Open Source: https://github.com/microsoft/tutel

Tutel:
Adaptive Mixture-of-Experts at Scale

Changho Hwang, **Wei Cui**, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, Joe Chau, Peng Cheng, Fan Yang, Mao Yang, Yongqiang Xiong

Microsoft / Microsoft Research

Mixture-of-Experts (MoE)

- **Dense Model:**

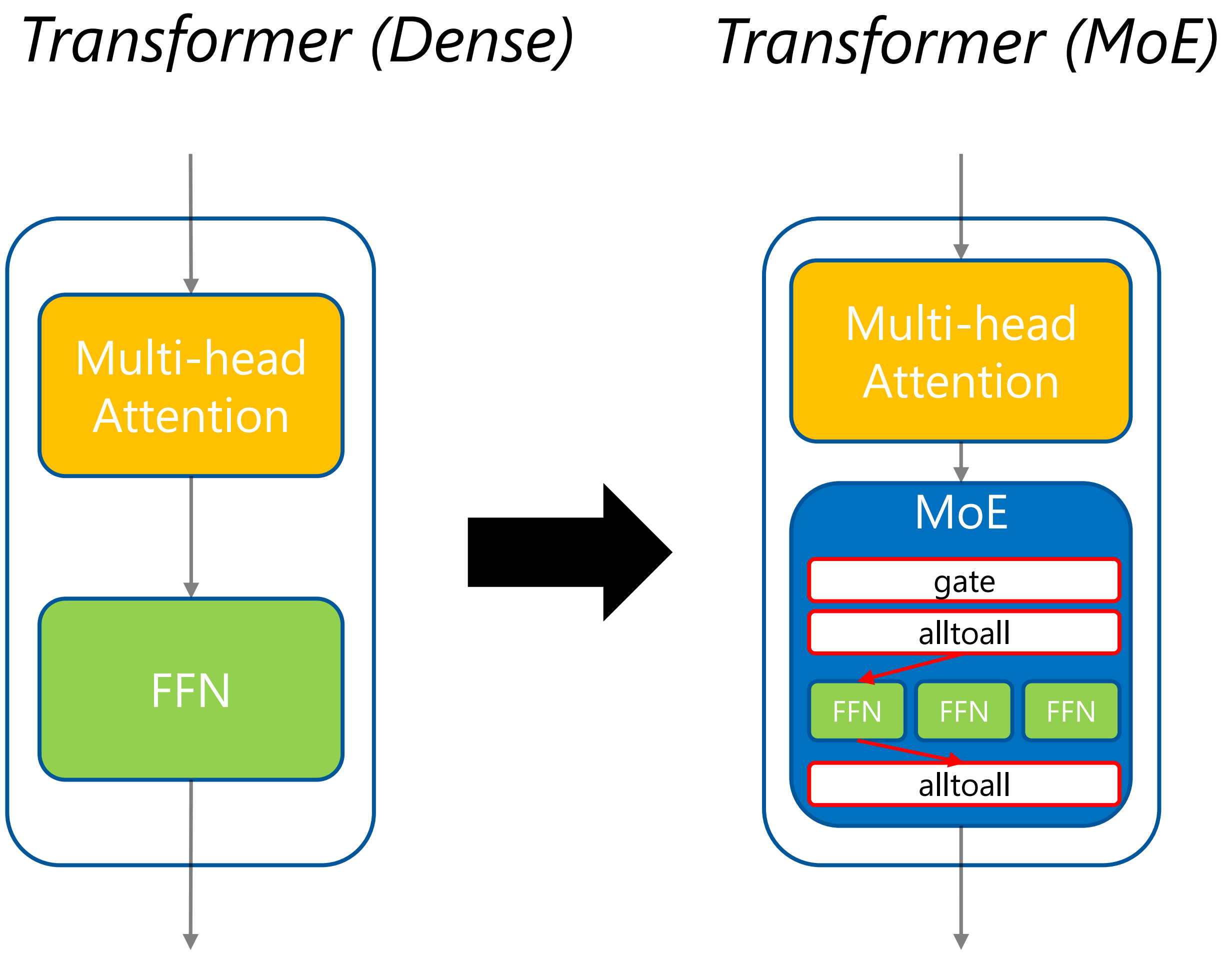
Scale Solutions: ZeRO / Model Parallel / ..

data ↑ param ↑ device ↑ → local mem ↑ net ↑

- **MoE based Model:**

"Key to unlock ***Exa-scale*** Training"

data ↑ param ↑ device ↑ → $\underbrace{\textit{local mem (-) net (-)}}_{\textit{"sub-linear" scaling}}$



Total Parameters:

|FFN| → |FFN| × expert_count

Mixture-of-Experts (MoE)

① Decide Expert ID:

