SCHEDULING SHARED-PREFIX DECODING WITH UNKNOWN OUTPUT LENGTHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Requests sharing a prompt prefix can reuse its key–value (KV) cache. A scheduler must decide which requests to run within a fixed memory budget before knowing their output lengths. We show that individual prompt lengths and generation progress cannot distinguish shared prefixes from disjoint prompts. Such a policy must count the shared prefix separately for each request to respect the memory budget for every compatible prompt layout. On a family of k simultaneous requests, this restriction causes an $\Omega(k)$ increase in total completion time relative to an optimal schedule. For simultaneous requests whose prompts consist of one shared prefix kept in memory, subtracting its footprint leaves only generated tokens to schedule. This reduction preserves feasible schedules and completion times. With staggered starts and doubling attempt lengths, the Geometric Slicing Algorithm achieves a $76/3$ competitive ratio for total completion time on this budget. The guarantee holds for unknown output lengths in a model where interrupted requests restart from the beginning. We also examine when an existing serving engine can use cached prefixes to admit more requests. In TGI, making prefixes available immediately after prefill yields $1.43\times$ and $1.95\times$ makespan speedups on constructed and natural-language request bursts, respectively, under constrained memory. Controls show that the benefit decreases as requests arrive farther apart; native-engine comparisons identify settings where existing scheduling already captures the shared-memory benefit.

1 INTRODUCTION

Language-model applications often issue several requests from the same document, tool context, or reasoning prompt. Their common prefix can share a key–value (KV) cache, while generated tokens require private memory. How should the scheduler allocate a fixed memory budget before knowing output lengths?[1]

Existing guarantees usually add per-request memory footprints. Jaillet et al. (2025) gave approximations with known output lengths; the Geometric Slicing Algorithm (GSA) of Feng et al. (2026b) handles unknown lengths with ratio ≤ 61.92. Feng et al. (2026a) extended constant guarantees to arbitrary prompt lengths, makespan, and online arrivals. Shared KV changes their memory constraint: the cost of another request depends on which requests are already running.

Consider $k = 32$ requests with the same 6k-token prompt, each producing a short answer: the shared prefix has length $W = 6144$, and each output has length $L = 64$. This configuration matches the scale of a shared-prefix group in the public Mooncake trace (Qin et al., 2025a). A radix cache (Zheng et al., 2024; Kwon et al., 2023) stores the prefix once, so the batch uses $W + kL \approx 8$k tokens of KV. Counting it once per request instead gives $k(W + L) \approx 199$k tokens and admits only one request at that capacity. Shared-document retrieval and Best-of-N sampling also create groups of related prompts (Lewis et al., 2020; Snell et al., 2024). Their request-specific questions or instructions add private prompt memory, which must also be counted.

[1] Code and experiment records: anonymous repository.

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

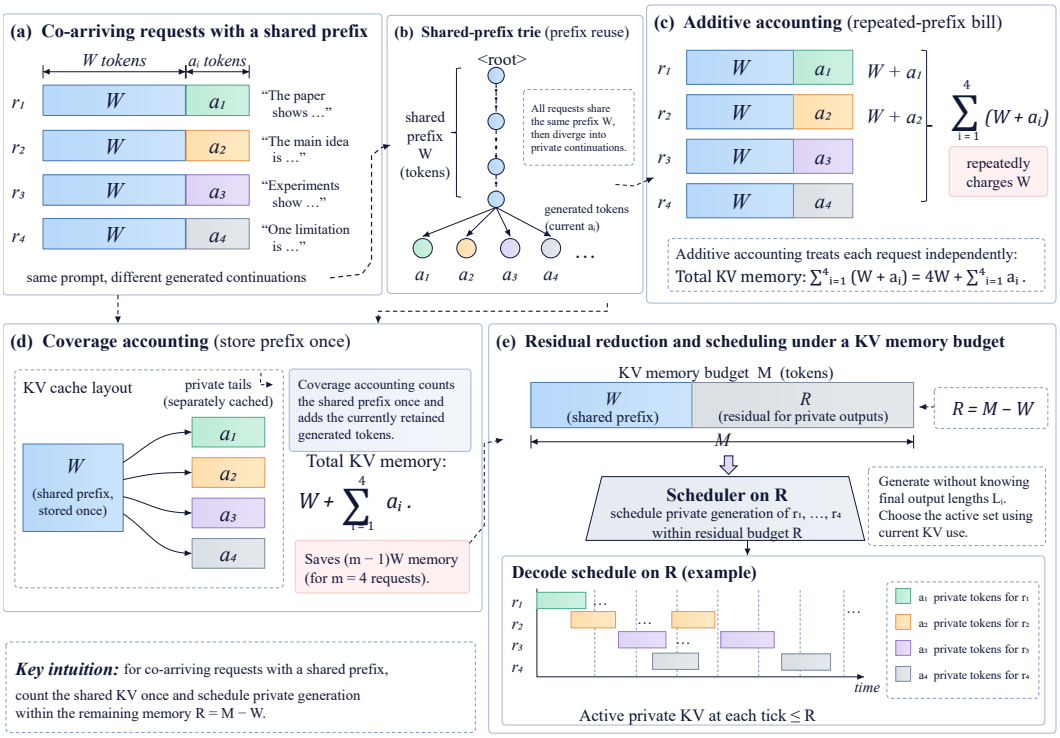

Figure 1: Shared-prefix accounting and residual scheduling, in token units. (a,b) Four simultaneous requests share one resident prompt of length W and retain a_i generated tokens each. (c,d) Additive accounting charges $4W + \sum_i a_i$, whereas shared memory uses $W + \sum_i a_i$. (e) Subtracting W leaves the budget $R = M - W$ for generation, without knowing final output lengths L_i. Private prompt suffixes are absent. The timeline is schematic; each decode tick also reserves one new token per active request.

We represent shared prompt blocks by a prefix tree, or trie. Request i follows a path P_i whose edges have memory weights w_e. For an active set S, each shared edge is counted once, followed by each request's generated tokens:

$$\mathrm{Mem}_{\mathrm{cov}}(S) = \sum_{e \in \cup_{i \in S} P_i} w_e + \alpha \sum_{i \in S} a_i,$$

where a_i is the number of generated tokens currently retained for request i and α is their per-token memory cost. We call this *coverage memory*, referring to the union of prompt paths. The simplest case has one resident shared prefix and no private prompt suffixes. Figure 1 shows the shared-memory constraint and the budget left for generation.

Our approach is to subtract the shared prefix from the memory budget and schedule generation on what remains. For the simple case above, this gives exactly the additive scheduling problem with zero prompt length and budget $R = M - W$ (Theorem 6). We call GSA on R *Coverage-GSA* and prove a $76/3$ competitive ratio for staggered starts with doubling attempt durations.

The reduction requires sharing information. Prompt lengths and generation progress alone cannot distinguish a common prefix from disjoint prompts. A policy safe in both cases must count prompts separately, causing an $\Omega(k)$ loss on long shared prefixes (Lemma 3).

We also study a simpler admission rule that reserves a known output cap $\widehat{L}_i$ for each request. It admits a set S when $W + \sum_{i \in S} (q_i + \widehat{L}_i) \le M$, where q_i is the private prompt length (Algorithm 4).

This tests shared-memory admission separately from GSA's repeated attempts to discover output lengths.

Contributions.

1. **The cost of not observing sharing.** We prove the $\Omega(k)$ lower bound for policies restricted to lengths and progress. For peak-reserved schedules, the gap is $\Theta(1 + \min\{W/L, k\})$ (Theorem 1).
2. **A scheduling reduction and a sharper guarantee.** With no private prompt suffixes, staggered GSA with doubling slices is $\leq 76/3$-competitive after subtracting the common prefix (Theorem 22). We also treat private suffixes and online total completion time (Theorem 38).
3. **When a serving engine can use a shared prefix.** Earlier prefix publication in TGI changes admission and completion time. Native controls and trace replays identify when this helps, including schedulers that already count shared blocks.

Scope. The main guarantee assumes simultaneous arrivals, a prefix kept in memory, no private prompt suffixes, and interrupted requests that restart from the beginning. Section 5 treats private prompt suffixes and multiple prefix groups. The online result concerns total completion time; arrival-to-completion latency is a different objective. Managing prefix residency and routing requests between devices require decisions beyond the fixed-prefix model.

2 Related Work

Additive theory. Jaillet et al. (2025); Feng et al. (2026b;a) establish scheduling guarantees under additive memory. D'Ambrosio (2026) give a $\Theta(n)$ deterministic latency ratio in the fully online nonpreemptive model. Fluid and control analyses also sum per-request memory (Ao et al., 2025; Dai et al., 2025; Bari et al., 2025). Kong et al. (2026) use memory-time volume; Wang et al. (2025) prove NP-hardness and give batching approximations with variable prefill and decode lengths. Subtracting a common prefix lets additive guarantees apply to the remaining memory.

Prefix-serving systems. Paged and radix caches share KV storage (Kwon et al., 2023; Zheng et al., 2024; Ye et al., 2024). BatchLLM groups shared-prefix requests (Zheng et al., 2026); Preble tracks sharing and residency across engines (Srivatsa et al., 2025); PRISM jointly manages retention and admission (Qu et al., 2026). These systems already use information beyond prompt lengths and progress. Our lower bound concerns that restricted observation class; our positive result gives a reduction and competitive analysis (Appendix I.2).

Preemption and recovery. SRF selects victims using current KV context, with a shared-prefix rule for removable segments (Kim et al., 2026). Marconi uses reuse and compute savings per byte for cache eviction (Pan et al., 2025a). An engine can retain generated tokens and rebuild their KV after interruption. Our model instead restarts private generation, so native comparisons must measure actual recovery work and request latency.

Relation to GSA. We keep GSA's restart rule and change its memory constraint. The reduction gives Coverage-GSA; counting prefixes separately serializes the long-prefix family (Corollaries 7, 10). For private prompt suffixes, the results of Feng et al. (2026a) apply on the remaining budget (Theorem 9). Continuous batching and compression address other execution costs (Yu et al., 2022; Agrawal et al., 2024; Qin et al., 2025b; Wu et al., 2025). Speculative and tree decoding create additional branches (Leviathan et al., 2023; Cai et al., 2024) (Corollary 42).

3 Model

We use the simultaneous-arrival model of Feng et al. (2026b), replacing additive memory with coverage memory. A policy is *non-clairvoyant* if it learns an output length only when that request finishes. Unless stated otherwise, unused jobs are killed and restart at age 0 (preemption that drops

private KV). The additive width-one lower bound also holds when progress is retained (Proposition 5). Appendix B.9 compares the kill-and-restart algorithm with a stronger, active-memory-only optimum that may retain progress; this comparison does not implement an engine that pins paused private KV.

Definition 1 (Star batch instance)**.** *k jobs share one resident prefix of weight W and have no private prompt suffixes. We call this a* star*: one common prompt path followed by separate output paths. Job i has unknown private length L_i; after a_i private decode ticks its private age is a_i. For an active set S,*

$$\mathrm{Mem}_{\mathrm{cov}}(S) = W + \alpha \sum_{i \in S} a_i, \qquad \mathrm{Mem}_{\mathrm{add}}(S) = \sum_{i \in S} (W + \alpha a_i).$$

Memory is measured in token-equivalent KV units, with private-token cost $\alpha = 1$ unless stated otherwise. At each tick the scheduler selects $S_t \subseteq \{1, \ldots, k\}$. Each selected request advances by one token, and its resulting memory must fit capacity M. All requests initially arrive at time zero, so total flow time is the sum of completion times $\sum_i C_i$. A look-ahead *check includes the next token of every selected job before deciding whether the set fits. A scheduler is* additive look-ahead feasible *if every S_t satisfies this check under $\mathrm{Mem}_{\mathrm{add}}$ with budget M. It is* additive completion-feasible *if it never runs a set that cannot reserve peak $W+L$ for every member: $|S|(W + L) \le M$ (the peak-reservation class of the additive baselines in Section 5).*

Definition 2 (Competitive ratio)**.** *For a family $\mathcal{I}$, the ratio of ALG vs OPT is $\sup_{I \in \mathcal{I}} \mathrm{ALG}(I)/\mathrm{OPT}(I)$, where OPT is the offline optimum under coverage memory.*

Peak reservation. A batch that fits before generation may run out of memory as outputs grow. Peak reservation avoids this by reserving a known upper bound on each output before starting the batch. For stars with an output upper bound $\widehat{L}$, peak reservation uses $W + |S|\widehat{L} \le M$ (coverage) or $|S|(W + \widehat{L}) \le M$ (additive). For request-specific prompt suffixes, replace each private cost by $q_i + \widehat{L}_i$. A known upper bound and an oracle's realized L_i are different inputs.

4 Why additive schedulers serialize

Remark 1 (Long shared prefix vs short shared prefix)**.** Long-prefix regime *(Theorem 4): $W \ge kL$, $M = W + kL$, $\alpha = 1$; coverage OPT runs all k jobs in parallel (flow kL). Additive cohort capacity $s^\star = \lfloor M/(W + L) \rfloor$ equals 1 at $W = kL$, so best additive flow is $L \cdot k(k + 1)/2$ and the ratio is $(k + 1)/2 = \Omega(k)$.* Short-prefix counterexample*: W=10, L=4, k=32, M=138 yields $s^\star$=9 and best-additive ratio 2.3125—not $\Omega(k)$ (Appendix B.5).*

At fixed k=32, L=64, the additive width $s^\star$ steps $16 \to 4 \to 2 \to 1$ as $W/(kL)$ grows to $W = kL$. A short shared prompt can yield only a small admission loss; for example, $k = 32$, $L = 64$ and $W = 64$ give additive peak width 16 (Appendix I.1).

Theorem 1 (Regime characterization)**.** *Fix identical lengths L, budget $M = W + kL$, and let $s^\star = \lfloor M/(W + L) \rfloor \ge 1$ be the additive completion width. Coverage OPT has flow kL; best additive uninterrupted $s^\star$-cohorts have flow*

$$F_{\mathrm{add}}(k, s^\star, L) = L \cdot \Phi(k, s^\star), \qquad \Phi(k, s) = \sum_{i=1}^{k} \left\lceil \frac{i}{s} \right\rceil = s\frac{q(q+1)}{2} + r(q+1)$$

for $k = qs + r$, $0 \le r < s$, giving the exact ratio $F_{\mathrm{add}}/(kL) = \Phi(k, s^\star)/k$. Asymptotically, over additive completion-feasible *(peak-reserved cohort) schedules—where $|S| \le s^\star$ at every tick and the slot exchange applies (Appendix B.6)—the ratio is governed by $(W/L,\ k)$ jointly:*

$$\frac{F_{\mathrm{add}}}{F_{\mathrm{cov}}} \;=\; \Theta\Big(1 + \min\Big\{\tfrac{W}{L},\ k\Big\}\Big) \qquad (k \ge 2),$$

with universal constants. In particular, $W = \Theta(L)$ gives $O(1)$, whereas any $W = \Theta(kL)$ (including $W = c\,kL$ with $c < 1$) gives $\Theta(k)$ as k grows. The exact threshold for $s^\star = 1$ is $W > (k - 2)L$. The stronger condition $W \ge kL$ also forces width one under the age-dependent look-ahead constraint (Lemma 2). For the wider look-ahead-feasible class the bound is proved at $W \ge kL$ (Lemma 2); whether it extends to intermediate W is open (Appendix B.4).

Lemma 2 (Additive width-one serialization). *Under* $\mathrm{Mem}_{\mathrm{add}}$ *look-ahead alone on the long-prefix family ($W \geq kL$, $M = W + kL$, $\alpha = 1$, $k \geq 2$), every additive look-ahead-feasible schedule serves at most* one *job per tick: two jobs at any ages $a, b \geq 0$ have look-ahead cost $2W + a + b + 2 \geq 2W + 2 > M$. In particular at most one job can advance from private age $L-1$ to L in any tick.*

Lemma 3 (Memory safety without sharing information). *Consider a policy using only the budget M and per-request states $\{(p_i, a_i)\}$: prompt length and current private age. Require its next-tick memory to fit M for every prompt layout consistent with these observations. On the star family with $p_i = W$, every admitted set S_t must satisfy $\sum_{i \in S_t}(W + a_i + 1) \leq M$. The policy is therefore additive look-ahead feasible, and Theorem 4's $\Omega(k)$ bound applies.*

The proof compares a shared prefix with disjoint prompts that produce the same observations (Appendix B.2). Prefix IDs or block-residency information place a policy outside this observation class. A quantitative guarantee for incomplete or inaccurate sharing information requires a separate signal model (Appendix B.2).

Theorem 4 (Additive serialization on long shared prefixes). *Fix $\alpha = 1$, $W \geq kL$, and $M = W + kL$. Coverage OPT has flow time kL. Any additive* completion-feasible *schedule (never runs S with $|S|(W+L) > M$) has flow-time ratio $\Omega(k)$ versus coverage OPT. At $W = kL$ the best additive completion-feasible schedule has ratio exactly $(k+1)/2$, and the same $\Omega(k)$ bound extends to every additive look-ahead-feasible schedule via the width-one lemma (Lemma 2); coverage simultaneous batching matches OPT.*

Proposition 5 (Makespan with retained progress). *Under pause-without-reset, the same long-prefix family ($W \geq kL$, $M = W + kL$) permits at most one request to advance per tick under additive look-ahead (Lemma 2). Its kL required service ticks give additive makespan at least kL, while simultaneous coverage execution finishes at time L. Hence the makespan ratio is $\Omega(k)$ without kill-and-restart.*

5 COVERAGE-GSA

Coverage-GSA subtracts the shared prefix weight W and runs GSA on the remaining budget $R = M - W$ with prompt length 0. It gives every unfinished request one attempt per phase. An attempt lasts at most τ_p ticks, and the number of overlapping attempts is chosen to fit the memory budget. Requests that finish leave the system. The others restart in a later phase with a longer attempt, as in the original GSA. Peak reservation instead uses a known output cap in a single admission test (Algorithm 4). The two policies count shared memory in the same way but use different execution schedules.

Theorem 6 (Exact residual reduction). *Fix a star instance: shared prefix W, private lengths $(L_i)_{i \leq k}$, budget M, $\alpha = 1$, residual $R = M - W$. A tick-by-tick schedule is coverage look-ahead feasible iff the* same *schedule is feasible in the additive model of Feng et al. (2026b) with prompt length $s = 0$ and budget R:*

$$W + \sum_{i \in S_t}(a_i + 1) \leq M \iff \sum_{i \in S_t}\big(0 + a_i + 1\big) \leq R \qquad \text{at every tick } t.$$

The same requests run at the same ticks in both models, with the same resets and completions. Their feasible schedules and optimal flow values are identical, so every competitive ratio is preserved (Appendix B.7).

Corollary 7 (Coverage-GSA: Constant competitiveness for unknown lengths). *Run GSA (Feng et al., 2026b, Alg. 2) with prompt length 0 on the residual budget $R = M - W$ (*Coverage-GSA*). For heterogeneous unknown lengths with $L_{\max} \leq R$, staggered Coverage-GSA at slice growth $\alpha_g = 2$ is non-clairvoyantly $\leq 76/3 < 25.34$-competitive versus coverage OPT on stars (Theorem 22, Appendix B.8).*

Why the ratio stays constant. Let N_p be the number of unfinished requests at the start of phase p, whose slice is τ_p. For the analysis, sort the realized lengths as $L_{(1)} \leq \cdots \leq L_{(k)}$; the algorithm never uses this ordering. A request that finishes after L tokens uses successive memory amounts $1, 2, \ldots, L$. Their sum, $\binom{L+1}{2}$, is its memory-time volume. An Abel summation over the length

classes gives

$$\mathrm{OPT} \;\geq\; V := \frac{1}{R}\sum_{i=1}^{k}(k-i+1)\binom{L_{(i)}+1}{2} \;\geq\; \frac{3}{64R}\sum_{p}\tau_p^2 N_p^2.$$

This bounds the cost of all phases together, avoiding a separate charge of V for each phase. On GSA's sequence of slices, each nonfinal slice is at most $R/2$ and the final slice equals R. Its feasible staggered width satisfies $k_p\tau_p \geq R$. The resulting phase duration is $D_p \leq N_p\tau_p^2/R + \tau_p$, giving

$$\begin{aligned}\sum_i C_i &\leq \frac{1}{R}\sum_p \tau_p^2 N_p^2 + 4\sum_i L_i \\ &\leq \frac{64}{3}V + 4\sum_i L_i \leq \frac{76}{3}\,\mathrm{OPT}.\end{aligned}$$

Appendix B.8 proves the volume, phase-duration and staggered-pipeline feasibility bounds. The constant applies at finite pipeline widths; the prior GSA analysis gives 61.92 at its tuned growth and approaches 32 at growth 2 only as the minimum width diverges. An epoch extension for online total completion time is given in Theorem 38. On 160 random heterogeneous tight-M stars the observed ratio against the volume lower bound is at most 7.7; Coverage-GSA also wins on the heterogeneous family that breaks greedy residual packing (flow 587 vs 1123; Appendix B.7).

A concrete execution. Let the shared prefix leave $R = 5$ token units for three requests with no private prompt suffixes. Their output lengths turn out to be $(1, 3, 5)$, but the scheduler does not know them. Doubling GSA uses slices $(1, 2, 5)$: the first attempt finishes request 1 at time 1; the second runs the remaining requests for two ticks and neither finishes. In the last phase, request 2 runs from 3 to 6 and request 3 from 8 to 13. The fixed staggered start leaves times 6–8 idle; starting immediately would change the schedule being analyzed. Completions are $(1, 6, 13)$ and private memory never exceeds 5. Appendix A gives every phase and its memory use.

Proposition 8 (Static prefix groups and a limit of additive reduction)**.** *Consider disjoint resident prefix groups, each with a fixed budget from $M = \sum_g M_g$. Theorem 6 applies separately to each group: $\sum_{i\in S_t\cap g}(a_i+1) \leq R_g = M_g - W_g$. The reduction preserves feasibility and completions for schedules that obey this partition. A general fixed-cost additive representation fails: for four jobs on two partially shared leaves $u\cdot v$, $u\cdot w$, the coverage-feasible family at some ages (same-branch pairs feasible, cross-branch pairs not) is not representable by any fixed per-job costs and threshold (Appendix B.16). General tries therefore require a set-dependent memory constraint in at least these instances.*

Theorem 9 (Adding request-specific prompt suffixes)**.** *Let job i have a private prompt suffix q_i after the common prefix, such as a question or instruction. Coverage memory becomes $W + \sum_{i\in S}(q_i + \alpha a_i)$.* (a) Identical suffixes. *With $q_i \equiv q$, subtracting the shared prefix yields the original GSA model at prompt length $s = q$ on budget R—its native setting—so GSA at $s = q$ transfers with constants unchanged whenever $q + L_{\max} \leq R$.* (b) Heterogeneous suffixes. *With arbitrary $q_i \geq 1$ and $q_i + L_i \leq R$, subtracting the shared prefix yields the arbitrary-prompt additive model of Feng et al. (2026a), so their regime-aware router run on the residual is constant-competitive (≤ 996 for their completion-time objective, with their separate makespan and online guarantees) with no relation assumed among the suffixes.* (c) Additive side. *For identical output length L, $\alpha = 1$, $M = W + \sum_i(q_i + L)$ and $W \geq \sum_i(q_i + L)$, every additive look-ahead-feasible schedule is width one, so the $\Omega(k)$ separation of Theorem 4 holds on suffixed stars (proofs in Appendix B.15).*

With partial sharing, adding a request can cost less memory when more of its prefix is already active. The prefix term of $\mathrm{cov}(S)$ is therefore monotone and submodular. Proposition 8 shows why fixed per-request costs cannot always represent it. For a trie with one common root and separate child prefixes, subtracting the root leaves disjoint prefix groups with budget $M - |U|$ (Lemma 46). Algorithm 2 packs these groups in each geometric phase (Appendix B.18). It runs groups sequentially when no two prefixes fit together. When every phase fits all groups, it is 4-competitive regardless of the number of groups. Mixed packing and depth-three overlap remain open.

Corollary 10 (Additive GSA serializes)**.** *GSA run with prompt $s = W$ per job and budget M keeps its staggered-pipeline peak $\leq M$ under additive memory, hence is additive look-ahead feasible. On*

Table 1: TGI stock versus prefill-time prefix publication. All requests arrive together; $k = 16$, output cap 64. Ratios are stock/patched. Natural-language prompts use distinct questions under one common prefix. The intervention can affect both admission and prefill compute.

