FSMoE: A Flexible and Scalable Training System for Sparse Mixture-of-Experts Models

黄霄童

2025年4月

目录

目录

Basic Information

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

- Conference/Journal: *ACM ASPLOS 2025*
- Paper link: https://doi.org/10.48550/arXiv.2501.10714

FSMoE: A Flexible and Scalable Training System for Sparse Mixture-of-Experts Models

Xinglin Pan*
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China
xpan413@connect.hkust-gz.edu.cn

Wenxiang Lin*
Harbin Institute of Technology, Shenzhen
Shenzhen, China
wenxianglin@stu.hit.edu.cn

Lin Zhang
Hong Kong University of Science and Technology
Hong Kong SAR, China
lzhangbv@connect.ust.hk

Shaohuai Shi
Harbin Institute of Technology, Shenzhen
Shenzhen, China
shaohuais@hit.edu.cn

Zhenheng Tang
The Hong Kong University of Science and Technology
Hong Kong SAR, China
zhtang.ml@ust.hk

Rui Wang
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China
rwang132@connect.hkust-gz.edu.cn

Bo Li
Hong Kong University of Science and Technology
Hong Kong SAR, China
bli@cse.ust.hk

Xiaowen Chu†
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China
xwchu@hkust-gz.edu.cn

SJTU

目录

Background: What is MoE ?

- **Gate:** assign tokens to specific experts.
- **Order:** transforms the input tensor layout before dispatched
- **Experts:** FFN layers

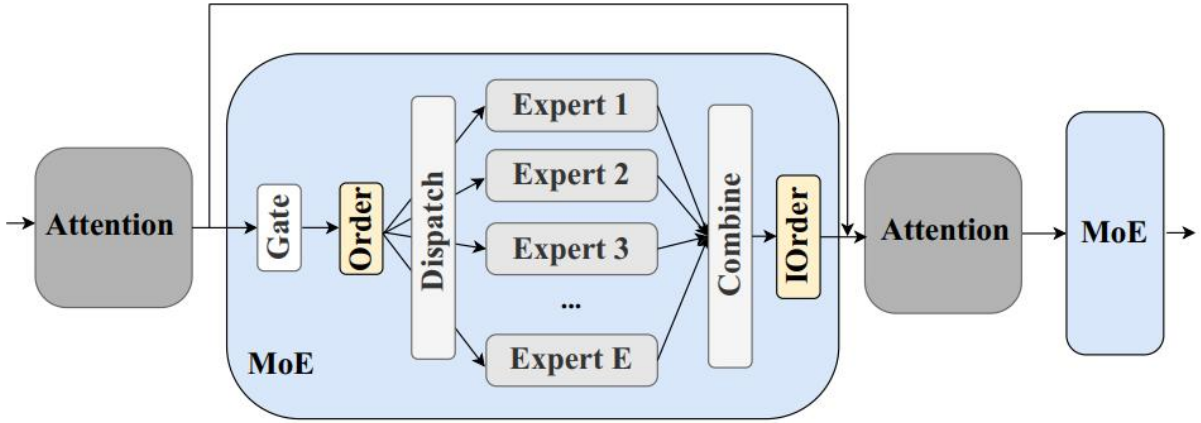
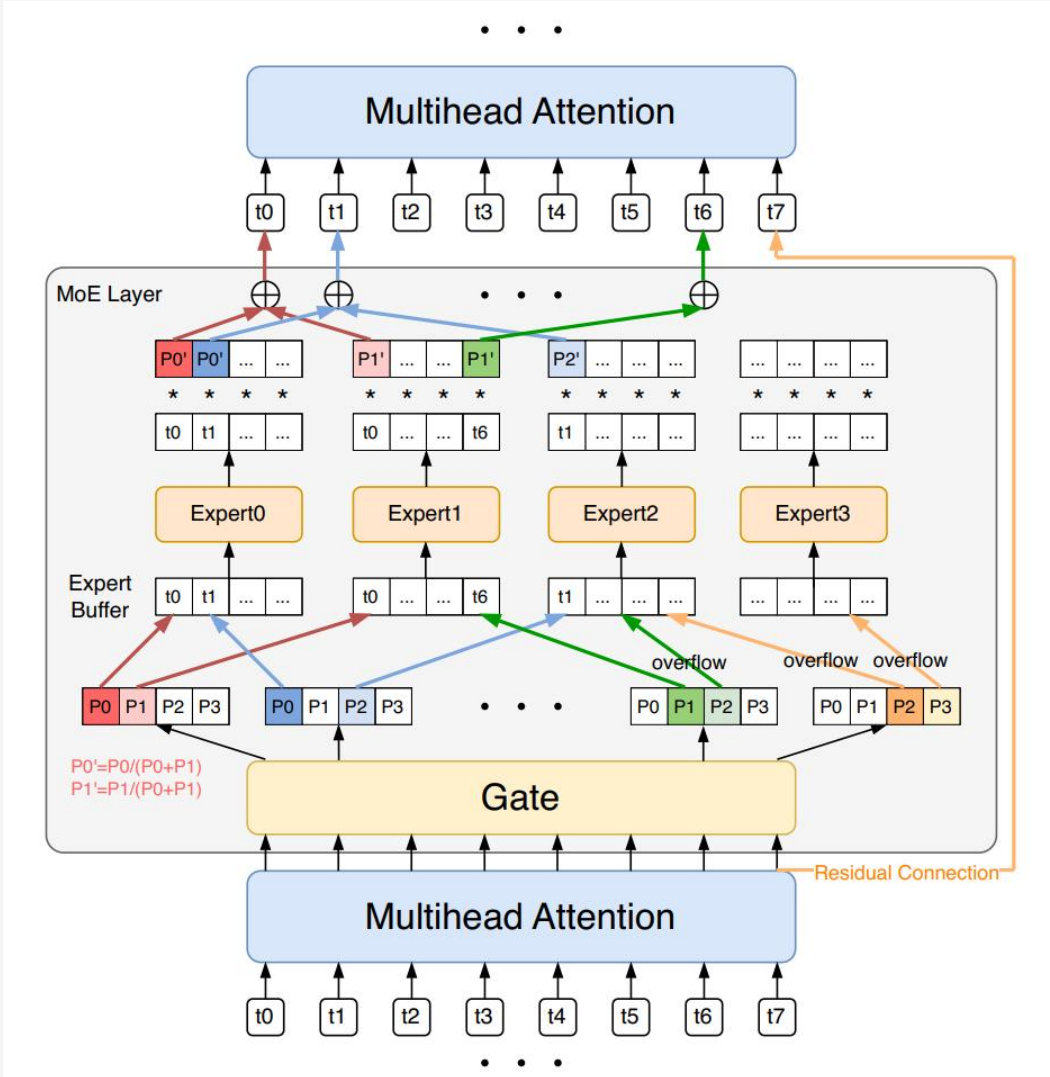


Figure 1. A typical MoE structure with E experts.



