CRITICAL POINTS FOR LLM ADMISSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Shared large language model (LLM) services must use limited capacity efficiently without repeatedly postponing certain workloads. Admission controllers can balance request value against service debt, the service owed but not yet delivered, by ranking requests with a weighted score and allocating resources in that order. However, fixed weight grids can repeat allocations while missing useful request orders in narrow weight intervals. We present TERRA-CP (Temporal Entitlement Ranking with Critical Points), an admission controller that searches queue-specific score crossings, where request rankings change. It combines explicit crossing enumeration for small queues with implicit search for large queues, constructs resource-feasible candidates, and selects the one with the smallest predicted maximum service debt. A welfare reserve constrains cumulative estimated utility relative to greedy allocations on each current queue. We prove coverage of the ranking-and-packing family under uncapped exact-arithmetic search and characterize fixed-grid omissions and finite-budget loss. Extensive experiments demonstrate that TERRA-CP reduces maximum cumulative deficit by 14.4% relative to a runtime-matched grid and by 29.0% relative to independently tuned, reserve-adapted drift-plus-penalty scheduling. In six-tenant vLLM experiments, it reduces completion deficit by 11.6% at moderate load and mean response time by 4.85% at high load relative to welfare-constrained token-counter scheduling.

1 INTRODUCTION

An LLM platform serves request streams from multiple teams or applications on shared graphics processing units (GPUs). We call each stream a workload, or a tenant in our serving experiments. When capacity is insufficient, an admission controller chooses which queued requests to start. Repeatedly favoring high-value requests can leave the same workload underserved, even while the platform processes many requests..

The administrator assigns service targets and request priorities, which we call values. The controller records target service not yet provided as service debt. Each active round adds the target amount, and recorded service pays it down, with the balance kept nonnegative. With a half-unit target, missing one round leaves a debt of 0.5. A full unit of service in the next active round clears this balance and meets the new target. Debt stays with the workload when its request changes. Figure 1 separates these accounts from the values and resource demands of current requests.

Such accounts are established tools for scheduling. Deficit scheduling carries service credit across rounds, while MaxWeight prioritizes queue backlogs (Shreedhar & Varghese, 1995; Tassiulas & Ephremides, 1990). For LLM serving, Virtual Token Counter (VTC) tracks token service, while Token Pools uses resource entitlements to guide admission (Sheng et al., 2024; Cunningham, 2026). Knowing who has been underserved, however, does not tell us which requests to serve together: serving one can leave too little capacity for another. Our question is how to find useful allocations from these competing requests with a limited search budget.

One way to generate allocations is to rank requests by a weighted combination of debt-repair priority and current value, then assign resources in that order within capacity. Small value weights ν favor debt repair, and large weights favor current value. Evaluating several weights gives the controller candidate allocations to compare.

Yet covering a numerical range does not ensure coverage of useful request orders. When scores vary linearly with ν, orders change only where two scores cross, at points determined by the current

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.

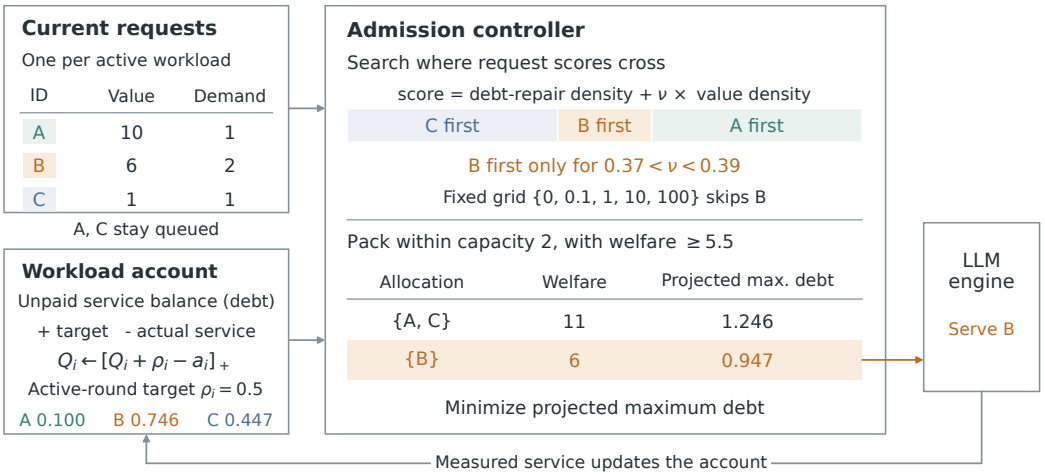


Figure 1: A full-bundle admission example with one resource, capacity 2, an active-round target of 0.5, and one service unit per served request. The illustrative 50% welfare floor admits both allocations with zero reserve. Critical-point search finds B, missed by the grid, reducing projected maximum debt from 1.246 to 0.947. Strip widths are not to scale. The greedy reference $\{A, C\}$ is always included.

queue. In Figure 1, B ranks first only for $0.37 < \nu < 0.39$. The grid $\{0, 0.1, 1, 10, 100\}$ misses this interval and admits A and C instead. Admitting B leaves smaller maximum service debt. Both allocations meet the same requirement on total estimated utility, or reported welfare. Adding weights outside B's interval repeats decisions without recovering this alternative.

