Pack Smart, Schedule Fair: Unifying Heterogeneity and Fragmentation Awareness in GPU Scheduling

Varun Ramesh
Microsoft, Stanford University
vramesh3@stanford.edu

Jiayu Chang
Stanford University
cjy1125@stanford.edu

Hyeonggyu Kim
Stanford University
hgk@stanford.edu

Stefan Gabriel Ene
Stanford University
enegstef@stanford.edu

***Abstract*—We integrate Fragmentation Gradient Descent (FGD) into Gavel's heterogeneity-aware scheduling framework through a two-stage design: Gavel's policy layer determines GPU type and quantity allocations via linear programming, while FGD selects the specific nodes that minimize fragmentation growth. We replicate both systems independently, validating Gavel on the Microsoft Philly trace (504 experiments) and FGD on the Alibaba GPU-sharing trace (6 placement policies across 1,200 nodes with 5,592 GPUs), then evaluate the combined scheduler on two Alibaba cluster topologies under both Max-Min Fairness and FIFO policies (900 total experiments). Our results reveal that fragmentation-aware placement reduces the fragmentation rate by 30–40% on clusters with mixed node sizes (1–8 GPUs per node), increasing cluster utilization by up to 8 percentage points and cluster throughput by 17% under high load, but provides no measurable benefit on uniform-node clusters, where all placement strategies converge. This finding demonstrates that the value of fragmentation-aware scheduling depends on *node topology diversity*, not GPU type heterogeneity, and that effective cluster scheduling requires jointly considering allocation policy, placement strategy, and physical topology.**

***Index Terms*—GPU sharing, fragmentation, heterogeneity-aware, scheduling policies, workload traces, distributed training**

I. Introduction

GPU cluster scheduling for deep learning has evolved rapidly as organizations deploy increasingly diverse hardware for training and inference workloads. In production environments, GPU utilization averages between 25% and 50% [2], motivating the adoption of GPU sharing where multiple tasks run on a single physical GPU through partial resource allocation.

While GPU sharing improves aggregate utilization, it introduces *resource fragmentation*: remaining capacity on individual devices may be too small to satisfy subsequent requests, and these fragments cannot be consolidated across physical GPU boundaries. Production trace analysis from Alibaba clusters shows that fragmentation renders 21% to 42% of unallocated GPU resources unusable during normal operation [2], [3].

Separately, heterogeneous GPU clusters require scheduling policies that account for varying per-model speedups across accelerator types. A ResNet-50 model sees a 10x throughput improvement moving from a K80 to a V100, while an A3C reinforcement learning model sees only 2x [1].

Two recent systems address complementary aspects of this problem. Gavel [1] expresses scheduling policies as optimization problems over an effective throughput abstraction, enabling heterogeneity-aware allocation with support for fairness, makespan minimization, and hierarchical policies. However, Gavel was validated exclusively on whole-GPU workloads where fragmentation does not arise. FGD [2] defines a statistical fragmentation measure and places tasks toward the steepest descent of this measure, reducing unallocated GPUs by up to 49%. However, FGD operates under FIFO ordering without a flexible policy framework.

Because these two mechanisms operate at different levels of abstraction – Gavel at the GPU-type level, FGD at the node level – their integration raises a natural question: whether limited coordination between allocation and placement could further improve packing efficiency.

We show that the value of fragmentation-aware placement depends not on GPU type heterogeneity but on *node topology diversity*, specifically the variance in GPUs per node across a cluster. On clusters with mixed node sizes (1, 2, 4, and 8 GPUs per node), FGD-style placement reduces fragmentation by 30–40% and recovers up to 280 GPUs of usable capacity. On uniform-node clusters, all placement strategies converge and fragmentation-aware scheduling provides no measurable benefit. This dimension is not evaluated by either Gavel or FGD in their original work.

A. Contribution

We bridge these approaches with five contributions:

1) **Replication.** Independent replication of both Gavel and FGD on their original traces, validating our implementations against published results.
2) **Integration.** A two-stage scheduler that combines Gavel's heterogeneity-aware LP allocation with FGD's fragmentation-aware placement.
3) **Topology insight.** An evaluation showing that fragmentation reduction depends on *node topology diversity* (mixed node sizes) rather than GPU type heterogeneity.
4) **Exploratory coordination.** An exploratory study of limited allocation–placement coordination using a Placement-Opportunity-Aware (POA) signal.

*This paper was written as a final project report for Stanford's Advanced Networking and Distributed Systems CS244 course in Winter 2026. Thanks to Dr. Keith Winstein and Dr. David Mazières for their guidance and review.

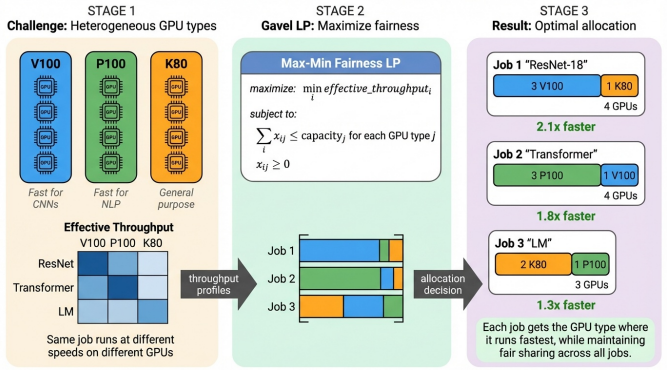

Fig. 1: Gavel's heterogeneity-aware allocation. Jobs exhibit different throughputs across GPU types. The Max-Min Fairness LP allocates time fractions to maximize minimum effective throughput. Each job receives GPUs on its best types.

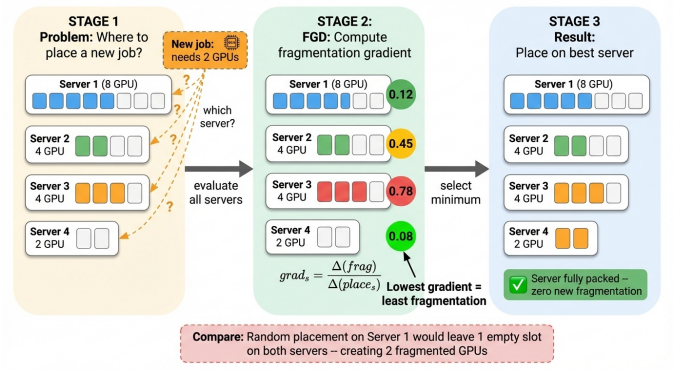

Fig. 2: FGD placement algorithm. For a 2-GPU job, FGD computes the fragmentation gradient at each server and selects the minimum. Server 4 (gradient = 0.08) fully packs, creating zero new fragmentation.

