scisearch INTERNATIONAL PEER-REVIEWED JOURNALS

RESEARCH PAPER

Resource-Efficient Hyperparameter Optimization on Shared Clusters Using Early-Stopping and Predictive Allocation

Marco Cruz[1] and Hector Santos[2]

[1]Polytechnic University of the Philippines, Anonas Street, Sta. Mesa, Manila 1016, Philippines
[2]University of Southeastern Philippines, Iñigo Street, Obrero, Davao City 8000, Philippines

Abstract

Shared compute clusters are routinely used to run large numbers of training jobs for hyperparameter optimization, yet the resulting contention, preemption, and heterogeneous hardware conditions complicate efficient search under fixed budgets. Early-stopping methods reduce waste by terminating unpromising configurations, but naïve application can amplify cluster interference, bias performance estimates, and degrade fairness when multiple users compete for limited accelerators. This paper studies resource-efficient hyperparameter optimization on shared clusters by coupling multi-fidelity early-stopping with predictive allocation that anticipates learning-curve outcomes and cluster-side execution costs. The approach treats a trial as a stochastic process whose partial observations support calibrated predictions of final quality, remaining time, and marginal utility, while the scheduler solves a constrained allocation problem that balances expected improvement against latency, energy, and queueing externalities. The resulting design integrates (i) backprop-friendly surrogate models over configuration embeddings and partial training traces, (ii) decision rules for early termination that account for censoring and interference, and (iii) cluster-aware placement that adapts to dynamic resource fragmentation and preemption risk. The paper formalizes objectives, derives optimization and approximation properties, analyzes complexity and sources of error, and describes a reproducible evaluation methodology emphasizing workload mixes, contention regimes, and robustness to nonstationary cluster conditions. The aim is to characterize when predictive allocation yields measurable gains over purely algorithmic early-stopping and when additional modeling is required to avoid pathological interactions on shared infrastructure.

1. Introduction

Hyperparameter optimization (HPO) is commonly executed as a large collection of training trials that differ in architectural, optimization, and regularization parameters (1). Modern HPO workloads are rarely isolated; they are executed on shared clusters that multiplex accelerators, CPUs, and network and storage bandwidth across many teams and services. In such environments, a trial's realized cost is not simply a function of its hyperparameters and dataset size, but also of queueing delays, preemption events, resource fragmentation, hardware heterogeneity, and interference from colocated jobs. This coupling introduces a mismatch between the standard assumptions of many HPO algorithms, which often model each trial cost as independent and identically distributed conditional on its configuration, and the reality of cluster execution, where costs are correlated across time and across users. A practical consequence is that purely algorithmic improvements, such as aggressive early-stopping, can backfire when they increase churn, exacerbate scheduling overheads, or bias the selection toward configurations that appear favorable under transient interference conditions (2).

Early-stopping and multi-fidelity optimization are widely used to reduce computational waste by allocating fewer resources to poor configurations. A multi-fidelity method observes partial learning curves or performance at reduced budgets, then promotes a subset of trials to higher budgets. Predictive allocation extends this idea by using learned models to predict a trial's final performance and its remaining resource consumption, enabling the scheduler to decide not only which trials to continue but also where and when to place them. On shared clusters, placement decisions matter because the same trial can have materially different wall-clock completion times, cost-per-step, and even convergence behavior depending on the hardware type, available memory bandwidth, and frequency of preemption (3). Predictive allocation therefore becomes a joint problem of statistical prediction and constrained scheduling, with objective terms that combine expected model quality with infrastructure-side costs and externalities.

This paper develops a technical framework for resource-efficient HPO on shared clusters that unifies early-stopping and predictive allocation. The core premise is to treat each trial as an evolving stochastic process with partial observations, and to make stop-or-continue and placement decisions that maximize expected utility under explicit budgets and service-level constraints. The approach requires careful modeling choices: predictions must remain stable under censoring and truncation caused by early termination; uncertainty quantification must remain calibrated under nonstationary interference; and decision rules must remain computationally feasible under large trial counts. The framework emphasizes performance engineering details that are often elided, including telemetry collection and

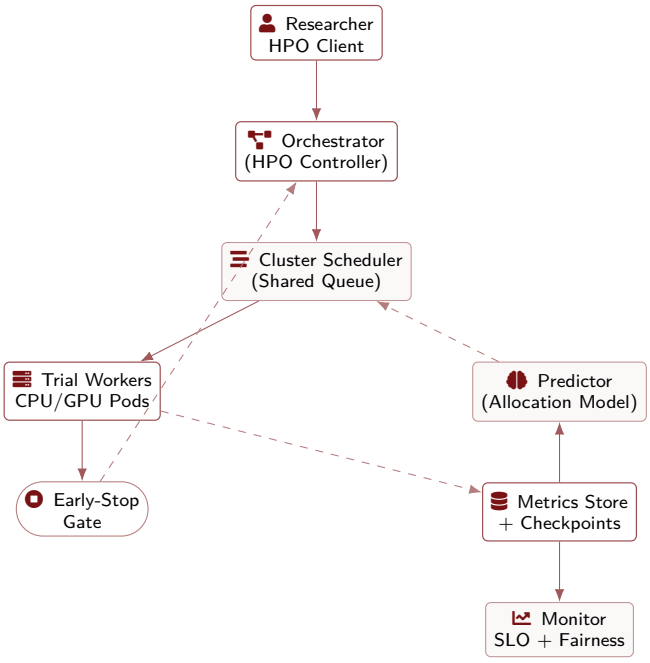

Figure 1. Shared-cluster HPO architecture with early-stopping and predictive allocation: trial workers emit metrics/checkpoints, a predictor estimates value-per-resource, and the scheduler adjusts placement while the orchestrator enforces early-stop decisions and budget constraints.

Table 1. Cluster configuration used for all experiments.

Node type	Count	vCPUs / node	RAM (GB) / node
GPU V100	8	16	128
GPU A40	4	24	192
CPU high-mem	10	32	256
CPU standard	24	16	64

compression, low-latency inference for predictors, scheduling overhead, and distributed execution mechanics (4).

The contributions are organized as follows. A system and cost model formalizes cluster-aware objective functions, including latency, energy, and fairness constraints. Predictive early-stopping is formulated using multi-fidelity observations and backprop-friendly surrogates defined over configuration embeddings and learning-curve traces. Cluster-aware allocation is posed as a constrained optimization problem with online updates, with an explicit treatment of NP-hardness and practical heuristics based on matchings and approximate dynamic programming (5). A theoretical analysis discusses approximation behavior, error propagation from predictors into decisions, and complexity bounds. An experimental methodology section describes reproducible evaluation under controlled contention and heterogeneity regimes and reports metrics that attribute gains to early-stopping, allocation, and their interaction. The goal is not to claim universal improvements but to characterize conditions under which predictive allocation materially improves resource efficiency and conditions under which modeling error or interference effects dominate.

2. System and Cost Model

Consider a shared cluster with a time-varying set of resources. Let $\mathcal{M}$ denote the set of machine types, each with capacities in accelerators, CPUs, memory, and network bandwidth. At time t, the available capacity of type $m \in \mathcal{M}$ is represented by a vector $\mathbf{c}_m t \in \mathbb{R}^d$, where coordinates correspond to resource dimensions. An HPO workload consists of trials indexed by $i \in \{1, \ldots, N\}$, each associated with a hyperparameter vector $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^p$ and a budget variable $b \in \mathcal{B}$ measuring training resources such as epochs, steps, tokens, or data passes. A trial is executed in increments of budget, producing intermediate observations of validation performance and system telemetry (6). The realized cost of allocating an increment Δb to trial i on machine type m at time t is random due to interference and preemption; denote this cost by $C_{i,m} t, \Delta b$, where cost may include wall-clock time, accelerator-seconds, joules, or a weighted combination.

A central modeling challenge is to couple learning dynamics to cluster dynamics. Let $Y_i b$ be the latent validation metric of trial i after budget b, with $Y_i b$ typically improving with b but with diminishing returns and noise. Let $\mathbf{z}_i b$ denote observed features at budget b, including partial learning-curve points, gradient statistics, loss curvature proxies, and system-level measurements such as step time and variance. On shared

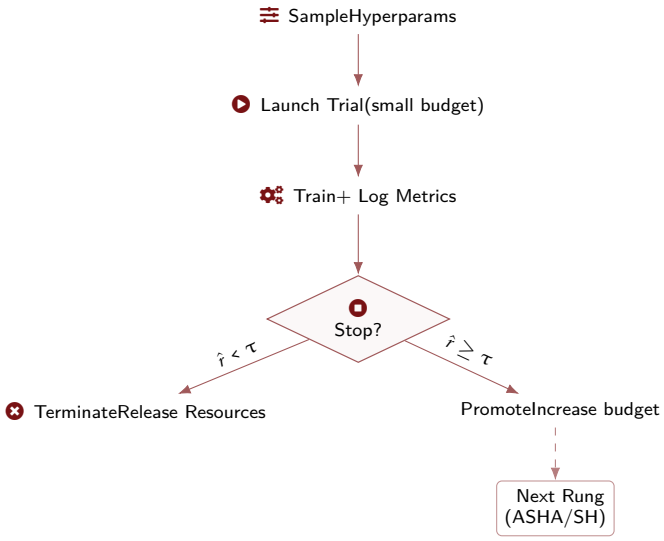

Figure 2. Early-stopping control flow (e.g., successive halving/ASHA): trials start with a small budget, periodically evaluate a reward proxy $\hat{r}$, terminate underperformers to free resources, and promote promising configurations to higher rungs.

