

HAIL: Prediction-Calibrated Hierarchical Resource Isolation for Multi-Tenant Cloud Platforms

Lana Kapoor *, Lena Huang, Serena Arora

Department of Computer Science and Engineering, University of Minnesota Twin Cities, Minnesota, United States

Corresponding author: Lana Kapoor. (Email: lana3849@umn.edu)

Abstract: Multi-tenant cloud platforms must reserve capacity for protected workloads while avoiding needless loss of best-effort service. This paper presents HAIL, a hierarchical adaptive isolation-and-lending controller operating across CPU, memory, network, and storage. It maps a three-layer isolation taxonomy to physical-capacity constraints, host-local quota control, and application-level demand coverage. Four gradient-boosted predictors forecast next-interval demand; independent validation residuals supply one-sided protection margins, and bounded online error feedback increases reservations for persistently underpredicted protected tenants. Unallocated priority capacity is made available to batch tenants through a deterministic feasible projection. We conduct reproducible experiments with synthetic but temporally correlated and bursty tenant traces, twelve disjoint test seeds, five comparison controllers, an oracle upper bound, and targeted ablations. With causal historical profiles, HAIL reduces modeled protected shortfalls from 24.11% for a hierarchical profiling-and-lending baseline to 8.86% (paired 95% interval: -19.47 to -11.04 percentage points); utilization decreases from 87.15% to 86.44%. Stress tests reveal that strict priority can starve best-effort demand under severe scarcity. These are simulated resource-coverage results, not measured request latencies. The released experiment package preserves code, generated demands, seed splits, and per-episode results.

Keywords: Multi-tenancy; Resource isolation; Cloud scheduling; Calibrated prediction; Quality of service; Capacity lending.

1. Introduction

Cloud consolidation makes isolation a scheduling decision rather than simply an access-control setting. Co-located services compete for processor time, memory bandwidth, network queues, and storage I/O even when their address spaces are disjoint. The tail-at-scale effect makes a small fraction of overloaded requests disproportionately important to user-visible performance [1]. Cluster orchestration systems such as Borg, Omega, and Kubernetes expose resource requests and limits, yet their operational goal remains high utilization across competing tenants [2]. Modern production schedulers compound this tension by admitting heterogeneous services and batch jobs onto the same shared hosts [3].

An informative starting point is the hierarchical resource-isolation taxonomy of Ding et al. [4]. Their survey identifies CPU, memory, network, and storage as four coupled resource domains, and distinguishes hardware-assisted, operating-system, and orchestration layers. This classification is directly actionable: a quota controller must respect physical capacity, generate enforceable per-tenant limits, and assess application outcomes. Its accompanying discussion of interference-aware profiling and recoverable idle-capacity lending also suggests a useful control structure. This work converts these elements into an executable four-domain, three-layer allocation model instead of treating isolation as a single CPU threshold.

Two difficulties arise when allocations are updated only once per control interval. First, reactive quotas lag a burst by at least one decision period. Second, a prediction that is unbiased on average can still be systematically unsafe for an SLO-protected tenant when underprediction is more costly than overprediction. Merely increasing all reservations is unattractive because the resulting idle headroom reduces best-effort service and resource efficiency. Consequently, the control problem is not just forecasting demand: it is deciding

where to place a finite amount of protective headroom and how to redistribute the residual capacity.

We develop HAIL (Hierarchical Adaptive Isolation and Lending), a prediction-calibrated quota controller. HAIL fits a separate gradient-boosted demand model for each resource, adds a validation-derived positive-error margin only to protected tenants, and adjusts that margin from recent forecast surprises. It then allocates priority quotas within each host capacity and lends the remainder to best-effort tenants. The policy has a deterministic feasibility constraint and leaves the forecasting method replaceable.

Three contributions are evaluated: (i) an explicit operational mapping from a four-domain, three-layer isolation taxonomy to a quota-and-service simulator; (ii) uncertainty-sensitive, feedback-adjusted allocation that distinguishes priority protection from opportunistic lending; and (iii) a reproducible comparison against fixed, proportional, reactive, hierarchical, and mean-forecast policies, including two targeted ablations and paired uncertainty intervals. All claims in this paper concern a fully specified synthetic trace experiment rather than a live cloud deployment.

2. Related work and positioning

Multi-resource allocation is richer than CPU proportional sharing. Dominant resource fairness provides an axiomatic account of fairness across heterogeneous demands [5]; Tetris shows that accounting for all resource dimensions helps avoid packing fragmentation [6]. HAIL addresses a different control horizon: tenants are already co-located on a host, and quotas must be set before the next demand realization. The proportional scaling policy in our experiments is an intentionally simple multi-resource comparator, not an implementation of dominant resource fairness.

Placement and interference profiling are complementary.

Paragon uses prior workload information to classify interference and guide online placement [7], while Quasar incorporates QoS objectives into cluster management [8]. Heracles pairs application feedback with mechanisms for safe co-location on shared servers [9]. These systems motivate domain-aware decisions, but their underlying placement or hardware mechanisms are not assumed in our simulator.

