CS 498: Machine Learning System
Spring 2025

Minjia Zhang

The Grainger College of Engineering

Today

Pipeline Parallelism (Cont.)

Combining Multiple Parallelism

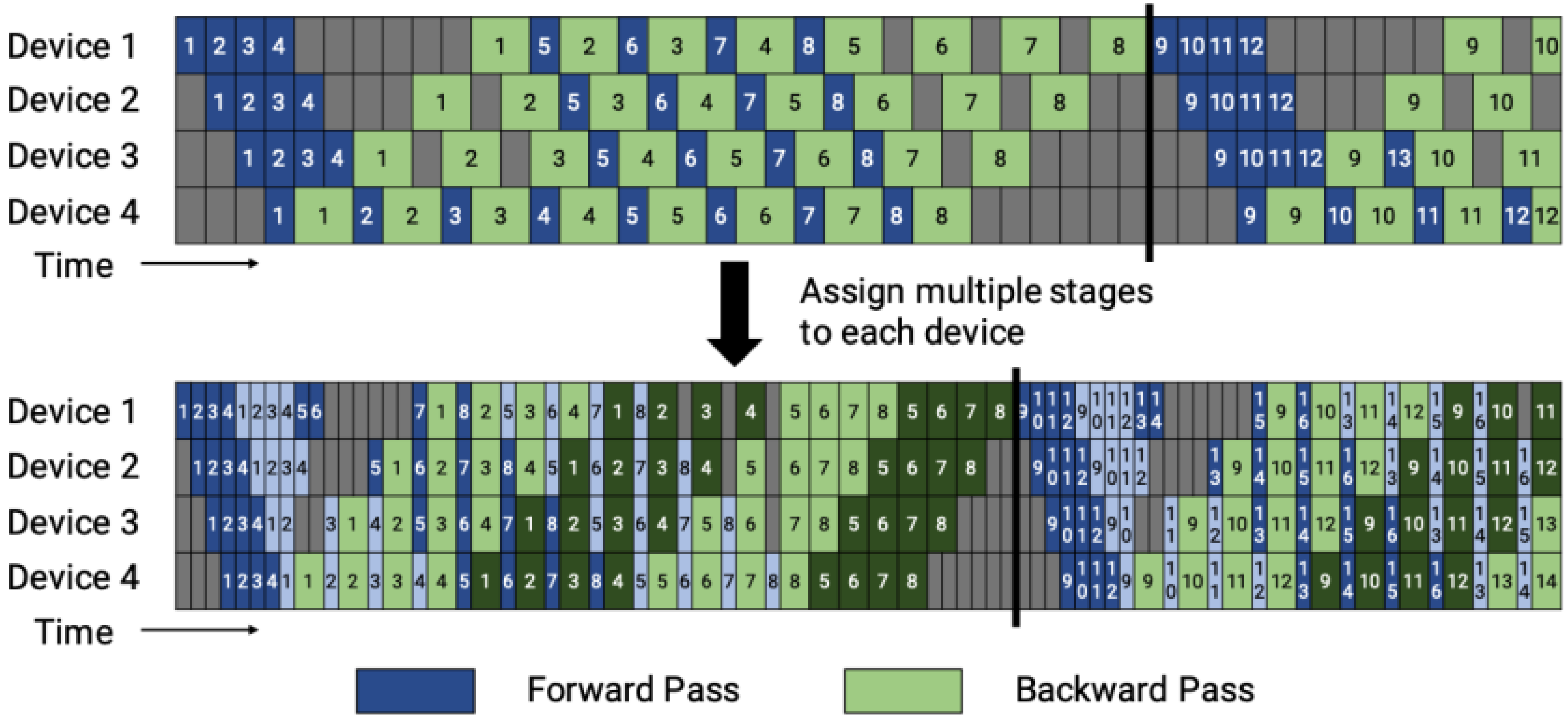
Pipeline Parallelism with 1F1B Schedule

Pipeline parallelism with 1F1B Schedule

$$BubbleFraction = \frac{p-1}{m}$$

Pipeline parallelism with interleaved 1F1B Schedule

$$BubbleFraction = \frac{1}{v} * \frac{p-1}{m}$$



Megatron-v2

Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM

Deepak Narayanan[‡★], Mohammad Shoeybi[†], Jared Casper[†], Patrick LeGresley[†], Mostofa Patwary[†], Vijay Korthikanti[†], Dmitri Vainbrand[†], Prethvi Kashinkunti[†], Julie Bernauer[†], Bryan Catanzaro[†], Amar Phanishayee[*], Matei Zaharia[‡]
[†]*NVIDIA* [‡]*Stanford University* [*]*Microsoft Research*

ABSTRACT

Large language models have led to state-of-the-art accuracies across several tasks. However, training these models efficiently is challenging because: a) GPU memory capacity is limited, making it impossible to fit large models on even a multi-GPU server, and b) the number of compute operations required can result in unrealistically long training times. Consequently, new methods of model parallelism such as tensor and pipeline parallelism have been proposed. Unfortunately, naive usage of these methods leads to scaling issues at thousands of GPUs. In this paper, we show how tensor, pipeline, and data parallelism can be composed to scale to thousands of GPUs. We propose a novel interleaved pipelining schedule that can improve throughput by 10+% with memory footprint comparable to existing approaches. Our approach allows us to perform training iterations on a model with 1 trillion parameters at 502 petaFLOP/s on 3072 GPUs (per-GPU throughput of 52% of theoretical peak).

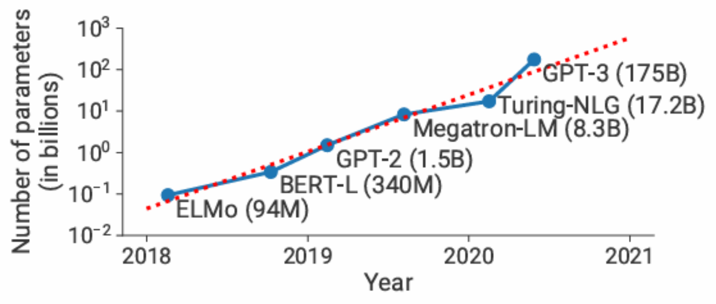


Figure 1: Trend of sizes of state-of-the-art Natural Language Processing (NLP) models with time. The number of floating-point operations to train these models is increasing at an exponential rate.

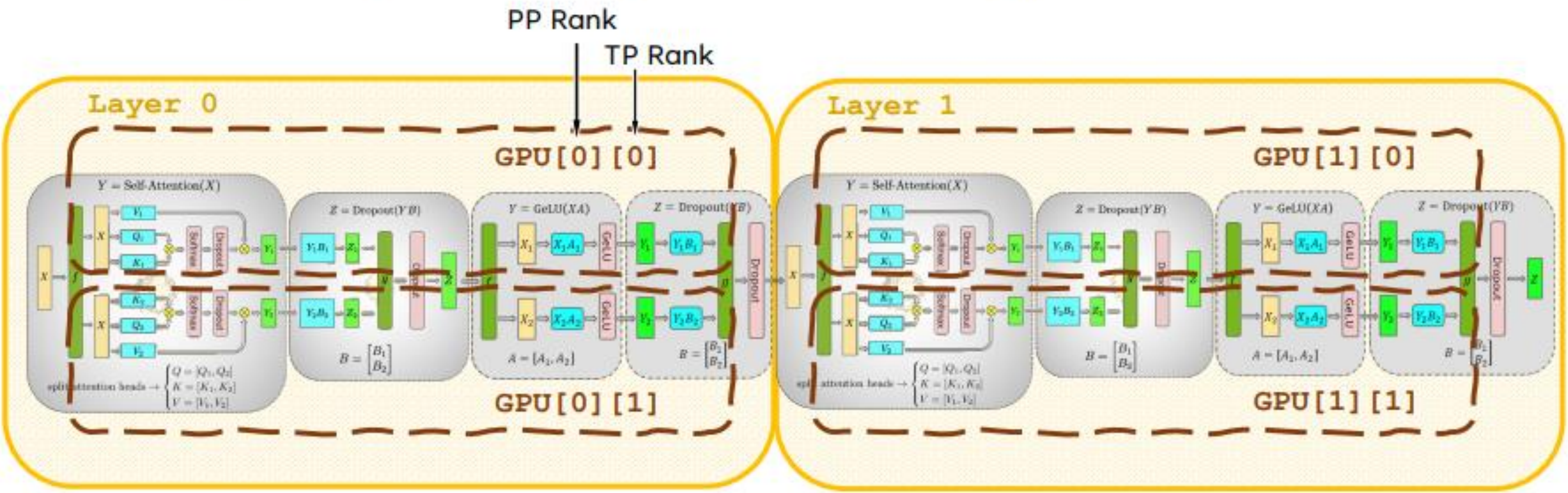
Various model parallelism techniques have been proposed to address these two challenges. For example, recent work [39, 40] has shown how tensor (intra-layer) model parallelism, where matrix multiplications within each transformer layer are split over multiple GPUs, can be used to overcome these limitations. Although this