Prompt setting	Stock (s)	Patched (s)	Makespan ratio	Mean-flow ratio
Constrained, constructed	9.59 ± 0.14	6.68 ± 0.08	1.43	1.66
Constrained, natural language	6.805	3.482	1.95	1.84
Loose capacity, constructed	4.01	3.59	1.12	—

the long-prefix family ($W \geq kL$, $M = W + kL$), Lemma 2 forces every phase width to one, so its flow is $\geq L \cdot k(k+1)/2$ and its ratio versus coverage OPT is $\geq (k+1)/2 = \Omega(k)$. The regime-aware router of Feng et al. (2026a) is additive look-ahead feasible by construction, so the same bound applies to it. Tick-level simulation gives additive-GSA ratios $7.9/17.8/41.2/44.5$ at $k = 4/8/16/32$ (the excess over $(k+1)/2$ is geometric-phase waste) and Coverage-GSA at 2.2–3.9 on the same family (Appendix B.7).

At $M = W + L_{\max}$, both peak-reservation rules admit only one request. Scheduling must therefore use memory as generation progresses. Greedy residual packing is exact on identical lengths and ≥ 7.4-bad heterogeneously (Theorem 27); Coverage-GSA retains its constant guarantee. A separate backfilling rule keeps the first unfinished request running and uses spare memory for others (Algorithm 3). A restarted request must fit for $d_j = \max\{1, 2h_j\}$ ticks, where h_j is its deepest observed unfinished attempt. Each request finishes no later than under serial FIFO and generates fewer than $3L_i$ private tokens (Theorem 51).

Coverage lockstep is guaranteed optimal at the two extreme budgets and can be suboptimal at intermediate budgets, though it remains a $\leq 5 \cdot \mathrm{OPT}$ benchmark (Theorem 11); non-clairvoyant coverage doubling is ≤ 4-competitive against that benchmark and per-job on full-parallel stars (Theorem 14). Best-of-N forks amplify Theorem 4: t tenants at fork width b give ratio $\Omega(tb)$ (Corollary 42, Appendix B.14). Proofs are in Appendix B.

6 Experiments

The experiments ask three questions: when can admission use a cached prefix, how large is the memory difference in public-trace groups, and what does discovering output lengths cost? We test earlier prefix publication in TGI, admission with reserved output caps, and GSA's repeated attempts. These are separate interventions, and each comparison identifies the one it changes. Makespan measures when the last request finishes; flow measures completion minus arrival for each request. Time to first token (TTFT) starts at request arrival, including queueing. Wall-clock comparisons use the same physical KV pool within each pair. A logical token budget and the engine's physical block pool are reported separately.

6.1 Making a resident prefix available at admission

At TGI commit `b4adbf2`, cached prefixes are published when a request completes. A 97-line modification publishes the completed prefill instead. It leaves the router's admission formula unchanged but makes the cached prefix available to later batch-building passes. Both arms run Qwen3-8B with the same flags and prefix caching on an A100-40GB. For the constrained cells, an identical reduced GPU memory fraction gives an inferred KV capacity of 3314 tokens in both arms. Table 1 reports three repetitions per setting. Figure 2 shows the individual runs and how the benefit changes with arrival density.

The constrained constructed batch changes from $1+12+3$ to $13+3$ completion waves; the natural-language batch changes from two waves to one. Greedy outputs match byte-for-byte on $45/48$ natural-language prompt–seed pairs; the remaining differences also arise from changing batch composition relative to a common single-request run. These measurements establish the total effect of earlier prefix publication, not an independently identified admission-only speedup. The loose-capacity control permits prefill reuse even when both arms have the same wave structure (Appendix H.1).

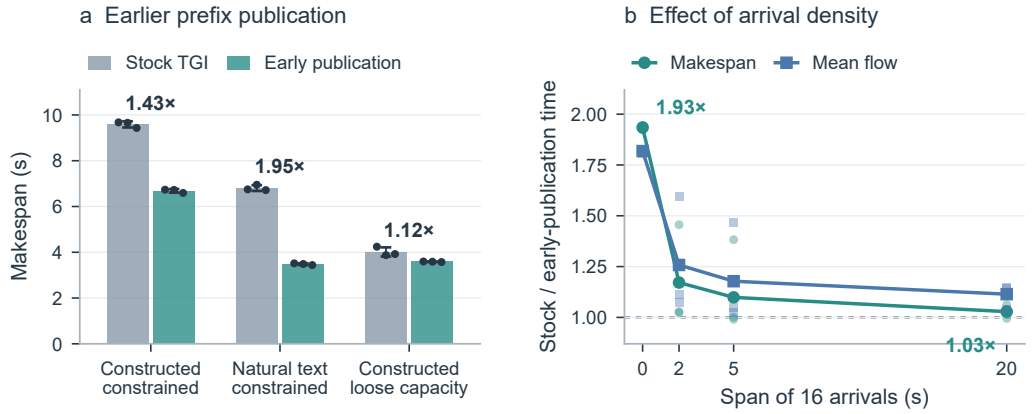

Figure 2: Earlier prefix publication in TGI (Qwen3-8B, A100-40GB, 16 requests, 64-token outputs). (a) Bars show mean makespan, whiskers show sample standard deviation, and dots show all three repetitions; labels are stock/patched ratios of means. (b) The same natural-language workload is spread over 0–20 seconds. Lines show ratios of three-run means; faint points show paired-seed ratios. The dashed line denotes equal time. Repetitions measure execution variability on fixed prompts. The intervention affects both admission and prefill reuse.

Arrival density. Spreading the same 16 natural-language requests over 0, 2, 5 and 20 seconds reduces makespan speedup to 1.93, 1.17, 1.10 and 1.03, respectively. The change follows the reduction in concurrent requests: at low overlap, admission has little contention to resolve (Appendix F).

6.2 Shared-prefix groups in a public trace

The Mooncake tool/agent hour contains 23,608 requests and measurable hashed prefix blocks (Qin et al., 2025a). Using 10-second windows, 51% of requests belong to groups with $k \geq 8$ and $W \geq k\bar{L}$. This is a group-structure statistic from one provider-hour, rather than a measured rate of simultaneous GPU residency. Lifetime reconstruction under specified service-time models gives different overlap estimates when timestamps are jittered (Appendix G).

For a controlled replay, we sample eight groups per budget from groups whose shared footprint fits $0.85M$ and whose additive footprint exceeds $1.5M$. Prompts preserve the trace's hashed-prefix structure; requests arrive together after prefix warmup, and output lengths are fixed to the realized trace lengths. Thus the static policies have oracle output lengths. At $M = 32$k, additive/coverage makespan is $2.22\times$; using a physical pool equal to M gives $2.21\times$. The native vLLM scheduler is within about 4% of the static coverage gate (Table A3); Figure 3(a,b) shows every selected group.

Scheduling cost beyond accounting. On eight uniformly sampled qualifying 64k-budget groups, geometric scheduling takes $1.77\times$ the makespan and $1.37\times$ the mean completion time of static coverage, with additional attempts and prefill. Static coverage knows realized lengths, so this is an oracle comparison rather than a contradiction of the unknown-length guarantee. It does show why a coverage-aware memory formula alone does not establish the practical superiority of GSA. Figure 3(c) shows the variation across the eight groups.

Preserving the arrival stream. We also retain every request and its published arrival time in the first five minutes of each Mooncake trace: 1,788 tool/agent requests and 918 conversation requests. With an imposed public cap of 512 tokens, a CPU service model gives a 2.28–6.00% mean-response reduction from coverage admission across two capacities (Appendix G.1). The model charges input-length-dependent prefill and fixed time per generated token; it excludes shared GPU compute contention. The published data contain prefix hashes and lengths, so this experiment studies queueing under those service assumptions rather than generation on the original prompts.

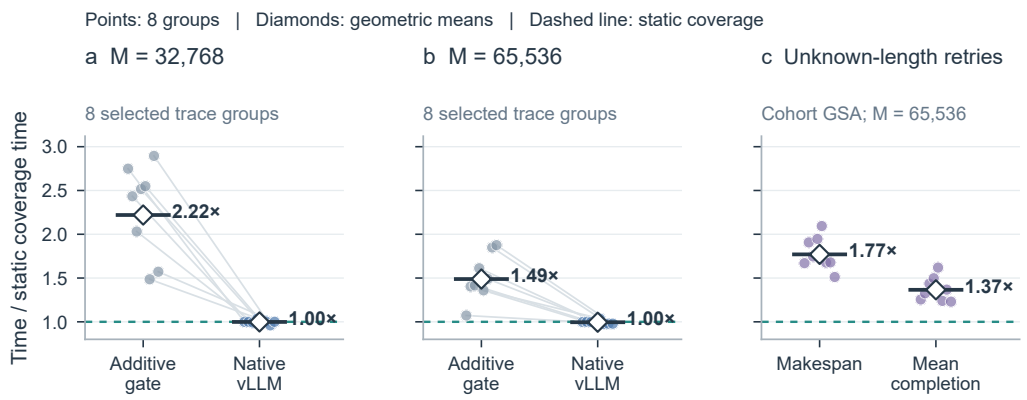

Figure 3: Selected Mooncake groups on vLLM (Qwen3-8B, A100-40GB). (a,b) Makespan relative to static coverage; thin lines connect the same group across additive admission and native vLLM. Each budget uses eight qualifying groups with warmed prefixes and simultaneous arrivals. (c) A separate eight-group replay measures cohort GSA relative to static coverage, which knows realized output lengths. Each point is a ratio of three-run means; diamonds are geometric means across groups. Values above one indicate longer time than static coverage. The physical KV pool is $1.5M$ in every arm; group selection and fixed-length replay details are in Appendix G.

6.3 Complete-prompt admission under a fixed physical pool

Admission charges the entire tokenized prompt, including each private question or instruction suffix. Only complete common KV pages are shared; a partial prefix page is charged with the private suffix. The physical pool, allocator-reserved pages and capacity available to admission are recorded separately (Appendix F). Every policy sees the same prompt tokens and public output cap, and receives completion events during execution. The controlled length fixture supplies stopping events through the environment; natural generation instead runs to EOS or the public cap.

Appendix J separately examines four-branch applications on native vLLM, including exact voting, cancellation and conditional submission.

7 Discussion and Conclusion

Storing a shared prefix once does not ensure that admission counts it once. A policy using only prompt lengths and generation progress cannot tell which memory is shared. For one resident common prefix, providing that information has a simple consequence: subtract its footprint and schedule generation on the remaining memory. For requests without private prompt suffixes, we prove a $76/3$ competitive ratio for staggered Coverage-GSA.

This guarantee concerns decode ticks and restarts under unknown output lengths. The online extension bounds total completion time, which includes arrival timestamps. It does not bound arrival-to-completion latency by the same factor: short requests can wait behind a long active batch (Appendix B.13). With unlimited memory, serving every released request continuously is optimal without predicting its length. Any remaining delay from the epoch policy then comes from its batching and restarts.

Real engines also incur private-suffix prefill, block rounding, cache eviction and batch-dependent compute costs. The TGI experiments show that exposing cached prefixes earlier can improve admission and completion time. The benefit shrinks when arrivals overlap less or native admission already counts shared blocks. Combining the scheduling guarantee with these execution costs remains the central systems challenge.

AI USE STATEMENT

AI assistants were used to assist with manuscript revision, code development and diagnostic calculations. The accompanying source and experiment records identify the implemented procedures and reported measurements.

REFERENCES

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming Throughput-Latency Tradeoff in LLM inference with Sarathi-Serve. In *OSDI*, 2024.

Ruicheng Ao, Gan Luo, David Simchi-Levi, and Xinshang Wang. Optimizing LLM inference: Fluid-guided online scheduling with memory constraints. *arXiv preprint arXiv:2504.11320*, 2025.

Agrim Bari, Parikshit Hegde, and Gustavo de Veciana. Optimal scheduling algorithms for LLM inference: Theory and practice. *POMACS*, 2025.

Tianle Cai et al. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

Kexin Chu, Zixu Shen, Yiwei Yang, et al. Selective KV-cache sharing to mitigate timing side-channels in LLM inference, 2025.

J.G. Dai, Tianze Deng, Yueying Li, and Tianyi Peng. Throughput-optimal scheduling algorithms for LLM inference and AI agents. *arXiv preprint arXiv:2504.07347*, 2025.

Ian D'Ambrosio. A tight linear deterministic competitive ratio for fully online KV-cache scheduling. *arXiv preprint arXiv:2608.16944*, 2026.

Gregory Dexter, Shao Tang, Ata Fatahi Baarzi, Qingquan Song, Tejas Dharamsi, and Aman Gupta. LLM query scheduling with prefix reuse and latency constraints. In *NeurIPS*, 2025.

Yiding Feng, Siyu Liu, Zonghan Yang, and Yuhao Zhang. General non-clairvoyant KV-cache scheduling via regime-aware routing. *arXiv preprint arXiv:2607.09248*, 2026a.

Yiding Feng, Zonghan Yang, and Yuhao Zhang. Competitive non-clairvoyant KV-cache scheduling for LLM inference. *arXiv preprint arXiv:2601.22996*, 2026b.

Patrick Jaillet et al. Online scheduling for LLM inference with KV cache constraints. *arXiv preprint arXiv:2502.07115*, 2025.

Kyoungmin Kim, Jiacheng Li, Kijae Hong, Qunyou Liu, Darong Huang, and Anastasia Ailamaki. Saving GPU hours in LLM inference system development and online workloads with simulation and DBMS-inspired cache replacement policies, 2026. URL https://arxiv.org/abs/2411.07447v5. Version 5, July 5, 2026.

Li Kong, Qi Qi, Yinyu Ye, and Zijie Zhou. Geometry-aware online scheduling for LLM serving: From theoretical bound to system practice, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *SOSP*, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *ICML*, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*, 2020.

Rui Pan, Zhuang Wang, Zhen Jia, Can Karakus, Luca Zancato, Tri Dao, Yida Wang, and Ravi Netravali. Marconi: Prefix caching for the era of hybrid LLMs. In *Proceedings of Machine Learning and Systems*, 2025a. URL https://arxiv.org/abs/2411.19379v3.

Zaifeng Pan, Ajjkumar Patel, Yipeng Shen, Zhengding Hu, Yue Guan, Wan-Lu Li, Lianhui Qin, Yida Wang, and Yufei Ding. KVFlow: Efficient prefix caching for accelerating LLM-based multi-agent workflows. In *NeurIPS*, 2025b.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. QuALITY: Question answering with long input texts, yes! In *Proceedings of NAACL-HLT*, pp. 5336–5358. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.naacl-main.391. URL https://aclanthology.org/2022.naacl-main.391/.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Aashaka Shah, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. In *ISCA*, 2024.

Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Feng Ren, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: Trading more storage for less computation — a KVCache-centric architecture for serving LLM chatbot. In *FAST*, pp. 155–170, 2025a.

Ziran Qin, Yuchen Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, Weiyao Lin, and Jianguo Li. CAKE: Cascading and adaptive KV cache eviction with layer preferences. In *ICLR*, 2025b.

Xingyu Qu, Tianhao Lin, Yiqi Li, Zhiyu Chen, and Sheng Wang. PRISM: Fast online LLM serving via scheduling-memory co-design. *arXiv preprint arXiv:2605.08581*, 2026.

Edi Shmueli and Dror G. Feitelson. Backfilling with lookahead to optimize the packing of parallel jobs. *Journal of Parallel and Distributed Computing*, 65(9):1090–1107, 2005.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Vikranth Srivatsa, Zijian He, Reyna Abhyankar, Dongming Li, and Yiying Zhang. Preble: Efficient distributed prompt scheduling for LLM serving. In *ICLR*, 2025.

Meixuan Wang, Yinyu Ye, and Zijie Zhou. LLM serving optimization with variable prefill and decode lengths, 2025.

Yuxin Wang, Yuhan Chen, Zeyu Li, et al. BurstGPT: A real-world workload dataset to optimize LLM serving systems. *arXiv preprint arXiv:2401.17644*, 2024.

Fangzhou Wu, Sandeep Silwal, and Qiuyi Zhang. Randomization boosts KV caching, learning balances query load: A joint perspective. In *ICLR*, 2026.

Jialong Wu, Zhenglin Wang, Linhai Zhang, Yilong Lai, Yulan He, and Deyu Zhou. SCOPE: Optimizing key-value cache compression in long-context generation. In *ACL*, 2025.

An Yang et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Lu Ye, Ze Tao, Yong Huang, and Yang Li. ChunkAttention: Efficient self-attention with prefix-aware KV cache and two-phase partition. In *ACL*, 2024.

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 *OSDI*, 2022.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. WildChat: 1m ChatGPT interaction logs in the wild. In *International Conference on Learning Representations (ICLR)*, 2024.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *NeurIPS*, 2024.

Zhen Zheng, Xin Ji, Taosong Fang, Fanghao Zhou, Chuanjie Liu, and Gang Peng. BatchLLM: Optimizing large batched LLM inference with global prefix sharing and throughput-oriented token batching. In *Proceedings of the 9th MLSys Conference, Industry Track*, 2026. URL https://arxiv.org/abs/2412.03594.

A WORKED EXECUTION

A three-request execution. Consider three requests with the same resident prefix and no private prompt suffixes. After subtracting the shared prefix, the remaining capacity is $R = 5$ tokens. Their output lengths happen to be $(L_1, L_2, L_3) = (1, 3, 5)$; the scheduler observes a length only when that request completes. GSA sets attempt lengths using growth factor $\alpha_g = 2$. Its slice sequence uses $\ell = \lfloor \log_2 5 \rfloor = 2$ and $\beta = 5/2^2 = 5/4$, giving $\tau_p = \lfloor \beta 2^p \rfloor = (1, 2, 5)$. The last slice is exactly R, rather than an unbounded continuation of the integer ladder $1, 2, 4, 8$.

Phase	Slice	Execution intervals	Peak private memory	Observation
0	1	requests 1,2,3: $[0, 1)$	3	request 1 finishes at 1
1	2	requests 2,3: $[1, 3)$	4	both remain unfinished
2	5	request 2: $[3, 6)$; request 3: $[8, 13)$	5	finish at 6 and 13

The staggered schedule allows widths $5, 3, 1$ in the three phases; only the remaining requests are started. Unfinished requests restart at age zero between attempts. In the last phase, width one fixes the second request's start offset at 5 ticks, so the scheduler is idle on $[6, 8)$ after request 2 finishes. This interval is part of the staggered schedule; an implementation that starts the next cohort immediately is a different timing rule and should be identified explicitly. The completion vector is $(1, 6, 13)$, total flow is 20, and every tick uses at most the residual capacity 5. The example shows when requests finish, when attempts restart, and how the memory check determines their start times. The competitive bound compares this scheduling rule with the optimum across all feasible output lengths.

B PROOFS AND SCHEDULING EXTENSIONS

Unless noted, theory plots and computational checks in the appendices use the long-prefix family $W{=}kL$, $M{=}W{+}kL$, $\alpha{=}1$.

B.1 PROOF OF LEMMA 2 AND THEOREM 4

Lemma 2. Fix $\alpha = 1$, $W \geq kL$, $M = W + kL$, $k \geq 2$. Two jobs at *any* ages $a, b \geq 0$ have additive look-ahead cost $2W + a + b + 2 \geq 2W + 2 > W + kL = M$ (using $W \geq kL$). Hence every additive look-ahead-feasible schedule serves at most *one* job per tick on this family; in particular at most one job can advance $L{-}1 \to L$ per tick.

Theorem 4. Coverage look-ahead admits the full set at every *pre-completion* age $a \in \{0, \ldots, L-1\}$: the look-ahead cost of advancing all k jobs is $W + k(a+1) \leq W + kL = M$, with equality exactly at the final transition $a = L{-}1 \to L$, upon which every job has decoded its L tokens and is removed. Hence OPT flow is kL. Any additive *completion-feasible* schedule obeys $|S| \leq s^\star = \lfloor M/(W+L) \rfloor$. For $W \geq kL$, $1 \leq (W + kL)/(W + L) < 2$, hence $s^\star = 1$ (and at $W = kL$, $\lfloor 2k/(k+1) \rfloor = 1$). Kill-and-restart then forces uninterrupted unit cohorts with flow $L \cdot k(k+1)/2$ and ratio $(k+1)/2 = \Omega(k)$. Any additive look-ahead-feasible schedule serves at most one job per tick (Lemma 2), and the flow-optimal width-1 kill-restart schedule is the uninterrupted serial one, so the same $L{\cdot}k(k{+}1)/2$ lower bound applies. Coverage simultaneous batching matches OPT. Ratios $2.5, 4.5, 8.5, 16.5, 32.5$ at $k \in \{4, 8, 16, 32, 64\}$.

B.2 PROOF OF LEMMA 3

Proof. Fix any reachable state with views $\{(W, a_i)\}$ and consider the *disjoint* instance $I_\perp$: k requests with pairwise-disjoint prompts, each of length W, at the same ages. The policy sees the same observations on the star and on $I_\perp$ and therefore admits the same set S_t. On $I_\perp$, shared-memory and additive costs coincide. Memory safety requires $\sum_{i \in S_t} (W + a_i + 1) \leq M$. The same inequality then constrains its behavior on the star. □

What changes when sharing information is available. Lemma 3 applies to policies that cannot distinguish shared prefixes from disjoint prompts. To remain safe in either case, they must use the additive memory constraint. Theorem 4 then gives an $\Omega(k)$ loss on the long-prefix family. Prefix IDs, cache-residency bits, group summaries, or conservative reuse estimates can distinguish some shared layouts from disjoint ones. Such observations place a policy outside the lemma's stated information restriction. Recovering part of the $W(k{-}1)$ overcount may permit more concurrent requests. The results establish two cases: no overlap information gives the lower bound, while exact common-prefix information permits the reduction in Theorem 6. A quantitative guarantee for partial or inaccurate information requires a model of that information.

B.3 Proof of Proposition 5

Lemma 2 permits at most one request to advance per tick on the long-prefix family, regardless of whether inactive progress is reset or retained. Completing all requests requires at least kL service ticks, so additive makespan is at least kL. Continuous simultaneous coverage execution completes every request at L. Charging retained private KV for paused requests further restricts the additive feasible set and cannot weaken this lower bound; the coverage schedule pauses no request. This argument does not identify the active-set retention relaxation with an engine's pinned-storage model.

B.4 Proof of Theorem 1 (asymptotic form)

Counting cohort completion times gives the exact Φ formula. It remains to establish $F_{\mathrm{add}}/F_{\mathrm{cov}} = \Theta(1 + \min\{W/L, k\})$ with universal constants. Write $x := W/L \geq 0$ and note

$$s^\star = \left\lfloor \frac{W + kL}{W + L} \right\rfloor = \left\lfloor \frac{x + k}{x + 1} \right\rfloor, \qquad \frac{x + k}{x + 1} - 1 < s^\star \leq \frac{x + k}{x + 1}.$$

Upper bound. Using $\Phi(k, s) \leq \sum_{i=1}^{k} (\frac{i}{s} + 1) = \frac{k(k+1)}{2s} + k$,

$$\frac{\Phi(k, s^\star)}{k} \leq \frac{k + 1}{2s^\star} + 1.$$

If $x \leq k$ then $s^\star > \frac{x+k}{x+1} - 1 = \frac{k-1}{x+1} \geq \frac{k-1}{2\min\{x,k\}+1}$ (since $x + 1 \leq 2x + 1$ and $x \leq k$ here), so for $k \geq 2$,

$$\frac{k + 1}{2s^\star} \leq \frac{(k + 1)(2\min\{x, k\} + 1)}{2(k - 1)} \leq 3\,(1 + \min\{x, k\}),$$

and the ratio is at most $4(1+\min\{x, k\})$. If $x > k$ then $s^\star \geq 1$ trivially and $\frac{\Phi(k,s^\star)}{k} \leq \frac{k+1}{2} + 1 \leq 2(1+k) = 2(1 + \min\{x, k\})$.

Lower bound. Each cohort of size $\leq s^\star$ occupies L ticks, so cohorts run serially and

$$\frac{\Phi(k, s^\star)}{k} \;\geq\; \frac{1}{2}\left\lceil \frac{k}{s^\star} \right\rceil \;\geq\; \frac{k}{2s^\star} \;\geq\; \frac{k(x + 1)}{2(x + k)}.$$

For $x \leq k$: $\frac{k(x+1)}{2(x+k)} \geq \frac{k(x+1)}{4k} = \frac{x+1}{4} = \frac{1}{4}(1 + \min\{x, k\})$. For $x > k$: $\frac{k(x+1)}{2(x+k)} \geq \frac{k \cdot x}{2 \cdot 2x} = \frac{k}{4} \geq \frac{1+k}{8} = \frac{1}{8}(1+\min\{x, k\})$. Hence the ratio is at least $\frac{1}{8}(1+\min\{x, k\})$ throughout (numerically verified over $k \leq 256$, $x \leq 10k$).

