Accelerating Distributed MoE Training and Inference with Lina

Jiamin Li[1], Yimin Jiang[2], Yibo Zhu, Cong Wang[1], Hong Xu[2]

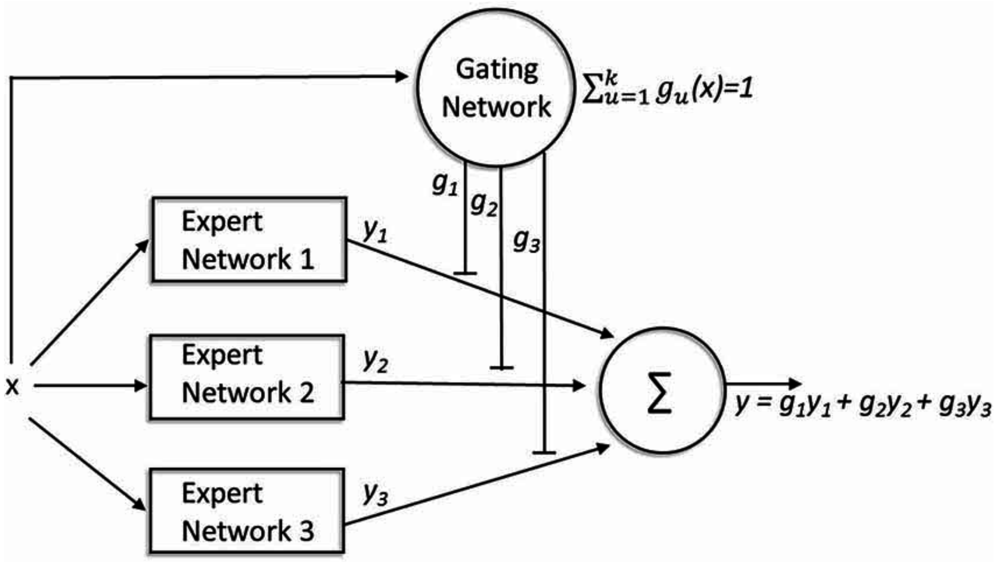
[1]City University of Hong Kong, [2]ByteDance Inc., [3]The Chinese University of Hong Kong

ATC 2023

香港城市大學
City University of Hong Kong

香港中文大學
The Chinese University of Hong Kong

Sparsely-Activated Mixture-of-Experts (MoE)

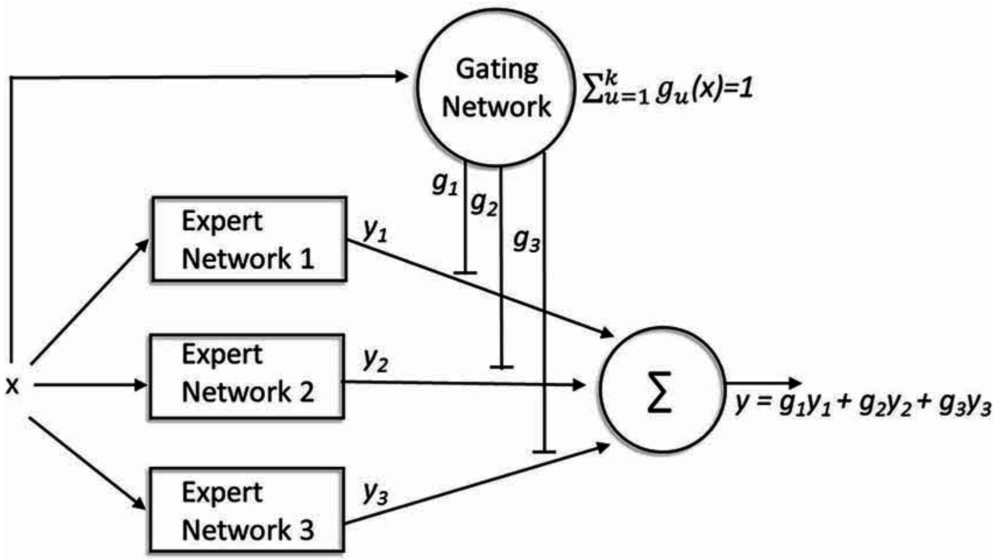


MoE architecture

An ensemble of experts.

Figure credit to Anatomical and Functional Plasticity in Early Blind Individuals and the Mixture of Experts Architecture

Sparsely-Activated Mixture-of-Experts (MoE)

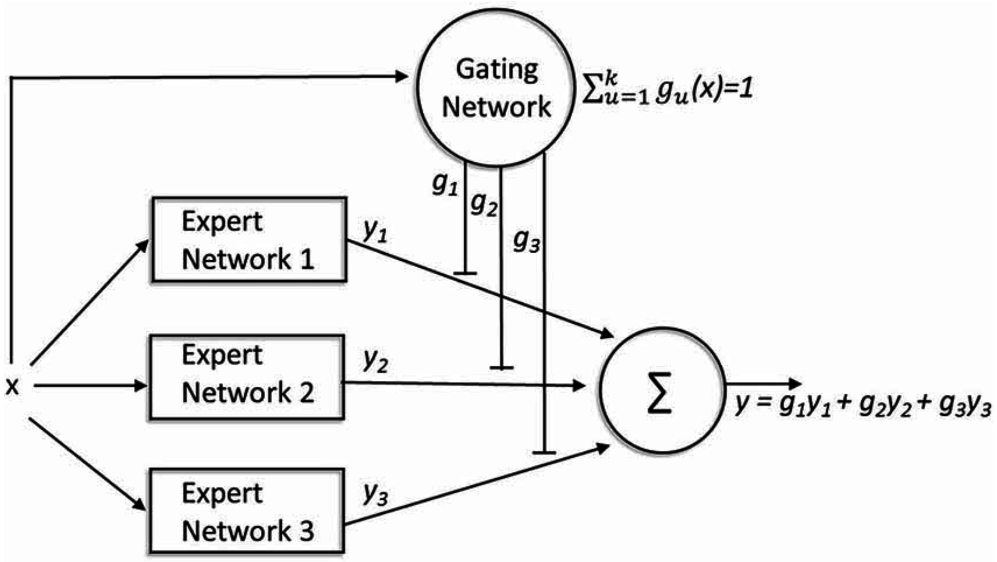


MoE architecture
An ensemble of experts.

- *Sparsely-activated* MoE: each input selects just a few (1 or 2) experts for processing
- Benefit: sub-linear scaling of FLOPS with model size

Figure credit to Anatomical and Functional Plasticity in Early Blind Individuals and the Mixture of Experts Architecture

Sparsely-Activated Mixture-of-Experts (MoE)



MoE architecture
An ensemble of experts.

- *Sparsely-activated* MoE: each input selects just a few (1 or 2) experts for processing
- Benefit: sub-linear scaling of FLOPS with model size

Massive model parameters with constant computation cost.

Figure credit to Anatomical and Functional Plasticity in Early Blind Individuals and the Mixture of Experts Architecture

Potential of MoE in Transformer Models

- GLaM by Google
  - GLaM outperforms GPT-3 on 29 tasks

		GPT-3	GLaM	relative
cost	FLOPs / token (G)	350	**180**	−48.6%
	Train energy (MWh)	1287	**456**	−64.6%
accuracy on average	Zero-shot	56.9	**62.7**	+10.2%
	One-shot	61.6	**65.5**	+6.3%
	Few-shot	65.2	**68.1**	+4.4%

- DeepSpeed MoE models
  - Model quality: 6.7B-parameter dense = 1.3B-parameter MoE - 128
  - Training compute reduction of 5x

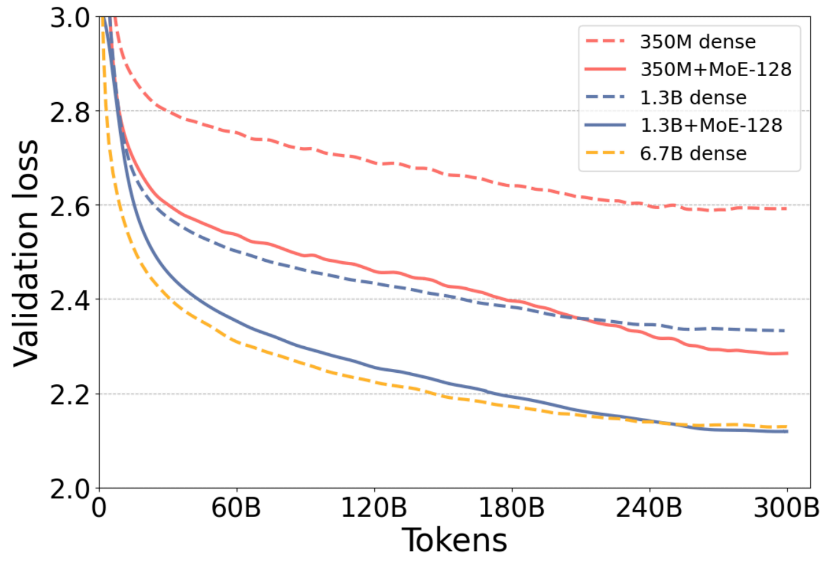


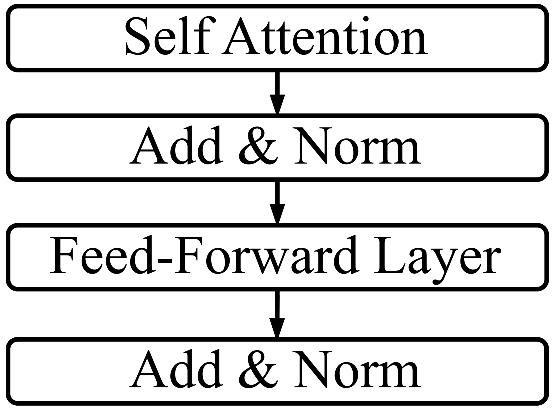
Figure credit to GLaM and DeepSpeed MoE.

MoE in Transformer-based Language Models

- Feed forward layers (FFN) are replaced with MoE layers.
- MoE layer = gate + experts
  - Expert: feed forward neural network

    => *same* architecture, *different* parameters
  - Gate: a *trainable* matrix to select expert for each data sample
    - Top-2 in training, Top-1 in inference

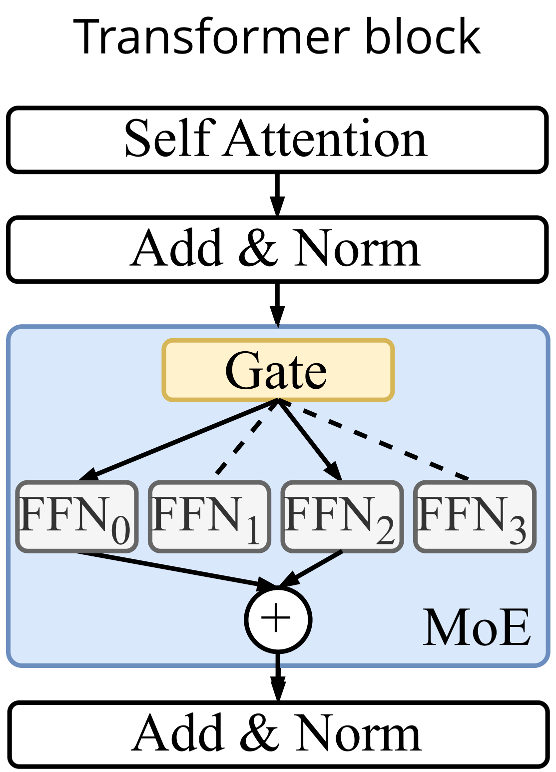
Transformer block



MoE in Transformer-based Language Models

- Feed forward layers (FFN) are replaced with MoE layers.
- MoE layer = gate + experts
  - Expert: feed forward neural network

    => *same* architecture, *different* parameters
  - Gate: a *trainable* matrix to select expert for each data sample
    - Top-2 in training, Top-1 in inference

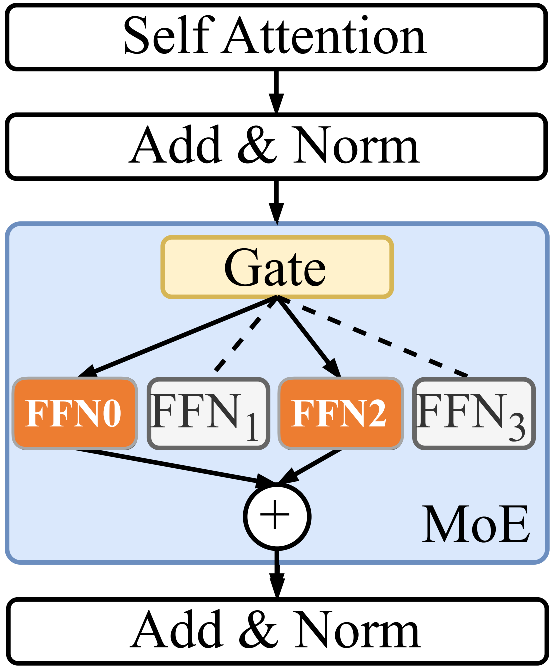


MoE in Transformer-based Language Models

- Feed forward layers (FFN) are replaced with MoE layers.
- MoE layer = gate + experts
  - Expert: feed forward neural network

    => <u>*same*</u> architecture, <u>*different*</u> parameters
  - Gate: a <u>*trainable*</u> matrix to select expert for each data sample
    - Top-2 in training, Top-1 in inference

