FasterMoE

Modeling and Optimizing Training of Large-Scale Dynamic Pre-Trained Models

Jiaao He Jidong Zhai[1] Tiago Antunes Haojie Wang Fuwen Luo Shangfeng Shi Qin Li

Tsinghua University

4, Apr, 2022 @ PPoPP'22

[1]Corresponding author

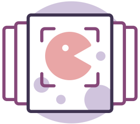
Pre-Trained Models

- The most popular DL model with **capability in multiple disciplines**.

Reading comprehension

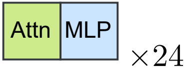
Conversation

Genome analysis



Computer vision

Trend of Pre-Trained Models: Giant Transformers



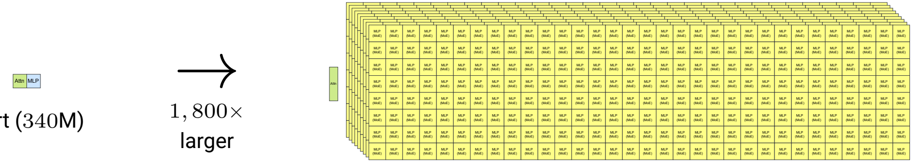
Transformer blocks in Bert[2] (340M Parameters)

- Massive computation for each input.
- Outstanding model capability.

[2] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." (2018).

[3] Lepikhin, Dmitry, et al. "Gshard: Scaling giant models with conditional computation and automatic sharding." (2020).

Trend of Pre-Trained Models: Giant Transformers



Bert (340M) $\longrightarrow$ $1,800\times$ larger GShard (600B)

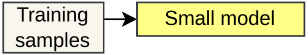
- GShard[3] is $1,800\times$ larger.
  - Significant improvement in model capability is observed.
- While GShard is only $1.5\times$ deeper, it adopts **Mixture-of-Experts (MoE)** structure, and employs $2,048$ experts in every MLP module.

---

[2] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." (2018).

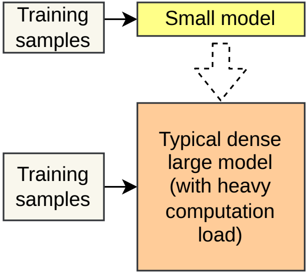
[3] Lepikhin, Dmitry, et al. "Gshard: Scaling giant models with conditional computation and automatic sharding." (2020).

MoE: A New Structure to Enlarge Models



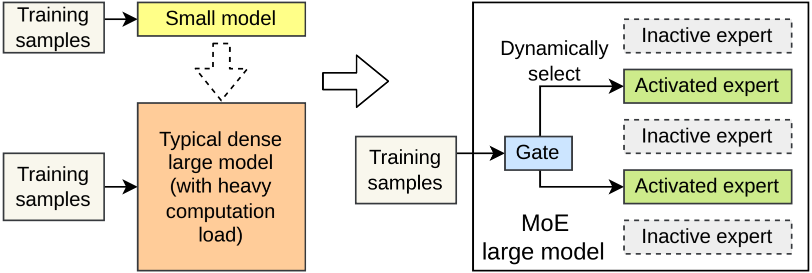
- Small models: limited capability

MoE: A New Structure to Enlarge Models



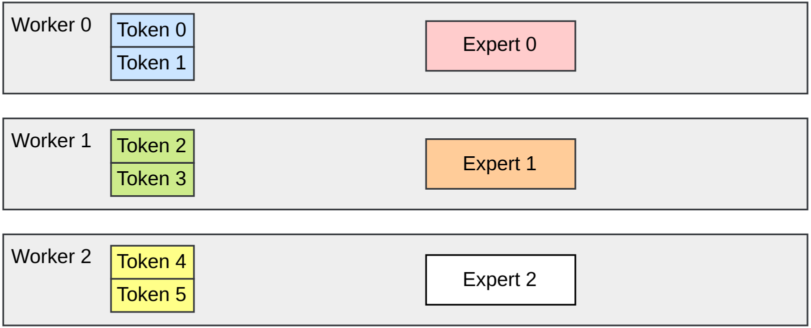
- Small models: limited capability; Dense large models: expensive computation.

MoE: A New Structure to Enlarge Models



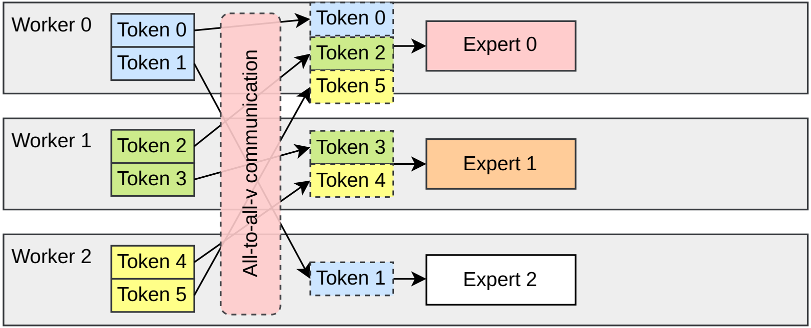
- Small models: limited capability; Dense large models: expensive computation.
- Mixture of **Experts**: Small models, selected by **Gate Module**.
  - The size of the model is enlarged, thus its capability is stronger.
  - The amount of computation remains small.

Expert Parallelism



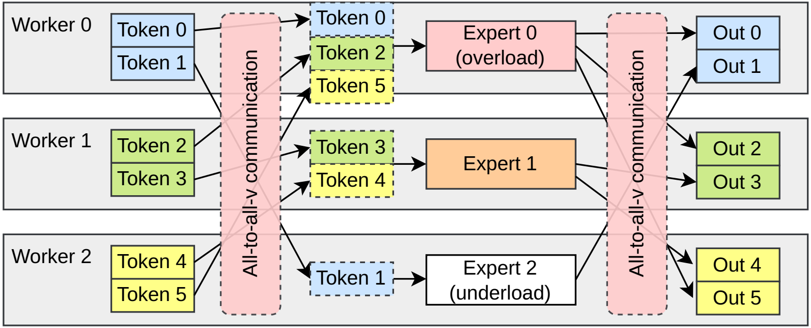
- Both experts and training data (tokens) are distributed across all workers.

Expert Parallelism



- Both experts and training data (tokens) are distributed across all workers.
- An `all-to-all` is performed to send tokens to their desired experts.

Expert Parallelism



- Both experts and training data (tokens) are distributed across all workers.
- An `all-to-all` is performed to send tokens to their desired experts.
- Another `all-to-all` sends the experts' answers back into original sequences.