$F_{gating}(input_x) \rightarrow expert\_id$

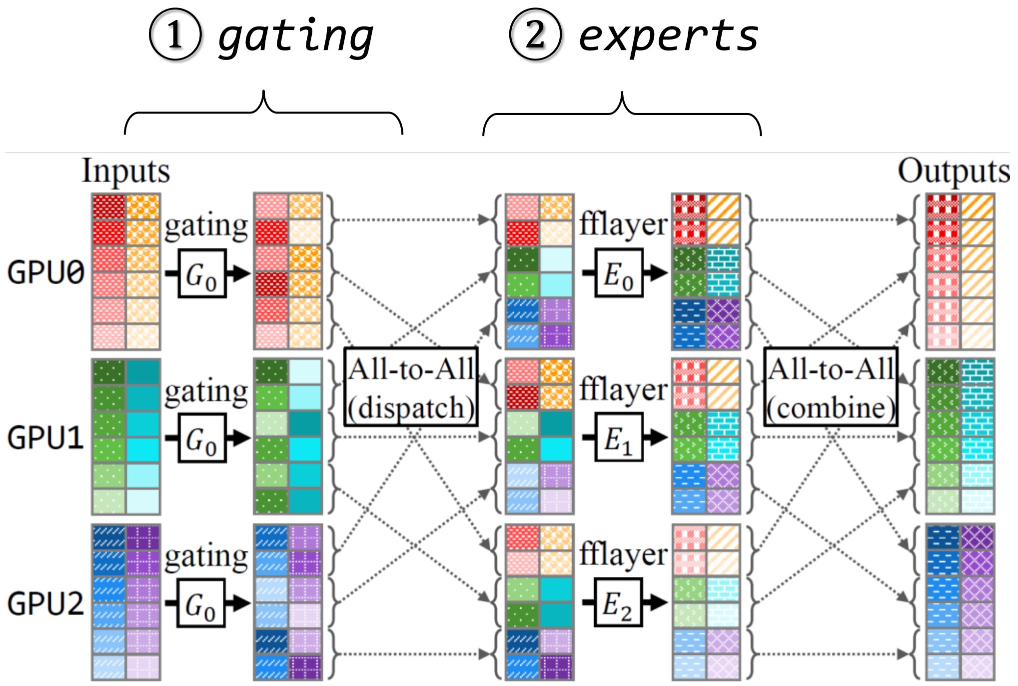
② Train With Target Expert ID:

$output = FFN_{expert\_id}(input_x)$

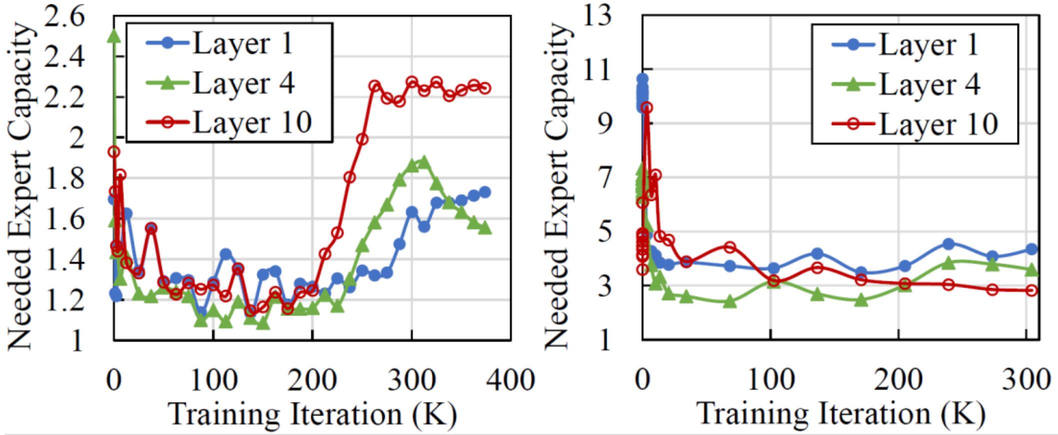
F_{gating} is trainable, so:

the dispatch from "Inputs → Experts":

- ***dynamically changed***
- ***potentially imbalanced***



Internal MoE layer Data Flow

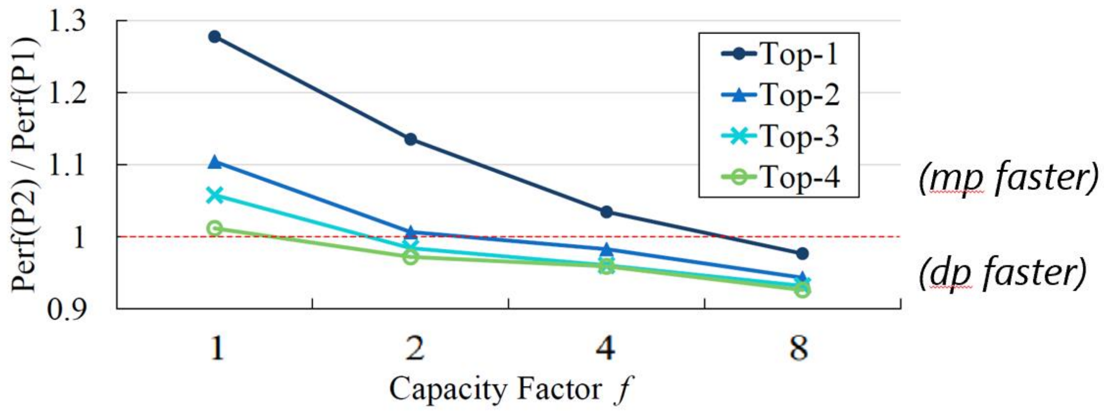


Imbalance States by Training Iterations

SwinV2-MoE tiny (left) and base (right)