In this paper, we present TERRA-CP (Temporal Entitlement Ranking with Critical Points). TERRA-CP chooses search weights using score crossings, which we call critical points. It enumerates crossings and interval representatives for small queues. For large queues, it locates search weights by counting pairwise order changes, without listing every crossing. Both branches pack retained orders within capacity and remove duplicate allocations. Our default implementation, TERRA-CP(64), retains at most 64 search weights. TERRA-CP then checks a cumulative reported-welfare requirement. A greedy allocation on each current queue defines the reference, and the controller retains a prescribed fraction of its welfare cumulatively. Welfare earned above the required level is kept in a reserve. This surplus can fund a lower-welfare allocation in a later round to serve a workload that has fallen behind. Among candidates passing this check, TERRA-CP selects the one with the smallest predicted maximum service debt.

We prove that uncapped TERRA-CP under exact arithmetic recovers every allocation produced by the chosen ranking and packing rules. For every finite grid fixed in advance, we construct a reachable state where it misses a strictly better debt-repair decision. We also characterize the tight one-step loss from limiting search and prove cumulative welfare preservation (Section 5).

We first evaluate fractional allocation, permitting partial resource bundles, to examine search quality before requiring requests to fit in full. Under a 90% welfare requirement, TERRA-CP(64) lowers maximum cumulative deficit by 14.4% over a runtime-matched grid and by 29.0% over tuned, reserve-adapted drift-plus-penalty (DPP). TERRA-CP(16) improves deficit and CPU cost over two grid controls. Implicit construction cuts allocation time by 33.5% at 1,000 workloads. We then test whole-request admission in vLLM (Kwon et al., 2023) with trace-derived arrivals. Across six windows, TERRA-CP lowers completion deficit by 11.6% at moderate load and mean response time by 4.85% at high load relative to welfare-constrained token-counter scheduling.

2 Related Work

Service accounting and value-aware allocation. Weighted sharing and dominant resource fairness (DRF) allocate capacity across competing users (Parekh & Gallager, 1992; Ghodsi et al., 2011).

Deficit scheduling carries unused service credit forward (Shreedhar & Varghese, 1995). MaxWeight schedules using queue pressure (Tassiulas & Ephremides, 1990), while virtual queues and drift-plus-penalty accommodate long-run constraints and utility objectives (Neely, 2010). We use these accounting ideas to search for allocations under a cumulative welfare constraint. Our reserve adapts the conservative reward budget of Wu et al. (2016) to admission, using a greedy allocation recomputed on the controller's current queue.

Parametric search. Parametric search resolves parameter-dependent comparisons (Megiddo, 1981), while kinetic data structures track changes in continuously varying data (Basch et al., 1999). Related changes arise in matroids with linearly varying weights (Eppstein, 1998). For affine-profit knapsack, approximation schemes cover all parameter values (Holzhauser & Krumke, 2017). We instead characterize the allocations produced by a specified ranking and packing rule, including fixed-grid omissions and the debt-repair loss from limiting this search.

LLM admission and scheduling. VTC accounts for weighted tokens within continuous batching (Sheng et al., 2024), while learning to rank (LTR) predicts output-length order for approximate shortest-job-first scheduling (Fu et al., 2024). We adapt these orders to the same nonpreemptive admission interface. Token Pools combines resource entitlements, debt, and threshold admission above vLLM (Cunningham, 2026). It rejects requests for client retry, whereas TERRA-CP retains them and selects resource-feasible sets. Engine-level systems manage batching and memory (Yu et al., 2022; Kwon et al., 2023; Agrawal et al., 2024; Zhong et al., 2024). FairBatching shares prefill/decode computation using latency slack (Lyu et al., 2025). Scorpio and ProServe coordinate admission, batching, or routing for latency and priority objectives (Tang et al., 2025; Huang et al., 2025). H-MAS adapts policies and execution parameters, while CONCUR controls agent-level concurrency using cache feedback (Liu et al., 2026; Chen et al., 2026). Online scheduling under KV-cache constraints studies latency and conditional competitive guarantees (Jaillet et al., 2025); Cascade coordinates SLO-aware scheduling with cache management (Adnan et al., 2026). TERRA-CP instead targets tenant debt at waiting admission. Its family-coverage result is not latency optimality or engine-level fairness.

3 Admission Model

We index workloads by i and admission rounds by t. A workload may have many queued or running requests. At each admission decision, the controller sees only the first not-yet-admitted request of each workload. Each active workload $i \in \mathcal{A}_t$ therefore has one eligible request, with demand $\widehat{d}_i^t$ and full-bundle value $\widehat{V}_i^t$, estimated before admission. The service account and tie-breaking key stay with the workload when the request changes.

The fractional model allocates GPU time, tokens, and bandwidth, represented by $x_i^t = (g_i^t, k_i^t, \ell_i^t)$. The serving instance instead packs integer token and sequence-slot budgets, while vLLM manages batch execution and KV memory. An allocation is $X = (x_i^t)_{i \in \mathcal{A}_t}$. For each resource r, with available capacity C_{tr}, it satisfies

$$0 \le x_{ir}^t \le \widehat{d}_{ir}^t, \qquad \sum_{i \in \mathcal{A}_t} x_{ir}^t \le C_{tr}.$$

Fractional allocation may serve part of a bundle, with service given by the least fulfilled resource fraction. Full-bundle admission reserves the complete bundle or leaves the request waiting. An order-only packer scans a request order and assigns resources within capacity, using the order rather than its numerical scores. Recorded service $a_i^t \in [0, 1]$ is the delivered resource fraction in the simulator, where each allocation step is a round. In vLLM, it credits first admission within an arrival epoch, which groups requests with the same prescribed arrival tick. Repeated callbacks within that epoch add no target. Section 6.3 defines the completion-based evaluation.

