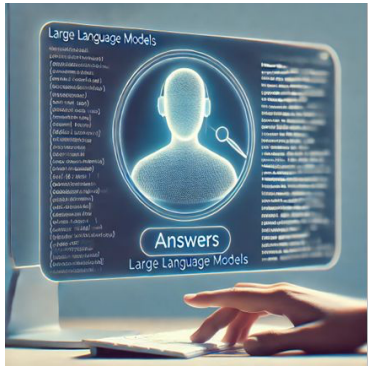
FreeRide: Harvesting Bubbles in Pipeline Parallelism

Jiashu Zhang, Zihan Pan, Molly Xu, Khuzaima Daudjee, and **Sihang Liu**

Middleware 2025

Large Language Models

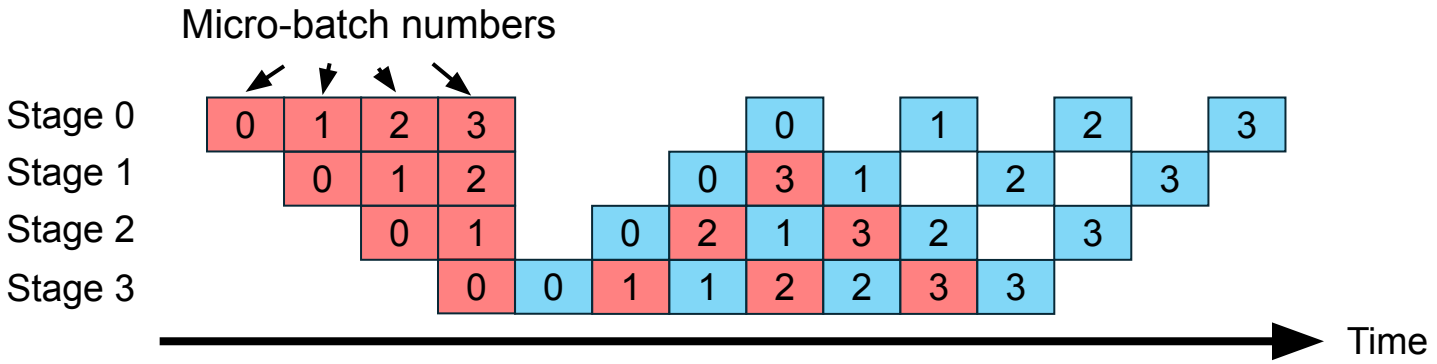
- Large language models (LLMs) are widely used in many applications
- Training these models is extremely costly
  - Require multiple GPUs
  - Take weeks to months
- Pipeline parallelism is a commonly used to train LLMs on multiple GPUs