Consequences. Fix $c = W/(kL) < 1$ and let k grow. Then $x = ck$ gives ratio $\Theta(k)$: a shorter prefix changes the constant but not the order of growth. For example, at $c = 0.25$, $s^\star = \lfloor \frac{(c+1)k}{ck+1} \rfloor \to 4$. The expression inside the floor approaches $\frac{c+1}{c}$ from below, as in the evaluated instances. The organizing role of $W/(kL)$ in the figures is as a normalized position within a fixed-k family; the competitive scaling is carried by $(W/L, k)$. □

Class scope. The asymptotic form is proved for completion-feasible (peak-reserved cohort) schedules; at $W \geq kL$ it extends to the full look-ahead-feasible class via Lemma 2. For intermediate W, it remains open whether schedules that check memory at each changing age can improve on this cohort bound. Staggered starts improve coverage schedules (Theorem 11(c)); the corresponding additive case is unresolved.

B.5 Short-prefix counterexample

At W=10, L=4, k=32, M=138, the additive width is $s^\star$=9 and the best-additive ratio is 2.3125. The long-prefix condition $W \geq kL$ of Theorem 4 does not hold here. A greedy policy favoring smaller current progress gives ratio 1026.5× on this instance, but that failure concerns the policy, not every additive schedule.

Theorem 11 (Coverage lockstep under kill-and-restart)**.** *Under kill-and-restart, some optimum is* atomic*: pausing resets progress to* 0*, so started jobs may be assumed to run uninterrupted (Lemma 12). For identical private lengths L with residual $R = M - W$ and $b^\star = \lfloor R/L \rfloor$, the $b^\star$-cohort lockstep schedule realized by peak-reserved coverage is* exactly *optimal in two regimes: (a) full parallel ($b^\star \geq k$; flow kL) and (b) tight residual ($R = L$; serial flow $L \cdot k(k + 1)/2$, since completions are $\geq L$ ticks apart). (c) At some intermediate budgets ($L < R < kL$), staggered pipelines strictly beat lockstep (e.g. k=5, L=3, R=7: OPT $23 < 27 = F^\star$; Appendix B.6), but lockstep remains a constant-factor benchmark: $F^\star \leq 4\,\mathrm{VLB} + kL \leq 5 \cdot \mathrm{OPT}$ (Lemma 13).*

B.6 When simultaneous cohorts are optimal

Setup. Fix identical private length L, budget M, residual $R = M - W$, and peak-reservation width $b^\star = \lfloor R/L \rfloor$. A kill-and-restart job needs L consecutive service ticks in its completing attempt. Any shorter attempt interrupted before completion is discarded. A *lockstep* schedule starts each cohort together and waits for it to finish before starting the next. Its flow is $F^\star := L \cdot \Phi(k, b^\star)$, with Φ as in Theorem 1.

Lemma 12 (Atomicity of OPT)**.** *Every feasible kill-restart coverage schedule can be transformed, without increasing any completion time, into an* atomic *schedule in which every started job runs uninterrupted to completion.*

Proof. Let i be the earliest interrupted job, at tick t_0. Its preceding uninterrupted service interval $[s_0, t_0)$ contains $t_0 - s_0 < L$ discarded ticks. Delete that interval for i, keeping every other job's schedule fixed. The deletion completes no job and only reduces memory use in (7), so no completion time increases. Repeat for all interrupted intervals. The argument allows different lengths and also justifies the atomic-schedule solver used in the finite-instance checks. □

(a) Full parallel ($b^\star \geq k$): lockstep is exact. Every job needs L service ticks, so flow $\geq kL$; simultaneous coverage batching achieves $C_i = L$ for all i, hence OPT $= kL = F^\star$.

(b) Tight residual ($R = L$): serial is exact. At a completion tick the finishing job has look-ahead weight $L = R$, so *no other job is served that tick*; every other unfinished job is therefore killed to age 0 and needs at least L further ticks before its own completion. Also no two jobs can finish at the same tick ($2L > R$), and the first completion needs $\geq L$ ticks. Hence consecutive completion times are $\geq L$ apart: $C_{(r)} \geq rL$, so flow $\geq L \cdot k(k+1)/2$, which the serial ($b^\star$=1 lockstep) schedule achieves. OPT $= F^\star$.

(c) Intermediate budgets ($L < R < kL$): lockstep can be suboptimal. In (7), a request's memory grows with its current age. Starting jobs at different times can therefore use spare memory that fixed-width, simultaneous cohorts leave idle. The fixed-slot exchange argument does not establish lockstep optimality under this constraint. Explicit counterexamples (exact atomic-schedule OPT via Lemma 12): k=5, L=3, R=7: OPT $= 23 < 27 = F^\star$ (two age-2 jobs at weight 3+3 leave slack 1 for an age-0 filler that banks progress); k=5, L=3, R=6 (divisible $L \mid R$): OPT $= 25 < 27 = F^\star$ via the *staircase* pipeline (ages $2, 1, 0$ at weights 3+2+1 = 6, one completion per tick after warm-up); k=3, L=2, R=3 ($b^\star$=1): OPT $= 9 < 12 = F^\star$; k=8, L=4, R=8: OPT $= 72 < 80$; k=9, L=5, R=15: OPT $= 81 < 90$. These examples show how staggered starts improve memory use at intermediate budgets.

Lemma 13 (Lockstep benchmark quality)**.** *For all identical-star instances,* $F^\star \leq 4\,\mathrm{VLB} + kL \leq 5 \cdot \mathrm{OPT}$.

Proof. $\Phi(k, b) \leq \sum_{i=1}^{k} \left(\frac{i}{b} + 1\right) = \frac{k(k+1)}{2b} + k$, so $F^\star \leq \frac{Lk(k+1)}{2b^\star} + kL$. For identical lengths, Lemma 29 gives $\mathrm{VLB} = \frac{L(L+1)}{2R} \cdot \frac{k(k+1)}{2}$. Since $R < (b^\star + 1)L \leq 2b^\star L$,

$$\frac{Lk(k+1)/(2b^\star)}{\mathrm{VLB}} = \frac{2R}{b^\star(L+1)} < \frac{4L}{L+1} < 4.$$

Hence $F^\star < 4\,\mathrm{VLB} + kL$; and OPT $\geq \max(\mathrm{VLB}, kL)$ yields $F^\star \leq 5 \cdot \mathrm{OPT}$. (Observed gaps are far smaller: $F^\star/\mathrm{OPT} \leq 1.34$ on all computationally checked counterexamples.) □

Scope of part (c). Lemma 12 uses that an interrupted block is pure waste (reset to age 0); under pause-*retain* the deletion step is unavailable, which is why Proposition 5 states a makespan lower bound. Part (c) concerns the coverage class only: the additive completion-feasible class is width-capped ($|S| \leq s^\star$ at every tick, independent of ages), where the slot exchange is valid, so the best-additive schedules in Theorems 4 and 1 remain $s^\star$-cohort lockstep.

B.7 The GSA model and the shared-prefix reduction

To compare Corollary 7 with the guarantee of Feng et al. (2026b), we restate their model and theorem hypotheses, then verify each hypothesis at the boundary $s = 0$ used by the reduction.

The model of Feng et al. (2026b). Discrete rounds $t = 0, 1, \ldots$; a single accelerator with KV budget $M_{\mathrm{GSA}} \in \mathbb{N}$. There are n jobs, all present at time 0 (offline batch); every job has the *same* prompt length $s \geq 0$ and an unknown response length $o_i \geq 1$ with per-job feasibility $s + o_i \leq M_{\mathrm{GSA}}$. An active job that has decoded u tokens occupies $s + u + 1$ units; the batch constraint is $\sum_{i \in B_t} (s + u_{i,t} + 1) \leq M_{\mathrm{GSA}}$ at every round. Actions: start an inactive job (at $u = 0$), or kill an active unfinished job, discarding all progress (kill-and-restart; no pause/swap). Objective: total completion time $\sum_i C_i$. A schedule is non-clairvoyant if its decisions do not depend on the unrevealed o_i.

Staggered and geometric schedules. The scheduler spaces request starts over time and gives each request a fixed-length attempt. In the Staggered Pipeline Schedule (SPS) of Feng et al. (2026b), $\mathrm{SPS}(\mathcal{T}, k, \tau)$ starts job i of $\mathcal{T}$ at round $\lfloor i\tau/k \rfloor$ and kills it τ rounds later if unfinished; its exact peak occupancy is

$$\mathrm{Peak}(k, \tau, s) = sk + \tfrac{1}{2}\big(\tau k + \tau + k - \gcd(\tau, k)\big) \qquad \text{(GSA Lem. 3.3)}$$

and the per-phase width is $k^\star(\tau, s) = \max\{k : \mathrm{Peak}(k, \tau, s) \le M_{\mathrm{GSA}}\}$ (Feng et al., 2026b, Lem. 3.4). GSA (Feng et al., 2026b, Alg. 2) fixes $\alpha_g > 1$ and sets $\ell = \lfloor \log_{\alpha_g}(M_{\mathrm{GSA}} - s) \rfloor$ and $\beta = (M_{\mathrm{GSA}} - s)/\alpha_g^\ell$. Here $1 \le \beta < \alpha_g$, and the final slice equals $M_{\mathrm{GSA}} - s$. Phase $p = 0, 1, \ldots$ runs SPS on all unfinished jobs with slice $\tau_p = \lfloor \beta \alpha_g^p \rfloor$ and width $k^\star(\tau_p, s)$. Jobs still unfinished at phase end are killed.

The prior GSA guarantee. Feng et al. (2026b, Thm. 3.15): For every $\alpha_g > 1$, on every instance of the model above (offline batch, identical prompts s, unknown o_i, per-job feasibility, kill-and-restart),

$$\Gamma(\mathrm{GSA}) \;\le\; \Big(2 + \tfrac{2}{\alpha_g - 1}\Big)\Big(\alpha_g^2\big(1 + \tfrac{2}{k_{\min}}\big) + \alpha_g + \tfrac{\alpha_g}{\alpha_g - 1}\Big),$$

where $k_{\min} \ge 1$ is the instance's minimum SPS parallelism. At $\alpha_g = (7 + \sqrt{13})/6$ this is ≤ 61.92 for all $k_{\min} \ge 1$; at $\alpha_g = 2$ it tends to 32 as $k_{\min} \to \infty$. The hypotheses are exactly: (H1) all jobs present at $t = 0$; (H2) identical prompt length $s \ge 0$ across jobs; (H3) unknown response lengths with $s + o_i \le M_{\mathrm{GSA}}$; (H4) kill-and-restart semantics; (H5) integer rounds and slices; (H6) $M_{\mathrm{GSA}} - s \ge 1$ so the ladder is defined.

Theorem 6: the same feasible schedules. Both models schedule the same ground set of jobs in discrete ticks with kill-and-restart and objective $\sum_i C_i$; only the per-tick feasibility predicate differs. Every active request uses the shared prefix, so its set S_t must satisfy $W + \sum_{i \in S_t}(a_i + 1) \le M$. Subtracting W gives $\sum_{i \in S_t}(a_i + 1) \le R := M - W$. This is exactly GSA's memory test at $s = 0$, $M_{\mathrm{GSA}} = R$: each job occupies $0 + u_{i,t} + 1 = a_i + 1$ units. Both tests accept the same selected sets at every tick. Keeping the schedule unchanged preserves every completion time and the total flow. The feasible optima and all competitive ratios are therefore identical. We state the isomorphism at $\alpha = 1$ (the paper's default and the unit used by the GSA model); for general bytes-per-age one rescales the budget axis to token units before applying it. □

Hypothesis check at $s = 0$ (Corollary 7). The star instance under the identity map satisfies each hypothesis: (H1) our model is the same offline batch (all k jobs at $t = 0$); (H2) prompt length 0 is identical across jobs—this is where the reduction *uses* the star structure: heterogeneity lives entirely in the response lengths $L_i = o_i$, which the original GSA model already allows; (H3) $L_i \le R$ iff $0 + o_i \le M_{\mathrm{GSA}}$, the corollary's stated hypothesis; (H4) both models kill to age 0, no pause; (H5) ticks and slices are integers on both sides; (H6) $R \ge L_{\max} \ge 1$. The GSA analysis also applies at $s = 0$. The peak formula $\mathrm{Peak}(k, \tau, 0) = \frac{1}{2}(\tau k + \tau + k - \gcd(\tau, k)) \ge 1$ remains exact and increasing in k. A single request fits whenever $\tau \le R$, so $k^\star(\tau, 0) \ge 1$; its executed weight never exceeds $L_{\max} \le R$. Condition H6 defines $\ell = \lfloor \log_{\alpha_g} R \rfloor$. The area $A(o, s) = so + o(o + 1)/2$ becomes $o(o + 1)/2 > 0$. Finally, the constant in Theorem 3.15 depends on α_g and $k_{\min}$, not on s. (At $s = 0$ the width $k^\star$ is *larger* than at any $s > 0$, so $k_{\min}$ can only increase, which improves—never degrades—the bound.) Hence the imported ratio applies verbatim against coverage OPT on the star. □

Corollary 7 (derivation). GSA run at $(s{=}0,\ R)$ is a non-clairvoyant algorithm of the GSA model restated above; Thm. 3.15's formula at $\alpha_g = (7 + \sqrt{13})/6$ and $k_{\min} \ge 1$ evaluates to ≤ 61.92, and $\to 32$ at $\alpha_g = 2$ as $k_{\min} \to \infty$ (large residual). By Theorem 6 and the hypothesis check, the same ratio holds verbatim against coverage OPT on the star. The hypothesis $L_{\max} \le R$ is the per-job feasibility any algorithm needs. The guarantee allows different unknown lengths and every feasible remaining budget $R \ge L_{\max}$, including $R = L_{\max}$. □

Corollary 10. Verbatim additive GSA at $(s{=}W, M)$ schedules within its exact staggered-pipeline peak $\mathrm{Peak}(k, \tau, s) = sk + \frac{1}{2}(\tau k + \tau + k - \gcd(\tau, k)) \le M$ (Feng et al., 2026b, Lem. 3.3), so its per-tick additive memory never exceeds M: it is additive look-ahead feasible. On the long-prefix family, Lemma 2 gives width ≤ 1 at every tick; alternatively $\mathrm{Peak}(2, \tau, W) \ge 2W + \tau + 1 > W + kL = M$ shows $k^\star(\tau, W) = 1$ for every slice directly from GSA's own width formula. A width-one kill-restart schedule has flow $\ge L \cdot k(k + 1)/2$ (Theorem 4), giving ratio $\ge (k + 1)/2$ versus coverage OPT $= kL$. The same reasoning applies to the router of Feng et al. (2026a). Every batch satisfies $\sum_{i \in B_t}(s_i + u_{i,t} + 1) \le M$ in its additive model. At $s_i = W$, Lemma 2 forces width one on the long-prefix family, regardless of the router's internal choice. □

Computational checks (released with the code). We compare GSA with greedy packing in descending order of current progress, denoted RPP in the formulas and code (Algorithm 1). The implementation uses the exact SPS peak formula and GSA's phase loop, with $\tau_p = \lfloor \beta \alpha_g^p \rfloor$ and the final slice capped at the remaining budget. It checks memory at each tick under both accounting rules. `prove_gsa_longspine_victim`: on $W{=}kL$, $M{=}W{+}kL$, verbatim-additive GSA has width-one phases at every k and flow ratios

7.9/17.8/41.2/44.5 at k=4/8/16/32 ($\geq k/2$; the excess over $(k+1)/2$ is geometric-phase waste), while Coverage-GSA measures 2.2–3.9. `prove_covgsa_constant_hetero_tightm`: over 160 random heterogeneous tight-M stars, Coverage-GSA's ratio against the rigorous VLB floor (Lemma 29; VLB $\leq$ OPT, so this bounds the true ratio) is $\leq$ 7.7, consistent with (and far inside) the inherited 61.92; on the k=16 RPP counterexample family of Proposition 31, Coverage-GSA's flow is 587 versus RPP's 1123. `prove_gsa_no_sharing_sanity`: at W=0 the two memory maps coincide, as the isomorphism predicts.

Theorem 14 (Non-clairvoyant coverage doubling)**.** (a) Identical stars. *Probe doubling ($\widehat{L} = 2^p$; each unsuccessful phase runs a single width-b probe cohort for 2^p ticks, which under identical lengths reveals whether $L \leq 2^p$) is $\leq$ 4-competitive against the lockstep benchmark $F^\star$ (hence $\leq$ 4-competitive vs OPT in the exact regimes of Theorem 11, and $\leq$ 20-competitive vs OPT at intermediate M via Lemma 13).* (b) Heterogeneous full-parallel. *On a k-star with unknown private lengths L_i, $L_{\max} = \max_i L_i$, and $M \geq W + \alpha k L_{\max}$ (coverage full-parallel feasible through completion), non-clairvoyant coverage doubling satisfies $C_i \leq 4L_i$ per job, hence $\sum_i C_i \leq 4\sum_i L_i \leq 4 \cdot \mathrm{OPT}$.*

B.8 A SELF-CONTAINED CONSTANT FOR COVERAGE-GSA AT $s = 0$

We first prove a $140/3 < 46.7$ competitive ratio for a family of schedules that use doubling attempt durations. The family includes both simultaneous cohorts and staggered starts. A sharper width bound then gives $76/3 < 25.34$ for staggered GSA with growth factor $\alpha_g = 2$. Both results apply after subtracting the resident shared prefix. The proof uses the volume lower bound in Lemma 29, proved in Appendix B.12, and does not invoke the GSA competitive theorem. The 61.92 prior bound remains available at GSA's tuned growth factor. The same argument also improves the $O(\log L_{\max})$ bound for cohort doubling in Lemma 32 to a constant.

Setting. Throughout this subsection we work in the *prompt-free residual model*: k jobs, all present at $t = 0$, with unknown integer private lengths $1 \leq L_i \leq R$; discrete ticks; an active job at age a has look-ahead weight $a + 1$; a set S_t may be run at a tick iff

$$\sum_{i \in S_t} (a_i + 1) \;\leq\; R, \tag{1}$$

i.e. exactly (7); unselected unfinished jobs are reset to age 0 (kill-and-restart); the objective is total flow $\sum_i C_i$. By Theorem 6 this *is* coverage scheduling on a star with residual $R = M - W$, so every ratio proven here holds verbatim against coverage OPT on stars. A schedule is non-clairvoyant if its decisions depend only on R and on observed completions, never on unrevealed L_i.

We use two lower bounds on OPT, both already established: $\mathrm{OPT} \geq \sum_i L_i$ (every job needs L_i service ticks), and the volume lower bound of Lemma 29,

$$\mathrm{OPT} \;\geq\; \mathrm{VLB} \;=\; \frac{1}{R}\sum_{i=1}^{k}(k-i+1)\binom{L_{(i)}+1}{2}, \qquad L_{(1)} \leq \cdots \leq L_{(k)}.$$

Doubling ladders. Call an integer sequence $\tau_0, \tau_1, \ldots$ a *doubling ladder* if

$$\tau_0 = 1, \qquad 2\,\tau_{p-1} \;\leq\; \tau_p \;\leq\; 2\,\tau_{p-1} + 1 \quad (p \geq 1). \tag{2}$$

Two ladders matter to the paper: (i) the *pure* ladder $\tau_p = 2^p$ of residual-slot doubling (Lemma 32); (ii) the *GSA ladder at* $\alpha_g = 2$: $\ell = \lfloor \log_2 R \rfloor, \beta = R/2^\ell \in [1, 2), \tau_p = \lfloor \beta\, 2^p \rfloor$ for $p = 0, \ldots, \ell$, whose final slice is $\tau_\ell = R$ exactly. Both satisfy (2): $\tau_0 = \lfloor \beta \rfloor = 1$, and for any real $x \geq 1$, $\lfloor 2x \rfloor \geq 2\lfloor x \rfloor$ and $\lfloor 2x \rfloor \leq 2x < 2(\lfloor x \rfloor + 1)$, so $2\lfloor x \rfloor \leq \lfloor 2x \rfloor \leq 2\lfloor x \rfloor + 1$ with $x = \beta 2^{p-1}$. (Checked exhaustively for all $R \leq 600$ in the verification script.)

The algorithm family. Fix a doubling ladder. The schedule runs phases $p = 0, 1, \ldots$ while unfinished jobs remain; phase p attempts *every* currently-unfinished job exactly once with reservation τ_p, killing it at the end of its reservation if unfinished, in one of two realizations:

- **Cohort realization** (the paper's `gsa_cov`-cohort arm and Lemma 32): the unfinished jobs are partitioned into cohorts of width $b_p = \max\{1, \lfloor R/\tau_p \rfloor\}$; cohorts run sequentially, each holding its reservation for τ_p ticks.
- **Staggered realization** (the staggered pipeline of GSA; the paper's token-level `gsa_cov` arm): require $\tau_p \leq R$ in every executed phase, as on the GSA ladder. The N_p unfinished jobs are started at offsets $\lfloor j\tau_p/k_p \rfloor$, $j = 0, \ldots, N_p - 1$, with width $k_p = \max\{k \geq 1 : \mathrm{Peak}(k, \tau_p) \leq R\}$, where Peak is bounded in Lemma 16 below.

With the GSA ladder at $\alpha_g = 2$ and the staggered realization this is precisely GSA run at prompt 0 on budget R—Coverage-GSA. Both versions use only observed completions. The analysis reserves the full duration of each phase or cohort, even if its requests finish early. Remark 3 treats implementations that advance as soon as a phase or cohort empties.

Step 1: within-phase feasibility.

Lemma 15 (Cohort feasibility)**.** *Every cohort tick satisfies* (1)*.*

Proof. A cohort of width $b \le b_p$ whose members are at common age $a \le \tau_p - 1$ has look-ahead weight $\le b_p(a+1) \le b_p\tau_p \le R$ when $b_p = \lfloor R/\tau_p \rfloor \ge 1$. If $\lfloor R/\tau_p \rfloor = 0$ (necessarily in the final phase, since $\tau_p > R \ge L_{\max}$), then $b_p = 1$ and a single job's weight never exceeds $L_i \le R$. Members that complete before the reservation ends only remove weight. □

For the staggered realization we prove the exact SPS peak at $s = 0$ from scratch. Recall the classical floor-sum identity: for integers $\tau, k \ge 1$ with $g = \gcd(\tau, k)$,

$$\sum_{m=0}^{k-1} \left\lfloor \frac{m\tau}{k} \right\rfloor = \frac{(\tau-1)(k-1)+g-1}{2}, \tag{3}$$

which follows from $\sum_m \lfloor m\tau/k \rfloor = \frac{\tau(k-1)}{2} - \frac{1}{k}\sum_m (m\tau \bmod k)$ and the fact that $m\tau \bmod k$ traverses each multiple of g in $\{0, g, \ldots, k-g\}$ exactly g times, so $\sum_m (m\tau \bmod k) = k(k-g)/2$.

Lemma 16 (SPS peak at $s = 0$, self-contained)**.** *Start n jobs at offsets $\lfloor j\tau/k \rfloor$ ($j = 0, \ldots, n-1$), each killed τ ticks after its start or removed earlier on completion. Then at every tick the total look-ahead weight is at most*

$$\mathrm{Peak}(k, \tau) \;:=\; \frac{k\tau + k + \tau - \gcd(\tau, k)}{2},$$

independently of n and of the completion pattern.

Proof. Fix a tick t and drop early completions (they only remove weight). Let j_0 be the least active index and $m = j - j_0 \ge 0$ for active j. By superadditivity of the floor, $\lfloor j\tau/k \rfloor \ge \lfloor j_0\tau/k \rfloor + \lfloor m\tau/k \rfloor$, so job j's age obeys $a_j \le (\tau-1) - \lfloor m\tau/k \rfloor$. Since $a_j \ge 0$ forces $\lfloor m\tau/k \rfloor \le \tau - 1$, i.e. $m \le k-1$, at most k jobs are active, and

$$\sum_{j \text{ active}} (a_j + 1) \;\le\; \sum_{m=0}^{k-1} \Big(\tau - \Big\lfloor \frac{m\tau}{k} \Big\rfloor\Big) \;=\; k\tau - \frac{(\tau-1)(k-1)+g-1}{2} \;=\; \mathrm{Peak}(k, \tau)$$

by (3). □

This is a direct proof of Lemma 3.3 of Feng et al. (2026b) at $s = 0$. The simulation reaches equality once $n \ge k$, using the same width rule.