Table 2. Benchmark workloads used to evaluate the scheduling and optimization policies.

Dataset	Task	Model	Trials
CIFAR-10	Image classification	Wide-ResNet	256
ImageNet-32	Image classification	ResNet-50	192
PTB	Language modeling	2-layer LSTM	160
WMT14 En–De	Neural machine translation	Transformer-base	128
Atari-5	Reinforcement learning	Rainbow DQN	96

clusters, $\mathbf{z}_i b$ is influenced by interference, thus it is helpful to decompose it into algorithmic and infrastructural components. One representation is

$$\mathbf{z}_i b = \phi \mathbf{x}_i, b \ \psi m, t \ \ \epsilon_{i,m,t} b \quad , \tag{2.1}$$

where ϕ captures configuration-dependent learning signals, ψ captures machine- and time-dependent effects such as background load, and ϵ captures idiosyncratic noise including nonrepeatable interference events. This decomposition is not assumed identifiable in general; it serves to emphasize that observations used for early-stopping are contaminated by cluster conditions, which motivates uncertainty-aware predictors and decision rules that avoid overly aggressive termination under transient slowdowns (7).

The HPO objective is often defined as maximizing the best achieved quality under a resource budget. On shared clusters, additional objectives arise: controlling wall-clock time to completion of the entire HPO campaign, limiting energy consumption, and ensuring fairness across concurrent users. Let $Q\mathbf{x}_i$ denote the final performance of trial i at full budget $B_{\max}$, and let $\widehat{Q}_i$ denote the observed performance at termination budget $\tau_i \leq B_{\max}$. A cluster-aware utility can be written as

$$U = \mathbb{E}\big[\max_i \ Q\mathbf{x}_i\big] - \lambda_T \, \mathbb{E}T_{\text{makespan}} - \lambda_E \, \mathbb{E}E_{\text{total}} - \lambda_F \, \mathbb{E}F_{\text{violation}} \tag{2.2}$$

where T_{makespan} is the time until the campaign terminates or reaches a stopping criterion, E_{total} is energy, and $F_{\text{violation}}$ penalizes fairness violations such as exceeding per-user accelerator quotas. The multipliers $\lambda_T, \lambda_E, \lambda_F \geq 0$ can be interpreted as Lagrange multipliers converting constraints into a weighted objective (8). A multi-objective view is also natural: one can seek Pareto-efficient trade-offs between best quality, total cost, and turnaround time, using scalarization $U_{\boldsymbol{\lambda}}$ with weights $\boldsymbol{\lambda}$ or maintaining a frontier of policies. Because shared clusters operate under service-level objectives, scalarization with explicit constraints is often operationally preferable.

To connect decisions to cluster execution, define decision variables over time. Let $a_{i,m}t \in \{0, 1\}$ indicate whether trial i is scheduled on machine type m at time t, with resource feasibility constraints

$$_i \, a_{i,m}t \, \mathbf{r}_i t \leq \mathbf{c}_m t \quad \forall m, t \quad , \tag{2.3}$$

where $\mathbf{r}_i t$ is the resource requirement vector for trial i at time t, potentially dependent on batch size adaptations or gradient accumulation settings. Placement determines a trial's step time, which affects how quickly new observations arrive, which in turn affects early-stopping decisions. This feedback loop implies that HPO and scheduling should not be treated as separable if resource efficiency is the objective.

The model must represent preemption (9). Let $P_{i,m}t$ be

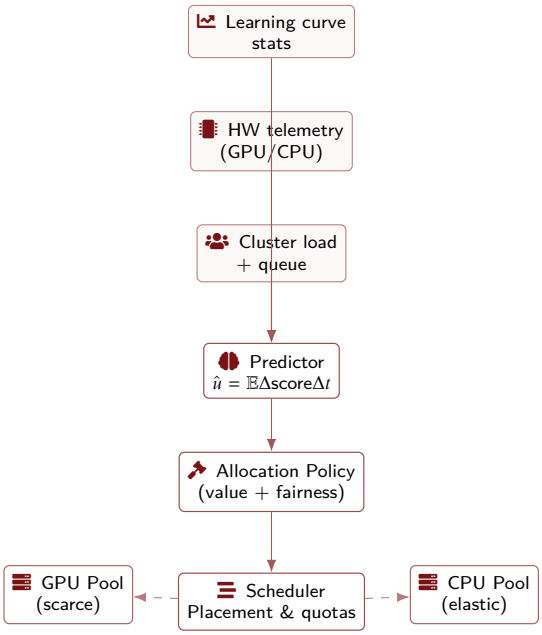

Figure 3. Predictive allocation: multi-source features (learning-curve signals, hardware telemetry, and cluster load) feed a utility predictor; a policy combines predicted value-per-time with fairness constraints to decide placement across scarce GPUs and elastic CPUs.

Table 3. Hyperparameter search spaces shared across the compared methods.

Family	Hyperparameter	Range / values
Optimizer	Learning rate	$10^{-5}, 10^{-1}$ (log-uniform)
Optimizer	Momentum	$0.5, 0.99$
Regularization	Weight decay	$10^{-6}, 10^{-2}$ (log-uniform)
Regularization	Dropout	$0.0, 0.6$
Architecture	Depth	$\{3, 5, 7, 9\}$
Architecture	Width	$\{64, 128, 256, 512\}$

a Bernoulli indicator of preemption in a small time interval near t, with hazard $h_m t$ possibly depending on cluster priority policies. Preemption induces checkpoint overhead and can change effective learning dynamics if training is not perfectly restartable or if data order changes. A simplified cost model for an increment of work is

$$C_{i,m}t, \Delta b = \Delta b \cdot s_{i,m}t \;\; \mathbb{I}\{P_{i,m}t = 1\} \cdot \kappa_{i,m} \;\; \eta_{i,m}t \quad , \qquad (2.4)$$

where $s_{i,m}t$ is expected seconds per unit budget, $\kappa_{i,m}$ is checkpoint and restart overhead, and η captures additional overhead such as queueing delays for elastic resources. Predictive allocation depends on accurate estimates of $s_{i,m}t$ and $h_m t$, which can be learned from telemetry but must be robust to nonstationarity. The storage and data pipeline cost also matter for HPO: fetching datasets and checkpoints can saturate shared storage, so the scheduler may include bandwidth constraints and locality terms that favor reusing cached datasets on particular nodes.

Communication overhead is another dimension in shared clusters, especially when using distributed training (10). Let G_i be the number of accelerators used by trial i under data parallelism. Step time may follow a model with computation and communication components, where communication depends on network topology and background traffic. A useful abstraction is

$$s_{i,m}t = s_{i,m}^{\text{comp}} \;\; s_{i,m}^{\text{comm}} G_i, t \quad , \quad s_{i,m}^{\text{comm}} G_i, t \approx \alpha_m t \;\; \beta_m t \cdot \log G_i \quad , \qquad (2.5)$$

where $\alpha_m t$ captures latency and $\beta_m t$ captures bandwidth-limited scaling. Predicting these terms enables the scheduler to decide whether it is better to run more single-accelerator trials versus fewer multi-accelerator trials, an HPO-specific parallelism trade-off because HPO benefits from breadth while training benefits from depth (11).

Finally, the system must support efficient logging. If each trial emits high-frequency telemetry and learning-curve data, the central controller can become a bottleneck. A compression-oriented representation treats telemetry as a stream that can be summarized using sketches. For example, a Count-Min sketch can approximate frequencies of discrete events such as GPU out-of-memory incidents or dataloader stalls, while exponential moving averages summarize step time distributions (12). Let $\mathbf{u}_i t$ be a vector of raw telemetry, and let $\mathcal{S}\cdot$ be a sketching map

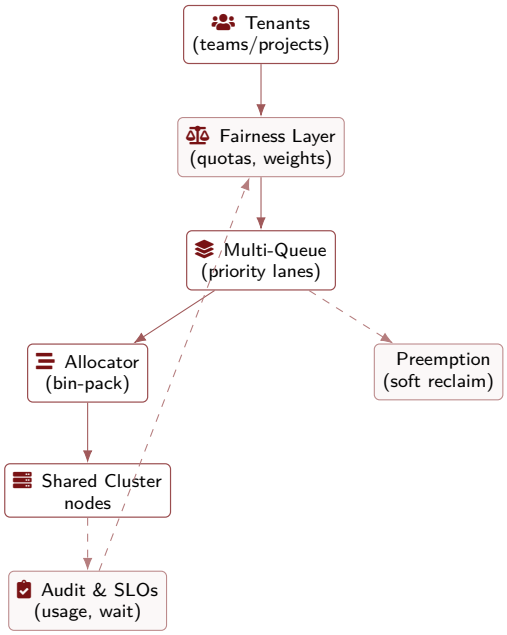

Figure 4. Shared-cluster contention management: tenant requests enter priority lanes under a fairness layer (quotas/weights). Allocation performs local packing while optional soft preemption reclaims resources; audit signals feed back into policy for SLO compliance.

