SPECGROVE: REUSABLE TREE CAPACITY FOR CONCURRENT DIFFUSION SPECULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Concurrent block-diffusion speculative decoding must share limited verification capacity across requests with heterogeneous returns from tree expansion. We introduce SpecGrove, which turns a single DFlash proposal and DDTree expansion per request into reusable, nested capacity tiers. SpecGrove scores tiers by covered draft-prefix mass and jointly selects heterogeneous tree sizes using measured packed-pass cost. An exact-row Bellman frontier optimizes the resulting finite-tier surrogate, and selected trees share one request-isolated target pass. We establish optimality for this finite-tier objective and characterize conditions under which packed verification preserves the joint autoregressive output law. Across Qwen3-4B and Qwen3-8B on four reasoning and code-generation benchmarks, SpecGrove improves complete wall-clock throughput by 13.0–16.8% over matched DFlash and 4.7–39.0% over capacity-feasible fixed-size DDTree at concurrency eight. Under Poisson arrivals, it reduces tail latency relative to matched allocation controls.

1 INTRODUCTION

Autoregressive generation requires a target pass per token, limiting throughput under concurrent demand. Speculative decoding (SD) drafts candidates cheaply and verifies them in parallel with the target (Leviathan et al., 2023; Chen et al., 2023). Tree verification covers several candidate prefixes, allowing a matching target walk to advance beyond one token (Miao et al., 2024; Cai et al., 2024; Chen et al., 2024). DFlash supplies block-wise position distributions, and DDTree expands high-mass prefixes into a tree (Chen et al., 2026; Ringel & Romano, 2026). Concurrent requests, however, must share finite verifier rows for their anchors and candidates. Tree size therefore changes both covered target prefixes and the capacity left for other requests in the same pass.

BASS batches speculative requests, TETRIS allocates capacity to promising candidates, and D-Cut and ECHO adapt speculative work to runtime conditions (Qian et al., 2024; Wu et al., 2025; Liu et al., 2026; Hu et al., 2026). For block-diffusion trees, however, one fixed node count need not serve every request well: extra rows may cover high-mass paths for one request but little for another. Every selected tree charges an anchor and candidate rows to the same pass, so giving rows to one request narrows the others' feasible choices. Large fixed trees can become infeasible as concurrency rises, while independent sizing ignores how the selected trees jointly determine verifier cost.

DDTree expands prefixes in best-first order, so cutoffs of one expansion offer candidate sizes without repeated construction. Yet these prefix-closed cutoffs still compete for a shared row cap. Maximizing draft coverage at the largest feasible packing need not maximize throughput when measured target-pass cost has plateaus or jumps. The remaining problem is to select request-level cutoffs jointly, accounting for each anchor and the cost at every reachable row count. The value of a tier thus depends on the other selected tiers: they determine whether its rows fit, which point of the packed-pass cost curve the wave reaches, and whether a smaller packing gives a higher rate.

We propose SPECGROVE to allocate these cutoffs jointly. One batched DFlash proposal and one DDTree expansion per request yield nested tiers; an exact-row Bellman frontier selects among them using draft-prefix mass, pre-sampling service weights, and measured cost. The frontier compares every reachable row total, including partially filled passes, while request-local masks isolate verification. We derive covered-prefix progress, prove finite-tier surrogate optimality, and establish the joint target law under exact same-prefix conditionals, causal decisions, and fresh draws. Experiments measure complete wall time, including draft construction and allocation.

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.

Our main contributions are summarized as follows:

- **Reusable tree capacity.** We expose nested, prefix-closed cutoffs of a single DDTree expansion as capacity choices. Their exact row charges include each request's anchor, allowing one DFlash proposal to support several feasible tree sizes without repeated expansion.
- **Exact-row allocation and analysis.** We derive expected target progress from covered-prefix probabilities and solve the finite-tier, measured-cost surrogate with an exact-row Bellman frontier. Separate results bound draft prediction error and preserve the target output law under stated kernel and sampling assumptions.
- **Matched empirical evaluation.** Across two Qwen3 models and four tasks at $C = 8$, complete-wall-time throughput improves by 13.0–16.8% over feasible DFlash ($R = 193$) and 4.7–39.0% over feasible fixed-45-node DDTree ($R = 384$). Ablations, native single-request comparisons, and BF16 quality measurements test different aspects of the method.

2 PRELIMINARIES

We review SD, block-diffusion trees, and packed verification to fix notation for SPECGROVE.

2.1 SPECULATIVE DECODING

SD drafts candidates cheaply and verifies target conditionals in parallel (Leviathan et al., 2023; Chen et al., 2023). Let x be a request's revealed context and y a continuation ending at its first EOS or a finite cap. The target defines the stopped law

$$p(y \mid x) = \prod_{t=1}^{|y|} p(y_t \mid x, y_{<t}),$$

which includes EOS when it occurs; without EOS, y has the capped length.

In single-chain SD, a drafted token z_t from draft kernel q is accepted with probability

$$\alpha_t = \min\left\{1, \frac{p(z_t \mid x, z_{<t})}{q(z_t \mid x, z_{<t})}\right\}.$$

Rejection samples from the positive residual of $p - q$, preserving the target law (Leviathan et al., 2023; Chen et al., 2023). Tree variants check several prefixes per pass; in our verifier, candidate paths select which conditionals are evaluated, while fresh target draws determine every emitted token.

2.2 BLOCK-DIFFUSION DRAFTS AND TREES

Block-diffusion drafting supplies position-wise distributions q_d in one DFlash pass (Chen et al., 2026). For a candidate prefix v, define

$$Q(v) = \prod_{d=1}^{|v|} q_d(v_d) \quad \text{and} \quad P_p(v) = \prod_{d=1}^{|v|} p(v_d \mid x, v_{<d})$$

as draft surrogate mass and target prefix probability. The q_d are position-wise rather than conditioned on the realized prefix, so $Q(v)$ need not equal $P_p(v)$. This distinction matters before verification: tier scores can use the available draft marginals, whereas actual expected progress depends on target-prefix probabilities available only after the corresponding target conditionals have been evaluated.

DDTree expands high-mass prefixes best-first under a node budget (Ringel & Romano, 2026). Because every retained prefix keeps its ancestors, cutoffs of one expansion give nested executable trees at several budgets. SPECGROVE uses them as capacity choices without rebuilding; an upgrade adds covered prefixes, whose draft mass estimates gain while their rows compete with other requests.

2.3 CONCURRENT PACKED VERIFICATION

Concurrent verification packs request-local trees into one target pass. The root anchor is an already-emitted target token whose evaluation supplies the next conditional. A tree node attends only to

its request's cached prefix, ancestors, and itself, not to other requests (Miao et al., 2024; Ringel & Romano, 2026). At a visited prefix, the verifier draws a fresh target token and follows a matching child if one exists. An off-tree draw, EOS, or the cap ends that request's walk; no rejection correction or resampling occurs on exit. Without EOS or cap truncation, a depth-D tree T has expected progress

$$\mu_p(T) = 1 + \sum_{v \in T \setminus \{\varnothing\}} P_p(v) \le D + 1. \tag{1}$$

The leading one counts a new draw; width affects coverage, not walk length.

In a concurrent wave, n admitted requests share row budget R for one target pass, while offered closed-batch concurrency is C. A full non-root tier with b nodes consumes $b + 1$ rows including its anchor. Progress depends on covered prefixes; target-stage cost depends on the packed shape.

3 SPECGROVE

SPECGROVE replaces fixed per-request tree sizes with joint allocation of reusable DDTree tiers. One DFlash proposal and one tree expansion per request supply the capacity choices; an exact-row Bellman frontier selects their sizes using draft-prefix coverage and measured packed-pass cost. The selected tiers share a request-isolated target pass, adapting capacity to concurrent demand without repeated tree construction. Algorithm 1 gives pseudocode for one allocation and verification wave.

3.1 REUSABLE TIERS AND SCORING

At a wave boundary, the allocator observes each admitted request's generated length g_i, remaining output capacity $h_i \ge 1$, waiting age a_i, KV handle, and draft marginals. An outer controller ensures that the admitted set is jointly feasible. These observations contain no upcoming target samples. Let $\mathcal{B} = \{\beta_1 < \cdots < \beta_K\}$ be the non-root budget labels. One batched DFlash pass and a canonical best-first expansion per request produce nested tiers $T_i(\beta_1) \subseteq \cdots \subseteq T_i(\beta_K)$. EOS/length handling, capacity fitting, and expansion gating leave eligible sets $A_i \subseteq \mathcal{B}$, frozen for this wave. Upgrades reuse the same draft distributions and expansion order. We distinguish a tier label from its realized shape. Let $m_i(b)$ count retained non-root nodes and $r_i(b)$ its charged rows. For an unpadded tier,

$$r_i(b) = 1 + m_i(b), \qquad m_i(b) = b \text{ when no nodes are removed.}$$

Any padding is also charged to r_i; only valid prefixes contribute utility. Reported full-tier feasibility uses $r_i(b) = b + 1$. Let $\mathcal{V}_i(b) = \{v \in T_i(b) : 1 \le |v| \le h_i - 1,\ v \text{ contains no EOS}\}$. Define

$$Q_i(v) = \prod_{d=1}^{|v|} q_{i,d}(v_d), \qquad U_i(b) = 1 + \sum_{v \in \mathcal{V}_i(b)} Q_i(v). \tag{2}$$

The utility in Equation (2) equals expected new-token progress only when the relevant draft and target prefix probabilities match. A nested upgrade adds the mass of its newly covered valid prefixes.

Path mass alone can favor requests with concentrated draft distributions. SPECGROVE also incorporates generated length and waiting age through a weight frozen before target sampling:

$$w_i = 1 + \lambda \left[\frac{\max_j g_j - g_i}{M} + \frac{a_i}{\max(1, \max_j a_j)} \right], \tag{3}$$

where $M > 0$ is a common output cap, $0 \le g_i \le M$, $a_i \ge 0$, and $\lambda \ge 0$. The terms favor lagging or waiting requests. Hence $1 \le w_i \le 1 + 2\lambda$; weights alter capacity, not sampling. Appendix A.4 bounds how this weighting changes the unweighted surrogate rate for the same finite tier set.

