Heterogeneity at Hyperscale

Characterization and Scheduling of Large Production AI Clusters at Alibaba

Suyi Li*, **Lingyun Yang***, Haoxuan Yu, Sheng Yao, Tianyuan Wu, Xiaoxiao Jiang, Hanfeng Lu
Kangjin Wang, Chenhao Wang, Shenglin Xu, Lun Wang, Qingyang Duan, Shenghao Liang, Xiu Lin
Meng Zhang, Wenchao Wu, Yinghao Yu, Guodong Yang, Liping Zhang, Wei Wang
(*Equal contribution)







Motivation

Heterogeneous Workloads

GenAI (LLMs, video) + classical DNNs (CTR, Rec) served as first-class citizens in shared clusters

Heterogeneous Hardware

Multi-vendor, multi-generation GPUs coexist: NVIDIA, AMD, and custom XPUs with different specs

The Central Paradox

High GPU demand does NOT yield high effective utilization — idle GPUs exist despite long wait queues

This talk: A 6-month study of 155,410 GPUs — diagnosing why, and what we did about it.

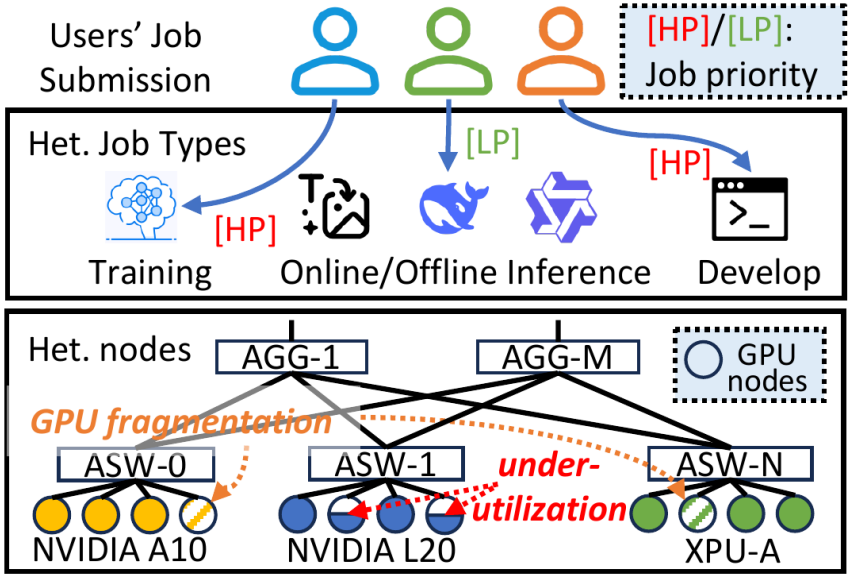
ASI: Alibaba Serverless Infrastructure

155,410 GPUs across 37,707 nodes

81 Internal departments served

14M Jobs over 6 months

11+ GPU models (e.g., NVIDIA/AMD GPUs)



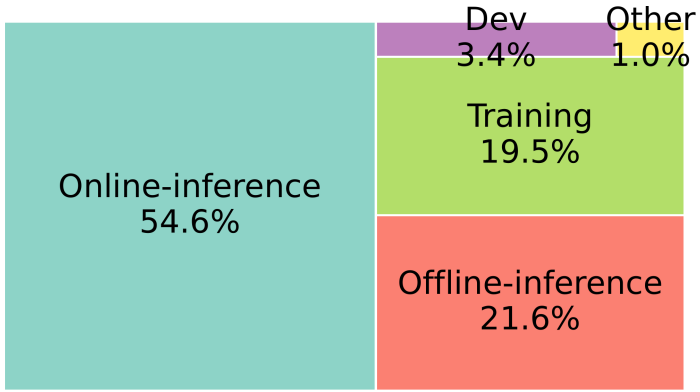
Fat-tree topology: 8-GPU nodes → ASW (256–512 GPUs) → AGG