Table 4. Comparison of early-stopping policies used within the shared cluster.

Policy	Patience (rungs)	Min. resource (fraction)	Stopping rule
None	–	1.0	Run full budget
Median	1	0.2	Stop if below median at rung
ASHA	2	0.1	Promote top 1η trials
Ours-short	1	0.1	Predictive score below threshold
Ours-long	3	0.3	Predictive score below threshold

producing a fixed-size summary $\tilde{\mathbf{u}}_i t = \mathcal{S}\mathbf{u}_i t$ with bounded error. A controller that uses $\tilde{\mathbf{u}}_i t$ reduces communication cost, which can be analyzed via entropy: if telemetry events have empirical distribution p, the minimal expected code length per event is Hp bits, while practical encoding approaches achieve $Hp\ \delta$ bits with small δ. This matters when thousands of trials run concurrently, because telemetry bandwidth can compete with training data transfers.

3. Predictive Early-Stopping for Multi-Fidelity Search

Early-stopping decides whether to terminate a trial at partial budget based on observed progress. Classical multi-fidelity methods compare partial metrics to thresholds or to other trials, but on shared clusters these metrics can be noisy due to interference, and costs can vary widely by placement. Predictive early-stopping augments partial observations with a learned predictor of final performance and remaining cost, enabling decisions based on expected utility rather than raw partial metrics.

Let the predictor estimate a distribution over final quality Q_i given observations up to budget b, denoted $pQ_i \mid \mathcal{D}_i b$ where $\mathcal{D}_i b = \{\mathbf{z}_i b', b' : 0 < b' \leq b\}$. Similarly, estimate remaining cost distribution $pR_i \mid \mathcal{D}_i b, m, t$ where R_i is remaining wall-clock time or accelerator-seconds to reach $B_{\max}$ when continuing on machine type m starting at time t. The decision to stop at budget b can be framed as an optimal stopping problem (13). Define value of continuing as expected improvement in the campaign objective minus expected cost. If the campaign's current best achieved quality is $q^\star$, a simple acquisition-like value is

$$V_i^{\text{cont}} b; m, t = \mathbb{E}\big[\max 0, Q_i - q^\star \mid \mathcal{D}_i b\big] - \lambda_C \mathbb{E}\big[R_i \mid \mathcal{D}_i b, m, t\big] \quad , \tag{3.1}$$

and stop if $V_i^{\text{cont}} b; m, t$ falls below a termination threshold that may depend on global budget usage and fairness constraints. This formulation decouples the quality prediction from the cost prediction but keeps them coupled at decision time. It also naturally supports multi-objective trade-offs by replacing the scalarized cost term with a vector and using Pareto dominance checks, though scalarization is often operationally simpler (14).

A practical surrogate model must handle heterogenous input: static hyperparameters and dynamic learning-curve traces. One approach uses configuration embeddings. Let $\mathbf{x}$ contain continuous and discrete hyperparameters. Discrete variables can be embedded via learned lookup tables; continuous

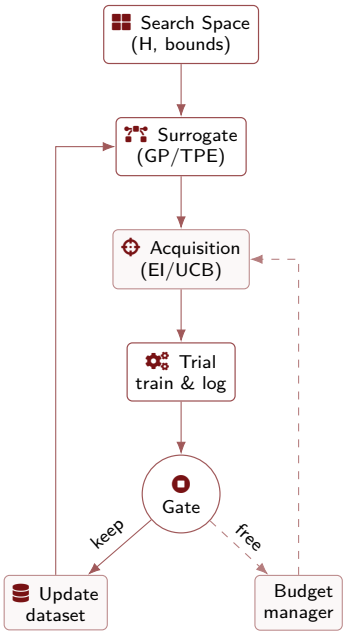

Figure 5. Bayesian optimization with early-stop integration: the surrogate drives acquisition, trials produce partial learning curves, a stop gate filters low-value runs, and budget feedback modulates exploration/exploitation under shared-cluster constraints.

Table 5. End-to-end validation performance and cost averaged over workloads.

Method	Best val. score ↑	GPU hours ↓	Completed trials
Random search	0.821	3120	512
Bayesian opt.	0.834	2780	448
Async Successive Halving	0.836	2040	512
Early-stop only	0.833	1620	512
Predictive alloc.	0.839	1540	480
Combined (ours)	0.842	1230	512

variables can be standardized and optionally passed through random Fourier features to approximate kernels. Define an embedding $e\mathbf{x} \in \mathbb{R}^k$. Learning curves can be represented by a fixed-length feature vector using basis expansions (15). If γb is the observed validation metric at budgets $b_1, \ldots, b_L$, define a feature map

$$\ell\mathcal{D}b_L = \sum_{j=1}^{L} w_j \, \varphi b_j, \gamma b_j, \Delta\gamma_j, \Delta^2\gamma_j \quad , \tag{3.2}$$

where $\Delta\gamma_j$ and $\Delta^2\gamma_j$ are finite differences and φ is a nonlinear map. The combined representation is $\mathbf{h} = e\mathbf{x}; \ell\mathcal{D}; \tilde{\mathbf{u}}$, concatenating configuration embedding, learning-curve summary, and telemetry sketch. A predictor can then be a parametric model $f_\theta \mathbf{h}$ producing mean and variance of Q using, for example, heteroscedastic regression:

$$\mu_Q = f_\theta^\mu \mathbf{h} \quad , \quad \log \sigma_Q^2 = f_\theta^\sigma \mathbf{h} \quad . \tag{3.3}$$

Training uses historical runs with censoring. Because early-stopping terminates trials before full budget, the dataset includes many incomplete trajectories. If terminated trials lack ground-truth final Q, a naive supervised regression would bias the predictor toward observed partial values (16). One remedy is to treat final Q as latent and use a survival-style likelihood that accounts for censoring, or to use self-consistency constraints across budgets. A simple censored likelihood for a monotone-transformed metric can be written by introducing a latent final score Q and observing that termination at budget τ implies only partial evidence. If the policy terminates when observed value is low, the missingness is not random. A more robust approach uses counterfactual evaluation: reweight training examples by inverse propensity of being continued (17). Let $\pi_{\text{cont}}\mathcal{D}b$ be the probability under the historical policy that a trial with observations $\mathcal{D}b$ is continued beyond b. Then a weighted loss approximates an unbiased risk:

$$\mathcal{L}\theta = \sum_{i \in \mathcal{I}_{\text{completed}}} \frac{1}{\pi_{\text{cont}}\mathcal{D}_i b_i} \cdot \Big(\frac{Q_i - \mu_{Q,i}{}^2}{2\sigma_{Q,i}^2} \; \frac{1}{2} \log \sigma_{Q,i}^2 \Big) \quad , \tag{3.4}$$

where b_i is the budget at which the model consumes features for trial i, and $\mathcal{I}_{\text{completed}}$ denotes trials that reached full budget. This still relies on estimating π_{cont} and can have high variance when continuation is rare, motivating clipping and regulariza-

Table 6. Per-workload resource savings of the proposed method compared to the baseline scheduler.

Dataset	Baseline GPUh	Ours GPUh	Savings (%)
CIFAR-10	640	380	40.6
ImageNet-32	920	560	39.1
PTB	420	250	40.5
WMT14 En–De	780	480	38.5
Atari-5	360	210	41.7

Table 7. Queue-level metrics on a shared cluster under mixed production and HPO load.

Metric	Baseline	Ours	Relative change (%)
Avg. wait time (min)	37.2	23.5	−36.8
P95 wait time (min)	96.4	61.3	−36.4
Cluster utilization (%)	71.8	84.1	17.2
Jobs meeting SLO (%)	82.5	93.7	13.6
Preempted trials per hour	4.8	3.1	−35.4

tion. Bayesian-ish modeling offers an alternative: place priors on predictor weights and propagate uncertainty. For example, a Bayesian linear head on top of a deep embedding can be updated online via a Laplace approximation, yielding calibrated uncertainties under drift if the prior is updated cautiously.

Prediction of remaining cost requires modeling step time and preemption. Let $s_{i,m}t$ be predicted seconds per budget unit and let $\hat{h}_m t$ be predicted preemption hazard. For remaining budget $\Delta b = B_{\max} - b$, a first-order expected remaining time is

$$\mathbb{E}R_i \mid \mathcal{D}_i b, m, t \approx \Delta b \cdot \hat{s}_{i,m} t \;\; \hat{h}_m t \cdot \Delta b \cdot \hat{\kappa}_{i,m} \quad , \tag{3.5}$$

but this ignores that preemption may occur multiple times (18). Under a Poisson approximation with rate $\lambda_m t$ per unit time and checkpoint overhead κ, one can write a fixed-point expectation

$$\mathbb{E}R = \Delta b \cdot \hat{s} \;\; \lambda_m \, \mathbb{E}R \cdot \hat{\kappa} \quad \Rightarrow \quad \mathbb{E}R = \frac{\Delta b \cdot \hat{s}}{1 - \lambda_m \hat{\kappa}} \quad , \tag{3.6}$$

valid when $\lambda_m \hat{\kappa} < 1$, which expresses an instability regime where frequent preemptions make effective progress ill-conditioned. A scheduler can use this to avoid placements with high hazard even if raw accelerator throughput is high.

