

Figure 1: **The problem, the lever, and what we measured.** (a) Taps arrive at a shared pool; each owes a clip within a hard deadline D_r, past which extra compute has *negative* value. (b) The premise this paper tests: if quality per GPU-second were content-dependent, one fixed configuration would be simultaneously too generous and too stingy. (c) It is not. A single static configuration delivers the most quality, the full allocator loses to it in every cell of the sweep, and an oracle given the true curves clears it only barely and only by spending several times more. On this backend there is nothing left for a per-request allocator to allocate.

AN EMPIRICAL CEILING ON PER-REQUEST COMPUTE ALLOCATION IN DIFFUSION SERVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Interactive generative products run under a hard latency constraint: a tap must return a rendered clip within a service-level objective or the interaction breaks. This makes compute an allocation problem—spend more on requests whose quality still climbs, less on those that have saturated—and a growing body of harness and serving work assumes such allocation is worth doing. Whether it is worth doing is an empirical question about the backend. We measure it.

We formalise budget-aware generative orchestration as a deadline-filtered multiple-choice knapsack over per-request quality–compute curves and build a complete reference allocator—online curve estimation, marginal-return allocation, admission control, mid-flight revision, speculative prefetch—not as a system to advocate but as the instrument the measurement requires. Against 7,200 profiled SDXL renders and numeric gates frozen before the first render, we report a *ceiling*: an oracle given the true per-request curves, an upper bound no online estimator can pass, beats the best static configuration by $+0.0118$ $[+0.0090, +0.0146]$ delivered quality while spending $3.5\times$ its compute. Per-request allocation is not merely hard to realise here; there is almost nothing for it to win. The realised system behaves as that bound predicts: it is beaten by a single static configuration in all 45 cells of the sweep, by the *same* configuration in every cell, and in 30 cells a static setting strictly dominates it—more quality and less compute. Cost, by contrast, is separable in the knobs to 0.6% held-out error, so one profiling pass prices any configuration. A pre-registered human study (259 pairs) shows the quality proxy is calibrated, making these facts about the backend rather than the score. The mechanism is distillation: a few-step sampler collapses the useful cost axis to a handful of non-dominated rungs. We give the conditions under which the ceiling should rise, and release the measured tables so the bound can be recomputed on any backend before an allocator is built.

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.

1 INTRODUCTION

Generative video is moving from an offline, best-effort service into the interaction loop of consumer products. In an interactive story or content app, a user taps a choice and expects a freshly generated clip to appear *now*; a creator previews an edit and expects it rendered before attention drifts. In these settings latency is not a quality dimension to be traded off—it is a *hard constraint*. A clip that is one point better on an aesthetic metric but arrives two seconds late has failed the interaction. This inverts the usual research framing, where more compute is unambiguously good: here, past the deadline, additional compute has negative value.

At the same time, the compute footprint of a single generation request is now a tunable, high-dynamic-range quantity. The number of denoising steps (Ho et al., 2020; Song et al., 2021), the numerical precision, the spatial resolution, and the choice of a distilled few-step sampler (Salimans & Ho, 2022; Song et al., 2023; Luo et al., 2023; Sauer et al., 2023) each move the GPU cost of a request by large factors, and each trades against output quality along a curve that is neither linear nor uniform across requests. Some prompts are easy and saturate quality after a handful of steps; others need the full budget. The result is a rich, per-request *quality–compute* surface—and, on a shared GPU pool serving many concurrent interactive requests, a genuine allocation problem: how should a fixed compute budget be spread across requests, under per-request deadlines, to maximize the quality users actually receive?

The gap. The systems that today sit between an application and a diffusion backend—generation *harnesses* and *agents*—do not answer this question. Workflow copilots, self-evolving repair agents and workflow benchmarks all optimise *quality* or automate *construction*; none treats the compute budget as a first-class, constrained resource allocated across requests under a hard latency SLO. The scheduling literature, conversely, has deep results on deadline-driven scheduling and latency-SLO inference serving, but those systems schedule fixed-cost, fixed-quality jobs; they do not exploit the fact that a generation request's cost *and* quality can be dialled continuously, per request, on one model. Budget-aware *generative* orchestration sits in the gap, and Section 2 places it against both lines in detail.

This paper. We formalise the problem and build BACH (Budget-Aware Compute Harness), a complete reference allocator: an online estimator of each request's quality–compute curve, a marginal-return allocator that solves the deadline-filtered knapsack, admission control for the tail it cannot serve, mid-flight revision of allocations already dispatched, and speculative prefetch of likely next branches. We build it not to advocate it but because the instrument is a prerequisite: one cannot bound what allocation is worth without an allocator that actually allocates. Every threshold was frozen before the first render.

What we establish.

- **The cost side is easy, the quality side is not.** Cost is separable, $c(x, a) \approx \gamma(x)\phi(a)$, to 0.6% held-out error, so one profiling pass prices any configuration. Quality curves do not fit a saturating family, do not separate by content stratum, and their difficulty parameters are *less* predictable from cheap features than a global constant.
- **A ceiling on what adaptation can buy.** Handing the allocator the *true* curves—an upper bound no estimator can pass—beats the best static configuration by $+0.0118$ $[+0.0090, +0.0146]$ while spending $3.5\times$ the compute. This is the paper's central number: the deficit below is not the estimator's fault and a better one would not repair it.
- **The realised consequence.** BACH is beaten by a single static configuration in all 45 cells—the *same* one (`fixed-4@768`) every time, so the workload does not even call for switching—and in 30 cells a static setting strictly dominates it. Every ablation is indistinguishable from the full system, which is what a ceiling this low predicts.
- **The mechanism, and where it lifts.** Distillation collapses the cost axis to a handful of non-dominated rungs whose best is nearly the same for every request. Section 6 gives the conditions under which the ceiling should rise; the released tables let a reader recompute the bound on their own backend before building anything.

A null invites two rebuttals and we close both. That the metric produced it: a pre-registered human study (259 pairs) shows the proxy is calibrated, agreement rising monotonically to 94.3% where the metric reports a large gap. That the baselines were weak: our own first baseline set put every static comparator on the expensive side of the axis that turns out to matter, and replacing it is what produced the result above rather than a flattering one.

Contributions. Beyond the findings above: a formalisation of budget-aware generative orchestration as a deadline-filtered multiple-choice knapsack with an allocator exact on the concave relaxation (Sections 3–4); an evaluation protocol for claims of this kind—frozen gates, a baseline set spanning the axis under test, an oracle arm separating design space from implementation, and a human validation of the proxy; and released artifacts—cost and quality tables, human labels, and the code that regenerates every number here.

2 Related Work

Generation harnesses and agents. The tools between an application and a diffusion backend optimise quality or automate construction, not cost under a constraint: ComfyUI-Copilot helps a human *author* a workflow (Xu et al., 2025), ComfyClaw evolves a skill library that repairs and refines them (Li et al., 2026), and ComfyBench measures whether agents can *construct* valid ones at all (Xue et al., 2024). Cost-aware LLM-agent orchestration (Yao et al., 2023; Schick et al., 2023; Wu et al., 2023) structures *which* calls to make. None takes an SLO and a shared budget as inputs and allocates across concurrent requests. Bach wraps whatever backend such a tool produces and decides only how much compute each invocation may spend.

The knobs Bach allocates. A large body of work reduces the cost of a *single* diffusion request: faster solvers (Song et al., 2021; Lu et al., 2022), progressive, consistency and adversarial distillation (Salimans & Ho, 2022; Song et al., 2023; Luo et al., 2023; Sauer et al., 2023), and training-free reuse of intermediate features (Ma et al., 2024; Bolya & Hoffman, 2023), atop latent and high-resolution backbones (Ho et al., 2020; Rombach et al., 2022; Podell et al., 2023; Esser et al., 2024) and their video counterparts (Ho et al., 2022; Blattmann et al., 2023). These *create* the action space: they reshape a request's quality–compute curve but do not decide, under a budget, which setting each request gets.