Background: Parallelism in MoE

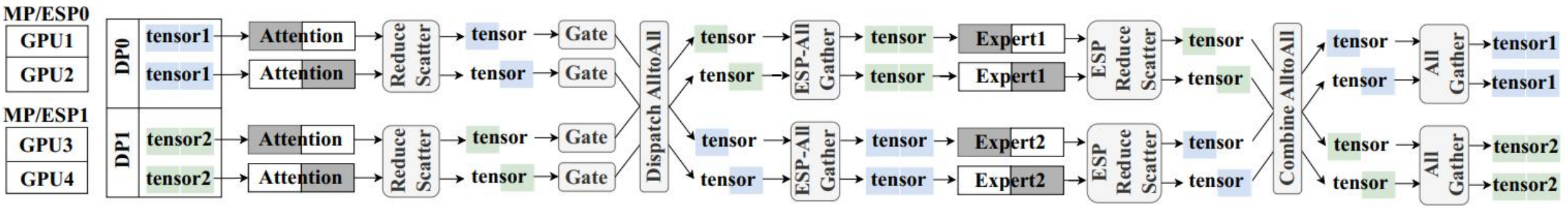


Figure 2. An example of $N_{DP} = N_{MP} = N_{EP} = N_{ESP} = 2$. The attention is partitioned into two parts across MP groups, and the two experts are distributed to the two EP groups (GPU1 and GPU3, as well as GPU2 and GPU4) in EP, and each expert is further partitioned into two shards across the ESP group. The blue and green rectangles indicate the data tensors.

- When the group of TP and ESP is aligned with the number of GPUs in a node:
  - ESP-AllGather and ESP-ReduceScatter: intra-node communications
  - AlltoAll Dispatch/Combine and Gradient-AllReduce: inter-node communications

Background: Communication Overhead

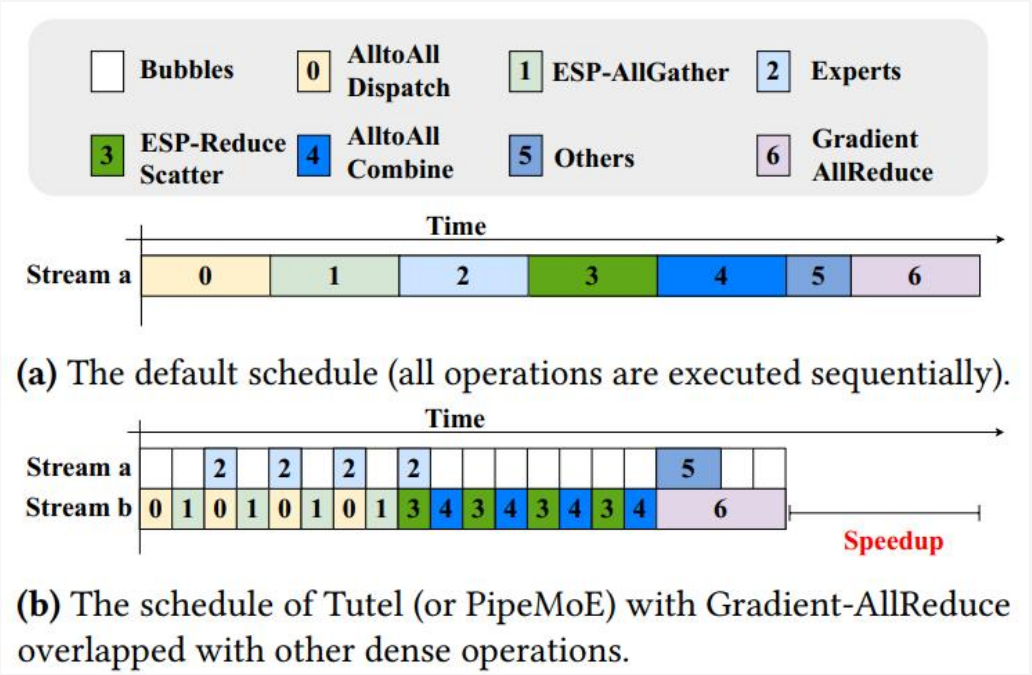
- Communication overhead typically contributes over 50% to the overall training.

Table 2. Time performance (iteration time in millisecond) of each operation in a transformer layer of two real-world models, GPT2-XL [38] and Mixtral7B [20], with $B = 4$ and $L = 1024$ for two testbeds in Table 3. The numbers in the brackets represent each operation's portion of the forward and backward time.

Testbeds/Breakdown		Communication				Computation			
		AlltoAll	AllReduce	AllGather	ReduceScatter	Experts	Routing	Order	Attention
A	GPT2-Forward	6.9(31.16%)	0(0%)	4.6(20.83%)	5.4(24.46%)	3.1(14.04%)	0.1(0.45%)	0.3(1.36%)	1.7(7.7%)
	GPT2-Backward	6.9(21.27%)	5.26(16.26%)	4.6(14.22%)	5.4(16.7%)	6.1(18.86%)	0.1(0.31%)	0.4(1.24%)	3.6(11.13%)
	Mixtral-Forward	19.5(29.8%)	0(0%)	12.3(18.73%)	13.7(20.86%)	15.6(23.76%)	0.1(0.15%)	0.3(0.46%)	4.1(6.24%)
	Mixtral-Backward	19.6(17.45%)	26.45(23.59%)	12.3(10.97%)	13.7(12.22%)	31.8(28.36%)	0.1(0.09%)	0.5(0.45%)	7.7(6.87%)
B	GPT2-Forward	11.2(20.7%)	0(0.0%)	15.5(28.7%)	15.7(29.1%)	6.7(12.4%)	0.1(0.2%)	0.3(0.6%)	4.5(8.3%)
	GPT2-Backward	11.2(15.7%)	7.3(10.3%)	15.5(21.8%)	15.2(21.3%)	13(18.3%)	0.1(0.1%)	0.3(0.4%)	8.6(12.1%)
	Mixtral-Forward	28.3(15.9%)	0.0(0.0%)	39.6(22.3%)	40.8(23.0%)	58.5(33.0%)	0.1(0.1%)	0.7(0.4%)	9.5(5.4%)
	Mixtral-Backward	30.8(10.8%)	32.1(11.3%)	40.1(14.1%)	41.8(14.7%)	119.7(42.1%)	0.2(0.1%)	1.2(0.4%)	18.1(6.4%)

Motivations

Object: figure out the optimal pipeline degree and scheduling for MoE parallelism.



(a) The default schedule (all operations are executed sequentially).

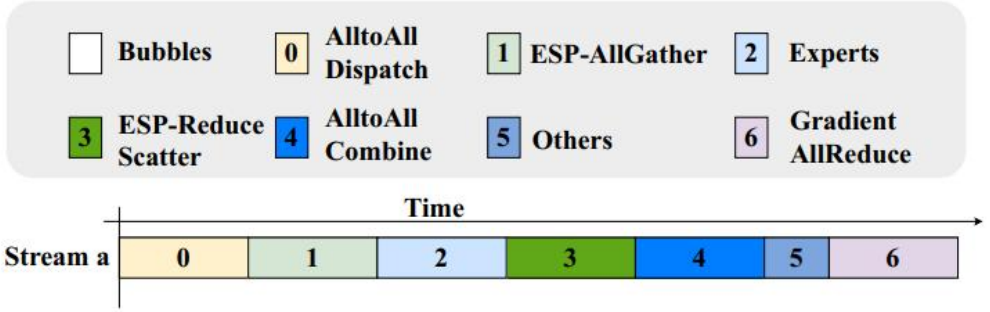
(b) The schedule of Tutel (or PipeMoE) with Gradient-AllReduce overlapped with other dense operations.

(c) Our proposed schedule FSMoE w/o partitioning the gradient.

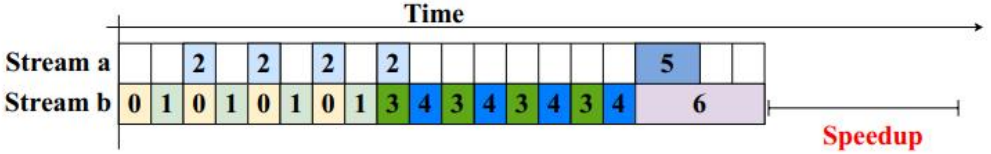
(d) Our proposed schedule FSMoE w/ partitioning the gradient.