* Images generated with ChatGPT

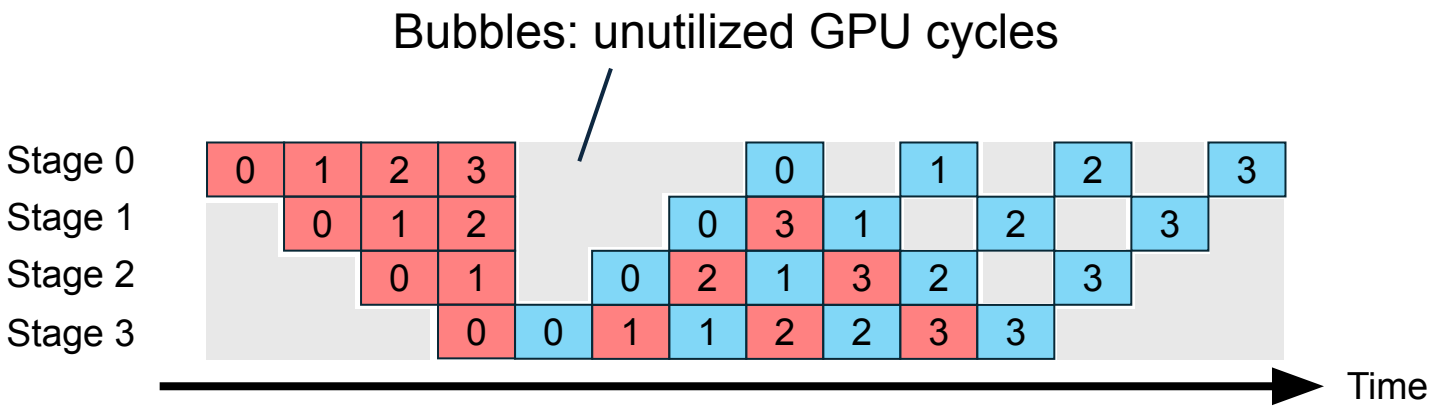
Pipeline Parallelism for LLM Training

- A model is divided into stages, each executing a few layers
- Stages are deployed on different GPUs
- Inputs are divided into smaller batches, i.e., micro-batches
- Each micro-batch goes through **forward propagation (FP)** and **backward propagation (BP)**, and then processes the next.



Resource Underutilization

- During training, resources can remain underutilized
  - We refer to these underutilized periods as **bubbles**
  - Bubbles take over 40% of GPU cycles in the 4-stage setup below

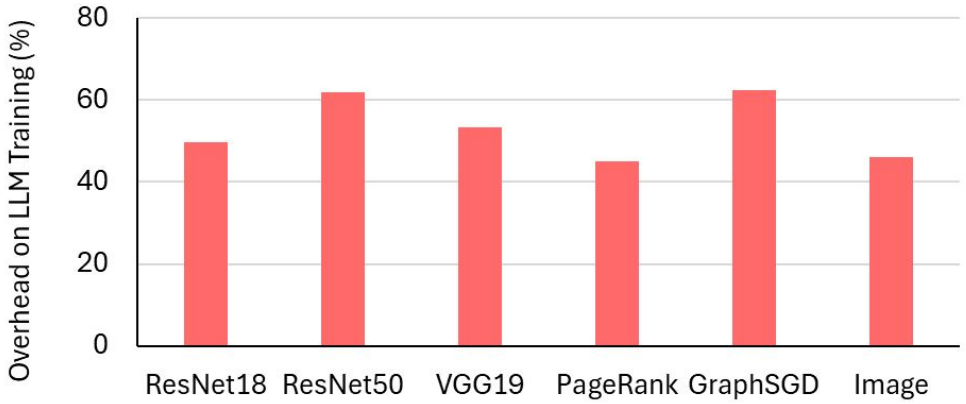


Bubble Mitigations

- Common approaches to reduce bubbles
  - Overlap communication and computation [Hanayo'21, Chimera'23]
  - Interleave forward and backward propagation [GPipe' 19, PipeDream'19, DAPPLE'21]
- Side effects
  - More complex coding
  - Potentially higher memory usage
- However, bubbles fundamentally exist …

Instead of mitigating bubbles, can we utilize them?