A positive lookup curve $C(s)$ models target-stage time at total packed rows s for each model/hardware configuration. It retains measured plateaus and jumps. Request lengths and layout can add variation beyond row count. At a fixed row total, choices share a denominator and differ by covered mass; across totals, an added tier must earn enough mass to offset its measured cost. A larger feasible packing can therefore have a lower surrogate rate. Eligibility, utilities, weights, and the curve are fixed before solving the wave. For allocation $\boldsymbol{b} = (b_1, \ldots, b_n)$, let $s(\boldsymbol{b}) = \sum_i r_i(b_i)$ and $V(\boldsymbol{b}) = \sum_i w_i U_i(b_i)$. The objective is

$$J(\boldsymbol{b}) = \frac{V(\boldsymbol{b})}{C(s(\boldsymbol{b}))}, \qquad b_i \in A_i, \quad s(\boldsymbol{b}) \le R. \tag{4}$$

For feasible allocations $\boldsymbol{a}, \boldsymbol{b}$, positivity of C makes $J(\boldsymbol{a}) \geq J(\boldsymbol{b})$ equivalent to $V(\boldsymbol{a})C(s(\boldsymbol{b})) \geq V(\boldsymbol{b})C(s(\boldsymbol{a}))$. This comparison accounts for cost jumps without a differentiable cost model. The weighted surrogate rate in Equation (4) trades draft coverage against measured target-stage cost. Throughput counts emitted tokens over complete wall time, including the draft pass, tree construction, row allocation, request packing, target execution, and token commit.

3.2 Exact Allocation

To allocate the fixed eligible tiers, we retain the best weighted utility at every reachable row count. Let $D_i(s)$ be the maximum weighted utility attainable by the first i requests using exactly s rows. Initialize $D_0(0) = 0$ and $D_0(s) = -\infty$ for $s \neq 0$; for $i \geq 1$, the Bellman frontier is

$$D_i(s) = \max_{\substack{b \in A_i \\ r_i(b) \leq s}} \{D_{i-1}(s - r_i(b)) + w_i U_i(b)\}. \tag{5}$$

An empty maximum is $-\infty$. In exact arithmetic, the allocator maximizes $D_n(s)/C(s)$ over reachable $s \leq R$ and backtracks one tier per request. Appendix A.2 proves optimality even for a nonconvex cost curve. The frontier is recomputed each wave. With $k = \max_i |A_i|$, back-pointers require $O(nRk)$ arithmetic, $O(R)$ reward memory, and $O(nR)$ pointer memory; the evaluated tuple-copying Python path has a conservative $O(n^2Rk)$ time bound because backtracked prefixes are copied.

3.3 Packed Verification

Selected trees share a target pass, with each node attending to its request's cached prefix, ancestors, and itself; request-local KV offsets isolate histories without duplicating anchors. At each visited prefix, a fresh target draw follows a matching child or stops on an off-tree draw, EOS, or the cap. Only the visited path is committed; the exit token becomes the next anchor if generation continues. Appendix C.10 checks the packed layout and its BF16 target conditionals separately.

Algorithm 1 One SpecGrove allocation and verification wave

Require: Admitted requests $\mathcal{I}$ with revealed states, tier labels $\mathcal{B}$, row cap R, cost curve C
1: Obtain one batched DFlash proposal for $\mathcal{I}$
2: **for** each request $i \in \mathcal{I}$ **do**
3: Expand one canonical prefix-closed tree; fit and gate eligible tiers A_i
4: Compute $r_i(b)$ and $U_i(b)$ for $b \in A_i$; freeze w_i
5: **end for**
6: Build the exact-row frontier $D_n(s)$ by Equation (5)
7: Choose $s^* \in \arg\max_{s \leq R} D_n(s)/C(s)$ over reachable rows; backtrack $\boldsymbol{b}^*$
8: Pack selected trees with request-local masks, positions, and KV indices
9: Run one target pass; sample and commit each visited path, then update request states

4 Theoretical Analysis

We analyze covered-prefix progress, finite-tier optimality, and the joint output law in turn. Appendix A provides the full statements, assumptions, and proofs for each result.

4.1 Expected Progress Is Covered Prefix Mass

For request i with tree depth D_i^{tree}, set $H_i = \min(h_i, D_i^{\text{tree}} + 1)$ and let $\mathcal{S}_{i,d}$ be its non-EOS depth-d prefixes (including the root). Tail summation over disjoint matching events gives (Proposition 1)

$$\mathbb{E}_p[N_i] = \sum_{d=0}^{H_i - 1} \Pr_p(N_i > d) = \sum_{d=0}^{H_i - 1} \sum_{v \in \mathcal{S}_{i,d}} P_p(v).$$

The root term is one, and at most H_i draws are possible; hence $1 \leq \mathbb{E}_p[N_i] \leq H_i$. Increasing width covers more prefixes but cannot extend a walk beyond its retained depth.

For nested eligible tiers $b < b'$, let $\mu_p(T_i(b), h_i)$ denote true expected progress under the output cap.

$$\begin{aligned} \mu_p(T_i(b'), h_i) - \mu_p(T_i(b), h_i) &= \sum_{v \in \mathcal{V}_i(b') \setminus \mathcal{V}_i(b)} P_p(v), \\ U_i(b') - U_i(b) &= \sum_{v \in \mathcal{V}_i(b') \setminus \mathcal{V}_i(b)} Q_i(v). \end{aligned} \tag{6}$$

The same added prefixes appear in both lines of Equation (6); only their probabilities differ. The root contributes the first draw, while width helps on target-reachable paths. If every target next-token kernel differs from its position-wise draft kernel by at most ϵ in total variation at every reachable prefix, coupling the first d draws bounds depth-d event error by $d\epsilon$. Thus $|\mu_p(T_i(b), h_i) - U_i(b)| \le \sum_{d=1}^{H_i - 1} d\epsilon = H_i(H_i - 1)\epsilon/2$, taking H_i for the deeper tier (Proposition 2).

4.2 Exact Allocation for The Finite Surrogate

The allocation guarantee applies to the fixed eligible tiers. Let $\mathcal{F}_R = \{\boldsymbol{b} \in \prod_i A_i : s(\boldsymbol{b}) \le R\}$ and $\mathcal{S}_R = \{s(\boldsymbol{b}) : \boldsymbol{b} \in \mathcal{F}_R\}$. For a finite nonempty feasible set and positive $C(s)$, the exact-row frontier in exact arithmetic returns

$$J^* = \max_{\boldsymbol{b} \in \mathcal{F}_R} J(\boldsymbol{b}) = \max_{s \in \mathcal{S}_R} \frac{D_n(s)}{C(s)}. \tag{7}$$

Partitioning feasible allocations by exact row count proves Equation (7): $C(s)$ is common within each part, so its largest numerator $D_n(s)$ gives that part's best ratio. Define $\Phi(\rho) = \max_{s \in \mathcal{S}_R}\{D_n(s) - \rho C(s)\}$. For any ρ, $V(\boldsymbol{b}) - \rho C(s(\boldsymbol{b})) = C(s(\boldsymbol{b}))[J(\boldsymbol{b}) - \rho]$, so $\Phi(\rho)$ changes sign at J^* (Dinkelbach, 1967). Let $c_{\min} = \min_{s \in \mathcal{S}_R} C(s)$ and $c_{\max} = \max_{s \in \mathcal{S}_R} C(s)$. For any feasible $\widehat{\boldsymbol{b}}$, the residual gives a two-sided certificate of its gap to the optimal finite-tier rate:

$$0 \le \frac{\Phi(J(\widehat{\boldsymbol{b}}))}{c_{\max}} \le J^* - J(\widehat{\boldsymbol{b}}) \le \frac{\Phi(J(\widehat{\boldsymbol{b}}))}{c_{\min}}. \tag{8}$$

At an optimizer the residual is at least the rate gap times $c_{\min}$; over all feasible allocations it is at most the gap times $c_{\max}$. Thus Equation (8) is zero exactly at a surrogate optimum. The comparison covers every reachable row count, including smaller packings favored by cost jumps. Theorem 3 bounds tier-projection loss by $L/c^{\mathcal{F}}_{\min}$; Theorem 4 bounds retained-set selection loss by $2\eta + \zeta$, where $\eta = \epsilon_V/\widehat{c}_{\min} + V_{\max}\epsilon_C/(c_{\min}\widehat{c}_{\min})$. Here ϵ_V, ϵ_C are uniform numerator and cost errors, $V_{\max}$ bounds reference utility, the two $c_{\min}$ terms bound positive costs, and ζ is estimated-solver error. Together the theorems bound the reference gap by $L/c^{\mathcal{F}}_{\min} + 2\eta + \zeta$ under these assumptions. Section 5 measures complete wall time, including costs outside the surrogate.

4.3 Conditional Output-law Preservation

For a fixed collection of requests, assume exact same-prefix target conditionals, causal decisions using no unrevealed target randomness, fresh independent target draws, and eventual completion at each request's EOS or fixed cap. Let $\mathcal{Y}_i$ contain the valid completed sequences for request i: EOS appears only as the final token, and a sequence without EOS has exactly the capped length. For every $(y_1, \ldots, y_n) \in \prod_i \mathcal{Y}_i$, coupling to independent reference streams gives

$$\Pr(Y_1 = y_1, \ldots, Y_n = y_n) = \prod_i \prod_{t=1}^{|y_i|} p_i(y_{i,t} \mid y_{i,<t}).$$

Strings outside this stopping support have probability zero. Because each scheduling choice uses only revealed state, the next selected request and its prefix are fixed before its target draw. Exact same-prefix conditionals make that fresh draw distributed as $p_i(\cdot \mid y_{i,<t})$; the chain rule multiplies these factors in reveal order, and regrouping them by request yields the displayed law (Theorem 5). An off-tree token is emitted and becomes the next anchor if generation continues. Independent target randomness across requests lets the factors be regrouped by request even when scheduling uses other requests' revealed histories. Each request has an independent target stream in the coupling; scheduling selects which stream advances next. Tree choice and interleaving change the number of draws exposed per pass, while preserving the joint law of complete stopped sequences.

5 EXPERIMENTS

We evaluate SPECGROVE on GSM8K, MATH-500, HumanEval, and MBPP with two DFlash/Qwen3 pairs (4B/8B), three baselines (AR/DFlash/DDTree), and two matched TETRIS/ECHO-style controls. We test throughput scaling, mixed workloads, Poisson serving, ablations, and BF16 quality.

5.1 EXPERIMENTAL SETUP

Datasets. We use 128 frozen prompts each from **GSM8K** (Cobbe et al., 2021), **MATH-500** (Hendrycks et al., 2021; Lightman et al., 2023), **HumanEval** (Chen et al., 2021), and sanitized **MBPP** (Austin et al., 2021); selection details are in Appendix C.1.

Target and draft models. We pair **Qwen3-4B** and **Qwen3-8B** (Yang et al., 2025) with their **DFlash-b16** drafts (Chen et al., 2026); checkpoint and tokenizer revisions are in Appendix C.1.

Baselines. AR is target-only Qwen3 decoding (Yang et al., 2025). Matched **DFlash** (Chen et al., 2026) and fixed-45-node **DDTree** (Ringel & Romano, 2026) share verifier, batching, precision, and timing boundaries with SPECGROVE; native $C = 1$ runs retain their original paths. On the shared backend, **TETRIS**-style ranks ancestor-closed upgrades by draft mass per row (Wu et al., 2025), while **ECHO**-style gates tiers by draft confidence (Hu et al., 2026). These controls isolate allocation on the same tiers and verifier; equal and greedy controls appear in Table 15.

