FlexTrain: Scalable Hybrid-Parallel Training with Elastic Resource Utilization and Consistent Accuracy

Weilin Cai*, Diandian Gu*, Baoquan Zhong*, Jun Wang*, Zhuolin Zheng*, Gaohong Liu, Kaihua Jiang, Shuguang Wang, Wencong Xiao, Jiayi Huang+

HKUST (GZ) | ByteDance Seed

May 20, 2026

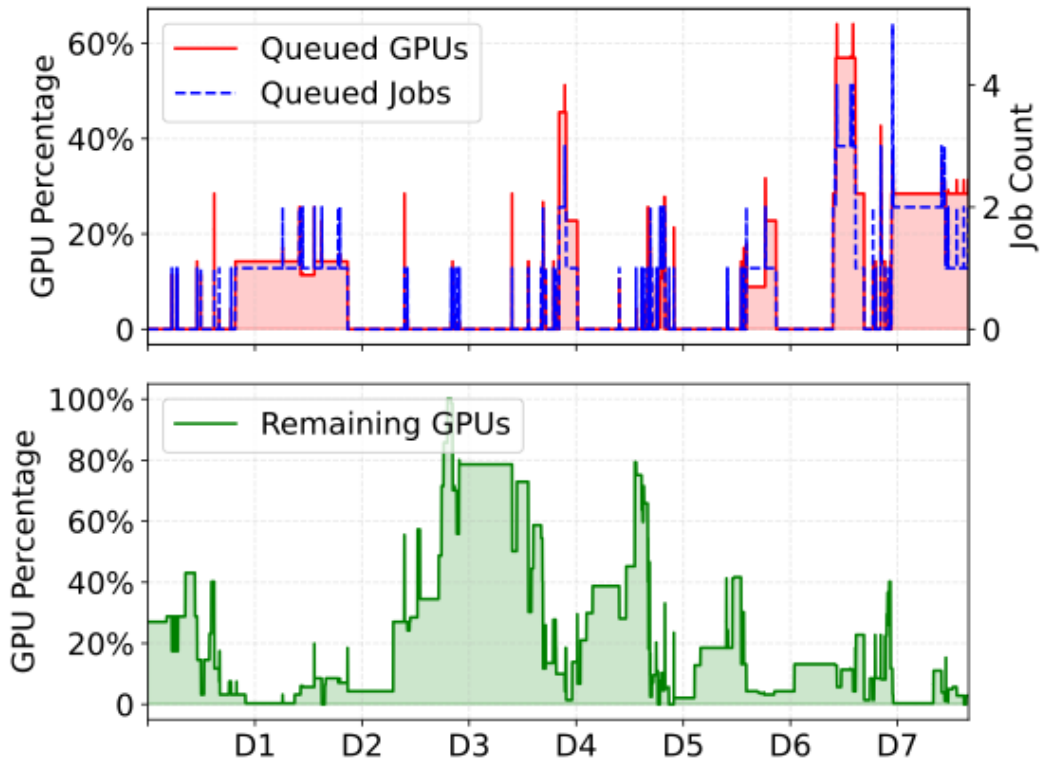
香港科技大學(廣州)
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

字节跳动 ByteDance

MLSys

Background & Motivation

- LLM models evolve rapidly
- GPU requirements for LLM training increase
- GPU clusters remain underutilized



Can we make use of these idle resources to speed up LLM training?

Elastic Training

Dynamically adjust job's GPUs according to cluster utilization and requirements.

Existing elastic training remain rarely adopted due to three major limitations:

1. Accuracy Inconsistency
2. High Profiling Overhead
3. Inflexibility in Utilizing Tidal Resource

Scenario	Related Work	Schedule Granularity	Consistent Accuracy	No Profiling in Advance	Supported Parallelism
Fault tolerance for one job	TorchElastic (pytorch, 2022)	DP	×	✓	-
	ElasWave (Kang et al., 2025)	DP&PP	△	×	DP/PP/TP/EP
Scheduling multiple jobs	ElasticFlow (Gu et al., 2023)	DP	×	×	DP
	Pollux (Qiao et al., 2021)	DP	×	✓	DP
	Rubick (Zhang et al., 2024)	DP&PP&TP	×	×	DP/PP/TP
	EasyScale (Li et al., 2023)	DP	✓	✓	DP
	FlexTrain	PP (DP&PP)	✓(×)	✓	DP/PP/TP/EP/CP

FlexTrain Design Goals

A practical elastic LLM training system in shared GPU clusters.

