Memory-Efficient Pipeline-Parallel DNN Training

Deepak Narayanan[3] Amar Phanishayee[1] Kaiyu Shi[2] Xie Chen[2] Matei Zaharia[3]

[1]Microsoft Research [2]Microsoft [3]Stanford University

Presenter: Shiwei Zhang

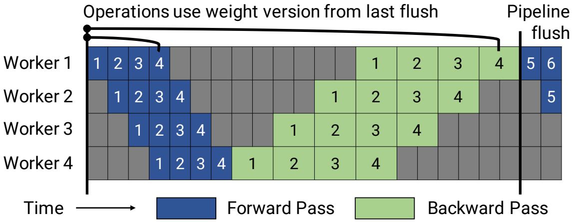
Introduction

Pipeline Parallelism

DNN models are becoming increasingly large. Traditional model parallelism can either lead to GPU under-utilization (inter-layer MP) or high communication overhead (tensor MP).

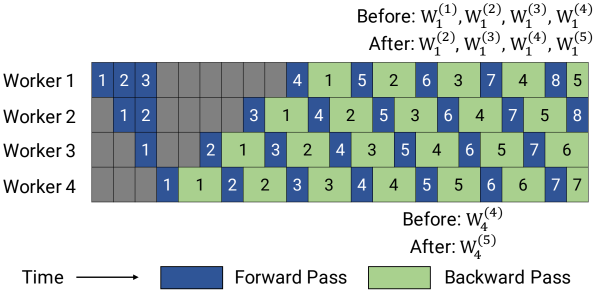
Pipelining pushes multiple inputs in sequence through a series of workers that each manage one part of the model, achieving both high utilization and good computation-communication overlap. However, existing pipelining schemes, **GPipe** and **PipeDream**, have their own trade-off between throughput and memory footprint.

GPipe



GPipe divides a minibatch into m microbatches that share the same version of parameters. Pipeline flushing is required after every m microbatches.

PipeDream



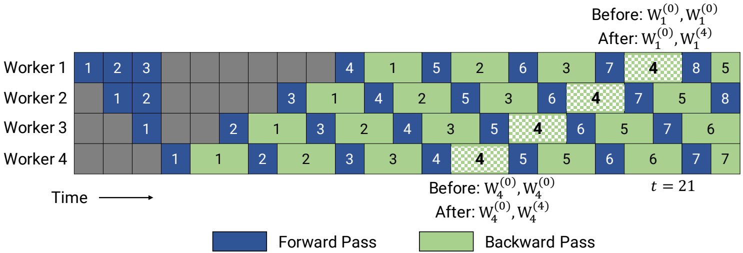
PipeDream introduces multiple weight versions and gradient staleness to maximize throughput. At most d versions of weights can present at the same time.

Methods

- Double-Buffered Weight Updates (2BW) that has both high throughput and low memory footprint
- A planing algorithm that yields effective parallelization schemes for many of today's large model architectures.

Double-Buffered Weight Updates (2BW)

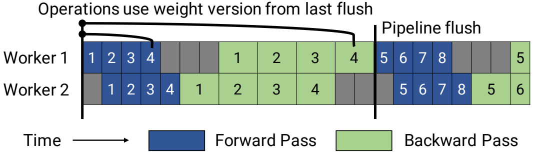
Double-Buffered Weight Updates (2BW)



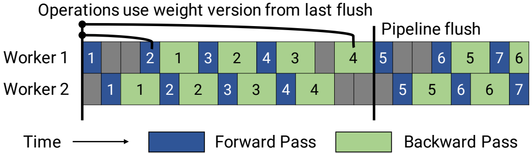
Double-Buffered Weight Updates introduces both m microbatches and 2 weight versions. It uses the same 1F1B scheduling as in PipeDream.

PipeDream-Flush

The paper also introduces PipeDream-Flush, which is simply GPipe with 1F1B scheduling. It have the same semantics as GPipe but has lower memory footprint.



(a) GPipe.



(b) PipeDream-Flush.

Memory footprint Comparison

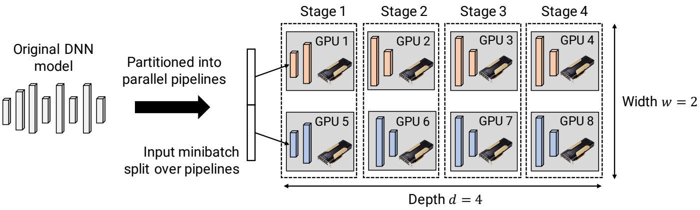
- **GPipe**: m activations of batchsize B/m, 1 weight version.
- **PipeDream**: d activations of batchsize B, d weight versions.
- **PipeDream-2BW**: d activations of batchsize B/m, 2 weight versions.
- **PipeDream-Flush**: d activations of batchsize B/m, 1 weight versions.

Semantics Comparison

- **GPipe**: $W^{(t+1)} = W^{(t)} - \nu \cdot \nabla f(W^{(t)})$
- **PipeDream**: $W^{(t+1)} = W^{(t)} - \nu \cdot \nabla f(W^{(t-d+1)})$
- **PipeDream-2BW**: $W^{(t+1)} = W^{(t)} - \nu \cdot \nabla f(W^{(t-1)})$
- **PipeDream-Flush**: $W^{(t+1)} = W^{(t)} - \nu \cdot \nabla f(W^{(t)})$

Planner

Planner



The planner of PipeDream-2BW is an *exhaustive* searching over a *reduced* space.
The strategy space is (w, d, b, r):

- w: the data parallelism replicas,
- d: the number of stages,
- b: the microbatch size,
- r: boolean whether activations should be recomputed

Cost Model

- In the experiments, the authors use end-to-end profile-based cost functions.
- They also presents a closed form cost function that needs less profiling but provides lower accuracy. It uses the forward and backward time of each block at different batchsizes to simulate the execution.