5) **Visualization.** An open-source interactive visualization tool[1] for exploring GPU allocation, fragmentation, and scheduling dynamics across experiments (Appendix).

All deliverable code, including our extended Gavel+FGD simulator, is available at https://github.com/Turquoise-T/cs244c-GPU_fragment.

II. Background & Related Work

A. Heterogeneity-Aware Scheduling with Gavel

Gavel [1] expresses scheduling policies as optimization problems over allocation matrices (Fig. 1). For each round, it solves a Max-Min Fairness LP to maximize the minimum effective throughput across all jobs, subject to per-type capacity constraints, then maps fractional allocations to physical assignments via priority scores. Gavel generalizes across policies (Max-Min Fairness, Finish-Time Fairness, FIFO, makespan minimization) but was validated exclusively on whole-GPU workloads where fragmentation does not arise.

B. Fragmentation-Aware Placement with FGD

FGD [2] defines a statistical fragmentation measure – the expected number of GPUs per node that cannot serve a randomly sampled task – and places each incoming task on the node with the minimum fragmentation gradient (Fig. 2). This captures both *deficient* fragmentation (insufficient GPU capacity) and *stranded* fragmentation (sufficient GPU but insufficient co-located CPU or memory). FGD was validated on Alibaba traces but operates under FIFO ordering without a flexible policy framework.

C. Other Related Work

GPU cluster scheduling spans fairness and priority (Tiresias [7], Themis [9]), time-slicing and migration (Gandiva [6]), topology-aware virtual clusters (HiveD [10]), fine-grained sharing (Salus [5]), co-adaptive scheduling (Pollux [11]), and heterogeneity-aware elastic scaling (Sia [12]). See Fig. 8 in the Appendix for a research lineage. Our work bridges the heterogeneity-aware and fragmentation-aware lines.

[1]https://github.com/varunr89/gpu-scheduling-viz

III. Methodology

Our approach proceeds in three phases: baseline replication, integration, and comparative evaluation (see Fig. 10 in the Appendix for how we bridge Gavel's round-based simulation with FGD's inflation methodology).

Phase 1: Baseline Replication. We replicate Gavel on the Microsoft Philly trace (108 GPUs: 36 V100, 36 P100, 36 K80) and replicate FGD on the Alibaba v2023 trace [3] (Cluster H: 5,592 GPUs, 1,200 nodes with mixed node sizes of 1, 2, 4, and 8 GPUs).

Phase 2: Integration. We integrate FGD into Gavel as a two-stage architecture (Fig. 5). Gavel's policy layer solves an LP for time-fraction allocations per GPU type, and FGD then selects the physical node with the minimum fragmentation gradient within the allocated type. This preserves policy-level optimization while adding fragmentation awareness.

Phase 2b: Allocation–Placement Coordination (GavelFGD+). We also explore augmenting the LP objective with a *Placement-Opportunity-Aware* (POA) signal. For each GPU type t and job demand d: poa_score$[t, d]$ = #nodes with $\geq d$ free GPUs / #nodes of type t. The LP objective becomes $\max \min_j \text{eff\_throughput}(j) + \mu \sum_{j,t} \text{poa\_score}[t, d_j] \cdot x_{j,t}$, steering flexible jobs toward GPU types where placement is currently more feasible.

Phase 3: Evaluation. We test on two Alibaba cluster topologies to isolate the effect of node-size diversity:

TABLE I: Cluster configurations used in evaluation.

Property	Alibaba Split	Cluster H
Total GPUs	6,212	5,592
Nodes	1,213	1,200
GPU types	12 (heterogeneous)	1 (homogeneous)
GPUs/node	8 (uniform)	1, 2, 4, 8 (mixed)
Node sizes	All same per type	462/310/228/200

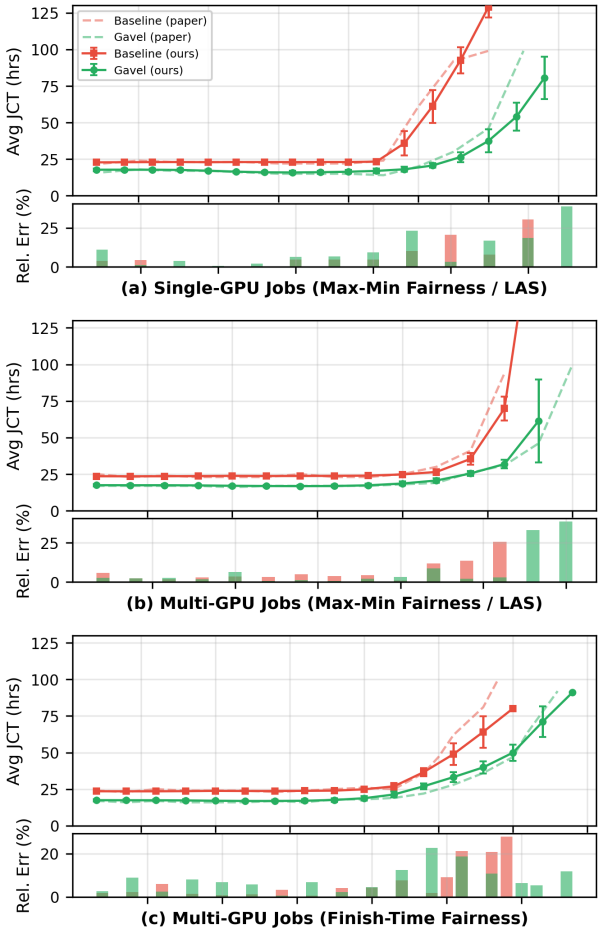

Fig. 3: Gavel replication results (Philly trace, 3 seeds). Solid: our results (mean ± std). Dashed: digitized paper reference. Relative error subplots below each panel. (a) Single-GPU (MMF). (b) Multi-GPU (MMF). (c) Multi-GPU (Finish-Time Fairness). Error is small at sub-saturation loads and diverges near the saturation knee.