Request value and workload targets. Let $\widehat{U}_i^t(X)$ be the controller's utility estimate for request i under allocation X, and $\widehat{V}_i^t$ its full-bundle value. Reported welfare is $\widehat{W}_t(X) = \sum_{i \in \mathcal{A}_t} \widehat{U}_i^t(X)$. Full-bundle values generate priorities before packing, while allocation-specific utilities evaluate the resulting candidates. Hats denote estimates available before execution. The platform administrator

configures each workload's service target $\rho_i \in [0, 1]$, in normalized service units per active admission round, not per second. One unit means that the resource bundle is fully served. The administrator supplies trusted priorities, targets, and latency weights, and the controller receives resource estimates. The reserve is defined on reported welfare. Realized welfare sums utilities computed from execution outcomes.

Tracking unpaid service. An active round adds the service target ρ_i to workload i's debt. Recorded service pays it down, so the account evolves as

$$Q_i^{t+1} = \begin{cases} [Q_i^t + \rho_i - a_i^t]_+, & i \in \mathcal{A}_t, \\ Q_i^t, & i \notin \mathcal{A}_t. \end{cases} \tag{1}$$

Here $[z]_+ = \max\{z, 0\}$, and $a_i^t = a_i^t(X_t)$ records service from the committed allocation X_t. The nonnegative balance Q_i^t records unpaid required service, preserving later shortfalls even after earlier over-service. Inactive workloads retain their balance without accruing new debt.

Measuring long-run service. The debt account guides the next decision. To evaluate long-run service, we compare each workload's total received service with its accumulated target. Let $h_i(T) = |\{t < T : i \in \mathcal{A}_t\}|$. We measure cumulative deficit and service fulfillment by

$$D_i(T) = \rho_i h_i(T) - \sum_{t<T:i\in\mathcal{A}_t} a_i^t, \qquad \phi_i(T) = \frac{\sum_{t<T:i\in\mathcal{A}_t} a_i^t}{\rho_i h_i(T)}.$$

Fulfillment is defined only for $\rho_i h_i(T) > 0$. The cumulative deficit credits all past service, including earlier over-service, whereas reflected debt discards negative balances. With $Q_i^0 = 0$, $Q_i^T \geq [D_i(T)]_+$. For round-based evaluation, our primary metrics are the largest cumulative deficit $\max_i [D_i(T)]_+$, normalized by T where stated, and the lowest fulfillment $\min_i \phi_i(T)$.

Table 1: Core notation. Service debt guides decisions, while cumulative deficit and fulfillment measure service over a trajectory.

Symbol	Meaning
Q_i^t, ρ_i	Service debt and active-round service target
$D_i(T), \phi_i(T)$	Cumulative deficit and service fulfillment
$\widehat{V}_i^t, \widehat{U}_i^t(X)$	Full-bundle value and allocation-specific utility estimate
$\widehat{W}_t(X), B_t$	Reported welfare and cumulative welfare reserve
$\widehat{s}_i^t, \widehat{a}_i^t(X)$	Full-bundle service credit and allocation-specific prediction
a_i^t	Recorded service used to update the debt account

4 Critical-Point Admission

Let $M_t(X)$ denote the largest predicted service debt after allocation X. TERRA-CP seeks an allocation that minimizes this quantity. Let $\mathcal{H}_t$ contain allocations allowed by fractional or full-bundle service, as applicable, and satisfying resource capacities and the cumulative welfare requirement defined below. The ideal decision has objective value

$$M_t^* = \min_{X\in\mathcal{H}_t} M_t(X). \tag{2}$$

TERRA-CP generates candidates by ranking and packing requests, then minimizes $M_t(X)$ over the candidates satisfying the welfare requirement.

Why debt enters the ranking. Debt-weighted service, $\sum_i Q_i^t \widehat{a}_i^t(X)$, gives priority to serving workloads with larger unpaid balances. It arises from bounding the one-step change in the quadratic balance $\frac{1}{2}\sum_i Q_i^2$ (Neely, 2010), as derived in Appendix A.16. This motivates the debt-weighted priority used to generate candidates. The selector compares their maximum projected debt.