Microservice controllers highlight the need to react to SLO stress: FIRM develops fine-grained, SLO-oriented resource management [10]. The variability of real serverless workloads is documented by Shahrad et al. [11]. At VM scale, Protean studies practical allocation in the presence of priority and heterogeneous VM types [12], while SLO-protected harvesting examines how idle capacity may be exploited for subordinate work [13]. Such studies provide relevant operational context; their experimental results are not transferred into our evaluation. PRESAGE offers a closely related cloud-systems example of translating workload forecasts and predictive uncertainty into SLO-aware elastic scheduling decisions [25].

Learning has been applied to cluster scheduling [14], and production autoscaling controllers use workload observations to reduce administrative overhead [15]. Resource Central provides an important example of prediction supporting cloud management at scale [16]. HAIL makes a narrower claim than these platforms: it evaluates a one-step predictor coupled to bounded, host-local isolation quotas. Its distinction is how one-sided forecast error interacts with priority and four separate capacities, rather than the particular gradient-boosting estimator.

3. Resource and objective model

Consider a single host with $n = 12$ tenants. The protected set P contains four tenants and the opportunistic set B contains eight. For each decision step t and resource d in $D = \{\text{CPU, memory, network, storage}\}$, the realized nonnegative demand $x(i, d, t)$ is expressed in normalized capacity units. Every resource has host capacity $C(d) = 10$. At the start of t , the controller chooses an allocation $a(i, d, t)$ using information available only through $t - 1$. It must satisfy $a(i, d, t) \geq 0$ and $\sum_i a(i, d, t) \leq C(d)$ in every domain.

The application-level service proxy for tenant i is $s(i, t) = \min \text{ over } d \text{ of } \{1, a(i, d, t) / \max[x(i, d, t), \epsilon]\}$. This bottleneck coverage ratio represents the fraction of a multidimensional demand vector that its most constrained resource can support. It is deliberately not a response-time prediction. A protected shortfall event occurs when $s(i, t) < 0.95$. Our primary metric V is the mean of that binary event over protected tenant-steps. Secondary metrics are mean batch service S_B , realized host utilization U , and the 95th percentile of protected service deficit $1-s$.

Realized resource use is $u(d, t) = \sum_i \min\{a(i, d, t), x(i, d, t)\}$, with U the average of $u(d, t)/C(d)$ over dimensions and time. This quantity discounts reserved but unconsumed quota; total provisioned quotas would be a misleading utilization surrogate. Batch service reflects the fraction of its own demand that a tenant receives, averaged equally across batch tenant-steps. These definitions make the protection-efficiency tradeoff auditable but do not infer kernel scheduling delays, throughput in requests/s, or wall-clock P99 latency.

Priority-aware allocation is feasible by construction. For each resource independently, requested protected quotas are proportionally reduced only if their sum exceeds $C(d)$. The

residual $C(d)$ minus protected allocation is then distributed among batch requests in proportion to their target demands, with no allocation beyond those targets. Each domain therefore has a nonnegative, capacity-bounded allocation, irrespective of forecast accuracy. The remaining research question is how the target vector should be formed before this deterministic projection.

4. Hail: calibrated hierarchical control

A. Making the isolation hierarchy operational. Figure 1 shows three stages implemented in the experiment. The hardware layer appears as the per-domain physical capacity vector C . The OS layer is represented by the feasible quota vector a , analogous to enforcing CPU, memory, traffic, and I/O ceilings. The application layer evaluates the bottleneck coverage proxy and reports the demand observations used in an error signal. This operationalizes the resource-domain and layer decomposition of Ding et al. [4] as an executable allocation model. In addition, the taxonomy-and-lending baseline uses four-domain demand profiles with priority-first allocation and reassigns predicted unneeded quota to batch tenants at the beginning of each interval. It provides a concrete operational comparator derived from that review's profiling and capacity-reuse discussion.

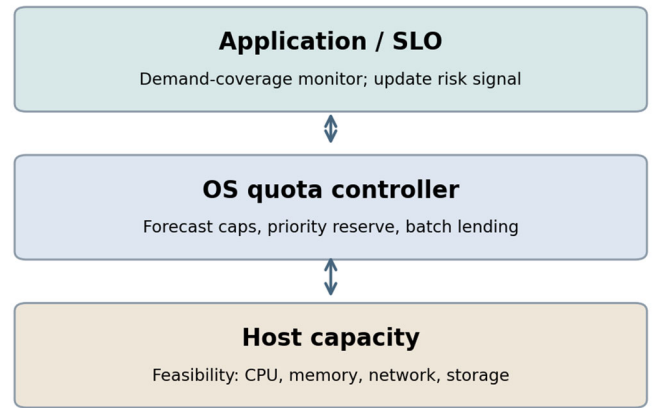


Fig. 1 Three-layer operational mapping applied independently to four resource domains.

