WHEN PREDICTIONS TAKE TIME: COST-AWARE PREDICTOR SELECTION FOR LEARNING-AUGMENTED SCHEDULING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning-augmented scheduling algorithms use predictions of unknown job processing times to improve over non-clairvoyant baselines, but prior work often treats predictions as costless. In practice, more accurate predictors require more complex models and longer inference times; this latency is itself a scheduling cost. We fill this gap by studying predictor selection for single-machine scheduling to minimize total completion time: given this accuracy–latency tradeoff, which predictor should be used? We call predictor selection *cost-aware* when it accounts for inference latency alongside prediction accuracy, and *cost-unaware* when it minimizes prediction error alone. We introduce an algorithm-agnostic framework in which any learning-augmented scheduler with an error-response curve can be analyzed. We prove that inference latency τ contributes an unavoidable $\Theta(n\tau)$ term to the objective via an adaptive-adversary lower bound, and establish that the cost-aware objective has a unique optimal complexity $\theta^\star$ following a power-law scaling: heavier workloads justify more complex predictors. Ignoring prediction cost incurs a non-vanishing performance ratio of $1 + \kappa$, where $\kappa = n\tau/\mathrm{OPT}$. Experiments on the ATLAS GPU-cluster trace, the Azure Functions serverless trace, and the Alibaba microservices trace confirm the theoretical results.

1 INTRODUCTION

Scheduling jobs without knowing processing times is a classical challenge in computing systems (Motwani et al., 1994), known as non-clairvoyant scheduling. This problem has gained renewed urgency as cloud platforms, GPU training clusters, and serverless infrastructures now process large volumes of jobs and requests (Weng et al., 2022; Zhang et al., 2021). Without job durations, a scheduler cannot use size-based algorithms such as Shortest Processing Time (SPT) (SMITH, 1956) and must rely on algorithms like Round Robin (RR) that treat all jobs equally. The gap between clairvoyant and non-clairvoyant performance has driven a rapidly growing research agenda on learning-augmented scheduling (Mitzenmacher & Vassilvitskii, 2022), which incorporates job size predictions into decision-making to aim at near-optimal performance with accurate predictions and worst-case guarantee when predictions are bad (Purohit et al., 2018). Prior competitive analysis results for learning-augmented scheduling treat predictions as freely available when scheduling decisions are made (Purohit et al., 2018; Lattanzi et al., 2020; Im et al., 2023). In practice, however, every prediction requires model inference. Inference takes time.

To illustrate the impact of prediction cost from model inference, consider single-machine scheduling to minimize total completion time $\sum C_j$, where C_j is the completion time of job j, in Figure 1. Four jobs with unknown sizes $p_j^* \in \{1, 3, 5, 8\}$ (in arbitrary time units) are available at $r_j = 0$. With access to the true sizes, SPT is optimal (SMITH, 1956) and achieves a total completion time of 31. Without any size information, RR achieves 45. Shortest Predicted Job First (SPJF) (Purohit et al., 2018) improves upon non-clairvoyant algorithms by sorting jobs according to predicted sizes $\widehat{p}_j$ and executing them non-preemptively. A fast predictor with inference latency $\tau = 1$ that misranks two similarly-sized jobs still achieves 37, well below RR: even imperfect predictions add substantial value when obtained quickly. A perfect predictor with $\tau = 4$, however, recovers the exact SPT ranking yet produces the worst outcome of 47, exceeding even RR. The bottleneck is not prediction quality but prediction latency: every job waits for inference to complete before execution begins, contributing

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

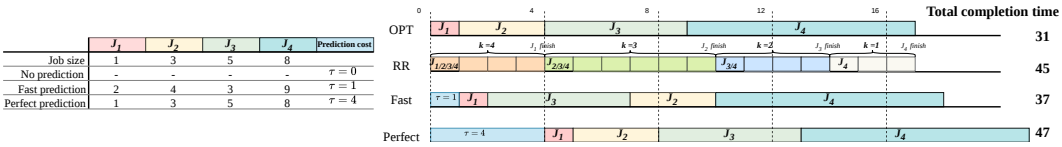

Figure 1: Toy example: four batch jobs with sizes $(p_1^*, p_2^*, p_3^*, p_4^*) = (1, 3, 5, 8)$ scheduled under four scheduling algorithms. OPT (SPT on true job sizes) achieves $\sum C_j = 31$. RR requires no predictions but pays $\sum C_j = 45$, under shared processor. SPJF with a fast predictor ($\tau = 1$, mild ranking error) yields $\sum C_j = 37$. In contrast, SPJF with a perfect but slower predictor ($\tau = 4$) achieves $\sum C_j = 47$, worse than no prediction, because inference latency outweighs the ranking improvement.

$n\tau = 16$ to the objective and offsetting the benefit of a perfect ranking. This accuracy-latency tradeoff is not hypothetical. More expressive predictors may improve size estimates, but they can also require more inference time or computational resources (Shahout et al., 2024). When the scheduling objective is time-sensitive, as $\sum C_j$ is, the scheduler must decide how much prediction quality is worth waiting for. A natural question arises:

When should a scheduler wait for a better prediction?

Related work. The competitive-analysis line of learning-augmented scheduling accounts for prediction error but treats predictions as costless. For batch preemptive non-clairvoyant single-machine $\sum C_j$, SPJF and Preferential Round Robin (PRR) (Purohit et al., 2018) were proposed under ℓ_1 job-size error η. SPJF satisfies $\sum_j C_j \leq \mathrm{OPT} + (n-1)\eta$, a tight additive bound that already shows how prediction error can be amplified by n in completion-time objectives; PRR combines SPJF with Round Robin to trade consistency against robustness. Subsequent work showed that ℓ_1 is not Lipschitz with respect to $\sum C_j$ and introduced a hybrid-instance error ν satisfying $\ell_1 \leq \nu \leq n\ell_1$ (Im et al., 2023). The prediction target has also been shifted from processing times to job permutations, using an inversion-weighted error, with extensions to weighted jobs and unrelated machines (Lindermayr & Megow, 2025). For online makespan under restricted assignment, compact per-machine weight predictions with multiplicative error $\eta = \max_i \widehat{w}_i/w_i$ break the $\Omega(\log m)$ worst-case barrier when predictions are accurate (Lattanzi et al., 2020). These works demonstrate that learning-augmented scheduling supports many prediction objects and error metrics. Our work addresses an orthogonal issue absent from these guarantees: the time required to obtain a prediction can itself affect the scheduling objective. Prediction cost has been studied in adjacent models. A fixed monetary cost per oracle query has been analyzed for ski rental and the Bahncard problem (Drygala et al., 2023). A budget on instantaneous advice queries, including a non-clairvoyant scheduling variant achieving ratio $2 - (B/N)^2$ with B revealed job sizes, was studied by Benomar & Perchet (2023). SKIPPREDICT (Shahout & Mitzenmacher, 2024) models per-job prediction cost under two regimes (external fixed cost; server-time cost) with a binary skip-or-predict decision; a recent survey identifies prediction cost as a central open challenge (Mitzenmacher & Shahout, 2025). Prediction portfolios (Dinitz et al., 2022) select among k discrete predictors with $O(\log k)$ overhead. In all cases the cost is a flat penalty, a query budget, or a per-job overhead in a stochastic queue; none captures the inference latency in $\sum C_j$, nor optimizes predictor complexity as a continuous variable. Jiang et al. (2025) show empirically that a more accurate predictor can be slower to run, so that beyond some point the added inference latency outweighs the scheduling benefit of higher accuracy. Building on this, we ask which predictor complexity a workload should select. We show that the latency cost cannot be removed by scheduling around it, that a unique best complexity exists and can be computed in closed form, and that a single workload quantity determines whether the choice matters at all.

Our contributions. We introduce a *cost-aware predictor-selection framework* for single-machine scheduling to minimize total completion time $\sum C_j$. The framework is algorithm-agnostic: it requires from the downstream scheduler only a function $\rho(\eta)$ that maps prediction error η to competitive ratio, without depending on the scheduler's internal mechanism. We prove that inference latency contributes an unavoidable $\Theta(n\tau)$ additive term to the objective, with matching upper and lower bounds via a wait-then-schedule decomposition and an adaptive-adversary construction. Under power-law assumptions on latency and error, the cost-aware objective $J(\theta) = n\tau(\theta) + \rho(\eta(\theta))\,\mathrm{OPT}$ has a unique optimal complexity satisfying $\theta^\star \propto (\mathrm{OPT}/n)^{1/(a+b)}$, where a and b govern how prediction

error decays and inference latency grows with predictor complexity respectively. Ignoring prediction cost incurs a non-vanishing performance ratio of $1+\kappa$, where $\kappa = n\tau/\mathrm{OPT}$ can be computed directly from the trace and predictor specification, telling whether cost-aware selection is necessary. We validate the framework on three production traces that span the regime from negligible to significant prediction cost: the ATLAS GPU-cluster trace, the Azure Functions serverless trace, and the Alibaba microservices trace, using a controlled noise-injection predictor family and trained predictor families. On the microservices trace, where accuracy improves monotonically with complexity, cost-aware selection chooses a predictor about $16\times$ less complex than the most accurate one, which would increase total cost by 8.5%. Figure 2 provides a visual road-map of the theoretical development; Algorithm 1 gives a reproducible evaluation protocol for any trace and model class.