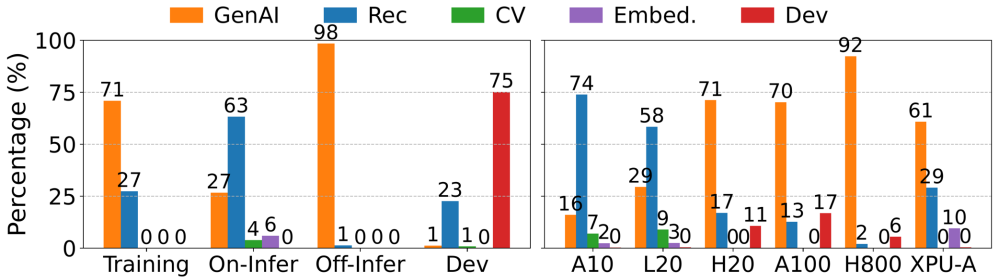
The ASI GPU Trace — Largest Public Dataset

	ASI	Acme (Shanghai AI Lab)	PAI (Alibaba)	Helios (SenseTime)
Year	**2025**	2023	2020	2020
Duration	**6 months**	6 months	2 months	6 months
# Jobs	**14M**	1.09M	1.26M	3.36M
# GPUs	**155,410**	4,704	6,742	6,416
GPU Models	**H20, H800, A10, A100, L20, XPU…**	A100	T4, P100, V100	1080Ti, V100
Job Types	**Train, Dev, Online/Offline Infer**	Train, Dev, Eval	—	—
Model Types	**LLM, DM, DNN, Rec**	LLM	DNN	DNN
Priorities	**High / Low (Spot)**	—	—	—

Publicly released: github.com/alibaba/clusterdata/tree/master/cluster-trace-gpu-v2026