B. Forecasting from observable history. HAIL trains four independent histogram-based gradient-boosting regressors, one per resource dimension. At each step the 33-component feature vector contains one-, two-, and three-step lagged demand in all domains; a six-step trailing mean; the difference between steps $t-1$ and $t-3$; a tenant profile estimated exclusively from the first six observed warmup steps; aggregate lagged host load; the protected/batch indicator; and four sinusoidal time features. In particular, no latent base profile from the synthetic generator is supplied to any controller. The regressor predicts step t using only data available through $t-1$. Gradient boosting is a conventional supervised choice for nonlinear tabular signals [17], not a new learning architecture.

C. One-sided calibration. For independent validation episodes, let $e(d) = \max\{0, x(d) - \hat{x}(d)\}$ denote positive underprediction residuals, pooled over tenants and steps. We set $q(d)$ to their empirical 90th percentile, obtaining 0.1312, 0.1304, 0.1078, and 0.1056 normalized units for CPU, memory, network, and storage, respectively. These values remain frozen at test time. The construction uses a simple

empirical validation quantile related to predictive uncertainty calibration [18]; it is not a formal conformal procedure and confers no distribution-free coverage or real-world SLO guarantee.

D. Online protection adjustment. For protected tenant i , HAIL maintains a four-dimensional nonnegative surprise state $h(i, d, t)$. After each step, h is updated by exponential smoothing of $\min\{\max[0, x(i, d, t) - \hat{x}(i, d, t)] / \max[q(d), 0.04], 3\}$; the new state is 0.85 times the old value plus 0.15 times this surprise. The next protected target is $z(i, d, t+1) = \hat{x}(i, d, t+1) + q(d)[1 + 0.65 \min\{h(i, d, t), 1\}]$. Batch targets remain their point forecasts. The uncertainty premium is therefore priority-selective, monotone in recent underprediction, and bounded. The bound avoids allowing a single adverse step to permanently remove capacity from batch work.

E. Priority projection and lending. Let $Z_P(d)$ be the sum of protected targets for domain d . If $Z_P(d) \leq C(d)$, each protected tenant receives its target; otherwise all protected targets are scaled by $C(d)/Z_P(d)$. If $R(d)$ is the remaining capacity and $Z_B(d)$ the total batch target, batch tenant i receives $z(i, d) \min\{1, R(d)/Z_B(d)\}$. This is applied independently to CPU, memory, network, and storage. Reserving predicted protection before batch admission makes the first stage conservative; the second is work-conserving relative to requested batch targets at decision time. Since consumption is revealed later, unused in-interval quotas are not reclaimed within that same interval.

F. Controller state and cost. The controller requires six historical steps for feature construction, four model outputs per tenant, and an $O(n|D|)$ quota projection. The learning stage is separate from online decisions, and the deterministic projection is independent of the ML library. Invalid or unavailable model predictions can be replaced with previous observations within the same feasibility mechanism, but this fallback is not activated in the reported experiments. Unlike placement optimization, our control loop neither migrates tenants nor changes VM admission.

G. Exact decision sequence. In each interval, all strategies first receive lagged tenant demand vectors and known host capacities. HAIL then (1) forms the 33-component input representation from telemetry up to $t-1$, (2) obtains four one-step predictions per tenant, (3) augments only protected predictions with the calibrated margin and previous surprise state, (4) projects protected targets independently onto each host capacity, (5) shares residual budget among batch predictions, and (6) observes actual service fractions and updates the surprise state for the following interval. Model fitting and the empirical calibration step are outside this per-step path. The order of operations matters: using the current step demand at stage (3) would create a non-causal oracle, and performing the batch projection before protected admission would invalidate priority protection. Each controller uses the same quota-projection routine of the released code, allowing direct comparison of the decision targets.

H. Design rationale. The extra protected target $q(d)$ directly addresses the asymmetric cost of underprediction. The online state distinguishes recurrent difficult tenants from consistently predictable ones, while the independent resource projection prevents a favorable CPU forecast from masking a network or memory bottleneck. A fixed reservation can reduce shortfalls, but only at the expense of constantly holding headroom. The learned target attempts to allocate headroom where future demand and past errors suggest it will

be useful.

5. Experimental design

A. Workload provenance. We use a fully synthetic, openly defined workload generator so that burst timing, cross-resource correlation and the priority mix are known and can be varied. No sampled or measured tenant trajectories from Azure, Alibaba, or Google were used. Each episode has 400 five-minute-equivalent control steps and 12 tenants. Demand is generated from tenant-specific four-dimensional base profiles, autoregressive noise, synchronized low-frequency load variation, periodic components, and randomly arriving finite-duration bursts. CPU, memory, network, and storage have correlated but nonidentical excursions. A generated control step is an abstract time unit; it is not a time measurement of a deployed system.