✓ Accuracy consistency during elastic scaling

✅ **Scaling PP** – selected elastic foundation

⭕ **Scaling DP** – hard to maintain consistent accuracy

❌ **Scaling TP/CP/EP** – complicated & extra intra-layer communication overhead

FlexTrain Design Goals

A practical elastic LLM training system in shared GPU clusters.

- ✓ Accuracy consistency during elastic scaling
- ✓ High training efficiency with lightweight online profiling
- ✓ High flexibility with arbitrary PP configuration

FlexTrain System Overview

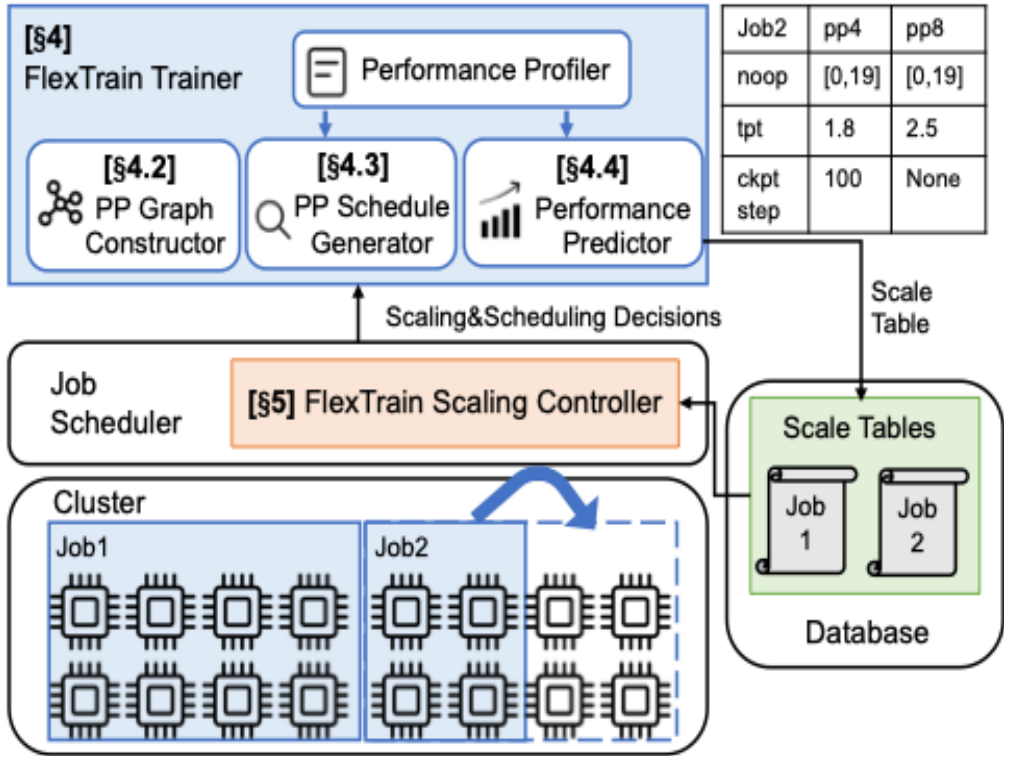
FlexTrain has two main components:

1. FlexTrain Trainer

- Integrated with the training framework
- Constructs PP execution DAGs with online profiling
- Generates memory-safe & throughput-optimal PP schedules
- Performance prediction to generate table for scaling decsion
- Checkpoint resharding and elastic reconfiguration