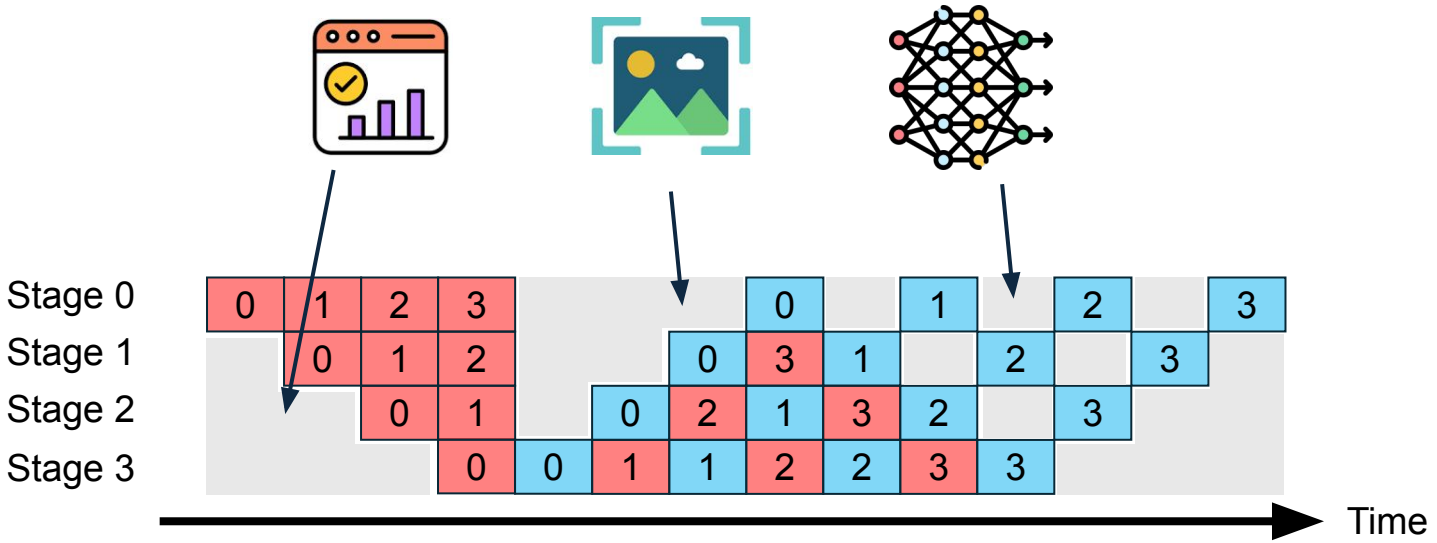
- Naively co-locating workloads can lead to significant overhead
- The primary LLM training workload is more critical



On average, naive co-location leads to over **53% overhead** on LLM training

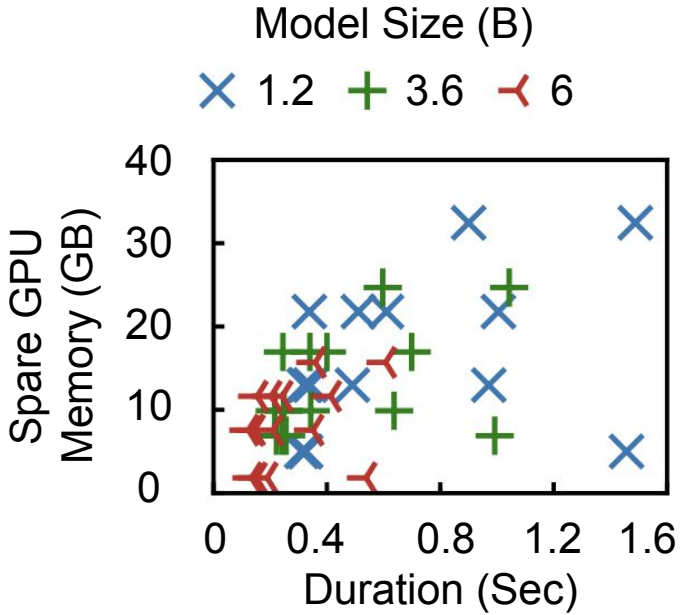
Goal of This Work

Utilize the bubbles in pipeline parallel LLM training with minimum overhead



Bubble Characteristics

- Underutilization during LLM training
  - Idle GPU cycles
  - Spare GPU memory
- Observations
  - Larger LLMs have better utilization
  - Bubbles vary in shape
    - Different spare memory sizes
    - Different durations



Our Approach: FreeRide for Utilizing Bubbles

Key observations

- Bubbles are usable resources
  - Bubbles are sometimes long, e.g., 0.2 ~ 1 sec
  - There is often unutilized memory
- Lightweight GPU applications that can work as ***side tasks*** widely exist, e.g.,
  - Smaller CNNs
  - Graph analytics
  - Image processing

Fit side tasks exactly into bubbles to minimize interference

Challenges of Using Bubbles

- Programming effort
  - Bubbles are intermittent and vary in duration
  - Side tasks should be tailored to fit into spare resources
- Overhead due to bubble usage
  - Main training workload consumes expensive and high-in-demand GPUs
  - Overhead of using bubbles should be ideally zero

