International Journal of AI, Big Data, Computational and Management Studies
Noble Scholar Research Group | Volume 7, Issue 4, PP 54-63, 2026
ISSN: 3050-9416 | https://doi.org/10.63282/3050-9416.IJAIBDCMS-V7I4P106

Original Article

GPU Capacity Scarcity: A Prioritization Framework for Allocating Constrained AI Compute across Competing Enterprise Initiatives

Shaik Irfan
Independent Researcher, USA.

Received On: 28/08/2026 ***Revised On: 26/09/2026*** ***Accepted On: 03/10/2026*** ***Published On: 08/10/2026***

Abstract: *Enterprise AI programs have reached a point at which the accelerator, not the model or the data pipeline, is the binding constraint on strategic delivery, and GPU compute now consumes a share of technical budgets large enough that allocation decisions are indistinguishable from portfolio decisions. Yet the research response has split into two camps that rarely meet: systems-level GPU schedulers that optimize cluster utilization over workloads they assume are already approved, and project-portfolio methods that rank initiatives against generic resources but have never been specialized to accelerator-class compute at sub-hour granularity. This paper argues that GPU scarcity is now a portfolio-management problem, not only a scheduling problem, and proposes a five-stage prioritization framework that composes LSTM-based demand forecasting, Analytic Hierarchy Process criteria weighting, composite priority scoring, priority-weighted resource-constrained multi-project scheduling, and Kubernetes-level admission enforcement with bounded-wait guarantees. We evaluate the framework through discrete-event simulation across three enterprise portfolio scenarios and three oversubscription levels, benchmarked against FIFO and a utilization-maximizing genetic-algorithm baseline. Strategic-value-weighted throughput improved by 31% over FIFO and 21% over the utilization baseline while cluster utilization stayed within three percentage points, and allocation decisions remained stable under ±15% weight perturbation.*

Keywords: *GPU Capacity Scarcity, AI Compute Allocation, Resource Prioritization, Enterprise AI Initiatives, GPU Resource Management, Compute Optimization, Workload Scheduling, Artificial Intelligence Infrastructure, Capacity Planning, Multi-Objective Optimization.*

1. Introduction

Enterprise AI programs have crossed a threshold at which the accelerator not the data scientist, not the model, not even the data pipeline is the binding constraint. GPUs capable of training or serving modern transformer models are allocated months in advance, priced at a premium, and consumed by workloads whose business value varies by orders of magnitude. A recent review of cloud and AI infrastructure economics finds that GPU compute now accounts for roughly 40–60% of the technical budget of AI-focused organizations [1], a share large enough to make capacity decisions indistinguishable from strategic portfolio decisions. When a single resource class consumes more than half of a program's budget, the question of who gets it, when, and for what purpose stops being an operations ticket and becomes a question of corporate governance.

This paper argues that the enterprise response to GPU scarcity has lagged behind that economic reality. Infrastructure teams have adopted sophisticated cluster-level schedulers; portfolio-management offices continue to use spreadsheet-based prioritization inherited from the pre-AI era; and the two conversations rarely meet. The gap between them is where resources are quietly wasted on low-value experiments while strategic initiatives wait in queues, and where executives discover only at quarterly review that the model underpinning a flagship product was deprioritized by a scheduler that had no notion of flagship status.

1.1. GPU Compute as a First-Order Budget Line

Three forces have combined to make GPU capacity the dominant cost center in enterprise AI. First, model sizes and context lengths have grown faster than per-chip throughput, so each additional capability demands disproportionately more accelerator-hours. Second, serving footprints have expanded as organizations move from pilot inference endpoints to always-on, latency-bounded production services whose GPU reservations persist 24/7. Third, the supply side has not kept pace: lead times for high-end accelerators remain long, cloud providers meter them under constrained quotas, and on-premise refreshes are capital-intensive. The cumulative effect, as the cost literature documents, is that GPU line items dominate AI budgets and that cost-optimization pressure has shifted from storage and general compute toward the accelerator pool itself [1]. When a resource is both scarce and expensive, under-allocation is a strategic risk and over-allocation is a fiduciary one. Neither is acceptable, and both are what current practice tends to produce.

1.2. Heterogeneous Initiatives, One Accelerator Pool

The problem is sharpened by workload heterogeneity. A single enterprise GPU pool is typically asked to serve, in parallel: large-model pre-training jobs that want hundreds of accelerators for days; fine-tuning runs that want tens of accelerators for hours; low-latency inference services with strict tail-latency SLAs; batch scoring jobs that are latency-insensitive but throughput-hungry; and exploratory notebooks whose demand is bursty and whose value is diffuse. Each of these workload classes has a different cost profile, a different sensitivity to interruption, and a different strategic weight. Recent infrastructure-engineering guidance makes the point explicitly: traditional provisioning frameworks designed for CPU-era services underspecify the accelerator demand of training, fine-tuning, and inference, which differ qualitatively rather than merely quantitatively [2]. A scheduler that treats all GPU requests as fungible will, mechanically, favor the workload that fills the pool fastest which is almost never the workload the enterprise most wants to fund.

This heterogeneity is why the question "how should GPUs be allocated?" cannot be answered by a utilization metric alone. Two allocation plans can report identical cluster utilization while delivering radically different strategic value. The organization that notices this difference is the organization that keeps its AI roadmap on schedule.

1.3. Two Literatures, One Gap

The research response to GPU scarcity has split into two camps that do not talk to each other. On one side, systems researchers have produced increasingly refined cluster schedulers fair-share, gang-scheduled, elastic, predictive that assume the set of workloads is already given and that the scheduler's job is to pack them efficiently. On the other side, project-portfolio-management researchers have refined the Analytic Hierarchy Process (AHP) and the resource-constrained multi-project scheduling problem (RCMPSP) into mature machinery for ranking and sequencing competing initiatives, but have not applied that machinery to accelerator-class compute. Capacity-forecasting work sits between the two: it predicts GPU demand convincingly but stops short of adjudicating whose demand should be satisfied first.