Trading quality for resources. Real-time scheduling handles hard deadlines (Liu & Layland, 1973); *anytime* algorithms formalise trading computation for solution quality (Zilberstein, 1996; Boddy & Dean, 1994); and approximate computing exposes accuracy knobs to a runtime that meets a resource target by degrading output in a controlled way (Baek & Chilimbi, 2010; Hoffmann et al., 2011; Sidiroglou-Douskos et al., 2011). The closest systems analogue is video analytics and streaming, where a knob vector is chosen per stream: VideoStorm schedules over a resource–quality profile (Zhang et al., 2017), Chameleon amortises re-profiling across correlated cameras (Jiang et al., 2018), AWStream learns a Pareto-optimal degradation policy (Zhang et al., 2018a), and adaptive bitrate streaming picks a quality level per chunk under a stall deadline (Yin et al., 2015; Spiteri et al., 2016; Mao et al., 2017). All assume what our measurement questions: that the quality bought by extra resources is knowable in advance. There it is, because the profile amortises over a stationary stream; a generation request is seen once. Speculative execution is the classical way to spend spare budget on futures (Leviathan et al., 2023; Chen et al., 2023), and interactive generative environments (Bruce et al., 2024) make the branching structure that invites it common.

Latency-SLO inference serving and GPU scheduling. Serving systems schedule inference under latency targets: latency-aware batching (Crankshaw et al., 2017), predictable execution (Gujarati et al., 2020), model-variant selection (Romero et al., 2021), preemptive serving (Zhang et al., 2023), statistical multiplexing (Li et al., 2023), and LLM-serving advances in batching, memory and scheduling (Yu et al., 2022a; Kwon et al., 2023; Agrawal et al., 2024; Zhong et al., 2024). INFaaS is the nearest neighbour: it selects among model variants to meet an SLO. Here the "variant" is a per-request allocation over steps, precision and resolution on the *same* model, whose quality is content-dependent and estimated online. Cluster schedulers for training workloads (Xiao et al., 2018; Gu et al., 2019; Narayanan et al., 2020; Mahajan et al., 2020; Qiao et al., 2021; Yu & Chowdhury,

2020) allocate *GPUs to jobs*; we allocate *compute within a job* whose output quality is the thing being traded.

Positioning. BACH brings the deadline-aware discipline of real-time scheduling to the generation-harness layer that the ComfyUI-agent line has optimised for quality alone. The allocator is the classical greedy for separable concave utilities under a knapsack constraint (Nemhauser et al., 1978; Khuller et al., 1999; Sviridenko, 2004; Kellerer et al., 2004); we contribute no new optimiser, but a measurement: whether the objects such an allocator needs exist in usable form on a modern backend.

3 PROBLEM FORMULATION

Requests and knobs. A request r arrives with a context x_r (prompt, conditioning image, interaction metadata) and a hard deadline D_r. A backend exposes a finite set of *configurations* $\mathcal{A} = \{a_1, \ldots, a_M\}$, each a joint setting of denoising steps s, numerical precision p, spatial resolution ℓ, and choice of (possibly distilled) sampler. Running r under a costs

$$c(x_r, a) \;=\; \text{GPU-seconds to render } r \text{ at } a, \tag{1}$$

which we measure by direct profiling (Section 5) and model as approximately separable in the knobs, $c(x_r, a) \approx \gamma(x_r)\,\phi(a)$, with a content factor γ and a configuration factor ϕ. Delivering r under a yields quality

$$q(x_r, a) \in [0, 1], \tag{2}$$

a normalized score (Section 5 specifies the metric). Neither c nor q is known exactly before running; BACH works with estimates $\hat{c}, \hat{q}$ (Section 4).

The quality–compute curve. Ordering configurations by cost traces $Q_r(b) = \max_{a:\, c(x_r,a)\le b} q(x_r, a)$, the best quality achievable for r within budget b. We assume Q_r is non-decreasing with diminishing returns, so the marginal gain $Q_r'(b)$ eventually falls below any threshold. Crucially the shape is assumed to depend on content: an "easy" request saturates early, a "hard" one keeps improving. Estimating that shape from cheap features of x_r is Hypothesis H1 — and, as Section 5 reports, the assumption that the shapes differ enough to matter is the one that fails.

Static allocation problem. Consider a batch of N requests sharing a compute budget B (GPU-seconds available within the scheduling horizon, e.g. set by the deadline and the number of GPUs). Choosing one configuration $a_r \in \mathcal{A}$ per request to maximize total quality is a multiple-choice knapsack:

$$\max_{\{a_r\}} \sum_{r=1}^{N} q(x_r, a_r) \quad \text{s.t.} \quad \sum_{r=1}^{N} c(x_r, a_r) \le B, \tag{3}$$

with the per-request deadline entering as a feasibility filter: a configuration is *admissible* for r only if its estimated finish time respects D_r,

$$\mathcal{A}_r \;=\; \{\, a \in \mathcal{A} : t_{\text{start}}(r) + \hat{c}(x_r, a)/G \le D_r \,\}, \tag{4}$$

where G is the effective per-request GPU throughput given batching and concurrency. Equations (3)–(4) are the static core: spend the shared budget where marginal quality per GPU-second is highest, subject to each request's deadline.

Objective and SLO. A hard deadline makes late delivery worthless, so we do not optimise mean quality. The *delivered* quality of r is

$$\tilde{q}(r) = q(x_r, a_r) \text{ if finished by } D_r, \quad 0 \text{ otherwise}, \tag{5}$$

and the SLO attainment rate is $\alpha = \Pr[\text{finished by } D_r]$. The orchestrator maximises $\mathbb{E}_r[\tilde{q}(r)]$, which couples two goals a quality-only harness separates: raise per-request quality *and* keep the miss rate low. This is why over-provisioning is not optimal under load—it inflates $\hat{c}$, shrinks the admissible set in Eq. (4), and drives misses that zero out quality.

Interactive extension. In a session the next request is not independent of the current one: after delivering r the user picks from a small branch set with estimable probabilities π_j, and *perceived* latency for the chosen branch is zero if it was already rendered. Given spare budget B_{idle} between user actions, speculative prefetch chooses a subset of branches maximising expected hidden latency $\sum_j \pi_j \Delta_j$ subject to $\sum_j c(x_{r^{(j)}}, a_{r^{(j)}}) \le B_{\text{idle}}$, where Δ_j is the latency pre-rendering would hide. This is a budgeted maximum-coverage problem over branches and is Hypothesis H3.

4 METHOD

BACH has three components, one per hypothesis: an online estimator of the quality–compute curve (H1), a budget-aware allocator solving Eq. (3) under deadlines (H2), and a branch-prefetch policy for interactive load (H3).

4.1 ONLINE QUALITY–COMPUTE CURVE ESTIMATION

Cost and curve. The configuration factor $\phi(a)$ in Eq. (1) is profiled once per backend; the content factor $\gamma(x_r)$ is predicted from cheap request features, giving $\hat{c}(x_r, a) = \hat{\gamma}(x_r)\phi(a)$ before any render. Estimating Q_r is the crux of H1. Rather than assume a universal curve we predict a small set of parameters of a monotone saturating family

$$\hat{Q}_r(b) \;=\; q_{\max}(x_r)\left(1 - e^{-\,b/\beta(x_r)}\right), \tag{6}$$

where $q_{\max}$ is the achievable ceiling and β a per-request *difficulty scale* (small β = saturates early = easy). A lightweight regressor maps request features to $(\hat{q}_{\max}, \hat{\beta})$, trained offline on (request, configuration, measured quality) triples and refreshed online. The allocator ranks on the marginal return $\hat{Q}'_r(b) = (\hat{q}_{\max}/\hat{\beta})e^{-b/\hat{\beta}}$. The family is deliberately simple: the scheduler needs the *shape*—where returns flatten—not calibrated absolute quality, and a mis-specified ceiling cancels in the marginal ranking.

Online correction. Predicted curves drift as the content mix shifts. Each completed render reveals one point (b, q) on the true curve—the realised cost and a fast quality proxy—and the regressor takes an exponentially-weighted online step on it, so $\hat{\beta}$ tracks the current workload rather than the offline corpus.