2 Model and Preliminaries

Prediction is costly when inference delays scheduling. We model this cost through three objects: the non-clairvoyant scheduling benchmark, a scheduler error-response curve ρ (formally, a competitive-ratio upper envelope), and a predictor family mapping complexity θ to batch latency $\tau(\theta)$ and induced normalized error $\eta(\theta)$. These objects define the cost-aware predictor-selection problem studied in Section 3. Definitions of notations and a summary of all symbols are given in Appendix A.

Problem definition. We study $1 \mid r_j, \mathrm{pmtn} \mid \sum_j C_j$: jobs arrive over time on a single machine, preemption is allowed, and the objective is total completion time. In the clairvoyant setting, Shortest Remaining Processing Time (SRPT) is optimal under release dates; in the batch case $r_j = 0$ for all j, SRPT coincides with SPT, also called Shortest Job First (SJF), so clairvoyant optimum is obtained by ordering jobs by increasing sizes (Schrage, 1968; SMITH, 1956). The batch assumption removes release-time structure, while the single-machine assumption removes parallel slack that could partially mask inference delay. The difficulty is non-clairvoyance: the scheduler does not know the true processing time p_j^* until job j completes. Learning-augmented scheduling relaxes this information barrier by giving the scheduler predictions $\widehat{p}_j$ of the true processing times. Prior algorithms state guarantees as functions of prediction error (Purohit et al., 2018; Im et al., 2023); after normalizing the native error measure of the chosen scheduler, we view such guarantees as an error-dependent competitive-ratio curve ρ, defined formally next.

Scheduler interface. We use a learning-augmented scheduler only through an error-dependent competitive guarantee. Let $\mathcal{I}$ be the instance class, $I \in \mathcal{I}$, and $\widehat{\mathbf{z}}$ be the prediction object supplied to scheduler $\mathcal{A}$; we set $\widehat{\mathbf{z}} = \widehat{\mathbf{p}}$ a vector of job-size predictions. Let $\mathrm{ALG}_{\mathcal{A}}(I, \widehat{\mathbf{z}})$ denote the resulting total completion time and $\mathrm{OPT}(I)$ be the clairvoyant optimum. Each scheduler defines a native error measure (e.g., ℓ_1 for SPJF), which we convert to a dimensionless error $\eta_{\mathcal{A}}(I, \widehat{\mathbf{z}}) \geq 0$, with $\eta_{\mathcal{A}} = 0$ for perfect predictions. A response curve $\rho_{\mathcal{A}}$ is any non-decreasing upper envelope satisfying $\rho_{\mathcal{A}}(x) \geq \sup_{I \in \mathcal{I}} \sup_{\widehat{\mathbf{z}}:\, \eta_{\mathcal{A}}(I,\widehat{\mathbf{z}}) \leq x} \frac{\mathrm{ALG}_{\mathcal{A}}(I,\widehat{\mathbf{z}})}{\mathrm{OPT}(I)}$. Thus, for every $I, \widehat{\mathbf{z}}$, $\mathrm{ALG}_{\mathcal{A}}(I, \widehat{\mathbf{z}}) \leq \rho_{\mathcal{A}}(\eta_{\mathcal{A}}(I, \widehat{\mathbf{z}}))\, \mathrm{OPT}(I)$. For an additive guarantee $\mathrm{ALG} \leq \alpha\, \mathrm{OPT} + g$, where $g \geq 0$ is the error-dependent additive term, we take $\eta_{\mathcal{A}} = g/\mathrm{OPT}$ and $\rho_{\mathcal{A}}(x) = \alpha + x$. For example, the SPJF bound $\mathrm{OPT} + (n-1)\ell_1$ corresponds to $\eta_{\mathcal{A}} = (n-1)\ell_1/\mathrm{OPT}$ and $\rho_{\mathcal{A}}(x) = 1 + x$. Thus $\eta_{\mathcal{A}}$ is not a universal error metric, but the normalized argument of the scheduler's own guarantee. We write η and ρ when $\mathcal{A}$ is clear.

Predictor model. We model predictors by a complexity parameter $\theta \in \Theta = [\theta_{\min}, \infty)$, where $\theta_{\min} > 0$ is the simplest available predictor. For a fixed instance I and scheduler $\mathcal{A}$, choosing θ produces, in our setting, job-size predictions $\widehat{\mathbf{p}}_\theta(I)$ after batch inference latency $\tau(\theta)$. Here $\tau(\theta)$ is total batch latency; since I is fixed, dependence on instance parameters such as n is suppressed. Equivalently, $\widehat{\mathbf{p}}_\theta(I)$ is the prediction object $\widehat{\mathbf{z}}_\theta(I)$ in the scheduler interface. The prediction induces normalized scheduler error $\eta_I(\theta) := \eta_{\mathcal{A}}(I, \widehat{\mathbf{z}}_\theta(I)) = \eta_{\mathcal{A}}(I, \widehat{\mathbf{p}}_\theta(I))$. When I and $\mathcal{A}$ are fixed, we write $\eta(\theta)$. Different native error measures, such as the ℓ_1 job-size error (Purohit et al., 2018) or the objective-sensitive ν measure (Im et al., 2023), are absorbed through normalization. We use only the induced triple $(\tau(\theta), \eta(\theta), \rho)$.

Assumption 1 (Inference latency). *$\tau : \Theta \to \mathbb{R}_{\geq 0}$ is differentiable and convex, with $\tau'(\theta) > 0$ for all $\theta \in \Theta$, and $\tau(\theta) \to \infty$ as $\theta \to \infty$.*

Assumption 2 (Prediction error). *$\eta : \Theta \to \mathbb{R}_{\geq 0}$ is differentiable and strictly convex, with $\eta'(\theta) < 0$ for all $\theta \in \Theta$, and $\inf_{\theta \in \Theta} \eta(\theta) = \eta_{\min} \geq 0$.*

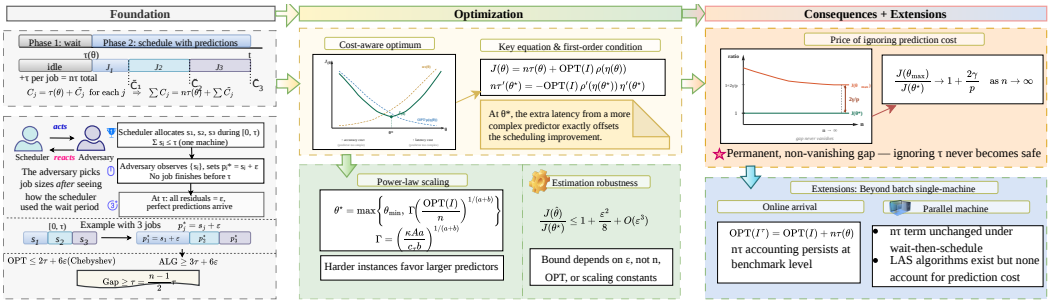

Figure 2: Overview flowchart for road-map of the prediction cost theory. Foundation (left): Theorem 1 decomposes wait-then-schedule cost into $n\tau(\theta) + \rho(\eta)\,\mathrm{OPT}(I)$; Theorem 2 proves the $n\tau$ term is unavoidable via an adaptive adversary. Optimization (center): Theorem 3 establishes strict convexity of $J(\theta)$ and a unique optimum $\theta^\star$; Corollary 2 gives a closed-form scaling law $\theta^\star \propto \left(\frac{\mathrm{OPT}}{n}\right)^{1/(a+b)}$. Consequences and extensions (right): Theorem 4 shows ignoring prediction cost yields a permanent gap $1 + \frac{2\gamma}{p}$ as $n \to \infty$; extensions to online arrivals and parallel machines are outlined with details.

Assumption 3 (Competitive-ratio curve)**.** *The selected response curve ρ is differentiable, convex, non-decreasing on $\mathbb{R}_{\geq 0}$, and strictly increasing on $\eta(\Theta)$, with $\rho(0) = \rho_0 \geq 1$.*

Derivatives at $\theta_{\min}$ are interpreted as right derivatives. These assumptions encode an accuracy-latency tradeoff with diminishing returns. They also make the cost-aware objective well behaved: $\rho(\eta(\theta))$ is convex by the standard composition rule (Boyd & Vandenberghe, 2004), and is strictly convex because ρ is strictly increasing on $\eta(\Theta)$ while η is strictly convex. Together with $\tau(\theta) \to \infty$, this yields a unique cost-aware optimum over Θ. The power-law family below gives one concrete instantiation, and the experiments test how well these structural assumptions match practical predictor families.

