GPU-Disaggregated Serving for Deep Learning Recommendation Models at Scale

Lingyun Yang†, Yongchen Wang, Yinghao Yu, Qizhen Weng†, Jianbo Dong, Kan Liu, Chi Zhang, Yanyi Zi, Hao Li, Zechao Zhang, Nan Wang, Yu Dong, Menglei Zheng, Lanlan Xi, Xiaowei Lu, Liang Ye, Guodong Yang, Binzhang Fu, Tao Lan, Liping Zhang, Lin Qu, Wei Wang†

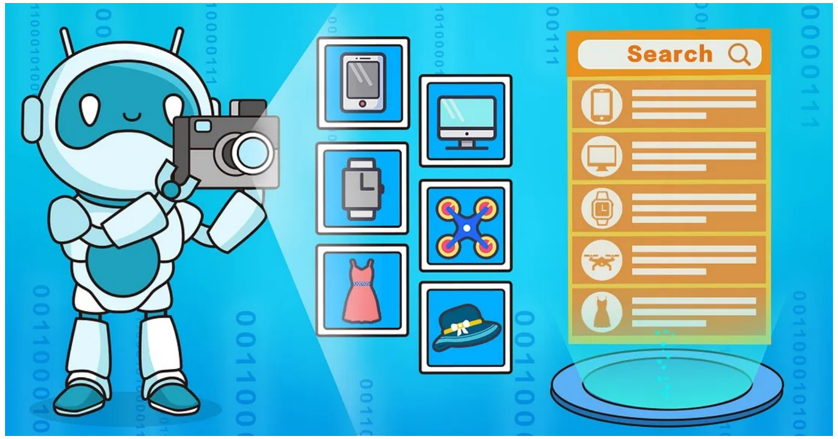
†HKUST *Alibaba Group*

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Alibaba Group
阿里巴巴集团

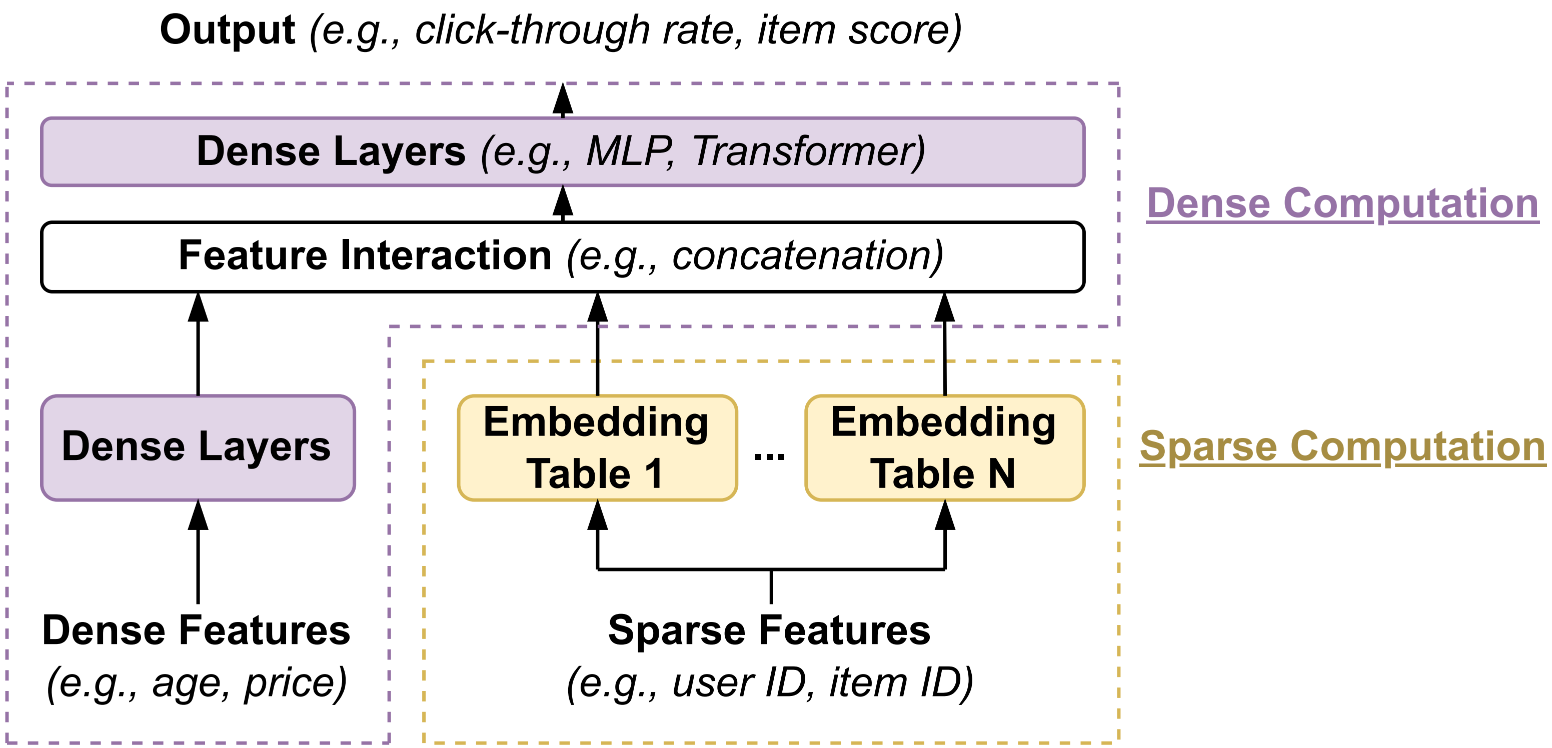
Deep Learning Recommendation Model (DLRM)

- DLRMs are widely used in e-commerce platform to provide accurate, personalized recommendations to improve customer experience
- Applications: web search, recommendation, advertisements, etc.



Images credit to: https://alibabatech.medium.com

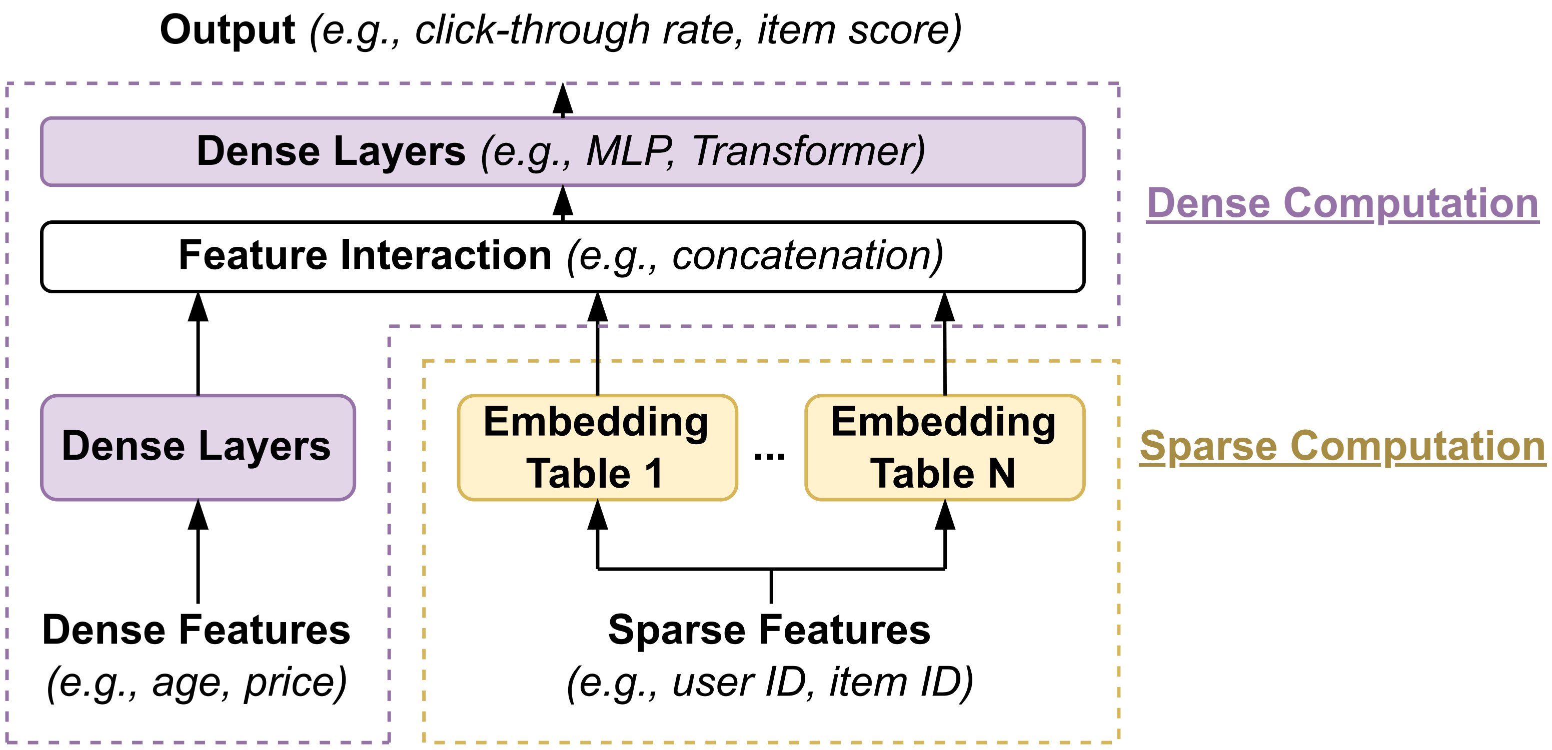
An Illustration for DLRM Serving



<u>Embedding operations</u> are characterized by large embedding tables (typically 100GB-1TB), low compute-intensity

<u>Dense network component</u> is better executed on GPUs

An Illustration for DLRM Serving

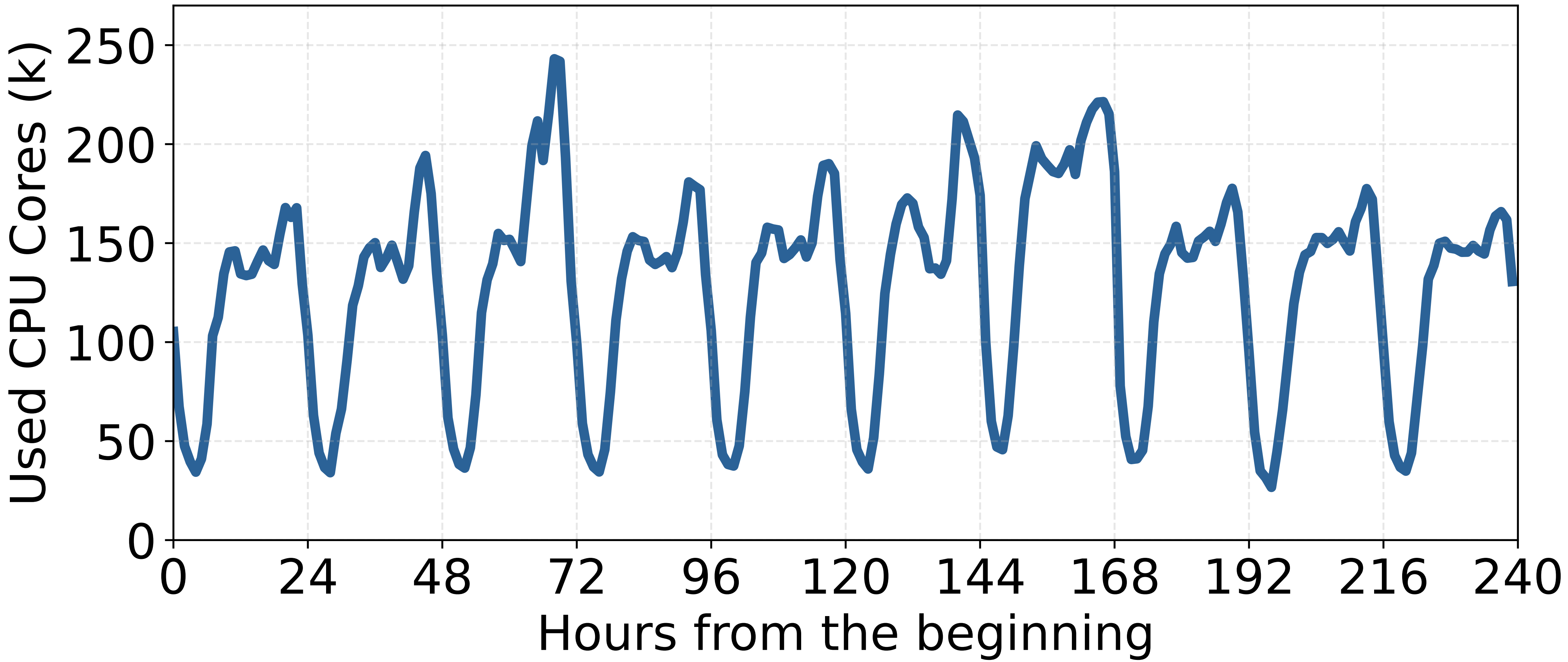


Embedding operations are characterized by large embedding tables (typically 100GB-1TB), low compute-intensity