Workload Composition

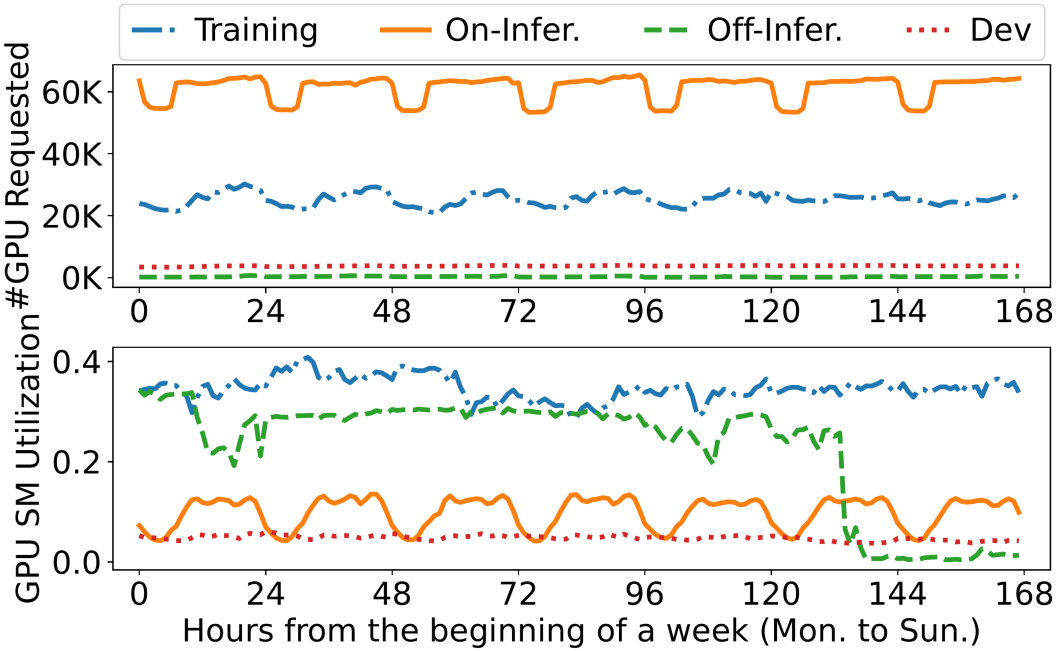


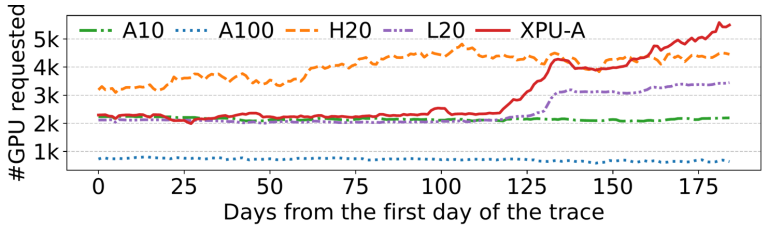


Key Observations

- **Online inference** dominates (>50% of GPU-hours)
- **GenAI** dominates training (71%) and offline-infer (98%)
- >99% of jobs pin a specific GPU model
- GPU sharing (fractional GPU) is negligible

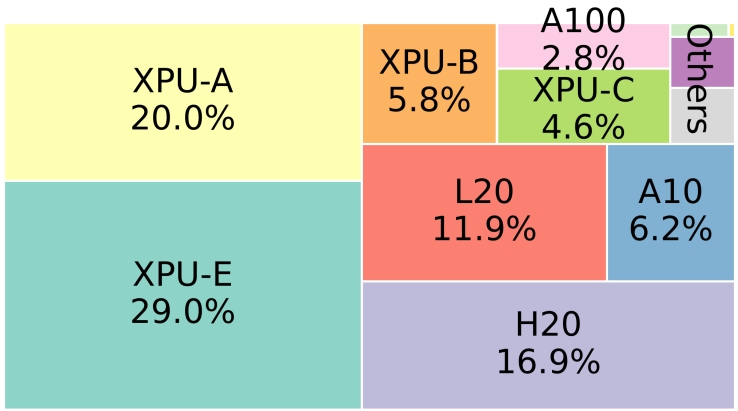
Temporal Patterns





- Strong diurnal cycles in GPU demand
- Median job execution: 5 hours (vs 2 min in Acme)
- Median scheduling latency: 1 second
- XPU-A adoption rises sharply after Day 120 (optimization release)

GPU Request Sizes: Bigger Than Ever



Growing request sizes

2 GPUs

ASI median
mean: 11

1 GPU

PAI median
mean: 2.6

Resource Utilization Reality

- **Online-inference: median SM utilization only 6%**
- GenAI serving uses 94% GPU memory but only 5% SM — severely SM-underutilized
- Training tasks have the highest utilization across all metrics

High GPU Demand ≠ High Utilization

Despite long wait queues, idle GPUs exist and cannot be allocated.