Parametric form. For closed-form results, we instantiate the predictor family with power laws for batch latency and for the reducible component of normalized error:

$$\tau(\theta) = c_\tau\,\theta^b, \qquad \eta(\theta) = \eta_{\min} + A\,\theta^{-a},$$

where $c_\tau, A > 0$, $a > 0$, and $b \geq 1$. Since $\theta_{\min} > 0$, the latency curve is increasing and convex for $b \geq 1$, while the reducible error term $A\theta^{-a}$ is decreasing and strictly convex in θ. The constant $\eta_{\min} \geq 0$ represents irreducible prediction error from inherent variability or limited features. This form is motivated by empirical scaling laws in which predictive loss decreases approximately as a power law with model scale (Hestness et al., 2017; Kaplan et al., 2020); here it is used as a tractable model for the induced normalized error $\eta(\theta)$. The scheduling response is abstracted separately by the upper envelope ρ.

3 Prediction Cost: Theory

We analyze the cost-aware predictor selection problem in three steps: the unavoidable cost of prediction latency, the optimal complexity choice, and the price of ignoring prediction cost. We first identify how inference delay enters $\sum_j C_j$. Under a wait-then-schedule policy, prediction delay shifts every completion time by $\tau(\theta)$ and contributes $n\tau(\theta)$. We then show that this dependence is unavoidable up to constants, even if an algorithm processes jobs before predictions are revealed.

3.1 The $n\tau$ Cost of Prediction

Theorem 1 (Wait-then-schedule bound)**.** *Fix a batch instance I, predictor complexity θ, and learning-augmented scheduler $\mathcal{A}$ with error-response curve ρ. Under a wait-then-schedule policy, which idles the machine during $[0, \tau(\theta))$ and runs $\mathcal{A}$ using predictions $\widehat{\mathbf{p}}$, the resulting completion times satisfy*

$$\sum_{j=1}^{n} C_j^{\mathrm{wait}} \leq n\tau(\theta) + \rho(\eta(\theta))\mathrm{OPT}(I).$$

Every wait-then-schedule policy also satisfies $\sum_{j=1}^{n} C_j^{\mathrm{wait}} \geq n\tau(\theta) + \mathrm{OPT}(I)$.

The proofs of Theorems 1, 3, and 4 and of Corollary 2 are given in Appendix B. Theorem 1 charges the full $n\tau(\theta)$ because the machine idles during inference. One might hope to do better by running jobs during the wait, so that some work is already done when predictions arrive. Theorem 2 shows that this lowers the loss by at most a constant factor.

Theorem 2 (Adaptive-adversary lower bound)**.** *Fix $n \geq 2$ and predictor complexity θ with $\tau(\theta) > 0$. For every deterministic preemptive non-clairvoyant algorithm $\mathcal{A}$ on a single machine that knows n and $\tau(\theta)$ but not the processing times before predictions arrive at $\tau(\theta)$, an adaptive adversary observing the service given to each job during $[0, \tau(\theta))$ constructs a batch instance I such that, even if perfect predictions arrive at $\tau(\theta)$,*

$$\sum_{j=1}^{n} C_j^{\mathcal{A}} \geq \mathrm{OPT}(I) + \frac{n-1}{2}\tau(\theta).$$

Proof. Write $\tau = \tau(\theta)$, and let s_j be the service given to job j during $[0, \tau)$, so $\sum_j s_j \leq \tau$. After observing the service vector, the adversary sets $p_j^* = s_j + \varepsilon$ for arbitrary $\varepsilon > 0$. No job completes before τ, and all jobs have residual processing time ε at time τ. Even the best continuation after τ therefore has completion times at least $\tau + \varepsilon, \tau + 2\varepsilon, \ldots, \tau + n\varepsilon$, so $\sum_j C_j^{\mathcal{A}} \geq n\tau + \frac{n(n+1)}{2}\varepsilon$. If $s_{(1)} \leq \cdots \leq s_{(n)}$ are the sorted service amounts, then the clairvoyant optimum schedules jobs in increasing $s_j + \varepsilon$ and satisfies $\mathrm{OPT}(I) = \sum_{k=1}^{n}(n-k+1)s_{(k)} + \frac{n(n+1)}{2}\varepsilon$. Since $s_{(k)}$ is non-decreasing and the weights $n-k+1$ are non-increasing, Chebyshev's sum inequality for oppositely sorted sequences gives

$$\sum_{k=1}^{n}(n-k+1)s_{(k)} \leq \frac{n+1}{2}\sum_{k=1}^{n} s_k \leq \frac{n+1}{2}\tau.$$

Therefore, $\mathrm{OPT}(I) \leq \frac{n+1}{2}\tau + \frac{n(n+1)}{2}\varepsilon$. Subtracting this upper bound on $\mathrm{OPT}(I)$ from the lower bound on $\sum_j C_j^{\mathcal{A}}$ cancels the ε terms and yields $\sum_j C_j^{\mathcal{A}} - \mathrm{OPT}(I) \geq n\tau - \frac{n+1}{2}\tau = \frac{n-1}{2}\tau$. □

Pre-prediction policies such as round-robin sharing, FIFO-like concentration, and adaptive mixtures behave differently, and cannot be enumerated. The proof avoids this: the only feature of an algorithm that affects the outcome is how long it ran each job before predictions arrive, the service vector $(s_1, \ldots, s_n)$. Because job sizes are unknown until then, the adversary sets each true size just above the time already spent on that job. No job finishes, every job still carries ε work, and the processing done during the wait is wasted. One adversary, replying to one vector, covers every algorithm at once, and the bound holds uniformly for every vector satisfying $\sum_j s_j \leq \tau$.

Corollary 1 (Tightness of the prediction-latency term)**.** *Theorems 1 and 2 together show that the additive prediction-latency dependence in total completion time is $\Theta(n\tau(\theta))$: wait-then-schedule pays exactly $n\tau(\theta)$, and no deterministic non-clairvoyant policy can avoid a worst-case additive loss of $(n-1)\tau(\theta)/2$.*

Adversary model. Theorem 2 is pathwise: for any realization of a randomized algorithm's internal randomness, the service vector during $[0, \tau)$ is deterministic and the same construction applies, so the bound holds almost surely, and in expectation, against adaptive adversaries that observe the realized service vector. The result does not claim a randomized lower bound against oblivious adversaries. For a deterministic algorithm, the service vector under the no-completion history is determined by the algorithm, n, and τ, so the hard instance $p_j^* = s_j + \varepsilon$ is itself a fixed instance specified before the run. No single instance can defeat every algorithm, since an algorithm that hard-codes the optimal order of a given instance pays nothing above OPT on it; the per-algorithm form is the standard one in non-clairvoyant scheduling, as in the Round Robin lower bound of Motwani et al. (1994). We prove no distributional lower bound for random workloads.

3.2 Cost-Aware Predictor Selection

Theorems 1–2 identify the latency term that must be balanced against prediction quality. This leads to the cost-aware objective $J(\theta) = n\tau(\theta) + \mathrm{OPT}(I)\rho(\eta(\theta))$, where the first captures the unavoidable aggregate prediction-latency cost and the second term is the downstream scheduling guarantee.

Theorem 3 (Cost-aware optimality condition)**.** *Under Assumptions 1–3, J is strictly convex on Θ and has a unique minimizer $\theta^\star \in \Theta$. If the right derivative satisfies $J'_+(\theta_{\min}) \geq 0$, the latency already dominates at the simplest predictor and $\theta^\star = \theta_{\min}$. Otherwise, $\theta^\star$ is the unique interior solution of*

$$n\tau'(\theta^\star) = -\mathrm{OPT}(I)\rho'(\eta(\theta^\star))\eta'(\theta^\star).$$

The objective J is not a surrogate chosen in advance: both terms are total completion time in the same units, the first being the latency term of Theorems 1–2 and the second the scheduler's own guarantee, so no weight is tuned. Strict convexity of J is derived from the qualitative shapes in Assumptions 1–3; no parametric form is needed for existence and uniqueness, and the power law enters only in the closed form below. The first-order condition balances marginal delay cost (left side) against marginal scheduling benefit (right side). Under the power-law forms of Section 2, this balance admits a closed form.

