SpiderFlow

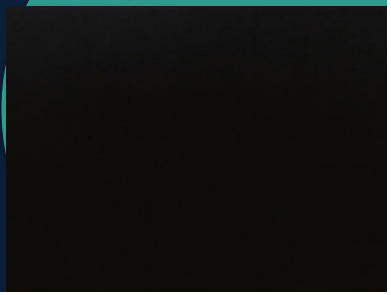
Topology-Aware LLM Training Scheduler for Decentralized GPU Clusters

Efficient Topology-Aware Scheduling for LLM Training Across Decentralized GPU Clusters

Zihan Chang, Shuibing He, Bo Zhou, Sheng Xiao , Siling Yang, Rui Wang, Zhe Pan

Intelligent Storage and Computing Systems (ISCS) Laboratory

Zhejiang University, Hangzhou, China



Outline

Presentation Outline

01

Background & Motivation

Why does decentralized GPU training slow down LLM workloads?

02

Key Observations

How do the number of clusters K and parallelism choices affect communication cost?

03

System Design

How do SpinSearch and TPA jointly optimize scheduling and parallelism automation?

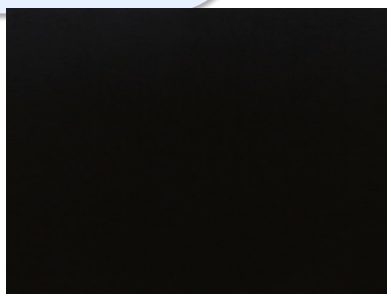
04

Evaluation Results

How much do throughput, JCT, and scheduling overhead improve over existing methods?

One-sentence summary

SpiderFlow formulates cross-cluster training scheduling as a topology-aware graph optimization problem, using low-overhead SpinSearch and two-level Parallel Automation to reduce cross-cluster communication while maintaining high throughput.



Motivation

Background & Core Problem

When a single cluster is insufficient, LLM training must leverage GPUs across geographically or logically distributed clusters.

Rising training demand

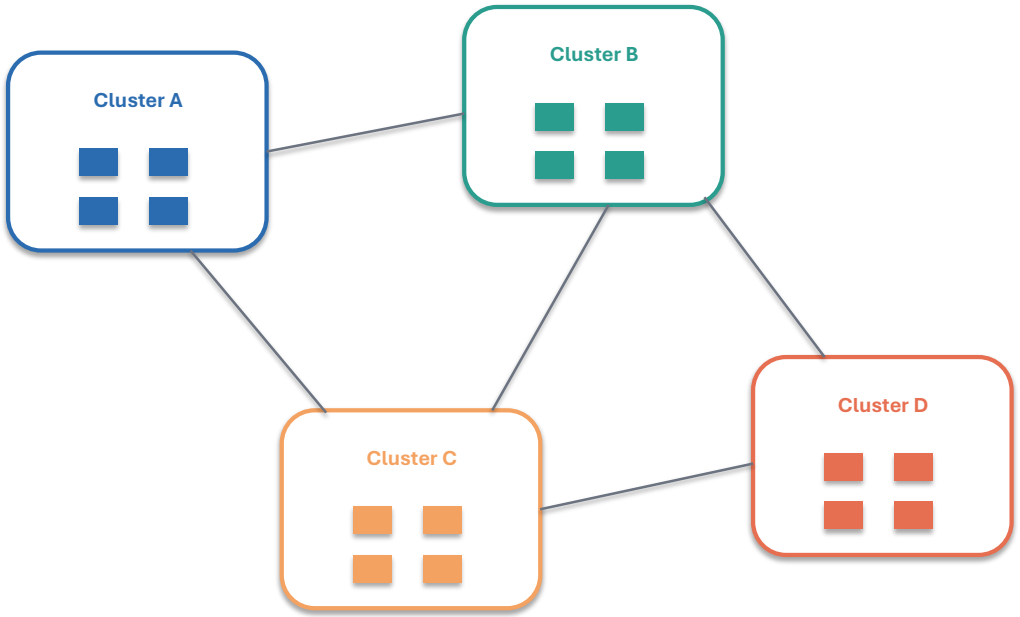
As LLMs scale up, a single cluster often cannot provide enough training capacity.

Heterogeneous links

Inter-cluster links are typically low-bandwidth and high-latency, much weaker than intra-cluster NVLink or PCIe.

Fragmented existing methods

Task-level schedulers ignore model parallelism, while parallelism automation ignores multi-task topology-aware scheduling.



