Expert Parallelism in LLM Training

Spring 2026

Lecturer: Yuedong (Steven) Xu

Fudan University

ydxu@fudan.edu.cn

Disclaimer

Machine learning systems is a broad and rapidly evolving field. The course material has been developed using a broad spectrum of resources, including research papers, lecture slides, blogposts, research talks, tutorial videos, and other materials shared by the research community.

Distributed LLM Training: Outline

- Data Parallelism
  - Parameter-Server, All-Reduce, Memory Optimization
- Model Parallelism
  - Pipeline Parallelism, Tensor Parallelism, Sequence Parallelism
- Mixture of Experts
  - **Principles**
  - Parallel Training
  - DeepEP

Mixture of Experts

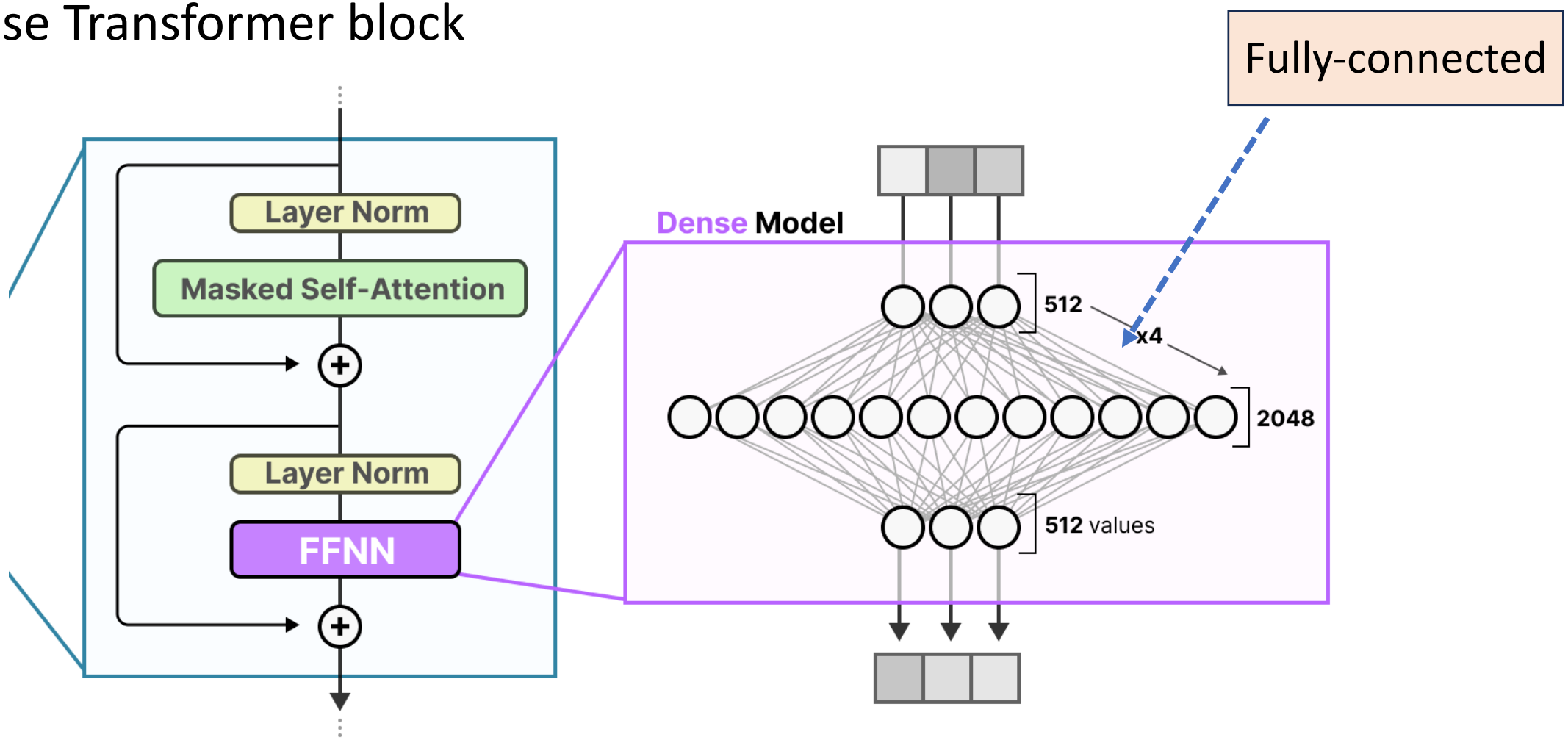
- Mixture of Experts in Early Days
  - Robert Jacobs, Michael Jordan, Steven Nowlan and Geoffrey Hinton (1991)
    - “a new supervised learning procedure for systems composed of many separate networks, each of which learns to handle a subset of the complete set of training cases.”
  - David Eigen, Marc'Aurelio Ranzato, Ilya Sutskever (2013)
    - “extend the Mixture of Experts to a stacked model, the Deep Mixture of Experts, with multiple sets of gating and experts.”
- Mixture of Experts in Transformer
  - GShard by Google (2020)

What shall we learn in this lecture

- We only learn
  - the structure of MoE in today's language models
  - the system challenges faced by MoE model training
  - recent progresses on addressing All-to-All issues
- We encourage students to learn by themselves
  - the instability of MoE model training
  - the design of new MoE framework
  - the fine-tuning techniques of MoE models

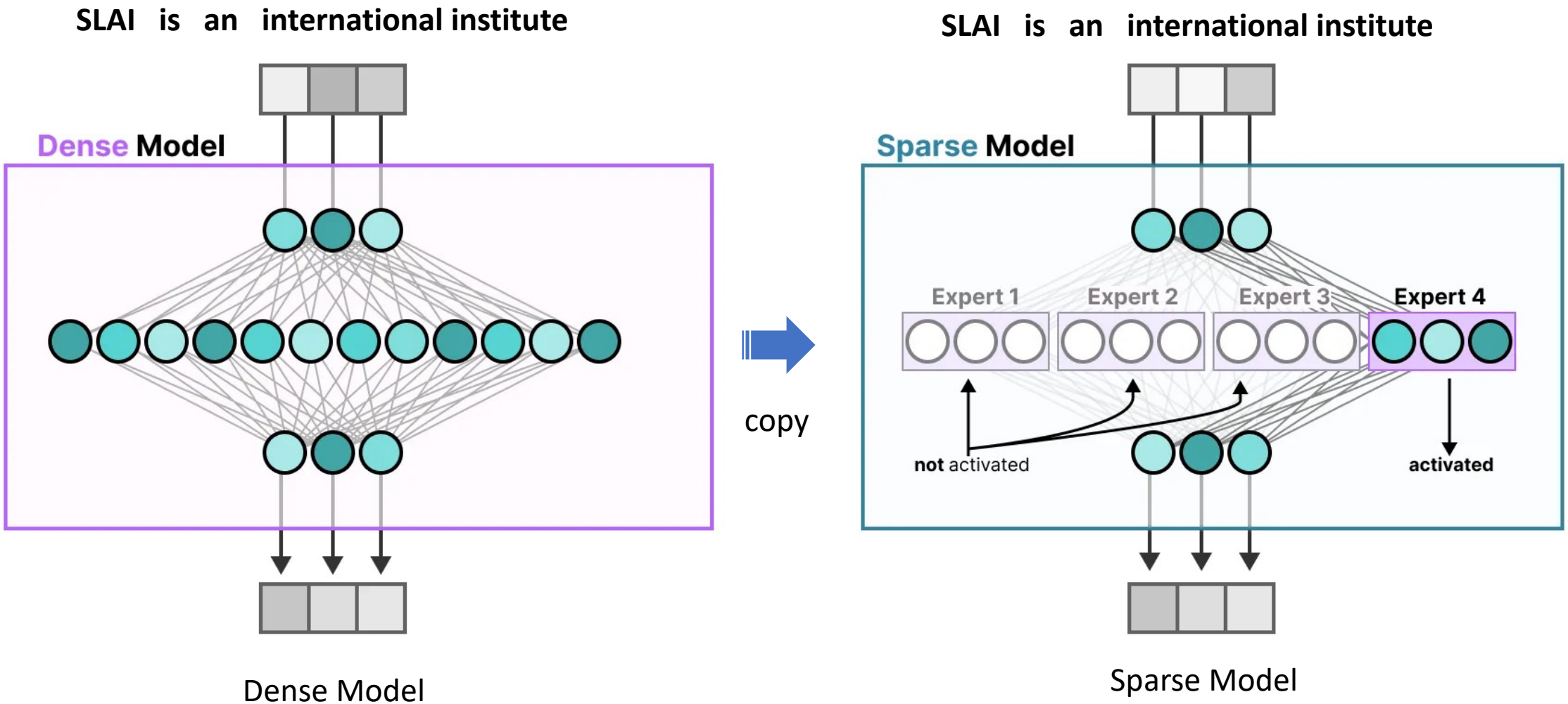
Mixture of Experts

- Dense Transformer block



《A Visual Guide to Mixture of Experts (MoE)》

Mixture of Experts

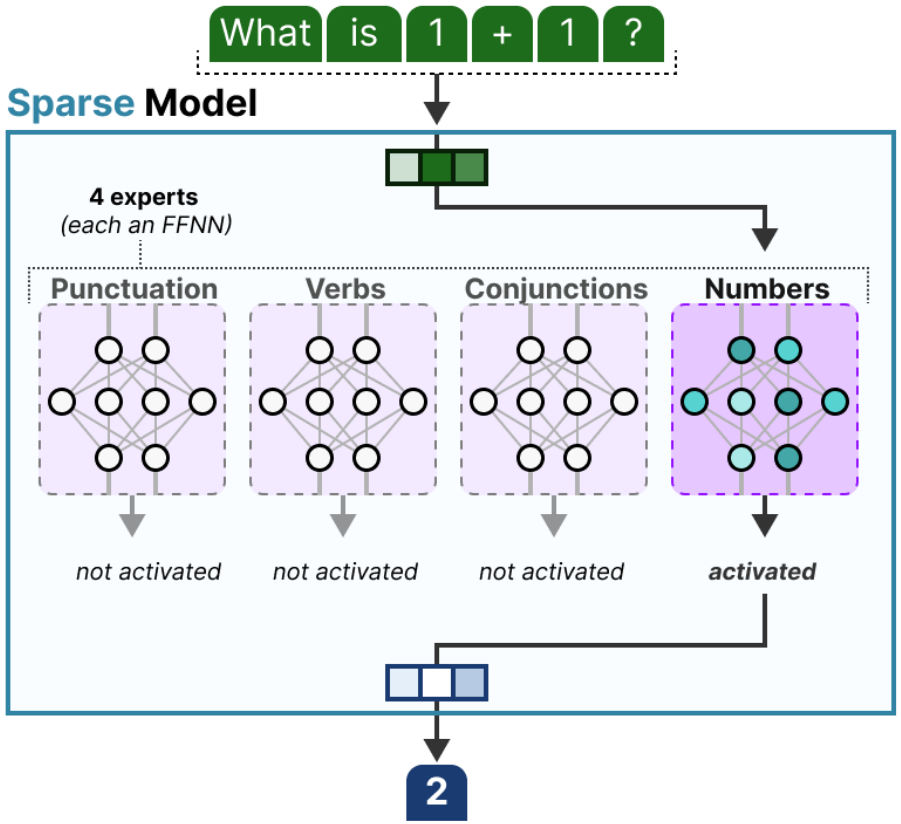


Dense Model

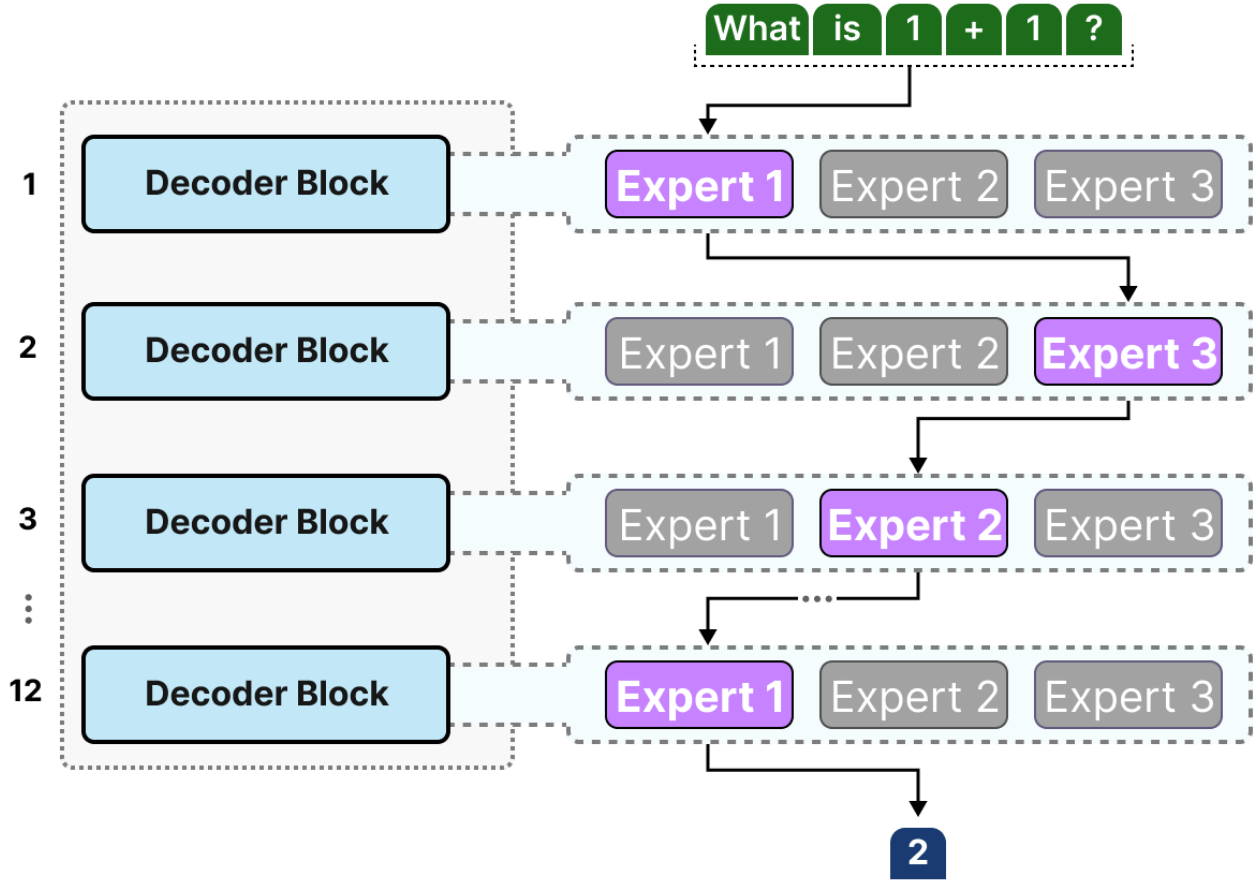
Sparse Model

《A Visual Guide to Mixture of Experts (MoE)》

Mixture of Experts



stacking



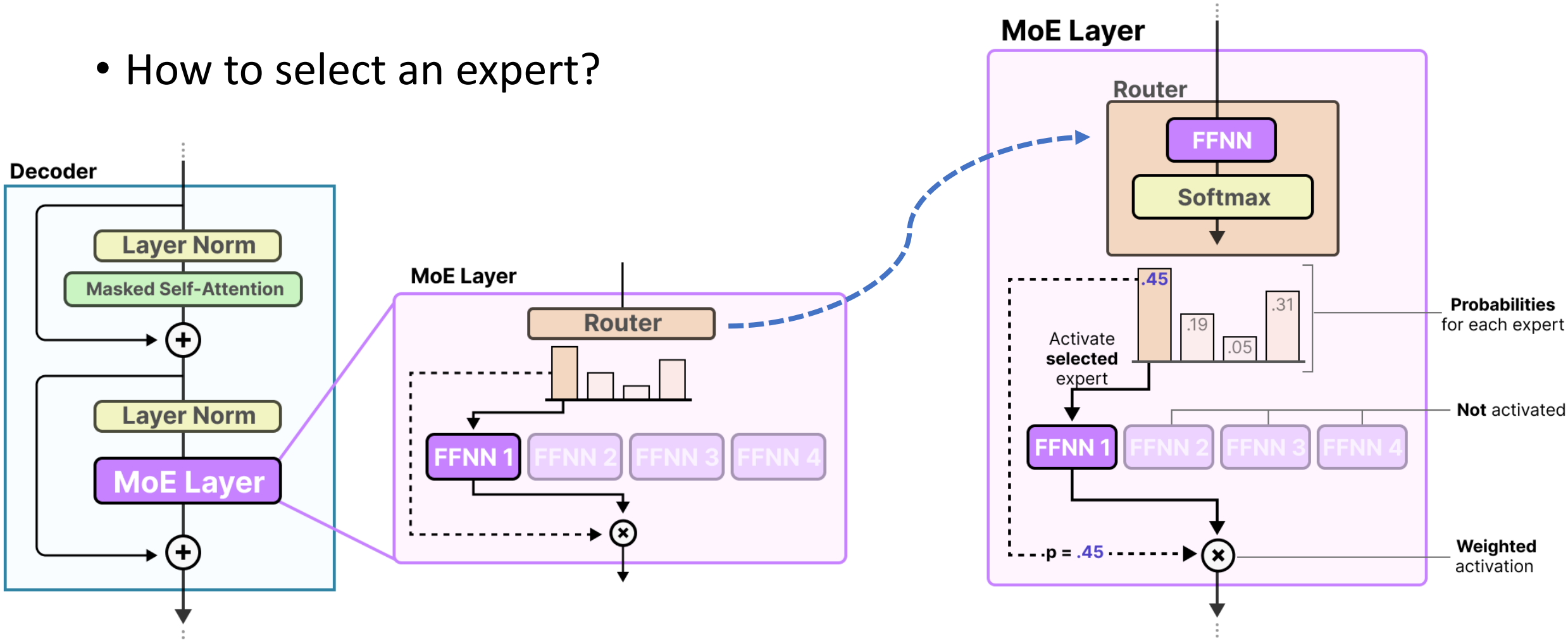
Each expert specializes in different syntactic tokens

More than one decoders → The chosen experts likely differ between tokens, and each token chooses different experts when traversing different encoders → a path exists

《A Visual Guide to Mixture of Experts (MoE)》

Mixture of Experts

- How to select an expert?