Our Solution: FreeRide Framework

FreeRide tackles both challenges by providing

- Easy-to-use programming interface
  - No modifications to LLM training
  - Minimize user effort when developing code to use the bubbles
- Profiling-guided side task management
  - Minimize interference to LLM training
  - Guarantee sufficient bubble resources for side tasks

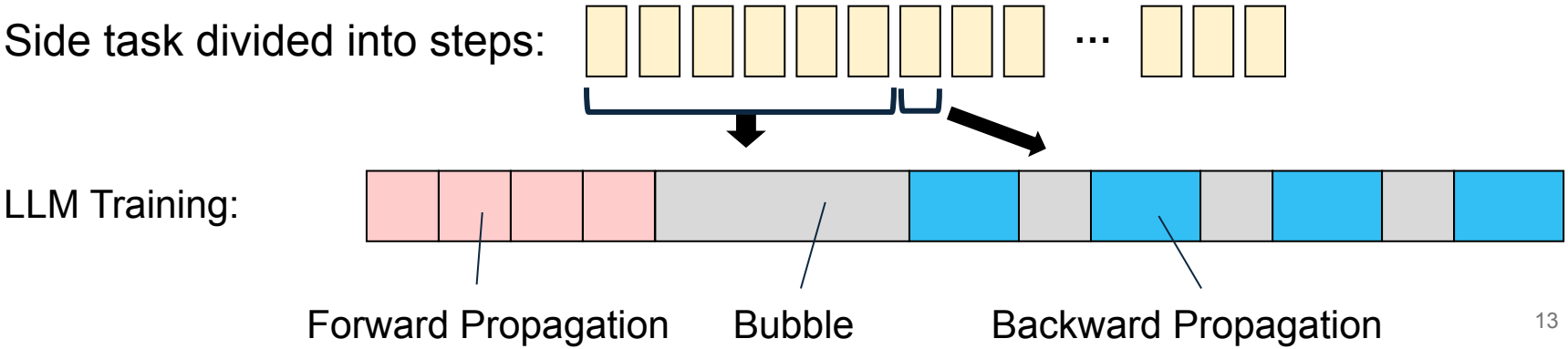
Programming Interface

FreeRide provides two types of interface:

1. Iterative
   Tailor side tasks for bubble execution using FreeRide API
2. Imperative
   FreeRide enforces side tasks to fit into bubbles

Iterative Interface

- Directly use FreeRide API to implement side tasks
  - Side-task implementation limits GPU execution time to fit into bubbles
  - Minimize performance overhead
- GPU programs usually follow iterative patterns, e.g.,
  - Iterations in ML training

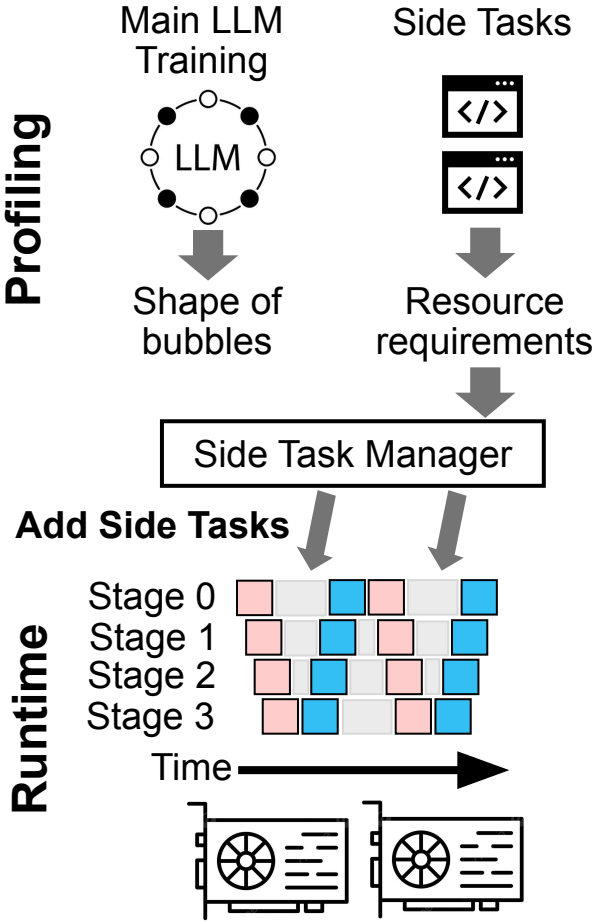


Imperative Interface

- Some GPU programs are not easily divisible into small steps, e.g., video rendering
- FreeRide manages side tasks directly to limit execution time
  - Minimum programming effort
  - Slightly higher overhead
  - Work as a *fallback solution*