Today

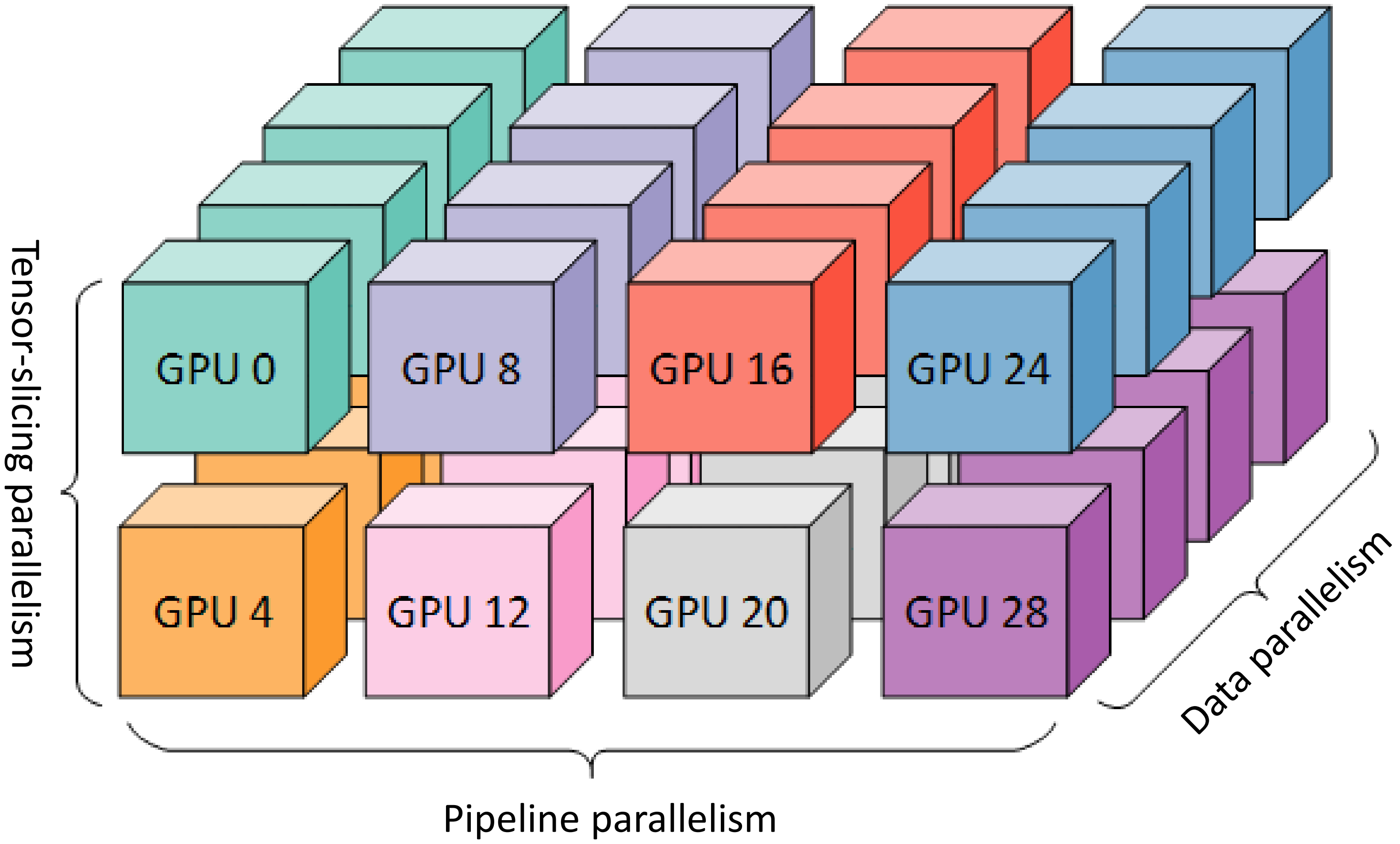
Combining Multiple Parallelism

Combining Model Parallelism

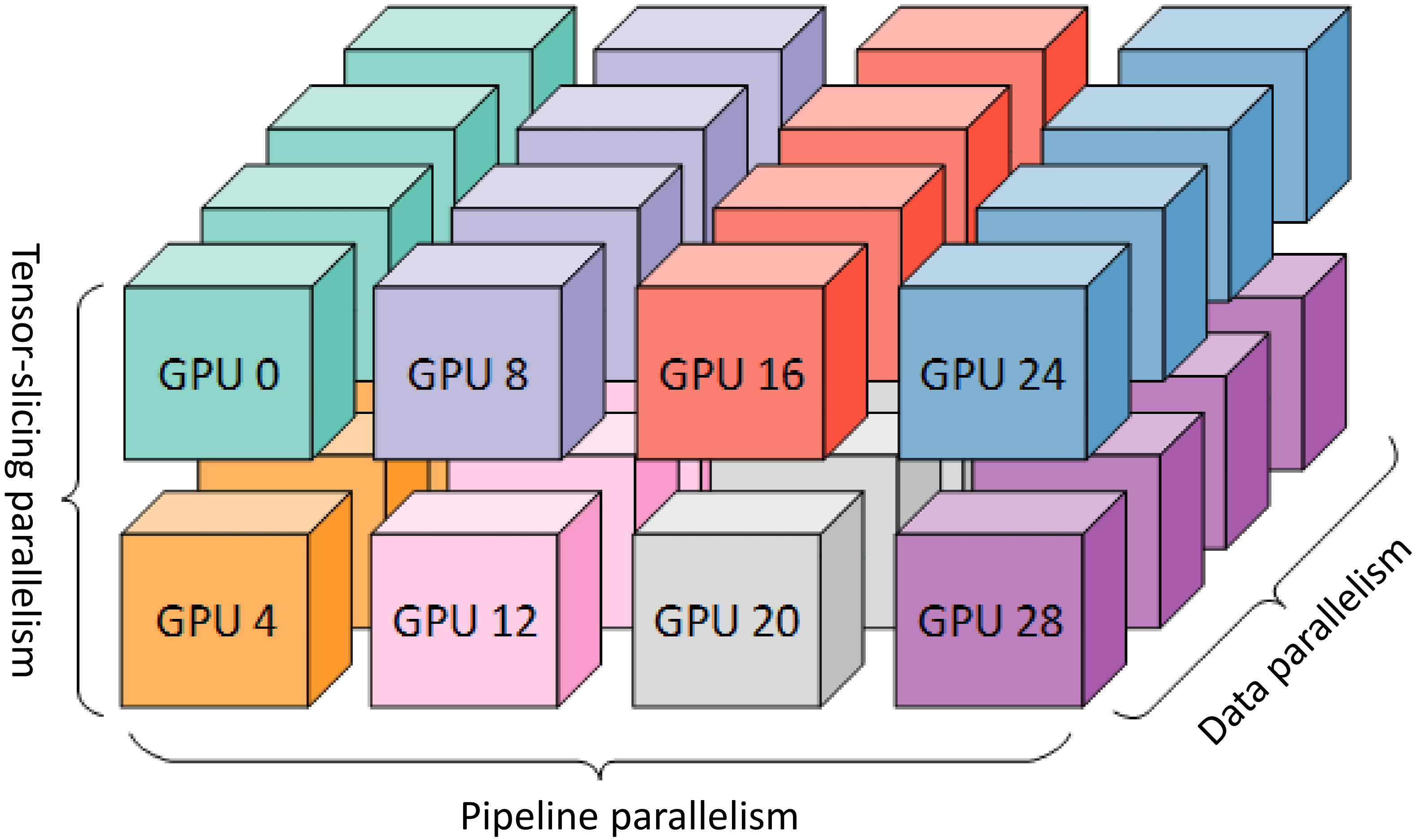
- Model_Parallelism = Tensor_Parallelism × Pipeline_Parallelism



3D Parallelism



3D Parallelism



Question: How do we know which parallelism to choose?

Performance Analysis of Combined Parallelism

- (p, t, d): Parallelization dimensions, where p is the pipeline-model-parallel size, t is the tensor-model-parallel size, and d is the data-parallel size.
- n: Number of GPUs, satisfying $p \cdot t \cdot d = n$.
- B: Global batch size.
- b: Microbatch size.
- $m = \frac{B}{b \cdot d}$: Number of microbatches per pipeline.

Performance Analysis of Combined Parallelism

- Tensor and Pipeline Model Parallelism

$$\frac{p-1}{m}$$ (BubbleFraction)

- (p, t, d): Parallelization dimensions, where p is the pipeline-model-parallel size, t is the tensor-model-parallel size, and d is the data-parallel size.
- n: Number of GPUs, satisfying $p \cdot t \cdot d = n$.
- B: Global batch size.
- b: Microbatch size.
- $m = \frac{B}{b \cdot d}$: Number of microbatches per pipeline.

Performance Analysis of Combined Parallelism

- Tensor and Pipeline Model Parallelism

Assume d = 1, n = p * t

$$\frac{p-1}{m} = \frac{n/t-1}{m}$$

- (p, t, d): Parallelization dimensions, where p is the pipeline-model-parallel size, t is the tensor-model-parallel size, and d is the data-parallel size.
- n: Number of GPUs, satisfying $p \cdot t \cdot d = n$.
- B: Global batch size.
- b: Microbatch size.
- $m = \frac{B}{b \cdot d}$: Number of microbatches per pipeline.

Performance Analysis of Combined Parallelism

- Tensor and Pipeline Model Parallelism
  - t ⇧, pipeline bubble ⇩

$$\frac{p-1}{m} = \frac{n/t-1}{m}$$

- (p, t, d): Parallelization dimensions, where p is the pipeline-model-parallel size, t is the tensor-model-parallel size, and d is the data-parallel size.
- n: Number of GPUs, satisfying $p \cdot t \cdot d = n$.
- B: Global batch size.
- b: Microbatch size.
- $m = \frac{B}{b \cdot d}$: Number of microbatches per pipeline.

Performance Analysis of Combined Parallelism

- Tensor and Pipeline Model Parallelism
  - t ⇧, pipeline bubble ⇩

$$\frac{p-1}{m} = \frac{n/t-1}{m}$$

- Communication overhead
  - All-reduce communication for tensor model parallelism is expensive!
  - Especially when cross servers

- (p, t, d): Parallelization dimensions, where p is the pipeline-model-parallel size, t is the tensor-model-parallel size, and d is the data-parallel size.
- n: Number of GPUs, satisfying $p \cdot t \cdot d = n$.
- B: Global batch size.
- b: Microbatch size.
- $m = \frac{B}{b \cdot d}$: Number of microbatches per pipeline.

Performance Analysis of Combined Parallelism

- Tensor and Pipeline Model Parallelism
  - t ⇧, pipeline bubble ⇩

$$\frac{p-1}{m} = \frac{n/t-1}{m}$$

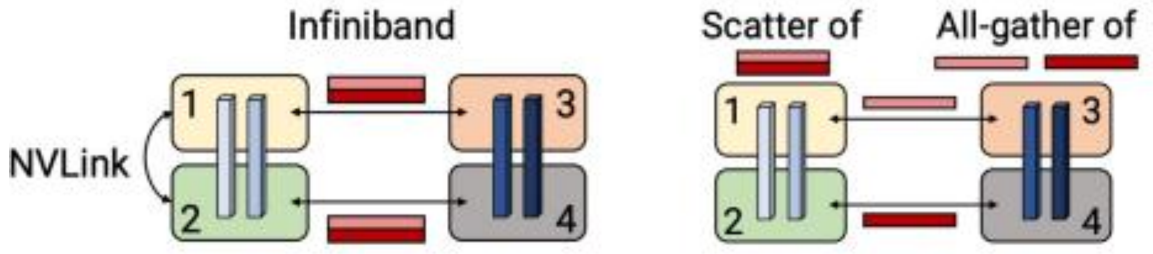
- (p, t, d): Parallelization dimensions, where p is the pipeline-model-parallel size, t is the tensor-model-parallel size, and d is the data-parallel size.
- n: Number of GPUs, satisfying $p \cdot t \cdot d = n$.
- B: Global batch size.
- b: Microbatch size.
- $m = \frac{B}{b \cdot d}$: Number of microbatches per pipeline.