Motivations

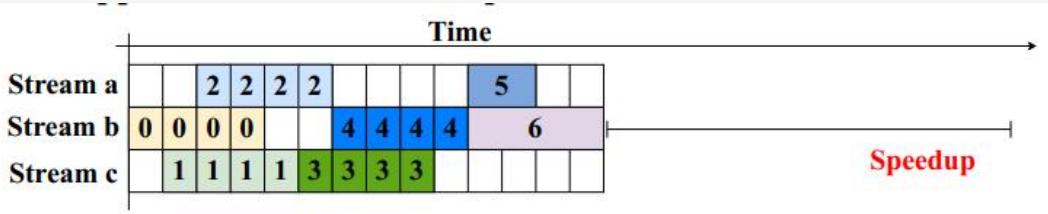
1. Explore the Overlapping Intra-node and Inter-node Communication.
2. Optimizing Forward and Backward Separately.
3. Co-Design in Backward Propagation and Gradient Synchronization.



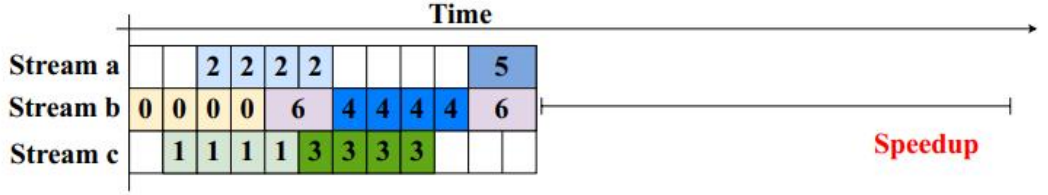
(a) The default schedule (all operations are executed sequentially).



(b) The schedule of Tutel (or PipeMoE) with Gradient-AllReduce overlapped with other dense operations.



(c) Our proposed schedule FSMoE w/o partitioning the gradient.



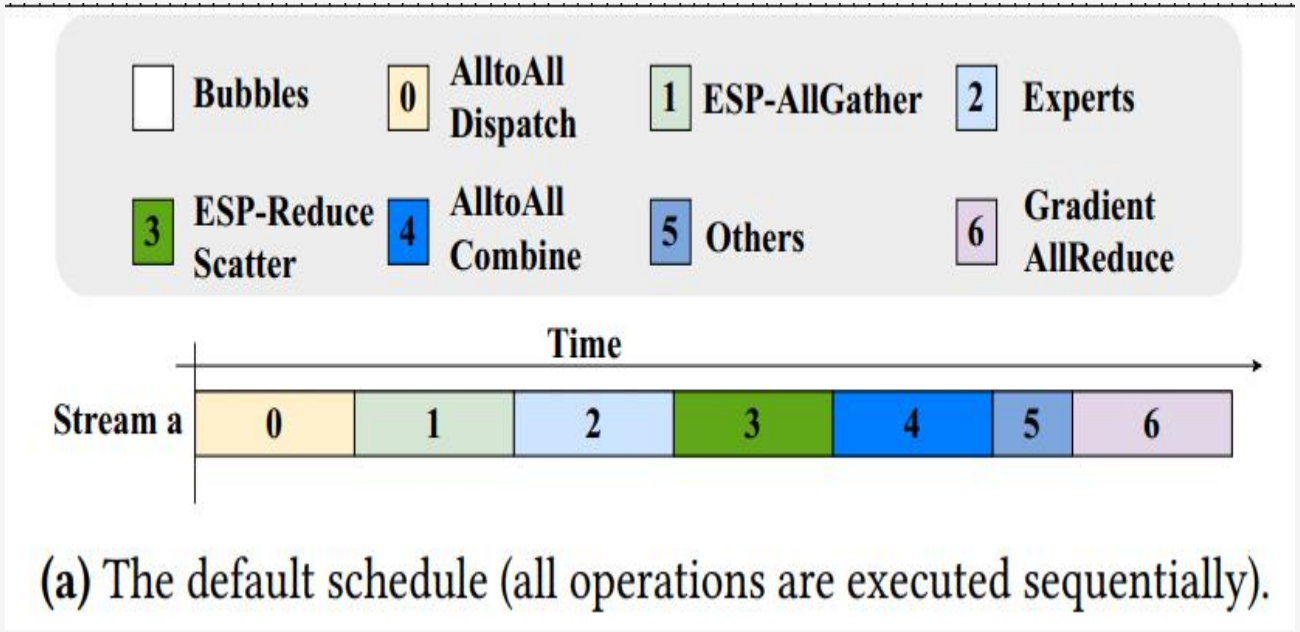
(d) Our proposed schedule FSMoE w/ partitioning the gradient.

目录

FSMoE: Performance Models

The time required for each chunk in the AlltoAll, AllGather, ReduceScatter, and expert computation processes on inputs divided into r chunks is represented by following equation respectively.

$$
\begin{aligned}
t_{a2a,r} &= \alpha_{a2a} + \frac{n_{a2a}}{r} \cdot \beta_{a2a}, \\
t_{ag,r} &= \alpha_{ag} + \frac{n_{ag}}{r} \cdot \beta_{ag}, \\
t_{rs,r} &= \alpha_{rs} + \frac{n_{rs}}{r} \cdot \beta_{rs}, \\
t_{exp,r} &= \alpha_{exp} + \frac{n_{exp}}{r} \cdot \beta_{exp},
\end{aligned}
$$



(a) The default schedule (all operations are executed sequentially).

FSMoE: Optimize the Pipeline Degree

Then, this paper classify all general cases into four scenarios.

- Case1:

$$t_1^{moe} = 2r \cdot t_{a2a,r} + t_{gar} = 2r\alpha_{a2a} + 2n_{a2a}\beta_{a2a} + t_{gar}.$$