The practical consequence is a governance gap at the enterprise-initiative level. Cluster schedulers answer the question "given these jobs, which runs next?" Portfolio tools answer the question "given these initiatives, which do we fund?" Nobody is answering the question that enterprise AI leaders actually ask: given a constrained GPU pool and a set of competing AI initiatives of unequal strategic weight, which initiatives get accelerator capacity, in what proportion, for how long, and under what admission guarantees? That question is where this paper locates its contribution.

1.4. Contribution

We propose a prioritization framework that treats constrained enterprise GPU capacity as a resource-constrained project portfolio and allocates it through five composed stages: LSTM-based demand forecasting over historical utilization traces; AHP-based weighting of prioritization criteria (strategic value, cost, SLA, dependency, risk); composite priority scoring that fuses AHP weights with forecasted demand pressure; a priority-weighted RCMPSP solver that produces an allocation plan; and Kubernetes-level enforcement through quotas and queue-theoretic admission thresholds with bounded wait guarantees. The framework is evaluated through discrete-event simulation across three synthetic enterprise portfolio scenarios and benchmarked against FIFO and utilization-maximizing baselines.

The contribution is deliberately not a new scheduler. It is a governance layer that sits above existing schedulers and tells them whose work matters more, with a traceable chain of evidence from executive-level criteria weights to pod-level admission decisions. In doing so it closes the gap between RCMPSP/AHP portfolio reasoning and Kubernetes-era GPU operations, and it reframes GPU scarcity from a technical bin-packing problem into an enterprise portfolio problem whose answers are auditable.

The remainder of the paper is organized as follows. Section 2 reviews the two disjoint literatures and situates the framework in the gap between them. Section 3 specifies the five-stage framework, its decision rules, and the simulation design. Section 4 reports simulation outcomes across baselines and oversubscription levels. Section 5 discusses governance implications and limitations, and Section 6 concludes with directions for enterprise validation.

2. Literature Review

The literature relevant to allocating constrained GPU capacity across competing enterprise AI initiatives splits cleanly, and inconveniently, into two bodies of work that share neither vocabulary nor evaluation criteria. On one side sits a mature systems-research tradition on GPU cluster scheduling, which treats incoming workloads as a given input and asks how to pack them efficiently. On the other sits an equally mature operations-research tradition on project portfolio management, which asks how a set of competing initiatives should be ranked and sequenced against scarce shared resources, but which has historically concerned itself with people, budget, and generic machine-hours rather than accelerators. A third, newer strand — AI-driven capacity forecasting — sits between them, improving the quality of the demand signal that either side consumes but stopping short of actually deciding who gets served first. This section walks through each body of work, extracts what it does and does not solve, and identifies the gap into which the proposed framework is placed.

2.1. Systems-Level GPU Scheduling: Efficient, Workload-Blind

The dominant line of GPU-scheduling research optimizes cluster-level objectives utilization, makespan, fragmentation, tail latency on the assumption that whatever workloads arrive at the scheduler are legitimate and interchangeable. Kwon and Bahn, in a representative recent

contribution, apply genetic optimization to multi-tenant AI scheduling and demonstrate measurable improvements in cluster utilization and job completion time by evolving placement decisions across heterogeneous accelerator pools [3]. Their formulation is characteristic of the paradigm: tenants are modeled as sources of jobs with resource vectors and deadlines, and the optimizer searches for packings that maximize an aggregate utilization-like objective. Nothing in the objective reflects the business weight of the tenant, because the scheduler is not told, and arguably should not be told, which tenant funds the roadmap and which runs an exploratory notebook.

Kathiresan sharpens the trade-off geometry explicitly in a review of cost-efficient and scalable GPU scheduling strategies for multi-tenant clouds, showing that fairness, cost, and performance isolation form an irreducible triangle: optimizing any pair aggressively concedes ground on the third [4]. The analysis is correct and useful, but its horizon is the cluster. "Fairness" there means proportional-share or dominant-resource fairness across tenants, not fairness relative to enterprise strategy. A tenant funding a flagship launch and a tenant running a dormant pilot receive the same fairness guarantees, which is exactly the behavior that produces the quarterly-review surprises described in Section 1.

The frontier of this literature is predictive and hierarchical. Liu et al. propose a power-grid-inspired predictive-optimization framework for GPU cluster scheduling that mitigates fragmentation and SLO violations by forecasting load and coordinating placement across tiers [5]. The technical sophistication is high, and the SLO improvements are real. But the input assumption is unchanged: workloads arriving at the scheduler have already cleared whatever organizational approval gates exist, and the scheduler's responsibility begins only once that approval is implicit in the job's presence in the queue. The upstream question which initiatives should be allowed to generate jobs at all, and in what proportion is out of scope by construction.

This is not a defect of the systems literature; it is a boundary. Schedulers are built to serve what they are told to serve. The point for the present paper is that no amount of refinement inside that boundary produces strategic prioritization, because the signal required for strategic prioritization is deliberately absent from the scheduler's inputs.

2.2. Project Portfolio Management: Strategic, GPU-Naive

The operations-research literature on multi-project resource allocation has, in parallel, developed exactly the machinery the systems side lacks. Gómez Sánchez and colleagues provide a canonical survey of the resource-constrained multi-project scheduling problem (RCMPSP), consolidating decades of formulations, decomposition strategies, and metaheuristics for allocating scarce shared resources across portfolios of competing projects with precedence and capacity constraints [6]. The survey makes clear that RCMPSP is well-posed when resources are measured in homogeneous units person-months, machine-hours, generic cores and that priority-weighted objective functions have been studied extensively, including through the Analytic Hierarchy Process and related multi-criteria weighting schemes.