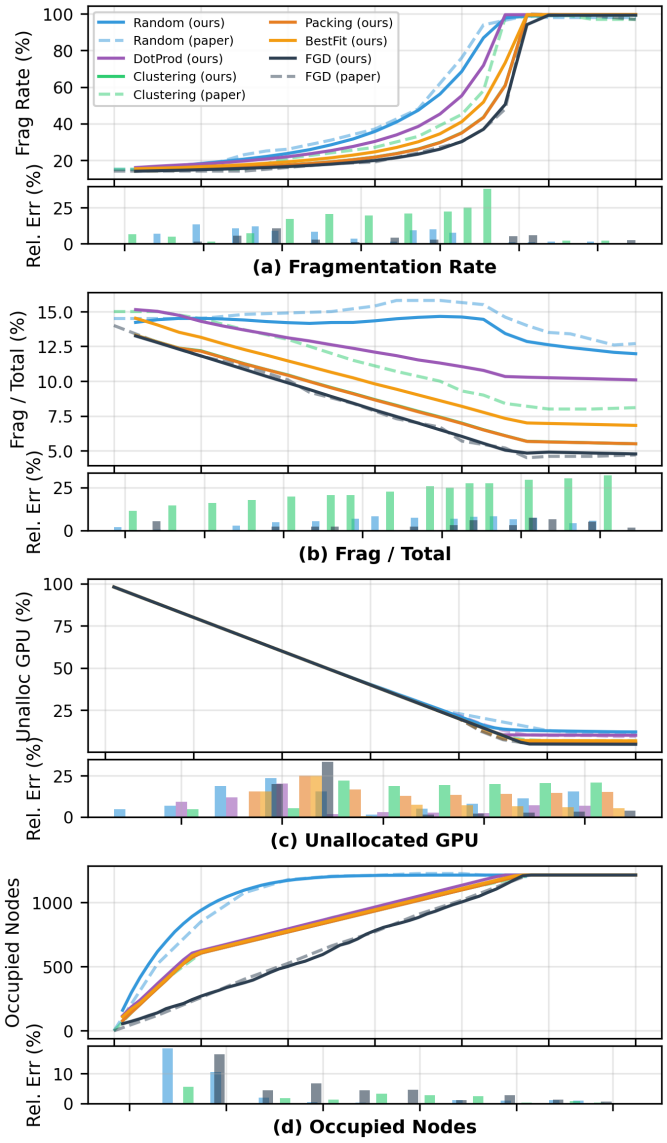

Fig. 4: FGD replication (Alibaba Cluster H, 6 policies, 10 runs). Solid: our results. Dashed: digitized paper reference. Relative error subplots below each panel. (a) Fragmentation rate. (b) Fragmented GPUs as % of total. (c) Unallocated GPU %. (d) Occupied nodes. FGD consistently achieves the lowest fragmentation and tightest packing.

Each combined configuration sweeps 15 arrival rates × 3 seeds under both Max-Min Fairness and FIFO allocation policies, with 4 placement strategies (Random, Strided, Best-Fit, FGD), totaling 720 experiments (900 including replication runs).

IV. Replication Results

A. Gavel Replication

We ran 504 experiments across three configurations from the original Gavel paper [1]: single-GPU jobs (MMF), multi-GPU jobs (MMF), and multi-GPU jobs (Finish-Time Fairness), each sweeping arrival rate across 3 seeds.

Fig. 3 overlays our results against digitized paper references. Gavel's heterogeneity-aware policy reduces average JCT by 20–70% compared to the baseline. Sub-saturation error is small (3.05 hours RMSE); larger deviations appear only near the saturation knee where our simulator saturates at a slightly different load point. We attribute this to queue management: the original Gavel likely uses backfilling or priority-based preemption for pending jobs that is not fully specified in the paper, whereas our implementation uses strictly FIFO ordering for the pending queue.

B. FGD Replication

We replicate FGD's standalone evaluation on Alibaba Cluster H using the inflation methodology [2], comparing six placement policies. Fig. 4 shows our replication of four key metrics alongside digitized references. FGD maintains the lowest fragmentation throughout the demand range. At 80% demand, Random placement fragments 85% of unallocated

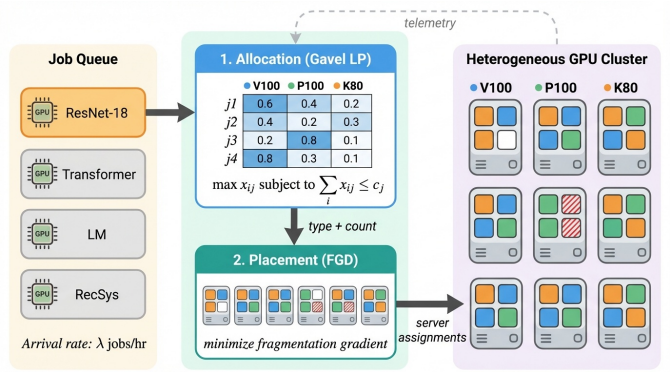

Fig. 5: Two-stage Gavel+FGD architecture. Stage 1 (Allocation): Gavel's LP computes time-fraction allocations across GPU types for each job. Stage 2 (Placement): FGD evaluates the fragmentation gradient at each candidate node and selects the node that minimizes fragmentation growth. Telemetry from the cluster feeds back into both stages.

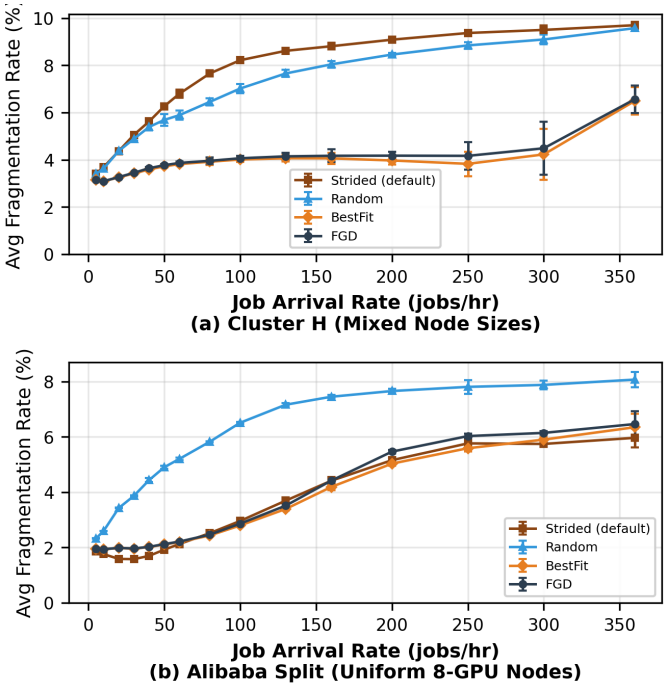

Fig. 6: Fragmentation rate vs. job arrival rate across four placement strategies. (a) Cluster H: BestFit and FGD reduce fragmentation by 40–50% compared to Strided at moderate-to-high load. (b) Alibaba Split: all strategies converge. Mean ± std over 3 seeds, Max-Min Fairness policy.

