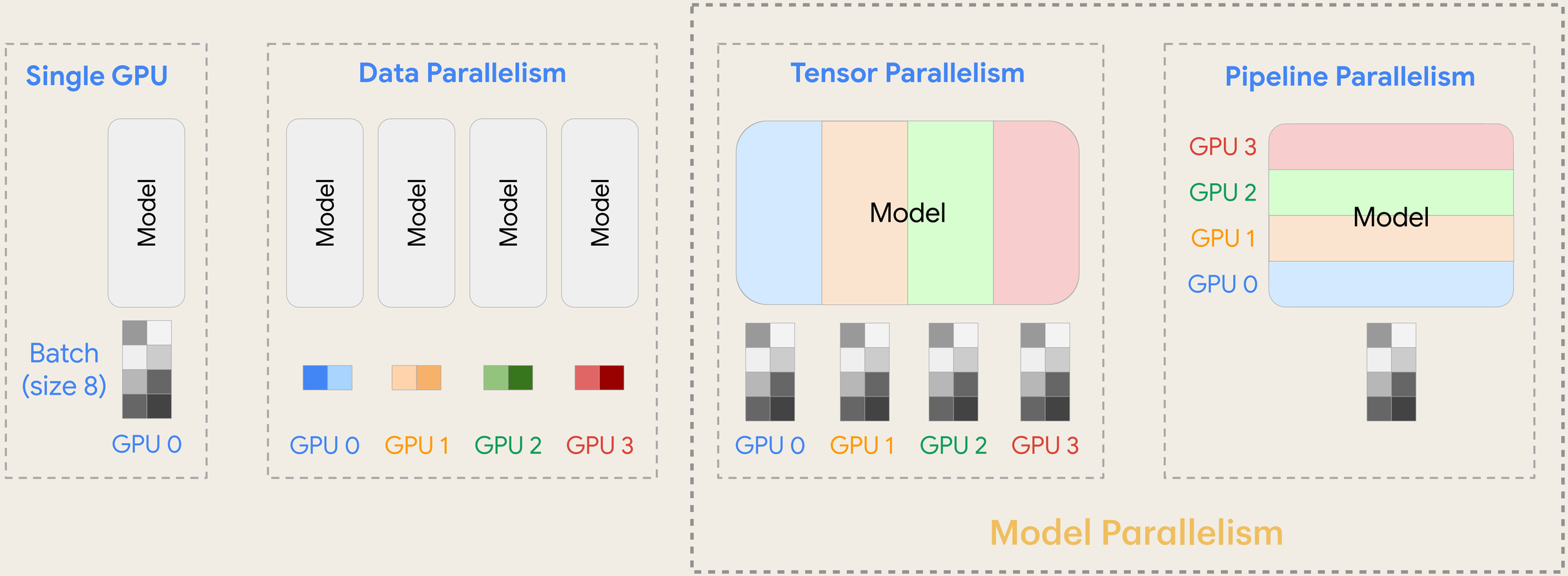
Elastic Averaging for Efficient Pipelined DNN Training

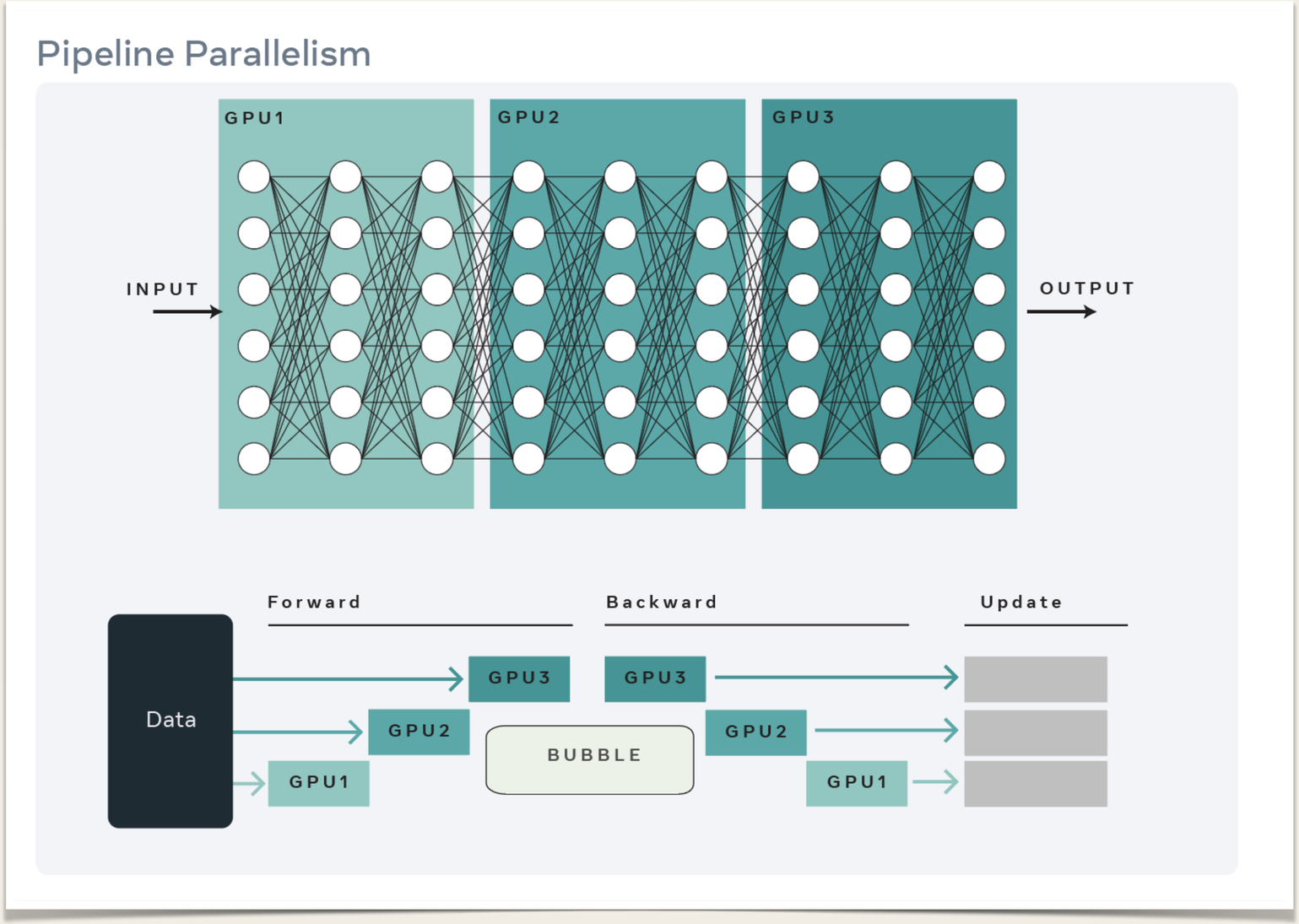
Chen, Zihao; Xu, Chen; Qian, Weining; Zhou, Aoying
East China Normal University