4.2 BUDGET-AWARE ALLOCATION UNDER DEADLINES

Given the pending set, their estimated curves and a shared budget B for the horizon, the allocator solves Eq. (3) restricted to the deadline-admissible sets of Eq. (4). Exact multiple-choice knapsack is NP-hard but admits a fast, near-optimal *marginal-return greedy*: seed every request at its cheapest admissible configuration, then repeatedly spend the next GPU-second on whichever upgrade yields the highest quality gain per unit cost,

$$\text{ratio}(r) \;=\; \frac{\hat{q}(x_r, a_r^{+}) - \hat{q}(x_r, a_r)}{\hat{c}(x_r, a_r^{+}) - \hat{c}(x_r, a_r)}, \tag{7}$$

until the budget is exhausted (Algorithm 2, Appendix C). Because each $\hat{Q}_r$ is concave, per-request upgrade ratios decrease, so a priority queue keyed on Eq. (7) gives an $O(NM \log N)$ schedule that is optimal in the continuous relaxation and within one item of the integral optimum (Appendix C). Measured ladders need not be concave—a distilled few-step sampler can dominate a multistep one at equal cost—so the allocator operates on their upper concave envelope, which restores the ordering the greedy needs and upper-bounds achievable quality.

Admission control. Under overload even the cheapest admissible configuration may not fit; forcing such requests in pushes others past their deadlines and zeroes their quality (Eq. (5)). BACH admits in deadline order and *sheds* the tail that cannot be served rather than uniformly starving the batch, maximising delivered quality $\mathbb{E}[\tilde{q}]$ rather than nominal quality: it converts what a fixed policy spends on doomed-late renders into on-time deliveries.

4.3 Two Further Mechanisms

Two mechanisms complete the harness. Both are brief here because, as Section 5 reports, neither earns its place on this backend; specifications are in Appendix D.

Mid-flight budget revision. Diffusion makes a dispatch-time decision revocable at no loss: a request that has consumed b GPU-seconds holds a partially denoised latent, so downgrading means stopping early and upgrading means continuing. Revision is pressure-driven: the trigger is that some queued request will miss its deadline even at its cheapest configuration, at which point BACH reclaims compute from the in-flight requests whose remaining spend buys the least quality per GPU-second. Upgrades are strictly opportunistic.

Speculative branch prefetch. Between user actions the pool is often idle. BACH spends that idle budget on the branches most likely to be requested next, ranking candidates by expected hidden latency per GPU-second (Algorithm 4). Prefetch never touches budget reserved for deadline-bound work, so its only cost is compute burned on branches the user never picks.

Putting it together. At runtime BACH loops over Algorithms 1–4, updating the estimators from every completion. The harness is model-agnostic: it wraps an existing backend workflow and changes only *how much* compute each invocation receives.

5 Experiments

This section states the experimental design, the frozen measurement protocol, the pre-registered decision rules, and what the measurements returned. Every quantitative cell in this paper is emitted by the released analysis code from result files; **no number is typed by hand**. One gate, G8, required a production trace that is an external dependency; it was not obtained, the gate is marked *not run*, and the paper's conclusions are scoped to the measured backend under a synthetic workload throughout.

Pre-registration. Before any render existed we froze the quality metric and its normalisation, the difficulty strata, the baseline list, the sweep grid, the statistical protocol, and numeric GO/NO-GO thresholds (Appendix A). Two commitments do most of the work. The *headline operating point*—offered load ρ=1.2, SLO D=5 s, pool G=4—was declared in advance, so the reported gain cannot be selected after the fact from a grid of load/SLO combinations. And quality is normalised by *global* corpus-wide quantiles rather than per request: per-request rescaling would force $q(a^{\text{ref}}) \equiv 1$ and $q(a^{\min}) \equiv 0$ by construction, making H1's claim that the ceiling $q_{\max}$ varies with content true by definition rather than by measurement.

Backend, action space, hardware. A latent diffusion image pipeline: SDXL (Podell et al., 2023) with a distilled few-step sampler (Sauer et al., 2023) for $s \leq 8$ and a multistep solver (Lu et al., 2022) above it. The frozen action space crosses steps $s \in \{1, \ldots, 50\}$, precision, and resolution $\ell \in \{512, 768, 1024\}$: $|\mathcal{A}| = 60$. All measurements come from one 24 GB slice of an NVIDIA L40S. The precision arm is `fp16` versus `bf16`: the hardware supports `fp8` but no quantisation toolchain for it was available, and the pre-registered protocol requires falling back rather than presenting an `fp8` arm that silently runs at another precision. A request is only ever scheduled at a configuration we actually measured for it, so no cost or quality is ever interpolated.

Corpus, strata, metric. Requests are 300 prompts sampled from PARTIPROMPTS (Yu et al., 2022b) proportionally across its challenge levels; those labels are used only to spread the sample and play no part in our strata. Difficulty strata come from pre-render textual complexity alone—noun-phrase count (regular-expression heuristic), token count, distinct content-word count—summed as z-scores and split at rank terciles into exactly 100 requests each, and written to disk before any image existed. Quality is a human-preference reward model (Kirstain et al., 2023) mapped into $[0, 1]$ by fixed global quantiles. Secondary metrics (Xu et al., 2023; Zhang et al., 2018b; Hessel et al., 2021) serve only as appendix robustness checks; corpus-level scores such as FID (Heusel et al., 2017) are unusable here because the allocator needs a *per-request* quality comparable across configurations of one content. Because everything rests on a proxy for human preference, we validate it directly on 200 same-content

Gate	Quantity	Pre-registered threshold	Measured	Verdict
G1	median per-request curve fit quality	≥ 0.85	-0.000	FAIL
G2	difficulty-scale ratio, hard vs easy stratum	≥ 2.0	1.061	FAIL
G3	curve-parameter prediction gain over a global constant	≥ 0.2	-0.055	FAIL
G4	separable cost model held-out MAPE	≤ 0.15	0.006	PASS
G5	BACH minus best fixed, delivered quality	CI excludes 0	-0.017	FAIL
G6	fraction of the oracle gain captured	≥ 0.7	-1.479	FAIL
G7	perceived P50 reduction from prefetch	P50 ≥ 0.2 and waste ≤ 0.3	P50 1.00, waste 1.34	FAIL
G8	real-trace sign agreement with E2	sign agreement	–	*not run*

Table 1: Every pre-registered gate, with the threshold frozen before any render existed and the measured value as it fell. G4 passes: the cost side of the problem is easy. Six of the seven testable gates fail, including G5, declared make-or-break in advance. G8 requires a production trace we could not obtain and is reported as *not run* rather than estimated; Appendix F states what that costs the paper.

pairs and treat agreement below 65% as disqualifying, with the caveat carried in the body rather than buried.

Workload, baselines, statistics. The expensive render pass yields measured cost and quality for every (content, configuration) pair, so the scheduling experiments *replay* those tables rather than re-rendering; that is what makes the full sweep affordable. Load is a synthetic interactive stream (Poisson sessions, a branch tree per session, exponential think times). The comparison set was fixed in advance and nothing may be dropped: `fixed-`$\{4,20,50\}$; deadline-ordered (EDF (Liu & Layland, 1973)) and shortest-job-first variants; three ablations of BACH (one shared curve instead of per-request curves; no mid-flight revision; no admission control); BACH; and an oracle reading the true curves. BACH is driven by *out-of-fold* curve predictions, so the gap to the oracle measures estimation error rather than leakage. All intervals are session-clustered bootstraps with $B{=}2000$ resamples (Efron & Tibshirani, 1993); a difference counts only when its 95% interval excludes zero, and cross-validation holds out whole content groups.

5.1 Is the Quality Proxy Trustworthy?

Every result below is a claim about a learned preference score, and a null result is exactly what a broken score also produces. We therefore ran the pre-registered validation before interpreting anything. An annotator labelled 259 same-content pairs blind to configuration and display order, choosing the better image or declaring them indistinguishable: the random pairs on which the ≥ 0.65 gate is defined, plus a probe pitting the cheapest configuration against the reference one (protocol in Appendix H).

Agreement on random pairs is 80.3%, clearing the gate. Because a single accuracy number cannot distinguish a good metric from an easy sample, we bucket pairs by the metric's own reported gap (Table 10): agreement rises monotonically to 94.3% in the largest quartile while sitting near chance in the smallest—the signature of a calibrated score. On the probe, annotators called the pair indistinguishable 65.0% of the time against 31.2% on random pairs, and 71.4% of the pairs they did call went to the *cheaper* render. The metric's refusal to separate cheap from expensive is agreement with the annotator, not blindness.