Metrics. Throughput (tok/s) divides emitted tokens by complete wall time; /AR intervals use 2,000 paired prompt-cluster bootstrap resamples for each task–comparator pair and have pairwise coverage. Serving reports **p95 TTFT (time to first token)**, **TPOT (time per output token)**, and **E2E (end-to-end latency)**. We also report completed requests per second, p95 queue delay, and Jain's service-rate index to distinguish aggregate throughput, request queueing, and fairness.

Implementation. On one NVIDIA H20, targets use eager BF16/SDPA without CUDA graphs; FP32 proposal bookkeeping does not change target precision. SPECGROVE uses tiers $\mathcal{B} = \{11, 23, 45\}$, $\lambda = 1$, capacity fitting, and gates larger tiers above eight active requests. The closed-batch matrix uses temperature one, cap 256, seeds 17/29/43, $R = 193$, and $C \in \{1, 4, 8, 16\}$; sweeps use $R \in \{96, 384\}$ and reach $C = 32$. At $R = 193$, evaluated DFlash fits through $C = 8$ and fixed DDTree through $C = 4$; a dash denotes row-cap infeasibility. Scheduler controls use a disjoint 16-prompt calibration slice; open-loop serving uses 256 balanced requests, $R = 384$, at most 32 active requests, and Poisson arrivals at 1, 2, and 4 requests/s. Quality uses cap 1,024 and five seeds at $C = 4$; full settings, scoring, and feasibility accounting are in Appendix C.

5.2 MAIN THROUGHPUT COMPARISON

Table 1 reports complete wall-clock throughput for every primary model–task–concurrency cell at $R = 193$; paired /AR intervals are in Table 8. At $R = 384$, Table 2 compares feasible fixed-45-node DDTree; the separate BF16 quality study is reported in Section 5.6 and Appendix C.8.

At $R = 193, C = 8$, SPECGROVE exceeds matched DFlash by 13.0–16.8% across all eight model–task pairs and reaches 2.70–4.11× matched-AR throughput (Table 1). Its 24 feasible DFlash comparisons over $C \in \{1, 4, 8\}$ favor SPECGROVE by 13.0–32.1%. From $C = 8$ to $C = 16$, aggregate throughput rises by 13.0–34.0% in every pair. At $C = 16$ it reaches 1.74–3.34× AR; the evaluated fixed DFlash and DDTree shapes exceed the row cap. Minimum tiers consume 192 of 193 rows there, making this cell a test of high-concurrency feasibility under a nearly saturated shared row cap.

With $R = 384, C = 8$, eight full DDTree tiers use $8(45 + 1) = 368$ rows. Against that matched fixed-45-node configuration, SPECGROVE is faster in all eight model–task pairs: 4.7–19.1% on 4B and 10.0–39.0% on 8B (Tables 2, 11 and 12). All eight fixed trees fit with 16 rows to spare, so this cell compares adaptive sizing with executable fixed-size verification under the same timing boundary. At $R = 193, C = 4$, gains range from 0.3% to 14.4% on 8B but -4.8% to $+0.6\%$ on 4B (Table 1). Fixed DDTree leads in three of four 4B tasks at this concurrency, while SPECGROVE leads in all four 8B tasks. At $R = 193, C = 1$, SPECGROVE is within 0.6% of matched DDTree, while native DDTree is 8–11% faster: GSM8K ratios are 0.929/0.911 on 4B/8B (Appendix C.7). Isolated requests offer no cross-request allocation opportunity. Equal and greedy controls in Section 5.5 compare allocation policies on the same tiers, separating tier sizing from the row-allocation rule.

Table 1: Throughput (tok/s) at $R = 193, T = 1$ (DF/DD/SG: matched DFlash/DDTree/SPECGROVE). At $C = 8$, SG outperforms DF by 13.0–16.8%, while fixed DD exceeds the row cap.

		Qwen3-4B				Qwen3-8B			
Data	C	**AR**	**DF**	**DD**	**SG**	**AR**	**DF**	**DD**	**SG**
GSM8K	1	40.0	144.3	**182.7**	182.6	40.3	143.2	185.7	**186.0**
	4	121.4	395.9	**512.7**	488.1	124.1	399.3	452.5	**485.3**
	8	239.4	653.9	–	**762.6**	255.6	654.5	–	**753.0**
	16	461.0	–	–	**978.6**	482.4	–	–	**861.1**
MATH	1	39.7	146.9	**185.0**	184.2	42.1	149.7	**193.8**	193.4
	4	128.3	392.1	**502.7**	481.1	136.1	411.9	457.9	**505.1**
	8	249.1	625.8	–	**728.2**	262.3	612.5	–	**708.4**
	16	470.0	–	–	**963.0**	470.7	–	–	**834.9**
HumanEval	1	39.3	144.8	186.0	**186.2**	41.7	146.7	**194.3**	193.8
	4	127.9	411.7	**530.0**	519.4	135.4	420.6	452.5	**517.6**
	8	249.5	674.3	–	**762.1**	263.8	609.6	–	**711.6**
	16	462.9	–	–	**912.5**	462.9	–	–	**803.9**
MBPP	1	39.6	133.2	**159.4**	159.2	36.5	126.8	**151.0**	150.1
	4	65.2	240.3	282.6	**284.4**	62.7	237.0	278.3	**279.1**
	8	93.3	305.5	–	**357.0**	93.4	329.3	–	**383.6**
	16	147.8	–	–	**468.7**	153.9	–	–	**514.2**

Table 2: Throughput at $R = 384, C = 8$. SPECGROVE outperforms fixed DDTree by 4.7–39.0%.

	Qwen3-4B				Qwen3-8B			
Data	**AR**	**DF**	**DD**	**SG**	**AR**	**DF**	**DD**	**SG**
GSM8K	245.0	667.9	662.8	**758.7**	250.4	644.7	529.5	**735.9**
MATH	247.5	625.6	637.0	**740.3**	261.4	609.1	522.2	**710.8**
HumanEval	249.7	680.8	640.9	**763.5**	239.7	584.6	492.7	**672.5**
MBPP	92.0	304.2	351.1	**367.6**	97.8	347.7	358.9	**394.7**

Table 3: Mixed-workload throughput results for Qwen3-4B and Qwen3-8B. SPECGROVE reaches 1003.2/844.7 tok/s at $R = 384, C = 32$, while DDTree leads on 4B at $C = 4$.

		Qwen3-4B				Qwen3-8B			
R	C	**AR**	**DF**	**DD**	**SG**	**AR**	**DF**	**DD**	**SG**
193	4	102.0	347.3	**450.6**	433.5	104.8	349.4	410.3	**439.3**
193	8	198.2	559.6	–	**675.3**	203.8	556.2	–	**652.8**
193	16	379.2	–	–	**895.6**	386.2	–	–	**781.3**
384	8	198.0	558.5	605.2	**661.1**	202.1	552.4	496.5	**643.1**
384	16	383.9	780.5	–	**906.4**	383.7	682.4	–	**781.4**
384	32	610.2	–	–	**1003.2**	570.8	–	–	**844.7**

5.3 MIXED WORKLOADS AND LARGER BUDGETS

Mixed requests interleave tasks and prompt lengths, testing allocation when request states differ. We report feasible comparisons through $C = 32$; the full matrices appear in Tables 7, 11 and 12.

At $R = 384$, mixed-request throughput rises from 661.1 to 1003.2 tokens/s for Qwen3-4B and from 643.1 to 844.7 tokens/s for Qwen3-8B as concurrency increases from $C = 8$ to $C = 32$ (Table 3). At $C = 8$, the matched fixed-45-node DDTree comparison is feasible: SPECGROVE reaches 661.1 versus 605.2 tokens/s on 4B and 643.1 versus 496.5 tokens/s on 8B. At $R = 193, C = 4$, results are mixed: fixed DDTree leads on 4B (450.6 versus 433.5 tokens/s), while SPECGROVE leads on 8B (439.3 versus 410.3 tokens/s for fixed DDTree). The sign split persists at $R = 384, C = 4$: -4.1% on 4B and $+6.4\%$ on 8B relative to the same fixed-45-node DDTree baseline (Table 7).

Table 4: SPECGROVE/AR throughput ratios under different row budgets (HE: HumanEval). Increasing R from 96 to 384 changes ratios by at most 0.18 at $C \leq 8$ but supports $C = 16, 32$.

		Qwen3-4B				Qwen3-8B			
R	***C***	**GSM8K**	**MATH**	**HE**	**MBPP**	**GSM8K**	**MATH**	**HE**	**MBPP**
96	1	4.56	4.64	4.81	4.02	4.59	4.54	4.65	4.28
	4	4.04	3.77	4.11	4.35	4.01	3.71	3.87	4.55
	8	3.06	2.92	3.06	4.02	2.95	2.75	2.77	3.92
384	1	4.54	4.46	4.79	4.02	4.56	4.55	4.64	4.33
	4	4.01	3.77	4.11	4.40	3.91	3.70	3.88	4.57
	8	3.10	2.99	3.06	3.99	2.94	2.72	2.81	4.04
	16	2.14	2.07	1.97	3.26	1.83	1.78	1.76	3.38
	32	1.35	1.52	1.47	2.55	1.21	1.43	1.33	2.78

Table 5: Poisson-serving results on Qwen3-8B (p95 latencies in ms; Jain's index J). SPECGROVE lowers p95 TTFT at stable loads, while ECHO-style has higher J at 4 req/s overload.

Req/s	**Method**	**tok/s**	**done/s**	**Queue**	**TTFT**	**TPOT**	**E2E**	***J***
1	SPECGROVE	192.5	**0.978**	**37.3**	**67.4**	**8.5**	**2049**	**0.940**
	TETRIS	**193.2**	**0.978**	39.6	70.2	9.6	2222	0.931
	ECHO	192.1	**0.978**	40.6	70.0	9.8	2198	0.926
2	SPECGROVE	**370.4**	**1.874**	**52.3**	**84.0**	**11.1**	**2633**	**0.917**
	TETRIS	366.7	1.870	75.0	111.1	15.5	3795	0.860
	ECHO	365.2	1.869	70.1	110.0	16.1	3979	0.869
4	SPECGROVE	**682.3**	**3.449**	**4708.0**	**4778.0**	61.0	**16702**	0.660
	TETRIS	671.1	3.418	5707.0	5738.0	**59.8**	18746	0.649
	ECHO	669.9	3.410	5234.0	5316.0	60.0	18029	**0.713**

At $R = 384, C = 16$, the matched DFlash comparison remains feasible and favors SPECGROVE by 16.1%/14.5% (4B/8B). At $C = 32$, throughput is 1.64×/1.48× matched AR; the evaluated fixed DFlash and DDTree shapes exceed the row budget. Across the 24 model–task–concurrency cells shared by $R = 96$ and $R = 384$ at $C \leq 8$, the /AR ratios differ by at most 0.18 (Table 4). A larger cap chiefly admits $C = 16$ and $C = 32$ in this tier set, while the ratio at a fixed concurrency changes little. At $C = 32$, both SPECGROVE and AR deliver more tokens per second than at $C = 8$, but the /AR ratio falls as AR also scales. Appendix C.6 gives per-task throughput and fixed-tree comparisons at both row budgets and model scales, including low-load DDTree reversals.