Searching where the request order changes. Each request's score is a line in the value weight ν: its intercept rewards debt repair, and its slope rewards current value. Normalized demand $b_i^t > 0$, defined in Equation (5), accounts for consumed capacity. The full-bundle credit $\widehat{s}_i^t$ is one for positive fractional bundles. Fractional candidates predict their allocated resource fraction. In serving, the credit is one if the tenant has not yet received an admission credit in the arrival epoch, otherwise zero. Here $\widehat{a}_i^t(X)$ multiplies that credit by the indicator that its eligible head is admitted, and recorded $a_i^t(X)$ credits only actual first admissions in the engine-accepted prefix. Generation across epochs adds no credit: this is not a completion prediction. Let $\sigma_t = \max_{j \in \mathcal{A}_t} |\widehat{V}_j^t|$. TERRA-CP ranks in descending score order by

$$f_i^t(\nu) = \alpha_i^t + \nu \beta_i^t, \quad \alpha_i^t = \frac{Q_i^t \widehat{s}_i^t}{b_i^t}, \quad \beta_i^t = \frac{\widehat{V}_i^t}{\max\{\sigma_t, 10^{-12}\} b_i^t}, \qquad \nu \geq 0. \tag{3}$$

Varying ν trades debt-repair density against value density. Under exact arithmetic, the request order can change only at pairwise crossings

$$\nu_{ij}^t = \frac{\alpha_j^t - \alpha_i^t}{\beta_i^t - \beta_j^t} \tag{4}$$

for unequal slopes. TERRA-CP uses a hybrid candidate-construction rule. Below 256 active requests, it sorts zero, all positive crossings, one representative per adjacent interval, and a tail weight, retaining at most $K \geq 2$ weights at evenly spaced indices, including both endpoints. For larger queues, it locates approximate crossing quantiles by counting order inversions, without enumerating every request pair (Appendix A.2). Uncapped TERRA-CP(∞) always enumerates the full list, including equality orders that can differ from both neighboring intervals. Both branches pack retained orders, add the greedy reference G_t, remove duplicate allocations, and apply the same reserve check and selector. The 100-workload and serving experiments exercise the small-queue branch.

Accounting for resource demand. To normalize resource demand, we combine each request's bottleneck demand with its demand across all resources. Let $\epsilon = 10^{-9}$ and $\bar{C}_{tr} = \max\{C_{tr}, \epsilon\}$. We define normalized resource demand as

$$b_i^t = \max\left\{\epsilon, \omega \max_r \frac{\widehat{d}_{ir}^t}{\bar{C}_{tr}} + (1-\omega) \sum_r \frac{\widehat{d}_{ir}^t}{\bar{C}_{tr}}\right\}, \qquad \omega = 0.5. \tag{5}$$

Demand $\widehat{d}_{ir}^t$ and capacity C_{tr} use the same resource units, making b_i^t dimensionless. The intercept measures debt-weighted normalized service, and the slope measures normalized value density. The maximum captures bottleneck demand and the sum captures total demand. Appendix A.23 varies ω. Coverage permits any fixed coefficients and a deterministic order-only packer.

Preserving cumulative welfare. Debt repair can require giving up welfare in the current round. The welfare reserve permits this trade whenever earlier decisions have earned enough surplus relative to the reference. Let G_t be the allocation obtained by ranking requests by reported-value density $\widehat{V}_i^t / b_i^t$ and applying the same packing rule as the candidates. We recompute it on the controller's current queue each round. In the fractional simulator, the reference skips nonpositive-value requests, while the score-based candidates scan every rank. For welfare floor $\gamma \in [0, 1]$, the required reference amount is

$$r_t = \mathrm{res}_\gamma(\widehat{W}_t(G_t)), \qquad \mathrm{res}_\gamma(x) = \begin{cases} \gamma x, & x \geq 0, \\ x, & x < 0, \end{cases}$$

For negative reference welfare, charging that amount itself keeps the reference admissible. Let B_t be the accumulated welfare surplus before round t. When the selected allocation is committed as a whole, it meets the welfare requirement exactly when

$$B_t + \widehat{W}_t(X) - r_t \geq 0. \tag{6}$$

After choosing X_t, we update the reserve by adding earned welfare and subtracting the reference amount, giving

$$B_{t+1} = B_t + \widehat{W}_t(X_t) - r_t, \qquad B_0 = 0. \tag{7}$$

Choosing whom to serve. For candidate X, let

$$\widehat{Q}_i^{t+1}(X) = \begin{cases} [Q_i^t + \rho_i - \widehat{a}_i^t(X)]_+, & i \in \mathcal{A}_t, \\ Q_i^t, & i \notin \mathcal{A}_t, \end{cases} \qquad M_t(X) = \max_i \widehat{Q}_i^{t+1}(X),$$

To distinguish allocations with the same maximum debt, we also measure service weighted by each workload's debt relative to its target:

$$P_t(X) = \sum_{i \in \mathcal{A}_t} \frac{Q_i^t}{\max\{\rho_i, 10^{-9}\}} \widehat{a}_i^t(X).$$

An inactive workload can set a maximum debt that no current allocation can reduce. When candidates tie, $P_t(X)$ favors service to active workloads with larger debt relative to their targets. The controller first minimizes the largest projected debt, then uses normalized repayment and reported welfare to break ties. Formally, it lexicographically maximizes

$$\Big(-M_t(X), P_t(X), \widehat{W}_t(X), \mathrm{sig}(X)\Big) \tag{8}$$

among allocations passing the reserve check. The allocation tuple $\mathrm{sig}(X)$ provides a fixed deterministic final tie-break. For $n = |\mathcal{A}_t|$, explicit event construction costs $O(n^2 \log n)$. Implicit construction costs $O((1+q)n \log n)$ for q inversion-count queries. Packing adds $O(Kn \log n)$, or $O(n^3 \log n)$ without a cap. These bounds use unit-cost arithmetic.

Nearby request orders often share packing decisions. We reuse unchanged prefixes and suffixes when the packer reaches the same state, preserving the selected allocation and its tie-breaks (Appendix A.26).