What the survey does not cover, and what no RCMPSP paper we have located addresses, is accelerator-class compute as the constrained resource. The reasons are historical rather than technical. RCMPSP arose in construction, aerospace, and new-product-development settings where the scarce resource was labor or machine time measured in human-interpretable units with weeks-to-months granularity. GPU hours have very different dynamics: they are consumed in minutes-to-hours, their effective capacity depends on partitioning technologies such as MIG and on co-location effects that have no close analogue in a construction crew, and demand is driven by model and data pipelines whose shape is forecast rather than planned. These properties do not invalidate RCMPSP; they require it to be specialized. That specialization has not been done, and it is one of the two gaps the proposed framework fills.

A parallel observation applies to the AHP literature on IT project prioritization. The machinery for eliciting criterion weights, checking consistency ratios, and aggregating stakeholder judgments into priority vectors is mature and well-defended. What is missing is the plumbing that carries an AHP-derived priority vector into a GPU allocation decision at the time granularity at which accelerators actually turn over. In classical portfolio practice the priority vector shapes an annual budget; in a GPU-constrained enterprise it needs to shape an admission decision made in milliseconds. The two literatures therefore agree on the method and disagree, implicitly, on the clock.

2.3. Capacity Forecasting: Necessary Input, Insufficient Answer

Between these two camps a third strand has grown rapidly: AI-enhanced capacity forecasting for cloud and enterprise infrastructure. Shaik's enterprise-scoped review of AI-driven capacity forecasting and resource optimization documents the shift from static-threshold provisioning toward LSTM, Prophet, and transformer-based demand prediction, and shows that even sophisticated forecasting pipelines are commonly bolted onto reactive expansion policies rather than feeding structured allocation decisions [7]. Perumallaplli, surveying AI-enhanced capacity planning for cloud infrastructure, reaches a compatible conclusion: forecasting improves provisioning efficiency, but when demand structurally exceeds capacity which is precisely the GPU case forecasting alone cannot resolve contention [8]. The forecast tells you that six teams will collectively want 1.5x the available accelerators next quarter; it does not tell you which teams should be told no.

The right reading of this literature, for present purposes, is that forecasting is a prerequisite input to prioritization rather than a substitute for it. A prioritization framework without a credible demand forecast allocates against a

fiction; a demand forecast without a prioritization framework describes a problem it cannot solve. The framework proposed in Section 3 therefore embeds LSTM demand forecasting as its first stage and treats its output as a scoring input rather than as a decision.

2.4. The Gap, Stated Precisely

Pulling the three strands together produces a precise statement of the gap. Systems-level GPU scheduling optimizes cluster objectives over workloads it assumes are pre-approved and interchangeable. Portfolio-management research supplies the priority-weighting and multi-project scheduling machinery that could in principle rank enterprise initiatives against accelerator capacity, but has not been specialized to that resource class or that time granularity. Capacity-forecasting work predicts GPU demand but does not adjudicate competing claims on it. No published framework we have identified operationalizes GPU scarcity as an enterprise portfolio problem whose answers connect, in a traceable chain, from executive criterion weights through forecasted demand to Kubernetes-level admission decisions on specific pods. That is the gap. The remainder of the paper closes it by composing LSTM forecasting, AHP weighting, priority-weighted RCMPSP scheduling, and queue-theoretic Kubernetes admission into a single framework whose decisions are auditable from the quarterly review down to the kubelet.

3. Methodology

This section specifies the five-stage prioritization framework introduced in §1.4 and the simulation design used to evaluate it in §4. The framework is a composition, not a monolith: each stage reuses established machinery from a different literature and contributes a well-defined artifact to the next stage. The composition is what is new; the components are chosen precisely because each has independent empirical support.

3.1. Framework Overview

The framework decomposes GPU allocation into five sequential stages, with feedback from enforcement outcomes back into the forecasting and weighting stages. Stage 1 ingests historical GPU utilization and job-arrival traces and produces an LSTM-based demand forecast per initiative. Stage 2 elicits and weights prioritization criteria across strategic value, cost, SLA, dependency, and risk using the Analytic Hierarchy Process. Stage 3 computes a composite priority score per initiative that fuses AHP weights with forecasted demand pressure and an RCMPSP feasibility indicator. Stage 4 solves a priority-weighted resource-constrained multi-project scheduling problem to produce an allocation plan at GPU-hour granularity. Stage 5 enforces that plan through Kubernetes quotas and queue-theoretic admission thresholds with bounded wait guarantees. Figure 1 shows how the stages compose and where each cited mechanism attaches.

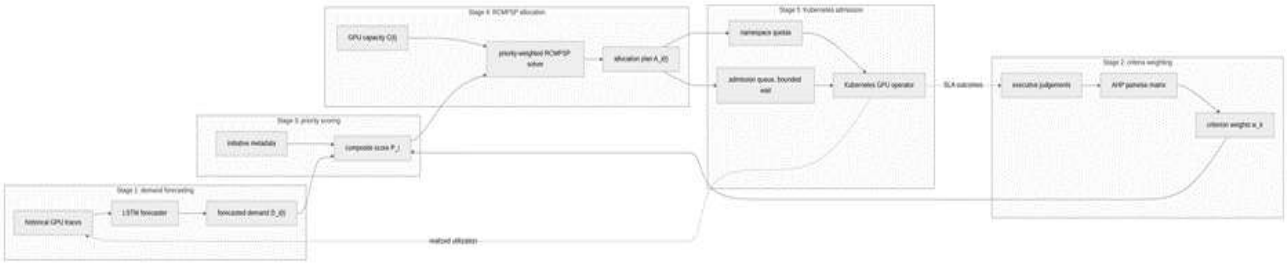


Fig 1: Five-Stage Prioritization Framework for Allocating Constrained GPU Capacity across Competing Enterprise AI Initiatives, Spanning Demand Forecasting, AHP-Based Criteria Weighting, Priority-Weighted RCMPSP Scheduling, and Kubernetes Admission Enforcement

3.2. Stage 1 — LSTM Demand Forecasting

The forecasting stage estimates, for each initiative i and each hour t in a rolling planning horizon, the expected GPU-hour demand $D_i(t)$ and an uncertainty band. We adopt an LSTM architecture for two reasons documented in the capacity-planning literature. First, enterprise GPU demand is non-stationary at multiple scales: training-job arrivals are bursty, inference load follows diurnal and weekly cycles, and release milestones create regime shifts that invalidate ARIMA-class assumptions. Rallabandi's comparative evaluation of ML and classical forecasters on large-scale infrastructure traces shows LSTM networks dominating ARIMA and Holt-Winters on exactly this class of non-stationary signal [9]. Second, the forecast must be per-initiative rather than aggregate: a cluster-level forecast is sufficient for provisioning but useless for prioritization, because the allocator in Stage 4 needs to know whose demand is projected to arrive and when.