Side Task Management Overview

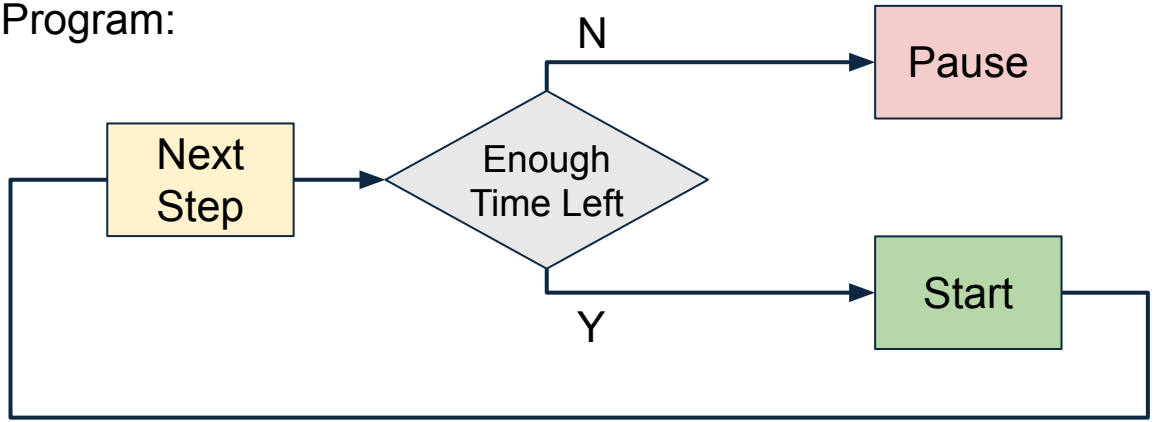
- Profiling
  - Profile *training framework* to get shape of bubbles (duration and spare memory)
  - Profile *side tasks* to understand their memory consumption
- Fit side task into bubbles during runtime
  - *Start* at the beginning of a bubble
  - *Pause* at the end of a bubble



Side Task Resource Control: Execution Time

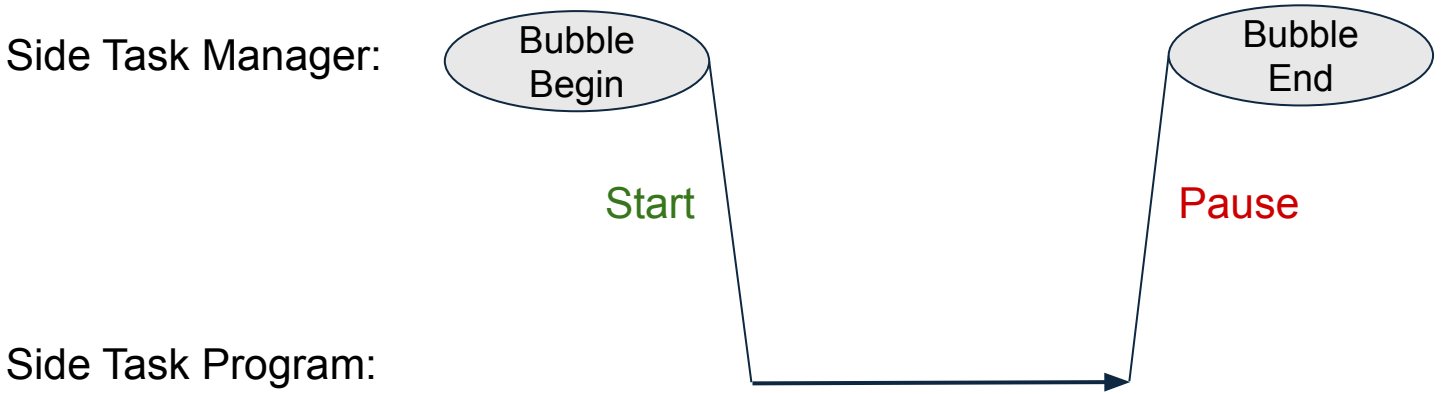
- Side task manager controls execution time for each interface
  - **Iterative interface: program-directed**
  - Imperative interface: framework-enforced