Lemma 17 (The staggered width dominates the cohort width)**.** *For $1 \le \tau \le R$: $k_p := \max\{k \ge 1 : \mathrm{Peak}(k, \tau) \le R\}$ is well defined and $k_p \ge \lfloor R/\tau \rfloor$.*

Proof. $\mathrm{Peak}(1, \tau) = \tau \le R$, so $k_p \ge 1$. For $b = \lfloor R/\tau \rfloor \ge 1$: $\tau b \le R$ and, since $(\tau-1)(b-1) \ge 0$ gives $\tau + b \le \tau b + 1 \le R + 1$,

$$\mathrm{Peak}(b, \tau) = \frac{\tau b + \tau + b - \gcd(\tau, b)}{2} \le \frac{R + (R+1) - 1}{2} = R.$$ □

Step 2: phase durations and progress. Let N_p denote the number of unfinished jobs at the start of phase p, and let $q_i := \min\{p : \tau_p \ge L_i\}$ be job i's *class*. Since every unfinished job is attempted with reservation τ_p in every phase p, job i completes during phase q_i and not earlier; hence, exactly,

$$N_p = \#\{i :\; L_i > \tau_{p-1}\} = \#\{i :\; q_i \ge p\} \qquad (N_0 = k). \tag{4}$$

For the GSA ladder, $\tau_\ell = R \ge L_{\max}$ guarantees $q_i \le \ell$; the pure ladder needs no such cap—it simply stops when no jobs remain, at $p = \lceil \log_2 L_{\max} \rceil$ at the latest.

Lemma 18 (Phase duration)**.** *In either realization, phase p lasts at most $D_p \le 2N_p\tau_p^2/R + \tau_p$ ticks.*

Proof. Cohort. Duration is $\lceil N_p/b_p \rceil \tau_p \le N_p\tau_p/b_p + \tau_p$. If $\tau_p \le R$ then $b_p = \lfloor R/\tau_p \rfloor \ge R/(2\tau_p)$ (as $\lfloor x \rfloor \ge x/2$ for $x \ge 1$), so $N_p\tau_p/b_p \le 2N_p\tau_p^2/R$. If $\tau_p > R$ (the cohort realization's final phase) then $b_p = 1$ and $N_p\tau_p \le 2N_p\tau_p^2/R$ because $2\tau_p > R$.

Staggered. The phase ends by $\lfloor (N_p - 1)\tau_p/k_p \rfloor + \tau_p \le N_p\tau_p/k_p + \tau_p$, and $k_p \ge b_p$ by Lemma 17, so the cohort bound applies verbatim. (For the GSA ladder $\tau_p \le R$ always, which is where Lemma 17 is invoked.) □

Step 3: the flow decomposition. Job i completes by the end of phase q_i, so by Lemma 18 and (4),

$$\sum_i C_i \;\le\; \sum_i \sum_{p\le q_i} D_p \;=\; \sum_p D_p\, N_p \;\le\; \frac{2}{R}\sum_p \tau_p^2 N_p^2 \;+\; \sum_p \tau_p N_p. \tag{5}$$

The second sum is a geometric tail: by (2), $\tau_p \le \tau_q\, 2^{p-q}$ for $p \le q$, so $\sum_{p\le q_i} \tau_p \le 2\tau_{q_i}$; and for $q_i \ge 1$, $\tau_{q_i} \le 2\tau_{q_i-1} + 1 \le 2(L_i - 1) + 1 < 2L_i$ (using the *integer* bound $\tau_{q_i-1} \le L_i - 1$ from $\tau_{q_i-1} < L_i$), while $q_i = 0$ gives $\tau_0 = 1 \le 2L_i$. Hence

$$\sum_p \tau_p N_p = \sum_i \sum_{p\le q_i} \tau_p \;\le\; 4\sum_i L_i. \tag{6}$$

Step 4: charging the quadratic term to the volume bound. This is the step that removes the log of Lemma 32: instead of charging each phase to VLB separately, we lower-bound VLB by the *whole* ladder sum at once via an Abel summation over length classes.

Lemma 19 (Ladder-class decomposition of the volume bound)**.** *For every doubling ladder and every instance ($1 \le L_i \le R$), with N_p as in (4),*

$$\mathrm{VLB} \;\ge\; \frac{3}{64R}\sum_{p\ge 0} \tau_p^2 N_p^2.$$

Proof. Class volumes. Let class q collect the jobs with $q_i = q$, i.e. $\tau_{q-1} < L_i \le \tau_q$ (class 0: $L_i \le \tau_0 = 1$, so $L_i = 1$). For $q \ge 1$, $L_i \ge \tau_{q-1} + 1 \ge (\tau_q + 1)/2$ by (2), so $\binom{L_i+1}{2} \ge \frac{1}{2}\cdot\frac{\tau_q+1}{2}\cdot\frac{\tau_q+3}{2} \ge \tau_q^2/8$; for $q = 0$, $\binom{2}{2} = 1 \ge \tau_0^2/8$.

Rank sums. Sort lengths ascending. Classes are length-ordered (all of class q is $\le \tau_q < \tau_q + 1 \le$ all of class $q{+}1$), so the class-q jobs occupy exactly the positions with reverse ranks $N_{q+1}+1, \dots, N_q$ in the sorted sum (8), whence

$$\mathrm{VLB} \;\ge\; \frac{1}{R}\sum_{q\ge 0} \frac{\tau_q^2}{8} \sum_{r=N_{q+1}+1}^{N_q} r \;\ge\; \frac{1}{16R}\sum_{q\ge 0} \tau_q^2\big(N_q^2 - N_{q+1}^2\big).$$

Abel summation. With $N_q = 0$ beyond the last nonempty class and $\tau_q^2 \ge 4\tau_{q-1}^2$ from (2),

$$\sum_{q\ge 0} \tau_q^2\big(N_q^2 - N_{q+1}^2\big) = \tau_0^2 N_0^2 + \sum_{q\ge 1}\big(\tau_q^2 - \tau_{q-1}^2\big)N_q^2 \;\ge\; \frac{3}{4}\sum_{q\ge 0}\tau_q^2 N_q^2. \qquad \square$$

Step 5: the theorem.

Theorem 20 (Self-contained constant at $s = 0$)**.** *In the prompt-free residual model (1) with $1 \le L_i \le R$, kill-and-restart, and unknown lengths, the schedule that attempts every unfinished job in each phase in the cohort realization on any doubling ladder, or in the staggered realization when every executed slice is at most R —in particular* Coverage-GSA with $\alpha_g = 2$ *(GSA ladder, staggered or cohort realization) and* residual-slot doubling*—is non-clairvoyant and satisfies*

$$\sum_i C_i \;\le\; 4\sum_i L_i \;+\; \frac{128}{3}\,\mathrm{VLB} \;\le\; \frac{140}{3}\cdot\mathrm{OPT} \;<\; 46.7\cdot\mathrm{OPT}.$$

Via Theorem 6, the same bound holds for Coverage-GSA against coverage OPT on every star with residual $R = M - W \ge L_{\max}$, tight M included.

Proof. Feasibility is Lemmas 15–17; combining (5), (6), and Lemma 19,

$$\sum_i C_i \;\le\; 4\sum_i L_i + \frac{2}{R}\cdot\frac{64R}{3}\,\mathrm{VLB} \;=\; 4\sum_i L_i + \frac{128}{3}\,\mathrm{VLB},$$

and $\mathrm{OPT} \ge \max\{\sum_i L_i, \mathrm{VLB}\}$ (Lemma 29) gives $4 + \frac{128}{3} = \frac{140}{3}$. $\square$

Lemma 21 (Width on GSA's residual-dependent ladder)**.** *Let $R \ge 1$ be an integer, $\ell = \lfloor \log_2 R \rfloor$, and $\tau_p = \lfloor R/2^{\ell-p} \rfloor$ for $0 \le p \le \ell$. The maximal feasible staggered width k_p satisfies $k_p\tau_p \ge R$.*

Proof. Write $K = k_p$ and $\tau = \tau_p$. Maximality and integer-valued Peak imply

$$R \le \mathrm{Peak}(K+1, \tau) - 1 = \frac{(K+2)\tau + K - 1 - \gcd(K+1, \tau)}{2}.$$

If $K \ge 2$, the right side is at most $K\tau$, since $(K-2)\tau \ge K - 2 \ge K - 1 - \gcd(K+1, \tau)$. Every nonfinal slice has $2\tau_p \le R$, so $\mathrm{Peak}(2, \tau_p) \le 2\tau_p \le R$ and $k_p \ge 2$. The final slice equals R and has feasible width one, which also gives $k_p\tau_p \ge R$. $\square$

The ladder condition matters: $R = 4$, $\tau = 3$ has width one and $k\tau < R$. The stronger width bound is not asserted for arbitrary slices or for cohort execution.

Theorem 22 (Sharper constant for staggered GSA)**.** *Under the model of Theorem 20, staggered GSA on the residual-dependent ladder at $\alpha_g = 2$ satisfies*

$$\sum_i C_i \le 4 \sum_i L_i + \frac{64}{3}\,\mathrm{VLB} \le \frac{76}{3}\,\mathrm{OPT}.$$

The same ratio holds on resident stars under the residual reduction of Theorem 6.

Proof. Lemma 21 sharpens the staggered phase duration to $D_p \le N_p\tau_p/k_p + \tau_p \le N_p\tau_p^2/R + \tau_p$. Substitute this bound into the flow decomposition (5), retaining coefficient $1/R$ in place of $2/R$. Equation (6) and Lemma 19 then give $\sum_i C_i \le 4\sum_i L_i + (64/3)\mathrm{VLB}$. Both $\sum_i L_i$ and VLB are lower bounds on OPT, so the ratio is $4 + 64/3 = 76/3$. Feasibility and non-clairvoyance are unchanged. For noninteger residual capacity, use $\lfloor R \rfloor$: all private ages, lengths and per-tick memory sums are integers, so this leaves the feasible schedules and OPT unchanged. □

Corollary 23 (Corollary 7, self-contained restatement)**.** *Run GSA with $\alpha_g = 2$ at prompt length 0 on the residual budget $R = M - W$. For heterogeneous unknown lengths with $L_{\max} \le R$, staggered Coverage-GSA is non-clairvoyantly $\le 76/3 < 25.34$-competitive versus coverage OPT on stars, at every residual budget—with a proof contained entirely in this paper. The GSA theorem of Feng et al. (2026b) also gives the tuned constant ($\alpha_g = (7+\sqrt{13})/6$, ≤ 61.92 for general α_g-optimized slicing) and the large-width refinement ($\to 32$ at $\alpha_g = 2$ as $k_{\min} \to \infty$), while the bound above follows from the proof in this appendix.*

Remark 2 (Conditions at zero prompt length)**.** *The proof never divides by s or uses induction over prompt tokens. Three conditions remain to check: (i)* ladder definition*: $\ell = \lfloor \log_2 R \rfloor$ needs $R \ge 1$, which is $L_{\max} \ge 1$ plus per-job feasibility $L_{\max} \le R$ (the analogue of condition H6 above); (ii)* per-job volumes*: the class bound $\binom{L_i+1}{2} \ge \tau_q^2/8$ uses only $L_i \ge 1$—at $s = 0$ the completing volume degrades to $L(L+1)/2$, which stays positive, so the volume bound never degenerates; (iii)* widths*: the staggered width $k_p \ge 1$ is defined when $\tau_p \le R$. Cohorts use $b_p \ge 1$, with a singleton fallback when $\tau_p > R$ that is feasible by $L_i \le R$. Unlike the import, the proof never uses that widths grow as $s \downarrow 0$; the constant is uniform in $k_{\min}$, so the tight-M regime ($k_{\min} = 1$) is covered with no boundary case.*

Remark 3 (Advancing when a phase finishes early)**.** *The proof reserves the full duration of each phase or cohort. The implemented variant (and `run_gsa`) starts the next cohort/phase as soon as the current one is drained. Which jobs finish in phase p depends only on $\{L_i \le \tau_p\}$, not on timing, so early advancement shifts every subsequent completion earlier and the bound carries over.*

Remark 4 (Growth factors, private prompts, and measured variants)**.** *(a) The theorem is specific to ladders with growth exactly 2 (where $\lfloor \beta 2^p \rfloor$ satisfies the clean sandwich (2)). At GSA's tuned $\alpha_g = (7+\sqrt{13})/6 \approx 1.77$ the floor-quantized slices can repeat ($\lfloor \beta\alpha^p \rfloor = \lfloor \beta\alpha^{p+1} \rfloor$), the growth ratio degrades, and the constants become α-dependent and worse; we therefore state the self-contained result at $\alpha_g = 2$ and keep 61.92 as the imported alternative. (b) Theorem 9(a,b) uses imported guarantees: the equal-prompt GSA theorem at $s = q > 0$ for part (a), and the arbitrary-prompt router of Feng et al. (2026a) for part (b); this appendix does not replace those results (the same class/Abel charge appears to extend to $s > 0$ against the area volume $so + \binom{o+1}{2}$, which we leave as an explicit open sharpening). (c) Constant comparison: the $76/3 \approx 25.33$ bound for staggered GSA is uniform in $k_{\min}$, including width one. It sharpens the imported bounds 61.92 at tuned growth and 32 in the large-width limit at growth two, in the prompt-free setting considered here. The $140/3$ bound remains valid for the broader cohort/doubling family. These are bounds on the same algorithms' theoretical completion times, not new engine-speed measurements. (d) The live `gsa_add`/`gsa_cov` arms and the computational checks run GSA's tuned ladder ($\alpha_g \approx 1.77$, recorded in the artifacts); their guarantee is the imported one, while the makespan comparison itself is ladder-identical across arms and so independent of α_g.*

Numerical verification. The program `polyserve/theory/s0_selfcontained_check.py` checks the ingredients of the $140/3$ bound. It exhaustively tests the floor-sum identity (3) for $k, \tau \le 60$ and compares the peak bound with tick-level simulation. The peak reaches equality once $n \ge k$. It tests the doubling inequalities for all $R \le 600$ and Lemma 19 on 4,000 random instances plus boundary-length cases. The latter reach 0.98 of the bound. Inside `run_gsa`, phase durations reach at most 0.995 of Lemma 18's bound, with no violation. Across 3,000 random instances, 1,650 staggered runs and structured adversarial families, the largest $\mathrm{ALG}/\max\{\mathrm{VLB}, \sum L\}$ is 12.8. On 491 small instances with exact OPT, the largest ratios are 3.83 for cohorts and 3.10 for staggered execution. For the $76/3$ bound, exact arithmetic checks all GSA phases with $R \le 10{,}000$ and 15,029 small executions. The check retains the arbitrary-slice counterexample above. These finite tests check the formulas; Theorem 22 supplies the proof for all instances.

B.9 Comparison with a progress-retaining optimum

The $140/3$ guarantee also holds against a relaxation that retains the progress of inactive jobs without charging their storage. Define three transition rules for an unselected unfinished job: *kill-and-restart* resets its age to 0; *retain-progress* keeps its age; and *retain-or-kill* allows either choice. In this subsection all three use the active-set constraint $\sum_{i\in S_t}(a_i+1)\le R$. Write $\mathrm{OPT}^{\mathrm{kr}}$, $\mathrm{OPT}^{\mathrm{pr}}$, and $\mathrm{OPT}^{\mathrm{flex}}$ for their offline optimal values. Then $\mathrm{OPT}^{\mathrm{flex}}\le\min\{\mathrm{OPT}^{\mathrm{pr}},\mathrm{OPT}^{\mathrm{kr}}\}$. These are abstract progress-retention benchmarks: inactive storage is free. For an engine retaining private KV in the same pool, the constraint is instead

$$\sum_{i\in S_t}(a_i+1)+\sum_{i\text{ paused with retained KV}}a_i\le R.$$

For example, at $R=2$ a paused job of age 1 leaves room for only one new token of an age-0 job; advancing that second job again costs $1+2>R$, although its active-set cost alone is 2.

Lemma 24 (The volume bound under progress retention)**.** *Every schedule feasible under (1) in any of the three transition rules satisfies $\sum_i C_i\ge\mathrm{VLB}$ and $\sum_i C_i\ge\sum_i L_i$. In particular, $\mathrm{OPT}^{\mathrm{flex}}\ge\max\{\mathrm{VLB},\sum_i L_i\}$.*

Proof. A job starts at age 0, rises by one on a served tick, never rises on an unserved tick, and completes on first reaching age L_i. Thus each level crossing $a\to a+1$, for $0\le a<L_i$, occurs at least once and consumes active-set weight $a+1$. Contiguity is unnecessary: job i contributes at least $\binom{L_i+1}{2}$ served weight before completion. The prefix-of-completions argument of Lemma 29 therefore gives VLB unchanged, and $C_i\ge L_i$ gives the linear bound. □

Corollary 25 (Kill-and-restart GSA against a retaining benchmark)**.** *With unknown lengths $1\le L_i\le R$, the kill-and-restart schedule of Theorem 20 satisfies*

$$\sum_i C_i\le 4\sum_i L_i+\frac{128}{3}\mathrm{VLB}\le\frac{140}{3}\mathrm{OPT}^{\mathrm{flex}}\le\frac{140}{3}\mathrm{OPT}^{\mathrm{pr}}.$$

The algorithm still discards unfinished progress at slice boundaries; the stronger benchmark does not change its transition rule.

Proof. Theorem 20 bounds the algorithm directly by $4\sum_i L_i+(128/3)\mathrm{VLB}$. Apply Lemma 24 and $4+128/3=140/3$. □

This comparison does not supply a scheduler that preserves every unfinished job's KV. In particular, simply retaining ages while replaying the same selected sets can make a later tick infeasible. The pinned-storage model above and the active-set relaxation must be distinguished when comparing implementations.

B.10 Proof of Theorem 14(a)

Here *probe doubling* has the following explicit meaning for identical unknown lengths. In phase p run one cohort of width $\min\{k,\max(1,\lfloor R/2^p\rfloor)\}$ for at most 2^p ticks. If no request finishes, reset that cohort and increment p. At the first completion, the identical-length promise reveals L; finish the remaining jobs in cohorts of the same width, releasing each cohort as soon as it completes. A singleton phase with $2^p>R$ is feasible because its request completes at $L\le R$. This is the single-probe policy of Theorem 14(a), distinct from GSA's policy of attempting every unfinished request in each phase.

Let $b(\ell)=\max\{1,\lfloor(M-W)/\ell\rfloor\}$, $b^\star=b(L)$, and $F^\star=L\cdot\Phi(k,b^\star)$. Final phase index $p^\star$ satisfies $2^{p^\star-1}<L\le 2^{p^\star}$, so $\widehat{L}=2^{p^\star}\in(L,2L]$.

Width. The map $\ell\mapsto b(\ell)$ is nonincreasing, so $b_f:=b(\widehat{L})\ge b(2L)=\max\{1,\lfloor(M-W)/(2L)\rfloor\}$. Since $\lfloor x/2\rfloor\ge\lfloor\lfloor x\rfloor/2\rfloor$ for $x=(M-W)/L$, $\lfloor(M-W)/(2L)\rfloor\ge\lfloor b^\star/2\rfloor$. Thus $b_f\ge\lfloor b^\star/2\rfloor$ when $\lfloor b^\star/2\rfloor\ge1$; if $b^\star=1$ then $(M-W)/(2L)<1$ and $b_f=1$.

Prior waste. Each unsuccessful phase runs exactly one probe cohort of duration 2^p, so the wall clock consumed before the final phase is $\sum_{p<p^\star}2^p=\widehat{L}-1<2L$, and the prior delay summed over jobs is $<2kL\le 2F^\star$ (using $F^\star\ge kL$). The single probe per phase is essential: attempting *every* unfinished job in every phase multiplies each phase length by $\lceil k/b_p\rceil$, and that variant is not ≤ 4-competitive (k=7, L=9, R=27: flow 497 against $F^\star$=108, ratio 4.60).

Final phase, $b^\star=1$. Serial cohorts of duration $\le\widehat{L}\le 2L$ give final flow $\le 2F^\star$; with prior waste, ALG $<4F^\star$.

Final phase, $b^\star \geq 2$**.** Let $m = \lfloor b^\star/2 \rfloor \geq 1$, so $b_f \geq m$. Final-phase flow $\leq L \cdot \Phi(k, m)$ (cohort count $\lceil k/b_f \rceil \leq \lceil k/m \rceil$; when $b^\star = 2m$ even, $\lceil k/m \rceil \leq 2\lceil k/b^\star \rceil$, but the odd case can violate the ceiling comparison, so we charge through Φ).

Lemma 26 (Half-width charging)**.** *For $s \geq 2$, $m = \lfloor s/2 \rfloor$, $k \geq 1$: $2k + \Phi(k, m) \leq 4\Phi(k, s)$.*

Proof. Enough: $\Phi(k, m) + k \leq 3\Phi(k, s)$ (then add $k \leq \Phi(k, s)$). Even $s = 2m$: $\lceil i/m \rceil \leq 2\lceil i/s \rceil \Rightarrow \Phi(k, m) \leq 2\Phi(k, s)$. For odd $s = 2m + 1$, write an index in the jth width-s block as $i = (j-1)s + u$, $1 \leq u \leq r \leq s$. Then $\lceil i/m \rceil = 2(j-1) + \lceil (j-1+u)/m \rceil$. Since $\lceil (j-1+u)/m \rceil \leq j - 1 + \lceil u/m \rceil$ and $\sum_{u=1}^{r} \lceil u/m \rceil \leq 2r$ for $r \leq 2m + 1$ (at the last index the sum is $3m + 3 \leq 4m + 2$), this block contributes at most $2r(j-1) + r + r(j-1) + 2r = 3rj$ to $\Phi(k, m) + k$. Summing full blocks and the last partial block gives $\Phi(k, m) + k \leq 3\Phi(k, s)$. □

Hence ALG $< 2kL + L \cdot \Phi(k, m) \leq 4F^\star$ (observed ratios ≤ 3.13 on the tested grid).

Benchmark vs OPT. The bound above is stated against the lockstep benchmark $F^\star$. In the two exact regimes of Theorem 11 ($b^\star \geq k$ or $R = L$) $F^\star = \mathrm{OPT}$, so doubling is ≤ 4-competitive vs OPT there. At intermediate budgets $F^\star$ can strictly exceed OPT (Theorem 11(c)); chaining with Lemma 13 gives ALG $\leq 4F^\star \leq 20 \cdot \mathrm{OPT}$, and sharpening this constant is open.

B.11 Proof of Theorem 14(b)

Under $M \geq W + \alpha k L_{\max}$, every unfinished set is look-ahead feasible through completion. This full-parallel policy starts every unfinished request at age 0 in phase p, serves the selected set together for at most 2^p ticks, and resets survivors before phase $p + 1$. Requests leave on completion. No peak reservation of size $k2^p$ is needed: the full-parallel instance promise bounds every actually executed private age by $L_i \leq L_{\max}$. This definition is separate from the known-cap admission baseline. Job i finishes in phase $p_i = \lceil \log_2 L_i \rceil$, so $C_i \leq \sum_{p \leq p_i} 2^p < 2 \cdot 2^{p_i} \leq 4L_i$ and $\sum C_i \leq 4 \sum L_i \leq 4 \cdot \mathrm{OPT}$ (measured ratios: mean 2.14, max 2.43; Table A2).

Remark 5 (A geometric policy under a tight budget)**.** *A geometric coverage policy (growth α_g) that retains the current cohort gives mean ratio ≈ 1.68 and maximum ≈ 2.49 over 40 trials. The comparator knows lengths and uses shortest remaining processing time (SRPT), with shared-memory accounting under $M = W + 2L_{\max}$. These are empirical comparisons without a competitive bound.*

Theorem 27 (Non-clairvoyance on tight-M stars)**.** *Under $M = W + L_{\max}$ (c=1 residual) on* star / shared-prompt *instances: (i) Greedy progress packing (RPP; Algorithm 1) is* exactly optimal *on identical-length stars (serial lockstep; Lemma 30), and from any state its progress anchor completes within $L_{\max}$ ticks. (ii) On heterogeneous lengths RPP is not ≤ 4-competitive: an explicit instance family has ratio ≥ 5.5 at k=16 and ≥ 7.4 at k=32 with $L_{\max}$ fixed (Proposition 31); whether the ratio is $\omega(1)$ remains open. (iii) Residual-slot doubling is $O(\log L_{\max})$-competitive (Lemma 32). Static peak reservation using a known cap U can have an unbounded ratio even with immediate release on completion: on $k = U$ unit-length jobs with $R = U$, its width is one, whereas coverage OPT runs all jobs together, giving ratio $(U + 1)/2 = \Theta(U)$. For heterogeneous lengths, Corollary 7 gives the $76/3$ bound for staggered Coverage-GSA at growth two. The broader cohort family has bound $140/3$, and the prior tuned GSA bound is 61.92. Under online arrivals, Theorem 38 gives Epoch-GSA a constant < 140 for total completion time. More general partial sharing requires separate analysis.*

B.12 Proof of Theorem 27 (RPP)

Regime: $M = W + L_{\max}$ (c=1), $\alpha = 1$, kill-and-restart on stars; write $R = L_{\max}$ for the residual private capacity.

