Data Parallel Training for Dense and MoE Models

Runyu Lu, Ruofan Wu, Jeff Ma

Agenda

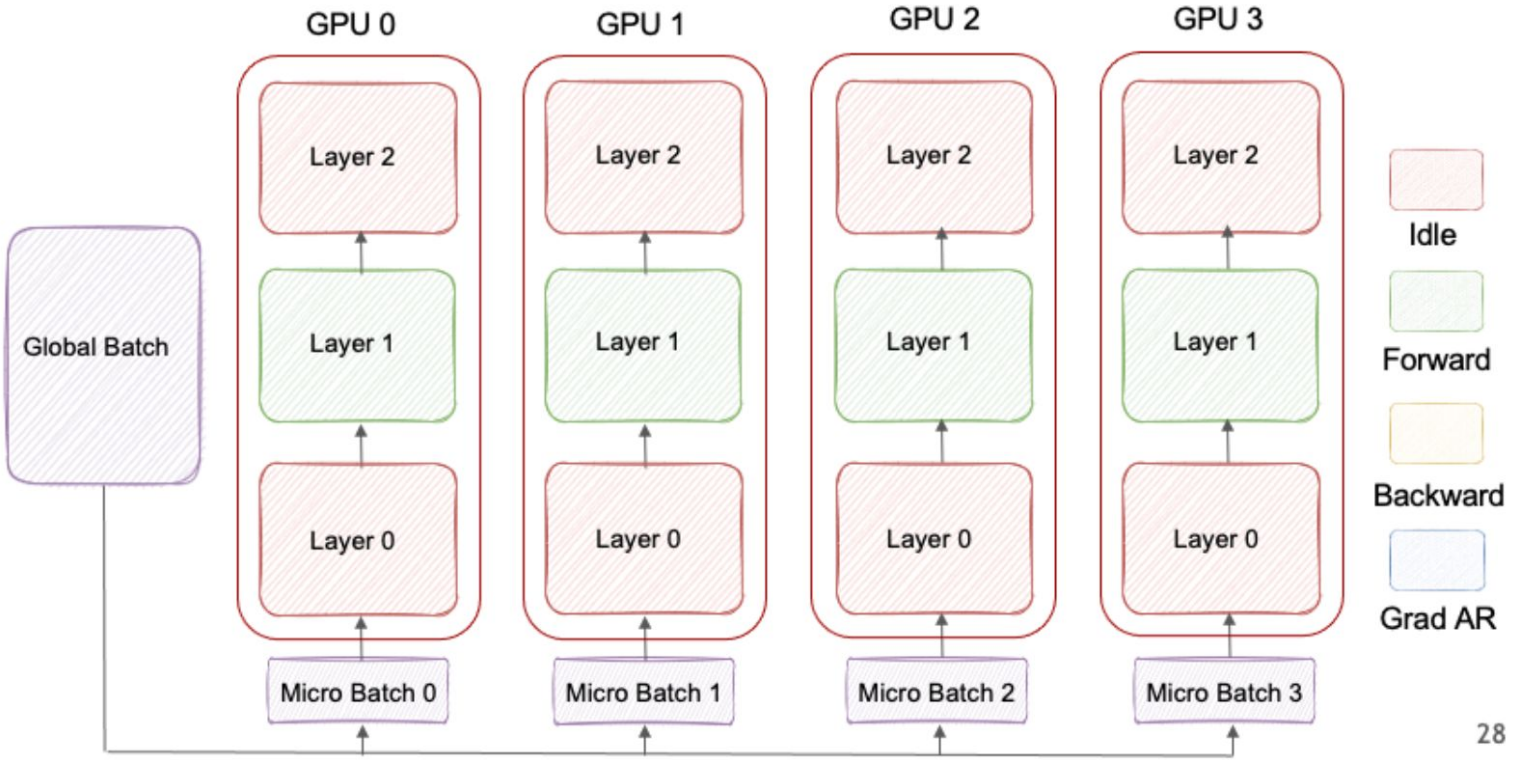
1. FSDP
2. TUTEL

PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel

Itinerary

1. Background
2. Data Parallel and its limitation
3. FSDP Design
4. FSDP Evaluation

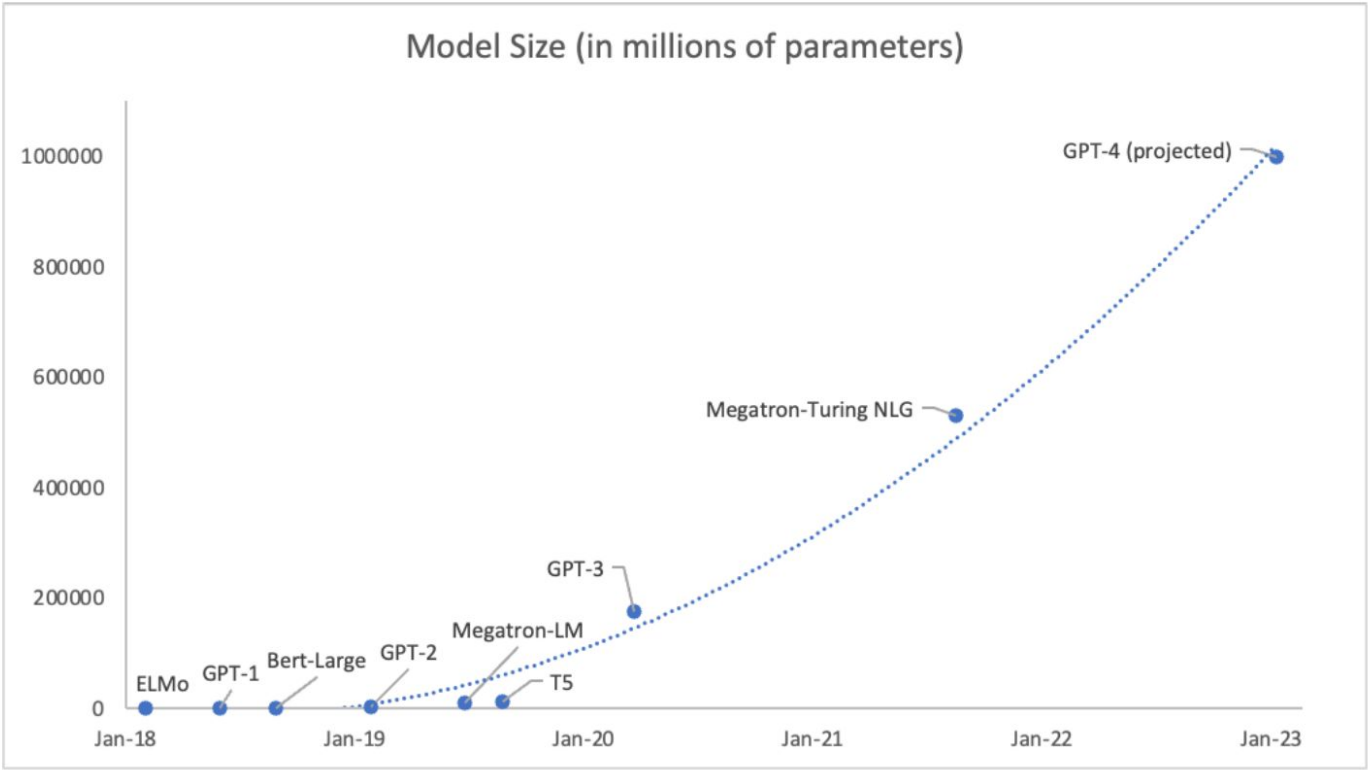
Parallelize LLM training: Naïve Data Parallelism



Problem: replicated LLM model does not fit in a GPU!

https://docs.nvidia.com/nemo-framework/user-guide/latest/nemotoolkit/features/parallelisms.html

LLM scaling law motivates its growth in size



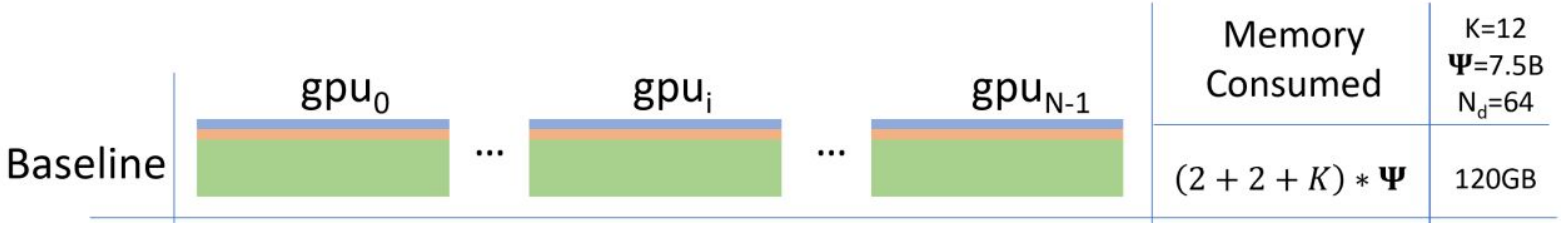
https://www.digitalnative.tech/p/ai-in-2023-the-application-layer

LLM Training Memory Consumption

In GPT training (W≈l*12*h=175B), we need memory to store the following:

- Model weights: 2W
- Gradients: 2W
- Optimizer state: 3*4*W

H100 HBM size: 80GB
H200 HBM size: 141GB



*If Ψ=175B, the Memory Consumed will be **2800 GB**!*

Ψ denotes model size (number of parameters), K denotes the memory multiplier of optimizer states, and Nd denotes DP degree. In the example, we assume a model size of Ψ = 7.5B and DP of Nd = 64 with K = 12 based on mixed-precision training with Adam optimizer.

ZeRO: Memory Optimizations Toward Training Trillion Parameter Models, Rajbhandari et al., 2020 https://arxiv.org/pdf/1910.02054

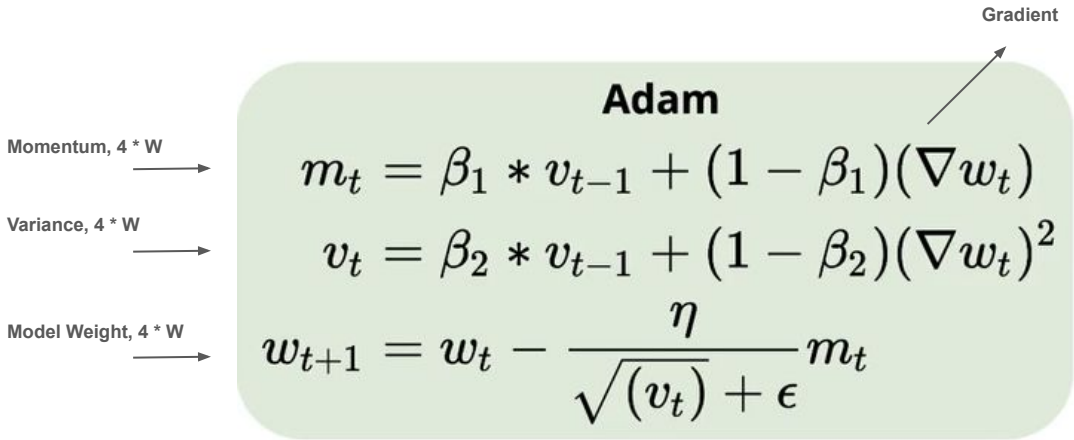
What is Optimizer State?

optimizer state *refers to the internal data that an optimizer maintains while training a model.*

ZeRO train a trillion-parameter model with Adam optimizer, so we use Adam Optimizer as an example.

