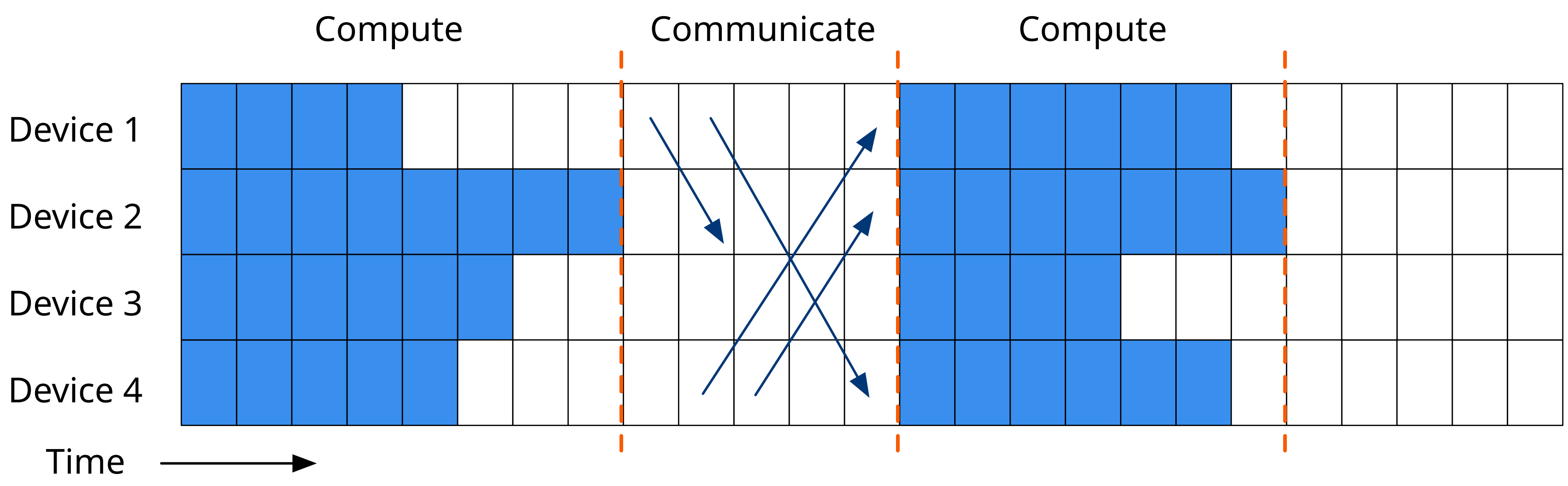
- Next round of local computing cannot continue until all working nodes have completed this communication.