Static Parallelism for Dynamic MoE

Static parallelism cannot satisfy all efficient preferences from ***dynamic*** workload

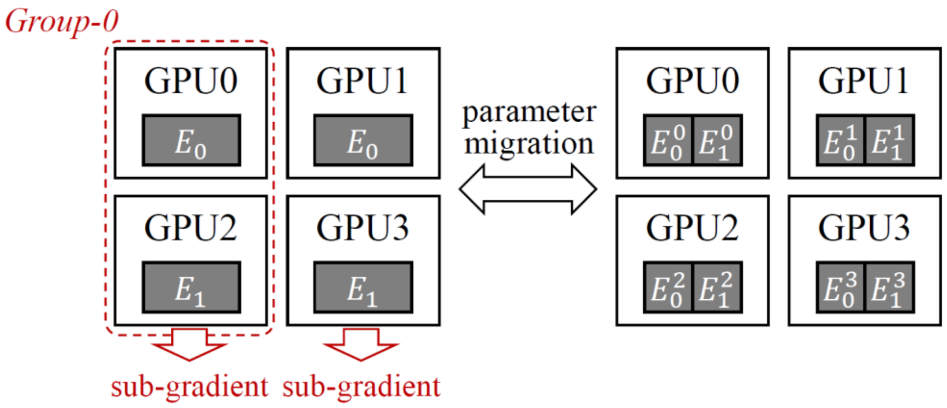


Parallelism Efficiency on Different Capacities

(P1: Data Parallel P2: Model Parallel)

Hard to Change Parallelism: Normal parallel solutions are not compatible to switch.

- Overhead of **parameter** migration
- Different **input** layout, **gradient** update, etc..



Existing data ↔ model parallelism for MoE

Tensor parallelism is not the only factor deserved to change in dynamic workload.

Tutel Design

- Adaptive MoE at Scale

Switchable Parallelism

One MoE → Multi-path Parallelism:

$P_1(+)\ \ P_2(-) \leftrightarrow P_1(-)\ \ P_2(+)$

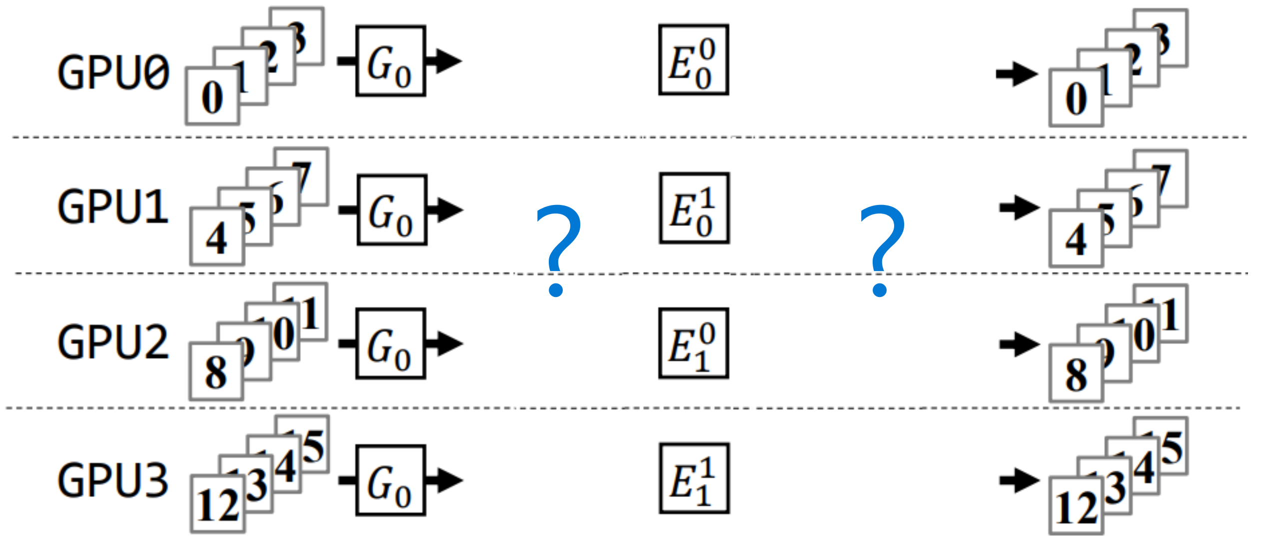
no-cost

No location collisions:

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

Eliminate sub-optimal options →

- simplified set = { ① , ⑦ }



Base Partition Framework

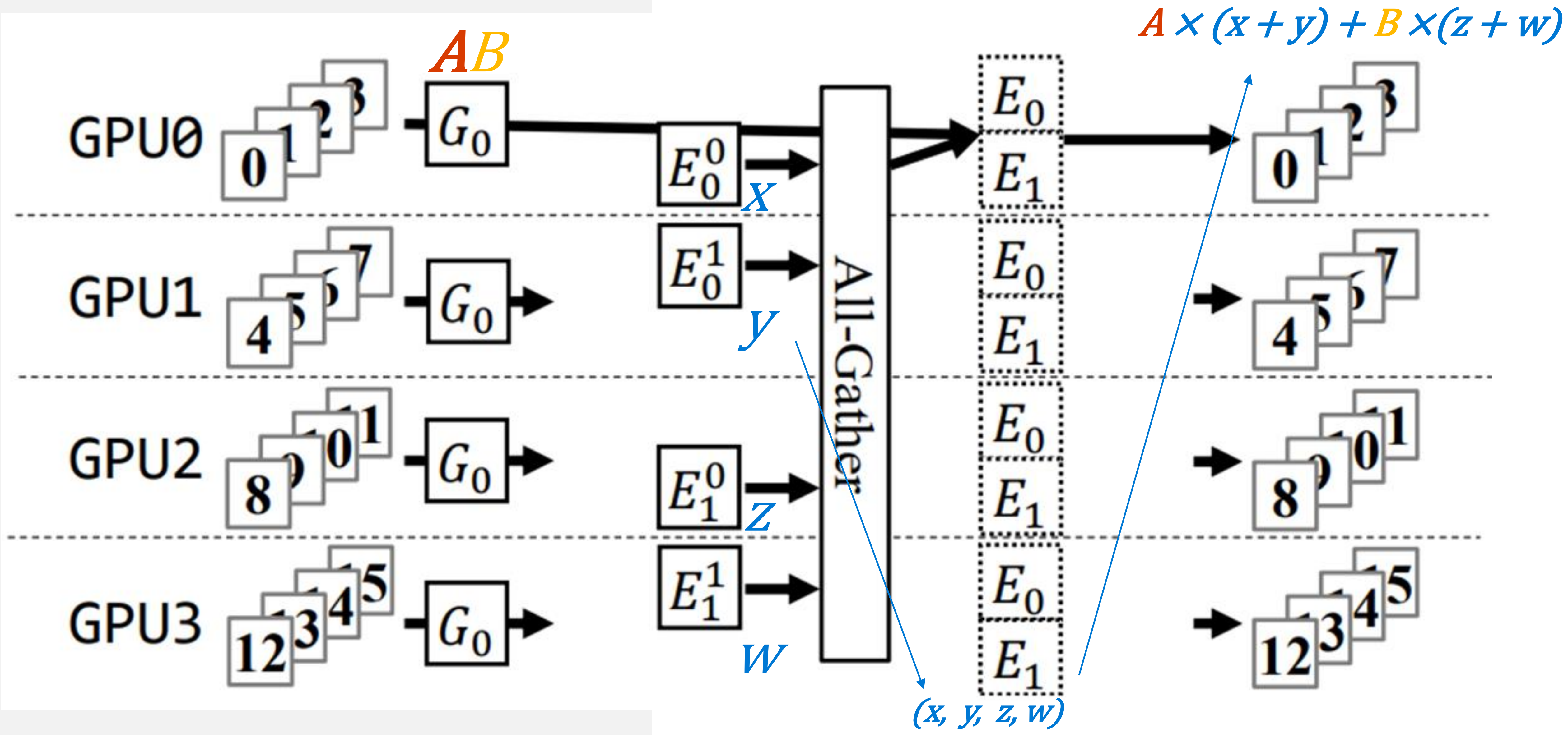
implement

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

Tensor Parallel Options

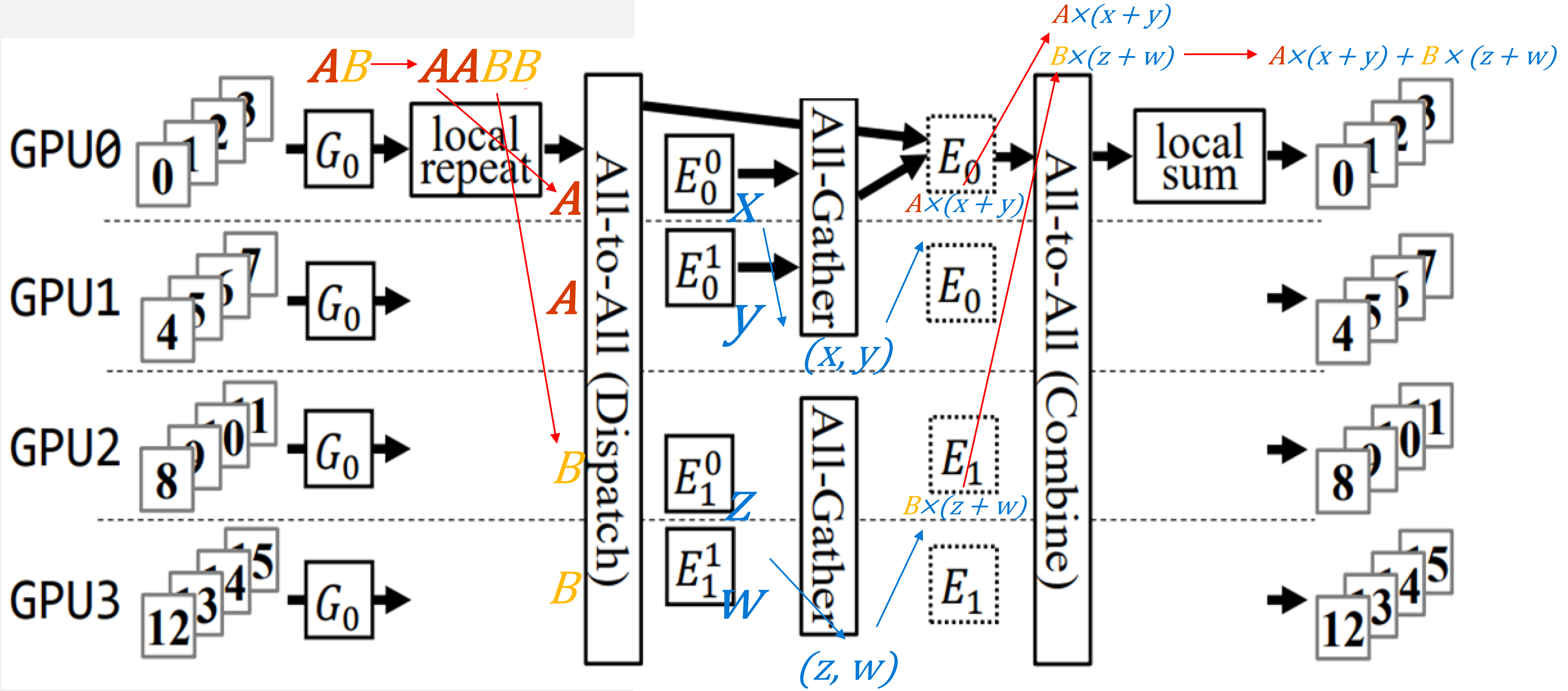