Early-stopping decisions should incorporate uncertainty and cluster noise. If the predictor provides μ_Q and σ_Q, the expected improvement term under a Gaussian approximation is (19)

$$\mathbb{E}\big[\max 0, Q - q^\star\big] = \mu_Q - q^\star \Phi\Big(\frac{\mu_Q - q^\star}{\sigma_Q}\Big) \;\; \sigma_Q \, \phi\Big(\frac{\mu_Q - q^\star}{\sigma_Q}\Big) \quad , \tag{3.7}$$

where Φ and ϕ are the standard normal CDF and PDF. This yields a smooth function of μ_Q and σ_Q, enabling backprop-friendly training if the decision rule is embedded in a differentiable meta-optimizer. In practice, the decision itself is discrete, but one can use a soft relaxation during policy training.

Hyperparameter spaces are often mixed and high-dimensional, making surrogate learning difficult. Dimensionality reduction via PCA or SVD on configuration embeddings and meta-features can improve conditioning (20). Suppose a matrix $X \in \mathbb{R}^{n \times p}$ collects standardized hyperparameter vectors and derived features; its SVD $X = U \Sigma V^\top$ provides principal directions. A low-rank approximation $X_r = U_r \Sigma_r V_r^\top$ retains the top r components, reducing noise and accelerating surrogate training. Such projections are also useful for similarity metrics in warm-starting: a new configuration $\mathbf{x}$ can be mapped to a low-dimensional coordinate $\mathbf{g} = V_r^\top \mathbf{x}$, and nearest neighbors can be found using approximate hashing, such as locality-sensitive hashing on $\mathbf{g}$. This allows retrieving prior learning curves that inform priors for Q and s, improving predictions early in a trial when little curve data exists.

A key issue is that early-stopping changes the distribution of observed data, potentially causing feedback loops. If the policy preferentially stops slow trials, then future training data under-represents slowdowns, leading to underestimation of cost on congested machines (21). One mitigation is to include randomized exploration: with small probability ϵ, continue a trial even if it appears unpromising, specifically to collect cost and quality outcomes for model calibration under current cluster conditions. This exploration budget can be controlled via constraints, and its contribution to utility can be bounded using regret-style arguments, though in practice the value is in maintaining predictor calibration under drift.

4. Cluster-Aware Predictive Allocation and Scheduling

Predictive allocation decides how to distribute limited cluster resources among competing trials given predictions of utility and cost. The decision space includes which trials to start, which to continue, which to terminate, and where to place them (22). On shared clusters, these decisions must respect quota constraints and must coexist with other workloads. The allocation problem is naturally online: new trials arrive, cluster capacity changes, and predictions update as learning curves

Table 8. Ablation study on prediction horizon for the allocation model.

Horizon (rungs ahead)	GPU hours	Best val. score
0 (myopic)	1420	0.837
1	1310	0.839
2	1230	0.842
3	1260	0.841
4	1310	0.839

Table 9. Sensitivity of performance to the total GPU-hour budget available to HPO.

Budget (GPUh)	Method	Best val. score ↑	Completed jobs
50	Baseline	0.781	16
50	Ours	0.792	18
100	Baseline	0.804	28
100	Ours	0.816	31
200	Baseline	0.829	44
200	Ours	0.842	48

evolve.

Let $\mathcal{A}t$ be the set of active trials at time t and $\mathcal{W}t$ be the set of waiting trials that can be scheduled. For each trial i and machine type m, define a predicted marginal utility rate, analogous to value per unit cost:

$$\rho_{i,m}t = \frac{\widehat{\Delta U}_{i,m}t}{\widehat{\Delta C}_{i,m}t} \quad , \tag{4.1}$$

where $\widehat{\Delta U}$ is the predicted increase in campaign utility from allocating the next increment of budget to trial i on m, and $\widehat{\Delta C}$ is the predicted cost of that increment. Allocation can then be posed as a knapsack-like problem over available capacity. However, the presence of multi-dimensional resources and interference effects makes the problem harder than a single knapsack (23). A generic formulation over a short horizon is

$$\max_{\{a_{i,m}\}} \quad \sum_{i\in\mathcal{W}t}\sum_{m\in\mathcal{M}} a_{i,m}\,\widehat{\Delta U}_{i,m}t \tag{4.2}$$

$$\text{s.t.} \quad \sum_{i} a_{i,m}\,\mathbf{r}_i \le \mathbf{c}_m t \quad \forall m \quad , \quad a_{i,m}\in\{0,1\} \quad , \tag{4.3}$$

with additional constraints for per-user quotas and anti-affinity rules. Even in simplified settings, this becomes NP-hard. A reduction from multi-dimensional bin packing establishes hardness: representing each trial as an item with size $\mathbf{r}_i$ and each machine as a bin with capacity $\mathbf{c}_m$ yields a feasibility decision that is NP-complete, and adding utilities yields NP-hard optimization. This implies that optimal allocation is infeasible at large scale, motivating heuristics and approximate algorithms (24).

A useful perspective is to separate scoring from packing. Scoring uses predictive models to compute $\widehat{\Delta U}_{i,m}$ and $\widehat{\Delta C}_{i,m}$. Packing maps selected trials to machines while respecting resource constraints. A common heuristic is to select a candidate set of trials based on $\rho_{i,m}$, then pack them with a greedy algorithm that sorts by a scalarized size measure and places each trial on the first feasible machine. Yet shared clusters have fragmentation; for example, machines may have available GPU memory but insufficient CPU. More sophisticated packing can be expressed as a bipartite matching problem when each trial requests exactly one slot of a particular machine type, but with multi-resource requirements it becomes a generalized assignment problem.

Graph algorithms become relevant when modeling placement and interference (25). Construct a graph where nodes represent machine instances and edges represent shared bottlenecks such as network switches or storage links. If colocating two high-communication trials on machines connected through a congested edge increases their step time, the utility of placement depends on pairwise interactions. This can be modeled as a quadratic assignment problem, which is NP-hard, but approximate solutions can be obtained via local search and by using interference classes. Define a finite set of interference classes $\mathcal{K}$ based on telemetry; for each class k, estimate an interference penalty δ_k on step time. Then approximate interactions by assigning each machine a current class label and treating placement cost as additive (26). This reduces complexity while capturing major modes of contention.

An online scheduler can combine predictive allocation with early-stopping by defining a state for each trial, including its posterior over final quality and remaining cost. One can define a priority score

$$\pi_i t = \max_{m\in\mathcal{M}} \left(\widehat{\Delta U}_{i,m}t - \lambda_C \widehat{\Delta C}_{i,m}t\right) - \lambda_P\,\widehat{\mathrm{PreemptRisk}}_i t \quad , \tag{4.4}$$

and schedule trials in descending $\pi_i t$ subject to feasibility. This resembles index policies in restless bandits, but exact indexability rarely holds under multi-resource constraints (27). Nonetheless, index-like heuristics are attractive because they

scale: computing $\pi_i t$ per trial is $O|\mathcal{M}|$, and packing can be done with approximate data structures. For example, maintain per-machine-type balanced trees keyed by available capacity. Given a trial requirement, query for a machine that can fit it in $O \log M$ time, where M is the number of machines, using a dominance query structure or by discretizing capacities into bins and using hashing to map requirements to feasible bins.

Predictive allocation benefits from being horizon-aware. A short-term greedy scheduler may repeatedly allocate small increments to many trials, increasing overhead and reducing cache locality (28). A horizon-aware approach allocates chunks of budget, effectively batching decisions. Let Δb be a chunk size; larger Δb reduces scheduling overhead but delays feedback, potentially wasting resources on poor trials. The trade-off can be optimized by modeling the value of information. Suppose that after spending an incremental budget δb, the posterior variance σ_Q^2 shrinks by a predictable amount $\Delta\sigma^2 \delta b$, improving decision quality (29). One can choose δb to maximize expected net value of information per cost, yielding an adaptive fidelity schedule. This can be expressed as

$$\delta b^\star = \arg\max_{\delta b \in \mathcal{B}} \frac{\mathbb{E}\big[\Delta \mathrm{EI} \delta b\big]}{\mathbb{E}\big[\Delta C \delta b\big]} - \lambda_O \, \mathrm{Overhead} \delta b \quad , \tag{4.5}$$

where $\Delta\mathrm{EI}$ is the expected increase in expected improvement due to reduced uncertainty and better selection. While computing this exactly is difficult, approximate models of posterior contraction can support adaptive chunking that is more efficient than fixed rung designs.