One scope limit carries through every quality claim in this paper: both images occupied the same on-screen width, so the cheaper render was shown at native size while the reference was downscaled. The equivalence holds *at that display size*—the operating condition of interactive products, but not a claim about full-screen viewing.

What we changed after seeing data, in both directions. We revised the protocol twice after results existed, and the two are worth distinguishing. The first restricted the action space to a fixed resolution, on our own inference that the metric could not rank that axis; the human study refuted the inference and we retracted it. The second added the reduced-resolution static baselines and is what flipped G5. One would have made our result easier to defend and did not survive scrutiny; the other made it harder and did. Both are logged in Appendix A; the primary metric, strata, operating point and thresholds were never touched.

5.2 E1: QUALITY–COMPUTE CURVES (H1)

Question and protocol. Are per-request curves well modelled by Eq. (6); does the difficulty scale β vary across content; and is it predictable *before* rendering? A subset of the corpus is rendered at the full knob grid and the remainder at a sparse ladder. We fit Eq. (6) per request, report fit quality and the spread of $\hat{\beta}$ across the pre-registered strata, and predict $(\hat{q}_{\max}, \hat{\beta})$ from cheap features under leave-content-group-out cross-validation.

Outcome. All three gates fail (Table 1). **G2 is the premise of the entire paper**—if curves do not differ by content there is nothing for a content-aware allocator to exploit—and the measured ratio is 1.06 against a threshold of 2.0. G1's R^2 is indistinguishable from zero because the measured ladder is non-monotonic where the sampler switches, and G3's predictor is worse than a global constant. Details are in Appendix E; they are why the ceiling below sits where it does.

5.3 E2: BUDGET-AWARE ALLOCATION (H2)

Question and protocol. Does the allocator beat fixed configurations on delivered quality and SLO attainment—and if so, which of its parts is doing the work? We sweep offered load, SLO tightness and pool size, and at the pre-declared operating point compare BACH against the strongest fixed configuration with a paired session-clustered interval (gate G5). Three ablations isolate mechanisms: one shared curve instead of per-request curves; no mid-flight revision; no admission control. Every interval below is a paired session-clustered bootstrap over all five replications, so its point estimate is the same quantity as the corresponding difference of means in the tables.

The baseline set, and why it was wrong at first. Our original fixed baselines were `fixed-4`, `fixed-20` and `fixed-50`, all at the reference resolution—a control set biased in our favour, since every static comparator sat on the expensive side of the axis that turns out to matter. We added four reduced-resolution static baselines, measured for all 300 contents, and re-ran the sweep unchanged (PREREG R7). The conclusion depends entirely on that correction.

The allocator loses to a constant. At the headline cell the strongest static configuration is `fixed-4@768`, and BACH is worse than it by -0.0174 $[-0.0194, -0.0155]$ delivered quality—an interval excluding zero, so gate G5, declared make-or-break before any render existed, fails. The result is not a property of the declared operating point: across all 45 load $\times$ SLO $\times$ pool cells BACH wins significantly in 0 and loses significantly in 45. Nor does the workload call for switching: `fixed-4@768` is the strongest static configuration in *every one* of the 45 cells.

One qualification, stated before a reader raises it: `fixed-4@768` wins on quality while spending *more* compute (522 against 208 GPU-seconds), so that comparison alone is a Pareto trade, not a defeat. The defeat is elsewhere. In 30 of 45 cells a static configuration *strictly dominates* BACH—more quality *and* less compute. At the headline cell `fixed-1@512` delivers 0.673 in 181 GPU-seconds against BACH's 0.669 in 208: a quality advantage of $[+0.0029, +0.0046]$, excluding zero, at 13% less compute. More quality, less compute, no machinery.

No component is doing anything. BACH-global discards per-request curves entirely and scores every request with one shared curve; it is statistically indistinguishable from the full system, as are the ablations without mid-flight revision and without admission control (Table 2). The ablations are not evidence that the components are badly implemented, but that there is nothing for them to do.

The ceiling: how much was ever on the table. A deficit invites the reply that a better estimator would close it, so we measured the ceiling. The oracle is the same allocator handed the *true* curves—an upper bound no online estimator can pass. It beats `fixed-4@768` by only $+0.0118$ $[+0.0090, +0.0146]$ while spending $3.5\times$ its compute, and wins significantly in just 28 of 45 cells. The gap between BACH and the oracle is a research problem; the gap between the *oracle* and a constant is not worth crossing. That is what makes this a statement about the design space rather than about our implementation.

Under light load with loose SLOs every policy converges, as the pre-registration predicted; that null cell is reported in Appendix E, not suppressed.

Policy	Deliv. quality	GPU-s	SLO attain. α	P50 (s)	P99 (s)
`fixed-4`	0.637	850	1.000	0.53	0.71
`fixed-20`	0.130	3432	0.208	5.63	7.81
`fixed-50`	0.000	3590	0.000	5.59	11.66
`fixed-1@512`	0.673	181	1.000	0.11	0.12
`fixed-4@512`	0.682	355	1.000	0.22	0.26
`fixed-8@512`	0.628	585	1.000	0.36	0.42
`fixed-4@768`	0.687	522	1.000	0.33	0.35
EDF (`fixed-20`)	0.130	3432	0.208	5.63	7.81
SJF (`fixed-20`)	0.144	3426	0.230	5.52	7.81
BACH-global	0.671	172	1.000	0.11	0.12
BACH (no preemption)	0.670	180	1.000	0.11	0.20
BACH (no admission ctrl)	0.670	180	1.000	0.11	0.20
BACH	0.669	208	1.000	0.11	0.29
Oracle	0.699	1815	1.000	0.47	4.91

Table 2: E2 (H2). Headline cell (ρ=1.2, D=5 s, G=4). Three static configurations deliver more quality than BACH, and one of them (`fixed-1@512`) does so while spending less compute. Every BACH ablation is indistinguishable from the full system.

5.4 E3: SPECULATIVE BRANCH PREFETCH (H3)

Prefetch does hide latency—a cached branch returns instantly—but burns more compute on branches the user never picks than the delivered work costs, failing the efficiency half of G7 by a wide margin, stably across the sweep (Appendix D). The reason is the one that runs through this paper: when the cheapest configuration is nearly free, on-demand rendering is already fast, so there is little latency left to hide and every mispredicted branch is pure loss. The pre-registered real-trace replay (G8) is described in Appendix F; the export is an external dependency we could not obtain in time, and it is the one gate left untested.

6 DISCUSSION

A bound, not a verdict. The ceiling is the result that transfers. Perfect per-request knowledge buys $+0.0118$ over a well-chosen constant at $3.5\times$ the compute, and that number is recomputable on any backend from a profiling pass and a quality table—the artifacts we release—without building an allocator. Where the bound is this low, profile once and pick the best rung; where it is high, the same measurement says so.

Why the bound is low here. An allocator is only as good as the differences between the things it allocates to, and distillation removes them: a few-step sampler collapses the cost axis to a few non-dominated rungs whose best is nearly the same for every request. The optimiser is exact on the concave envelope; the premise that requests differ is what fails, which is also why no ablation separates from the full system. The result should reverse where that premise does—a wide non-dominated ladder (a non-distilled or video model), legible difficulty, or a request mix whose best rung differs—the first being cheapest to test.

Conclusion. We built a complete budget-aware allocator to measure what one could be worth, froze the tests before the first render, and found the ceiling already at floor level. The evidence is one backend, one proxy, one annotator and a synthetic workload (Appendix I); the bound, not the system, is what transfers.

AI USE STATEMENT

Large language models were used as a general-purpose assistant during this work: for drafting and copy-editing prose, for scaffolding and refactoring the experiment code, and as a search aid when surveying related work. Their use was confined to execution, not to the substantive claims.

Specifically: every reference in the bibliography was checked by the authors against a primary source (publisher page, proceedings entry, or arXiv record) for author list, title, venue and year, and entries that could not be verified were removed rather than retained; no experimental number in this paper was produced, suggested, or edited by a language model—every quantitative cell is emitted programmatically by the released analysis code from measured result files, and a gate whose experiment was not run is marked as such rather than filled in. The authors read, tested and take full responsibility for all code, all claims, and all text in this submission.