Committing requests sequentially. In vLLM execution, the engine may accept only a prefix of an admission plan. The controller checks the reserve at each prefix index against the greedy prefix at the same index, extending each sequence's cumulative welfare constantly beyond its end. Unaccepted requests remain queued. Appendix A.7 specifies the prefix guard and waiting-time value adjustment.

5 What Critical-Point Search Guarantees

We first establish which allocations critical-point search recovers and why a fixed grid can miss a better decision. We then bound the loss from limiting search and show that the welfare reserve preserves welfare across admissions.

Lemma 1 (Coverage of ranking-induced allocations)**.** *Fix capacities, reports, state, and tie keys. Let $\mathcal{P}_t$ be a deterministic packer depending on ν only through the total request order. Treat canonical coefficients as exact rationals and do not truncate. The orders at zero and every positive crossing, one per bounded open interval, and one in the tail recover every allocation obtained by ranking at any $\nu \geq 0$ and applying $\mathcal{P}_t$. For c_t distinct positive crossings and $n = |\mathcal{A}_t|$, at most $2c_t + 2 \leq n(n-1) + 2$ weights suffice. The exact-arithmetic enumeration branch of TERRA-CP(64) therefore optimizes Equation* (8) *over this family plus greedy when $c_t \leq 31$, including whenever $n \leq 8$.*

Orders are constant between crossings, with fixed tie-breaking at equality (Appendix A.30). This argument applies to any fixed affine scores and deterministic order-only packer under the lemma's exact-arithmetic and tie-breaking conditions. It covers the induced allocation family. Appendix A.18 measures the remaining gap to constrained allocation oracles. Even within this family, a grid can miss a better decision.

Proposition 1 (Fixed grids can miss better allocations)**.** *Under the round-level guard in Equation* (6)*, for every finite state-independent grid $\mathcal{G} \subset [0, \infty)$ and welfare floor $\gamma \in [0, 1]$, a realizable three-request, one-slot state with rational demands, coefficients, debts, service targets, and utilities exists in which exact-arithmetic TERRA-CP(64) selects a reserve-feasible allocation with projected maximum debt strictly lower than that of every reserve-feasible candidate from $\mathcal{G}$ plus the same greedy reference.*

The construction places a better request order in an interval missed by the grid. With only three requests, TERRA-CP(64) enumerates the complete family and finds it. Appendix A.30 constructs the state and a history reaching its debt and reserve balances.

We now compare the budgeted search TERRA-CP(K), defined in Section 4, with the uncapped search TERRA-CP(∞). Both use the same packing, reserve check, and allocation selection.

Proposition 2 (Worst-case loss from limiting the search budget)**.** *Let X_K and X_∞ be selected from the same state by the idealized exact-arithmetic TERRA-CP(K) and TERRA-CP(∞) controllers, and assume $\widehat{a}_i^t(X) \in [0,1]$ for every request and candidate. Then $0 \le M_t(X_K) - M_t(X_\infty) \le 1$. For every $K \ge 2$ and $\delta > 0$, a reachable rational round-level state attains a gap above $1-\delta$ and $M_t(X_K)/M_t(X_\infty) > 1/\delta$. Thus the additive-one bound is tight and no finite multiplicative guarantee depends only on K.*

The one-unit ceiling follows from bounded service. The tightness construction applies to either coefficient-based candidate rule, which retains zero and the tail but at most K weights (Appendix A.30).

Appendix A.29 separates truncation loss from the family gap. The welfare invariant below holds independently of either gap.

Theorem 1 (Cumulative welfare preservation)**.** *Let $\gamma \in [0,1]$, initialize $B_0 = 0$, and retain the current-queue greedy reference G_t. The round-level guard and reserve update give $B_t \ge 0$ and $\sum_{\tau<T} \widehat{W}_\tau(X_\tau) \ge \sum_{\tau<T} r_\tau$ for every temporal prefix T. Under the stronger prefix guard, the same inequality holds at every committed prefix of the checked plan, provided its matched greedy prefix is charged. For nonnegative charged reference welfare, this retains at least a γ fraction of the cumulative reported reference.*

Summing reserve updates proves the invariant (Appendix A.30). It concerns estimated welfare on each controller's visited queues, not realized utility against a shared counterfactual greedy trajectory.

6 Experiments

We ask whether critical-point search finds better allocations at matched cost, how its budget trades computation for service quality, and how the resulting controller performs in real LLM serving.

Setup and comparisons. The fractional simulator uses five resource-demand and value profiles, 100 workloads, and 1,000 rounds per trajectory, with administrator-assigned priorities and targets. Paired policies receive the same external inputs and evolve their own service accounts. Critical-point and grid methods share packing, selection, and a development-selected welfare floor of 0.90. Grid names give their weight counts. Dense136 uses zero and 135 log-spaced positive weights from 10^{-8} to 10^{8}, while Adaptive88 adjusts the range to each queue. Their budgets are chosen by development timing to match TERRA-CP(64), with 30 new paired trajectories per comparison.

We measure maximum cumulative deficit per round, $\max_i [D_i(T)]_+/T$, and minimum fulfillment, $\min_i \phi_i(T)$. A separate 12-trajectory study compares four search budgets with drift-plus-penalty (DPP) (Neely, 2010), dominant resource fairness (DRF) (Ghodsi et al., 2011), and virtual token counter (VTC) scheduling (Sheng et al., 2024). These "banked" policies enforce the same 90% welfare requirement. Serving comparisons also include learned output-length ranking (LTR). Protocols, adaptations, and paired intervals appear in Appendices A.5, A.13, and A.15.