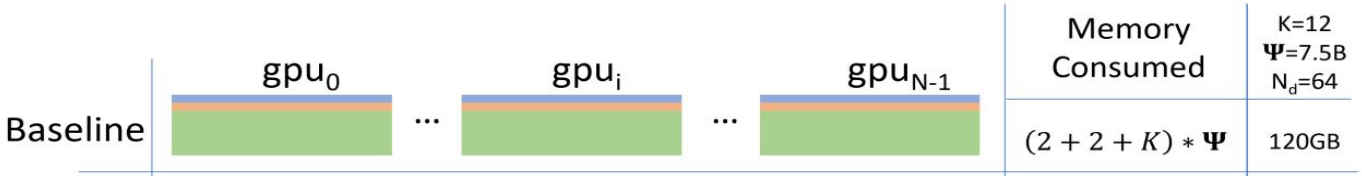
Adam optimizer

We need to store a copy of model weight, momentum and variance, all in FP32 format, which is the **Optimizer State** for Adam.



Limitations of Data Parallel

- DP requires **all model parameters, gradients,** and **optimizer states** to fit in the memory of one GPU device.
- Inadequate for supporting Large models

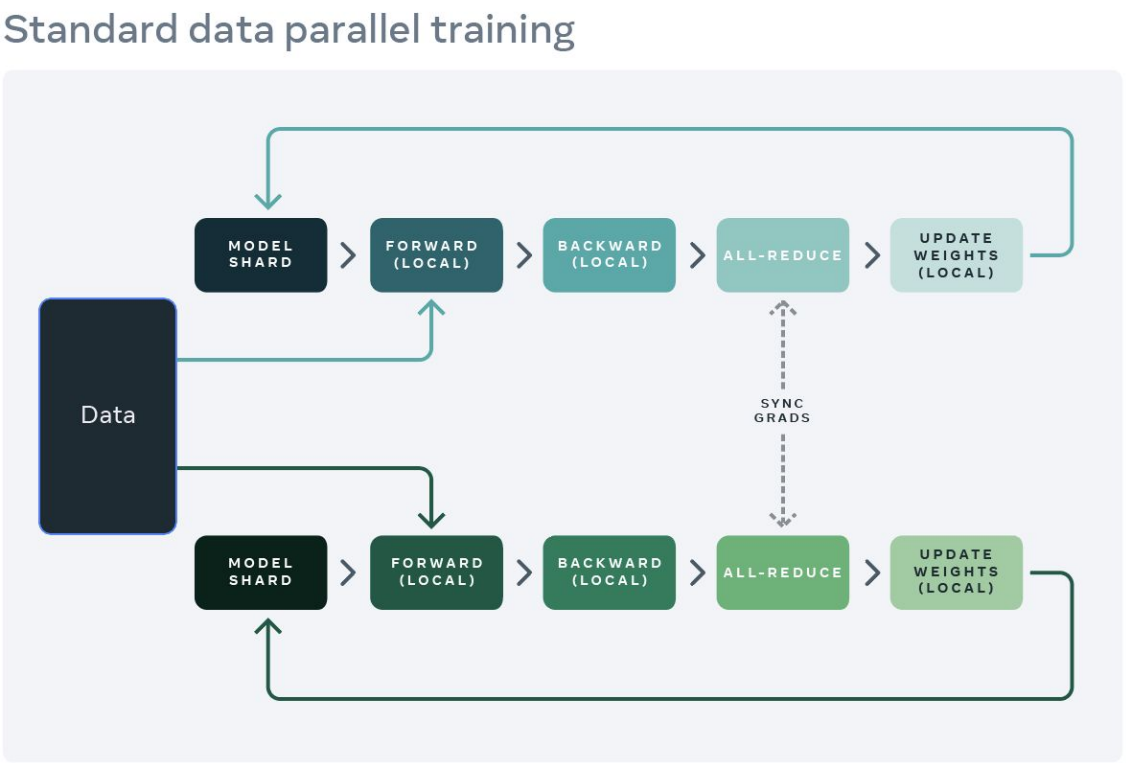


H100 HBM size: 80GB
H200 HBM size: 141GB

Ψ denotes model size (number of parameters), K denotes the memory multiplier of optimizer states, and Nd denotes DP degree. In the example, we assume a model size of Ψ = 7.5B and DP of Nd = 64 with K = 12 based on mixed-precision training with Adam optimizer.

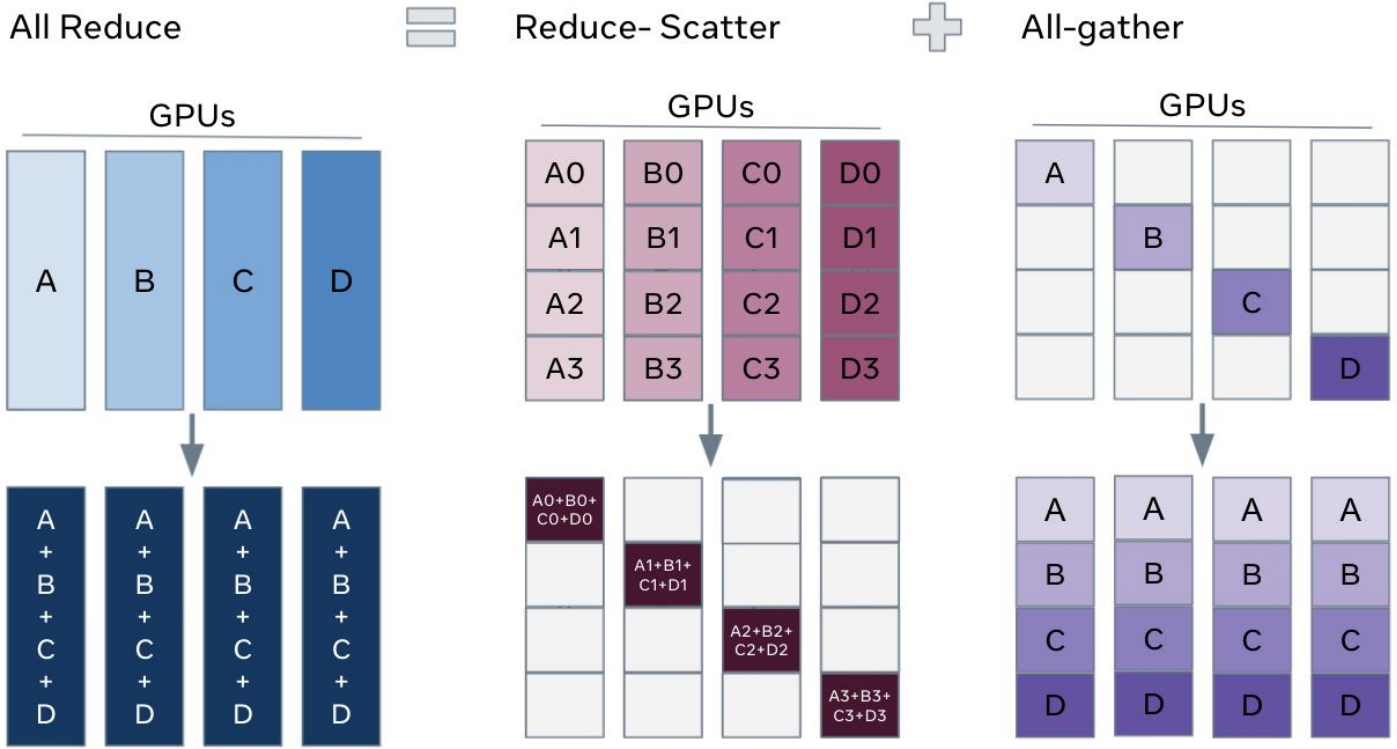
ZeRO: Memory Optimizations Toward Training Trillion Parameter Models, Rajbhandari et al., 2020
https://arxiv.org/pdf/1910.02054

How DP (Data Parallelism) Works?



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

What is All Reduce?



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

FSDP: Fully Sharded Data Parallelism

- A new variant of DP
- Trade off extra communication for memory reduction
- First introduced by DeepSpeed Zero
- Adopted by PyTorch FSDP

ZeRO: Memory Optimizations Toward Training Trillion Parameter Models, Rajbhandari et al., 2020 https://arxiv.org/pdf/1910.02054
PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel https://arxiv.org/pdf/2304.11277

FSDP: Fully Sharded Data Parallelism

DP (Data Parallel)

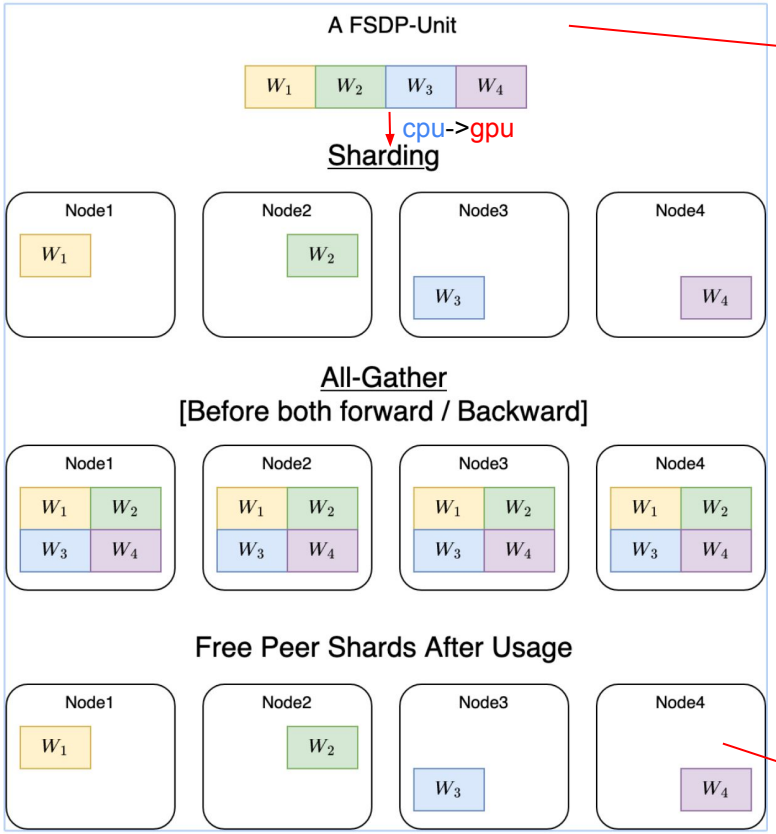
- Split the Data
- Replicates the full model on each GPU.
- Gradients are synchronized across GPUs after each backward pass.