ETHICS STATEMENT

Who bears the degradation. A budget-aware scheduler improves aggregate delivered quality by spending less on some requests than a fixed policy would, and under overload by shedding a tail of requests entirely. Aggregate gains can therefore hide a distributional harm: if the difficulty estimator systematically assigns small budgets to a recognisable class of content, the users who ask for that content receive systematically worse output while the reported average improves. We treat this as a first-class evaluation question rather than an afterthought: alongside mean delivered quality we report the distribution across difficulty strata, and the shed rate by stratum, so that a policy which wins on the mean by starving a subgroup is visible rather than hidden. We do not claim to have solved fair allocation; we claim only that the paper reports the quantity a reader would need in order to notice the problem.

Human annotation. The human-preference validation collects pairwise judgements of image quality. Annotators are shown model-generated images and prompts from the corpus, are asked only which of two images they prefer, and provide no personal data. No identifying information about annotators is collected or released.

Production data. The real-workload replay uses an anonymised export of a deployed service's job queue. The export contains arrival times relative to the start of the window, job durations, configuration parameters and stable pseudonymous grouping keys. It contains no user identifiers, no prompt text, no media, no absolute timestamps and no billing records. The service is not named, and no absolute figures that would characterise its scale are reported.

Compute and environmental cost. The stated goal of this work is to spend *less* compute for the same delivered quality, so the direct effect of the method is a reduction. The speculative-prefetch component cuts the other way: it deliberately burns otherwise-idle GPU time on work that may be discarded. We therefore report wasted GPU-seconds as a headline cost of that component rather than only its latency benefit, and we report the regime in which the trade stops being worthwhile.

Dual use. The method is a scheduling layer and is agnostic to what the underlying generator produces; it neither improves nor constrains the content of any individual generation. It inherits, and does not mitigate, whatever risks attach to the generative backend it wraps.

REPRODUCIBILITY STATEMENT

The frozen experimental protocol—quality metric and its global normalisation, difficulty strata, baseline list, sweep grid, statistical procedure, and the numeric GO/NO-GO thresholds—is stated in Appendix A and was fixed before any render was produced. The cost model and its held-out separability check are in Appendix B; the allocator's optimality argument and the concave-envelope construction it relies on are in Appendix C; the interaction and prefetch model is in Appendix D; corpus construction, knob grid, baseline configurations, the human-preference validation protocol, the bootstrap procedure, and the trace schema and anonymisation are in Appendix E. Known limitations that bound what the results can show are collected in Appendix I. The supplementary material contains the complete pipeline: corpus construction, backend profiling, grid rendering, quality scoring, the discrete-event simulator, all policies, the bootstrap, the gate evaluator, and the script that regenerates every table and figure in this paper from result files—so that a reader can reproduce each reported cell, or observe which gate was not run, by re-running one command.

REFERENCES

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

Woongki Baek and Trishul M. Chilimbi. Green: A framework for supporting energy-conscious programming using controlled approximation. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, 2010.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

Mark Boddy and Thomas L. Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67(2):245–285, 1994.

Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2023.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *International Conference on Machine Learning (ICML)*, 2024.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Daniel Crankshaw, Xin Wang, Giulio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. Clipper: A low-latency online prediction serving system. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, 1993.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning (ICML)*, 2024.

Juncheng Gu, Mosharaf Chowdhury, Kang G. Shin, Yibo Zhu, Myeongjae Jeon, Junjie Qian, Hongqiang Liu, and Chuanxiong Guo. Tiresias: A GPU cluster manager for distributed deep learning. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2019.

Arpan Gujarati, Reza Karimi, Safya Alzayat, Wei Hao, Antoine Kaufmann, Ymir Vigfusson, and Jonathan Mace. Serving DNNs like clockwork: Performance predictability from the bottom up. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2020.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Henry Hoffmann, Stelios Sidiroglou, Michael Carbin, Sasa Misailovic, Anant Agarwal, and Martin Rinard. Dynamic knobs for responsive power-aware computing. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2011.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. Chameleon: Scalable adaptation of video analytics. In *ACM SIGCOMM Conference*, 2018.

Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Springer, 2004.

Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-Pic: An open dataset of user preferences for text-to-image generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning (ICML)*, 2023.

Zhuohan Li, Lianmin Zheng, Yinmin Zhong, Vincent Liu, Ying Sheng, Xin Jin, Yanping Huang, Zhifeng Chen, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. AlpaServe: Statistical multiplexing with model parallelism for deep learning serving. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 663–679, 2023.

Zongxia Li, Dawei Liu, Fuxiao Liu, Yuhang Zhou, Xiyang Wu, Jingxi Chen, Jing Xie, Xiaomin Wu, and Lichao Sun. ComfyClaw: Self-evolving skill harnesses for image generation workflows. *arXiv preprint arXiv:2607.01709*, 2026.

C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating diffusion models for free. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Kshiteej Mahajan, Arjun Balasubramanian, Arjun Singhvi, Shivaram Venkataraman, Aditya Akella, Amar Phanishayee, and Shuchi Chawla. Themis: Fair and efficient GPU cluster scheduling. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2020.

Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *ACM SIGCOMM Conference*, 2017.

Deepak Narayanan, Keshav Santhanam, Fiodar Kazhamiaka, Amar Phanishayee, and Matei Zaharia. Heterogeneity-aware cluster scheduling policies for deep learning workloads. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2020.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294, 1978.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

Aurick Qiao, Sang Keun Choe, Suhas Jayaram Subramanya, Willie Neiswanger, Qirong Ho, Hao Zhang, Gregory R. Ganger, and Eric P. Xing. Pollux: Co-adaptive cluster scheduling for goodput-optimized deep learning. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Francisco Romero, Qian Li, Neeraja J. Yadwadkar, and Christos Kozyrakis. INFaaS: Automated model-less inference serving. In *USENIX Annual Technical Conference (ATC)*, 2021.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Stelios Sidiroglou-Douskos, Sasa Misailovic, Henry Hoffmann, and Martin Rinard. Managing performance vs. accuracy trade-offs with loop perforation. In *ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE)*, 2011.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning (ICML)*, 2023.

Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. BOLA: Near-optimal bitrate adaptation for online videos. In *IEEE International Conference on Computer Communications (INFOCOM)*, 2016.

Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed H. Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Wencong Xiao, Romil Bhardwaj, Ramachandran Ramjee, Muthian Sivathanu, Nipun Kwatra, Zhenhua Han, Pratyush Patel, Xuan Peng, Hanyu Zhao, Quanlu Zhang, Fan Yang, and Lidong Zhou. Gandiva: Introspective cluster scheduling for deep learning. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2018.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. ImageReward: Learning and evaluating human preferences for text-to-image generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zhenran Xu, Xue Yang, Yiyu Wang, Qingli Hu, Zijiao Wu, Longyue Wang, Weihua Luo, Kaifu Zhang, Baotian Hu, and Min Zhang. ComfyUI-Copilot: An intelligent assistant for automated workflow development. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 632–643, 2025.

Xiangyuan Xue, Zeyu Lu, Di Huang, Zidong Wang, Wanli Ouyang, and Lei Bai. ComfyBench: Benchmarking LLM-based agents in ComfyUI for autonomously designing collaborative AI systems. *arXiv preprint arXiv:2409.01392*, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *ACM SIGCOMM Conference*, 2015.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for transformer-based generative models. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2022a.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research (TMLR)*, 2022b.

Peifeng Yu and Mosharaf Chowdhury. Salus: Fine-grained GPU sharing primitives for deep learning applications. In *Proceedings of Machine Learning and Systems (MLSys)*, 2020.

Ben Zhang, Xin Jin, Sylvia Ratnasamy, John Wawrzynek, and Edward A. Lee. AWStream: Adaptive wide-area streaming analytics. In *ACM SIGCOMM Conference*, 2018a.

Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J. Freedman. Live video analytics at scale with approximation and delay-tolerance. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017.

Hong Zhang, Yupeng Tang, Anurag Khandelwal, and Ion Stoica. SHEPHERD: Serving DNNs in the wild. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 787–808, 2023.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.