6.1 Search quality at matched scheduling time

At essentially the same scheduling time, the least-served workload receives more of its target service: mean minimum fulfillment rises from 63.6% with Dense136 to 68.3% with TERRA-CP(64) (Appendix A.5). Maximum cumulative deficit per round falls from 0.04745 to 0.04064, a 14.4% reduction. Both metrics improve in all 30 paired trajectories. The elapsed-time ratio, TERRA-CP(64)/Dense136, is 0.993. Realized-welfare retention decreases by 0.0103 percentage points, within the prespecified 0.20-point allowed loss.

The gain also holds against the queue-adaptive grid: at matched cost, TERRA-CP(64) raises fulfillment over Adaptive88 by 2.05 percentage points and lowers deficit by 0.00280. Thus, adjusting the range of a numerical grid does not recover all of the benefit of searching at the order changes themselves.

Table 2: Search quality at matched scheduling time: means over the same 30 paired 100-workload trajectories. Both methods share packing, selection, and the 90% welfare requirement. Bold marks the best observed mean. Appendix A.5 gives paired intervals and the independent Adaptive88 cohort.

Policy	Elapsed allocation (ms/round)	Maximum deficit/round	Minimum fulfillment (%)
Dense136	45.92	0.04745	63.64
TERRA-CP(64)	**45.57**	**0.04064**	**68.29**

Large-queue construction. On a separate set of five 200-round trajectories at 1,000 workloads, implicit construction handles 317–515 eligible heads and reduces total allocation time by 33.5% relative to enumeration. Mean deficit changes from 0.06536 to 0.06564 per round and minimum fulfillment from 44.16% to 44.02%. Appendix A.2 details the construction ablation and absolute timings.

6.2 Baseline comparisons and search-budget tradeoffs

Under the same 90% welfare requirement, TERRA-CP(64) reduces mean maximum deficit by 29.0% relative to independently tuned banked DPP, 70.5% relative to banked VTC, and 74.9% relative to banked DRF (Figure 2). The reduction over DPP holds in 11 of 12 trajectories. Banked DRF and VTC each submit one proposal and fall back to greedy if it fails the reserve check, even when another debt-repair allocation is feasible. Our banked DPP adapter fixes its debt-value weight offline, whereas TERRA-CP searches queue-specific order changes each round.

DPP remains cheaper per allocation round. Even with 16 search weights, however, TERRA-CP achieves lower mean deficit and lower CPU cost than both Dense96 and Adaptive64, with paired 95% intervals excluding zero for all four differences. It retains 90.4% of TERRA-CP(64)'s deficit reduction over DPP at 48.8% less CPU time. Development selects $K = 64$ as the default. Appendix A.15 gives the complete budget curve, and Appendix A.13 reports costs and paired intervals for the published-policy comparisons.

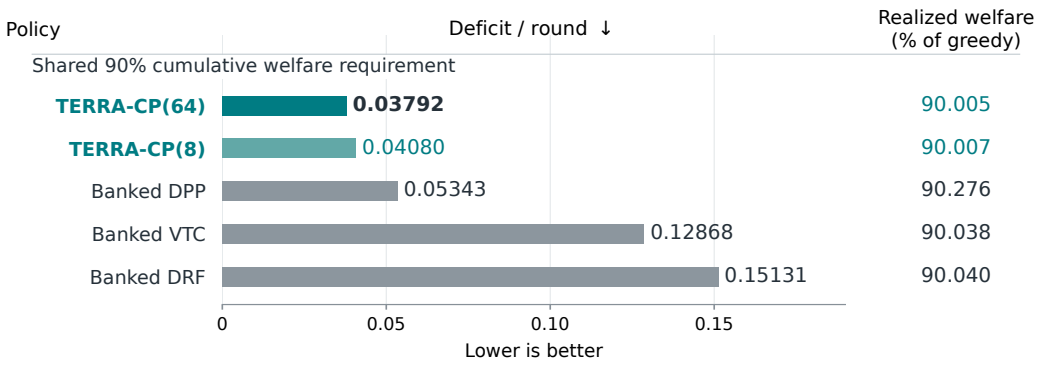


Figure 2: Service deficits on 12 paired fractional trajectories. All policies satisfy the same 90% cumulative reported-welfare requirement. Rows are ordered by mean maximum cumulative deficit per round, not by realized welfare. Teal identifies TERRA-CP and bold marks the lowest deficit. The final column reports realized welfare as a percentage of greedy.

6.3 Whole-request admission in vLLM

We implement TERRA-CP at vLLM's waiting-admission interface, with CPU admission decisions and GPU model execution. Six Azure windows each supply 216 question-answering, summarization, and code-generation requests from six tenants to Qwen2.5-7B at two loads. Dense64, guarded VTC, and guarded LTR share the 90% welfare reserve, one-head-per-tenant eligibility, and no-free-

slot early return. Dense64 isolates weight placement, while guarded VTC and guarded LTR isolate the ranking rule. The 64-weight budget covers the ideal family for these queues (Lemma 1). Response time starts at prescribed arrival. Completion deficit time-averages the largest tenant shortfall against a common external reference, crediting a request when its complete response is received.