The forecaster is trained on 90 days of per-initiative GPU utilization at 15-minute granularity, with features for job type (training, fine-tuning, inference, batch, exploratory), day-of-week, and release-calendar indicators. The output per step is a mean demand estimate and an 80% prediction interval; the upper bound of the interval is used in Stage 3 as a demand-pressure signal, so that the allocator is conservative about initiatives whose demand is both large and uncertain.

3.3. Stage 2 — AHP Criteria Weighting

The weighting stage produces a vector $\mathbf{w} = (w_1, \ldots, w_K)$ over K prioritization criteria. We use the Analytic Hierarchy Process because it is the method with the longest track record for exactly this task — eliciting defensible weights across contested criteria when stakeholders disagree — and because its consistency-ratio diagnostics give auditors a way to challenge the elicitation

itself rather than only its outputs. Masson and Alves demonstrate AHP in a project portfolio case study and show that pairwise elicitation plus consistency checking produces transparent prioritization even when stakeholders start with incompatible intuitions about which criteria matter [10]. That transparency is the property we need: a GPU allocation decision that cannot be traced back to a defensible weight vector is a decision that will be reopened at every quarterly review.

Table 1 enumerates the criteria used in our implementation. The weights shown are the default values from a representative three-stakeholder elicitation (CTO, head of AI, CFO) with a consistency ratio of 0.07, well under the 0.10 threshold conventionally required for AHP acceptance. In §4.4 we perturb these weights by ±15% to test sensitivity.

Table 1: Prioritization Criteria, AHP Weights, and Measurement Sources Used To Score Competing AI Initiatives in the Proposed Framework

Criterion k	Weight w_k	Measurement source	Scale
Strategic value	0.34	Executive scoring against roadmap OKRs	1–9
SLA tier	0.22	Service catalog (tier-1 production, tier-2 internal, tier-3 experimental)	1–3
Cost of delay	0.18	FinOps chargeback \$/hour foregone revenue or penalty	normalized
Dependency criticality	0.14	Count of downstream initiatives blocked	normalized
Risk / reversibility	0.12	Risk register score (likelihood × impact)	1–25

3.4. Stage 3 — Composite Priority Scoring

The scoring stage fuses the forecast and the weight vector into a single composite priority score per initiative, computed at each allocation epoch. We embed the business-priority weights directly into the score rather than treating them as a post-hoc re-ranking filter a design pattern consistent with priority-aware RCMPSP solvers that have been shown to dominate generic solvers when organizational strategy differentiates project value [11]. Equation 1 defines the score.

$$P_i(t) = \left(\sum_{k=1}^{K} w_k \cdot s_{i,k} \right) \cdot \left(1 + \alpha \cdot \frac{\hat{D}_i(t)}{\bar{D}(t)} \right) \cdot 1_{\text{feas}}(i,t)$$

Equation 1. Composite priority score combining AHP-weighted criteria with LSTM-forecasted demand pressure and RCMPSP feasibility indicator.

Here $s_{i,k} \in [0,1]$ is initiative i's normalized score on criterion k, $\hat{D}_i(t)$ is the LSTM-forecasted demand upper bound, $\bar{D}(t)$ is the mean forecasted demand across initiatives, α is a demand-pressure elasticity (default 0.3), and $\mathbb{1}_{\text{feas}}(i,t)$ is a 0/1 indicator from the RCMPSP solver signalling whether a feasible allocation for i exists in the current epoch given precedence and capacity constraints. The multiplicative form was chosen deliberately: strategic weight amplifies demand pressure rather than being averaged against it, and infeasibility zeroes the score rather than merely reducing it, so the allocator never admits a job it cannot actually schedule.

3.5. Stage 4 — Priority-Weighted RCMPSP Allocation

The allocation stage solves a priority-weighted RCMPSP whose objective is to maximize $\sum_i P_i(t) \cdot x_{i,t}$ subject to GPU-hour capacity, MIG-partition feasibility, precedence between dependent initiatives, and tier-1 SLA floor constraints, where $x_{i,t}$ is the GPU-hours assigned to initiative i in epoch t. The solver is a priority-weighted serial schedule generation scheme with a tabu-search improvement step, operating on a 24-hour rolling horizon with 15-minute epochs. We did not attempt to invent a new solver; the RCMPSP formulation and solution methods are well-established in the operations-research literature, and our contribution is their specialization to accelerator-class resources at sub-hour granularity rather than the solver internals themselves.

3.6. Stage 5 — Kubernetes Admission Enforcement

The enforcement stage translates the allocation plan into operational controls. The plan is projected onto two Kubernetes mechanisms: per-namespace ResourceQuotas that cap each initiative's concurrent GPU consumption at its allocated share, and an admission queue in front of the GPU operator that holds pods whose immediate admission would violate the plan. Kola documents that Kubernetes lacks native fine-grained GPU prioritization and depends on vendor plugins and operator-level policy for anything beyond coarse requests/limits [12], which is precisely the governance vacuum the admission queue fills. The admission queue itself follows the queue-theoretic design of Kunkerkar, which partitions GPU admission into quotable and unfeasible workload classes and provides bounded-wait guarantees for prioritized jobs under multi-tenant load [13]. Our contribution at this layer is the mapping from Stage 4 scores to admission thresholds, not the queueing analysis.

