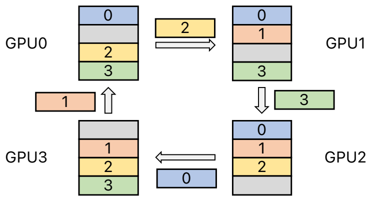
Research summary – Junyeol Ryu

jyeol.ryu@gmail.com
https://junyeol.me/about.html

Congestion avoidance for collective communication in PCIe-based systems

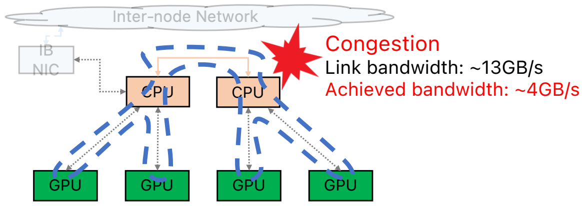
Heehoon Kim, Junyeol Ryu, Jaejin Lee. TCCL: Discovering Better Communication Paths for PCIe GPU Clusters. ASPLOS '24. [link]

Collective communication is essential for parallelism in training AI models!



Ring algorithm of AllGather for four GPUs

GPU communication libraries exhibit **low performance for PCIe-based systems**



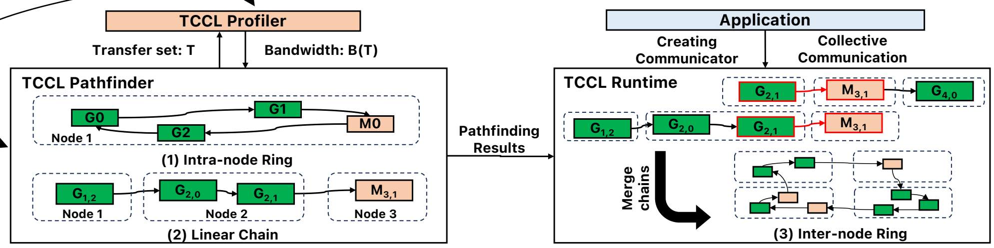
Ring example of a PCIe-based system

Existing libraries can neither identify nor avoid congestion!

Key findings from my analysis of NCCL

- Existing libraries find paths based **solely on the bandwidths of individual links**
- However, multiple transfers are executed **simultaneously across the PCIe host bridge** during collective communication

Insight 1: Profiler specialized in measuring simultaneous multiple transfers



Insight 2: Enumerate all possible paths while minimizing the search time for performant path

Overview of TCCL

Result:

- Up to $\mathbf{2.07}\times$ speedup for collective communication
- Up to $\mathbf{1.11}\times$ speedup for training AI models

Further research directions

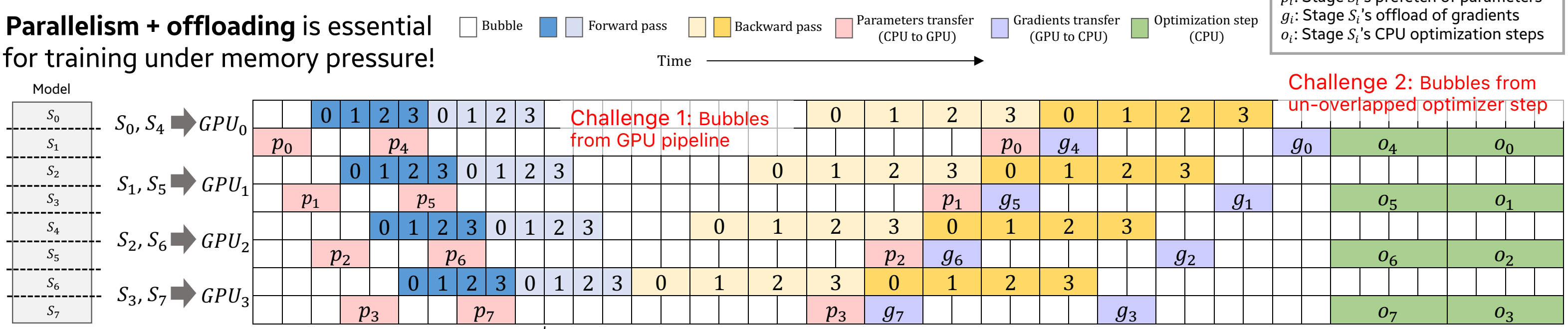
- Extending beyond ring algorithm (e.g., double binary tree, all-to-all)
- Overlapping dependent communication and computation by decomposition
- Utilizing multi-path opportunities