1

Stranded GPUs

Partial node allocation leaves unused GPUs

2

CPU Shortage

CPU/GPU ratio mismatch blocks GPU allocation

3

Topology

ASW-locality constraints for large GenAI jobs

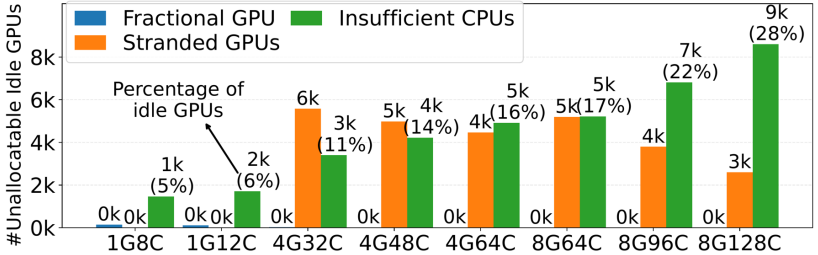
4

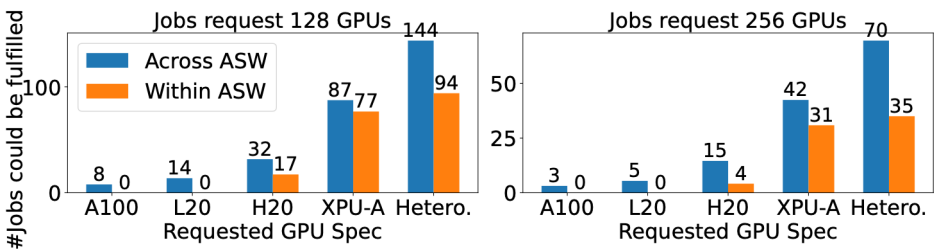
Headroom

Production reservation for burst capacity

Note: Fractional GPU fragmentation is negligible — GPU sharing is rarely used in practice.

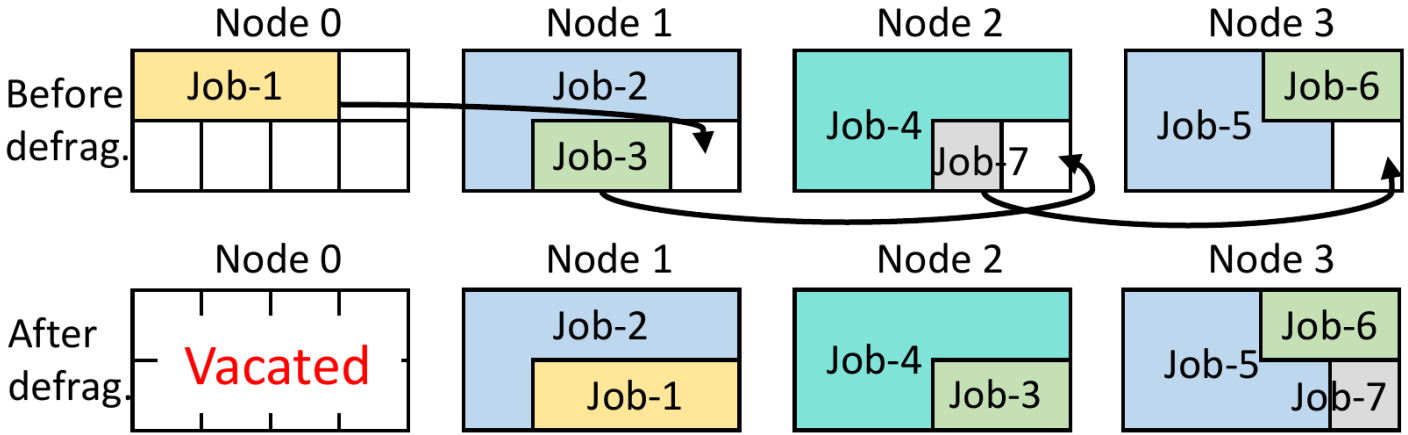
GPU Fragmentation: Quantified





- Fractional GPU fragmentation is negligible; **stranded capacity and CPU mismatches** are the dominant causes.
- Topology-induced fragmentation (ASW-locality) is a newly identified contributor for large GenAI workloads.
- **Heterogeneous** allocation retain more capacity under topology constraints.

IPC: Iterative Partitioned Consolidation



Divide & Conquer

Random partition of nodes into parallel subproblems

Ejection Chains

Recursive task migration (max depth K=3)

Iterative

Up to 5 rounds; diminishing returns after 3

Make-before-Break

New instance starts before old one terminates (zero downtime)

