Zero Bubble Pipeline Parallelism

Penghui Qi, Xinyi Wan, Guangxing Huang, Min Lin

(Paper Presented by Aydan Pirani)

Parallelism Techniques

- Training large models often requires a vast amount of interconnected GPUs

- Data Parallelism → splits data across multiple GPUs, then computes in "chunks"
  - Works until a single model is too big (too many parameters)

- Model Parallelism → splitting a model into multiple parts
  - Tensor Parallelism: splits the matrix multiplication to several devices
  - Pipeline Parallelism: model split into different stages, to be run on devices
- ZeRO → shards parameters across devices, but maintains simplicity

Which Technique?

- Are we limited by GPU-GPU communication bandwidth?

- No: DP, TP and ZeRO
- Yes: PP

- **Goal: making pipeline parallelism more performant**

Neural Networks

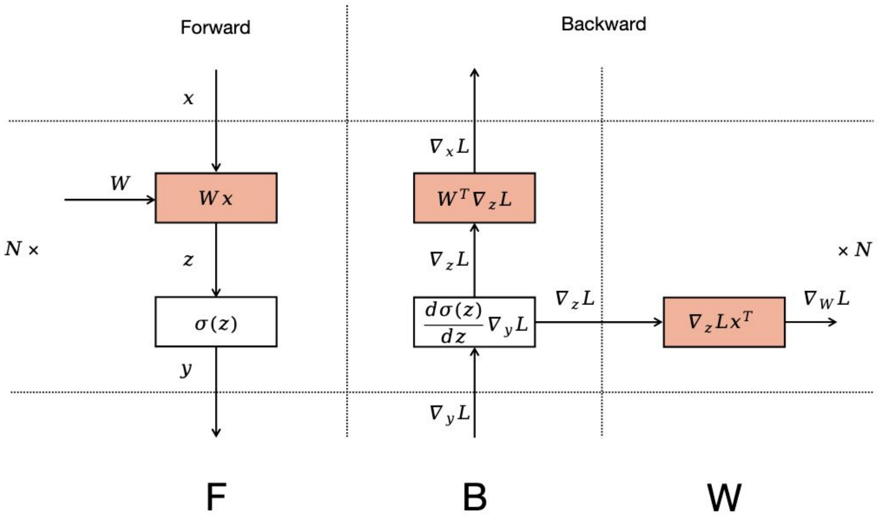
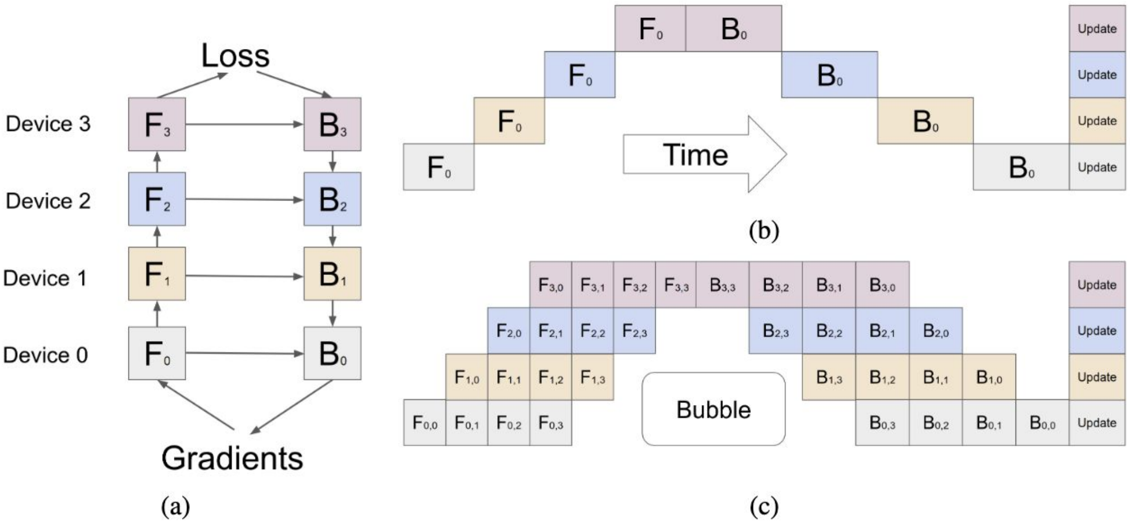


Figure 1: Computation Graph for MLP.