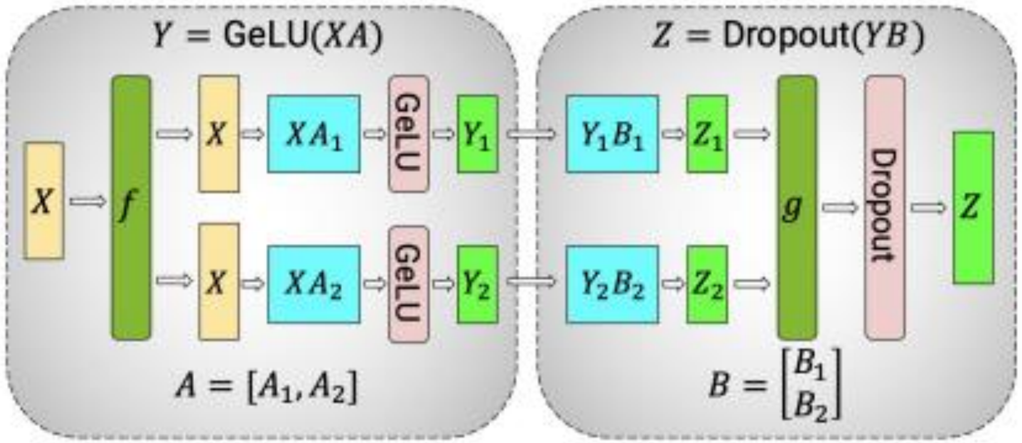
- Communication overhead
  - All-reduce communication for tensor model parallelism is expensive!
  - Especially when cross servers

Takeaway #1: Use tensor model parallelism within a server and pipeline model parallelism to scale to multiple servers.

Scatter/gather Communication Optimization

- Scatter/gather optimization as an extension to the Megatron-LM
    - This reduced pipeline bubble size does not come for free
    - The output of each transformer layer is replicated (after g in MLP block)
    - They are sending and receiving the exact same set of tensors
    - Split the sending message to equal size of chunk and perform an all-gather on receivers

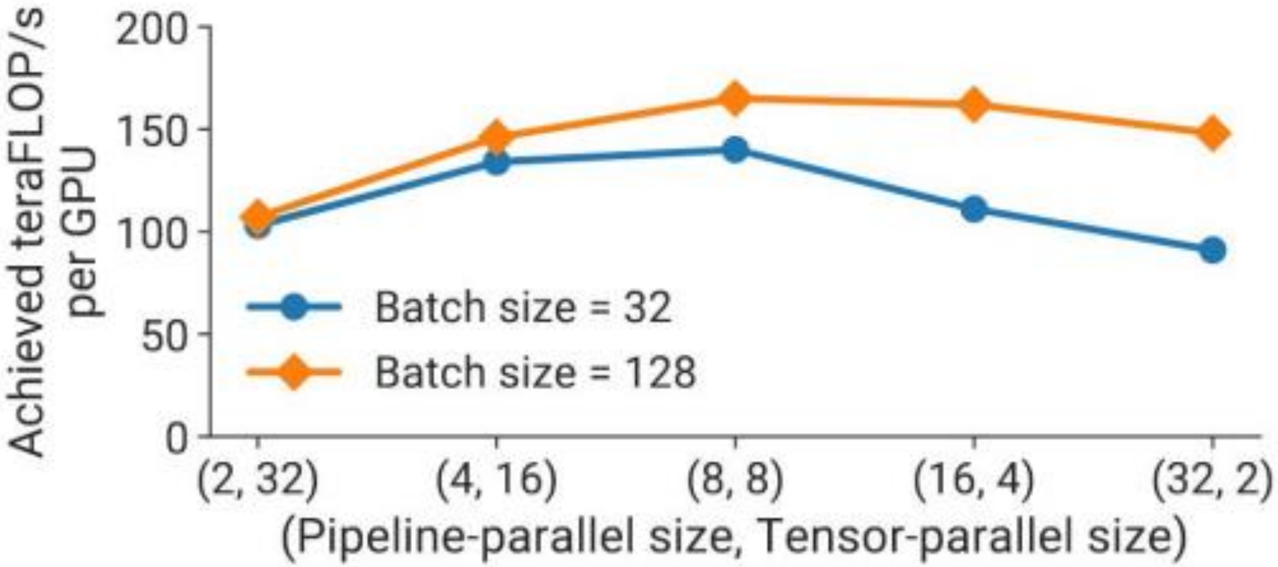




(a) MLP.

Evaluation – TP vs. PP

- Tensor versus Pipeline Parallelism
  - 161-billion param. GPT



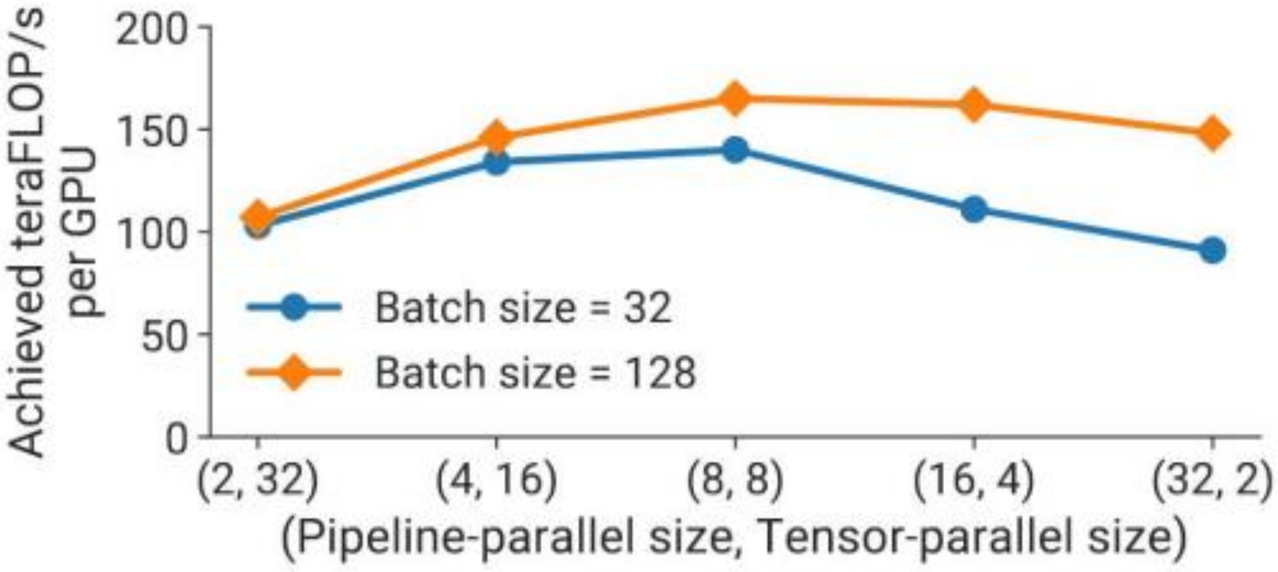
8 Nvidia 80GB A100 cards per node, 8 nodes are connected through fat-tree topology

Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM, Narayanan et al. SC'21

Evaluation – TP vs. PP

- Tensor versus Pipeline Parallelism
  - 161-billion param. GPT

Question: Why does the performance peak at t = p = 8?



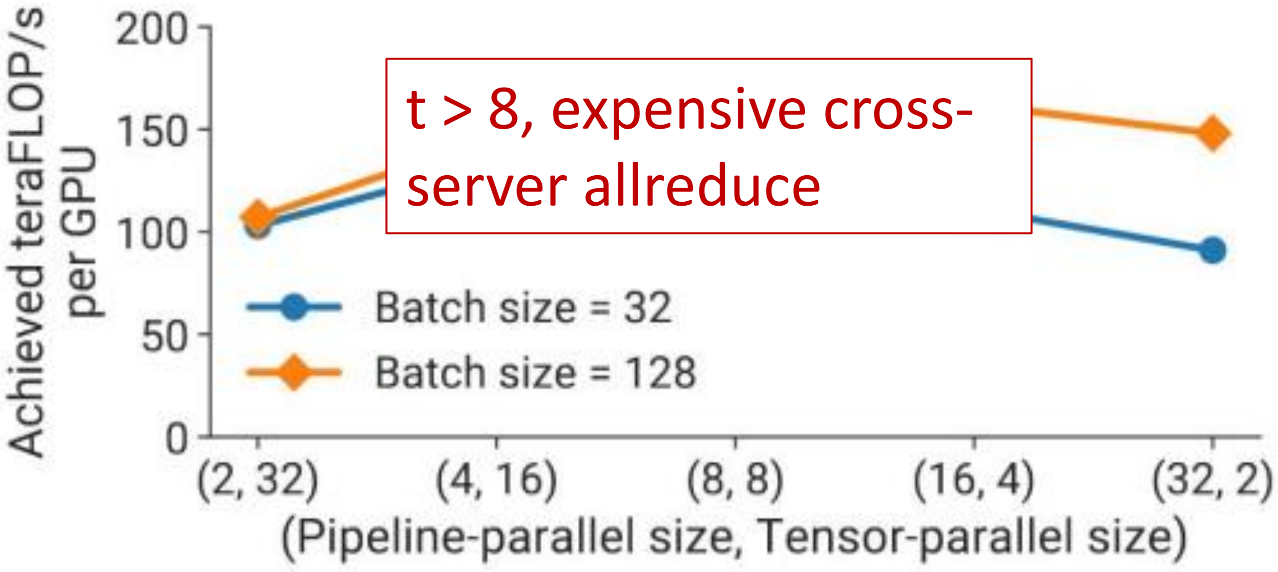
8 Nvidia 80GB A100 cards per node, 8 nodes are connected through fat-tree topology

Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM, Narayanan et al. SC'21

Evaluation – TP vs. PP

- Tensor versus Pipeline Parallelism
  - 161-billion param. GPT

Question: Why does the performance peak at t = p = 8?



