SCHEDULING BACKWARDS FOR LLM BATCHING UNDER PROGRESS CONSTRAINTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continuous batching lets language-model requests share a decode iteration despite differences in their accumulated contexts and attention workloads. We study the tradeoff between batching efficiency and active-batch progress disparity through a budgeted formulation: an operator specifies a disparity tolerance, and the scheduler minimizes logical makespan subject to it. In an offline nonpreemptive model with B slots, this constraint creates admission blackouts: a slot can be free although no new request may enter. We introduce Right-Aligned scheduling (RA), which constructs schedules backwards to coordinate starts. A window invariant proves that RA never takes longer than sorted static batching. Refined idle-time accounting yields a $\big(2 - (11B - 2)/(9B^2)\big)$-approximation, with a tight $3/2 - 1/(2B)$ factor on instances of at most $2B$ requests for $B \geq 3$. Observation-based retiming preserves feasibility under valid duration upper bounds. Experiments compare with 20,163 exact budget optima, isolate algorithm components, and examine achieved tradeoffs on public trace subsets and independent synthetic workloads. At an established conversational operating point, plain RA improves logical throughput by 8.9% over the tested forward baselines; retiming retains a 4.6% gain under bounded forecast perturbations.

1 INTRODUCTION

Large language models generate responses one token at a time. Continuous batching admits new requests as others finish, sharing model execution without waiting for an entire batch to drain (Yu et al., 2022; Kwon et al., 2023; Agrawal et al., 2024b). In the synchronous decode setting considered here, active requests advance together. However, *one token is not a fixed amount of work*: in full-attention models, a longer accumulated context requires more attention computation and memory access (Chen, 2023; Sheng et al., 2024).

Consider two requests with equal prompt lengths that have generated 32 and 512 tokens. Both receive one token in a shared iteration, despite different context-dependent resource demands. A lightweight request can therefore experience delay associated with heavier batch companions. Equal token-generation rates do not remove this disparity. Limiting progress differences offers one response: strict synchronization eliminates them but can leave slots idle, whereas unrestricted replacement permits greater heterogeneity. The interesting choices lie between these extremes.

We study this tradeoff between logical makespan and worst-case active-batch progress disparity. Following the ϵ-constraint approach to multi-objective optimization (Mavrotas, 2009), the operator specifies a disparity budget α, and the scheduler minimizes makespan within it. The budget is neither a latency-conversion coefficient nor a universal definition of user fairness. The question is *how much efficiency can be retained at each chosen level of homogeneity*, not whether a constrained optimum can outperform its unconstrained counterpart.

The scheduling difficulty lies in admission timing. We consider an offline backlog with at most B concurrent requests, each advancing without interruption once admitted. If an active request has advanced more than α steps, a fresh request at progress zero cannot enter, even when a slot is free. Starting a long request immediately can therefore prevent useful admissions later. With two slots, $\alpha = 1$, and lengths $(3, 2, 2)$, longest-first finishes at five; delaying the longest request by one step permits completion at four (Figure 1). The decision is not just *which* request to start, but *when*.

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.

We propose *Right-Aligned scheduling (RA)* to coordinate these admissions. Under contiguous unit-rate service, overlapping requests' progress difference equals their start-time difference, so each sufficiently long request creates an admission *blackout*. RA exploits this structure by constructing schedules backwards, processing requests in decreasing length order and placing each as late as feasibility permits.

Our main contributions are as follows:

- **A window invariant for backward scheduling.** We bound the schedule's growth over every group of at most B placements. This proves $T_{\mathrm{RA}} \le L_{\mathrm{static}}$: RA never takes longer than sorted static batching on any instance. The guarantee is intrinsic to RA, not obtained by comparison with a fallback schedule.
- **Approximation guarantees across progress budgets.** Relative to the same-budget optimum, refined idle-time accounting yields the factor $2 - (11B - 2)/(9B^2)$, with a tight $3/2 - 1/(2B)$ factor for at most $2B$ requests when $B \ge 3$. RA is optimal when every request exceeds 2α and recovers the longest-processing-time (LPT) makespan when the budget is nonbinding. These guarantees also support an approximate representation of the two-objective tradeoff.
- **Observation-based retiming.** We derive a rule that adjusts the backward plan using observed completions while preserving capacity and progress feasibility under valid duration upper bounds (Section 5).
- **Numerical evaluation and component attribution.** We separate exact budget optima from algorithmic gaps, isolate RA from the portfolio and retiming components, and report both efficiency and achieved progress disparity (Section 6).

Positioning and scope. For equal prompts, progress disparity is exactly current-context disparity; with prompt range δ, the latter is at most $\alpha + \delta$. Our theory studies this scheduling discipline in logical decode iterations, not a calibrated GPU-latency objective. Admission control may shift delay into the queue, so waiting and completion times are reported separately. Client-level service accounting, including resource-aware cost functions in VTC (Sheng et al., 2024), addresses cumulative allocation rather than batch composition. Memory management, prefill coordination, and online KV-constrained or prediction-aware scheduling are complementary concerns (Kwon et al., 2023; Agrawal et al., 2024b; Jaillet et al., 2025; Ao et al., 2025; Wang et al., 2025; Chen et al., 2025). Our offline guarantees do not assert an online competitive ratio. Public Vidur traces (Agrawal et al., 2024a; Microsoft and Vidur Contributors, 2026) and independent synthetic workloads are replayed in a logical simulator; no native Vidur or GPU execution is claimed.

2 Model and the Blackout Structure

An instance is $I = (o, B, \alpha)$, where $o_i \in \mathbb{Z}_{>0}$ is the contiguous service requirement of request i, $B \in \mathbb{Z}_{>0}$ is the number of slots, and $\alpha \in \mathbb{Z}_{\ge 0}$. All requests are available at time zero. A schedule assigns a start $s_i \in \mathbb{Z}_{\ge 0}$ to each request. Request i occupies one slot on $I_i = [s_i, s_i + o_i)$ and has progress $a_i(t) = t - s_i$ while active. It is not interrupted. Feasibility requires

$$|S(t)| \le B, \qquad \max_{i \in S(t)} a_i(t) - \min_{i \in S(t)} a_i(t) \le \alpha, \tag{1}$$

where $S(t) = \{i : t \in I_i\}$ and the disparity of an empty set is zero. We call this progress homogeneity; below, "α-fair is shorthand for this active-batch constraint. A feasible set of execution intervals can always be assigned to B slots: greedily place each interval on a slot whose previous interval has ended.

For a capacity-respecting schedule s, define $D(s) = \max_t[\max_{i \in S(t)} a_i(t) - \min_{i \in S(t)} a_i(t)]$ and $T(s) = \max_i(s_i + o_i)$. Minimizing over capacity-respecting schedules gives

$$V(\alpha) = \min_{s:\, D(s) \le \alpha} T(s).$$

It is non-increasing. Exact minimizers with disparity tie-breaking generate nondominated objective pairs; an approximate scheduler need not. Write $T^* = V(\alpha)$ for the fixed-budget optimum, $q_1 \ge$

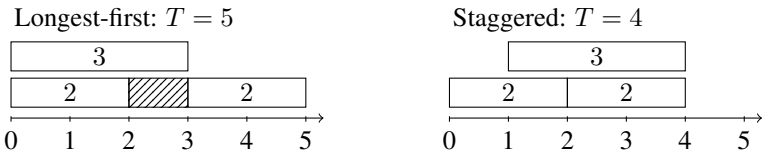

Figure 1: Two slots, $\alpha = 1$, and lengths $(3, 2, 2)$. Left: at time two, the free slot cannot be filled because the long request has progress two (hatched interval). Right: delaying the long request permits both short requests to share the other slot consecutively.

$\cdots \geq q_n$ for the sorted lengths, and $W = \sum_i o_i$. Set $q_j = 0$ for $j > n$. Logical throughput is $W/T \leq B$. Makespan factors $T_A/T^* \geq 1$ are reciprocals of the corresponding throughput guarantees. They do not automatically apply to batch-dependent GPU time.

Lemma 1 (Start times and blackouts)**.** *A capacity-respecting contiguous schedule satisfies* (1) *if and only if overlapping requests have* $|s_i - s_j| \leq \alpha$*. Equivalently,*

$$s_j \notin (s_i + \alpha, s_i + o_i) \quad \textit{for all } i \neq j. \tag{2}$$

The interval is empty when $o_i \leq \alpha$*.*

Proof. On an overlap, $a_i(t) - a_j(t) = s_j - s_i$. If $s_i \leq s_j$ and the pair overlaps, then $s_j < s_i + o_i$; excluding the open interval in (2) is exactly the requirement $s_j \leq s_i + \alpha$. □

Thus a forward policy is fair whenever every admission occurs with maximum current progress at most α. It may insert requests into a running batch, but it cannot bypass a blackout merely because capacity is idle. Endpoints matter: starts at $s_i + \alpha$ and $s_i + o_i$ are permitted.

The admission obstruction. Figure 1 contrasts starts $(0, 0, 3)$ with $(1, 0, 2)$ for lengths $(3, 2, 2)$. Each overlap in the latter schedule has start difference one, and its makespan four matches the rounded work bound $\lceil 7/2 \rceil$. Intentional initial idleness can therefore reduce total idle slot-time under the constraint.

Complexity and a benchmark. For $\alpha \geq q_1$ the constraint is nonbinding and the problem is $P\|C_{\max}$. It is therefore strongly NP-hard with B in the input, NP-hard already for $B = 2$, and has no FPTAS with variable B unless P=NP (Garey & Johnson, 1979). This reduction is a statement about the embedded classical problem, not a new hardness caused specifically by fairness. Sorted static batching starts each consecutive block of at most B sorted requests simultaneously and waits for the block to drain. It is fair for every α and has

$$L_{\text{static}} = \sum_{r \geq 1} q_{(r-1)B+1} \leq \frac{W}{B} + \left(1 - \frac{1}{B}\right) q_1 \leq \left(2 - \frac{1}{B}\right) T^*. \tag{3}$$

This is a static baseline, not continuous LPT. If every length exceeds 2α, overlaps form disjoint cohorts of size at most B; a partition argument gives $T^* = L_{\text{static}}$. In particular this includes $\alpha = 0$. Proofs and the integrality justification are in Appendix A.

Fairness-specific lower bounds. Besides $T^* \geq \max\{q_1, W/B\}$, we use

$$BT^* \geq W + \Theta, \qquad \Theta := \sum_{j=2}^{B} (q_1 - \alpha - q_j)_+, \tag{4}$$

$$T^* \geq q_1 + q_{B+1} - \alpha, \qquad n > B. \tag{5}$$

During the longest request's blackout, another slot can receive no new request, which gives the idle-time charge Θ. For (5), either one of the $B + 1$ longest requests starts after that blackout, or two of them must fit on one slot before its end. A strengthened version, needed later, is

$$T^* \geq q_1 + \min\left\{ q_{B+k}, \left\lceil \frac{k}{B-1} \right\rceil q_{B+k} - \alpha \right\}, \quad B \geq 2,\ q_1 > \alpha,\ 1 \leq k \leq n - B. \tag{6}$$

These bounds are computable from the lengths; no exact unconstrained scheduling optimum is required to evaluate them.

3 SCHEDULING BACKWARDS

3.1 RIGHT-ALIGNED SCHEDULING

Construct the schedule on $(-\infty, 0]$. Let r_j be the left edge of the occupied region on slot j, initially zero. For a request of length ℓ, a placement t on slot j is admissible if $t + \ell \le r_j$ and, for each already placed request (s, o),

$$\big[o > \alpha \Rightarrow t \notin (s + \alpha, s + o)\big] \ \wedge \ \big[\ell > \alpha \Rightarrow s \notin (t + \alpha, t + \ell)\big]. \tag{7}$$

The second condition is essential: the new request must not place its own blackout around an old start.

Algorithm 1 Right-Aligned scheduling (RA)

Require: Lengths o, slots B, progress budget α
1: $r_j \leftarrow 0$ for $j \in [B]$; $P \leftarrow \varnothing$
2: **for** each request i in non-increasing order of o_i **do**
3: **for** $j = 1, \ldots, B$ **do**
4: $\tau_j \leftarrow \max\{t \le r_j - o_i : (7) \text{ holds against } P\}$
5: **end for**
6: $j^* \leftarrow \arg\max_j \tau_j$; break ties by smallest slot index
7: $t_i \leftarrow \tau_{j^*}$; $r_{j^*} \leftarrow t_i$; $P \leftarrow P \cup \{(t_i, o_i)\}$
8: **end for**
9: **return** $s_i = t_i - \min_h t_h$ and the slot assignment

The feasible start set is a finite union of closed intervals. Its relevant endpoints are the fit limit and the endpoints induced by (7), giving a direct $O(Bn^3)$ implementation. If m is the smallest placed start, then $t = m - \ell$ is always admissible: it ends before every old request begins. RA is therefore well defined, never overlaps two requests on one slot, and checks every pairwise blackout relation when the later-placed request is inserted. Shifting the completed schedule preserves feasibility. The first request ends at zero and all others end no later, so $T_{\mathrm{RA}} = -\min_i t_i$.