Neural Networks (Explained)

- Forward Pass: Input → Output
- Backward Pass:
  - Gradient with respect to input → used to backprop to previous layers
  - Gradient with respect to weights → used to update the weights
- Break up these passes, then perform pipeline parallelism

Pipeline Parallelism



(GPIPE)

GPIPE's Approach to Pipeline Bubbles

- GPipe attempted to mitigate these bubbles
  - Incrementing concurrent batches
  - Discards (and recomputes) some intermediate activations

- Asynchronous PP allows each stage of the pipeline to process data without waiting
  - Improvement over GPipe
- Synchronous setting: one-forward-one-backward (1F1B)

One Forward, One Backward (PipeDream)

- Each worker alternates between performing a forward pass and a backward pass
- GPUs always actively working on some part of the computation
- Reduces the need to store multiple activations
- Asynchronous updates between mini-batches, reducing pipeline stalls

1F1B Interleaving

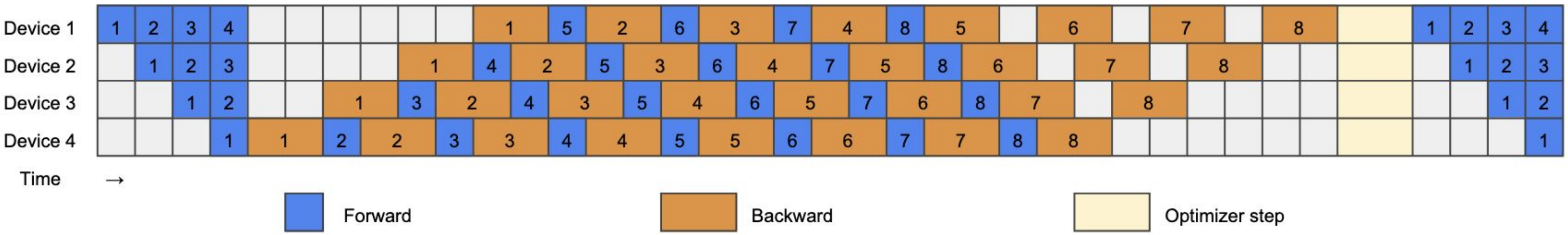


Figure 2: 1F1B pipeline schedule.