FSDP (Fully Sharded Data Parallel)

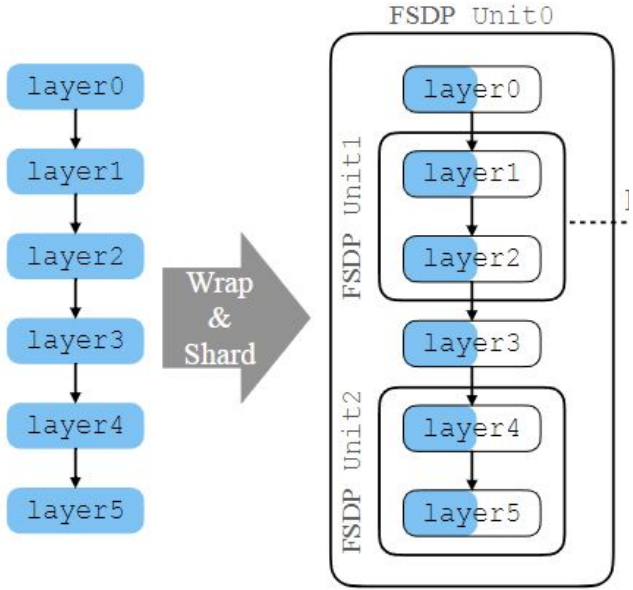
- Split the Data
- Shards both the model parameters and optimizer states across GPUs.
    - Load layers' weight when needed, free them when not needed
- Gradients are synchronized locally for the sharded parameters. Each GPU only holds part of the model and gradients.

ZeRO: Memory Optimizations Toward Training Trillion Parameter Models, Rajbhandari et al., 2020 https://arxiv.org/pdf/1910.02054
PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel https://arxiv.org/pdf/2304.11277

FSDP Forward



A FSDP Unit could be a layer, a group of layers, like:

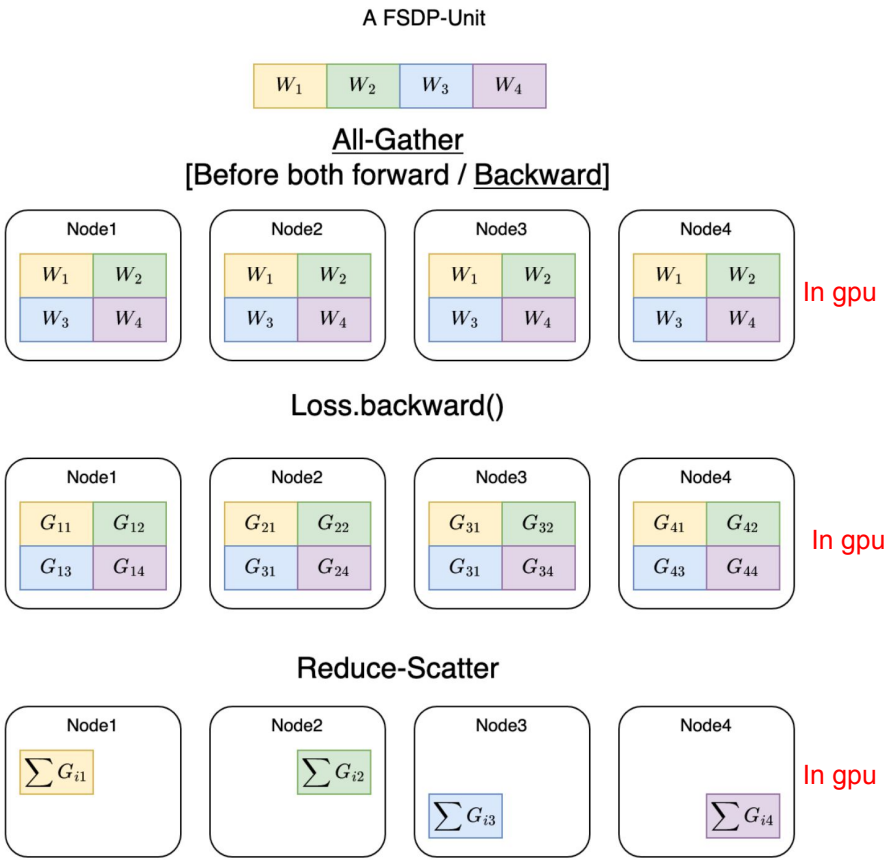


Free the weight to reserve memory for next fsdp unit

https://docs.google.com/presentation/d/1ntPSYg-WphI8sErwjUI0AztOY1i4SZmQuvmGhkeREIA/edit#slide=id.g2318fd43235_0_111

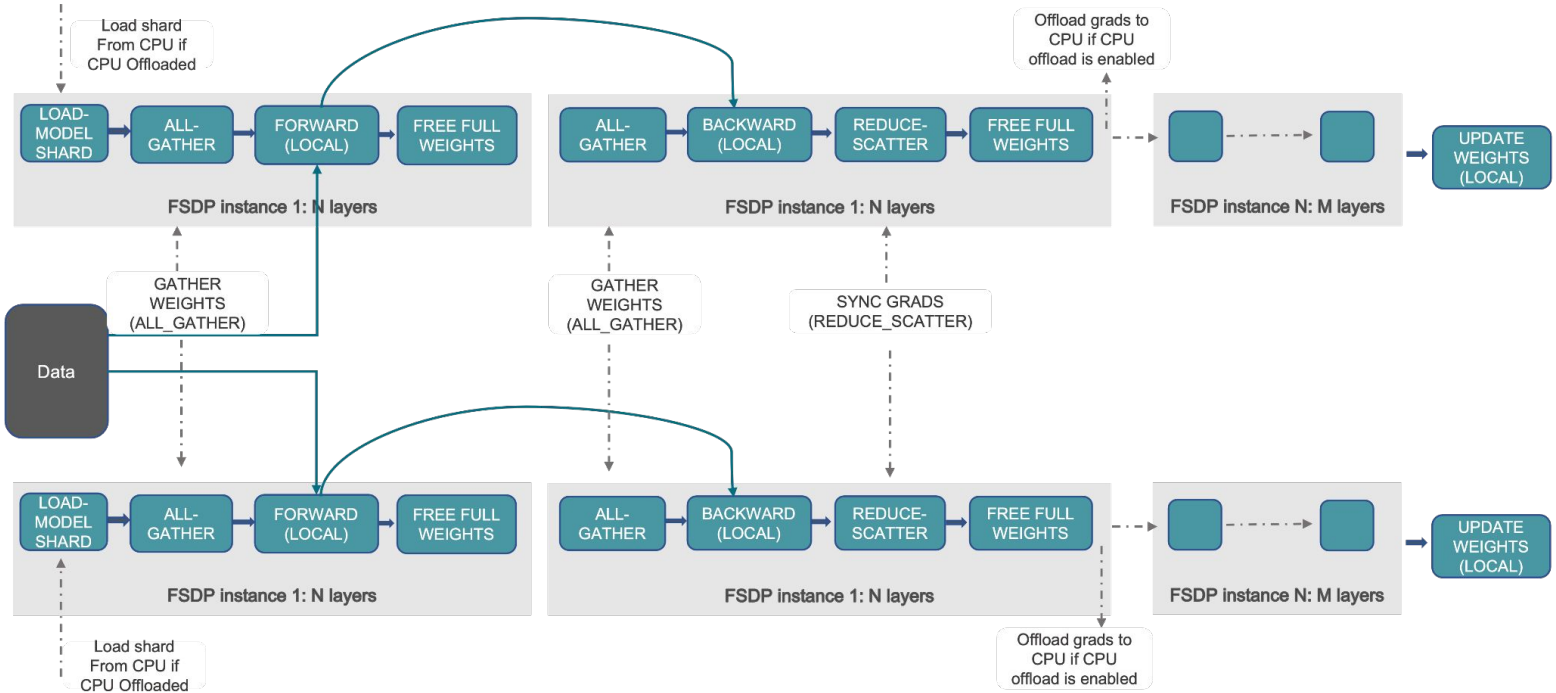
FSDP Backward Workflow

- After all-gather
  - all nodes have the same W_i
  - different nodes have different G_i



https://docs.google.com/presentation/d/1ntPSYg-Wphl8sErwjUl0AztOY1i4SZmQuvmGhkeREIA/edit#slide=id.g2318fd43235_0_111

FSDP Full Workflow



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

FSDP Optimization

- Overlap communication and computation
    - AllGather needed for the next compute operator can be overlapped with the the current computation operator
    - In the backward path, issue AllGather before Reduce-Scatter.
- Group model weights into units and handles each unit independently
    - Collective communication is more efficient with larger unit

FSDP Kernel Execution workflow

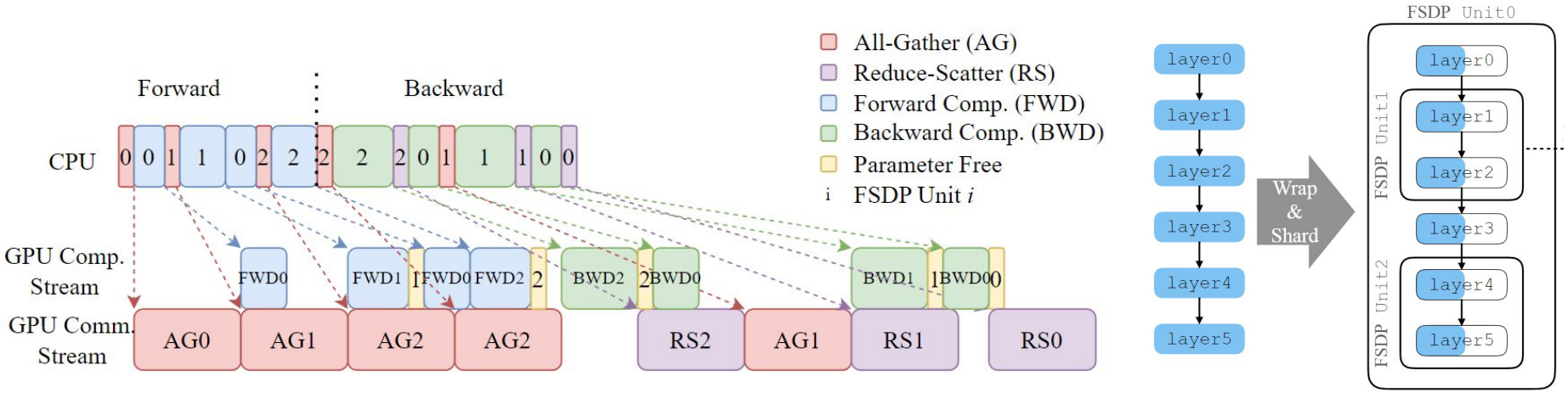
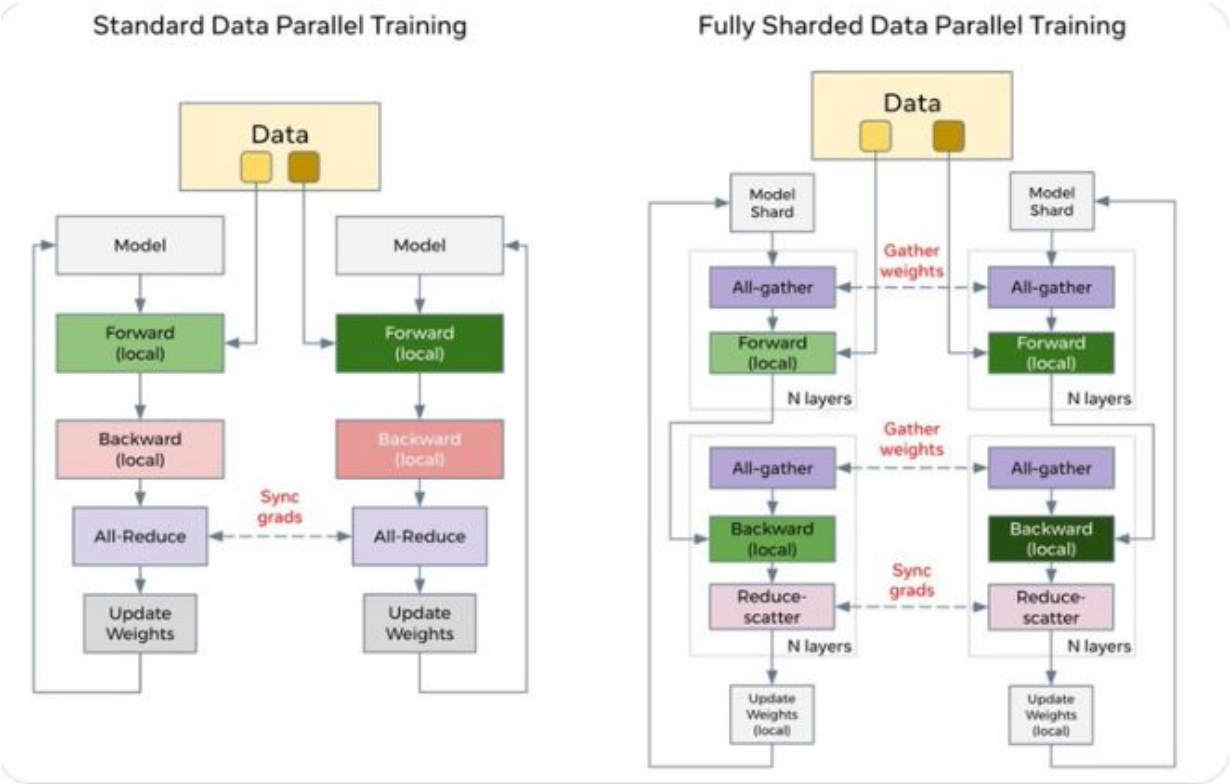