The comparison comprises 48 fresh-server runs and 10,368 completions (Appendix A.9). Cross-model execution, sustained backlog, and search reuse provide complementary serving studies in Appendices A.10, A.11, and A.12.

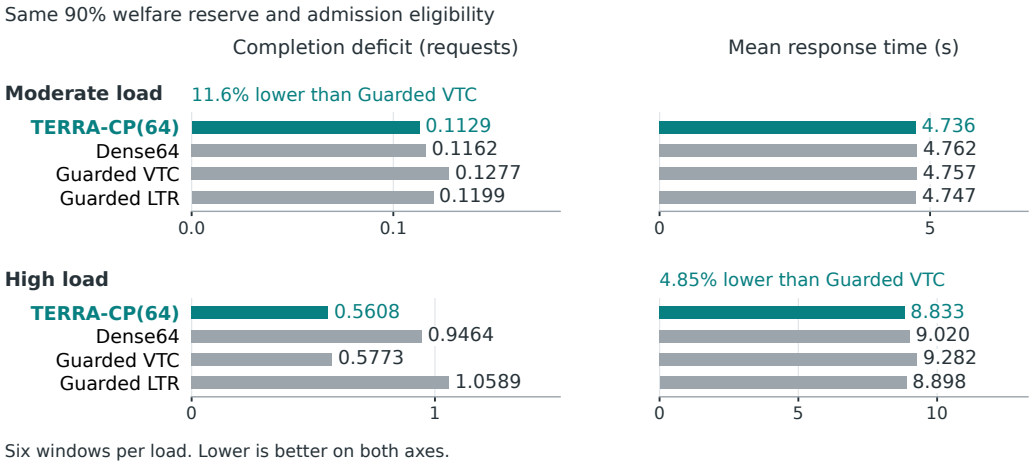


Figure 3: Completed service and response time over six windows per load under the same welfare reserve and implementation rules. Bars give absolute means, with TERRA-CP(64) in teal. Completion deficit is a time-average, so its request units can be fractional. Annotated reductions compare TERRA-CP with guarded VTC. Appendix A.9 reports paired intervals, P95 response, welfare, and policy cost.

Completed service. At moderate load, TERRA-CP reduces completion deficit from 0.1277 requests with guarded VTC to 0.1129, an 11.6% reduction. The paired difference is -0.0148 requests, with a 95% interval of $[-0.0266, -0.0031]$. At high load, the means are 0.5608 and 0.5773 for TERRA-CP and guarded VTC, respectively. All 10,368 requests complete without a reported-welfare reserve violation.

Response time. At high load, TERRA-CP reduces mean response from 9.282 to 8.833 s relative to guarded VTC, a 4.85% reduction. The paired difference is -0.450 s, with a 95% interval of $[-0.942, -0.079]$ s. Most of this reduction (0.427 s) occurs before admission.

Search cost on small queues. TERRA-CP and Dense64 share packing, selection, and callback handling, comparing queue-specific order changes with 64 fixed weights under matched implementation rules. At moderate and high load, TERRA-CP records 1.011 and 1.847 ms/completed request versus Dense64's 2.174 and 2.942 ms, reductions of 53.5% and 37.2%. Guarded VTC ranks requests once per search and costs 0.33 and 0.36 ms/completed request less than TERRA-CP.

Appendix A.9.1 reports a 48-tenant extension with 1,646 requests per window.

7 Conclusion

TERRA-CP searches queue-specific score crossings to improve target service at matched grid cost, with a configurable search budget and efficient large-queue construction. Six-tenant vLLM experiments show lower completion deficit at moderate load and mean response time at high load than guarded VTC, with less policy time than Dense64. For shared LLM services, this turns tenant targets into admission decisions, directing capacity toward underserved workloads under a cumulative reported-welfare requirement relative to current-queue greedy allocations.

AI USE STATEMENT

We used generative AI tools for literature retrieval, research ideation, assistance with experimental design, implementation and analysis, and drafting and polishing the manuscript. We reviewed the AI-assisted work and take responsibility for the final content of this paper.

ETHICS STATEMENT

This study concerns resource-allocation fairness in shared LLM serving. It collects no new human-subject data and uses publicly released arrival traces and prompt sources only for systems evaluation. Tenant values, service targets, latency sensitivities, service objectives, and strategic reports are synthetic and must not be interpreted as measurements of real users or protected groups. Production deployment could redistribute delay or access across tenants and would require stakeholder-defined service targets, privacy-preserving telemetry, abuse-resistant reporting, and audits for disparate impact. TERRA-CP is not strategy-proof and does not protect realized welfare under arbitrary misreports.

REPRODUCIBILITY STATEMENT

We provide an anonymized implementation with configuration files, saved measurements, and analysis scripts. Our serving workloads use Azure's 2023 and 2024 LLM inference traces (Patel et al., 2024; Stojkovic et al., 2025) and BurstGPT (Wang et al., 2025) for arrival timing, with prompt inputs from LongBench (Bai et al., 2024) and HumanEval (Chen et al., 2021). The appendix specifies the sources used in each study. These prompt sources supply systems-evaluation inputs, not the original production requests or a full model-accuracy benchmark. The artifact records source provenance, selected slices, and preprocessing, and its README maps results to inputs and reproduction commands. Saved-data analysis is separate from new simulations and GPU-backed serving runs. Model weights and large candidate-audit logs are not bundled. The appendix specifies comparator selection, timing conventions, statistical comparisons, and proofs. Development and evaluation records are kept separate. The serving analysis distinguishes the controller's admission clock from the common external reference used to evaluate completed service.