ZB-H1 and ZB-H2 Schedules

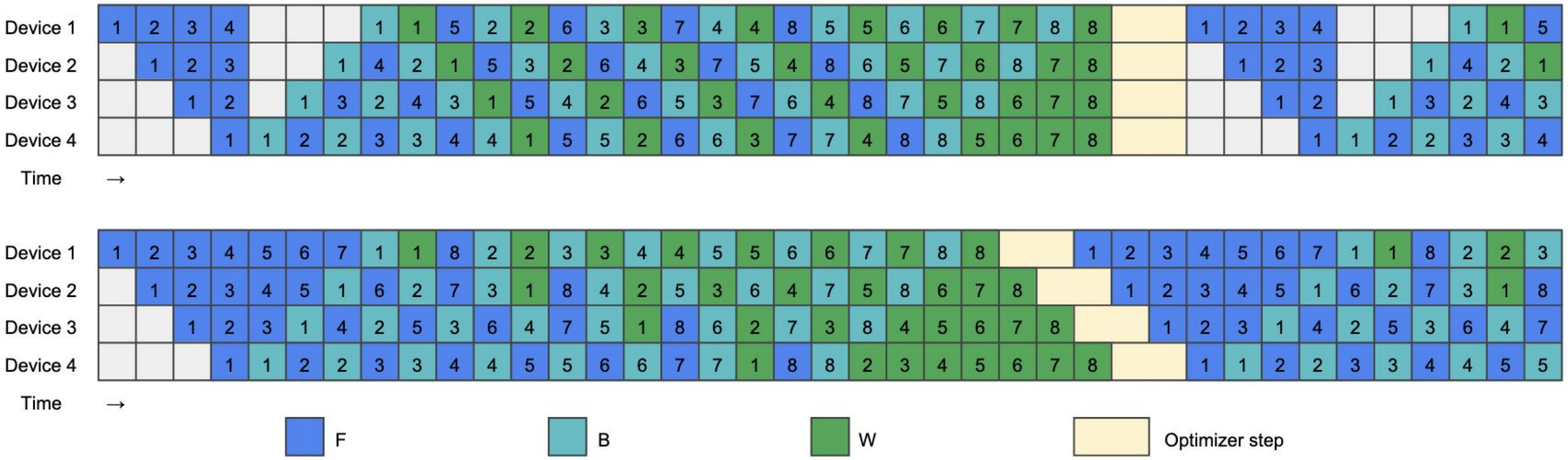


Figure 3: Handcrafted pipeline schedules, top: ZB-H1; bottom: ZB-H2

ZB-H1 and ZB-H2 Schedules

- **ZB-H1:**
  - Ensures max peak memory usage doesn't exceed 1F1B
  - All workers maintain the same number of in-flight microbatches
  - B initiated earlier, hence bubble size drops

- **ZB-H2:**
  - Larger memory footprint than 1F1B, zero bubble schedule
  - Reorder passes
  - synchronization between the optimizer steps is removed here

1F1B, ZB-H1 and ZB-H2

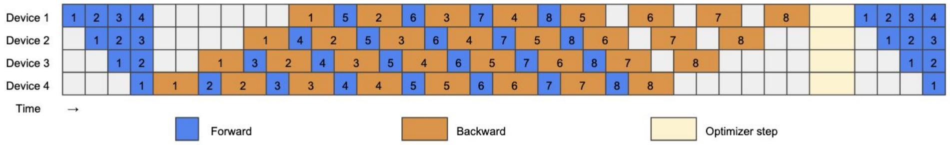


Figure 2: 1F1B pipeline schedule.

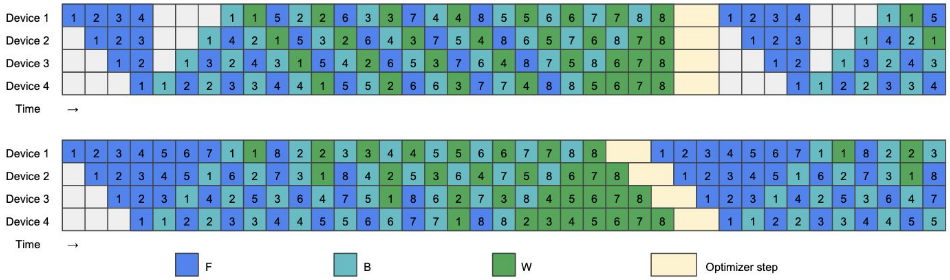


Figure 3: Handcrafted pipeline schedules, top: ZB-H1; bottom: ZB-H2

Peak Activation Memories

M_B ,M_W represent the memories taken by a B/W pass

T_F ,T_B, T_W represent the time taken by a F/B/W pass

p represents the phases/number of pipelines

Table 2: Comparison between 1F1B and our handcrafted schedules.

Schedule	Bubble size	Peak activations memory
1F1B	$(p-1)(T_F+T_B+T_W)$	pM_B
ZB-H1	$(p-1)(T_F+T_B-T_W)$	pM_B
ZB-H2	$(p-1)(T_F+T_B-2T_W)$	$(2p-1)M_B$

Issues with ZB-Hx

- Cannot assume that $T_F = T_B = T_W$
- There is a communication latency (that is ignored)
- *Balancing bubble size/memory limit* is challenging

Design a heuristic, and then present an ILP (Integer Linear Programming) problem

The Heuristic

- **Warm-Up:** Schedule as many F passes as possible before the first B pass to minimize bubbles, staying within memory limits.

- **1F1B:** Schedule 1F1B, and insert W (weight update) if a bubble is large enough.

- Ensure stage i always schedules one more F than stage i+1.
- After all F and B passes are done, schedule remaining W passes sequentially.