Both RPP and the coverage optimum obey the same one-tick *look-ahead* budget: a set S may run at a tick only if, after the tick's $+1$ age growth, $\mathrm{Mem}_{\mathrm{cov}} \leq M$, i.e.

$$\sum_{i \in S} (a_i + 1) \leq R, \qquad R := M - W = L_{\max}. \tag{7}$$

Unselected unfinished jobs are reset to age 0 (kill-and-restart). The exhaustive OPT solver uses the same feasibility, so OPT below is the offline coverage optimum.

Lemma 28 (Greedy maximality; anchor completion)**.** *Write $\sigma := \sum_{i \in S} (a_i + 1)$ and $\rho := R - \sigma \geq 0$ for the leftover budget. (i) Every excluded unfinished job j has $a_j + 1 > \rho$; in particular, if an age-0 job is excluded then $\rho = 0$. (ii) The maximum-progress unfinished job (the* anchor*) is served every tick until it completes; hence from any state RPP completes a job within $L_{\max}$ ticks and finishes all k jobs.*

Algorithm 1 Greedy packing by current progress (RPP)

Require: unfinished $\mathcal{U}$ with private ages $(a_i)_{i\in\mathcal{U}}$, residual $R = M - W$
sort $\mathcal{U}$ by key $(-a_i,\, i)$ ▷ desc. progress; tie-break smaller job id
$S \leftarrow \emptyset$
for $i \in \mathcal{U}$ in sorted order **do**
  if $\sum_{j\in S\cup\{i\}}(a_j+1) \le R$ **then**
    $S \leftarrow S \cup \{i\}$
  end if
end for
kill-restart: $a_i \leftarrow 0$ for all $i \in \mathcal{U} \setminus S$
advance $a_i \leftarrow a_i + 1$ for $i \in S$; remove i with $a_i \ge L_i$
return S

Proof. (i) When the scan reaches j the partial sum is some $\sigma_j \le \sigma$ and j is skipped iff $\sigma_j + (a_j+1) > R$, so $a_j + 1 > R - \sigma = \rho$; an excluded age-0 job has weight 1, forcing $\rho = 0$. (ii) The anchor ℓ is first in the scan order $(-a_i, i)$ and feasible alone ($a_\ell + 1 \le L_\ell \le R$), so it is selected and advances; being served keeps its progress maximal, so it stays first until it reaches age L_ℓ. □

Lemma 29 (Volume lower bound)**.** *Sort private lengths ascending, $L_{(1)} \le \cdots \le L_{(k)}$. Every schedule feasible under (7) (in particular the kill-restart coverage optimum) has flow time at least*

$$\mathrm{VLB} := \frac{1}{R}\sum_{r=1}^{k}\sum_{i=1}^{r}\binom{L_{(i)}+1}{2} = \frac{1}{R}\sum_{i=1}^{k}(k-i+1)\binom{L_{(i)}+1}{2}. \tag{8}$$

Hence $\mathrm{OPT} \ge \mathrm{VLB}$. *Separately, every request needs L_i service ticks, so $C_i \ge L_i$ and* $\mathrm{OPT} \ge \sum_i L_i$.

Proof. Fix a feasible schedule with completion order $\pi_1, \ldots, \pi_k$ ($C_{\pi_1} \le \cdots \le C_{\pi_k}$) and let $\sigma(t) = \sum_{i\in S(t)}(a_i+1) \le R$ be the served weight at tick t. Each completed job j finishes via one uninterrupted run through ages $0, 1, \ldots, L_j - 1$ ending at C_j, contributing weight $1 + 2 + \cdots + L_j = \binom{L_j+1}{2}$ to $\sum_t \sigma(t)$ within $[0, C_j]$. These runs belong to distinct jobs, so

$$R \cdot C_{\pi_r} \;\ge\; \sum_{t\le C_{\pi_r}} \sigma(t) \;\ge\; \sum_{s=1}^{r}\binom{L_{\pi_s}+1}{2} \;\ge\; \sum_{i=1}^{r}\binom{L_{(i)}+1}{2},$$

the last step because any r lengths have weight-sum at least that of the r smallest. Summing over $r = 1, \ldots, k$ gives $\sum_r C_{\pi_r} \ge \mathrm{VLB}$. □

Lemma 30 (Identical-star exactness under tight residual)**.** *On identical-length stars with $R = L_{\max} = L$, under kill-and-restart, RPP is exactly optimal: $\sum_i C_i^{\mathrm{RPP}} = L \cdot k(k+1)/2 = \mathrm{OPT}$.*

Proof. The anchor is served every tick until it completes at age L (Lemma 28(ii)). While the anchor is at age j (weight $j{+}1$), any filler admitted by Lemma 28(i) has weight $\le R - (j{+}1) = L - j - 1$, i.e. age $\le L - j - 2 < L - 1$: a filler never reaches its completing age while the anchor is alive, and no two jobs complete at the same tick ($2L > R$). At the anchor's completion tick its weight is $L = R$, so every other job is excluded and reset to age 0, leaving one fewer job with all ages 0. By induction, completions occur exactly at $L, 2L, \ldots, kL$, so $\sum_i C_i^{\mathrm{RPP}} = L \cdot k(k+1)/2 = \mathrm{OPT}$ by Theorem 11(b). For $R > L$ the exactness fails in both directions: on the k=5, L=3, R=7 instance of Theorem 11(c), RPP's age-0 fillers give flow $23 < 27 = F^\star$, and in general neither RPP nor lockstep need equal OPT. □

Proposition 31 (Heterogeneous RPP is not $\le$ 4-competitive)**.** *There are heterogeneous star instances under $M = W + L_{\max}$ on which RPP's flow exceeds $5.5 \cdot \mathrm{OPT}$, and the computed ratio lower bound grows with k at fixed $L_{\max}$ (reaching 7.4 at k=32); we conjecture the growth is unbounded. Concretely, for $L = (1, 3, 3, 3, 4, 4, 5, 6, 7, 7, 8, 9, 11, 18, 30, 32)$ with ids assigned in* descending *length order (k=16, R=$L_{\max}$=32, W=R, M=2R), simulated RPP flow is* 1123*, while an explicitly verified feasible atomic schedule (each job runs once, uninterrupted; (7) re-checked tick by tick) completes with flow* 203*, so*

$$\frac{\sum_i C_i^{\mathrm{RPP}}}{\mathrm{OPT}} \;\ge\; \frac{1123}{203} \;\approx\; 5.53 \;>\; 4.$$

Analogous computationally checked instances give $\ge$ 6.9 at k=24 and $\ge$ 7.4 at k=32 (same $L_{\max}$=32). Descending-progress priority lets long jobs continue while shorter jobs repeatedly restart near completion. The adversary chooses request IDs to control ties. Discarded progress increases with k in this constructed family.

Lemma 32 (Residual-slot doubling is $O(\log L_{\max})$-competitive)**.** *On heterogeneous stars with $M = W + R$, $R \geq L_{\max}$, kill-and-restart, residual-slot doubling—phases $p = 0, 1, \ldots, P = \lceil \log_2 L_{\max} \rceil$ run in ascending order; phase p attempts every currently-unfinished job once, in cohorts of width $b_p = \max(1, \lfloor R/2^p \rfloor)$ each reserved for 2^p ticks—has*

$$\textstyle\sum_i C_i \;\leq\; \big(32\,(\lceil \log_2 L_{\max} \rceil + 1) + 4\big) \cdot \mathrm{OPT}.$$

Proof. Feasibility. A width-b_p cohort held for 2^p ticks has look-ahead weight $\leq b_p 2^p \leq R$ when $b_p = \lfloor R/2^p \rfloor \geq 1$; when $2^p > R$ the fallback $b_p = 1$ is feasible because a single job's weight never exceeds $L_{\max} \leq R$.

Class structure. Let $C_q = \{i : 2^{q-1} < L_i \leq 2^q\}$ and $N_p = \sum_{q \geq p} |C_q|$ (jobs with $L_i > 2^{p-1}$). A job attempted in phase $p \geq q_i$ completes within its reservation, so at the start of phase p the unfinished set is contained in the N_p jobs of class $\geq p$, and every class-q job completes by the end of phase q.

Phase durations. Phase p lasts $\leq \lceil N_p/b_p \rceil 2^p \leq N_p 2^p/b_p + 2^p$. If $2^p \leq R/2$ then $b_p \geq R/2^{p+1}$ and $N_p 2^p/b_p \leq 2N_p 4^p/R$; if $2^p > R/2$ then $b_p \geq 1$ and $N_p 2^p \leq 2N_p 4^p/R$ as well (since $4^p/R > 2^{p-1} \geq 2^p/2$). Hence phase p lasts $\leq 2N_p 4^p/R + 2^p$ in all cases.

Flow. A class-q job finishes by $\sum_{p \leq q}(2N_p 4^p/R + 2^p) \leq \sum_{p \leq q} 2N_p 4^p/R + 2^{q+1}$, so

$$\textstyle\sum_i C_i \;\leq\; \underbrace{\sum_q |C_q|\, 2^{q+1}}_{\leq\, 4 \sum_i L_i \,\leq\, 4\,\mathrm{OPT}} \;+\; \sum_q |C_q| \sum_{p \leq q} \frac{2N_p 4^p}{R} \;=\; 4\,\mathrm{OPT} + \sum_p \frac{2\,4^p N_p^2}{R},$$

where the swap uses $\sum_{q \geq p} |C_q| = N_p$.

Charging each phase to VLB. The N_p longest jobs all have completing volume $\binom{L+1}{2} \geq 4^p/8$ (as $L > 2^{p-1}$), and in the sorted VLB sum (8) their reverse ranks cover $1, \ldots, N_p$, so for every p

$$\mathrm{VLB} \;\geq\; \frac{1}{R} \cdot \frac{4^p}{8} \cdot \frac{N_p(N_p+1)}{2} \;\geq\; \frac{4^p N_p^2}{16R}, \qquad \text{hence} \qquad \frac{2\,4^p N_p^2}{R} \;\leq\; 32\,\mathrm{VLB}.$$

Summing over the $P + 1$ phases and using $\mathrm{VLB} \leq \mathrm{OPT}$ (Lemma 29) gives the claim. □

Against clairvoyant feasible schedules, the observed ratios stay below $4(\lceil \log_2 L_{\max} \rceil + 1)$. Those comparison schedules upper-bound OPT, so these observed ratios do not upper-bound the true competitive ratio. Lemma 32 is an elementary bound; Appendix B.8 gives the stronger constant guarantee.

Static peak reservation is $\omega(1)$. Take $k = U$ unit-length jobs, shared prefix $W = U$, and $M = W + U$, so $R = U$. The cap U is a valid known upper bound; it need not equal the realized maximum output length. Coverage OPT runs all jobs in one tick and has flow k. The fixed-cap policy reserves U private tokens for every admitted request, so its width is $\lfloor R/U \rfloor = 1$. Even releasing each request immediately upon its observed completion, it completes jobs at $1, 2, \ldots, k$, with flow $k(k+1)/2$ and ratio $(k+1)/2 = (U+1)/2$. The example isolates a conservative per-request cap; it does not require keeping memory reserved after a request has finished. Coverage-GSA provides the constant guarantee for different unknown lengths (Corollary 7). RPP and fixed-cap reservation have the counterexamples above. Remark 5 reports a separate empirical comparison, and Appendix B.13 treats online arrivals.

B.13 Online arrivals: release-time floor and Epoch-GSA

The results so far assume all k jobs are present at $t = 0$. This appendix removes that assumption. Add release times $r_i \geq 0$ to the prompt-free residual model of Appendix B.8: at tick t a scheduler may serve only released, unfinished jobs,

$$S_t \subseteq \{i :\, r_i \leq t,\; a_i < L_i\}, \qquad \sum_{i \in S_t} (a_i + 1) \;\leq\; R, \tag{9}$$

lengths $1 \leq L_i \leq R$ stay unknown until completion, unselected unfinished jobs reset to age 0, and idling is permitted (with arrivals it can be forced). An algorithm is *online* if its tick t decision depends only on R, on $\{i : r_i \leq t\}$, and on observed completions—never on unrevealed lengths or future arrivals. We report total completion time $\mathrm{TC} = \sum_i C_i$ as the primary objective and total flow $\mathrm{TF} = \sum_i (C_i - r_i)$ as secondary; the benchmark is the offline optimum in the same release-time model.

Proposition 33 (Release-time isomorphism)**.** *Fix a star with shared prefix W, budget M, and release times (r_i). A release-respecting schedule is coverage-feasible at every tick iff it is feasible under (9) with $R = M - W$. The map is the identity on schedules and preserves C_i, hence* TC *and* TF.

Proof. Theorem 6 equates the two per-tick predicates $W + \sum_{i \in S_t}(a_i + 1) \le M$ and $\sum_{i \in S_t}(a_i + 1) \le R$ for *every* candidate set S_t, with no reference to how S_t was chosen. Release times restrict which S_t are admissible, identically on both sides, and leave the predicate untouched; the identity map therefore carries release-respecting coverage-feasible schedules to release-respecting residual-feasible ones and back, preserving ages and so completion times. □

Online guarantees in the remaining-memory model therefore apply to shared-prefix scheduling, including at $M = W + L_{\max}$. The volume bound in Lemma 29 remains valid, but its interval $[0, C_i]$ includes time before late jobs arrive. A common release-time cutoff gives a more informative bound.

Lemma 34 (Release-time volume lower bound)**.** *Write $T(L) = \binom{L+1}{2} = L(L+1)/2$ and, for a job set A with lengths sorted ascending $L^A_{(1)} \le \cdots \le L^A_{(m)}$,*

$$\mathrm{VLB}_R(A) \;:=\; \frac{1}{R}\sum_{r=1}^{m}\sum_{j=1}^{r} T\big(L^A_{(j)}\big).$$

Then for every schedule feasible under (9), every cutoff $\theta \ge 0$, and every $A \subseteq \{i : r_i \ge \theta\}$,

$$\sum_{i \in A}\big(C_i - \theta\big) \;\ge\; \mathrm{VLB}_R(A). \tag{10}$$

This holds under kill-and-restart, under pause-retain, and under the flexible semantics that chooses per job and per tick.

Proof. Job i is served at least once at each age $0, 1, \ldots, L_i - 1$: its completing pass runs through those ages consecutively under kill-and-restart, while under pause-retain the age never resets, so each level is visited exactly once; mixtures only repeat levels. A serve at age a carries weight $a + 1$, so the total weight $\sigma(t) = \sum_{i \in S_t}(a_i + 1)$ accumulated by job i over its own serves is at least $1 + 2 + \cdots + L_i = T(L_i)$, and every one of those ticks lies in $[r_i, C_i)$.

Fix θ and A, and order A by completion, $C_{\pi_1} \le \cdots \le C_{\pi_m}$. For each r, the jobs $\pi_1, \ldots, \pi_r$ are distinct and all of their serves happen at ticks t with $\theta \le r_{\pi_s} \le t < C_{\pi_s} \le C_{\pi_r}$, so

$$\sum_{\theta \le t < C_{\pi_r}} \sigma(t) \;\ge\; \sum_{s=1}^{r} T(L_{\pi_s}) \;\ge\; \sum_{j=1}^{r} T\big(L^A_{(j)}\big),$$

the last step because any r of the lengths have T-sum at least that of the r smallest. Feasibility gives $\sigma(t) \le R$ at every tick, so the left side is at most $R\,(C_{\pi_r} - \theta)$, whence $C_{\pi_r} - \theta \ge \frac{1}{R}\sum_{j \le r} T(L^A_{(j)})$. Summing over $r = 1, \ldots, m$ yields (10). Nothing in the argument used which jobs were reset, so all three semantics are covered. □

Two specializations are what the implementation evaluates. Taking $A = \{i : r_i \ge \theta\}$ and bounding the remaining jobs by the trivial $C_i \ge r_i + L_i$,

$$\mathrm{OPT}_{\mathrm{TC}} \;\ge\; \max_{\theta}\Big[\,\theta\,|A_\theta| + \mathrm{VLB}_R(A_\theta) + \sum_{i:\, r_i < \theta}(r_i + L_i)\Big], \qquad A_\theta = \{i : r_i \ge \theta\}, \tag{11}$$

and it suffices to range θ over the distinct release times. For flow, $C_i - r_i = (C_i - \theta) - (r_i - \theta)$ gives

$$\mathrm{OPT}_{\mathrm{TF}} \;\ge\; \max_{\theta}\Big[\,\mathrm{VLB}_R(A_\theta) - \sum_{i \in A_\theta}(r_i - \theta) + \sum_{i \notin A_\theta} L_i\Big]. \tag{12}$$

At $\theta = 0$ both reduce to the t=0 statements of Lemma 29, so (10) is a strict generalization. Since adding release constraints only shrinks the feasible set, the t=0 floor VLB of (8) also remains valid online—a fact the competitive proof below uses.

Epoch-GSA. The t=0 ladder cannot be run as-is: it attempts *every* unfinished job once per phase, and a job arriving mid-phase has no phase to join. Restarting the ladder on each arrival is equally bad (an adversary releasing one job per tick never lets a phase finish). Epoch-GSA resolves this by batching arrivals on a geometric grid.

Definition 3 (Epoch-GSA)**.** *Partition time into epochs $E_e = [2^e - 1,\ 2^{e+1} - 1)$, $e = 0, 1, 2, \ldots$. At the start of an epoch, form B_e from all released, unfinished jobs and start the schedule that attempts every unfinished job in each phase of Appendix B.8 at phase 0: cohorts may use any doubling ladder, while staggered execution requires every slice to be at most R. Coverage-GSA uses the GSA ladder at $\alpha_g = 2$. Arrivals wait while*

this batch remains unfinished. Whenever a batch drains, immediately form a new batch from all released, unfinished jobs and restart at phase 0*; if none exist, wait for the next arrival. All batches remain subject to the same absolute epoch boundary: there, kill every in-flight job (age* $\leftarrow 0$*), discard the current batch's phase state, and form the next epoch's batch. Within a batch, a drained cohort or phase advances immediately. Thus a new arrival waits until the current batch drains or the next epoch starts, whichever occurs first.*

Epoch-GSA is online: batch membership depends only on release times already observed, and the ladder inside an epoch is the non-clairvoyant schedule of Theorem 20. Feasibility is inherited tick by tick from Lemmas 15 and 16, since batches run one at a time and each starts with zero ages. This definition includes the early re-batching used by `simulate_online_ladder`; a unit job arriving to an idle system starts immediately rather than waiting for a boundary.

Two guarantees follow. The first is per epoch and concerns *makespan*: each batch clears in time proportional to the best possible clearing time for that batch (Lemma 35, Theorem 36). The second is the online main result: *total completion time* is within an explicit constant of the release-time optimum (Theorem 38), by a charge that is per job rather than per epoch.

Lemma 35 (The ladder is $O(1)$-competitive for makespan)**.** *In the prompt-free residual model* (1) *on a job set* B *with* $1 \le L_i \le R$*, the schedule that attempts every unfinished job in each phase under the ladder and realization conditions of Theorem 20 clears* B *within* $\frac{76}{3} < 26$ *times the minimum possible makespan.*

Proof. By Lemma 18 the total duration is $\sum_p D_p \le \frac{2}{R}\sum_p \tau_p^2 N_p + \sum_p \tau_p$. For the second sum, (2) gives $\sum_{p\le P}\tau_p \le 2\tau_P$ and $\tau_P < 2L_{\max}$ (the integer bound of (6)), so $\sum_p \tau_p \le 4L_{\max}$. For the first, $\sum_p \tau_p^2 N_p = \sum_i \sum_{p\le q_i} \tau_p^2$ and $\tau_p^2 \le 4^{p-q_i}\tau_{q_i}^2$, so $\sum_{p\le q_i}\tau_p^2 \le \frac{4}{3}\tau_{q_i}^2 < \frac{16}{3}L_i^2$, giving $\frac{2}{R}\sum_p \tau_p^2 N_p < \frac{32}{3R}\sum_i L_i^2$. Any feasible schedule needs total served weight at least $\sum_i T(L_i) \ge \frac{1}{2}\sum_i L_i^2$ (Lemma 34's level count) while spending at most R per tick, so its makespan is at least $\frac{1}{2R}\sum_i L_i^2$; it is also at least $L_{\max}$. Hence $\sum_p D_p < \frac{32}{3}\cdot 2\,\mathrm{MS}^\star + 4\,\mathrm{MS}^\star = \frac{76}{3}\mathrm{MS}^\star$. □

Theorem 36 (Epoch-GSA: per-epoch clearing and transfer)**.** *In the release-time model* (9) *with unknown lengths and kill-and-restart, Epoch-GSA is online and always feasible. An epoch-start batch* B_e *that is not interrupted by the next epoch boundary clears within* $26\,\mathrm{MS}^\star(B_e)$ *ticks of its start. Every job completes within* $O(1)\cdot\max\{r_i,\ \mathrm{MS}^\star(B_{e_i})\}$ *of its release whenever its batch is not preempted by a later epoch boundary. Via Proposition 33 every such bound holds verbatim for Coverage-GSA against the coverage optimum on stars with residual* $R = M - W \ge L_{\max}$*, tight* M *included.*

Proof. Onlineness and feasibility are immediate from Definition 3: batch membership uses only observed release times, and within an epoch a single ladder runs, so Lemmas 15–16 apply tick by tick. All of B_e is present at the epoch start, so Lemma 35 bounds the clearing time by $26\,\mathrm{MS}^\star(B_e)$. The epoch containing r_i ends by $2\,(r_i+1)$, which gives the stated per-job form. □

From per-epoch makespan to total completion time. Charging total completion time *per epoch* does not work: under the re-batching discipline of Definition 3 a job killed at a boundary reappears in the next batch, so the natural charge "epoch E_e is short unless OPT is large on B_e" is multiplied by $|B_e|$ when summed. (Running batches strictly sequentially instead—each job in exactly one batch—is not merely hard to analyze but bad: Proposition 39.) The charge that works is *per job*, against the arrival-free run of the same ladder on the whole instance, and it rests on a single monotonicity property of attempt-all ladders.

Lemma 37 (Reference phase ends)**.** *Fix a doubling ladder and a version that reserves each full phase duration satisfying the conditions of Theorem 20. For a job set* B *write* $N_p(B) = \#\{j \in B :\ L_j > \tau_{p-1}\}$ *(with* $\tau_{-1} = 0$*), and let* $D_p(N)$ *be the duration of a phase that attempts* N *jobs with slice* τ_p*:* $\lceil N/b_p\rceil\,\tau_p$ *for cohorts,* $\lfloor (N-1)\tau_p/k_p\rfloor + \tau_p$ *for the staggered realization—in both cases nondecreasing in* N*. Let* J *be the whole instance and, for* $i \in J$ *with class* $q_i = \min\{p : \tau_p \ge L_i\}$*,*

$$\widehat{C}_i \ :=\ \sum_{p\le q_i} D_p\big(N_p(J)\big),$$

the end of job i*'s completing phase when the ladder runs on all of* J *from* $t = 0$*. Then (a) for every* $B \subseteq J$ *with* $i \in B$*, the ladder started fresh on* B *(all members at age* 0*) completes* i *within* $\widehat{C}_i$ *ticks of its start, in either realization and also under early-advancing timing (Remark 3); and (b)* $\sum_{i\in J}\widehat{C}_i \le 4\sum_i L_i + \frac{128}{3}\mathrm{VLB}(J) \le \frac{140}{3}\,\mathrm{OPT}(J)$*, where* $\mathrm{OPT}(J)$ *is the arrival-free optimum and* $\mathrm{VLB}(J)$ *the* t=0 *volume bound* (8)*.*

Proof. (a) Which jobs finish in phase p depends only on $\{L_j \le \tau_p\}$, so on B the count entering phase p is $N_p(B) \le N_p(J)$ and job i still completes during phase q_i. Phase p on B lasts $D_p(N_p(B)) \le D_p(N_p(J))$ by monotonicity, and early advancement only ends phases sooner; summing over $p \le q_i$ gives the claim.