Outline of the Paper

Challenge 1

Stragglers due to load imbalance

Challenge 2

Inefficient coarse-grained operators

Challenge 3

Network congestion

Outline of the Paper

Challenge 1
Stragglers due to load imbalance

→ **Optimization 1**
Expert shadowing

Challenge 2
Inefficient coarse-grained operators

→ **Optimization 2**
Smart fine-grained schedule

Challenge 3
Network congestion

→ **Optimization 3**
Topology-aware gate

Outline of the Paper

Challenge 1
Stragglers due to load imbalance

→ **Optimization 1**
Expert shadowing

Challenge 2
Inefficient coarse-grained operators

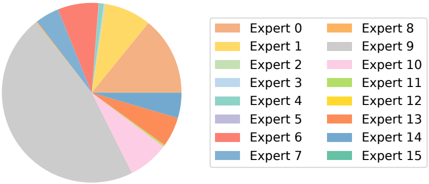
→ **Optimization 2**
Smart fine-grained schedule

Challenge 3
Network congestion

→ **Optimization 3**
Topology-aware gate

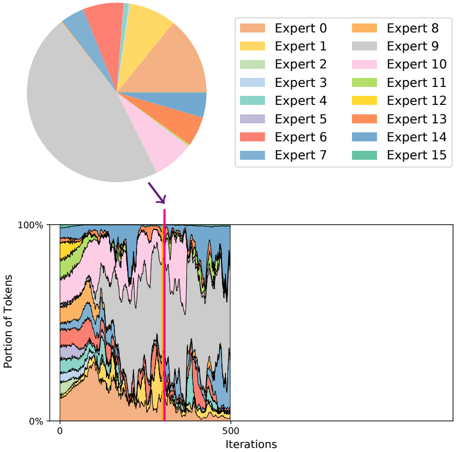
Performance Modeling
DDL-Roofline: Characterization of distributed DL workloads.

Challenge 1: Imbalanced Assignment



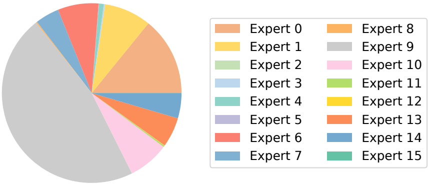
- Expert selection can be severely imbalanced.

Challenge 1: Imbalanced Assignment

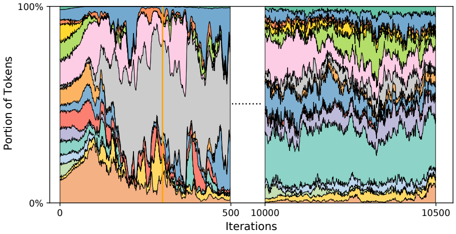


- Expert selection can be severely imbalanced.
- The distribution changes between iterations.

Challenge 1: Imbalanced Assignment

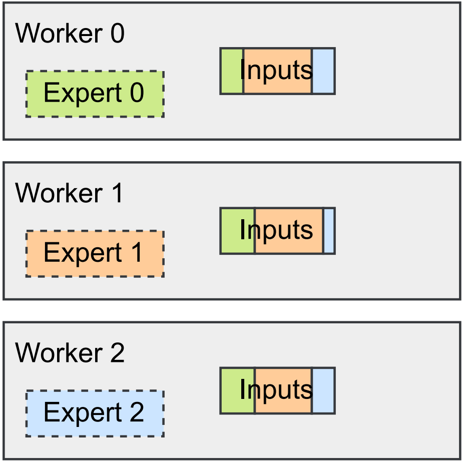


- Expert selection can be severely imbalanced.



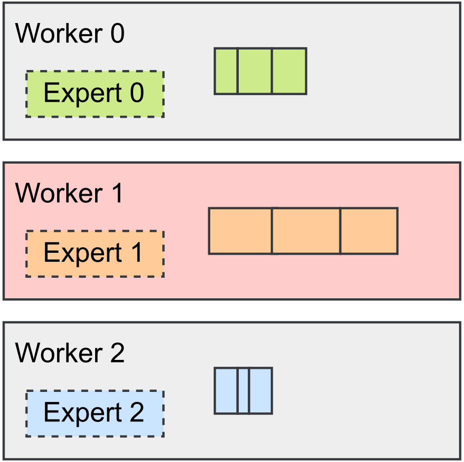
- The distribution changes between iterations.
- The skew varies a lot throughout the training process.

Expert Shadowing



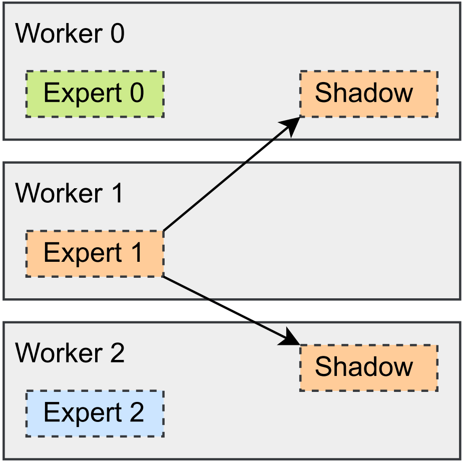
- Expert 1 is very popular.

Expert Shadowing



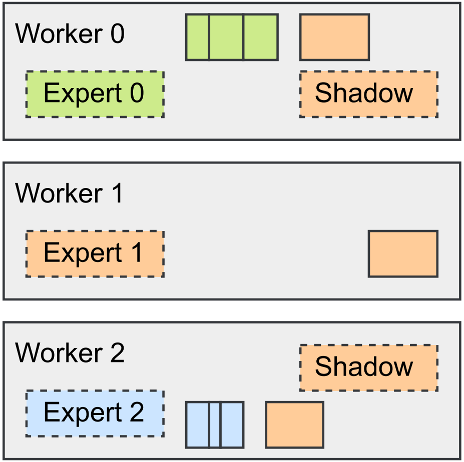
- Expert 1 is very popular.
- Worker 1 is a **straggler**.

Expert Shadowing



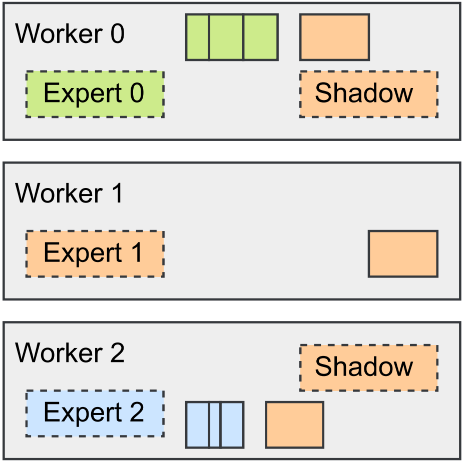
- **Shadow** expert 1 by broadcasting its parameters.

Expert Shadowing



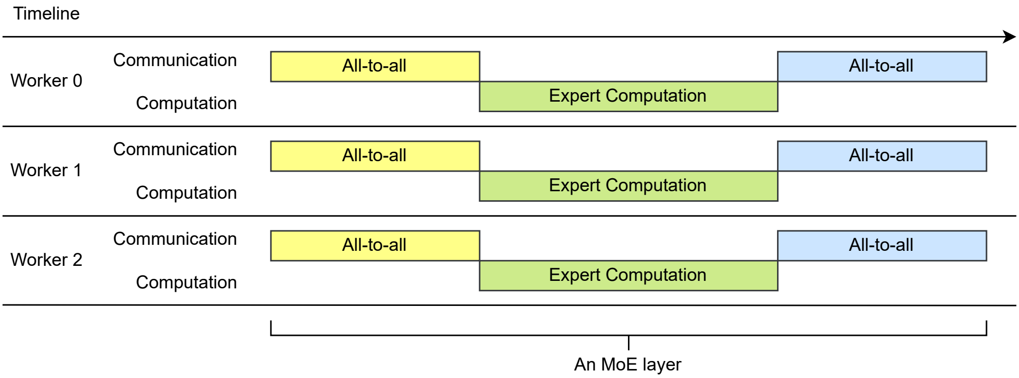
- **Shadow** expert 1 by broadcasting its parameters.
- It becomes more balanced.