Optimizer Synchronization

- Generally requires a “barrier” for all pipeline stages
  - Makes zero bubble impossible, because of stragglers

- Most of the time the global states have no effects
- Replace the before-hand synchronizations with a post-update validation

(Similar to optimistic vs. pessimistic concurrency control)

Optimizer Synchronization

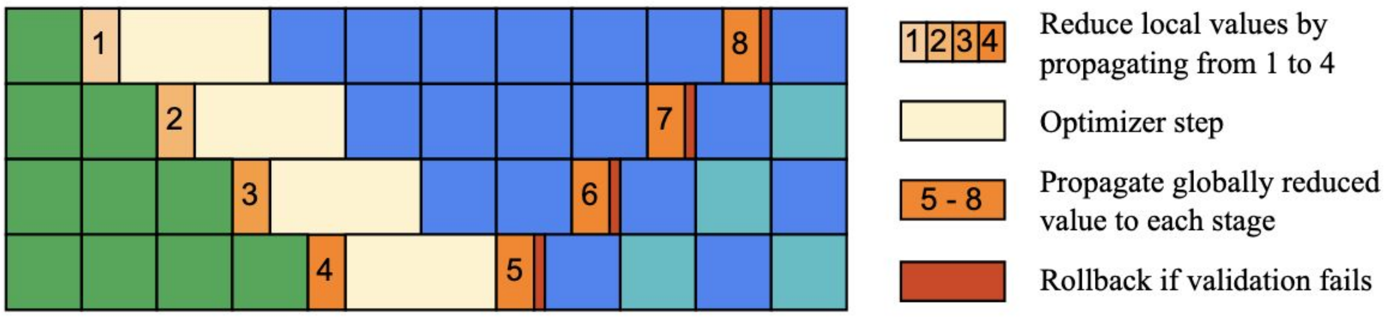


Figure 4: The post-validation strategy to replace optimizer synchronization.

Experiment Setup

- **Methods:** 1F1B, 1F1B-I, ZB-1p, ZB-2p
- Open-source Megatron-LM project, models analogous to GPT-3
- Specific number of iterations for profiling, collecting empirical measurement
- Up to 32 A100s, 4 interconnected nodes, verifiable correctness

Table 3: Models and fixed settings used in experiments

Model	Layers	Attention Heads	Hidden Size	Sequence Length	Pipelines (GPUs)	Microbatch Size	Number of Microbatches
1.5B	22	24	2304	1024	8	6	24 / 32 / 64
6.2B	30	32	4096	1024	8	3	24 / 32 / 64
14.6B	46	40	5120	1024	16	1	48 / 64 / 128
28.3B	62	48	6144	1024	32	1	96 / 128 / 256

Main Results

5.2 Main results

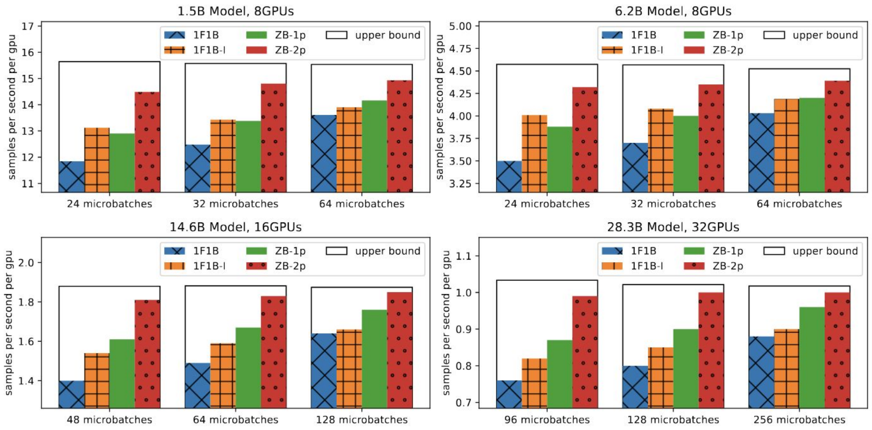


Figure 5: Comparison of throughput across different pipeline schedules.

Bubble Rates

Table 5: Bubble rates of 1F1B, 1F1B-I, ZB-H1, ZB-H2, ZB-1p, ZB-2p under different settings.