5.4 MATCHED SCHEDULERS AND POISSON ARRIVALS

Matched TETRIS- and ECHO-style controls share DDTree tiers and a verifier, leaving allocation policy as the changing component. We test closed batches and matched Poisson arrivals, reporting throughput and latency for stable and overloaded Poisson traces at matched arrival rates.

SPECGROVE wins all 12 seed–capacity comparisons against each control; paired median throughput gains range from 1.7% to 9.2% against TETRIS-style and from 1.9% to 9.2% against ECHO-style allocation (Table 9). At 1 request/s, paired throughput changes are -0.3%/+0.2% versus TETRIS/ECHO-style scheduling. At the stable 2-request/s point, throughput remains similar, while median p95 TTFT across three traces is 84.0 ms for SPECGROVE versus 111.1/110.0 ms for TETRIS/ECHO-style scheduling; p95 end-to-end latency is 2633 versus 3795/3979 ms (Table 5). At the same load, p95 TPOT is 11.1 versus 15.5/16.1 ms, and p95 queue delay is 52.3 versus 75.0/70.1 ms. Across the two stable loads, the TTFT advantage over TETRIS/ECHO-style grows from 2.8/2.6 to 27.1/26.0 ms. With nearly the same completed-request rate, the lower queue delay and TTFT show a latency benefit not captured by aggregate throughput. SPECGROVE spends more time in allocation: median allocator p95 is 0.110 ms at 2 requests/s, versus 0.030/0.036 ms for the controls (Appendix C.5). At 4 requests/s, offered load exceeds the 3.41–3.45 completed requests/s; these traces probe overload. Here SPECGROVE's median Jain index is 0.660 versus ECHO-style's 0.713, recording a fairness tradeoff alongside its lower p95 latency (Table 5). Appendix C.5 reports full metrics and tail variation.

5.5 Component Ablations

On GSM8K/MATH-500 across $C \in \{1, 4, 8, 16\}$, SpecGrove exceeds fixed-11 and random-tier controls by 9.6%/8.5% and 3.6%/5.5% in geometric-mean throughput (4B/8B), with positive changes in every cell (Appendix C.9). The measured cost curve adds 1.4%/2.4% over a linear estimate; the no-weights contrast is +0.5%/-0.1%. Equal, greedy, and fixed-45 controls yield -0.9%/-0.3%, +0.0%/-0.6%, and -2.5%/+2.6% gains, respectively, with fixed-45 evaluated on four feasible cells per model. Removing the active gate changes throughput by at most 0.2% in magnitude; removing tier fitting changes it by +0.3%/-0.3% (4B/8B). Fixed-11 and random tiers distinguish variable capacity from cutoff choice. Tier sizing drives the largest measured gains; equal and greedy allocation remain close to the full policy on the same eligible tiers and shared verifier.

5.6 Quality and Numerical Scope

At $C = 4$ and cap 1,024, BF16 task-quality differences from matched AR range from -2.03 to $+1.25$ percentage points; seven of eight paired 95% intervals include zero (Appendix C.8). The 4B MBPP difference is -2.03 points, with interval $[-3.75, -0.31]$. Each task comparison uses 128 frozen prompts and five seeds, and the intervals describe pairwise score differences rather than equivalence. The throughput matrix uses cap 256 and reaches $C = 16$, a distinct regime from this quality study. Appendix C.10 reports temperature-zero sequence matches in 1,248/1,536 repeated 4B conditions and 1,155/1,536 repeated 8B conditions. Direct same-prefix BF16 posterior checks also reveal numerical differences. The exact output-law result therefore applies to the stated target-kernel premise, while these measurements characterize BF16 task quality under the separate longer-cap protocol at $C = 4$.

6 Related work

Speculative decoding and tree verification. SD verifies cheap drafts with the target (Leviathan et al., 2023; Chen et al., 2023). SpecInfer verifies trees, Medusa adds heads, EAGLE drafts features, and Sequoia optimizes tree shape (Miao et al., 2024; Cai et al., 2024; Li et al., 2024; Chen et al., 2024). These methods change candidate generation or verification within a request. We allocate nested trees across requests; fresh target draws determine the emitted tokens along each visited path.

Block-diffusion tree construction. DFlash proposes a block using target features; DDTree expands likely prefixes under a node budget (Chen et al., 2026; Ringel & Romano, 2026). DDTree's best-first order exposes the nested cutoffs that form our capacity ladder. CaDDTree selects tree shape and size with a throughput model (Zhang et al., 2026). We jointly choose DDTree cutoffs for concurrent requests, charging anchors under one row cap and using the measured cost of their shared target pass.

Shared-resource allocation. BASS batches SD; TETRIS allocates rows to promising tokens (Qian et al., 2024; Wu et al., 2025). D-Cut prunes by confidence and cost; ECHO adapts depth and width (Liu et al., 2026; Hu et al., 2026). Our allocation unit is one prefix-closed tier per request, with anchors and packed-pass cost charged jointly. The TETRIS- and ECHO-style controls in Section 5.4 share our tiers and verifier, exposing policy differences on a common backend. Orca, vLLM, and SGLang address the complementary serving problems of batching, runtime scheduling, and KV-cache management (Yu et al., 2022; Kwon et al., 2023; Zheng et al., 2024).

7 Conclusion

SpecGrove reuses one DDTree expansion as nested tiers, allocating them by measured row cost before request-isolated target verification. Covered-prefix mass explains tier utility, and the exact-row frontier solves the finite surrogate over every reachable packing size. On one H20, matched high-concurrency throughput improves by 13.0–16.8% over DFlash and 4.7–39.0% over feasible fixed-size DDTree. At stable Poisson load, p95 TTFT is lower than for matched policy controls. Adaptive sizing has the clearest ablation gain; equal and greedy allocation remain competitive. Native DDTree is faster at $C = 1$, and 4B low-load comparisons are mixed. The joint output-law theorem applies under exact same-prefix target conditionals and fresh independent draws. Observed BF16 posterior differences leave implementation-level exactness unresolved; higher-concurrency quality and broader hardware measurements remain important tests of whether the system gains generalize.

AI USE STATEMENT

Generative AI assisted manuscript drafting and editing, literature retrieval, formulation and checking of mathematical claims, proof drafting, interpretation of existing results, and LaTeX, table, and citation checks. AI-assisted scripts checked finite examples and table consistency but did not independently reproduce GPU experiments. The authors reviewed all AI-assisted work and take responsibility for the final claims, proofs, data, and citations.

ETHICS STATEMENT

Faster inference does not remove model biases, harmful outputs, or privacy risks. Numerical and task-quality differences require consideration before deployment. Throughput alone does not establish energy or carbon savings.

REPRODUCIBILITY STATEMENT

Section 5 and Appendix C specify settings and full results; Appendix A gives assumptions and proofs. We will release the implementation, evaluation scripts, checkpoint and tokenizer identifiers and revisions, prompt identifiers, calibrated cost tables, per-prompt timing and quality records, and the complete experimental protocol, subject to applicable model and dataset licenses.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *PMLR*, pp. 5209–5235, 2024.

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

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code, 2021.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable and robust speculative decoding. In *Advances in Neural Information Processing Systems 37*, pp. 129531–129563, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021.

Werner Dinkelbach. On nonlinear fractional programming. *Management Science*, 13(7):492–498, 1967.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS 2021 Datasets and Benchmarks Track*, volume 1, 2021.

Xinyi Hu, Yuhao Shen, Baolin Zhang, Hengxin Zhang, Jun Dai, Shuang Ge, Lei Chen, Yue Li, and Mingcheng Wan. ECHO: Elastic speculative decoding with sparse gating for high-concurrency scenarios, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626. ACM, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *PMLR*, pp. 19274–19286, 2023.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *PMLR*, pp. 28935–28948, 2024.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023.

Tianyu Liu, Yuhao Shen, Rui Cen, Junhan Shi, Jiebin Zhang, Guangshuo Qin, Hong Liu, Song Liu, Guanghua Yu, and Jianchen Zhu. D-Cut: Adaptive verification depth pruning for batched speculative decoding, 2026.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. SpecInfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 932–949. ACM, 2024.

Haifeng Qian, Sujan Kumar Gonugondla, Sungsoo Ha, Mingyue Shang, Sanjay Krishna Gouda, Ramesh Nallapati, Sudipta Sengupta, Xiaofei Ma, and Anoop Deoras. BASS: Batched attention-optimized speculative sampling. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 8214–8224. Association for Computational Linguistics, 2024.

Liran Ringel and Yaniv Romano. Accelerating speculative decoding with block diffusion draft trees, 2026.

Zhaoxuan Wu, Zijian Zhou, Arun Verma, Alok Prakash, Daniela Rus, and Bryan Kian Hsiang Low. TETRIS: Optimal draft token selection for batch speculative decoding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33329–33345. Association for Computational Linguistics, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report, 2025.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for transformer-based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation*, pp. 521–538. USENIX Association, 2022.

Shuai Zhang, Huachuan Qiu, Hongliang He, and Yong Dai. Cost-aware diffusion draft trees for speculative decoding, 2026.

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 *Advances in Neural Information Processing Systems 37*, pp. 62557–62583, 2024.

A THEORETICAL STATEMENTS AND PROOFS

This appendix distinguishes progress under target kernels, finite-surrogate optimization, and preservation of the output law. The prediction and numerical-error bounds require uniform assumptions and provide no direct estimate of implementation error on measured traces.

A.1 EXPECTED PROGRESS WITH EOS AND AN OUTPUT CAP

Fix a nonterminated request, a finite prefix-closed candidate tree T with distinct sibling labels and maximum depth D, and $h \geq 1$ remaining output positions. Verification samples the target kernel at the current prefix and follows the corresponding tree edge when one exists. The wave stops after an off-tree token, EOS, or the output cap; the terminating sampled token is emitted. Let N count newly emitted tokens. Write $\mathcal{S}_d(T)$ for the depth-d tree prefixes containing no EOS, and $P_p(v)$ for their target probabilities, with $P_p(\varnothing) = 1$.