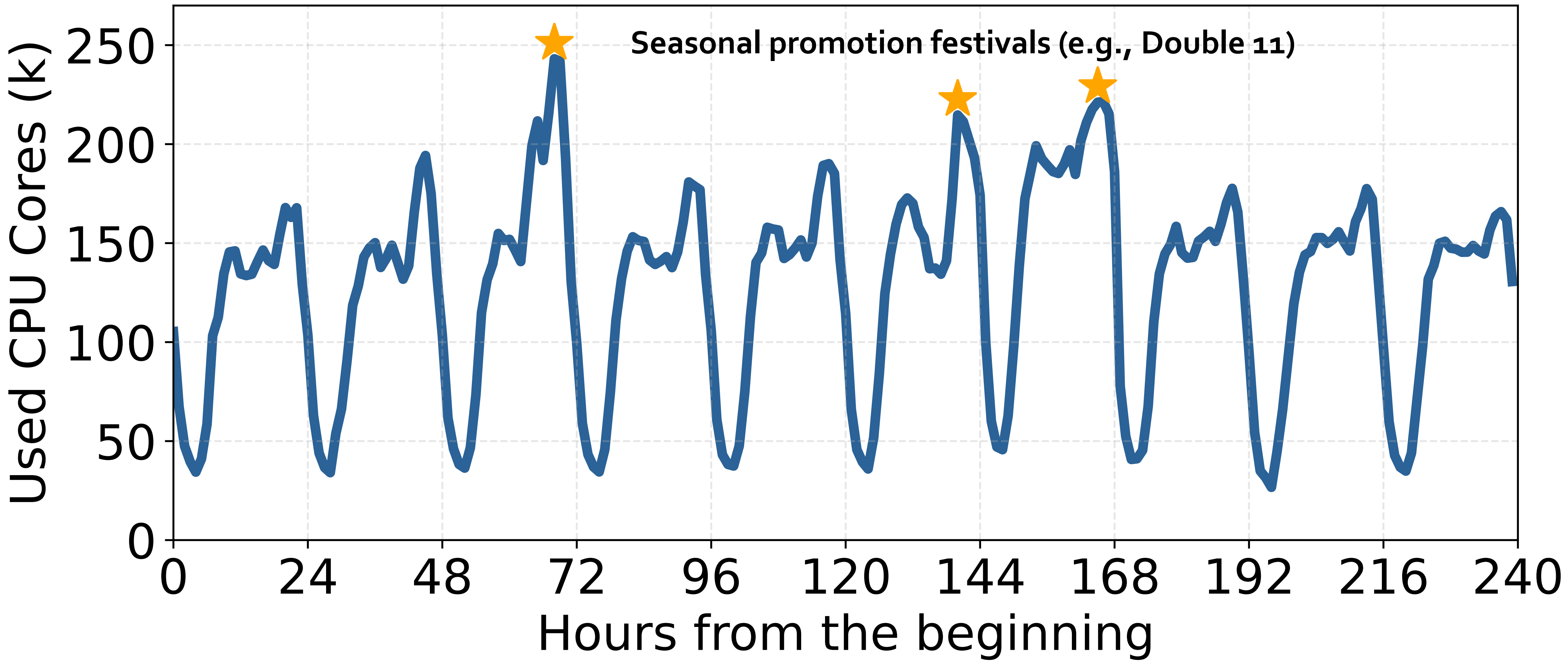
Dense network component is better executed on GPUs

A typical DLRM task may require <48 CPUs, 1 GPU>

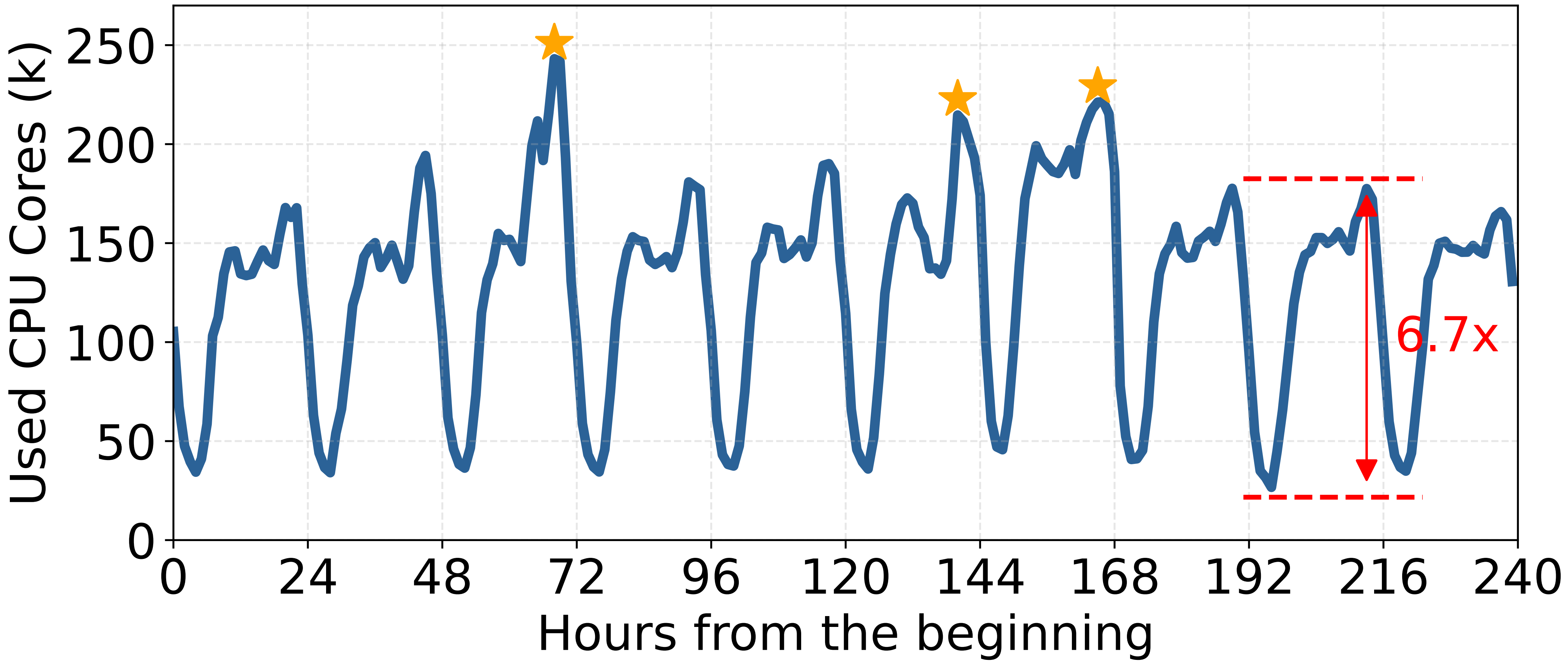
Daily and Seasonal Variations of DLRM Services



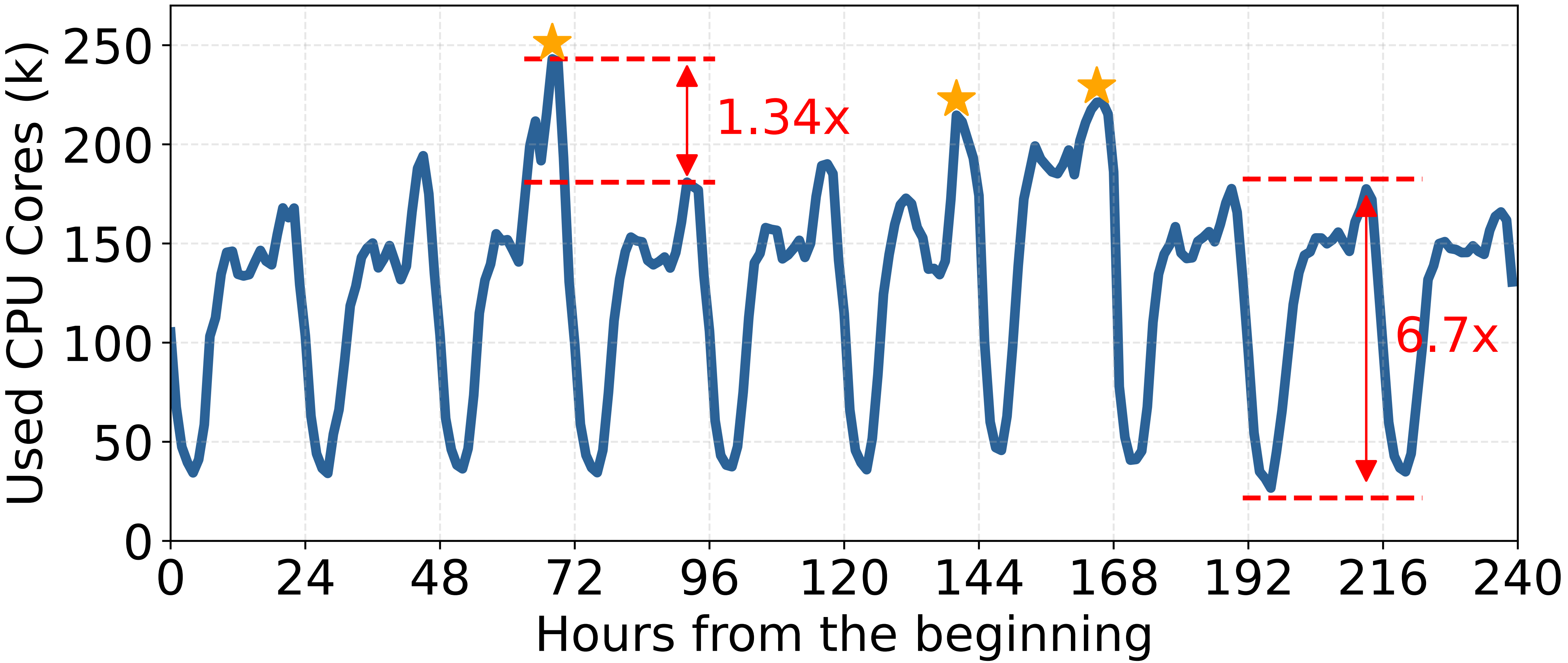
Daily and Seasonal Variations of DLRM Services



Daily and Seasonal Variations of DLRM Services



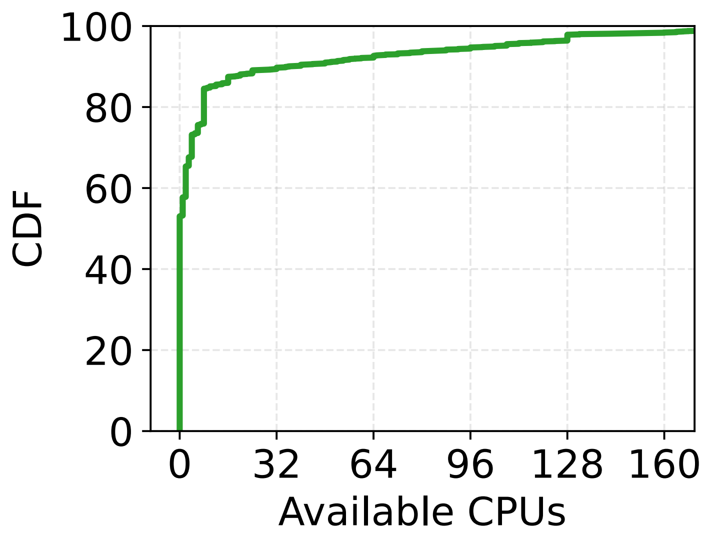
Daily and Seasonal Variations of DLRM Services

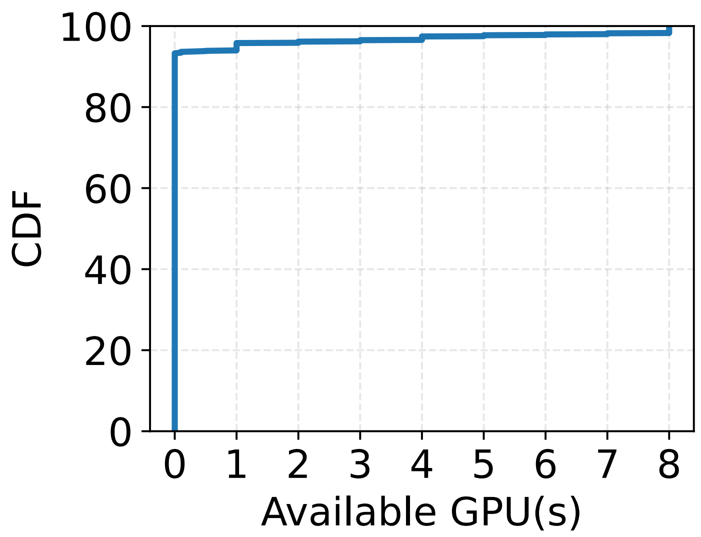


C1: Resource Fragmentation for Daily DLRM Serving

- Shared clusters have high allocation rates (e.g., > 90%)

- Hard to scale DLRM instances due to severe fragmentation



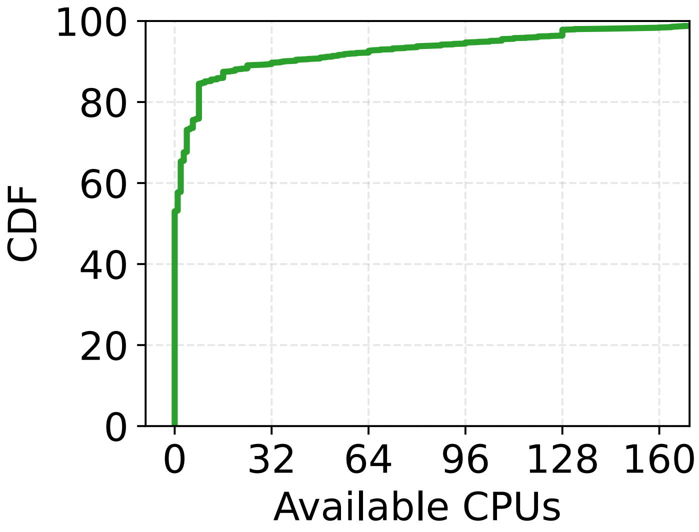


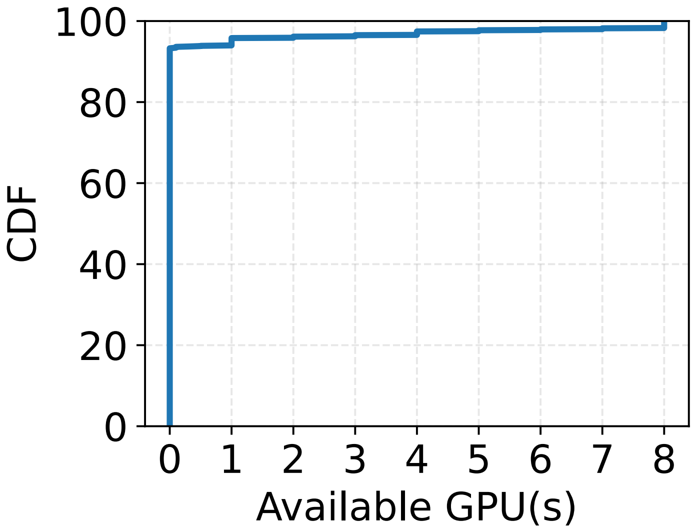
- 4k nodes
- 640k CPU cores
- 11k GPUs

C1: Resource Fragmentation for Daily DLRM Serving

- Shared clusters have high allocation rates (e.g., > 90%)

- Hard to scale DLRM instances due to severe fragmentation
  - DLRM instances typically have high CPU-to-GPU ratio