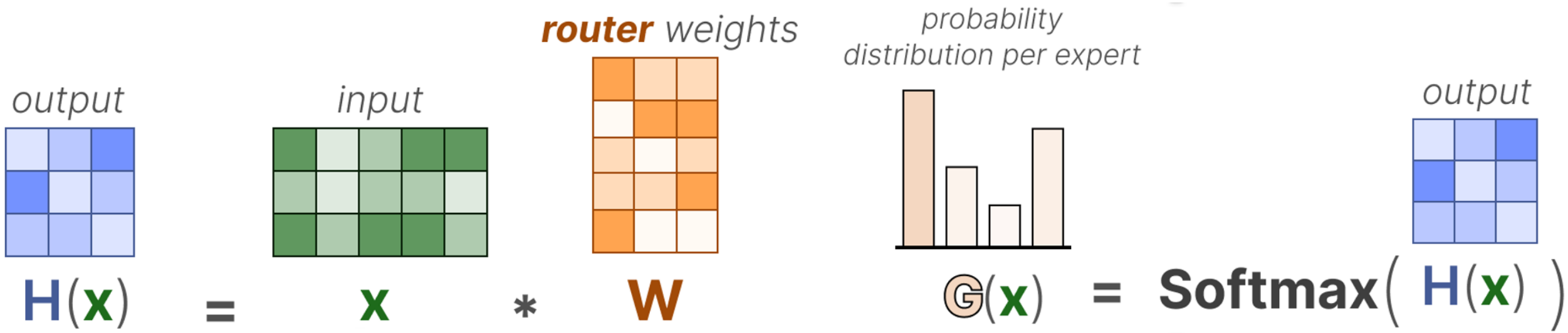
Transformer layer

Determines which tokens are sent to which experts

《A Visual Guide to Mixture of Experts (MoE)》

Mixture of Experts

- How to select an expert?

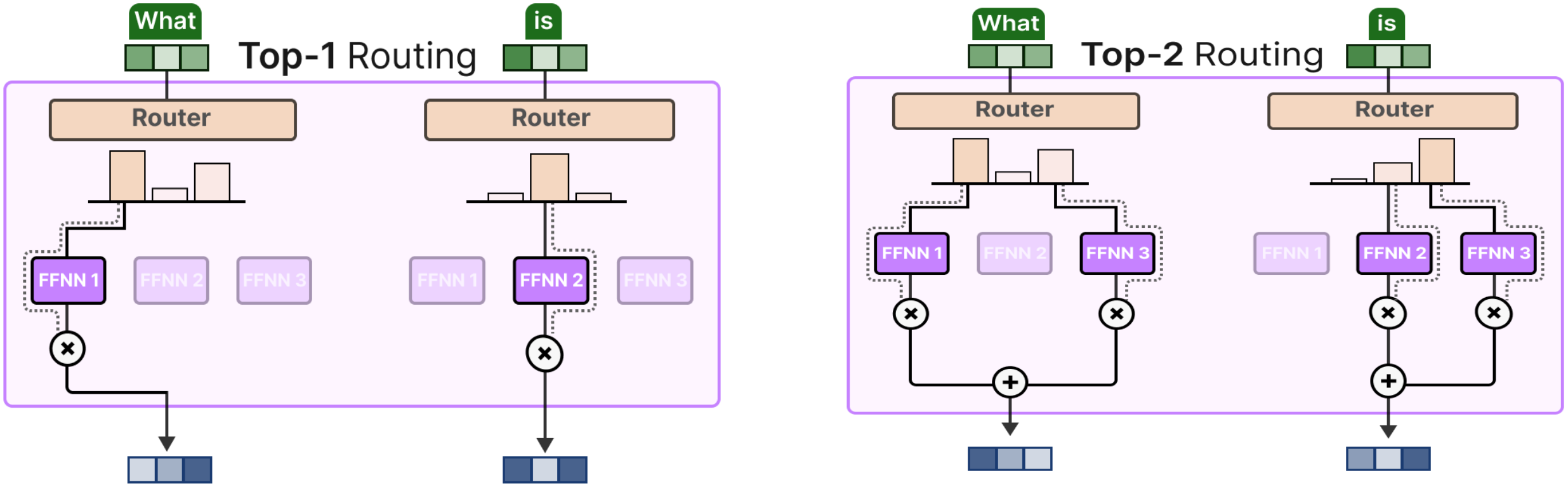


A very basic router

《A Visual Guide to Mixture of Experts (MoE)》

Mixture of Experts

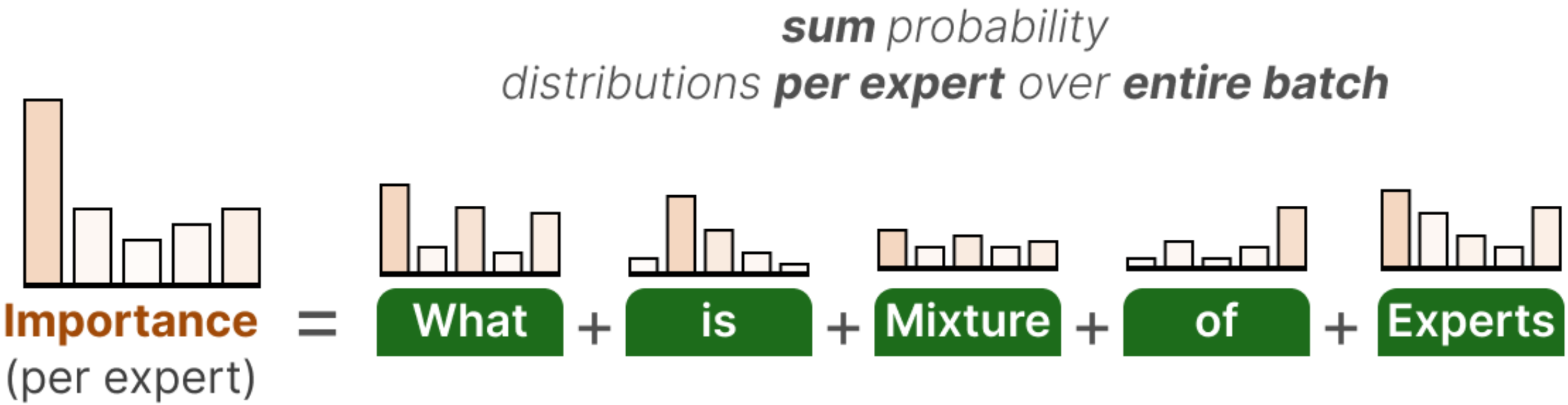
- Top-K routing



《A Visual Guide to Mixture of Experts (MoE)》

Mixture of Experts

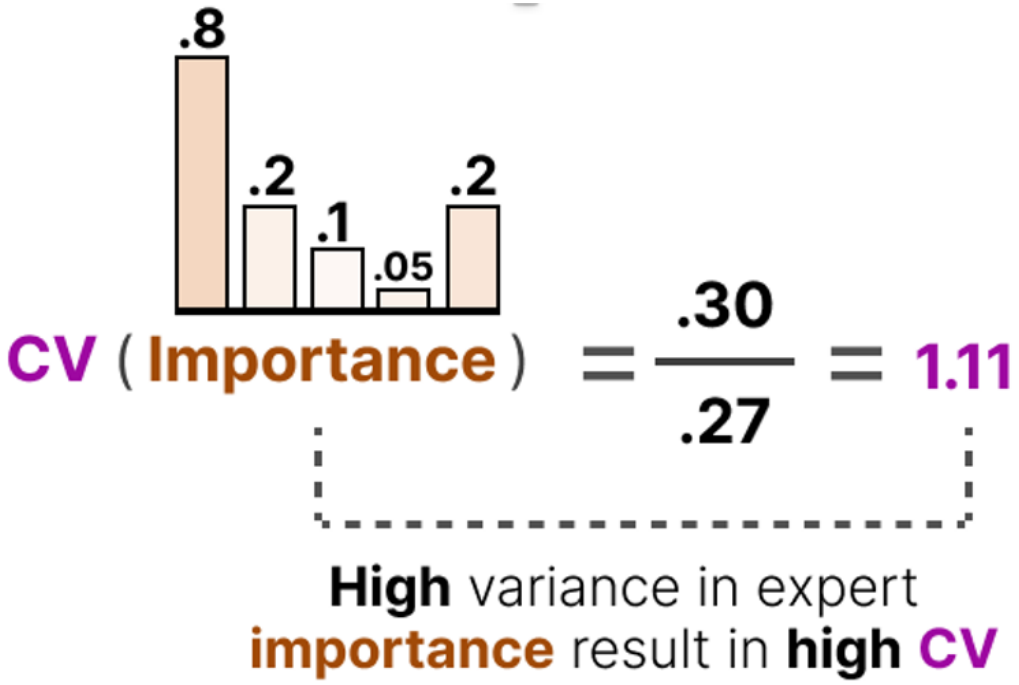
- Load balancing from an algorithmic perspective
  - Quantifying importance of experts



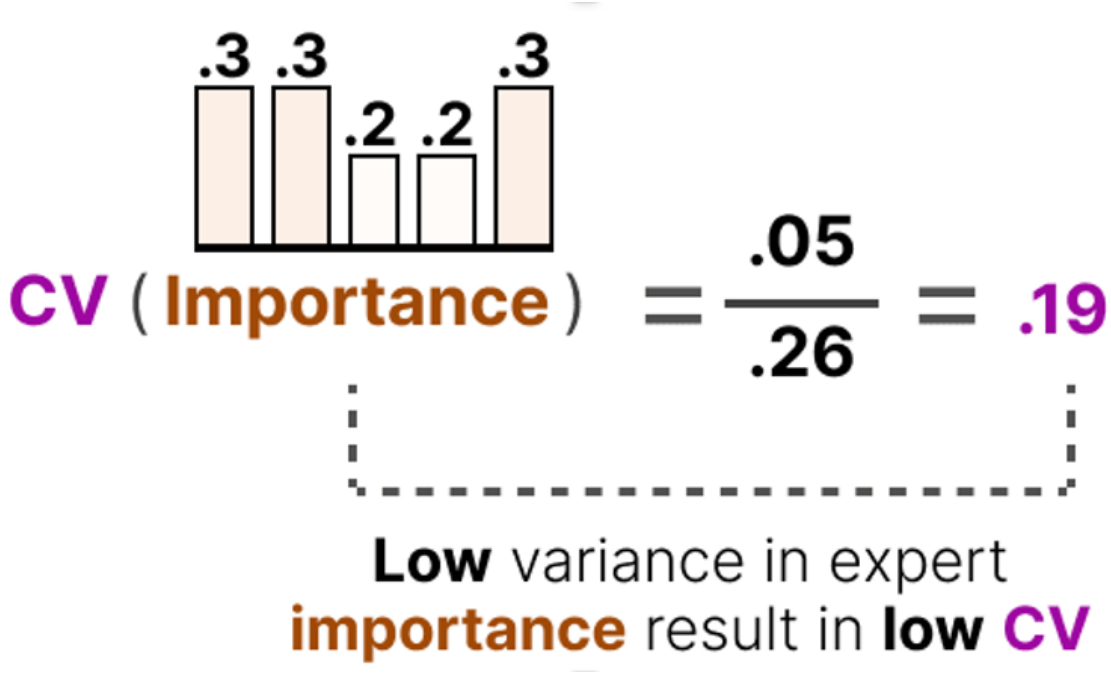
《A Visual Guide to Mixture of Experts (MoE)》

Mixture of Experts

- Load balancing from an algorithmic perspective
  - Coefficient of variation to capture the diversity across experts



Unbalanced

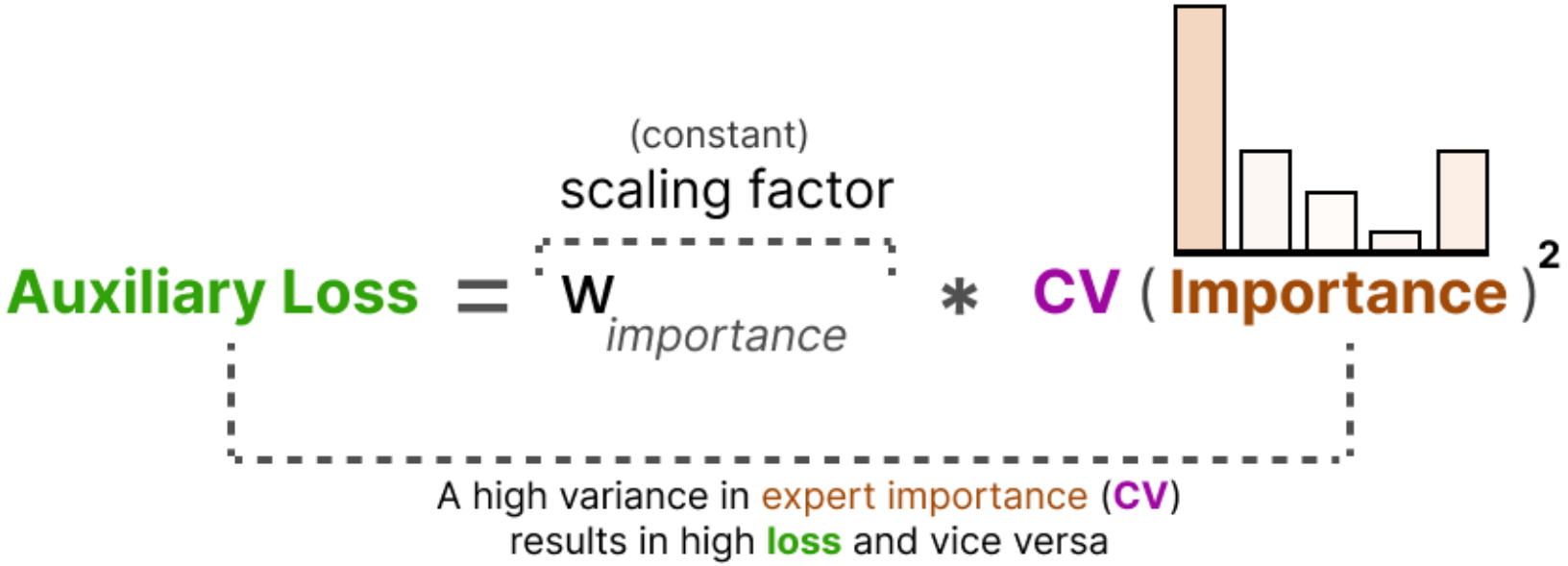


More balanced

《A Visual Guide to Mixture of Experts (MoE)》

Mixture of Experts

- Load balancing from an algorithmic perspective
  - Adding “auxiliary loss” to the overall training objective function

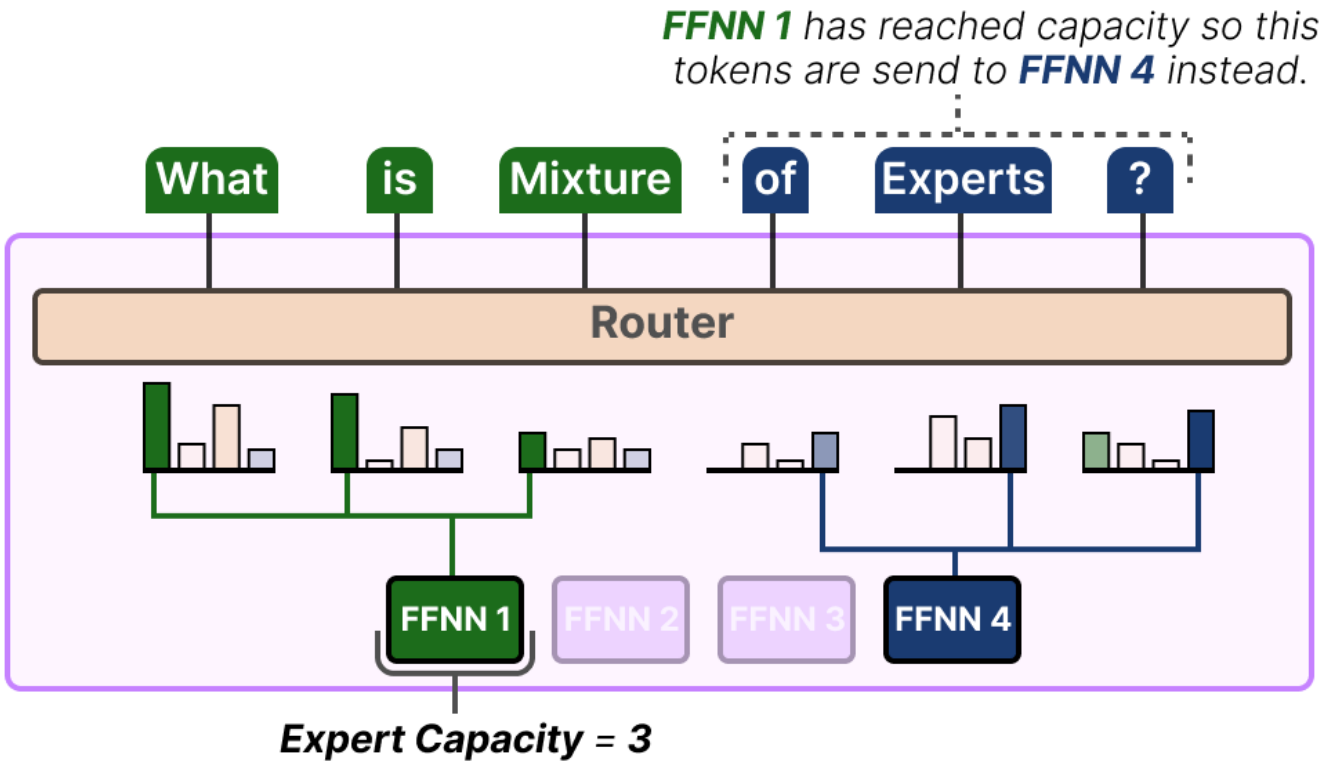


$$l_{aux} = \frac{1}{E}\sum_{e=1}^{E}\frac{c_e}{S} * m_e$$

《A Visual Guide to Mixture of Experts (MoE)》

Mixture of Experts

- Expert Capacity
  - limiting the amount of tokens that a given expert can handle