(b) $\sum_i \widehat{C}_i = \sum_p D_p(N_p(J))\, N_p(J)$ is the middle term of (5); the chain (5)–(6) with Lemma 19 bounds it by $4\sum_i L_i + \frac{128}{3}\mathrm{VLB}(J)$, and $\mathrm{OPT}(J) \geq \max\{\sum_i L_i, \mathrm{VLB}(J)\}$ (Lemma 29) gives the constant, exactly as in Theorem 20. □

Theorem 38 (Epoch-GSA is constant-competitive for total completion time)**.** *In the release-time model (9) with unknown lengths $1 \leq L_i \leq R$ and kill-and-restart, Epoch-GSA (Definition 3, with the ladder and realization conditions of Theorem 20) is online, always feasible, and satisfies*

$$\sum_i C_i \;\leq\; 2\sum_i (r_i + L_i) \;+\; 10\sum_i L_i \;+\; 128\,\mathrm{VLB}(J) \;<\; 140\cdot \mathrm{OPT}_{\mathrm{TC}},$$

where $\mathrm{OPT}_{\mathrm{TC}}$ is the offline optimum total completion time in the same release-time model. Via Proposition 33, coverage Epoch-GSA is < 140-competitive against the coverage optimum on every star with residual $R = M - W \geq L_{\max}$ under online arrivals, tight M included.

Proof. Let $s_e = 2^e - 1$ be the start of epoch E_e, whose length is 2^e, and fix a job i. Let $e_i = \min\{e : s_e \geq r_i\}$; then $s_{e_i} \leq 2r_i$ (if $r_i = 0$ then $e_i = 0$ and $s_{e_i} = 0$; otherwise $s_{e_i - 1} \leq r_i - 1$ and $s_{e_i} = 2s_{e_i-1} + 1 \leq 2r_i - 1$). Let $f_i = \lceil \log_2 \widehat{C}_i \rceil$, so that $2^{f_i} \leq 2\widehat{C}_i - 1$ (as $\widehat{C}_i \geq 1$ is an integer and $2^{f_i - 1} < \widehat{C}_i$). Put $\hat{e} = \max\{e_i, f_i\}$. If i is still unfinished at $s_{\hat{e}}$, it belongs to $B_{\hat{e}}$ (it was released by $s_{e_i} \leq s_{\hat{e}}$), every member of $B_{\hat{e}}$ starts the epoch at age 0, and Lemma 37(a) makes the first batch of that epoch complete i within $\widehat{C}_i \leq 2^{f_i} \leq 2^{\hat{e}}$ ticks—inside the epoch, before the boundary can kill it. Hence in every case

$$C_i \;\leq\; s_{\hat{e}} + \widehat{C}_i \;=\; \max\{s_{e_i},\, 2^{f_i} - 1\} + \widehat{C}_i \;\leq\; \max\{2r_i,\, 2\widehat{C}_i - 2\} + \widehat{C}_i \;\leq\; 2r_i + 3\widehat{C}_i - 2.$$

Summing over i and applying Lemma 37(b),

$$\sum_i C_i \;\leq\; 2\sum_i r_i + 3\sum_i \widehat{C}_i - 2k \;\leq\; 2\sum_i (r_i + L_i) + 10\sum_i L_i + 128\,\mathrm{VLB}(J) - 2k.$$

Three lower bounds on $\mathrm{OPT}_{\mathrm{TC}}$ finish the proof: $C_i \geq r_i + L_i$ for every job in every schedule, so $\mathrm{OPT}_{\mathrm{TC}} \geq \sum_i (r_i + L_i)$; and dropping the release constraints only enlarges the feasible set, so $\mathrm{OPT}_{\mathrm{TC}} \geq \mathrm{OPT}(J) \geq \max\{\sum_i L_i, \mathrm{VLB}(J)\}$ (Lemma 29). Therefore $\sum_i C_i \leq (2 + 10 + 128)\,\mathrm{OPT}_{\mathrm{TC}} - 2k < 140\,\mathrm{OPT}_{\mathrm{TC}}$. Onlineness and feasibility are as in Theorem 36. □

The constant is three times the offline one, and the mechanism is worth stating plainly: the epoch grid costs a factor 2 in waiting (a job released at r_i waits at most until $2r_i$) and a factor 2 in ladder time (the first epoch long enough to hold $\widehat{C}_i$ has length $< 2\widehat{C}_i$), and the release-time floor $\sum_i (r_i + L_i)$ pays for the waiting while the arrival-free floor pays for the ladder. Any improvement of Theorem 20 transfers with the same factor three. Early re-batching cannot invalidate this argument: if i is unfinished at $s_{\hat{e}}$, it is in that epoch's first batch, which cannot drain before completing i. The argument bounds absolute completion times, not time since arrival.

Remark 6 (Related online guarantees and the flow-time question)**.** *Feng et al. (2026a) prove, in the additive model with arbitrary prompt lengths, that a doubling-stage router is $O(1)$-competitive for total completion time under online arrivals (their Theorem 9). At prompt length 0 every job is response-heavy in their taxonomy, their router degenerates to a doubling-stage ladder of the same shape as Epoch-GSA, and their proof charges per job in the same way; Proposition 33 therefore transports their $O(1)$ to stars as well, while Theorem 38 makes the constant explicit and self-contained. Whether some algorithm attains a constant for* flow *time in the prompt-free residual model remains open; Epoch-GSA itself does not, as shown below. In the prompt-bearing additive model every deterministic online algorithm has super-constant ratio for total flow under adversarial arrivals (Jaillet et al., 2025; Feng et al., 2026b), and the fully online nonpreemptive variant is $\Theta(n)$ (D'Ambrosio, 2026); those constructions use large prompts, so whether the prompt-free residual model—and hence coverage scheduling on stars—admits a constant for flow is open. Total completion time is the objective for which constant online ratios are known to be attainable, which is why it is the objective of Theorem 38. It is distinct from request latency: subtracting $\sum_i r_i$ from a competitive bound on total completion time does not preserve the competitive factor.*

Proposition 39 (Batching without geometric epochs is $\Omega(R)$)**.** Batch-and-run—*admit the currently released jobs, run the ladder until the batch is empty, and only then admit anything that arrived meanwhile—has $\mathrm{TC}/\mathrm{OPT}_{\mathrm{TC}} = \Omega(R)$ in the release-time model. Thus waiting for a batch to finish can fail even for total completion time; epoch boundaries are one way to limit this blocking.*

Proof. Take $L_1 = R$ released at $r_1 = 0$ and R unit jobs released at $r_i = 1$ (so $k - 1 = R$). Batch-and-run commits to the singleton batch $\{1\}$ and serves it for R ticks (its ladder must reach reservation $\tau_P \geq R$), so every unit job completes at $\geq R$, giving $\mathrm{TC} \geq R \cdot R = R^2$. The optimum runs job 1 at ages $0, \ldots, R-1$ and, at tick $t = 1$, fills the residual $R - (a_1 + 1) = R - 2$ with unit jobs, and serves the last two at $t = 2$ when $R \geq 5$; all R unit jobs therefore finish in $O(1)$ ticks and $\mathrm{OPT}_{\mathrm{TC}} = O(k + R) = O(R)$. The ratio is $\Omega(R)$. Epoch-GSA instead abandons the singleton at the first boundary past $t = 1$ and re-forms the batch, so the unit jobs are admitted after $O(1)$ ticks. □

Proposition 40 (Epoch-GSA has unbounded total-flow ratio)**.** *For powers of two $R \geq 8$, release one job of length R at time R and R unit jobs at time $R+1$. Epoch-GSA has*

$$\frac{\mathrm{TF}_{\mathrm{Epoch}}}{\mathrm{OPT}_{\mathrm{TF}}} \geq \frac{R(R-1)}{2R+2} = \Omega(R),$$

even with the immediate re-batching of Definition 3.

Proof. Time R lies in the epoch $[R-1, 2R-1)$. The system was idle, so the long job starts a singleton batch at R. It cannot complete before $2R$, even without restarts. Consequently this batch cannot drain before the epoch boundary $2R-1$, and none of the unit jobs enters a batch before that boundary. Each unit job therefore completes no earlier than $2R$, a flow of at least $R-1$, giving $\mathrm{TF}_{\mathrm{Epoch}} \geq R(R-1)$. A feasible comparison schedule runs the long job uninterrupted from R to $2R$. At tick $R+1$ its look-ahead weight is 2, so $R-2$ unit jobs run beside it and finish at $R+2$. At tick $R+2$ the long job has weight 3; the remaining two unit jobs fit since $R \geq 8$ and finish at $R+3$. Total flow is $R + (R-2) \cdot 1 + 2 \cdot 2 = 2R+2$, which upper-bounds $\mathrm{OPT}_{\mathrm{TF}}$ and proves the ratio. □

For the cohort implementation with the long job first in the fixed order and the GSA $\alpha_g = 2$ ladder, the same family gives exactly $\mathrm{TF} = R^2 + 2R$ and $\mathrm{TC} = 2R^2 + 4R$. The comparison schedule has $\mathrm{TC} = R^2 + 4R + 2$. Thus an increasing flow ratio coexists with a completion-time ratio below 2 on this family; the completion-time theorem does not establish an online response-time guarantee.

Proposition 41 (Unlimited capacity needs no length information)**.** *In the tick model with unlimited memory, the non-clairvoyant policy that serves every released unfinished job at every tick is optimal for both total completion time and total flow.*

Proof. Each job completes at $r_i + L_i$, the earliest possible time for that job under any schedule. These simultaneous per-job lower bounds are attained without reading L_i before completion. □

Computational checks. The program `polyserve/theory/online_arrivals_check.py` checks five parts of the online analysis:

- The release-time reduction agrees on 76,156 memory-test pairs from random schedules that respect arrivals.
- The lower bounds (11)–(12) hold on 1,320 instances against random feasible schedules and exact optima under all three transition rules. The ratio reaches 1.000, attaining Lemma 34.
- Against exact OPT on small instances, Epoch-GSA's largest ratios are 4.32 for total completion time and 5.56 for flow, using the strongest FLEX benchmark.
- On Proposition 39's family, batch-and-run worsens by a factor 2 per doubling of R ($3.04 \to 43.05$ over $R = 8 \ldots 128$). Epoch-GSA stays at 1.96–2.00 of the lower bound.
- On 1,500 random arrival instances with 48,527 jobs, $C_i \leq 2r_i + 3\widehat{C}_i - 2$ holds for every job and sometimes with equality. The instances include uniform, unit-flood, boundary-length and long-tail mixtures, with arrivals spread over at most $8R$. The offline bound reaches at most 0.67 of its budget. The largest total-completion ratio against $\max\{\sum_i (r_i + L_i), \mathrm{VLB}, \text{release-VLB}\}$ is 17.0.

Measured against the windowed floor alone the worst ratio drifts mildly with R ($8.25 \to 13.71$) while the ratio against a clairvoyant feasible schedule is flat (5.1–8.2): the drift is in the floor, which is why the theorem charges each job to its phase end in the arrival-free run. A paired sweep of 26 instances evaluates this particular epoch ladder on $(1+\varepsilon)R$ against the optimum on R. Its worst measured ratio stays between 4.70 and 6.16 for the tested augmentations; a large finite budget leaves a ratio of 4.96. These observations concern the epoch-and-restart policy, whose batch membership and restarts persist after capacity ceases to bind. They do not give a lower bound for other policies or identify an unavoidable cost of unknown lengths (Proposition 41). The regression suite additionally checks the exact cohort values in Proposition 40 for $R = 8, 16, 32, 64, 128, 256$.

B.14 Fork-width amplification: statement and proof

Corollary 42 (Fork-width amplification)**.** *Under Best-of-N forks with t tenants and width b on one long shared prefix ($W = kL$, $k = t \cdot b$, $M = W + kL$), Theorem 4 with $k = tb$ gives best-additive / coverage-OPT ratio $(k+1)/2 = \Omega(tb)$, with values $2.5, 4.5, 8.5, 16.5$ for $b \in \{1, 2, 4, 8\}$ at t=4.*

Proof. Set $k = tb$, $W = kL$, $M = W + kL$. Theorem 4 gives ratio $(k+1)/2 = \Omega(tb)$. At t=4: ratios $2.5, 4.5, 8.5, 16.5$ for $b \in \{1, 2, 4, 8\}$. □

Live admission with a fixed KV-block pool (A800, t=4 tenants on one long shared prefix) reproduces the fork shape: additive admission underuses available concurrency by factors $4, 8, 16, 32$ at $b \in \{1, 2, 4, 8\}$; two-group overlap at shared-root fraction $\rho \in \{0, 0.5, 1\}$ gives $4/4/8$ (g=2) and $4/4/6$ (g=3), the star at ρ=1 (`live_overlap_admission.json`).

B.15 Proof of Theorem 9

(a) Fix identical suffixes q. Coverage memory of an active set is $W + \sum_{i \in S}(q + a_i)$: the shared prefix is stored once, and each active job's private path (suffix plus decode) once. Look-ahead feasibility subtracts the constant W on both sides, leaving exactly the GSA predicate at prompt length $s = q$ and budget $R = M - W$ ($u_{i,t} = a_i$, occupancy $q + u + 1$ per active job). The identity map on schedules preserves kill-and-restart, completions, and flow, as in Theorem 6; the GSA hypotheses hold with $s = q$ (identical prompts by construction; per-job feasibility $q + L_i \le R$ by assumption; the remaining hypotheses are unchanged from Appendix B.7). The imported guarantee at $s = q$ is the theorem's native setting—no boundary case is needed.

(b), arbitrary suffixes. Subtracting the resident prefix leaves $\sum_{i \in S_t}(q_i + a_i + 1) \le R$ at every tick. This is the arbitrary-prompt model of Feng et al. (2026a), with $s_i = q_i$, budget R, and unknown response lengths $o_i = L_i$. Keeping the schedule unchanged preserves resets, completions, and total completion time, as in Theorem 6. The model hypotheses hold as follows. Jobs are present at $t = 0$, or arrive over time for their online Theorem 9. The private prompt lengths $s_i = q_i \ge 1$ are known at arrival, and response lengths $o_i \ge 1$ are unknown until completion. Individual feasibility is $s_i + o_i \le R$, equivalent to $q_i + L_i \le R$. Both models use integer rounds and kill-and-restart. Their regime-aware router run on the residual is therefore constant-competitive for total completion time against the coverage optimum on the suffixed star (≤ 996 by the explicit analysis in their proof of Theorem 1), constant-competitive for makespan (their Theorem 8), and $O(1)$-competitive for total completion time under online arrivals (their Theorem 9), with no relation assumed among the q_i. The import is needed only for arbitrary suffixes; the star case $q_i \equiv 0$ is outside their model ($s_i \in \mathbb{N}^+$) and is covered self-contained by Theorem 20 and Theorem 38.

(c) With coverage-feasible budget $M = W + \sum_i (q_i + L) \le W + k(\bar{q} + L)$, two jobs at any ages have additive look-ahead cost $\ge 2W + 2 > W + k(\bar{q} + L) \ge M$ whenever $W \ge k(\bar{q} + L)$, so width one is forced at every tick; the serial lower bound and the coverage upper bound (all k jobs fit at every pre-completion age by the budget's definition) then reproduce the proof of Theorem 4 verbatim. □

B.16 Partially shared prompts and the limit of fixed request costs

Two-group overlap: $\mathrm{Mem}_{\mathrm{cov}} = (2 - \rho)W + \alpha \sum a_i$; additive overcount $W(2k - 2 + \rho)$ increases in ρ. A g-group trie has overcount $W(n - g + (g - 1)\rho)$; at ρ=1 this is a star. Depth-2 shared-bundle forests with private ε-edges reduce to stars up to $\varepsilon|S|$ and preserve the long-prefix $\Omega(k)$ separation for $\varepsilon = o(L)$.

Proof of Proposition 8. *Forest direction.* With disjoint shared prefixes and a static split $M = \sum_g M_g$, an active set S costs $\sum_{g:\, S \cap g \neq \emptyset} \big(W_g + \sum_{i \in S \cap g} a_i\big) \le \sum_g \big(W_g + \sum_{i \in S \cap g} a_i\big)$, and enforcing the per-group predicates $\sum_{i \in S \cap g}(a_i + 1) \le R_g$ tick-by-tick is exactly the product of the star identity maps of Theorem 6; kill-and-restart semantics, completions, and flow are componentwise. (A *dynamic* split that moves budget between groups is an admission policy on top of the reduction, not part of the identity map.) *Trie counterexample.* Take a root edge u with two child edges v, w ($|u| = |v| = |w| = 10$), jobs $1, 2$ on leaf $u \cdot v$, jobs $3, 4$ on leaf $u \cdot w$, all ages 0 (each active job contributes $a_i + 1 = 1$), budget $M = 25$. Coverage costs: any singleton 21; same-branch pairs $\{1, 2\}, \{3, 4\}$ cost 22 (feasible); cross-branch pairs cost 32 (infeasible); all larger sets ≥ 33 (infeasible). Suppose some fixed costs $c_1, \ldots, c_4$ and threshold R' represented this family. Feasibility of the two same-branch pairs gives $(c_1 + c_2) + (c_3 + c_4) \le 2R'$; infeasibility of the two cross-branch pairs $\{1, 3\}, \{2, 4\}$ gives $(c_1 + c_3) + (c_2 + c_4) > 2R'$. The two left-hand sides are the same sum $c_1 + c_2 + c_3 + c_4$—contradiction. Hence no fixed-cost additive threshold reproduces coverage feasibility on this trie even at a single fixed budget and fixed ages. This obstruction rules out a general fixed-cost additive representation. (A single partially shared node with one branch, as in the three-job instance, is *not* enough: its feasible families are threshold-representable; two branches create the exchange obstruction.) Visiting each prefix group in turn during every doubling phase gives Interleaved Group Doubling (IGD). Packing requests from different groups into feasible waves gives Branch-packed Geometric Scheduling (BPGS). The same four jobs are the exclusive forest of Lemma 46 at residual $M - |U| = 15$: coverage scheduling is forest IGD/BPGS there (Theorem 48), which is a different question from representing the feasible family by additive costs. □

Admission with a general prefix tree. For a prefix tree T with edge weights w_e, an active set costs $\mathrm{cov}(S) = \sum_{e \in \bigcup_{i \in S} P_i} w_e + \alpha \sum_{i \in S} a_i$. The prefix term is monotone and submodular: adding a request never reduces memory, but can cost less when more of its prefix is already active. Even selecting as many requests as fit, "maximize $|S_t|$ subject to $\mathrm{cov}(S_t) \le M$," must account for which requests share memory. Proposition 8 shows that fixed per-request costs cannot always replace this calculation. Three structured cases remain

tractable. First, fully resident prefix groups with a fixed budget per group admit the separate reductions in Proposition 8. Second, private edges of size ε add a corresponding private cost below a common prefix. Third, subtracting a common root from a depth-two trie leaves disjoint groups with budget $M - |U|$ (Lemma 46). The sequential and packed group schedules below apply to this last case under their respective capacity conditions. Competitive scheduling for deeper partial sharing, with memory reassigned between groups over time, remains open.

B.17 Dynamic cross-group budget on disjoint star forests

Setting. Groups $g = 1, \dots, G$; group g has shared prefix weight W_g and k_g jobs with unknown private lengths; all jobs present at $t = 0$; kill-and-restart; $\alpha = 1$. One *global* budget M replaces the static split of Proposition 8: at every tick the scheduler picks $S_t \subseteq \{1, \dots, k\}$ ($k = \sum_g k_g$) subject to the coverage look-ahead predicate

$$\sum_{g:\, S_t \cap g \neq \emptyset} W_g \;+\; \sum_{i \in S_t} (a_i + 1) \;\le\; M. \tag{13}$$

A shared prefix occupies memory exactly while its group has an active member, and loading or evicting a shared prefix is free in time (the tick model charges decode ticks only, matching the imported GSA model). Write $R_g := M - W_g$, $R_{\min} = \min_g R_g$, $R_{\max} = \max_g R_g$, $\beta := R_{\max}/R_{\min} \ge 1$, and let OPT be the offline optimum under (13).

Why the split must move. If groups are serialized, or the split is frozen, the penalty is linear in the batch. A dynamic schedule completing all jobs exists iff $\max_g(W_g + L_{\max,g}) \le M$, whereas a static split $M = \sum_g M_g$ completing all jobs exists iff $\sum_g (W_g + L_{\max,g}) \le M$: on the region between the two thresholds the static-vs-dynamic ratio is $+\infty$. Where a static split is feasible, the penalty is exactly the per-group batch size.

Theorem 43 (Static-split penalty is $\Theta(k)$)**.** *Let* $\mathrm{STAT}(I)$ *be the minimum flow over static splits* $M = \sum_g M_g$*, each group scheduled offline-optimally on its residual* $M_g - W_g$*.* (i) Lower bound. *For the family* F_k*:* $G = 2$*, identical lengths* L*,* k *jobs per group,* $W_1 = W_2 = W = kL$*,* $M = 2(W + L)$*,* $\mathrm{STAT}(F_k) = L\,k(k+1)$*,* $\mathrm{OPT}(F_k) \le 3kL$*, and* $\mathrm{STAT}/\mathrm{OPT} \ge (k+1)/3$*.* (ii) Upper bound (identical lengths). *For every instance with identical lengths* L *on which some static split is feasible,* $\mathrm{STAT} \le \frac{k_{\max}+1}{2}\mathrm{OPT}$ *with* $k_{\max} = \max_g k_g$*; hence over identical-length forests the worst-case penalty is* $\Theta(k_{\max})$*.*

Proof. (i) The split constraints $M_g \ge W + L$ and $M_1 + M_2 = 2(W + L)$ force per-group residual L, the tight residual of Theorem 11(b) where serial lockstep is exactly optimal with flow $L\,k(k+1)/2$ per group. Dynamically, serialize the groups: group 1 alone has residual $M - W = kL + 2L$, so all k jobs run in one wave finishing at time L with flow kL, and group 2 repeats on $[L, 2L]$; total $3kL \ge \mathrm{OPT}$. Exact enumeration confirms equality of both quantities at $k \in \{2, 3, 4\}$; results are recorded in `e1_separation.json`. *(ii)* Any feasible split has $R_g^{\mathrm{stat}} \ge L$, so serial lockstep is feasible in every group and $\mathrm{STAT} \le \sum_g L\,k_g(k_g+1)/2 \le \frac{k_{\max}+1}{2} L \sum_g k_g$, while every schedule has $\sum_i C_i \ge \sum_i L_i = L \sum_g k_g$. □

Completing one entire group before serving another can also be costly. Take $G = 2$, $W_1 = W_2 = 3T$, and $M = W + T$, with $k - 1$ unit jobs and one length-T job per group. The prefixes cannot fit together, and finishing either group takes at least T ticks. Any whole-group order therefore has ratio $\ge k/7$, whereas OPT can switch between groups to finish short jobs earlier (`e5_serialize_bad.json`). The positive result therefore needs a schedule that visits all groups every phase.

Group-wise volume bound. For group g let VLB_g be the star VLB of its jobs at residual R_g, over its sorted lengths.

Lemma 44 (Forest VLB)**.** *Every schedule feasible under* (13) *has* $\sum_i C_i \ge \sum_g \mathrm{VLB}_g$ *and* $\sum_i C_i \ge \sum_i L_i$*.*

Proof. At any tick where g is active, (13) gives $\sum_{i \in S_t \cap g}(a_i + 1) \le M - \sum_{h \text{ active}} W_h \le R_g$, so group g's per-tick served weight never exceeds its dynamic residual regardless of what other groups do. Applying the argument of Lemma 29 to group g's completions alone gives $\sum_{i \in g} C_i \ge \mathrm{VLB}_g$, which sums over g; the linear bound is $C_i \ge L_i$. □

The algorithm. Interleaved group doubling (IGD) visits every prefix group in each doubling phase. It applies the cohort rule of Lemma 32 within each group. In phase p, groups are visited in a fixed order; the visit of group g attempts every currently-unfinished job of g once, in cohorts of width $b_p^g = \max\{1, \lfloor R_g/2^p \rfloor\}$ reserved 2^p ticks. Only one group is active at a time. When $2^p \le R_g$, its private peak is at most $b_p^g 2^p \le R_g$. When $2^p > R_g$, the width is one and the job finishes before its private weight exceeds $L_i \le R_g$. Thus IGD is

feasible and non-clairvoyant: its width reads only W_g, M, and p, and the fallback uses the individual-feasibility promise without reading L_i.