Presenter
Muhan Zhang
University of North Texas

Common Types of Parallelism in DNN Training



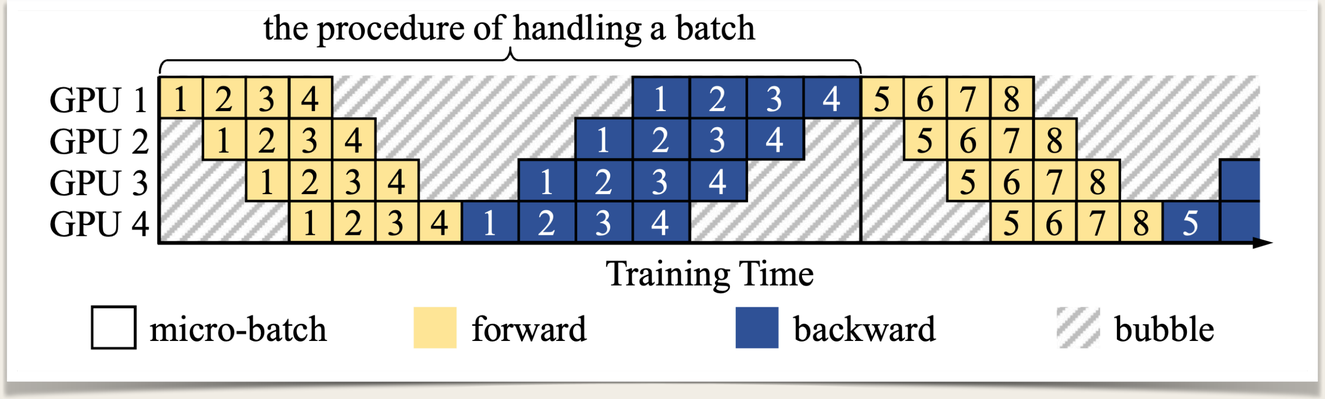
What is Pipeline Parallelism?



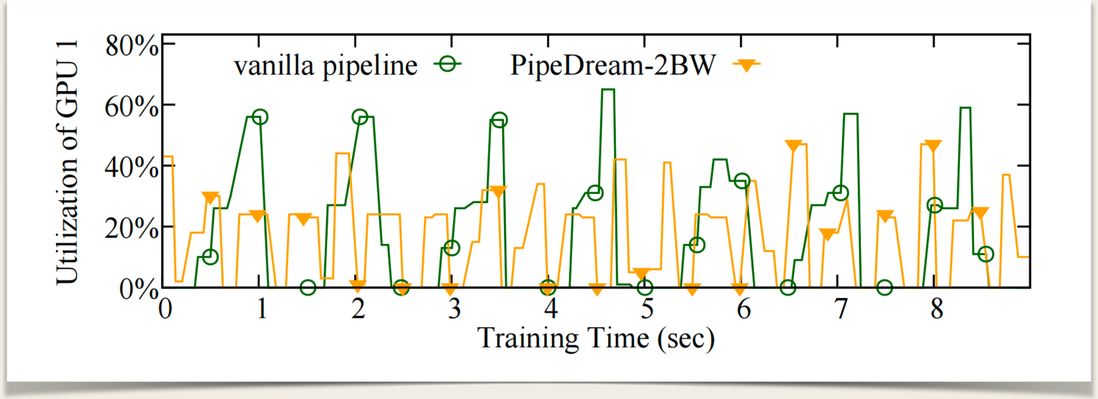
Pipeline Parallelism Challenges

- **Bubble issue:** GPUs become idle during backward propagation.
- **Low peak utilization:** communication delays and underutilized GPU resources.
- Existing solutions: multi-version pipelines like PipeDream introduce high memory footprints but fail to fully overlap communication and computation.

Vanilla Pipeline Parallelism



Underutilized GPU in the Example of BERT

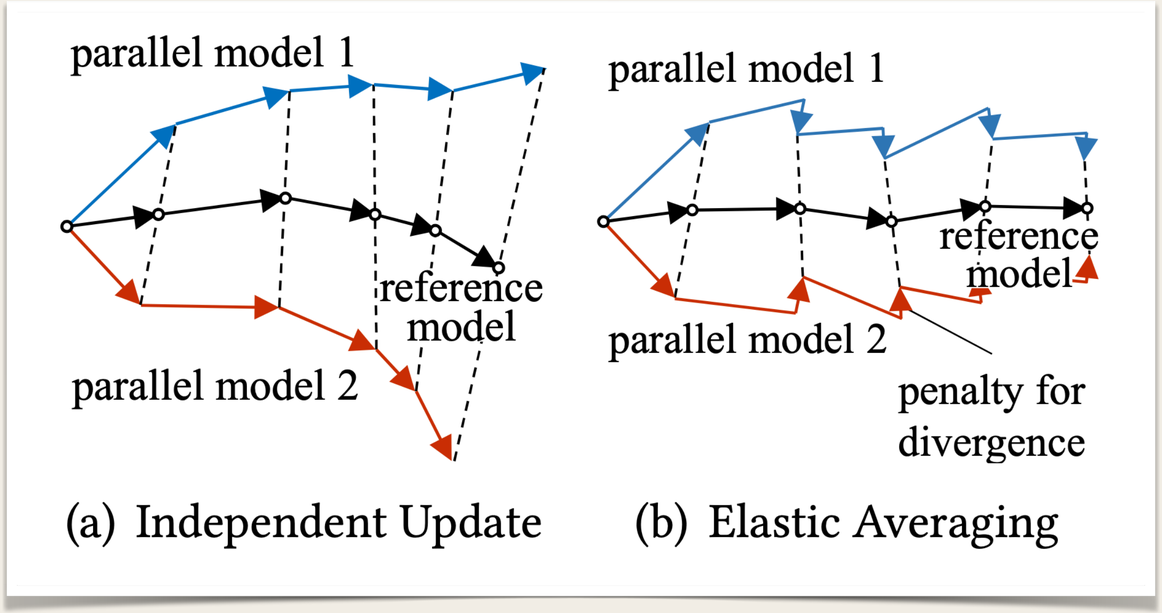


What is Elastic Averaging?