Synchronous Data Parallelism

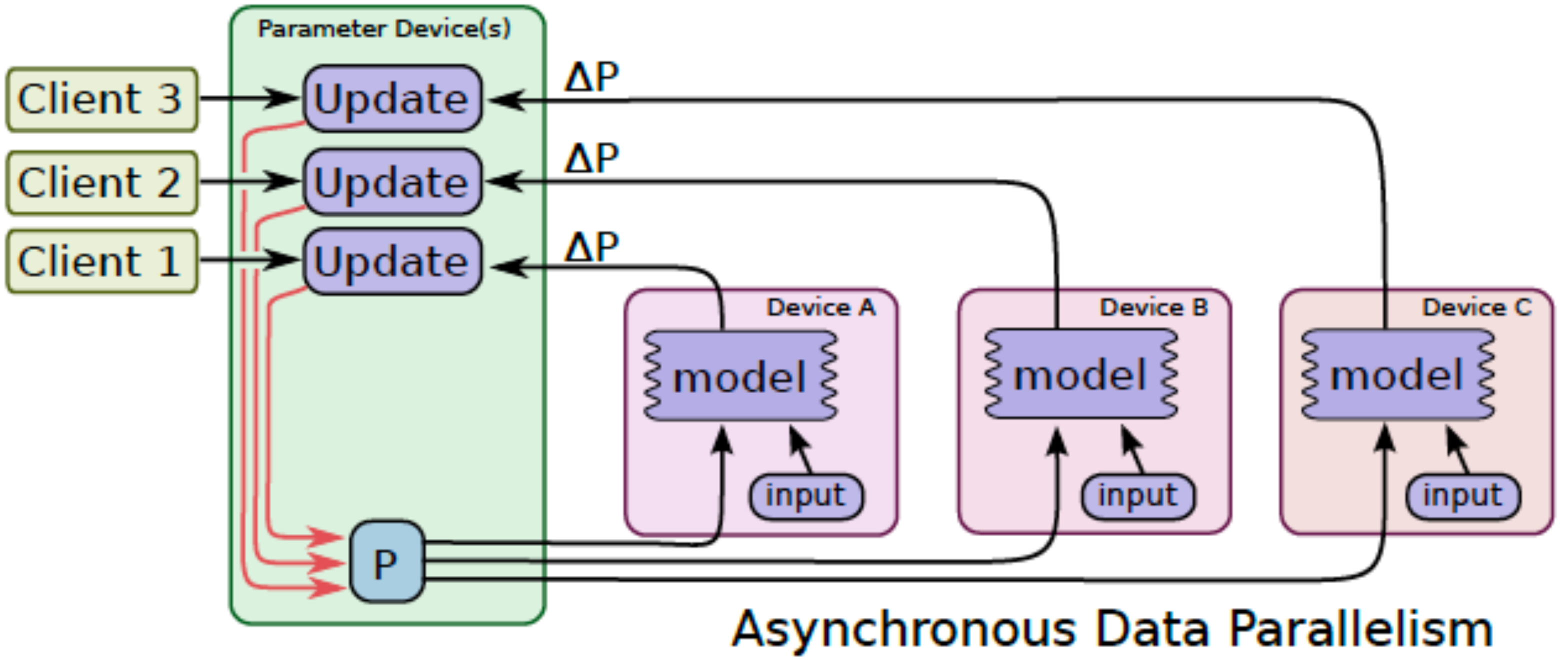
Gradient Accumulation: Synchronize

- **Advantages:** computing and communication are strictly synchronized, ensure parallel execution logic is the same as the serial execution logic
- **Disadvantages:** The working node of the local computing earlier needs to wait for other working nodes to process, which causes a waste of computing hardware.



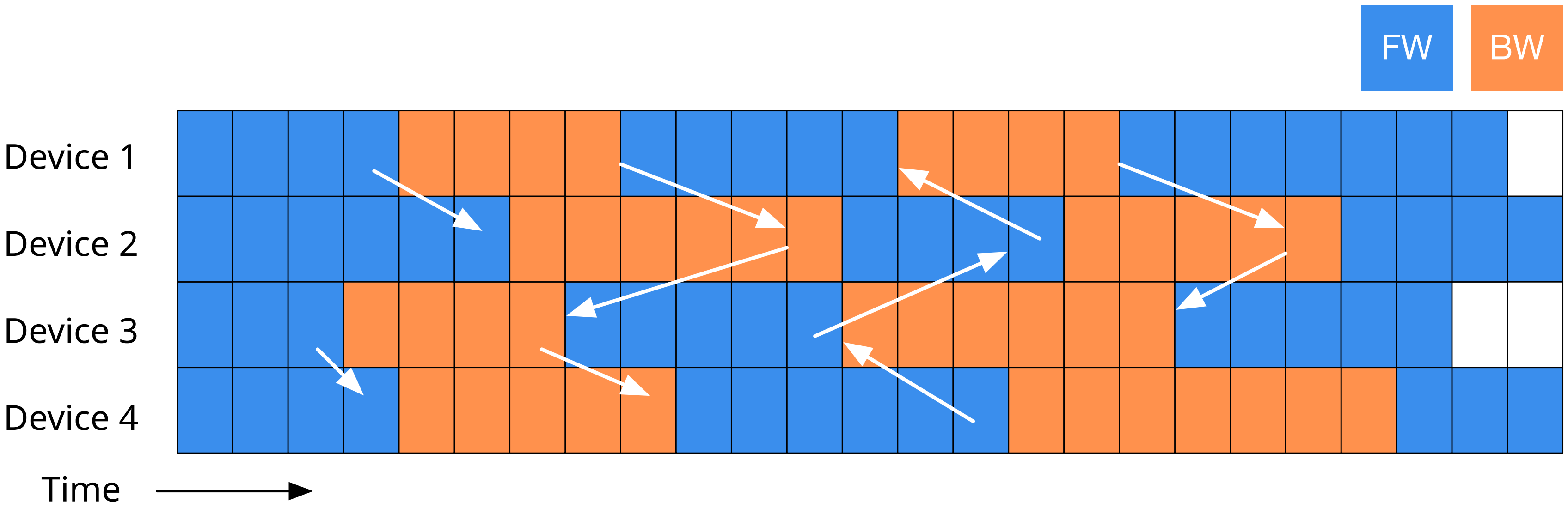
Gradient Accumulation: Asynchronous

- After the current batch iteration, communicate with other servers to transmit network model parameters



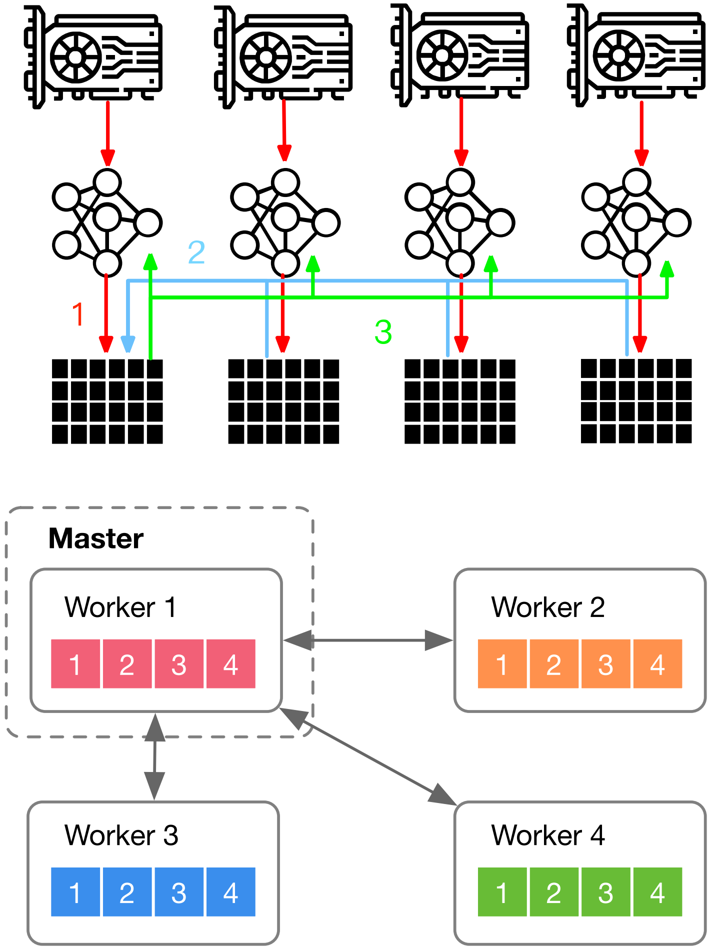
Gradient Accumulation: Asynchronous

- **Advantages:** High execution efficiency, no blocking and waiting between communication and execution except for single machine communication time
- **Disadvantages:** the network model training is not convergent, the training time is long, and the repeated use of model parameters leads to industrialization failure