Corollary 2 (Power-law predictor scaling)**.** *Under the parametric forms of Section 2 with the linear error-response upper envelope $\rho(\eta) = \rho_0 + c_\rho\eta$, where $c_\rho > 0$ is the slope constant, the cost-aware optimum is*

$$\theta^\star = \max\left\{\theta_{\min},\ \Gamma\left(\frac{\mathrm{OPT}(I)}{n}\right)^{1/(a+b)}\right\}, \qquad \Gamma = \left(\frac{c_\rho\, A\, a}{c_\tau\, b}\right)^{1/(a+b)}.$$

Corollary 2 shows that the relevant workload scale is $\mathrm{OPT}(I)/n$, not n alone: holding n fixed, a larger optimum increases the value of accuracy; holding $\mathrm{OPT}(I)$ fixed, a larger batch increases aggregate latency pressure. Thus larger average completion-time scale favors larger predictors. In particular, more jobs do not call for a cheaper predictor. One might expect a larger batch to raise the delay $n\tau$ and push the choice toward a faster predictor, but the n that inflates the delay also scales the total work. When inference is charged per job, $\tau(\theta) = nf(\theta)$, and job sizes are drawn from a fixed workload, both $n\tau(\theta)$ and $\mathrm{OPT}(I)$ grow as n^2, so n cancels: $\theta^\star$ and κ depend on the workload only through the per-job scale $p_{\text{eff}} = 2\mathrm{OPT}(I)/n^2$, and doubling the number of jobs from the same workload leaves the best predictor unchanged. If instead the batch latency does not grow with n, then $\mathrm{OPT}(I)/n$ grows with n and $\theta^\star$ increases. When the interior solution is active, the scaling term for $\theta^\star$ depends on the slope $c_\rho = \rho'(0)$ but not on ρ_0: predictor selection is driven by the scheduler's *marginal* sensitivity to error, not by its fixed perfect-prediction baseline. The linear upper envelope $\rho(\eta) = \rho_0 + c_\rho\eta$ is used to obtain a closed-form scaling law: any smooth response curve with $\rho(0) = \rho_0$ and $\rho'(0) > 0$ admits the local approximation $\rho(\eta) = \rho_0 + \rho'(0)\eta + o(\eta)$ near $\eta = 0$. This does not assert that LAS response curves are globally linear—prior bounds can be nonlinear or problem-specific (Purohit et al., 2018; Balkanski et al., 2023). For nonlinear smooth ρ, Theorem 3 still characterizes the optimum, but a closed form need not exist in general.

3.3 The Price of Ignoring Prediction Cost

We quantify the penalty, ignoring prediction cost at the predictor-selection stage. A cost-unaware policy selects the lowest-error predictor, regardless of latency; we show this can incur a non-vanishing constant-factor loss in $\sum_j C_j$, even when the downstream scheduler is optimal. Consider n batch-arriving jobs with identical job size $p > 0$ and two predictors: a predictor θ_s with $\tau(\theta_s) = 0$ with error $\eta(\theta_s) = \varepsilon > 0$, and a complex predictor θ_c with perfect predictions $\eta(\theta_c) = 0$ but per-job inference cost $\gamma > 0$, giving batch latency $\tau(\theta_c) = \gamma n$. Since all jobs are identical, prediction error does not affect schedule optimality, so both predictors achieve $\mathrm{OPT}(I)$ after their respective delays.

Theorem 4 (Price of ignoring prediction cost)**.** *In the setting above, a cost-unaware policy selects θ_c while a cost-aware policy selects θ_s. Their realized total-completion-time ratio is*

$$\frac{\sum_{j=1}^n C_j(\theta_c)}{\sum_{j=1}^n C_j(\theta_s)} = 1 + \frac{2\gamma n}{p(n+1)} \xrightarrow[n\to\infty]{} 1 + \frac{2\gamma}{p}.$$

The limiting ratio is at least 2 whenever $\gamma \geq p/2$.

The theorem separates accuracy from value. Perfect predictions are not always worthwhile. The latency of the complex predictor affects every job's completion time, contributing $n\tau(\theta_c) = \gamma n^2$ to the total, precisely the $n\tau$ mechanism identified in Section 3.1. We define the *regime parameter*

$$\kappa(\theta) \;=\; \frac{n\,\tau(\theta)}{\mathrm{OPT}(I)}, \tag{1}$$

which measures prediction cost in units of the optimal schedule. The ratio in Theorem 4 is then exactly $1 + \kappa(\theta_c)$: the fractional loss equals the regime parameter of the chosen predictor. When $\kappa \ll 1$ prediction cost is invisible; when $\kappa = \Omega(1)$ it materially distorts the schedule.

3.4 Sensitivity and Extensions

The optimum $\theta^\star$ depends on $\mathrm{OPT}(I)$, which is unknown before execution. Plugging in an estimate $\widehat{\mathrm{OPT}} = (1+\epsilon)\mathrm{OPT}(I)$ costs only a second-order relative excess of at most $\epsilon^2/8 + O(|\epsilon|^3)$ in J (Proposition 1, Appendix C). For online arrivals, the $n\tau$ accounting persists at the benchmark level, $\mathrm{OPT}(I^\tau) = \mathrm{OPT}(I) + n\tau(\theta)$; under wait-then-schedule on m identical machines, the additive $n\tau(\theta)$ term is unchanged, so the selection framework carries over, while extending Theorem 2 requires a separate argument (Appendix C).

4 Experimental study

We evaluate the operational predictions of the cost-aware framework: (i) the total-cost objective $J(\theta)$ should exhibit an interior optimum, because scheduling regret decreases with predictor complexity while aggregate inference latency $n\tau(\theta)$ increases; (ii) under the power-law model, the optimal complexity should scale with workload pressure as $\theta^\star \propto (\mathrm{OPT}(I)/n)^{1/(\alpha+b)}$; and (iii) a cost-unaware selector that minimizes prediction error alone should become harmful when the regime parameter $\kappa = n\tau(\theta)/\mathrm{OPT}(I)$ is non-negligible. The closest prior work on prediction cost studies M/G/1 queuing models via simulation (Shahout & Mitzenmacher, 2024; Mitzenmacher & Shahout, 2025); broader learning-augmented scheduling experiments treat inference as free (Balkanski et al., 2023). We test on three production traces chosen to span the regime from negligible to significant prediction cost. The Alibaba PAI trace (ATLAS) (Jiang et al.), where GPU-training jobs run for minutes to hours, is the long-job regime, where prediction cost is negligible. The Azure Functions 2021 invocation trace (Azure) (Zhang et al., 2021), where serverless invocations are predominantly sub-second, is the short-job regime. The Alibaba microservices trace (Luo et al., 2022), where a job is a single RPC call, is a regime where measured inference latency alone changes which trained predictor should be deployed. The point of the contrast is that one selection rule gives different answers in different regimes, which is why "how much to predict" has no single answer. We use a semi-synthetic predictor family that matches Assumptions 1–2 by construction, and trained LightGBM (Ke et al., 2017), k-NN, and MLP families.

4.1 Setup

Traces and regime parameter. The two production traces span opposite ends of the job-duration spectrum. **ATLAS** (Jiang et al.) contains 732,355 completed GPU-training jobs from Alibaba's PAI cluster (Weng et al., 2022), with durations ranging from seconds to days. **Azure Functions** (Azure) (Zhang et al., 2021) contains 1.98 million serverless invocations over two weeks, with predominantly sub-second durations. This contrast is deliberate: the same inference latency has very different values of the regime parameter on the two traces. In our test batches, the effective job scale $p_{\text{eff}} = 2\mathrm{OPT}(I)/n^2$ is 694.0 seconds on ATLAS and 0.1259 seconds on Azure, a $5513\times$ difference. Consequently, a 10 ms-per-job predictor has $\kappa = 2.88 \times 10^{-5}$ on ATLAS but $\kappa = 0.159$ on Azure. Thus ATLAS tests the long-job regime, where prediction cost is negligible under measured LightGBM latencies, while Azure tests the short-job regime, where inference latency can change the predictor-selection decision. **Alibaba microservices** is the first 3-minute CallGraph shard of the public cluster-trace-microservices-v2022 release (Luo et al., 2022). A job is one RPC call and p_j^* is its recorded call duration. After removing duplicate records and restricting to remote-invocation calls, the shard contains 2.59M calls.

4.2 Benchmark and evaluation

Processing-time labels. On ATLAS, p_j^* is the job-level task span across the multi-task PAI job hierarchy (Jiang et al.; Weng et al., 2022); on Azure and on the microservices trace, it is the recorded duration of the atomic invocation or call. These are observed durations from a multi-machine

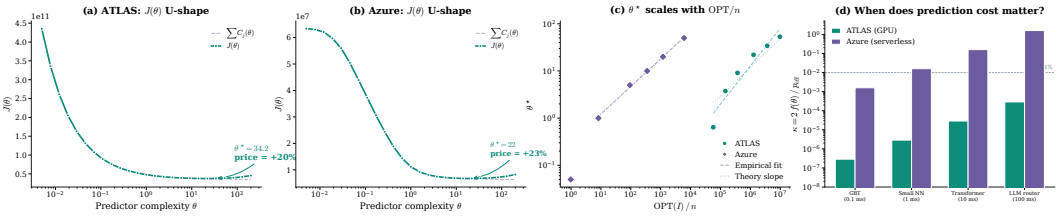

Figure 3: Experimental validation. **(a)–(b)** Total cost $J(\theta)$ on ATLAS and Azure Functions traces, confirming the U-shape predicted by Theorem 3; the cost-unaware choice ($\theta_{\max}$) incurs a 21–23% penalty over $\theta^\star$. **(c)** Optimal complexity $\theta^\star$ scales with workload pressure OPT/n on both traces (empirical slopes 1.24 and 0.61 vs. theoretical 0.61 and 0.57); per-bucket fit with local α yields a median ratio of 1.02 on Azure. **(d)** Regime parameter $\kappa = 2f(\theta)/p_{\text{eff}}$ across predictor types: on ATLAS prediction cost is negligible at all scales; on Azure a transformer-class predictor already reaches $\kappa \approx 16\%$.

environment rather than intrinsic single-machine runtimes, a gap absorbed by the irreducible error floor $\eta_{\min}$ (Appendix D).