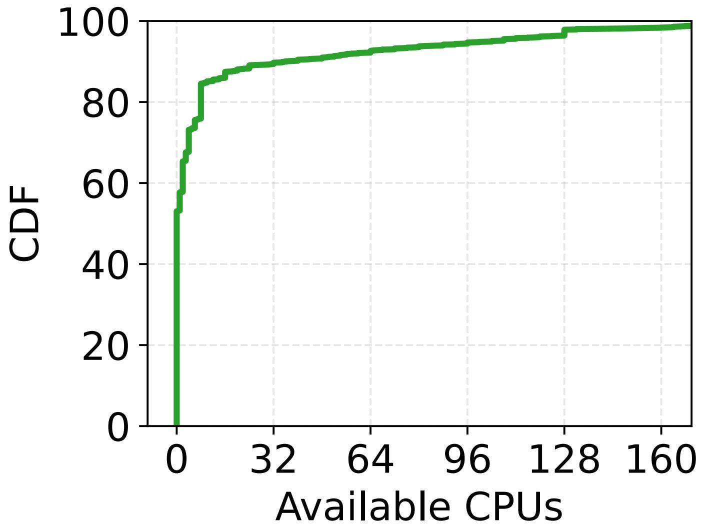


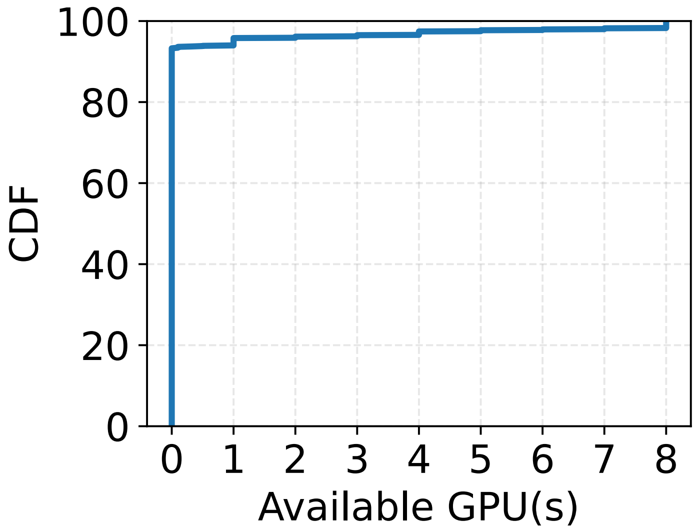


- 4k nodes
- 640k CPU cores
- 11k GPUs

C1: Resource Fragmentation for Daily DLRM Serving

- Shared clusters have high allocation rates (e.g., > 90%)
- Hard to scale DLRM instances due to severe fragmentation
  - DLRM instances typically have high CPU-to-GPU ratio
  - Over 30k fragmented CPUs and more than 200 fragmented GPUs





- 4k nodes
- 640k CPU cores
- 11k GPUs

C2: Load Spikes in Seasonal Promotional Festivals

- Over-provisioning for the peak load
  - Results in significant underutilization

- Capacity loaning during load spikes

C2: Load Spikes in Seasonal Promotional Festivals

- Over-provisioning for the peak load
  - Results in significant underutilization

- Capacity loaning during load spikes
  - Existing datacenter include *multiple* purpose-specific infrastructures: some for **training** and the others for **inference**

C2: Load Spikes in Seasonal Promotional Festivals

- Over-provisioning for the peak load
  - Results in significant underutilization

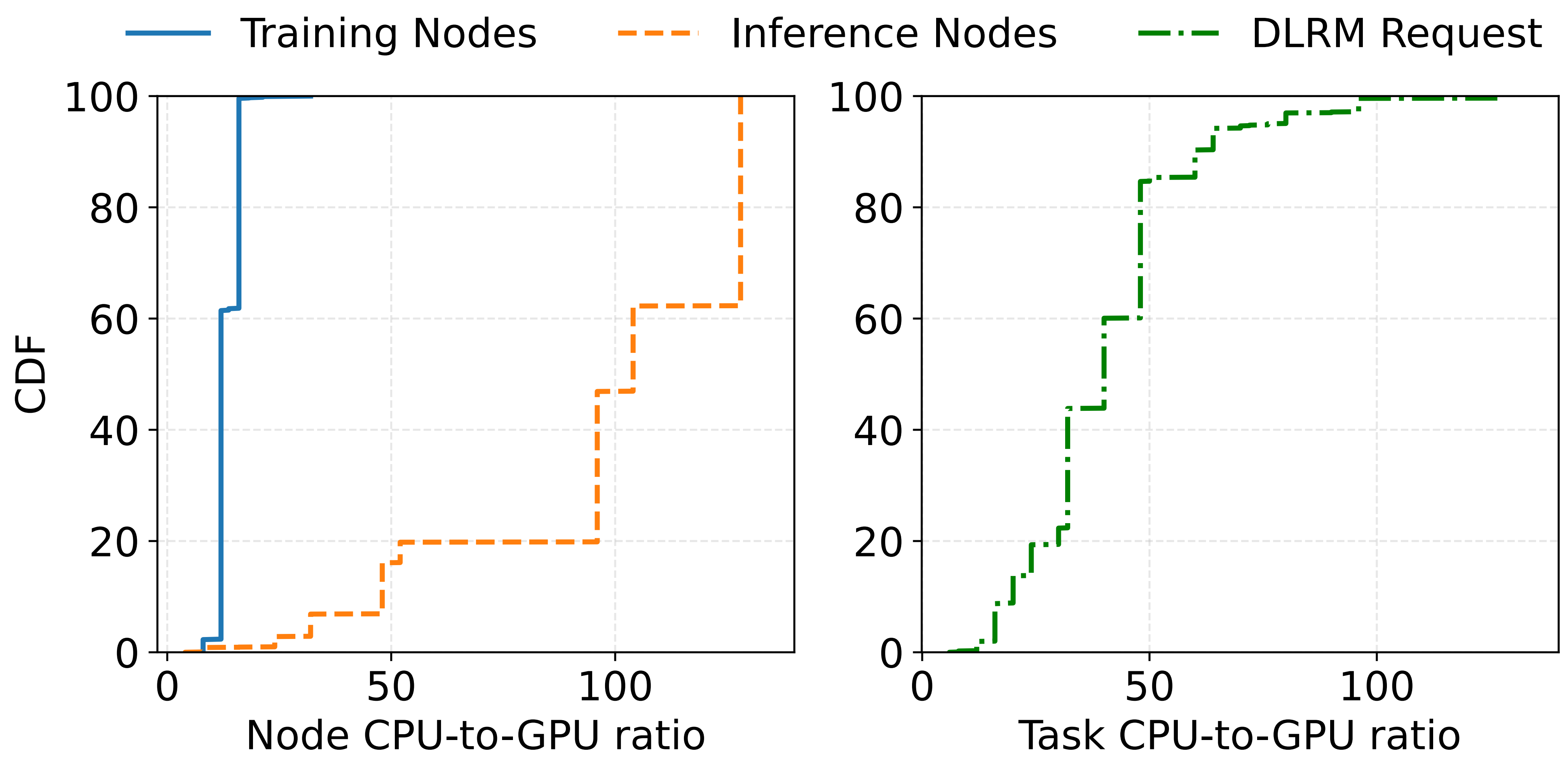
- Capacity loaning during load spikes
  - Existing datacenter include *multiple* purpose-specific infrastructures: some for **training** and the others for **inference**
  - Can we temporarily loan GPU servers from **training clusters** to handle excessive recommendation queries?

C2: Load Spikes in Seasonal Promotional Festivals

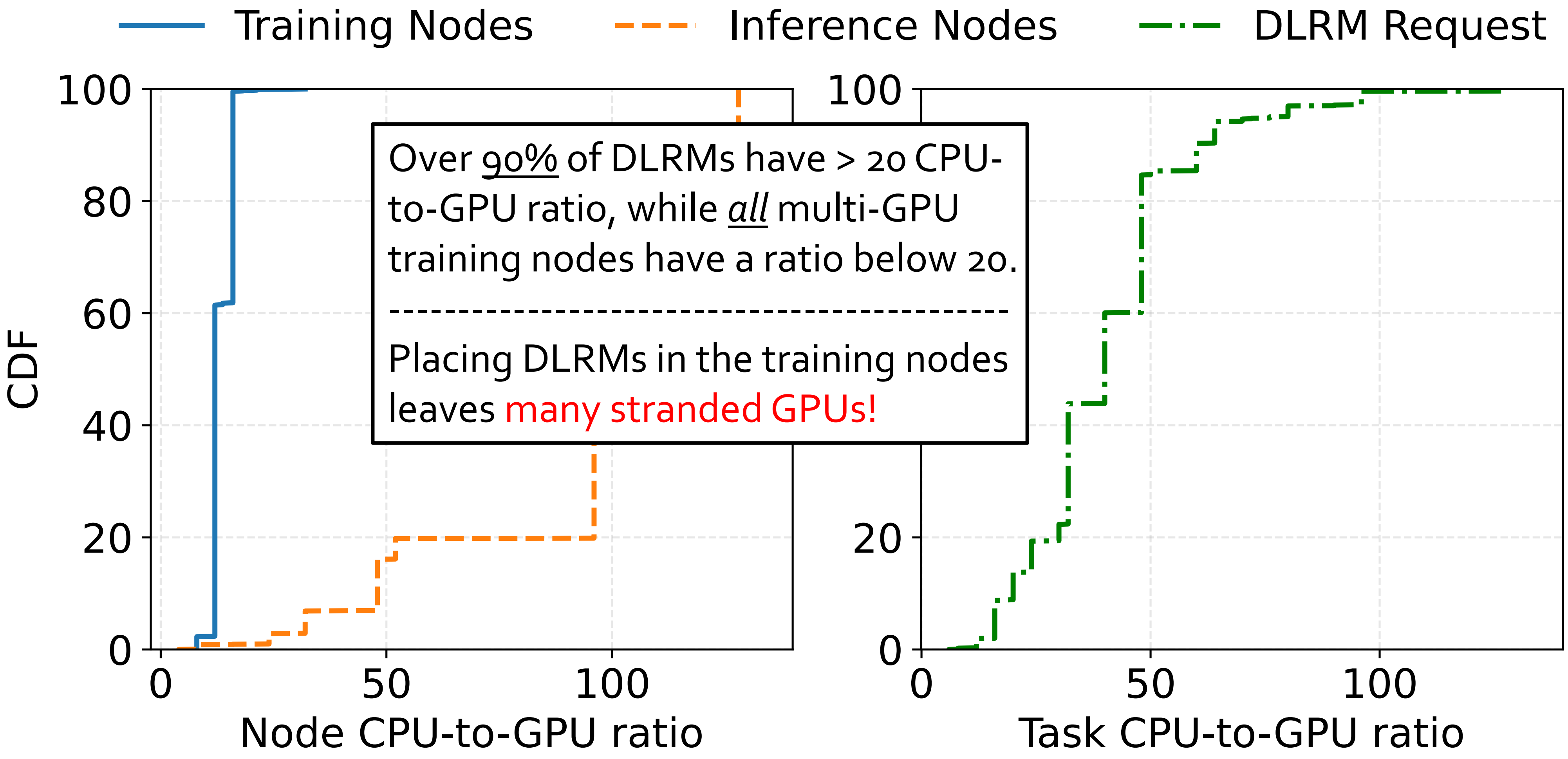
- Over-provisioning for the peak load
  - Results in significant underutilization

- Capacity loaning during load spikes
  - Existing datacenter include *multiple* purpose-specific infrastructures: some for **training** and the others for **inference**
  - Can we temporarily loan GPU servers from **training clusters** to handle excessive recommendation queries?
  - The mismatch between server configuration and resource demand renders capacity loaning ineffective!

Resource Heterogeneity in GPU Clusters



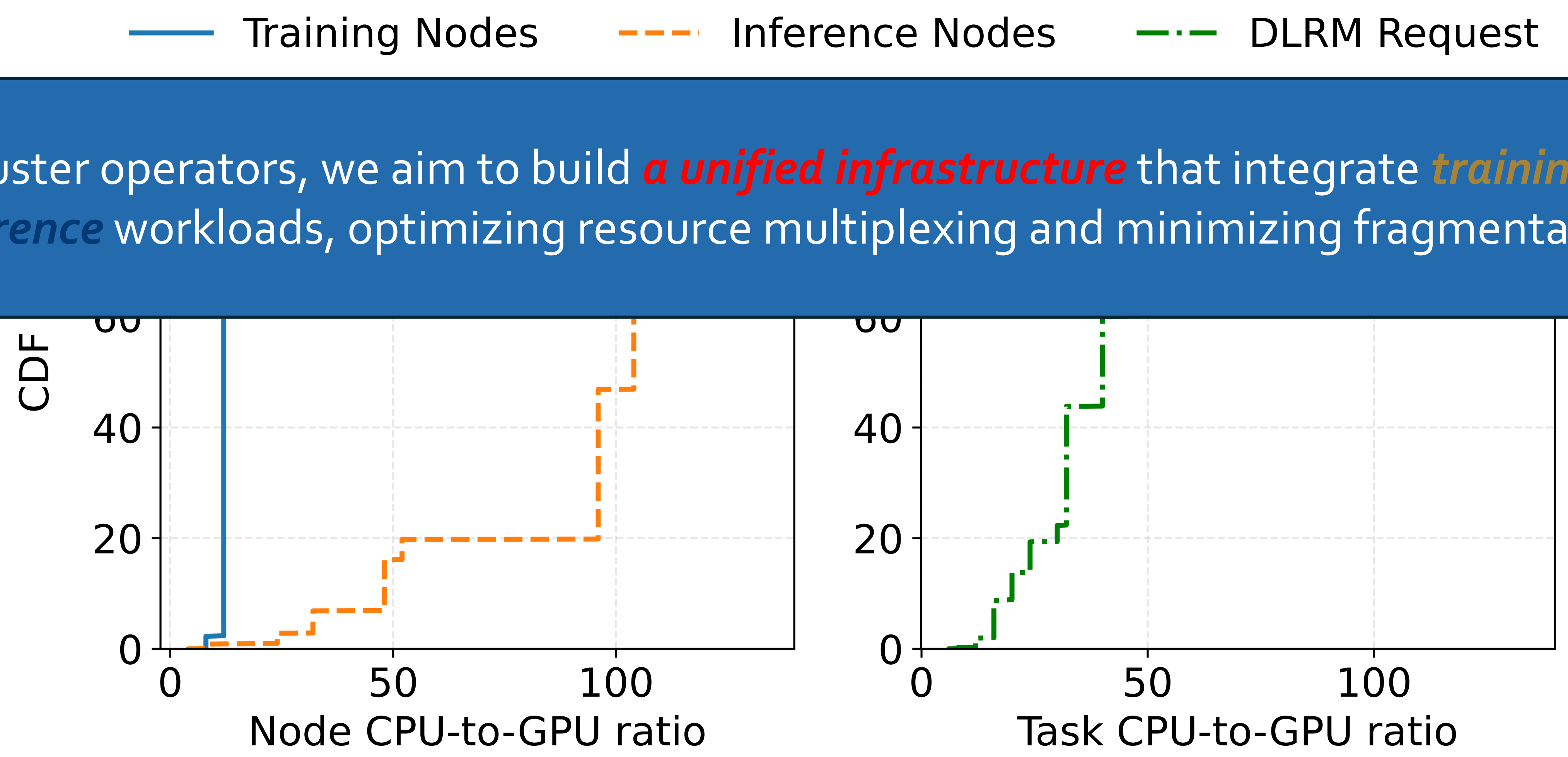
Resource Heterogeneity in GPU Clusters



Over <u>90%</u> of DLRMs have > 20 CPU-to-GPU ratio, while <u>*all*</u> multi-GPU training nodes have a ratio below 20.