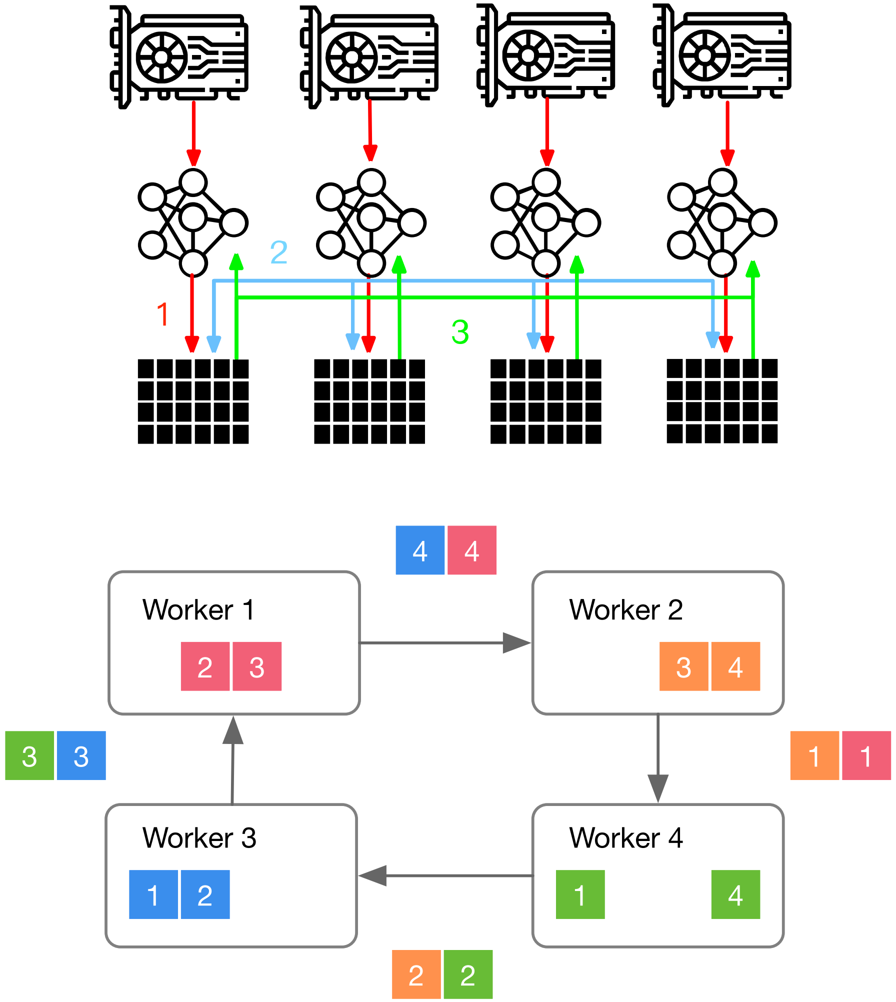


Gradient Accumulation Communication Method

GPU0 作为参数服务器



参数服务器分布在所有GPU



Distribution Data Parallel, DDP

- Adopted multi process parallelism, not restricted by GIL.
- Parameters of each process are not synchronized, but the change of parameters.
- Using Ring all Reduce improves the communication efficiency.

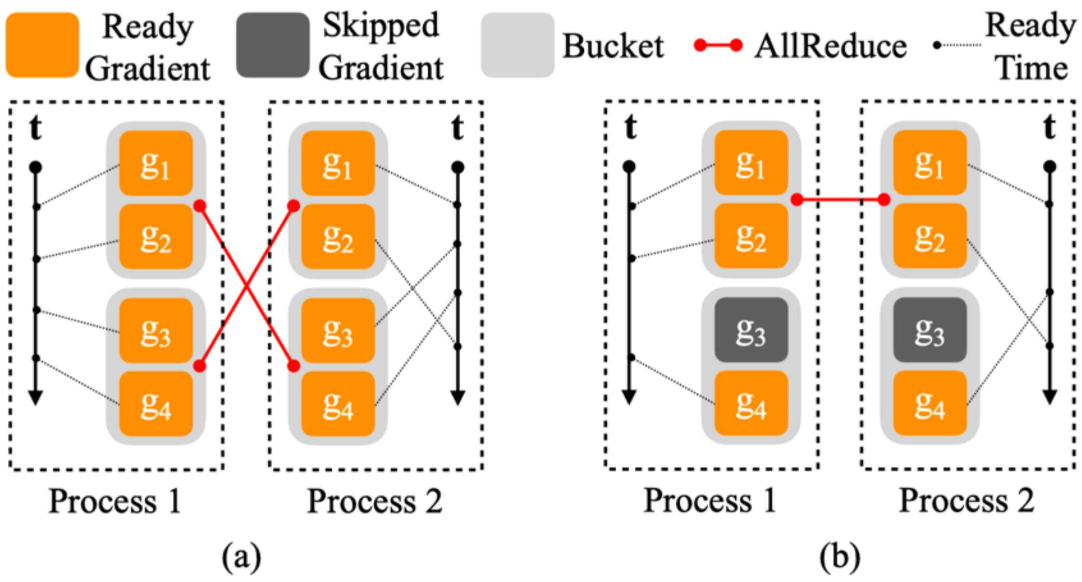


Figure 3: Gradient Synchronization Failures

Distribution Data Parallel, DDP

1. gradient bucket
2. keep reduce order
3. skip gradient
4. Collective communication