t > 8, expensive cross-server allreduce

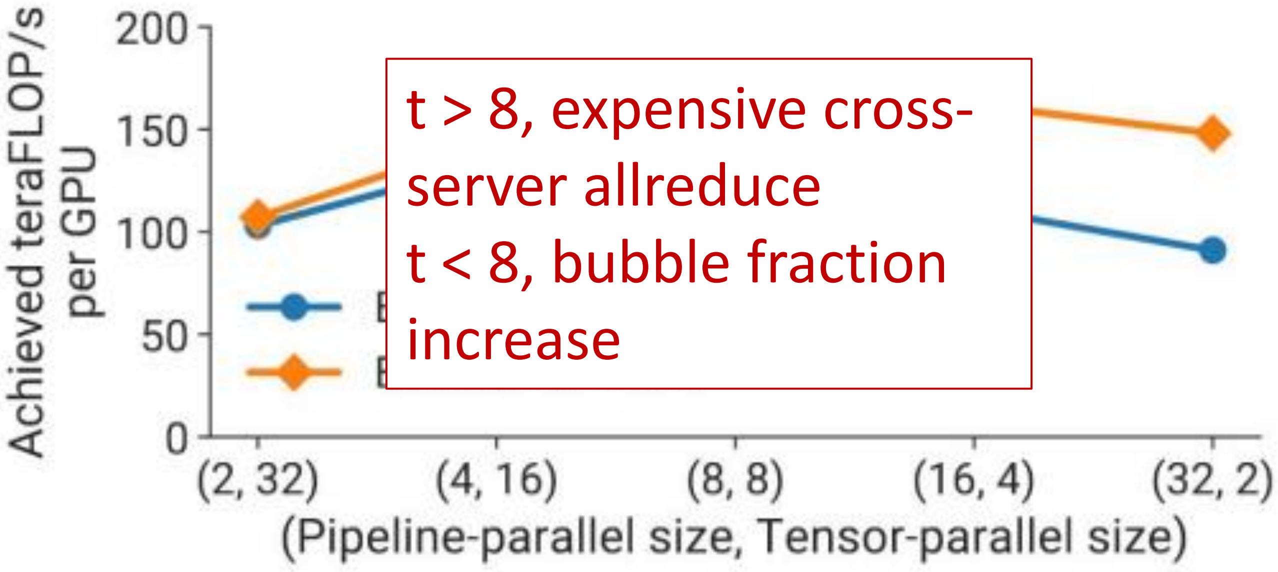
8 Nvidia 80GB A100 cards per node, 8 nodes are connected through fat-tree topology

Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM, Narayanan et al. SC'21

Evaluation – TP vs. PP

- Tensor versus Pipeline Parallelism
  - 161-billion param. GPT

Question: Why does the performance peak at t = p = 8?



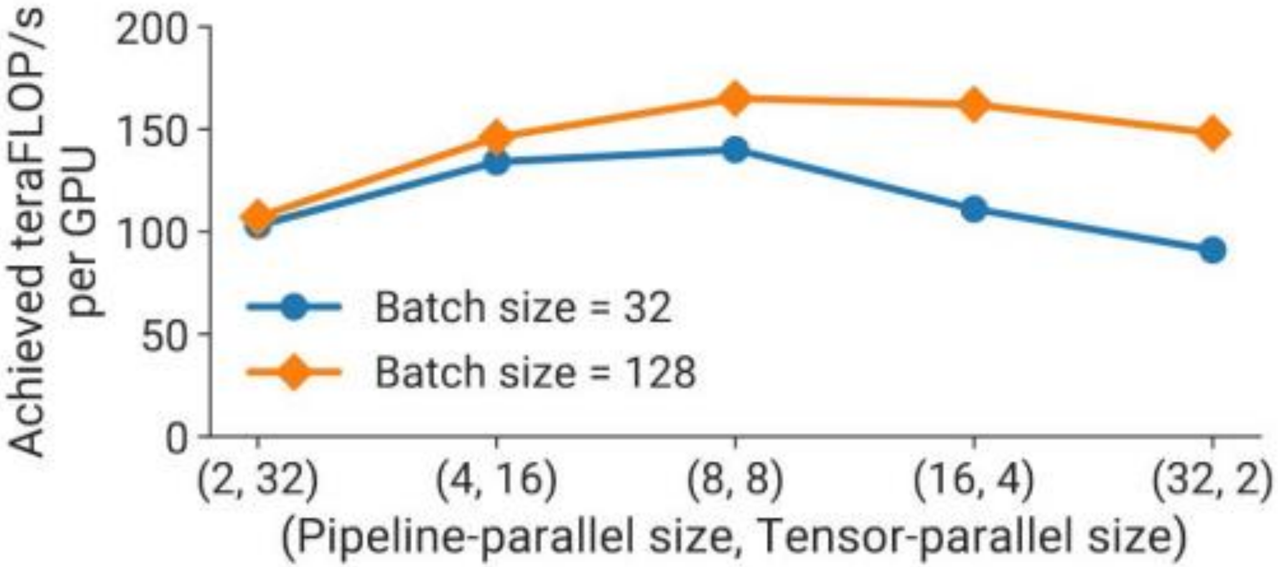
t > 8, expensive cross-server allreduce
t < 8, bubble fraction increase

8 Nvidia 80GB A100 cards per node, 8 nodes are connected through fat-tree topology

Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM, Narayanan et al. SC'21

Evaluation – TP vs. PP

- Tensor versus Pipeline Parallelism
  - 161-billion param. GPT
  - Peak performance achieved when t = p = 8
  - Need a conjunction of both types of model parallelisms

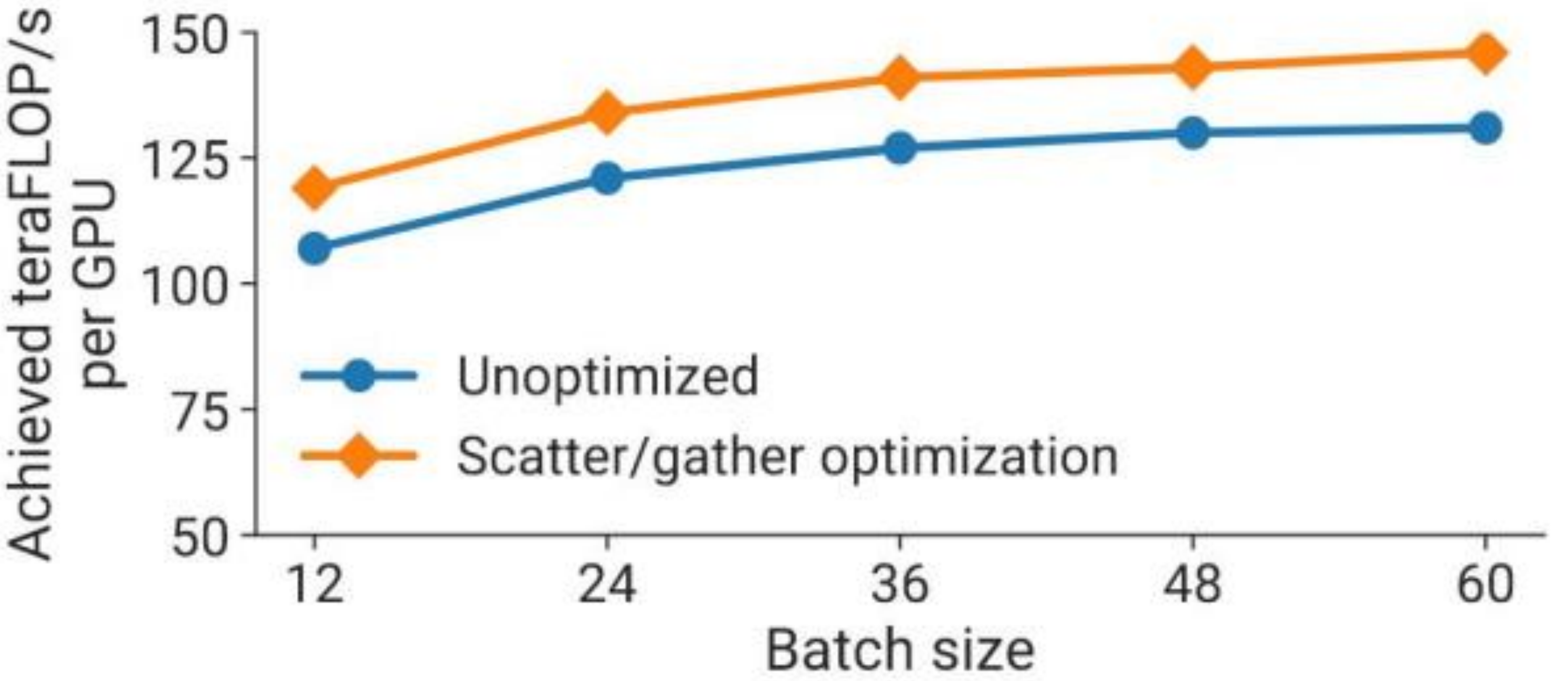


8 Nvidia 80GB A100 cards per node, 8 nodes are connected through fat-tree topology

Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM, Narayanan et al. SC'21

Evaluation - Scatter-gather optimization

- GPT model with 175 billion parameters using 96 A100 GPUs
- Up to 11% in throughput
    - Large batch size with interleaved schedules
    - Reduce cross-node communication cost



Performance Analysis of Combined Parallelism

- Data versus Pipeline Parallelism

$$\frac{p-1}{m}$$

- (p, t, d): Parallelization dimensions, where p is the pipeline-model-parallel size, t is the tensor-model-parallel size, and d is the data-parallel size.
- n: Number of GPUs, satisfying $p \cdot t \cdot d = n$.
- B: Global batch size.
- b: Microbatch size.
- $m = \frac{B}{b \cdot d}$: Number of microbatches per pipeline.

Performance Analysis of Combined Parallelism

- Data versus Pipeline Parallelism

$$\frac{p-1}{m}$$

*Assume t=1, n = d*p*
*m = B / (d * b)*
Assume b' = B/b (ratio of batch size to microbatch size)
m = b' / d

- (p, t, d): Parallelization dimensions, where p is the pipeline-model-parallel size, t is the tensor-model-parallel size, and d is the data-parallel size.
- n: Number of GPUs, satisfying $p \cdot t \cdot d = n$.
- B: Global batch size.
- b: Microbatch size.
- $m = \frac{B}{b \cdot d}$: Number of microbatches per pipeline.

Performance Analysis of Combined Parallelism

- Data versus Pipeline Parallelism

$$\frac{p-1}{m} = \frac{n/d-1}{b'/d}$$

*Assume t=1, n = d*p*
*m = B / (d * b)*
Assume b' = B/b (ratio of batch size to microbatch size)
m = b' / d

- (p, t, d): Parallelization dimensions, where p is the pipeline-model-parallel size, t is the tensor-model-parallel size, and d is the data-parallel size.
- n: Number of GPUs, satisfying $p \cdot t \cdot d = n$.
- B: Global batch size.
- b: Microbatch size.
- $m = \frac{B}{b \cdot d}$: Number of microbatches per pipeline.