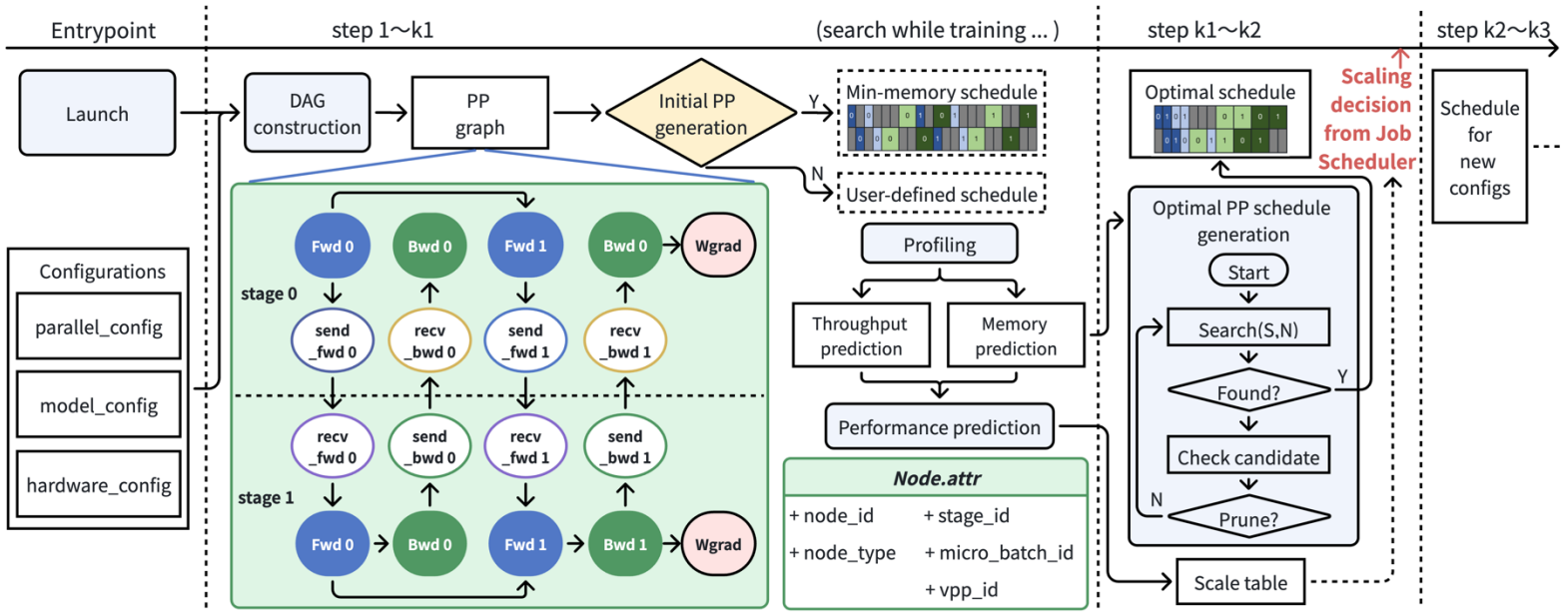
2. FlexTrain Scaling Controller

- Integrated with the cluster job scheduler
- Monitors idle resources
- Maintains scale tables
- Makes Poisson-based scale-up/down decisions
- Minimizes impact on non-elastic jobs



Elastic Training Workflow

1. Launch with user-specified settings.
2. Online profiling upon initial PP schedule to collect time and memory of DAG nodes.
3. Search for better PP schedules under memory constraints.
4. Performance prediction to build scale tables for different GPU counts.
5. Scaling PP or DP+PP degree based on scheduler decisions.



PP Graph Constructor

FlexTrain abstracts the PP exectution of each training iteration as a DAG.

- **Compute Node:**

Forward, Backward, Wgrad

- **Communication Node:**

Send forward, Receive forward, Send backward, Receive backward

- **Edge:**

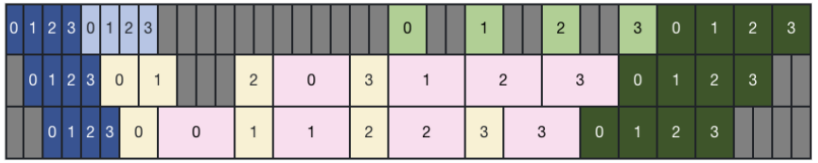
Computation-computation dependency, Computation-communication dependency

Why DAG abstraction matters:
- Allows fine-grained profiling of time and memory.
- Enables schedule search.
- Captures dependencies and avoids deadlocks.
- Provides input to the performance predictor.