- Elastic Averaging (EA) is introduced to handle **multiple parallel pipelines** and **improve GPU utilization.**

- It ensures that parallel models **do not diverge** too much by periodically averaging weights with a reference model.

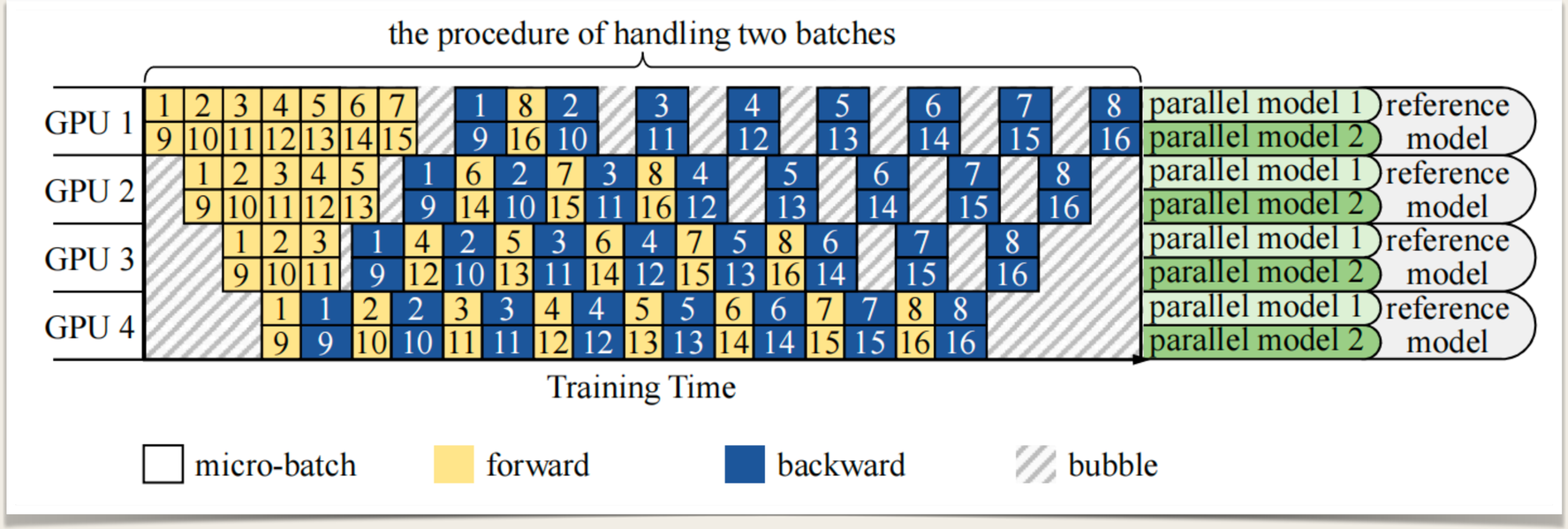
Elastic Averaging



Challenges in Direct Combination

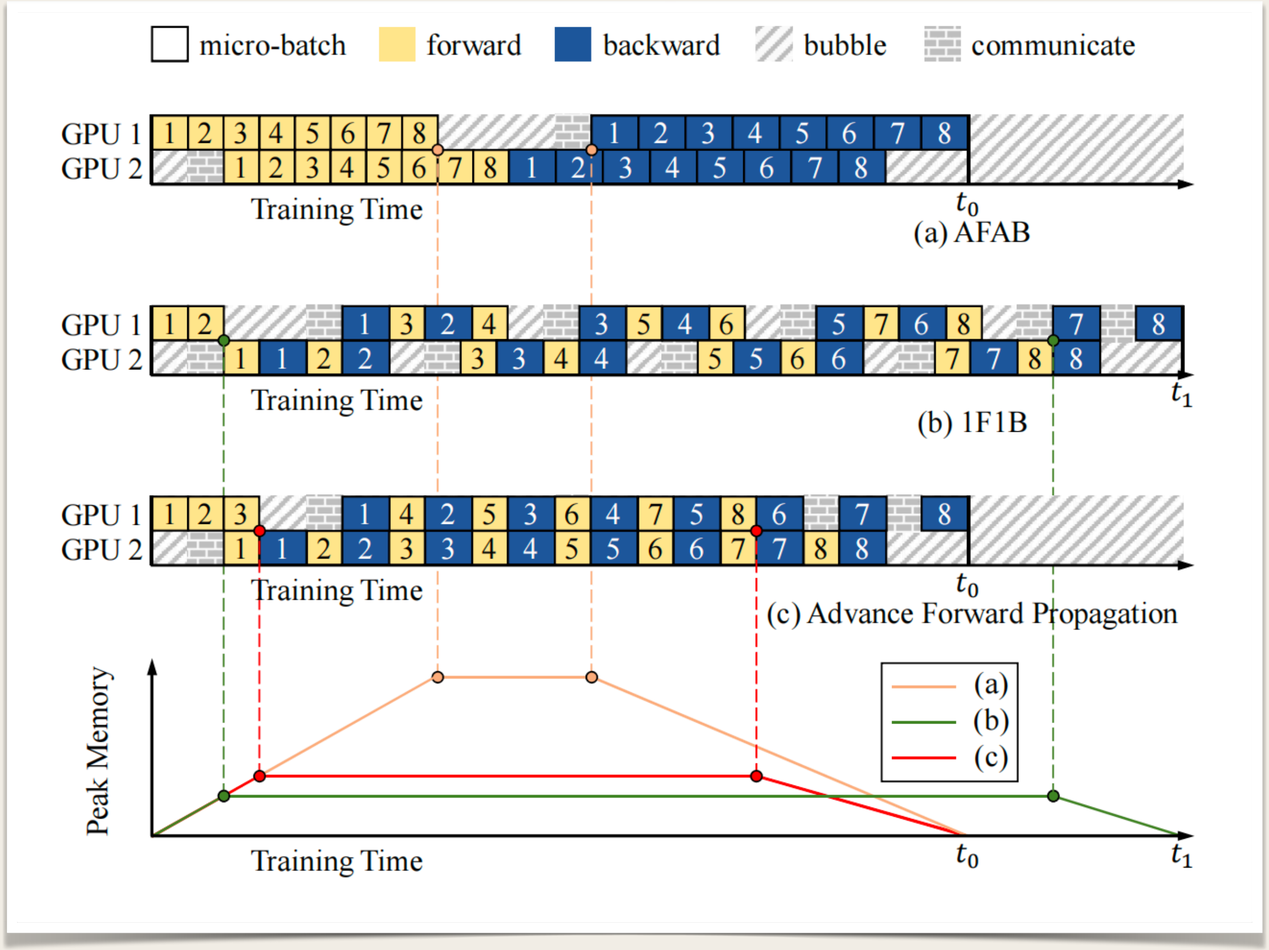
- **Increased memory usage** due to model replicas.
- Need a performance-aware schedule to solve this.
- Introducing **Advance Forward Propagation.**