Switchable Parallelism

Path 1: Data parallelism



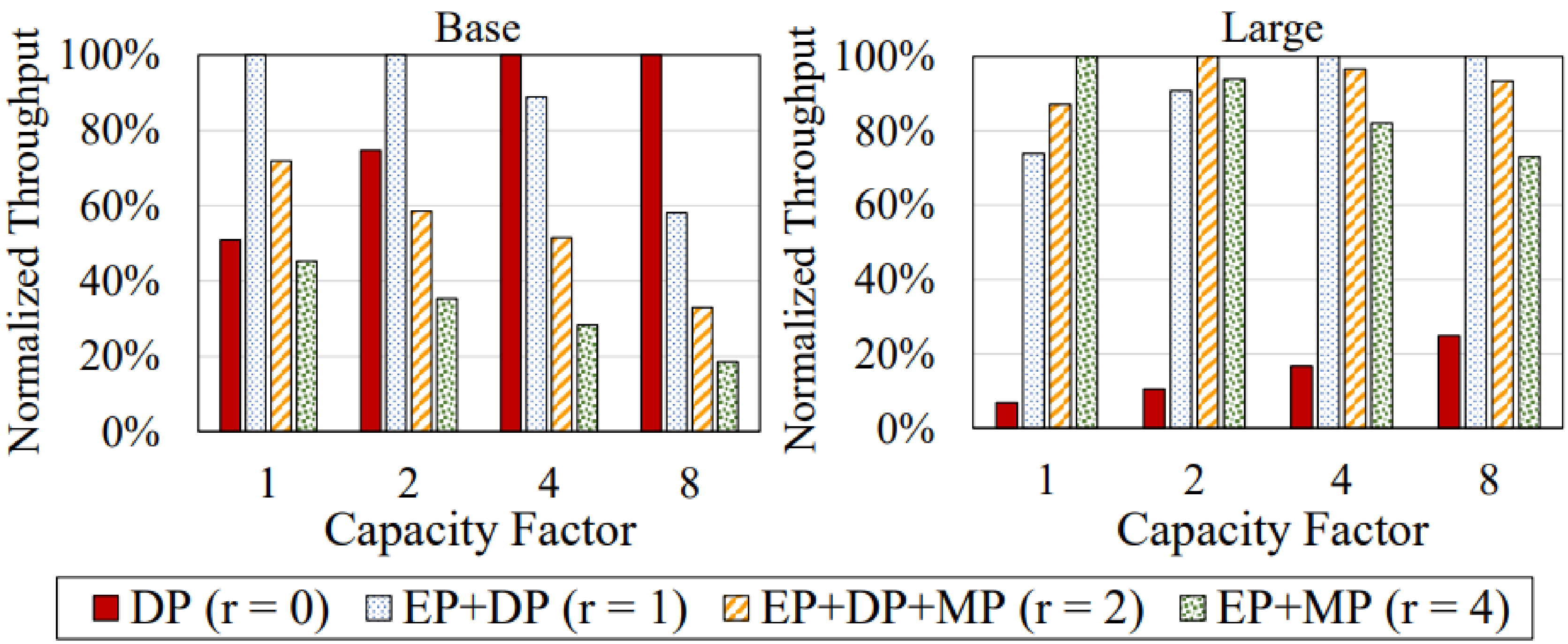
Switchable Parallelism

Path 2: Expert + Data + Model parallelism



Evaluation of Switchable Parallelism

- Multiple Parallelism Throughput on Different Capacity States



64 GPUs (A100) for 16 MoE Experts