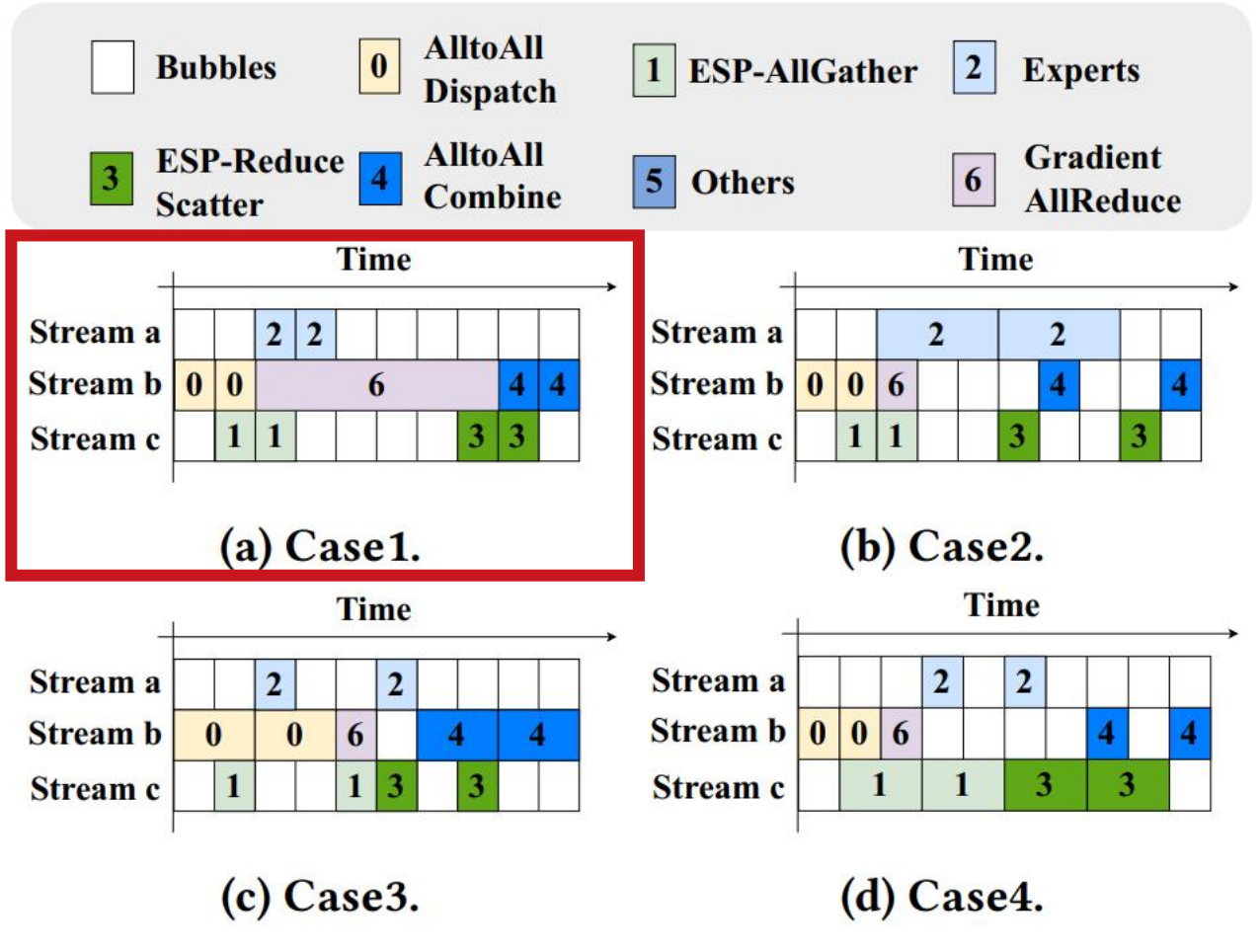
Therefore, to find its minima, t_1^*, we should solve

$$\text{minimize: } f_1(r) = t_1^{moe},$$

$$\text{s.t. } r \geq 1,$$

$$(Q1 \wedge \neg Q2 \wedge Q4) \vee (Q1 \wedge Q2 \wedge Q5)$$

$$\vee (\neg Q1 \wedge \neg Q3 \wedge Q6) \vee (\neg Q1 \wedge Q3 \wedge Q7).$$



FSMoE: Optimize the Pipeline Degree

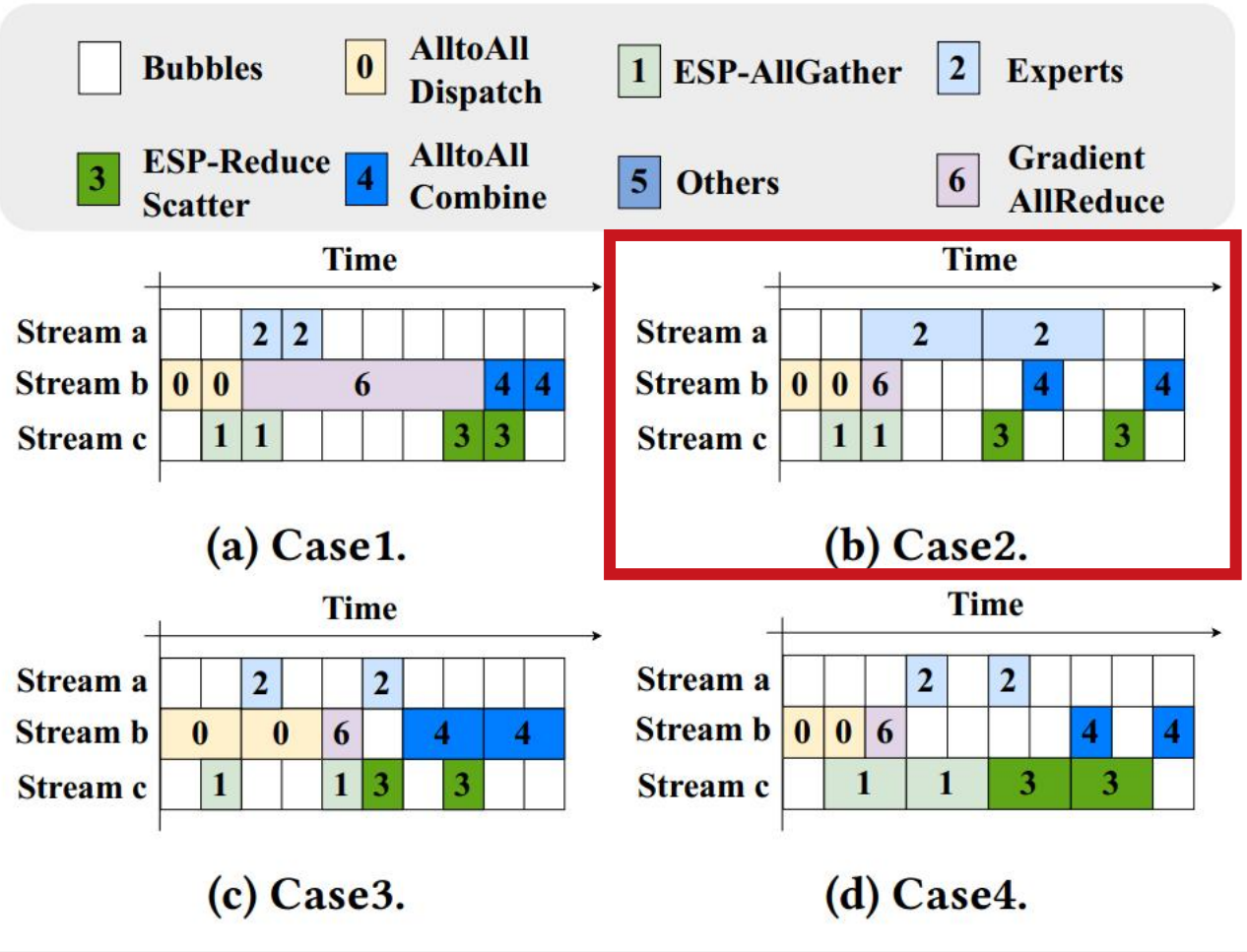
Then, this paper classify all general cases into four scenoris.

- Case2:

$$
\begin{aligned}
t_2^{moe} &= 2t_{a2a,r} + t_{ag,r} + t_{rs,r} + r \cdot t_{exp,r} \\
&= 2\alpha_{a2a} + \frac{2n_{a2a}}{r}\beta_{a2a} + \alpha_{ag} + \frac{n_{ag}}{r}\beta_{ag} \\
&\quad + \alpha_{rs} + \frac{n_{rs}}{r}\beta_{rs} + r\alpha_{exp} + n_{exp}\beta_{exp}.
\end{aligned}
$$

Therefore, to find its minima, t_2^*, we should solve

$$
\begin{aligned}
&\text{minimize: } f_2(r) = t_2^{moe}, \\
&\text{s.t. } r \geq 1, \\
&(Q1 \wedge Q2 \wedge \neg Q5) \vee (\neg Q1 \wedge Q3 \wedge \neg Q7).
\end{aligned}
$$