Steps s	fp16, 512	fp16, 768	fp16, 1024	bf16, 1024
1	0.016	0.030	0.050	0.036
4	0.033	0.048	0.079	0.065
8	0.054	0.073	0.117	0.102
20	0.121	0.236	0.424	0.374
50	0.288	0.556	1.000	0.895

Table 3: Profiled configuration cost factor $\phi(a)$, normalised to the reference configuration (50 steps, fp16, 1024). The precision axis is fp16/bf16: fp8 required a quantisation backend that was not available on the measurement host, so the pre-registered fallback applied.

A Pre-Registration and Decision Rules

Before producing a single render we froze the measurement protocol and the decision rules, so that the reported result cannot be a selection over analysis choices. The frozen document is released with the code. Its substantive commitments are:

Metric. The primary quality score, its *global* normalisation constants, and the list of secondary metrics were fixed in advance; secondary metrics may only support robustness statements and may not overturn a headline result. Global rather than per-request normalisation is deliberate: rescaling each request between its own cheapest and reference configuration would force the ceiling $q_{\max}$ to be identical across content *by construction*, making H1 unfalsifiable.

Strata. Difficulty strata are defined by a score computed only from pre-render features (prompt noun-phrase count, requested resolution, conditioning-image gradient energy), split at terciles, and written to disk before any quality is measured. Re-stratifying by the *fitted* β after the fact is explicitly forbidden, since it would guarantee the separation E1 sets out to test.

Operating point. The headline cell (ρ=1.2, D=5 s, G=4, skewed branching) was declared in advance. Every other cell of the load $\times$ SLO $\times$ pool grid is reported, marked exploratory, including the light-load cell where we predict and expect no effect.

Statistics. All intervals are session-clustered bootstraps with B=2000; a difference is called real only when its 95% interval excludes zero. Cross-validation holds out whole content groups, so different configurations of one content cannot appear on both sides of a split.

Gates. Table 1 in the body lists the numeric GO/NO-GO thresholds and their outcomes on both action spaces. G5 is the make-or-break test: if BACH does not beat the best fixed configuration at the declared operating point with an interval excluding zero, the central claim is not supported, and the pre-registered response is to report an honest characterisation of why fixed configurations are hard to beat rather than to move the threshold.

B Cost Model and Profiling Details

Table 3 reports the profiled configuration cost factor $\phi(a)$ over the knob grid, normalised so that the reference configuration has $\phi = 1$. Profiling sweeps each knob with three timing repeats after discarding warmup iterations, and fits $\log c = \log \gamma(x) + \log \phi(a)$ by least squares with the reference configuration pinned for identifiability.

Separability is a modelling convenience, so we test it rather than assume it: the fit reports the mean absolute percentage error on *held-out content*, and the pre-registered rule (gate G4) is that above 15% we abandon the separable form for a direct regression of $\hat{c}(x, a)$ and say so in the body. Table 3 gives the profiled factor across steps, resolution and precision. Two properties matter downstream: cost is linear in steps at fixed resolution, which is what makes a step budget interpretable as a compute budget; and held-out MAPE is 0.6%, so the separable form is accurate enough that none of the scheduling results depend on cost-model error.

Algorithm 1 Online quality–compute curve estimation

Require: request x, profiled $\phi(\cdot)$, regressor f_θ
1: $\hat{\gamma} \leftarrow$ content-cost head of $f_\theta(x)$
2: $(\hat{q}_{\max}, \hat{\beta}) \leftarrow$ curve head of $f_\theta(x)$
3: **return** cost $\hat{c}(x, a) = \hat{\gamma}\,\phi(a)$; curve $\hat{Q}_x(\cdot)$ via Eq. (6)
on completion of x **at cost** b **with quality proxy** q**:**
4: $\theta \leftarrow \theta - \eta\,\nabla_\theta \ell\big(f_\theta(x);\,(b, q)\big)$ ▷ EW online update

Algorithm 2 Budget-aware allocation (marginal-return greedy)

Require: requests $\{x_r\}$, curves $\hat{Q}_r$, costs $\hat{c}$, budget B, throughput G, deadlines D_r
1: admissible $\mathcal{A}_r \leftarrow$ Eq. (4) for each r; drop r if $\mathcal{A}_r = \varnothing$ ▷ shed
2: $a_r \leftarrow \arg\min_{a \in \mathcal{A}_r} \hat{c}(x_r, a)$; used $\leftarrow \sum_r \hat{c}(x_r, a_r)$
3: priority queue $\mathcal{Q} \leftarrow$ each r keyed by ratio Eq. (7)
4: **while** used $< B$ **and** $\mathcal{Q} \neq \varnothing$ **do**
5: pop r with max ratio; let a_r^+ be its next admissible config
6: **if** used $+ (\hat{c}(x_r, a_r^+) - \hat{c}(x_r, a_r)) \le B$ **then**
7: used $+= \hat{c}(x_r, a_r^+) - \hat{c}(x_r, a_r)$; $a_r \leftarrow a_r^+$
8: re-insert r with updated ratio if a further upgrade exists
9: **end if**
10: **end while**
11: **return** allocation $\{a_r\}$

C Optimality of the Marginal-Return Greedy

We record the standard guarantee behind Algorithm 2.

Proposition 1. *If each per-request quality–compute curve $\hat{Q}_r$ is non-decreasing and concave in allocated compute, then the marginal-return greedy (Algorithm 2) is optimal for the continuous relaxation of the budgeted allocation Eq.* (3)*, and its integral solution differs from the optimum by at most the quality of a single upgrade step.*

Proof sketch. Concavity makes the per-request marginal returns decreasing, so the union of all upgrade steps, sorted by quality-per-cost ratio Eq. (7), has non-increasing ratios within each request. Greedily taking the globally highest-ratio remaining upgrade is the classical fractional-knapsack rule, optimal for the relaxation (Kellerer et al., 2004); rounding down at the budget boundary loses at most the last (partial) upgrade.

When curves are not concave. Measured configuration ladders need not be concave—a distilled few-step sampler can dominate a multistep one at equal cost, putting a kink in the frontier. The implementation therefore operates on the *upper concave envelope* of the measured (cost, quality) points. Points strictly inside the envelope are discarded: for each of them some cheaper envelope rung achieves at least the same quality, so no allocation ever wants them. The envelope upper-bounds achievable quality and restores the non-increasing marginal ordering the greedy needs. The gap this introduces relative to the true integral optimum is bounded by one envelope segment and is measured empirically against the exhaustive solution on small instances.

D Revision and Prefetch Mechanisms

The branch model $\mathcal{B}(r)$, the estimation of branch probabilities π_j from session history, and the expected-hidden-latency objective of Section 3. That objective is a budgeted maximum-coverage problem (Khuller et al., 1999), for which the ratio greedy of Algorithm 4 is the standard $(1 - 1/e)$ approximation (Nemhauser et al., 1978; Sviridenko, 2004).

Figure 2 reports sensitivity of prefetch value to branch factor k and probability skew. The sign of the finding is stable across the sweep: latency reduction saturates quickly while wasted compute grows with k, so there is no region of this space in which prefetch passes G7.

Algorithm 3 Mid-flight budget revision

Require: in-flight $\{(r, a_r, b_r^{\text{spent}})\}$, pending queue $\mathcal{P}$, now t, pool G
$F \leftarrow$ projected finish times of the in-flight set
$d \leftarrow \textsc{Doomed}(\mathcal{P}, F)$ ▷ EDF onto earliest-free GPUs at each request's floor cost
if $d > 0$ **then**
  for r in in-flight, ascending remaining marginal return **do**
    $a_r^- \leftarrow$ cheapest configuration with $c(x_r, a_r^-) \geq b_r^{\text{spent}}$ meeting D_r
    if $a_r^- \neq a_r$ **then** revise r to a_r^-; update F; $d \leftarrow \textsc{Doomed}(\mathcal{P}, F)$
    if $d = 0$ **then break**
  end for
else if $\mathcal{P} = \varnothing$ **then**
  spend remaining slack upgrading in-flight requests by marginal return
end if

Algorithm 4 Speculative branch prefetch

Require: branch candidates $\{r^{(j)}\}$ with π_j, Δ_j; idle budget B_{idle}
sort branches by $\pi_j \Delta_j / \hat{c}(x_{r^{(j)}}, a_{r^{(j)}})$ (desc.)
used $\leftarrow 0$; $S \leftarrow \varnothing$
for $r^{(j)}$ in sorted order **do**
  if used $+ \hat{c}(x_{r^{(j)}}, a_{r^{(j)}}) \leq B_{\text{idle}}$ **then**
    render $r^{(j)}$; cache by (context, config); $S \leftarrow S \cup \{r^{(j)}\}$
    used $+= \hat{c}(x_{r^{(j)}}, a_{r^{(j)}})$
  end if
end for
on user choice $r^\star$**:** return cached result if hit, else allocate on-demand (Alg. 2)

E Additional Experimental Detail

Figure 3 gives the full load sweep behind the headline cell of Section 5.3, including the light-load regime where the pre-registration predicts—and this appendix reports—no effect.