When $\alpha \ge q_1$, all blackout constraints disappear. RA then chooses a slot of smallest current load, in decreasing length order, and has exactly the LPT makespan. Thus Graham's factor $4/3 - 1/(3B)$ applies (Graham, 1969). Building from the right is consequential only when fairness binds.

3.2 THE WINDOW INVARIANT

A placement *drops* if its start is smaller than the current leftmost start m. Its drop amount is the increase in the partial makespan. Write the slot edges as a_j, with $a_{\max} = \max_j a_j$. At a drop of length ℓ, after the first placement, the following facts hold:

$$a_j < m + \ell \quad \text{for every } j, \qquad a_{\max} - m \le \alpha. \tag{8}$$

Indeed, if $a_j \ge m + \ell$, the request could start at the already feasible start m without dropping. Otherwise all slots are nonempty and their leftmost requests, each of length at least ℓ, overlap each other; fairness gives the second inequality. Moreover, the only placed starts in $[m, a_{\max}]$ are these edges.

For a distinct edge a, define its *lower gap* as its distance to the next smaller distinct edge, or $+\infty$ for the smallest edge. At a drop, (7) simplifies to

$$t^* = a^* - \ell, \qquad a^* = \max\{a : \text{lower gap of } a \ge \ell - \alpha\}. \tag{9}$$

We call a^* the anchor. The smallest edge always qualifies. When $\ell \le \alpha$, every edge qualifies and the request is placed flush against a largest edge. Appendix B gives the endpoint argument establishing (9).

Lemma 2 (Window invariant)**.** *Fix any window of at most B consecutive placements, and let μ be the leftmost start before the window. Every drop in the window has anchor at least μ. If the window contains a drop, its total drop amount is at most the length of its last dropping request, and hence at most the first length in the window.*

Proof. Consider a drop of length ℓ in the window. An edge $v < \mu$ was created by a placement inside the window. At its first drop, the edges are all at most $\mu + \alpha$ by (8); they only decrease afterwards. The first drop is at least its request length minus α, hence at least $\ell - \alpha$. Any subsequent dropping placement moves the minimum still farther left. A non-dropping placement creating $v < \mu$ has length $\ell' \geq \ell$ and ends by an edge $a \leq \mu + \alpha$, so $v \leq a - \ell' \leq \mu - (\ell - \alpha)$. Thus every edge below μ lies at least $\ell - \alpha$ below it.

At most $B - 1$ placements in the window precede the current drop, so at least one slot retains its pre-window edge, which is at least μ. Let b be the smallest edge at least μ. Either b is the smallest edge of all, or its lower neighbor lies at most $\mu - (\ell - \alpha)$. In both cases b qualifies in (9); hence $a^* \geq b \geq \mu$. The very first placement satisfies the same conclusion with $\mu = a^* = 0$.

If the last drop in the window places length ℓ, the leftmost start after the window is $a^* - \ell$. Its total movement from μ is $\mu - a^* + \ell \leq \ell$. □

Theorem 3 (Instancewise static domination)**.** *For every instance, RA produces an α-fair schedule and*

$$T_{\mathrm{RA}} \leq L_{\mathrm{static}} \leq \left(2 - \frac{1}{B}\right) T^*. \tag{10}$$

The first inequality holds also for every prefix of the decreasing-length construction. If every request is longer than 2α, RA is optimal.

Proof. Partition the placements into consecutive windows of B requests. The window headed by $q_{(r-1)B+1}$ contributes at most that length by Lemma 2. Summing all drops gives $T_{\mathrm{RA}} \leq L_{\mathrm{static}}$; (3) gives the factor. A prefix is itself a run on the corresponding subset. In the stated regime, $L_{\mathrm{static}} = T^*$, so domination implies optimality. □

Proposition 4 (The cost of restricting schedules to rounds)**.** *For $B \geq 2$, $k \geq 2$, consider $(2k - 1, k^{2(B-1)})$ with $\alpha = k - 1$, where the exponent denotes multiplicity. Every policy that uses simultaneous-start groups of size at most B and drains a group before starting the next has makespan at least $3k - 1$. RA attains the optimal makespan $2k$.*

Proof. The group containing the longest request contributes $2k - 1$ and can contain at most $B - 1$ short requests. At least one length-k request remains outside it, contributing another k. In contrast, start the long request at one on one slot; on each other slot, start two length-k requests at zero and k. The co-active start differences are at most $k - 1$, so this schedule is fair and finishes at $2k$. The work bound is $\lceil (2Bk - 1)/B \rceil = 2k$. RA realizes these relative positions: on its negative time axis the long request starts at $-(2k - 1)$ and the two short requests on each other slot start at $-2k$ and $-k$. □

This separates backward interleaving from any reordering of independent static batches: the latter can have an asymptotic throughput loss of one third on a family where RA is optimal.

4 Sharper Approximation Guarantees

4.1 A Tight Constant on Structured Instances

Let the last drop place length ℓ, with anchor $-y$ and largest edge $-E$. The final makespan is $T_{\mathrm{RA}} = \ell + y$. If s is the number of distinct edges above the anchor, then $s \leq B - 1$. Writing $d = \ell - \alpha > 0$, consecutive edges above the anchor have gaps smaller than d, hence $y - E \leq sd$. Let μ be the leftmost start before the layer containing the drop. The window invariant gives $y \leq |\mu|$, while (8) gives $E \geq |\mu| - \alpha$. Also $E \geq \ell$, because the top-edge request has length at least ℓ and ends by zero.

If $T^* \geq D := E + \ell$, these observations yield two complementary bounds:

$$T_{\mathrm{RA}} \leq D + sd, \qquad T_{\mathrm{RA}} \leq \tfrac{3}{2} D - d. \tag{11}$$

Balancing them gives $T_{\mathrm{RA}}/T^* \leq 3/2 - 1/(2(s+1)) \leq c_B$, where $c_B := 3/2 - 1/(2B)$. The case $\ell \leq \alpha$ is simpler: $y = E$ and the same hypothesis makes RA optimal. It also gives the additive guarantee $T_{\mathrm{RA}} \leq T^* + \min\{\alpha, \ell\}$, but that hypothesis must not be dropped.

Theorem 5 (Sharp two-layer guarantee and an obstruction)**.** *For $B \geq 3$ and $n \leq 2B$, $T_{\mathrm{RA}} \leq c_B T^*$. The constant is exact for RA on this class. Specifically, for $B \geq 2$, $k \geq 1$, and*

$$q_j = (2B-1)k - (j-1)(k-1)\ (j \leq B), \quad q_{B+1} = Bk, \quad \alpha = (B-1)k, \tag{12}$$

RA has makespan $(3B-1)k$, whereas $T^* = 2Bk + B - 1$.

The missing branch in (11) has a useful interpretation. If $T^* < E + \ell$, the optimal assignment must separate requests of length at least E from other requests of length at least ℓ. A counting argument forces $T^* \geq 3\ell$. On two-layer runs, $y \leq q_1 \leq T^*$, so $T_{\mathrm{RA}} \leq \frac{4}{3}T^* \leq c_B T^*$ for $B \geq 3$. Appendix C supplies the count, including the first-layer snapping case, and proves the family formula. At $B = 2$ this route gives $4/3$, not $5/4$, for arbitrary two-layer instances.

The family is an obstruction for RA and certain length-greedy policies, not a hardness-of-approximation result for the optimization problem. An optimal schedule delays the longest request; an assignment-first backward construction can recover the optimum on the family. In particular, no inference that every individual numerical instance must attain this worst-case factor is justified.

4.2 An Improved Unconditional Bound

Theorem 6 (Unconditional and short-tail guarantees)**.** *For every $B \geq 2$ and every instance,*

$$T_{\mathrm{RA}} \leq C_B T^*, \qquad C_B := 2 - \frac{11B-2}{9B^2} < 2 - \frac{1}{B}. \tag{13}$$

If $n > B$ and $q_{B+1} \leq \alpha$, then

$$T_{\mathrm{RA}} \leq T^* + \left(1 - \frac{1}{B}\right) q_{B+1} \leq c_B T^*. \tag{14}$$

Here the proof must bypass the static benchmark, whose factor in (3) is tight. Put $c = 1 - 1/B$. Subtracting the blocking lower bound from a refined summation of the static layers gives

$$T_{\mathrm{RA}} - T^* \leq c \min\{q_1, q_{B+1} + \alpha\}. \tag{15}$$

The first B placements lie on different slots with initial depths $M_j = \max\{q_j, q_1 - \alpha\}$. Their total right-end idle is exactly Θ, already charged to the optimum by (4). Their blackouts are inert for all later placements. Therefore, when $q_{B+1} \leq \alpha$, the remaining requests run as ordinary LPT with initial loads M_j, proving (14).

The difficult case is $q_{2B+1} \leq \alpha < q_{B+1}$. Let $\varepsilon = q_{B+1} - \alpha$. Every gap created after the first layer is covered by blackout obstructions. Each obstruction can be charged at most once to each other slot. There are at most B relevant later long requests and $2B-1$ relevant earlier starts, so their total induced gap length is at most $(B-1)(3B-1)\varepsilon$. If a placement after the first two layers determines the makespan, this gives

$$T_{\mathrm{RA}} - T^* \leq c q_{2B+1} + c(3B-1)(q_{B+1} - \alpha). \tag{16}$$

Combine (15) and (16), use $q_{B+1} \leq T^*/2$ and $q_{2B+1} \leq T^*/3$, and balance the two affine expressions in α/T^* to obtain (13). If the first two layers determine the makespan, Theorem 5 and its $B = 2$ version apply. If instead $\alpha < q_{2B+1}$, (6) gives $T^* > q_1 + \alpha$; combining this with (15) gives the smaller factor $7/4 - 3/(4B)$. The full obstruction accounting and algebra appear in Appendix D.

The unconditional factor and the bad-family factor remain different: we do not claim that c_B holds on every instance. Nor does increasing α necessarily improve the greedy schedule, even though it improves the optimal objective by enlarging the feasible set.

4.3 Reading the guarantee along the tradeoff

For any capacity-respecting comparator s, Theorem 6 at $\alpha = D(s)$ gives

$$D(\mathrm{RA}(D(s))) \leq D(s), \qquad T_{\mathrm{RA}}(D(s)) \leq C_B T(s).$$

The parameterized algorithm thus approximates makespan without relaxing the comparator's disparity budget. On a finite budget grid, retaining the best schedule found at any smaller budget

makes makespan non-increasing and preserves the current-budget guarantee. This is schedule reuse, not monotonicity of raw RA. A logarithmic grid yields a polynomial-size $(C_B, 1+\varepsilon)$-approximate Pareto set, with a $(1+\varepsilon)$ disparity allowance; Appendix I proves this corollary and distinguishes it from exact-budget feasibility. The sharper last-drop certificate remains polynomially checkable (Appendix C). Streaming arrivals require a different information model; Appendix H gives an impossibility bound.

5 From a Backward Plan to an Executable Rule

A direct execution of nominal start times can waste capacity when requests finish earlier than forecast. Conversely, starting every request as soon as a slot is free can violate the progress budget. Our practical variant, RA-R, separates a backward plan from observation-based retiming. It is a snapshot implementation, not an online algorithm with an inherited adversarial guarantee.

A small fixed portfolio. First compute RA and two assignment-first plans. The latter assign requests to slots by LPT, then process the slots in decreasing or increasing total-load order; inside each slot they place requests backwards in decreasing length order using (7). Keep the plan with smallest predicted makespan. We call this three-plan portfolio RA-P2; it is a restricted, fixed-size alternative to trying many slot orders. Retaining RA means it inherits every RA makespan upper bound on its planning lengths.

Retiming from observations. Let u_i be a valid duration upper bound and $\bar{s}_i$ the chosen plan for u. Preserve the order of requests on each planned slot. If $\bar{s}_j - \bar{s}_i > \alpha$, retain the precedence $x_j \geq x_i + u_i$. If $|\bar{s}_j - \bar{s}_i| \leq \alpha$, retain the two start-proximity inequalities $|x_i - x_j| \leq \alpha$ while both requests are unfinished. At each completion or proposed-start event, fix starts already executed, constrain pending starts to be no earlier than the current iteration, replace completed durations by their observed values, and remove proximity constraints involving completed requests. The least feasible solution of these difference constraints gives the next admissions. No future realized duration is used by the controller.