Utilizing compute and memory of GPUs and CPUs for large-model training

Junyeol Ryu*, Yujin Jeong*, Daeyoung Park, Jinpyo Kim, Heehoon Kim, Jaejin Lee. SPipe: Hybrid GPU and CPU Pipeline for Training LLMs under Memory Pressure. Under submission to OSDI '25. [link]

Parallelism + offloading is essential for training under memory pressure!

p_i: Stage S_i's prefetch of parameters
g_i: Stage S_i's offload of gradients
o_i: Stage S_i's CPU optimization steps

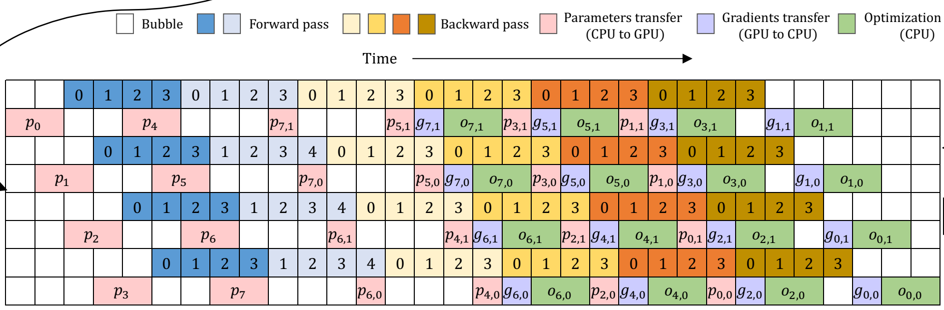


- Stage's memory should not exceed GPU memory
- Multiple stages are assigned to each GPU

Stages are fetched to GPU memory and gradients are offloaded to CPU memory in an overlapped manner

Optimizer updates the stages' parameters on the CPU

Insight 1: Shared parameters on CPU memory allows **decoupling** a stage's forward/backward pass onto different GPUs



Insight 2: CPU optimizer steps can execute **in parallel** with GPU's forward/backward pass

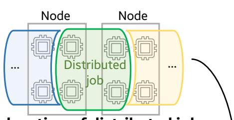
Result: Avg. $\mathbf{1.26}\times$ speedup for training LLaMA2 models (~100B) using ~32 V100 32GB GPUs

Further research directions: Hybrid parallelism (e.g., PP+TP) + offloading technique

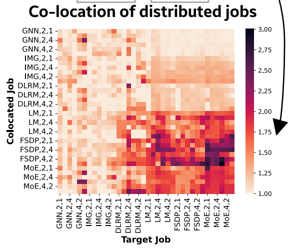
Reinforcement learning-based resource management

Junyeol Ryu, Jeongyoon Eo. Network Contention-Aware Cluster Scheduling with Reinforcement Learning. ICPADS '23. [link]

Mitigating network congestion is essential when scheduling distributed jobs in GPU clusters!



Co-location of distributed jobs

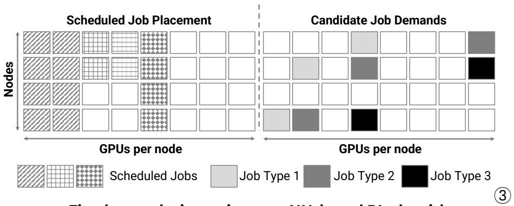


Insight: Co-locating jobs yields **varying performance effects** due to model type, parallelism, placement

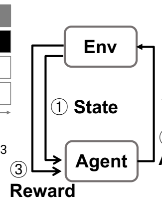
However, it is infeasible to try all co-location options on every new job request

Reinforcement learning (RL)

- Repetitive decisions leave abundant training data to RL algorithm
- Reward reflects complex objectives (e.g., min. congestion, max GPU utilization)
- Adapt to shifting or unseen circumstances by **explore-and-exploit**



Fixed state design as input to NN-based RL algorithm



- Penalize increase in congestion
- Incentivize increase in GPU utilization

- Schedule
- Migration
- Preemption

⊕ Insight: **Simple heuristics** can effectively assist RL (e.g., selective multiplexing with greedy approach)

Result: Up to $\mathbf{18.2}\%$ reduction for average job completion time