Side Task Program:



Side Task Resource Control: Execution Time

- Side task manager controls execution time for each interface
  - Iterative interface: program-directed
  - **Imperative interface: framework-enforced**

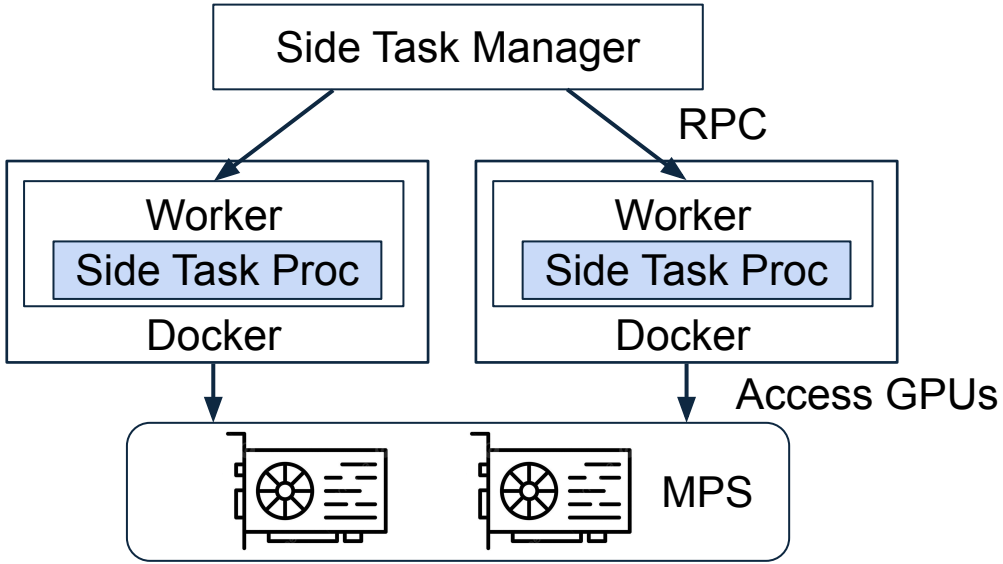


Side Task Resource Control: GPU Memory

- Profiler provides GPU memory requirements of each side task
- FreeRide assigns a side task only if its memory requirement fits into the spare GPU memory
- To handle unexpected memory usage
  - FreeRide limits side task GPU memory using Nvidia MPS
  - Side task is terminated if its memory exceeds the limit
  - Prioritize main LLM training

FreeRide System

- Side task manager coordinates a set of workers
  - Workers run in Dockers for isolation
  - Assigns side tasks to workers through RPCs



Evaluation Methodology

- Main training workload
  - 3.6B-parameter NanoGPT
  - Using DeepSpeed pipelining framework
- Side task workloads
  - CNN Inference:
    ResNet18, ResNet50, VGG19
  - GPGPU workloads:
    Page Rank, Graph SGD, Image processing
  - Mix of workloads above

Evaluation Methodology (cont.)