Model	#Stage (p)	#Microbatch (m)	1F1B	1F1B-I	ZB-H1	ZB-H2	ZB-1p	ZB-2p
1.5B	8	24	0.2431	0.1055	0.1585	0.1083	0.1585	**0.0433**
		32	0.1985	0.0818	0.1242	0.0837	0.1242	**0.0039**
		64	0.1240	0.0443	0.0674	0.0444	0.0674	**0.0026**
6.2B	8	24	0.2347	0.0808	0.1323	0.0698	0.1323	**0.0029**
		32	0.1898	0.0628	0.1045	0.0559	0.1045	**0.0022**
		64	0.1091	0.0320	0.0554	0.0294	0.0554	**0.0010**
14.6B	16	48	0.2552	0.1104	0.1397	0.0672	0.1397	**0.0066**
		64	0.2082	0.0852	0.1088	0.0516	0.1088	**0.0054**
		128	0.1251	0.0445	0.0576	0.0266	0.0576	**0.0028**
28.3B	32	96	0.2646	0.1493	0.1421	0.0641	0.1421	**0.0038**
		128	0.2168	0.1164	0.1106	0.0490	0.1106	**0.0029**
		256	0.1352	0.0624	0.0594	0.0257	0.0594	**0.0018**

Motivation Behind ZB-V

- ZB-2p eliminates pipeline bubbles, but 2x memory consumption.

- Minimize idle time while maintaining the same memory constraints as 1F1B.
- Divide the model into 2p chunks, assigning two chunks per worker.

- Workers are assigned chunks in a sequential manner, alternating from start to end:
    - Ensures forward and backward passes originate from the same worker.

Advantages of ZB-V

- First worker initiates backward pass without waiting, leading to quicker memory clearance.
- Uniform memory consumption across all workers.

- Half the memory compared to ZB-H2.
- Achieves zero bubble with memory usage equivalent to 1F1B.
    - **ONLY** under a very specific setting

ZB-V Bubble Rates

Table 8: Bubble rates of 1F1B, 1F1B-I, ZB-H1, ZB-H2 and ZB-V under different settings.

Model	#Stage (p)	#Microbatch (m)	1F1B	1F1B-I	ZB-H1	ZB-H2	ZB-V
6.2B	16	48	0.2668	0.1499	0.1536	0.0823	**0.0697**
		64	0.2206	0.1169	0.1198	0.0630	**0.0533**
		128	0.1390	0.0621	0.0637	0.0325	**0.0274**
14.6B	24	72	0.2699	0.1519	0.1439	**0.0628**	0.0638
		96	0.2229	0.1184	0.1121	**0.0480**	0.0483
		192	0.1403	0.0630	0.0595	**0.0247**	0.0250
28.3B	32	96	0.2676	0.1509	0.1429	0.0629	**0.0593**
		128	0.2204	0.1177	0.1111	0.0478	**0.0451**
		256	0.1362	0.0626	0.0593	0.0251	**0.0236**

Bubble Rates vs. Memory Limits

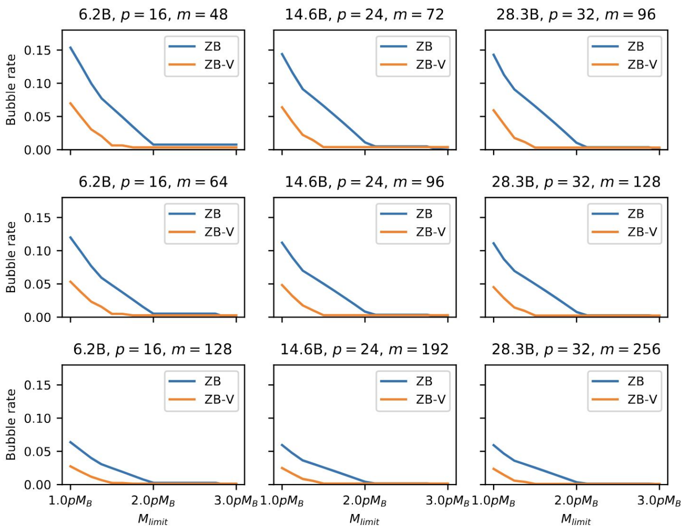


Figure 9: The relation between memory limit and bubble rate for ZB-V, compared with the heuristic method in Section 3.1.

Thoughts

- Very strong work

- Innovative approach to reduce pipeline bubbles without significant memory increase (ZB-p1).
- Outperforms 1F1B and GPipe, but ZB-H2 has a higher memory footprint
- ZB-V is **VERY** impactful, does a great job limiting bubbles

- Would be valuable to integrate multiple parallelism methods, do we see speedup?

Thanks for Listening!

Any questions?