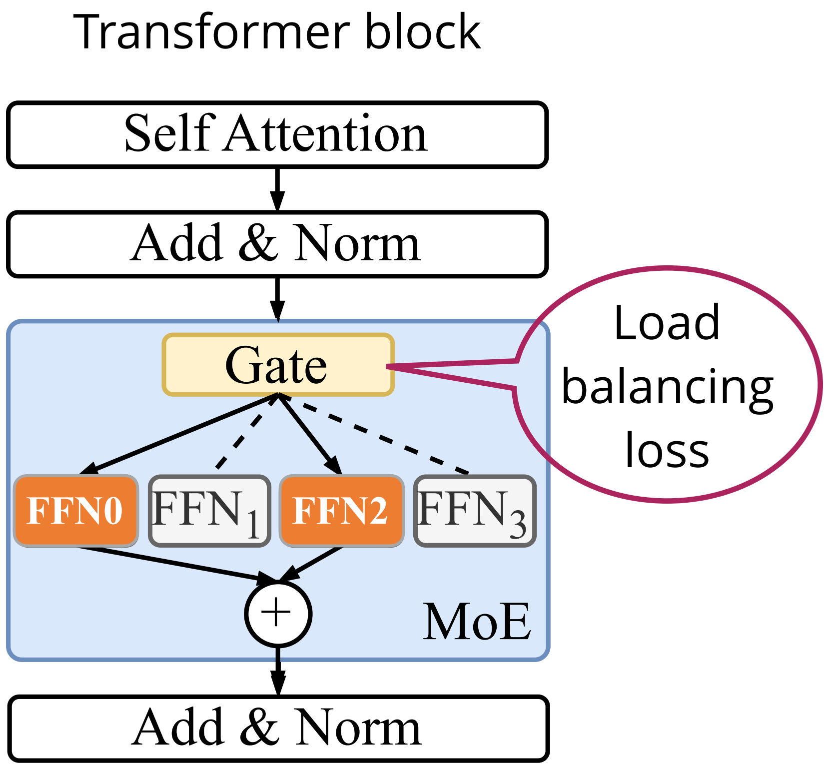
Transformer block



MoE in Transformer-based Language Models

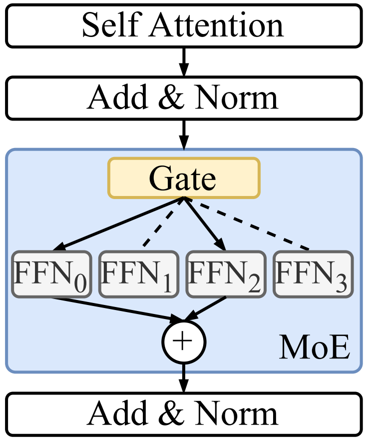
- Feed forward layers (FFN) are replaced with MoE layers.
- MoE layer = gate + experts
  - Expert: feed forward neural network

    => <u>*same*</u> architecture, <u>*different*</u> parameters
  - Gate: a <u>*trainable*</u> matrix to select expert for each data sample
    - Top-2 in training, Top-1 in inference
  - Load balancing loss during training: <u>*even*</u> distribution among experts.



Distributed MoE

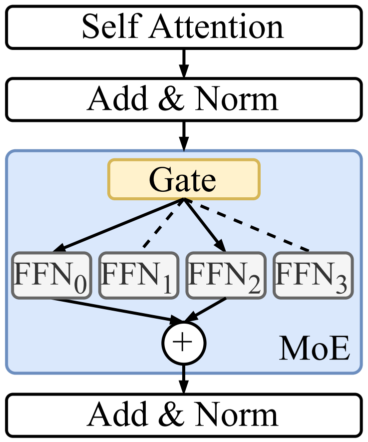
- Hybrid parallelism:
  - Expert parallelism: each device hosts one unique expert
  - Data parallelism: replicate non-expert parameter on each device



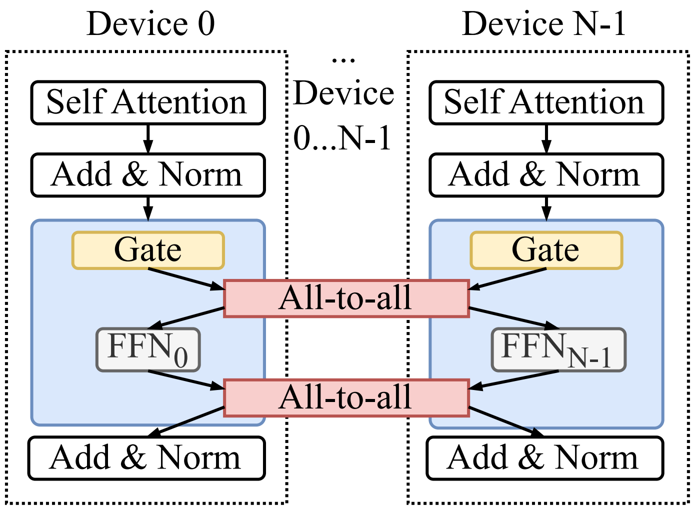
Single-device

Distributed MoE

- Hybrid parallelism:
  - Expert parallelism: each device hosts one unique expert
  - Data parallelism: replicate non-expert parameter on each device



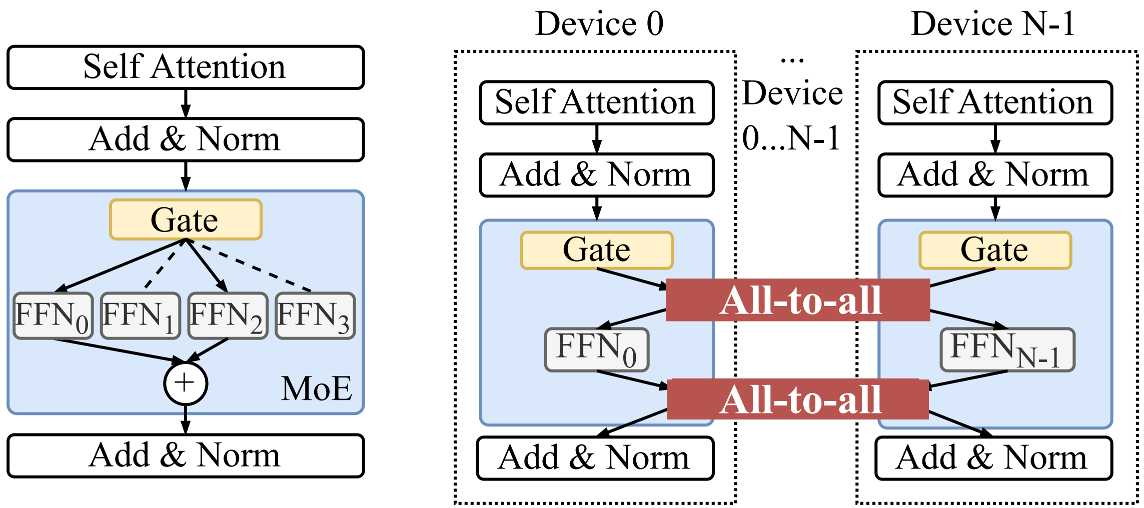
Single-device



Distributed

Distributed MoE

- Hybrid parallelism:
  - Expert parallelism: each device hosts one unique expert
  - Data parallelism: replicate non-expert parameter on each device

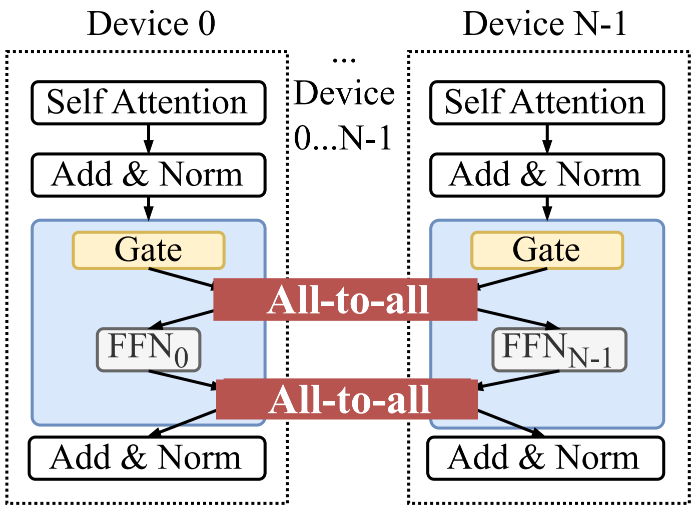


Single-device　　　Distributed

- All-to-all communication
  - 1st: send data samples to experts.
  - 2nd: restore data samples back

Distributed MoE

- Hybrid parallelism:
  - Expert parallelism: each device hosts one unique expert
  - Data parallelism: replicate non-expert parameter on each device

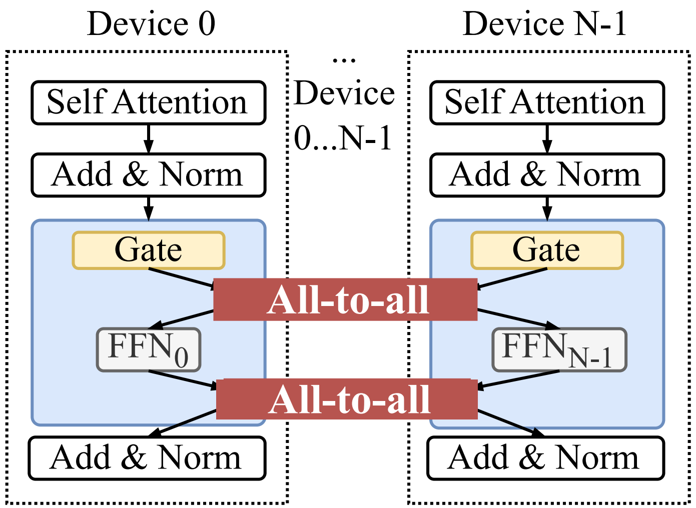


Distributed

- All-to-all communication
  - 1st: send data samples to experts.
  - 2nd: restore data samples back

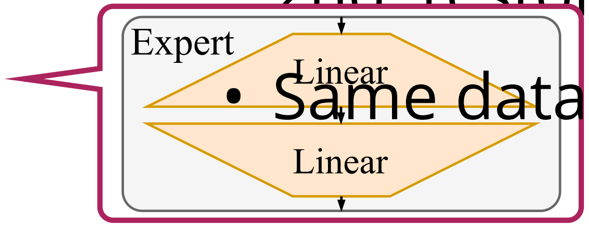
Distributed MoE

- Hybrid parallelism:
  - Expert parallelism: each device hosts one unique expert
  - Data parallelism: replicate non-expert parameter on each device



Distributed

- All-to-all communication
  - 1st: send data samples to experts.
  - 2nd: restore data samples back
  - Same data transfer size



Distributed MoE is not efficient

Model	Training (ms)		Inference (ms)	
#Layers & Params	All-to-all	Ratio	All-to-all	Ratio
12L + 117M	259	36.7%	73	27.4%
24L + 233M	589	35.4%	103	26.2%
36L + 349M	979	38.2%	153	28.3%
12L + 419M	333	39.5%	102	32.5%
24L + 838M	715	37.6%	177	31.7%
36L + 1.2B	1145	36.8%	243	27.4%

All-to-all takes an average of 34.1% of the step time.

Distributed MoE is not efficient

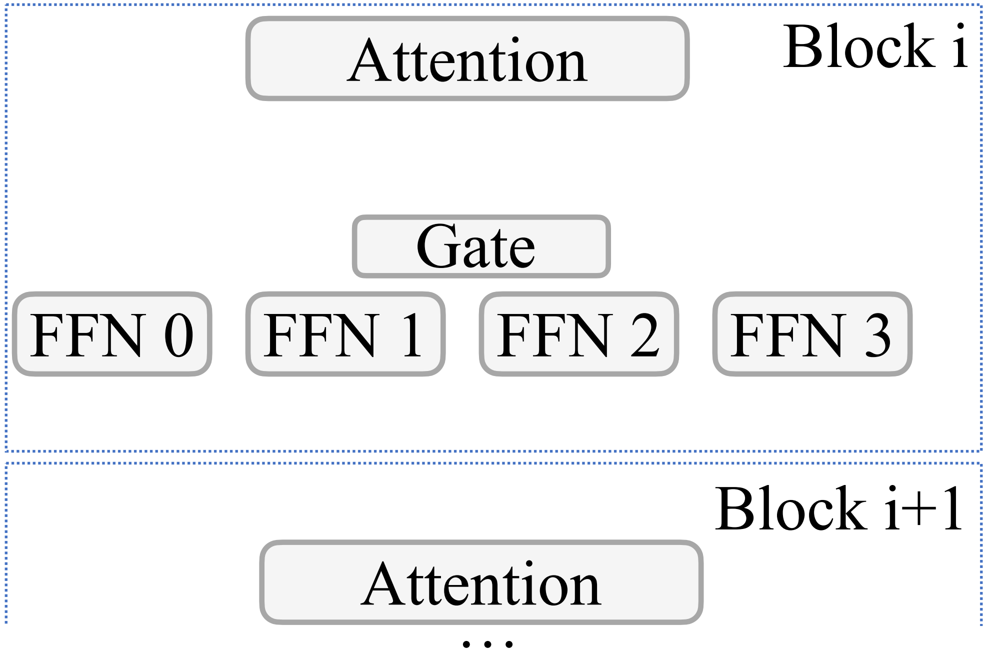
All-to-all takes an average of 34.1% of the step time.

ce (ms)	
l	Ratio
	27.4%
	26.2%
	28.3%
	32.5%
	31.7%
	27.4%

Distributed MoE is not efficient

All-to-all takes an average of 34.1% of the step time.

ce (ms)	
l	Ratio
	27.4%
	26.2%
	28.3%
	32.5%
	31.7%
	27.4%



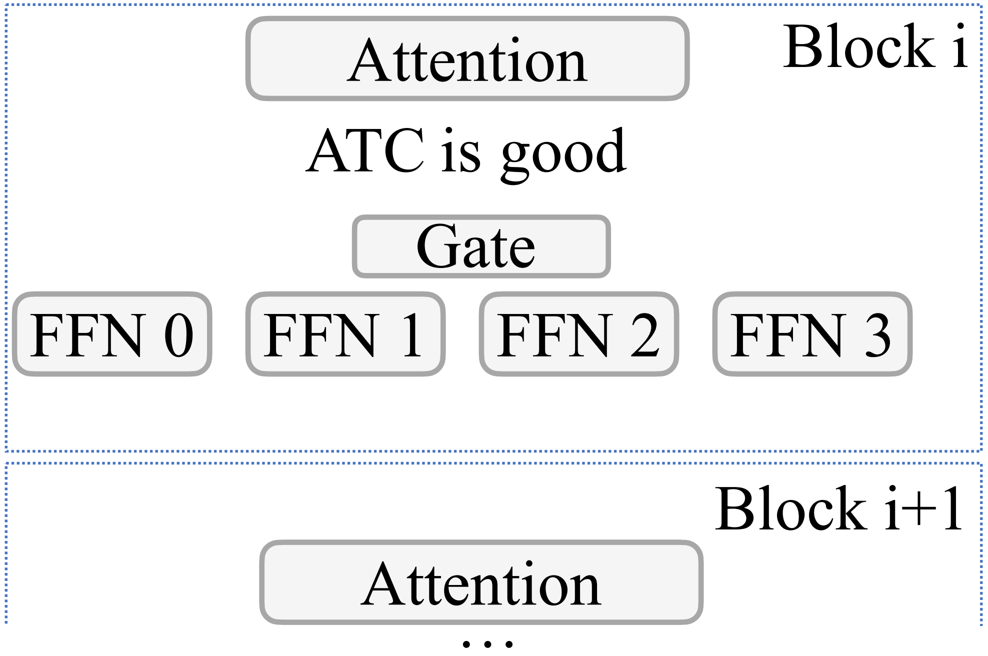
In MoE Transformer:

- Attention -> a complete sequence
- FFN expert -> one single token

Distributed MoE is not efficient

All-to-all takes an average of 34.1% of the step time.

ce (ms)	
l	Ratio
	27.4%
	26.2%
	28.3%
	32.5%
	31.7%
	27.4%



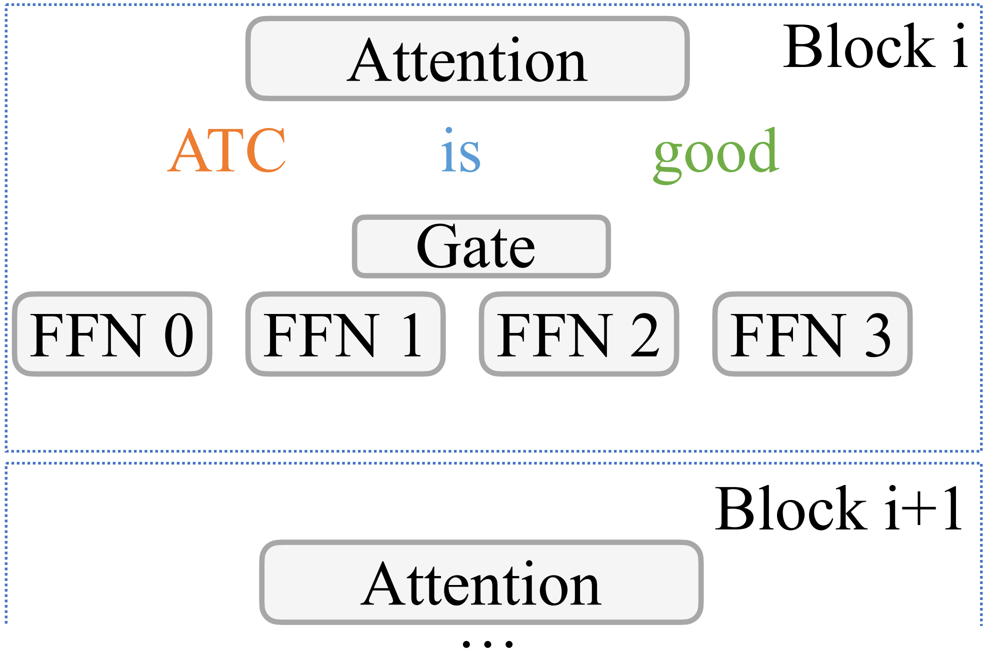
In MoE Transformer:

- Attention -> a complete sequence
- FFN expert -> one single token

Distributed MoE is not efficient

All-to-all takes an average of 34.1% of the step time.

ce (ms)	
l	Ratio
	27.4%
	26.2%
	28.3%
	32.5%
	31.7%
	27.4%



In MoE Transformer:

- Attention -> a complete sequence
- FFN expert -> one single token

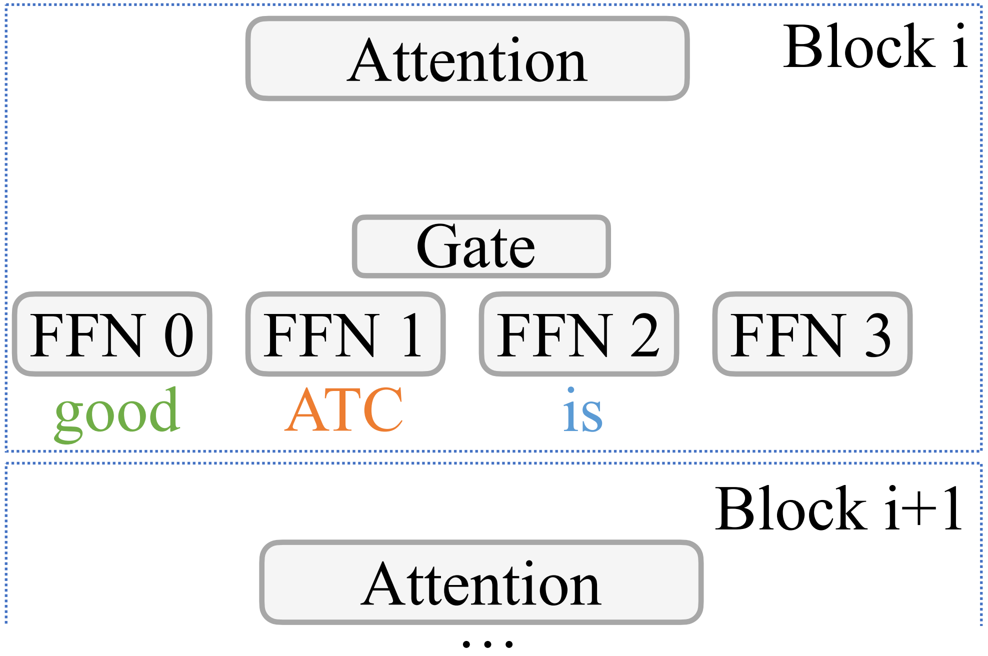
Distributed MoE is not efficient

All-to-all takes an average of 34.1% of the step time.

ce (ms)	
l	Ratio
	27.4%
	26.2%
	28.3%
	32.5%
	31.7%
	27.4%

In MoE Transformer:

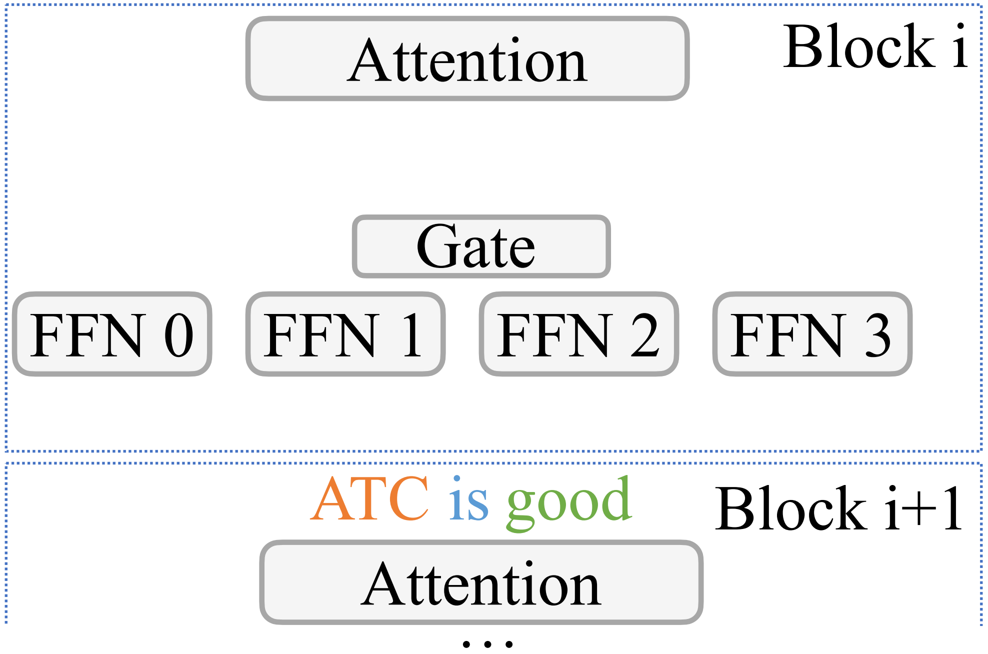
- Attention -> a complete sequence
- FFN expert -> one single token



Distributed MoE is not efficient

All-to-all takes an average of 34.1% of the step time.

ce (ms)	
l	Ratio
	27.4%
	26.2%
	28.3%
	32.5%
	31.7%
	27.4%



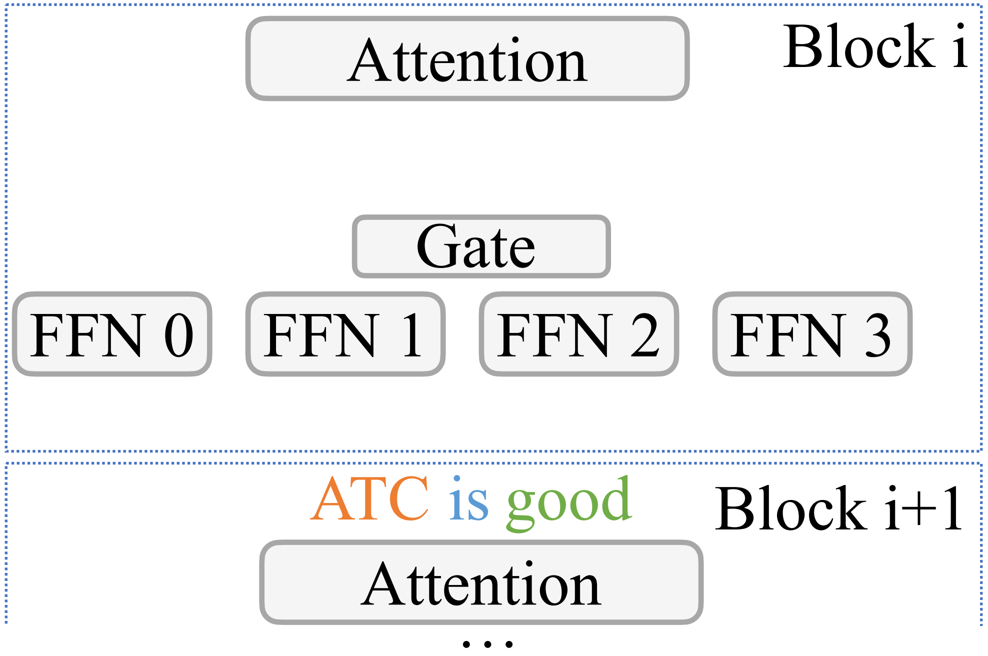
In MoE Transformer:

- Attention -> a complete sequence
- FFN expert -> one single token

Distributed MoE is not efficient

All-to-all takes an average of 34.1% of the step time.

ce (ms)	
l	Ratio
	27.4%
	26.2%
	28.3%
	32.5%
	31.7%
	27.4%



In MoE Transformer:

- Attention -> a complete sequence
- FFN expert -> one single token

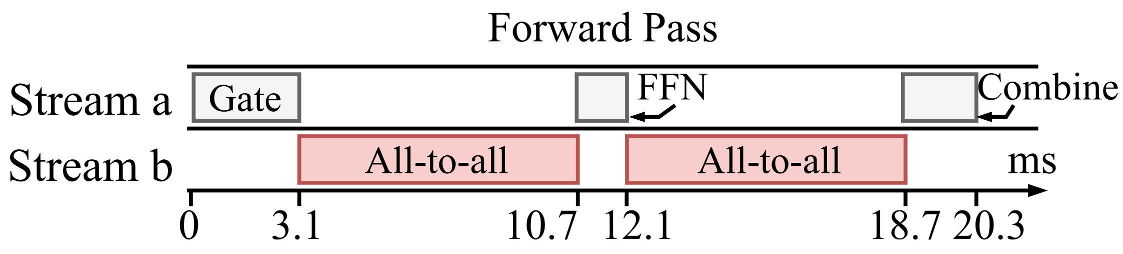
To restore to a sequence, we must wait for the processing of all tokens.

All-to-all is the bottleneck

- Synchronous and blocking operation & large amounts of data transfer.

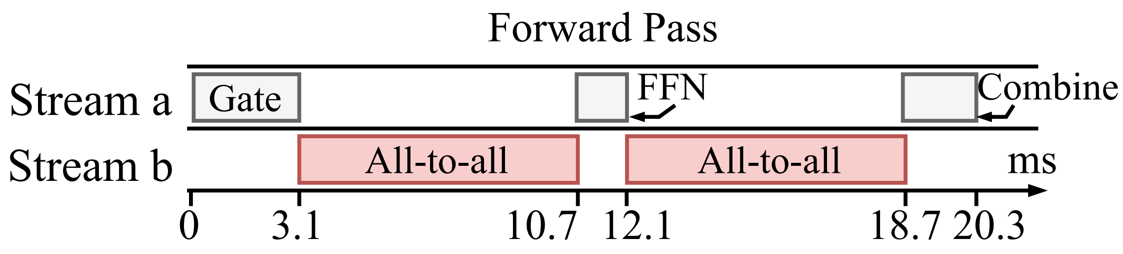
All-to-all is the bottleneck

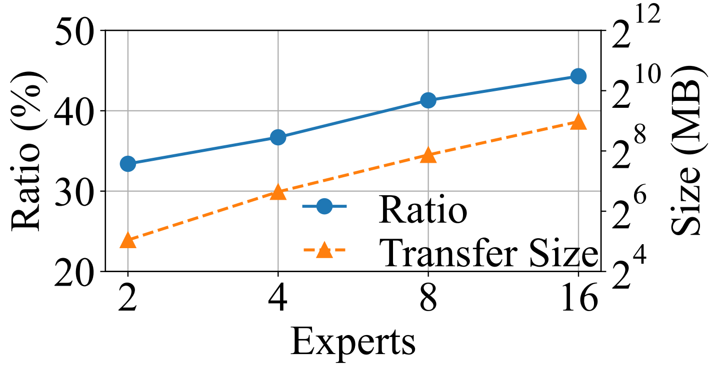
- Synchronous and blocking operation & large amounts of data transfer.