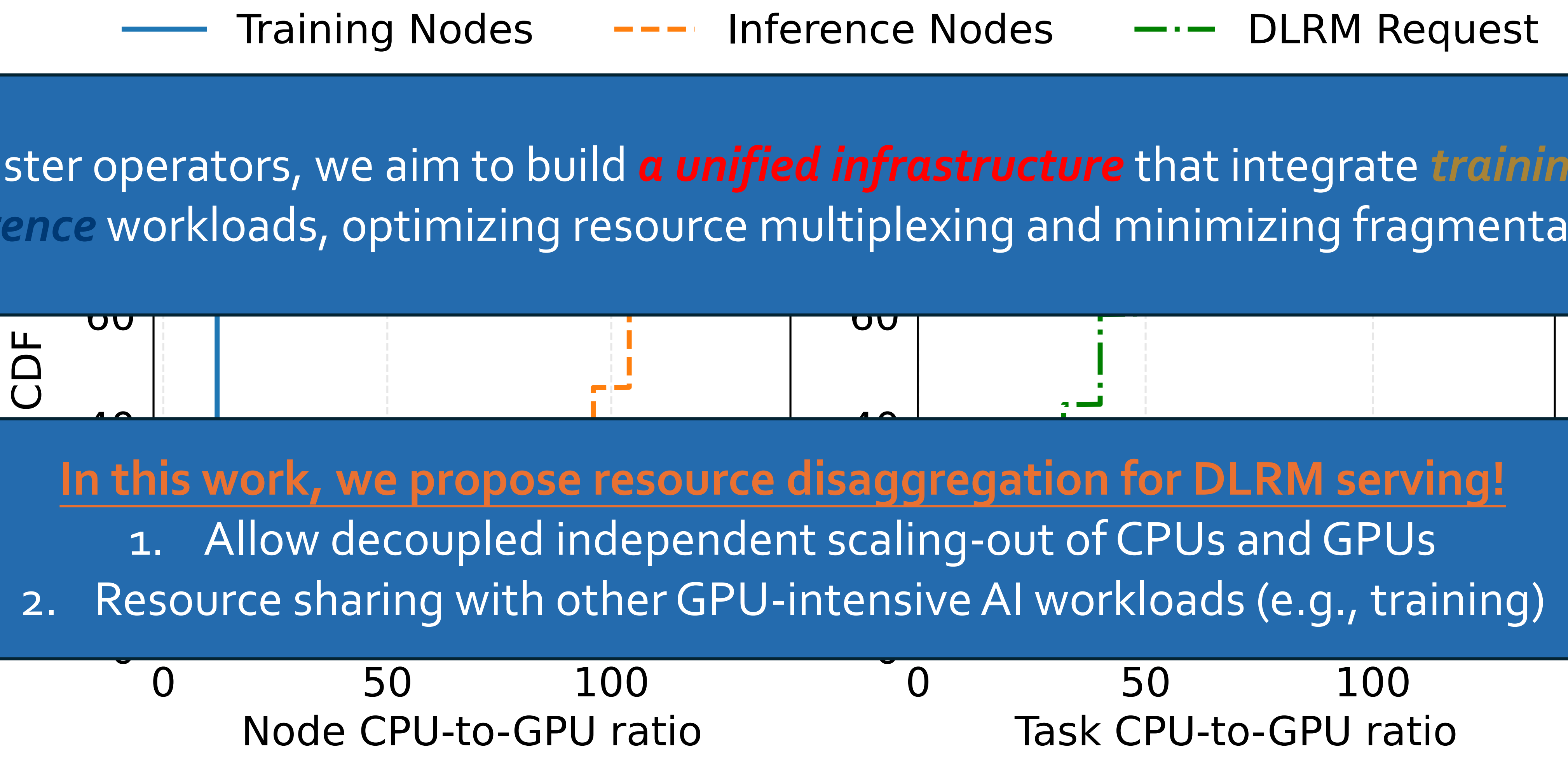
Placing DLRMs in the training nodes leaves many stranded GPUs!

Resource Heterogeneity in GPU Clusters



As cluster operators, we aim to build ***a unified infrastructure*** that integrate ***training*** and ***inference*** workloads, optimizing resource multiplexing and minimizing fragmentation.

Resource Heterogeneity in GPU Clusters



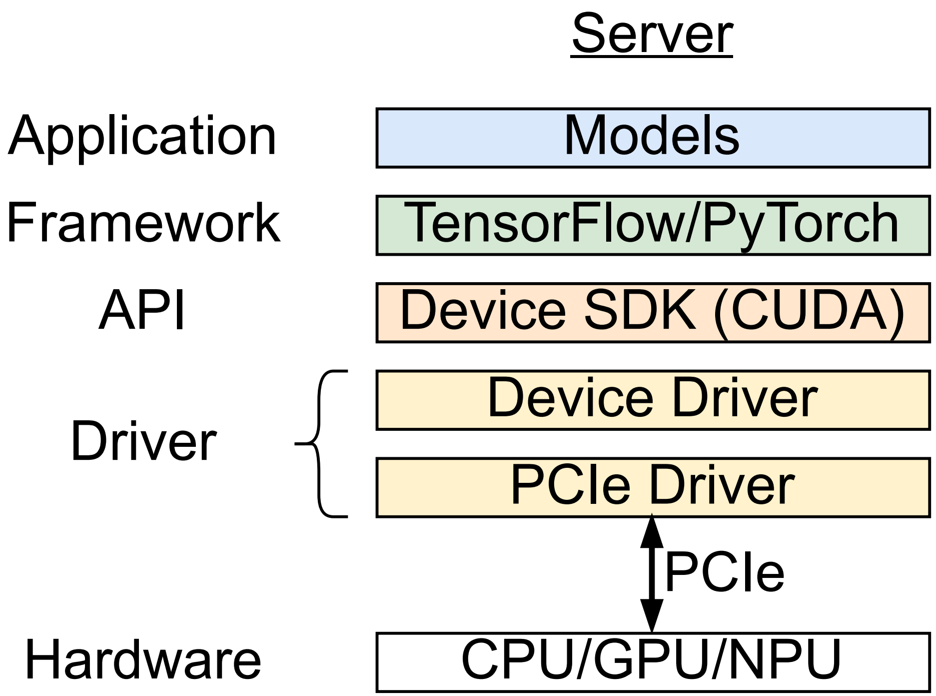
As cluster operators, we aim to build ***a unified infrastructure*** that integrate ***training*** and ***inference*** workloads, optimizing resource multiplexing and minimizing fragmentation.

In this work, we propose resource disaggregation for DLRM serving!

1. Allow decoupled independent scaling-out of CPUs and GPUs
2. Resource sharing with other GPU-intensive AI workloads (e.g., training)

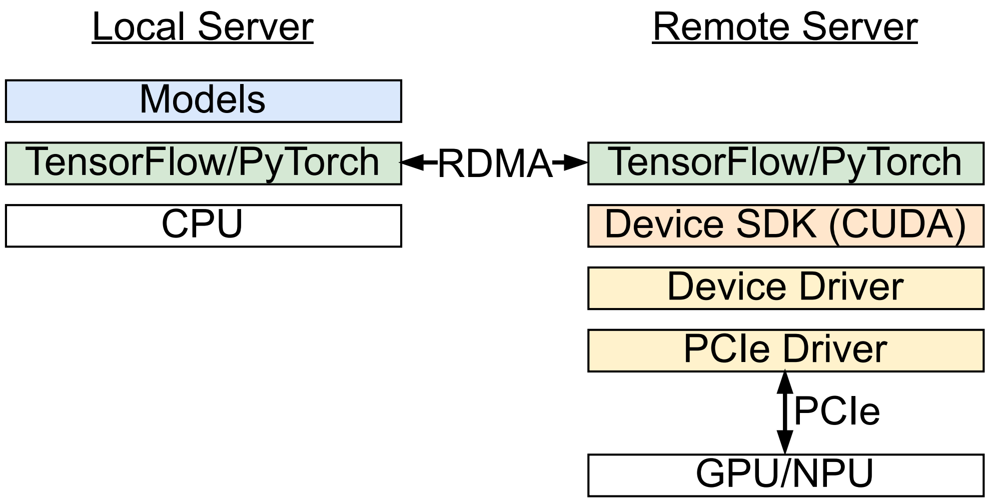
Approaches to GPU Disaggregation

- GPU disaggregation at different levels
  - Graph-level disaggregation
    - Partition the compute graph into a CPU sub-graph and a GPU sub-graph
    - Schedule sub-graphs on selected CPU and GPU nodes for disaggregated execution
  - API-level disaggregation
    - Intercept program calls to CUDA APIs (rCUDA [HPCS'10])
    - Redirect them to a remote GPU node for execution
  - Hardware-level disaggregation
    - Enabled with specialized hardware
    - Examples: customized multi-hop PCIe switches (DxPU [TACO'23]) and CXL 3.0



Approaches to GPU Disaggregation

- GPU disaggregation at different levels
  - Graph-level disaggregation ✅
  - API-level disaggregation (rCUDA [HPCS’10])
  - Hardware-level disaggregation (DxPU [TACO’23])



- Design considerations
  - DLRM exhibits distinct resource consumption
  - Easy to support heterogeneous AI accelerators
  - Adapt to the existing infrastructure (i.e., no specialized hardware, high-bandwidth RDMA)

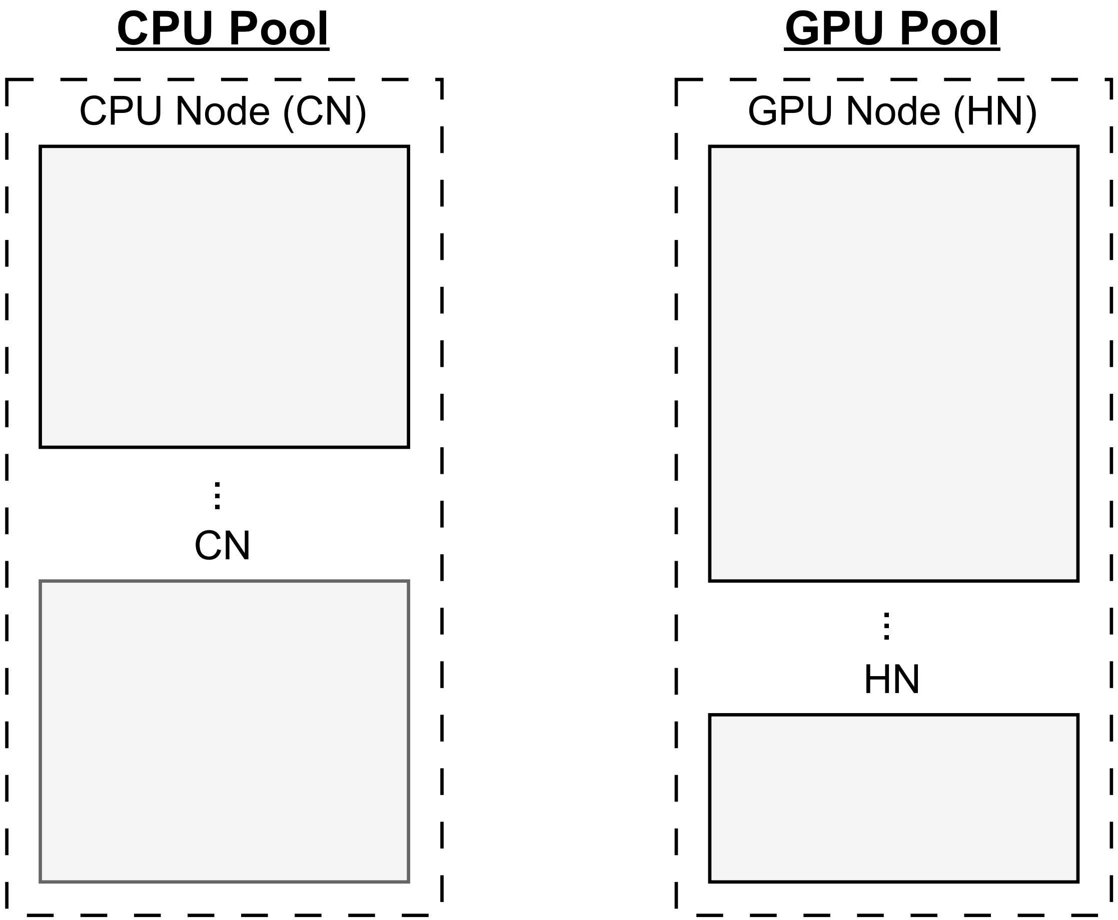
Prism Overview

- Prism is a large-scale DLRM system that enables GPU-disaggregated serving by means of graph partitioning