Pipeline Parallelism with Elastic Averaging



Advance Forward Propagation

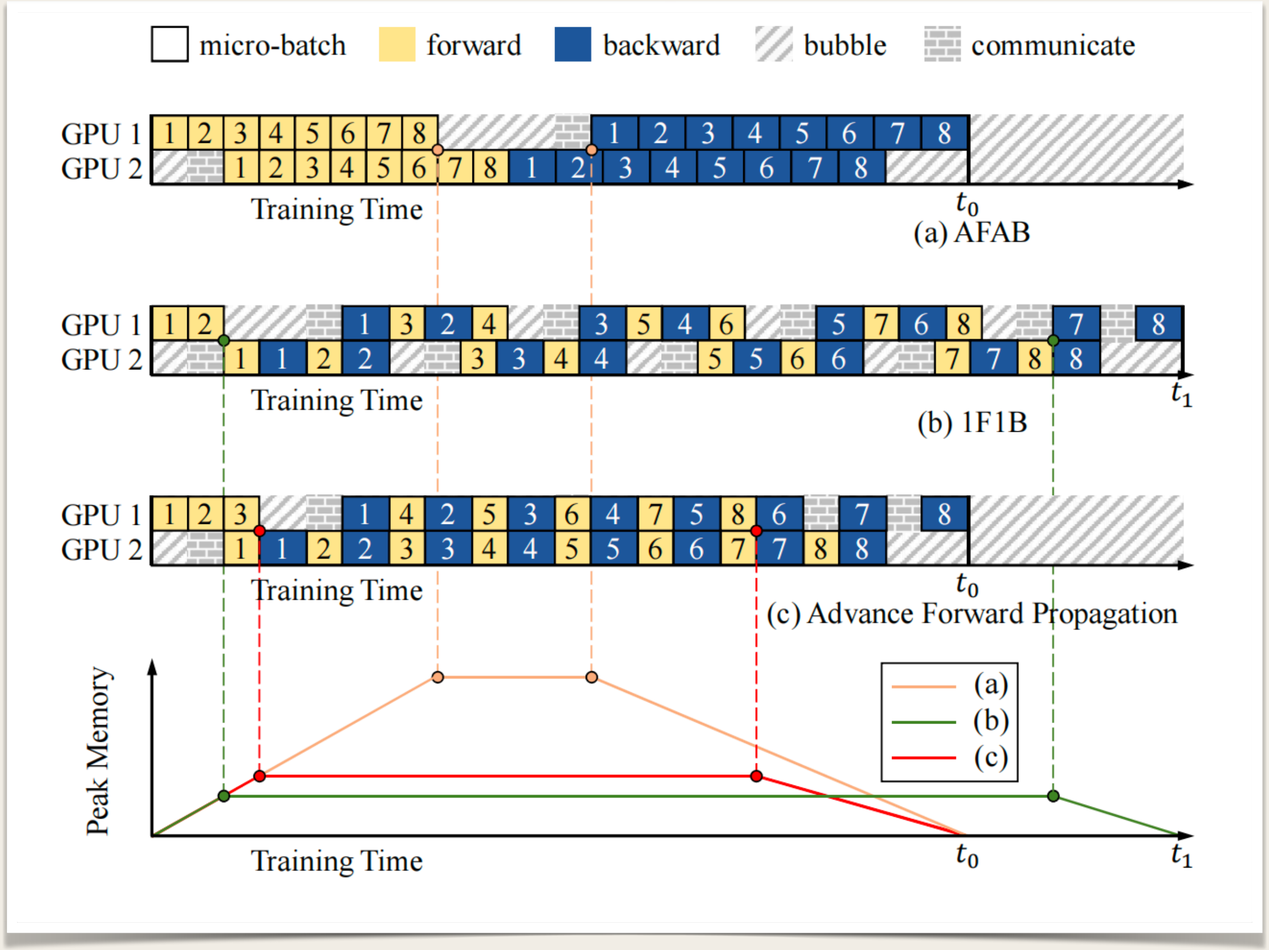
- AFAB: High performance but requires more memory.
- 1F1B: Lower memory usage but higher communication overhead, longer training time.