All-to-all is the bottleneck

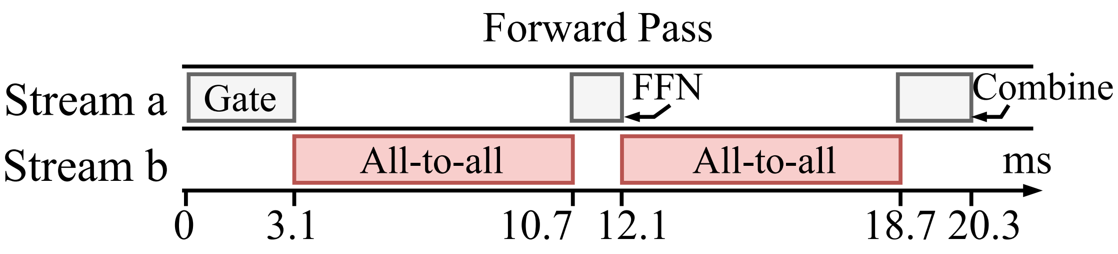
- Synchronous and blocking operation & large amounts of data transfer.

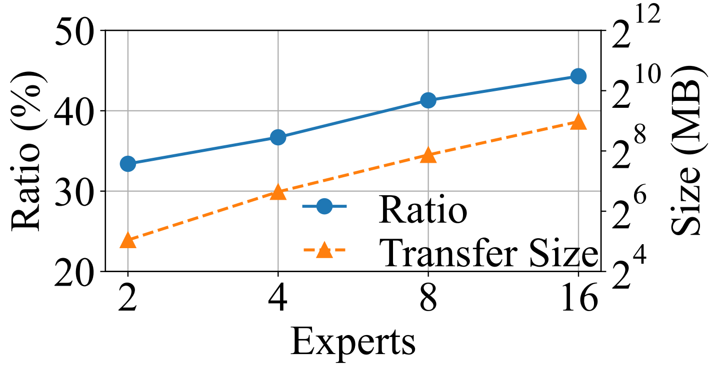




All-to-all is the bottleneck

- Synchronous and blocking operation & large amounts of data transfer.

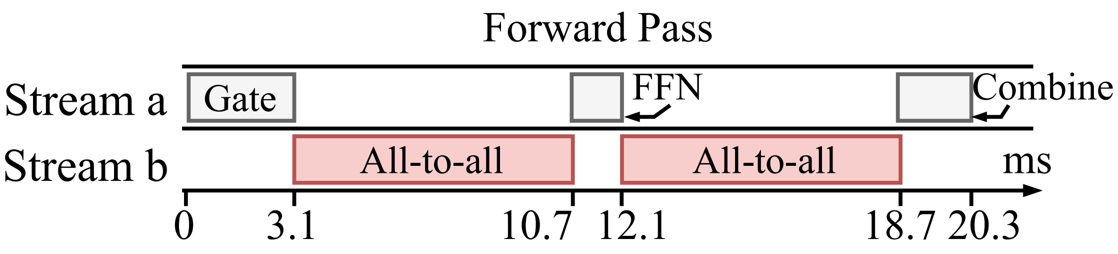


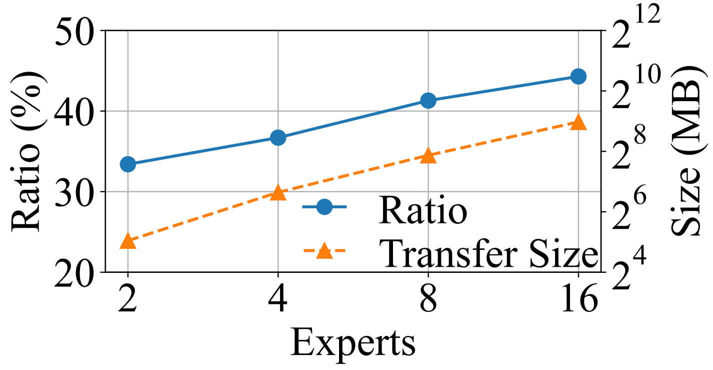


Is it the only cause of all-to-all being the bottleneck?

All-to-all is the bottleneck

- Synchronous and blocking operation & large amounts of data transfer.



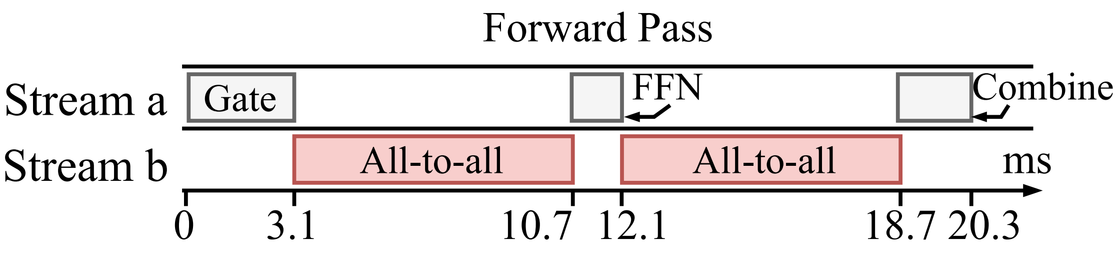


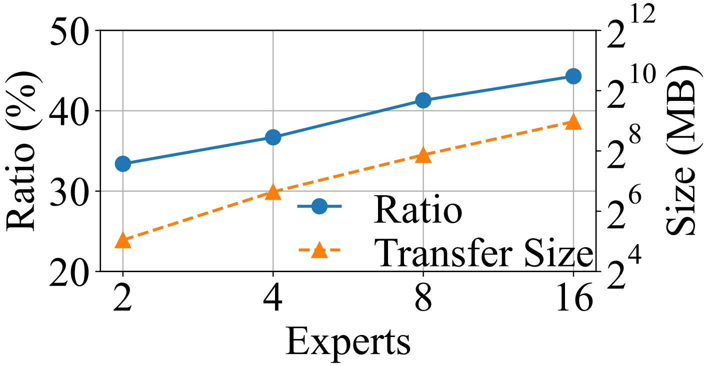
Is it the only cause of all-to-all being the bottleneck?

- MoE training and inference have their *<u>unique</u>* problems.

All-to-all is the bottleneck

- Synchronous and blocking operation & large amounts of data transfer.





Is it the only cause of all-to-all being the bottleneck?

- MoE training and inference have their *unique* problems.
  - Training has backward pass
  - Inference is purely workload-driven

MoE Training in Data & Expert Parallel

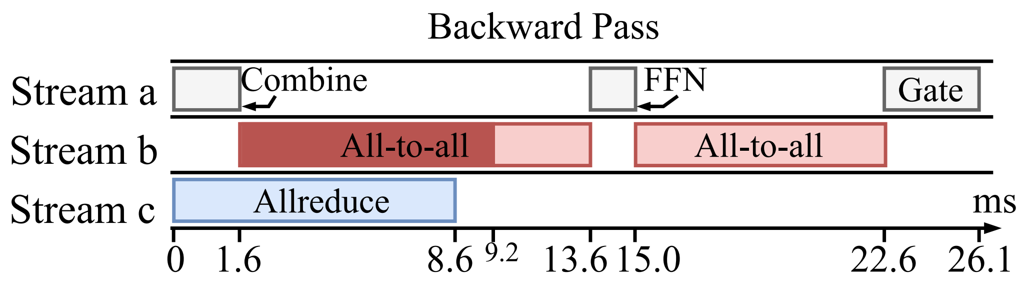
In backward pass,

- Allreduce: asynchronously aggregate *non-expert* gradients in data parallel.
- All-to-all: exchange token gradients to compute *expert* gradients.

MoE Training in Data & Expert Parallel

In backward pass,

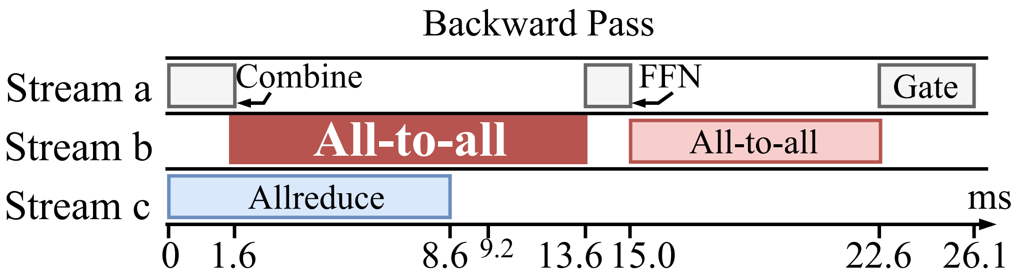
- Allreduce: asynchronously aggregate *non-expert* gradients in data parallel.
- All-to-all: exchange token gradients to compute *expert* gradients.



MoE Training in Data & Expert Parallel

In backward pass,

- Allreduce: asynchronously aggregate *non-expert* gradients in data parallel.
- All-to-all: exchange token gradients to compute *expert* gradients.

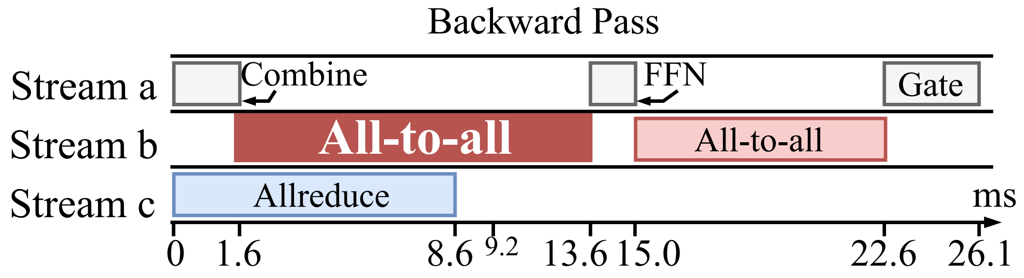


All-to-all is prolonged when it overlaps with allreduce and directly impacts step time.

MoE Training in Data & Expert Parallel

In backward pass,

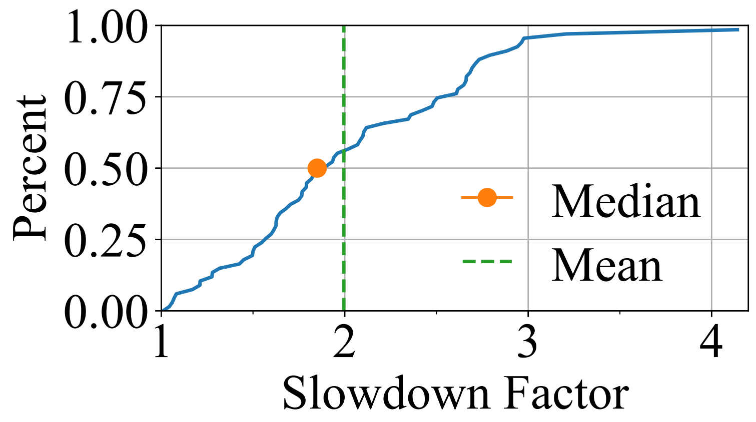
- Allreduce: asynchronously aggregate *non-expert* gradients in data parallel.
- All-to-all: exchange token gradients to compute *expert* gradients.



All-to-all is prolonged when it overlaps with allreduce and directly impacts step time.

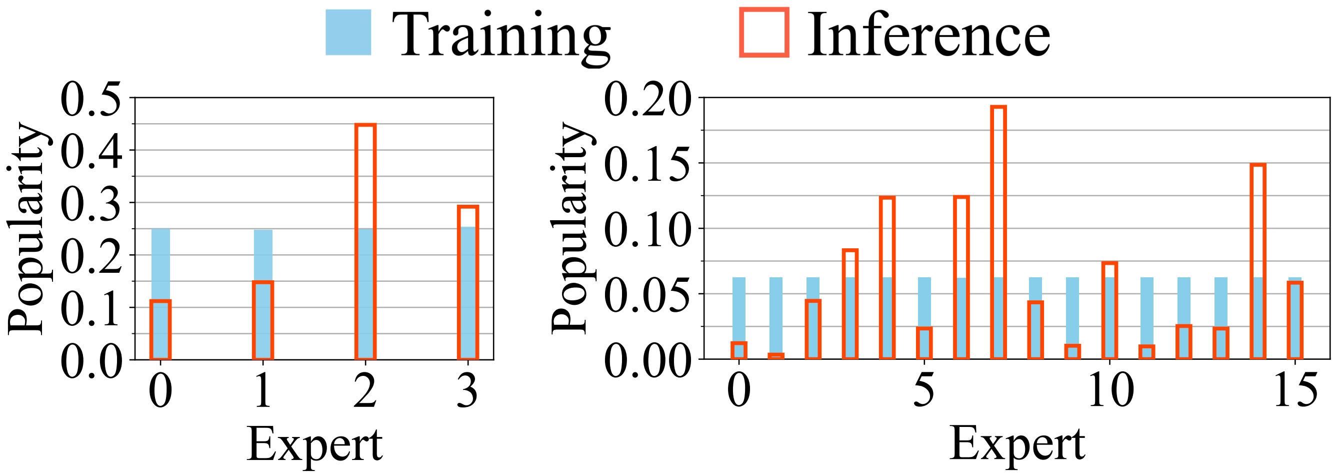
Slowdown of all-to-all varies:

Median: 2x; Maximum: ~4x



Expert Popularity in MoE Inference

- Inference: no load balancing constraints => expert selection is workload-driven, therefore, much more *biased*.