Performance Analysis of Combined Parallelism

- Data versus Pipeline Parallelism

$$\frac{p-1}{m} = \frac{n/d-1}{b'/d} = \frac{n-d}{b'}$$ b'=B/b

*Assume t=1, n = d*p*
*m = B / (d * b)*
Assume b’ = B/b (ratio of batch size to microbatch size)
m = b’ / d

- (p, t, d): Parallelization dimensions, where p is the pipeline-model-parallel size, t is the tensor-model-parallel size, and d is the data-parallel size.
- n: Number of GPUs, satisfying $p \cdot t \cdot d = n$.
- B: Global batch size.
- b: Microbatch size.
- $m = \frac{B}{b \cdot d}$: Number of microbatches per pipeline.

Performance Analysis of Combined Parallelism

- Data versus Pipeline Parallelism

$$\frac{p-1}{m} = \frac{n/d-1}{b'/d} = \frac{n-d}{b'} \quad b' = B/b$$

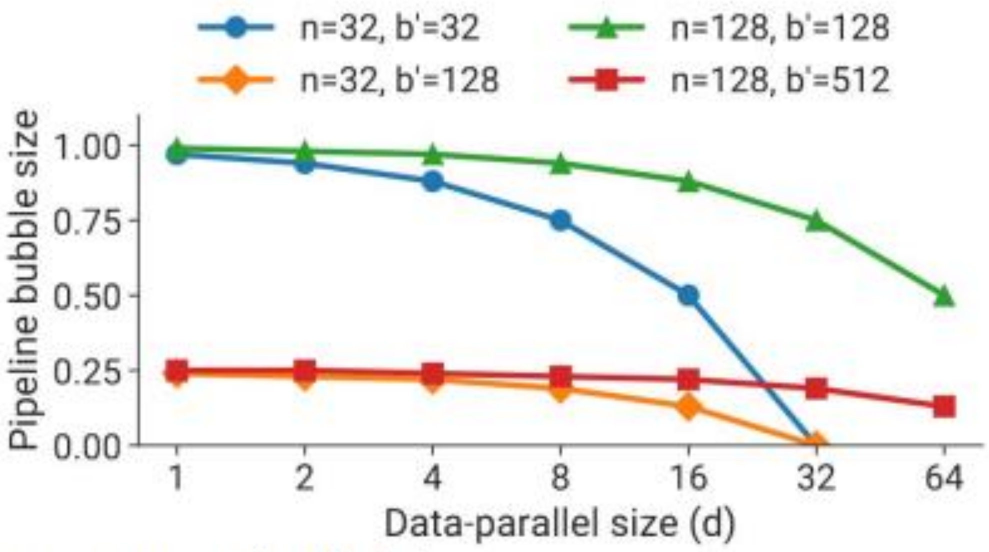
- (p, t, d): Parallelization dimensions, where p is the pipeline-model-parallel size, t is the tensor-model-parallel size, and d is the data-parallel size.
- n: Number of GPUs, satisfying $p \cdot t \cdot d = n$.
- B: Global batch size.
- b: Microbatch size.
- $m = \frac{B}{b \cdot d}$: Number of microbatches per pipeline.

- Data versus Tensor Parallelism
  - DP is less communication heavy than TP
    - All-reduce once per batch vs. All-reduce once per microbatch
  - Tensor parallelism can lead to hardware underutilization

Performance Analysis of Combined Parallelism

- Data versus Pipeline Parallelism

$$\frac{p-1}{m} = \frac{n/d-1}{b'/d} = \frac{n-d}{b'}$$ b'=B/b



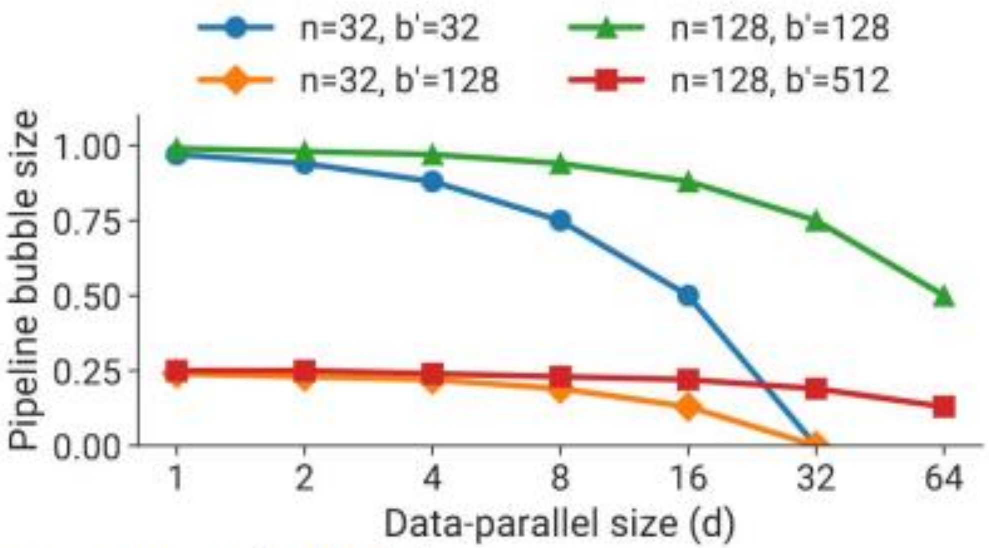
- Data versus Tensor Parallelism
  - DP is less communication heavy than TP
    - All-reduce once per batch vs. All-reduce once per microbatch
  - Tensor parallelism can lead to hardware underutilization

Performance Analysis of Combined Parallelism

- Data versus Pipeline Parallelism

$$\frac{p-1}{m} = \frac{n/d-1}{b'/d} = \frac{n-d}{b'}$$

b'=B/b



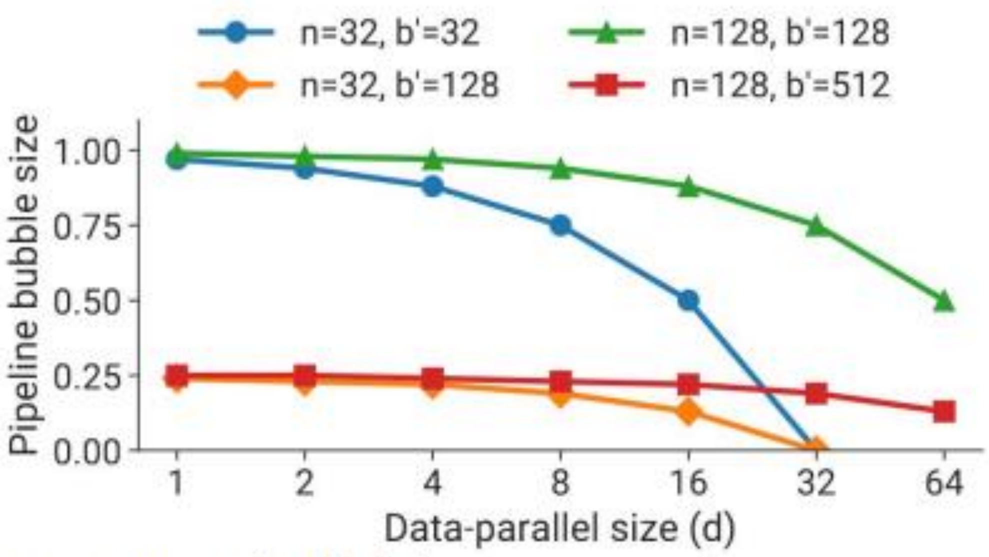
- Data versus Tensor Parallelism
  - DP is less communication heavy than TP
    - All-reduce once per batch vs. All-reduce once per microbatch
  - Tensor parallelism can lead to hardware underutilization

Question: How do *n, b', d* affect the bubble fraction?

Performance Analysis of Combined Parallelism

- Data versus Pipeline Parallelism

$$\frac{p-1}{m} = \frac{n/d-1}{b'/d} = \frac{n-d}{b'} \quad b'=B/b$$

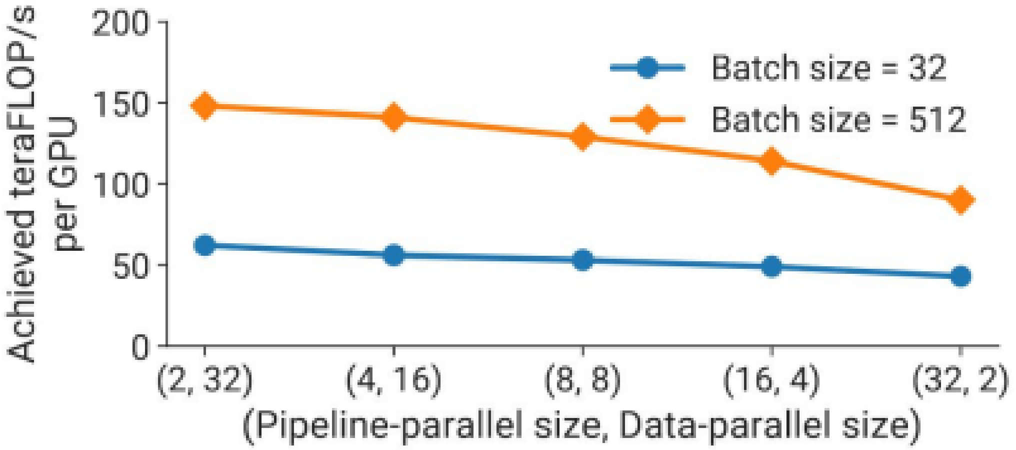


- Data versus Tensor Parallelism
  - DP is less communication heavy than TP
    - All-reduce once per batch vs. All-reduce once per microbatch
  - Tensor parallelism can lead to hardware underutilization

Takeaway #2: Decide tensor-parallel size and pipeline-parallel size based on the GPU memory size; data parallelism can be used to scale to more GPUs.