Proposition 7 (A safe retiming certificate)**.** *If $o_i \leq u_i$ for every request,* RA-R *is capacity-respecting and α-fair, and $T_{\text{RA-R}}(o) \leq T_{\text{RA-P2}}(u)$. With $u = o$, it therefore inherits Theorems 3 and 6. For uncertain lengths this is a certificate against the upper-bound instance, not a C_B-approximation to $T^*(o; B, \alpha)$.*

Preserved slot precedences ensure capacity; each cross-slot pair is either non-overlapping or governed by a proximity constraint while co-active. Completed requests only relax the remaining system. The previous feasible continuation is therefore retained, and its least solution cannot move an unfinished request later. This proves the makespan certificate; Appendix E gives the induction. Under the tested error model $\widehat{o}_i = \mathrm{round}(o_i(1+\epsilon_i))$, $|\epsilon_i| \leq 0.2$, we use the rounding-safe upper bound $u_i = \lceil (\widehat{o}_i + 0.5)/0.8 \rceil$. This is a bounded-error assumption, not a learned prediction guarantee.

6 Numerical Study: Tradeoffs and Algorithmic Loss

The experiments address three questions: how closely RA approaches the same-budget optimum, which implementation components explain its gains, and how performance depends on workload heterogeneity. All executions use an independent event-driven logical simulator. Prefill, KV allocation, online arrivals, and hardware timing are excluded. Original trace results are supplemented by post-review component, exact-frontier, and synthetic studies; no parameter is retuned on these new outcomes.

Metrics and protocol. We report makespan, achieved disparity D, mean completion and waiting times, and token-weighted companion-progress exposure

$$\bar{E} = \frac{1}{W} \sum_t \sum_{i \in S(t)} \left(\max_{j \in S(t)} a_j(t) - a_i(t) \right).$$

Any $D \leq \alpha$ schedule satisfies $\bar{E} \leq (1 - 1/B)\alpha$ (Appendix I). This is a progress-based diagnostic, not latency or GPU cost. The primary second objective remains worst-case disparity, not its average.

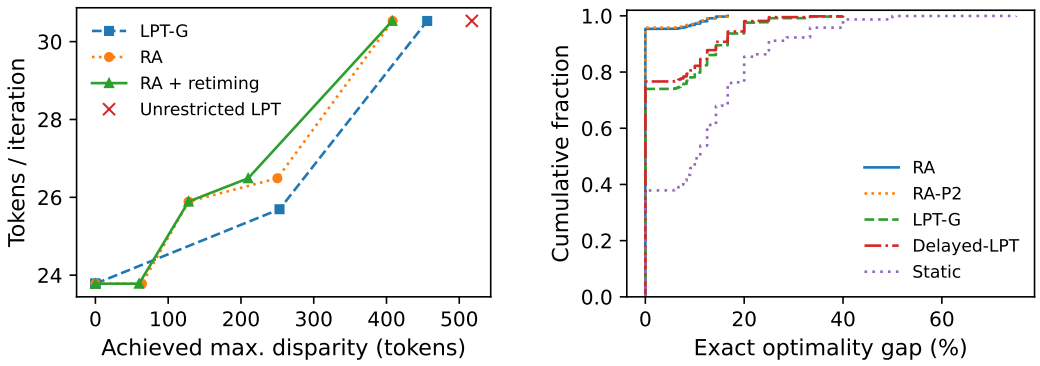

Figure 2: Left: achieved throughput–disparity tradeoffs on conversational snapshots, $B = 32$, exact lengths. Points are means at each tested budget, not claims of exact Pareto optimality; Static coincides with the zero-disparity LPT-G endpoint. Right: exact optimality-gap distributions on 10,360 exhaustive instances with $n > B$ and $0 < \alpha < q_1$.

The trace study reuses 24 paired 128-request snapshots from each of the Arxiv and Splitwise conversational prefixes distributed with Vidur (Agrawal et al., 2024a; Microsoft and Vidur Contributors, 2026). After the original development split and 4096-total-token filter, the pools contain only 150 and 143 requests; snapshots overlap substantially. We retain $B \in \{16, 32\}$, $\alpha \in \{0, 64, 128, 256, 512\}$, and the previously selected representative budget 128. Separately, four artificial length distributions generate 24 independent 128-request workloads each: uniform, lognormal, bimodal, and narrow. They use the same grid plus $\alpha = 1024$. These 96 draws are new synthetic workloads, not additional real traces. Exact lengths and the original signed 10–20% perturbations are tested with shared inputs across policies. Full distribution definitions, all outcomes, and paired 20,000-resample bootstrap intervals appear in Appendix J.

Comparators and attribution. Static, FIFO-G, SPT-G, and LPT-G use the same admission constraint. A delayed-LPT baseline reserves one slot for the longest predicted request, tries delays $0, \lfloor \alpha/2 \rfloor, \alpha$, and chooses the smallest predicted makespan before execution. Its observed-progress gate does not access future realized lengths. Unrestricted LPT is a tradeoff reference, not a same-budget competitor. We separately evaluate plain RA, the RA-P2 portfolio, and each with retiming. Under errors, fixed plans use the valid upper bounds from Section 5; retiming reacts only to completions.

Exact optima separate two different losses. An exhaustive dynamic program permitting intentional delay supplies 20,163 exact budget optima for $B \in \{2, 3, 4\}$, integer lengths up to five, and up to ten requests, including runs beyond two layers. Appendix J specifies both complete enumeration sets. Assignment enumeration independently supplies unconstrained optima. This distinguishes the policy cost $V(\alpha)/V(\infty)$ from the algorithmic loss $T_A/V(\alpha)$, rather than treating their product as a scheduling defect. On the 10,360-instance subset in Figure 2, RA attains the exact optimum on 95.37% of instances, with a 0.515% mean gap, versus 3.896% for LPT-G and 3.432% for delayed LPT. The largest observed RA gap is 16.67%. These bounded-length finite checks do not establish a stronger worst-case theorem; the separate tight-family experiment remains in Appendix F.

The backward rule produces the primary throughput gain. At conversational $B = 32$, $\alpha = 128$, plain RA already reproduces the 8.87% paired gain over LPT-G; the portfolio changes none of the 24 plans' makespans at this point. Delayed LPT coincides with LPT-G here. Across all 960 trace scenarios, RA-P2 improves the nominal RA makespan only nine times. This does not make the portfolio redundant: it resolves the constructed obstruction in Theorem 5 and can help at other budgets. The main trace gain nevertheless comes from the analyzed backward construction.

Retiming has a distinct effect. With exact lengths, it leaves throughput at 25.89 but reduces mean completion from 672.1 to 629.5 and $\bar{E}$ from 73.34 to 21.32 (Table 1). Under perturbed lengths, retiming raises throughput from 21.16 to 23.73, recovering the original 4.63% paired gain over

Table 1: Conversational snapshots, $B = 32$, $\alpha = 128$, 24 paired replications. Throughput is tokens per logical iteration; completion is mean completion in iterations; $\bar{E}$ is average companion-progress exposure in tokens. The retimed portfolio is the original RA-R policy.

	Exact lengths			10–20% errors	
Policy	Throughput	$\bar{E}$	Completion	Throughput	Completion
LPT-G / delayed LPT	23.78	0.00	1029.5	22.68	1057.6
RA	25.89	73.34	672.1	21.16	803.3
RA-P2	25.89	73.34	672.1	21.16	803.3
RA + retiming	25.89	21.32	629.5	23.73	703.6
RA-P2 + retiming	25.89	21.32	629.5	23.73	703.6
Unrestricted LPT	30.53	116.14	863.2	30.97	859.3

Table 2: Independent synthetic workloads at the unchanged representative budget $\alpha = 128$, $B = 32$. Paired throughput gains of plain RA with retiming over LPT-G, in percent; brackets are 95% bootstrap intervals across 24 independently generated workloads per family. All budgets are reported in Appendix J.

Length distribution	Exact lengths	10–20% errors
Uniform	3.99 [2.41, 5.56]	0.24 [-0.38, 1.10]
Lognormal	4.00 [3.39, 4.88]	1.24 [0.67, 1.77]
Bimodal	4.17 [3.74, 4.60]	1.42 [0.94, 1.92]
Narrow	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]

LPT-G. These exposure changes are observations, not a monotonicity theorem for retiming. Static batching has zero exposure but lower throughput; unrestricted LPT is faster but has larger maximum progress disparity. Figure 2 therefore presents operating choices rather than implying that one policy dominates every objective.

Heterogeneity and prediction slack determine the useful regime. The independent synthetic draws support the mechanism beyond the small trace pools. At the unchanged budget 128, plain RA with retiming gains about 4% on uniform, lognormal, and bimodal lengths, but has no gain on narrow lengths (Table 2). The full grid is essential: at $B = 32$, $\alpha = 512$, lognormal gains are 22.13% exact and 7.42% with errors, whereas uniform and bimodal noisy gains are -14.53% and -14.57%. Conservative bounds can outweigh the scheduling advantage: at this budget they move 20 of 24 uniform and 19 of 24 conversational workloads out of Theorem 6's short-tail regime, but no lognormal workload crosses that boundary. This is a structural diagnostic, not a sufficient explanation of every regression. The mechanism is regime-dependent, not universally dominant. All 193,944 added scheduler evaluations pass independent capacity and, where applicable, progress checks. Confidence intervals on the original trace replay remain conditional on its finite pools; synthetic intervals describe the specified generators, not real-serving populations.

7 Limitations and Conclusion

The paper studies how efficiently a chosen progress-homogeneity requirement can be implemented. Backward admission planning and the window invariant give instancewise and worst-case guarantees, while exact-frontier comparisons distinguish the constraint's intrinsic cost from the algorithm's loss. Component experiments attribute the primary gain to RA and show how retiming changes execution efficiency and exposure. The results support a regime-dependent scheduling mechanism, not universal service fairness or GPU speedups. The unrestricted sharp approximation factor and the integration of this discipline with calibrated latency, prompt processing, memory pressure, and streaming arrivals remain open directions.

AI USE STATEMENT

Generative AI assisted mathematical checking and writing revision. Responsibility for the claims, code, and final submission remains with the authors.

REFERENCES

Amey Agrawal, Nitin Kedia, Jayashree Mohan, Ashish Panwar, Nipun Kwatra, Bhargav S. Gulavani, Ramachandran Ramjee, and Alexey Tumanov. Vidur: A large-scale simulation framework for LLM inference. In *Proceedings of Machine Learning and Systems*, volume 6, 2024a.

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput–latency tradeoff in LLM inference with Sarathi-Serve. In *18th USENIX Symposium on Operating Systems Design and Implementation*, pp. 117–134, 2024b.

Ruicheng Ao, Gan Luo, David Simchi-Levi, and Xinshang Wang. Optimizing LLM inference: Fluid-guided online scheduling with memory constraints, 2025. Version 4, revised June 2026.

Lequn Chen. Dissecting batching effects in GPT inference. Technical blog, 2023. URL `https://le.qun.ch/en/blog/2023/05/13/transformer-batching/`. May 13, 2023.

Zixi Chen, Yinyu Ye, and Zijie Zhou. Adaptively robust LLM inference optimization under prediction uncertainty, 2025.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2):416–429, 1969.

Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *Journal of the ACM*, 34(1):144–162, 1987.

Patrick Jaillet, Jiashuo Jiang, Konstantina Mellou, Marco Molinaro, Chara Podimata, and Zijie Zhou. Online scheduling for LLM inference with KV cache constraints, 2025. Version 5, revised January 2026.

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 *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

George Mavrotas. Effective implementation of the epsilon-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 213(2):455–465, 2009. doi: 10.1016/j.amc.2009.03.037.

Microsoft and Vidur Contributors. Vidur: Source code, profiling data, and request traces, 2026. URL `https://github.com/microsoft/vidur`. Source inspected September 17, 2026. Source inspection is not native simulator execution.

Ying Sheng, Shiyi Cao, Dacheng Li, Banghua Zhu, Zhuohan Li, Danyang Zhuo, Joseph E. Gonzalez, and Ion Stoica. Fairness in serving large language models. In *18th USENIX Symposium on Operating Systems Design and Implementation*, pp. 965–988, 2024.

Meixuan Wang, Yinyu Ye, and Zijie Zhou. LLM serving optimization with variable prefill and decode lengths, 2025. Version 4, revised June 2026.

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 *16th USENIX Symposium on Operating Systems Design and Implementation*, pp. 521–538, 2022.