B. Reproducibility and causal separation. Six seeds (101, 103, 107, 109, 113, 127) generate training episodes, two seeds (131, 137) calibrate the residual margins, and twelve disjoint seeds (149-199; fully enumerated in the archive) evaluate held-out performance. The first six observations in every episode supply an estimated tenant demand profile and warmup statistics for every controller. No generator-internal tenant parameters, current-step realizations, or future samples enter predictive features or policy decisions. An executable causal test changes future demands and verifies that all earlier feature rows remain unchanged. Traces, predictors, metrics and experiment configuration are retained.

C. Model and diagnostics. Four histogram gradient boosting models use a maximum depth of five, 95 boosting iterations, learning rate 0.07, L2 regularization 2.0 and random states 2026-2029. Models are fitted on training episodes, while positive residual quantiles use validation examples; no test-seed optimization was performed. We record resource-wise held-out forecast error and the empirical proportion of protected observations that exceed each margin. The same workloads, capacity vectors, quota feasibility routine, and observed-history restrictions apply to the compared controllers.

D. Policies. Fixed uses the average of the six warmup demands (scaled by 1.04) as a constant cap vector. Proportional scales the preceding-step demand to fit each capacity without a priority stage. Reactive targets 1.05 times the previous observation, serving protected tenants first. Taxonomy-and-lending implements the interference-aware profiling and idle-capacity lending discussed in [4]: it uses a four-domain weighted mean of previous demand and warmup profile, with a 1.08 protected multiplier, then performs the priority/lending projection. ML mean uses the uncalibrated prediction. HAIL adds calibrated margins and feedback. Oracle sees current demand and establishes a non-causal capacity upper bound, not a deployable baseline.

E. Ablations and fairness. The no-uncertainty variant is equivalent to ML mean, isolating the one-sided margin. No-feedback keeps the validation margin but freezes its online multiplier. CPU-only retains ML/margin adaptation in CPU but uses lagged protected caps for memory, network, and storage. All methods except the proportional comparator employ the same protected-first projection; no method receives additional physical capacity. This isolates target construction from quota-feasibility implementation.

F. Inference and statistical unit. Primary results average each metric first within an entire episode, then across twelve independent test seeds. Paired differences use the same seed

for both policies. We report two-sided 95% Student-t intervals on the twelve differences; such intervals summarize variation across this generator's scenarios, not sampling uncertainty about a real cloud population. The per-tenant event is a modeled demand shortfall, not observed SLO latency. Prior studies on end-to-end microservice latency [19] underline why this proxy cannot be interpreted as actual P99 response time.

6. Results

A. Protection and efficiency. Table I and Fig. 2 show that HAIL reaches an 8.86% protected shortfall rate, compared with 24.11% for taxonomy-and-lending, 57.99% for reactive, 69.57% for ML mean, and 36.95% for fixed caps. The reduction from the hierarchical baseline is 15.25 percentage points (63.3% relative). Across twelve paired test episodes, the 95% interval is -19.47 to -11.04 percentage points. The improvement is conditional on the synthetic burst generator and the 0.95 resource-coverage threshold.

TABLE I. Test-episode mean performance (12 seeds)

Method	Shortfall %	Host use %	Batch svc %
Fixed	36.95	83.95	78.76
Proportional	79.56	87.19	85.95
Reactive	57.99	86.98	82.80
Hierarchy + lending	24.11	87.15	83.12
ML mean	69.57	87.02	85.23
HAIL	8.86	86.44	80.67
Oracle (hindsight)	0.00	90.82	88.87

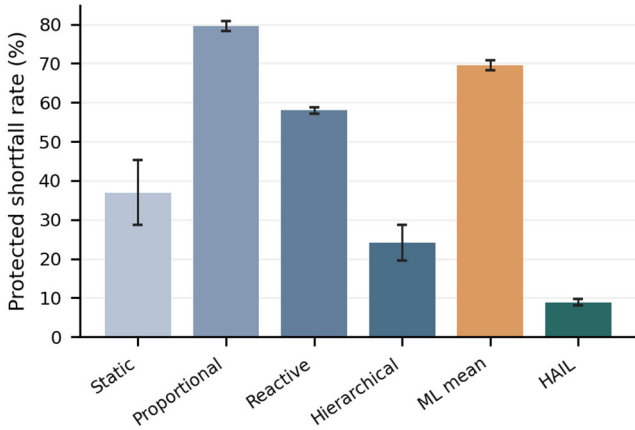


Fig. 2 Protected shortfall rates; error bars show approximate 95% intervals across seeds.

The protection gain has a measurable cost. HAIL realizes 86.44% mean host utilization against 87.15% for taxonomy-and-lending, a 0.70-percentage-point reduction (paired interval -0.98 to -0.43). Batch demand coverage decreases from 83.12% to 80.67% (-2.45 percentage points). Reserving predictive headroom for the protected set therefore reduces best-effort supply. The fixed controller realizes 83.95% utilization but suffers more protected shortfalls, illustrating that lower utilization alone does not guarantee better isolation.