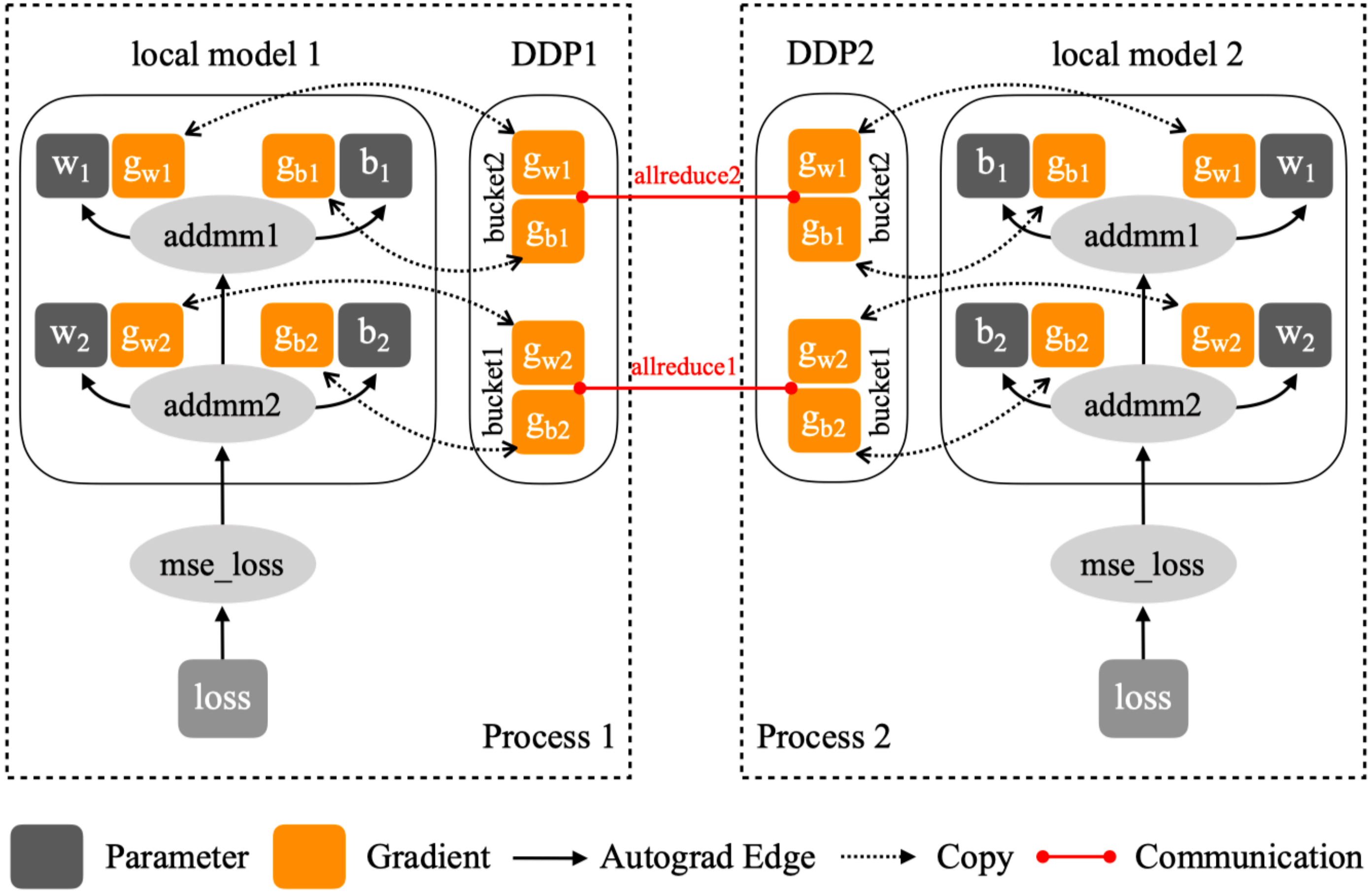
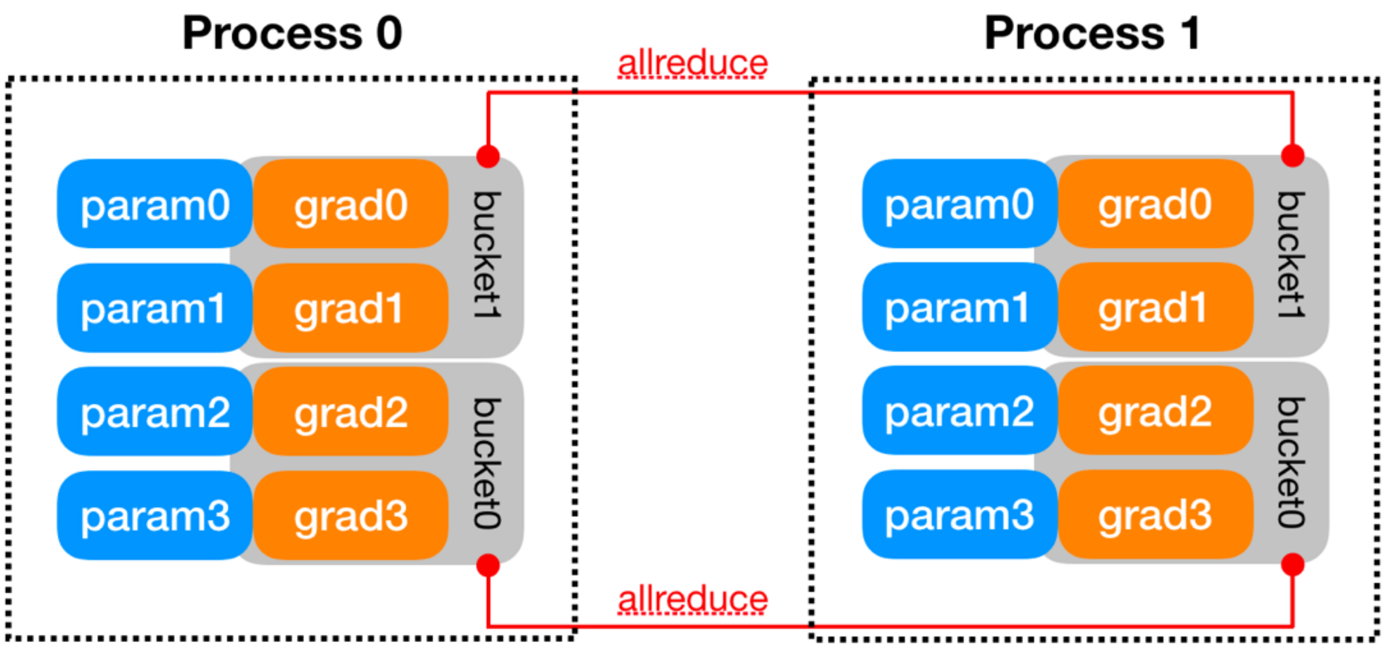