A Structural Facts and Lower Bounds

A.1 Integral schedules and complexity

The model is stated with integer starts. Allowing real starts does not change its optimum when all lengths and α are integers. Indeed, fix any real feasible schedule and a slot assignment. Replace every s_i by $\lfloor s_i \rfloor$. A same-slot precedence $s_j \geq s_i + o_i$ is preserved because o_i is integer. Every originally disjoint pair remains disjoint for the same reason. Every originally overlapping pair has $|s_i - s_j| \leq \alpha$, hence $|\lfloor s_i \rfloor - \lfloor s_j \rfloor| \leq \alpha$ because α is integer. No new overlap violates fairness, and the makespan does not increase. There is at least one feasible integer schedule (serial execution), and its nonempty set of integer makespans has a minimum. This minimum is no greater than any real feasible makespan, proving integrality and attainment.

If $\alpha \geq q_1$, intersecting pairs automatically satisfy the budget: for $s_i \leq s_j < s_i + o_i$, $s_j - s_i < o_i \leq \alpha$. Thus $P\|C_{\max}$ is a special case. Its variable-machine strong NP-hardness and two-machine NP-hardness transfer (Garey & Johnson, 1979). An FPTAS with $\varepsilon < 1/W$ would return a feasible makespan less than $T^* + 1$, since $T^* \leq W$. Rounding starts down makes this an exact integer optimum in time polynomial in the unary input size, contradicting strong NP-hardness unless P=NP. This does not assert an approximation-hardness threshold such as $3/2$; the unconstrained special case itself admits a PTAS (Hochbaum & Shmoys, 1987).

A.2 Static batching and the rigid regime

For any partition into groups of size at most B, order the group maxima as $M_1 \geq M_2 \geq \cdots$. Among the largest $(r-1)B + 1$ requests, at least r groups are needed, so $M_r \geq q_{(r-1)B+1}$. Consequently the sum of group maxima is at least L_{static}, with equality for consecutive sorted groups. Also

$$\sum_{r \geq 2} q_{(r-1)B+1} \leq \frac{1}{B} \sum_{i=2}^{n} q_i,$$

because the B indices $(r-2)B + 2, \ldots, (r-1)B + 1$ each have length at least the corresponding layer head. This proves (3). It is tight: one request of length B and $B(B-1)$ unit requests have static makespan $2B - 1$, whereas at $\alpha \geq B - 1$ one long request on one slot and successive unit requests on the other slots finish in B iterations.

Consider only the requests of length greater than 2α. If i overlaps j and j overlaps k, then $|s_i - s_k| \leq 2\alpha$. The earlier of i, k has length greater than 2α, so they overlap too. Thus overlap is an equivalence relation on these requests. Each class is a clique of intervals and therefore shares a common time point; capacity bounds its size by B. The union spans of different classes are disjoint. Each span has length at least its largest request, so the partition argument gives

$$T^* \geq L_{\text{static}}(\{o_i : o_i > 2\alpha\}; B).$$

When all requests meet the threshold, this equals L_{static}, which is achieved by static batching. For $\alpha = 0$, intersecting requests must have exactly the same start, giving the same conclusion directly.

A.3 Blocking and stacking

Fix a longest request, starting at s_1. In its blackout $(s_1 + \alpha, s_1 + q_1)$, no request starts. On another slot, at most one request has positive processing inside this interval, namely one active at its left endpoint. If its length is x, that slot is idle for at least $(q_1 - \alpha - x)_+$. The $B - 1$ requests that can cover these slots are distinct and are not the longest request. Giving the slots the $B - 1$ largest remaining lengths only decreases the idle bound. Since total slot-time is work plus idleness, $BT^* \geq W + \Theta$. When $q_1 \leq \alpha$ the statement reduces to the work bound.

For stacking, suppose $q_1 > \alpha$; otherwise (5) is immediate from $T^* \geq q_1$. Every request is early, with start at most $s_1 + \alpha$, or late, with start at least $s_1 + q_1$. Consider the $B + 1$ longest requests. If one is late, $T \geq q_1 + q_{B+1}$. Otherwise two share a slot, and the second cannot start before q_{B+1}; hence $s_1 + \alpha \geq q_{B+1}$ and $T \geq q_1 + q_{B+1} - \alpha$.

For (6), apply the same dichotomy to the $B + k$ largest requests. A late member gives $T \geq q_1 + q_{B+k}$. If another member uses the longest request's slot, it must precede that request, also giving this bound.

Otherwise the longest request is alone on its slot among this subset. The remaining $B + k - 1$ jobs share $B - 1$ slots, so one slot has at least $1 + \lceil k/(B-1) \rceil$ of them. Its last start is at least $\lceil k/(B-1) \rceil q_{B+k}$ and at most $s_1 + \alpha$. Combining the alternatives proves the strengthened bound. Finally, among $kB + 1$ requests some slot holds $k + 1$, yielding $T^* \geq (k+1)q_{kB+1}$; this gives $q_{B+1} \leq T^*/2$ and $q_{2B+1} \leq T^*/3$ when the indices exist.

B Details of the Backward Geometry

B.1 Existing starts are safe snapping points

Suppose an old request has start s_0 and length $o_0 \geq \ell$. A new request of length ℓ can start at s_0 on any slot on which it fits. The point s_0 lies in no old blackout because the partial schedule is feasible. The new blackout is contained in the old request's blackout, so it contains no old start either. This establishes the snapping fact used to rule out a drop when some slot has room for $[m, m + \ell)$.

B.2 Where a drop lands

At a noninitial drop all slots are nonempty and $a_{\max} < m + \ell$. Their leftmost requests have lengths at least ℓ and mutually overlap, establishing (8). A non-leftmost start on a slot is at least $a_j + \ell_j \geq m + \ell > a_{\max}$. Thus the old starts in $[m, a_{\max}]$ are precisely the distinct left edges.

Every candidate $t \leq a_{\max} - \ell$ is below m, so it lies in no old blackout. Its only possible violation is that $(t + \alpha, t + \ell)$ contains an old start. Starts above $a_{\max}$ cannot lie there; therefore the forbidden set for this placement is the union of $(a - \ell, a - \alpha)$ over the left edges a. The largest feasible point below the fit limit is an endpoint $a - \ell$, with the fit limit itself included when feasible. Such a point is excluded exactly when some smaller edge a' has $0 < a - a' < \ell - \alpha$. Equivalently, the lower gap of a must be at least $\ell - \alpha$. This proves (9). When $\ell \leq \alpha$ the new blackout is empty and the largest edge qualifies. At the very first placement the unique edge value is zero and the anchor is zero.

B.3 First-layer positions

For $n \geq B$, the first B placements use distinct slots and have starts

$$e_j = -M_j, \qquad M_j = \max\{q_j, q_1 - \alpha\}, \quad j \leq B. \tag{17}$$

Induct on j. An already used slot $i < j$ admits at most $-M_i - q_j < -M_j$, so it suffices to identify the latest feasible start on an unused slot. If $q_j \geq q_1 - \alpha$, the fit point $-q_j$ is feasible. Its own blackout contains no old start because $-M_i \leq -q_i \leq -q_j$. Membership in an old blackout would require $q_j < M_i - \alpha \leq q_1 - \alpha$, a contradiction. If $q_j < q_1 - \alpha$, the longest request's blackout excludes every point in $(-(q_1 - \alpha), 0)$; the latest candidate is therefore $-(q_1 - \alpha)$. It avoids all old blackouts because their left endpoints are at least $-q_1 + \alpha$, and its own blackout contains no old start because every $M_i \geq q_1 - \alpha$. This proves (17). In particular placements $2, \ldots, B$ do not drop and the minimum after layer one is $-q_1$.

After layer one, all edges are at most $-M_B \leq -(q_1 - \alpha)$ and only decrease. All nonempty first-layer blackouts lie in $[-(q_1 - \alpha), 0)$. Every later fit point, or a point below it, is strictly to the left of this region. This proves the inertness used in the short-tail and obstruction arguments. The only idle introduced in layer one is the right-end gap $M_j - q_j = (q_1 - \alpha - q_j)_+$, which totals Θ.

C The Sharp Constant and Its Certificates

C.1 Conditional last-drop analysis

Let $\ell, -y, -E, \mu, s$ be as in the main text. The last drop starts at $-y - \ell$, so $T_{\mathrm{RA}} = y + \ell$. The window invariant gives $y \leq |\mu|$. At the drop, (8) gives $E \geq |\mu| - \alpha$, and the leftmost request on the largest-edge slot gives $E \geq \ell$. If $d = \ell - \alpha > 0$, the s distinct edges above the anchor fail the lower-gap test. Summing their gaps gives $y - E \leq sd$. Thus $D = E + \ell$ satisfies $D \geq 2\ell$ and $D \geq |\mu| + d$, and

$$T_{\mathrm{RA}} \leq D + sd, \qquad T_{\mathrm{RA}} \leq \ell + |\mu| \leq \tfrac{3}{2}D - d.$$

Under $T^* \geq D$, divide by D and balance $1 + s\rho$ with $3/2 - \rho$, where $\rho = d/D$. Their minimum is at most $3/2 - 1/(2(s+1)) \leq c_B$. If $d \leq 0$, every edge qualifies, $y = E$, and $T_{\text{RA}} = D \leq T^*$.

For the additive claim, the minimum current edge m satisfies $|m| \leq E + \alpha$ and $|m| < E + \ell$ by (8). The anchor has $y \leq |m|$, hence $T_{\text{RA}} \leq E + \ell + \min\{\alpha, \ell\} \leq T^* + \min\{\alpha, \ell\}$. This proof uses $T^* \geq E + \ell$ essentially; it is not an unconditional price-of-fairness bound.

C.2 The crowding alternative and two layers

Suppose $T^* < E + \ell$ and define $p = \#\{i : q_i \geq E\}$, $Q = \#\{i : q_i \geq \ell\}$. In any slot assignment with makespan smaller than $E+\ell$, two length-at-least-E requests cannot share a slot because $E \geq \ell$. Nor can such a request share a slot with any other request of length at least ℓ. The p largest requests therefore occupy distinct slots, leaving the remaining $Q - p$ relevant requests on $B - p$ slots. Since the last drop has index $k > B$, $Q \geq k > B$; this excludes $p \geq B$. Pigeonhole and average-work bounds give

$$T^* \geq \Gamma := \max\left\{ \left\lceil \frac{Q-p}{B-p} \right\rceil \ell, \frac{q_{p+1} + \cdots + q_Q}{B-p} \right\}. \tag{18}$$

If $k > 2B$, immediately $Q > 2B - p$, so the ceiling is at least three. Suppose instead the last drop is in layer two. The stacking bound implies $T^* \geq q_1 + \ell - \alpha$, and the failed condition forces $E > q_1 - \alpha$. Thus $M_j \geq E$ in (17) holds exactly when $q_j \geq E$. The $B - p$ first-layer edges with $j > p$ would lie strictly above the present largest edge $-E$ unless overwritten. There are only $k - B - 1$ preceding second-layer placements, each changing one slot. Hence $B - p \leq k - B - 1$, or $Q \geq k > 2B - p$. Again $T^* \geq 3\ell$.

On a two-layer run, the window invariant also gives $y \leq q_1 \leq T^*$. Therefore $T_{\text{RA}} = \ell + y \leq (4/3)T^*$. Combining this with the conditional c_B bound proves $T_{\text{RA}} \leq \max\{c_B, 4/3\}T^*$ on every run whose last drop is in layer two. If $n \leq B$, or the only drop is the first, $T_{\text{RA}} = q_1 = T^*$. Thus for all $n \leq 2B$ and $B \geq 3$ the factor is c_B, proving the upper-bound part of Theorem 5.

C.3 The tight family

For (12), $q_B = Bk + B - 1$ and $q_{B+1} = Bk$. Any B-slot assignment of $B + 1$ requests places two together, so $T^* \geq q_B + q_{B+1} = 2Bk + B - 1$. A matching fair schedule runs q_{B+1} at zero and q_B at Bk on one slot, and starts every q_j, $j < B$, at k on the other slots. The distinct start differences for overlapping pairs are k and $(B-1)k$, both at most α; the jobs starting at 0 and Bk do not overlap.

The first layer of RA is unsnapped, with edges $-q_1, \ldots, -q_B$. For the next request $\ell = Bk$, adjacent distinct edge gaps are $k - 1 < \ell - \alpha = k$. The smallest edge is the only qualifying anchor, including the all-equal-edge case $k = 1$. The drop therefore starts at $-q_1 - Bk$, giving $T_{\text{RA}} = (3B-1)k$. Its ratio tends to c_B as $k \to \infty$, establishing sharpness on the two-layer class for $B \geq 3$.

For $k \geq 2$, every schedule shorter than $(3B-1)k$ must delay the longest request until at least k. Some slot holds two requests, x then z. If x is the longest, the claimed threshold already follows. If z is the longest, it starts after a length-at-least-Bk request. Otherwise $s_z \geq Bk$. Were $s_1 < k$, the start s_z would be strictly after $s_1 + \alpha$ and would therefore have to be at least $s_1 + q_1$, again giving makespan at least $(3B-1)k$.