B. Resource-specific implications. Figure 3 shows episode-level utilization versus shortfall. Proportional scaling achieves 87.19% utilization but a 79.56% protected shortfall rate because it does not give priority to the protected set. In contrast, the hindsight oracle uses 90.82% of capacity with no protected shortfall in these episodes. It accesses current demand, and the synthetic generator rarely saturates protected

aggregate demand at full capacity; the oracle is therefore a diagnostic upper bound, not a deployable policy.

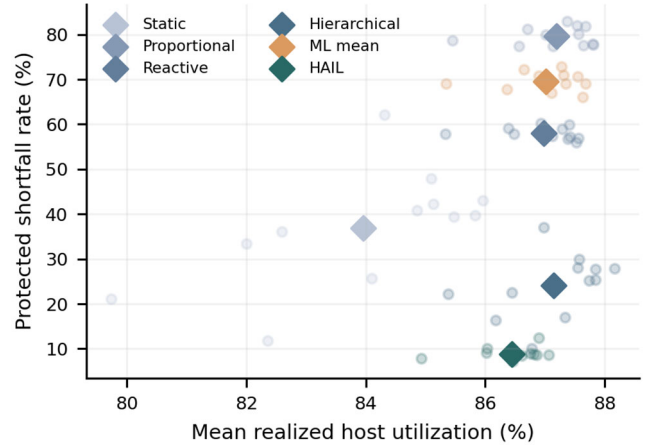


Fig. 3 Protection-utilization operating points; circles are episode results, diamonds are means.

C. Effect of predictive calibration. Uncalibrated ML mean produces a 69.57% shortfall rate at 87.02% utilization; strict minimum coverage over four resource domains amplifies small point-prediction errors. Adding the fixed validation margin reduces shortfalls to 13.34%, and activating bounded per-tenant surprise feedback lowers them further to 8.86%. The paired HAIL minus no-feedback change is -4.48 percentage points (95% interval -5.00 to -3.97). This identifies a measurable contribution from online feedback beyond a static global margin under the tested stochastic process.

TABLE II. Calibration and feedback ablations

Policy variant	Shortfall %	P95 deficit
ML mean	69.57	0.239
+ fixed margin	13.34	0.114
CPU-only adaptation	54.33	0.267
HAIL full	8.86	0.091

D. Importance of all four domains. CPU-only adaptation yields 54.33% protected shortfalls, much higher than full HAIL despite sharing its CPU predictor and calibrated margin. The service proxy is the minimum resource coverage across domains; stale memory, network, or storage quotas can therefore dominate adequate CPU provision. This ablation operationally tests the four-domain architecture and isolates the benefit of joint quota adjustment.

E. Tail of resource deficit. Mean episode-wise 95th-percentile protected deficit (1-s) is 0.0906 for HAIL, compared with 0.1501 for taxonomy-and-lending, 0.2760 for reactive, and 0.2390 for ML mean. Thus HAIL improves the hierarchical baseline by 0.0595 absolute (paired 95% interval -0.0745 to -0.0445). These quantities describe resource demand coverage rather than the 95th-percentile response time. The distinction remains important because latency can also depend on queuing and fan-out.

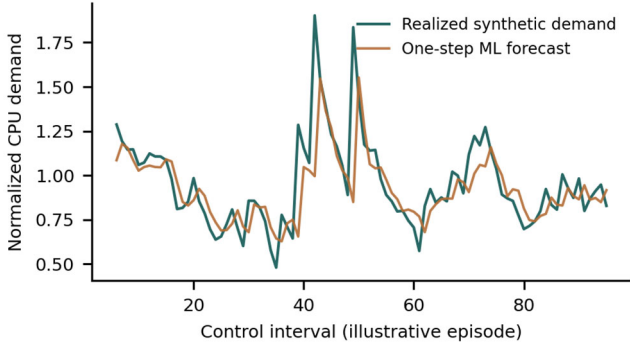


Fig. 4 Example synthetic CPU demand and one-step ML forecast for one protected tenant.

F. Prediction and calibration diagnostics. Figure 4 illustrates a one-step synthetic CPU forecast; sudden burst onsets remain difficult to predict from lagged observations. Resource-specific validation margins range from 0.1056 to 0.1312 capacity units. On the held-out protected observations, exceedance of the corresponding margin averages 9.55%, 9.81%, 9.69%, and 10.54% for CPU, memory, network, and storage. These are empirical diagnostics of the pooled quantile, not guaranteed joint four-domain coverage; the controller therefore relies additionally on feedback and priority projection.

G. Margin sensitivity. Re-running HAIL on identical held-out workloads at five multiples of the four validation quantiles yields protected shortfalls of 69.57%, 25.10%, 8.86%, 4.21%, and 2.90% for multipliers 0, 0.5, 1, 1.5, and 2. Corresponding mean batch demand coverage declines from 85.23% to 82.90%, 80.67%, 78.17%, and 75.43%. At twice the margin, utilization falls to 84.38%, compared with 87.02% without headroom. Figure 5 shows that prioritization involves a continuous protection-throughput trade-off rather than cost-free performance gains.