Theorem 45 (IGD is $O(G\,\beta \log L_{\max})$-competitive)**.** *On every dynamic disjoint-star-forest instance,*

$$\sum_i C_i^{\mathrm{IGD}} \;\le\; G\left(32\,\beta\left(\lceil \log_2 L_{\max}\rceil + 1\right) + 4\right)\cdot \mathrm{OPT}.$$

In particular, for $G = 2$ with comparable shared prefixes ($\beta = O(1)$) IGD is $O(\log L_{\max})$-competitive against dynamic OPT, where every static split is $\Omega(k)$-bad (Theorem 43).

Proof. Let $q_i = \lceil \log_2 L_i \rceil$, $n_p^g = \#\{i \in g : q_i \ge p\}$, $n_p = \sum_g n_p^g$, $P = \max_i q_i$. A job attempted in phase $p \ge q_i$ completes within its reservation, so group g's phase-p visit attempts at most n_p^g jobs and the algorithm terminates by phase P. A visit lasts $\lceil n_p^g/b_p^g\rceil 2^p \le 2n_p^g 4^p/R_g + 2^p[n_p^g \ge 1]$; both cases of 2^p relative to $R_g/2$ give this bound. Summing visits gives $D_p \le \sum_g (2n_p^g 4^p/R_g + 2^p[n_p^g \ge 1])$ and $\sum_i C_i \le \sum_p D_p n_p$. The resulting terms are $\sum_p 2\cdot 4^p n_p \sum_g n_p^g/R_g$ and $\sum_p 2^p n_p \#\{g : n_p^g \ge 1\}$. For the main term, class-$\ge p$ jobs have $\binom{L+1}{2} \ge 4^p/8$ and occupy the top reverse-ranks of the sorted VLB sum, so $(n_p^g)^2 \le 16 R_g \mathrm{VLB}_g/4^p$ and, by Cauchy–Schwarz and $R_g \le R_{\max}$, $n_p \sum_g n_p^g/R_g \le G/R_{\min}\sum_g (n_p^g)^2 \le 16G\beta 4^{-p}\sum_g \mathrm{VLB}_g \le 16G\beta\, 4^{-p}\mathrm{OPT}$; summing over the $P+1$ phases gives $\le 32G\beta(P+1)\mathrm{OPT}$. For the rounding term, $\#\{g : n_p^g \ge 1\} \le G$ and $\sum_p 2^p n_p \le 4\sum_i L_i \le 4\mathrm{OPT}$, so it is $\le 4G\,\mathrm{OPT}$. □

IGD's dependence on G can be necessary: on G singleton groups, its ratio is exactly $(G+1)/2$. When all groups fit together, packing them in the same phase finishes this instance in one tick. The packing algorithm in Appendix B.18 is 4-competitive under its all-groups-fit condition. When no two prefixes fit together, it reduces to IGD. The measured IGD ratio falls from 2.6 to 1.7 as β grows from 1 to 7 (`e8_beta_sweep.json`); this observation does not establish that dependence on β can be removed from the worst-case bound.

B.18 Shared-root depth-two tries and branch-packed geometric scheduling

The four-request counterexample in Proposition 8 has a common root followed by two child prefixes. Subtracting the root gives the same memory constraint as two disjoint prefix groups for every nonempty active set.

Lemma 46 (Shared-root depth two is a forest)**.** *Let T be a trie with a single root edge of weight U and disjoint child shared prefixes of weights $V_1, \ldots, V_G$, each carrying a set of private jobs. For an active set S write S_g for the jobs of S on child g, and let a_i be private kill-and-restart age. Then*

$$\mathrm{Mem}_{\mathrm{cov}}(S) \;=\; \begin{cases} 0 & \text{if } S = \emptyset, \\ U + \displaystyle\sum_{g:\, S_g \ne \emptyset} V_g + \sum_{i \in S}(a_i + 1) & \text{otherwise.} \end{cases}$$

Hence on every nonempty S, coverage look-ahead feasibility $\mathrm{Mem}_{\mathrm{cov}}(S) \le M$ is exactly forest feasibility (13) on shared prefixes (V_g) at budget $M - U$.

Proof. Every nonempty path uses the root, so U is paid once; child g is paid iff S meets that child; private look-ahead is additive. The empty set occupies nothing on either model. (Computational check: every subset of the four-job instance, at private ages in $\{0,1,2\}$, matches the forest formula; singletons cost 21, same-branch pairs 22, cross-branch pairs 32 at $U = V_1 = V_2 = 10$, $M = 25$, which is the exclusive forest $V_1 = V_2 = 10$, $M - U = 15$; `bpgs_certs.json`.) □

The four-request example is exclusive in the forest: $V_1 + V_2 = 20 > 15 = M - U$, so no schedule co-runs the two branches. Additive look-ahead still serializes even same-branch pairs (cost $42 > 25$). Coverage does not: a same-branch pair fits. This example can be scheduled by treating each branch as a prefix group. It still cannot be represented by one fixed additive cost per request, as Proposition 8 proves.

The algorithm. Branch-packed geometric scheduling (BPGS) uses the same attempt durations as IGD but packs different groups together whenever they fit. Phases $p = 0, 1, \ldots$ run while unfinished jobs remain; phase p attempts every currently-unfinished job once at reservation $\tau = 2^p$. Inside the phase, waves are packed by first-fit in a fixed group order.

Assume every request is individually feasible: $W_g + L_i \le M$ for $i \in g$. This is an instance promise, not an observed output length. If a wave is empty and $\tau > R_g$, the singleton fallback serves one request in isolation. It completes by age $L_i \le R_g$ even though the reserved interval is longer. The wave then idles until the end of that interval; no additional group is packed beside the singleton. Both admission branches read only the group shared prefixes, the budget, the phase slice, and unfinished membership. Completions are the only length-dependent observations.

Algorithm 2 Branch-packed geometric scheduling (one phase)

Require: unfinished jobs by group, shared prefixes (W_g), budget M, reservation τ

$\text{Need}_g \leftarrow$ unfinished jobs of group g ▷ one attempt each this phase
while some $\text{Need}_g \neq \emptyset$ **do**
  $G' \leftarrow \emptyset,\ \text{used} \leftarrow 0$ ▷ conservative reservation $\sum W + w\tau$
  for groups g in order with $\text{Need}_g \neq \emptyset$ **do**
    $w \leftarrow \min\big(|\text{Need}_g|, \lfloor (M - \text{used} - W_g)/\tau \rfloor\big)$
    if $w \geq 1$ **then**
      admit the next w jobs of g; $\text{used} \leftarrow \text{used} + W_g + w\tau$; $G' \leftarrow G' \cup \{g\}$
    else if $\text{used} = 0$ and $\tau > M - W_g$ **then**
      admit one job of g alone; $G' \leftarrow \{g\}$
      break ▷ singleton fallback; no later group joins this wave
    end if
  end for
  run admitted jobs for a reserved τ ticks, removing each on completion
  kill unfinished members; drop all attempted jobs from Need
end while

Proposition 47 (BPGS is coverage look-ahead feasible)**.** *Every BPGS wave satisfies*

$$\sum_{g \in G'} W_g + \sum_{i \text{ admitted}} \min(L_i, \tau) \leq M.$$

Consequently every tick of the reservation obeys (13)*. On a shared-root depth-two trie the same schedule is feasible at budget M by Lemma 46.*

Proof. The conservative pack maintains $\text{used} \leq M$ with used an upper bound on shared prefix weight plus τ per admitted job, hence on peak look-ahead as ages grow. A fallback wave contains exactly one request, whose peak is $W_g + \min\{L_i, \tau\} \leq W_g + L_i \leq M$ by individual feasibility. Early completions only drop occupancy. Every nonempty Need yields at least one admitted job, so each phase makes progress, and by phase $\lceil \log_2 L_{\max} \rceil$ every job has completed. The admission rule never uses $\min\{L_i, \tau\}$; that expression is only an upper bound in this feasibility proof. □

Theorem 48 (Exclusive shared prefixes: BPGS is IGD)**.** *If $W_g + W_h > M$ for all $g \neq h$, every BPGS wave contains one group g and takes $\min\{|\text{Need}_g|, b_p^g\}$ jobs, where $b_p^g = \max\{1, \lfloor R_g/\tau \rfloor\}$ is IGD's nominal width. For the same group order, the sequence of reserved intervals is identical to IGD. In particular BPGS inherits Theorem 45: it is $O(G\,\beta \log L_{\max})$-competitive on exclusive forests, and $O(\log L_{\max})$-competitive for two comparable exclusive shared prefixes.*

Proof. A nonempty wave already paying W_g cannot add h: the next conservative increment is at least $W_h + \tau$, and $W_g + W_h > M$. The unique group g therefore sees residual $R_g = M - W_g$ and width $\min(n_g, \lfloor R_g/\tau \rfloor)$, or width 1 when that floor is 0, which is b_p^g. First-fit in the IGD group order then drains group g's Need in successive cohorts before touching the next group, which is IGD's visit. The four-request forest ($M - U = 15$) is exclusive: BPGS and IGD both complete at flow 6 (makespan 2), while additive width-one geometric flow is 10. □

Theorem 49 (Prefixes that fit together: one wave and 4-competitive independent of G)**.** *Write n_p for the number of unfinished jobs at the start of phase p, and G_p for the number of groups that still have one.* (i) Phase. *If $\sum_{g:\, n_p^g \geq 1} W_g + n_p \cdot 2^p \leq M$, then BPGS packs every unfinished job into a single wave of duration 2^p. IGD's phase lasts at least $G_p \cdot 2^p$ whenever $G_p \geq 1$.* (ii) Global. *If $\sum_g W_g + N \cdot 2^{\lceil \log_2 L_{\max} \rceil} \leq M$ with $N = \sum_g k_g$, then every phase satisfies (i), BPGS is 4-competitive against forest OPT, and the ratio does not depend on G.*

Proof. (i) First-fit takes the entire Need of each unfinished group in order: after packing a prefix of groups at full width, the leftover budget is at least the shared prefixes plus $n\tau$ of the groups not yet packed, so the next group enters at full Need. The wave therefore contains every unfinished job and lasts $\tau = 2^p$. IGD visits groups serially, at least τ ticks each. *(ii)* The hypothesis implies (i) at every $p \leq \lceil \log_2 L_{\max} \rceil$, since $n_p \leq N$ and unfinished shared prefixes are a subset. A job of class $q_i = \lceil \log_2 L_i \rceil$ completes in phase q_i, hence by time $\sum_{p \leq q_i} 2^p = 2^{q_i+1} - 1 \leq 4L_i$. Flow is at most $4 \sum_i L_i \leq 4\,\text{OPT}$ (Lemma 44). On G zero-prefix unit jobs with $M \geq G$ the hypothesis holds with room 0: BPGS flow equals $\text{OPT} = G$ (one tick) while IGD's ratio is

Algorithm 3 Backfill with a restart-depth bound

Require: unfinished $\mathcal{U}$, ages (a_i), depths (h_i), attempt order (π_i), residual R
$p \leftarrow \min \mathcal{U}$ ▷ FIFO head; served every tick until completion
$S \leftarrow \{p\}$, $\sigma \leftarrow a_p$
for $i \in \mathcal{U} \setminus \{p\}$ with $a_i > 0$, in increasing π_i **do**
  if $\sigma + a_i + |S| + 1 \leq R$ **then**
    $S \leftarrow S \cup \{i\}$, $\sigma \leftarrow \sigma + a_i$
  end if
end for
for $j \in \mathcal{U} \setminus \{p\}$ with $a_j = 0$, in increasing j **do**
  $d_j \leftarrow \max\{1, 2h_j\}$
  if $\sigma + (|S| + 1)d_j \leq R$ **then**
    $S \leftarrow S \cup \{j\}$ ▷ guaranteed service; do not kill at this depth
  end if
end for
assign new increasing attempt numbers π_i to selected jobs with $a_i = 0$
reset $a_i \leftarrow 0$ for $i \in \mathcal{U} \setminus S$; advance $a_i \leftarrow a_i + 1$ for $i \in S$
observe completions and remove completed jobs from $\mathcal{U}$
$h_i \leftarrow \max\{h_i, a_i\}$ for every selected job still unfinished
return S

exactly $(G+1)/2$. The bound uses only individual completion lengths and the one-wave condition; it does not require a length prediction. □

One group is residual-slot doubling (Lemma 32), not the staggered pipeline of Coverage-GSA. On mixed instances, where only some groups fit together, the same observable greedy pack remains feasible. The exclusive and global all-groups-fit bounds above do not supply a constant guarantee for this intermediate regime; such a result needs an additional packing analysis. On 300 random two-group forests, the observable implementation has mean flow/OPT 1.83 versus IGD's 2.39; it improves flow on 225 instances and ties on 75. This is a finite-instance comparison, not a dominance guarantee (`bpgs_suite.json`). Depth-three overlap (a root with children that themselves share further prefixes across groups) is not a forest, and Lemma 46 stops there.

C BACKFILL WITH A RESTART-DEPTH BOUND

Coverage-GSA is constant-competitive versus coverage OPT (Theorem 20). At tight M, repeated attempts can generate more private tokens than running requests to completion; Table A1 compares this work directly in the tick model. Greedy progress packing (Algorithm 1) backfills the leftover residual by current age and is ≥ 5.5-bad on an explicit family (Proposition 31), because fillers can be killed just short of completion and restarted without a deeper window. Algorithm 3 always serves the first unfinished request in FIFO order. It uses spare memory for other requests, but restarts one only if it can fit for $d_j = \max\{1, 2h_j\}$ ticks. Here h_j is the greatest number of tokens reached by an attempt that was still unfinished. This records an observation, not retained KV or a predicted output length. Reaching d_j does not kill the job.

The admission test guarantees that each newly started secondary request can run to the chosen depth unless it finishes first (Lemma 52). Protecting a head request while backfilling spare capacity is classical in parallel-job scheduling (Shmueli & Feitelson, 2005). Here the additional condition limits repeated attempts when output lengths are unknown and interrupted progress is discarded. Length-predicting volume rules such as SVF (Kong et al., 2026) are a different information model.

All jobs arrive at time 0, with $1 \leq L_i \leq R$. A selected job advances one private token per tick; feasibility is $\sum_{i \in S}(a_i + 1) \leq R$. An unselected unfinished job loses its private progress. Initialize all ages and observed depths at zero. A global increasing counter orders newly started attempts; simultaneous starts use selection order, with the protected head first. The depth h_i records only an observed incomplete age, never a completed job's length. The policy reads only (i, a_i, h_i, π_i, R).

Proposition 50 (Feasibility and termination)**.** *The head is always individually feasible. Continuing attempts are admitted only when the next-tick weight fits. A cold restart with $d_j \geq 1$ implies the next-tick weight fits. The head's index is nondecreasing and the head is never reset, so every instance finishes in finitely many ticks.*

Theorem 51 (Per-job FIFO dominance)**.** *For every input and every job i, $C_i \leq \sum_{j \leq i} L_j = C_i^{\mathrm{FIFO}}$. Hence $C_{\max} \leq C_{\max}^{\mathrm{FIFO}}$ and $\sum_i C_i \leq \sum_i C_i^{\mathrm{FIFO}}$.*

Table A1: Tick-model comparison on the length vectors specified below ($R = 256$; completion at the true last token). These are CPU ticks, not GPU seconds. Backfill with a restart-depth bound is request-wise no later than serial FIFO (Theorem 51) and uses 28/83 attempts against RPP's 381/2680. RPP's makespan is lower on these two instances; the proved bound controls the number of attempts and generated tokens.

k	Policy	Makespan	Mean-CT	Generated	Attempts
16	serial FIFO	1792	840	1792	16
16	cohort-GSA	2655	944	4252	130
16	depth-bounded	1152	328	2394	28
16	RPP	1024	320	4535	381
32	serial FIFO	3632	1810	3632	32
32	cohort-GSA	5168	1756	8624	262
32	depth-bounded	2176	690	5408	83
32	RPP	2048	770	15295	2680

Proof. Until i completes, the head index is at most i. Every tick serves the current head by one token. Once a job becomes head it is never reset, so every head-token charged to job $j \le i$ lies on j's unique successful run and there are at most L_j such tokens. Partitioning the ticks before C_i by head identity yields $C_i \le \sum_{j \le i} L_j$. Successful fillers can only complete additional jobs earlier. □

The comparator is serial FIFO. For identical lengths $L_i \equiv R$, serial execution is already optimal (Theorem 11). It is not a constant-competitive claim versus coverage OPT, and it does not inherit $140/3$.

Lemma 52 (Service guaranteed before a restart)**.** *If j is admitted as a cold filler against a set S_0, then either j completes at depth $< d_j$, or j is not excluded for capacity before reaching age d_j.*

Proof. Attempts started after j receive later order and cannot evict j. Attempts excluded at j's admission reset and, if restarted, also receive later order. The only job that can outrank j and not already lie in S_0 is a later head. While the admission-time head $p_0 \in S_0$ is unfinished, it remains first. A new head can appear only after p_0 completes, and its current attempt must have started after j was admitted. At the ℓ-th subsequent tick ($\ell \le d_j$), this new head's weight is at most ℓ. The reserved continuation $a_{p_0}(0) + \ell$ of the old head covers that weight. At most one such foreign head exists at a time. Thus the weight of everyone in front of j, plus j, is at most $\sum_{i \in S_0} a_i(0) + (|S_0| + 1)\ell \le R$ for all $\ell \le d_j$. □

Corollary 53 (Private work and attempts)**.** *Failed attempt depths $f_1, \dots, f_m$ of any job i satisfy $f_{r+1} \ge 2f_r$ and $f_m < L_i$, so $\sum_r f_r < 2L_i$ and the private tokens generated for i are $< 3L_i$. The number of attempts is at most $1 + \lceil \log_2 L_i \rceil$.*

The work bound counts private decode tokens, not prefill, kernel launch, or page allocation.

The $k = 16$ vector, in FIFO order, is $(256, 128, 64, 32, 32, 16, 16, 16, 16, 256, 128, 256, 64, 128, 256, 128)$. The $k = 32$ vector appends $(128, 64, 64, 256, 32, 256, 128, 16, 32, 256, 64, 16, 128, 128, 256, 16)$. These are prompt-free residual instances; the table measures decoded-token ticks and does not model private prompt suffixes or GPU throughput. On these two vectors the tick-model makespan gain versus serial FIFO is $1.56\times/1.67\times$ and versus legal cohort-GSA $2.30\times/2.38\times$ (mean-CT $2.56\times/2.62\times$ and $2.88\times/2.54\times$). Peak private occupancy is 256 in both cells. A 15,029-instance enumeration ($R \le 6$, $k \le 5$, every ordered length vector) found no violation of feasibility, Theorem 51, Lemma 52, or Corollary 53. On the RPP counterexample of Proposition 31 the new flow is 591 against RPP's 1123 (offline atomic 203). We also test two hundred held-out synthetic vectors from five length families, with $k \in \{16, 32\}$ and seeds 1000–1019. All improve on serial FIFO and cohort-GSA in pure ticks. Under the hypothetical step cost $1 + 0.01(b - 1) + 5s$, the long-output family improves on both in only $2/40$ instances.

D FINITE-INSTANCE CHECKS

The deterministic programs in `polyserve/theory/` compare closed forms, feasible schedules, and exact optima on finite instances (Table A2). The GSA, $s = 0$, retaining-benchmark, online-arrival, and dynamic-budget checks are described with the corresponding proofs above. Backfill with a restart-depth bound is checked by `prove_depth_backfill_bundle` (15,029 small instances plus the published tight-M vectors).

Table A2: Finite-instance comparisons (`coverage_scheduling.json`). Long-prefix family $W{=}kL$, $M{=}W{+}kL$, $\alpha{=}1$ unless stated; OPT is the exact atomic-schedule optimum.

Family	Result
Long-prefix, $k \in \{4, 8, 16, 32, 64\}$	coverage/OPT $= 1.00$; best additive/OPT $= (k{+}1)/2$ exactly $(2.5, 4.5, 8.5, 16.5, 32.5)$; $s^\star{=}1$; sticky wave ratio $= k$
Identical lockstep	full-parallel and $R{=}L$ exact; intermediate R: lockstep $>$ OPT ($23{<}27$ etc.)
Doubling (identical)	full/tight at $(k, L) \in \{(4, 5), (8, 4), (8, 8), (16, 4)\}$: ratios ≤ 3.13 vs $F^\star$
Doubling (heterogeneous, full parallel)	$W{=}kL_{\max}$ / $W{=}\max(kL_{\max}, 8)$: mean 2.14, max 2.43 (bound 4)
Short-prefix counterexample	$W{=}10$, $L{=}4$, $k{=}32$: best additive 2.3125

E Reproducibility

The CPU regression suite `polyserve/theory/revision_regression.py` replays the observable branch-packing rule, the online flow family, the per-job completion-time charge, and a worked GSA schedule. It fixes the random forest seed at 20260925 and records the tested instances and aggregate results in `artifacts/theory_revision/`. The separate `s0_selfcontained_check.py` program checks the floor sum, staggered occupancy, doubling ladder, Abel charge, and both offline realizations against exact small-instance optima. These numerical checks accompany the general proofs; their finite instance counts do not establish the universal competitive bounds.

F Systems experiments: protocols and additional results

F.1 Prompt and block accounting

A request contains a common prefix of length W and a private prompt suffix q_i. At a generated length a_i, a token-level coverage estimate is $W + \sum_{i \in S}(q_i + a_i)$. For the $s = 16$-token pages in our vLLM configuration, we assume that only the complete prefix pages can be shared. Writing $h = \lfloor W/s \rfloor$ and $r = W - sh$, the block count is

$$h + \sum_{i \in S} \left\lceil \frac{r + q_i + a_i}{s} \right\rceil .$$

The additive comparator counts each full prompt separately. An output cap is an API input limit available before generation; it is distinct from a realized output length. The engine's total pool and allocator-reserved pages determine the capacity available to admission. Comparisons use the same physical pool, model, prefix-cache setting and output cap across policies.

What timing includes. Completion times include queueing after the common arrival, private-suffix prefill, decode, repeated attempts and scheduler overhead. Model loading is excluded. The common prefix is prefetched before the measured burst, so these experiments assume a resident prefix. Engine-step observations provide allocated-block counters at the step boundary; they do not measure a within-step peak. A decode-round guarantee treats one token per active request as one tick, whereas elapsed time also depends on batch width and prefill work.

F.2 Coverage peak reservation pseudocode

Algorithm 4 reserves a public output cap before admitting each request. Replacing that cap with the request's realized output length would give the policy information unavailable before generation. GSA instead discovers completion through repeated attempts of increasing duration.

The additive comparator starts with $u = 0$ and adds $W + q_i + \widehat{L}_i$ for every admitted request. At block granularity, replace token sums by the corresponding shared and private block counts. The cap-based policy requires every input request to fit alone; completed jobs release their private reservation.