Different Schedules on One Batch

Advance Forward Propagation

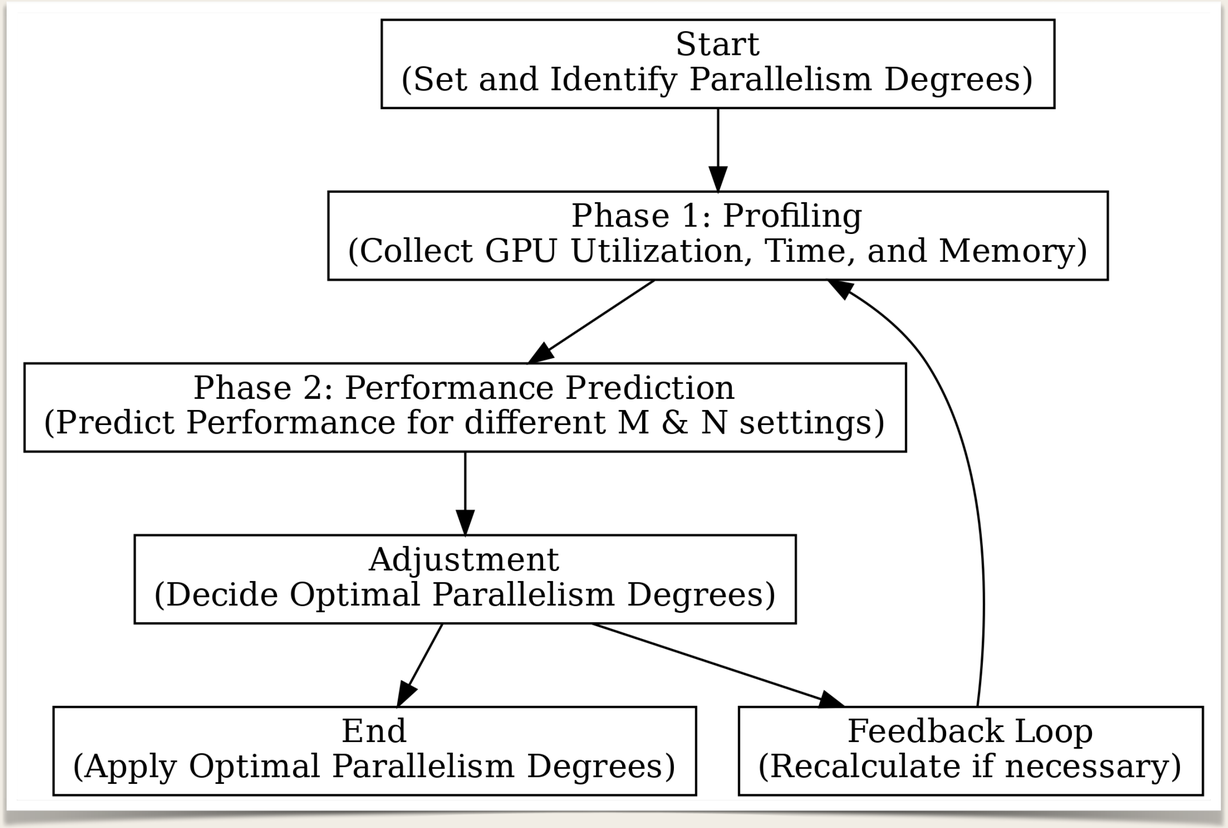
- Advance Forward Propagation:
  - Allows forward propagation to be done in advance, utilizing idle GPU time.
  - Balances memory and performance.
  - Overlaps computation with communication.
  - Reduces idle time (uses bubble time effectively).



Different Schedules on One Batch

Profiling-Based Tuning

- Automatically adjusts parallelism degrees (the micro-batch number M & the parallel pipeline number N).
- Predicts performance based on a short profiling run.
- Focuses on predicting computation, communication, and bubble times.



Profiling-Based Tuning Flowchart

Experimental Setup

- Tested on a 3-node cluster with Tesla V100 GPUs.
- Baselines:
  - PyTorch: Represents data parallelism.
  - GPipe: Represents pipeline parallelism.
  - PipeDream, PipeDream-2BW, Dapple: Variants of pipeline parallelism.
- Comparison:
  - AvgPipe compared with each baseline under the same or lower memory constraints.

Comparison of AvgPipe and Baselines: Training Time, GPU Memory, Utilization, and Statistical Efficiency

GNMT (Neural Machine Translation):

- AvgPipe(G) achieves a 1.3x speedup over GPipe and 1.6x over PipeDream.
- The performance improves due to reduced bubbles and improved GPU utilization.