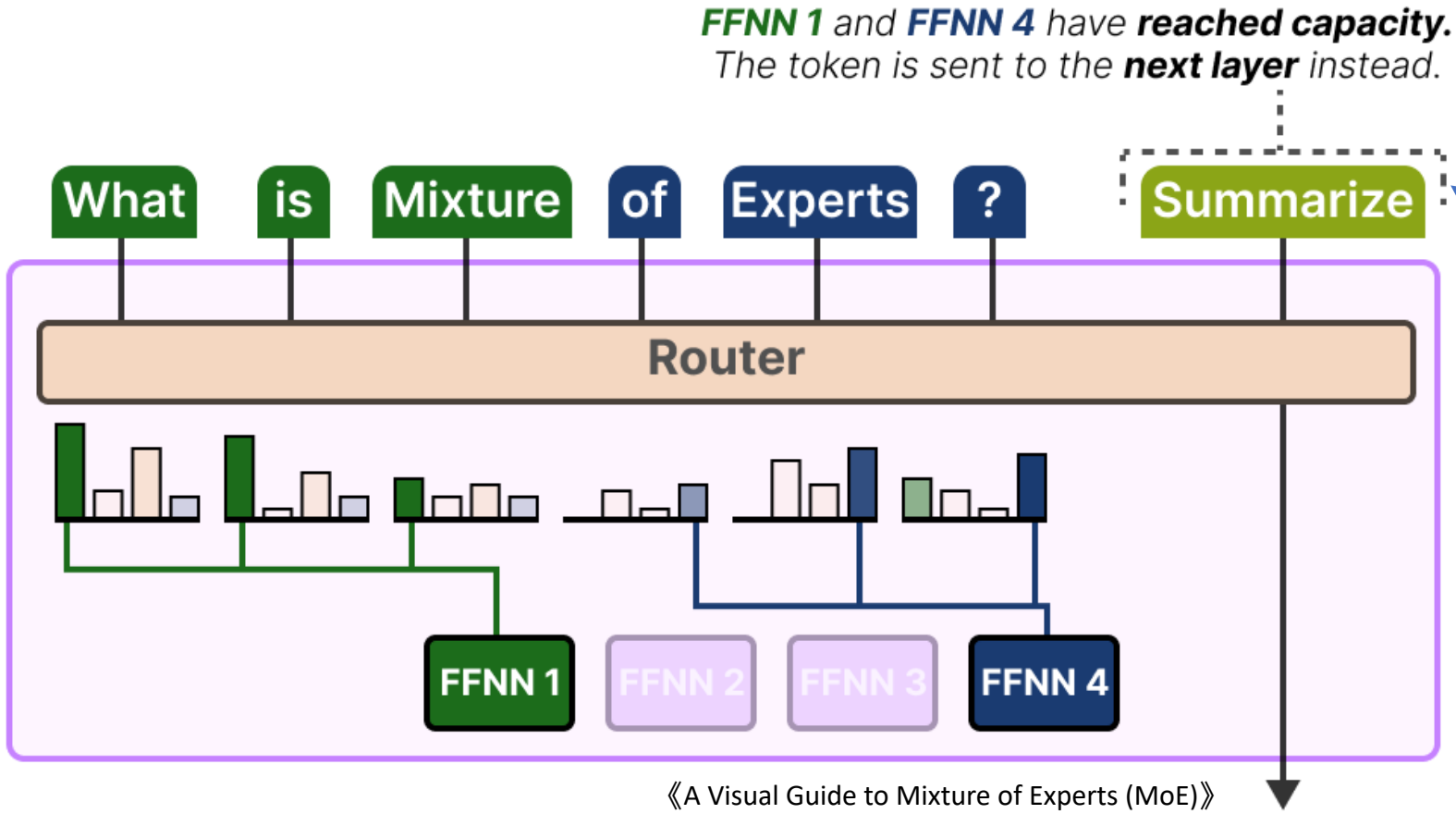
$$capacity = max(\frac{S}{E} * K * capacity\_factor, min\_capacity)$$

- Expert Capacity
  - S: # of tokens
  - E: # of experts
  - K: Tok-K parameter

《A Visual Guide to Mixture of Experts (MoE)》

Mixture of Experts

- Expert Capacity
  - limiting the amount of tokens that a given expert can handle

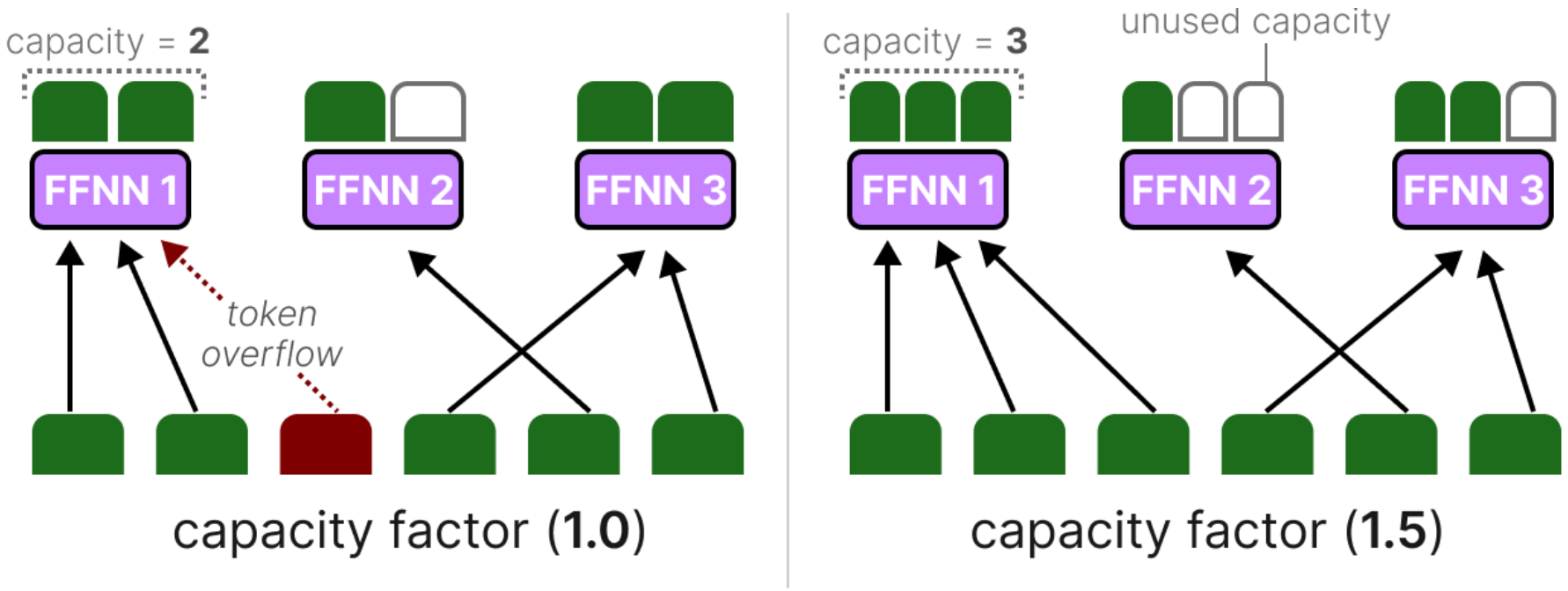


Overflow: not trained by any expert

《A Visual Guide to Mixture of Experts (MoE)》

Mixture of Experts

- Adjusting Expert Capacity

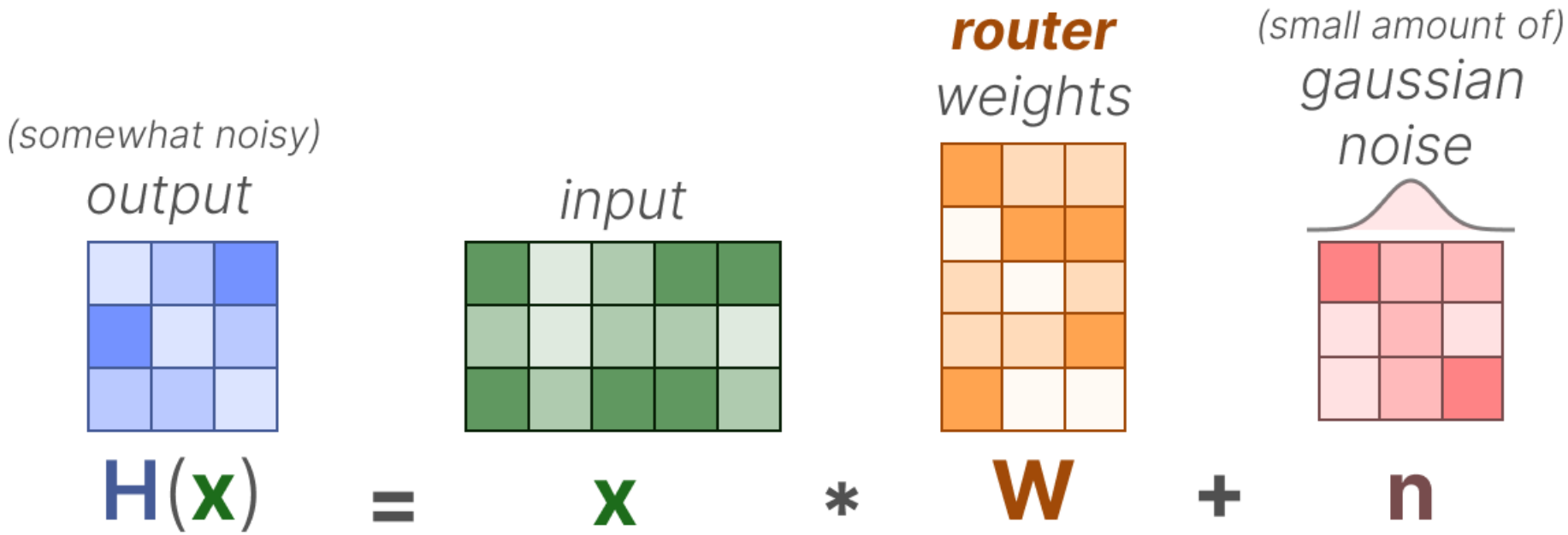


Increasing the capacity factor increases the quality but increases communication costs and memory of activations.

《A Visual Guide to Mixture of Experts (MoE)》

Mixture of Experts

- Random Routing/Noisy Routing

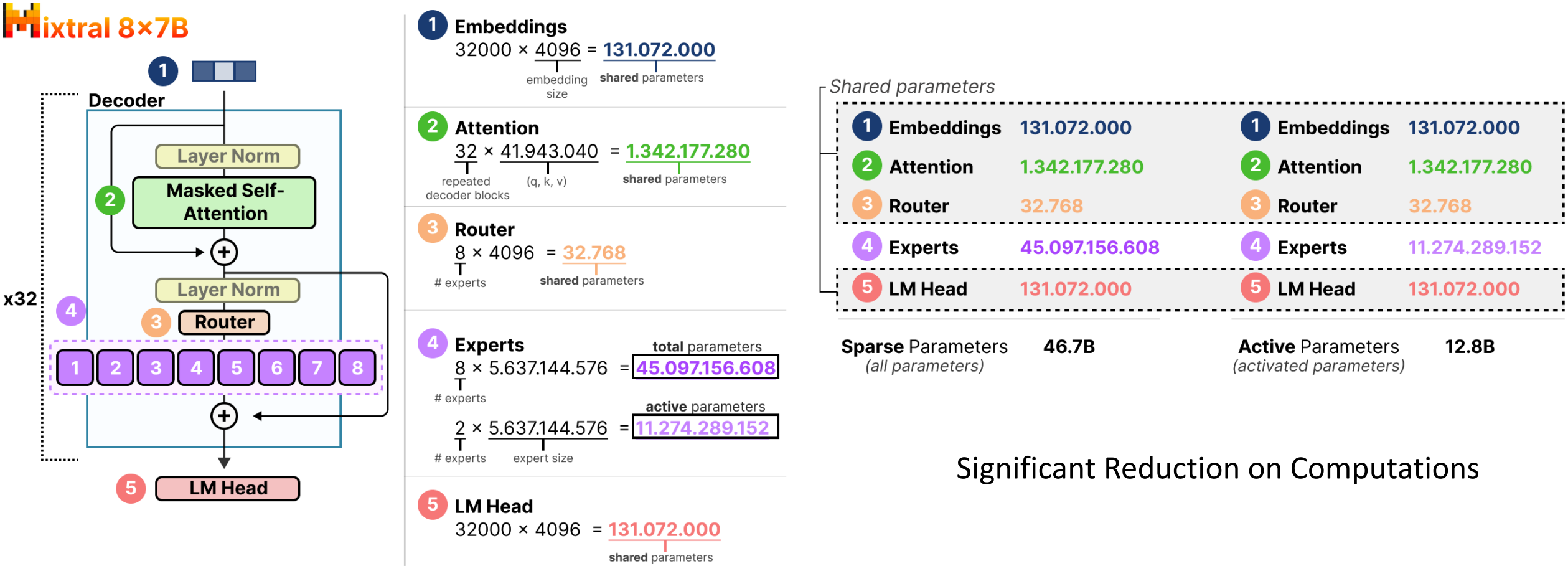


preventing the same experts from always being selected

《A Visual Guide to Mixture of Experts (MoE)》

Mixture of Experts

- Scaling up model parameters



Significant Reduction on Computations

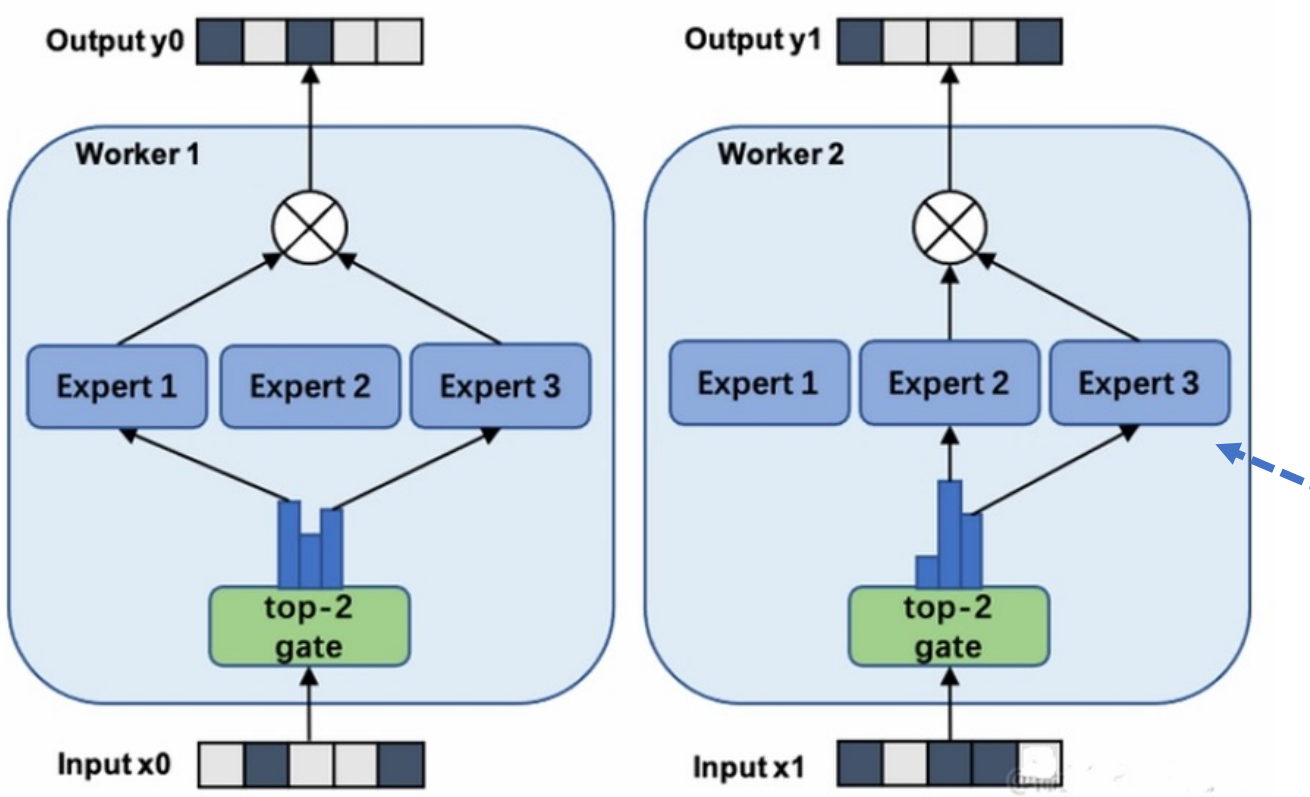
《A Visual Guide to Mixture of Experts (MoE)》

Distributed LLM Training: Outline

- Data Parallelism
- Model Parallelism
- Mixture of Experts
  - Principles
  - **Parallel Training**
  - Recent Progresses

Mixture of Experts

- Parallel Training
  - How to distribute experts on different GPUs?



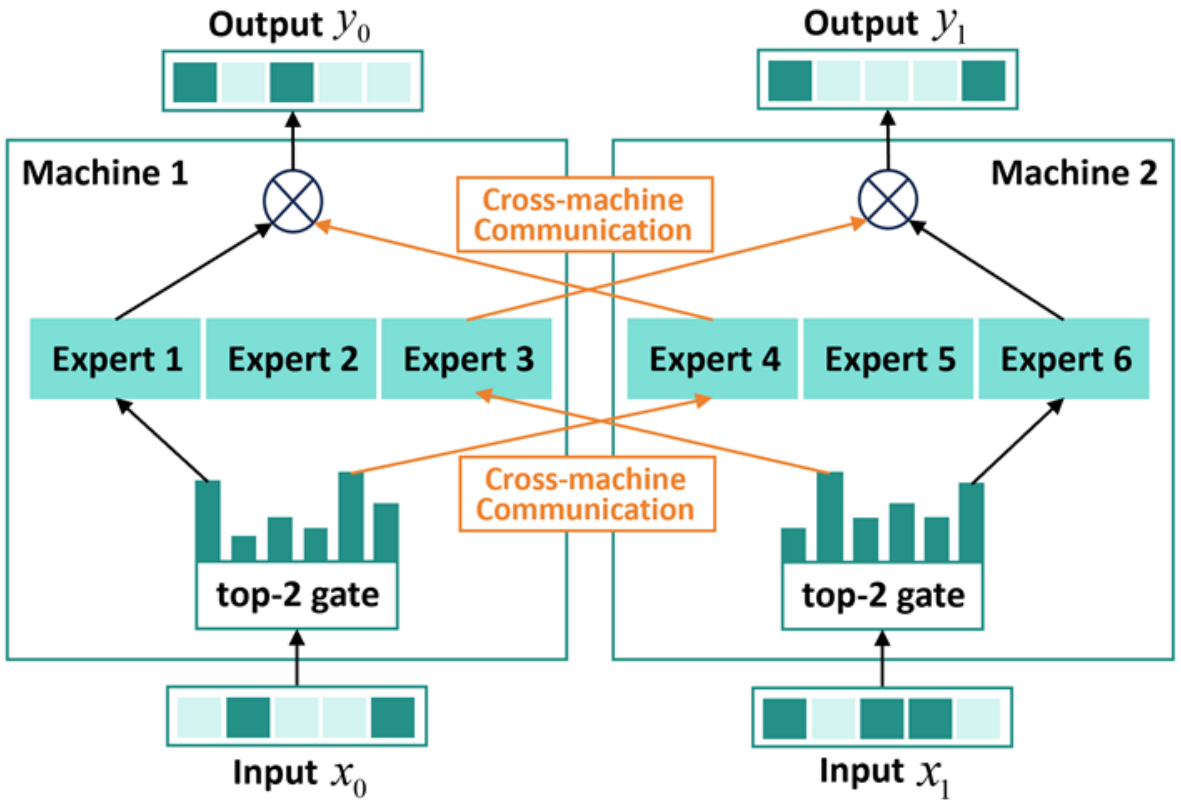
- Data and Expert Parallelism
  - Small # of experts
  - Experts are small in sizes

Every GPU/Machine stores the complete replicas of experts

《FastMoE and Wu Dao 2.0 - behind the scenes with two authors of the recent Chinese AI breakthrough》

Mixture of Experts

- Parallel Training
  - How to distribute experts on different GPUs?