Evaluation - DP vs. Model Parallelism

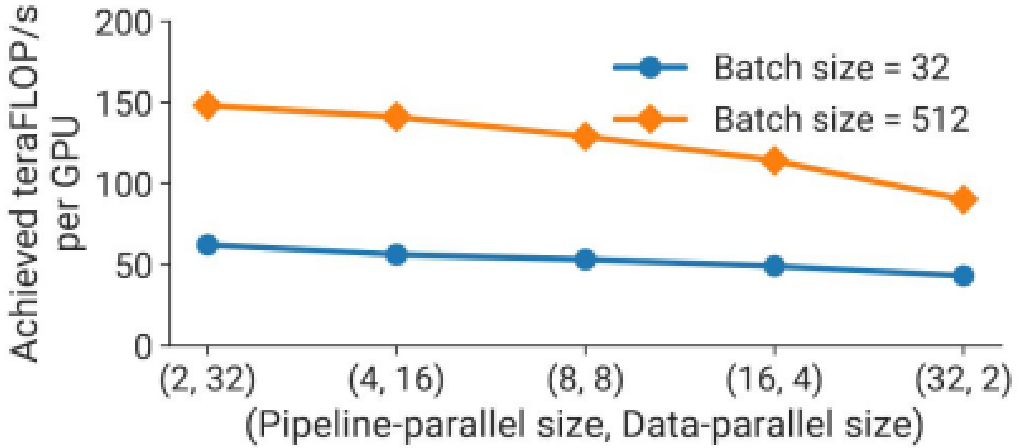
- Pipeline-parallelism vs. Data-parallelism
  - 5.9-billion param. GPT



Question: Why does throughput decrease as pipeline parallel size increase?

Evaluation - DP vs. Model Parallelism

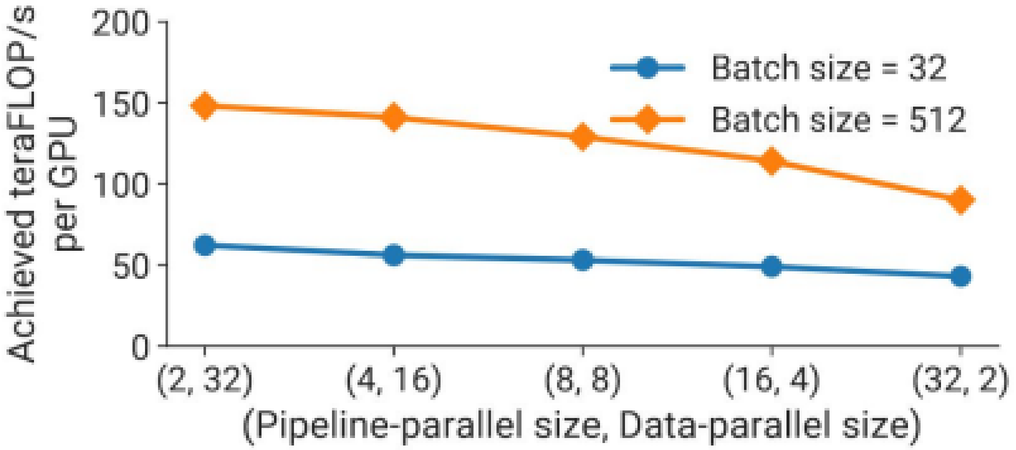
- Pipeline-parallelism vs. Data-parallelism
  - 5.9-billion param. GPT
  - Throughput decreases as pipeline-parallel size increases



$$\frac{p-1}{m} = \frac{n/d-1}{b'/d} = \frac{n-d}{b'} \quad b' = B/b$$

Evaluation - DP vs. Model Parallelism

- Pipeline-parallelism vs. Data-parallelism
  - 5.9-billion param. GPT
  - Throughput decreases as pipeline-parallel size increases

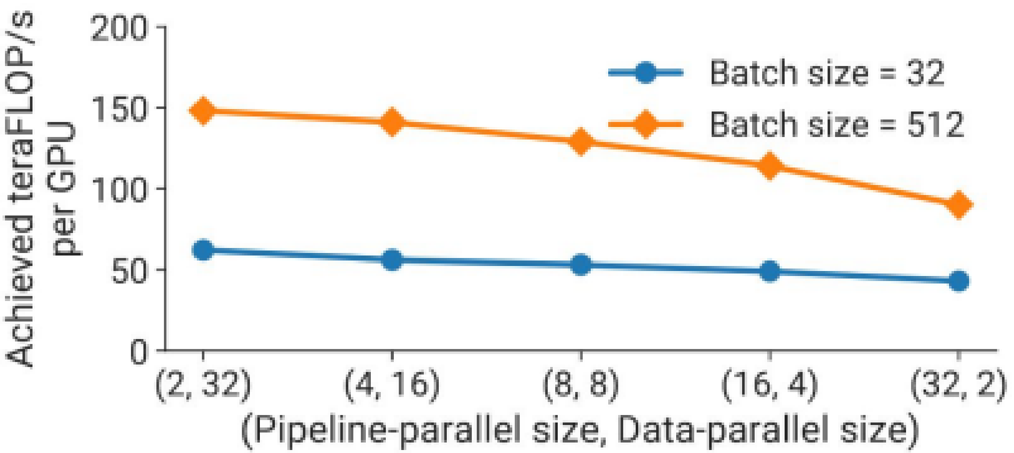


Limitations of data-parallelism:

1. Memory capacity
2. Scaling limitation proportional to the batch size

Evaluation - DP vs. Model Parallelism

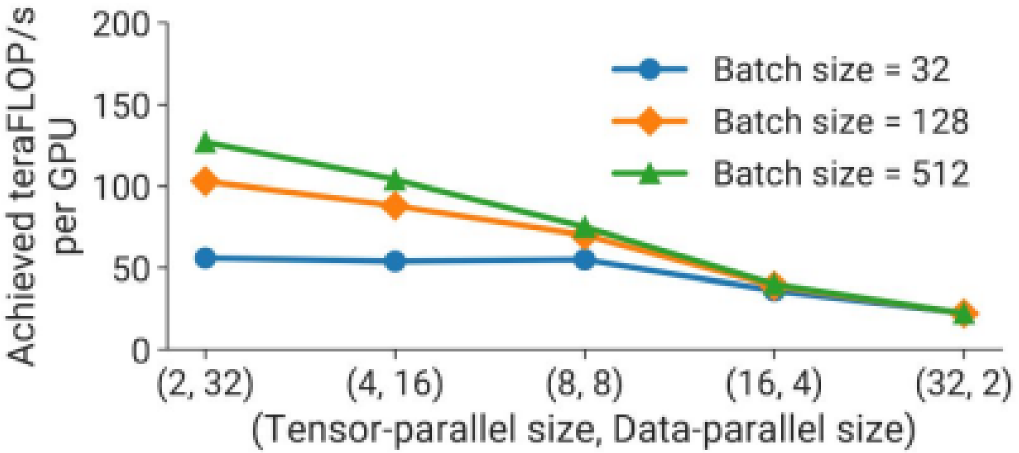
- Pipeline-parallelism vs. Data-parallelism
  - 5.9-billion param. GPT
  - Throughput decreases as pipeline-parallel size increases



Limitations of data-parallelism:

1. Memory capacity
2. Scaling limitation proportional to the batch size

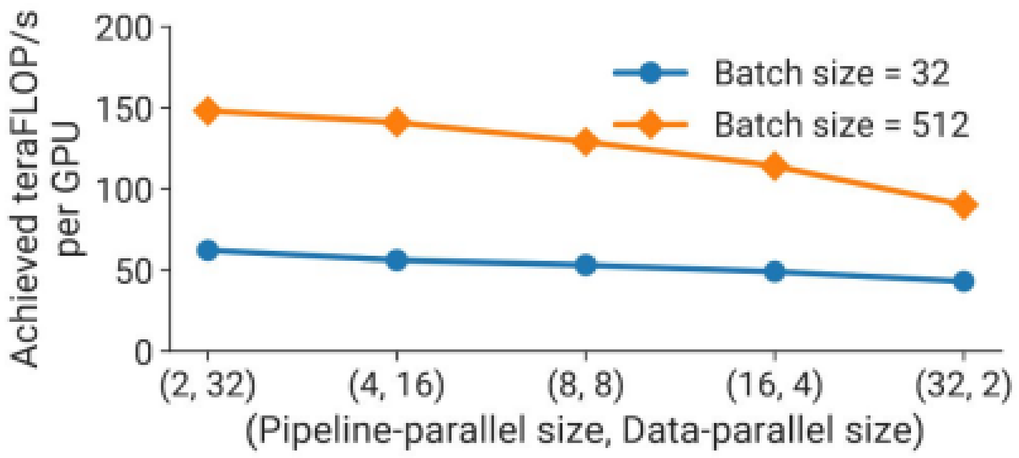
- Tensor-parallelism vs. Data-parallelism
  - 5.9-billion param. GPT



Question: Why does throughput decrease as tensor-parallel size increase?

Evaluation - DP vs. Model Parallelism

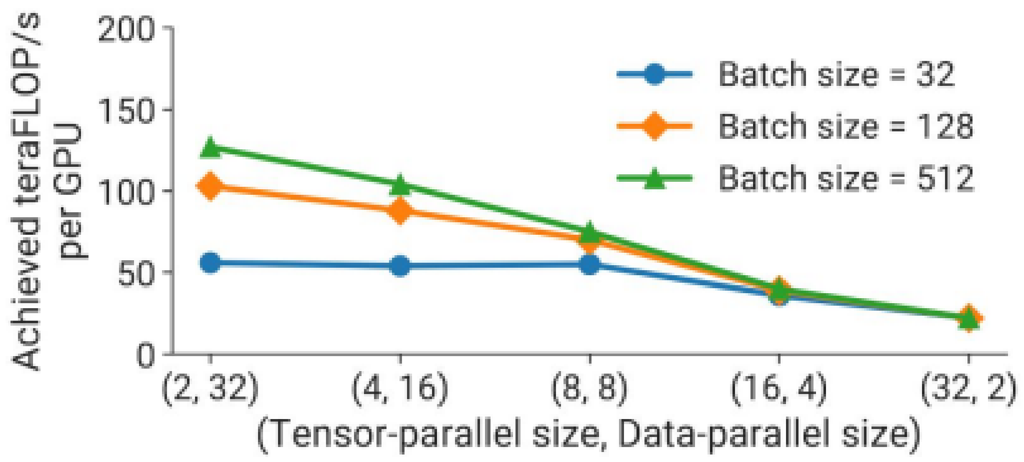
- Pipeline-parallelism vs. Data-parallelism
  - 5.9-billion param. GPT
  - Throughput decreases as pipeline-parallel size increases



Limitations of data-parallelism:

1. Memory capacity
2. Scaling limitation proportional to the batch size

- Tensor-parallelism vs. Data-parallelism
  - 5.9-billion param. GPT
  - Throughput decreases as tensor-parallel size increases