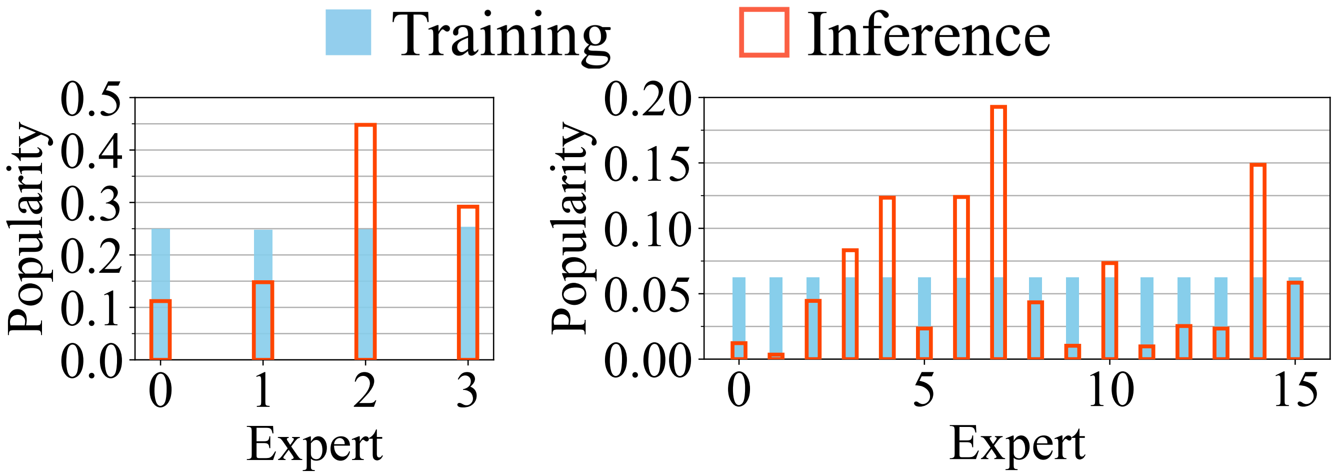


4-expert MoE

16-expert MoE

Expert Popularity in MoE Inference

- Inference: no load balancing constraints => expert selection is workload-driven, therefore, much more *biased*.



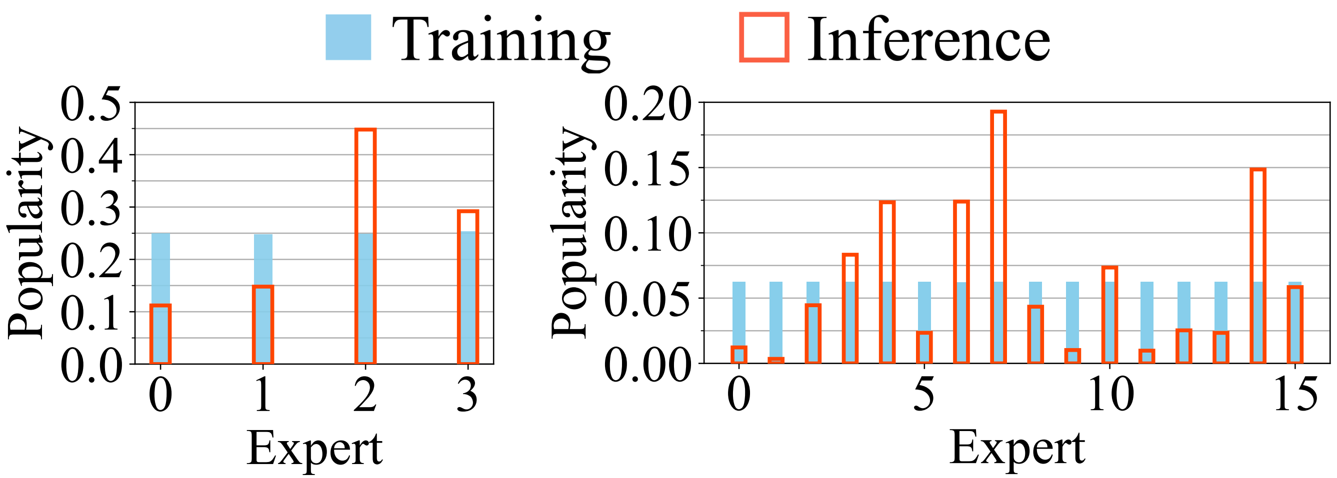
4-expert MoE

16-expert MoE

Devices with unpopular experts have to wait for those with popular experts.

Expert Popularity in MoE Inference

- Inference: no load balancing constraints => expert selection is workload-driven, therefore, much more *biased*.



4-expert MoE

16-expert MoE

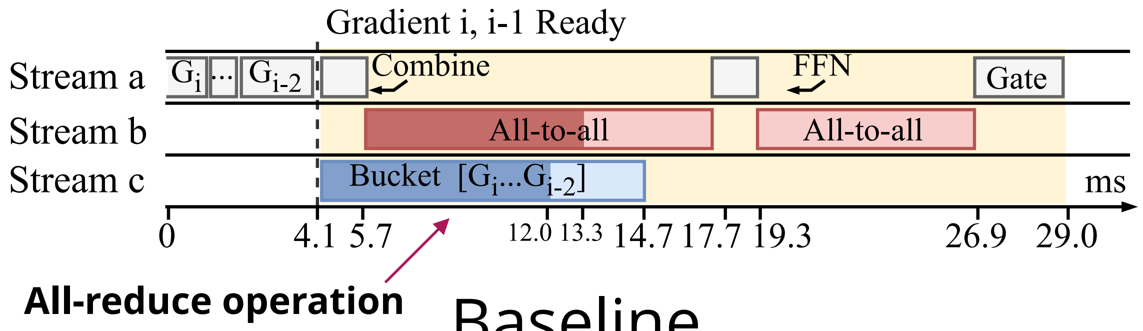
Maximum idle time of the least popular expert
=> 29.4% of the inference time.

Devices with unpopular experts have to wait for those with popular experts.

Training: Challenges

Intuition: always prioritise all-to-all and avoid bandwidth sharing.

- Minimise the blocking period incurred by all-to-all

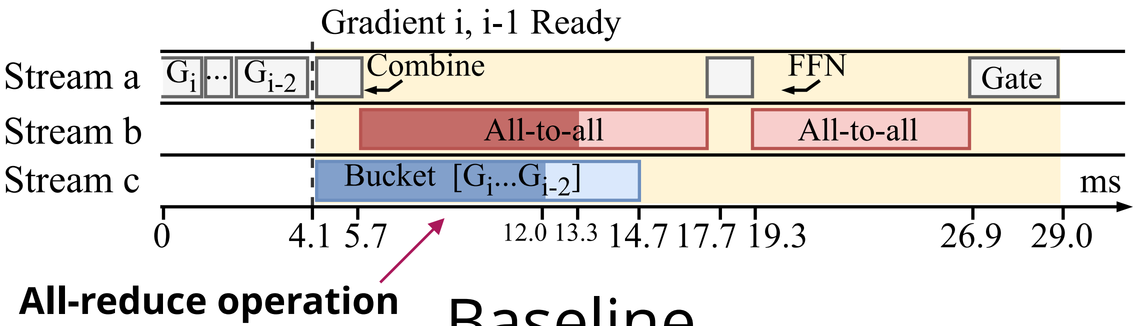


Baseline

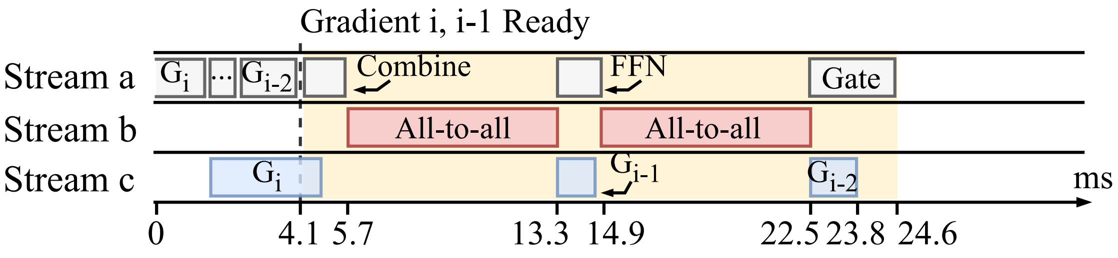
Training: Challenges

Intuition: always prioritise all-to-all and avoid bandwidth sharing.

- Minimise the blocking period incurred by all-to-all



Baseline



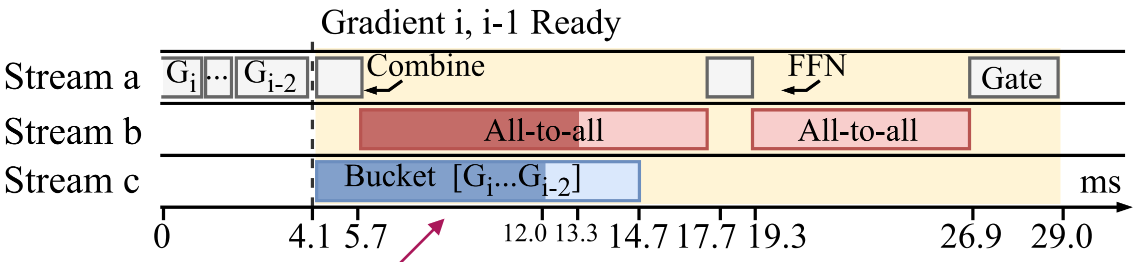
Optimal

Training: Challenges

Intuition: always prioritise all-to-all and avoid bandwidth sharing.

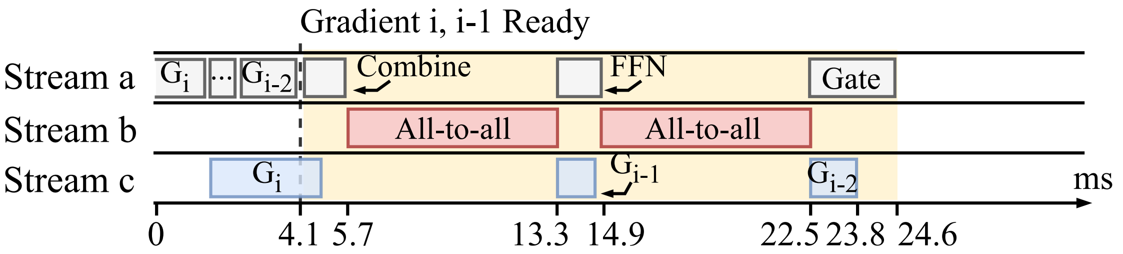
- Minimise the blocking period incurred by all-to-all

Backward pass



All-reduce operation

Baseline



Optimal

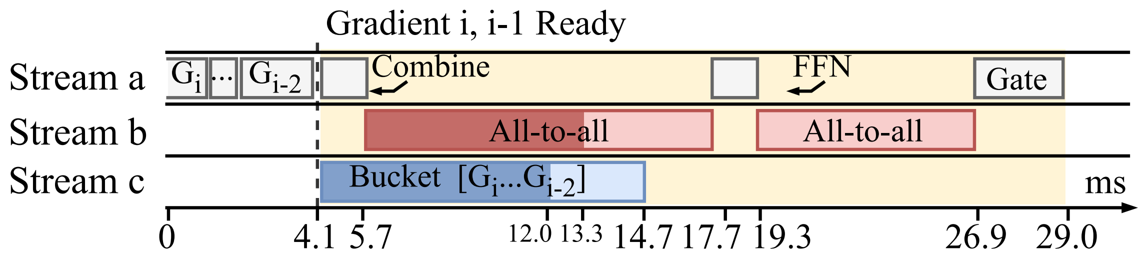
Challenges:

1. NCCL Communication primitives cannot be preempted.

2. No control knob to adjust resource sharing (GPU SM, network bandwidth…).

Training: Micro-op Scheduling

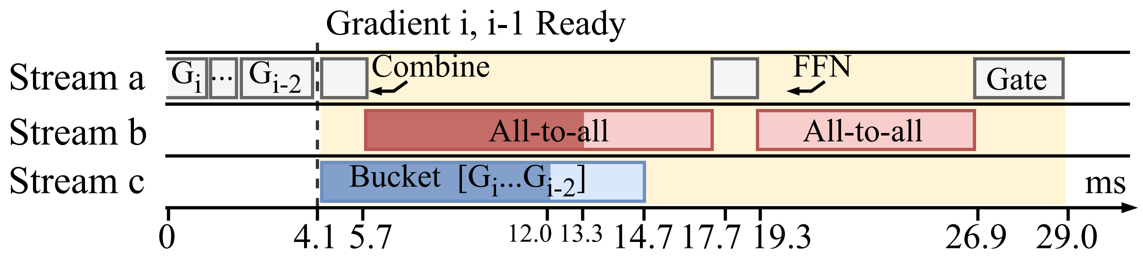
- Tensor Partitioning
  - Partition allreduce into micro-ops
  - Prioritise all-to-all whenever possible



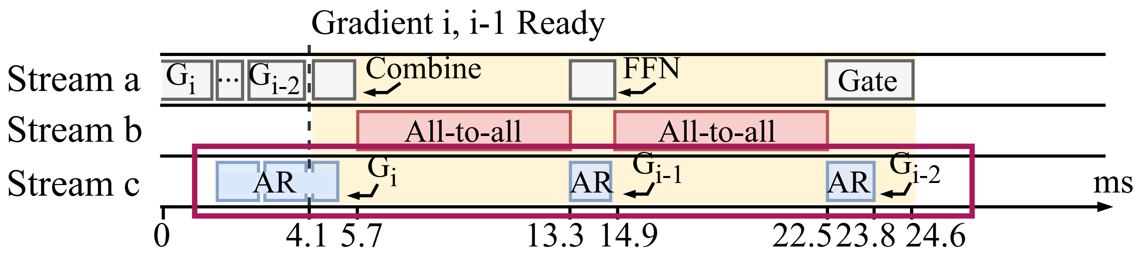
Baseline

Training: Micro-op Scheduling

- Tensor Partitioning
  - Partition allreduce into micro-ops
  - Prioritise all-to-all whenever possible