GPUs while FGD fragments only 42%. The fragmentation-to-total ratio stays below 8% with FGD, and relative error subplots confirm close agreement across the full demand range.

A separate arrival-order sensitivity study (Appendix Table II) shows that FGD's effectiveness depends on workload ordering assumptions; we discuss this robustness caveat in Section VI-E.

V. Combined Gavel+FGD Results

We evaluate the integrated system (Fig. 5) on both Alibaba cluster topologies under four placement strategies: Strided (Gavel's default), Random, BestFit, and FGD.

A. Fragmentation Across Arrival Rates

Fig. 6 compares fragmentation rate versus job arrival rate across both cluster topologies under Max-Min Fairness with four placement strategies. On Cluster H (panel a), Strided placement produces the highest fragmentation (9–10% at 300+ jobs/hr), while BestFit and FGD reduce it to 4–6% – a consistent 40–50% reduction across the full arrival rate sweep. Random placement falls between these two groups. This ordering holds consistently: FGD ≈ BestFit < Random < Strided, under both Max-Min Fairness and FIFO, indicating that the placement benefit is *independent of the allocation policy*.

On the Alibaba Split cluster (panel b), the gap narrows substantially: all strategies converge to within 1–2 percentage points. The residual separation arises because even uniform 8-GPU nodes exhibit fragmentation when partial-GPU jobs leave mismatched remainders, but this effect is far smaller than the mixed-node case.

B. Utilization Impact of Placement

Fragmentation directly reduces usable cluster capacity. Fig. 7 shows that on Cluster H, Strided placement caps at 89% utilization under high load while BestFit and FGD reach 94%, a gap of 5 percentage points representing approximately 280 recoverable GPUs on a 5,592-GPU cluster. Panel (b) shows *effective capacity*, defined as utilization minus fragmentation rate. Because fragmented GPUs are allocated but unusable by any waiting job, subtracting the fragmentation rate from raw utilization yields the fraction of cluster capacity doing useful work. At 250 jobs/hr, Strided achieves 64% effective capacity while BestFit reaches 78% – a 13-point gap, translating to 17% more completed jobs.

Individual job completion times, measured over the steady-state window (jobs 4000 to 5000), are identical across all placements to within numerical precision. Placement affects cluster-level capacity but not per-job throughput, confirming that allocation and placement operate on orthogonal dimensions.

VI. Analysis

A. The Role of Node Topology

The contrast between panels (a) and (b) in Fig. 6 reveals the central finding of our evaluation: **fragmentation-aware placement matters only when clusters have diverse node sizes.**

Fig. 9 illustrates why. On uniform-node clusters (left panel), any job that fits on one node fits equally well on any other node of the same type. The placement decision is trivial because all choices produce the same fragmentation. On mixed-node

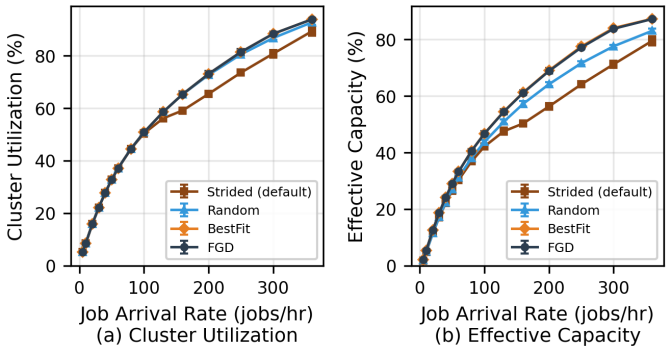

Fig. 7: Cluster H: (a) Cluster utilization and (b) effective capacity (utilization minus fragmentation) vs. arrival rate. Strided underperforms by up to 8 utilization points and 13 effective-capacity points at high load. Mean ± std over 3 seeds, Max-Min Fairness policy.

clusters (right), a 3-GPU job cannot use scattered 1-GPU fragments across different nodes, so placement decisions directly determine whether capacity remains usable.

This finding has practical implications: cloud providers deploying new clusters with uniform node sizes (e.g., standard 8-GPU servers) gain little from fragmentation-aware placement, while clusters that accumulate heterogeneous hardware over time, mixing older 1–2 GPU machines with newer 4–8 GPU servers, benefit substantially.

B. Policy Independence of Placement Effects

Our evaluation under both Max-Min Fairness and FIFO allocation policies shows that FGD's placement benefit is consistent regardless of the allocation policy. On Cluster H, the fragmentation reduction from FGD is 30–40% under both policies, with nearly identical curves (see Appendix Fig. 11 and Fig. 12 for per-utilization breakdowns). Combined with the utilization impact shown in Fig. 7, this suggests that allocation and placement affect largely orthogonal outcomes: placement shapes cluster capacity, while allocation shapes per-job throughput.

C. Gavel at Alibaba Scale

Running Gavel's LP-based allocation on the Alibaba cluster (6,200 GPUs, up to 1,700 concurrent jobs) revealed performance bottlenecks not present at Philly scale (108 GPUs, 50 jobs). At Alibaba scale, the LP solver consumes 48% of per-round runtime (3.0s per solve), compared to 8% at Philly scale. We addressed this through solver warm-starting and logging optimizations, reducing per-round time from 1.58s to 0.64s – sufficient for our evaluation but highlighting that Gavel's LP formulation faces scalability limits at production cluster sizes.

D. POA Coordination Results

Fig. 13 (Appendix) compares Gavel, GavelFGD, and GavelFGD+ across utilization levels. FGD placement alone reduces fragmentation by 40–50% relative to baseline Gavel. GavelFGD+ further improves packing at high load: at 85.4% utilization (360 jobs/hr), fragmentation drops from 2.01% to 0.47%, a 76.7% reduction relative to GavelFGD. The percentage of unallocated GPUs remains nearly identical across all three systems (Fig. 13c), confirming that the improvement comes from more efficient packing rather than additional resources.