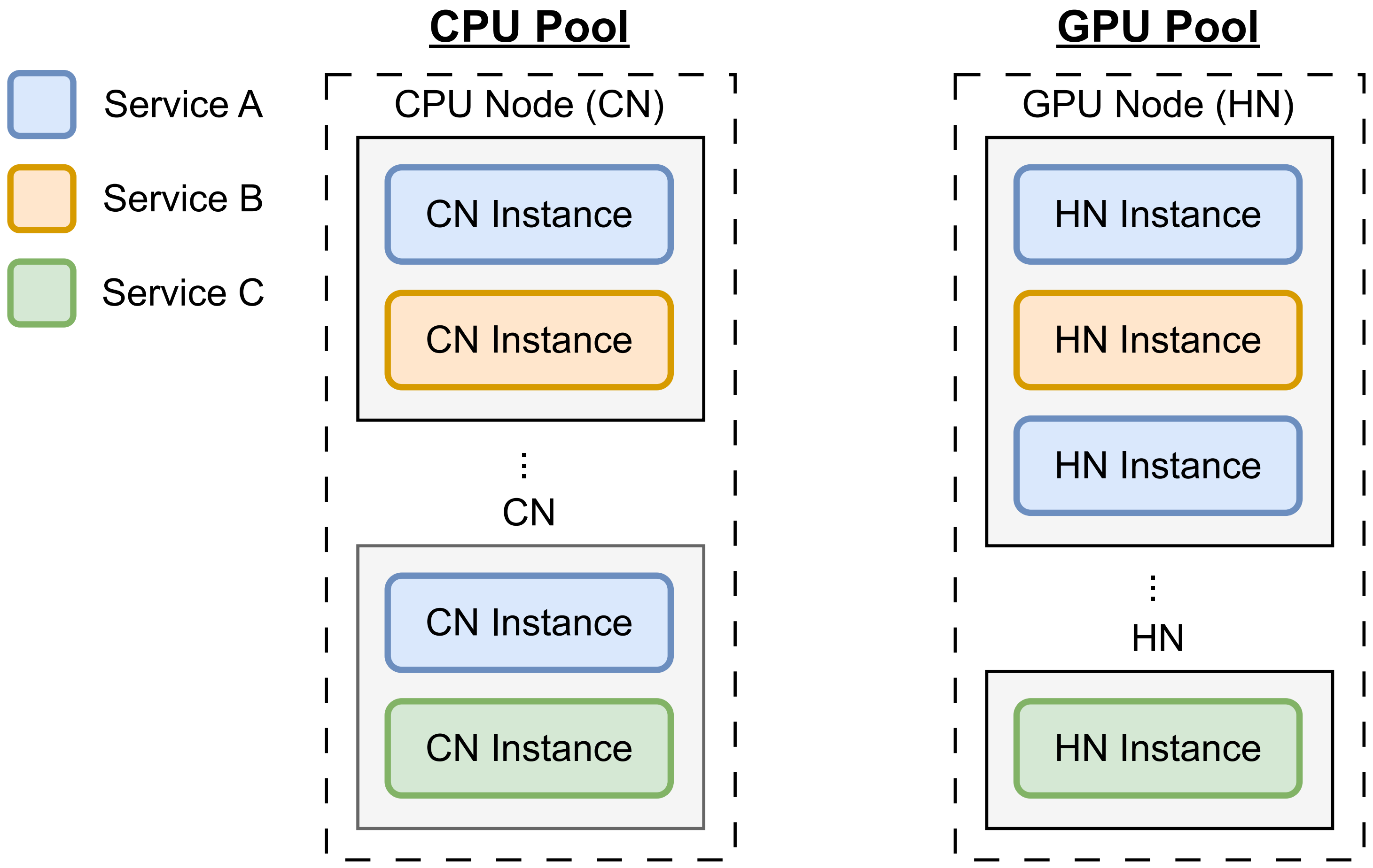
- Prism operates on a cluster where a fleet of heterogeneous GPU nodes (HNs) interconnects with a number of CPU nodes (CNs) via a high-speed RDMA network; automatically partitions recommendation models for distributed inference on CNs and HNs

- Prism has been deployed in production clusters for over two years and now runs over 10k GPUs

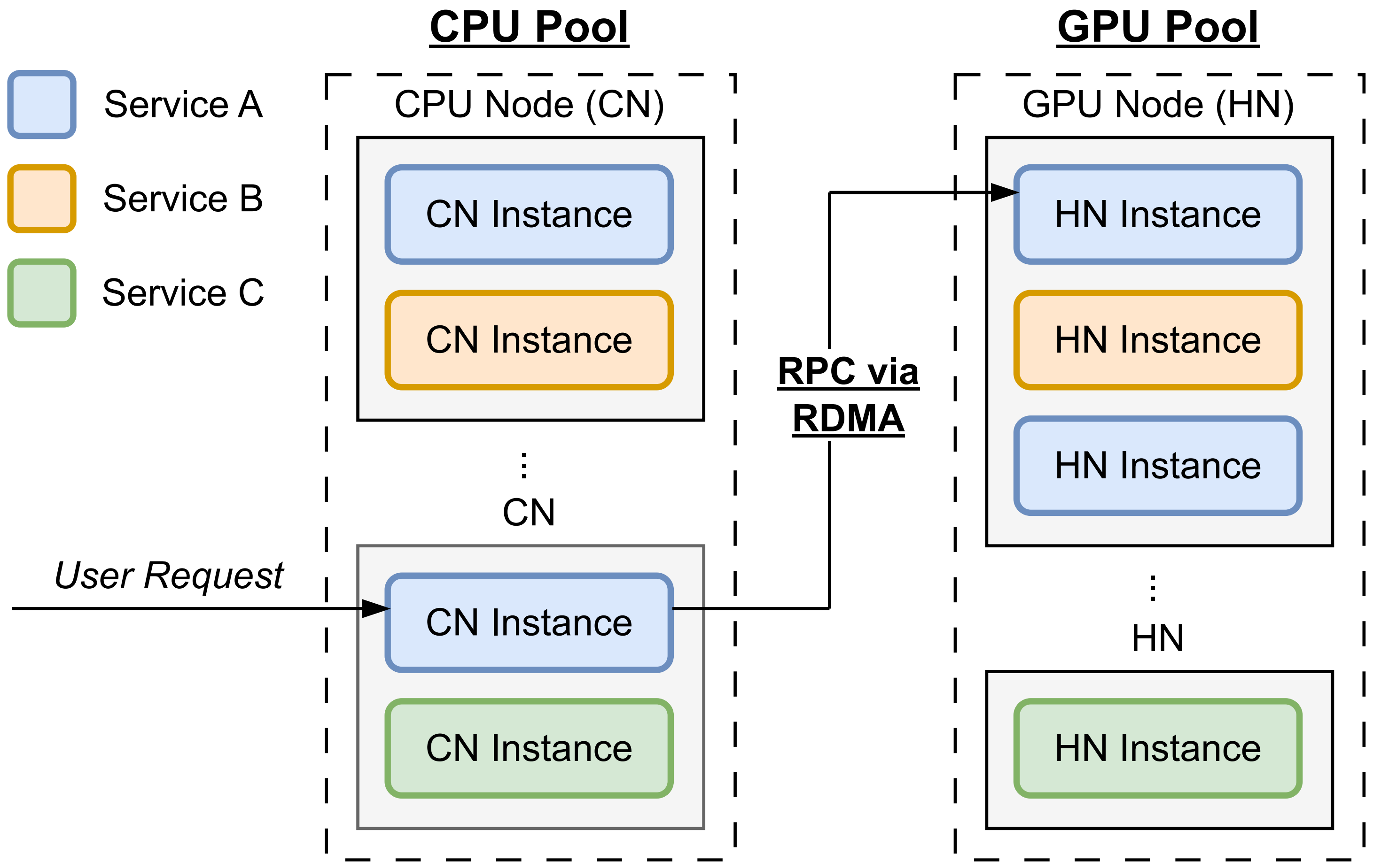
Rectified Execution Flow with **Prism**



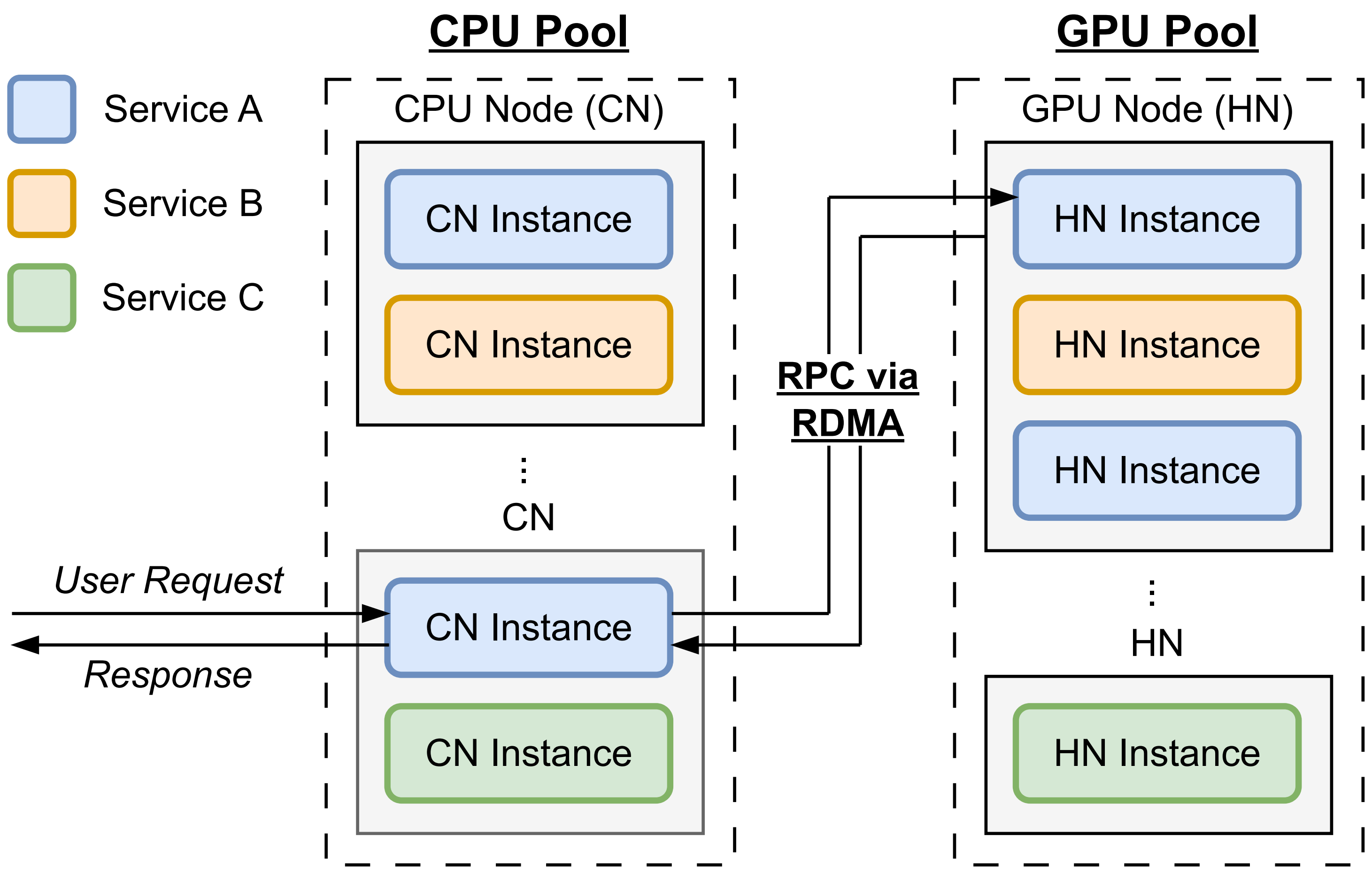
Rectified Execution Flow with Prism



Rectified Execution Flow with Prism



Rectified Execution Flow with **Prism**



Requirements of Disaggregated DLRM Serving

Requirements of Disaggregated DLRM Serving

- Transparency to model development and optimization
  - Automated graph partitioning to support disaggregated inference for various recommendation models
  - Don't affect users or invalidate original graph optimization strategies

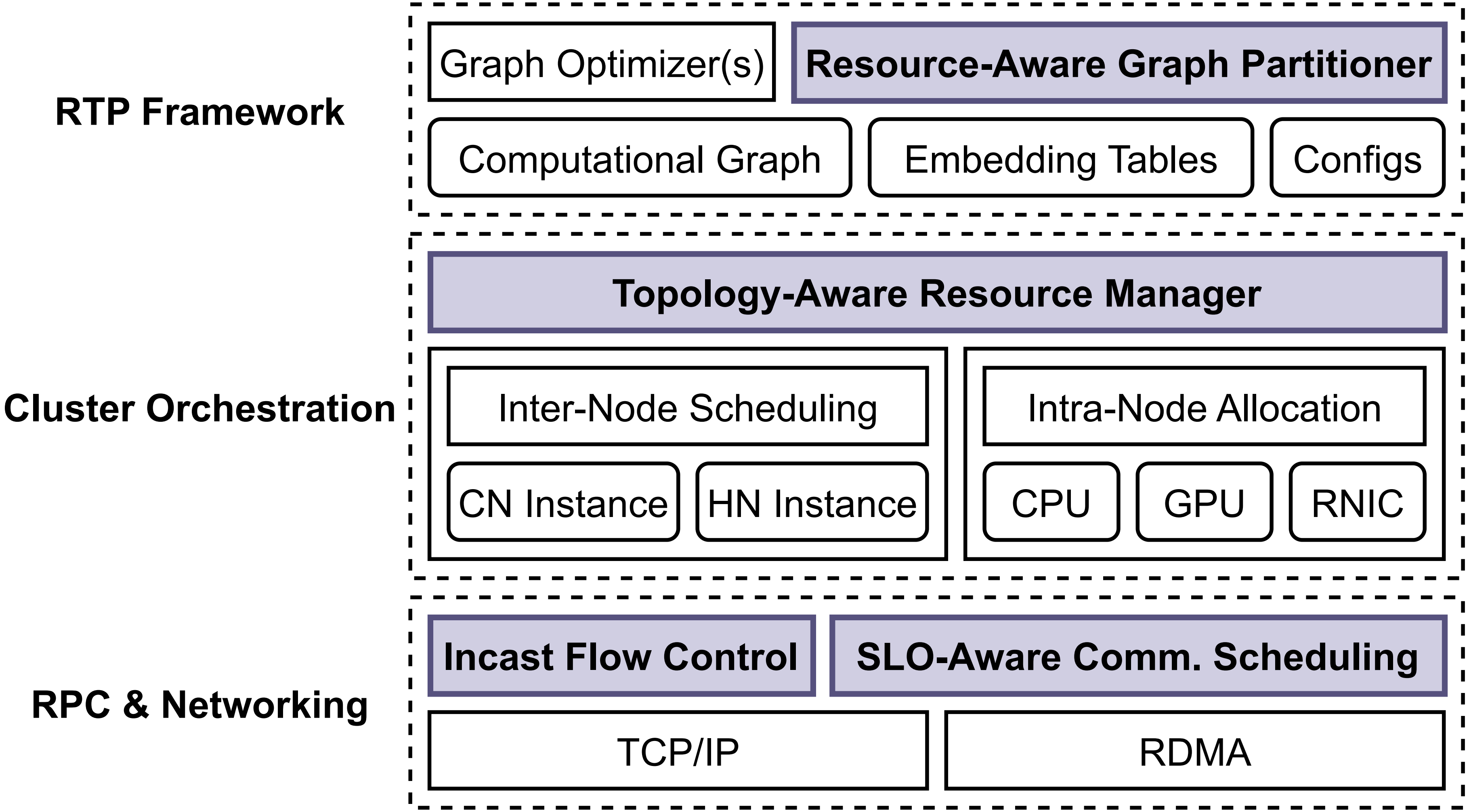
Requirements of Disaggregated DLRM Serving