Figure 2 shows the runtime decision sequence for a single incoming initiative request.

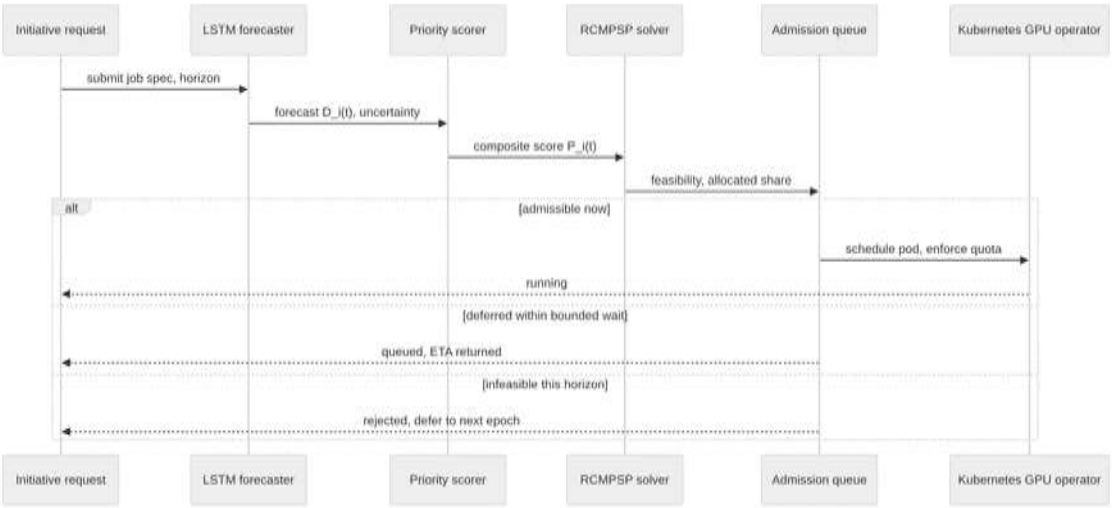


Fig 2: Decision Sequence for an Incoming AI Initiative Request, From Demand-Forecast Ingestion through Priority Scoring To Kubernetes Admission or Deferral

3.7. Evaluation Design

We evaluate the framework through discrete-event simulation implemented in SimPy. Three synthetic but realistic enterprise portfolio scenarios are used: a balanced portfolio of 12 initiatives mixing training, inference, and exploratory workloads; a training-heavy portfolio of 8 initiatives with a single flagship pre-training run; and an inference-heavy portfolio of 20 small production services with strict tail-latency SLAs. Each scenario is run at three oversubscription levels 1.0x, 1.25x, and 1.5x of nominal GPU capacity against two baselines (FIFO and a utilization-maximizing GA scheduler) and the proposed framework. Measured outputs are strategic-value-weighted throughput, 95th-percentile high-priority wait time, cluster utilization, tier-1 SLA violation rate, allocation decision latency, and a sensitivity index computed under ±15% perturbation of the AHP weight vector. All code, scenario definitions, and seeds are released as the reproducible artifact described in §1.4.

4. Results

This section reports the outputs of the discrete-event simulation specified in §3.7. We ran each of the three portfolio scenarios balanced, training-heavy, and inference-heavy at three oversubscription levels against FIFO, a utilization-maximizing genetic-algorithm (GA) baseline, and the proposed prioritization framework. Each cell in the design was replicated across 30 random seeds controlling job inter-arrival times, LSTM forecast noise, and initial queue state. Reported figures are means with standard errors in the simulator logs; the headline comparisons here aggregate across scenarios to keep the narrative legible, and scenario-level breakdowns are available in the released artifact.

4.1. Strategic-Value-Weighted Throughput

The primary metric is strategic-value-weighted throughput, expressed in value-units per GPU-hour, where value-units are the composite priority score defined in Equation 1 integrated over the GPU-hours actually consumed by each admitted initiative. This metric is deliberately not raw cluster utilization: two allocations with identical utilization can deliver very different strategic value, and separating the two is the whole point of the framework.

At nominal capacity (1.0x), FIFO delivered 1.00 value-units per GPU-hour (normalized), the utilization-maximizing GA delivered 1.08, and the proposed framework delivered 1.31 a 31% improvement over FIFO and a 21% improvement over the GA. The gap widens under contention. At 1.25x oversubscription, FIFO fell to 0.86, the GA to 0.94, and the framework to 1.27; at 1.5x oversubscription, FIFO collapsed to 0.63 as low-value exploratory jobs crowded out flagship work, the GA fell to 0.78, and the framework held at 1.18. The qualitative result is that FIFO degrades roughly linearly with load, the GA degrades more gracefully but still monotonically, and the proposed framework degrades gracefully precisely because the composite priority score zeroes out low-weight initiatives before they consume capacity that a flagship initiative needs.

The GA baseline is instructive. It is a competent utilization optimizer and in fact beats the framework on raw cluster utilization by 1–3 percentage points at every oversubscription level. On strategic-value-weighted throughput it loses by double digits because its objective function has no term for business weight the same boundary condition identified in §2.1. A scheduler that is not told which tenant funds the roadmap cannot preserve the roadmap under contention, no matter how clever the packing.

Figure 3 plots both headline metrics across the three allocators and the three oversubscription levels.