Figure 4: Distributed Gradient Reduction

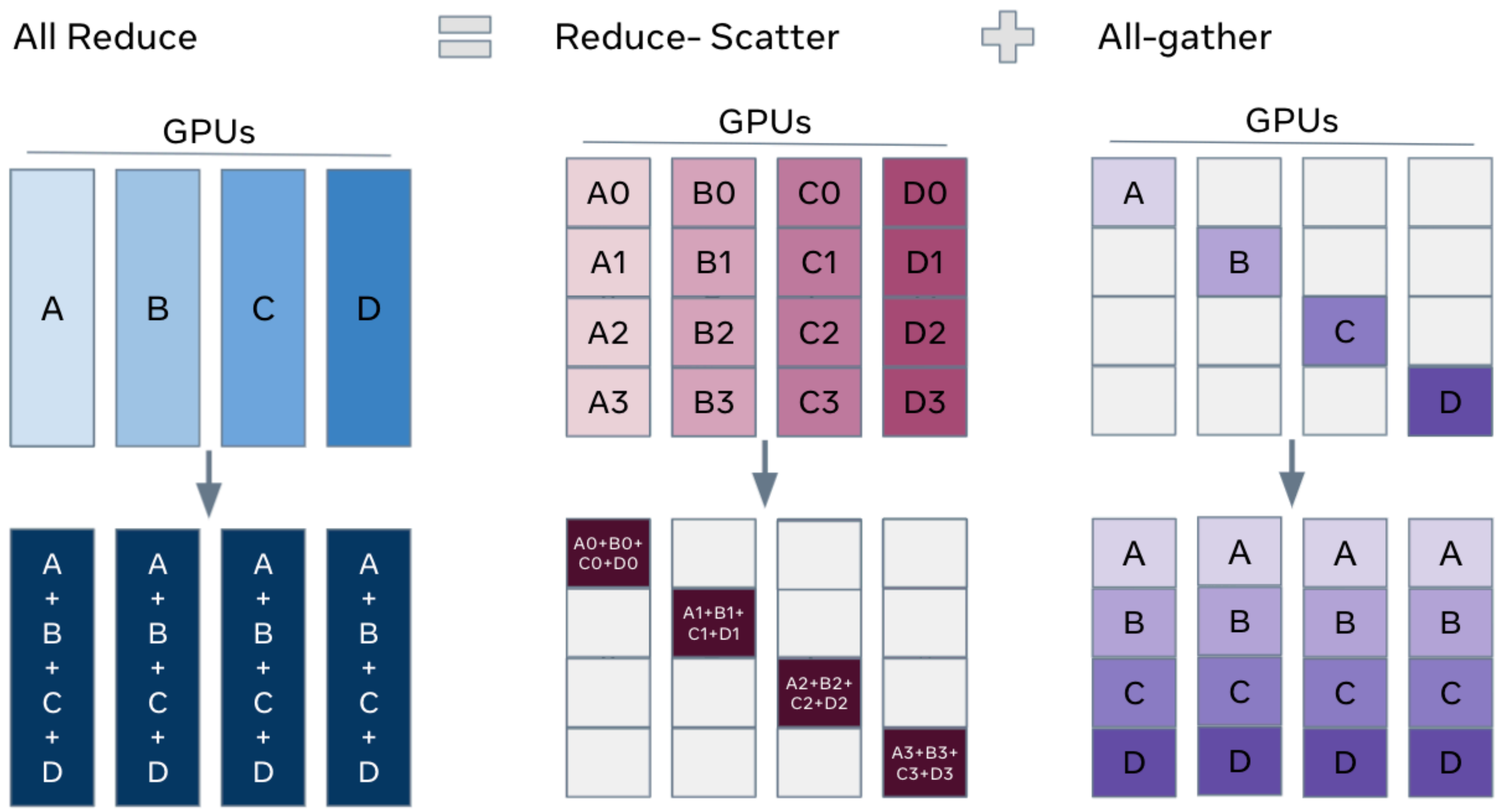
Distribution Data Parallel, DDP

1. gradient bucket
2. keep reduce order
3. skip gradient
4. Collective communication