Algorithm 4 Shared-prefix admission with known output caps

Require: resident shared-prefix weight W, capacity M, queue $\mathcal{U}$, private prompt lengths q_i, output caps $\widehat{L}_i$

$S \leftarrow \emptyset$; $u \leftarrow W$
for i in queue order **do**
  if $u + q_i + \widehat{L}_i \le M$ **then**
    $S \leftarrow S \cup \{i\}$; $u \leftarrow u + q_i + \widehat{L}_i$
  else
    stop this admission pass
  end if
end for
return S

Corollary 54 (Tick-to-wall-clock transfer)**.** *Let a decode step at batch width b take wall time $g(b)$, nondecreasing in b. Define $c = \max_b g(b)/g(1)$ over the widths a given schedule admits. Each completion time lies between $g(1)$ and $c\,g(1)$ times its tick count. The same scaling applies to flow and makespan. A ρ-competitive tick schedule is therefore $c\rho$-competitive in wall time against the optimum over the same feasible schedules. The statement assumes that every charged operation follows this step-cost model. Prefill, recomputation and scheduler overhead require their own costs before applying it to a complete serving system.*

Proof. A completion at tick C_i has wall time $\sum_{t \le C_i} g(b_t) \in [g(1)\,C_i,\ c\,g(1)\,C_i]$. The wall-clock optimum is a feasible tick schedule, so its wall flow is at least $g(1)$ times the tick optimum's flow; the algorithm's wall flow is at most $c\,g(1)$ times its tick flow, which is at most $c\,g(1)\,\rho$ times the tick optimum's flow. Chaining the two gives the $c\rho$ ratio; the same argument applies verbatim to makespan. □

G SERVING-TRACE STRUCTURE AND SELECTED-GROUP REPLAY

Prefix-group structure in public serving traces. The Mooncake trace (Qin et al., 2025a) records prefix-consistent hashes for complete 512-token blocks and each request's output length. These hashes identify shared prefixes without exposing prompt content. They determine W up to a 512-token floor. The tool/agent hour contains 23,608 requests. Within 10 s arrival windows, 51.0% belong to a shared-prefix group with $k \ge 8$ and $W \ge k\bar{L}$. The fraction is 53.7% when groups of any $k \ge 2$ are included. Median output and input lengths are 30 tokens and 6.3k tokens, respectively. Three template families have shared prefixes of 2.5–6.1k tokens, followed by private suffixes (Table A4). Their prefix and output lengths match the scale of the long-prefix examples. The conversation hour has much shorter sharing: all 12k requests share one 512-token block. Every group with $k \ge 8$ has $P(W \ge k\bar{L}) = 0$. WildChat (Zhao et al., 2024) contains 59,857 timestamped conversations and, like UltraChat, has $P(W \ge kL) = 0$ at k=16/32. Other traces describe concurrency but lack the prefix information needed to measure W. In the Azure LLM inference traces (Patel et al., 2024), every 2024 request has ≥ 16 co-arrivals within 10 s. At an assumed 50 ms per output token, 89% of decodes overlap ≥ 8 new arrivals. BurstGPT (Wang et al., 2024) API traffic has strongly correlated bursts, with interarrival coefficient of variation ≈ 98; its human conversation streams are near-Poisson. The longest common prefix between consecutive arrivals is ≈ 0 even in Mooncake's tool/agent hour, because different template families interleave. Grouping by shared prefix reveals overlap that consecutive-arrival comparisons miss. Among arrivals sharing ≥ 2048 tokens, 64.6% occur within their predecessor's recorded-output decode duration. These prefix measurements cover one provider-hour with ~ 3 s timestamp quantization. They establish that long shared prefixes occur in this workload, without estimating their frequency across providers. Scripts and per-trace results are in `polyserve/experiments/trace_analysis/` and `artifacts/traces/`.

Runtime co-residency of the 51% groups. We estimate request lifetimes as $[t_i,\ t_i + \text{input}_i/5\text{k tok/s} + \text{output}_i \cdot \text{TPOT}]$, where TPOT denotes time per output token. We evaluate both raw timestamps and uniformly jittered timestamps. These are service-model scenarios, not bounds on actual overlap. At TPOT 50 ms, 99% of members in the qualifying 51% set overlap at least one group member, and 84% overlap ≥ 4. The corresponding fractions with jittered timestamps and TPOT 20 ms are 97%/75%. In the first scenario, 56% of qualifying groups, containing 74% of members, have ≥ 8 simultaneous live requests. At each group's estimated peak overlap, additive reservation exceeds the shared footprint by a median 4.6× (4.2× with jittered timestamps). The ratio is $\ge 2\times$ in 99% of groups. Shared footprints are 10–17k tokens, compared with additive estimates of 47–54k. These values identify capacities satisfying $\text{coverage} < M < \text{additive}$ within the studied 32–64k-token budgets. Two thirds of the qualifying 51% set, or 35% of all requests, satisfy three modeled conditions: co-arrival, ≥ 8 live members, and $\ge 4\times$ peak overcount. With jittered arrivals and halved service times, the fraction passing the width-8 condition falls to 6–12%. Pairwise overlap remains $\ge 91\%$, and 79% of groups

still have $\geq 2\times$ overcount. The conversation hour has more modeled overlap, with median k_{live} of 30. Its prefix is only 512 tokens, and the median peak overcount is 1.04. Concurrency alone therefore does not imply a large memory-accounting difference.

Replaying the trace groups on a live engine. We replay selected shared-prefix groups on vLLM using an A100-40GB and Qwen3-8B. The engine's KV-block override fixes the physical pool at $1.5\,M$, prefix caching is enabled, and the common prefix is resident before every arm. The logical admission budget is M, studied at 32–64k tokens. We synthesize token IDs deterministically from each 512-token hash block, preserving the trace's prefix structure and recorded input and output lengths. For each budget $M \in \{32{,}768,\ 65{,}536\}$, we select 8 groups uniformly from those whose shared footprint fits $0.85\,M$ and whose additive footprint exceeds $1.5\,M$. There are 334 and 324 eligible groups, respectively. Each policy takes the longest arrival-order prefix of unfinished requests that fits its memory estimate. The additive policy charges $\text{input}_i + \text{out}_i$ per request, using the realized output length out_i. The shared-memory policy counts the common prefix once. Both receive realized lengths; the additive reservation is therefore no larger than a valid `max_new_tokens` reservation. Table A3 reports 3 repetitions per arm, with completion timestamps taken in the engine-step loop. The eight selected groups at each budget are the independent workload units; repetitions measure execution variation within each group. The replay submits each group together and does not reproduce the original inter-group arrival stream or semantic generation task. The additive policy needs 3–8 waves at 32k and 2–6 at 64k, as its memory estimate predicts. The shared-memory policy runs one wave, and native vLLM v1 finishes within about 4% of it (the largest absolute deviation is 4.03%). Reducing the physical pool from $1.5M$ to exactly M retains ratios of $2.21\times$ for makespan and $1.60\times$ for mean completion time. The gain therefore does not depend on the extra physical capacity (`mooncake_group_replay_vllm_pool1.json`). These groups emit 3 tokens at the median (at most 52), so suffix prefill is a substantial part of each wave. Scaling the same groups' outputs $\times 8$ (capped at 2048) makes that tail the majority of wall time and raises add/cov makespan to $3.31\times$/$2.06\times$ at 32k/64k (Table A3). Script and artifacts: `run_mooncake_group_replay.py`, `mooncake_group_replay_vllm.json`, the output-scaled replay JSONs, and group manifest `artifacts/traces/mooncake_group_replay_manifest.json`.

Geometric scheduling on the same prefix groups. A further replay uses eight uniformly sampled qualifying groups at $M = 65{,}536$ and compares coverage GSA with static coverage. GSA is slower on all eight groups: the geometric-mean makespan ratio is 1.770 (range 1.512–2.094), and its mean-completion ratio is 1.366 (range 1.230–1.619). Static coverage uses the realized output lengths and completes each group in one wave; GSA incurs seven or eight waves of geometric attempts. This oracle comparison isolates the cost of discovering completion through retries, and does not compare two policies with equal length information. It also explains why a static coverage-gate benefit should not be read as an observed GSA benefit. Results are in `mgr_fu_uniform_gsa_M65536.json`.

Table A3: Mooncake tool/agent groups replayed on vLLM (A100-40GB, Qwen3-8B, fixed KV-block pool, prefix caching on both arms; 8 groups per budget, 3 seeds; ratios are add/cov of seed means, geometric mean over groups with the per-group range). Additive wave counts equal the footprint prediction in every group; coverage runs one wave throughout; the engine-default arm is within 0.96–1.04 of coverage on makespan in every group. The $\times 8$ rows keep the trace prefixes and scale each request's realized output (cap 2048).

M (tok)	k	Waves add/cov	Makespan add/cov	Mean CT add/cov	Arrival TTFT add/cov
32,768	18–32	3–8 / 1	2.22× (1.49–2.89)	1.66× (1.01–2.05)	1.97× (1.06–2.68)
65,536	18–46	2–6 / 1	1.49× (1.07–1.88)	1.11× (0.88–1.40)	1.16× (0.87–1.63)
32,768, out ×8	18–32	3–8 / 1	3.31× (2.11–4.84)	3.80× (1.61–4.84)	11.9× (4.44–22.4)
65,536, out ×8	18–46	2–6 / 1	2.06× (1.16–3.53)	2.32× (1.79–3.05)	4.68× (2.52–6.72)

G.1 Continuous arrivals with a public output cap

We compare additive and coverage admission under FIFO in a CPU service model using the first five minutes of each public Mooncake trace (Qin et al., 2025a). The windows retain all 1,788 tool/agent requests and all 918 conversation requests, with their published arrival timestamps and ordering. We do not select requests by prefix overlap or length. We impose the same public output cap of 512 tokens on every request, truncating 195 recorded outputs in each window. Admission sees the public cap, not the future output length. The simulator stops each request at the smaller of its recorded output length and the cap. Both policies use the same token pool and strict FIFO order. Additive admission reserves each active request's full input and output cap. Coverage admission counts shared prompt blocks once, then adds private prompt tails and the same output caps.

Table A4: Mooncake tool/agent hour: co-arriving trie shared prefix groups in 10 s windows (a group = the k requests sharing a maximal hashed prefix W; $\bar{L}$ = the group's real mean output length, $p95$-truncated). The conversation hour, by contrast, has $P(W \geq k\bar{L}) = 0$ in every stratum with $k \geq 8$.

k stratum	n groups	W $p50/p90$ (tok)	$P(W \geq k\bar{L})$	$P(W \geq 64k)$	$P(W \geq 256k)$
2–7	123	2560/2560	1.000	1.000	1.000
8–15	242	2560/6144	0.988	0.988	0.545
16–31	458	6144/6144	0.587	0.587	0.279
≥ 32	249	512/6144	0.349	0.349	0.000

Trace	Pool (tokens)	Additive (s)	Coverage (s)	Reduction
Tool/agent	131,072	776.955	742.355	4.45%
Tool/agent	262,144	274.732	258.256	6.00%
Conversation	131,072	777.932	760.180	2.28%
Conversation	262,144	276.977	265.885	4.00%

Table A5: Mean response time in simulated seconds for complete five-minute arrival windows at 25 ms per output token. Reduction is $100(1 - \text{Coverage}/\text{Additive})\%$. Each row compares the same requests, pool capacity, and public output cap. All requests complete without drops or modeled capacity violations.

Only complete 512-token blocks with matching prefix hashes are shared; partial blocks and input tails without hashes remain private. The pool starts empty, and blocks without active users are released on completion. Each admitted request independently prefills at 5,000 input tokens/s and then emits one token every 25 ms. This idealized parallel service model has no slowdown from a larger batch. The public traces contain no original prompts or token sequences, so this experiment does not measure model-generated natural EOS or task quality. After five minutes of arrivals, we continue service until the queue is empty. Mean response time includes every request's time from arrival to completion, including requests still queued when arrivals stop.

Sensitivity. Coverage reduces mean response time by 2.28–6.00% in these four configurations (Table A5). At 50 ms per output token, with the same prefill rate and all other settings fixed, the additive-to-coverage mean response ratios are 1.0378, 1.0620, 1.0242, and 1.0436, in table order. The large absolute delays reflect queue buildup under the specified service rates and capacities, rather than measured production latency. The two arrival windows are the workload units in this deterministic comparison. Requests sharing the same queue are not independent experimental replicates.

H Exposing cached prefixes to TGI admission

H.1 Publishing cached prefixes when prefill finishes

Stock TGI (commit `b4adbf2`) populates the radix trie only at request *completion* (`radix.rs`'s `free()`; admission in `queue.rs`). The initial admission pass sees `prefix_len` $= 0$ for all k simultaneous requests and charges each full prefix. Under tight capacity, this limits concurrency even though the cache could store their common prefix once. A 97-line patch publishes each request's prefix to the radix trie when prefill finishes, once those KV blocks are ready. The scheduler and budget formulas are unchanged. After the first request's prefill, later batch-building passes see the prefix as cached and charge the other requests only for suffix + L. The initial pass is unchanged, so both arms initially admit the same requests.

We submit k=16 requests together, sharing a $\sim$1.8k-token prefix and carrying distinct private suffixes. The output length is L=64 with greedy decoding. Both arms use Qwen3-8B on an otherwise-idle A100-40GB, TGI commit `b4adbf2`, FlashInfer attention, prefix caching, and identical launcher flags. Only the router binary differs: stock or prefill-insert, built from the same tree and profile. We run 3 seeds. For Flash-Attention models, TGI's v3 router infers capacity from free VRAM and ignores `--max-batch-total-tokens`. We set the constrained budget through `--cuda-memory-fraction`. The recorded inferred KV capacity is 3314 tokens in every run. Warmup requires capacity above `max-total-tokens` plus a margin. The experiment therefore uses the inferred capacity, which holds the prefix and $\sim$13 private request footprints. It does not use $M = W + L_{\max}$, which would be $\approx$ 1890 tokens here.

Table A6: Two-arm TGI experiment (A100-40GB, TGI `b4adbf2`, Qwen3-8B, flashinfer + prefix caching both arms, 3 seeds; k=16 co-arriving sharers, shared prefix $\sim$1.8k tokens, L=64; inferred KV capacity 3314 tokens in all runs). Stock inserts into the radix trie only at request completion; the patched arm inserts at prefill completion. A loose-capacity control (engine-inferred M=138042) has identical wave structure across arms and a residual 1.12× gap (4.01 vs 3.59 s) from prefill-*compute* reuse: early insertion lets later sharers skip recomputing the shared prefix's KV. The constrained ratios measure the combined intervention. The $3 \rightarrow 2$ wave change is observed, but dividing the constrained speedup by the loose-capacity speedup does not independently identify an admission-only effect.

	stock (completion-time insert)	prefill-insert	Ratio
Makespan	9.59 ± 0.14 s	6.68 ± 0.08 s	1.43×
Mean-CT	6.87 ± 0.10 s	4.13 ± 0.05 s	1.66×
Wave structure (seed 1)	$1 + 12 + 3$ (three waves)	$13 + 3$ (two waves)	—

Per-request completion times show the changed batches directly (Table A6). Stock TGI initially runs one request; after it finishes and publishes its prefix, the next 12 run, followed by the remaining requests. Prefill-insert allows 13 concurrent requests after the first prefill, eliminating one completion wave. Greedy outputs match byte-for-byte across arms on $48/48$ constructed prompt–seed pairs. Those random-word prompts produce near-repetitive outputs, so we also use a coherent natural-language prefix and 16 distinct questions. Each run produces 16 distinct answers. Stock TGI completes in two waves, 2+14; prefill-insert completes in one, with 1.95× makespan and 1.84× mean-completion speedups. Outputs match on $45/48$ pairs. In each differing pair, one arm also differs from the common batch-size-1 output, consistent with batch-dependent floating-point effects (`tgi_two_arm_nl_equivalence_a100.json`). The observed $3/2$ wave-count change is consistent with the measured makespan change at this capacity. Artifacts: `tgi_two_arm_residual_admission_a100.json` (all runs, per-request traces, output-equivalence check, loose-capacity control) and the prefill-insert patch, pinned to `b4adbf2`.

Effect of spacing out arrivals. We next vary how much the same requests overlap in time. We spread the 16 natural-language requests over different arrival spans. The model, output length, A100, and 3314-token capacity match Appendix H.1. Each span uses 3 seeds and a cold server restart for each configuration. The arms differ only in the router binary; the sweep varies one number, the span over which the 16 arrivals are spread. The makespan speedup falls from 1.93× for simultaneous arrivals to 1.17×, 1.10×, and 1.03× at spans of 2 s, 5 s, and 20 s. The mean-flow speedup falls from 1.82× to 1.11×. All 24 configurations complete $16/16$ requests with 16 distinct outputs (`tgi_online_span_sweep_a100.json`). The simultaneous control gives 1.93×/1.82×, close to the earlier 1.95×/1.84× natural-language measurements. At a 20 s arrival span, mean flow is only 3.8–4.2 s. Requests overlap less, leaving little contention for the admission change to resolve. At low contention, changing when a prefix becomes visible has little effect on makespan.

I Additional discussion

I.1 Prefix-length regimes

For $k = 32$, $L = 64$ and $M = W + kL$, the additive peak width $\lfloor M/(W + L) \rfloor$ is $16, 4, 2, 1$ at $W = 64, 512, 1024, 2048$. The corresponding best additive flow ratios are $1.5, 4.5, 8.5, 16.5$. Thus a short shared prefix need not cause a large admission loss even when requests arrive together.

I.2 Extended related work

Feng et al. (2026a) extend additive scheduling guarantees to arbitrary prompt lengths, makespan, and online arrivals. Their router assigns jobs to sub-schedulers that share the memory budget over time. Because it uses additive memory, Corollary 10 applies when it counts a shared prefix separately per request. After subtracting the prefix, the same result supplies the guarantee for arbitrary private suffixes in Theorem 9(b). Whether the volume-based analysis of Kong et al. (2026) survives the residual reduction as GSA's does is open.

Prefix-aware systems already couple memory management and scheduling. BatchLLM identifies shared-prefix groups and uses memory-centric batching (Zheng et al., 2026); Preble keeps sharing and residency metadata above individual engines (Srivatsa et al., 2025); PRISM co-designs retention and admission (Qu et al., 2026). These systems are outside the length-and-age-only observation class whenever they use that extra information. Our competitive results concern the stated shared-memory model, rather than priority over the general idea

Table A7: How this paper differs from related systems and additive theory. The systems papers use sharing metadata beyond prompt lengths and decode ages.

Work	Objective	Memory	Shared prefix	Type
GSA / Jaillet / regime routing (Feng et al., 2026b; Jaillet et al., 2025; Feng et al., 2026a)	flow / latency	additive	no	theory
Dexter (Dexter et al., 2025)	TTFT + reuse	reuse-aware	yes (online)	theory+sys
Wu et al. (Wu et al., 2026)	hit rate + routing	eviction	yes (multi-LLM)	theory+sys
ChunkAttn (Ye et al., 2024)	kernel latency	exact dedup	yes	systems
Preble (Srivatsa et al., 2025)	dist. schedule	prefix share	yes	systems
BatchLLM (Zheng et al., 2026)	batch throughput	KV capacity	yes	systems
KVFlow (Pan et al., 2025b)	agentic eviction	tree reuse	yes	systems
CAKE / SCOPE (Qin et al., 2025b; Wu et al., 2025)	compress	lossy	intra-req	systems
This paper	flow competitiveness	**cov. vs add.**	yes (stars)	theory+systems

of prefix-aware admission. Eviction, compression and distributed placement can change whether a prefix is resident; composing their guarantees with the fixed-residual scheduling result remains open.

I.3 Multi-tenant isolation: details

Trusted sharing groups are tenants of one application sharing its system prompt, branches of one request, or documents one principal is authorized to read. Sharing a prefix cache across trust boundaries can expose whether a prefix is already resident through cache-hit TTFT (Chu et al., 2025). Admission decisions based on shared-prefix structure can reveal additional information through the number of requests admitted together. A deployment can restrict sharing to requests within the same tenant or namespace. This presents the scheduler with separate trusted prefix groups. Under a fixed per-group budget, the reduction in Proposition 8 applies within each group. Cache isolation remains necessary wherever sharing would otherwise cross a trust boundary.

J Native execution of four-branch voting

We compare two stopping rules for four-branch voting under native first-come, first-served (FCFS) scheduling in vLLM (Kwon et al., 2023). One waits for all branches. The other returns once the unique plurality cannot change, then cancels unfinished branches through the native abort API.

Task and configuration. We use 52 multiple-choice questions from 26 QuALITY articles (Pang et al., 2022), with two questions per article. Each question launches four reasoning branches whose final answers are combined by voting. For both stopping rules above, all 208 branches are submitted at the common application arrival. Each branch uses Qwen3-14B (Yang et al., 2025) in thinking mode, with temperature 0.6, top-p 0.95, top-k 20, and min-p 0. Generation stops at a natural end-of-sequence token or a public limit of 8192 output tokens. We run vLLM 0.30.0 with FCFS scheduling on one A100 40 GB GPU using BF16 weights. GPU memory utilization is 0.9, and the context limit is 32768 tokens. The scheduler allows at most 64 sequences and 4096 tokens per step. Prefix caching, chunked prefill, asynchronous scheduling, and CUDA graphs remain enabled. All four configurations have the same profiled physical KV pool of 2925 pages, each holding 16 token positions; one page is reserved by the allocator. The native admission watermarks 0 and 0.025 provide 0 and 73 pages of admission headroom, respectively, without changing the physical pool. Each configuration receives one complete workload warmup followed by three measurements with seed 0 and the same per-branch seed mapping. The engine is drained and the prefix cache reset between rounds. Unlike the resident-prefix replays above, each timed round includes the initial common-prefix prefill.

Exact stopping rule. Only terminal answers returned by the API contribute votes. Let c_1 and c_2 be the two largest counts of valid answer choices; missing counts are zero. Let u be the number of unfinished branches. The unique winner cannot change once

$$c_1 > c_2 + u.$$

At that point the application returns the winner and cancels its unfinished branches. An invalid terminal answer contributes no vote and is no longer unfinished. If all four branches finish without an earlier decision, we return their unique plurality, or abstain on a tie or when no valid answer is present. We register all terminal results in an API return before checking the stopping condition. An answer already returned is therefore never counted as a cancelled branch. Application completion time runs from the common arrival through the actual vote computation. Labels are used only to score the delivered answers after service.

Watermark	Completion rule	Mean (s)	Mean range (s)	P95 (s)	Last (s)	Correct
0	All four	149.526	149.501–149.526	324.203	374.925	36/52
0	Exact vote	125.254	125.096–131.035	255.210	309.304	38/52
0.025	All four	133.918	133.900–133.933	290.653	350.117	37/52
0.025	Exact vote	122.429	122.081–127.548	273.970	318.736	38/52

Table A8: Four-branch voting with native scheduling. Mean, P95, and Last are medians across three measured rounds after a complete warmup; Last is the delivery time of the final application, rather than the completion of the subsequent engine drain. Mean range gives the smallest and largest per-round application means. Correct counts are identical across the three rounds of each configuration.

Results. Exact stopping reduces median mean application time by 16.23% at watermark 0 and 8.58% at watermark 0.025 (Table A8). The corresponding P95 reductions are 21.28% and 5.74%, and final-application reductions are 17.50% and 8.96%. At watermark 0, each exact-vote round has 163 natural-EOS completions and 45 cancelled branches; at 0.025, these counts are 164 and 44. Every full-completion round has 208 natural-EOS completions, and no configuration reaches the output limit. Cancelled branches are recorded separately from natural EOS. Both exact-vote configurations answer 38 of 52 questions correctly. The respective full-completion configurations answer 36 and 37 correctly. No question answered correctly under full completion becomes incorrect in these comparisons.

These gains come from terminating a known voting application when its answer is fixed, with allocation and scheduling still performed by native vLLM. Cancellation changes the subsequent batches and can change generated answers despite fixed branch seeds, so we evaluate the answers actually delivered by each execution. The first measured exact-vote round is slower than the later two, as reflected in the reported ranges. The three repetitions quantify timing variability for the same 52 questions from 26 articles. These results motivate exact stopping as the native reference for this workload.

Starting the fourth branch only when necessary. We also test whether avoiding speculative work improves this baseline. The first three branches start at the common application arrival. The fourth keeps its original prompt and random seed, but starts only when the returned terminal answers require it. After n of the initial three branches have terminated, let c_1, c_2 denote their two largest valid-vote counts. After checking for an already fixed winner, we release the fourth branch when

$$c_1 + (3 - n) \leq c_2 + 1.$$

At this point, the leading answer cannot be guaranteed a unique plurality after the fourth vote, even if all unfinished initial branches support it. For example, two different valid answers trigger the fourth branch, whereas two identical answers wait for the third. Once released, the fourth branch must finish. The application clock still starts at the original common arrival; the native request arrival records when that branch actually becomes ready.

Watermark	Mean (s)	Mean range (s)	P95 (s)	Last (s)	Correct	Unstarted
0	137.285	137.280–137.364	324.112	369.821	37/52	45
0.025	138.332	138.329–138.534	331.524	381.474	37/52	44

Table A9: Conditional submission of the fourth branch, with the same physical pool and sampling configuration as Table A8. Times follow the same three-round convention. Unstarted counts branches never submitted; all submitted branches reach natural EOS, with no cancellation or output-limit termination.

This rule leaves 45 or 44 branches unstarted per round, but increases mean application time by 9.61% or 12.99% relative to simultaneous submission with exact stopping. P95 increases by 27.00% or 21.01%, and final-application time by 19.57% or 19.68%. Each configuration answers 37 of 52 questions correctly, losing one previously correct answer and gaining none. The finite-vote rule preserves the outcome for a fixed set of branch answers; changing when branches run can still change the generated answers. Fewer submitted branches therefore do not establish either lower latency or equal realized answer quality.