Proposition 1 (Exact wave utility)**.** *The expected progress is*

$$\mathbb{E}_p[N] = \sum_{d=0}^{\min(D,h-1)} \sum_{v \in \mathcal{S}_d(T)} P_p(v), \qquad 1 \leq \mathbb{E}_p[N] \leq \min(h, D+1). \tag{9}$$

If no tree path contains EOS and $h \geq D+1$, this expression reduces to Equation (1)*.*

Proof. The tail-sum identity gives $\mathbb{E}[N] = \sum_{k=1}^{h} \Pr(N \geq k)$. Emitting the kth token requires the first $k-1$ samples to follow a non-EOS path in T. Sibling labels are distinct, so different prefixes at depth $k-1$ define disjoint events. Consequently,

$$\Pr(N \geq k) = \sum_{v \in \mathcal{S}_{k-1}(T)} P_p(v)$$

for $k - 1 \leq D$, and is zero otherwise. Substitution proves the identity. The depth-zero sum is one and each depth sum is at most one; with no clipping or EOS, every node contributes. □

Node count controls the number of covered prefixes, not the number of tokens on a realized walk. In particular, width cannot increase progress beyond $D+1$. Substituting draft masses Q for target probabilities gives the implementable utility after excluding EOS-containing paths and nodes deeper than $h-1$. Let $\mu_p(T,h)$ denote Equation (9). For another family of next-token kernels $q(\cdot \mid v)$, define $\mu_q(T,h)$ using the same tree and stopping rule. Position-wise diffusion marginals induce such a reference family by taking $q(\cdot \mid v) = q_{|v|+1}(\cdot)$; this does not assume they equal target conditionals. We use $\mathrm{TV}(p,q) = \frac{1}{2}\sum_z |p(z) - q(z)|$.

Proposition 2 (Draft-to-target utility error)**.** *Suppose $0 \leq \epsilon \leq 1$ and $\mathrm{TV}(p(\cdot \mid v), q(\cdot \mid v)) \leq \epsilon$ at every relevant nonterminal prefix. For $H = \min(h, D+1)$,*

$$|\mu_p(T,h) - \mu_q(T,h)| \leq \sum_{d=1}^{H-1} \left[1 - (1-\epsilon)^d\right] \leq \frac{H(H-1)}{2}\epsilon.$$

Proof. Couple the two traversals maximally at each next-token draw, sending either process to an absorbing stop state on tree exit or EOS. While their nonterminal prefixes agree, the next tokens agree with probability at least $1-\epsilon$; after a common stop, agreement persists. The stopped processes agree through step d with probability at least $(1-\epsilon)^d$. Agreement implies the same membership in the depth-d non-EOS tree event. The difference between the two event probabilities is thus at most $1-(1-\epsilon)^d$. Summing over the nonzero depths in Equation (9) proves the first bound. The second follows from $1-(1-\epsilon)^d \leq d\epsilon$. □

A.2 FINITE-SET OPTIMALITY AND A CHECKABLE CERTIFICATE

Let $\mathcal{B} = \{\beta_1 < \cdots < \beta_K\}$ be a finite ordered budget set. Each admitted request i has a nonempty eligible set $A_i \subseteq \mathcal{B}$ after clipping, gating, and capacity fitting. Define

$$\mathcal{F}_R = \left\{ \boldsymbol{b} \in \prod_i A_i : \sum_i r_i(b_i) \leq R \right\}, \qquad \mathcal{S}_R = \{s(\boldsymbol{b}) : \boldsymbol{b} \in \mathcal{F}_R\}.$$

Assume $\mathcal{F}_R \neq \varnothing$ and $c_{\min} = \min_{\boldsymbol{b}\in\mathcal{F}_R} C(s(\boldsymbol{b})) > 0$.

Detailed proof of exact-row optimality. The costs $r_i(b)$ are the fixed positive integer row charges, including any retained padding. No choice-dependent interaction is permitted in U_i, w_i, or A_i beyond the declared total-row constraint; any such interaction would require additional Bellman state. The proof uses exact arithmetic on these fixed inputs.

Proof. We prove by induction that $D_i(s)$ is the maximum weighted utility attainable by the first i requests using exactly s rows. Initialization establishes the claim for $i = 0$. An exact-s allocation for i requests selects some $b \in A_i$ for its final request and leaves $s - r_i(b)$ rows for the others. By the induction hypothesis, its utility cannot exceed the corresponding term of the Bellman recurrence. Conversely, a maximizing final tier together with an induction-achieving allocation attains that term. Finiteness ensures that each reachable maximum is attained.

At fixed s, every allocation shares $C(s)$, so the maximal numerator gives the maximal ratio; scanning all reachable $s \leq R$ attains the feasible-set optimum. □

The frontier also defines a residual certificate. Let

$$\begin{aligned}\Phi(\rho) &= \max_{\boldsymbol{b}\in\mathcal{F}_R}\{V(\boldsymbol{b}) - \rho C(s(\boldsymbol{b}))\}\\ &= \max_{s\in\mathcal{S}_R}\{D_n(s) - \rho C(s)\}.\end{aligned}$$

As the maximum of finitely many affine functions, Φ is continuous. For $\rho_2 > \rho_1$, take a maximizer $\boldsymbol{b}_2$ at ρ_2. Then

$$\Phi(\rho_2) \leq \Phi(\rho_1) - c_{\min}(\rho_2 - \rho_1) < \Phi(\rho_1).$$

Thus Φ is strictly decreasing. Writing a term as $C(s(\boldsymbol{b}))[J(\boldsymbol{b}) - \rho]$ shows that its unique zero is J^*. Set $c_{\max} = \max_{s\in\mathcal{S}_R} C(s)$. Any feasible allocation satisfies

$$0 \leq \frac{\Phi(J(\boldsymbol{b}))}{c_{\max}} \leq J^* - J(\boldsymbol{b}) \leq \frac{\Phi(J(\boldsymbol{b}))}{c_{\min}}.$$

Indeed, for an optimizer $\boldsymbol{b}^*$,

$$\Phi(J(\boldsymbol{b})) \geq C(s(\boldsymbol{b}^*))[J^* - J(\boldsymbol{b})] \geq c_{\min}[J^* - J(\boldsymbol{b})].$$

For the other inequality, every feasible $\boldsymbol{a}$ has $C(s(\boldsymbol{a}))[J(\boldsymbol{a}) - J(\boldsymbol{b})] \leq c_{\max}[J^* - J(\boldsymbol{b})]$: negative terms are harmless, and nonnegative terms satisfy both bounds factorwise. Maximizing over $\boldsymbol{a}$ proves the lower gap bound. This also proves that zero residual is equivalent to optimality. The reported traces contain selected allocations and row counts. Applying this certificate to a wave additionally requires its input utilities, costs, complete frontier, and back-pointers. The mathematical certificate should not be confused with a claim that these full records were collected for every reported wave.

A.3 Tier Restriction, Prediction Error, and Overhead

Theorem 3 (Tier restriction)**.** *Let $\mathcal{F}$ be a finite nonempty action set with $V(a) \geq 0$ and $C(s(a)) > 0$. Define $J(a) = V(a)/C(s(a))$, $J^*_{\mathcal{H}} = \max_{a\in\mathcal{H}} J(a)$ for nonempty $\mathcal{H} \subseteq \mathcal{F}$, and $c^{\mathcal{F}}_{\min} = \min_{a\in\mathcal{F}} C(s(a))$. Suppose $\varnothing \neq \mathcal{G} \subseteq \mathcal{F}$ and a map $\mathcal{P} : \mathcal{F} \to \mathcal{G}$ satisfies, for some $L \geq 0$ and every $a \in \mathcal{F}$,*

$$C(s(\mathcal{P}a)) \leq C(s(a)), \qquad V(\mathcal{P}a) \geq \max\{0, V(a) - L\}.$$

Then

$$0 \leq J^*_{\mathcal{F}} - J^*_{\mathcal{G}} \leq L/c^{\mathcal{F}}_{\min}.$$

Proof. Set inclusion gives the lower bound. Let a^* maximize J over $\mathcal{F}$. If $V(a^*) \geq L$, the projection assumptions give

$$J^*_{\mathcal{G}} \geq J(\mathcal{P}a^*) \geq \frac{V(a^*) - L}{C(s(a^*))} \geq J^*_{\mathcal{F}} - L/c^{\mathcal{F}}_{\min}.$$

If $V(a^*) < L$, then $J^*_{\mathcal{F}} < L/c^{\mathcal{F}}_{\min}$ and $J^*_{\mathcal{G}} \geq 0$, which yields the same conclusion. □

For nested integer-budget trees, let $\widetilde{A}_i$ be a richer finite candidate set and $A_i \subseteq \widetilde{A}_i$ the retained set. Let $\mathcal{F}$ and $\mathcal{G}$ be their row-feasible product sets. Assume $A_i \cap (-\infty, b]$ is nonempty for every $b \in \widetilde{A}_i$, weights are nonnegative, utilities and row charges are nondecreasing with budget, and C is nondecreasing in total rows. Define

$$\pi_i(b) = \max\{a \in A_i : a \le b\}, \qquad L = \sum_i w_i \max_{b \in \widetilde{A}_i} [U_i(b) - U_i(\pi_i(b))].$$

Componentwise projection preserves feasibility, does not increase cost, and loses at most L weighted utility. Theorem 3 therefore applies. The monotonic-cost assumption is sufficient for this particular projection, not required by the Bellman optimizer itself. A numerical tier-loss certificate would require evaluating L for the deployed eligible sets.

Theorem 4 (Uniform prediction error)**.** *Let $F = V/C$ and $\widehat{F} = \widehat{V}/\widehat{C}$ be reference and estimated ratios on a common finite nonempty set $\mathcal{A}$. Suppose $\epsilon_V, \epsilon_C \ge 0$, $0 \le V \le V_{\max}$, $C \ge c_{\min} > 0$, $\widehat{C} \ge \widehat{c}_{\min} > 0$, and $|\widehat{V} - V| \le \epsilon_V$, $|\widehat{C} - C| \le \epsilon_C$ uniformly on $\mathcal{A}$. Then*

$$|\widehat{F} - F| \le \eta = \frac{\epsilon_V}{\widehat{c}_{\min}} + \frac{V_{\max}\epsilon_C}{c_{\min}\widehat{c}_{\min}}.$$

If $\zeta \ge 0$ and $\widehat{F}(\widehat{a}) \ge \max_{a \in \mathcal{A}} \widehat{F}(a) - \zeta$, then $F(\widehat{a}) \ge F^* - 2\eta - \zeta$. If $\zeta = 0$ and $\widehat{F}(\widehat{a}) - \widehat{F}(a) > 2\eta$ for all $a \ne \widehat{a}$, then $\widehat{a}$ uniquely maximizes F.