IPC: Deployment Results

20.2%

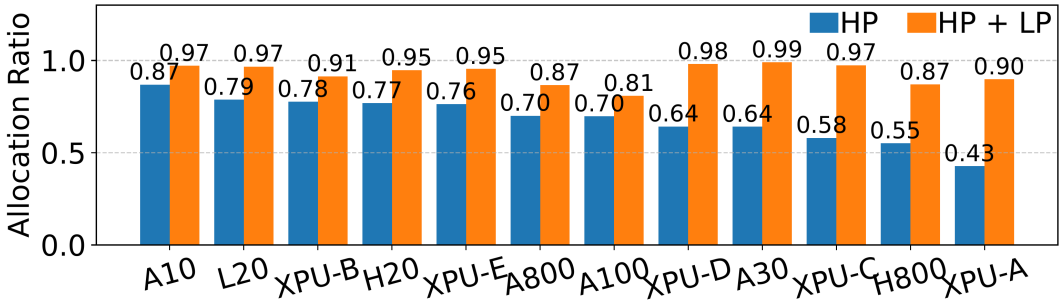
Reduction in nodes with slack resources

<2 min	Decision time for migration plans
40%	Of tasks are locked (cannot migrate)
2 days	ILP solve time for 100 nodes (impractical)

Topology-Aware GPU Allocation

- **Entropy-based metric** quantifies GPU distribution across ASWs (lower = more concentrated = better)
- Greedy algorithm: pick ASW that minimizes entropy each iteration
- **Same-ASW placement improves allreduce bandwidth by 27%**

SpotGPU: Reclaiming Idle HP Capacity



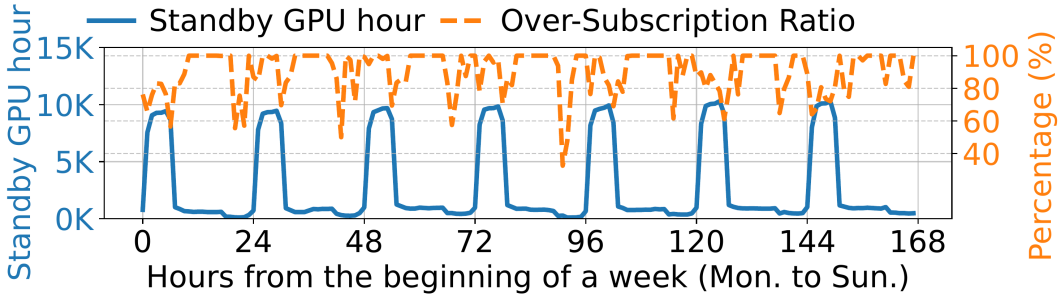
68% → 93%

GPU allocation ratio
(HP only → HP + LP Spot)

How SpotGPU Works

- HP users label tasks as **"Standby"** — releasing GPU while keeping container warm (replaces main process with sleep)
- LP (Spot) jobs run on reclaimed capacity at discounted rate
- When HP needs resources back: graceful eviction with 60-second termination window
- Preemptive scheduling: minimize wasted computation (preemption cost = GPUs × time since last checkpoint)

SpotGPU: Deployment Results



10K **GPU-hours** Peak standby at midnight

90% Standby harvest rate

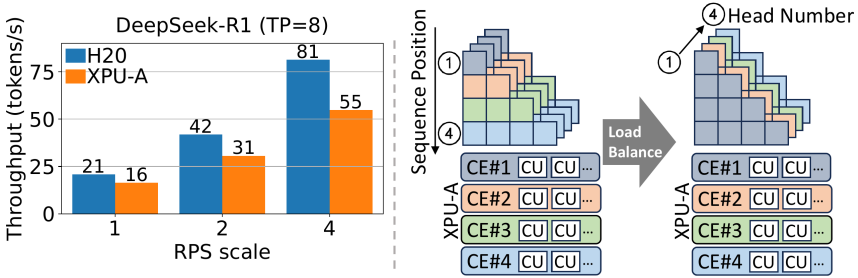
13s **(P95: 48s)** Average eviction time

Preemption-Cost-Aware Scheduling

- **Preemptive (MILP):** minimize total preemption cost = Σ(GPUs × time since last checkpoint). NP-hard in general; practical heuristic used in production.
- **Non-preemptive:** task packing, homogeneous co-location, eviction history — for jobs that cannot be preempted.
- *<5% of evictions require SIGKILL — graceful termination succeeds for >95% of tasks.*