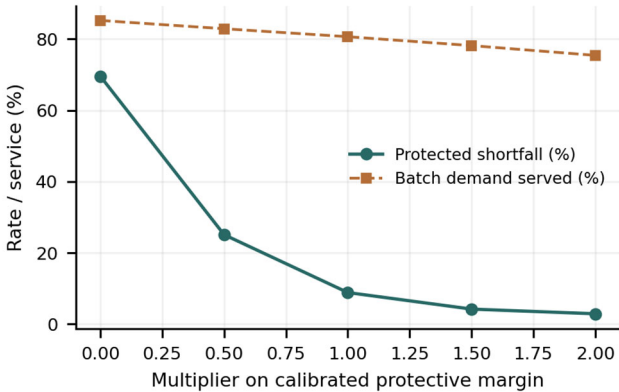


Fig. 5 Sweep of the error-margin multiplier, showing protection versus batch service.

H. Burst-shift sensitivity. We increase burst amplitude by 40% in twelve matched test episodes without refitting models or recalibrating margins. HAIL obtains 9.26% protected shortfalls and 87.32% utilization; taxonomy-and-lending obtains 23.07% and 87.98%. Batch coverage falls to 78.57% under HAIL versus 80.59% under the hierarchy. This supports a narrowly defined amplitude-shift comparison, not general robustness to workload drift or altered tenant classes.

I. Contention and starvation stress. We reduce all four capacity dimensions to 60% and 40% of their training setting, holding the generated demand, model inputs, and trained

predictors fixed; this is a counterfactual capacity-envelope test rather than a host simulation with capacity-dependent workloads. At 40% capacity, HAIL still reports fewer protected shortfalls (18.08%) than taxonomy-and-lending (30.64%), but mean batch coverage collapses to 1.93% versus 3.00%. At 60% capacity, protected shortfalls are 8.86% versus 24.11%, while batch coverage is 27.86% versus 30.03%. Figure 6 exposes a key limitation: strict priority without a minimum batch-share constraint can starve opportunistic tenants under severe scarcity.

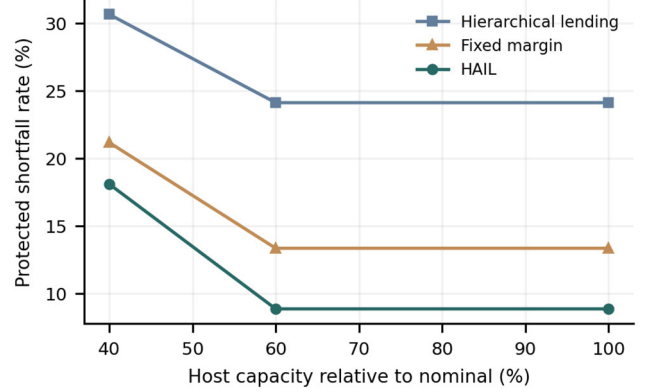


Fig. 6 Protected shortfall rates under reduced host capacities; batch coverage also falls sharply.

J. Comparison boundaries. The study evaluates prediction and quota-setting rather than virtual machine implementation. Firecracker [20] demonstrates practical isolation at the virtualization layer, and Twine [21] illustrates engineering requirements for cluster-wide management. Our simulator does not implement these mechanisms; it tests whether a risk-aware four-domain quota rule is promising enough to motivate subsequent host-level deployment.

7. Discussion

The experiments show a controllable operating-point shift rather than unconditional dominance. At nominal capacity, HAIL sacrifices 0.70 utilization points and 2.45 batch-service points relative to hierarchical lending to protect priority demand. Under 40%-capacity stress, batch coverage is only 1.93%, revealing that strict protected-first projection requires an explicit starvation or minimum-share constraint before production deployment. A future design could reserve a configurable batch floor while placing a tunable maximum on protected headroom. The calibration quantile (0.90) and online amplification bound (0.65) are explicit control parameters; a full multi-objective search remains future work. Because HAIL holds more headroom than the hierarchical baseline, an iso-utilization comparison obtained by sweeping the baseline's protected multiplier is a necessary complement to Table I. Neuro-Elastic provides a complementary empirical example of evaluating adaptive execution decisions against measured latency, energy, and hardware operating constraints [26].

Capacity reuse has precedents in protected harvesting systems, including SmartHarvest, which predicts spare CPU capacity [22]. Our policy uses a smaller, host-local four-domain simulation and explicitly restricts lending to remaining decision-time quota, so it avoids claiming preemption or immediate in-period reclamation. In a real host, memory placement such as TPP [23], contention within last-level cache, and network queuing will create isolation effects

not captured by independent quota axes. A practical controller should observe per-domain pressure and translate caps into mechanisms such as cgroups v2 and traffic shaping with a hardware-aware admission gate.