FSMoE: Optimize the Pipeline Degree

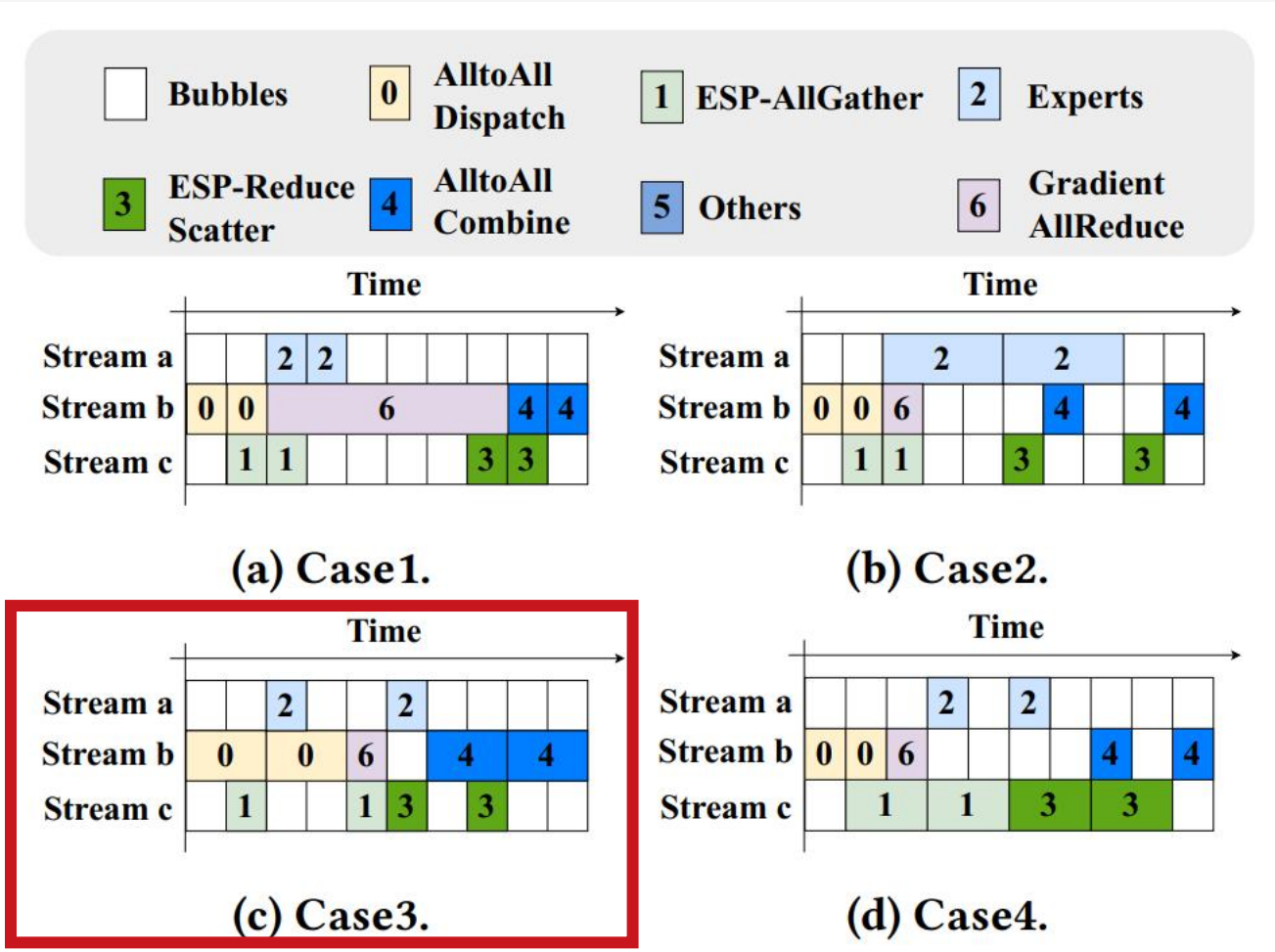
Then, this paper classify all general cases into four scenoris.

- Case3:

$$t_3^{moe} = 2r \cdot t_{a2a,r} + t_{ag,r} + t_{rs,r}$$
$$= 2r\alpha_{a2a} + 2n_{a2a}\beta_{a2a} + \alpha_{ag} + \frac{n_{ag}}{r}\beta_{ag} + \alpha_{rs} + \frac{n_{rs}}{r}\beta_{rs}.$$

Therefore, to find its minima, t_3^*, we should solve

$$\text{minimize: } f_3(r) = t_3^{moe},$$
$$\text{s.t. } r \geq 1,$$
$$Q1 \wedge \neg Q2 \wedge \neg Q4.$$



FSMoE: Optimize the Pipeline Degree

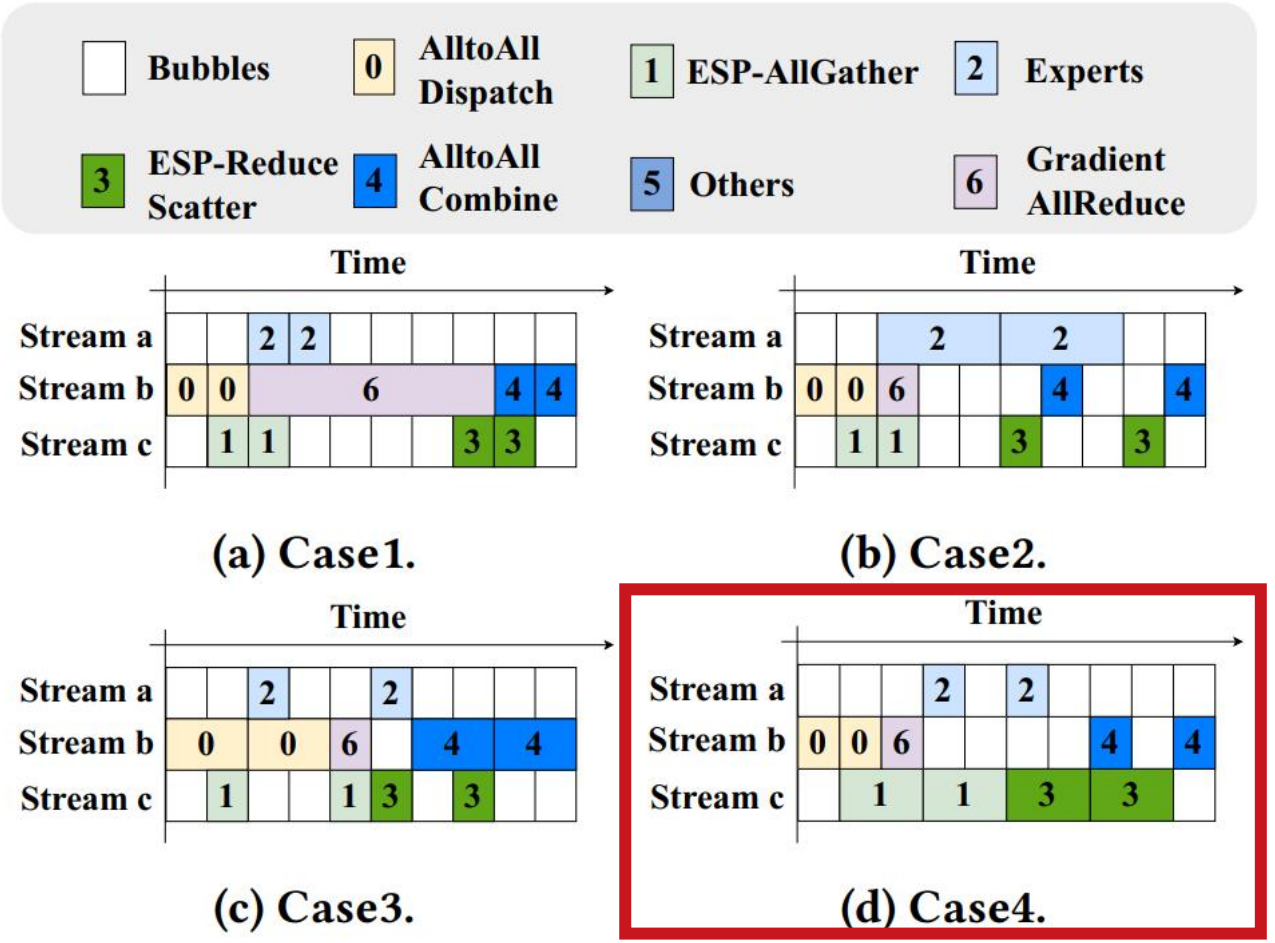
Then, this paper classify all general cases into four scenoris.