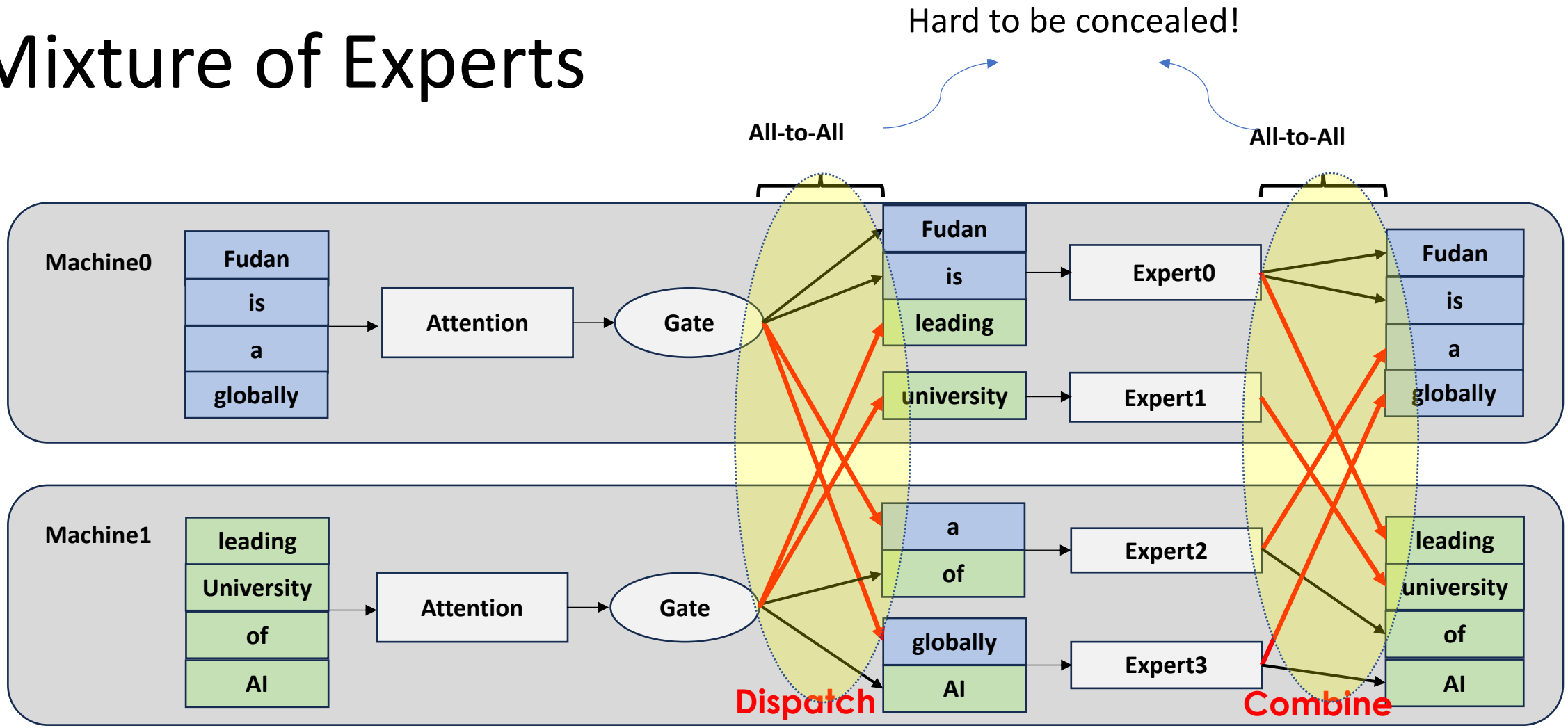


6 experts, DP = 2, EP = 2 (assuming one machine one GPU)

- Data and Expert Parallelism
  - Scaling up to gigantic experts
  - Experts are placed across DP groups
- Size of a "token"
  - *hidden_dim* * 2 Bytes (bf16)
  - Examples
    - QWEN-235B: 2*4096 = 8192 Bytes
    - DeepSeek-v3: 2*7168 = 14336 Bytes
- Traffic volume
  - Proportional to batch size, sequence length, hidden dimension, top-K

Cross-Machine Communication

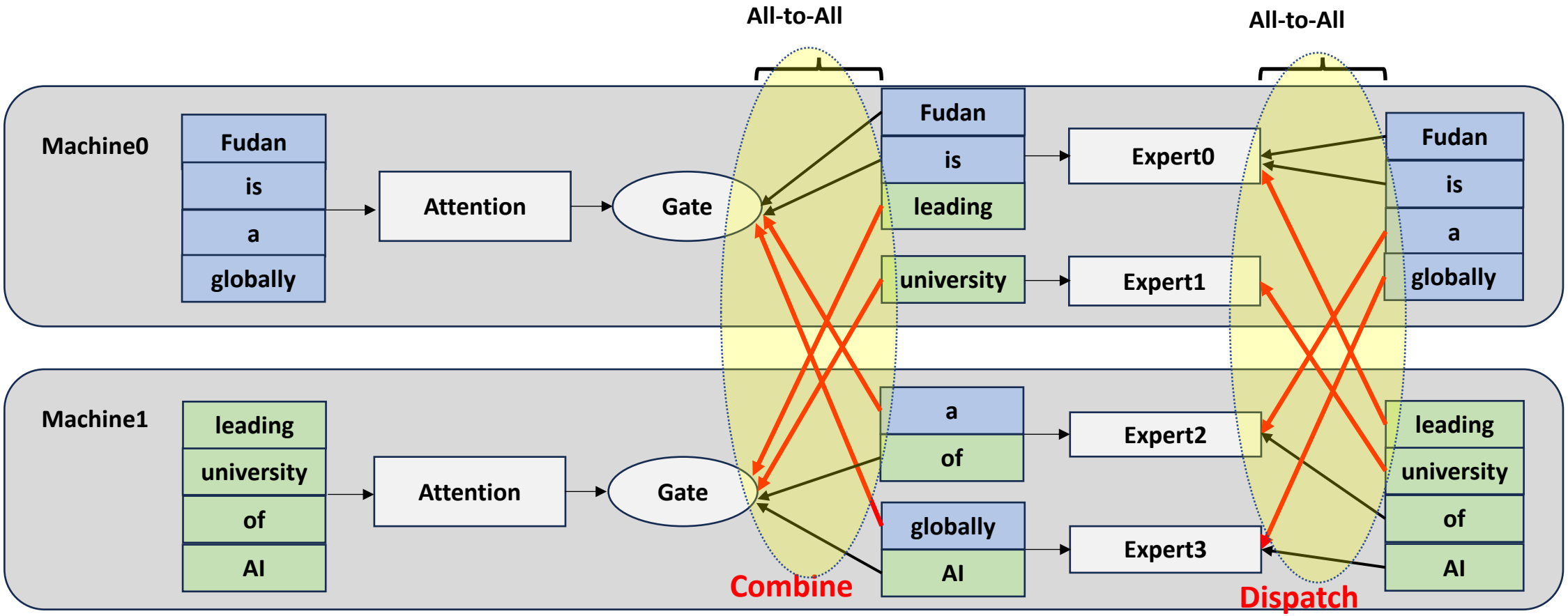
Mixture of Experts



Compute	Communication	Compute	Communication

Forward Pass: heavy token communications that hard to be concealed

Mixture of Experts



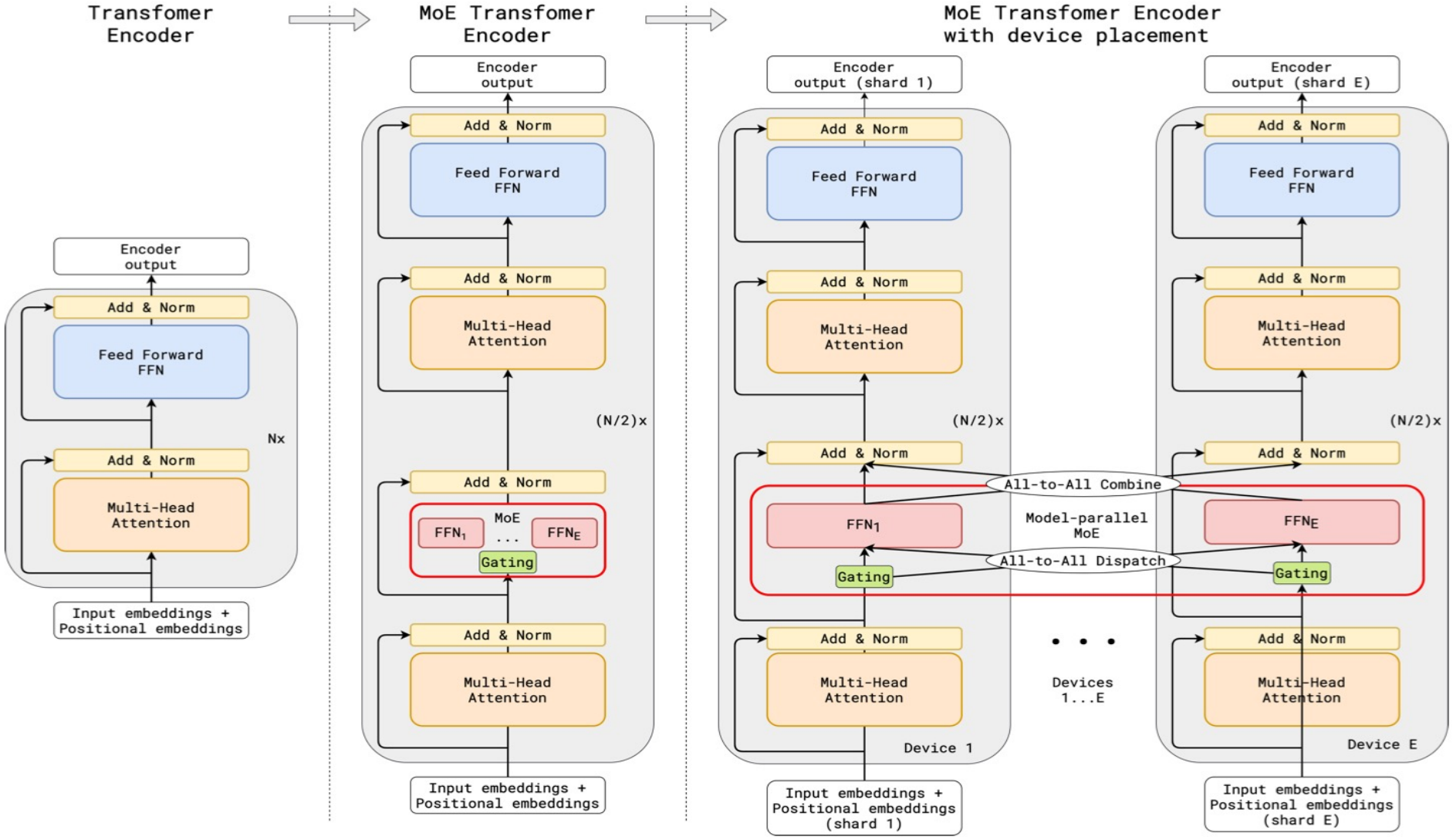
Compute | Communication | Compute | Communication

Backward Pass: heavy token communications that hard to be concealed

Mixture of Experts: Traffic Pattern

Forward: Dispatch and Combine

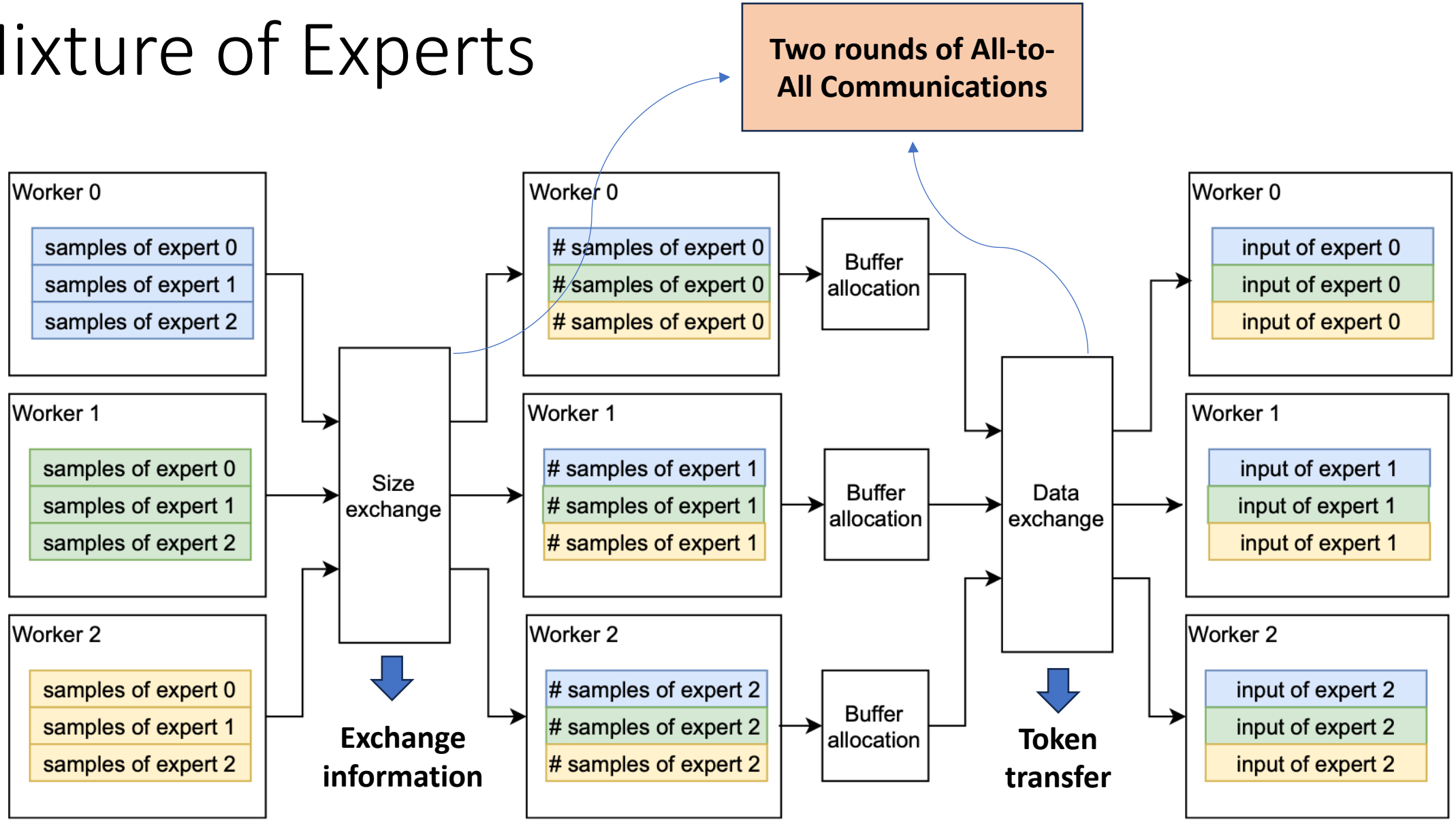
Backward: Dispatch and Combine



GShard (2020)

- MoE within a single machine: forward and backward
- MoE spanning multiple machines: all-to-all communications

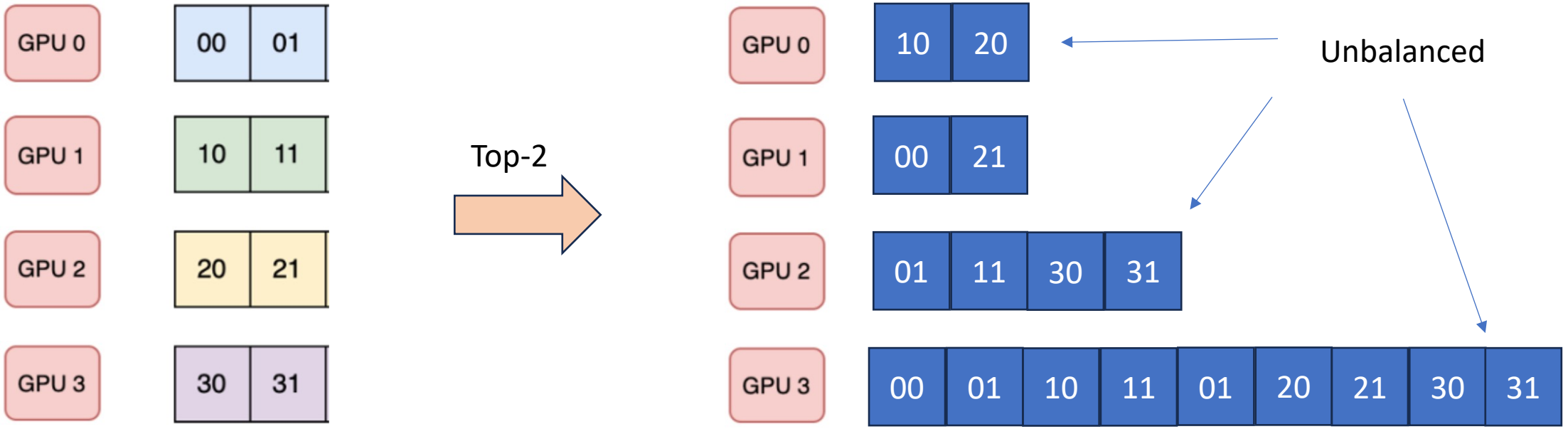
Mixture of Experts



FastMoE (2021): First MoE training system that supports Pytorch

Mixture of Experts

- All-to-All Communication
  - Asymmetric data volume (tokens) between GPUs hosting experts
    - **ZERO padding** to transmit ***dummy*** tokens under standard All-to-All?