E. Robustness to Workload Ordering

FGD's placement decisions depend on a workload-popularity distribution that determines the fragmentation gradient. Table II (Appendix) evaluates FGD under seven arrival orderings – trace order, ascending/descending by GPU demand, and four phased orderings that group tasks by demand tier. FGD underperforms the simpler Clustering baseline in 5 of 7 orderings, accumulating 2–3× more pending GPU demand. It wins only when large jobs arrive first and saturate big nodes before small jobs fill remaining slots – a pattern that aligns with FGD's gradient assumptions. Ascending orderings are worst: small jobs scatter early, leaving fragmented remainders that FGD cannot consolidate. This sensitivity suggests FGD's benefit is conditioned on workload stationarity and may degrade under distributional shift.

VII. Limitations

Our evaluation relies entirely on trace-driven simulation, which introduces several sources of potential bias.

Simulation Fidelity. Trace-driven simulation assumes the intervention does not affect the validity of the replayed trace [4]. Scheduling decisions can alter queuing delays and resource contention, feeding back into job performance; our simulator does not model such feedback effects.

Workload Stationarity. As shown in Section VI-E, FGD's effectiveness degrades under non-stationary workload orderings. Monte Carlo evaluation smooths arrivals toward a stationary distribution, which may overstate FGD's benefit under real distributional drift.

Migration and Preemption Costs. Gavel assumes jobs can be checkpointed and migrated with negligible overhead. In practice, migration incurs costs from checkpoint serialization, data transfer, and GPU warm-up that our simulator does not model.

VIII. Conclusion

We presented an integrated GPU cluster scheduler combining Gavel's heterogeneity-aware allocation with FGD's fragmentation-aware placement. Through replication of both systems (504 Gavel experiments on Philly, standalone FGD on Alibaba) and 900 combined experiments across two cluster topologies, we demonstrated that:

1) Gavel's heterogeneity-aware policies reduce JCT by 20–70% compared to heterogeneity-agnostic baselines, and this benefit holds at Alibaba scale.
2) FGD placement reduces fragmentation by 30–40% on clusters with mixed node sizes (1–8 GPUs per node), translating to up to 8 percentage points higher cluster

utilization and 17% more completed jobs under high load.

3) Fragmentation-aware placement provides no benefit on uniform-node clusters, where all strategies converge.

The central insight is that fragmentation is driven by *node topology diversity* (variance in GPUs per node), not GPU type heterogeneity. Cluster architects can mitigate fragmentation through hardware homogeneity or software (FGD-style placement), and our two-stage design shows that allocation and placement address orthogonal concerns that compose independently.

ACKNOWLEDGMENT

We thank Dr. Keith Winstein, Dr. David Mazières, and Akshay Shrivatsan for their guidance and feedback throughout this project.

AI USAGE

AI assisted with experiment infrastructure (SLURM pipeline, SSH multiplexing, result extraction), debugging Gavel–FGD integration, and building the visualization tool. The research design, methodology, and analysis are our own.

REFERENCES

[1] Narayanan, D., Santhanam, K., Kazhamiaka, F., et al. "Heterogeneity-Aware Cluster Scheduling Policies for Deep Learning Workloads." OSDI 2020. Usenix Web Link.

[2] Weng, Q., Yang, L., Yu, Y., et al. "Beware of Fragmentation: Scheduling GPU-Sharing Workloads with Fragmentation Gradient Descent." USENIX ATC 2023. Usenix Web Link.

[3] Alibaba Group. "Cluster Data Collected from Production Clusters." 2020-2023. GitHub Web Link.

[4] Alomar, A., Hamadanian, P., Nasr-Esfahany, A., et al. "CausalSim: A Causal Framework for Unbiased Trace-Driven Simulation." NSDI 2023. Usenix Web Link.

[5] Yu, P. and Chowdhury, M. "Salus: Fine-Grained GPU Sharing Primitives for Deep Learning Applications." MLSys, 2020. arXiv:1902.04610.

[6] Xiao, W., Bhardwaj, R., Ramjee, R., et al. "Gandiva: Introspective Cluster Scheduling for Deep Learning." OSDI 2018. Usenix Web Link.

[7] Gu, J., Chowdhury, M., Shin, K.G., et al. "Tiresias: A GPU Cluster Manager for Distributed Deep Learning." NSDI 2019. Usenix Web Link.

[8] Peng, Y., Bao, Y., Chen, Y., et al. "Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters." EuroSys 2018. ACM Web Link.

[9] Mahajan, K., Balasubramanian, A., Singhvi, A., et al. "Themis: Fair and Efficient GPU Cluster Scheduling." NSDI 2020. Usenix Web Link.

[10] Zhao, H., Han, Z., Yang, Z., et al. "HiveD: Sharing a GPU Cluster for Deep Learning with Guarantees." OSDI 2020. Usenix Web Link.

[11] Qiao, A., Choe, S.K., Subramanya, S.J., et al. "Pollux: Co-adaptive Cluster Scheduling for Goodput-Optimized Deep Learning." OSDI 2021. Usenix Web Link.

[12] Jayaram Subramanya, S., Zhong, D., Choudhury, S., et al. "Sia: Heterogeneity-aware, Goodput-optimized ML-Cluster Scheduling." SOSP 2023. ACM Web Link.

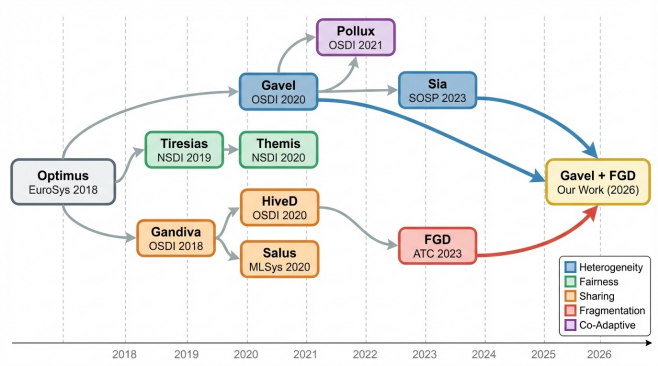

Fig. 8: Research lineage of GPU cluster scheduling. Color-coded by primary contribution: heterogeneity awareness (blue), fairness/priority (green), sharing/consolidation (orange), fragmentation (red), and co-adaptive scheduling (purple). Our work (gold) bridges the heterogeneity-aware and fragmentation-aware lines.