- Case4:

$$t_4^{moe} = 2t_{a2a,r} + r \cdot t_{ag,r} + r \cdot t_{rs,r}$$
$$= 2\alpha_{a2a} + \frac{2n_{a2a}}{r}\beta_{a2a} + r\alpha_{ag} + n_{ag}\beta_{ag} + r\alpha_{rs} + n_{rs}\beta_{rs}.$$

Therefore, to find its minima, t_4^*, we should solve

$$\text{minimize: } f_4(r) = t_4^{moe},$$
$$\text{s.t. } r \geq 1,$$
$$\neg Q1 \wedge \neg Q3 \wedge \neg Q6.$$



FSMoE：Optimize the Pipeline Degree

- FSMoE supports **varied pipeline degrees** in both phases. The algorithm executes once before training, following the estimation of cluster-related coefficients.

Algorithm 1 FindOptimalPipelineDegree

Input: $\alpha_{a2a}, \beta_{a2a}, n_{a2a}, \alpha_{ag}, \beta_{ag}, n_{ag}, \alpha_{rs}, \beta_{rs}, n_{rs}, \alpha_{exp}, \beta_{exp}, n_{exp}$ t_{gar}

Output: r and t^{moe}

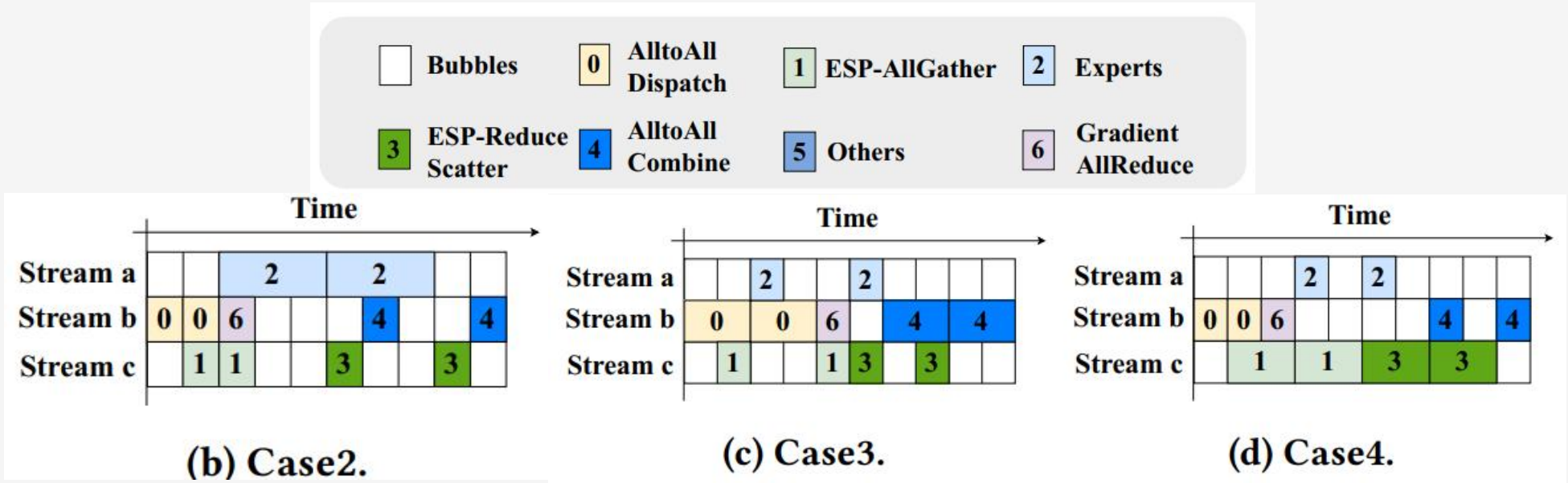
1: $r1, t1 = solve(f_1)$ ▷ Solve with SLSQP
2: $r2, t2 = solve(f_2)$
3: $r3, t3 = solve(f_3)$
4: $r4, t4 = solve(f_4)$
5: candidate_mins = $[t1, t2, t3, t4]$
6: candidates = $[r1, r2, r3, r4]$
7: r = candidates[argmin(candidate_mins)]
8: $t^{moe} = min$(candidate_mins)
9: return r and t^{moe}.

FSMoE: Scheduling for Backpropagation

- Performance Model:

$$t_{ar}(n_{ar}) = \alpha_{ar} + n_{ar} \cdot \beta_{ar}, \qquad g^{inv}_{grad}(t_{ar}) = (t_{ar} - \alpha_{ar})/\beta_{ar}.$$

- Scheduling approach for backpropagation:
  - Step 1: Calculate the time cost of overlappable parts. (with t_gar = 0)



FSMoE: Scheduling for Backpropagation

- Performance Model:

$$t_{ar}(n_{ar}) = \alpha_{ar} + n_{ar} \cdot \beta_{ar}, \qquad g^{inv}_{grad}(t_{ar}) = (t_{ar} - \alpha_{ar})/\beta_{ar}.$$

- Scheduling approach for backpropagation:
  - Step 1: And try to **slice and assign** the gradient to these overlappable parts.

$$n^i_{first} = g^{inv}_{grad}(\min(t_{grad}(n^{i-1}_{grad}), t^i_{olp})). \quad (3)$$

If n^{i-1}_{grad} is not fully overlapped, n^i_{grad} should be updated by

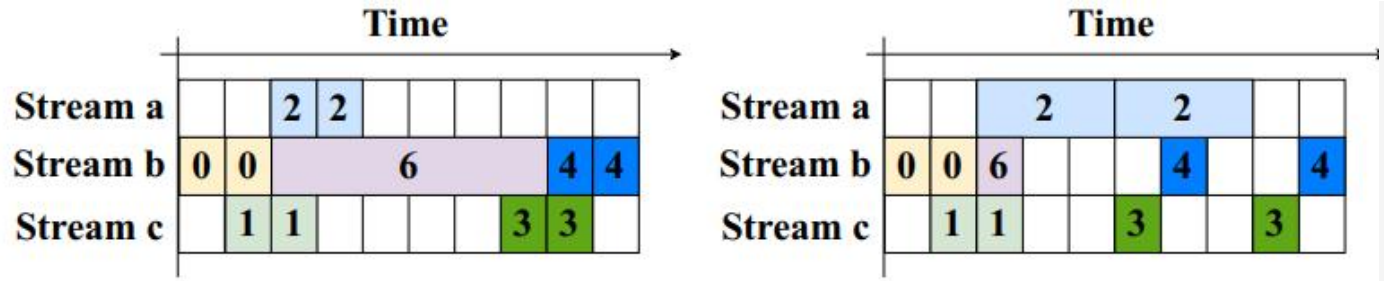
$$n^i_{grad} = n^i_{grad} + g^{inv}_{grad}(\max(t_{grad}(n^{i-1}_{grad}) - t^i_{olp}, 0)). \quad (4)$$

FSMoE: Scheduling for Backpropagation

- Scheduling approach for backpropagation:
  - Step 2: Assign the remaining gradients after the first step.

$$\text{minimize: } f_g(X_g) = \sum_{i=1}^{n_l} f_{moe}^i \left(t_{grad}(x_g^i) \right),$$

$$\text{s.t. } 0 \le x_g^i < n_{rem}^i + \sum_{j=i-1}^{n_l} (n_{rem}^j - x_{gar}^j), 0 < i < n_l, \quad (5)$$



(a) Case1. (b) Case2.