(Larger Capacity Factor Implies Stronger Imbalance)

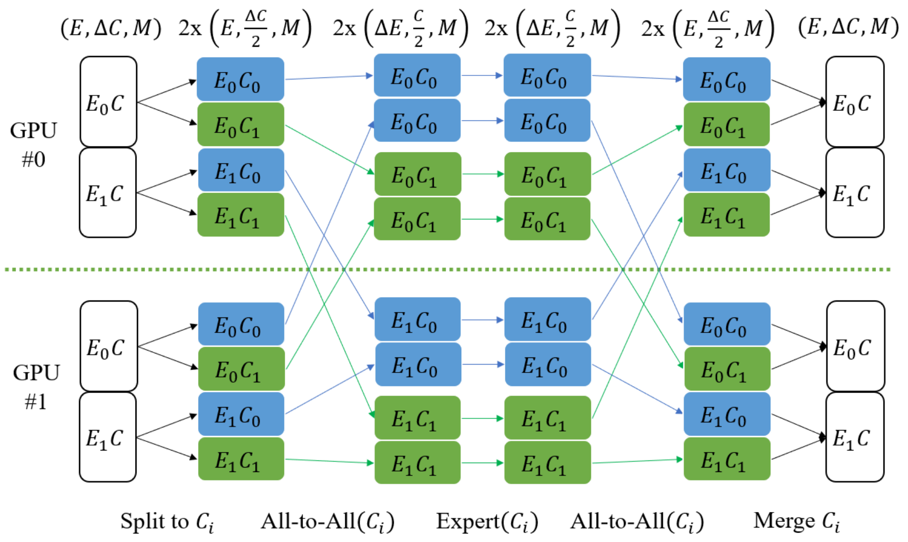
*Capacity factor is **monotonic decreasing** with r.*

Adaptive Pipelining

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

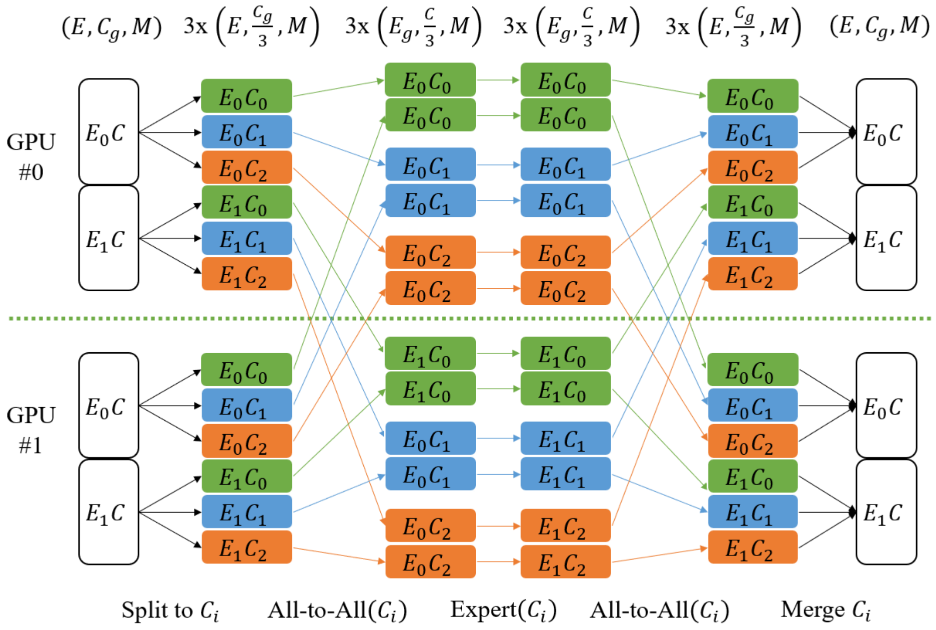
"MoE graph → multiple subgraph"