- Transparency to model development and optimization
  - Automated graph partitioning to support disaggregated inference for various recommendation models
  - Don’t affect users or invalidate original graph optimization strategies
- Compliance to SLOs (e.g., 20ms latency)
  - Require a joint optimization approach across various system components to minimize the impact on service performance

Requirements of Disaggregated DLRM Serving

- Transparency to model development and optimization
  - Automated graph partitioning to support disaggregated inference for various recommendation models
  - Don’t affect users or invalidate original graph optimization strategies
- Compliance to SLOs (e.g., 20ms latency)
  - Require a joint optimization approach across various system components to minimize the impact on service performance
- Good scalability
  - Ensure service performance remains unaffected, even under conditions of high traffic loads in production environments

System Components in Prism



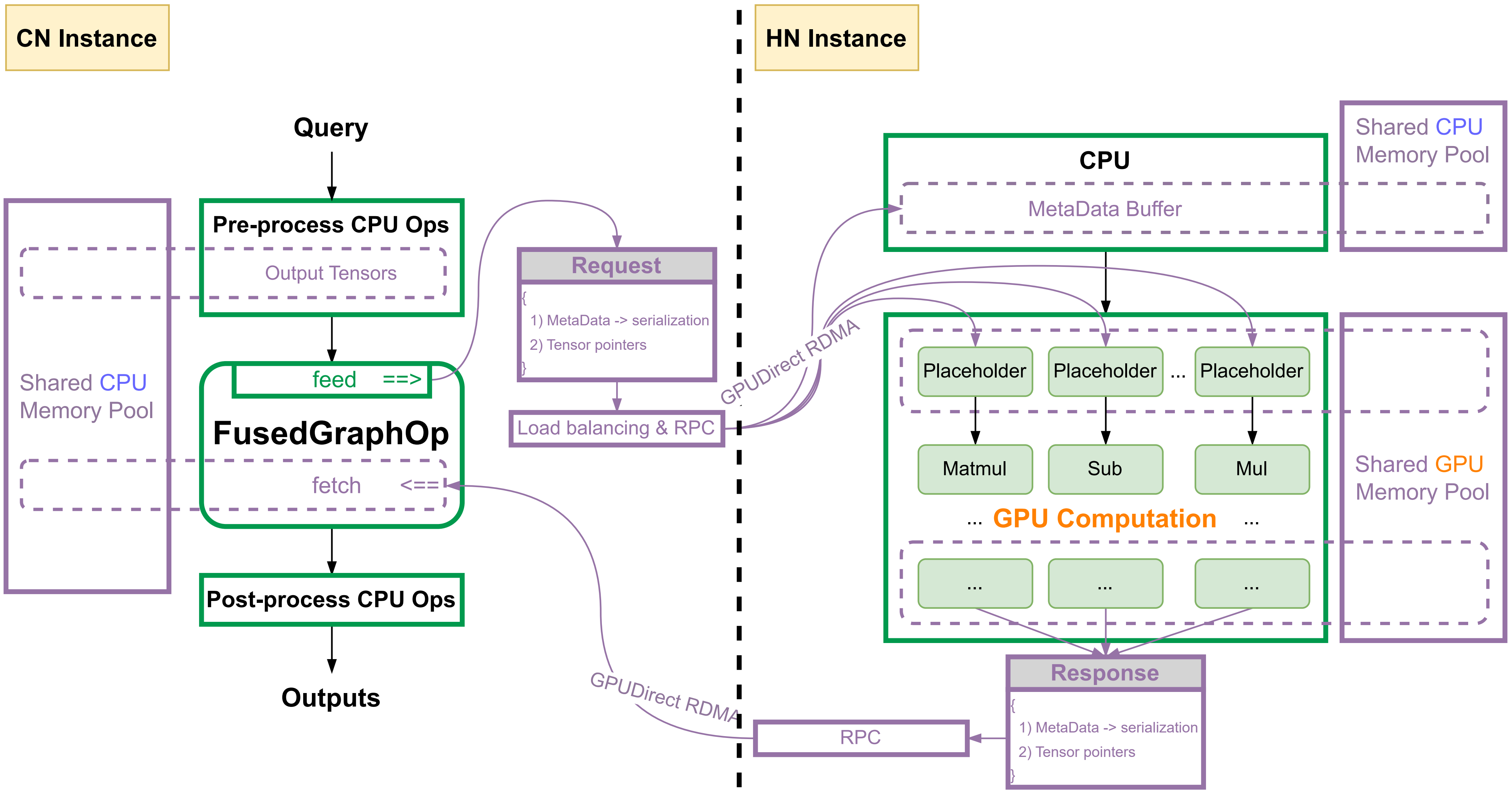
① Resource-Aware Graph Partitioner

- Retrofit for the existing workflow
  - Existing optimizers are typically applied *in a sequential manner* → Rewrite the original computation graph and generate an optimized computation graph tailored for deployment
  - *Graph partitioning* and *disaggregation optimization* serve as the final stages

① Resource-Aware Graph Partitioner

- Retrofit for the existing workflow
  - Existing optimizers are typically applied *in a sequential manner* → Rewrite the original computation graph and generate an optimized computation graph tailored for deployment
  - *Graph partitioning* and *disaggregation optimization* serve as the final stages
- Employ a heuristic approach to split the GPU subgraph
  - Offline profiling and operator categorization → CPU-intensive ops (e.g., embedding table lookup) and GPU-efficient ops (e.g., MatMul, Attention)
  - Perform a DFS coloring process to encompass the maximum number of operators feasible for GPU computation
- In 80% of DLRM services, RDMA transfer size per request < 10 MiB

① Resource-Aware Graph Partitioner



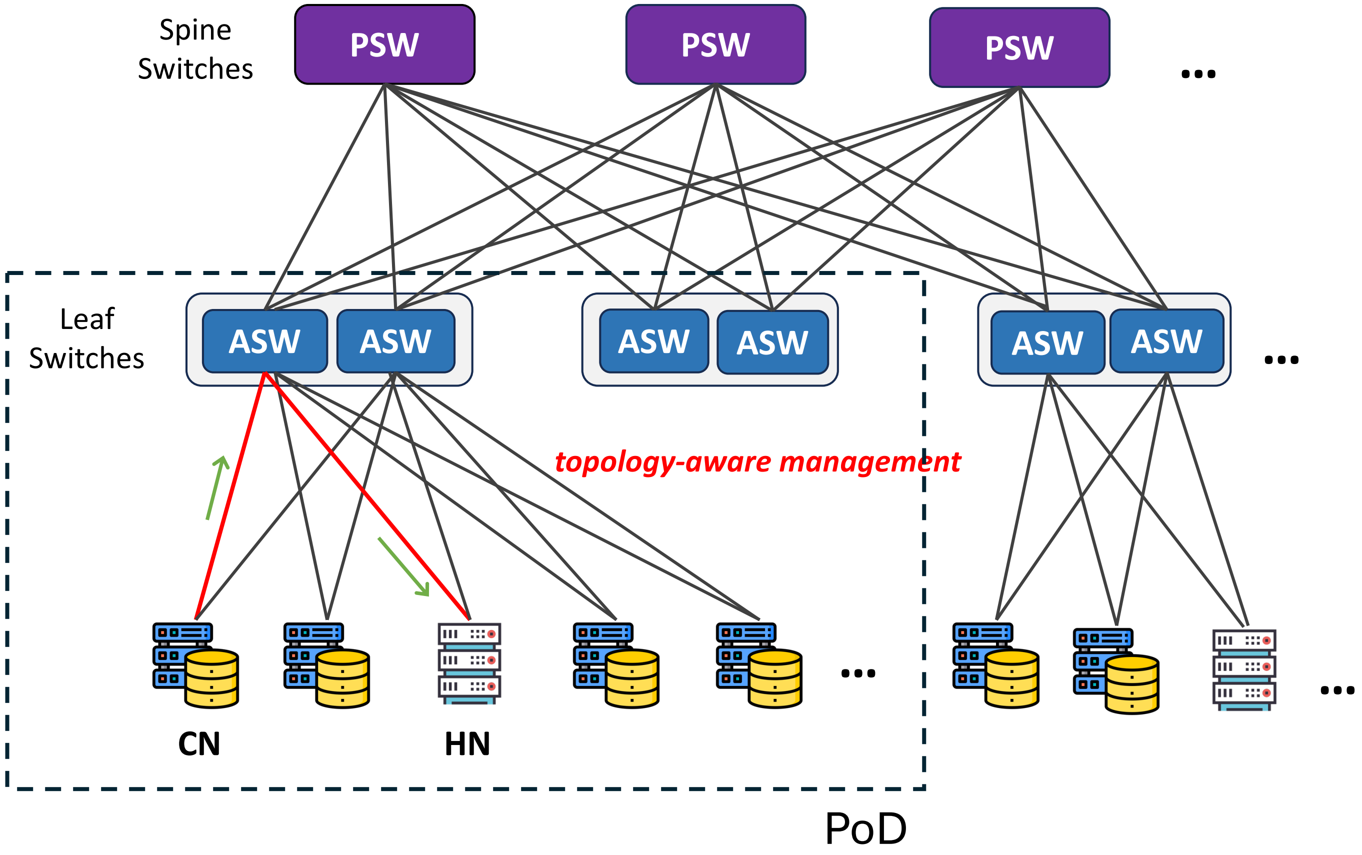
② Topology-Aware Resource Manager

- Place a group of CN and HN instances into a shared cluster
- Different role of instances can be scaled independently
- Principle: <u>topology-aware</u> node scheduling and resource allocation

② Resource Manager: Inter-node scheduling

- Two policies
  - Confine all instances within the same PoD
  - Schedule new instance(s) to the ASW with the most existing instances

- Deployment constraints at different levels
  - Node
  - NIC switch

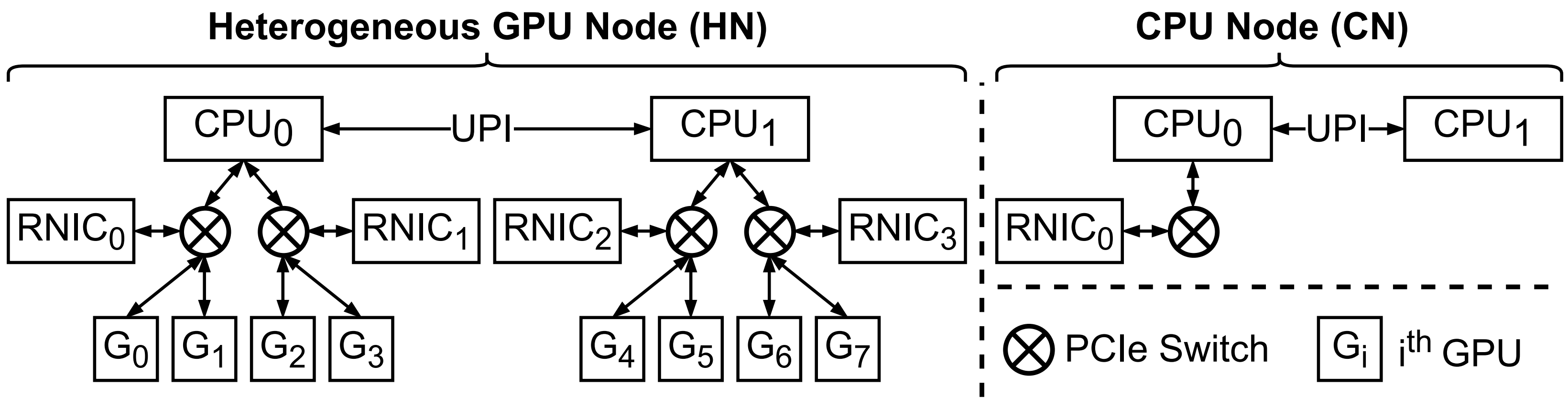


For more details of data center network, please refer to:
"Alibaba HPN: A Data Center Network for Large Language Model Training", SIGCOMM'24

② Resource Manager: Intra-node allocation

- HN instance
  - Arbitrary bindings of GPUs and RNICs can induce 21–36% performance loss
  - Assign RNIC and GPU on the same PCIe switch; enable GPUDirect RDMA

- CN instance
  - Prioritize CPU allocation under the same PCIe switch connected to the RNIC



③ SLO-Aware Communication Scheduler

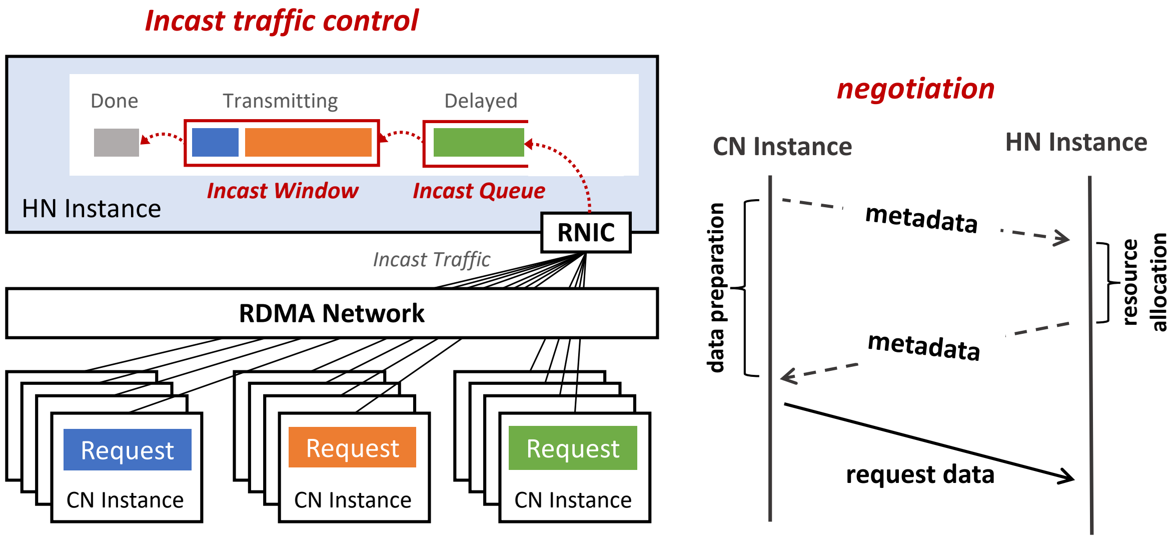
- Extend the native RoCEv2 stack and implement a middleware to leverage RDMA capabilities in a *virtualized* environment