Goal: reduce the number of spanned clusters K and commu cost while keeping scheduling latency low

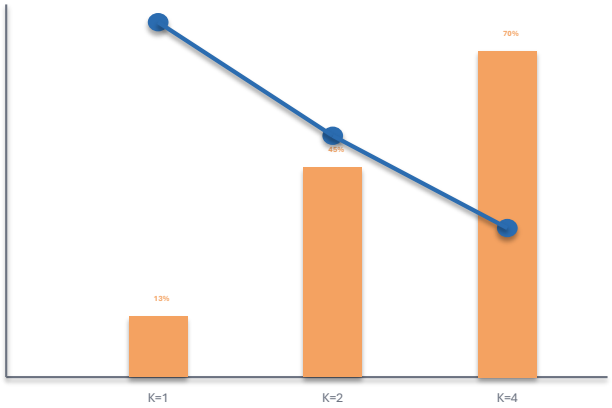
Observation

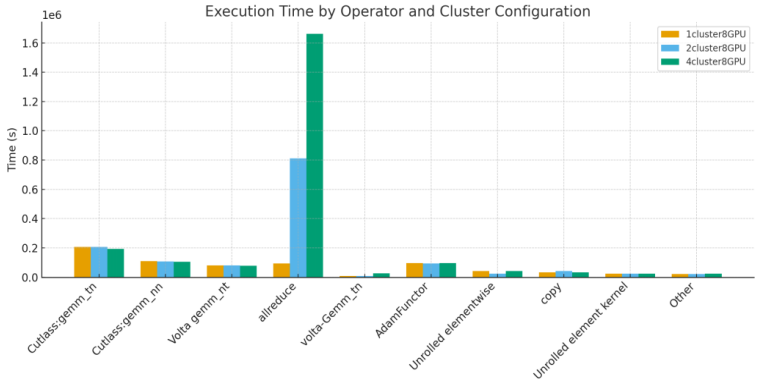
Two Key Observations

The main bottleneck in cross-cluster training is communication, and parallelism choices strongly shape communication patterns.

Observation 1: As K grows, AllReduce becomes the dominant bottleneck

Blue line: throughput scaling; orange bars: AllReduce share (schematic).





Observation 2: Parallelism strategy determines cross-cluster communication efficiency

Cross-cluster TP

Frequent fine-grained AllReduce
high synchronization cost

Prefer DP / PP

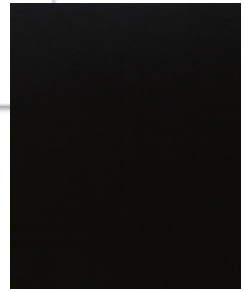
Fewer high-frequency syncs
better for WAN-like links

Design insight 1

Prioritize minimizing the number of participating clusters K instead of blindly expanding resource usage.

Design insight 2

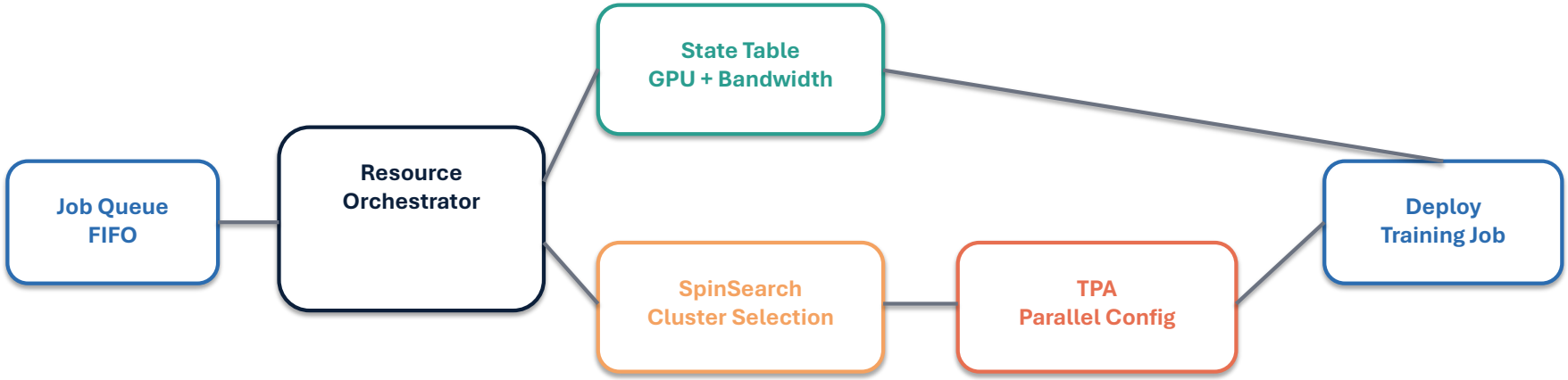
Parallelism automation must explicitly consider topology to avoid expensive cross-cluster cuts.