Heterogeneous GPUs: The XPU-A Challenge

Spec	H20	XPU-A
Memory	96 GB	**192 GB**
HBM BW	4.0 TB/s	**5.2 TB/s**
FP16	148 TFLOPS	**203 TFLOPS**
Out-of-box	100% (baseline)	**80%**



The Problem

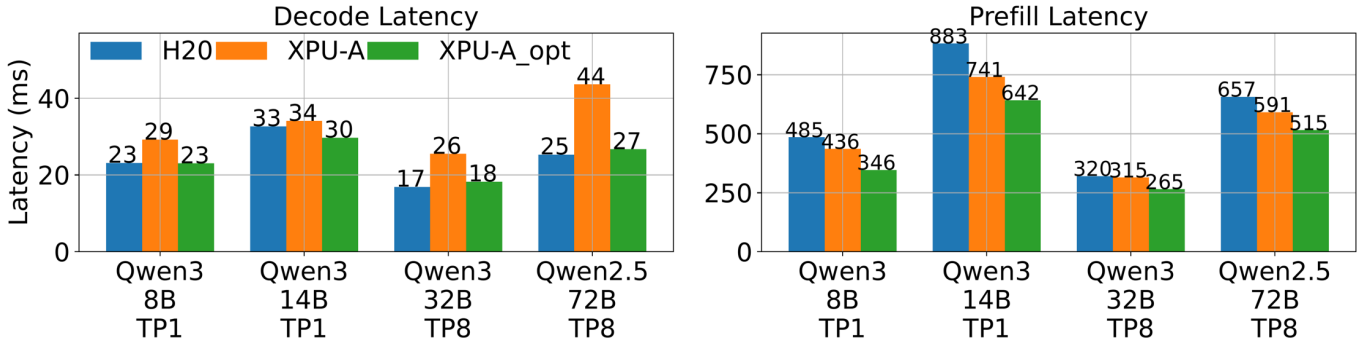
XPU-A has stronger specs than H20, but out-of-box GenAI performance is only 80% — due to kernel mismatches with hardware features.

Two Key Optimizations

Prefill (FlashAttention): Load imbalance across Compute Engines → parallelize along attention-head dimension → 1.58× speedup

Decoding (PagedAttention): Triton compiler inverts thread-block layout → custom compiler passes (contributed back to Triton)

XPU-A: Performance Benchmarks



Results After Optimization

- **Optimized XPU-A matches H20** for GenAI serving workloads across all tested models and configurations
- **vLLM latency reduction:** 33% at RPS=1, 43% at RPS=2 for DeepSeek-R1
- **HP demand for XPU-A increased 2.5×** after optimizations were deployed

Open Challenges

Topology × Heterogeneity

PD-disaggregated LLM serving (12K GPUs/day) requires both ASW-locality and heterogeneous GPU matching — two conflicting constraints. How to jointly optimize?

Online Inference Underutilization

Median SM: 6%, Memory: 30%. GenAI is memory-bound (94% mem, 5% SM) while DNN is light on both (20% mem, 6% SM). Complementary profiles, but colocation is hard.

CPU/GPU Interference

Colocating CPU-only jobs with GPU workloads reduces P90 SM utilization of GPU training by 18%. Safe colocation policies needed.

Conclusion

High GPU demand ≠ high effective utilization

Idle GPUs are stranded, CPU-blocked, topology-constrained, or reserved for headroom

IPC Defragmentation

20.2%

Reduction in nodes
with slack resources

SpotGPU Scheduling

68% → 93%

GPU allocation ratio
(24% faster LP tasks)

XPU-A Optimization

2.5×

HP demand increase;
up to 43% latency ↓

Trace: github.com/alibaba/clusterdata/tree/master/cluster-trace-gpu-v2026

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

Alibaba

FUDAN UNIVERSITY 1905

Thank You

Questions?

Paper: **Heterogeneity at Hyperscale: Characterization and Scheduling of Large Production AI Clusters at Alibaba** (*Operational Systems*)
Trace: github.com/alibaba/clusterdata
Contact: Lingyun Yang (yanglingyun.yly@alibaba-inc.com)