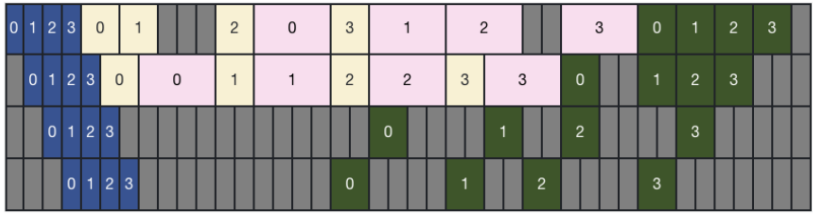
PP Schedule Generator

1. **PP Schedule Search**

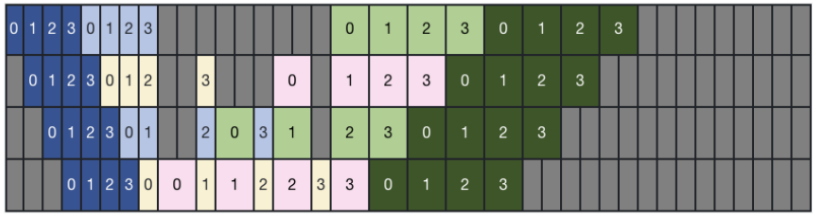
FlexTrain searches for high-throughput PP schedules while satisfying GPU memory constraints.



(a) Pipeline when PP degree is 3.



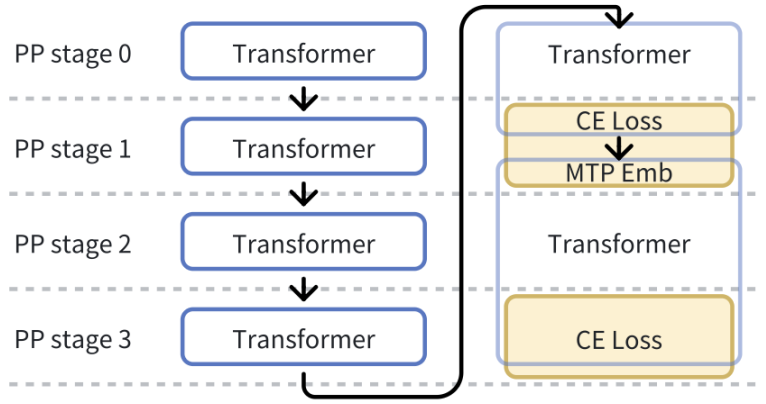
(b) Pipeline when PP degree is 4 (default No-Op insertion).



(c) Pipeline when PP degree is 4 (No-Op optimized for MTP).

2. **No-Op Insertion**

- Arbitrary PP Degree
- Balancing Multi-Token Prediction Workloads



Performance Predictor

FlexTrain predicts the best configuration for each possible GPU count.

- Modeling-based prediction
- Profiling-based prediction

Filtering configurations that:

- violate hyperparameter constraints
- cause OOM
- provide insignificant speedup
- produce inconsistent predictions

Scaling Controller

Using idle GPUs is not always beneficial because scaling has overhead and may hurt incoming jobs.

- **Scale-up decision**

FlexTrain uses Poisson-based benefit prediction that scales up only when the expected benefit is high.

It considers:

- expected idle time window
- scaling ov
- predicted

FlexTrain accelerates elastic jobs while protecting other non-elastic jobs.

- **Scale-down decision**

Triggered by resource contention.

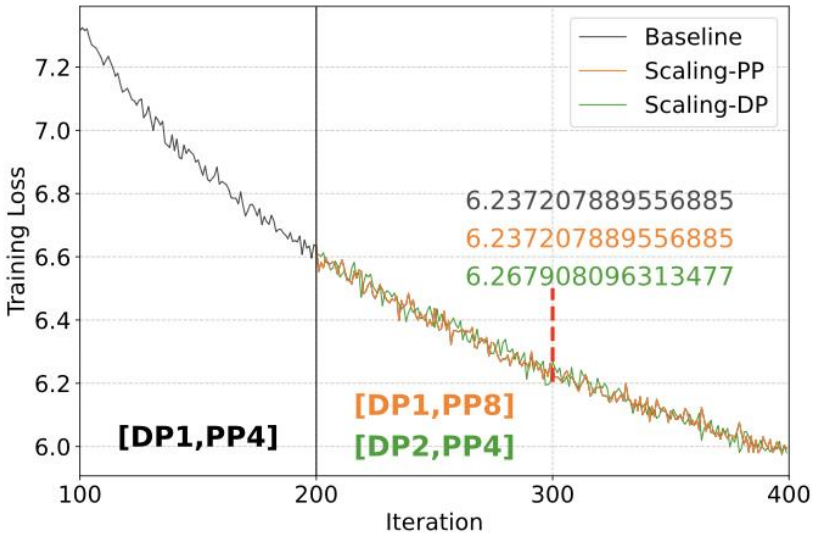
- Preempt elastic resources from jobs.
- Prefer jobs with lower benefit from extra GPUs.
- Support partial scale-down.