Mixture of Experts

- All-to-All Communication
  - NCCL (*NVIDIA Collective Commun. Library*): no all-to-all implementation *(before NCCL 2.7)*
    - ***ncclSend*** and ***ncclRecv*** point-to-point communication instead

```
ncclGroupStart();
for (int r=0; r<nranks; r++) {
  ncclSend(sendbuff[r], sendcount, sendtype, r, comm, stream);
  ncclRecv(recvbuff[r], recvcount, recvtype, r, comm, stream);
}
ncclGroupEnd();
```

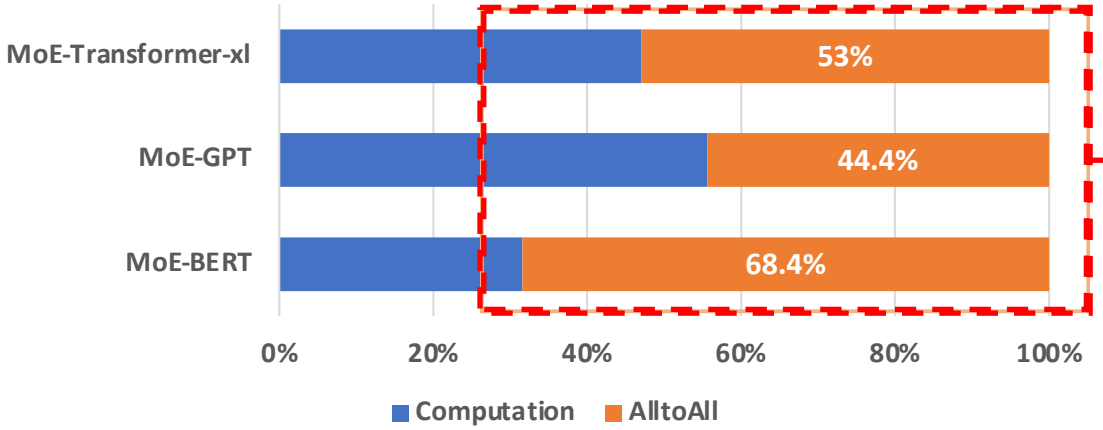
    - Refer to new NCCL version for all-to-all implementation

Mixture of Experts

- MoE All-to-All Communication Practice
  - Dispatch
    - **Gathering # of tokens to send and receive**: one standard All-to-All to exchange the # of tokens to transfer, e.g. via ***torch.distributed.all_to_all_single*** in DeepSpeed-MoE
    - Creating sending and receiving buffers, e.g. by ***append***ing or ***concatenat***ing vectors
    - Exchanging token vectors: non-standard All-to-All, e.g. via ***torch.distributed.all_to_all_single*** while the # of tokens differ
    - Assigning received tokens to local experts
  - Combine
    - Creating sending and receiving buffers
    - Exchanging the expert outputs and meta data to the corresponding nodes

Mixture of Experts

- Challenge in MoE All-to-All



iteration time vs **time of All-to-All** (32 GPUs)

All-to-All communication accounts for a significant proportion (44.4% ~ 68.4%),

	#Parameter	#layer	#AlltoAll per iteration
GPT* + MoE	175 B	96	96 * 4 = 384 times

Upscaling with # of transformer blocks

Distributed LLM Training: Outline

- Data Parallelism
- Model Parallelism
- Mixture of Experts
  - Principles
  - Parallel Training
  - **Recent Progresses**

Mixture of Experts

- Optimizing MoE All-to-All Communication
  - Expert-centric: keeping **experts in-place** and moving tokens via All-to-All
  - Data-centric: keeping **tokens in-place** and moving experts between GPUs

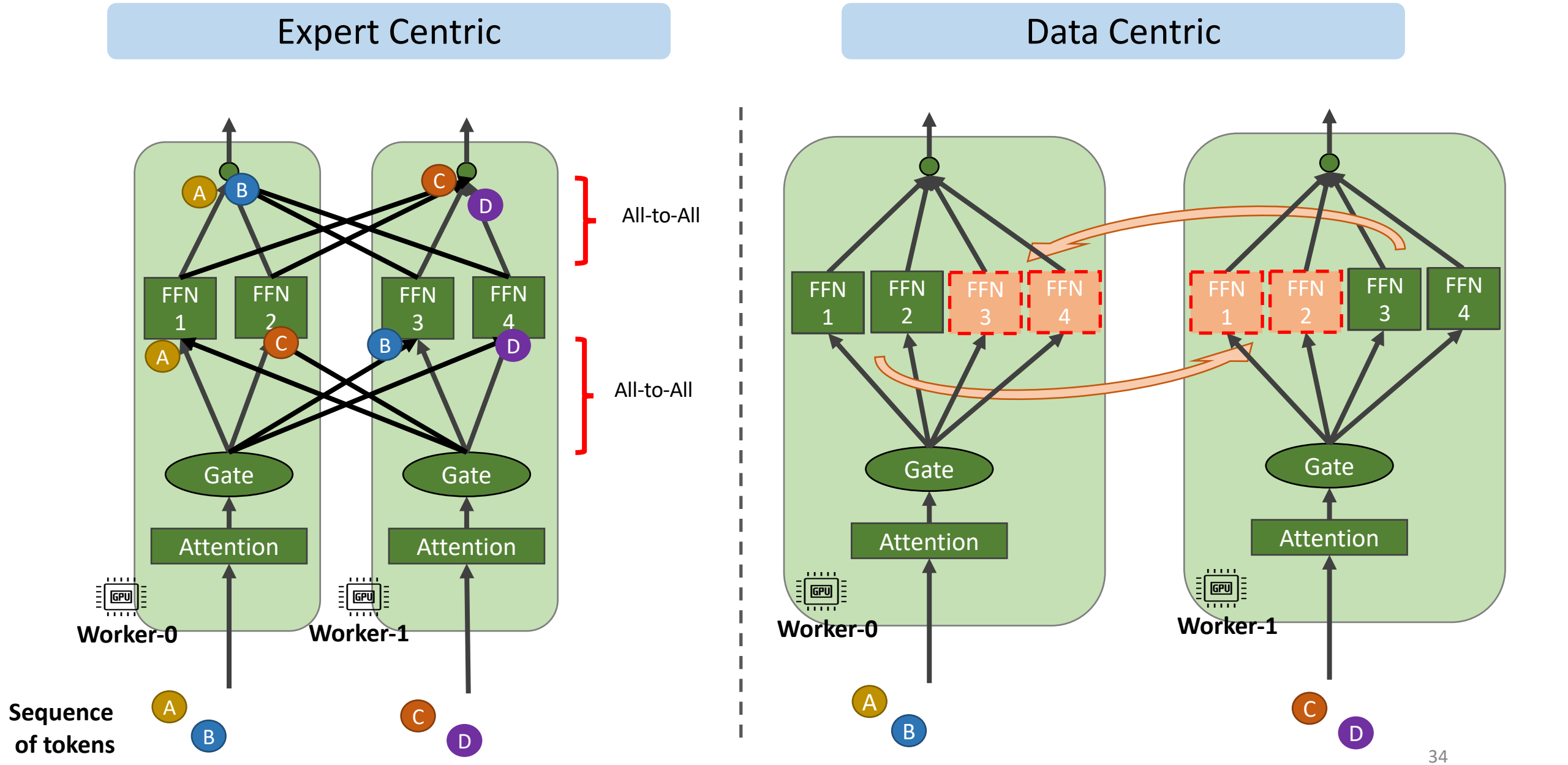
Case Study:

Running on 4 8-A100 machines (32 GPUs)
Batch=64, Seq=512, Gate's topk=2, hidden size=256, #expert=32

	Operation	**Volume of Traffic across a machine**
Expert Centric	send out tokens	**0.75GB**
Data Centric	fetch experts	**0.047GB**

Data Centric has lower traffic workload !

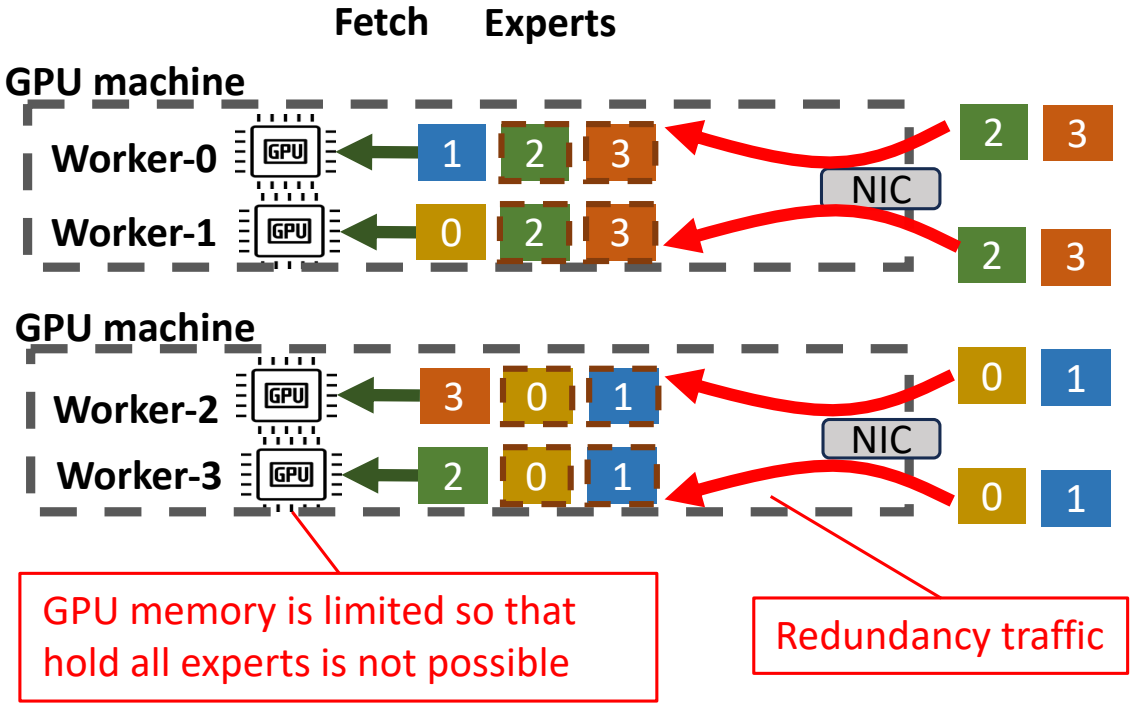
《Janus: A Unified Distributed Training Framework for Sparse Mixture-of-Experts Models》

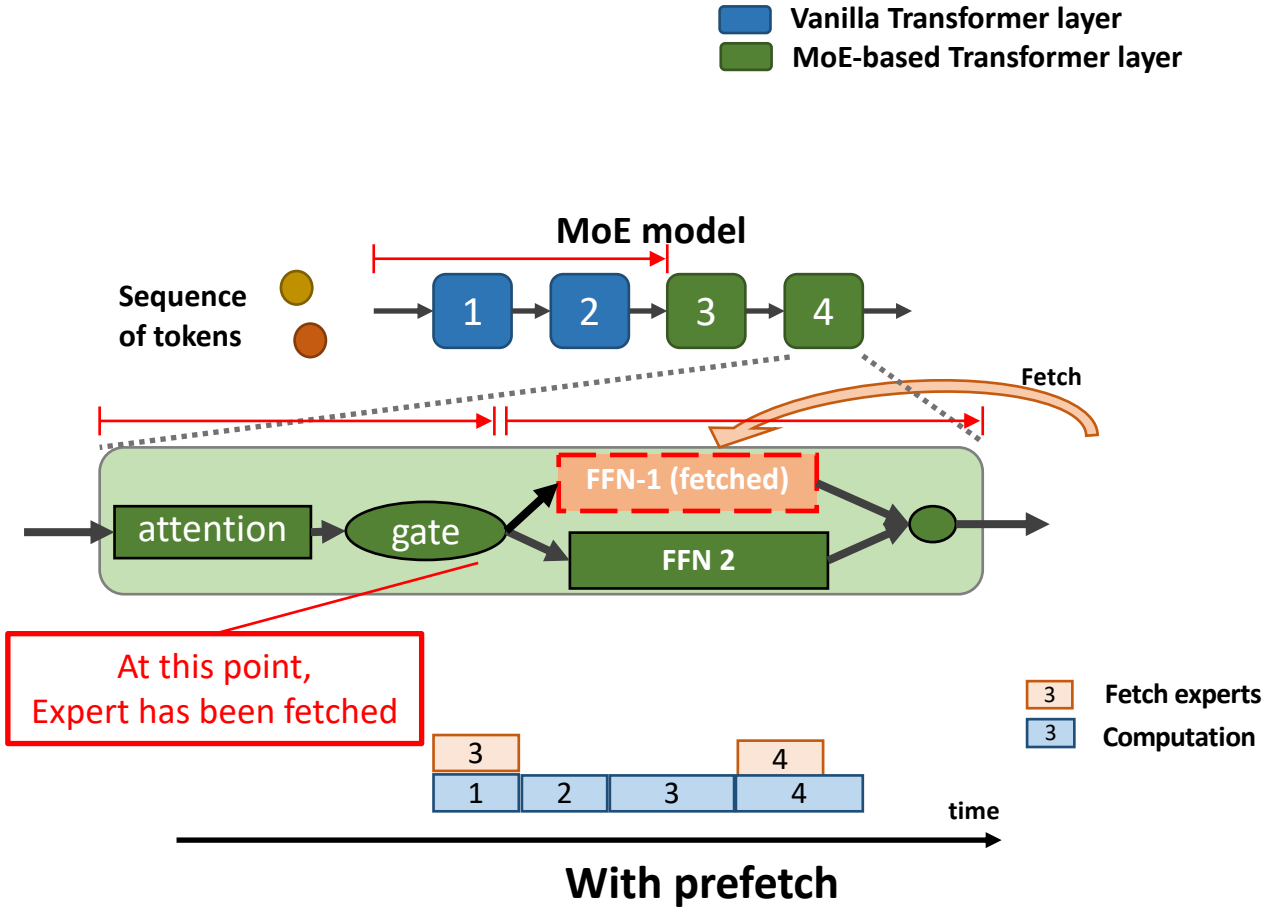


《Janus: A Unified Distributed Training Framework for Sparse Mixture-of-Experts Models》

Mixture of Experts

- Optimizing MoE All-to-All Communication
  - Potential challenges in Janus



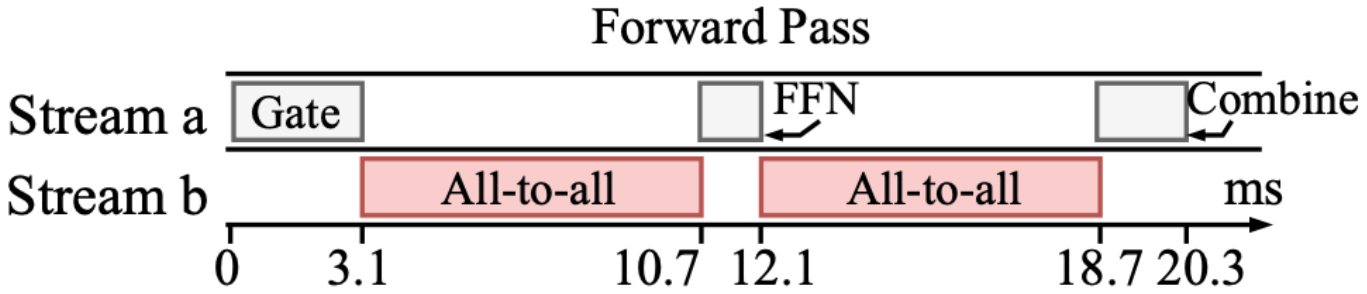


《Janus: A Unified Distributed Training Framework for Sparse Mixture-of-Experts Models》

Mixture of Experts

- Optimizing MoE All-to-All Communication
  - Lina (2023): forward pass is synchronous and blocking

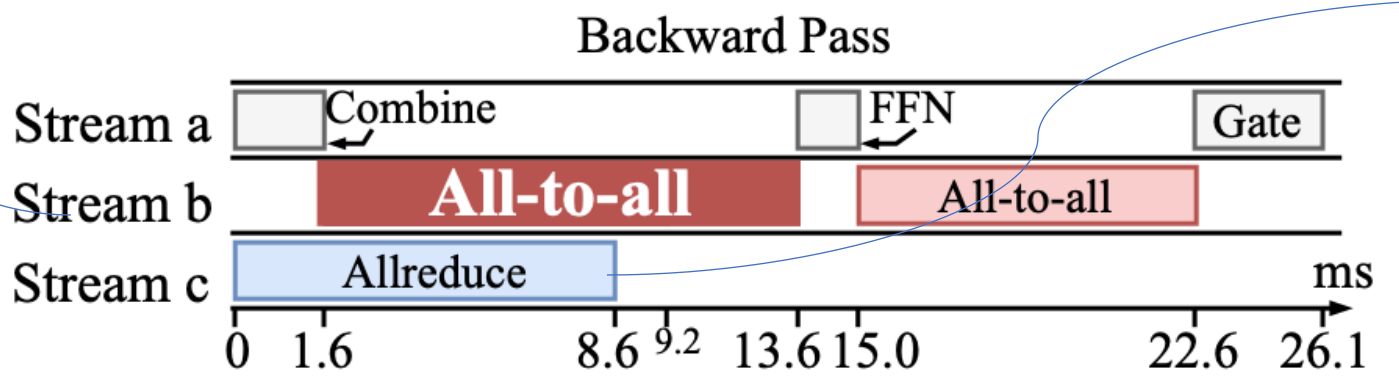
CUDA stream



  - Lina: backward pass is

Prolonged because of simultaneous non-expert AllReduce

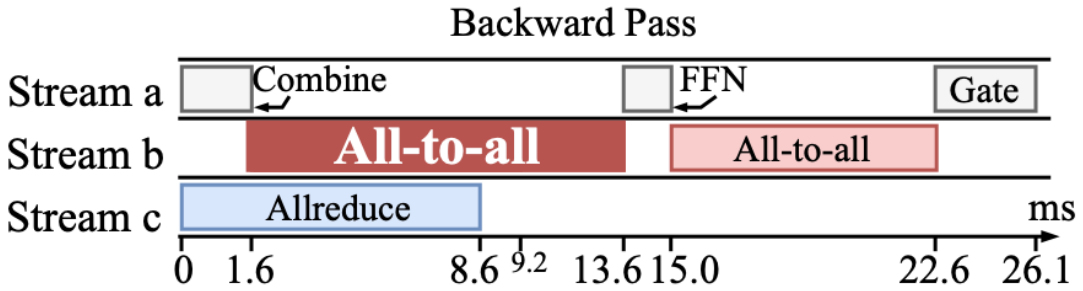
asynchronously aggregate non-expert gradients in data parallel