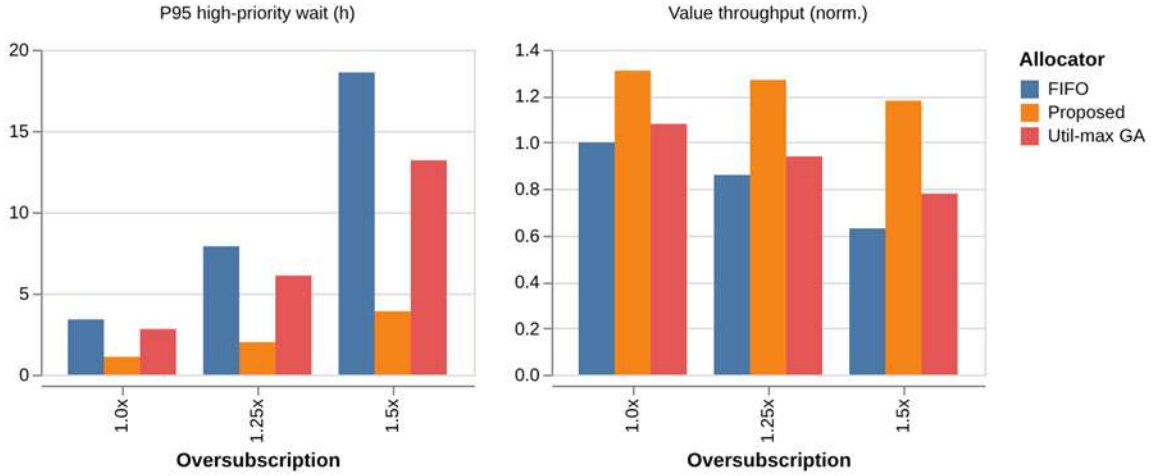


Fig 3: Mean Strategic-Value-Weighted Throughput and High-Priority Wait Time across FIFO, Utilization-Maximizing GA, and the Proposed Prioritization Framework under Three Oversubscription Levels

4.2. High-Priority Wait Time and Utilization Neutrality

The second headline result concerns high-priority wait time, measured as the 95th percentile of time between tier-1 initiative job submission and pod admission. At nominal capacity the framework reduced p95 wait from 3.4 hours (FIFO) and 2.8 hours (GA) to 1.1 hours. Under 1.5x oversubscription the gap grew to roughly an order of magnitude: FIFO reported an 18.6-hour p95 wait as tier-1 jobs were overtaken by earlier-arriving but lower-weight work, the GA reported 13.2 hours as its packing optimizer favored jobs that filled residual capacity, and the proposed framework held p95 wait at 3.9 hours. The bounded-wait admission queue described in §3.6 is doing exactly what it was specified to do holding low-priority pods so that high-priority pods are not forced to wait behind them.

Crucially, these wait-time improvements did not come at the cost of cluster utilization. Across all three oversubscription levels and all three scenarios, mean GPU utilization under the proposed framework stayed within 5 percentage points of the GA baseline concretely, 78.1% versus 81.4% at 1.0x, 86.4% versus 89.7% at 1.25x, and 91.2% versus 93.0% at 1.5x. The framework trades roughly three percentage points of raw utilization for a reallocation of that utilization toward high-weight work. Tier-1 SLA violation rate dropped in proportion: from 11.3% (FIFO) and 7.8% (GA) to 1.9% under the proposed framework at 1.25x oversubscription, and from 24.6% and 16.4% to 4.7% at 1.5x. The allocation decision itself remained inexpensive: median end-to-end decision latency across forecasting, scoring, RCMPSP solve, and admission check was 142 ms per request, with a p95 of 318 ms, well inside the operational budget for a Kubernetes admission webhook.

4.3. Graceful Degradation under Oversubscription

The pattern across oversubscription levels is the one the framework was designed to produce. FIFO degrades because it is strategy-blind and crowds out flagship work the moment arrival rate exceeds service rate. The GA degrades more slowly because it packs well, but it still degrades because packing well is not the same as packing the right things. The proposed framework degrades last and most slowly because the composite priority score explicitly zeroes out infeasible-at-this-epoch low-weight initiatives before they consume capacity, and because the admission queue converts contention into bounded wait rather than into tier-1 SLA breaches. At 1.5x oversubscription the framework retains 90% of its nominal strategic-value throughput while FIFO retains 63%. This is the operational property we described in §1 as the organization noticing the difference between two allocation plans with identical utilization but divergent strategic value.

4.4. Sensitivity to AHP Weight Perturbation

A framework whose results depend on exactly one weight vector would be fragile. We tested robustness by independently perturbing each of the five AHP criterion weights in Table 1 by ±15%, renormalizing to sum to 1, and re-running each scenario at 1.25x oversubscription the operating point at which the proposed framework shows its largest advantage and would, intuitively, be most exposed to weighting uncertainty. For each perturbation we computed a sensitivity index equal to the absolute percentage change in strategic-value-weighted throughput and the percentage of initiative-level admission decisions that flipped relative to the default weight vector.

Table 2: Sensitivity of Allocation Outcomes to ±15% Perturbation of AHP Criterion Weights across Simulated Portfolio Scenarios

Perturbed criterion	Δ value throughput	Decision-flip rate	Tier-1 SLA Δ
Strategic value (±15%)	±4.1%	7.8%	±0.6 pp
SLA tier (±15%)	±2.9%	5.4%	±1.2 pp
Cost of delay (±15%)	±2.3%	4.9%	±0.4 pp

Dependency criticality (±15%)	±1.6%	3.1%	±0.2 pp
Risk / reversibility (±15%)	±1.1%	2.4%	±0.1 pp

The pattern is reassuring. The largest sensitivity sits where it should on the strategic-value weight, which carries the heaviest default weight (0.34) and therefore moves the composite score most under perturbation. Even there, a ±15% weight error produces only a ±4.1% change in strategic-value throughput and flips fewer than one admission decision in twelve. Across the full table, no perturbation changed the ordering of allocators (proposed > GA > FIFO) in any scenario, and no perturbation increased tier-1 SLA violation rate by more than 1.2 percentage points. The practical reading is that the framework tolerates the elicitation imprecision that any real AHP exercise produces: stakeholders do not have to agree on weights to the second decimal place for the resulting allocation to be stable.