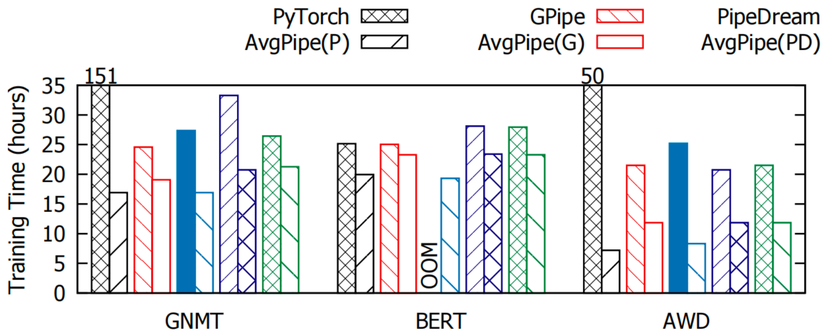


Figure 11. Training Time

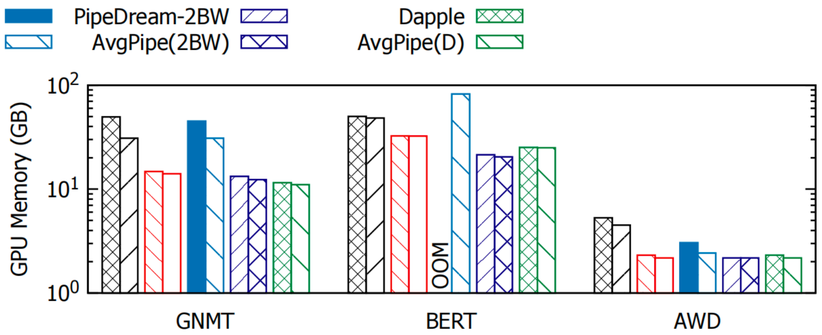


Figure 12. GPU Memory Footprints

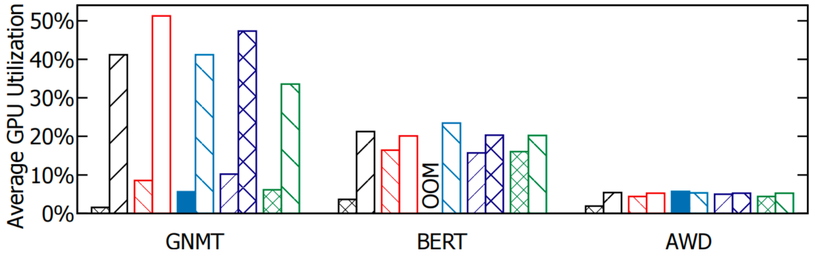


Figure 13. Averaged GPU Utilization

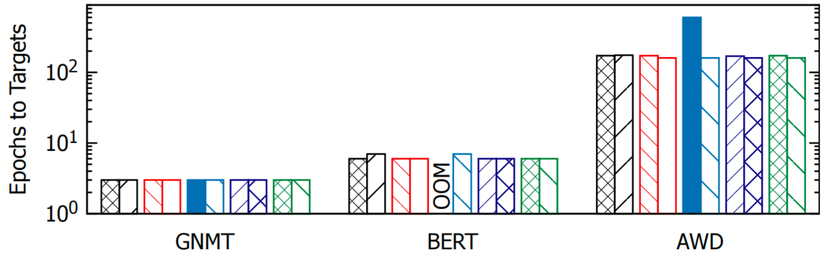


Figure 14. Statistical Efficiency

Comparison of AvgPipe and Baselines: Training Time, GPU Memory, Utilization, and Statistical Efficiency

BERT:

AvgPipe achieves 1.3x speedup over PyTorch and 1.1x over GPipe.

PipeDream fails due to memory issues, whereas AvgPipe optimizes memory and communication overhead.

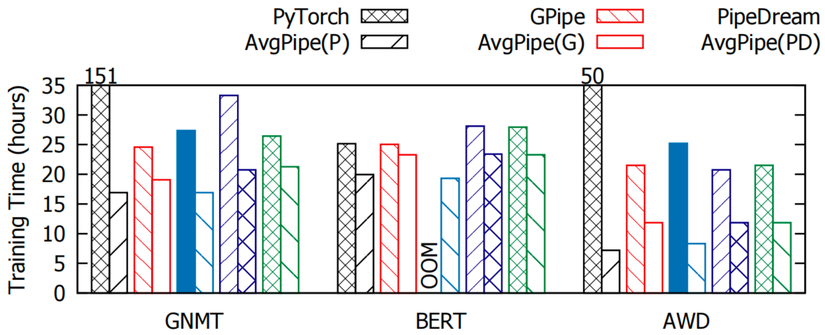


Figure 11. Training Time

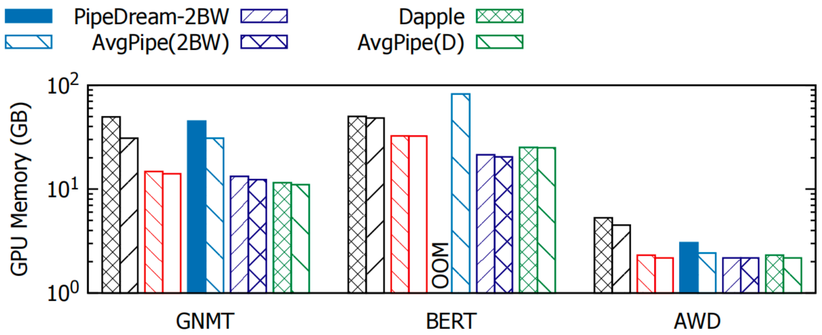


Figure 12. GPU Memory Footprints

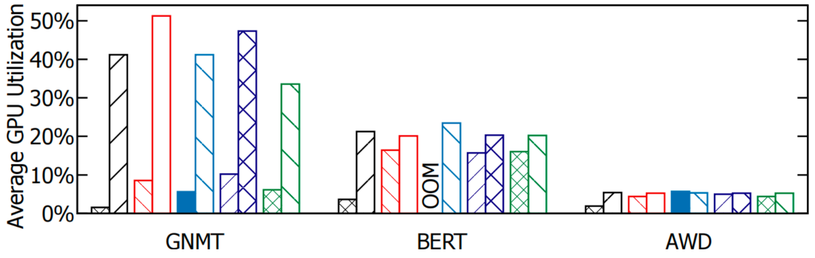


Figure 13. Averaged GPU Utilization

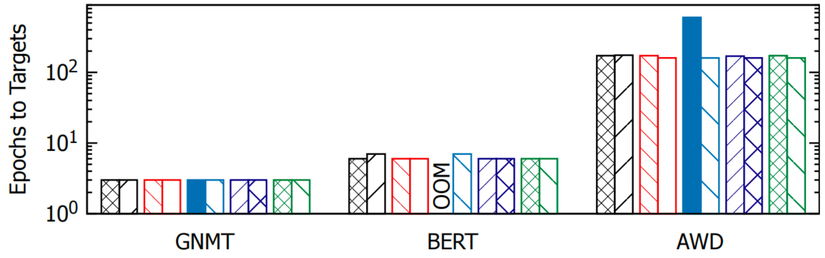


Figure 14. Statistical Efficiency

Comparison of AvgPipe and Baselines: Training Time, GPU Memory, Utilization, and Statistical Efficiency

AWD:

AvgPipe(P) outperforms PyTorch by 7.0x and achieves 1.8x speedup over GPipe and PipeDream-2BW.