FSMoE: Scheduling for Backpropagation

- Objective function:

```
188
189         def obj_func(xs):
190             outv = 0
191             sumx = 0
192             for item in xs:
193                 sumx += item
194                 if self.gap > self.alpha[4]:
195                     _, v = self.optimize_degree(self.gap + self.beta[4] * (item))
196                 else:
197                     _, v = self.optimize_degree(self.gar_time * (item))
198                 outv += v
199             outv += (sumup - sumx) * self.beta[4]
200             return outv
```

目录

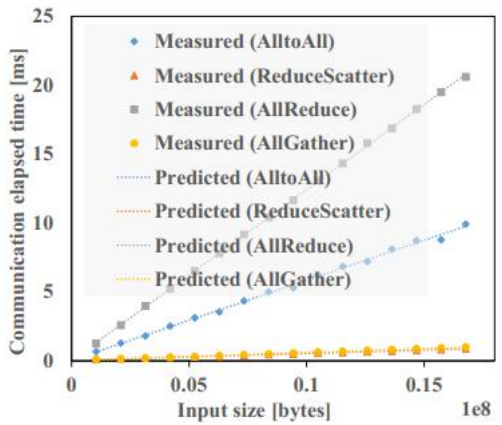
FSMoE: Server configurations

Table 3. The server configurations in our testbeds.

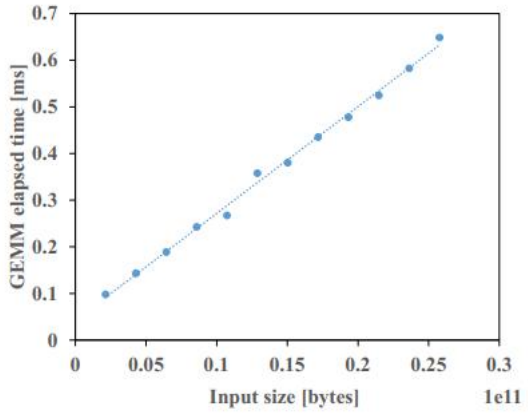
Name	Testbed A	Testbed B
CPU	Dual Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60GHz	Dual Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz
GPU	8x Nvidia RTXA6000 @1.46GHz 48GB Mem	4x Nvidia RTX2080Ti @1.35GHz 11GB Memory
Memory	512GB DDR4	512GB DDR4
NVlink	112.5GB/s (4x)	-
PCIe	4.0 (x16)	3.0 (x16)
Network	Mellanox MT28908 @ 200Gb/s	Mellanox MT27800 @ 100Gb/s

Testbeds: Experiments are carried out on two distinct testbeds: Testbed-A, a 48-GPU cluster comprising six interconnected nodes, and each node is equipped with four Nvidia A6000 GPUs. Testbed-B, a 32-GPU cluster comprising eight interconnected nodes, and each node is equipped with four Nvidia GeForce RTX2080Ti GPUs. More details on the server configuration can be found in Table 3. The software environments are Ubuntu-20.04, CUDA-11.3, PyTorch-1.12 and NCCL-2.12.

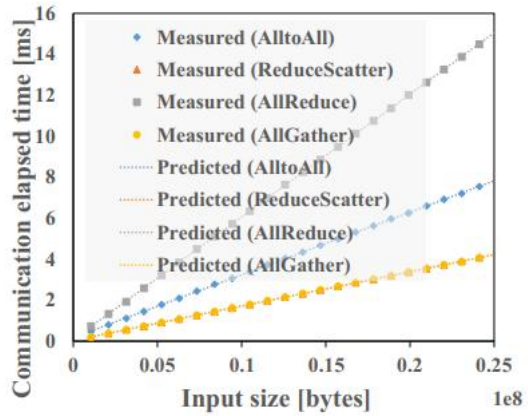
FSMoE: Performance Model Evaluation



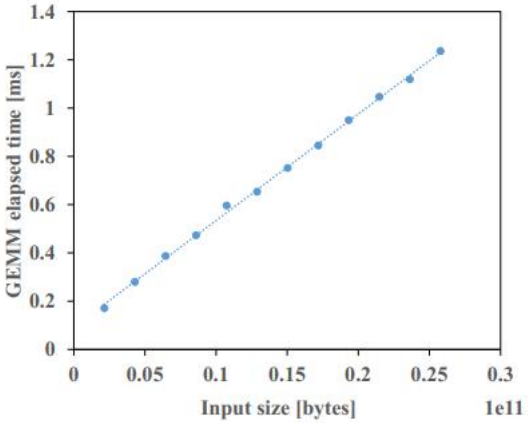
(a) Communication (A6000).



(b) GEMM (A6000).



(c) Communication (2080Ti).



(d) GEMM (2080Ti).

- Markers are measured values and lines are predicted values with estimated parameters.