	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



Example of 2-expert pipelining with degree=2

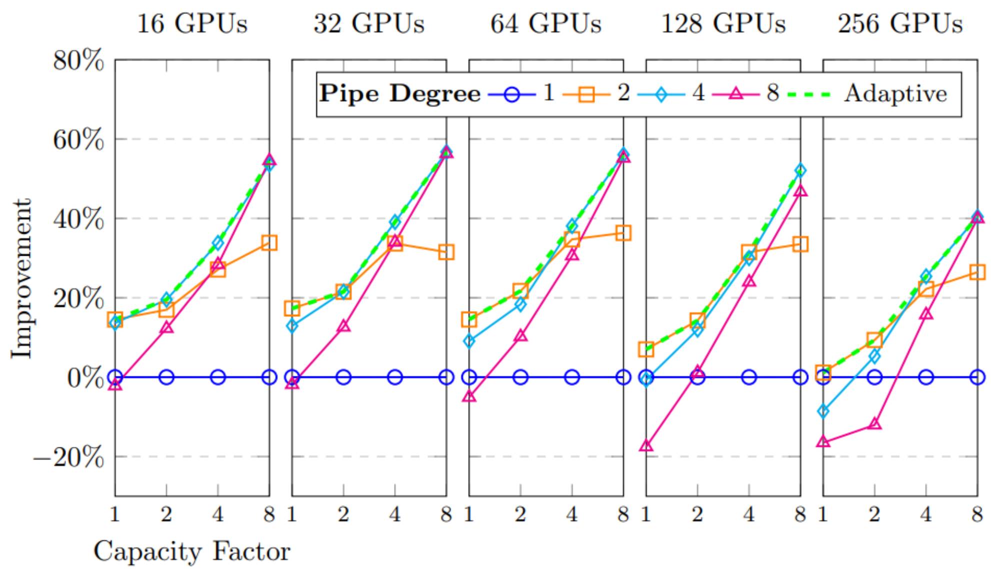
different colors are independent



Example of 2-expert pipelining with degree=3

Evaluation of Adaptive Pipelining

- Efficiency of Pipeline Degree on Different Capacity States



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

(Larger Capacity Factor Implies Less Balanced)

Optimal Degree Selection is more random, however:

① *Small Pipeline degree*: Not take advantage of overlap.

② *Large Pipeline degree*: Overhead of small-slice execution.

Combined Example to Select Optimal Parallel Options:

dict	1.00	1.01	…	4.10	…	8.00
value	r=2, o=1	r=2, o=1	…	r=2, o=2	…	r=1, o=4

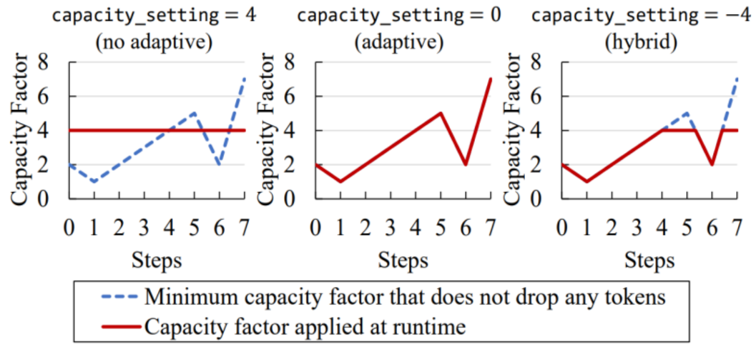
```
# Training Time:

for step_id, input_xs, .. in data_loader(..):
 cap_factor, .. = tutel_moe.top_k_routing(input_xs, 1)
 tutel_moe.forward(.., adaptive_r=dict[cap_factor].r,
                       a2a_ffn_overlap_degree=dict[cap_factor].o)
```

3 Extra Adaptive Mechanisms or Optimizations

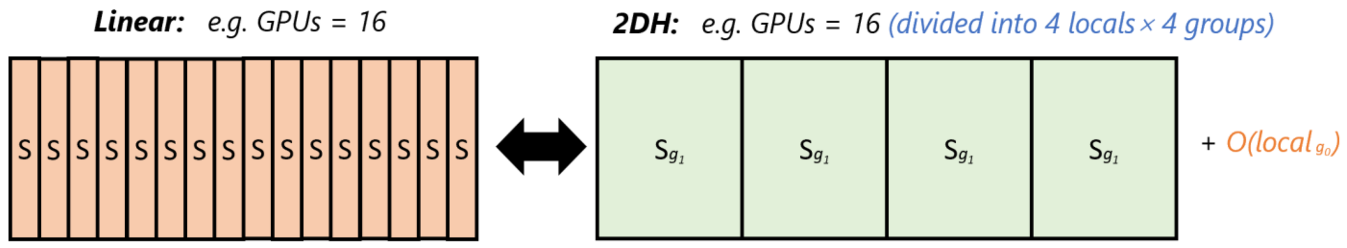
① Dynamic sparsity of Top-K & capacity controls (all “switchable”);

② Adaptive All-to-All algo. for different scales (Linear/2DH + Flexible);