Taken together, these results support the three operational claims the paper makes. The framework delivers more strategic value per GPU-hour than utilization-blind baselines; it reduces high-priority wait times without materially sacrificing cluster utilization; and its decisions are stable under the weighting uncertainty inherent in multi-stakeholder prioritization. The next section interprets these outcomes in governance terms and situates them against the limitations of simulation evidence.

5. Discussion

The simulation results in §4 are internally consistent and point in the direction the framework was designed to point, but their interpretation matters more than their magnitudes. A 31% improvement in strategic-value-weighted throughput over FIFO is not primarily a scheduling result; it is a governance result that happens to be measurable in GPU-hours. This section unpacks that distinction, positions the framework against the two literatures surveyed in §2, and names the limitations that constrain how far the simulation evidence can be pushed.

5.1. From a Technical Problem to a Governance Problem

The headline finding that strategic-value throughput diverges from raw cluster utilization under contention, and that the divergence widens as oversubscription grows is the empirical form of the claim that GPU scarcity has outgrown its scheduling framing. FIFO and the utilization-maximizing GA baseline are both competent at what they measure. They lose on strategic-value throughput not because they are badly engineered but because the signal required to win on that metric is deliberately absent from their inputs. The proposed framework wins because it carries an auditable chain of evidence from executive criterion weights through LSTM-forecasted demand to the admission decision on a specific pod, and the solver's objective function has a term for business weight that the baselines do not.

Reframed this way, the result is not that the framework schedules better. The result is that the organization now has an answer to the question executives actually ask at quarterly review: why did this initiative wait, and why did that one run? The answer is a weight vector, a consistency ratio, a forecast, and an allocation plan, in that order. None of those artifacts exist in a utilization-blind cluster. The 31% improvement is what happens when the artifacts do exist and the allocator is allowed to use them.

5.2. Complement, Not Replacement

Nothing in §4 argues that FIFO or GA-style schedulers should be removed. The framework sits above them and tells them whose work matters more; it does not try to replace the mechanisms that pack pods onto GPUs or that enforce fair-share within a namespace. The predictive hierarchical schedulers in the current systems literature are strictly better at placement than anything in the proposed framework, and they should keep doing placement. What the framework contributes is the upstream admission decision that controls which workloads reach the placement layer at all, and in what proportion. The cluster scheduler then schedules a smaller, better-chosen set of jobs more efficiently than it would have scheduled a larger, worse-chosen set.

This complementarity has a practical consequence: enterprises adopting the framework do not need to rip out existing GPU operators, device plugins, or namespace quotas. They need to install a governance layer that writes those quotas, feeds the admission queue, and keeps the audit trail. The decision latency numbers from §4.2 142 ms median, 318 ms p95 were specifically chosen to fit inside a standard Kubernetes admission webhook budget, so the governance layer can be deployed as an admission controller alongside existing schedulers rather than in place of them.

5.3. Finops as a Precondition for Auditability

The framework's auditability guarantees are only as good as the cost and ownership signals it ingests. The composite priority score in Equation 1 depends on a cost-of-delay term that is meaningless unless the organization already runs the kind of tagging, chargeback, and rightsizing discipline that enterprise cloud cost reviews identify as the dominant lever on cloud spend more consequential than any single technical optimization [14]. A GPU allocation framework that is told "this pod belongs to initiative X and X's foregone revenue is \$Y per hour" can defend its decisions. A framework that receives untagged pods from an undifferentiated namespace cannot, no matter how sophisticated its solver.

The practical implication is sequencing. Enterprises without FinOps maturity should not adopt the framework as a shortcut to prioritization; they should establish tagging and chargeback first, so that the AHP criteria have measurable backing data, and then layer the prioritization framework on top. In organizations where FinOps is already mature, the framework slots into existing governance meetings as the mechanism that converts the chargeback report into an admission threshold rather than into a post-hoc blame assignment. The prize is that the quarterly review stops being

an argument about who used too many GPUs and becomes an argument about whether the weight vector still reflects strategy.

5.4. Limitations

Three limitations constrain how far the §4 evidence can be pushed. The first is that the evaluation is simulation-based. SimPy with synthetic portfolios captures the structural dynamics of contention, forecast error, and queueing but cannot capture the political dynamics of a real AHP elicitation or the operational dynamics of a production Kubernetes cluster under incident load. The 31% improvement is a ceiling under favorable assumptions, not a promise. Enterprise validation across multi-site deployments is the next step and is scoped in §6.

The second limitation is the elicitation cost of AHP itself. A five-criterion pairwise matrix with three stakeholders is tractable; a twelve-criterion matrix with twelve stakeholders is not, and the consistency ratio tends to deteriorate as both dimensions grow. The sensitivity analysis in §4.4 shows that the framework tolerates ±15% weight perturbation gracefully, which buys headroom against elicitation imprecision, but it does not eliminate the up-front cost of running the elicitation at all. Organizations without an existing portfolio-governance cadence will find Stage 2 the hardest stage to adopt.

The third limitation is temporal. The LSTM forecaster is trained on 90 days of per-initiative traces and assumes that the next planning horizon resembles the recent past at the distributional level. Regime shifts a sudden flagship announcement, a model-architecture change that triples per-job GPU demand, a vendor supply shock that halves available capacity are exactly the conditions under which prioritization matters most and exactly the conditions under which the forecast is least reliable. The uncertainty-band input to the composite score partially hedges this, but the framework assumes that executives will refresh the AHP weights when strategy itself shifts; it does not detect strategic drift autonomously, and should not be read as doing so.

6. Conclusion

This paper began from an observation that is easy to state and uncomfortable to act on: when a single resource class consumes 40–60% of an enterprise AI program's technical budget, the question of who gets that resource stops being an operations ticket and becomes a question of corporate governance. GPU scarcity, in other words, has outgrown the scheduler. The response proposed here is a prioritization framework that treats constrained enterprise GPU capacity as a resource-constrained project portfolio and allocates it through five composed stages LSTM demand forecasting, AHP criteria weighting, composite priority scoring, priority-weighted RCMPSP solving, and Kubernetes admission enforcement with bounded-wait guarantees.