Expert Shadowing



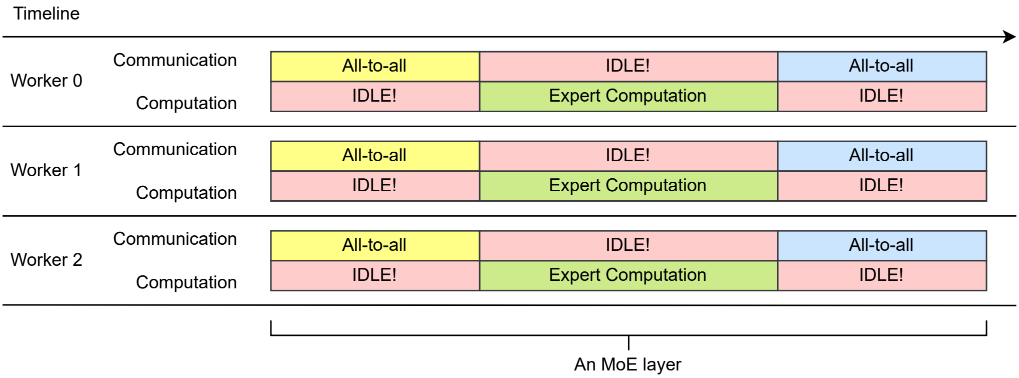
- We select shadow experts before every MoE layer, guided by a **performance predictor**. (detailed in the paper)

Challenge 2: Inefficient Coarse-grained Operators



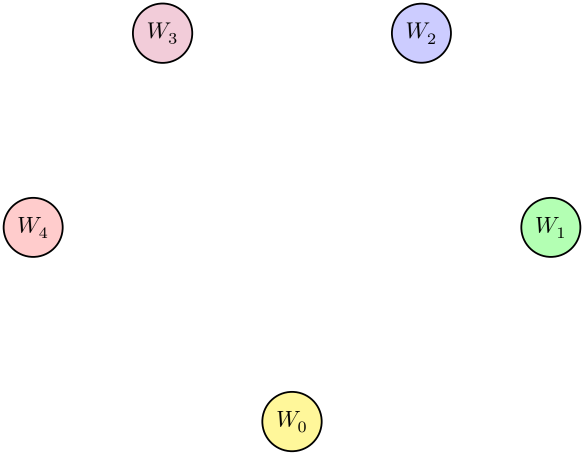
- Each MoE layer involves computation between 2 `all-to-all`s.

Challenge 2: Inefficient Coarse-grained Operators



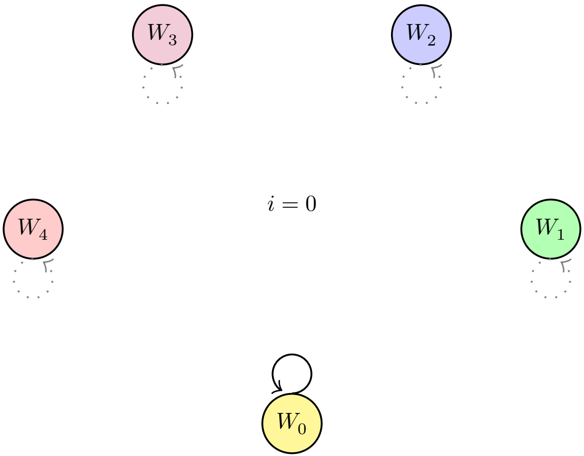
- There is always some hardware idling.

Pair-wise Exchange Algorithm for All-to-all



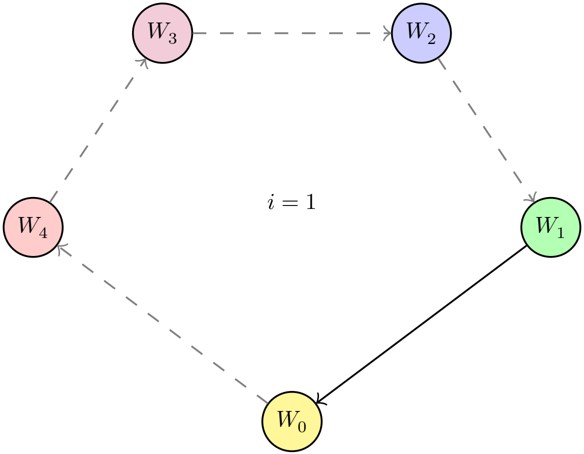
- n steps for n workers.
- At time step i, W_j:
  - Sends to $W_{(j-i) \bmod n}$
  - Receives from $W_{(j+i) \bmod n}$

Pair-wise Exchange Algorithm for All-to-all



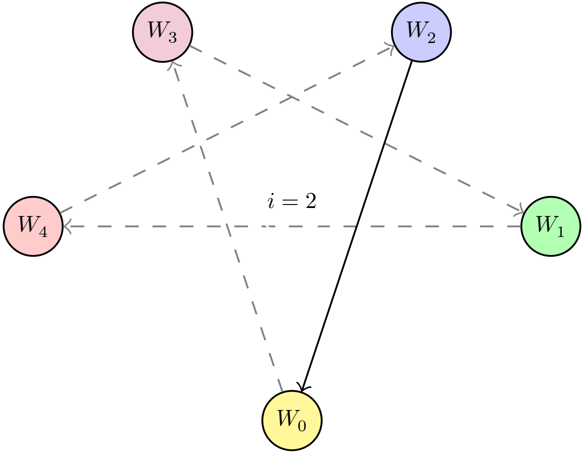
- n steps for n workers.
- At time step i, W_j:
  - Sends to $W_{(j-i) \bmod n}$
  - Receives from $W_{(j+i) \bmod n}$
- Example: 5 workers
  - For W_0:
    - Receives from W_0

Pair-wise Exchange Algorithm for All-to-all



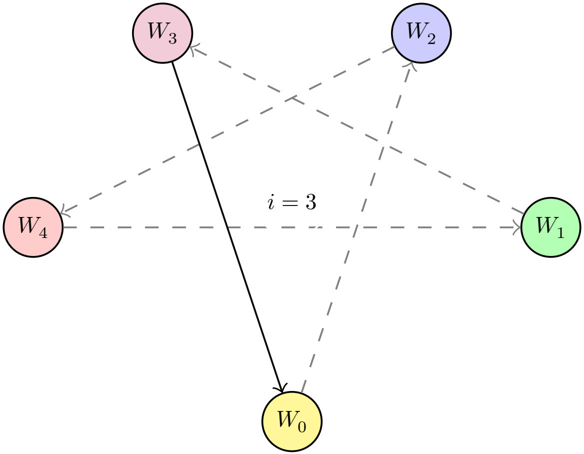
- n steps for n workers.
- At time step i, W_j:
  - Sends to $W_{(j-i) \bmod n}$
  - Receives from $W_{(j+i) \bmod n}$
- Example: 5 workers
  - For W_0:
    - Receives from W_0
    - Receives from W_1