This appendix collects the material the body points to: per-stratum curve fits (Table 5) and the quality–compute frontier, both panels of Figure 4, and the load sweep in full (Figure 3), including the light-load cells in which every policy converges and no difference is detectable. The human-preference protocol is in Appendix H; the pre-registration, including the bootstrap procedure and every threshold quoted in the body, is in Appendix A.

F Real-Trace Replay

Question. Do the synthetic conclusions survive a real production workload?

Protocol and decision rule. We replay an anonymised export of the deployed service's `ai_generation_jobs` queue through each policy in the trace-driven simulator calibrated to the profiled cost model, split into steady, bursty and tight-SLO regimes. Gate G8 asks only for *sign*

Setting	Perceived P50 (s)	Perceived P99 (s)	Hit rate	Wasted GPU-s
No prefetch	0.11	0.20	—	0
Uniform π	0.00	0.16	0.407	893
Learned π	0.00	0.17	0.413	880

Table 4: E3 (H3). Perceived latency, hit rate and wasted compute under uniform versus estimated branch probabilities. Prefetch removes essentially all perceived queueing delay and hit rates are respectable, but the wasted compute exceeds the delivered work, which is what fails gate G7.

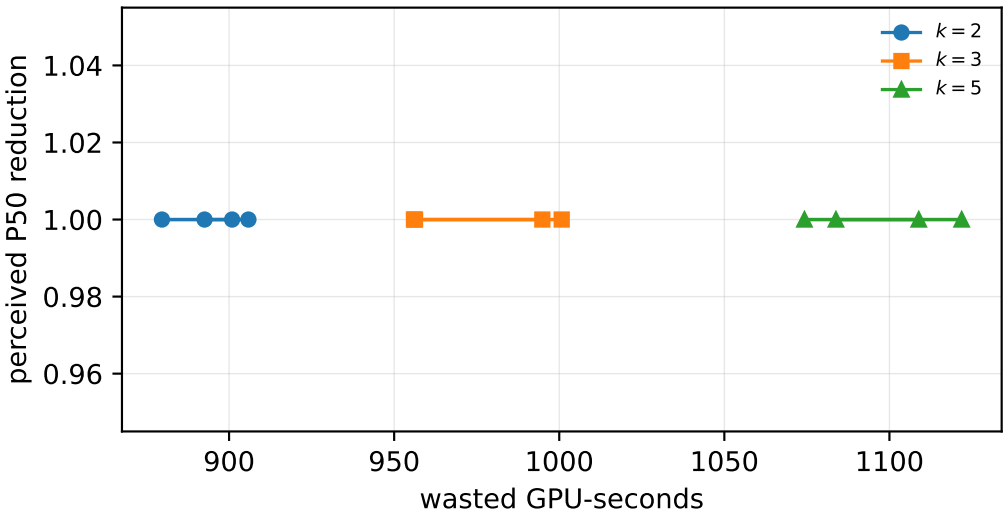

Figure 2: E3 sensitivity: perceived-latency reduction against wasted GPU-seconds, by branch factor. Latency reduction saturates almost immediately while wasted compute grows with the branch factor, so no point in this space passes G7.

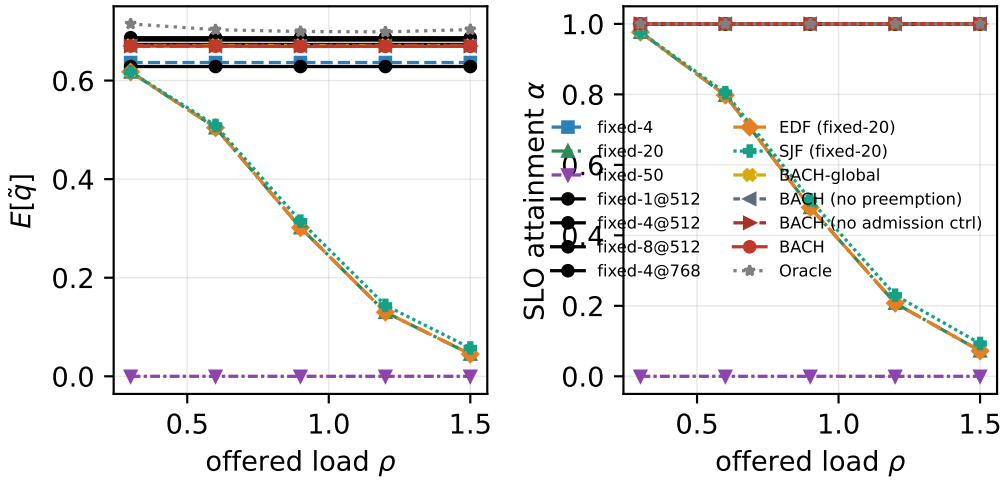

Figure 3: E2 (H2), full sweep. Delivered quality (left) and SLO attainment (right) against offered load ρ for every policy. The pre-registered prediction was a crossover: indistinguishable policies under light load, a widening BACH advantage as the deadline begins to bite, and graceful degradation under overload. The measured sweep does not show it; the curves are ordered the same way at every load.

agreement with E2: production arrival patterns, content mix and contention differ enough from the synthetic stream that claiming transferable magnitudes would overreach.

Status. The anonymised export was not available in time for this submission, so G8 is the one pre-registered gate that remains untested. We state this rather than substituting a synthetic stand-in: the whole point of G8 is that its arrivals are *not* ours to generate. Every conclusion in this paper is therefore a conclusion about a measured backend driven by a synthetic workload, and the external-validity limit that implies is stated in Appendix I.

Difficulty stratum	Curve R^2	$\hat{\beta}$ (rel.)	$\hat{q}_{\max}$	Param. pred. err.
Easy	-0.000 [-0.000, -0.000]	1.00	0.549 [0.524, 0.574]	1.336
Medium	-0.000 [-0.000, -0.000]	0.63	0.608 [0.585, 0.632]	1.336
Hard	-0.000 [-0.000, -0.000]	1.06	0.611 [0.584, 0.640]	1.336
All	-0.000 [-0.000, -0.000]	0.76	0.589 [0.574, 0.604]	1.336

Table 5: E1 (H1). Curve fit, difficulty-scale spread relative to the easy stratum, fitted ceiling, and leave-group-out parameter-prediction error, with session-clustered bootstrap intervals. All three gates fail: R^2 is indistinguishable from zero because the measured ladder is non-monotonic where the sampler switches (Appendix E), $\hat{\beta}$ barely separates the strata, and the prediction error exceeds what a single global constant achieves.

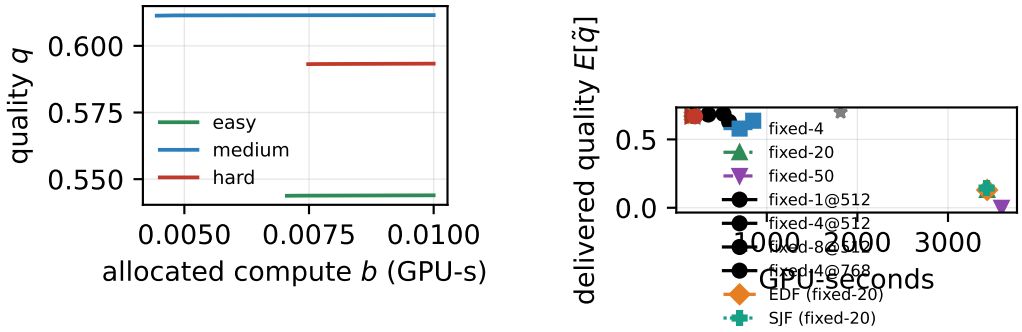

Figure 4: **Left, E1:** fitted quality–compute curves by difficulty stratum; the strata are nearly superimposed, which is gate G2's failure seen directly. **Right, E2:** quality–cost frontier at the pre-registered operating point, on which BACH sits far to the cheap side of every fixed configuration.