A second obstruction concerns forcing all B longest requests to end at the makespan T. The remaining request, of length Bk, must end before one of them begins. If it starts at s, put $a = T - s - Bk \geq q_B$. Any $q_i \in (a, a + k)$ would overlap it with a start difference larger than α, which is infeasible. If $a \geq q_1$, already $T \geq q_1 + Bk$. Otherwise the successive gaps $q_i - q_{i+1} = k - 1$ ensure that such a q_i exists. Hence this endpoint commitment also costs at least $(3B-1)k$. Neither argument is an approximation-hardness result for all algorithms.

Snapping need not create a uniform gap. For $B = 3$, $q = (5k+1, 4k+1, 3k, 3k)$ and $\alpha = 2k-1$, $k \geq 2$, the third first-layer placement is snapped to $-(3k+2)$. The fourth placement is a total drop, giving $T_{\text{RA}} = 8k+1$. Two $3k$ requests can run successively at 0 and $3k$ on one slot, while the others start at $k + 1$. This meets the stacking lower bound $T^* = 6k + 2$, and the ratio still tends to $4/3$. Thus a strict bound on every finite snapped instance does not imply that only unsnapped instances approach the sharp constant.

C.4 Polynomial certificate

The last-drop argument also gives a polynomially checkable certificate. Let L be any positive, polynomial-time lower bound on T^*, for example the maximum of the elementary, cohort, blocking, and stacking bounds, rounded up. At the last noninitial drop define $p = \#\{i : q_i \geq E\}$, $Q = \#\{i : q_i \geq \ell\}$, and, only when $p < B$,

$$\Gamma = \max\left\{ \left\lceil \frac{Q-p}{B-p} \right\rceil \ell, \frac{\sum_{i=p+1}^{Q} q_i}{B-p} \right\}.$$

Then $T_{\mathrm{RA}} \leq c_B T^*$ is certified by any one of

$$p \geq B, \qquad L \geq E + \ell, \qquad \left[p < B \text{ and } \frac{\ell}{\Gamma} + \frac{y}{L} \leq c_B \right]. \tag{19}$$

The first two conditions imply the hypothesis of (11). In the remaining branch, the crowding argument gives $T^* \geq \Gamma$, while $T_{\mathrm{RA}} = \ell + y$ and $T^* \geq L$. This is a sufficient certificate, not a complete characterization. Crucially, L need not contain the exact unconstrained optimum, whose computation would make the advertised certificate non-polynomial.

If $p \geq B$, the assignment argument preceding (18) makes $T^* < E + \ell$ impossible, because $Q > B$. If $L \geq E + \ell$, the same condition follows from $T^* \geq L$. In either case the conditional proof applies. For the third test of (19), either that condition holds or (18) gives $T^* \geq \Gamma$; therefore

$$T_{\mathrm{RA}} = \ell + y \leq \left(\frac{\ell}{\Gamma} + \frac{y}{L} \right) T^* \leq c_B T^*.$$

The test is polynomial when L is polynomially evaluable. The exact unconstrained optimum is not part of the implementable lower bound. The denominator $B - p$ is used only in the branch $p < B$.

D Proof of the Improved Unconditional Factor

Let $c = 1 - 1/B$, $n > B$, and Θ, M_j be as above. The case $n \leq B$ is optimal and needs no analysis.

D.1 The additive estimate

For static layers $r \geq 3$, charge each head to the B requests with indices $(r-2)B+2, \ldots, (r-1)B+1$. These sets are disjoint and lie after index $B + 1$. Thus

$$T_{\mathrm{RA}} \leq L_{\mathrm{static}} \leq q_1 + q_{B+1} + \frac{1}{B} \sum_{i \geq B+2} q_i.$$

Subtracting $T^* \geq (W + \Theta)/B$ gives

$$T_{\mathrm{RA}} - T^* \leq c(q_1 + q_{B+1}) - \frac{1}{B} \sum_{j=2}^{B} M_j.$$

Each $M_j \geq \max\{q_{B+1}, q_1 - \alpha\}$, so the right side is at most $c \min\{q_1, q_{B+1} + \alpha\}$, proving (15).

D.2 The short-tail regime

If $q_{B+1} \leq \alpha$, every tail request has no blackout. First-layer blackouts are inert by Appendix B, so every tail placement sits flush against a largest edge. This is LPT list scheduling with initial slot loads $M_1, \ldots, M_B$. If a slot attaining the largest final load receives no tail job, the makespan is at most q_1. Otherwise, let $\lambda \leq q_{B+1}$ be its last assigned job. Its previous load is at most the average load at that assignment, hence at most $(W + \Theta - \lambda)/B$. Therefore

$$T_{\mathrm{RA}} \leq \max\left\{ q_1, \frac{W + \Theta}{B} + c q_{B+1} \right\} \leq T^* + c q_{B+1} \leq c_B T^*.$$

D.3 Charging internal gaps to obstructions

A placement of length λ on a slot with previous edge a creates a gap if its chosen start $t < a - \lambda$. Every point of $(t, a - \lambda]$ is then forbidden. For a previously placed request x with start s_x, let $X_x = (s_x + \alpha, s_x + o_x)$, empty if $o_x \le \alpha$. For an old start s, let $Y_s^\lambda = (s - \lambda, s - \alpha)$, empty if $\lambda \le \alpha$. The gap is covered by the union of these intervals, so its length is at most the sum of their intersection lengths with $(t, a - \lambda]$.

An involved old request lies on another slot: on the current slot its start is at least a, whereas the gap's candidate starts are at most $a - \lambda$. Moreover, a pair consisting of one old request and another slot can be involved through X_x at most once. If it is involved at this placement, t is feasible and lies to the left of some point of X_x, so $t \le s_x + \alpha$. All later candidate gaps on this slot lie strictly below t, and cannot meet X_x again. Similarly, involvement through Y_s^λ gives $t \le s - \lambda$. A later placement of length λ' has fit limit at most $t - \lambda' \le s - \lambda - \lambda' < s - \lambda'$, below the next interval $Y_s^{\lambda'}$. A start–slot pair is therefore charged at most once through this family as well.

Suppose $q_{2B+1} \le \alpha < q_{B+1}$ and let $h = \#\{i : q_i > \alpha\}$, so $B < h \le 2B$. First-layer blackouts cannot cause later gaps. Thus at most $h - B$ requests contribute X-obstructions, each of length at most $\varepsilon = q_{B+1} - \alpha$. A Y-obstruction occurs only during a later long placement, and there are at most $h - 1$ earlier starts involved; each interval has length at most ε. Each source can be charged only to the other $B - 1$ slots. The total internal gap length I after layer one satisfies

$$I \le \varepsilon\big[(h - B)(B - 1) + (h - 1)(B - 1)\big] \le (B - 1)(3B - 1)\varepsilon.$$

The right-end idle Θ is separate and was already charged to the optimum.

If the last drop has index at most $2B$, the final makespan equals that of the corresponding two-layer prefix. Otherwise its length $\ell \le q_{2B+1} \le \alpha$ makes the anchor a largest edge. Let E be the smallest slot depth immediately before it. Summing the occupied spans of all slots gives

$$BE \le W - \ell + \Theta + I.$$

The still-unplaced work has merely been overcounted on the right. Hence

$$T_{\mathrm{RA}} = \ell + E \le T^* + c\ell + I/B \le T^* + cq_{2B+1} + c(3B - 1)(q_{B+1} - \alpha),$$

which proves (16) and the alternative needed when only the first two layers determine the makespan.

D.4 Combining the regimes

If $\alpha \ge q_{B+1}$, use the short-tail bound. If $\alpha < q_{2B+1}$, apply (6) with $k = B$ to obtain $T^* \ge q_1 + q_{2B} > q_1 + \alpha$. Then

$$T_{\mathrm{RA}} - T^* \le c \min\{T^* - \alpha, T^*/2 + \alpha\} \le \tfrac{3}{4} c T^*.$$

Thus the factor is at most $7/4 - 3/(4B) \le C_B$.

In the remaining regime, a last drop within two layers gives factor at most $\max\{c_B, 4/3\} \le C_B$. Otherwise put

$$x = q_{B+1}/T^* \le \tfrac{1}{2}, \quad a = \alpha/T^* < x, \quad K = c(3B - 1).$$

The additive and gap estimates give

$$\frac{T_{\mathrm{RA}} - T^*}{T^*} \le \min\{c(x + a), c/3 + K(x - a)\}.$$

The first expression increases in a and the second decreases. Their intersection is at $a = (c/3 + (K - c)x)/(c + K)$, and the value there is $c(2Kx + c/3)/(c + K)$. A minimum of these affine expressions is no larger than its global peak, even if the intersection lies outside the permitted range of a. Using $x \le 1/2$ bounds that peak by

$$c\frac{K + c/3}{c + K} = c\frac{9B - 2}{9B}.$$

Adding one gives $C_B = 2 - (11B - 2)/(9B^2)$. Finally, direct subtraction verifies that $C_B \ge c_B$, $C_B \ge 4/3$, and $C_B \ge 7/4 - 3/(4B)$ for $B \ge 2$. This completes Theorem 6.

E Observation-Based Retiming

The retiming result is an additional implementation statement, separate from the original RA analysis. Fix an α-fair plan $(\bar{s}, z)$ for upper lengths u, where z_i is the assigned slot. Initially form a system of lower difference constraints $x_v \geq x_u + w_{uv}$. Every pair on one slot receives its planned-order precedence, with weight the earlier request's upper length. Every ordered cross-slot pair with $\bar{s}_j - \bar{s}_i > \alpha$ receives precedence $x_j \geq x_i + u_i$. All remaining pairs receive the two proximity edges of weight $-\alpha$. The nominal plan satisfies this system. Start variables are bounded below by zero.

At an event iteration t, completed requests have their true, observed durations substituted for their upper lengths. Constraints into completed requests and proximity edges out of completed requests are removed; started requests retain their actual fixed starts, and pending starts are bounded below by t. The componentwise least feasible vector can be obtained by longest-path relaxation from these lower bounds, with a check that no relaxation increases a fixed start. The implementation stops explicitly on an upper-bound violation rather than silently continuing under an invalid contract.

Induction on events. Assume all actual durations are at most their upper bounds. At initialization the nominal plan is a feasible witness. Execute all starts at the current least solution's earliest pending time. Fixing those starts preserves feasibility. Advance to the earlier of the next proposed start and the next actual completion. In a native system only the latter event is observed; the environment, not the controller, determines its time. The new time lower bound is no larger than any previously proposed pending start. A completion only decreases a duration bound and removes constraints. Thus the previous feasible continuation remains feasible, and the new least solution cannot move any pending start later than that continuation. Every executed start is therefore no later than its initial planned counterpart, giving $s_i + o_i \leq \bar{s}_i + u_i$.

Slot precedences preserve capacity. For a cross-slot pair with a precedence, actual execution remains non-overlapping. For a proximity pair, if they overlap, the constraint is present until the second request starts and fixes their start difference within α. Removing it when one completes cannot change that fact retroactively. This proves feasibility and $T_{\text{RA-R}} \leq T_{\text{RA-P2}}(u)$.

The error experiment uses $\widehat{o}_i = \max\{1, \text{round}(o_i(1+\epsilon_i))\}$ with $|\epsilon_i| \leq 0.2$. Rounding implies $0.8o_i \leq \widehat{o}_i + 0.5$, hence

$$u_i = \left\lceil \frac{\widehat{o}_i + 0.5}{0.8} \right\rceil = \left\lfloor \frac{10\widehat{o}_i + 12}{8} \right\rfloor \geq o_i.$$

The formula accounts for integer rounding rather than using an unsafe bare multiplier. It does not cover arbitrary outliers, and the current integration prototype does not claim an analyzed fallback for such outliers. Scaling a length bound also does not transfer a makespan factor at fixed α: time scaling changes progress gaps. The certificate against $T_{\text{RA-P2}}(u)$ is therefore intentionally not restated as a multiplicative guarantee against $T^*(o; B, \alpha)$.

F Complete Numerical Results and Simulator Scope

F.1 Data provenance and execution status

The included CSVs are compact copies of the public Vidur Arxiv and Splitwise conversational trace prefixes, obtained from an earlier reproducibility archive. The public raw-file schemas and leading entries were checked against the upstream repository. Source URLs and SHA-256 checksums are recorded in the package. These are finite prefixes, not representative random samples of the complete trace collections. The Arxiv held-out pool has mean output length 293.1 tokens and maximum 3480; the conversational pool has mean 286.1 and maximum 594. The simulation uses only the output lengths as contiguous decode work. Prompt lengths are used for the 4096-token filter and for the separately labeled sensitivity exercise, not for the logical scheduling clock.