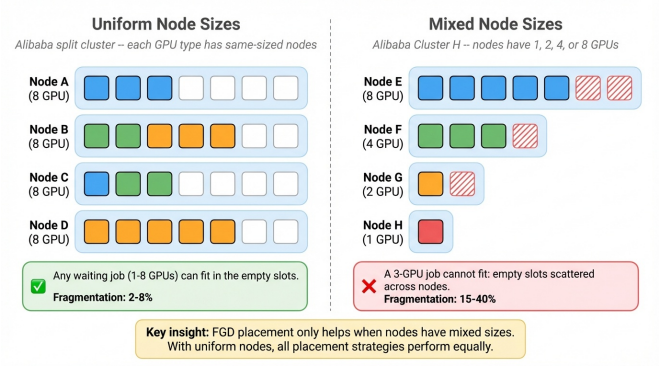

Fig. 9: Node topology determines fragmentation severity. Left: uniform 8-GPU nodes produce 2–8% fragmentation regardless of placement. Right: mixed node sizes (1–8 GPUs) produce 15–40% fragmentation because scattered fragments across small nodes cannot be consolidated. FGD placement helps only in the mixed case.

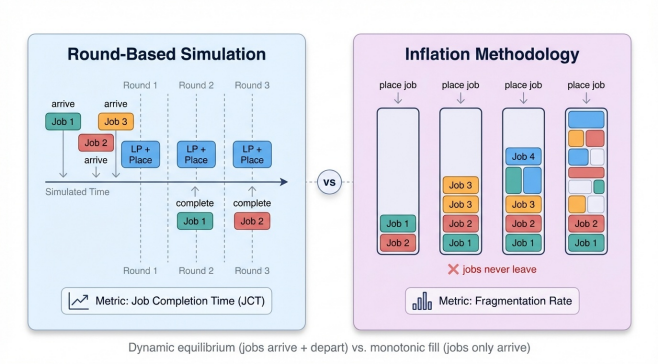

Fig. 10: Evaluation methodology bridging Gavel's round-based simulation (dynamic arrivals/departures, measuring JCT) with FGD's inflation methodology (monotonic fill, measuring fragmentation). Phase 1 replicates each system independently; Phases 2–3 evaluate the combined system using Gavel's round-based approach with FGD placement.

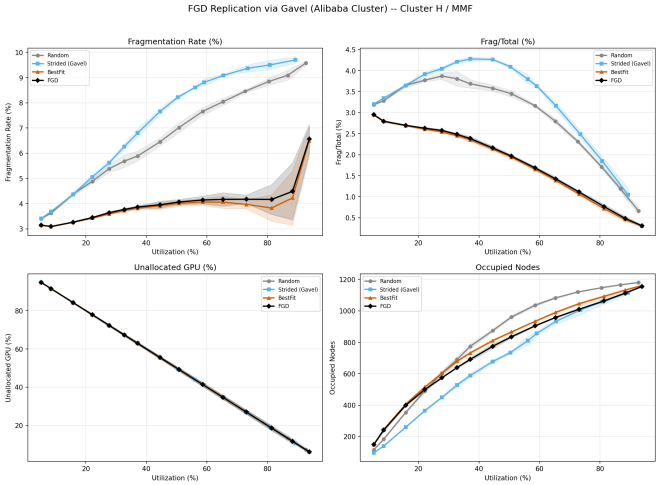

Fig. 11: Gavel+FGD on Cluster H (mixed node sizes, Max-Min Fairness) vs. cluster utilization. FGD placement (black) reduces fragmentation rate by 30–40% vs. Random (blue) at moderate utilization. BestFit (orange) is intermediate. Strided (brown) performs similarly to Random. Error bands: ±1 std over 3 seeds.

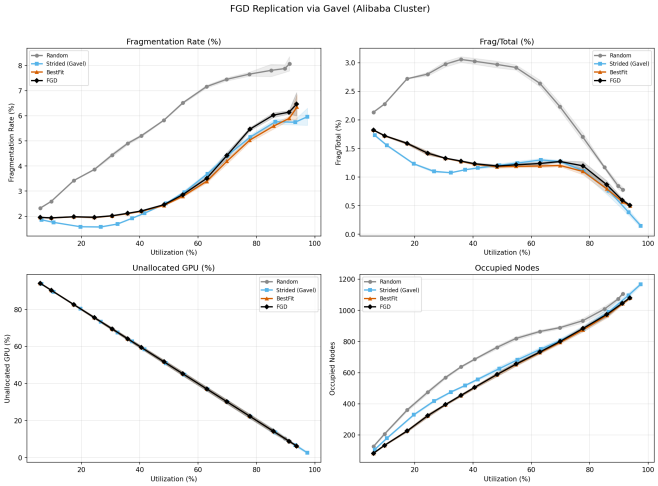

Fig. 12: Gavel+FGD on Alibaba Split (uniform 8-GPU nodes) vs. cluster utilization. All placement strategies converge – fragmentation rates are identical regardless of placement policy. Node topology uniformity eliminates the fragmentation problem FGD targets.