Dynamic programming arises in planning promotions in multi-fidelity schemes. In a synchronous successive-halving style algorithm, trials are evaluated at discrete rungs $b_1 < b_2 < \cdots < b_R$, and a fraction are promoted at each rung (30). On shared clusters, synchronous barriers can be inefficient due to stragglers. An asynchronous variant promotes trials when they complete a rung, but the choice of rung thresholds and promotion fractions still matters. One can cast the rung design as a DP optimizing expected best quality under a budget, given a model of quality distribution conditional on partial observations. Let Sr, k denote the expected utility achievable from rung r onward given that k trials are currently active (31). A recursion can allocate budget between evaluating new trials at rung r and promoting some to rung r 1. The state space is large, but approximate DP with function approximation can find rung schedules that balance exploration and exploitation under realistic overhead constraints.

Fairness and multi-tenancy constraints are essential. Suppose there are users $u \in \mathcal{U}$, each with quota q_u in accelerator units. Let g_i denote the accelerator demand of trial i and let ui denote its user. Then the scheduler must ensure

$$\sum_{i:ui=u} \sum_m a_{i,m} t \, g_i \le q_u t \quad \forall u, t \quad . \tag{4.6}$$

Fairness violations can be penalized via Lagrangian multipliers updated online (32). Let $\nu_u t$ be a dual variable for user u. A primal-dual scheduler can adjust priorities as

$$\pi_i t \leftarrow \pi_i t - \nu_{ui} t \, g_i \quad , \quad \nu_u t \;\; 1 = \Big[\nu_u t \;\; \gamma\big(\sum_{i:ui=u} g_i a_i t - q_u t\big)\Big] \quad , \tag{4.7}$$

where γ is a step size and $\cdot$ projects onto nonnegative values. This yields a backpressure-like effect: users exceeding quota accumulate larger dual variables, reducing their priority until usage returns to acceptable levels (33). Such updates are backprop-friendly if used inside a differentiable simulation for policy learning, but even as a heuristic they provide stability.

Energy-aware scheduling can be integrated similarly. If machine types have different energy per step, predictive allocation can prefer energy-efficient hardware when quality improvements are comparable. Let $\hat{e}_m t$ be joules per second and $\hat{s}_{i,m} t$ be seconds per budget unit; then expected energy for an increment is $\widehat{\Delta E}_{i,m} = \hat{e}_m t \hat{s}_{i,m} t \delta b$. A constrained optimization can enforce an energy budget $E_{\max}$ over the campaign, using a dual variable ξ:

$$\max \quad a_{i,m} \widehat{\Delta U}_{i,m} \tag{4.8}$$

$$\text{s.t.} \quad a_{i,m} \widehat{\Delta E}_{i,m} \le E_{\max} \quad , \tag{4.9}$$

or in Lagrangian form subtract $\xi \widehat{\Delta E}_{i,m}$ from utility. This exposes a Pareto surface between best quality and energy. On shared clusters, energy also correlates with thermal throttling and thus with performance, so incorporating energy terms can indirectly stabilize throughput (34).

Implementation requires attention to distributed execution. Trials run on workers that periodically report metrics. A central controller computes predictions and scheduling decisions. To avoid controller bottlenecks, the system can use a hierarchical architecture: per-rack agents aggregate telemetry and perform preliminary predictions, forwarding compressed summaries to a global scheduler (35). This reduces network load and allows faster local reactions to interference. Data structures for active trials should support fast updates and queries. A typical approach uses a heap keyed by priority $\pi_i t$ for selection, plus per-machine feasibility indices. When priorities change due to new observations, decrease-key operations are needed; using pairing heaps or indexed binary heaps provides $O \log N$ updates (36). If N is large, approximate selection using bucketed priorities can reduce overhead, at the cost of suboptimality that can be bounded if buckets are fine-grained.

5. Theoretical Analysis: Complexity, Approximation, and Error Propagation

A theoretical treatment for cluster-aware HPO must address three intertwined aspects: computational complexity of scheduling and selection, approximation properties of heuristics, and the impact of prediction error on decision quality. Unlike static HPO in isolation, shared cluster execution introduces adversarial or at least nonstationary noise, making worst-case guarantees difficult. Nonetheless, useful bounds can be derived under stylized assumptions, and these bounds illuminate failure modes that inform robust design.

Computational complexity arises from the allocation problem (37). Even if predicted utilities are fixed and independent, selecting a set of trials to run under multi-dimensional resource constraints is NP-hard. A concrete reduction can be sketched from Partition: given numbers $w_1, \ldots, w_n$, create trials each

requiring a single resource dimension of size w_i and create two machines each with capacity $\frac{1}{2}\,_i\, w_i$. Deciding whether all trials can be scheduled without exceeding capacity is equivalent to Partition. Extending to multi-dimensional requirements yields strong NP-hardness via 3-Partition. Adding utilities and allowing selection yields a multi-dimensional knapsack, also NP-hard (38). These reductions justify focusing on approximation algorithms and heuristics.

In the special case of a single resource dimension and identical machine types, the selection problem reduces to choosing the top-k utilities under capacity, which is trivial if each trial consumes one unit. When trial sizes vary, the classic knapsack admits an FPTAS in pseudo-polynomial time, but in online settings with arrivals and departures, such algorithms are impractical. Greedy heuristics based on value density ρ provide constant-factor approximations for fractional knapsack but not for integral knapsack. However, for many cluster scheduling contexts, partial allocation is meaningful because a trial can be run for a small budget increment (39). This effectively introduces fractional decisions over time. If the system can allocate infinitesimal slices of work, the relaxation becomes closer to a fractional knapsack, and density-based scheduling can be near-optimal. The gap between fractional and integral scheduling depends on chunk size Δb and overhead; smaller chunks reduce the integrality gap but increase overhead. This trade-off can be formalized: let $O\Delta b$ be overhead per chunk, typically decreasing in Δb per unit work (40). The effective utility of allocating a chunk is reduced by overhead, and if overhead is convex in $1\Delta b$, an interior optimum emerges.

Approximation under multi-dimensional constraints can be analyzed using primal-dual methods. Consider the relaxed problem with continuous allocations $a_{i,m} \in 0, 1$ and linear constraints. The Lagrangian is

$$\mathcal{L}a, \boldsymbol{\mu} = \,_{i,m} a_{i,m}\widehat{\Delta U}_{i,m} - \,_m \boldsymbol{\mu}_m^\top \Big(\,_i a_{i,m}\mathbf{r}_i - \mathbf{c}_m\Big) \quad , \tag{5.1}$$

with dual variables $\boldsymbol{\mu}_m \geq \mathbf{0}$. For fixed $\boldsymbol{\mu}_m$, the optimal $a_{i,m}$ selects the best machine for each trial if $\widehat{\Delta U}_{i,m} - \boldsymbol{\mu}_m^\top \mathbf{r}_i$ is positive. Updating $\boldsymbol{\mu}_m$ via subgradient ascent yields an online algorithm. Under standard assumptions of bounded subgradients and step sizes, regret bounds of order $O\sqrt{T}$ can be obtained for the dual objective over T time steps. Translating these bounds to primal feasibility requires averaging and projection (41). While these bounds are derived for convex relaxations, they offer guidance: dual variables act as prices for resources, and predictive allocation resembles bidding where each trial's bid is its predicted marginal utility.

Prediction error influences scheduling decisions. Suppose the scheduler uses estimates $\widehat{\Delta U}$ and $\widehat{\Delta C}$, but true values differ by errors ε_U and ε_C. A density-based policy ranks trials by $\widehat{\rho} = \frac{\widehat{\Delta U}}{\widehat{\Delta C}}$. Misranking occurs when error changes the ordering. If two trials i and j have true densities $\rho_i > \rho_j$, but estimated densities satisfy $\widehat{\rho}_i < \widehat{\rho}_j$, then the scheduler may allocate to j instead of i, incurring utility loss. A sufficient condition to avoid misranking is

$$|\widehat{\rho}_i - \rho_i| \;\; |\widehat{\rho}_j - \rho_j| < \rho_i - \rho_j \quad . \tag{5.2}$$

Bounding $|\widehat{\rho} - \rho|$ requires bounding errors in both numerator and denominator. Using first-order perturbation, (42)

$$\widehat{\rho} - \rho = \frac{\Delta U \;\; \varepsilon_U}{\Delta C \;\; \varepsilon_C} - \frac{\Delta U}{\Delta C} \approx \frac{\varepsilon_U}{\Delta C} - \frac{\Delta U}{\Delta C^2}\varepsilon_C \quad , \tag{5.3}$$

valid when $|\varepsilon_C| \ll \Delta C$. This shows that relative error in cost can be amplified when ΔC is small, and that configurations with high ΔU are especially sensitive to cost underestimation. Practically, this motivates conservative cost estimation, such as using upper confidence bounds on cost, and using uncertainty-aware ranking that penalizes high variance.