FSMoE: Performance Model Evaluation

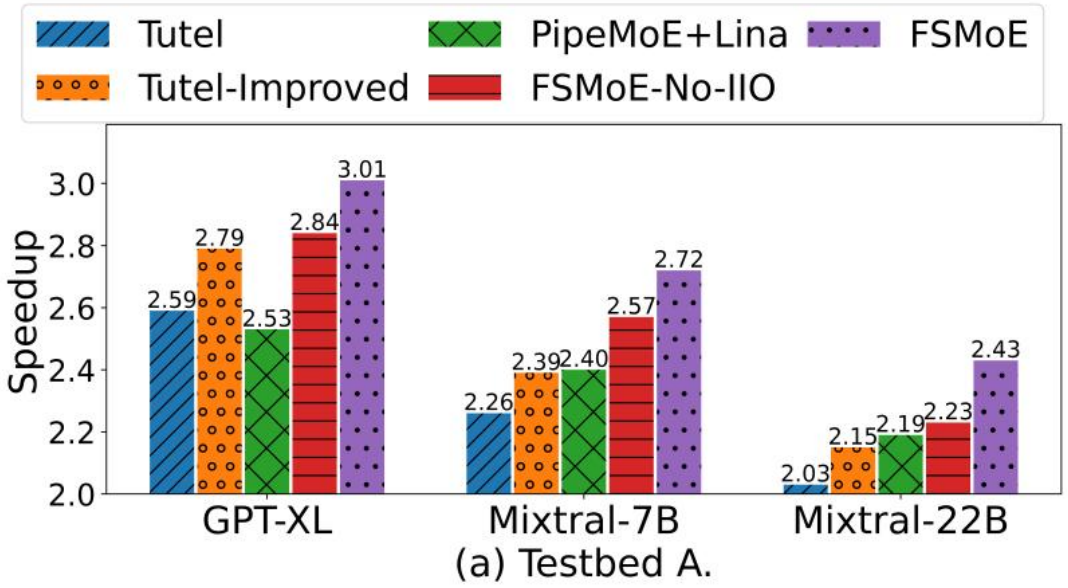
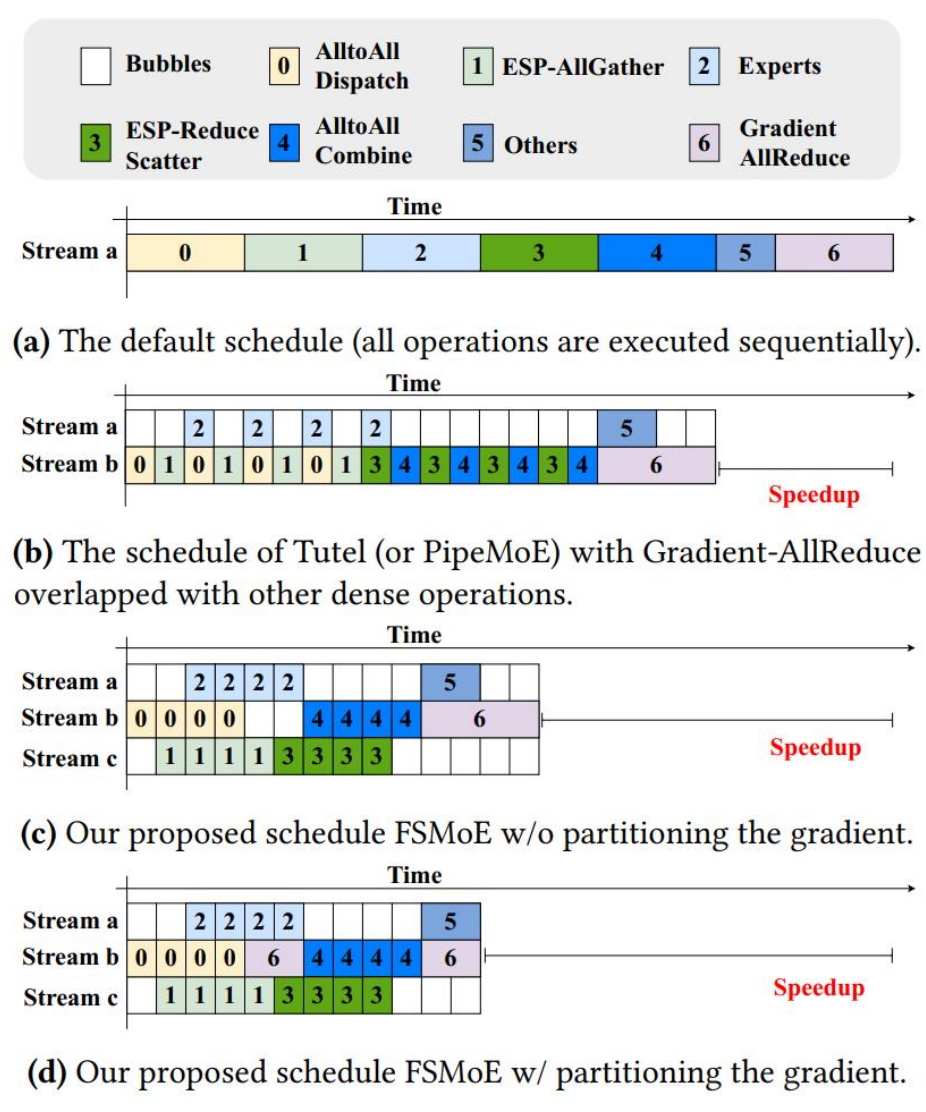


Figure 6. Speedups of FSMoE, FSMoE-No-IIO, Tutel, Tutel-Improved, PipeMoE+Lina (PipeMoE with the additional schedule introduced by Lina [24] that partitions the gradient into fixed chunk size) over DeepSpeed-MoE (DS-MoE) on MoE models (GPT2-XL, Mixtral-7B and Mixtral-22B).

目录

FSMoE：Analysis

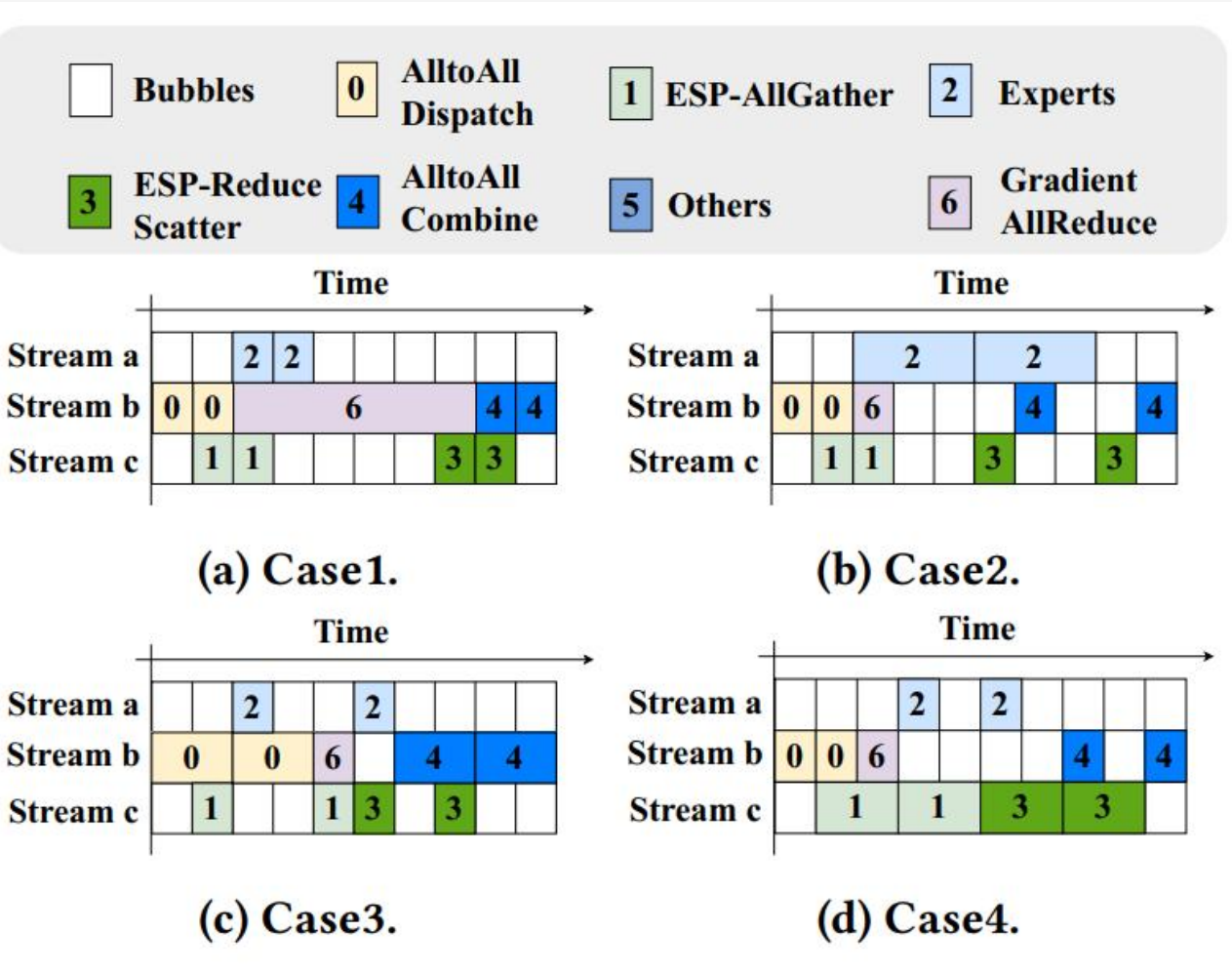


(a) The default schedule (all operations are executed sequentially).

(b) The schedule of Tutel (or PipeMoE) with Gradient-AllReduce overlapped with other dense operations.

(c) Our proposed schedule FSMoE w/o partitioning the gradient.

(d) Our proposed schedule FSMoE w/ partitioning the gradient.



目录

Q&A

上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

Thank You

饮 水 思 源　爱 国 荣 校