Pair-wise Exchange Algorithm for All-to-all



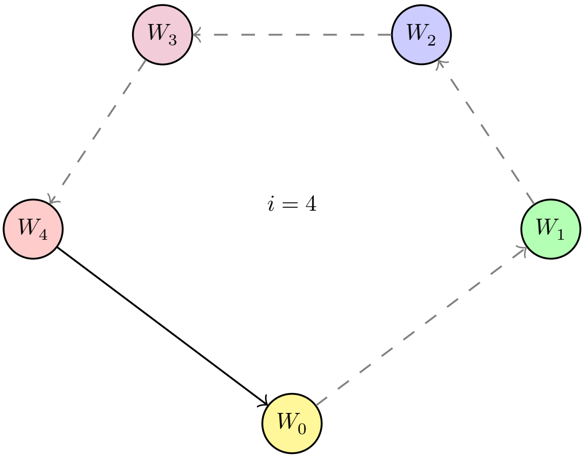
- n steps for n workers.
- At time step i, W_j:
  - Sends to $W_{(j-i) \bmod n}$
  - Receives from $W_{(j+i) \bmod n}$
- Example: 5 workers
  - For W_0:
    - Receives from W_0
    - Receives from W_1
    - Receives from W_2

Pair-wise Exchange Algorithm for All-to-all



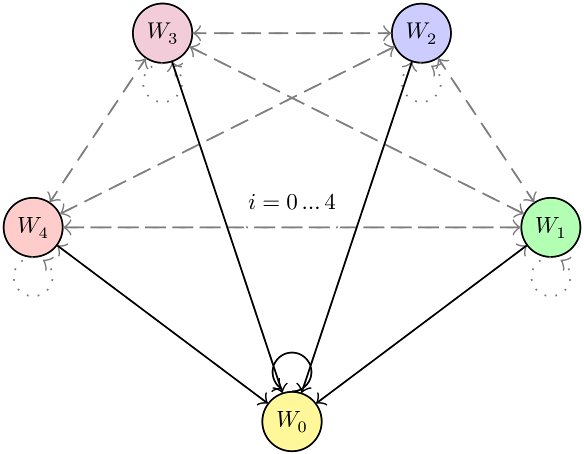
- n steps for n workers.
- At time step i, W_j:
  - Sends to $W_{(j-i) \bmod n}$
  - Receives from $W_{(j+i) \bmod n}$
- Example: 5 workers
  - For W_0:
    - Receives from W_0
    - Receives from W_1
    - Receives from W_2
    - Receives from W_3

Pair-wise Exchange Algorithm for All-to-all



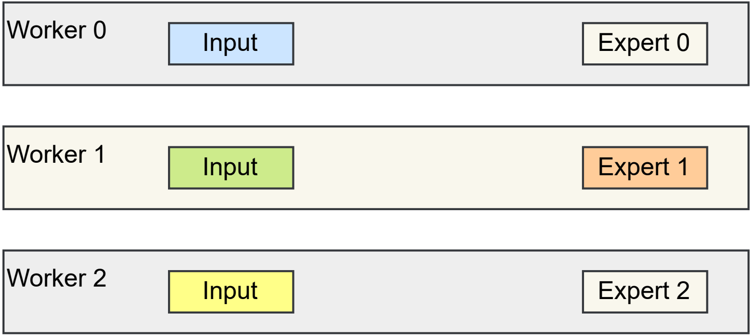
- n steps for n workers.
- At time step i, W_j:
  - Sends to $W_{(j-i) \bmod n}$
  - Receives from $W_{(j+i) \bmod n}$
- Example: 5 workers
  - For W_0:
    - Receives from W_0
    - Receives from W_1
    - Receives from W_2
    - Receives from W_3
    - Receives from W_4

Pair-wise Exchange Algorithm for All-to-all



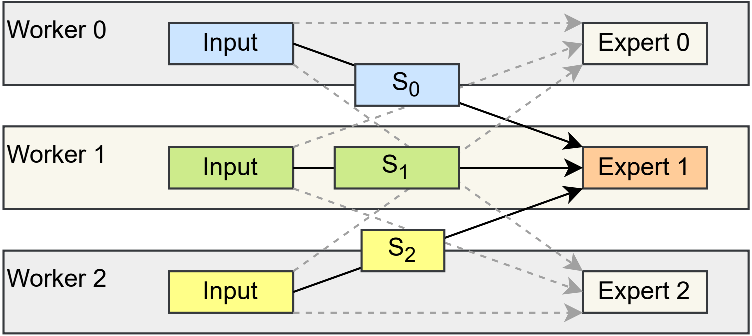
- n steps for n workers.
- At time step i, W_j:
  - Sends to $W_{(j-i) \bmod n}$
  - Receives from $W_{(j+i) \bmod n}$
- Example: 5 workers
  - For W_0:
    - Receives from W_0
    - Receives from W_1
    - Receives from W_2
    - Receives from W_3
    - Receives from W_4
    - All data received.

Smart Scheduling: Fine-grained Task Split-up



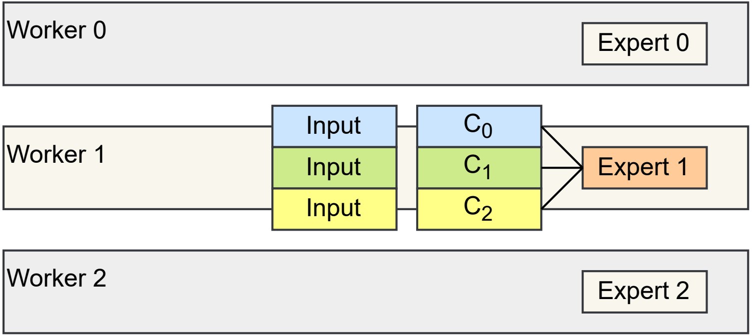
- Take worker 1 for example, expert 1 on it has to process input on worker $0, 1, 2$.

Smart Scheduling: Fine-grained Task Split-up



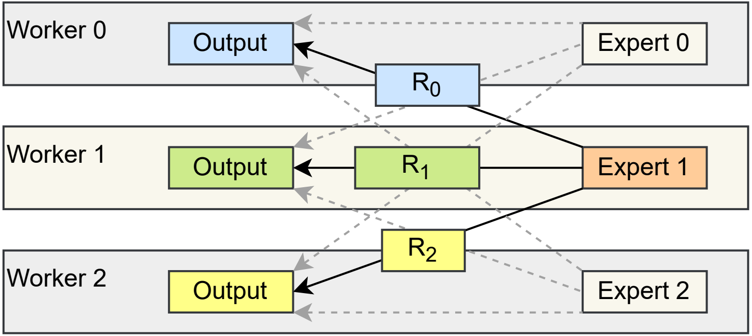
- Take worker 1 for example, expert 1 on it has to process input on worker $0, 1, 2$.
  - S_i: Worker i sends input to worker 1 (as a part of the first `all-to-all`).