Proof. Subtract the ratios:

$$\widehat{F} - F = \frac{\widehat{V} - V}{\widehat{C}} + \frac{V(C - \widehat{C})}{C\widehat{C}}.$$

The triangle inequality and denominator lower bounds give $|\widehat{F} - F| \le \eta$. For a reference maximizer a^*,

$$F(\widehat{a}) \ge \widehat{F}(\widehat{a}) - \eta \ge \widehat{F}(a^*) - \zeta - \eta \ge F(a^*) - 2\eta - \zeta.$$

For uniqueness, every alternative a satisfies $F(\widehat{a}) - F(a) \ge \widehat{F}(\widehat{a}) - \widehat{F}(a) - 2\eta > 0$. □

These statements bound ratios on a specified action set. Transferring them to end-to-end throughput requires the reference numerator and denominator to represent the desired output metric and full execution time, together with the stated uniform error estimates.

Combining restriction and estimation bounds. Choose $a^* \in \arg\max_{a \in \mathcal{F}} F(a)$ and $a^*_{\mathcal{G}} \in \arg\max_{a \in \mathcal{G}} F(a)$. By Theorem 3, $F^*_{\mathcal{F}} - F^*_{\mathcal{G}} \le L/c^{\mathcal{F}}_{\min}$. Uniform ratio error and ζ-optimality imply

$$F(\widehat{a}) \ge \widehat{F}(\widehat{a}) - \eta \ge \widehat{F}(a^*_{\mathcal{G}}) - \zeta - \eta \ge F^*_{\mathcal{G}} - \zeta - 2\eta.$$

Adding these inequalities gives the bound stated in Section 4.2; $\widehat{a} \in \mathcal{F}$ gives its lower bound. For an exactly solved estimated frontier, $\zeta = 0$. A positive ζ may instead account for an approximate solve. The result does not estimate any of L, η, ζ from the reported throughput tables.

The cost of an upgrade. For $V, C > 0$ and $C + \Delta C > 0$, cross-multiplication gives

$$\frac{V + \Delta V}{C + \Delta C} > \frac{V}{C} \quad \Longleftrightarrow \quad C\Delta V > V\Delta C.$$

For full execution time, let the old non-target overhead be $H_0 \ge 0$ and the new overhead $H_1 \ge 0$. With $V > 0$ and both full-time denominators positive,

$$\frac{V + \Delta V}{C + \Delta C + H_1} > \frac{V}{C + H_0} \quad \Longleftrightarrow \quad (C + H_0)\Delta V > V(\Delta C + H_1 - H_0).$$

Indeed, multiplying the positive denominators and cancelling $V(C + H_0)$ gives the right-hand inequality. The incremental overhead $H_1 - H_0$ may be negative. The special case $H_0 = 0, H_1 = H$ reduces to $H < C\Delta V/V - \Delta C$; it must not be used as a general comparison when the baseline also has overhead. Before extra overhead, $\Delta C > 0$ requires the marginal utility rate to exceed the old average rate, while $\Delta C = 0$ makes every positive utility increment beneficial. These inequalities compare idealized rates, not measured implementation latency.

A.4 Service Weights and Unweighted Progress

On a common finite nonempty action set with positive cost and nonnegative utilities, assume $0 < w_{\min} \le w_i \le w_{\max}$. Let a_w maximize the weighted ratio and a_0 maximize the unweighted rate $F_0(a) = \sum_i U_i(a_i)/C(s(a))$. Then

$$F_0(a_w) \ge \frac{w_{\min}}{w_{\max}} F_0(a_0).$$

The weight sandwich gives

$$w_{\max} F_0(a_w) \ge J_w(a_w) \ge J_w(a_0) \ge w_{\min} F_0(a_0).$$

Dividing proves the claim. Under Equation (3), the factor becomes $1/(1 + 2\lambda)$. This one-wave surrogate bound implies no queueing or fairness guarantee without an arrival model.

A.5 Joint Sequence Law and Numerical Perturbations

Theorem 5 (Causal joint sequence law)**.** *Consider finitely many requests with reference autoregressive kernels p_i, each decoded until its first EOS or a fixed finite request-local cap. Let $\mathcal{Y}_i$ be the corresponding valid stopped-sequence support: EOS may occur only as the final token, and a sequence with no EOS has the capped length. Assume:*

1. *every visited target conditional equals p_i at the same prefix;*
2. *scheduling and tree construction depend only on revealed histories and draft randomness independent of unrevealed target randomness;*
3. *target samples use fresh independent randomness across requests and emitted positions, every emitted token is a target sample, and unverified draft tokens are never emitted;*
4. *every request eventually reaches its prescribed stopping position.*

Then any such shared-budget schedule assigns zero probability outside $\prod_i \mathcal{Y}_i$ and, for every valid tuple, satisfies

$$\Pr(Y_1 = y_1, \dots, Y_n = y_n) = \prod_i \prod_{t=1}^{|y_i|} p_i(y_{i,t} \mid y_{i,<t}).$$

Equivalently, $\mathcal{L}(Y_1, \dots, Y_n) = \bigotimes_i \mathcal{L}_{p_i}(Y_i)$ for the stopped AR laws.

Proof. Construct independent reference streams on one probability space. For each request, independent uniforms $Z_{i,t}$ and inverse transforms of $p_i(\cdot \mid Y_{i,<t})$ generate its AR sequence up to EOS or the cap; EOS is absorbing. The scheduler retains revealed prefixes and a next-token cursor for each request. Causality ensures that it selects the request, tree, and budget before reading the uniform at that cursor. Exact conditional equality allows its target draw to use the same $Z_{i,t}$, producing the next reference token. If the token follows a tree edge, verification continues. If it leaves the tree or terminates the request, the wave stops, but the emitted token still agrees with the reference. Random variables for unvisited candidate nodes do not affect the output. Induction over emitted tokens shows that the shared execution reveals prefixes of the independent reference streams without altering them. Rebuilding trees and reallocating capacity change only when and how many tokens are revealed. Eventual completion reveals each reference sequence through its first EOS or cap, so every output lies in $\mathcal{Y}_i$ and has the stated stopped-sequence mass. Independence across requests makes the stopped-sequence probabilities factor into the stated joint product law. □

Theorem 6 (Posterior perturbation)**.** *For an output process of at most M tokens, suppose $0 \le \epsilon_t \le 1$ and, at every common revealed prefix at position t,*

$$\mathrm{TV}(p_t(\cdot \mid y_{<t}), \widetilde{p}_t(\cdot \mid y_{<t})) \le \epsilon_t.$$

With EOS padded by an absorbing token,

$$\mathrm{TV}(\mathcal{L}_p(Y), \mathcal{L}_{\widetilde{p}}(Y)) \le 1 - \prod_{t=1}^{M}(1 - \epsilon_t) \le \sum_{t=1}^{M} \epsilon_t.$$

For every score $f(Y) \in [0, 1]$, the difference in expected score is bounded by the same quantity.

Proof. Conditioned on agreement so far, maximal coupling makes the next samples agree with probability at least $1 - \epsilon_t$. Agreement of the entire padded sequence therefore has probability at least $\prod_t (1 - \epsilon_t)$. The coupling inequality proves the first bound, and the union bound gives the second. For a score in $[0, 1]$, its expectation differs by at most total variation. □

If an implemented conditional also depends on a packed layout or other requests, the bound must hold uniformly over those states, not only the local token prefix. Conditional mixtures then obey the same bound by convexity of total variation. The perturbation result requires bounds over all relevant common histories. Finite traces inspect only observed histories and cannot provide that certificate. Structural mask/KV checks, sampled-sequence matches, and task scores are distinct observations; none proves exact equality of target conditionals.

A.6 From Wave Progress to Measured Throughput

Assume an i.i.d. regenerative-wave model with emitted-token count N_t, full duration $T_t > 0$ almost surely, $0 < \mathbb{E}[T_t] < \infty$, and $\mathbb{E}[N_t] < \infty$. The strong law yields

$$\frac{\sum_{t=1}^{m} N_t}{\sum_{t=1}^{m} T_t} \xrightarrow{\text{a.s.}} \frac{\mathbb{E}[N_t]}{\mathbb{E}[T_t]}.$$

For two such systems $j \in \{0, 1\}$, write $\bar{T}_j = \mathbb{E}[T^{(j)}] = \bar{T}_{\text{target},j} + \bar{T}_{\text{other},j}$. Assume $\bar{T}_j > 0$, $\bar{T}_{\text{target},j} > 0$, and $\mathbb{E}[N^{(j)}] > 0$. Define

$$\alpha = \frac{\bar{T}_{\text{target},0}}{\bar{T}_0}, \qquad \gamma = \frac{\bar{T}_{\text{target},0}}{\bar{T}_{\text{target},1}},$$
$$\delta = \frac{\bar{T}_{\text{other},1} - \bar{T}_{\text{other},0}}{\bar{T}_0}, \qquad a = \frac{\mathbb{E}[N^{(1)}]}{\mathbb{E}[N^{(0)}]}.$$

The ratio of long-run token rates is

$$S = \frac{a}{1 - \alpha + \alpha/\gamma + \delta}.$$

To verify this identity, divide system 1's time decomposition by $\bar{T}_0$:

$$\frac{\bar{T}_1}{\bar{T}_0} = 1 - \alpha + \alpha/\gamma + \delta.$$

Substitution into $(\mathbb{E}[N^{(1)}]/\bar{T}_1)/(\mathbb{E}[N^{(0)}]/\bar{T}_0)$ proves the result. The empirical metric uses the same ratio-of-totals form: emitted tokens divided by complete wall time. The convergence statement requires the regenerative assumptions; evolving closed-batch states are not asserted to form i.i.d. waves. The decomposition shows how other-stage costs can offset a target-stage gain, but does not show that the per-wave surrogate maximizes measured batch throughput.

B Wave procedure and verification layout

The procedure expands each request once, freezes eligible tiers before target sampling, and commits only its sampled path. Table 6 specifies row-level visibility for the packed pass in Algorithm 1.

Algorithm 2 One SPECGROVE verification wave

Require: Admitted requests $\mathcal{I}$, revealed states, DFlash marginals, budgets $\mathcal{B}$, row cap R, cost curve C, coefficient λ

for each request $i \in \mathcal{I}$ **do**
    Build one canonical best-first tree order from the draft marginals
    Clip invalid paths; fit and gate tiers to obtain A_i
    Compute $U_i(b)$ for $b \in A_i$ and freeze w_i
end for
Compute the exact-row frontier $D_n(s)$ using Equation (5)
Choose reachable $s^* \leq R$ maximizing $D_n(s)/C(s)$
Backtrack the selected tiers $\boldsymbol{b}^*$
Pack trees with request-local ancestry, positions, and KV indices
Run one target forward pass
Ancestrally sample and commit target tokens; update request states

Table 6: Request-local visibility and commit rules for packed verification. Request-local masks and path-based commits keep packed requests isolated during target verification.