Figure 5: Overlap Communication and Computation

PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel https://arxiv.org/abs/2304.11277

FSDP vs DP



https://docs.habana.ai/en/latest/PyTorch/PyTorch_FSDP/Theory_of_operations.html

FSDP Evaluation

- Objective
  - Compare the capability with DP
  - Assess the scalability of FSDP
- Workload
  - Hugging- Face T5-11B transformer
  - minGPT-175B transformer
  - DHEN recommendation model (768B and 550M parameters)
- Environments
  - 8 to 512 A100 80GB GPUs
  - Interconnected by a 2Tb/s RoCE network

FSDP vs DP

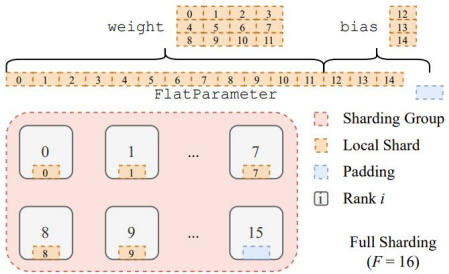


Figure 3: Full Sharding Across 16 GPUs

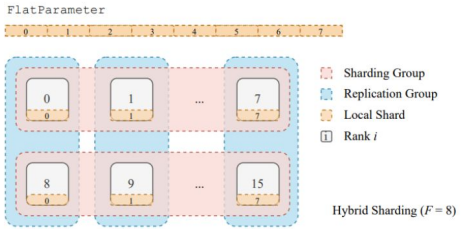
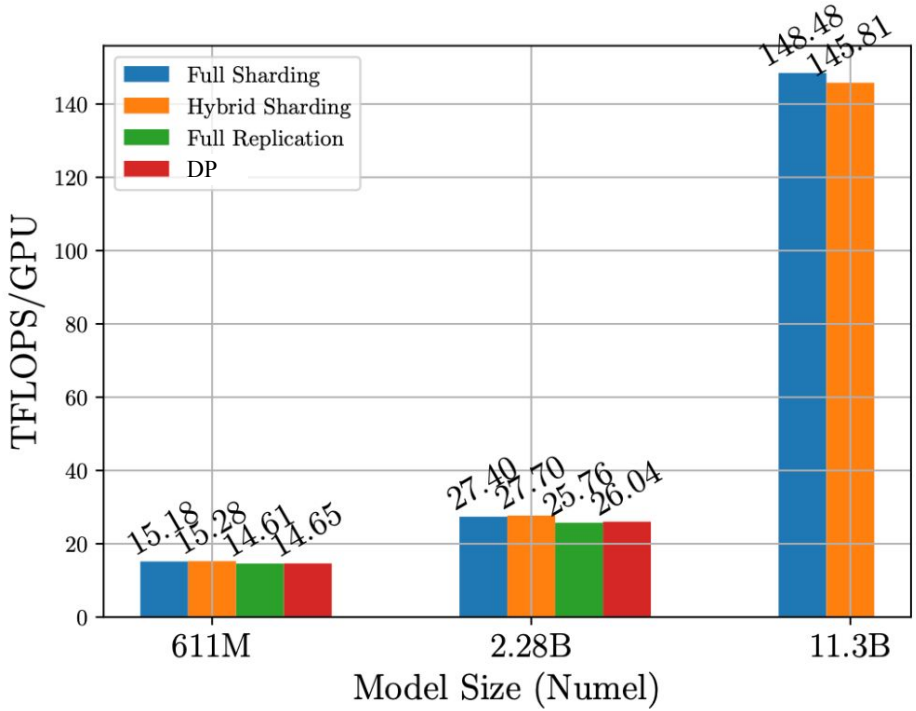
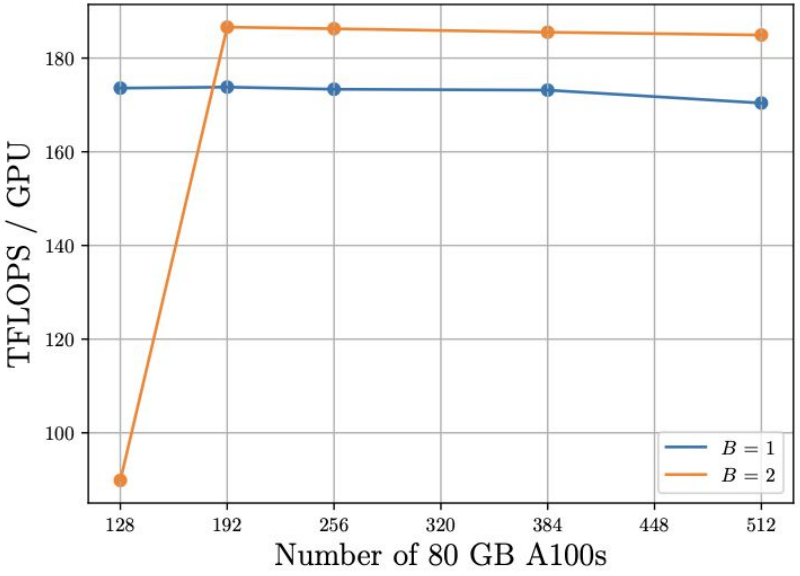


Figure 4: Hybrid Sharding on 16 GPUs: GPUs are configured into 2 sharding groups and 8 replication groups