Service performance and security isolation are different properties. Microarchitectural execution attacks and their mitigations [24] lie outside the bottleneck coverage proxy, and no security guarantee follows from a low simulated shortfall rate. A production implementation would need policy isolation of tenants, noisy-neighbor attack testing, quota enforcement latency, and a clear distinction between reservations and actual usage. Model training and decision costs should also be measured under realistic host hardware, rather than inferred from this Python simulator.

An important deployment issue is distribution shift. The positive-error quantile is calibrated on a small number of synthetic validation episodes from the same generator family as nominal testing. A new tenant mix, burst duration, priority ratio or calendar periodicity may invalidate the margin. The burst-amplitude and capacity-scarcity tests probe two narrow changes, but neither substitutes for independent production workloads. Monitoring calibration drift, off-path retraining, and conservative fallbacks are necessary next steps.

8. Limitations and reproducibility

First, all demand trajectories are synthetic. The generator includes arbitrary normalized base profiles, correlated dynamics and bursts rather than distributions estimated from cloud telemetry. Second, the service metric is resource-coverage shortfall, not application latency, throughput, security isolation or an observable Kubernetes SLO. No kernel scheduler, storage queue, link congestion, NUMA or device interference is measured. The controller observes per-tenant demand retrospectively, which must be instrumented in any subsequent deployment. Latent generator profiles are excluded from model inputs and policy decisions.

Third, four-domain capacity feasibility does not model nonlinear contention in shared caches, memory buses, network links or storage devices. Fourth, twelve disjoint test seeds provide comparisons within one synthetic family but do not span realistic tenant mixes or hardware configurations. The design also has no guarantee of shortfall-free service even within the simulator.

Fifth, the counterfactual 40%-capacity experiment exposes near-starvation of best-effort demand because no explicit minimum batch share is enforced. In that experiment, workloads, model inputs and point forecasts are held fixed; only the capacity envelope of the quota projection changes. Finally, the archive contains all generated inputs, trained predictors, seeds, code, calibrated margins, and per-episode results so that every quantitative statement can be audited.

9. Conclusion

HAIL combines causal-history four-domain forecasts, validation-calibrated protective margins, bounded online surprise feedback and a capacity-feasible, priority-first lending projection. In twelve held-out synthetic episodes, protected demand shortfalls decrease from 24.11% under a profiling-and-lending controller to 8.86%; mean utilization decreases by 0.70 percentage points. Ablations confirm the relevance of error calibration and multi-domain decisions. Severe capacity stress additionally exposes starvation of best-effort tenants, making a minimum-service safeguard a

necessary extension before deployment. The main contribution is an executable, transparent resource-coverage controller, not a measurement of production request latency.

References

- [1] Dean, J., & Barroso, L. A. (2013). The tail at scale. *Communications of the ACM*, 56(2), 74-80. <https://doi.org/10.1145/2408776.2408794>
- [2] Wang, J., Ge, Y., Liu, J., & Jiang, B. (2024). Early warning of liquidity stress in U.S. banks with seasonally aligned frailty hazards and delayed-feedback conformal alarms. *Journal of Trends in Finance and Economics*, 1(1), 85-100.
- [3] Tirmazi, M., Barker, A., Deng, N., Haque, M. E., Qin, Z., Hand, S., Harchol-Balter, M., & Wilkes, J. (2020). Borg: The next generation. In *Proceedings of the 15th European Conference on Computer Systems (EuroSys)* (Article 30, pp. 1-14). ACM. <https://doi.org/10.1145/3342195.3387517>
- [4] Ding, J., Chen, T., & Qin, Y. (2026). Achieving resource isolation in multi-tenant cloud platforms without sacrificing performance. *Journal of Computing and Electronic Information Management*, 21(1), 10-18. <https://doi.org/10.54097/b5qrla18>
- [5] Ghodsi, A., Zaharia, M., Hindman, B., Konwinski, A., Shenker, S., & Stoica, I. (2011). Dominant resource fairness: Fair allocation of multiple resource types. In *Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (pp. 323-336). USENIX Association.
- [6] Grandl, R., Ananthanarayanan, G., Kandula, S., Rao, S., & Akella, A. (2014). Multi-resource packing for cluster schedulers. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (pp. 455-466). ACM. <https://doi.org/10.1145/2619239.2626334>
- [7] Zhou, F. (2026). Moment-constrained side-force allocation for wind-assisted merchant ships: Hydrodynamic performance of retractable anti-leeway fins and flap rudders. *Innovation and Technology Studies*, 3(1), 27-41.
- [8] Delimitrou, C., & Kozyrakis, C. (2014). Quasar: Resource-efficient and QoS-aware cluster management. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (pp. 127-144). ACM. <https://doi.org/10.1145/2541940.2541941>
- [9] Lo, D., Cheng, L., Govindaraju, R. K., Ranganathan, P., & Kozyrakis, C. (2015). Heracles: Improving resource efficiency at scale. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture (ISCA)* (pp. 450-462). ACM. <https://doi.org/10.1145/2749469.2749475>
- [10] Qiu, H., Banerjee, S. S., Jha, S., Kalbarczyk, Z. T., & Iyer, R. K. (2020). FIRM: An intelligent fine-grained resource management framework for SLO-oriented microservices. In *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (pp. 805-825). USENIX Association.
- [11] Shahradd, M., Fonseca, R., Goiri, Í., Chaudhry, G., Batum, P., Cooke, J., Laureano, E., Tresness, C., Russinovich, M., & Bianchini, R. (2020). Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In *Proceedings of the 2020 USENIX Annual Technical Conference (ATC)* (pp. 205-218). USENIX Association.
- [12] Hadary, O., Marshall, L., Menache, I., Pan, A., Greeff, E. E., Dion, D., Dorminey, S., Joshi, S., Chen, Y., Russinovich, M., & Moscibroda, T. (2020). Protean: VM allocation service at scale. In *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (pp. 845-861). USENIX Association.