Object	Visible context	Excluded context	Verification role
Cached prefix	Request-local revealed tokens and KV state	Other requests' prefixes and KV state	Reuses the admitted request state
Selected tree node	Its cached prefix, strict tree ancestors, and itself	Siblings, descendants, and every other request	Evaluates one valid candidate conditional
Packed request block	Nodes from one selected request-local tier	Cross-request attention and unselected branches	Preserves independent request histories
Position/KV metadata	Indices derived from the same request-local path	Indices from another request or branch	Aligns logits with their intended prefixes
Commit step	Fresh target samples along the visited path	Unvisited draft tokens and hidden target draws	Emits only sampled target tokens
Off-tree outcome	The first sampled token leaving the candidate tree	Any continuation beyond that token in the same wave	Emits once and terminates the wave

C EXPERIMENTAL PROTOCOL AND COMPLETE RESULTS

This appendix gives the complete experimental protocol, throughput and serving matrices, quality comparisons, ablations, and numerical diagnostics underlying the main-text summaries.

C.1 PROTOCOL, COUNTS, AND INTERPRETATION

The frozen checkpoint pairs and revisions are:

Checkpoint	Revision
`Qwen/Qwen3-4B`	`1cfa9a7208912126459214e8b04321603b3df60c`
`z-lab/Qwen3-4B-DFlash-b16`	`b74e3a329c4d963783143b1e970d95b002be72bd`
`Qwen/Qwen3-8B`	`b968826d9c46dd6066d109eabc6255188de91218`
`z-lab/Qwen3-8B-DFlash-b16`	`9b41424b7109f9c5413454f481b09a82b85333f4`

Each target tokenizer is loaded at its target revision. Both model paths use BF16 and SDPA, with thinking mode and TF32 disabled. The frozen experiment record reports completion of every planned phase. Per model, it contains 2,208 main-throughput groups (8,448 method executions), 640 quality groups (2,560 executions), 1,096 correctness groups (4,160 executions), 1,024 native-baseline groups (6,144 executions), 4,368 budget groups (16,512 executions), 1,104 ablation groups

(14,208 executions), and 348 mixed-workload groups (1,248 executions): 10,788 groups and 53,280 executions in total. The Qwen3-8B scheduler supplement adds 108 matched closed-batch groups (324 method executions) and 27 final 256-request Poisson traces. These are protocol counts, not additional independent benchmark tasks. The main and ablation studies use temperature one, cap 256, seeds 17/29/43, and 128 frozen prompts per dataset. The quality study uses cap 1,024, concurrency four, and seeds 17/29/43/67/101. Accuracy and code pass rates are aggregated over prompt clusters. Reported confidence intervals are prompt-cluster bootstrap intervals for the specified dataset–comparator pair, not simultaneous family-wise intervals. Every throughput ratio below compares total emitted tokens per complete wall-clock time. The denominator includes online planning rather than reporting only target-forward latency. The original matrix does not separately log planning time; the scheduler supplement records it per wave and reports p95 values. A dash indicates that the specified fixed shape exceeds the row budget for that offered concurrency; it is neither a measured zero throughput nor an impossibility result for the baseline method. The tables retain the reported numeric precision. MATH denotes the MATH-500 evaluation subset throughout.

C.2 Complete Mixed-workload Matrix

Table 7: Mixed-workload throughput and gains (%) over matched DFlash/DDTree on Qwen3-4B/8B. SpecGrove scales to $C = 32$ at $R = 384$, while fixed DDTree remains faster on 4B at $C = 4$.

Model	R	C	SG tok/s	/AR (×)	vs DFlash	vs DDTree
	193	4	433.5	4.250	**+24.8**	-3.8
	193	8	675.3	3.408	**+20.7**	–
	193	16	**895.6**	**2.361**	–	–
4B	384	4	433.5	4.265	**+24.8**	-4.1
	384	8	661.1	3.339	**+18.4**	**+9.2**
	384	16	**906.4**	**2.361**	**+16.1**	–
	384	32	**1003.2**	**1.644**	–	–
	193	4	439.3	4.194	**+25.7**	**+7.1**
	193	8	652.8	3.203	**+17.4**	–
	193	16	**781.3**	**2.023**	–	–
8B	384	4	436.5	4.188	**+25.7**	**+6.4**
	384	8	643.1	3.182	**+16.4**	**+29.5**
	384	16	**781.4**	**2.037**	**+14.5**	–
	384	32	**844.7**	**1.480**	–	–

C.3 Complete Main Throughput Matrix

The 32 rows span two models, four tasks, and four concurrency levels at $R = 193$. DFlash is feasible in the 24 rows at $C \in \{1, 4, 8\}$, while 45-node DDTree is feasible at $C \in \{1, 4\}$. Thus $C = 8$ supports matched AR/DFlash comparisons, not an unmeasured DDTree shape.

Table 8: Throughput and autoregressive speedups at $R = 193$ with paired 95% bootstrap intervals. SPECGROVE outperforms DFlash in all 24 feasible cells, with mixed gains over DDTree.

Model	Data	C	tok/s	/AR [95% CI]	/DFlash	/DDTree
4B	GSM8K	1	182.6	4.566 [4.444, 4.694]	**1.265**	0.999
4B	GSM8K	4	488.1	4.019 [3.886, 4.163]	**1.233**	0.952
4B	GSM8K	8	**762.6**	**3.185** [3.071, 3.311]	**1.166**	–
4B	GSM8K	16	**978.6**	**2.123** [2.052, 2.203]	–	–
4B	MATH	1	184.2	4.638 [4.451, 4.824]	**1.253**	0.996
4B	MATH	4	481.1	3.751 [3.598, 3.929]	**1.227**	0.957
4B	MATH	8	**728.2**	**2.923** [2.829, 3.023]	**1.164**	–
4B	MATH	16	**963.0**	**2.049** [1.997, 2.100]	–	–
4B	HumanEval	1	186.2	4.734 [4.600, 4.879]	**1.286**	1.001
4B	HumanEval	4	519.4	4.061 [3.890, 4.215]	**1.262**	0.980
4B	HumanEval	8	**762.1**	**3.055** [2.924, 3.202]	**1.130**	–
4B	HumanEval	16	**912.5**	**1.971** [1.886, 2.052]	–	–
4B	MBPP	1	159.2	4.024 [3.799, 4.247]	**1.195**	0.999
4B	MBPP	4	284.4	4.364 [3.961, 4.768]	**1.184**	1.006
4B	MBPP	8	**357.0**	**3.826** [3.431, 4.199]	**1.168**	–
4B	MBPP	16	**468.7**	**3.172** [2.790, 3.550]	–	–
8B	GSM8K	1	186.0	4.614 [4.483, 4.741]	**1.299**	1.001
8B	GSM8K	4	485.3	3.910 [3.749, 4.086]	**1.215**	**1.072**
8B	GSM8K	8	**753.0**	**2.946** [2.820, 3.081]	**1.150**	–
8B	GSM8K	16	**861.1**	**1.785** [1.701, 1.870]	–	–
8B	MATH	1	193.4	4.591 [4.406, 4.769]	**1.291**	0.998
8B	MATH	4	505.1	3.712 [3.567, 3.886]	**1.226**	**1.103**
8B	MATH	8	**708.4**	**2.701** [2.626, 2.779]	**1.157**	–
8B	MATH	16	**834.9**	**1.774** [1.701, 1.847]	–	–
8B	HumanEval	1	193.8	4.643 [4.498, 4.795]	**1.321**	0.998
8B	HumanEval	4	517.6	3.822 [3.679, 3.973]	**1.230**	**1.144**
8B	HumanEval	8	**711.6**	**2.697** [2.604, 2.801]	**1.167**	–
8B	HumanEval	16	**803.9**	**1.737** [1.680, 1.796]	–	–
8B	MBPP	1	150.1	4.117 [3.861, 4.377]	**1.184**	0.994
8B	MBPP	4	279.1	4.451 [3.979, 4.969]	**1.178**	1.003
8B	MBPP	8	**383.6**	**4.109** [3.615, 4.642]	**1.165**	–
8B	MBPP	16	**514.2**	**3.342** [2.928, 3.812]	–	–

C.4 COMPLETE FIXED-CONCURRENCY SCHEDULER CONTROLS

All four original capacity settings are retained. These are matched DDTree-tier policy controls, not reproductions of the native TETRIS and ECHO systems. Throughput is median-over-seeds; paired gains are medians of per-seed ratios and need not equal ratios of the displayed medians.

Table 9: Qwen3-8B throughput with matched TETRIS-style/ECHO-style allocation controls (T/E). SPECGROVE improves paired throughput by 1.7–9.2%/1.9–9.2% over the respective controls.

R	C	SG (tok/s)	T (tok/s)	E (tok/s)	gain T/E (%)
193	8	650.7	595.7	595.9	+9.2/+9.2
193	16	772.6	751.8	751.4	+1.7/+1.9
384	16	764.8	717.6	719.7	+6.8/+6.6
384	32	826.1	809.8	810.3	+2.0/+2.1

C.5 COMPLETE OPEN-LOOP SERVING METRICS

Each entry is the median of three complete 256-request traces. The same arrival times, request seeds, prompt order, active-request cap, and request-local cache implementation are used by all three methods within a seed. Queue is admission time minus arrival time; TTFT and end-to-end latency are measured from arrival. Planning is the allocator's per-wave p95 latency.

Table 10: Complete Poisson-serving results for Qwen3-8B, with latencies in ms and Jain's service-rate index J. At 2 req/s, SPECGROVE reduces p95 TTFT to 84.0 ms versus 111.1/110.0 ms for TETRIS/ECHO-style controls, while ECHO-style has higher J at overloaded 4 req/s.

Req/s	Method	tok/s	done/s	Queue_{95}	TTFT_{95}	TPOT_{95}	E2E_{95}	J
1	SPECGROVE	192.5	**0.978**	**37.3**	**67.4**	**8.5**	**2049**	**0.940**
	TETRIS-style	**193.2**	0.978	39.6	70.2	9.6	2222	0.931
	ECHO-style	192.1	**0.978**	40.6	70.0	9.8	2198	0.926
2	SPECGROVE	**370.4**	**1.874**	**52.3**	**84.0**	**11.1**	**2633**	**0.917**
	TETRIS-style	366.7	1.870	75.0	111.1	15.5	3795	0.860
	ECHO-style	365.2	1.869	70.1	110.0	16.1	3979	0.869
4	SPECGROVE	**682.3**	**3.449**	**4708**	**4778**	61.0	**16702**	0.660
	TETRIS-style	671.1	3.418	5707	5738	**59.8**	18746	0.649
	ECHO-style	669.9	3.410	5234	5316	60.0	18029	**0.713**

Done/s is completed requests divided by finite-trace wall time, so it can differ from the nominal Poisson rate. The 4-req/s point is overloaded and its tail changes vary across traces. J uses per-request output tokens divided by end-to-end latency. Median allocator p95 time for SPECGROVE is 0.054/0.110/0.141 ms at 1/2/4 requests/s; TETRIS-style uses 0.023/0.030/0.053 ms and ECHO-style 0.027/0.036/0.054 ms. Bold numbers mark the best values within each arrival rate.