- Incast: A substantial number of CN instances concurrently transmit data to a limited number of HN instances

Incast Size	**Latency**	**% of Failed Requests**
10	10 ms	$\approx 33\%$
20	40 ms	$\approx 50\%$
100	10 s	$\approx 100\%$

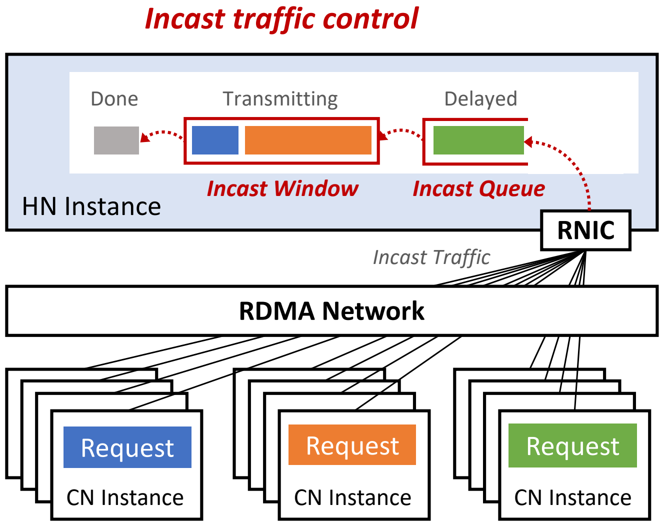
③ SLO-Aware Communication Scheduler

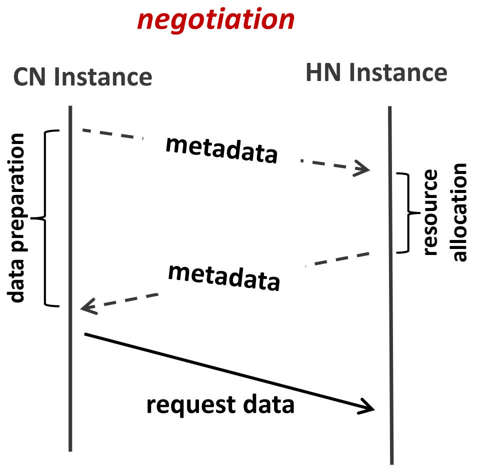
- Adaptive incast window
  - Adapt to the congestion level of network links and PCIe links
  - The number of CNPs serves as an estimator of the congestion level



③ SLO-Aware Communication Scheduler

- Adaptive incast window
  - Adapt to the congestion level of network links and PCIe links
  - The number of CNPs serves as an estimator of the congestion level
- Deadline-aware request scheduling
  - The <u>deadline</u> of a comm request: the latest time to initiate parameter transmission to meet the SLO
  - <u>Reorder</u> the requests in the incast queue to maximize the number of requests meeting their SLOs



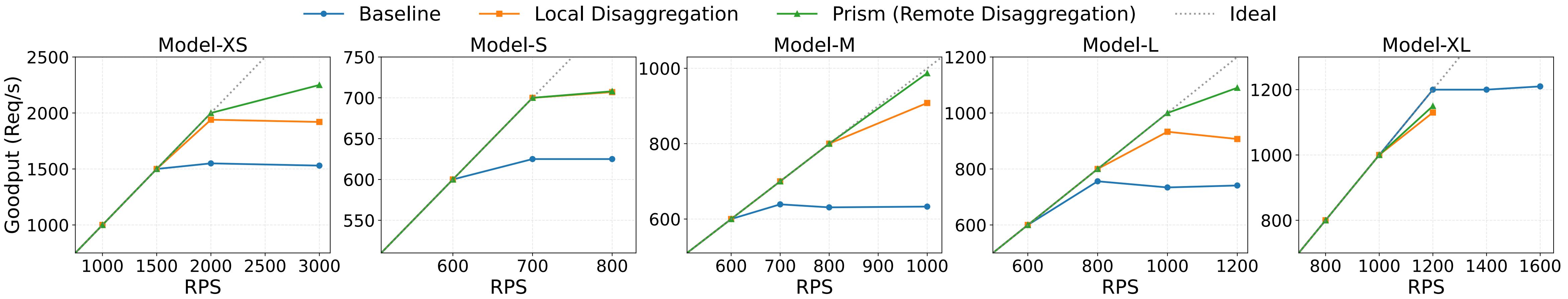


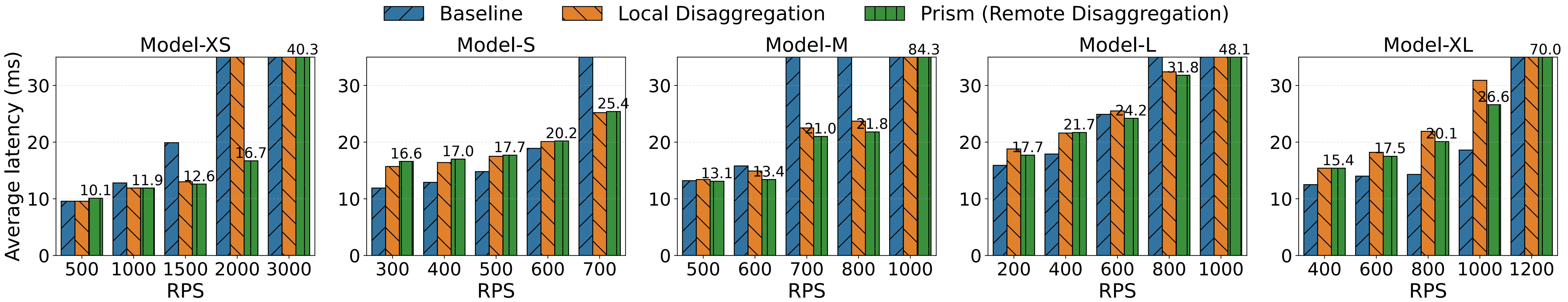
Evaluation

- Production workloads
- Machine specifications
  - CPU node (CN)
    - **128** vCPU cores
    - **1** 200 Gbps RNIC
  - GPU node (HN)
    - **128** vCPU cores
    - **8** A100 GPUs with 80 GiB GPU memory each
    - **4** 200 Gbps RNICs
  - All nodes use Intel(R) Xeon(R) Platinum 8369B CPUs, with **1024 GiB** memory

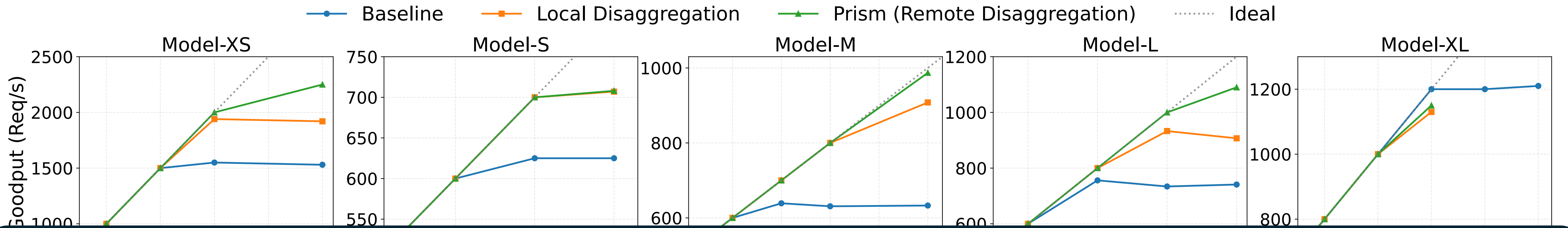
Model	**Emb Size** (Approximate)	**RDMA TX** (Per Req)	**Dense Features**
Model-XS	100 GiB	552.96 KiB	338.67 MiB
Model-S	450 GiB	6.84 MiB	57.20 MiB
Model-M	500 GiB	3.87 MiB	21.46 MiB
Model-L	600 GiB	3.69 MiB	20.79 MiB
Model-XL	700 GiB	9.03 MiB	8.73 GiB

Performance under varying traffic loads

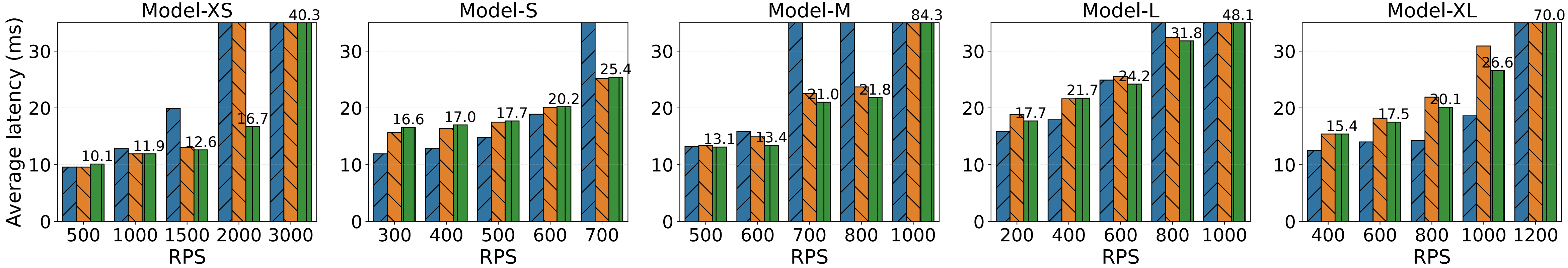




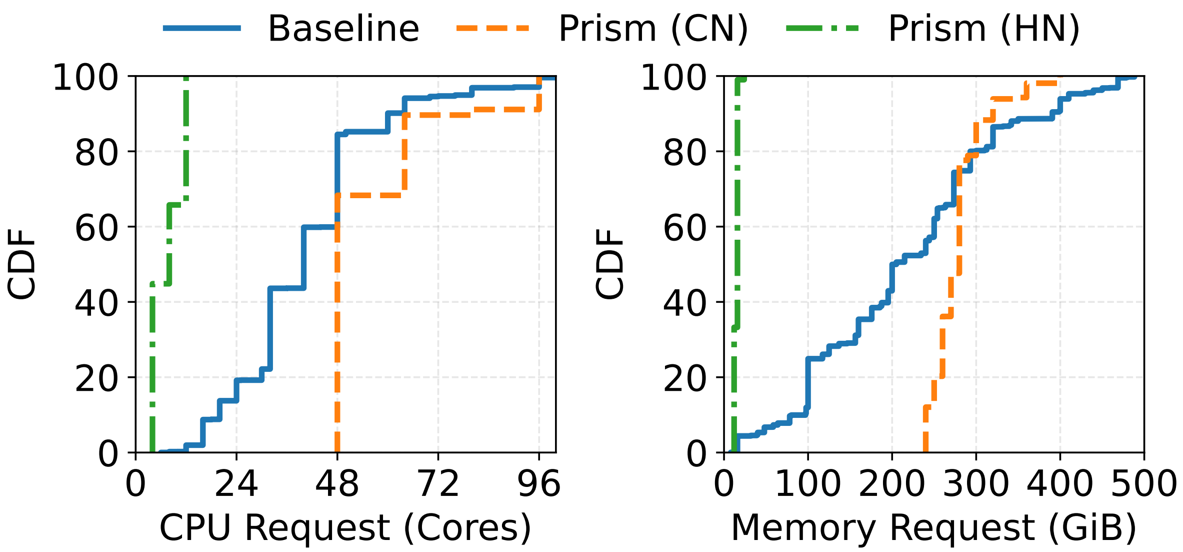
Performance under varying traffic loads



Prism can maintain service performance under <u>high traffic</u> scenarios!