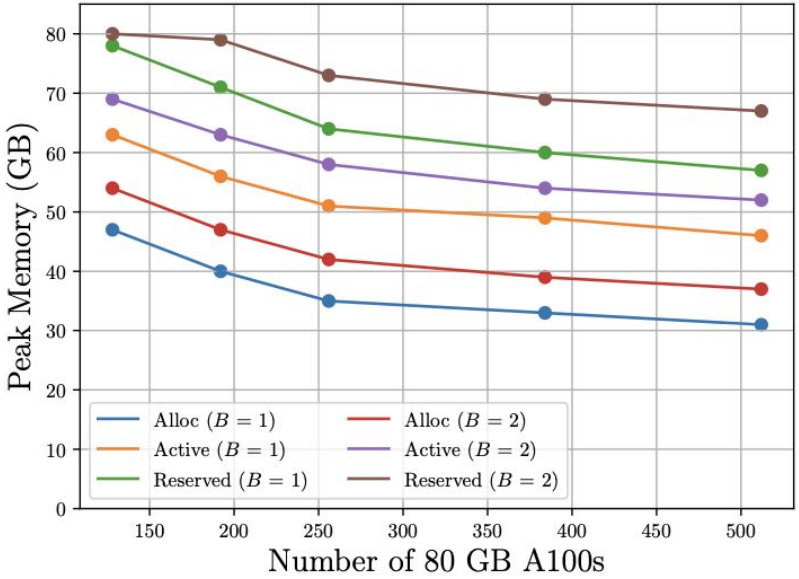


(a) Model Scale

Scalability

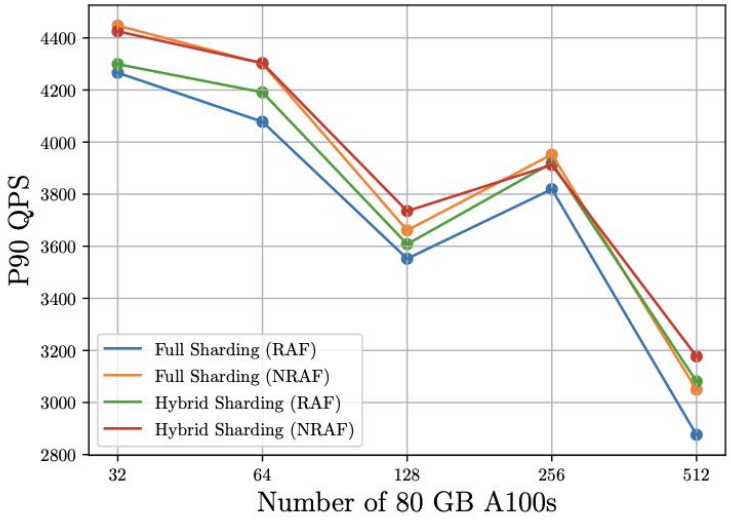


(b) GPT-175B TFLOPS

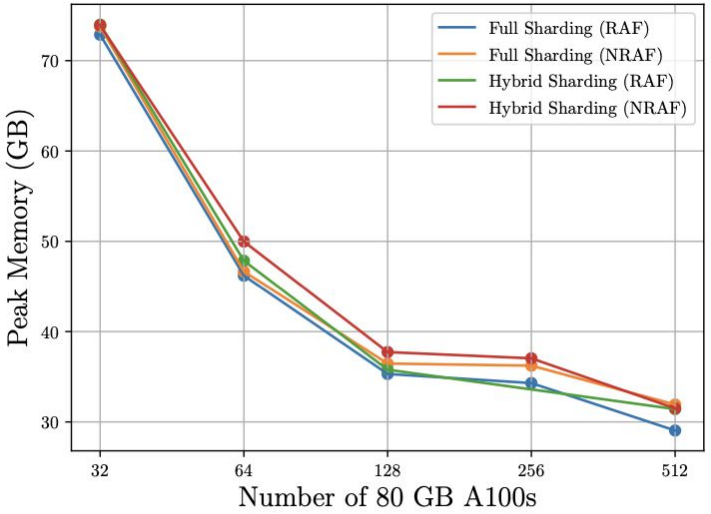


(b) GPT-175B

Throughput vs Memory Footprint

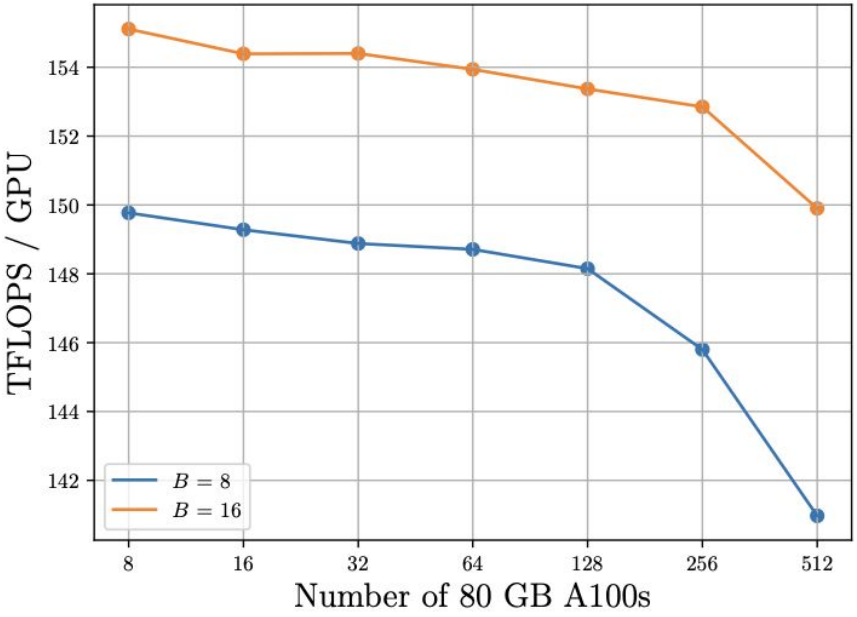


(a) DHEN QPS

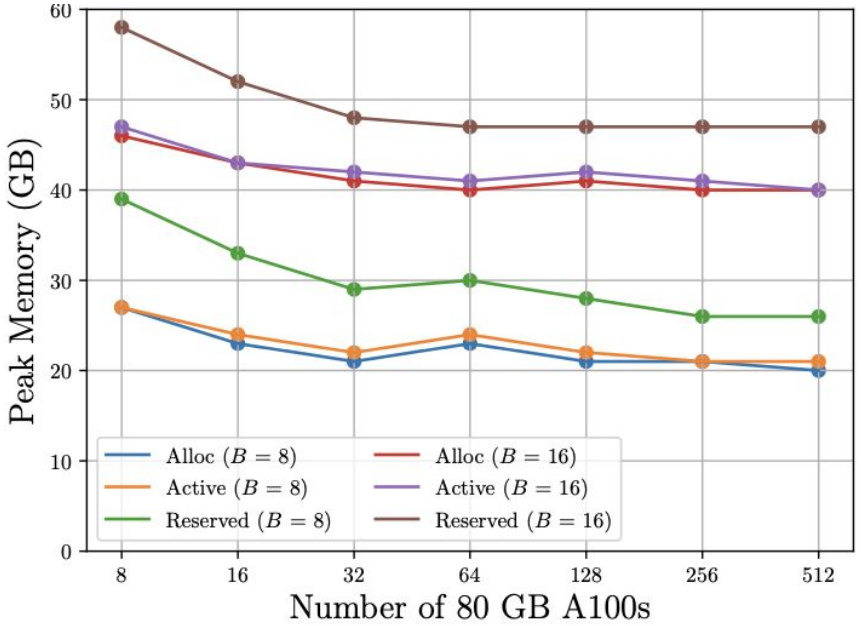


(a) DHEN

Overkill



(c) T5-11B TFLOPS



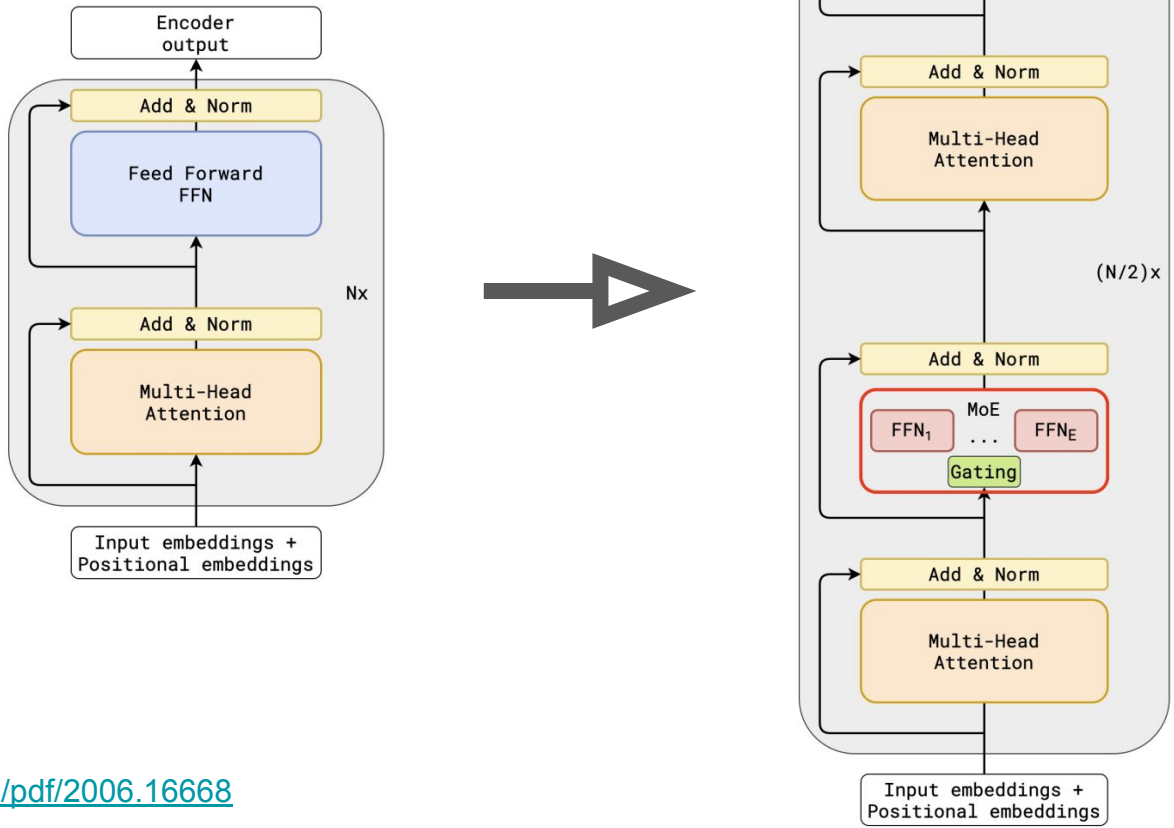
(c) T5-11B

Reference

1. Pytorch FSDP: https://dl.acm.org/doi/pdf/10.14778/3611540.3611569
2. FSDP slides, from WUSTL: https://www.ccrc.wustl.edu/~roger/566S.s24/presentations/Shen_Presentation.pdf
3. How FSDP works, from youtube video: https://www.youtube.com/watch?v=By_O0k102PY
4. ZeRO By MicroSoft: https://arxiv.org/pdf/1910.02054
5. Meta Blog: https://engineering.fb.com/2021/07/15/open-source/fsdp/

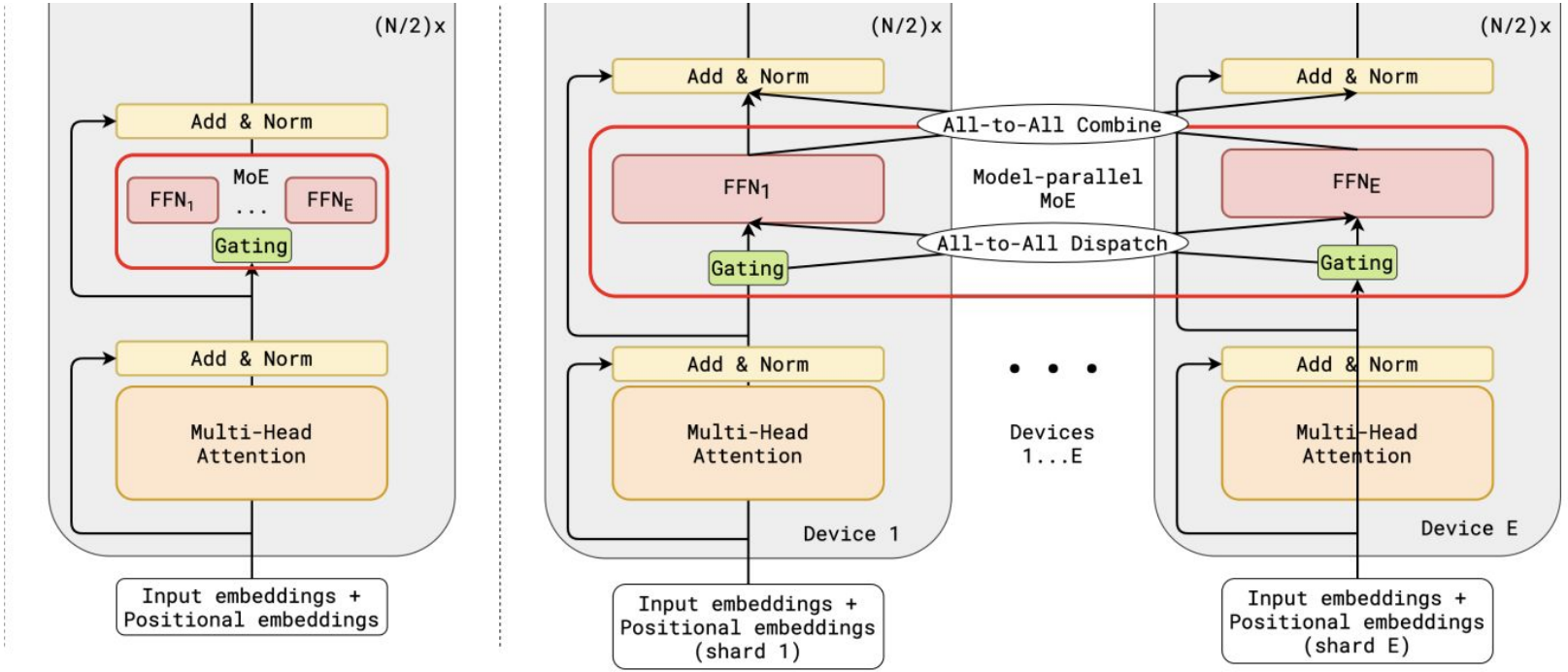
Tutel: Adaptive Mixture-of-Experts at Scale