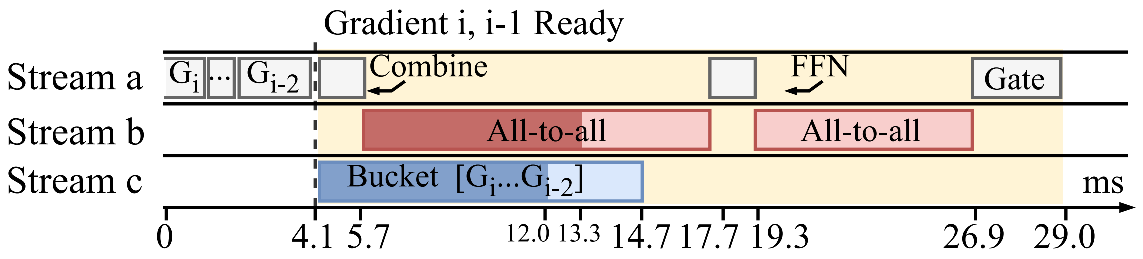
Baseline



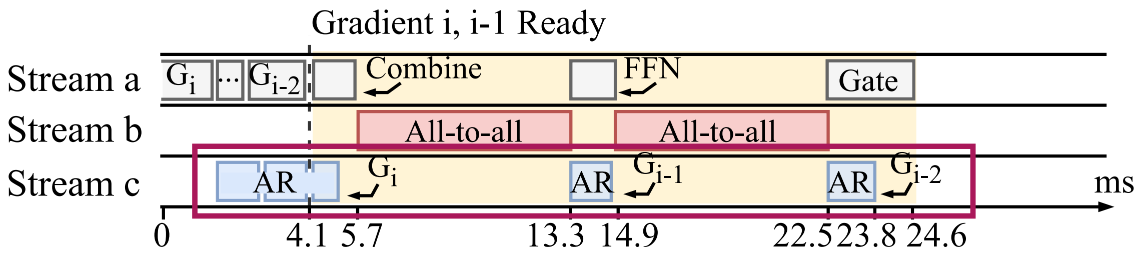
Prioritise all-to-all

Training: Micro-op Scheduling

- Tensor Partitioning
  - Partition allreduce into micro-ops
  - Prioritise all-to-all whenever possible



Baseline

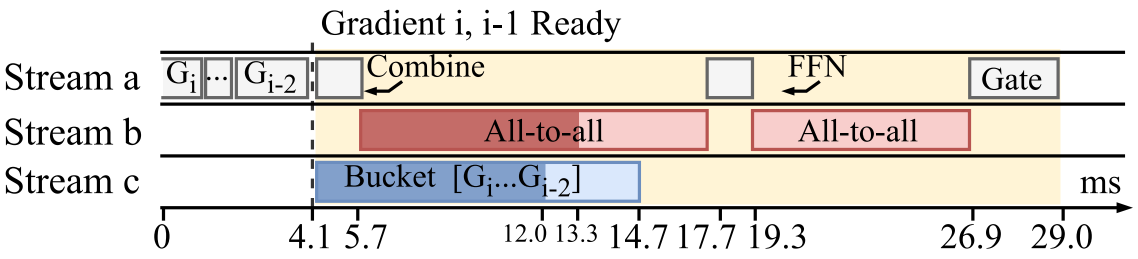


Prioritise all-to-all

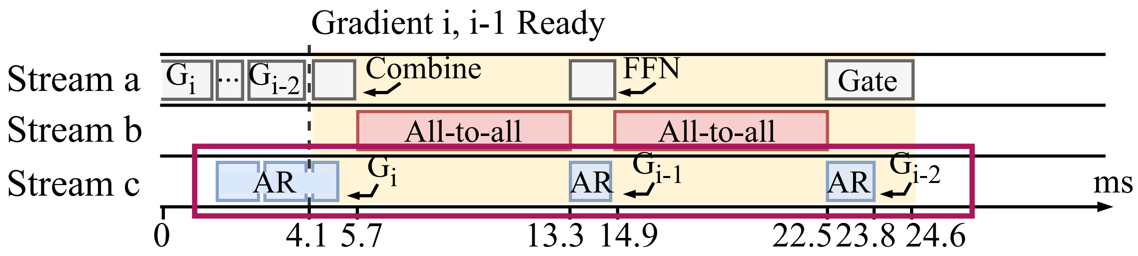
- Partition all-to-all into micro-ops
- Pipelining computation and all-to-all

Training: Micro-op Scheduling

- Tensor Partitioning
  - Partition allreduce into micro-ops
  - Prioritise all-to-all whenever possible

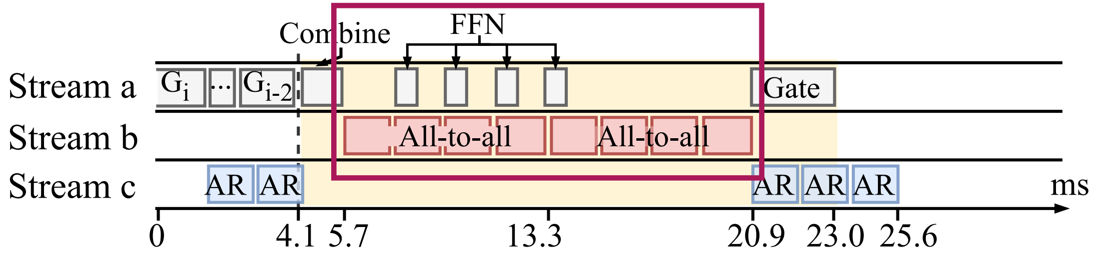


Baseline



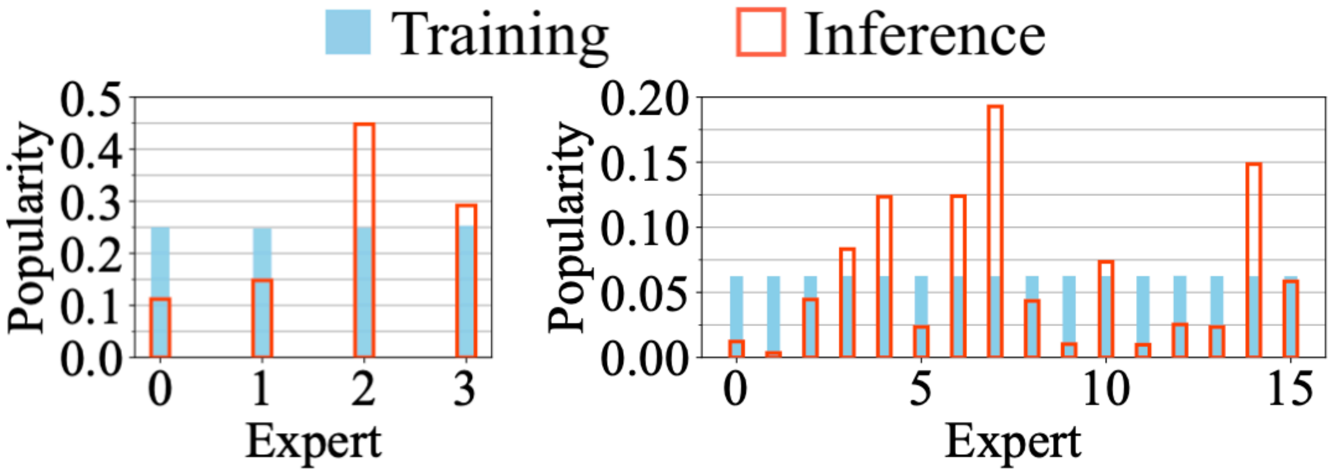
Prioritise all-to-all

- Partition all-to-all into micro-ops
- Pipelining computation and all-to-all



Inference: Challenges

- MoE inference: no load balancing constraints => expert selection is workload-driven, therefore, much more *biased*.



4-expert MoE

16-expert MoE

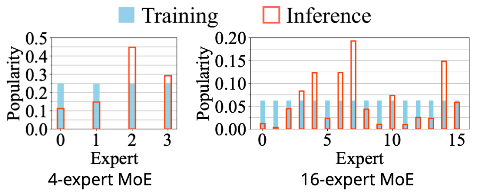
Maximum idle time of the least popular expert
=> 29.4% of the inference time.

Devices with unpopular experts have to wait for those with popular experts.

Inference: Challenges

Q: How to deal with imbalance computation load?

- MoE inference: no load balancing constraints => expert selection is workload-driven, therefore, much more *biased*.



Maximum idle time of the least popular expert
=> 29.4% of the inference time.

Devices with unpopular experts have to wait for those with popular experts.

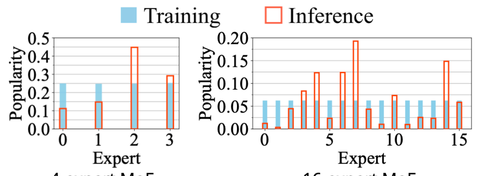
Inference: Challenges

Q: How to deal with imbalance computation load?

Allocate resources based on expert popularity:

Popular experts => more resources.

- MoE inference: no load balancing constraints => expert selection is workload-driven, therefore, much more *biased*.



4-expert MoE

16-expert MoE

Maximum idle time of the least popular expert
=> 29.4% of the inference time.

Devices with unpopular experts have to wait for those with popular experts.

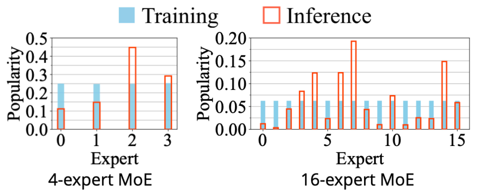
Inference: Challenges

Q: How to deal with imbalance computation load?

Allocate resources based on expert popularity:

Popular experts => more resources.

- MoE inference: no load balancing constraints => expert selection is workload-driven, therefore, much more *biased*.



Maximum idle time of the least popular expert
=> 29.4% of the inference time.

Devices with unpopular experts have to wait for those with popular experts.

Token's expert selection cannot be determined a prior to the actual gating computation.

Model& Dataset	Layer	Top-4			
Transformer-XL & Enwik8 (Text generation)	3	9	4	5	10
	4	5	7	8	10
	8	9	2	3	13
	12	4	5	15	8
BERT-Large & WMT En-De (Translation)	6	7	6	10	1
	8	10	6	2	15
	10	9	4	11	8
	12	1	8	10	14

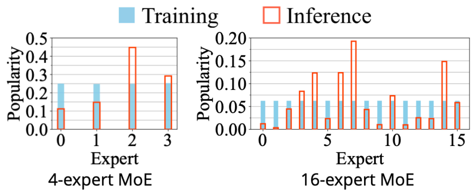
Inference: Challenges

Q: How to deal with imbalance computation load?

Allocate resources based on expert popularity:

Popular experts => more resources.

- MoE inference: no load balancing constraints => expert selection is workload-driven, therefore, much more *biased*.



Maximum idle time of the least popular expert
=> 29.4% of the inference time.

Devices with unpopular experts have to wait for those with popular experts.

Token's expert selection cannot be determined a prior to the actual gating computation.

Model& Dataset	Layer	Top-4			
Transformer-XL & Enwik8 (Text generation)	3	9	4	5	10
	4	5	7	8	10
	8	9	2	3	13
	12	4	5	15	8
BERT-Large & WMT En-De (Translation)	6	7	6	10	1
	8	10	6	2	15
	10	9	4	11	8
	12	1	8	10	14

Resource scheduling after gating network.

Inefficient practice for latency-sensitive tasks!

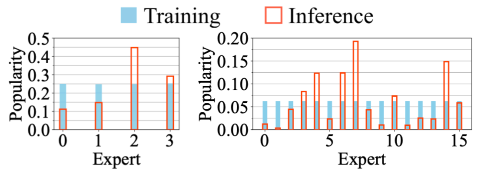
Inference: Challenges

Q: How to deal with imbalance computation load?

Allocate resources based on expert popularity:

Popular experts => more resources.

- MoE inference: no load balancing constraints => expert selection is workload-driven, therefore, much more *biased*.



4-expert MoE   16-expert MoE

Maximum idle time of the least popular expert
=> 29.4% of the inference time.

Devices with unpopular experts have to wait for those with popular experts.

Token's expert selection cannot be determined a prior to the actual gating computation.

Model& Dataset	Layer	Top-4			
Transformer-XL & Enwik8 (Text generation)	3	9	4	5	10
	4	5	7	8	10
	8	9	2	3	13
	12	4	5	15	8
BERT-Large & WMT En-De (Translation)	6	7	6	10	1
	8	10	6	2	15
	10	9	4	11	8
	12	1	8	10	14

Resource scheduling after gating network.

<u>Inefficient practice for latency-sensitive tasks!</u>

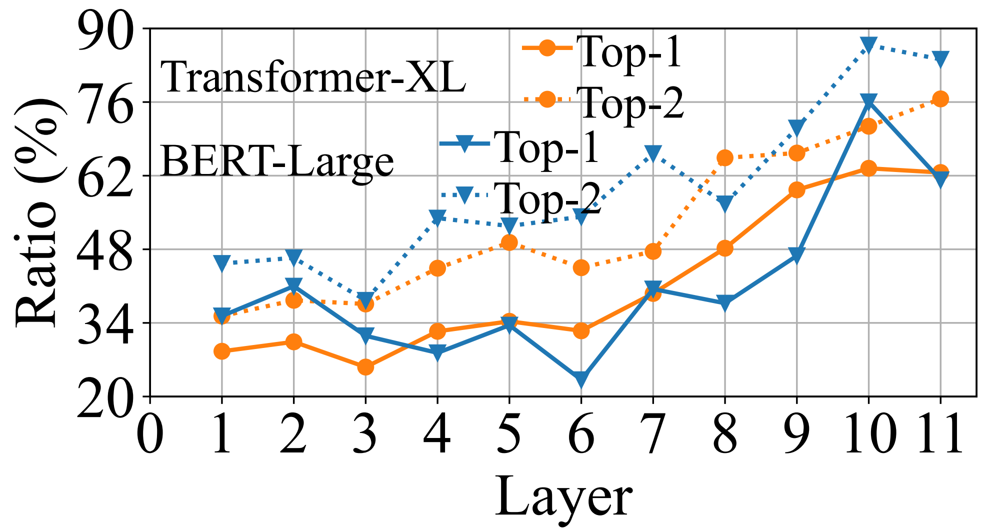
How to achieve low-overhead resource scheduling to balance device load?

Inference: Pattern in Expert Selection

- Findings: similar tokens tend to be processed by the same or similar experts in each layer.
- Tokens selecting the same expert in layer i tend to select the same expert again in layer i + 1.