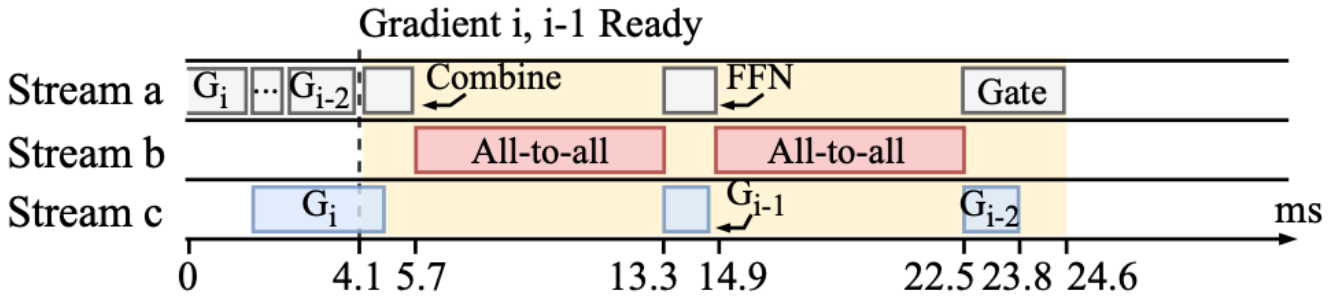
《Accelerating Distributed MoE Training and Inference with Lina》

Mixture of Experts

- Optimizing MoE All-to-All Communication
  - Lina: separate CUDA streams for the expert-parallel and data-parallel groups



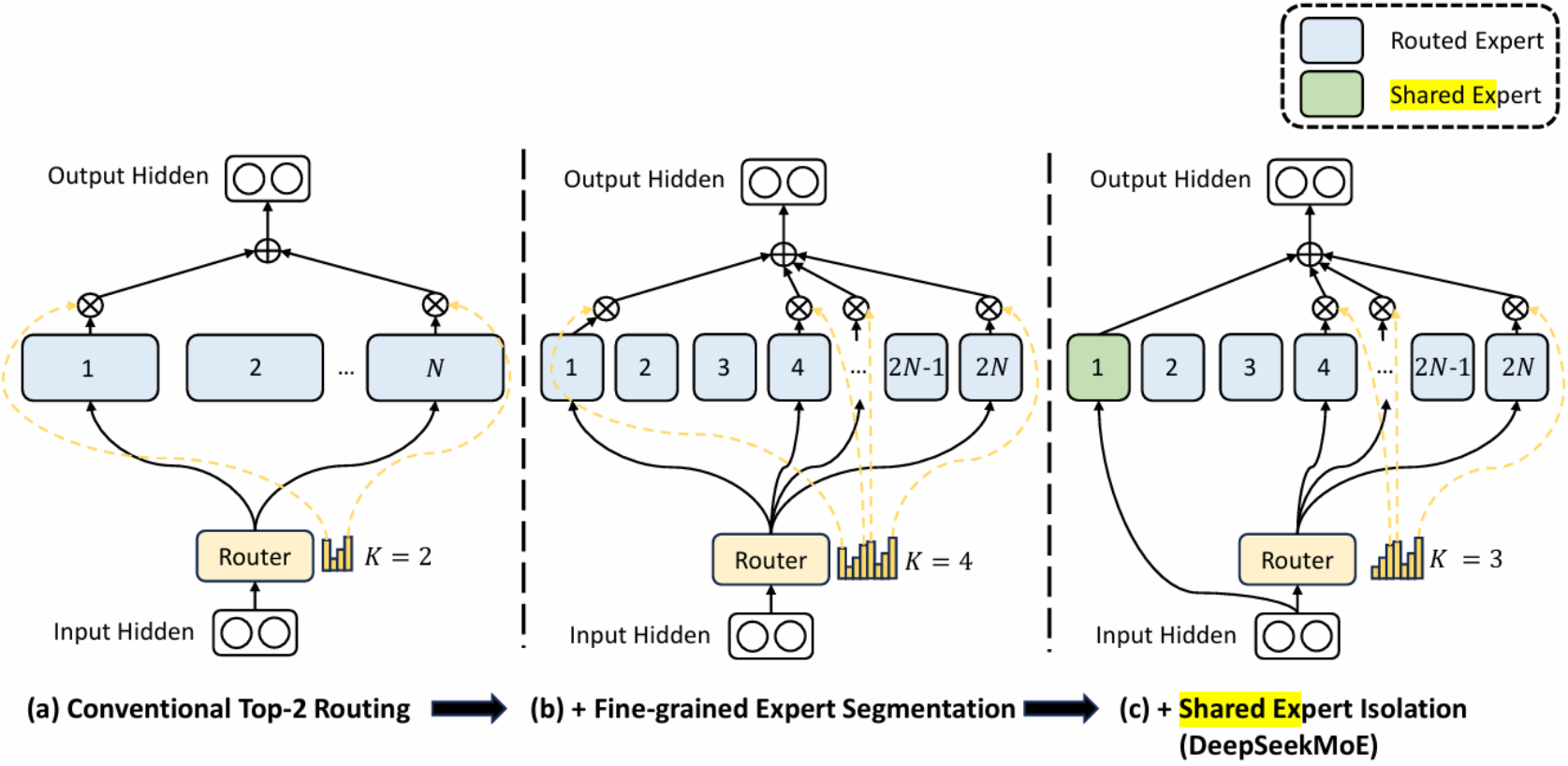
  - Lina: always prioritize all-to-all and avoid bandwidth sharing



  - More recent approaches: FSMoE (2025), ScheMoE (2024), etc.

Mixture of Experts

- Overlapping MoE All-to-All in DeepSeek v3



DeepSeek v3: 61 layers, 671B parameters, **256 routing experts**, **Top-8 selection,** 37B parameters activated by one token

Mixture of Experts

- Overlapping MoE All-to-All in DeepSeek v3

Token-expert affinity: $s_{i,t} = Sigmoid(u_t^T e_i)$

Token-expert affinity: $s_{i,t} = Sigmoid(u_t^T e_i) + b_i$

Adjusting b_i dynamically

Token-level load balancing

The Shenzhen Loop Area Institute (SLAI) is a premier global institution dedicated to artificial intelligence, serving as a national pilot for innovative AI talent cultivation under the auspices of China's Ministry of Education.

Sequence-level load balancing

Each token can be sent to at most 4 machines

Cross-machine restriction

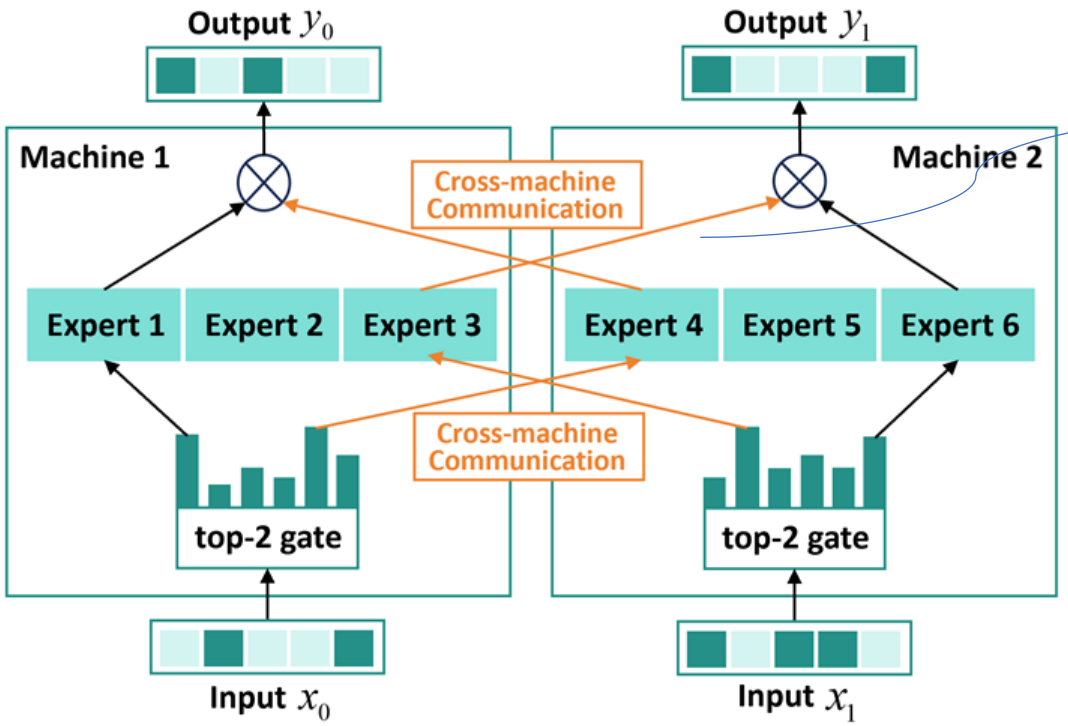
Mixture of Experts

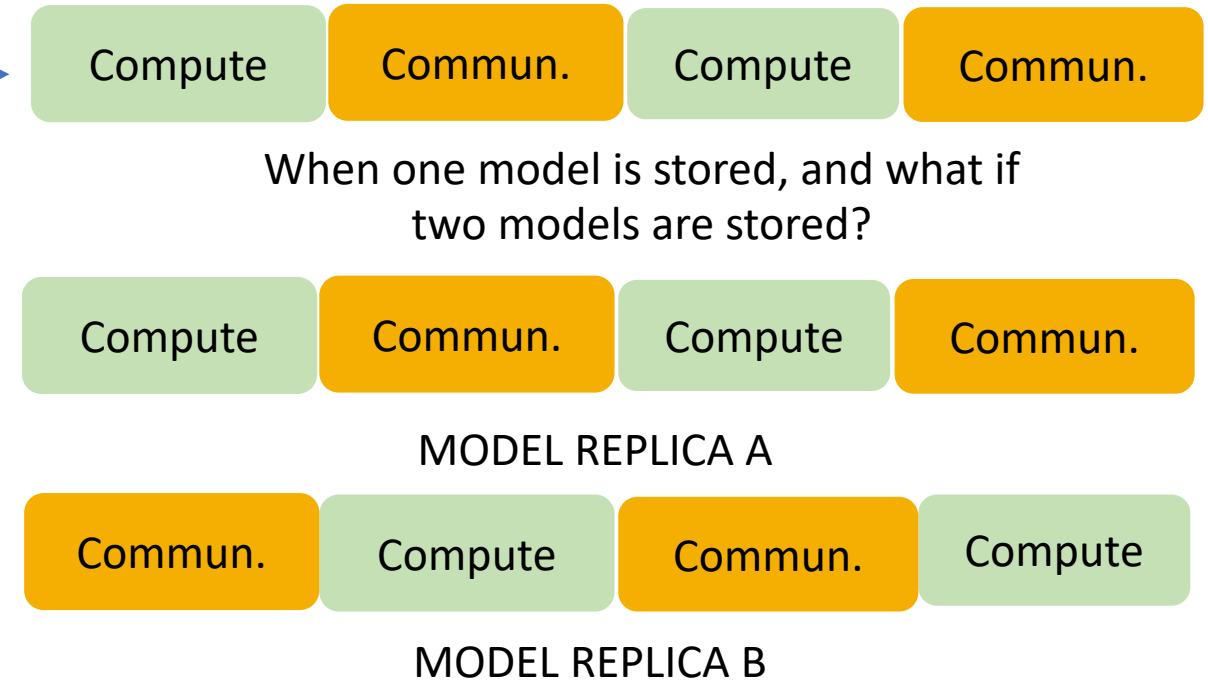
- Overlapping MoE All-to-All in DeepSeek v3
  - 1F1B pipeline: forward pass followed by the immediate backward pass
  - **ZERO bubble**: decomposing backward into backwards for input and parameter
  - **Chimera**: bidirectional pipeline for squeezing bubbles

DualPipe scheduling for 8 PP ranks and 20 micro-batches in two directions. The micro-batches in the reverse direction are symmetric to those in the forward direction

Mixture of Experts

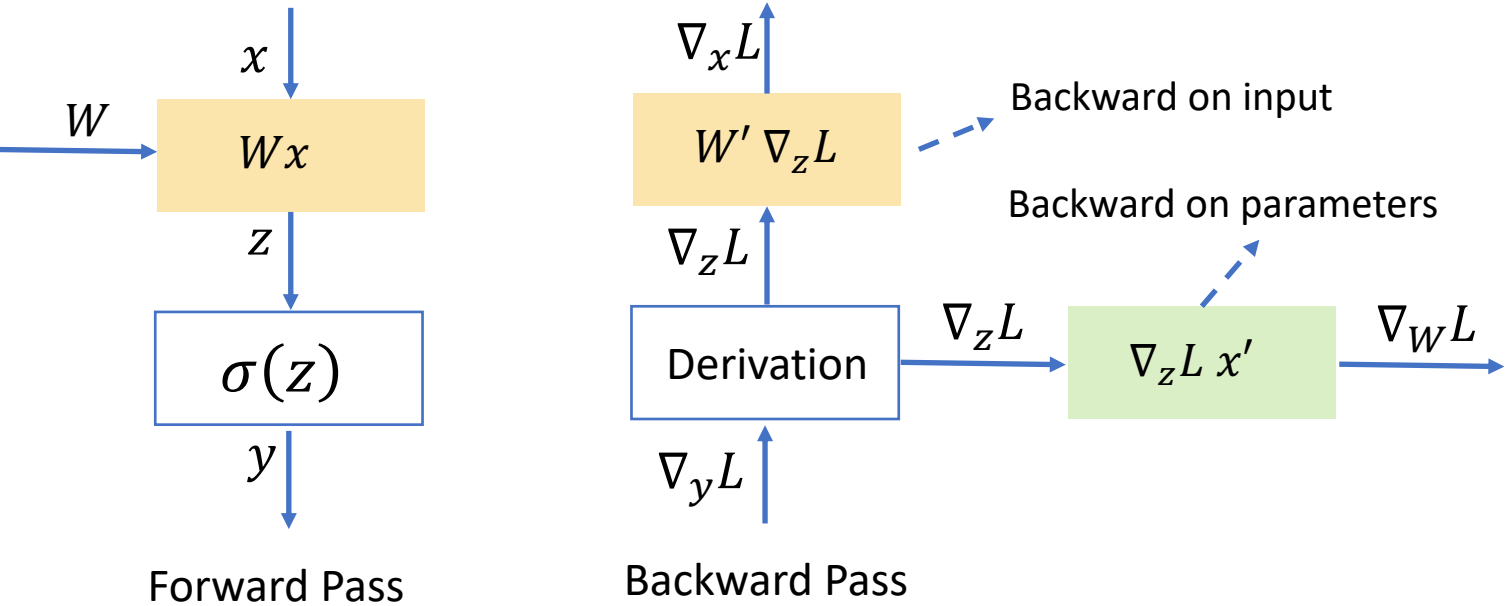
- Overlapping MoE All-to-All in DeepSeek v3
  - Why All-to-All inter-machine communication can hardly be hided?
    - Dependencies between compute and communication need to be decoupled!

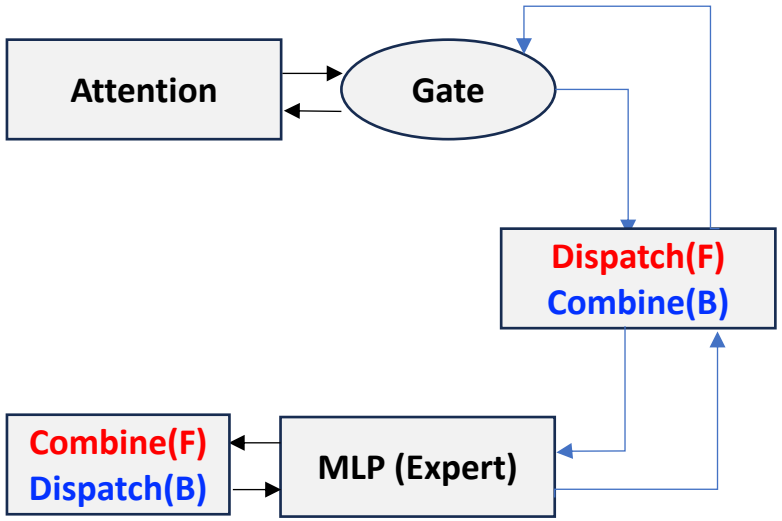




Mixture of Experts

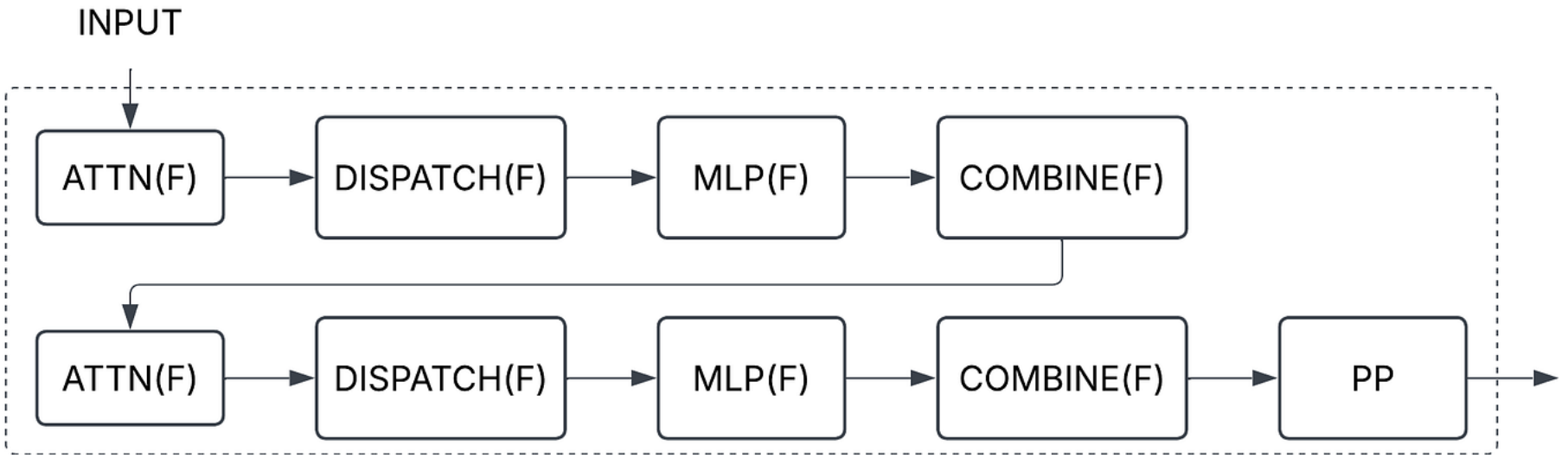
- Overlapping MoE All-to-All in DeepSeek v3
  - Decomposing computations into ***ATTN(F), MLP(F), ATTN(BI), ATTN(BW), MLP(BI), MLP(BW)***
  - Differentiating ***DISPATCH(F), DISPATCH(B), COMBINE(F), COMBINE(B)***
  - ***DISPATCH(B)*** after ***MLP(BI)***, ***COMBINE(B)*** after ***ATTN(BI)***