Background – MoE



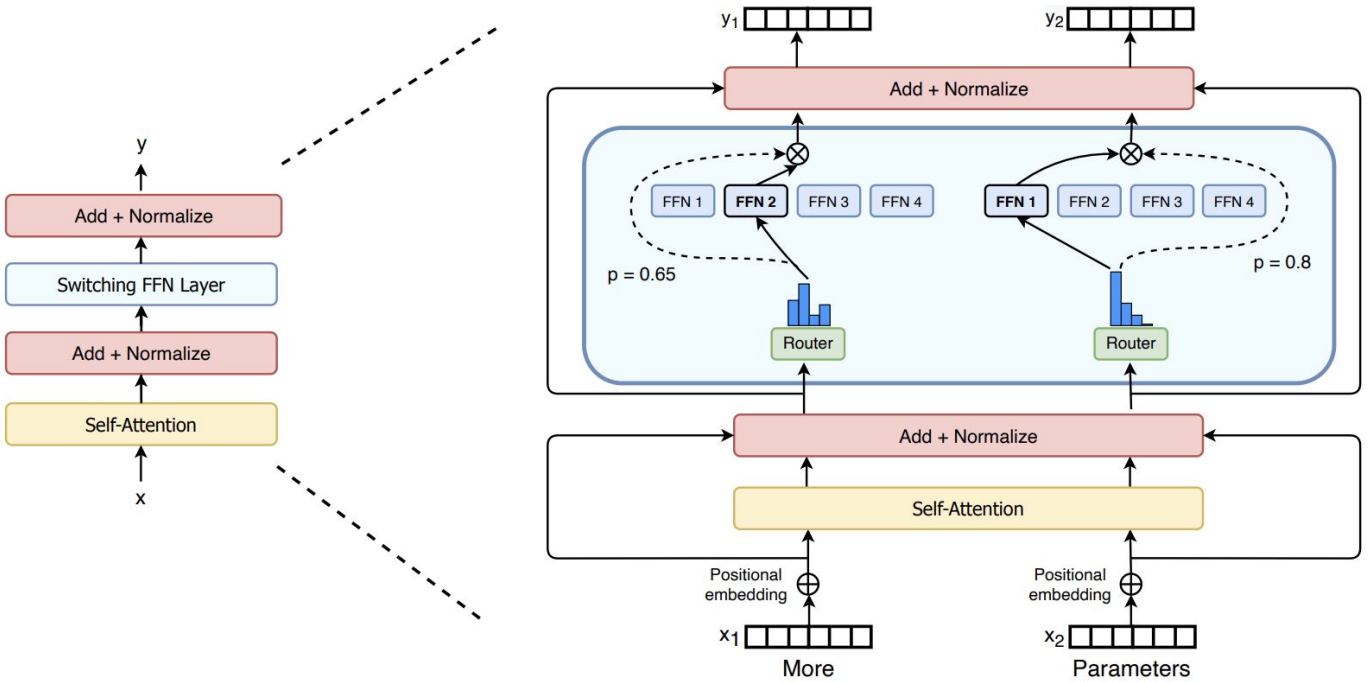
Gshard https://arxiv.org/pdf/2006.16668

MoE - Expert Parallelism

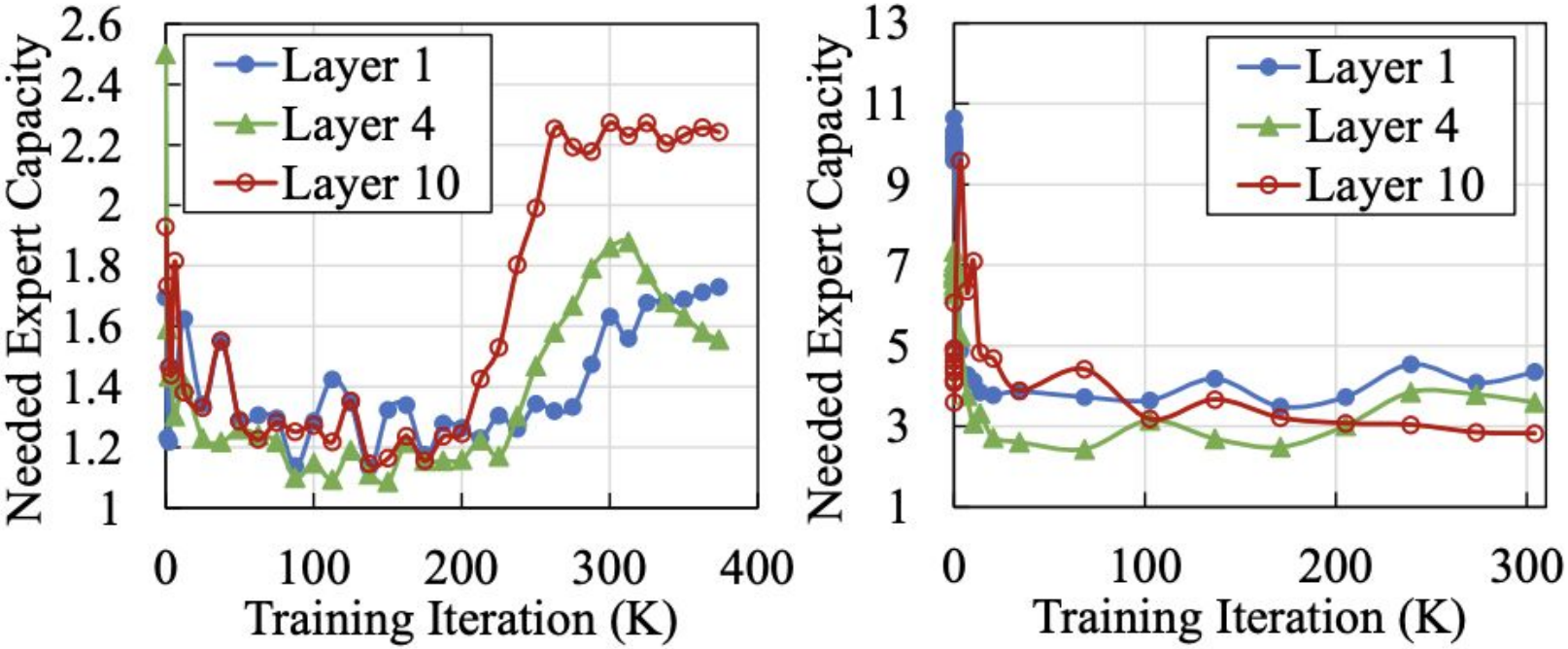


Gshard https://arxiv.org/pdf/2006.16668

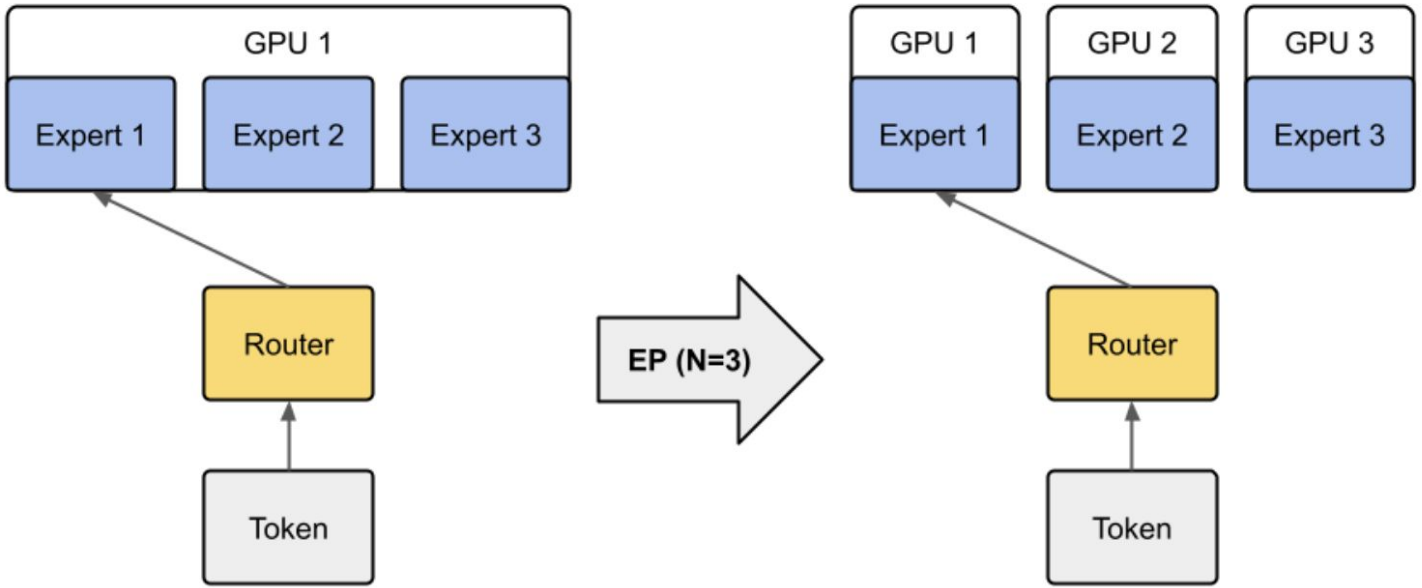
Background – MoE



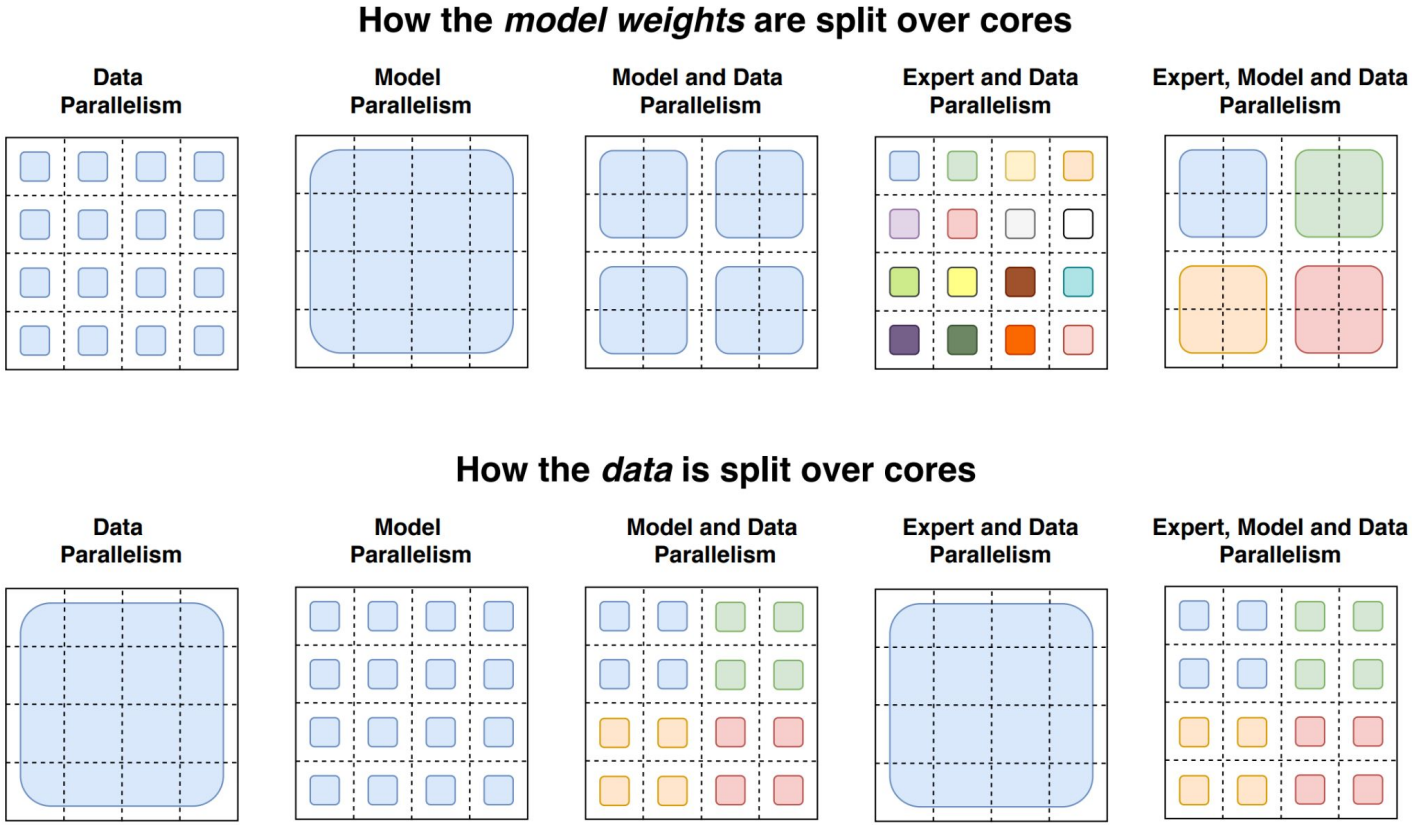
MoE - Dynamic Workload



MoE - Expert Parallelism

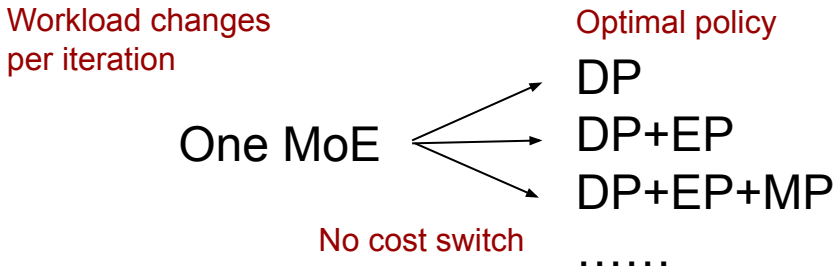


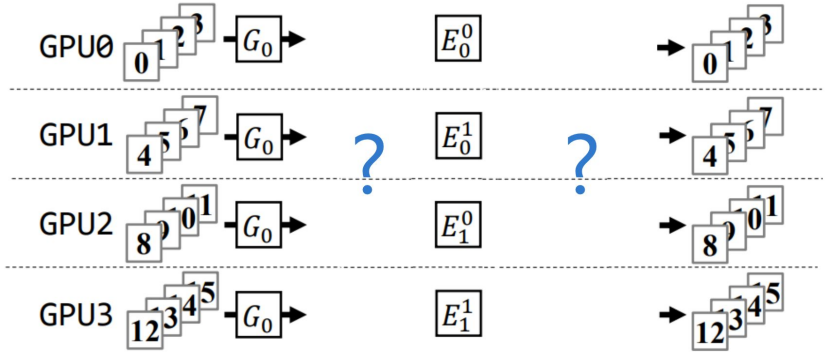
Background & Motivation



Tutel Design - Adaptive MoE at Scale

Idea: Switchable Parallelism





No location collisions: (the same as FSDP)

- **Parameter Placement**: evenly sharded
- **Input Layout**: the same as DP
- **Expert Gradients**: exclude all_reduce

Tutel Design - Adaptive MoE at Scale

- Switchable Parallelism
- Evaluation of Switchable Parallelism
- Adaptive Pipelining
- Evaluation of Adaptive Pipelining
- 3 Extra Adaptive Mechanisms or Optimizations
- Evaluation of Tutel MoE

Tutel Design - Adaptive MoE at Scale

Idea: Switchable Parallelism

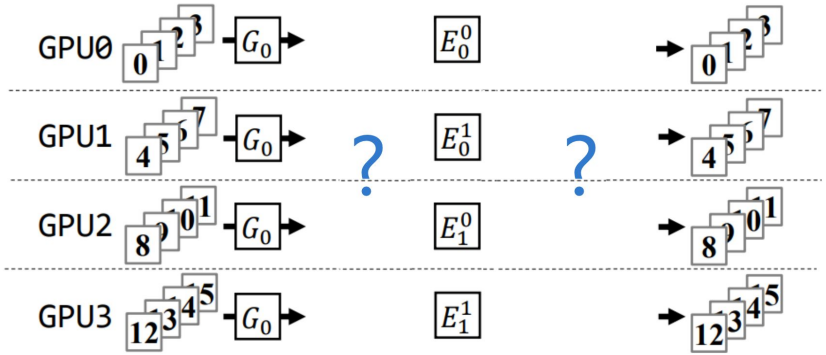