Example Configurations

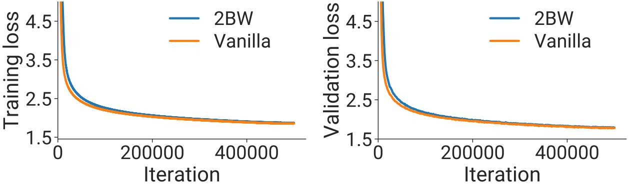
Model	Machine	(w, d)	Thpt.	b	r	Opt.
BERT-24	2 8×1080Ti	$(8,2)$	151	8	F	✓
		$(8,2)$	141	8	T	
BERT-24	4 8×V100	$(8,4)$	42	1	F	
		$(8,4)$	111	4	T	✓
BERT-48	8 8×V100	$(4,16)$	59	1	T	
		$(8,8)$	69	1	T	✓
		$(16,4)$	25	1	T	
BERT-192	8 8×V100	$(1,64)$	15	1	T	
		$(2,32)$	18	1	T	
		$(4,16)$	21	1	T	✓

Evaluation

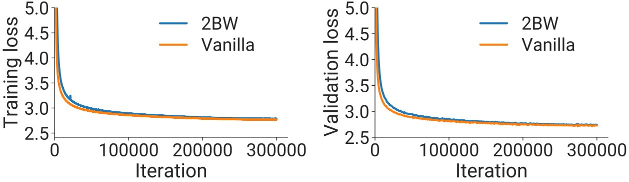
Experiments Setup

- Hardware:
  (a) 8 8x V100 (16GB) with NVLink.
  (b) single machine with 8x V100 (32GB).
- Implementation: PyTorch implementation adapted from Megatron.
- Models: BERT and GPT with 1.3, 2.2 and 3.9 billions of parameters.
- Baselines: Model parallelism (inter-layer MP and tensor MP) and GPipe.

Convergence

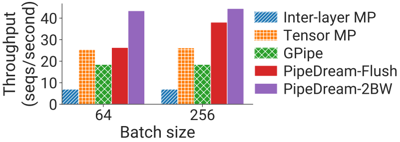


(a) BERT, 355M (batch size = 1024).

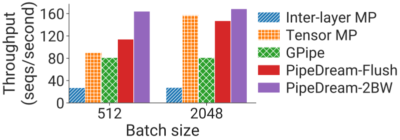


(b) GPT, 355M (batch size = 512).

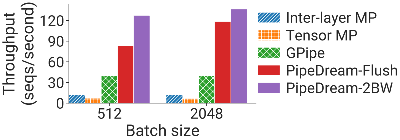
Throughput



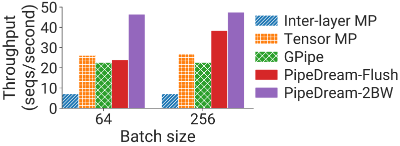
(a) GPT, 2.2B, 8-way model parallelism (8×V100s).



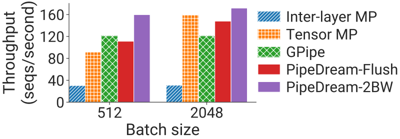
(b) GPT, 2.2B, 8-way model parallelism (64×V100s).



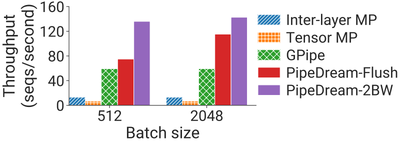
(c) GPT, 3.8B, 16-way model parallelism (64×V100s).



(a) BERT, 2.2B, 8-way model parallelism (8×V100s).



(b) BERT, 2.2B, 8-way model parallelism (64×V100s).

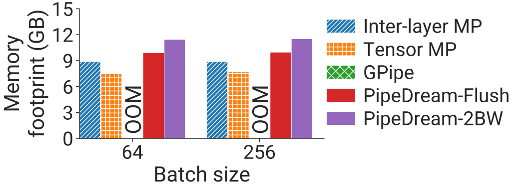


(c) BERT, 3.8B, 16-way model parallelism (64×V100s).

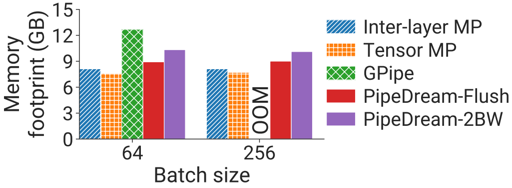
Throughput cont'd

- 16-way Tensor MP requires cross-server all-to-all communication in each layer.
- GPipe's high memory footprint limits its maximum m, which causes large bubbles in the pipeline.
- PipeDream-Flush performs better with high batchsizes. However, batchsizes cannot scale to infinity without affecting convergence.

Memory Footprint



(a) GPT, 2.2B.



(b) BERT, 2.2B.

Conclusion

Summary

- A new pipelining scheme that combines the high throughput of PipeDream and the low memory footprint and staleness of GPipe.
- A simple planner to automatically find optimal parallelism configurations.

Limitation

- It assumes repetitive model structures (therefore stage division is trivial) and homogeneous clusters.

Takeaways

- Combine two existing methods to build a new method that has the advantages of both.
- Simple searching algorithm over carefully designed space.

Thank you!