Experiment

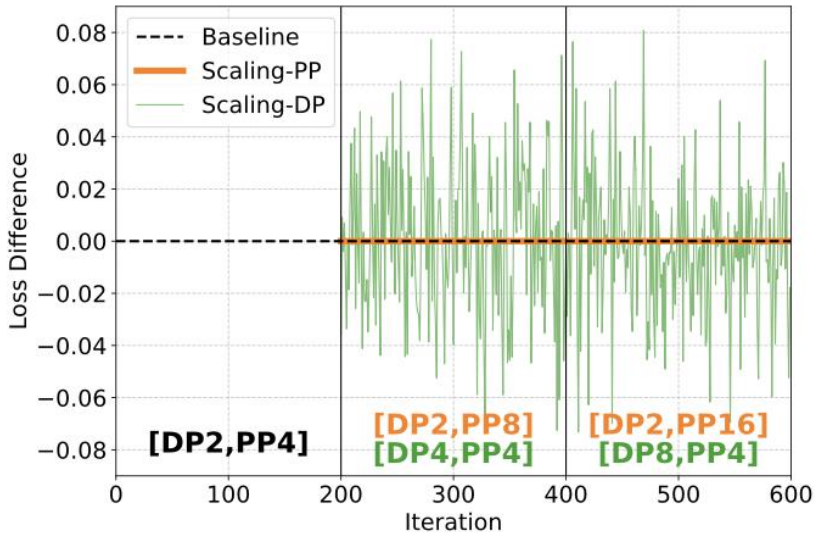
- Real-world tests on three distinct configurations.
- End-to-end scheduling simulation with three-month trace.

Hyperparameter	Small	Medium	Large
Param (Active/Total)	1B/7B	6B/60B	30B/600B
Layer Num	15	29	61
MTP Head Num	2	2	2
Batch Size	1024	256	768
Max Sequence Length	8K	32K	8K
DP/PP/TP/CP/EP Degree	1/4/1/8/8	2/4/1/8/8	4/8/1/8/16
Initial GPU Num	32	64	256
Elastic GPU Num	$32 \sim 128$	$64 \sim 256$	$256 \sim 1024$