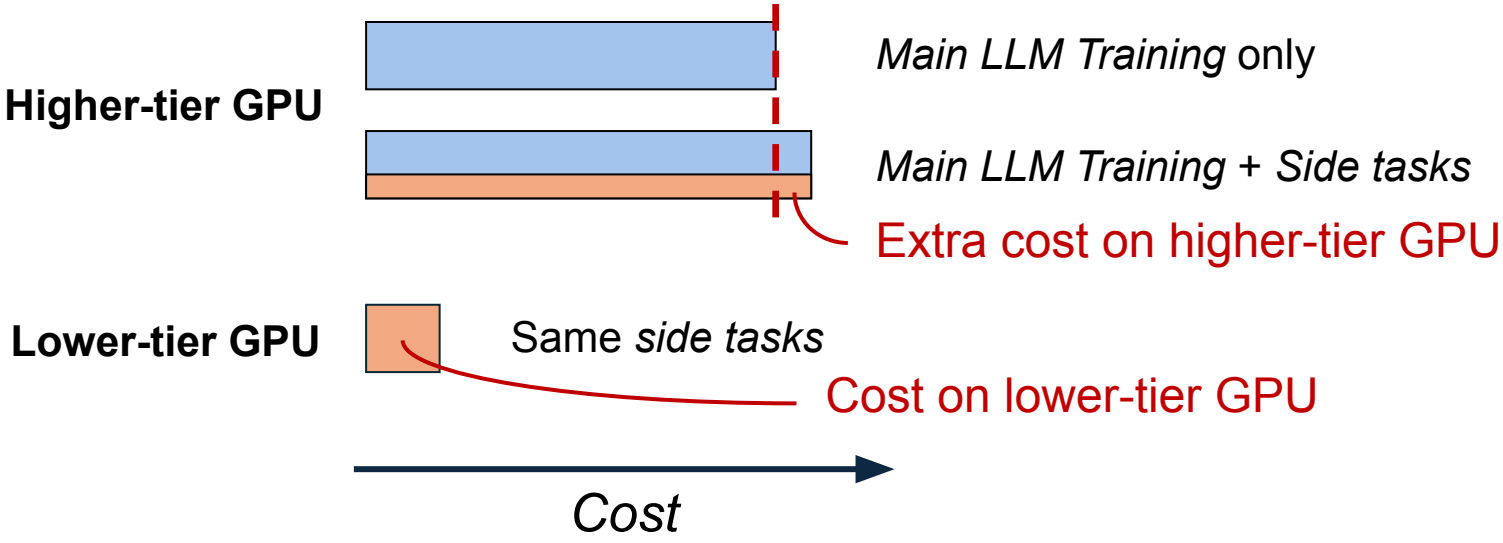
- Hardware Platforms
  - Higher-tier GPU: 4x Nvidia RTX6000 ADA (48GB): $3.96/hour
  - Lower-tier GPU: 1x Nvidia RTX 3080 (10GB): $0.18/hour
- Baselines
  - Naive co-location:
    Run multiple GPU kernels concurrently
  - Nvidia MPS:
    Use MPS to isolate side tasks and limit their memory