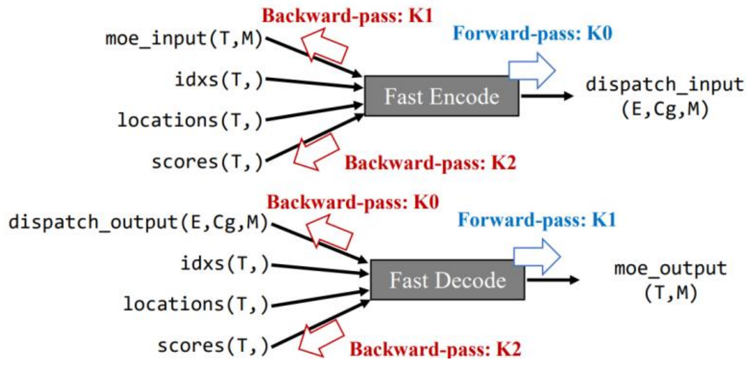
③ Deeply fused ops for “Fast Encode” and “Fast Decode” (90%↓ mem);



① Different Modes to Adapt Capacity Load



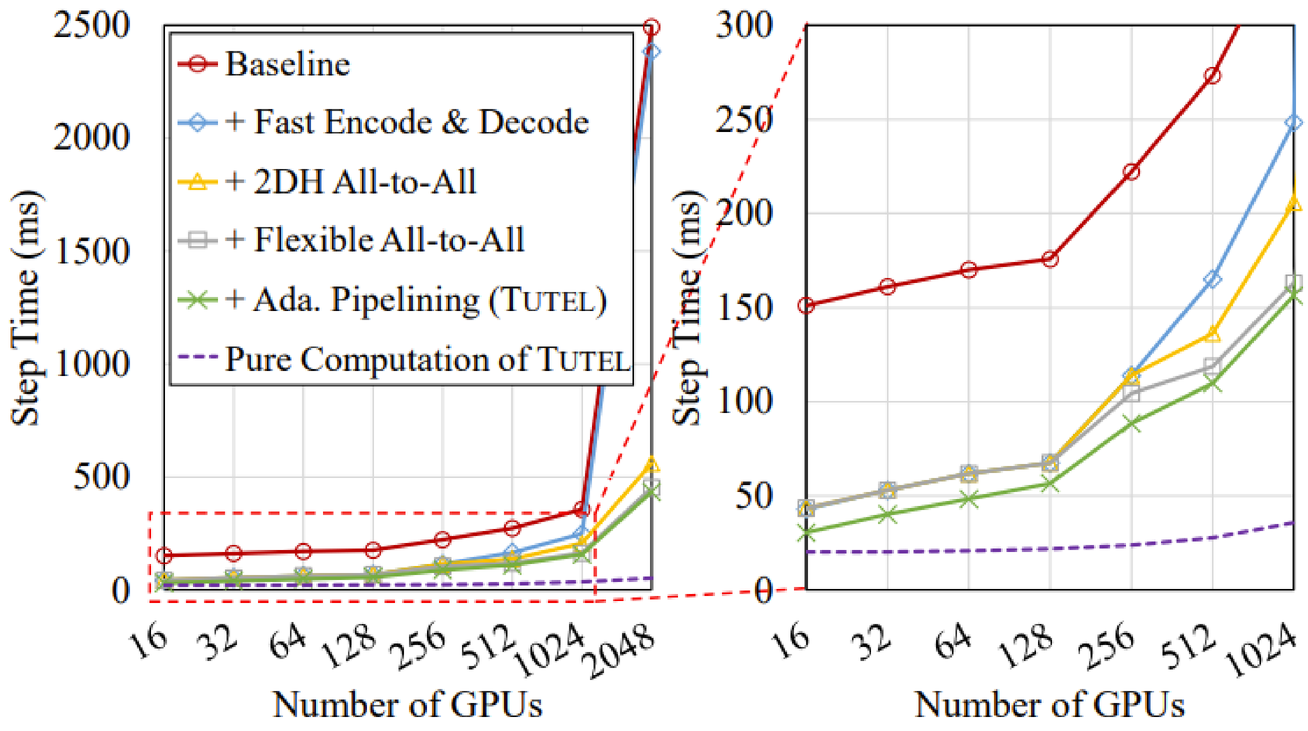
② Linear All2All (Left) for Small Scale 2DH All2All (Right) for Large Scale



③ Fused & Optimized Fast Encode and Decode

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

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



Single MoE layer Breakdown

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

Summary

1 **Adaptive**

The first MoE solution to design online parallelism modification, switch between different algorithm options and adapt across dynamic MoE workloads.

2 **At Scale**

Tutel [1] tackles non-scalable MoE, and achieves up to 5.75x speedup on 2,048 A100 in Azure.

3 **Deterministic Gains**

Tutel provides a gain with reproducible guarantee for different states of capacity. No predictors, no penalties and no math-inequivalence is involved, all of which may result in more harm against static. (throughput. & acc.)

Tutel [1]: *https://github.com/microsoft/tutel*

Thank you