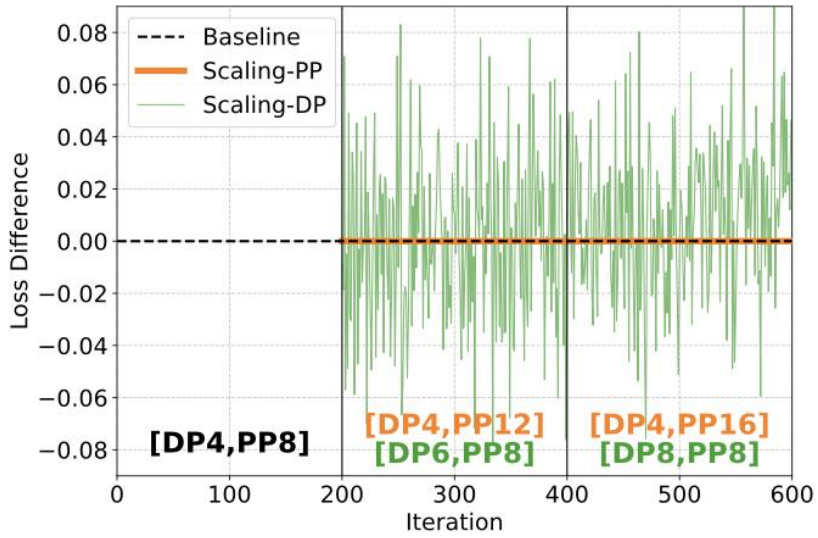
Accuracy Consistency



(a) Small Case



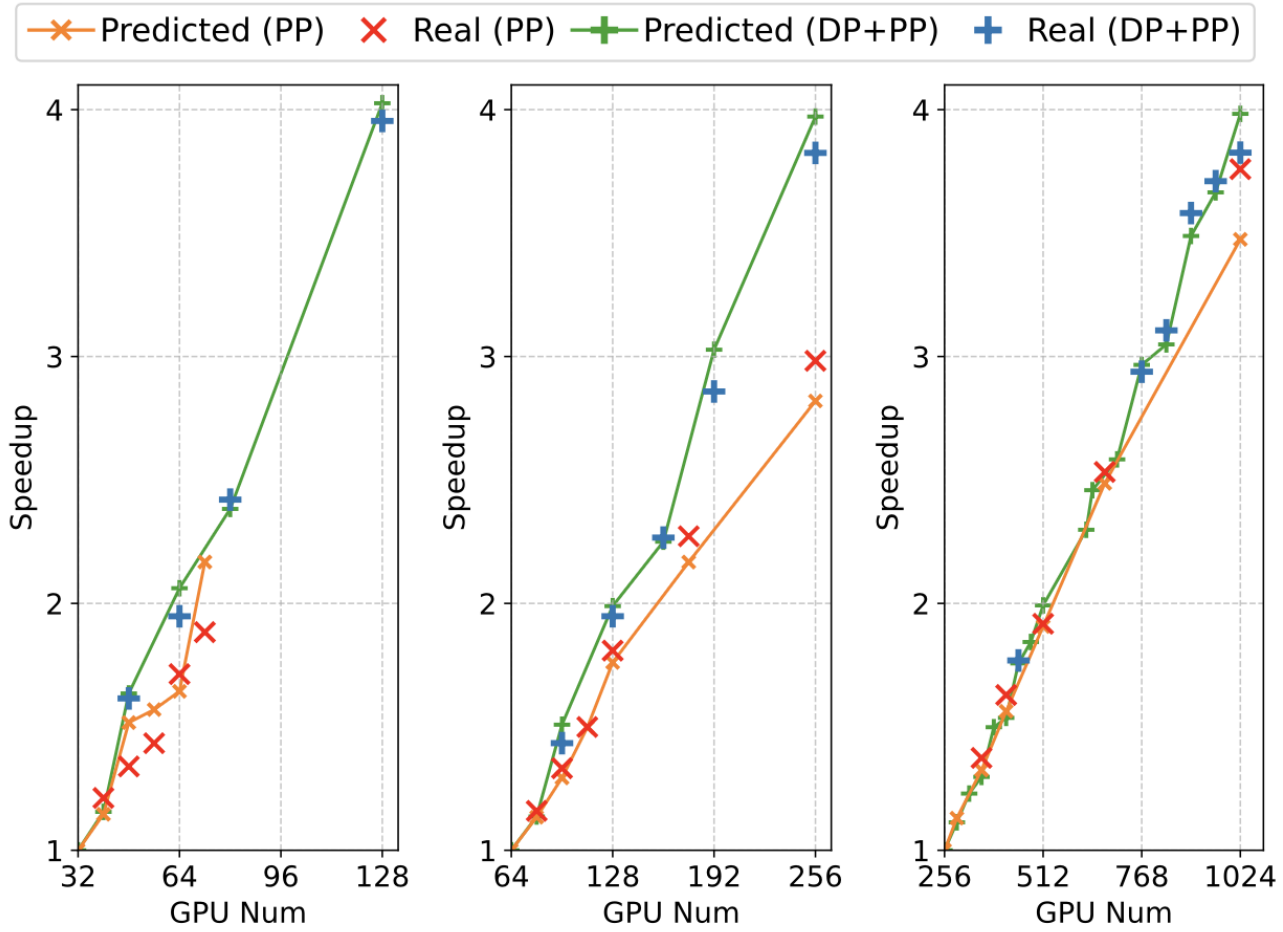
(b) Medium Case



(c) Large Case

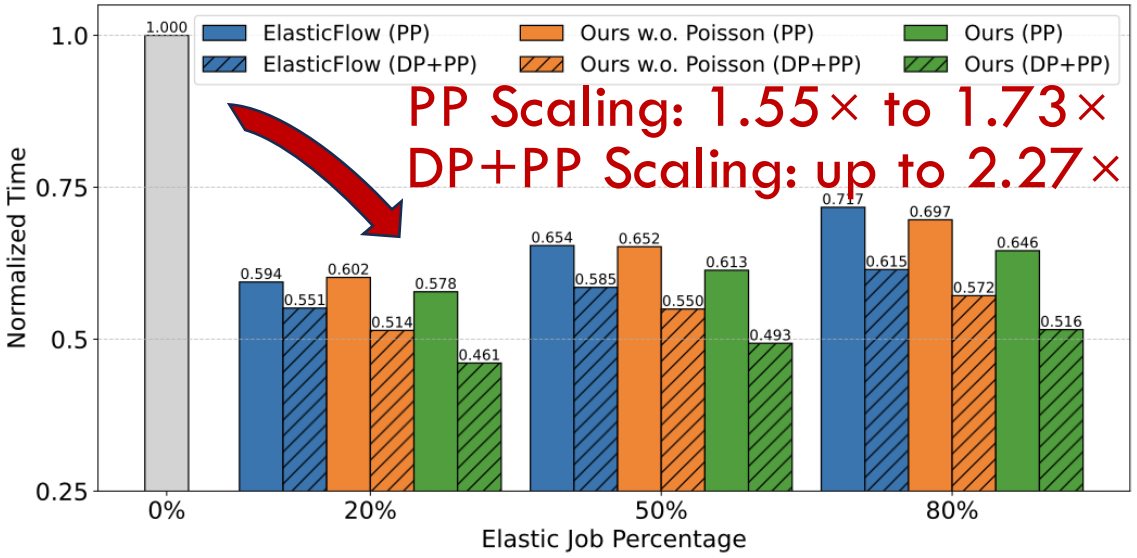
- PP Scaling: Bitwise identical Loss
- DP scaling: Visible loss divergence