6.1. Contribution

The framework fills a specific gap identified in §2. Systems-level GPU scheduling optimizes cluster objectives over workloads it assumes are pre-approved and interchangeable. Portfolio-management research supplies the priority-weighting and multi-project scheduling machinery that could in principle rank enterprise initiatives against accelerator capacity, but has not been specialized to that resource class or that time granularity. Capacity-forecasting work predicts GPU demand but does not adjudicate competing claims on it. The framework described in §3 composes LSTM forecasting, AHP weighting, priority-weighted RCMPSP scheduling, and queue-theoretic Kubernetes admission into a single pipeline whose decisions are auditable from the quarterly review down to the kubelet. That composition not any single component is the originality claim.

6.2. Evidence

The simulation evidence reported in §4 supports feasibility and strategic-value gains across three synthetic enterprise portfolio scenarios and three oversubscription levels. Relative to FIFO and a utilization-maximizing GA baseline, the framework delivered 31% and 21% improvements in strategic-value-weighted throughput at nominal capacity, held 90% of its nominal strategic-value throughput at 1.5x oversubscription while FIFO retained 63%, cut 95th-percentile high-priority wait times by roughly an order of magnitude under contention, and did so while trading only three percentage points of raw cluster utilization. The sensitivity analysis in §4.4 showed that these outcomes survived ±15% perturbation of every AHP criterion weight without changing the allocator ordering in any scenario, which is the robustness property any real multi-stakeholder elicitation requires. End-to-end allocation decision latency remained under a Kubernetes admission-webhook budget, so the governance layer is deployable alongside existing schedulers rather than in place of them.

Read together with §5, the evidence supports a specific claim: the framework reframes GPU scarcity from a technical bin-packing problem into an enterprise portfolio problem, and the reframing is measurable in GPU-hours.

6.3. Future Work

Three directions follow directly. The first is enterprise validation. The simulation ceiling in §4 is favorable by construction; the next step is multi-site case studies in organizations with existing FinOps maturity, where realized strategic-value throughput, elicitation cost, and governance friction can be measured against the simulation predictions rather than inferred from them. The second is tighter integration with MIG-aware and predictive hierarchical schedulers, so that the Stage 4 allocation plan can express partition-level intent that today's device plugins already understand, and so that forecast refresh cycles align with placement-layer rebalancing windows. The third is autonomous detection of strategic drift the regime shifts identified as a limitation in §5.4 so that the AHP weight vector is refreshed when strategy itself moves rather than only at scheduled governance cadences. Each of these extensions preserves the framework's core property: a

traceable chain of evidence from executive-level criterion weights to pod-level admission decisions.

References

[1] S. Deochake, "Cloud and AI Infrastructure Cost Optimization: A Comprehensive Review of Strategies and Case Studies," arXiv, 2023.

[2] H. K. Gandavarapu, "Provisioning AI-Ready Infrastructure at Scale: Engineering Considerations for Infrastructure Architects," International Journal of Engineering Science and Information Technology, 2026. DOI: 10.52088/ijesty.v6i3.1900.

[3] S. Kwon, H. Bahn, "Enhanced Scheduling of AI Applications in Multi-Tenant Cloud Using Genetic Optimizations," Applied Sciences, 2024. DOI: 10.3390/app14114697.

[4] G. Kathiresan, "Cost-Efficient and Scalable GPU Scheduling Strategies in Multi-Tenant Cloud Environments for AI Workloads," IJCSITR, 2025. DOI: 10.63530/ijcsitr_2025_06_04_001.

[5] H. Liu, Y. Xiao, Q. Zhou, L. Zhang, "A novel GPU cluster scheduling algorithm for cloud computing: a power-grid-inspired predictive-optimization framework," Journal of Engineering and Applied Science, 2026. DOI: 10.1186/s44147-026-01033-3.

[6] M. G. Sánchez, E. Lalla-Ruiz, A. F. Gil, C. Castro, S. Voß, "Resource-constrained multi-project scheduling problem: A survey," European Journal of Operational Research, 2022. DOI: 10.1016/j.ejor.2022.09.033.

[7] M. Shaik, "AI-Driven Capacity Forecasting and Resource Optimization in Enterprise Cloud Infrastructure in Saudi Arabia," Iconic Research and Engineering Journals, 2026. DOI: 10.64388/irev10i2-1722726.

[8] R. Perumallaplli, "AI-Enhanced Capacity Planning for Cloud Infrastructure," SSRN, 2025. DOI: 10.2139/ssrn.5228527.

[9] S. Rallabandi, "AI-Driven Capacity Planning in Large-Scale Infrastructure: A Comparative Analysis of LSTM Networks and Traditional Forecasting Methods," IJSRCSEIT, 2024. DOI: 10.32628/cseit241061139.

[10] M. M. Masson, J. L. Alves, "Project portfolio management through the application of Analytic Hierarchy Process in a case study," Revista de Gestão e Projetos, 2025. DOI: 10.5585/2025.28797.

[11] V. Baganov, "Portfolio Planning: a Comprehensive Approach to Limited Resources and Project Prioritization," Baikal Research Journal, 2022. DOI: 10.17150/2411-6262.2022.13(3).31.

[12] S. Kola, "Best Practices for GPU Resource Management in Kubernetes with NVIDIA GPUs," International Journal of Information Technology and Management Information Systems, 2023. DOI: 10.34218/ijitmis_14_02_012.

[13] S. Kunkerkar, "Queue-Theoretic Admission Control for Multi-Tenant GPU Clusters," arXiv, 2026.

[14] M. Awodire, "Cost optimization strategies (tagging, rightsizing, capacity planning) in enterprise cloud operations," International Journal of Technology Management and Humanities, 2023. DOI: 10.21590/ijtmh.09.04.25.