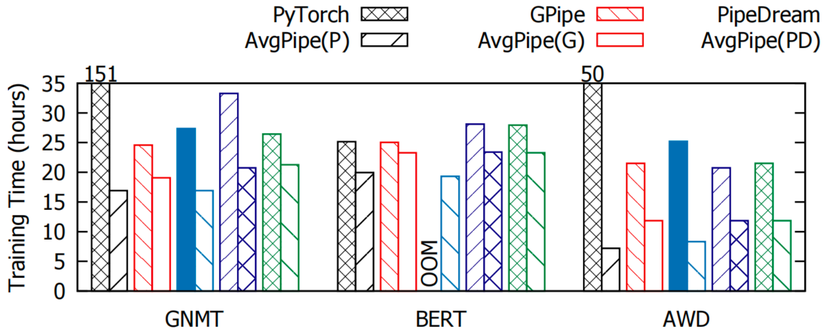


Figure 11. Training Time

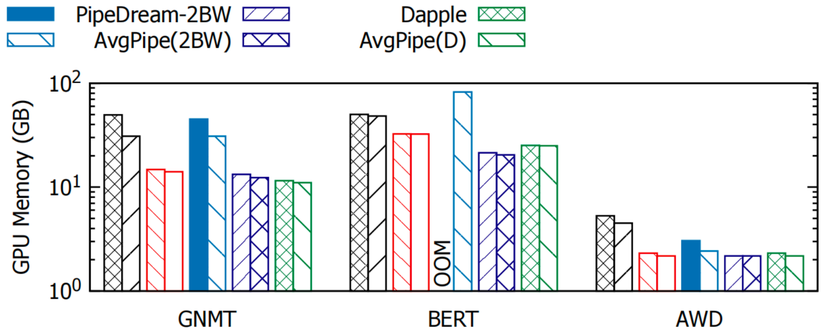


Figure 12. GPU Memory Footprints

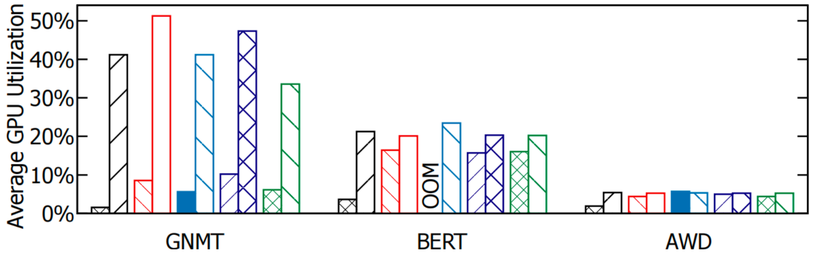


Figure 13. Averaged GPU Utilization

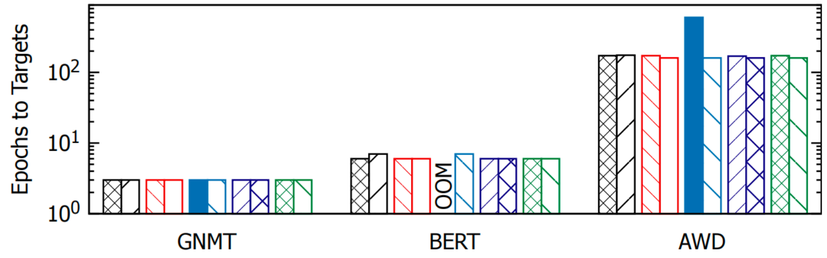


Figure 14. Statistical Efficiency

Detailed Analysis of GNMT: Batch Size Impact and GPU Utilization

AvgPipe(G) consistently outperforms GPipe, especially at larger batch sizes (192, 256).

AvgPipe(G) consistently outperforms GPipe, especially at larger batch sizes (192, 256).

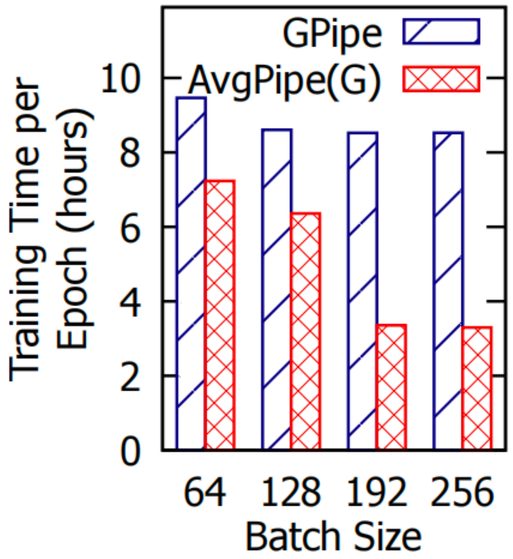


Figure 15. Varying Batch Size for GNMT

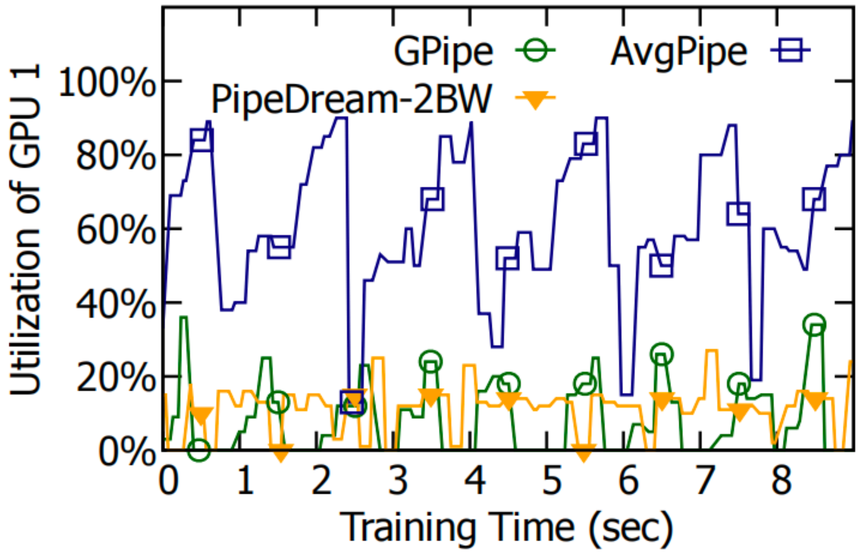
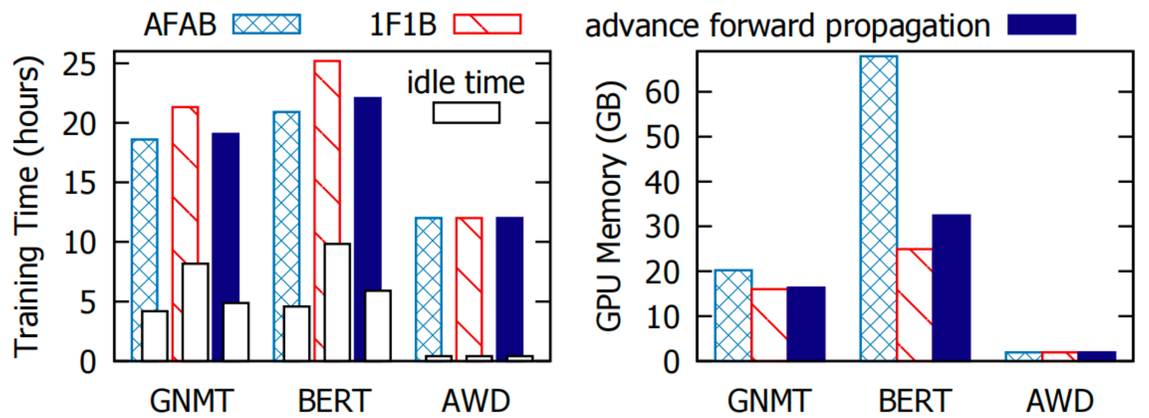