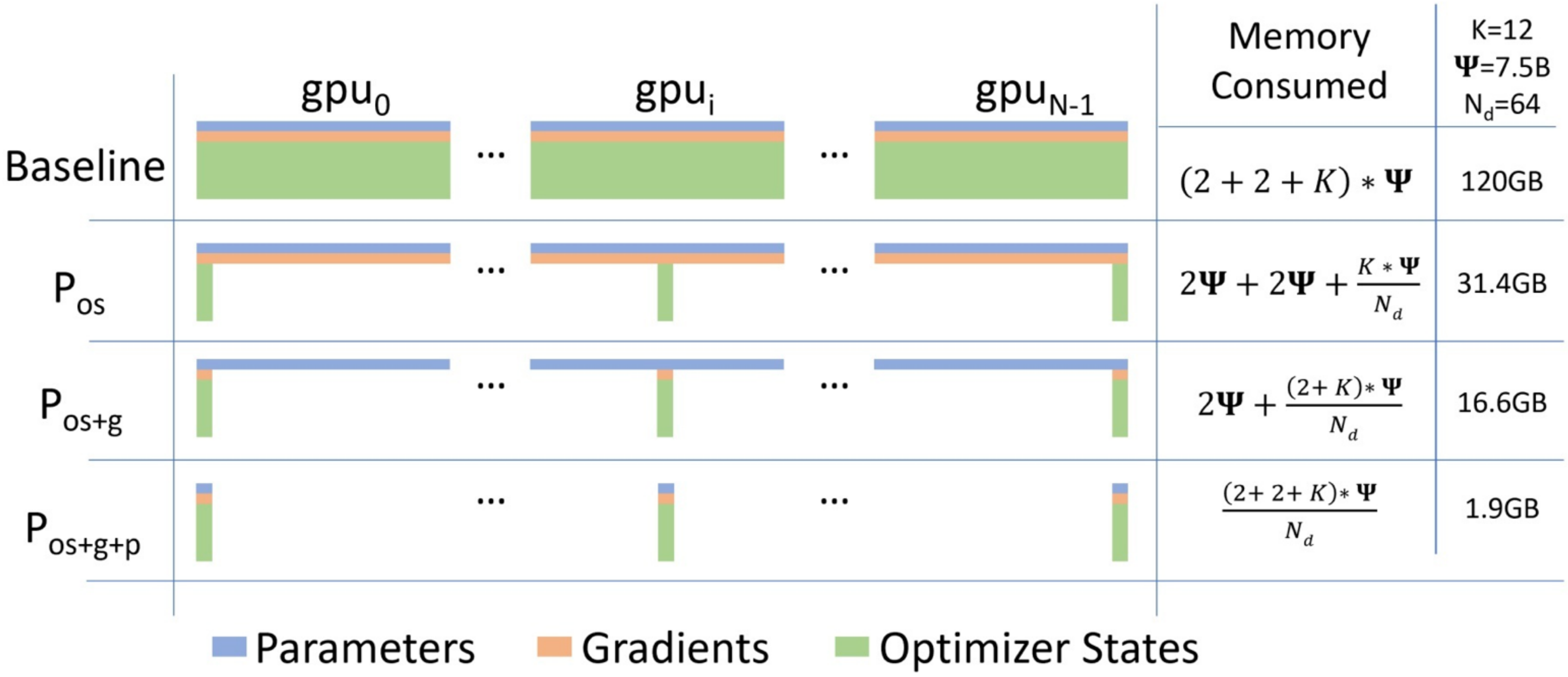


https://pytorch.org/docs/stable/notes/ddp.html

Collective communication

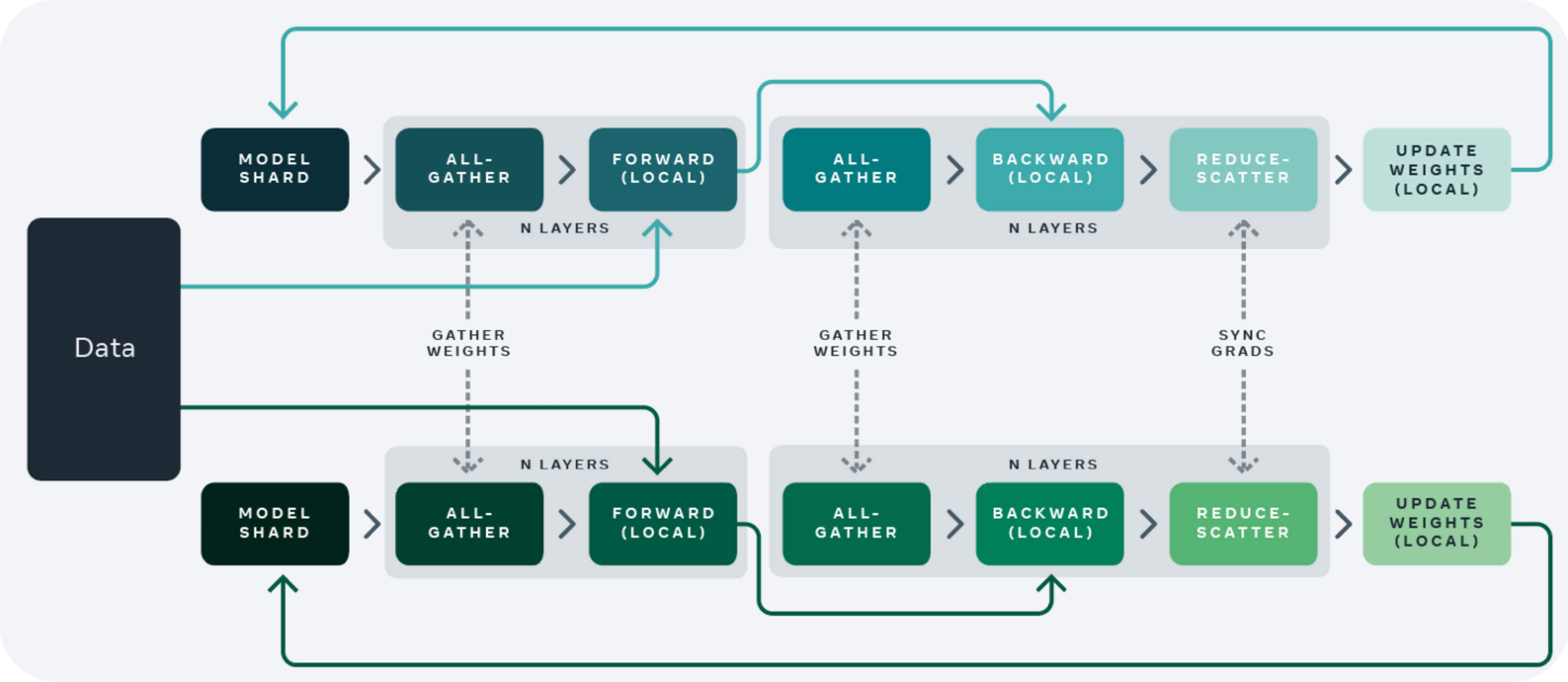


Model's States

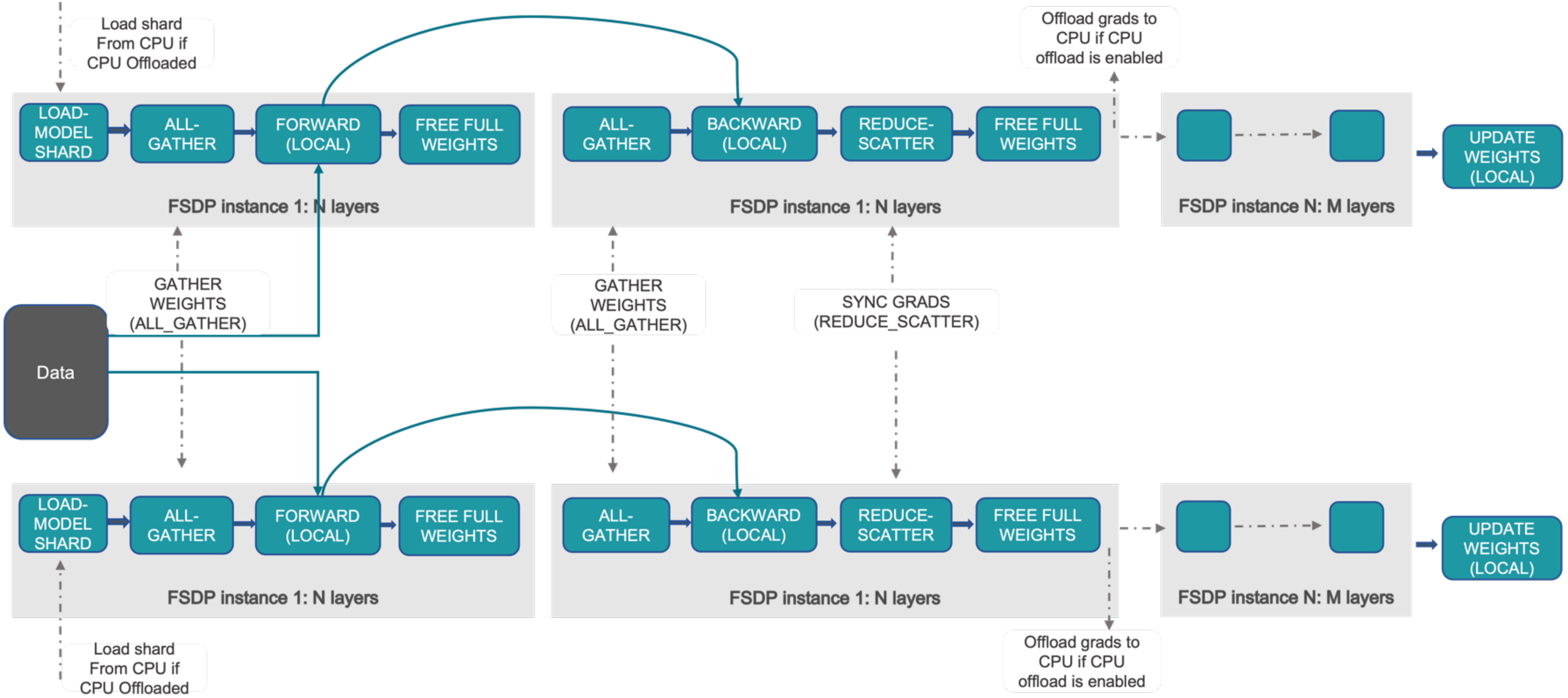


Fully Sharded Data Parallel, FSDP

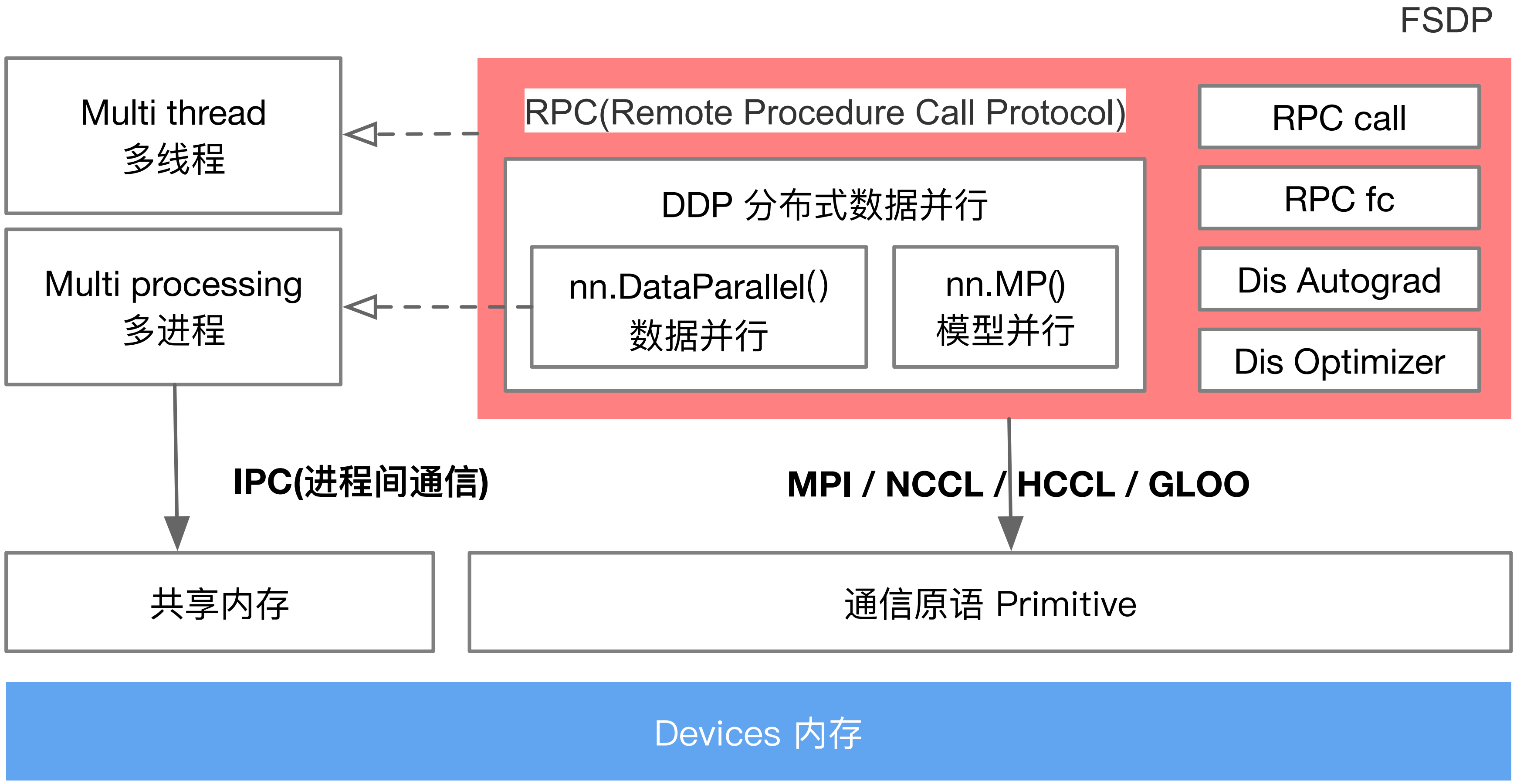
- FSDP shards all of model's parameters, gradients and optimizer states across data-parallel workers and can optionally offload the sharded model parameters to CPUs.



Fully Sharded Data Parallel, FSDP



PyTorch Implication



Inference

I. https://zhuanlan.zhihu.com/p/450854172 全网最全-超大模型+分布式训练架构和经典论文

II. https://pytorch.org/blog/introducing-pytorch-fully-sharded-data-parallel-api/

III. https://pytorch.org/tutorials/intermediate/FSDP_tutorial.html

IV. https://engineering.fb.com/2021/07/15/open-source/fsdp/

V. Li, Shen, et al. "Pytorch distributed: Experiences on accelerating data parallel training." arXiv preprint arXiv:2006.15704 (2020).

VI. Xu, Yuanzhong, et al. "Automatic cross-replica sharding of weight update in data-parallel training." arXiv preprint arXiv:2004.13336 (2020).

BUILDING A BETTER CONNECTED WORLD

THANK YOU