Smart Scheduling: Fine-grained Task Split-up



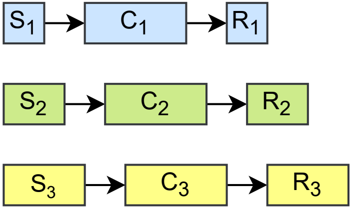
- Take worker 1 for example, expert 1 on it has to process input on worker $0, 1, 2$.
  - S_i: Worker i sends input to worker 1 (as a part of the first `all-to-all`).
  - C_i: The input from worker i is processed by expert 1 on worker 1.

Smart Scheduling: Fine-grained Task Split-up



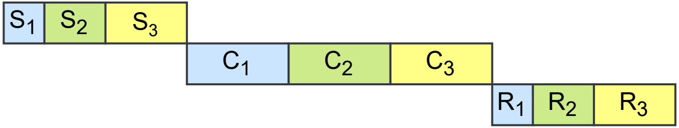
- Take worker 1 for example, expert 1 on it has to process input on worker $0, 1, 2$.
  - S_i: Worker i sends input to worker 1 (as a part of the first `all-to-all`).
  - C_i: The input from worker i is processed by expert 1 on worker 1.
  - R_i: Worker i retrieves output from worker 1 (as a part of the second `all-to-all`).

Smart Scheduling: Re-ordering Fine-grained Operations



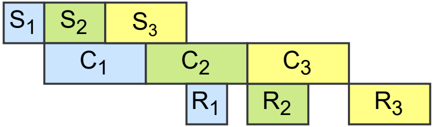
- Dependencies between S, C, R.

Smart Scheduling: Re-ordering Fine-grained Operations



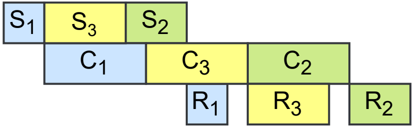
- Follow *Pair-wise Exchange algorithm* to organize Ss and Rs.
- Baseline: execute sequentially.

Smart Scheduling: Re-ordering Fine-grained Operations



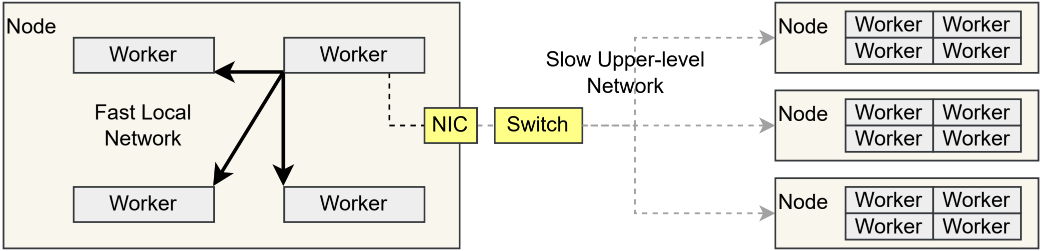
- Lower the latency: Perform C and R **as soon as possible**.

Smart Scheduling: Re-ordering Fine-grained Operations



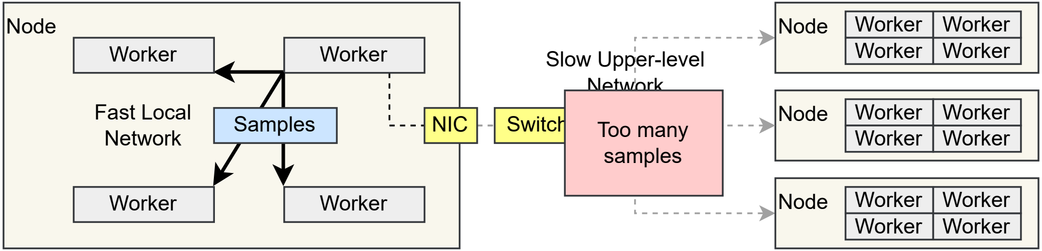
- To maximize efficiency: (detailed in the paper)
  - Use a **group of workers**, instead of a single worker, as the granularity.
  - Minimize first S and last R by grouping heuristics.

Challenge 3: Congested Cross-node Connection



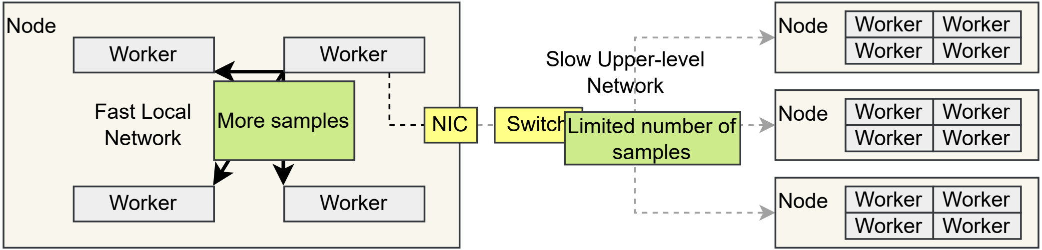
- Commonly, workers are in a tree-like topology, with lower upper-level bandwidth.

Challenge 3: Congested Cross-node Connection



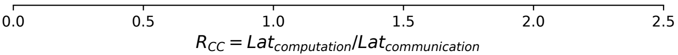
- Commonly, workers are in a tree-like topology, with lower upper-level bandwidth.
- With unconstrained expert selection, there is **congestion** in the slow connections.

Topology-aware Gates



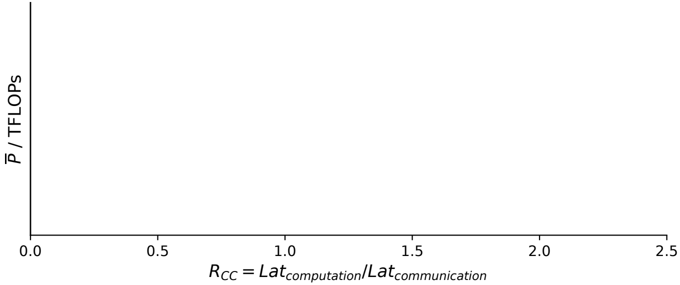
- We design a **Topology-aware gate**.
  - Limit the number of tokens transmitted through upper-level links.
  - Redirect overfilling tokens to experts in the local node (for better data utilization).
  - **Specific gates shall be designed for different cases**.

DDL-Roofline: Roofline for Distributed Deep Learning



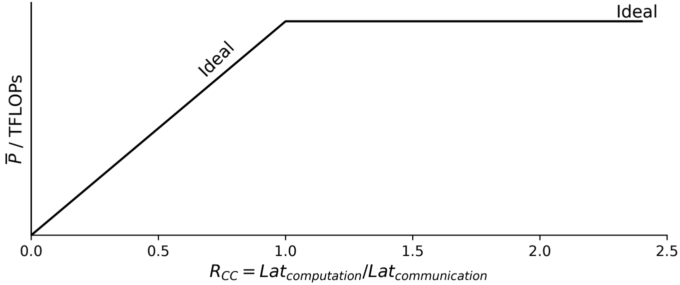
- X-axis: $R_{CC} = \frac{Lat_{\text{computation}}}{Lat_{\text{communication}}}$

DDL-Roofline: Roofline for Distributed Deep Learning



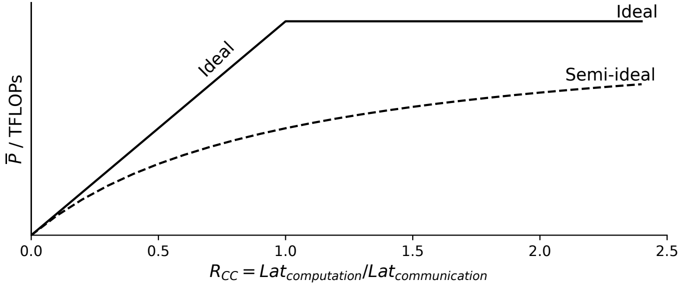
- X-axis: $R_{CC} = \frac{Lat_{\text{computation}}}{Lat_{\text{communication}}}$
- Y-axis: $\bar{P}$, **Per-accelerator** computation throughput.