Scheduling algorithms. All experiments use batch instances, so SPT is the exact clairvoyant optimum, $\mathrm{OPT}(I) = \sum_{k=1}^{n}(n-k+1)\,p^*_{(k)}$ (SMITH, 1956). Given predictions $\widehat{p}_j$, the scheduler runs SPJF, the standard learning-augmented baseline (Mitzenmacher, 2019; Purohit et al., 2018), processing jobs non-preemptively in non-decreasing $\widehat{p}_j$. Fixing one scheduling policy isolates predictor selection from scheduler design: any variation in $\sum_j C_j$ across θ is attributable to prediction quality and inference cost. Algorithm 1 (Appendix D) states the evaluation protocol.

Model classes. We evaluate two predictor families. *Layer 1 (controlled).* For each seed, we draw one Gaussian vector Z and reuse it across all complexities, setting $\widehat{p}_j = p^*_j \exp(\sigma_\theta Z_j), \qquad \sigma_\theta = c_\sigma \theta^{-a_{\text{noise}}}$, with $a_{\text{noise}} = 0.5$ and $c_\sigma = 2.0$. Thus only the noise amplitude changes with θ. Batch latency is $\tau(\theta) = c_\tau \theta^b$ with $b = 1$, and c_τ is calibrated so that $\kappa(\theta_{\max}) = 0.3$ on each trace. This controlled family isolates the power-law accuracy–latency mechanism from feature engineering. We use 20 independent seeds and report medians. *Layer 2 (trained LightGBM).* We train a LightGBM (Ke et al., 2017) family indexed by θ. On ATLAS, the grid contains 12 configurations from $\theta = 40$ (4 leaves, 10 trees) to $\theta = 204{,}000$ (255 leaves, 800 trees). Models predict log-duration from submit-time features (resource requests, task counts, temporal encodings, and categorical codes), with no early stopping so that complexity is controlled by the grid. We measure $\tau(\theta)$ as the median wall-clock time over 30 single-threaded inference passes after one warm-up pass. Since measured GBT latency is negligible on ATLAS, we also apply a common multiplier λ to all latencies and sweep $\kappa(\theta_{\max})$ from 0.1 to 10. On Azure and on the microservices trace we train the same 12-configuration grid (θ = leaves $\times$ trees) on the first 70% of jobs in time order (on the microservices trace, the 1.36M training calls with non-zero duration) and evaluate on a held-out batch of 10,000 jobs (the last 10,000 calls on the microservices trace), with all latencies measured and no multiplier.

4.3 Results and Discussions

Figure 3(a)–(b) confirms that $J(\theta)$ has an interior minimum on both traces: $\theta^\star = 34.2$ on ATLAS and $\theta^\star = 22.0$ on Azure, with the cost-unaware choice incurring a 21% and 23% penalty respectively. The scaling law (Figure 3(c), Table 3 in Appendix D) is quantitatively accurate on Azure, where the empirical log-log slope is 0.613 versus the predicted 0.572 and the median per-bucket ratio $\theta^\star_e/\theta^\star_t = 1.02$. ATLAS shows the same monotone trend but a steeper global slope (1.24 vs. 0.61), explained by substantial variation in the local regret exponent α_{local} across workload buckets (Table 3): when job sizes are nearly uniform, small noise disrupts rankings (α small), whereas diverse sizes require larger noise to misrank (α large); accounting for this local α restores per-bucket agreement on both traces. Finally, the regime parameter κ (Figure 3(d)) cleanly separates the two traces: on ATLAS ($p_{\text{eff}} = 694\,\text{s}$) even an LLM-class predictor yields $\kappa < 10^{-4}$, making prediction cost negligible, whereas on Azure ($p_{\text{eff}} = 0.13\,\text{s}$) a transformer-class predictor reaches $\kappa \approx 16\%$, confirming that short-job workloads require explicit cost–accuracy balancing.

Table 1: Trained LightGBM on the Alibaba microservices trace (10,000 held-out calls, measured; times in seconds). J is minimized at $\theta^\star = 12{,}600$ (shaded), not at the most accurate model.

θ	κ	$\sum_j C_j$	J	$\sigma_{\log}$	ℓ_1	θ	κ	$\sum_j C_j$	J	$\sigma_{\log}$	ℓ_1
40	0.04%	140,084.0	140,115.3	1.257	117.39	12,600	1.08%	115,937.9	**116,887.4**	0.704	82.53
200	0.09%	128,519.8	128,595.1	1.018	107.74	25,200	2.43%	115,233.6	117,362.2	0.677	75.15
400	0.11%	122,145.9	122,243.2	1.001	106.87	50,800	3.38%	114,886.7	117,845.2	0.670	74.75
800	0.16%	119,301.1	119,441.4	0.994	106.49	76,200	6.46%	114,498.1	120,149.4	0.665	73.94
3,200	0.71%	117,771.7	118,396.4	0.715	83.50	153,000	9.44%	114,509.9	122,774.8	0.659	73.64
6,400	0.85%	116,729.3	117,470.8	0.709	82.96	204,000	14.23%	**114,408.3**	126,866.3	0.658	73.48

Trained predictors on Azure. With measured latencies, the trained LightGBM grid on Azure has an interior minimum of J at $\theta^\star = 76{,}200$, and the largest model costs $+1.69\%$ more ($\theta^\star$ is interior in all 1,000 bootstrap resamples); on ATLAS the same procedure selects the largest model at no penalty. On Azure, however, the cost-unaware selector $\arg\min_\theta \sum_j C_j$ picks the same $\theta^\star$: the penalty of the largest model is 93.9% scheduling and 6.1% latency, and four held-out accuracy measures all degrade past $\theta^\star$ (Appendix E.1), so the right arm here reflects overfitting rather than latency. Overfitting departs from Assumption 2; Appendix E.1 shows it can only move the minimizer left.

When latency alone moves the choice. Separating cost-aware from cost-unaware selection through latency requires a trace where accuracy improves monotonically and κ is non-negligible. Table 1 reports the trained LightGBM grid on the microservices trace ($\mathrm{OPT} = 87{,}529.9$ s on the held-out batch). Accuracy improves at every step: the log-residual spread falls from 1.257 to 0.658, the ℓ_1 error from 117.4 to 73.5, and $\sum_j C_j$ falls at ten of eleven steps (the other rises by 0.01%). The two selectors nevertheless separate: cost-aware selection takes $\theta^\star = 12{,}600$, while the cost-unaware choice is the largest and most accurate model, $204{,}000$, which costs $+8.5\%$ in J. From $\theta^\star$ to the largest model the scheduling term falls by $1{,}530$ s while the latency term rises by $11{,}509$ s, so latency accounts for 115.3% of the penalty. The larger model schedules better and still loses. Over 1,000 bootstrap resamples the two selectors differ in 987, $\theta^\star = 12{,}600$ in 984, and the penalty has median $+8.4\%$ with 95% interval $[+7.0\%, +10.7\%]$. The separation is not specific to gradient-boosted trees: a k-NN family on the same batch minimizes $\sum_j C_j$ at complexity $367{,}310$ but J at $7{,}161$.

Which regimes need cost-awareness. The regime parameter decides in advance which predictors need cost-aware selection. Using each family's measured prediction time, gradient-boosted trees, an MLP up to 1536×1536, and k-NN all keep κ below 0.5%, and the two selections do not separate; only when κ is raised to the calibrated scale of Figure 3(a)–(b) do they separate, at a 21–23% penalty. The latency share of the penalty also rises steeply as jobs get shorter: across job-length slices of the Azure batch, latency is under 1% of the penalty on the longest jobs and around 80% on the shortest, with the shortest slice above the longest in all 1,000 bootstrap resamples, reproducing the ATLAS-versus-Azure contrast inside a single trace. Appendix E.2 compares each class's floor $1 + \kappa$ on J/OPT with Round Robin.

5 Conclusion

We showed that in learning-augmented scheduling, inference latency contributes an unavoidable $\Theta(n\tau)$ penalty, that the optimal predictor complexity follows a power-law scaling with workload pressure, and that $\kappa = n\tau/\mathrm{OPT}$ identifies when cost-aware selection matters. Experiments on three production traces confirm these predictions; on the Alibaba microservices trace, latency alone makes the most accurate trained predictor the wrong one to deploy. The analysis assumes a single machine and batch arrivals; extending the lower bound to parallel machines and online arrivals remains open. The power-law scaling holds locally within workload types but not globally; how the regret exponent α varies across workloads is open.