The core simulator was executed in Python with NumPy and a Numba-compiled placement/retiming kernel. It records 9,120 policy-run outcomes, corresponding to 960 workload/parameter/information scenarios. The environment chooses actual completion events; the controller sees predictions, already executed starts, and completion observations only. A separate causal controller, which has no true-length argument, matches the compiled retiming kernel in 100 random API-equivalence tests.

Table 3: Complete paired throughput-gain matrix for RA-R relative to LPT-G, in percent. Parentheses contain 95% bootstrap intervals. Negative values are regressions.

Trace	B	α	Exact lengths	10–20% perturbations
Arxiv	16	0	0.00 (0.00, 0.00)	0.00 (0.00, 0.00)
		64	0.00 (0.00, 0.00)	-0.03 (-0.08, 0.00)
		128	0.54 (0.48, 0.61)	-1.07 (-1.16, -0.97)
		256	6.75 (6.61, 6.91)	-1.22 (-2.26, -0.16)
		512	7.67 (7.52, 7.82)	-1.15 (-2.56, 0.20)
Arxiv	32	0	0.00 (0.00, 0.00)	0.00 (0.00, 0.00)
		64	0.00 (0.00, 0.00)	-0.03 (-0.07, -0.01)
		128	3.67 (3.08, 4.18)	0.37 (0.24, 0.50)
		256	6.70 (6.54, 6.88)	0.81 (-0.42, 2.04)
		512	6.74 (6.32, 7.17)	0.53 (-1.43, 2.43)
Conversation	16	0	0.00 (0.00, 0.00)	0.00 (0.00, 0.00)
		64	0.04 (0.00, 0.12)	-0.02 (-0.06, 0.00)
		128	4.82 (4.60, 5.04)	0.41 (0.15, 0.67)
		256	0.29 (0.25, 0.33)	-3.29 (-3.52, -3.08)
		512	0.21 (0.15, 0.29)	-10.77 (-11.71, -9.82)
Conversation	32	0	0.00 (0.00, 0.00)	0.00 (0.00, 0.00)
		64	0.00 (0.00, 0.00)	0.00 (0.00, 0.00)
		128	8.87 (8.78, 8.96)	4.63 (3.44, 5.72)
		256	4.81 (3.32, 6.52)	1.00 (0.03, 1.95)
		512	0.00 (0.00, 0.01)	-10.92 (-12.23, -9.58)

Each returned schedule is checked using the original active-set capacity and progress definitions, independently of its placement or retiming constraints.

The native Vidur package was not available in this environment, and attempts to retrieve it and its dependencies failed at network resolution. No native end-to-end Vidur result is claimed. A separate `run_native_vidur.py` adapter is included for an installed Vidur checkout. It calls native Request/Batch entities and the execution-time predictor for a decode-only checkpoint replay; it is source-inspected and syntax-checked, but was not executed end-to-end. It does not implement the full upstream online-arrival, prefill, or KV-allocation pipeline. The main numerical tables use neither this adapter nor its nonexistent output.

F.2 All matched-budget gains

Table 3 reports every point of the fixed grid. Each entry is the mean paired throughput gain $100(T_{\mathrm{LPT-G}}/T_{\mathrm{RA\text{-}R}} - 1)$, followed by its 95% bootstrap interval. These are interval estimates for resampling from the particular finite pools. No multiple-comparison correction or population-level significance claim is made. Larger-budget and noisy-length regressions are retained. The raw CSV also includes every baseline, actual progress gap, utilization, completion metric, and prototype runtime for every replication.

F.3 Ablations and development decisions

The final policy uses a fixed three-plan portfolio and observation-based retiming. Earlier development variants retained only the ordering of nominal starts, or treated nominal starts as release thresholds with an observed-progress admission gate. Their development outputs are included under `development.csv` and `robust_development.csv`; they were not relabeled as the final algorithm or substituted into the held-out comparisons. The final retiming rule was selected on the separate development rows, and the held-out protocol was then frozen. Retuning after seeing the held-out matrix would require another independent evaluation pool.

At the prespecified conversational setting, the fixed conservative plan has noisy throughput 21.16 tokens per logical iteration, versus 23.73 for retiming. The latter is a 12.22% paired gain, while still respecting the same progress budget. In the exact-length setting, retiming leaves throughput

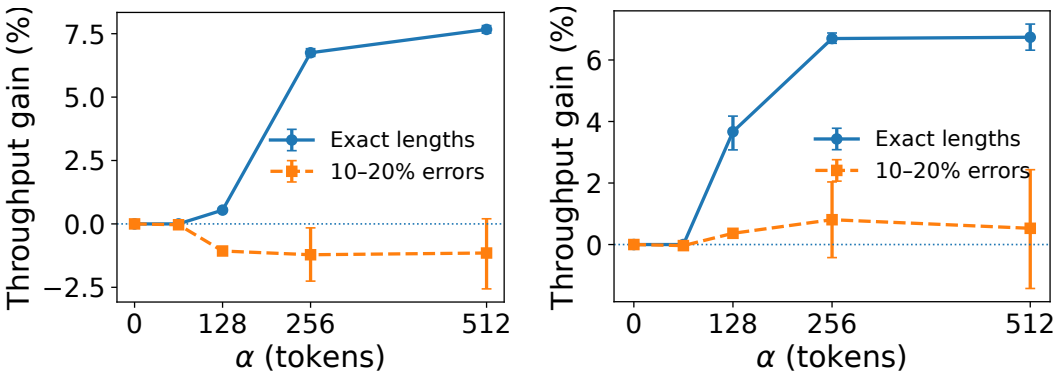

Figure 3: All Arxiv throughput gains relative to LPT-G, including adverse outcomes. Left: $B = 16$. Right: $B = 32$. Solid lines use exact lengths; dashed lines use the specified pre-rounding 10–20% multiplicative perturbations.

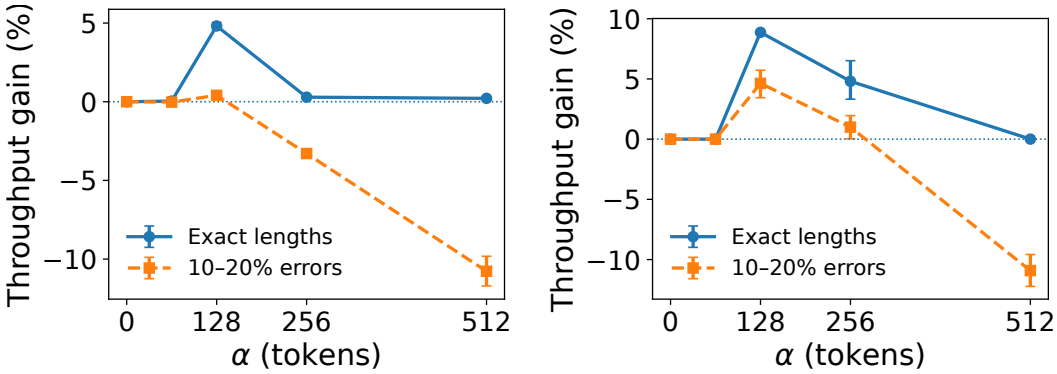

Figure 4: All conversational throughput gains relative to LPT-G. Left: $B = 16$. Right: $B = 32$. Exact-length gains concentrate at intermediate budgets, whereas conservative duration envelopes can hurt under large budgets and noisy lengths.

unchanged at 25.89 but reduces mean completion from 672.1 to 629.5 iterations by removing unnecessary future proximity restrictions after observed completions. The extra flexibility comes from retiming the feasible plan, rather than changing only a length-ordering key.

F.4 Timing sensitivity is not native Vidur calibration

To make the logical/physical distinction explicit, the artifact also evaluates a deliberately restricted decode-time surrogate. It interpolates nine non-attention per-layer A100/Llama-2-7B profiling points from the earlier archive. Batch token counts are rounded to multiples of eight, and counts below eight use the smallest available point. The per-iteration surrogate in milliseconds is

$$d_t = 0.35 + 32m(b_t) + \beta \sum_{i \in S(t)} (h_i + a_i(t) + 1), \tag{20}$$

where m is the interpolated non-attention component sum, h_i is prompt length, and $\beta = \kappa \cdot 0.00038$ is an *assumed*, not measured, attention coefficient. We sweep $\kappa \in \{0, 0.25, 0.5, 1, 2, 4\}$. Prefill and cache-management costs are absent. This is not the upstream Vidur predictor and has not been validated against hardware.

The sum-context term has a useful diagnostic property: across a fixed completed workload, $\sum_t \sum_{i \in S(t)} (h_i + a_i(t) + 1) = \sum_i h_i o_i + Q$ is schedule-independent. Thus this surrogate cannot attribute a gain directly to reducing a maximum footprint. At $\kappa = 1$, the conversational throughput gains shrink to 4.12% for exact lengths and 2.21% for noisy lengths; the logical gains must not

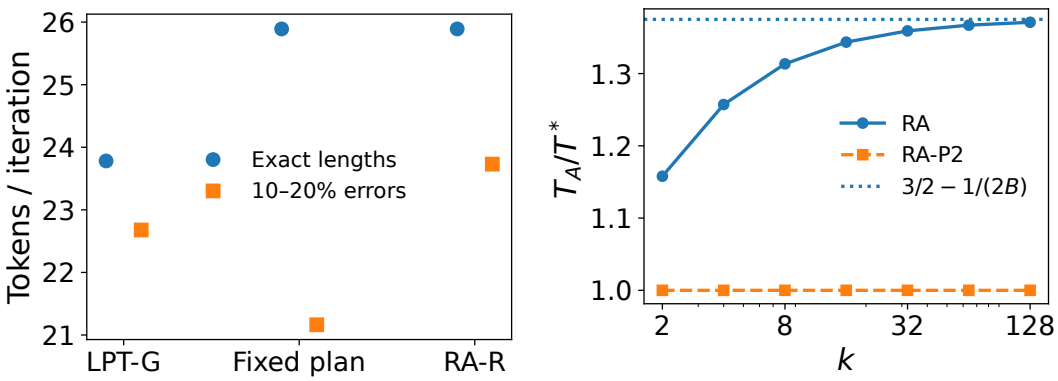

Figure 5: Left: fixed-plan versus retimed execution at the prespecified conversational point ($B = 32$, $\alpha = 128$). Right: the explicit bad family for RA at $B = 4$; the restricted portfolio includes an assignment-first order that reaches the constructive optimum on the tested family. The horizontal line is a limiting worst-case factor, not a pointwise lower bound on observed ratios.

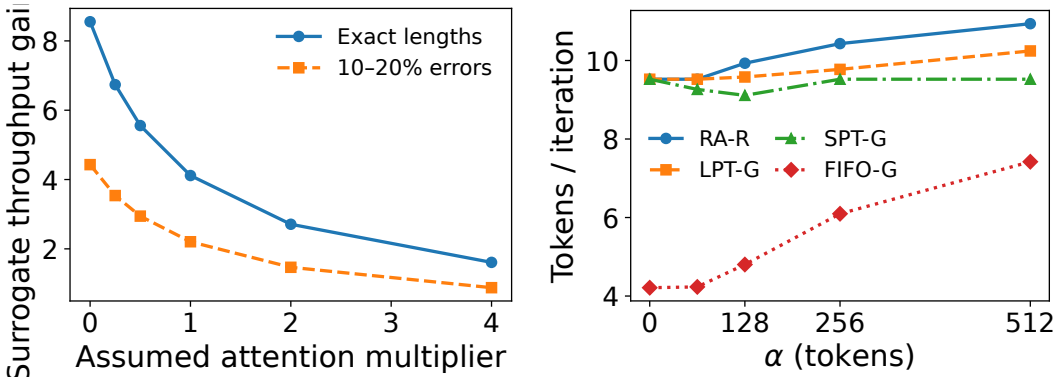

Figure 6: Left: sensitivity of inferred throughput gains to the assumed attention coefficient in (20), at the conversational operating point; this panel is not a hardware measurement. Right: exact-length Arxiv logical throughput over the full budget grid at $B = 32$.

be reported as GPU speedups. Figure 6 displays the full coefficient sweep rather than choosing a favorable coefficient as a calibrated model.

There is a separate limitation of the inspected stock Vidur decode predictor: its attention lookup summarizes a decode batch by its size and rounded average processed-token count (Microsoft and Vidur Contributors, 2026). For example, equal-size batches with context lengths $(64, 1984)$ and $(1024, 1024)$ have the same mean and predictor key despite different maxima. This does not mean scheduling cannot affect predicted total time; it means an equal-mean, different-spread comparison cannot directly validate a maximum-footprint mechanism with that feature map. A full systems study would need a suitable kernel-specific latency model, actual prefill and memory handling, and fresh test workloads.