C.6 SENSITIVITY TO THE ROW BUDGET

The following sweeps retain the same task and model distinctions while varying the row cap. Budget 96 admits fewer fixed-shape configurations; budget 384 supports comparisons at larger concurrency. These are capacity sweeps within the evaluated implementation, not cross-hardware measurements.

Table 11: Throughput results and baseline ratios for Qwen3-4B under row budgets $R \in \{96, 384\}$. At $R = 384, C = 8$, SPECGROVE outperforms fixed DDTree by 4.7–19.1%, while some low-concurrency comparisons favor DDTree.

R	Data	C	tok/s	/AR	/DFlash	/DDTree
96	GSM8K	1	186.6	4.560	**1.264**	0.998
96	GSM8K	4	505.5	4.038	**1.236**	–
96	GSM8K	8	751.1	3.064	–	–
96	MATH	1	188.6	4.640	**1.255**	0.997
96	MATH	4	497.2	3.773	**1.223**	–
96	MATH	8	747.7	2.918	–	–
96	HumanEval	1	183.9	4.814	**1.287**	**1.017**
96	HumanEval	4	515.5	4.113	**1.267**	–
96	HumanEval	8	746.2	3.056	–	–
96	MBPP	1	153.5	4.016	**1.193**	0.992
96	MBPP	4	274.8	4.353	**1.179**	–
96	MBPP	8	362.0	4.016	–	–
384	GSM8K	1	186.0	4.539	**1.263**	0.998
384	GSM8K	4	502.6	4.010	**1.233**	0.958
384	GSM8K	8	758.7	3.097	**1.136**	**1.145**
384	GSM8K	16	**999.1**	**2.139**	**1.142**	–
384	GSM8K	32	**1059.8**	**1.353**	–	–
384	MATH	1	179.2	4.457	**1.203**	0.953
384	MATH	4	485.3	3.768	**1.223**	0.956
384	MATH	8	740.3	2.991	**1.183**	**1.162**
384	MATH	16	**951.0**	**2.071**	**1.143**	–
384	MATH	32	**1005.5**	**1.521**	–	–
384	HumanEval	1	181.8	4.786	**1.288**	1.000
384	HumanEval	4	515.6	4.113	**1.261**	0.978
384	HumanEval	8	763.5	3.058	**1.121**	**1.191**
384	HumanEval	16	**916.1**	**1.967**	**1.129**	–
384	HumanEval	32	**950.9**	**1.469**	–	–
384	MBPP	1	152.4	4.022	**1.193**	0.992

Table 11 (continued)

R	Data	C	tok/s	/AR	/DFlash	/DDTree
384	MBPP	4	278.5	4.398	**1.196**	**1.011**
384	MBPP	8	367.6	3.993	**1.208**	**1.047**
384	MBPP	16	**470.9**	**3.263**	**1.144**	–
384	MBPP	32	**583.2**	**2.554**	–	–

Table 12: Qwen3-8B throughput and baseline ratios at $R \in \{96, 384\}$. At $R = 384, C = 8$, SPECGROVE outperforms fixed DDTree by 10.0–39.0% and remains feasible through $C = 32$, beyond the evaluated fixed shapes.

R	Data	C	tok/s	/AR	/DFlash	/DDTree
96	GSM8K	1	192.4	4.591	**1.296**	1.000
96	GSM8K	4	518.5	4.006	**1.238**	–
96	GSM8K	8	746.7	2.946	–	–
96	MATH	1	190.0	4.536	**1.288**	0.999
96	MATH	4	499.0	3.714	**1.234**	–
96	MATH	8	714.7	2.749	–	–
96	HumanEval	1	192.3	4.650	**1.317**	1.000
96	HumanEval	4	516.0	3.872	**1.241**	–
96	HumanEval	8	712.9	2.771	–	–
96	MBPP	1	163.1	4.280	**1.188**	1.003
96	MBPP	4	306.6	4.551	**1.181**	–
96	MBPP	8	382.6	3.925	–	–
384	GSM8K	1	192.3	4.562	**1.296**	1.000
384	GSM8K	4	503.1	3.908	**1.210**	**1.090**
384	GSM8K	8	735.9	2.939	**1.141**	**1.390**
384	GSM8K	16	**866.3**	**1.827**	**1.152**	–
384	GSM8K	32	**907.9**	**1.213**	–	–
384	MATH	1	189.5	4.546	**1.286**	1.000
384	MATH	4	498.5	3.702	**1.225**	**1.096**
384	MATH	8	710.8	2.720	**1.167**	**1.361**
384	MATH	16	**838.9**	**1.783**	**1.163**	–
384	MATH	32	**880.3**	**1.431**	–	–
384	HumanEval	1	183.3	4.637	**1.313**	0.995
384	HumanEval	4	482.1	3.885	**1.233**	**1.111**
384	HumanEval	8	672.5	2.806	**1.150**	**1.365**
384	HumanEval	16	**771.2**	**1.760**	**1.165**	–
384	HumanEval	32	**792.6**	**1.327**	–	–
384	MBPP	1	162.5	4.330	**1.198**	0.998
384	MBPP	4	304.2	4.570	**1.183**	**1.016**
384	MBPP	8	394.7	4.036	**1.135**	**1.100**
384	MBPP	16	**515.2**	**3.379**	**1.153**	–
384	MBPP	32	**596.8**	**2.779**	–	–

C.7 NATIVE SINGLE-REQUEST IMPLEMENTATIONS

These comparisons preserve the official baselines' native execution paths rather than sharing SPECGROVE's verifier. They measure an implementation-level end-to-end gap at $C = 1$ and should not be conflated with the common-framework DDTree ratios in Table 8. No component-level timing decomposition is inferred from the native gap.

Table 13: Native single-request throughput on Qwen3-4B/8B at $R = 193$. SPECGROVE outperforms DFlash in all eight model–task pairs, while native DDTree remains faster in every pair.

Model	Data	SG tok/s	/native DFlash	/native DDTree
4B	GSM8K	186.5	**1.175**	0.929
4B	MATH	190.6	**1.158**	0.917
4B	HumanEval	193.5	**1.182**	0.919
4B	MBPP	163.5	**1.095**	0.914
8B	GSM8K	186.0	**1.210**	0.911
8B	MATH	188.0	**1.202**	0.923
8B	HumanEval	189.1	**1.195**	0.900
8B	MBPP	175.6	**1.078**	0.915

C.8 TASK-QUALITY RESULTS

The quality experiment compares SPECGROVE with matched BF16 AR at a longer output cap than the throughput study. The table reports paired point differences and intervals without attributing any difference to a particular numerical mechanism. The study does not cover $C > 4$.

Table 14: BF16 quality of SPECGROVE versus matched AR at $C = 4$ (paired 95% intervals for score differences). Seven of eight intervals include zero; 4B MBPP is 2.03 percentage points lower.

Model	Data	AR (%)	SG (%)	Δ pp	paired 95% CI
4B	GSM8K	90.62	91.88	+1.25	[-0.16, +2.66]
4B	MATH	70.62	71.56	+0.94	[-1.88, +3.75]
4B	HumanEval	83.91	84.22	+0.31	[-2.19, +2.81]
4B	MBPP	67.03	65.00	-2.03	[-3.75, -0.31]
8B	GSM8K	92.97	92.50	-0.47	[-2.50, +1.72]
8B	MATH	72.97	73.91	+0.94	[-1.56, +3.59]
8B	HumanEval	87.03	85.94	-1.09	[-4.06, +1.88]
8B	MBPP	70.62	70.31	-0.31	[-2.19, +1.72]

C.9 COMPLETE COMPONENT ABLATIONS

Each reported gain is SPECGROVE's geometric-mean end-to-end throughput change relative to one ablated policy on GSM8K and MATH-500 across $C \in \{1, 4, 8, 16\}$. Fixed 45-node trees fit only at $C = 1$ and $C = 4$, giving four comparable cells per model. Each ablation is a separate controlled contrast against the full policy; its gain cannot be added to the others.

Table 15: GSM8K/MATH-500 throughput ablations of SPECGROVE. Geometric-mean gains are 9.6%/8.5% over fixed-11-node trees on 4B/8B; equal/greedy gaps stay within 0.9%.

Ablated policy	**4B gain**	**wins**	**8B gain**	**wins**
fixed 11-node trees	**+9.6%**	**8/8**	**+8.5%**	**8/8**
random tier allocation	**+3.6%**	**8/8**	**+5.5%**	**8/8**
linear cost model	**+1.4%**	**7/8**	**+2.4%**	**8/8**
constant cost model	-1.3%	2/8	**+3.9%**	**6/8**
no service weights	+0.5%	**6/8**	-0.1%	3/8
no active gate	-0.0%	3/8	-0.2%	3/8
no tier fitting	+0.3%	**7/8**	-0.3%	3/8
equal allocation	-0.9%	4/8	-0.3%	1/8
greedy marginal allocation	+0.0%	**6/8**	-0.6%	1/8
fixed 45-node trees	-2.5%	0/4	**+2.6%**	2/4
age-only admission	-0.1%	4/8	-0.2%	4/8

Gain is the geometric-mean end-to-end throughput change of SPECGROVE relative to the named ablated policy across GSM8K/MATH-500 and $C \in \{1, 4, 8, 16\}$. Fixed 45-node trees are feasible in four cells per model. Wins count cells with a positive change; each row is a separate comparison.

Reading the mixed outcomes. Adaptive sizing and informed selection beat fixed-small and random-tier controls in every evaluated cell; the measured cost curve beats linear cost in 15/16. Equal and greedy allocation remain close to SPECGROVE on the same adaptive DDTree tiers. Service weights, gating, and tier fitting have mixed signs. These results support tier sizing, not a Bellman-specific gain: the action sets are small, and the surrogate omits full wall time.

C.10 STRUCTURAL AND NUMERICAL DIAGNOSTICS

The original BF16 structural audit passes checks of request-isolated ancestry masks, position metadata, and KV bookkeeping. Numerical agreement is tested separately. At temperature zero, generated sequences match AR in 1,248 of 1,536 repeated condition executions for Qwen3-4B and 1,155 of 1,536 for Qwen3-8B. These denominators count repeated condition executions, not independent prompt samples. At temperature one, direct same-prefix posterior comparisons reveal numerical differences. The experiment record consequently marks `strict_sequence_law_certified=false`. The structural checks do not establish equality between implemented packed and autoregressive conditionals. The audit gives no uniform total-variation bound over reachable histories, shows no repair, and does not identify the cause of task-quality differences. Neither the theorem nor finite quality measurements certify exact BF16 output laws.