Workloads change per iteration

One MoE

Optimal strategy

- DP
- DP+EP
- DP+EP+TP
- ……

No cost switch

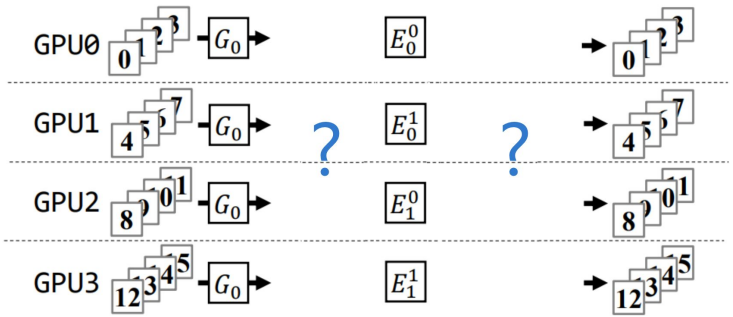


No location collisions (One data layout for all strategies)

- **Input Layout**: evenly split (the same as DP)
- **Parameter Placement**: evenly sharded
- **Expert Gradients**: exclude all-reduce (reduce-scatter + all-gather)

Switchable Parallelism

DP
EP
MP (TP)



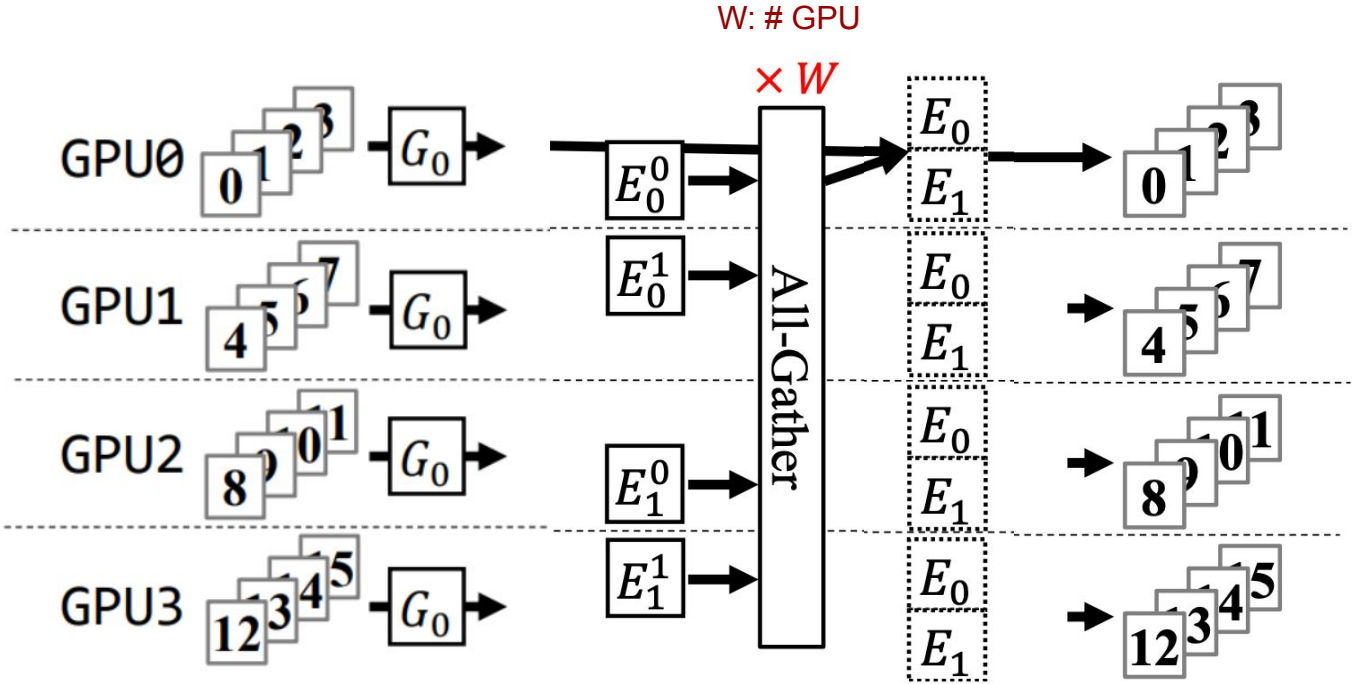
implement

Parallelism Method	Communication Complexity	Limitation	Comment
① DP	$\mathcal{O}(P)$	-	Possibly optimal
② MP	$\mathcal{O}(C_g \cdot W)$	-	No better than ⑥
③ EP	$\mathcal{O}(C_g)$	$E/W \geq 1$	No better than ⑥
④ DP+MP	$\mathcal{O}(C_g \cdot r + P/r)$	$1 \leq r \leq W$	No better than ⑦ for any r
⑤ EP+DP	$\mathcal{O}(C_g + P/E)$	-	A special case of $r = 1$ in ⑦
⑥ EP+MP	$\mathcal{O}(C_g \cdot max\{1, W/E\})$	-	A special case of $r = W/E$ in ⑦
⑦ EP+DP+MP	$\mathcal{O}(C_g \cdot W/E)$ – if $r \geq W/E$ $\mathcal{O}(C_g \cdot r + P/E/r)$ – if $1 \leq r < W/E$	-	Possibly optimal