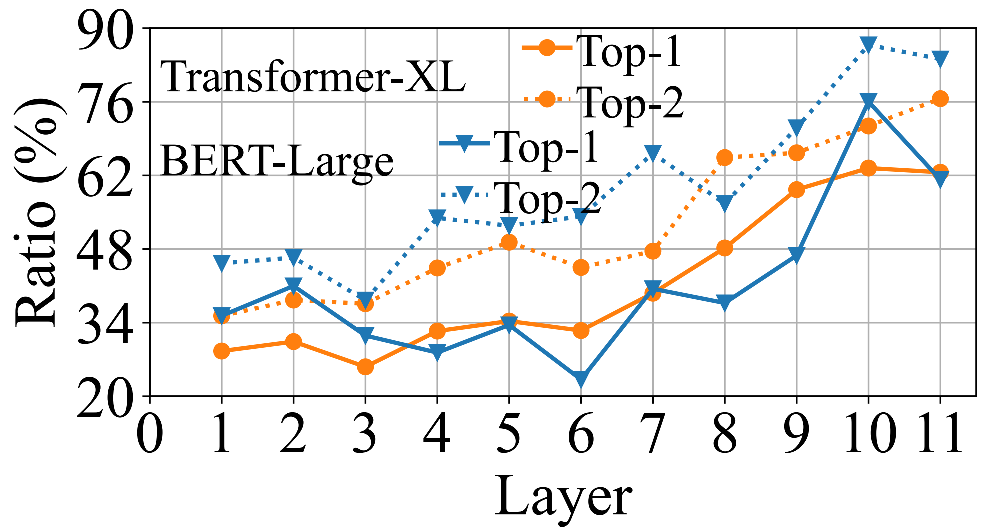
Inference: Pattern in Expert Selection

- Findings: similar tokens tend to be processed by the same or similar experts in each layer.
- Tokens selecting the same expert in layer i tend to select the same expert again in layer i + 1.



Inference: Pattern in Expert Selection

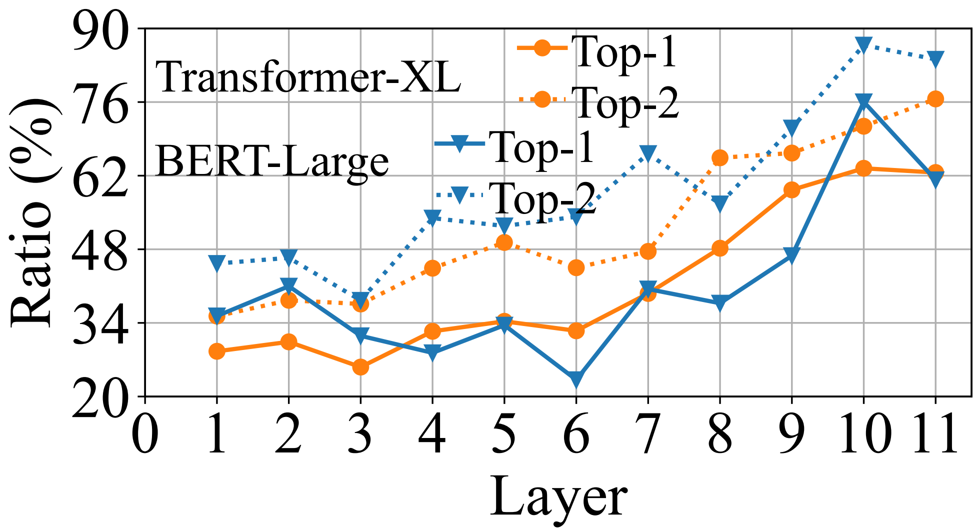
- Findings: similar tokens tend to be processed by the same or similar experts in each layer.
- Tokens selecting the same expert in layer i tend to select the same expert again in layer i + 1.



41.94% tokens when k is 1
54.59% tokens when k is 2

Inference: Pattern in Expert Selection

- Findings: similar tokens tend to be processed by the same or similar experts in each layer.
- Tokens selecting the same expert in layer i tend to select the same expert again in layer i + 1.



41.94% tokens when k is 1
54.59% tokens when k is 2

Exploit this pattern to estimate the overall expert popularity

Inference: Expert Popularity Estimation

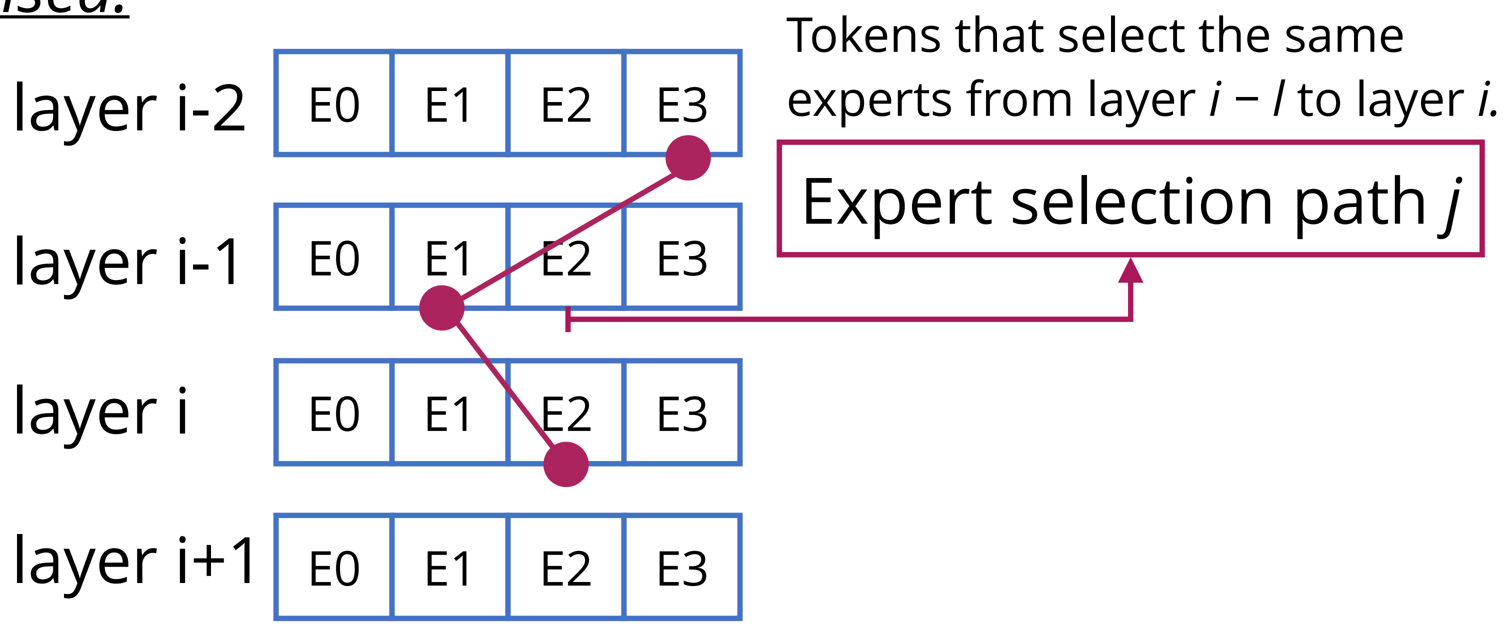
Inference: Expert Popularity Estimation

- Idea: Collect the expert selection distribution during training *after the load balancing loss is minimised.*

layer i-2	E0	E1	E2	E3
layer i-1	E0	E1	E2	E3
layer i	E0	E1	E2	E3
layer i+1	E0	E1	E2	E3

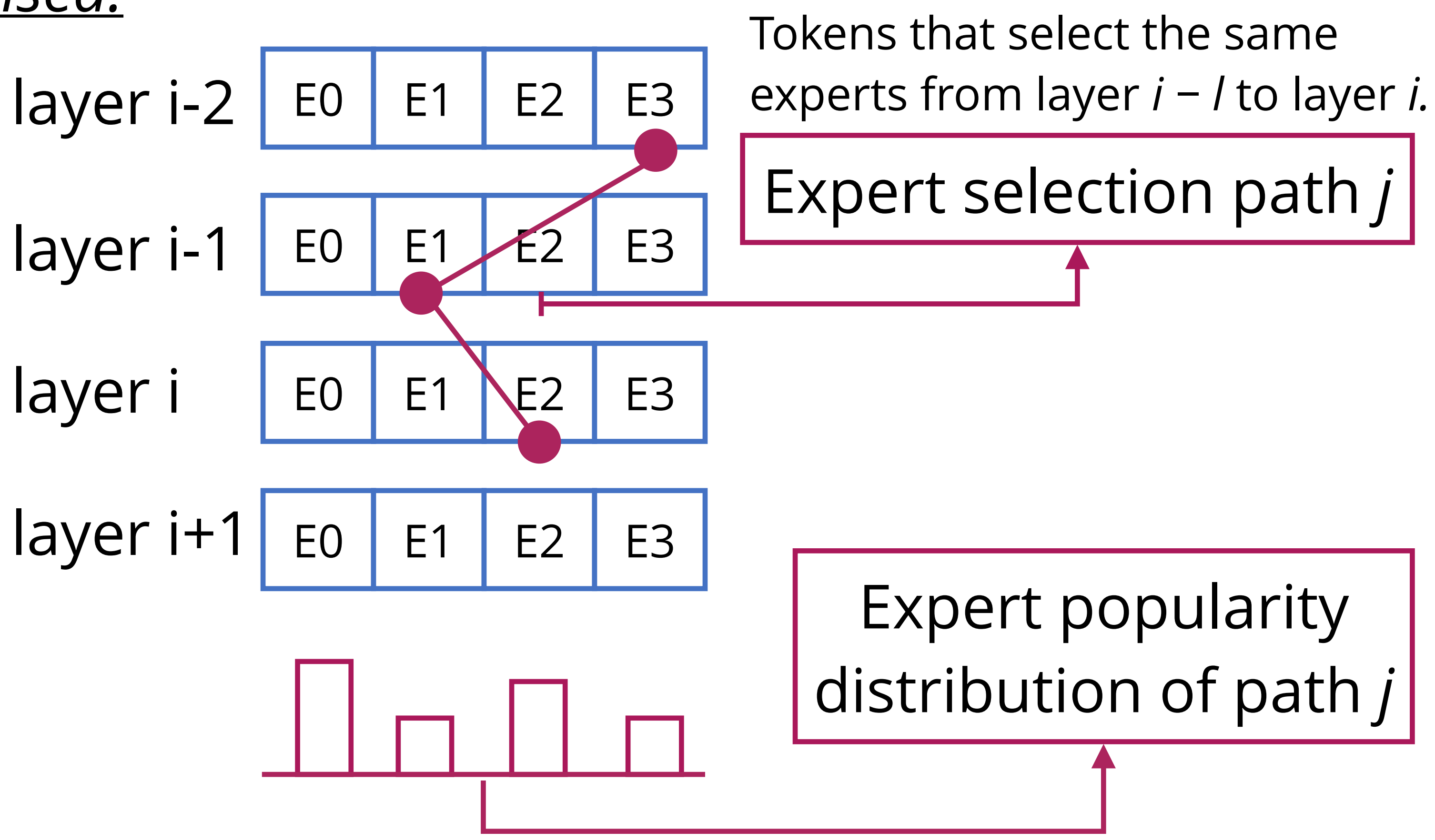
Inference: Expert Popularity Estimation

- Idea: Collect the expert selection distribution during training <u>*after the load balancing loss is minimised.*</u>



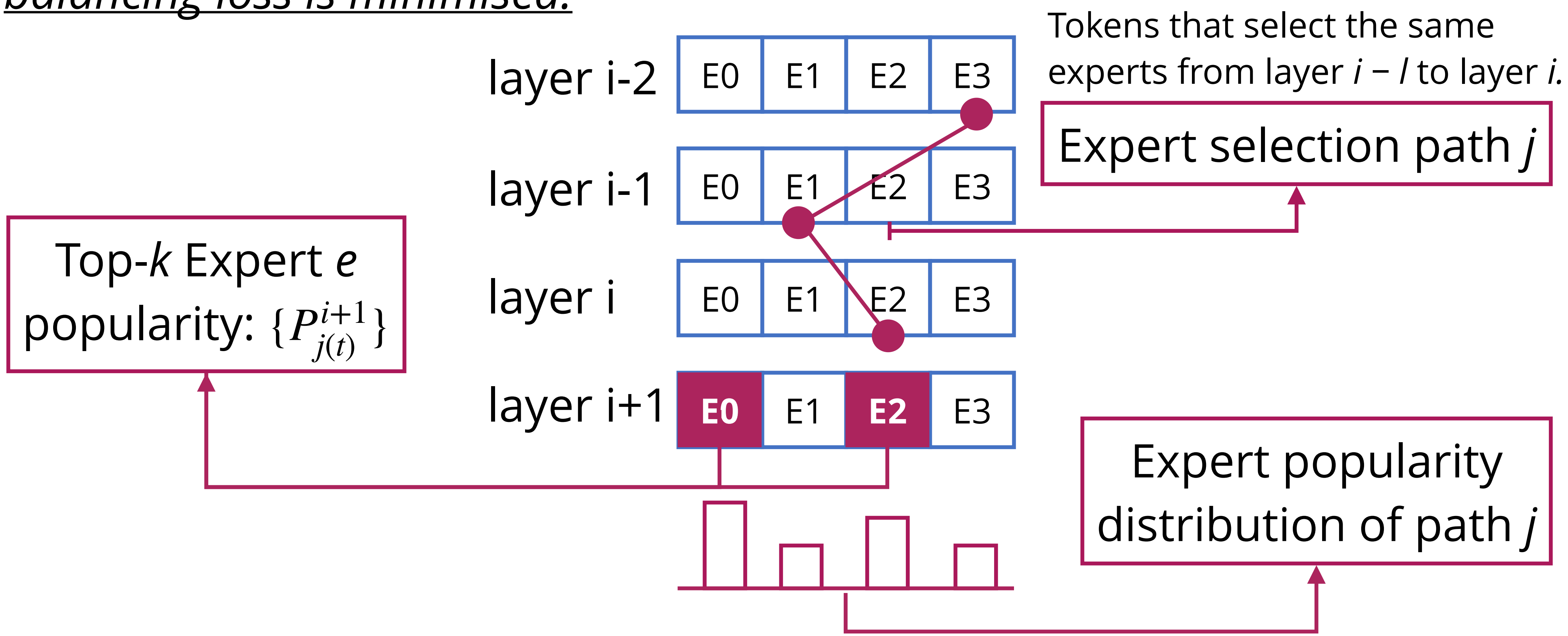
Inference: Expert Popularity Estimation

- Idea: Collect the expert selection distribution during training *after the load balancing loss is minimised.*