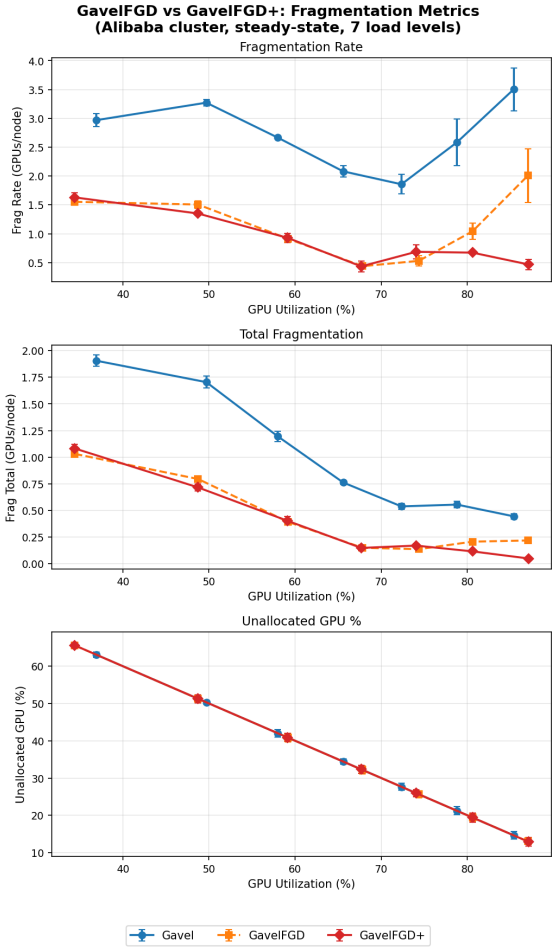

Fig. 13: Performance comparison between Gavel, GavelFGD, and GavelFGD+ on the Alibaba cluster under steady-state workloads. (a) Fragmentation rate, (b) total cluster fragmentation, and (c) percentage of unallocated GPUs across seven utilization levels.

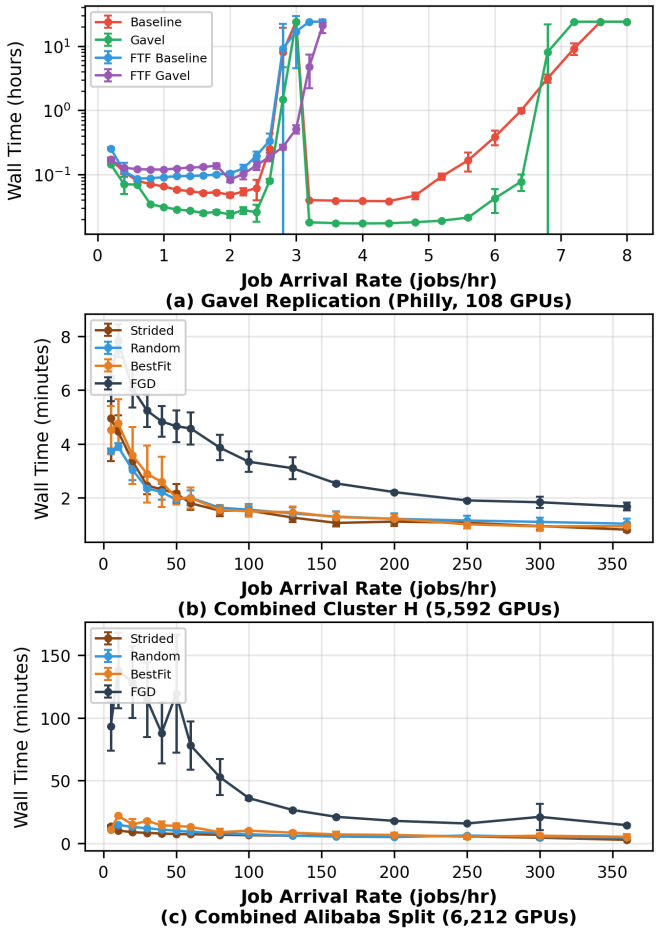

Fig. 14: Simulation wall time vs. arrival rate across experiment sets. (a) Gavel replication (Philly, log scale): exponential blowup near saturation. (b) Combined Cluster H: flat at 1–6 minutes. (c) Combined Alibaba: decreasing with load as fewer rounds are needed. Mean $\pm$ std over seeds.

Fig. 14 reports simulation wall time across all experiment sets (over 900 experiments total) on Stanford FarmShare (Intel Xeon, no GPU required for simulation). Panel (a) shows Gavel replication on the Philly trace: wall time increases exponentially near the saturation knee, where experiments hit the 24-hour SLURM limit; sub-saturation runs complete in under 1 hour. Panel (b) shows Cluster H experiments: wall time remains flat at 1–6 minutes across all arrival rates. Panel (c) shows Alibaba Split experiments: wall time decreases from 5+ minutes at low load to under 1 minute at high load, reflecting fewer scheduling rounds needed at higher arrival rates.

To support interactive exploration of scheduling behavior across 843 experiments, we developed a web-based visualization tool (Fig. 15). The tool loads experiment telemetry from a custom binary format hosted on Azure Blob Storage and decodes it in a Web Worker for responsive interaction. Key features include side-by-side comparison of two experiments with synchronized playback, time-series charts for utilization and fragmentation, an aggregate results dashboard with digitized paper reference overlays (Fig. 16), and filtering by trace, figure, scheduler, placement mode, load, and seed.