Eliminate sub-optimal options

- simplified set = { ①, ⑦ }

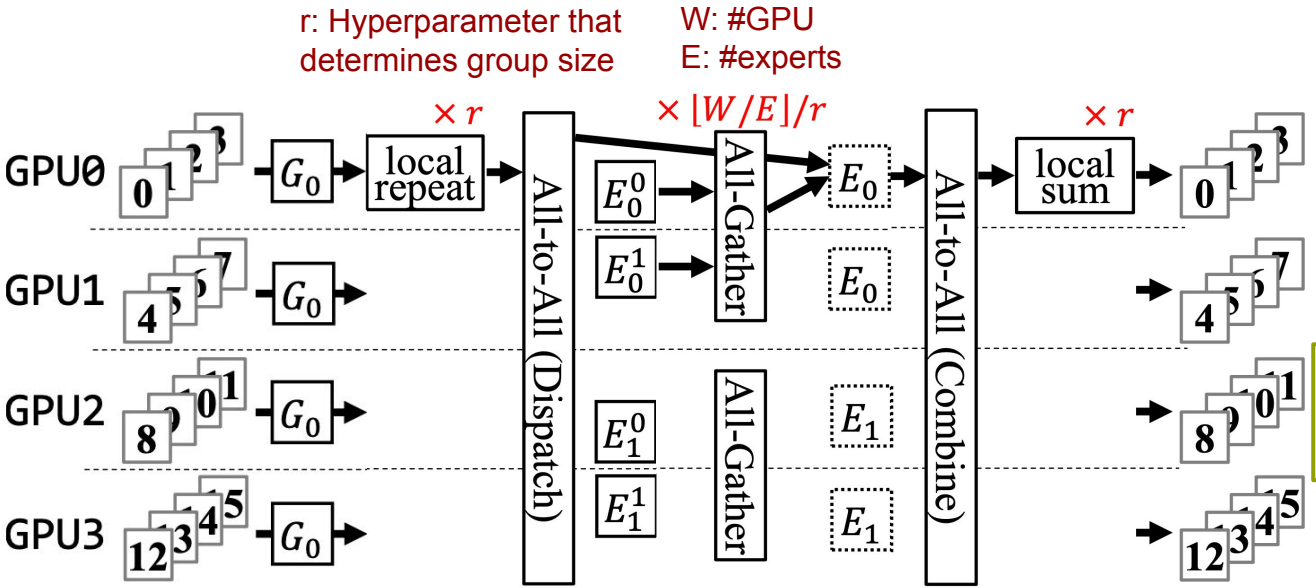
Switchable Data Parallelism



Transfer expert weights to each GPU

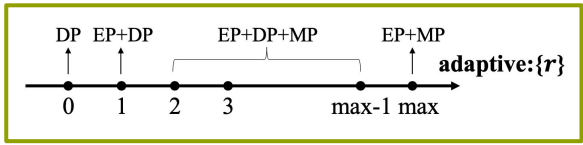
Computation

Switchable Data + Expert + Model Parallelism



If r = 2, no all-gather

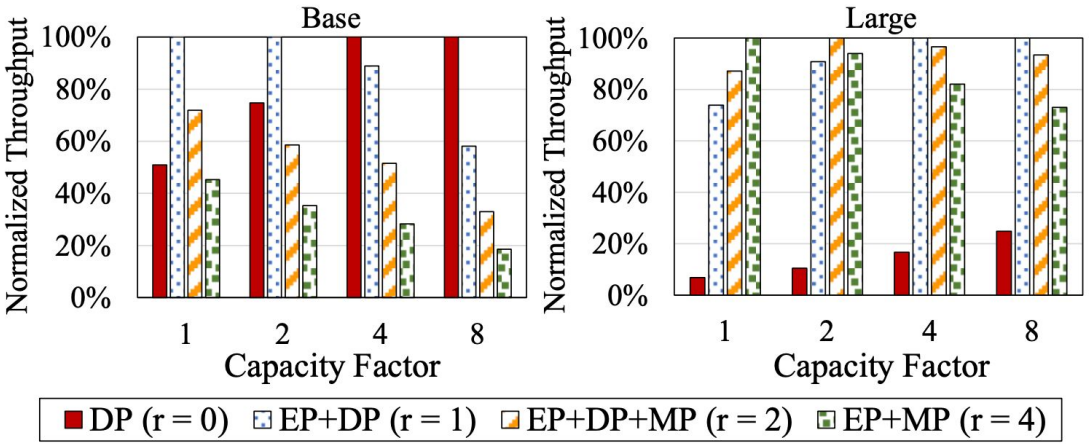
If r = 1, all-gather within 2 GPUs



Adjust strategy by tuning r

Evaluation of Switchable Parallelism

- Multiple Parallelism Throughput on Different Capacity Factors

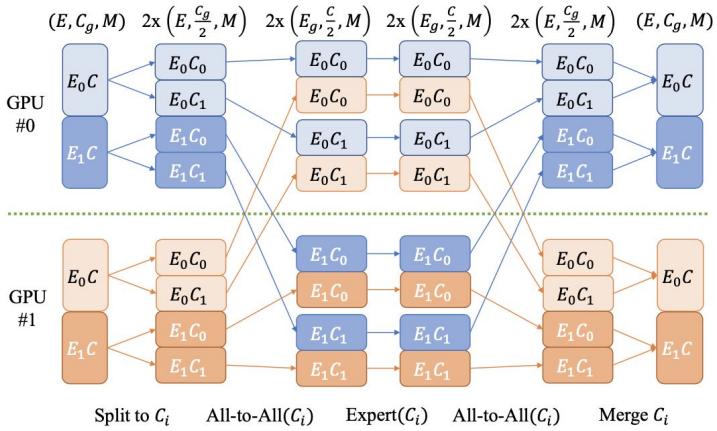


64 GPUs (A100) for 16 MoE Experts

(Larger Capacity Factor Implies Stronger Imbalance)

Adaptive Pipelining

- Concurrent Overlap between network communication and processor computation in dynamic workloads with proper granularities.

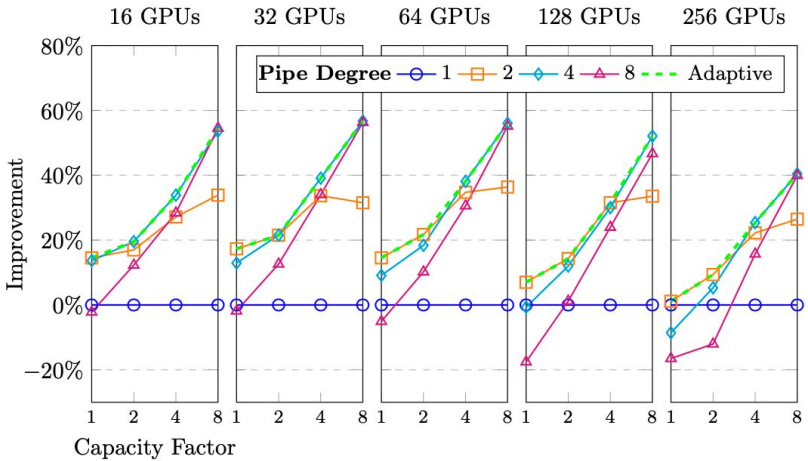


Data split by token

	Input Dispatch (Nvlink/InfiniBand)	Expert FFN (Processor)	Output Combine (Nvlink/InfiniBand)
t0	A2A		
t1	A2A	T ⊗ E	
t2	A2A	T ⊗ E	A2A
t3		T ⊗ E	A2A
t4			A2A

Evaluation of Adaptive Pipelining

- Efficiency of Pipeline Degree on Different Capacity Factors



16-256 GPUs (A100) with 2 MoE Experts / GPU

(Larger Capacity Factor Implies Stronger Imbalance)

- Small Pipeline degree:

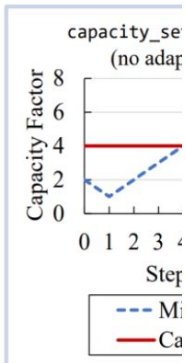
  Not take advantage of overlap.

- Large Pipeline degree:

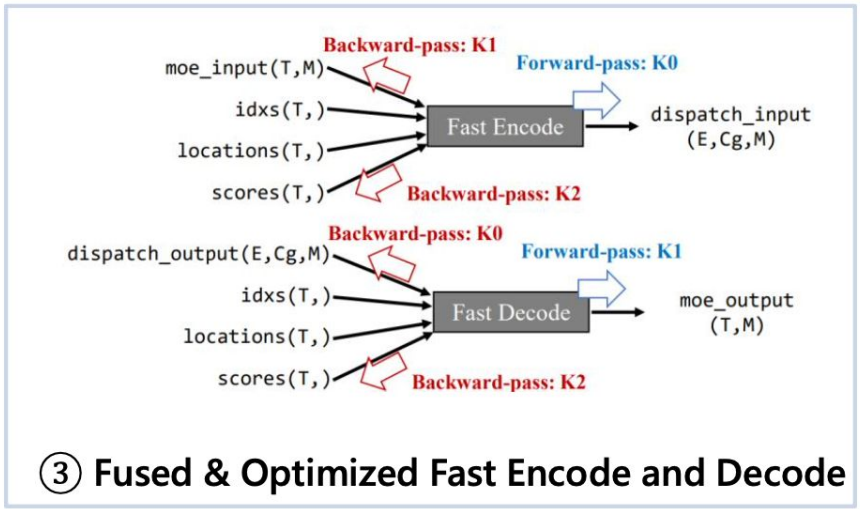
  Overhead of small-slice execution.

3 Extra Adaptive Mechanisms or Optimizations

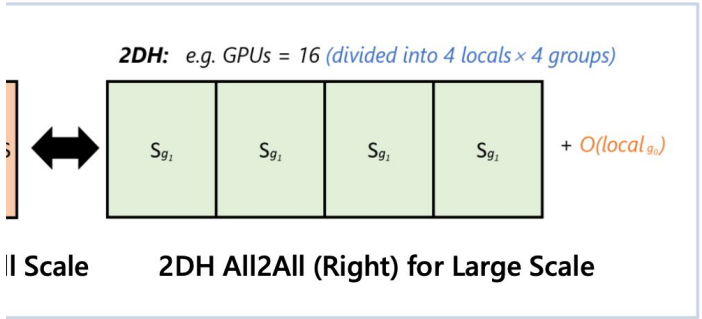
- Dynamic capacity controls
- Adaptive All-to-All algo. for different scales (Linear/2DH)
- Deeply fused ops for “Fast Encode” and “Fast Decode” (90%↓ mem)



① Differe

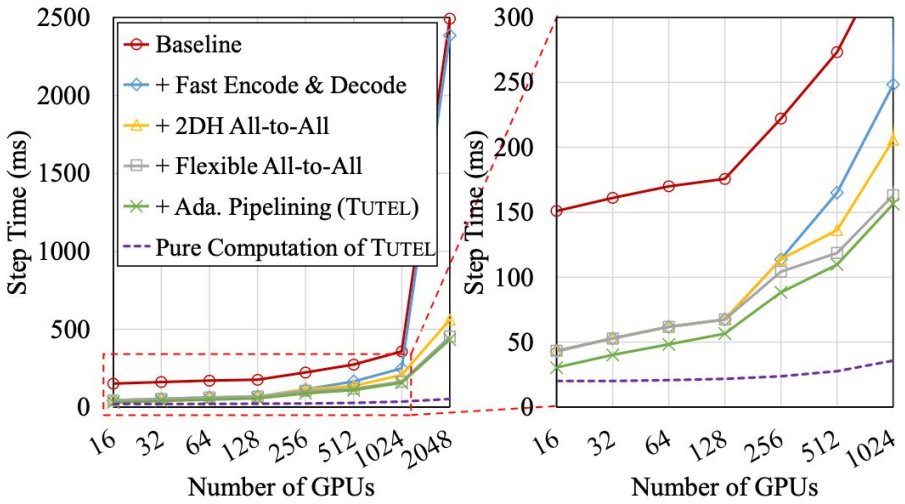


③ Fused & Optimized Fast Encode and Decode



Evaluation of Tutel MoE on 2,048 GPUs (A100)

- Baseline: Fairseq MoE / Deepspeed MoE
- Tutel optimization: Fast Encode & Decode
- above + 2DH All-to-All
- above + Flexible All-to-All
- above + adaptive parallelism
- Tutel computation time per device



Tutel MoE Layer delivers 4.96x and 5.75x speedup on 16 A100 and 2,048 A100, respectively

Thank you!

Q & A

Reference

1. Switch Transformers https://arxiv.org/pdf/2101.03961
2. Gshard https://arxiv.org/pdf/2006.16668