AI USE STATEMENT

In this work, we used generative AI tools to polish the writing and to draft the experiment code. We have reviewed all AI-assisted work, and we take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Assumptions and complete proofs are given in Sections 2–3 and Appendix B, and the experimental protocol in Section 4.1 and Appendix D. All traces are publicly available. Code and a notebook reproducing the ATLAS and Azure experiments are available at the anonymous repository `https://anonymous.4open.science/r/experiment-2D93/`.

REFERENCES

Eric Balkanski, Tingting Ou, Clifford Stein, and Hao-Ting Wei. Scheduling with speed predictions. In *International Workshop on Approximation and Online Algorithms*, pp. 74–89. Springer, 2023.

Ziyad Benomar and Vianney Perchet. Advice querying under budget constraint for online algorithms. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 75026–75047. Curran Associates, Inc., 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/eda830e16044587b5082a853c4f25a90-Paper-Conference.pdf`.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Matteo Dell'Amico. Scheduling with inexact job sizes: The merits of shortest processing time first. *arXiv preprint arXiv:1907.04824*, 2019.

Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Algorithms with prediction portfolios. *Advances in neural information processing systems*, 35: 20273–20286, 2022.

Marina Drygala, Sai Ganesh Nagarajan, and Ola Svensson. Online algorithms with costly predictions. In *International Conference on Artificial Intelligence and Statistics*, pp. 8078–8101. PMLR, 2023.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Non-clairvoyant scheduling with predictions. *ACM Transactions on Parallel Computing*, 10(4):1–26, 2023.

Z. Jiang, T. Zhao, C. Xia, W. Li, and A. Y. Zomaya. The cost of accurate predictions in learning-augmented scheduling. In *2025 IEEE 31st International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 221–232, 2025. doi: 10.1109/RTCSA66114.2025.00031.

Zhiyun Jiang, Tianming Zhao, Chunqiu Xia, and Albert Y Zomaya. ATLAS: Alibaba dataset and benchmark for learning-augmented scheduling.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1859–1877. SIAM, 2020.

Alexander Lindermayr and Nicole Megow. Algorithms with predictions. *URL: https://algorithms-with-predictions. github. io*, 2022.

Alexander Lindermayr and Nicole Megow. Permutation predictions for non-clairvoyant scheduling. *ACM Transactions on Parallel Computing*, 12(2):1–26, 2025.

S. Luo, H. Xu, K. Ye, G. Xu, L. Zhang, G. Yang, and C. Xu. The power of prediction: Microservice auto scaling via workload learning. In *Proceedings of the 13th ACM Symposium on Cloud Computing (SoCC)*, pp. 355–369, 2022.

Michael Mitzenmacher. Scheduling with predictions and the price of misprediction. *arXiv preprint arXiv:1902.00732*, 2019.

Michael Mitzenmacher and Rana Shahout. Queueing, predictions, and large language models: Challenges and open problems. *Stochastic Systems*, 15(3):195–219, 2025.

Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *Communications of the ACM*, 65(7):33–35, 2022.

Rajeev Motwani, Steven Phillips, and Eric Torng. Nonclairvoyant scheduling. *Theoretical computer science*, 130(1):17–47, 1994.

Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. In *Advances in Neural Information Processing Systems*, 2018.

Linus Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16(3):687–690, 1968.

Rana Shahout and Michael Mitzenmacher. Skippredict: When to invest in predictions for scheduling. *Advances in Neural Information Processing Systems*, 37:105663–105701, 2024.

Rana Shahout, Eran Malach, Chunwei Liu, Weifan Jiang, Minlan Yu, and Michael Mitzenmacher. Don't stop me now: Embedding based scheduling for llms. *arXiv preprint arXiv:2410.01035*, 2024.

W SMITH. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3: 59–66, 1956.

Qizhen Weng, Wencong Xiao, Yinghao Yu, Wei Wang, Cheng Wang, Jian He, Yong Li, Liping Zhang, Wei Lin, and Yu Ding. {MLaaS} in the wild: Workload analysis and scheduling in {Large-Scale} heterogeneous {GPU} clusters. In *19th USENIX symposium on networked systems design and implementation (NSDI 22)*, pp. 945–960, 2022.

Yanqi Zhang, Íñigo Goiri, Gohar Irfan Chaudhry, Rodrigo Fonseca, Sameh Elnikety, Christina Delimitrou, and Ricardo Bianchini. Faster and cheaper serverless computing on harvested resources. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, pp. 724–739, 2021.

A NOTATION

Table 2: Summary of notation.

Symbol	Meaning
Scheduling instance and objective	
I	Scheduling instance
$\mathcal{I}$	Class of scheduling instances
n	Number of jobs
j	Job index
r_j	Release time of job j
p_j^*	True processing time of job j
$\widehat{p}_j$	Predicted processing time of job j
$\widehat{\mathbf{p}}$	Vector of job-size predictions
C_j	Completion time of job j
$\sum_j C_j$	Total completion time objective
$\mathrm{OPT}(I)$	Clairvoyant optimal total completion time on instance I
Scheduler interface	
$\mathcal{A}$	Learning-augmented downstream scheduler
$\widehat{\mathbf{z}}$	Generic prediction object supplied to $\mathcal{A}$; in the main setting, $\widehat{\mathbf{z}} = \widehat{\mathbf{p}}$
$\mathrm{ALG}_{\mathcal{A}}(I, \widehat{\mathbf{z}})$	Total completion time achieved by $\mathcal{A}$ on instance I with prediction object $\widehat{\mathbf{z}}$
$\eta_{\mathcal{A}}(I, \widehat{\mathbf{z}})$	Normalized error argument entering scheduler $\mathcal{A}$'s competitive guarantee
$\rho_{\mathcal{A}}$	Non-decreasing competitive-ratio upper envelope for scheduler $\mathcal{A}$
ρ	Response curve $\rho_{\mathcal{A}}$ when $\mathcal{A}$ is clear
ρ_0	Perfect-prediction baseline, $\rho(0) = \rho_0 \geq 1$
Predictor family	
θ	Predictor complexity parameter
Θ	Complexity domain, $\Theta = [\theta_{\min}, \infty)$
$\theta_{\min}$	Simplest available predictor complexity
$\theta^\star$	Cost-aware optimal predictor complexity
$\tau(\theta)$	Total batch inference latency of predictor θ; dependence on fixed-instance parameters such as n may be suppressed
$\eta_I(\theta)$	Normalized scheduler error induced by predictor θ on instance I
$\eta(\theta)$	Shorthand for $\eta_I(\theta)$ when I and $\mathcal{A}$ are fixed
$\eta_{\min}$	Irreducible normalized prediction error
Cost-aware objective and power-law model	
$J(\theta)$	Cost-aware upper-bound objective, $n\tau(\theta) + \mathrm{OPT}(I)\rho(\eta(\theta))$
c_τ	Latency scaling constant in $\tau(\theta) = c_\tau \theta^b$
b	Latency growth exponent
A	Reducible-error scaling constant in $\eta(\theta) = \eta_{\min} + A\theta^{-a}$
a	Error decay exponent
c_ρ	Slope constant in the linear response upper envelope $\rho(\eta) = \rho_0 + c_\rho \eta$
Γ	Scaling constant in Corollary 2, $\Gamma = \left(\frac{c_\rho A a}{c_\tau b}\right)^{1/(a+b)}$
Latency lower bound	
s_j	Service given to job j before predictions arrive in Theorem 2
$s_{(k)}$	kth smallest service amount among $s_1, \ldots, s_n$
ε	Small positive perturbation used in lower-bound constructions and toy examples
Price of ignoring prediction cost	
θ_s	Simple fast predictor in Theorem 4
θ_c	Complex slow predictor in Theorem 4
p	Common processing time of identical jobs in Theorem 4
γ	Per-job inference cost in Theorem 4, giving batch latency $\tau(\theta_c) = \gamma n$
$\kappa(\theta)$	Regime parameter, $\kappa(\theta) = n\tau(\theta)/\mathrm{OPT}(I)$
OPT estimation and extensions	
$\widehat{\mathrm{OPT}}$	Estimate of $\mathrm{OPT}(I)$ used for plug-in predictor selection
$\hat{\theta}$	Plug-in predictor complexity chosen using $\widehat{\mathrm{OPT}}$
ϵ	Relative OPT-estimation error, $\widehat{\mathrm{OPT}} = (1+\epsilon)\mathrm{OPT}(I)$

Continued on next page

Table 2: Summary of notation continued.