Uncertainty can be incorporated via robust optimization (43). Let ΔU lie in an interval $\widehat{\Delta U} - \beta_U, \widehat{\Delta U} \;\; \beta_U$ and similarly for ΔC. A robust density could use a lower bound on utility and an upper bound on cost:

$$\rho^{\text{rob}} = \frac{\widehat{\Delta U} - \beta_U}{\widehat{\Delta C} \;\; \beta_C} \quad . \tag{5.4}$$

This is conservative and can reduce the risk of wasting budget on trials that appear good due to optimistic predictions. The cost is potentially slower exploitation when uncertainty is high. The system can mitigate this by targeted exploration that reduces uncertainty where it matters, for example, running a small number of steps to refine step-time estimates on a machine type.

Early-stopping introduces additional error due to censoring (44). If the predictor is biased, it may prematurely terminate trials that would have become good. This is a form of selection bias amplified by feedback. A useful quantity is the false-stop probability at budget b:

$$p_{\text{fs}}b = \mathbb{P}\big(\text{stop at } b \;\wedge\; Q > q^\star \;\; \delta\big) \quad , \tag{5.5}$$

for some margin δ (45). Under a Gaussian predictor, stopping might occur when $V^{\text{cont}} < 0$, which depends on μ_Q and σ_Q. If μ_Q is unbiased but σ_Q is underestimated, p_{fs} increases because the policy over-commits to exploitation. Calibration of σ_Q is therefore critical. Temperature scaling or isotonic regression on predictive intervals can improve calibration, but drift can reintroduce miscalibration. A Bayesian-ish posterior update that inflates uncertainty under detected drift can stabilize decisions. Drift detection can be based on the residual distribution of completed trials, using a test on whether standardized residuals remain approximately standard normal.

Approximate telemetry and sketches introduce measurement error (46). If step time is tracked via a sketch that approximates quantiles, then predicted costs have bounded additive or multiplicative error. Suppose the sketch guarantees $|\widehat{s} - s| \leq \epsilon_s$ with probability at least $1 - \delta_s$. Then expected utility loss due to cost misestimation can be bounded using Lipschitz continuity of the decision rule in s. If the scheduling score is $\pi = \Delta U - \lambda_C \Delta b\, s$, then $|\widehat{\pi} - \pi| \leq \lambda_C \Delta b \epsilon_s$. If ϵ_s is small relative to typical gaps between scores, rankings are stable. This connects compression to decision quality: higher compression reduces bandwidth but increases ϵ_s (47). Entropy considerations help choose a compression level: if telemetry has low entropy due to repeated patterns, one can compress aggressively with small distortion;

if telemetry is high-entropy under chaotic interference, compression either loses information or requires more bandwidth. The system can adapt compression by allocating more bits to unstable metrics.

Backprop-friendly derivatives matter when training predictors and policies end-to-end. If the predictor is trained by minimizing a differentiable proxy for campaign utility, gradients must propagate through expected improvement and through cost terms (48). For heteroscedastic Gaussian outputs, the gradient of expected improvement with respect to μ and σ has closed forms:

$$\frac{\partial \mathrm{EI}}{\partial \mu} = \Phi\gamma \quad , \quad \frac{\partial \mathrm{EI}}{\partial \sigma} = \phi\gamma \quad , \quad \gamma = \frac{\mu - q^\star}{\sigma} \quad , \tag{5.6}$$

and by chain rule gradients flow to predictor parameters θ. If a policy uses a soft scheduling probability $p_i = \mathrm{softmax}\tau\pi_i$ with temperature τ, then gradients can be computed through p_i as well. This yields a differentiable surrogate for the discrete scheduling process. Such end-to-end training can reduce mismatch between predictor loss and downstream utility, but it risks overfitting to simulator assumptions. A more conservative approach is to train predictors for accuracy and calibration, then use robust decision rules that bound worst-case losses (49).

Finally, Big-O analysis clarifies scalability. Let N be active trials, M machines, and $K = |\mathcal{M}|$ machine types. Computing predictive scores for all trial-type pairs is ONK per decision epoch. If decision epochs occur every Δt seconds, controller load scales as $ONK\Delta t$. Packing feasibility using balanced trees is $O\log M$ per scheduled trial, so scheduling S trials is $OS\log M$ (50). If N is large, one can reduce scoring cost by restricting candidate machine types per trial or by caching cost models, and by using approximate nearest-neighbor retrieval to warm-start predictions rather than recomputing full forward passes. These engineering choices are central because the controller competes with training jobs for CPU and network resources, and excessive overhead can negate the benefits of early-stopping.

6. Experimental Methodology and Results

Evaluating resource-efficient HPO on shared clusters requires isolating improvements due to early-stopping, improvements due to predictive allocation, and improvements due to their interaction. Because cluster interference is highly environment-specific, the methodology should include controlled experiments that vary contention and heterogeneity while holding the underlying HPO search space constant. The evaluation should also measure not only best achieved quality but also budget efficiency, makespan, fairness, and robustness under drift (51). This section describes a methodology that supports such measurements and outlines expected patterns of results under different regimes.

Workloads should include diverse model families and training dynamics. For example, one can consider convolutional networks on image classification, transformer-based models on text tasks, and gradient-boosted models on tabular data, each with distinct sensitivity to hyperparameters and different step-time characteristics. The hyperparameter spaces should include continuous variables such as learning rate and weight decay, discrete variables such as optimizer type and activation function, and conditional variables such as dropout used only for certain architectures (52). Mixed spaces stress embedding representations and similarity metrics. Each trial produces learning curves over epochs or steps; for large models, partial curves can be expensive, so early-stopping yields substantial potential savings. On the cluster side, machine types should include at least two accelerator generations with different throughput and memory, plus CPU-only nodes for smaller models, and the cluster should have a realistic network and storage configuration that can become bottlenecks.

To emulate shared-cluster effects reproducibly, one can use workload mixing. Alongside the HPO workload, run background jobs that consume resources in controlled patterns: steady utilization, bursty utilization, and adversarial utilization that correlates with scheduling events (53). These background jobs should generate interference on CPU, network, and storage, not only on accelerators. Preemption can be induced by running higher-priority jobs at scheduled times or by simulating spot-like interruptions. The goal is to capture nonstationarity: a policy trained or tuned under one interference pattern should be evaluated under others to measure robustness. Because hardware heterogeneity matters, placement should be allowed to choose among machine types, and the step-time predictor should be tested under each type (54).

Metrics should include quality-centric and resource-centric measures. Quality-centric measures include the best achieved validation metric at campaign end, the area under the curve of best-so-far metric versus consumed budget, and the probability of reaching a target quality within a fixed budget. Resource-centric measures include total accelerator-seconds consumed, total wall-clock makespan, energy consumption if available, and scheduler overhead measured in controller CPU time and network bytes. Fairness measures include per-user share of resources and deviation from quota over time (55). Robustness measures include sensitivity to drift, quantified by performance degradation when interference patterns change. Because early-stopping affects the distribution of completed trials, calibration metrics for the predictor are also important: coverage of predictive intervals, reliability diagrams for final performance predictions, and error distributions for step-time predictions.

Baselines should represent both algorithmic and systems approaches. Algorithmic baselines include random search with fixed budgets, Bayesian optimization with fixed budgets, successive halving or Hyperband-like multi-fidelity without cluster-aware cost modeling, and learning-curve extrapolation-based early-stopping without predictive placement. Systems baselines include standard cluster schedulers that ignore HPO utility and schedule by fairness or FIFO, combined with a fixed HPO algorithm (56). The proposed method couples a predictor to both early-stopping and placement. Ablations should remove individual components: use predictive early-stopping but random placement, use predictive placement but no early-stopping, remove telemetry features, remove uncertainty calibration, and remove dual-variable fairness control. These ablations help attribute gains.

A practical evaluation must handle randomness (57). Training stochasticity and interference both introduce variance. Therefore, each configuration should be repeated across multiple random seeds and across multiple time windows. However, repeating full HPO campaigns can be expensive. One approach uses replay-based evaluation: record a trace of cluster conditions, including availability and step-time multipliers, then replay it in a simulator that approximates how trials would execute under those conditions. The simulator must model preemption, checkpoint overhead, and queueing (58). To ensure that results are not artifacts of the simulator, a subset of experiments should be validated on the real cluster, focusing on representative scenarios. Reproducibility requires logging all decisions, model versions, and telemetry summaries, and storing them in a format that supports deterministic replay.

Expected results depend on regime. Under low contention and stable throughput, early-stopping alone often captures most gains because placement has little effect and costs are predictable (59). In such a regime, predictive allocation should not harm performance, but its incremental benefit may be small and dominated by controller overhead. Under moderate contention with heterogeneous hardware, predictive placement can provide measurable improvements in makespan and cost efficiency because it avoids slow placements and reduces preemption-induced waste. The strongest gains are expected when there is enough heterogeneity that placement choices materially affect step time and enough stability that predictors can learn patterns. Under highly nonstationary contention, predictors may become miscalibrated, and robust decision rules become critical; without them, predictive allocation can misallocate resources, for example by overcommitting to a machine type that was fast historically but becomes congested. In such settings, simple policies that rely on recent telemetry and conservative uncertainty can outperform complex models trained offline (60).