DDL-Roofline: Roofline for Distributed Deep Learning

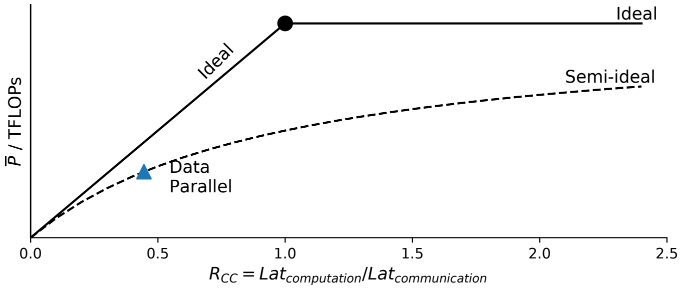


- X-axis: $R_{CC} = \frac{Lat_{\text{computation}}}{Lat_{\text{communication}}}$
- Y-axis: $\bar{P}$, **Per-accelerator** computation throughput.
- Reference lines:
  - Ideal: Computation and communication are **perfectly overlapped**.

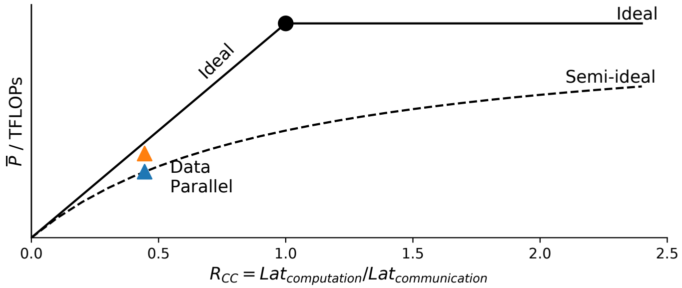
DDL-Roofline: Roofline for Distributed Deep Learning



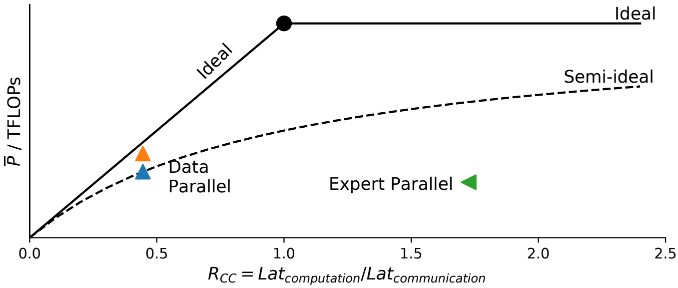
- X-axis: $R_{CC} = \frac{Lat_{\text{computation}}}{Lat_{\text{communication}}}$
- Y-axis: $\bar{P}$, **Per-accelerator** computation throughput.
- Reference lines:
  - Ideal: Computation and communication are **perfectly overlapped**.
  - Semi-ideal: They are performed **sequentially**.



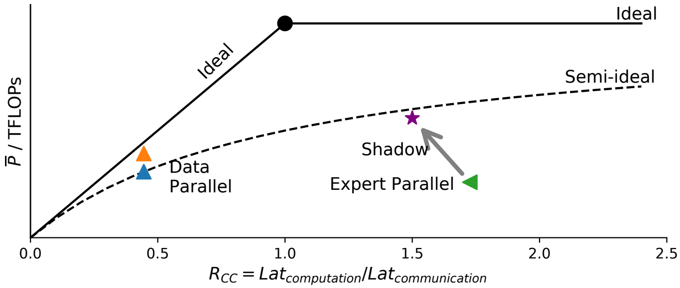
- Data parallelism: too much **communication overhead** of gradients for MoE.



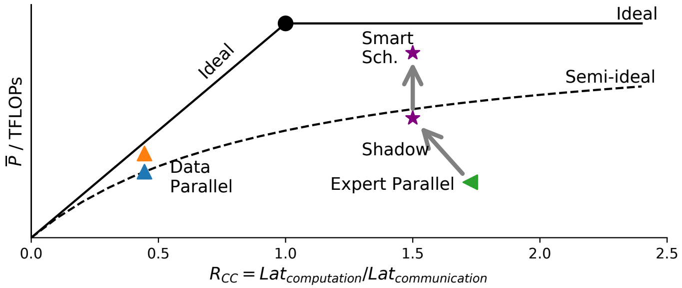
- Data parallelism: too much **communication overhead** of gradients for MoE.
  - Leap beyond semi-ideal by *overlapping*.



- Data parallelism: too much **communication overhead** of gradients for MoE.
  - Leap beyond semi-ideal by *overlapping*.
- Expert parallelism: less communication, but **load imbalance**.

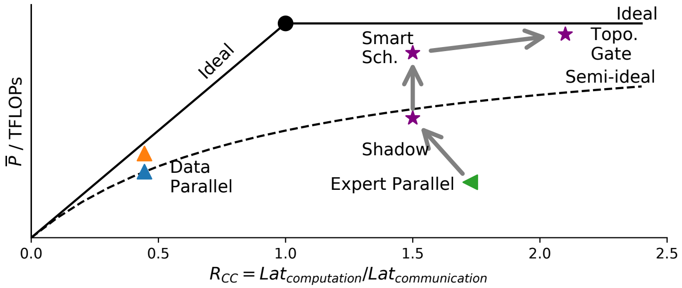


- Data parallelism: too much **communication overhead** of gradients for MoE.
  - Leap beyond semi-ideal by *overlapping*.
- Expert parallelism: less communication, but **load imbalance**.
- Our optimizations:
  - **Shadow** relieves load imbalance.



- Data parallelism: too much **communication overhead** of gradients for MoE.
  - Leap beyond semi-ideal by *overlapping*.
- Expert parallelism: less communication, but **load imbalance**.
- Our optimizations:
  - **Shadow** relieves load imbalance.
  - **Smart schedule** overlaps communication with computation.

PACMAN
pacman.cs.tsinghua.edu.cn



- Data parallelism: too much **communication overhead** of gradients for MoE.
  - Leap beyond semi-ideal by *overlapping*.
- Expert parallelism: less communication, but **load imbalance**.
- Our optimizations:
  - **Shadow** relieves load imbalance.
  - **Smart schedule** overlaps communication with computation.
  - **Topology-aware gate** reduces communication volume.

Implementation Details

- Based on *FastMoE*[4], a PyTorch-based MoE training framework.
  - With modified operators and customized gates.
- Using Megatron-LM[5] as transformer codebase.
- Open source at https://github.com/thu-pacman/fastermoe

[4] He, Jiaao, et al. "Fastmoe: A fast mixture-of-expert training system." (2021).
[5] Shoeybi, Mohammad, et al. "Megatron-lm: Training multi-billion parameter language models using model parallelism." (2019).