Mixture of Experts

- Overlapping MoE All-to-All in DeepSeek v3
  - Forward pass

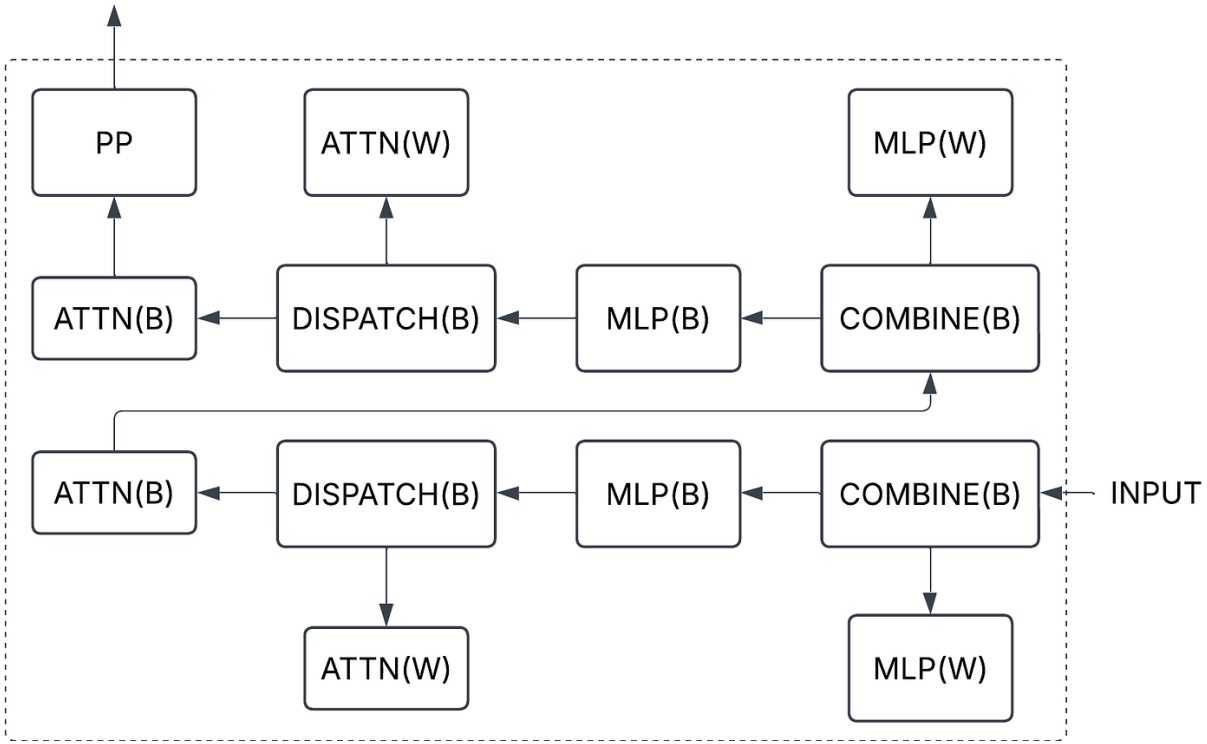


Decouple compute and communication as much as possible

https://medium.com/@awiteck/how-deepseeks-dualpipe-works-2f0764660de6

Mixture of Experts

- Overlapping MoE All-to-All in DeepSeek v3
  - Backward pass

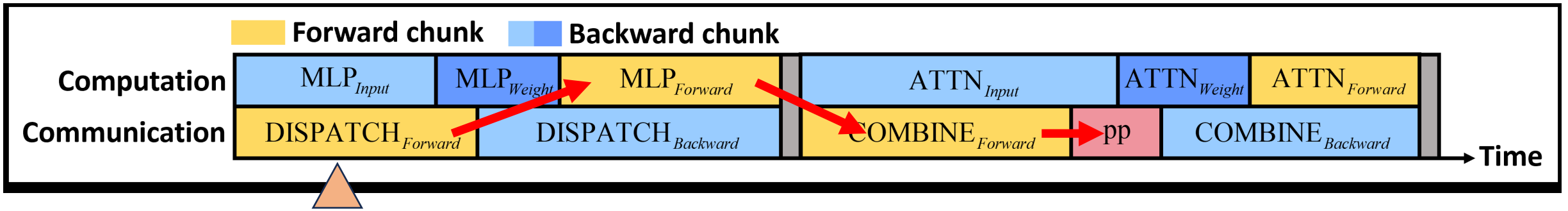


Decouple compute and communication as much as possible

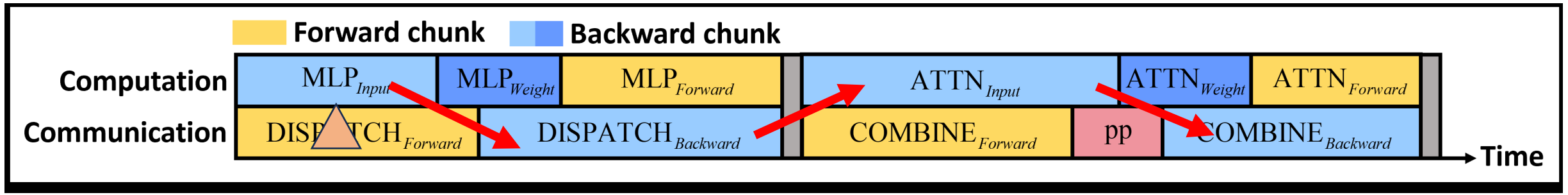
https://medium.com/@awiteck/how-deepseeks-dualpipe-works-2f0764660de6

Mixture of Experts

- Overlapping MoE All-to-All in DeepSeek v3
  - Forward pass



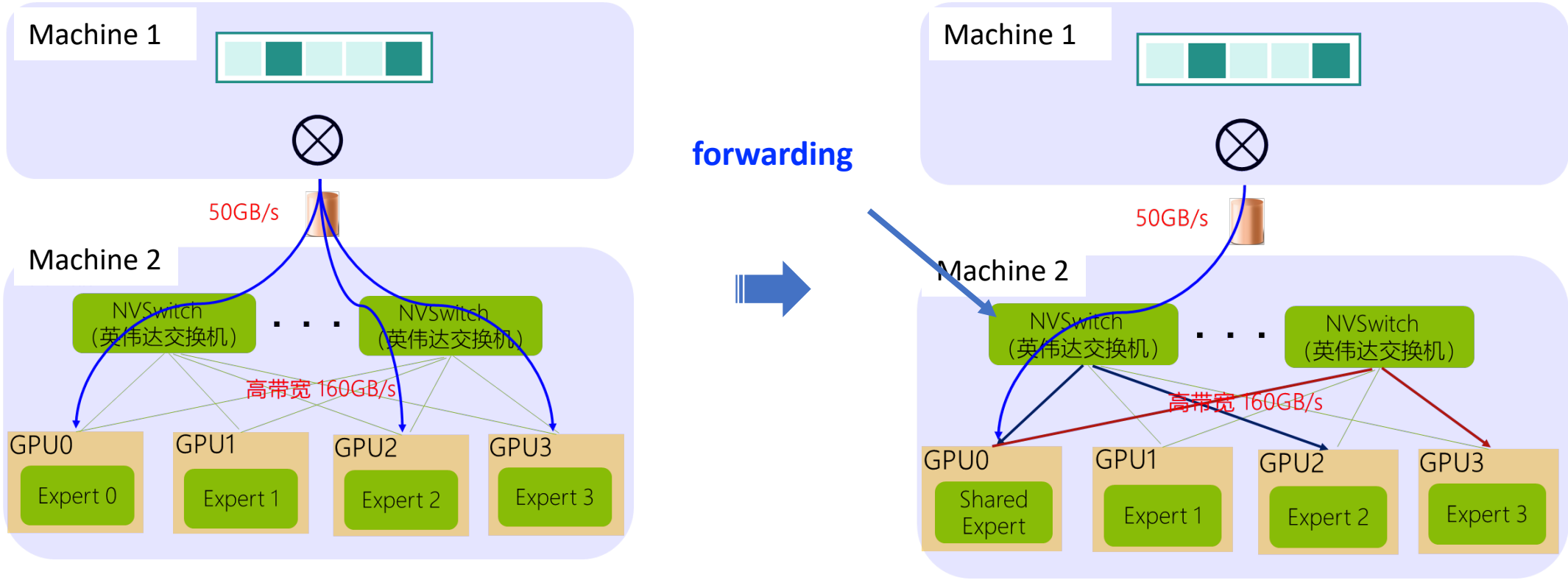
  - Backward pass



Source: DeepSeek-v3

Mixture of Experts

- Overlapping MoE All-to-All in DeepSeek v3
  - Token transfer: forwarding from RDMA to NVLink by use of asymmetric bandwidth

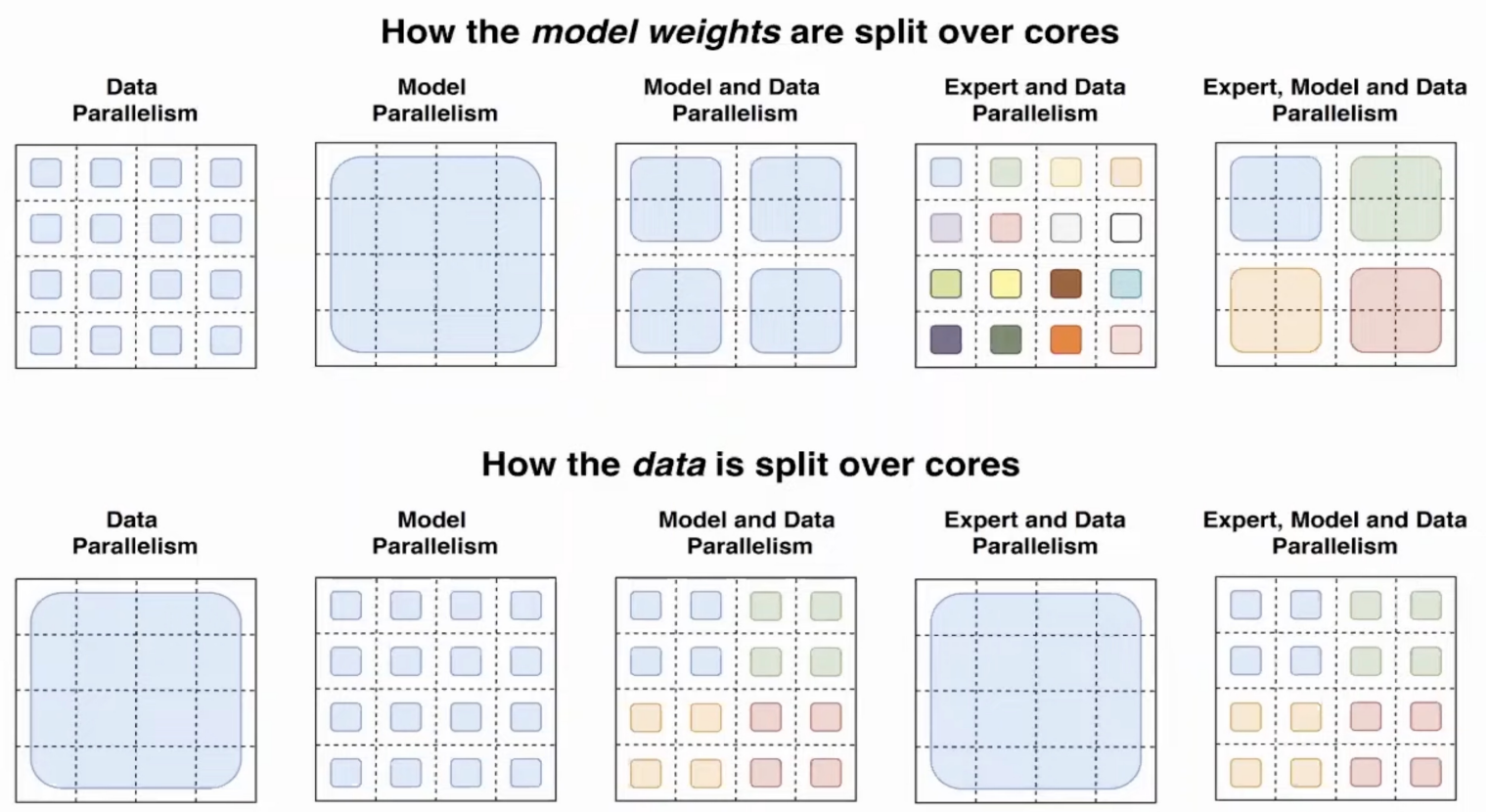


Mixture of Experts

- DeepEP (supplementary)
  - IBGDA versus IBRC
    - IBRC (InfiniBand Reliable Connection) requires a CPU to initiate operations over InfiniBand networks, usually for high throughput transmission
    - IBGDA (InfiniBand GPUDirect Async) enables direct GPU-initiated operations, bypassing the CPU to reduce latency, usually for low-latency small message transmission
  - Normal mode and low-latency mode
    - Forwarding
    - Buffer construction
    - …

Hybrid Parallelism

- Segmenting data, model and activation



Source: Transforming Natural Language Understanding: Attention, GPT, BERT, and Switch

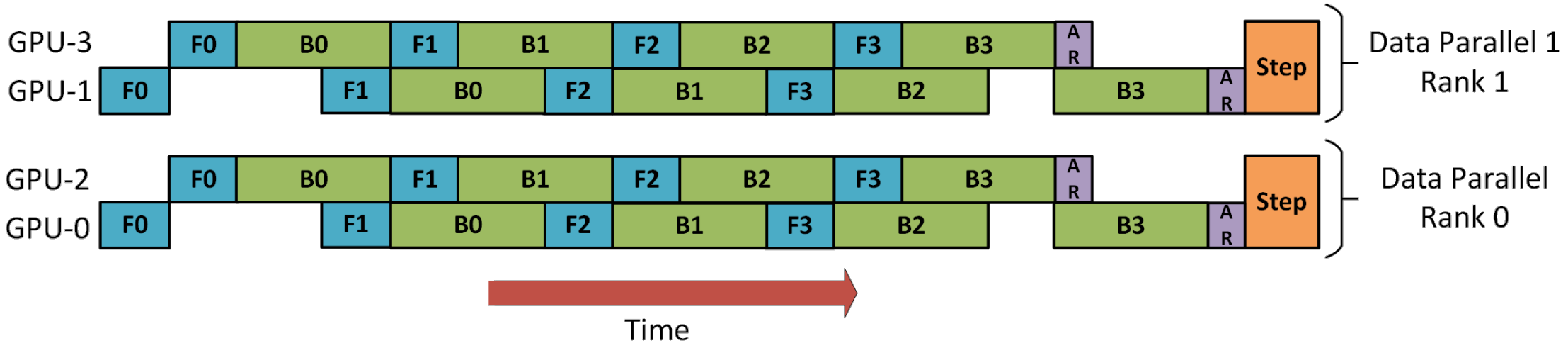
A Brief Summary

- Hybrid parallelism (classical models)

Models	**TP**	**DP**	**PP**	**Micro batchsize**	**global batchsize**	**sequence length**	**ZeRO**	GPUs	GPU types
Bloom-176B	4	8	12	2	2048	2048	ZeRO-1	384	A100-80GB
Megatron-175B	8	\	16	1	1536	2048	\	\	\
OPT-175B	8	124	\	\	\	2048	\	996	A100-80GB
GPT3-175B	\	\	\	\	\	2048	\	\	\
Megatron-NLG 530B	8	16	35	\	1960	2048	\	4480	A100-80GB
GLM-130B	4	24	8	\	\	4096	ZeRO-1	768	A100-80GB

A Brief Summary

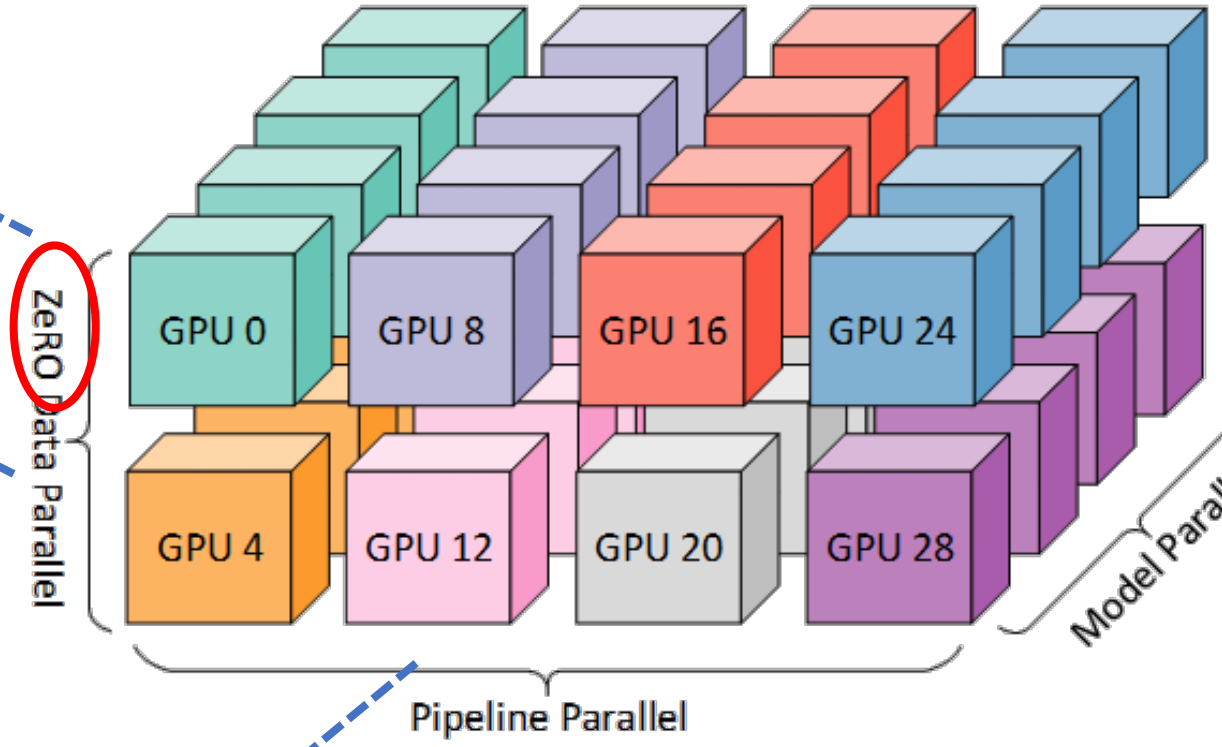
- Data Parallelism + Pipeline Parallelism



A Brief Summary

- Data Parallelism + Pipeline Parallelism + Tensor Parallelism + ZeRO

How to implement ZeRO-3 with pipeline?