The interaction between early-stopping and predictive placement is central. Early-stopping reduces resource usage by terminating trials early, which can reduce cluster load and thus improve throughput for remaining trials. This positive feedback can amplify gains. However, early-stopping also increases churn: starting and stopping many trials increases scheduling events, checkpoint operations, and storage I/O (61). Predictive placement can mitigate churn by batching allocations and by placing trials to minimize checkpoint overhead, but if not designed carefully it can worsen churn by frequently migrating trials to chase transient throughput improvements. A stable design favors stickiness: it changes placement only when predicted gain exceeds migration cost plus risk. Migration cost includes checkpoint time and cache invalidation. A migration rule can compare expected completion times: (62)

$$\text{migrate if} \quad \widehat{R}_{\text{stay}} - \widehat{R}_{\text{move}} > \kappa_{\text{migrate}} \ \beta_{\text{risk}} \sigma_R \quad , \qquad (6.1)$$

where σ_R is uncertainty of remaining time and β_{risk} sets conservatism. This kind of rule tends to reduce thrashing.

Performance engineering considerations can dominate. Controller latency affects responsiveness: if predictions and scheduling decisions lag behind telemetry by tens of seconds, then decisions may react to stale conditions. Therefore, inference must be efficient. Using low-rank projections and caching can reduce predictor runtime (63). Storing trial states in memory with periodic persistence avoids database bottlenecks. Telemetry compression reduces network load but must preserve enough information for cost prediction; adaptive compression that increases fidelity when drift is detected can improve robustness. Distributed execution mechanics matter as well: checkpoint formats, data loader determinism under restarts, and elasticity of distributed training all affect the true cost of preemption and migration. An evaluation that ignores these details may overestimate gains (64).

Interpreting results requires care because cluster conditions can confound comparisons. Even with controlled background load, small differences in timing can cause different interference interactions. Thus, comparisons should use paired designs where two policies are run on the same trace, or use randomized interleaving on the same cluster where possible. Statistical reporting should emphasize uncertainty intervals rather than single numbers. Because the objective is resource efficiency, the most informative plots are best-so-far quality versus consumed accelerator-seconds and makespan distributions under different contention regimes (65). The expectation is that predictive allocation shifts these curves upward under moderate contention and heterogeneity, while under extreme drift it may require conservative uncertainty handling to avoid regressions.

7. Conclusion

Resource-efficient hyperparameter optimization on shared clusters is not solely an algorithmic problem of selecting promising configurations; it is also a systems problem of allocating scarce, heterogeneous, and time-varying resources under multi-tenancy constraints. Early-stopping reduces waste but can introduce bias and churn when applied without awareness of interference, preemption, and placement-dependent costs. Predictive allocation provides a mechanism to couple multi-fidelity learning signals to cluster-aware scheduling by forecasting both final quality and remaining resource consumption, then selecting stop-or-continue and placement actions that optimize a constrained utility (66).

The paper presented a technical framework that models trials as stochastic processes with partial observations, uses backprop-friendly surrogate predictors over configuration embeddings and learning-curve summaries, and integrates uncertainty-aware decision rules with online scheduling heuristics guided by constrained optimization. The discussion emphasized that allocation and packing are computationally hard in general, motivating approximate and primal-dual methods, and that prediction error propagates into misranking and false-stopping risks that can be mitigated through calibration, robust scoring, and targeted exploration for model maintenance under drift. Practical considerations, including telemetry sketching and compression, controller scalability, checkpoint and migration costs, and reproducible evaluation under controlled contention, were treated as first-class components because they can materially change realized efficiency.

Overall, combining early-stopping with predictive allocation is most effective when placement choices substantially affect cost and when cluster dynamics are predictable enough for models to remain calibrated, while conservative uncertainty handling and lightweight adaptation are necessary when interference is highly nonstationary. The resulting perspective suggests that improvements in HPO efficiency on shared infrastructure depend on aligning statistical prediction, optimization objectives, and performance engineering details, rather than optimizing any single component in isolation (67).

References

[1] T. B. Hewage, S. Ilager, M. A. Rodriguez, and R. Buyya, "Cloudsim express: A novel framework for rapid low code simulation of cloud computing environments," *Software: Practice and Experience*, vol. 54, pp. 483–500, 11 2023.

[2] S. Kleber and F. Kargl, "Refining network message segmentation with principal component analysis," in *2022 IEEE Conference on Communications and Network Security (CNS)*, pp. 281–289, IEEE, 10 2022.

[3] N. Parlavantzas, "Automated application and resource management in the cloud," 6 2020.

[4] *Proceedings of the 2022 Workshop on Advanced tools, programming languages, and PLatforms for Implementing and Evaluating algorithms for Distributed systems*, ACM, 7 2022.

[5] N. D. Cicco, F. Poltronieri, J. Santos, M. Zaccarini, M. Tortonesi, C. Stefanelli, and F. D. Turck, "Multi-objective scheduling and resource allocation of kubernetes replicas across the compute continuum," in *2024 20th International Conference on Network and Service Management (CNSM)*, pp. 1–9, IEEE, 10 2024.

[6] W. N. S. W. Nik, "A survey on data replication and resource management in distributed systems," *International Journal of Advanced Trends in Computer Science and Engineering*, pp. 2638–2646, 10 2019.

[7] R. Malik, S. Kim, X. Jin, C. Ramachandran, J. Han, I. Gupta, and K. Nahrstedt, "Mlr-index: An index structure for fast and scalable similarity search in high dimensions," in *International Conference on Scientific and Statistical Database Management*, pp. 167–184, Springer, 2009.

[8] *On Strict Homogeneous Lyapunov Function for Generalized Homogeneous PI Controller*, 9 2021.

[9] F. L. Mouël, "Youpi: retour d'expérience et travaux futurs sur une plateforme fog/edge computing," 7 2019.

[10] P. Weisenburger, M. Köhler, and G. Salvaneschi, "Distributed system development with scalaloci," *Proceedings of the ACM on Programming Languages*, vol. 2, pp. 129–30, 10 2018.

[11] L. A. Martins, G. A. M. Sborz, F. Viel, and C. A. Zeferino, "Sbcci - an svm-based hardware accelerator for onboard classification of hyperspectral images," in *Proceedings of the 32nd Symposium on Integrated Circuits and Systems Design*, pp. 18–6, ACM, 8 2019.

[12] C. M. Zurita and F. Assis, "Hybrid control strategies for greenhouse climate regulation: Pid, fuzzy, and neuro-fuzzy comparative implementation in temperate-dry crop systems," in *2025 XV Symposium on Computing Systems Engineering (SBESC)*, pp. 1–6, IEEE, 11 2025.

[13] A. A. Klishin, D. J. Singer, and G. van Anders, "Avoidance, adjacency, and association in distributed systems design," *Journal of Physics: Complexity*, vol. 2, pp. 025015–, 4 2021.

[14] Y. Teng, J. Qi, L. Liu, S. Wang, L. Xu, and C. Fan, "Secure synchronized spatio-temporal trajectory similarity search," in *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 964–973, IEEE, 11 2023.

[15] T. Srinivasan, R. Chandrasekar, V. Vijaykumar, V. Mahadevan, A. Meyyappan, and A. Manikandan, "Localized tree change multicast protocol for mobile ad hoc networks," in *2006 International Conference on Wireless and Mobile Communications (ICWMC'06)*, pp. 44–44, IEEE, 2006.

[16] C. Zhu, W. Zheng, X. Fan, X. Deng, S. Liu, L. Yi, W. Xi, and Y.-S. Jeong, "Graph-empowered multidimensional target full-coverage reliability for internet of everything," *IEEE Internet of Things Journal*, vol. 12, pp. 3707–3719, 2 2025.

[17] A. Mampage and R. Buyya, "Cloudsimsc: A toolkit for modeling and simulation of serverless computing environments," in *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, pp. 550–557, IEEE, 12 2023.

[18] M. Erradi, "Message from the general chairs," in *2023 42nd International Symposium on Reliable Distributed Systems (SRDS)*, pp. ix–ix, IEEE, 9 2023.

[19] J. Han, X. Wei, R. Chen, and H. Chen, "Seraph: A performance-cost aware tuner for training reinforcement learning model on serverless computing," in *Proceedings of the 15th ACM SIGOPS Asia-Pacific Workshop on Systems*, pp. 95–101, ACM, 9 2024.

[20] P. Afanasev, A. Ilyushina, S. Kolesnichenko, P. Komissarov, and E. Zheleznov, "Environmental monitoring using distributed system theory," in *SGEM International Multidisciplinary Scientific GeoConference EXPO Proceedings*, vol. 21, pp. 247–254, STEF92 Technology, 12 2021.

[21] A. M. Ahmed, M. A. Saeed, A. A. Hamood, A. A. Alazab, and K. A. Ahmed, "Comparative study of static analysis and machine learning approaches for detecting android banking malware," in *2023 3rd International Conference on Emerging Smart Technologies and Applications (eSmarTA)*, pp. 1–8, IEEE, 10 2023.