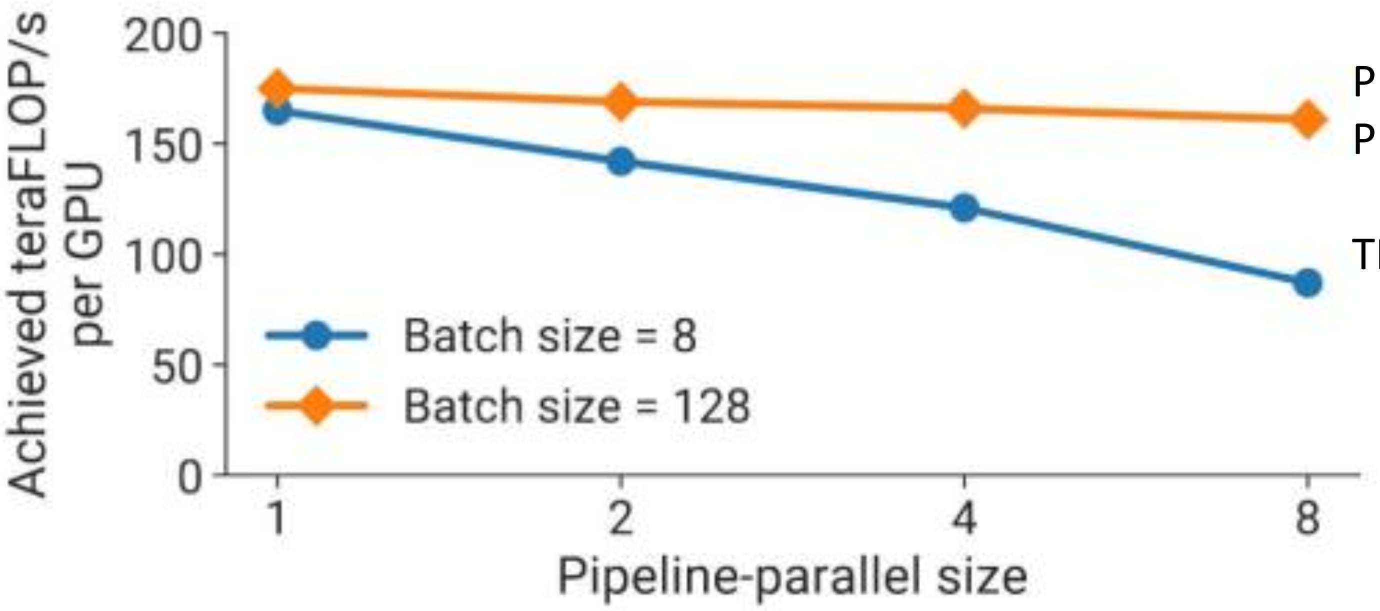


Limitations of tensor-parallelism:

1. More frequent Allreduce
2. Allreduce is on critical path

Evaluation - Pipeline Parallelism (Non-Interleaved)

- Weak Scaling - increase the #layers while increasing PP size



GPT-3 style:
  #heads: 128
  hidden_dim: 20480
  micro-batchsize: 1

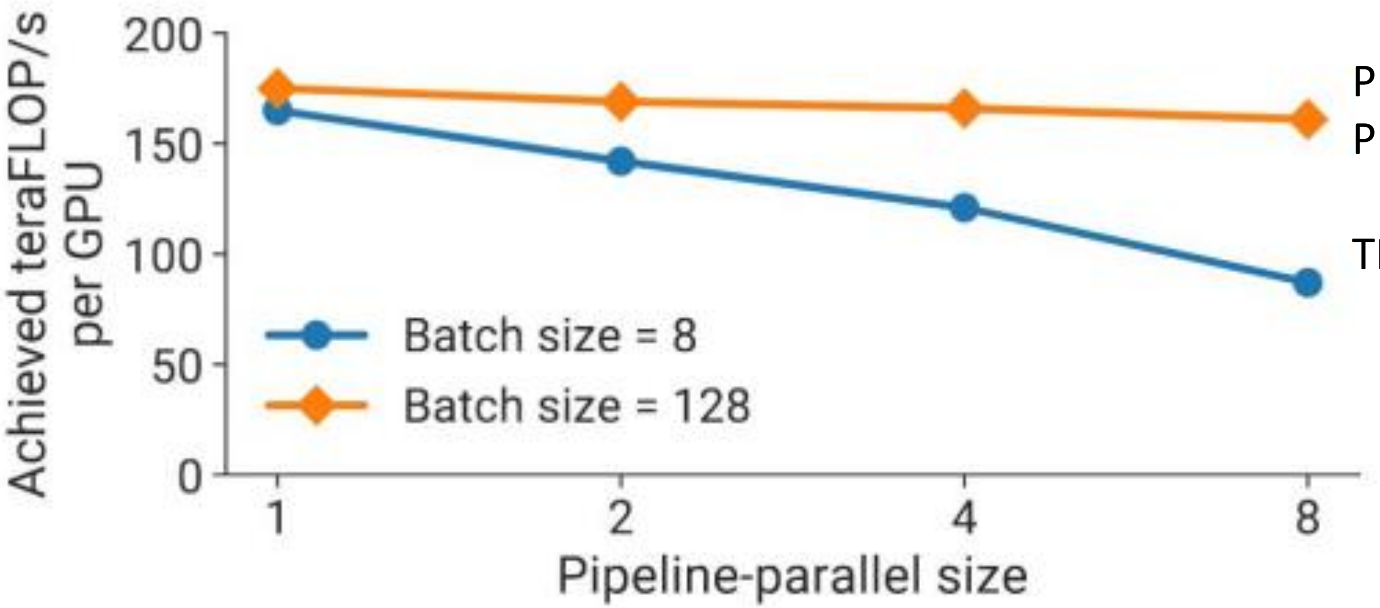
PP = 1, 3-layer Transformer 15B
PP = 8, 24-layer Transformer 121B

TP= 8 fixed, #GPUs from 8 to 64

Evaluation - Pipeline Parallelism (Non-Interleaved)

- Weak Scaling - increase the #layers while increasing PP size

Question: Why does larger batch size scale better?



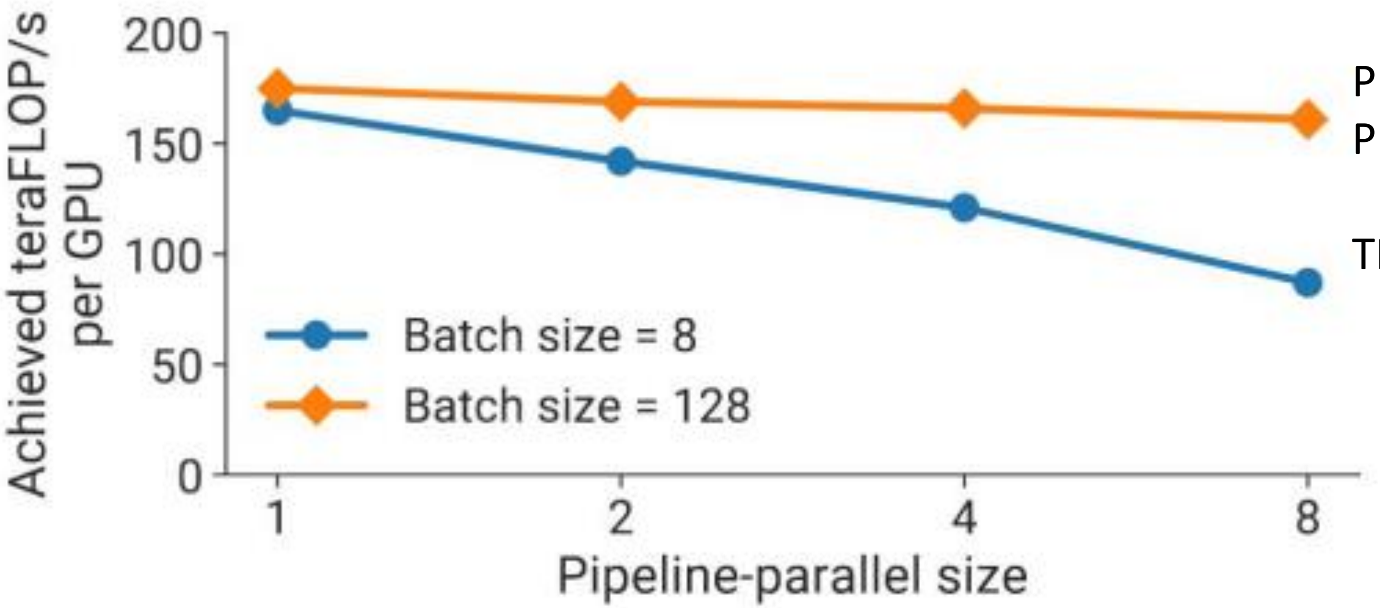
GPT-3 style:
- #heads: 128
- hidden_dim: 20480
- micro-batchsize: 1

PP = 1, 3-layer Transformer 15B
PP = 8, 24-layer Transformer 121B

TP= 8 fixed, #GPUs from 8 to 64

Evaluation - Pipeline Parallelism (Non-Interleaved)

- Weak Scaling - increase the #layers while increasing PP size
- Higher batch size scales better (p-1)/m



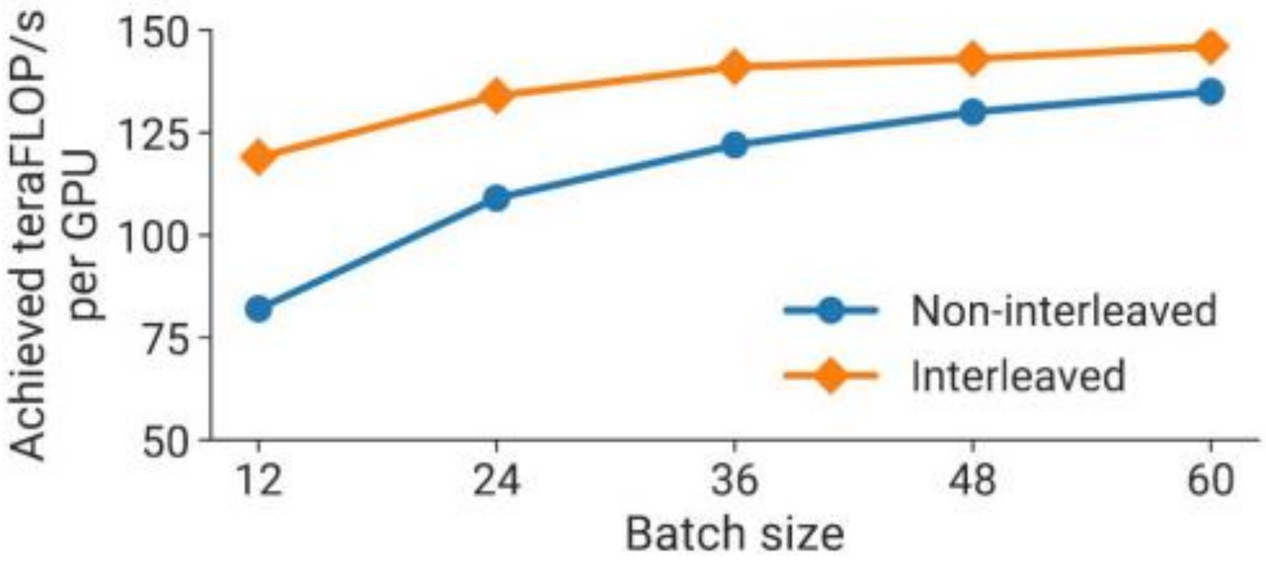
GPT-3 style:
- #heads: 128
- hidden_dim: 20480
- micro-batchsize: 1