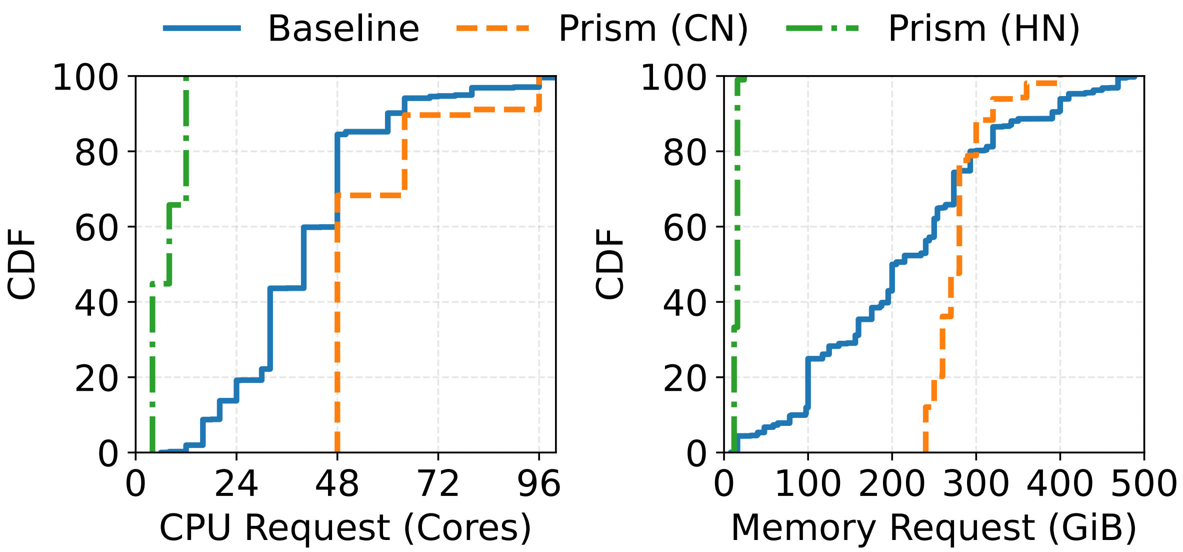


Mitigated Resource Fragmentation



- HN instances that require GPU allocation, their CPU requests are < 12 cores, and memory requests < 24 GiB
- CN instances have CPU requests > 48 cores and memory requests > 240 GiB

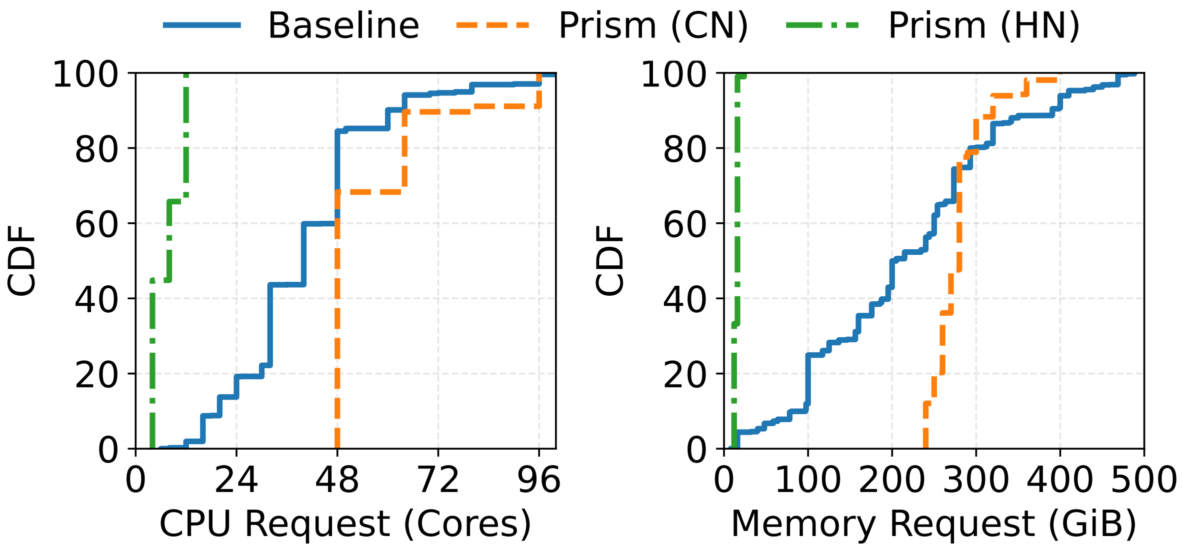
Mitigated Resource Fragmentation

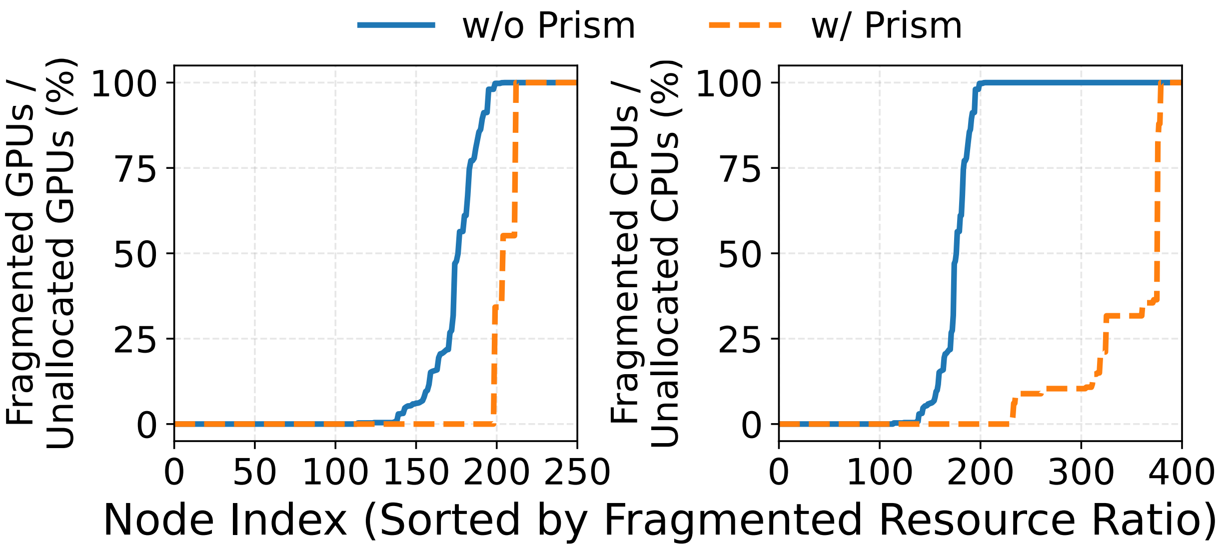


- HN instances that require GPU allocation, their CPU requests are < 12 cores, and memory requests < 24 GiB
- CN instances have CPU requests > 48 cores and memory requests > 240 GiB

Prism can separates resource requirements of DLRM inference services.

Mitigated Resource Fragmentation

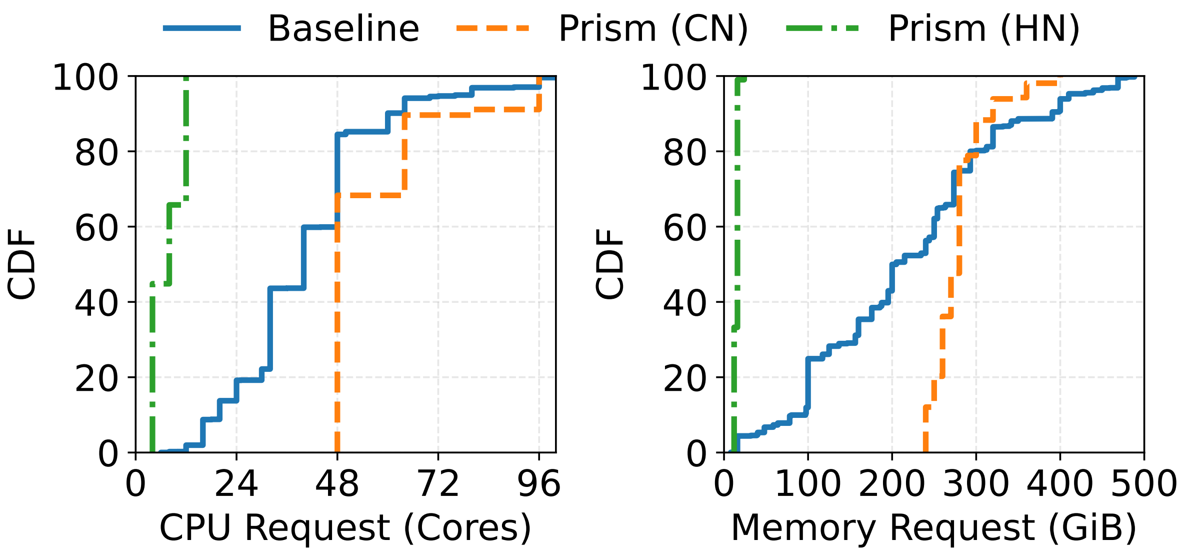


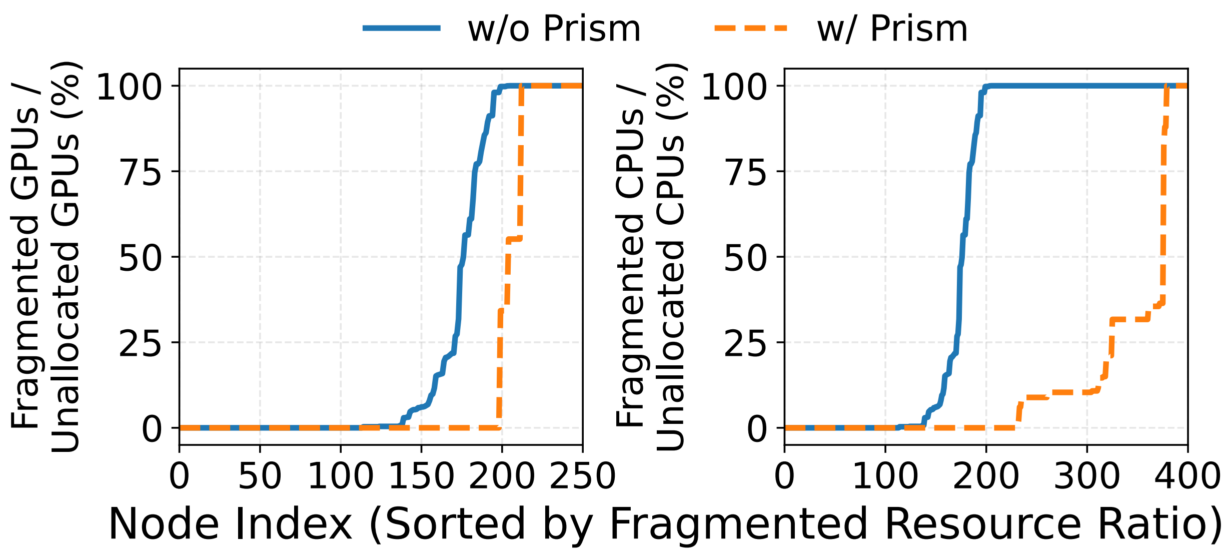


- HN instances that require GPU allocation, their CPU requests are < 12 cores, and memory requests < 24 GiB
- CN instances have CPU requests > 48 cores and memory requests > 240 GiB

Prism reduces the cluster's CPU fragments by 53% (18k cores) and GPU fragments by 27% (60 GPUs).

Mitigated Resource Fragmentation





- HN instances that require GPU allocation, their CPU requests are < 12 cores, and memory requests < 24 GiB
- CN instances have CPU requests > 48 cores and memory requests > 240 GiB

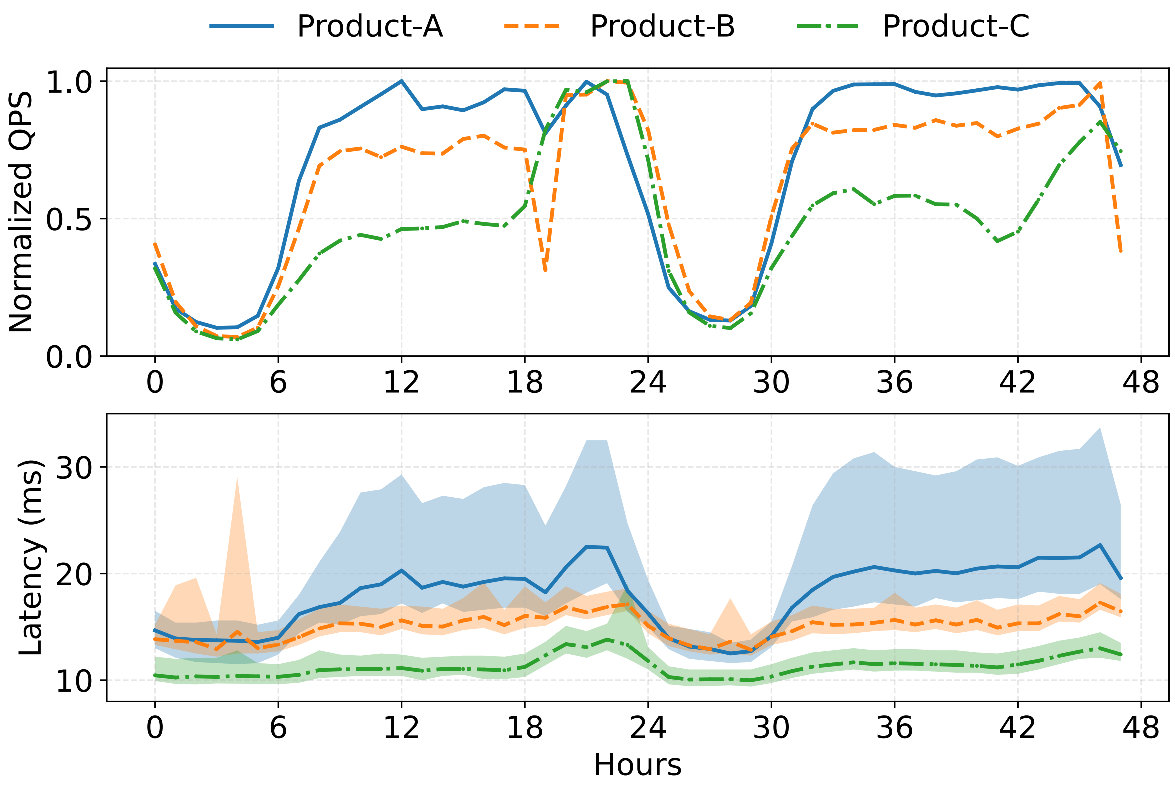
Prism reduces the cluster's CPU fragments by <u>53%</u> (18k cores) and GPU fragments by <u>27%</u> (60 GPUs).

Prism can effectively reduce the cluster's fragmented resources.

Efficient Resource Loans for Peak Demand

- During e-commerce promotional events, Prism can borrow a portion of training nodes to scale out DLRM inference services

Service	Role	# of Instances	CPU	GPU
Product-A	CN	25	48	-
	HN	45	4	MIG 2g.20gb
Product-B	CN	15	48	-
	HN	40	4	MIG 2g.20gb
Product-C	CN	15	48	-
	HN	55	2	MIG 2g.20gb



Discussion and Future Explorations

- Disaggregated serving for different workloads
  - LLM PD disaggregation: decouple the GPU computation and I/O bandwidth
  - DLRM disaggregation: decouple the use of CPU and GPU computation
  - Transform the workload from the perspective of resource provisioning
  - Decouple *diverse* resource requirements to accommodate the infrastructure

Discussion and Future Explorations

- Disaggregated serving for different workloads
  - LLM PD disaggregation: decouple the GPU computation and I/O bandwidth
  - DLRM disaggregation: decouple the use of CPU and GPU computation
  - Transform the workload from the perspective of resource provisioning
  - Decouple *diverse* resource requirements to accommodate the infrastructure
- Fault tolerance
  - Performance isolation between networking resources
  - Dense instance deployment on a single node → High blast radius!

Takeaways

- Prism enables DLRMs to harvest resources from CPU nodes and heterogeneous GPU nodes by means of **disaggregated serving**

- Prism effectively **mitigates resource fragmentation** in daily high-allocation GPU clusters; and enables efficient **capacity loaning** from training clusters during seasonal promotion events

- Prism has been deployed in production clusters for <u>over two years</u> and now runs <u>over 10k GPUs</u>

A production DLRM serving trace is released at: https://github.com/alibaba/clusterdata/tree/master/cluster-trace-gpu-v2025