Baseline Systems for Comparison

Systems that support faithfully selecting top-k experts for each token

- *FastMoE*: Baseline of expert parallelism.

Baseline Systems for Comparison

Systems that support faithfully selecting top-k experts for each token

- *FastMoE*: Baseline of expert parallelism.
- *DeepSpeed + ZeRO*: Baseline for data parallelism. (*by Microsoft*)
  - MoE model implemented by single-GPU version of *FastMoE*.
  - Use ZeRO Stage 3 to fit the model into GPU memory.

Baseline Systems for Comparison

Systems that support faithfully selecting top-k experts for each token

- *FastMoE*: Baseline of expert parallelism.
- *DeepSpeed + ZeRO*: Baseline for data parallelism. (*by Microsoft*)
  - MoE model implemented by single-GPU version of *FastMoE*.
  - Use ZeRO Stage 3 to fit the model into GPU memory.

Systems that require specific expert selection method

- *GShard*: A state-of-the-art MoE system. (*by Google*)
  - Limit capacity of each expert, and drop the overfilling input.
  - Ported to PyTorch + NVIDIA with customized gate in *FastMoE*.

Baseline Systems for Comparison

Systems that support faithfully selecting top-k experts for each token

- *FastMoE*: Baseline of expert parallelism.
- *DeepSpeed + ZeRO*: Baseline for data parallelism. (*by Microsoft*)
  - MoE model implemented by single-GPU version of *FastMoE*.
  - Use ZeRO Stage 3 to fit the model into GPU memory.

Systems that require specific expert selection method

- *GShard*: A state-of-the-art MoE system. (*by Google*)
  - Limit capacity of each expert, and drop the overfilling input.
  - Ported to PyTorch + NVIDIA with customized gate in *FastMoE*.
- *FairSeq + BASE Layers*: Another state-of-the-art MoE system. (*by Facebook*)
  - Run a matching algorithm over the input and expert score sheet for perfectly balanced assignment.

Evaluation Setup

Hardware

- *johnny* cluster: 16 `NVIDIA V100 PCIe` GPUs in 2 nodes.
- *trevor* cluster: 64 `NVIDIA V100 SXM2` GPUs in 16 nodes.

Evaluation Setup

Hardware

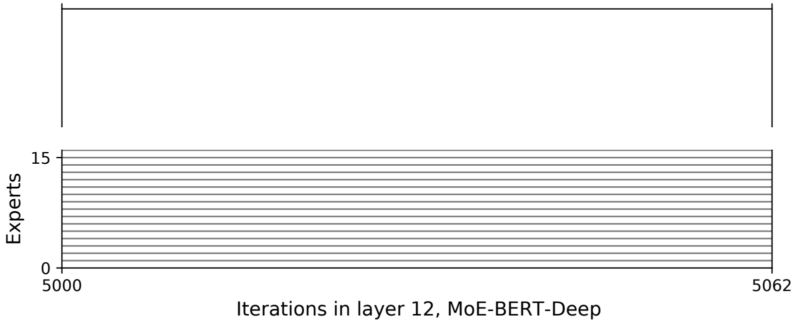
- *johnny* cluster: 16 `NVIDIA V100 PCIe` GPUs in 2 nodes.
- *trevor* cluster: 64 `NVIDIA V100 SXM2` GPUs in 16 nodes.

Models

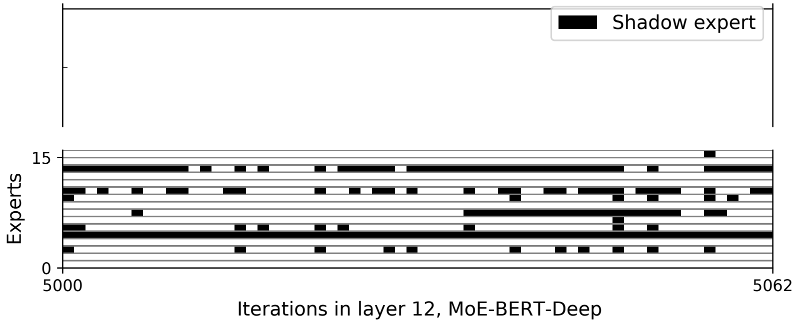
- 4 MoE-Bert and 3 MoE-GPT models at different sizes.

Model name	Size	Layers	Experts	H	α	Cluster
MoE-GPT-S	0.86B	12	16	1024	2	johnny
MoE-GPT	3.42B			2048		
MoE-GPT-L	13.7B			4096		
MoE-BERT-Deep	1.71B	24	16	1024	2	johnny
MoE-BERT-Deep-L	27.4B			4096		
MoE-BERT-Wide	3.27B	12	64	1024	2	trevor
MoE-BERT-Wide-L	13.1B			2048		

Case Study of Expert Shadowing

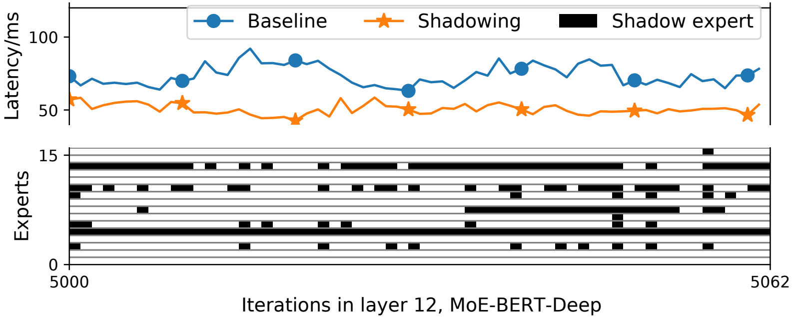


Case Study of Expert Shadowing



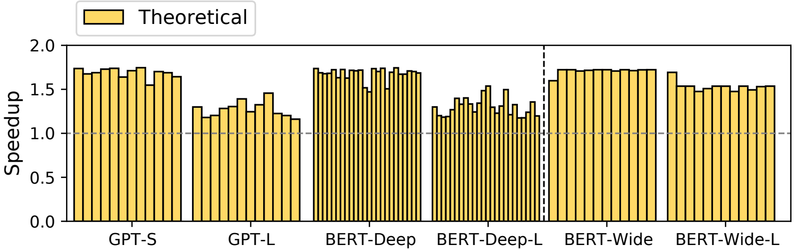
- Different experts are shadowed dynamically, according to different input.
  - On average, 19% experts are shadowed.

Case Study of Expert Shadowing



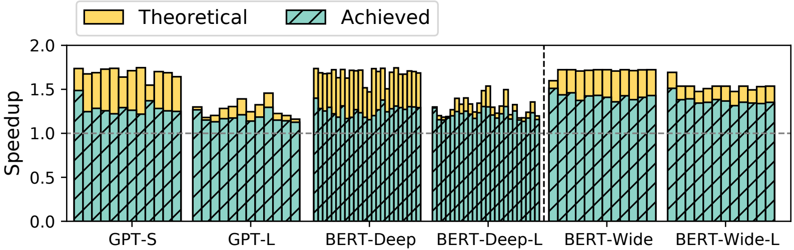
- Different experts are shadowed dynamically, according to different input.
  - On average, 19% experts are shadowed.
- End-to-end latency is effectively reduced.
  - Up to $1.97\times$ speedup in one iteration.