Symbol	Meaning
I^{τ}	Online-arrival instance obtained by shifting release dates by $\tau(\theta)$

B Deferred Proofs

B.1 Proof of Theorem 1

Proof. Let $\widetilde{C}_j$ be the completion time measured from the moment predictions become available. Then $C_j^{\text{wait}} = \tau(\theta) + \widetilde{C}_j$ for each job, so summing over all n jobs gives $\sum_j C_j^{\text{wait}} = n\tau(\theta) + \sum_j \widetilde{C}_j$. Since I is a batch instance, the optimal relative completion cost after time $\tau(\theta)$ is still $\mathrm{OPT}(I)$. By the definition of ρ in Section 2, the scheduler interface gives $\sum_j \widetilde{C}_j \le \rho(\eta(\theta))\mathrm{OPT}(I)$, proving the upper bound. For the lower bound, the post-wait schedule is feasible for I, so $\sum_j \widetilde{C}_j \ge \mathrm{OPT}(I)$. □

B.2 Proof of Theorem 3

Proof. The latency term $n\tau(\theta)$ is convex by Assumption 1. For the accuracy term, ρ is convex and non-decreasing while η is convex, so $\rho \circ \eta$ is convex by the standard composition rule (Boyd & Vandenberghe, 2004, Sec. 3.2.4). Strict convexity of η and the strict increase of ρ on $\eta(\Theta)$ imply that $\rho \circ \eta$ is strictly convex. Thus J is strictly convex. Since $\tau(\theta) \to \infty$ as $\theta \to \infty$, we have $J(\theta) \to \infty$, so a minimizer exists on $\Theta = [\theta_{\min}, \infty)$. Strict convexity gives uniqueness, and the boundary/interior alternatives follow from first-order optimality. For an interior minimizer, differentiating J gives the stated condition. □

B.3 Proof of Corollary 2

Proof. Substituting the parametric forms $\tau(\theta) = c_\tau \theta^b$ and $\eta(\theta) = \eta_{\min} + A\theta^{-a}$, together with $\rho(\eta) = \rho_0 + c_\rho \eta$, into $J(\theta) = n\tau(\theta) + \mathrm{OPT}(I)\rho(\eta(\theta))$ gives $J(\theta) = nc_\tau\theta^b + \mathrm{OPT}(I)\big(\rho_0 + c_\rho\eta_{\min} + c_\rho A\theta^{-a}\big)$. Note that ρ_0 and $\eta_{\min}$ enter only through the constant term $\mathrm{OPT}(I)(\rho_0 + c_\rho\eta_{\min})$ and therefore play no role in the optimality condition. Setting $J'(\theta) = 0$ gives $nc_\tau b\,\theta^{b-1} = \mathrm{OPT}(I)c_\rho A a\,\theta^{-a-1}$; rearranging yields $\theta^{a+b} = \mathrm{OPT}(I)c_\rho Aa/(nc_\tau b)$. Since J is strictly convex (Theorem 3), this is the unique minimum; $\max\{\cdot, \theta_{\min}\}$ enforces the constraint $\theta \ge \theta_{\min}$. □

B.4 Proof of Theorem 4

Proof. Under θ_s, no inference delay is paid. Since all jobs have the same true processing time, any non-idling one-at-a-time completion order is optimal, so $\sum_{j=1}^n C_j(\theta_s) = p\sum_{j=1}^n j = \frac{pn(n+1)}{2}$. Under θ_c, every job waits γn before scheduling begins, and the post-prediction schedule is again optimal for identical jobs. Thus $\sum_{j=1}^n C_j(\theta_c) = n\gamma n + \frac{pn(n+1)}{2} = \gamma n^2 + \frac{pn(n+1)}{2}$. Dividing gives a ratio of $1 + \frac{2\gamma n}{p(n+1)}$, which tends to $1 + \frac{2\gamma}{p}$ as $n \to \infty$. □

B.5 Proof of Proposition 1

Proof. Under the conditions of Corollary 2, the cost-aware upper-bound objective can be written as

$$J(\theta) = C + G(\theta), \qquad C = \mathrm{OPT}(I)(\rho_0 + c_\rho\eta_{\min}) \ge 0,$$

where

$$G(\theta) = nc_\tau\theta^b + \mathrm{OPT}(I)c_\rho A\theta^{-a} =: L\theta^b + M\theta^{-a}.$$

The interior first-order condition $Lb\,\theta^{b-1} = Ma\,\theta^{-a-1}$ implies

$$L\theta^{\star b} = \frac{a}{a+b}G(\theta^\star), \qquad M\theta^{\star -a} = \frac{b}{a+b}G(\theta^\star).$$

Since

$$\theta^\star \propto \mathrm{OPT}(I)^{1/(a+b)},$$

replacing $\mathrm{OPT}(I)$ by $\widehat{\mathrm{OPT}} = (1+\epsilon)\mathrm{OPT}(I)$ gives

$$\hat{\theta} = \theta^\star q, \qquad q = (1+\epsilon)^{1/(a+b)}.$$

Therefore

$$\begin{aligned}\frac{G(\hat{\theta})}{G(\theta^\star)} &= \frac{L(\theta^\star q)^b + M(\theta^\star q)^{-a}}{G(\theta^\star)} \\ &= \frac{a}{a+b}q^b + \frac{b}{a+b}q^{-a}.\end{aligned}$$

Define

$$f(\epsilon) = \frac{a}{a+b}(1+\epsilon)^{b/(a+b)} + \frac{b}{a+b}(1+\epsilon)^{-a/(a+b)}.$$

Then $f(0) = 1$ and $f'(0) = 0$. A Taylor expansion at $\epsilon = 0$ gives

$$f(\epsilon) = 1 + \frac{ab}{2(a+b)^2}\epsilon^2 + O(|\epsilon|^3).$$

Thus

$$\frac{G(\hat{\theta})}{G(\theta^\star)} = 1 + \frac{ab}{2(a+b)^2}\epsilon^2 + O(|\epsilon|^3).$$

Finally, because $C \geq 0$ and $G(\hat{\theta}) \geq G(\theta^\star)$,

$$\frac{J(\hat{\theta})}{J(\theta^\star)} = \frac{C + G(\hat{\theta})}{C + G(\theta^\star)} \leq \frac{G(\hat{\theta})}{G(\theta^\star)}.$$

Hence

$$\frac{J(\hat{\theta})}{J(\theta^\star)} \leq 1 + \frac{ab}{2(a+b)^2}\epsilon^2 + O(|\epsilon|^3).$$

Since $ab/(a+b)^2 \leq 1/4$, the coefficient is at most $1/8$, proving the stated bound. □

C Deferred Theory

C.1 Sensitivity to OPT Estimation

The cost-aware optimum $\theta^\star$ depends on $\mathrm{OPT}(I)$, unknown before execution. Replacing $\mathrm{OPT}(I)$ with an estimate $\widehat{\mathrm{OPT}}$, we show that the resulting sub-optimality degrades with estimation error.

Proposition 1 (Sensitivity to estimated OPT)**.** *Under the conditions of Corollary 2, assume $\theta^\star$ and $\hat{\theta}$ are both interior. If $\widehat{\mathrm{OPT}} = (1+\epsilon)\mathrm{OPT}(I)$ with $|\epsilon| \leq 1/2$, the plug-in optimum $\hat{\theta}$ satisfies*

$$\frac{J(\hat{\theta})}{J(\theta^\star)} \leq 1 + \frac{ab}{2(a+b)^2}\,\epsilon^2 + O(|\epsilon|^3).$$

Since $ab/(a+b)^2 \leq 1/4$, the relative excess is at most $\epsilon^2/8 + O(|\epsilon|^3)$.

The proof (Appendix B.5) uses $\hat{\theta} = \theta^\star(1+\epsilon)^{1/(a+b)}$ and optimality of $\theta^\star$ to show that the objective excess is second-order. The exponent $1/(a+b)$ implies that $\hat{\theta}$ is a damped function of ϵ: when $a+b$ is large, even substantial estimation errors produce small shifts in complexity. Cost is therefore robust to moderate OPT misestimation. The bound depends on ϵ, not n, $\mathrm{OPT}(I)$, or the scaling constants.

C.2 Extensions

The batch single-machine model isolates the prediction-latency mechanism cleanly, but real systems face online arrivals and parallel machines. We discuss how the accounting changes in these settings.

Online arrivals. If job j arrives at time r_j and its prediction requires $\tau(\theta)$ from arrival, a prediction-gated policy cannot schedule j before $r_j + \tau(\theta)$. Since $\tau(\theta)$ depends only on θ, not on j, this is a uniform shift of the release dates. The clairvoyant optimum of the shifted instance therefore satisfies

$$\mathrm{OPT}(I^\tau) = \mathrm{OPT}(I) + n\tau(\theta),$$

so the $n\tau$ accounting persists at the benchmark level. However, predictions now become available at different calendar times, and latency may interact with release-time structure, idleness, and online decisions. The exact batch bound of Theorem 1 thus does not transfer verbatim: a scheduler guarantee for the shifted online instance is naturally stated relative to $\mathrm{OPT}(I^\tau)$, rather than as $n\tau(\theta) + \rho(\eta)\,\mathrm{OPT}(I)$. The cost-aware selection principle, balancing latency against accuracy, continues to apply. Characterizing the tradeoff under general release times is left to future work.