G A Model-Dependent Cost Interpretation

This interpretation is supplementary; it is not used to generate the numerical throughput comparisons. Let $t = 0, \dots, T_A - 1$ index logical steps, $r_i(t) = t - s_i + 1$ for active requests, and $W = \sum_i o_i$, $Q = \frac{1}{2}\sum_i o_i(o_i + 1)$. Under the explicitly stipulated occupied-slot primitive $C_A(t) = c + \tau|S(t)| \max_{i \in S(t)} r_i(t)$, with $C_A(t) = c$ on empty elapsed steps,

$$C_A = cT_A + \tau Q + \tau E_A, \qquad E_A = \sum_t \sum_{i \in S(t)} \big(\max_j r_j(t) - r_i(t)\big).$$

The identity holds because each request visits token indices $1, \dots, o_i$ exactly once. A nonempty batch of size b_t has at least one zero contribution to its externality, so $E_A \le \alpha \sum_t (b_t - 1)_+$. If N_A is the number of nonempty steps, then $W \le BN_A$ and

$$E_A \le \alpha(W - N_A) \le (1 - 1/B)\alpha W.$$

Under schedule-independent revenue, this yields the additive RA profit comparison

$$\Pi_{\mathrm{RA}} \ge \Pi^*_\alpha - c(C_B - 1)T^* - \tau(1 - 1/B)\alpha W.$$

Indeed, the profit-optimal fair schedule has makespan at least T^* and nonnegative externality. This is an algebraic consequence of the chosen cost primitive, not a claim about arbitrary GPU billing.

For a pure max-driven time charge $\widetilde{C}_A(t) = c + \tau \max_i r_i(t)$, let $\bar{r}(t)$ be the mean token index of a nonempty batch and define

$$U_A = \sum_{t:S(t)\neq\varnothing} \left(1 - \frac{|S(t)|}{B}\right) \bar{r}(t), \qquad \widetilde{E}_A = \sum_{t:S(t)\neq\varnothing} (\max_i r_i(t) - \bar{r}(t)).$$

Then $\widetilde{C}_A = cT_A + \tau Q/B + \tau U_A + \tau \widetilde{E}_A$. Fairness bounds $\widetilde{E}_A \le \alpha N_A \le \alpha T_A$ but does not control U_A through the same argument. Dropping this occupancy term would incorrectly declare the mean-footprint contribution schedule-independent on partially occupied batches.

H Online Release Times

Fix $B = 2$ and a finite integer α. Release one request of length $L > \alpha+1$ at zero. Let a deterministic online policy start it at s. If it never starts, stop the instance and its ratio is zero. Otherwise, one continuation releases nothing else, giving $T^*_{\mathrm{online}}/T_A \le L/(s + L)$. Another releases a second length-L request at $r = s + \alpha + 1$. It lies inside the running request's blackout, so the policy cannot admit it until the first finishes, yielding $T_A \ge s + 2L$. A clairvoyant policy can wait until r and run the requests together, so $T^*_{\mathrm{online}} \le s + \alpha + 1 + L$. With $x = s/L$ and $\eta = (\alpha + 1)/L$, the adversary can force a ratio at most

$$\min\left\{\frac{1}{1+x}, \frac{1+x+\eta}{2+x}\right\}.$$

The perturbation caused by η is at most $\eta/2$, uniformly in $x \ge 0$. At $\eta = 0$, the decreasing and increasing terms meet at $x^2 + x - 1 = 0$, giving value $(\sqrt{5} - 1)/2$. Taking $L \to \infty$ proves the claimed bound, strictly below $2/3$. Release times are integer throughout; no infinitesimal offset is needed.

I Budgeted Tradeoffs and Resource-Disparity Proxies

This section gives consequences of the model and existing RA guarantee. The scalarization and geometric-grid arguments are standard constructions; their role is to state precisely what the parameterized scheduling guarantee implies. They are not additional claims about GPU execution.

I.1 What the budget controls

Let $D(s)$ be the maximum active-batch progress disparity defined in Section 2. For integer logical steps define the token-weighted companion-progress exposure

$$\bar{E}(s) = \frac{1}{W} \sum_{t=0}^{T(s)-1} \sum_{i\in S(t)} \left(\max_{j\in S(t)} a_j(t) - a_i(t)\right).$$

This is a progress-based diagnostic: it measures how far the leading active request is ahead of the request receiving a token. It is not GPU consumption or milliseconds. Unlike D, it averages across serviced tokens and can rank schedules differently. In particular, bounding a maximum need not minimize this average.

Proposition 8 (Progress and resource spread)**.** *Every schedule with $D(s) \le \alpha$ satisfies*

$$0 \le \bar{E}(s) \le \left(1 - \frac{1}{B}\right)\alpha.$$

If prompt lengths have range at most δ, current context lengths $\ell_i(t) = h_i + a_i(t)$ have active range at most $\alpha + \delta$. Hence any common L-Lipschitz per-request resource function $g(\ell)$ has active resource range at most $L(\alpha + \delta)$.

Proof. For a nonempty batch of size b_t, at least one exposure term is zero and every other term is at most α. Let N count nonempty steps. Then $W = \sum_t b_t \le BN$, and

$$W\bar{E}(s) \le \alpha \sum_{t:b_t>0} (b_t - 1) = \alpha(W - N) \le \left(1 - \frac{1}{B}\right)\alpha W.$$

For co-active i, j, $|\ell_i - \ell_j| \le |h_i - h_j| + |a_i - a_j| \le \delta + \alpha$. Apply the Lipschitz inequality to obtain the final statement. □

The exposure inequality is the same algebra underlying Appendix G, but requires no cost primitive or pricing assumption. For equal prompts, context disparity equals progress disparity. For heterogeneous prompts the displayed inequality is only a sufficient bound, not an equivalence. Choosing a particular resource model or translating its spread into latency requires additional system assumptions.

I.2 Exact-budget and approximate-frontier guarantees

Write $A_\alpha = \mathrm{RA}(o, B, \alpha)$ and fix $B \ge 2$. Theorem 6 states $D(A_\alpha) \le \alpha$ and $T(A_\alpha) \le C_B V(\alpha)$. Thus for every comparator s, choosing $\alpha = D(s)$ gives

$$D(A_\alpha) \le D(s), \qquad T(A_\alpha) \le C_B T(s).$$

These are pointwise budget guarantees. They do not say that every output is Pareto optimal, nor that a coarse grid contains every exact Pareto point.

Corollary 9 (A polynomial-size approximate Pareto set)**.** *For $0 < \varepsilon \le 1$, let*

$$\mathcal{G}_\varepsilon = \{0\} \cup \left\{\min\{q_1, \lfloor (1+\varepsilon)^k \rfloor\} : 0 \le k \le \left\lceil \log_{1+\varepsilon} q_1 \right\rceil\right\}.$$

After deleting duplicates, the schedules $\{A_\alpha : \alpha \in \mathcal{G}_\varepsilon\}$ number $O(1 + \varepsilon^{-1} \log q_1)$. For every capacity-respecting schedule s, this family contains a schedule A with

$$T(A) \le C_B T(s), \qquad D(A) \le (1+\varepsilon) D(s).$$

Discarding dominated objective pairs preserves the guarantee.

Proof. If $D(s) = 0$, use $\alpha = 0$. Otherwise, $d = D(s)$ is a positive integer, bounded by q_1 because overlapping intervals have start difference smaller than the earlier request's length. Choose $k = \lceil \log_{1+\varepsilon} d \rceil$ and $\alpha = \min\{q_1, \lfloor (1+\varepsilon)^k \rfloor\}$. Integrality gives $d \le \alpha \le (1+\varepsilon)d$. Feasible-set inclusion implies $V(\alpha) \le T(s)$, so Theorem 6 proves the two inequalities. The number of grid points follows from $\log(1+\varepsilon) \ge \varepsilon/2$. A schedule dominating a removed point also dominates its guaranteed objective bounds. □

A geometric-grid answer may use up to $(1+\varepsilon)$ times a comparator's disparity; it must not be advertised as exact-budget feasibility for that comparator. For a hard requested budget α, simply include an RA call at α and select only schedules whose actual disparity is at most α. This preserves the exact-budget C_B guarantee.

Monotone schedule reuse. For an increasing query grid $\alpha_1 < \cdots < \alpha_m$, select at query k a schedule of minimum makespan from $\{A_{\alpha_j} : j \le k\}$, breaking ties by smaller actual disparity. Every candidate is feasible at α_k, the minimum is non-increasing in k, and inclusion of A_{α_k} retains $T \le C_B V(\alpha_k)$. This construction is valid for any exact-length baseline and is applied equally to all baselines in the supplementary reuse analysis. With upper-bound planning lengths it applies to the nominal upper-bound objective, not automatically to unknown realized makespans. No noisy experiment selects a plan by comparing unobserved realized outcomes.

I.3 Separate the cost of the constraint from the algorithm's loss

Let $V(\infty)$ denote the unconstrained optimum. For any feasible algorithm A,

$$\frac{T_A(\alpha)}{V(\infty)} = \underbrace{\frac{V(\alpha)}{V(\infty)}}_{\text{price of the homogeneity requirement}} \underbrace{\frac{T_A(\alpha)}{V(\alpha)}}_{\text{algorithmic approximation loss}}.$$

The first factor is a policy tradeoff; the second is the approximation algorithm's target. Comparing a constrained policy only with unrestricted LPT cannot identify these two factors, and unrestricted LPT need not equal $V(\infty)$.

Corollary 10 (Worst-case price of homogeneity)**.** *For every instance and budget,*

$$1 \le \frac{V(\alpha)}{V(\infty)} \le 2 - \frac{1}{B}.$$

For every fixed finite α and $B \ge 2$, the upper bound is attained on a suitable integer-length instance.

Proof. Feasible-set inclusion gives the lower bound. Static batching is feasible at every budget, and its usual work/longest-job estimate gives $V(\alpha) \le L_{\text{static}} \le (2 - 1/B)V(\infty)$. For tightness take an integer $k > 2\alpha$, one request of length Bk, and $B(B-1)$ requests of length k. All requests lie in the rigid regime, so $V(\alpha) = (2B-1)k$. Without the progress constraint, the long request occupies one slot while each other slot processes B short requests; this attains the work lower bound $V(\infty) = Bk$. □

The worst-case ratio at a fixed absolute budget does not determine an instance's tradeoff. Workload scale and length composition matter, motivating the full budget curves and controlled heterogeneity experiments below.

J Additional Frontier, Attribution, and Heterogeneity Experiments

J.1 Design and reproducibility

The extension keeps the original RA placement and retiming kernels unchanged. Its design and raw data are recorded separately from the original study. The trace extension reuses the archived workload and prediction vectors exactly; it is not an additional held-out trace population. The synthetic study uses independent artificial draws with the following distributions, always with $n = 128$:

- Uniform: discrete uniform lengths in $\{1, \dots, 600\}$.
- Lognormal: round a lognormal variable with log-mean $\log 250 - 0.8^2/2$ and log-standard deviation 0.8, then clip to $[1, 4096]$.
- Bimodal: an equal mixture of discrete uniforms on $\{40, \dots, 100\}$ and $\{350, \dots, 650\}$.
- Narrow: discrete uniform lengths in $\{240, \dots, 360\}$.

There are 24 workloads per family. NumPy seed sequences use the tuple $(260921, \text{family index}, \text{replication index})$, with family order as listed above and replication indices zero through 23. Every workload is used at both capacities $16, 32$, all six budgets $0, 64, 128, 256, 512, 1024$, and both information conditions. Prediction signs and magnitudes are drawn once per workload and shared across policies and budgets. All 2,304 synthetic scenarios and their 23,040 policy executions are retained, together with the 960 trace scenarios and 9,600 component/comparator executions. The broad distributions were specified before inspecting synthetic performance; they are not fitted descriptions of the original finite trace pools.

Table 4: Exact comparisons on the 10,360-instance subset with $n > B$ and $0 < \alpha < q_1$. These are descriptive statistics of an exhaustive finite set, not sampling confidence estimates.

Policy	Mean gap (%)	Largest gap (%)	Exact optima (%)
Static	11.313	75.00	37.85
SPT-G	14.458	83.33	31.43
LPT-G	3.896	40.00	73.99
Delayed-LPT	3.432	40.00	76.63
RA	0.515	16.67	95.37
RA-P2	0.464	16.67	95.80

The forward delayed-admission comparator. Sort requests by predicted length and designate the longest as the reserved request. For each candidate delay $d \in \{0, \lfloor \alpha/2 \rfloor, \alpha\}$, simulate a forward schedule on predicted lengths. At $d = 0$ this is ordinary LPT-G. At positive d, reserve one slot until the designated request is admitted; the other slots follow LPT order. Once time is at least d, give the designated request admission priority whenever capacity and the observed-progress gate permit. After it starts, all slots follow ordinary LPT-G. Select the candidate with smallest predicted makespan, breaking ties by smaller delay. Execute the selected policy using actual completion events and observed progress, not future realized durations. This comparator is an additional heuristic, not an optimal forward-planning benchmark. It improves over LPT-G on some enumerated and synthetic instances, but not on the 960 archived trace scenarios.