Relatively lower bandwidth (e.g. RDMA)

High bandwidth domain (e.g. NVLink, HCCS)

Intra-datacenter RDMA or even Inter-datacenter links

Distributed LLM Training: Reliability

- Reliability

GPU

Network Switch

Smart NIC

Hardware Failures

Out-of-memory

PyTorch

Framework or CCL failures

Software Failures

- Meta AI encounters 110 failures when training OPT-175B using 992 NVIDIA A100 GPUs
- Meta utilized 16384 H100 GPUs to train Llama3-405B, and encountered 466 failures over a 54-day period
- BigScience trains BLOOM-176B incurs an average loss of 1.5 hours of training time on each hardware failure

Distributed LLM Training: Reliability

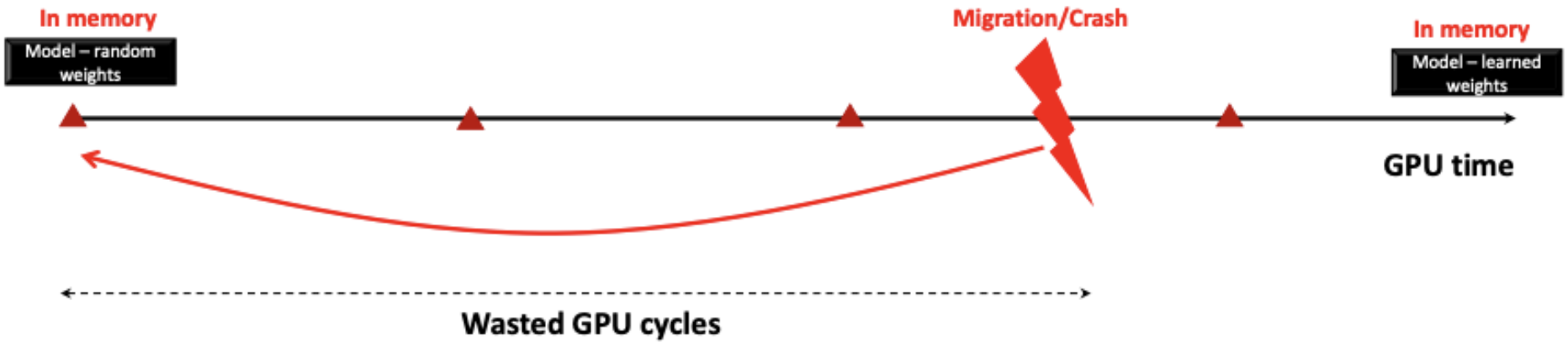
- Root-cause of training failures

Component	Category	Interruption Count	% of Interruptions
Faulty GPU	GPU	148	30.1%
GPU HBM3 Memory	GPU	72	17.2%
Software Bug	Dependency	54	12.9%
Network Switch/Cable	Network	35	8.4%
Host Maintenance	Unplanned Maintenance	32	7.6%
GPU SRAM Memory	GPU	19	4.5%
GPU System Processor	GPU	17	4.1%
NIC	Host	7	1.7%
NCCL Watchdog Timeouts	Unknown	7	1.7%
Silent Data Corruption	GPU	6	1.4%
GPU Thermal Interface + Sensor	GPU	6	1.4%
SSD	Host	3	0.7%
Power Supply	Host	3	0.7%
Server Chassis	Host	2	0.5%
IO Expansion Board	Host	2	0.5%
Dependency	Dependency	2	0.5%
CPU	Host	2	0.5%
System Memory	Host	2	0.5%

Table 5 Root-cause categorization of unexpected interruptions during a 54-day period of Llama 3 405B pre-training. About 78% of unexpected interruptions were attributed to confirmed or suspected hardware issues.

Distributed LLM Training: Reliability

- Checkpointing



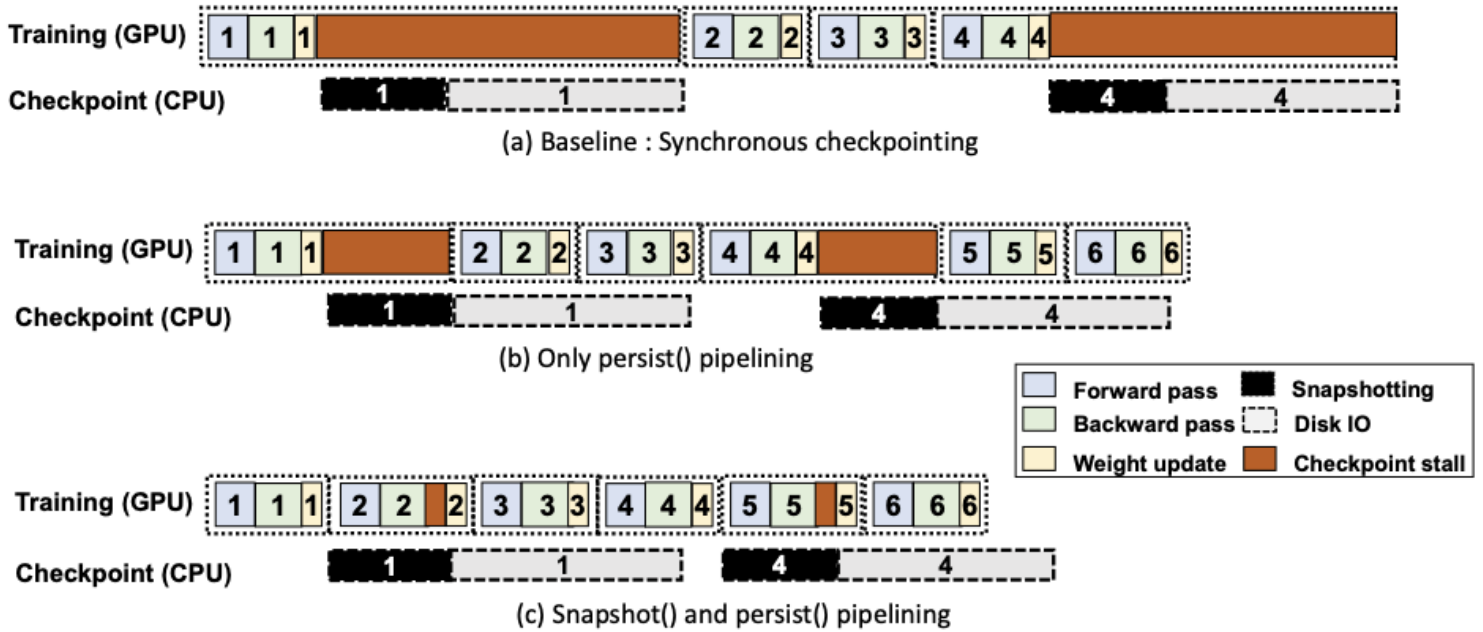
  - Frequent (e.g. synchronous) checkpointing causes training stalls
    - How frequent to checkpoint?
    - How to avoid contention with training traffic (e.g. data parallelism, expert parallelism, etc)?
    - How to resume correctly and quickly from a checkpoint?

Distributed LLM Training: Reliability

- Checkpointing: some confusions
  - A checkpoint includes
    - Model parameters
    - Momentum and variance in optimizer states
    - Hyperparameters such as learning rate and random seeds, etc.
  - Is checkpointing really necessary?
    - Data parallelism naturally duplicates model parameters and optimizer states!
  - ZeRO in data parallelism demands checkpoints for failure recovery
    - An optimizer shard is unique!

Distributed LLM Training: Reliability

- Checkpointing
  - Early trials: CheckFreq



  - Snapshot: Serialize and copy into a user-space buffer
  - Persist: Write out the serialized contents to disk

Source: CheckFreq: Frequent, Fine-Grained DNN Checkpointing

Distributed LLM Training: Reliability

- Checkpointing
  - Early trials: CheckFreq
    - To further reduce the checkpoint cost, CheckFreq snapshots on the GPU, and asynchronously writes it to CPU memory

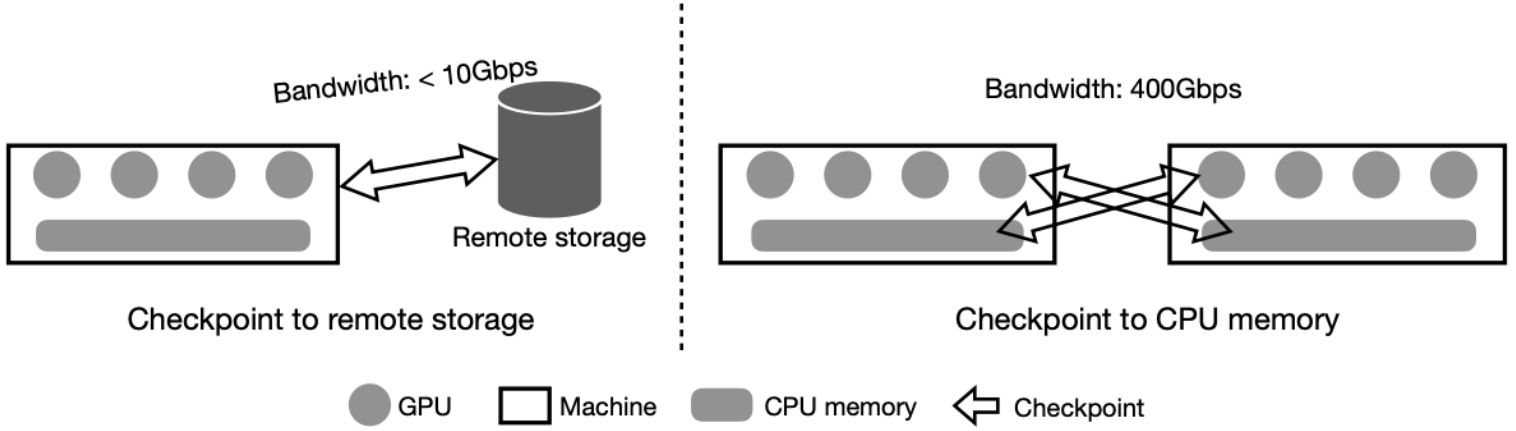
Model	Parameters	Checkpoint size	Checkpoint time (20Gbps)
Gopher [56]	280B	3.4 TB	23 min
MT-NLG [62]	530B	6.4 TB	43 min
PaLM [23]	540B	6.5 TB	44 min

Distributed LLM Training: Reliability

- Checkpointing
  - Early trials: GEMINI
    - As model grows large, checkpoint frequency is throttled by checkpointing time
    - CPU memory is much larger than GPU memory

Instance type	Cloud	GPU	GPU memory	CPU memory
p3dn.24xlarge [14]	AWS	8 V100	8 × 32 GB	768 GB
p4d.24xlarge [15]	AWS	8 A100	8 × 40 GB	1152 GB
ND40rs_v2 [10]	Azure	8 V100	8 × 32 GB	672 GB
ND96asr_v4 [11]	Azure	8 A100	8 × 40 GB	900 GB
n1-8-v100 [9]	GCP	8 V100	8 × 32 GB	624 GB
a2-highgpu-8g [9]	GCP	8 A100	8 × 40 GB	640 GB
DGX A100 [12]	NVIDIA	8 A100	8 × 80 GB	2 TB

CPU memory is an order of magnitude larger than GPU HBM

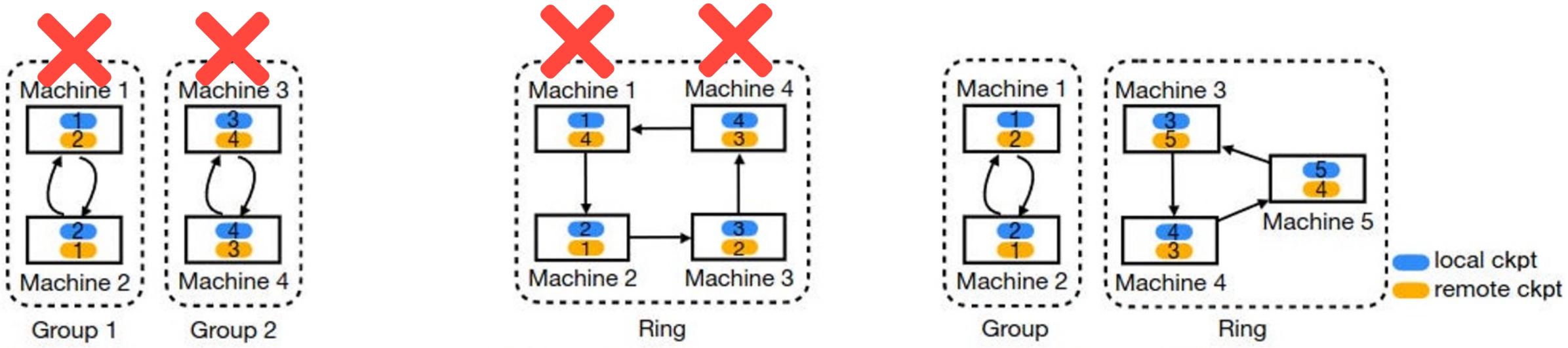


Storing checkpoints in remote CPU memory

Source: GEMINI: Fast Failure Recovery in Distributed Training with In-Memory Checkpoints

Distributed LLM Training: Reliability

- GEMINI Checkpointing
  - How many shards to checkpoint
  - How to distribute (group) different shards in the presence of multiple failures

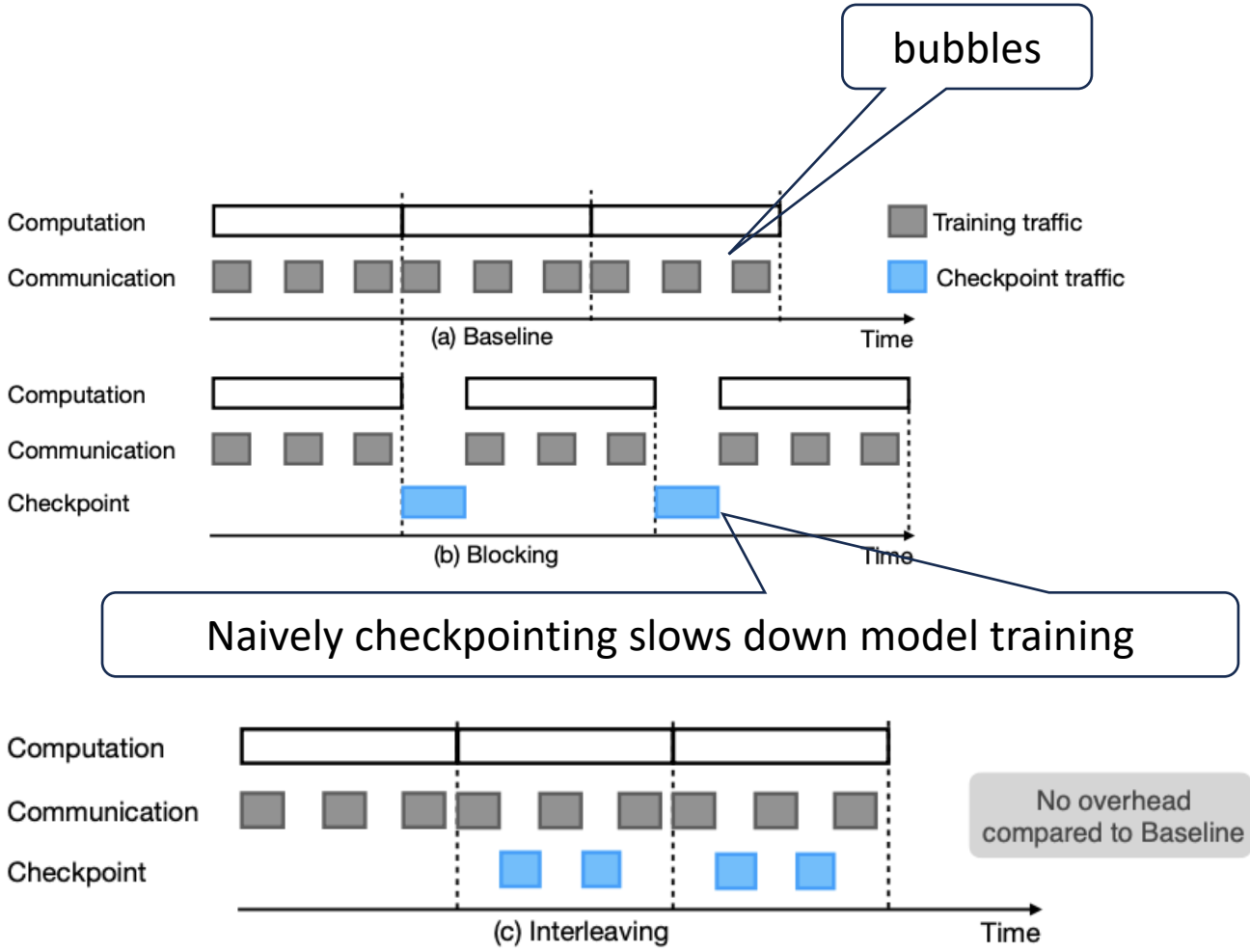


(a) Group placement strategy. (b) Ring placement strategy. (c) Mixed placement strategy.

Bad strategy

Distributed LLM Training: Reliability

- GEMINI Checkpointing
  - Interference with normal training traffic
  - Techniques
    - GPU-to-GPU communication (GDR: GPU Direct over RDMA)
    - Application-layer scheduling of chunked checkpoint transmission
    - Proper management of GPU memory for avoiding out-of-memory (OOM)



Idea checkpointing: inserting checkpoint transmission in the bubbles

Distributed LLM Training: Reliability

- Checkpointing
  - Recent advances
    - Further reducing checkpointing cost/improving checkpointing frequency
  - Selected Methods (sorry if my survey is incomplete)
    - Just-in-Time checkpointing
    - Bamboo
    - ByteCheckpoint
    - PhoenixOS
    - FlowCheck
    - Checkmate/Zoetic
    - PCcheck …

Thanks!