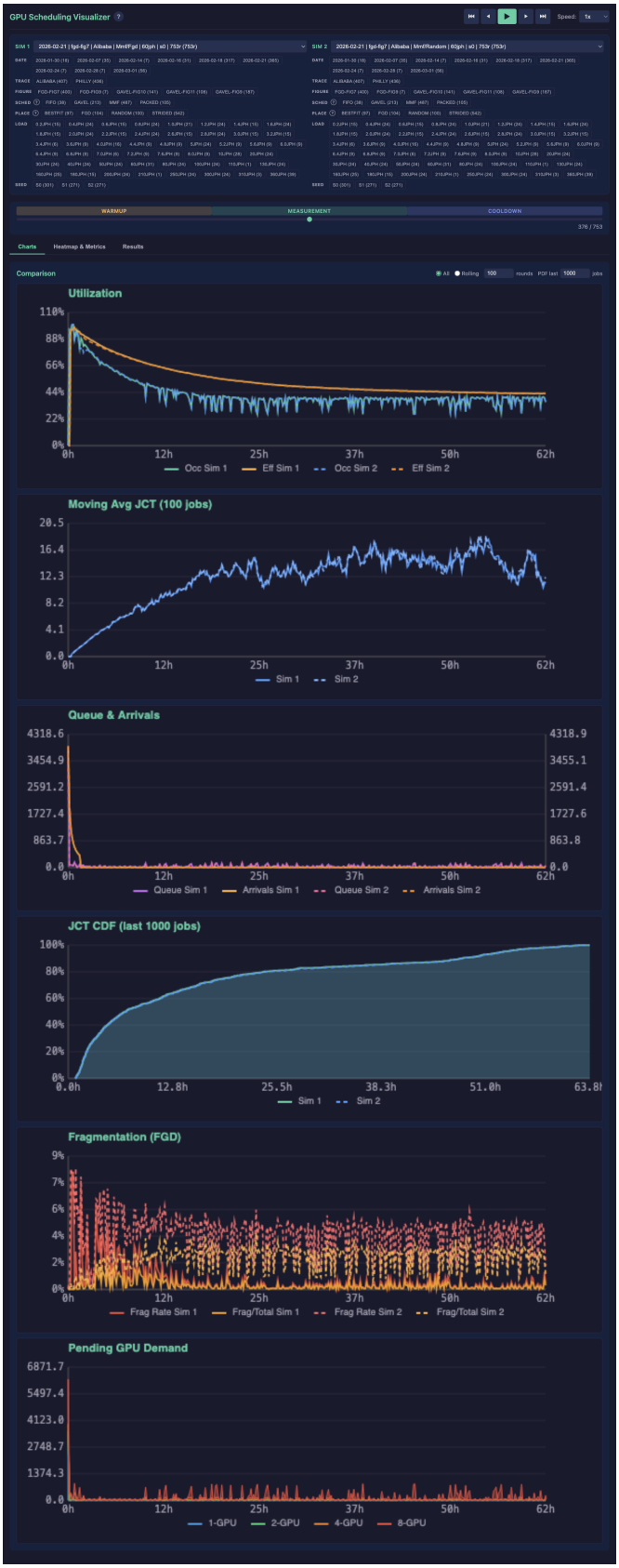

Fig. 15: Visualization tool: side-by-side comparison view. Two Alibaba experiments (FGD vs. Random placement at 60 jobs/hr) with synchronized playback. Filter panels allow selection by trace, figure type, scheduler, placement, load, and seed.

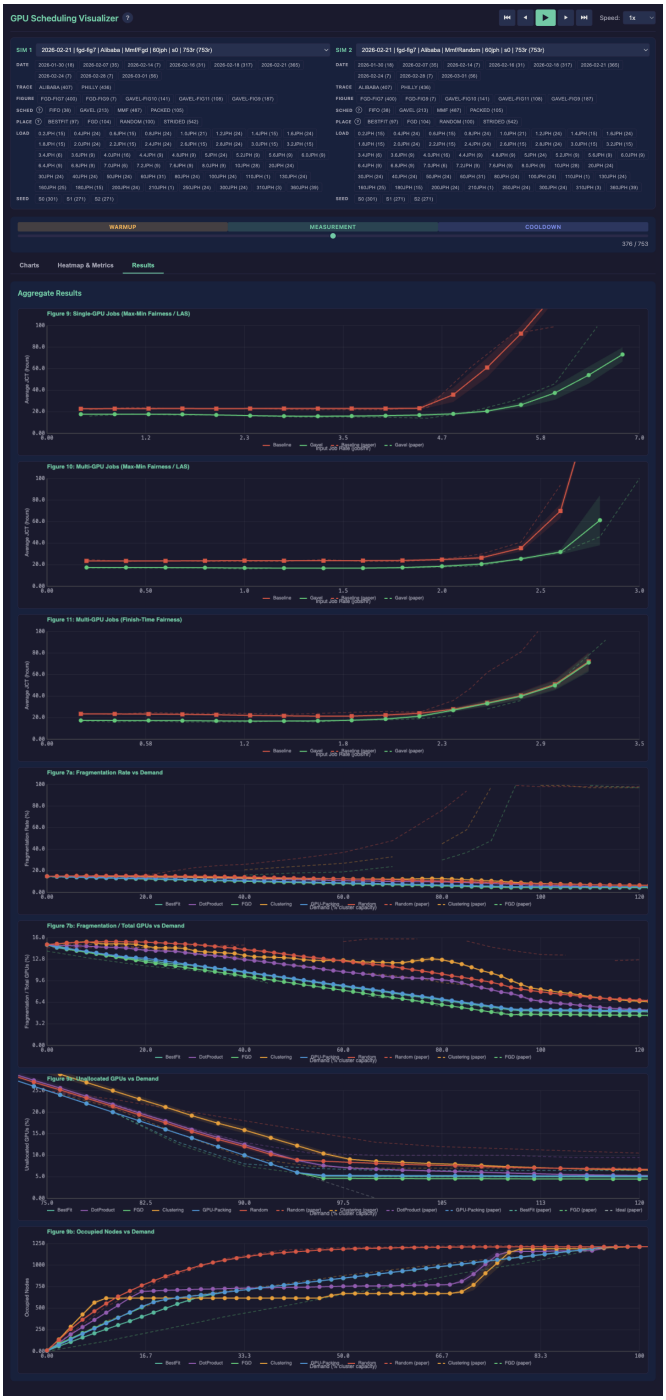

Fig. 16: Visualization tool: aggregate results dashboard. Replication curves for all Gavel and FGD figure types with digitized paper reference overlays for visual validation across the full experiment matrix.

Table II reports total pending task GPUs when cumulative arrived workload reaches 96% of cluster GPU capacity. Higher values indicate more unscheduled GPU demand at the same near-saturation load point.

We evaluate four task-arrival-order families:

- **Trace**: original creation-time order from the Alibaba trace.
- **Ascending**: tasks sorted from smaller to larger GPU requests.
- **Descending**: tasks sorted from larger to smaller GPU requests.
- **Phased**: tasks grouped by GPU-demand tier and submitted phase by phase.

The phased tiers are:

- Tier 0: $gpu = 0$ (CPU-only tasks)
- Tier 1: $0 < gpu < 0.5$ (small fractional GPU)
- Tier 2: $0.5 \leq gpu < 1$ (large fractional GPU)
- Tier 3: $gpu = 1$ (single full GPU)
- Tier 4: $gpu > 1$ (multi-GPU tasks)

TABLE II: Total pending task GPUs at 96% arrived workload across task arrival orders (Alibaba, 90% cluster scale).

Task arrival order	**FGD**	**Clustering**	**Random**
trace	480.0	205.5	522.0
ascending	2167.0	835.0	1029.0
descending	110.2	129.0	107.6
phased 01234	1890.0	808.0	993.0
phased 12034	1318.0	332.0	925.0
phased 24031	949.0	162.0	586.0
phased 31042	124.0	267.9	421.7

The table shows that FGD is highly sensitive to task arrival order: it performs worse than Clustering in several configurations, but outperforms both baselines in one phased order. This pattern is consistent with a distribution-conditioned placement policy whose effectiveness depends on how closely the assumed workload mix matches short-horizon arrivals.