Statistical interpretation. Every confidence interval is a paired, 20,000-resample percentile bootstrap interval over 24 workload indices. For a fixed synthetic family, this quantifies variation under the specified artificial generator. For the trace replay, it remains conditional on the small finite pool. Neither interval quantifies GPU or population-wide serving uncertainty. The "best forward" CSV reference takes the smallest realized makespan of Static, FIFO-G, SPT-G, LPT-G, and delayed LPT independently on each instance. It is an evaluation reference, not an implementable noisy-length selection rule. Main-text percentage gains use the explicitly identified LPT-G comparator.

J.2 Exact value functions and optimality gaps

The first exhaustive set contains all 461 nonempty multisets of at most six lengths from $\{1, 2, 3, 4, 5\}$, tested at six budgets and three capacities. This gives 8,298 exact budget instances. After establishing computational feasibility, an additional complete set covers all 791 multisets of seven through ten lengths from $\{1, 2, 3, 4\}$, at five budgets and the same capacities, giving 11,865 further exact instances. This extension covers more than two layers at each tested capacity without choosing instances by observed performance.

The exact dynamic program has state (waiting multiset, running length–age pairs). At each integer step it enumerates every admissible subset of waiting requests, including the empty subset when a request is already active. Existing running requests always advance, so the oracle respects nonpreemption. A globally empty delay is excluded because deleting it preserves feasibility and reduces makespan. Completion occurs at the modeled service length. Memoization only identifies states with the same future problem; it does not impose a request ordering. A separate assignment enumeration supplies the unconstrained optimum and agrees with the dynamic program at the nonbinding budget on every tested multiset.

Across all 20,163 budget instances, plain RA is optimal on 96.71%, with a mean gap of 0.372%. Table 4 removes $n \leq B$, zero budgets, and $\alpha \geq q_1$. The remaining 10,360 cases need not all have a binding constraint at the optimum; the filter is stated to avoid conflating this subset with a population of uniquely difficult instances. Observed RA optimality and the low average gap do not replace the worst-case analysis, particularly because all these enumerated lengths are small integers.

J.3 What the full ablation establishes

At the conversational representative point, the two assignment-first candidates improve neither the exact nor the upper-bound RA plan's makespan in any of the 24 workloads. The primary throughput

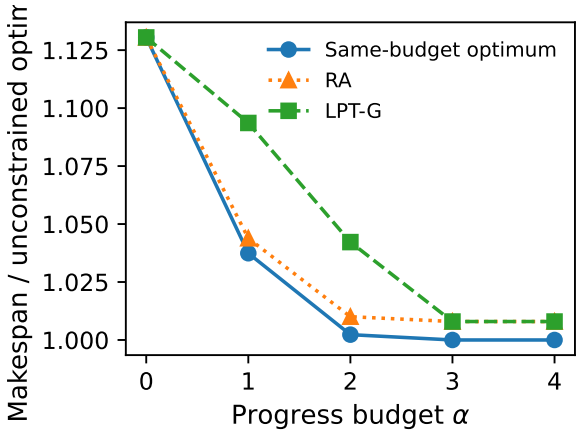

Figure 7: Budget value functions averaged over the fixed exhaustive set of 791 multisets with seven through ten lengths from $\{1, 2, 3, 4\}$, $B = 3$. The optimum curve measures the cost of the homogeneity requirement; the additional vertical gap measures algorithmic loss. Each curve uses the same instances at every budget.

improvement is therefore obtained without the portfolio. Across the full trace grid, nominal portfolio improvements occur in nine of 960 scenarios; the complete component CSV retains where these occur. The portfolio remains useful on the explicit family from Theorem 5, as shown in the original appendix experiment.

Retiming changes more than the effect of conservative duration bounds. In the exact-length representative case, RA and RA with retiming have the same makespan, but mean waiting falls from 387.4 to 344.7 iterations, mean completion from 672.1 to 629.5, and average companion-progress exposure from 73.34 to 21.32 tokens. The maximum disparity remains 128. Thus maximum and average exposure are distinct diagnostics. No theorem asserts that retiming decreases the average exposure on every instance; for example it can increase that average at a loose budget. The raw CSV reports all such changes.

A structural prediction-error diagnostic. At $B = 32, \alpha = 512$, all 24 uniform and all 24 conversational realized workloads satisfy $q_{B+1} \leq \alpha$, the short-tail condition in Theorem 6. After replacing lengths by conservative upper bounds, 20 uniform and 19 conversational instances no longer satisfy it. By contrast, all 24 lognormal workloads satisfy it both before and after the replacement. Mean $(B+1)$st lengths are 445.0 versus 544.8 for uniform, 409.9 versus 535.0 for conversation, and 316.0 versus 392.3 for lognormal. Thus modest per-request forecast errors can move a planning instance across a structural regime boundary. This is a descriptive diagnostic, not a causal decomposition or a necessary condition for regression: narrow workloads can regress without crossing the boundary. The code records these indicators for every noisy scenario, not only these examples.

J.4 Complete synthetic outcomes

Tables below include every budget, capacity, and information setting for plain RA with retiming. The full machine-readable results additionally include the portfolio, fixed plans, all forward policies, actual disparities, exposure, waiting/completion metrics, and execution starts. At budget 128 and capacity 32, the narrow distribution yields zero gain; the other exact-length families yield approximately four percent. Larger budgets can produce much larger lognormal gains but severe noisy-length regressions for other distributions. Neither the largest favorable gain nor the largest unfavorable gain is a universal characterization of the method.

Table 5: Complete synthetic gains for plain RA with retiming versus LPT-G, $B = 16$. Negative gains are regressions.

Distribution	α	Exact lengths	10–20% errors
Uniform	0	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
Uniform	64	0.92 [0.59, 1.25]	0.18 [-0.13, 0.50]
Uniform	128	1.40 [1.17, 1.64]	0.36 [0.06, 0.68]
Uniform	256	1.96 [1.69, 2.24]	-2.39 [-2.71, -2.09]
Uniform	512	1.33 [1.22, 1.45]	-10.97 [-11.67, -10.27]
Uniform	1024	0.00 [0.00, 0.00]	-9.68 [-10.55, -8.92]
Lognormal	0	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
Lognormal	64	0.39 [0.21, 0.58]	-0.01 [-0.09, 0.08]
Lognormal	128	0.81 [0.68, 0.96]	-0.99 [-1.21, -0.78]
Lognormal	256	8.63 [6.77, 10.43]	-2.96 [-3.93, -1.87]
Lognormal	512	15.30 [14.09, 16.47]	0.55 [-1.60, 2.51]
Lognormal	1024	9.08 [6.14, 12.13]	-4.97 [-7.75, -2.08]
Bimodal	0	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
Bimodal	64	0.67 [0.46, 0.89]	-0.09 [-0.36, 0.22]
Bimodal	128	1.29 [1.16, 1.41]	-0.29 [-0.59, -0.00]
Bimodal	256	1.45 [1.20, 1.72]	0.30 [0.01, 0.57]
Bimodal	512	1.96 [1.69, 2.24]	-8.13 [-8.89, -7.39]
Bimodal	1024	0.00 [0.00, 0.00]	-8.48 [-9.30, -7.70]
Narrow	0	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
Narrow	64	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
Narrow	128	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
Narrow	256	0.36 [0.29, 0.43]	-1.31 [-1.53, -1.08]
Narrow	512	0.00 [0.00, 0.00]	-2.14 [-2.58, -1.75]
Narrow	1024	0.00 [0.00, 0.00]	-1.61 [-2.20, -1.06]

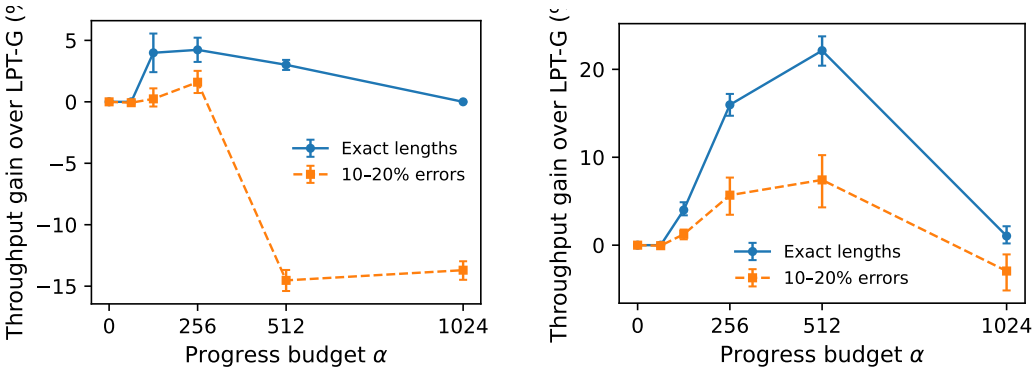

Figure 8: Complete synthetic budget curves, $B = 32$, plain RA with retiming. Left: uniform; right: lognormal. Error bars are paired bootstrap intervals.

Table 6: Complete synthetic gains for plain RA with retiming versus LPT-G, $B = 32$. Negative gains are regressions.

Distribution	α	Exact lengths	10–20% errors
Uniform	0	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
Uniform	64	0.00 [0.00, 0.00]	-0.07 [-0.21, 0.00]
Uniform	128	3.99 [2.41, 5.56]	0.24 [-0.38, 1.10]
Uniform	256	4.23 [3.25, 5.22]	1.60 [0.72, 2.52]
Uniform	512	3.01 [2.59, 3.41]	-14.53 [-15.39, -13.68]
Uniform	1024	0.00 [0.00, 0.00]	-13.70 [-14.48, -12.97]
Lognormal	0	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
Lognormal	64	0.00 [0.00, 0.00]	-0.06 [-0.14, -0.00]
Lognormal	128	4.00 [3.39, 4.88]	1.24 [0.67, 1.77]
Lognormal	256	15.96 [14.72, 17.20]	5.69 [3.46, 7.69]
Lognormal	512	22.13 [20.42, 23.75]	7.42 [4.30, 10.24]
Lognormal	1024	1.05 [0.20, 2.15]	-2.94 [-5.15, -1.05]
Bimodal	0	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
Bimodal	64	2.10 [1.73, 2.43]	-0.54 [-1.27, 0.24]
Bimodal	128	4.17 [3.74, 4.60]	1.42 [0.94, 1.92]
Bimodal	256	3.88 [2.65, 5.10]	0.25 [-0.90, 1.30]
Bimodal	512	4.02 [3.20, 4.80]	-14.57 [-16.47, -12.65]
Bimodal	1024	0.00 [0.00, 0.00]	-13.51 [-14.63, -12.43]
Narrow	0	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
Narrow	64	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
Narrow	128	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]
Narrow	256	0.00 [0.00, 0.01]	-1.47 [-1.72, -1.22]
Narrow	512	0.00 [0.00, 0.00]	-3.69 [-4.19, -3.17]
Narrow	1024	0.00 [0.00, 0.00]	-4.23 [-4.74, -3.71]

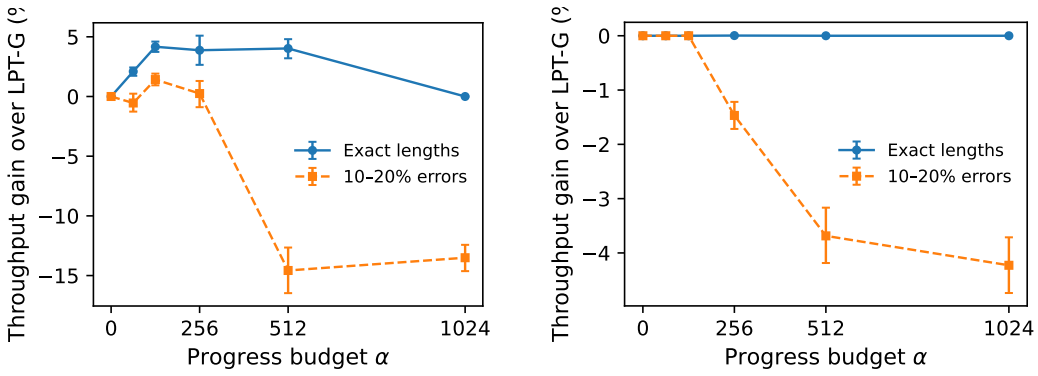

Figure 9: Complete synthetic budget curves, $B = 32$, plain RA with retiming. Left: bimodal; right: narrow. All regressions are retained.