

Queue–Idle Paradox

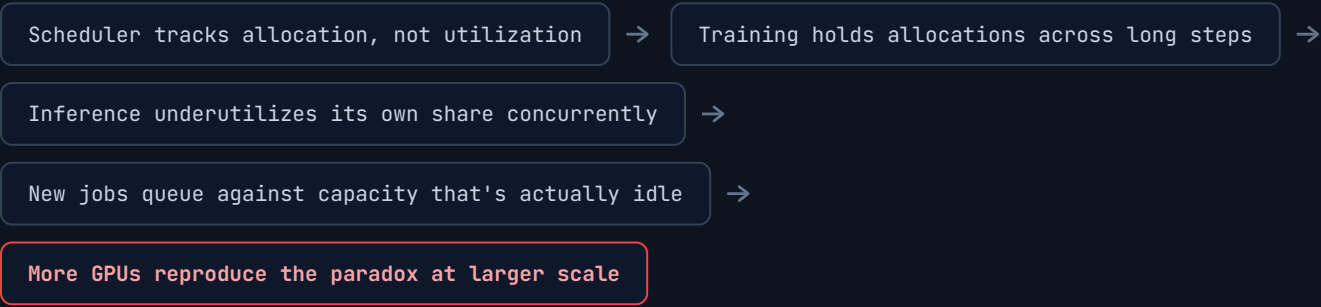
DEFINITION

The simultaneous presence of job queues and idle GPUs in the same cluster.

FRAMEWORK FLOW

01 The Condition Training and inference workloads share a cluster; the scheduler tracks GPUs as occupied the moment they're allocated, not whether they're doing useful work.	02 The Boundary Training jobs hold large allocations across long-running steps while inference underutilizes its own share — the point where "occupied" and "producing work" diverge for two workload classes simultaneously.	03 Failure State New jobs queue because the scheduler sees no free capacity, while allocated GPUs sit idle inside both the training and inference allocations — queue and idle capacity coexist in the same cluster, at the same time.	04 Consequence The organization reads the queue as a hardware shortage and adds GPUs; without workload class isolation, the same paradox reproduces at the new, larger scale.
--	--	---	--

Failure State: Occupied Is Not the Same as Working



THE DIAGNOSTIC SIGNAL

OCCUPANCY OWNERSHIP Ask the scheduler whether a GPU is occupied. It answers correctly — allocated is allocated. It was never asked whether the GPU is doing useful work.	COMPOSITE OWNERSHIP Ask who reconciles queue depth against idle-within-allocation. Nobody does by default — the scheduler and the utilization dashboard are two systems that never cross-reference each other.
TRIGGER CONDITION Mixed training + serving on a shared cluster is the primary activation condition — workload classes with uniform, saturating utilization patterns rarely trigger it.	REMEDiation SHAPE Workload class isolation, density-aware scheduling, an explicit idle-within-allocation check before any queue is read as a capacity signal.

WHERE THE PARADOX CONCENTRATES

MIXED TRAINING + SERVING The epicenter — training jobs hold allocations while inference underutilizes its share simultaneously; the scheduler sees both as occupied and queues new work.	INTERACTIVE EXPLORATION ENVIRONMENT Notebook sessions hold cards without computing — the highest Allocation–Utilization Gap of any archetype, a related but distinct failure shape.
WORKLOAD CLASS ISOLATION The fix, not a pattern — separating training and serving scheduling domains removes the condition that produces the paradox in the first place.	

ARCHITECTURAL RELATIONSHIPS

#89 Effective GPU Yield RELATED MODERATE Reduced Effective GPU Yield produces the Queue–Idle Paradox as its operational symptom: work remains queued while allocated GPU capacity is simultaneously idle or underutilized.	#91 Phantom Scarcity RELATED MODERATE Co-resident in the GPU Utilization Analyzer — Phantom Scarcity and the Queue-Idle Paradox describe the same allocation failure from different angles.
#93 Fragmentation Tax RELATED MODERATE Co-resident in the GPU Utilization Analyzer — fragmentation produces the Queue-Idle Paradox.	#94 Economic Density Loss RELATED MODERATE Co-resident in the GPU Utilization Analyzer as paired paradox/loss metrics.

ANCHOR /gpu-utilization-analyzer/

FRAMEWORK HOME

Vendor and scheduler agnostic — the occupied-vs-working distinction applies to any cluster mixing workload classes on shared capacity.

"The scheduler and the queue agree that the cluster is full. The utilization dashboard disagrees — and the dashboard is the one telling the truth."