* Pricing from runpod.io

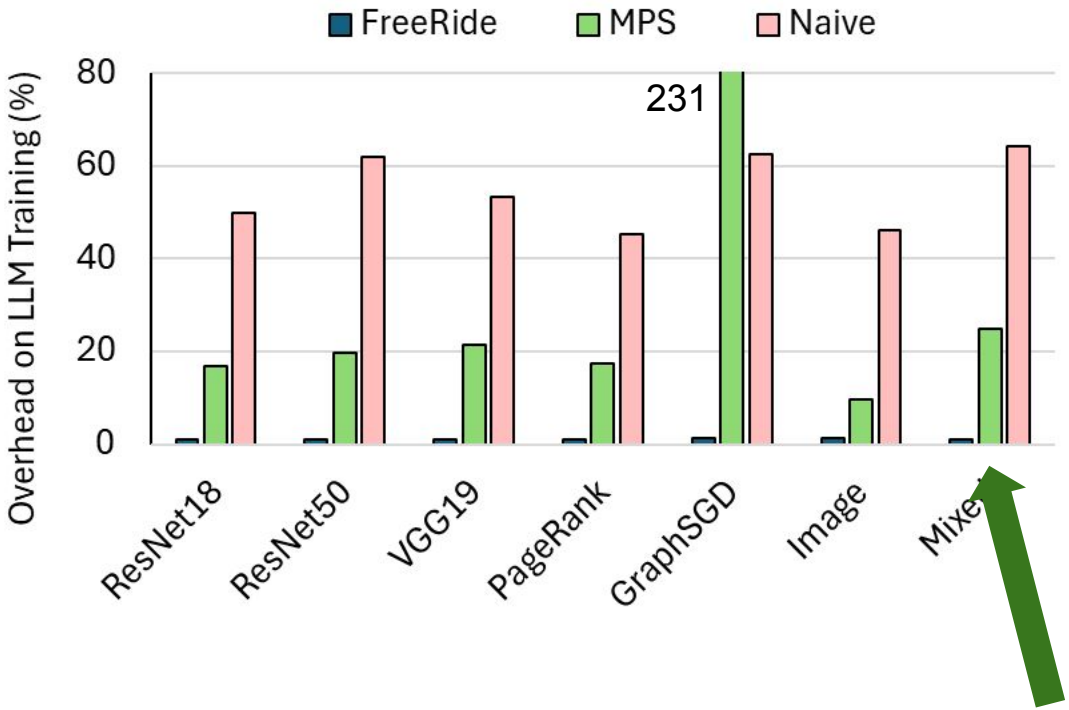
Evaluation Metric

- Performance overhead on LLM training
- Cost savings from bubble utilization

Cost on lower-tier GPU vs Extra cost on higher-tier GPU

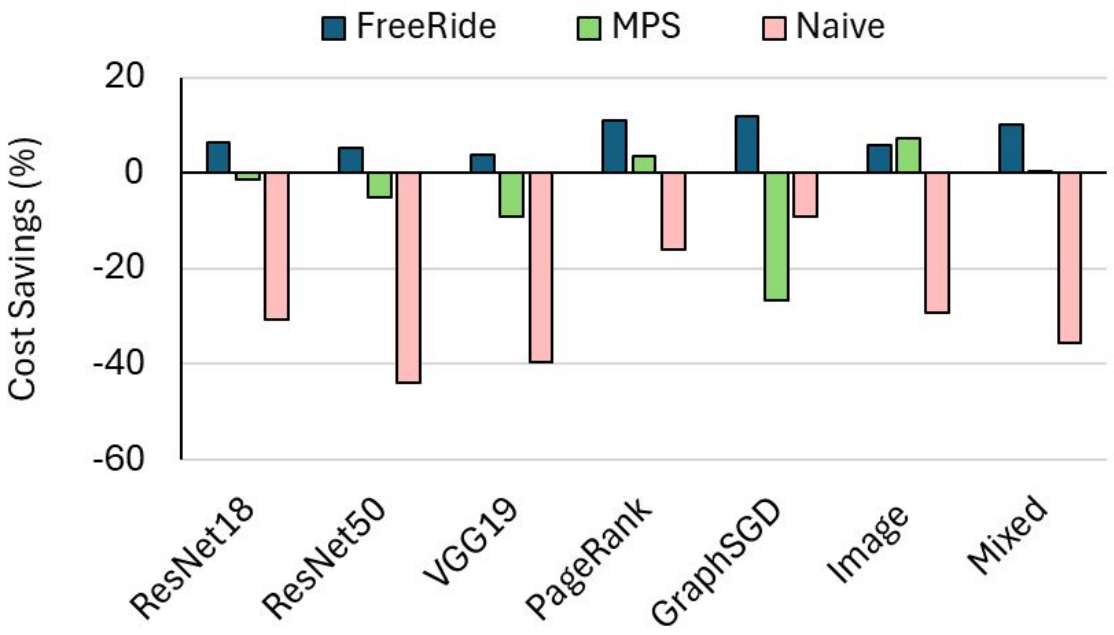


Performance Overhead



FreeRide has <1.5% overhead

Cost Savings



Negative savings = cost increase

FreeRide saves 8% of costs on average

Summary

- Pipeline parallelized LLM training leads to underutilized resources, i.e., **bubbles**
- FreeRide is a framework that enables generic GPU programs to utilize bubbles
- FreeRide achieves 8% cost savings by using bubbles and introduces negligible overhead to LLM training