PP = 1, 3-layer Transformer 15B
PP = 8, 24-layer Transformer 121B

TP= 8 fixed, #GPUs from 8 to 64

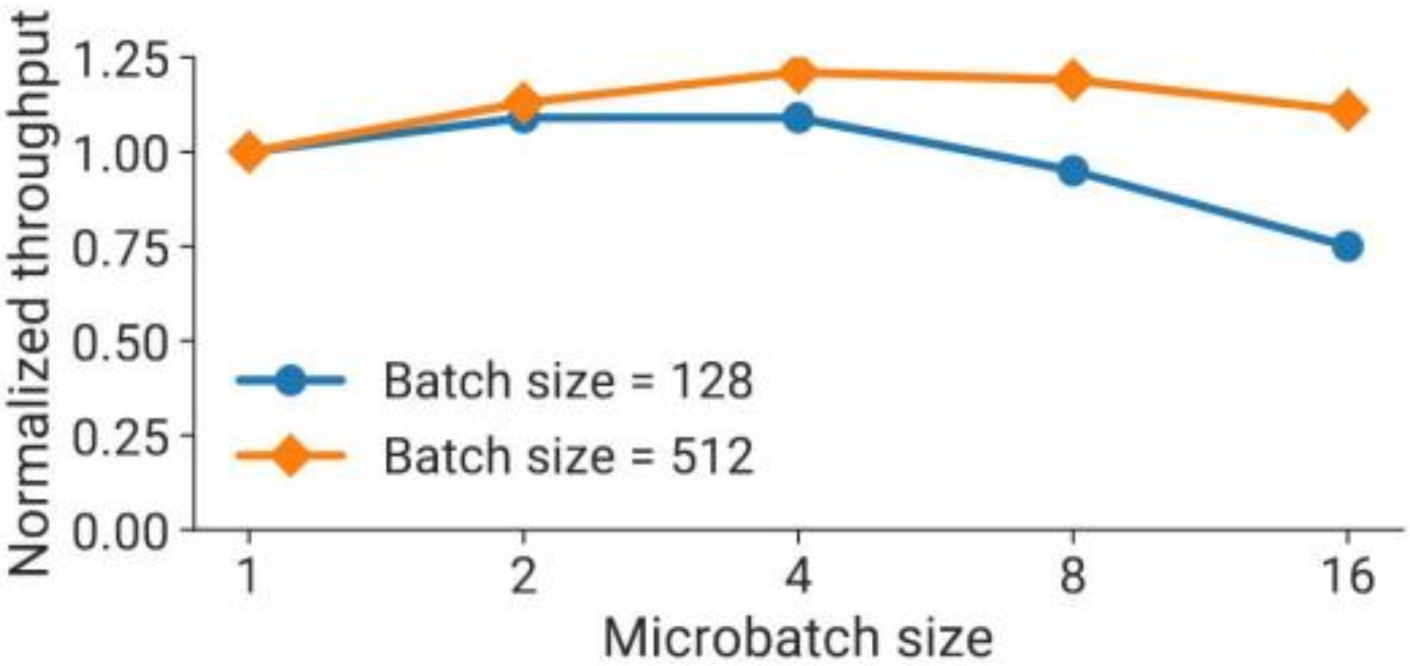
Evaluation – Non-Interleaved vs. Interleaved

- Interleaved schedule with scatter/gather optimization has higher throughput
  - The gap closes as the batch size increases
    - Bubble size decreases when batch size increases (i.e., more micro-batches)
    - Interleaved schedule features more communication cost per sample



Evaluation - Selection of Microbatch size

- Optimal microbatch size is **model dependent**
  - Arithmetic intensity
  - Pipeline bubble size

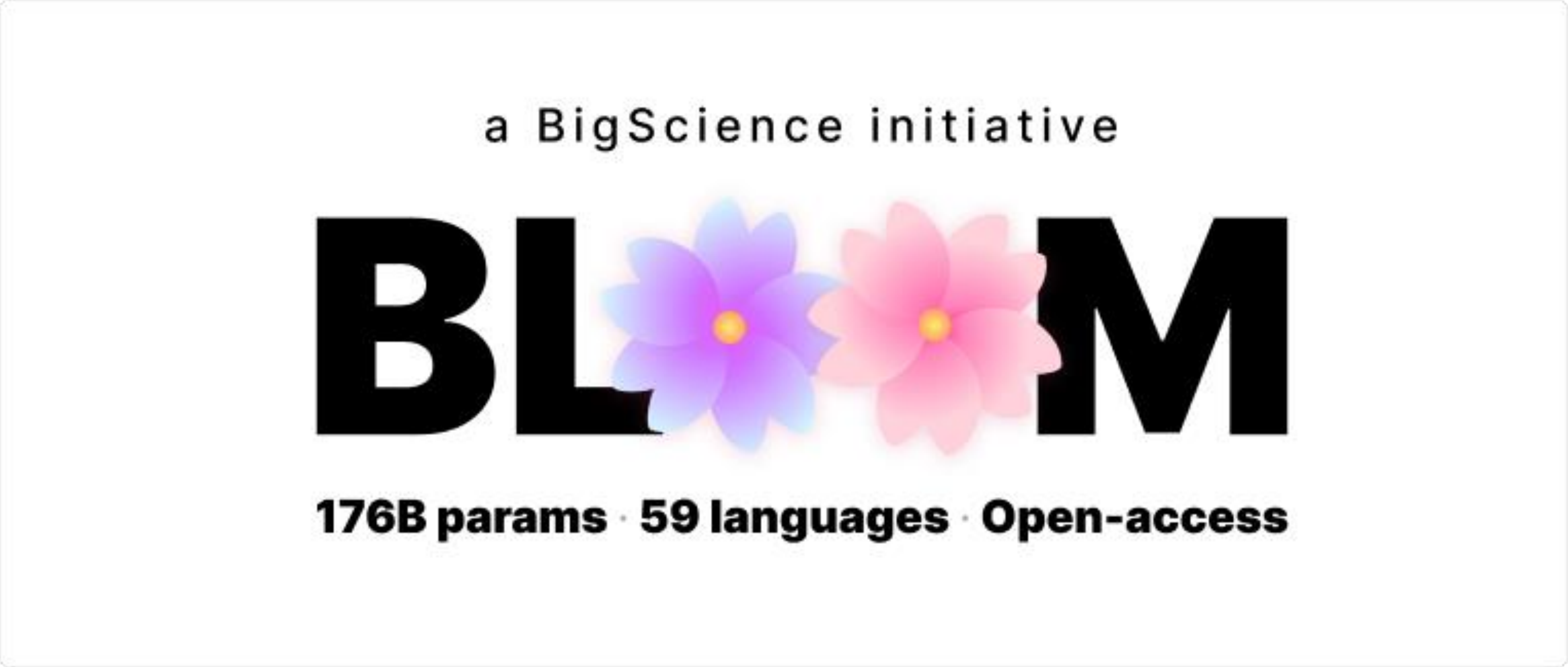


Evaluation - End-to-end Performance

- Superlinear scaling of throughput
    - Per-GPU utilization improves as the model get larger
    - Communication overhead is not significant

Number of parameters (billion)	Attention heads	Hidden size	Number of layers	Tensor model-parallel size	Pipeline model-parallel size	Number of GPUs	Batch size	Achieved teraFlOP/s per GPU	Percentage of theoretical peak FLOP/s	Achieved aggregate petaFLOP/s
1.7	24	2304	24	1	1	32	512	137	44%	4.4
3.6	32	3072	30	2	1	64	512	138	44%	8.8
7.5	32	4096	36	4	1	128	512	142	46%	18.2
18.4	48	6144	40	8	1	256	1024	135	43%	34.6
39.1	64	8192	48	8	2	512	1536	138	44%	70.8
76.1	80	10240	60	8	4	1024	1792	140	45%	143.8
145.6	96	12288	80	8	8	1536	2304	148	47%	227.1
310.1	128	16384	96	8	16	1920	2160	155	50%	297.4
529.6	128	20480	105	8	35	2520	2520	163	52%	410.2
1008.0	160	25600	128	8	64	3072	3072	163	52%	502.0

Models Trained with 3D Parallelism



First open-source project on LLM training through collaboration of AI researchers around the world
Combined multi-dimensional parallelism on Jean Zay cluster in France (estimated cost €3M)

Models Trained with 3D Parallelism

GPUs	TP	CP	PP	DP	Seq. Len.	Batch size/DP	Tokens/Batch	TFLOPs/GPU	BF16 MFU
8,192	8	1	16	64	8,192	32	16M	430	43%
16,384	8	1	16	128	8,192	16	16M	400	41%
16,384	8	16	16	8	131,072	16	16M	380	38%

Table 4 Scaling configurations and MFU for each stage of Llama 3 405B pre-training. See text and Figure 5 for descriptions of each type of parallelism.

Questions?

Evaluation - End-to-end Performance

- Estimated Training Time
  - T: number of tokens
  - P: number of parameters
  - n: number of GPUs
  - X: throughput
  - E.g. GPT3

$$\text{End-to-end training time} \approx \frac{8TP}{nX}$$

T (billion)	P (billion)	n	X (teraFLOPs/s per GPU)	#Days
300	175	1024	140	34
1000	450	3072	163	84

288 years with a single V100 NVIDIA GPU