Effectiveness of Smart Scheduling



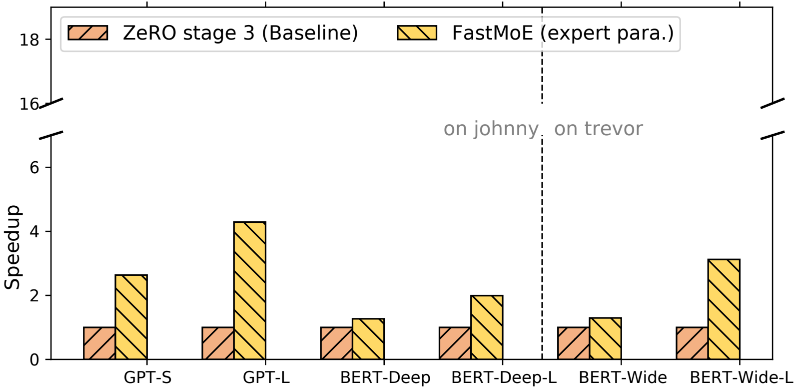
- Theoretical upper bound is computed by the performance predictor.
  - Up to $1.71\times$

Effectiveness of Smart Scheduling

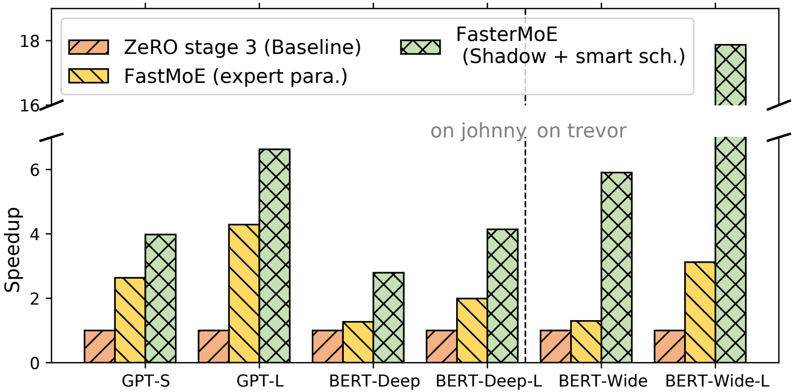


- Theoretical upper bound is computed by the performance predictor.
  - Up to $1.71\times$
- The schedule provides significant speedup.
  - Up to $1.42\times$ speedup achieved.
  - The effectiveness varies between models.

Speedup over FastMoE and ZeRO

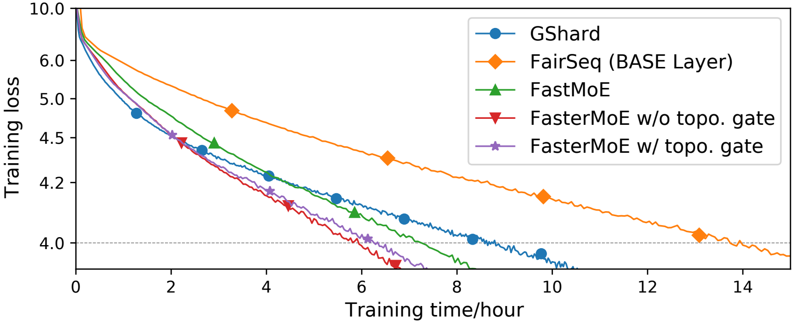


Speedup over FastMoE and ZeRO



- More than $17\times$ speedup over *DeepSpeed + ZeRO* baseline.
- More than $5\times$ speedup over *FastMoE* baseline.

Convergence Speed w.r.t. GShard and BASE Layers



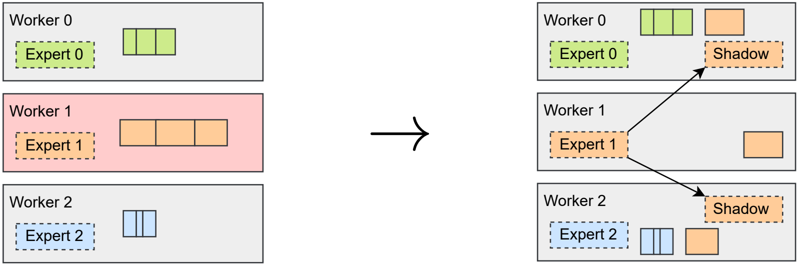
- Both baseline systems take significantly more steps to converge.
- Convergence speed is $1.37\times$ shorter than *GShard*, and $2.19\times$ shorter than *BASE Layers*.
- The topology-aware gate makes iterations 9.4% faster.

FasterMoE in a Nutshell

- MoE models are **large** and **dynamic**.

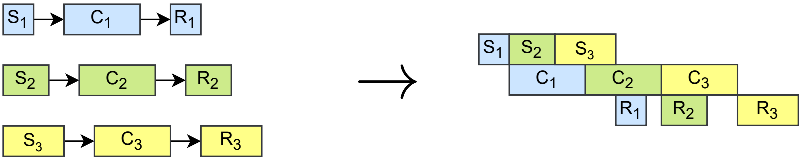
FasterMoE in a Nutshell

- MoE models are **large** and **dynamic**.
- To train the trending pre-trained models more efficiently,
  - **Shadow Experts** to relieve load imbalance;



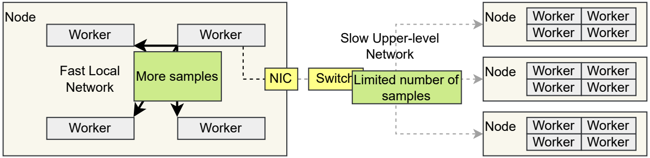
FasterMoE in a Nutshell

- MoE models are **large** and **dynamic**.
- To train the trending pre-trained models more efficiently,
  - **Shadow Experts** to relieve load imbalance;
  - Use **Smart Schedule** to overlap computation and communication;



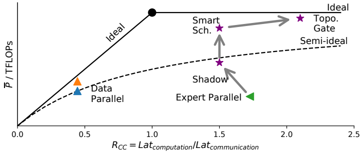
FasterMoE in a Nutshell

- MoE models are **large** and **dynamic**.
- To train the trending pre-trained models more efficiently,
  - **Shadow Experts** to relieve load imbalance;
  - Use **Smart Schedule** to overlap computation and communication;
  - Design **Topology-aware Gates** to reduce network congestion;



FasterMoE in a Nutshell

- MoE models are **large** and **dynamic**.
- To train the trending pre-trained models more efficiently,
    - **Shadow Experts** to relieve load imbalance;
    - Use **Smart Schedule** to overlap computation and communication;
    - Design **Topology-aware Gates** to reduce network congestion;
- All above are guided by the **DDL-Roofline** model.



The End

Thanks and Questions are Welcomed

- Contact: `hja20@mails.tsinghua.edu.cn`
- Source code: https://github.com/thu-pacman/FasterMoE