Parallel machines. Under wait-then-schedule on m identical machines, the additive $n\tau(\theta)$ term is unchanged: all machines wait until predictions are available, so every job's completion time is shifted by $\tau(\theta)$, regardless of m. The cost-aware selection framework therefore carries over directly. Scheduling algorithms that overlap prediction with execution can reduce the effective latency cost (Shahout et al., 2024), but depart from the wait-then-schedule model analyzed here. Extending the adaptive-adversary lower bound of Theorem 2 to parallel machines requires a separate argument, since the single-machine proof uses both the unit-capacity constraint and the single-machine SPT/SRPT weight structure. Learning-augmented algorithms for parallel machines exist (Lindermayr & Megow, 2025), but none account for prediction cost. A comprehensive parallel-machine prediction cost study belongs to future research direction.

D Experimental Details

Workload characterization. Two workload properties are central to cost-aware predictor selection. First, both traces are highly right-skewed: the top 10% of jobs account for 70% of total compute on ATLAS and 96% on Azure, with coefficients of variation 3.2 and 5.5 and Gini coefficients 0.82 and 0.95, respectively. This skew increases the value of accurate ranking: a long job misranked as short can delay many shorter jobs and substantially increase $\sum_j C_j$, which appears in the framework through the scheduler response $\rho(\eta)$. Second, the absolute time scales are very different. The median duration is 434 seconds on ATLAS and 31 milliseconds on Azure, while the effective completion-time scale p_{eff} differs by more than $5000\times$. This scale difference creates the sharp contrast in κ.

Processing-time labels. Learning-augmented scheduling assumes a scalar p_j^* per job; most empirical work avoids defining it on real traces by using synthetic instances (Lindermayr & Megow, 2022; Mitzenmacher, 2019). On production traces, p_j^* must be aggregated from multi-task executions (Dell'Amico, 2019; Jiang et al.). For **ATLAS**, following Jiang et al., we set p_j^* to the job-level task span (earliest task start to latest task end) across the multi-task, multi-instance PAI job hierarchy (Weng et al., 2022). For **Azure Functions**, each invocation is atomic, so p_j^* is the recorded duration in seconds (Zhang et al., 2021). Non-positive durations are removed. These labels are observed durations from a multi-machine environment, shaped by gang scheduling, fragmentation, interference, and queuing, not intrinsic single-machine runtimes. This gap is inherent to *all* empirical LAS evaluation on production traces, since no public trace provides controlled single-tenant re-execution. The irreducible error floor $\eta_{\min}$ (Section 2) absorbs exactly this system-induced label noise, and $\theta^\star$ (Theorem 3) remains well-defined for any $\eta_{\min} \geq 0$.

Per-bucket scaling. Table 3 reports the per-bucket validation of Corollary 2 for the Layer 1 family.

E Additional Experimental Results

E.1 Trained LightGBM on Azure

Table 4 reports selected configurations of the 12-point LightGBM grid on the Azure held-out batch of 10,000 invocations, with all latencies measured. J is minimized at $\theta^\star = 76{,}200$, in the middle of the grid. The $+1.69\%$ penalty of the largest model has a 95% bootstrap interval of $[+1.05\%, +2.60\%]$ over 1,000 resamples, and $\theta^\star = 76{,}200$ in 902 of them. Scaling all measured latencies by a common

Algorithm 1 Cost-aware predictor evaluation

Require: Batch instance $I = \{p_j^*\}_{j=1}^n$, predictor family $\{\mathcal{M}_\theta\}_{\theta\in\Theta}$, latency multiplier λ
  $\mathrm{OPT}(I) \leftarrow \sum_{k=1}^n (n-k+1)p^*_{(k)}$
  for $\theta \in \Theta$ **do**
    Obtain predictions $\widehat{\mathbf{p}}_\theta$ and batch latency $\tau(\theta)$
    $S(\theta) \leftarrow \mathrm{SPJF}(I, \widehat{\mathbf{p}}_\theta)$ ▷ total completion time
    $J(\theta) \leftarrow n\lambda\tau(\theta) + S(\theta)$
  end for
  $\theta^\star \leftarrow \arg\min_\theta J(\theta)$ ▷ cost-aware
  $\theta_{\mathrm{ua}} \leftarrow \arg\min_\theta S(\theta)$ ▷ cost-unaware
  return $\theta^\star, \theta_{\mathrm{ua}},\ J(\theta_{\mathrm{ua}})/J(\theta^\star) - 1$

Table 3: Per-bucket scaling law validation (Layer 1). Values of $\theta_e^\star/\theta_t^\star$ near 1.0 indicate agreement with Corollary 2. Cell shading: ratio (darker = further from 1.0), α_{local} (darker = higher exponent).

	ATLAS					Azure				
Bucket	OPT $/n$	$\theta_e^\star$	α_l	$\theta_t^\star$	R	OPT $/n$	$\theta_e^\star$	α_l	$\theta_t^\star$	R
Short-Med	57 364	0.64	0.319	0.77	0.83	0.981	0.266	0.525	0.276	0.96
Medium	146 240	3.76	0.411	2.16	1.74	8.265	2.418	0.537	1.447	1.67
Mixed	371 322	9.09	0.627	6.46	1.41	92.836	9.094	0.772	8.896	1.02
Long-biased	1 278 201	21.99	0.551	12.64	1.74	352.6	21.99	0.709	18.91	1.16
Long	3 648 818	34.20	0.451	22.88	1.49	1 195	34.20	0.621	36.92	0.93
Very long	9 694 549	53.18	0.372	40.88	1.30	6 050	82.70	0.500	100.14	0.83

Summary. ATLAS: $\theta^\star = 34.2$, price $= 21\%$, $\alpha = 0.645$, slope $= 1.24/0.61$, median ratio $= 1.41$. Azure: $\theta^\star = 22.0$, price $= 23\%$, $\alpha = 0.749$, slope $= 0.61/0.57$, median ratio $= 1.02$. Here $R = \theta_e^\star/\theta_t^\star$.

factor, the cost-aware and cost-unaware choices first differ at 0.94 ms per job on Azure but only at 37.9 s per job on ATLAS. Retraining on ten different 90% subsamples of the training data selects 76,200 nine times and 153,000 once, interior every time, and the largest model is never selected. At the largest model the latency share is $\kappa = 2.4 \times 10^{-3}$ on Azure against 5.7×10^{-7} on ATLAS. Because $\sum_j C_j$ is also minimized at 76,200, cost-aware and cost-unaware selection coincide on this trace: the rise of J past $\theta^\star$ comes mainly from the scheduling term, which accounts for 93.9% of the largest model's penalty, as the model overfits: total completion time, log-residual spread, rank correlation, and ℓ_1 error on the held-out batch all degrade past $\theta^\star$. Overfitting departs from Assumption 2, which takes η decreasing. Suppose the realized error satisfies $\eta_{\mathrm{real}}(\theta) \geq \eta(\theta)$ and the excess $g(\theta) = \mathrm{OPT}(I)\big[\rho(\eta_{\mathrm{real}}(\theta)) - \rho(\eta(\theta))\big]$ is non-decreasing in θ, as when overfitting sets in beyond some complexity. Then $J_{\mathrm{real}} = J + g$, and for every $\theta > \theta^\star$ both $J(\theta) \geq J(\theta^\star)$ and $g(\theta) \geq g(\theta^\star)$, so J_{real} has a minimizer at most $\theta^\star$. The theoretical $\theta^\star$ is thus an upper bound on the complexity worth deploying. This is a single 10,000-job batch; while $\theta^\star$ is stable across resamples, the size of the degradation past it was not tested.

Table 4: Trained LightGBM on the Azure trace (selected grid points). $\theta^\star = 76{,}200$ is shaded.

θ	$n\tau$ (s)	$\sum_j C_j$ (s)	J (s)
40	28	8,914,321	8,914,349
800	111	8,179,454	8,179,565
6,400	567	7,749,830	7,750,397
25,200	2,001	7,387,556	7,389,558
76,200	7,317	7,319,392	7,326,708
153,000	10,536	7,424,304	7,434,840
204,000	14,927	7,435,898	7,450,825

E.2 PREDICTOR CLASSES ON A NORMALIZED AXIS

Figure 3(d) reports the latency side $\kappa = n\tau/\mathrm{OPT}$ of each predictor class. Dividing J by OPT gives $J/\mathrm{OPT} = \kappa + \sum_j C_j/\mathrm{OPT}$, where the second term is at least 1 and equals 1 only under perfect scheduling. The floor $1 + \kappa$ is therefore the best a class can reach. Round Robin needs no predictions and is 2-competitive (Motwani et al., 1994), so a class whose floor exceeds 2 is dominated by Round Robin even with perfect predictions. Table 5 lists the floors on the Azure batch. The floor settles only the LLM-router row; the small-NN and transformer rows use representative per-job latencies rather than trained models, and whether such a class beats Round Robin in practice requires its measured $\sum_j C_j$, as in Table 1.

Table 5: Floor $1 + \kappa$ of each predictor class on the Azure batch, against the Round Robin bound of 2.

Predictor class	κ	$1 + \kappa$	vs. Round Robin (2)
GBT, measured	0.12%	1.001	below
Small NN, 1 ms/job	1.6%	1.016	below
Transformer, 10 ms/job	15.9%	1.159	below
LLM router, 100 ms/job	158.9%	2.589	above (dominated)