[22] R. Chandrasekar, R. Suresh, and S. Ponnambalam, "Evaluating an obstacle avoidance strategy to ant colony optimization algorithm for classification in event logs," in *2006 International Conference on Advanced Computing and Communications*, pp. 628–629, IEEE, 2006.

[23] N. Bronson, A. Aghayev, A. Charapko, and T. Zhu, *HotOS - Metastable failures in distributed systems*. ACM, 6 2021.

[24] D. Telezhenko, "Forecasting and analytics in virtual distributed systems: Using machine learning models and analytical tools," *Bulletin of V.N. Karazin Kharkiv National University, series «Mathematical modeling. Information technology. Automated control systems»*, pp. 46–51, 12 2023.

[25] K. Umapathy, S. Prabakaran, T. Dineshkumar, M. A. Archana, D. K. Sri, and A. N. Aledaily, *Theories of blockchain and distributed systems*, pp. 52–68. CRC Press, 9 2023.

[26] F. Wu, M. Dong, G. Mo, and H. Chen, "Treesls: A whole-system persistent microkernel with tree-structured state checkpoint on nvm," in *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 1–16, ACM, 10 2023.

[27] S. S. Supase and R. Ingle, *Election-Quorum-Based Coordinator Election Algorithm for Distributed Systems*, pp. 125–134. Springer Singapore, 10 2019.

[28] D. Le-Phuoc and M. Hauswirth, *Semantic Stream Processing*, pp. 1505–1513. Springer International Publishing, 2 2019.

[29] S. M. Shithil and M. A. Adnan, "A prediction based replica selection strategy for reducing tail latency in geo-distributed systems," *IEEE Transactions on Cloud Computing*, vol. 11, pp. 2954–2965, 7 2023.

[30] "Ieee transactions on parallel and distributed systems," 10 2023.

[31] G. Kosec and J. Slak, "Rbf-fd based dynamic thermal rating of overhead power lines," *WIT transactions on engineering sciences*, vol. 120, pp. 255–262, 7 2018.

[32] F. Schnell, "Multicast communication for increased data exchange in data-intensive distributed systems," 1 2018.

[33] T. Nowak, U. Schmid, and K. Winkler, "Topological characterization of consensus in distributed systems," *Journal of the ACM*, vol. 71, pp. 1–48, 11 2024.

[34] J. Pennekamp, "Evolving the industrial internet of things: The advent of secure collaborations," in *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, pp. 1–6, IEEE, 5 2024.

[35] S. Durga, P. Gupta, L. Kharb, P. Ranjit, V. H. R. Dornadula, K. C. Modak, and G. Manoharan, "9leveraging distributed systems for improved educational planning and resource allocation," 3 2024.

[36] R. Chandrasekar and T. Srinivasan, "An improved probabilistic ant based clustering for distributed databases," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI*, pp. 2701–2706, 2007.

[37] Y. Zhong, Y. Ao, D. Li, J. Li, and D. Zeng, "Adswitch: Adaptive cpu scheduling for distributed systems," in *2023 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pp. 34–41, IEEE, 12 2023.

[38] R. A. Al-Saphory and N. J. Al-Jawari, "Sensor and regional gradient detectability of distributed systems," *Journal of Physics: Conference Series*, vol. 1999, pp. 012084–, 9 2021.

[39] R. Lichtenthäler, M. Prechtl, C. Schwille, T. Schwartz, P. Cezanne, and G. Wirtz, "Requirements for a model-driven cloud-native migration of monolithic web-based applications," *SICS Software-Intensive Cyber-Physical Systems*, vol. 35, pp. 89–100, 8 2019.

[40] G. Hu, X. Yao, and Y. Yu, "Collie: Federated query processing via reinforcement learning," in *2025 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, pp. 1284–1291, IEEE, 10 2025.

[41] Z. Yuan, J. Shi, P. Zhou, N. Z. Gong, and L. Sun, "Badtoken: Token-level backdoor attacks to multi-modal large language models," in *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29927–29936, IEEE, 6 2025.

[42] T. Pusztai, S. Nastic, P. Raith, S. Dustdar, D. Vij, and Y. Xiong, "Vela: A 3-phase distributed scheduler for the edge-cloud continuum," in *2023 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 161–172, IEEE, 9 2023.

[43] H. Geppert, F. Dürr, and K. Rothermel, "Efficient conflict graph creation for time-sensitive networks with dynamically changing communication demands," *IEEE Transactions on Network and Service Management*, pp. 1–1, 1 2025.

[44] S. Joshi, S. Ameta, and G. Lavania, "Balanced load in distributed system with nosql middleware," *Journal of emerging technologies and innovative research*, 5 2019.

[45] V. K. Yadav, "Improve the scalability of system through distributed system," 3 2021.

[46] A. Rutkas and L. A. Vlasenko, "On a differential game in a nondamped distributed system," *Mathematical Methods in the Applied Sciences*, vol. 42, pp. 6155–6164, 7 2019.

[47] S. B. Kaleel and A. Abhari, "Tafe-ai: A trust-augmented fog and edge ai enabled network integrated with decentralized identity," in *2024 Annual Modeling and Simulation Conference (ANNSIM)*, pp. 1–13, IEEE, 5 2024.

[48] "2023 4th international conference on information science, parallel and distributed systems (ispds)," in *2023 4th International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*, pp. 1–1, IEEE, 7 2023.

[49] D. Lindsay, S. S. Gill, D. Smirnova, and P. Garraghan, "The evolution of distributed computing systems: from fundamental to new frontiers," *Computing*, vol. 103, pp. 1859–1878, 1 2021.

[50] V. Vijaykumar, R. Chandrasekar, and T. Srinivasan, "An obstacle avoidance strategy to ant colony optimization algorithm for classification in event logs," in *2006 IEEE Conference on Cybernetics and Intelligent Systems*, pp. 1–6, IEEE, 2006.

[51] P. Gaj, I. Smołka, and J. Stój, "Reliability analysis of m2m cyclic data transfer based on ad-hoc wireless p2p link," in *2023 IEEE International Conference on Big Data (BigData)*, pp. 5048–5056, IEEE, 12 2023.

[52] M. Bisen, "Integration technology of distributed system based on multi-objective hotopy algorithm," *Distributed Processing System*, vol. 3, 7 2022.

[53] X. Sui, Z. Zhang, Y. Zeng, D. Liu, L. Li, and L. Liu, "The design and implementation on distributed system," 6 2018.

[54] A. Rashidov, A. Akhatov, I. Aminov, D. Mardonov, and A. Dagur, *Distribution of data flows in distributed systems using hierarchical clustering*, pp. 207–212. CRC Press, 6 2024.

[55] X. Li, Z. Yang, M. Wu, H. Chen, and B. Zang, "Object-aware memory compression for smartphones," *ACM Transactions on Architecture and Code Optimization*, vol. 22, pp. 1–26, 12 2025.

[56] M. Nosrati and M. Fazlali, "Community-based replica management in distributed systems," *International Journal of Web Information Systems*, vol. 14, pp. 41–61, 4 2018.

[57] H. Zhang, N. Lu, R. Song, and J. Li, "Modelling and optimizing for distributed systems with limited resource migration capacity," *Journal of Physics: Conference Series*, vol. 1670, pp. 012026–, 11 2020.

[58] F. Ansari and A. Afzal, *A Grid-Connected Distributed System for PV System*, pp. 919–924. Springer Singapore, 1 2021.

[59] R. Yadav and R. Gaurkar, "Migration from virtualization to dockerization in cloud," *International Journal of Science and Research (IJSR)*, vol. 13, pp. 545–548, 6 2024.

[60] Z. Dong, C. Tang, J. Wang, Z. Wang, H. Chen, and B. Zang, "Optimistic transaction processing in deterministic database," *Journal of Computer Science and Technology*, vol. 35, pp. 382–394, 3 2020.

[61] A. Furutanpey, P. A. Frangoudis, P. Szabo, and S. Dustdar, "Adversarial robustness of bottleneck injected deep neural networks for task-oriented communication," in *2025 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*, pp. 1–6, IEEE, 5 2025.

[62] R. Ross, *The Business of Managing Distributed Systems*, pp. 10–1–10–14. Auerbach Publications, 7 2019.

[63] Z. J. Hamad and S. R. M. Zeebaree, "Recourses utilization in a distributed system: A review," 10 2021.

[64] R. Malik, C. Ramachandran, I. Gupta, and K. Nahrstedt, "Samera: a scalable and memory-efficient feature extraction algorithm for short 3d video segments.," in *IMMERSCOM*, p. 18, 2009.

[65] O. Okusanya, "Consensus in distributed systems: Raft vs crdts," 5 2019.

[66] I. Argyroulis, "Recent advancements in distributed system communications," 1 2021.

[67] E. A. Saksonov, Y. L. Leokhin, and V. N. Azarov, "Information interaction in distributed systems," in *2020 International Conference Quality Management, Transport and Information Security, Information Technologies (IT&QM&IS)*, pp. 15–19, IEEE, 9 2020.