Inference: Expert Popularity Estimation

- Idea: Collect the expert selection distribution during training *after the load balancing loss is minimised.*



Inference: Two-phase Scheduling

- Phase 1:
  - Compute resource allocation for expert *e* based on popularity estimation:

$$n_e = N \times \sum_{t=1}^{N_t} P_{j(t)}^{i+1}(e)/N_t$$

No. of tokens in a batch

No. of GPUs

Inference: Two-phase Scheduling

- Phase 1: **Pipelined with model computation => nearly zero overhead**
  - Compute resource allocation for expert *e* based on popularity estimation:

$$n_e = N \times \sum_{t=1}^{N_t} P_{j(t)}^{i+1}(e)/N_t$$

No. of tokens in a batch

No. of GPUs

Inference: Two-phase Scheduling

- Phase 1: **Pipelined with model computation => nearly zero overhead**
  - Compute resource allocation for expert *e* based on popularity estimation:

$$n_e = \underbrace{N}_{\text{No. of GPUs}} \times \sum_{t=1}^{N_t} P_{j(t)}^{i+1}(e) / \underbrace{N_t}_{\text{No. of tokens in a batch}}$$

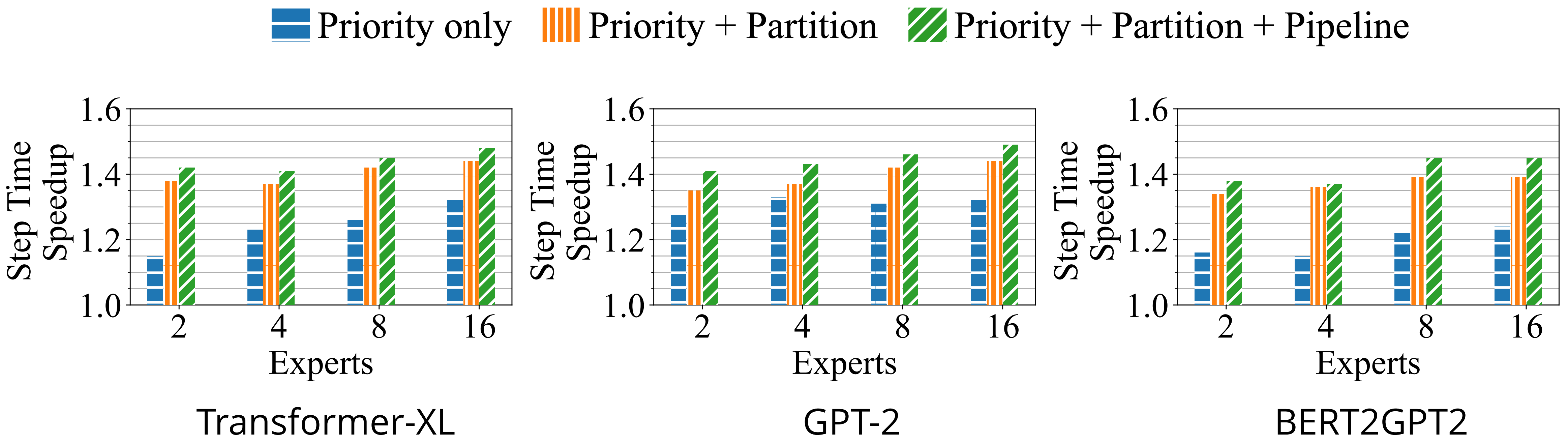
- Phase 2:
  - Fine-tune the allocation with the actual expert selection.
    - Re-compute the allocation when the actual selection deviates significantly from the estimation.

Evaluation

- Testbed: Our testbed has four worker nodes. Each node has 4 Ampere A100 GPUs with 40GB memory and is equipped with 100Gbps InfiniBand.
- Every FFN layer in Transformer is replaced with the MoE layer.
- Training models:
  - Transformer-XL: a 24-layer encoder model.
  - BERT2GPT2: a 12-layer encoder-decoder model.
  - GPT-2: a 12-layer decoder model.
- Inference models:
  - Transformer-XL: text generation with Enwik8 test set.
  - BERT: a 12-layer decoder model for translation using WMT En-De test set.

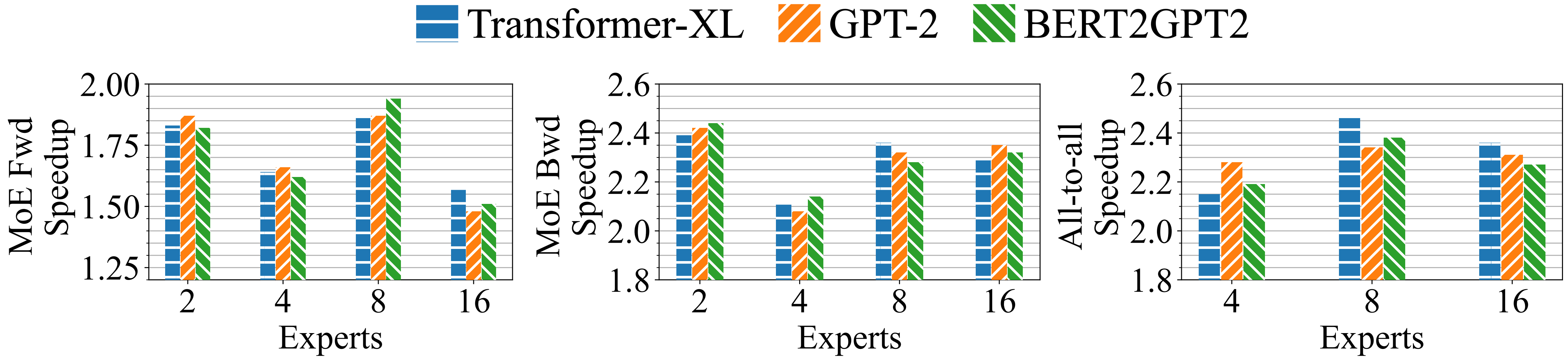
Training Step Time

Training step time speedup over Baseline (DeepSpeed) with different design choices.



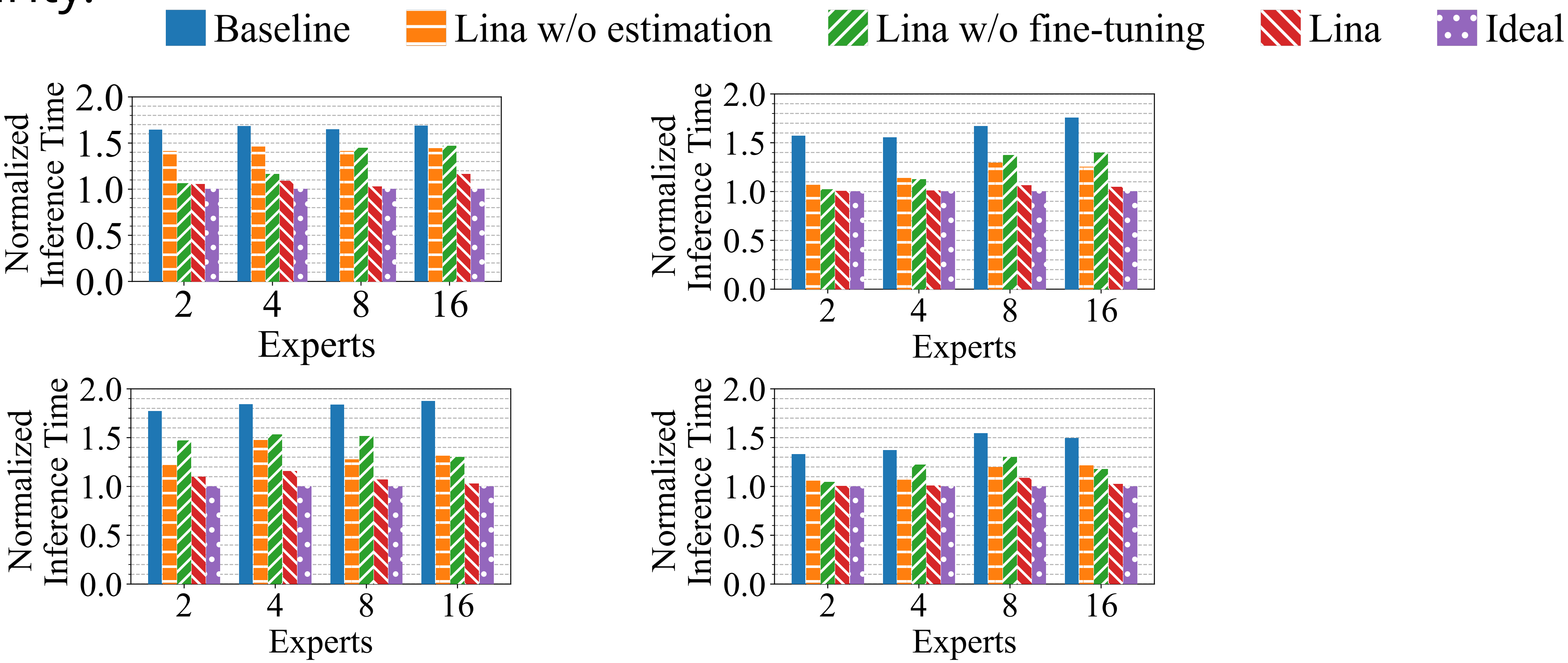
MoE Layer in Training

MoE layer forward, backward, all-to-all running time speedup over Baseline.



Inference Latency

Ideal: schedule resources assuming we have the prior knowledge of exact expert popularity.



Median Latency

95%ile Latency

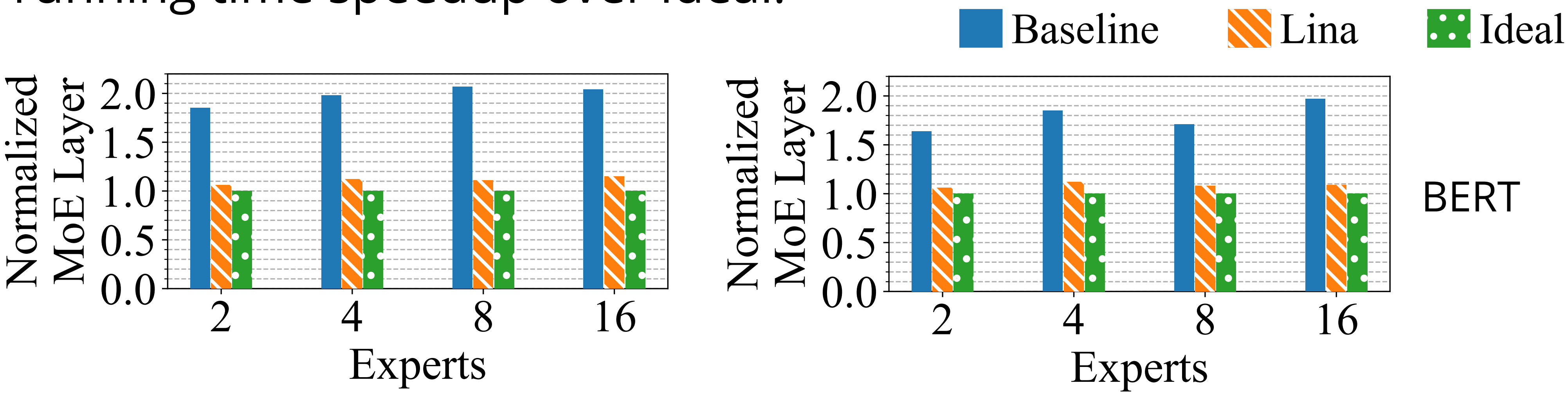
Transformer-XL

BERT

MoE Layer in Inference

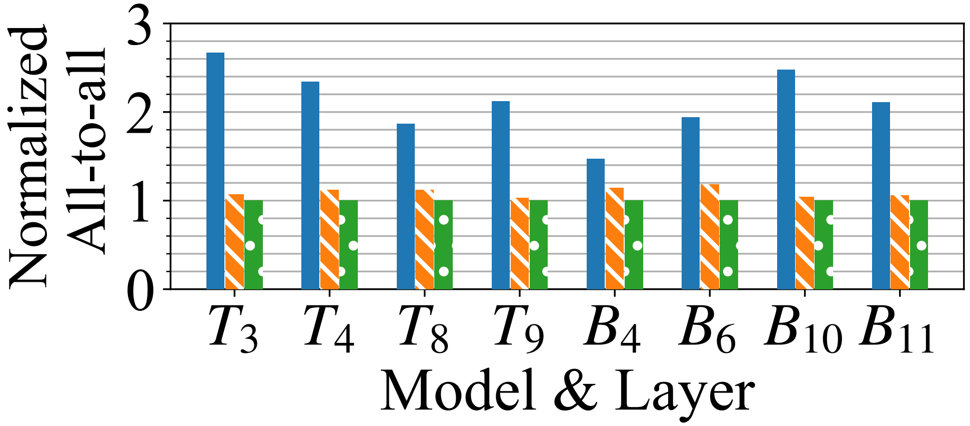
MoE layer running time speedup over Ideal.

Transformer-XL



BERT

All-to-all running time speedup over Ideal in selected layers.



Lina’s Contributions

- An in-depth empirical analysis of distributed MoE
  - Main causes for all-to-all to be the performance bottleneck in training and inference.
- [Training] A scheduler prioritises all-to-all over allreduce to improve its bandwidth and reduce its blocking period.
- [Inference] An estimation method of expert popularity to conduct two-phase resource scheduling.

Thanks!