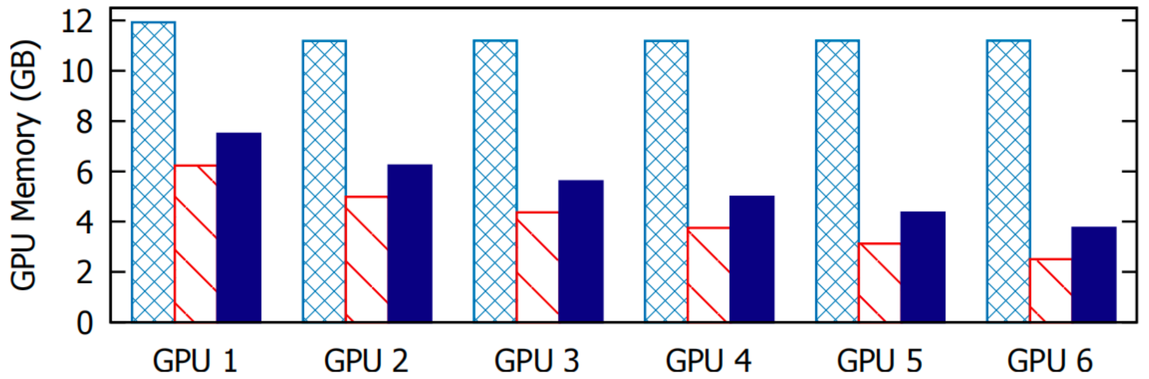
Figure 16. GPU Utilization Over Time For GNMT

Detailed Analysis of GNMT: Batch Size Impact and GPU Utilization



(a) Training Time

(b) GPU Memory Footprints



(c) Memory Footprint of each GPU for BERT

Advance forward propagation reduces training time and idle periods while efficiently managing memory usage, offering a balance between AFAB's speed and 1F1B's memory savings across all GPUs.

Detailed Analysis of GNMT: Batch Size Impact and GPU Utilization

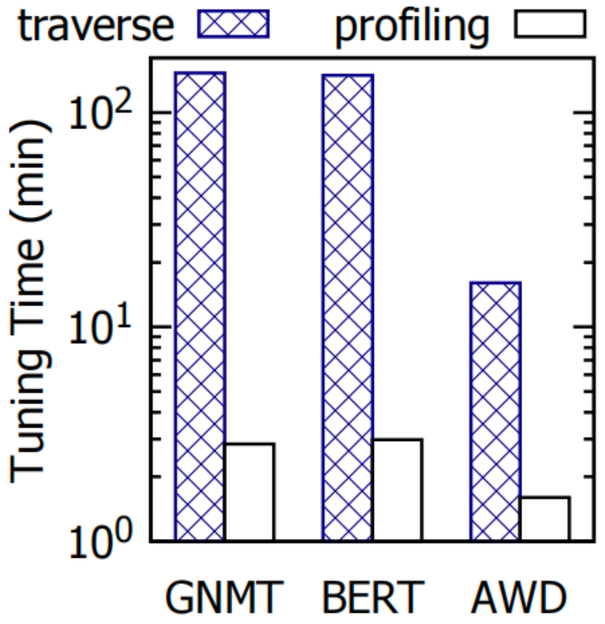


Figure 18. Tuning Cost

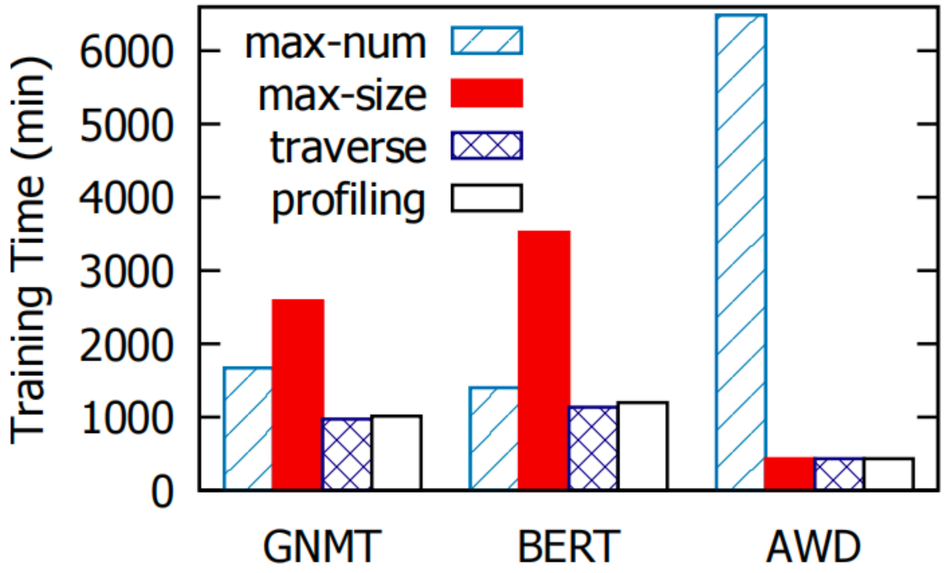


Figure 19. Tuning Result

Despite the faster tuning, profiling achieves similar or better training times compared to traversal and other methods like max-num and max-size.

Key Contributions of the Paper

1. Elastic Averaging-Based Framework

   Enhances pipeline parallelism performance.

   Decouples from specific optimizers, increasing flexibility.

2. Advance Forward Propagation

   Conserves memory while maintaining performance.

   Reduces communication overhead, preventing degradation in large models.

3. Profiling-Based Tuning Method

   Efficient tuning of parallelism degrees with minimal profiling time.

   Maximizes AvgPipe’s performance with negligible tuning cost.

Limitation

1. Scalability Validation:

   The paper primarily conducted experiments on 6 GPUs. Further validation of performance and scalability on larger GPU clusters is needed.

   The applicability to ultra-large scale models (such as GPT-3 level) has not been verified.

2. Convergence Analysis:

   The paper mainly focuses on training speed. A more in-depth theoretical analysis could be conducted on the potential changes in convergence behavior brought by using Elastic Averaging.

Thank you