Design

SpiderFlow Architecture

Integrates task scheduling, topology state, and parallelism automation in a single control loop.



SpinSearch output

Selected cluster IDs and per-cluster GPU allocations.

TPA output

DP/PP configuration and topology-aware GPU placement.

System objective

Low scheduling overhead, high thr
JCT.

SpinSearch

SpinSearch: Low-overhead Topology-aware Scheduling

Core goal: satisfy GPU demand with as few clusters as possible while prioritizing high-bandwidth, low-hop connections.

1

Bandwidth-first ordering

Rank candidate clusters by link bandwidth or widest-path bandwidth.

2

First-Fit Decreasing

Allocate GPUs from high-priority clusters first to satisfy demand compactly.

3

Hop-based expansion

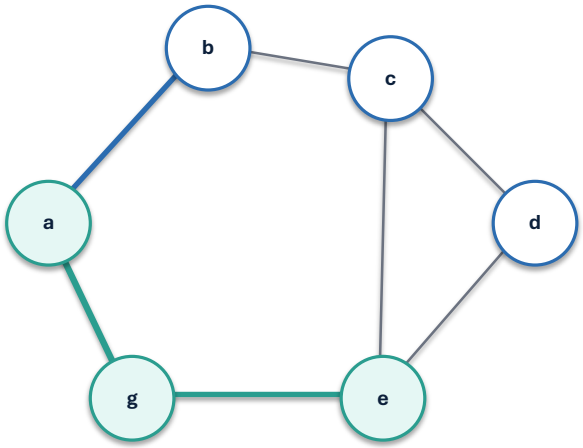
If resources are insufficient, expand hop by hop from a high-bandwidth anchor.

4

Low overhead

Complexity is O(m + n log n), avoiding the explosion of global ILP or genetic search.

Scheduling intuition: short distance, high bandwidth, fewer clusters



Example choice: fewer clusters and lower communica
distance rather than naive greedy allocation

TPA

TPA: Topology-aware Parallelism Automation

Decomposes global optimization into inter-cluster heuristics and intra-cluster ILP to balance quality and overhead.

Inter-cluster heuristic

Select clusters and place pipeline stages to minimize communication across high-cost edges.

Intra-cluster ILP

Optimize DP/PP, GPU grouping, and local placement within each cluster.

Buddy Allocator

Prefer physically close GPUs to reduce intra-cluster communication and fragmentation.

Core idea: reduce edge-cut communication cost across clusters

Design value

Preserves ILP accuracy within clusters while avoiding global ILP across up to 64 clusters, greatly reducing decision time.

Parallelism automation overhead (representative results)

Method	Topo 128/512 Runtime	Topo 512/2048 Runtime	Quality
ILP (ALPA)	182.4s	Timeout	1.00
DT-FM	145.7s	812.6s	1.18-1.23
Heuristic-ILP	3.37s	12.15s	1.05-1.08

Evaluation

Experimental Setup & Main Results

Combines physical testbed validation and large-scale simulation across multiple workloads and baselines.

Physical testbed

8 decentralized servers/clusters with 16 NVIDIA A100 GPUs in total; inter-server bandwidth of 1-15 Gb/s.

Simulator

PAI simulator with up to 64 clusters and about 3,584 GPUs; less than 5% deviation from physical results.

Workloads & Baselines

NewWorkload, Philly, and Helios; compared with Opportunistic, Lyra, Mast, DT-FM, and ALPA.

1.12-1.25×

Throughput improvement (samples/sec)

1.2-1.3×

JCT reduction (job completion time)

20-90×

Scheduling overhead reduction (avg.)

< 1%

Scheduling overhead share of total training time

SpinSearch contribution

Maintains low overhead even with 10,000 scheduling tasks while achieving QT close to ILP-based methods.

TPA contribution

Heuristic-ILP achieves near-ILP edge-cut quality while reducing runtime by

Summary

Summary & Takeaways

SpiderFlow's core value is bringing topology awareness into both task scheduling and parallelism automation.

Main contributions

Proposes a topology-aware LLM training scheduler for decentralized GPU clusters.

Models scheduling as graph search and prioritizes minimizing the number of spanned clusters K.
Uses inter-cluster heuristics plus intra-cluster ILP to reduce the parallelism automation search space.
Improves throughput and reduces JCT and scheduling overhead in both physical and large-scale simulated environments.

Key takeaway

Cross-cluster training is not simply "more resources is better"; better topology-aware placement is faster.

Current limitations

Cross-cluster TP is not yet integrated; heterogeneous GPU support is limited.

Future directions

Support heterogeneous GPUs, intra-cluster TP, EP, and more flexible hybrid parallelism.

Thanks !