Acceleration

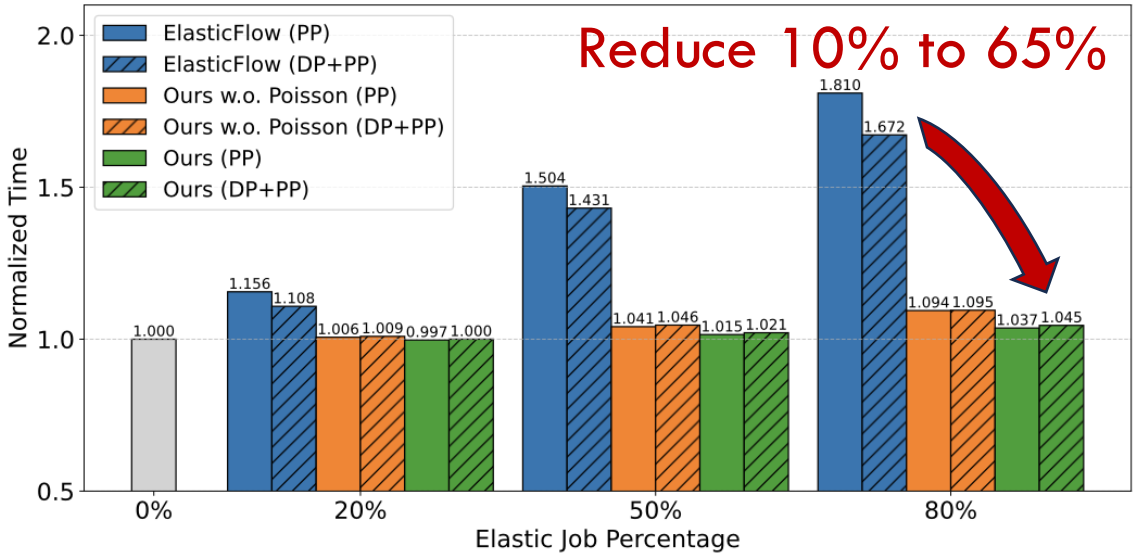


(a) Small Case (b) Medium Case (c) Large Case

End-to-End Scheduling Simulation



(a) Normalized JCT of Elastic Jobs



(b) Normalized Queue Time of Other Non-Elastic Jobs

Conclusion

FlexTrain Makes Elastic LLM Training Production-Ready.

➢ **What FlexTrain solves:**

- Improves GPU utilization by exploiting idle resources.
- Maintains accuracy consistency through PP scaling.
- Avoids harming non-elastic jobs with benefit-aware scheduling.

➢ **Results:**

- Preserves training accuracy under PP scaling.
- 1.55–1.73× JCT reduction with PP scaling.
- 2.27× JCT reduction with DP+PP scaling under relaxed accuracy constraints.
- 50–100% reduction in queue time for non-elastic jobs.

Thank You

Weilin Cai*, Diandian Gu*, Baoquan Zhong*, Jun Wang*, Zhuolin Zheng*, Gaohong Liu,
Kaihua Jiang, Shuguang Wang, Wencong Xiao, Jiayi Huang+

HKUST (GZ) | ByteDance Seed

May 20, 2026

Presenter: Weilin Cai – wcai738@connect.hkust-gz.edu.cn

香港科技大學(廣州)
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

字节跳动 ByteDance

MLSys