- [13] Ambati, P., Irwin, D., Kaufmann, L., Lantz, P., Rai, A., & Zeryihun, A. (2020). Providing SLOs for resource-harvesting VMs in cloud platforms. In Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI) (pp. 735-751). USENIX Association.
- [14] Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z., & Alizadeh, M. (2019). Learning scheduling algorithms for data processing clusters. In Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM) (pp. 270-288). ACM. <https://doi.org/10.1145/3341302.3342080>
- [15] Yue, X., & Zhu, R. (2024). An explainable hybrid machine learning framework for cash-flow-conditioned multi-horizon liquidity risk early warning: An SME-oriented benchmark study. *World Journal of Management Science*, 2(1), 63-72.
- [16] Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., & Bianchini, R. (2017). Resource Central: Understanding and predicting workloads for improved resource management in large cloud platforms. In Proceedings of the 26th Symposium on Operating Systems Principles (SOSP) (pp. 153-167). ACM. <https://doi.org/10.1145/3132747.3132772>
- [17] Chen, J., Zhang, X., & Han, X. (2023). Explainable identification of tax-related financial reporting anomalies in U.S. public companies: Cross-fitted tax-consistency residuals and Owen-value Shapley propagation. *World Journal of Economics and Business Research*, 1(1), 33-45.
- [18] Romano, Y., Patterson, E., & Candès, E. J. (2019). Conformalized quantile regression. In *Advances in Neural Information Processing Systems 32 (NeurIPS)* (pp. 3543-3553).
- [19] Rahman, J., & Lama, P. (2019). Predicting the end-to-end tail latency of containerized microservices in the cloud. In Proceedings of the IEEE International Conference on Cloud Engineering (IC2E) (pp. 200-210). IEEE. <https://doi.org/10.1109/IC2E.2019.00034>
- [20] Agache, A., Brooker, M., Iordache, A., Liguori, A., Neugebauer, R., Piwonka, P., & Popa, D.-M. (2020). Firecracker: Lightweight virtualization for serverless applications. In Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI) (pp. 419-434). USENIX Association.
- [21] Chen, J., Han, X., Zhang, X., & Yang, Y. (2023). Exact concept-level explanations for early detection of financial misstatements. *Eurasia Journal of Science and Technology*, 5(2), 34-48.
- [22] Wang, Y., Ma, X., Li, Z., & Chen, H. (2021). SmartHarvest: Harvesting idle CPUs safely and efficiently in the cloud. In Proceedings of the 16th European Conference on Computer Systems (EuroSys) (pp. 1-16). ACM. <https://doi.org/10.1145/3447786.3456225>
- [23] Al Maruf, H., Chowdhury, M., & Mahajan, A. (2023). TPP: Transparent page placement for CXL-enabled tiered-memory. In Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) (Vol. 3, pp. 742-755). ACM. <https://doi.org/10.1145/3582016.3582063>
- [24] Canella, C., Van Bulck, J., Schwarz, M., Lipp, M., von Bergmann, B., Payer, M., & Gruss, D. (2019). A systematic evaluation of transient execution attacks and defenses. In Proceedings of the 28th USENIX Security Symposium (pp. 249-266). USENIX Association.
- [25] Wang, Y., Zou, X., Lu, Z., Wei, K., & Rhee, M. (2026). PRESAGE: Uncertainty-aware predictive elastic scheduling for cloud analytics and GenAI inference. *IEEE Access*, 14, 1-16. <https://doi.org/10.1109/ACCESS.2026.3734498>
- [26] Shang, W., Teng, D., Chen, T., Zhang, F., & Ding, J. (2026). Neuro-Elastic: A unified framework for hardware-aware adaptive quantization and dynamic sparsity in real-time edge intent prediction. *IEEE Access*, 14, 1-15. <https://doi.org/10.1109/ACCESS.2026.3715276>