G THE RESTRICTED ACTION SPACE

The primary analysis uses the pre-registered 60-configuration action space. This appendix reports a secondary arm: the identical sweep—same measurements, same policies, same thresholds, same headline cell—with resolution held at 1024 so the allocator may vary only steps and precision. Nothing changes except which configurations the scheduler may choose.

The arm exists for a historical reason worth recording, because it bears on how the paper's amendment log should be read. This restriction was originally introduced (PREREG R4) as the *primary* analysis, on our own inference that the quality metric could not rank resolution and that BACH's apparent advantage over the then-current baselines was therefore an artifact. The human-preference study in Appendix H refuted that inference: the metric is calibrated, and annotators independently call the cheap-versus-expensive comparison a tie two thirds of the time. PREREG R6 retracts R4's justification and returns the pre-registered grid to primary, where it began. What survives is a straightforward action-space ablation: Table 7 is the headline cell under the restriction, and Table 8 applies every gate to both arms.

Rendered at	Mean score	Δ vs 1024	|Δ| / metric SD	Frac. 1024 higher
1024×1024	22.5284	+0.0000	0.000	0.000
768×768	22.5266	+0.0018	0.014	0.503
512×512	22.4682	+0.0602	0.043	0.780

Table 6: The frozen preference metric, scoring the *same* image at three resolutions. Downsampling moves the score by only a few percent of the metric's own spread. Read together with Table 9, this is the metric agreeing with the annotator that these renders are close—not failing to see a difference the annotator could.

Policy	Deliv. quality	GPU-s	SLO attain. α	P50 (s)	P99 (s)
`fixed-4`	0.637	850	1.000	0.53	0.69
`fixed-20`	0.133	3403	0.212	5.61	7.81
`fixed-50`	0.000	3567	0.000	5.59	11.66
EDF (`fixed-20`)	0.133	3403	0.212	5.61	7.81
SJF (`fixed-20`)	0.146	3396	0.235	5.50	7.81
BACH-global	0.630	3245	0.999	3.89	4.98
BACH (no preemption)	0.629	3197	0.999	3.75	4.97
BACH (no admission ctrl)	0.629	3197	0.999	3.75	4.97
BACH	0.628	3217	0.999	3.76	4.97
Oracle	0.682	2762	1.000	2.83	4.92

Table 7: Secondary arm: the headline cell with resolution held at 1024, so the allocator may vary only steps and precision. Compare Table 2, the same cell on the full action space.

			Measured	
Gate	Quantity	Threshold	60-config grid	1024-only
G1	median per-request curve fit quality	≥ 0.85	-0.000 ×	0.294 ×
G2	difficulty-scale ratio, hard vs easy stratum	≥ 2.0	1.061 ×	1.370 ×
G3	curve-parameter prediction gain over a global constant	≥ 0.2	-0.055 ×	-0.156 ×
G4	separable cost model held-out MAPE	≤ 0.15	0.006 ✓	0.006 ✓
G5	BACH minus best fixed, delivered quality	CI excludes 0	-0.017 ×	-0.008 ×
G6	fraction of the oracle gain captured	≥ 0.7	-1.479 ×	-0.197 ×
G7	perceived P50 reduction from prefetch	P50 ≥ 0.2 and waste ≤ 0.3	P50 1.00, waste 1.34 ×	1.000 ×
G8	real-trace sign agreement with E2	sign agreement	*not run*	*not run*

Table 8: Every pre-registered gate applied unchanged to both action spaces: the pre-registered 60-configuration grid and the 1024-only restriction. Thresholds are identical in both columns.

The episode is the clearest illustration we can give of why the amendment log matters more than the pre-registration. R4 was a post-hoc change that made our position easier to defend and did not survive its own validation study; R7—the reduced-resolution static baselines that decided the paper—was a post-hoc change that made our position harder to defend, and it held. Neither is visible from the final numbers alone.

H HUMAN-PREFERENCE VALIDATION

Protocol. The pre-registered validation (Appendix A) asks whether the quality proxy tracks human judgement well enough to carry the paper's conclusions. One annotator labelled 259 pairs drawn from the rendered corpus. Every pair shows two renders of the *same* prompt at different compute settings; configuration is hidden, left/right order is randomised per pair, and the annotator may declare a pair indistinguishable. Two sets are drawn: the *random* pairs on which the pre-registered ≥ 0.65 agreement gate is defined, and a *probe* that always pits the cheapest configuration against the reference one, included because that specific comparison is what the headline result rests on. Agreement is computed only over pairs the annotator actually called; ties are reported separately rather than being scored as disagreement, since a tie is evidence about the images, not about the metric.

Pair set	role	labelled	ties	scored	agreement
Random same-content pairs	pre-registered gate	199	62 (31%)	137	0.803 ✓
1-step 512 vs. 50-step 1024	supplementary	60	39 (65%)	21	0.571

Table 9: Pre-registered human-preference validation. The gate is defined on the random pairs and is met. The tie column carries as much of the finding as the agreement column: the cheap-versus-reference comparison is called indistinguishable roughly twice as often as a random pair.

Metric-gap quartile	$\lvert\Delta q\rvert$ range	n	human agreement
Q1	0.022–0.066	34	0.647
Q2	0.066–0.166	34	0.735
Q3	0.166–0.316	34	0.882
Q4	0.316–0.778	35	0.943

Table 10: Human agreement as a function of the metric's own confidence, over the random pairs the annotator called. Chance is 0.500. The monotone rise is the signature of a calibrated score; the low-gap quartile is *supposed* to be near chance, because those images really are close.

Calibration, not just accuracy. A single agreement number cannot separate "the metric is good" from "the sampled pairs were easy". Bucketing the random pairs by the metric's *own* reported gap $|\Delta q|$ separates them directly: a calibrated score should sit near chance where it reports a small difference and approach certainty where it reports a large one. Table 10 shows exactly that shape, rising monotonically from near chance in the smallest quartile to 94.3% in the largest. We regard this, rather than the headline agreement figure, as the evidence that the proxy can be trusted for the comparisons this paper makes.

Scope: display size. Both images in a pair were presented at equal on-screen width (roughly 500 px). A 512 render is therefore shown at native size while a 1024 render is downscaled by a factor of two. The equivalence the annotator reports is consequently established *at that display size*. This is the operating condition of the interactive products this work targets—a generated clip returned into a phone or browser viewport—but it is not a claim about full-screen or print viewing, and we do not make one. Every quality comparison in this paper, ours and the baselines' alike, inherits this scope.

Reproducibility. The labelling interface, the exact pair set with its hidden configurations, and the scoring script are included in the supplementary material, so the validation can be repeated with different annotators or a different display size without re-rendering anything.

I Threats to Validity

The quality proxy. Every claim about "delivered quality" is a claim about a learned preference score, not about human judgement directly. We validate it on held-out pairwise human comparisons and report the agreement rate; if the proxy tracks preference poorly, the allocator may be optimising the wrong objective even when it wins on the metric. This is the paper's main caveat and is stated in the body. A single scalar preference score is also a coarse summary of what makes a generation good: multi-dimensional benchmarks such as VBench (Huang et al., 2024) decompose generative quality into separately scored axes, and an allocator ranked on one of those axes rather than an aggregate could well behave differently.

Scale. The testbed profiles one GPU and reaches larger pools through a simulator calibrated to those measurements. Simulation abstracts away batching interactions, memory pressure, model-swap costs and network/storage latency for cached prefetch results, all folded into the throughput term G and the cost model. Conclusions about ordering between policies are more robust than conclusions about absolute magnitudes at scale, and we phrase them accordingly.

Backend specificity. Results are for one diffusion image/video backend. The formulation applies to any generator with a compute–quality knob, but the curves must be re-profiled; we do not claim transfer without it.

Trace representativeness. The production trace comes from a single deployed service over a bounded window. It establishes that the synthetic conclusions survive contact with a real arrival pattern, not that the magnitudes generalise across services—which is why the pre-registered gate asks only for sign agreement.