REFERENCES

Muhammad Adnan, Rohan Mahapatra, Prashant J Nair, Daniel Berger, Pantea Zardoshti, Rodrigo Fonseca, and Esha Choukse. Cascade: Exploiting slo-aware latency budget for fair and high goodput LLM inference serving. *arXiv preprint arXiv:2608.06557*, 2026.

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 *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 117–134, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/agrawal.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.172. URL http://dx.doi.org/10.18653/v1/2024.acl-long.172.

Julien Basch, Leonidas J Guibas, and John Hershberger. Data structures for mobile data. *Journal of Algorithms*, 31(1):1–28, April 1999. ISSN 0196-6774. doi: 10.1006/jagm.1998.0988. URL http://dx.doi.org/10.1006/jagm.1998.0988.

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. *arXiv preprint arXiv:2107.03374*, 2021.

Qiaoling Chen, Zhisheng Ye, Tian Tang, Peng Sun, Boyu Tian, Guoteng Wang, Shenggui Li, Yonggang Wen, Zhenhua Han, and Tianwei Zhang. CONCUR: High-throughput agentic batch inference of LLM via congestion-based concurrency control. 2026. URL https://openreview.net/forum?id=U7c6glPdDW.

Richard Cole, Jeffrey S. Salowe, W. L. Steiger, and Endre Szemerédi. An optimal-time algorithm for slope selection. *SIAM Journal on Computing*, 18(4):792–810, August 1989. ISSN 1095-7111. doi: 10.1137/0218055. URL http://dx.doi.org/10.1137/0218055.

William J Cunningham. Token management in multi-tenant ai inference platforms. *arXiv preprint arXiv:2603.00356*, 2026.

D. Eppstein. Geometric lower bounds for parametric matroid optimization. *Discrete & Computational Geometry*, 20(4):463–476, December 1998. ISSN 0179-5376. doi: 10.1007/pl00009396. URL http://dx.doi.org/10.1007/pl00009396.

Yichao Fu, Siqi Zhu, Runlong Su, Aurick Qiao, Ion Stoica, and Hao Zhang. Efficient LLM scheduling by learning to rank. In *Advances in Neural Information Processing Systems 37*, NeurIPS 2024, pp. 59006–59029. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024. doi: 10.52202/079017-1882. URL http://dx.doi.org/10.52202/079017-1882.

Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: fair allocation of multiple resource types. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, pp. 323–336, USA, 2011. USENIX Association.

Michael Holzhauser and Sven O. Krumke. An FPTAS for the parametric knapsack problem. *Information Processing Letters*, 126:43–47, October 2017. ISSN 0020-0190. doi: 10.1016/j.ipl.2017.06.006. URL http://dx.doi.org/10.1016/j.ipl.2017.06.006.

Weizhe Huang, Tao Peng, Tongxuan Liu, Donghe Jin, Kang Meng, Xianzhe Dong, and Ke Zhang. ProServe: Unified Multi-Priority Request Scheduling for LLM serving. *arXiv preprint arXiv:2512.12928*, 2025.

Patrick Jaillet, Jiashuo Jiang, Konstantina Mellou, Marco Molinaro, Chara Podimata, and Zijie Zhou. Online scheduling for LLM inference with kv cache constraints. *arXiv preprint arXiv:2502.07115*, 2025.

R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. 1998. URL https://arxiv.org/abs/cs/9809099.

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 *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626. ACM, October 2023. doi: 10.1145/3600006.3613165. URL http://dx.doi.org/10.1145/3600006.3613165.

Yuhan Liu, Cong Xu, Qi Jia, Yihua Wang, Feiyu Chen, Liang Jin, Lu Liu, Yaqian Zhao, Yuting Ding, and Xiang Li. H-MAS: Hierarchical multi-agent scheduling for multi-tenant LLM serving. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 39051–39071. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1946. URL http://dx.doi.org/10.18653/v1/2026.findings-acl.1946.

Hongtao Lyu, Boyue Liu, Mingyu Wu, and Haibo Chen. Fairbatching: Fairness-aware batch formation for LLM inference. *arXiv preprint arXiv:2510.14392*, 2025.

Nimrod Megiddo. Applying parallel computation algorithms in the design of serial algorithms. pp. 399–408, October 1981. doi: 10.1109/sfcs.1981.11. URL http://dx.doi.org/10.1109/sfcs.1981.11.

Michael Neely. *Stochastic network optimization with application to communication and queueing systems*. Morgan & Claypool Publishers, 2010.

A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks-the single node case. pp. 915–924 vol.2, 1992. doi: 10.1109/infcom.1992.263509. URL http://dx.doi.org/10.1109/infcom.1992.263509.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 118–132. IEEE, 2024.

Ying Sheng, Shiyi Cao, Dacheng Li, Banghua Zhu, Zhuohan Li, Danyang Zhuo, Joseph E Gonzalez, and Ion Stoica. Fairness in serving large language models. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 965–988, 2024.

M. Shreedhar and George Varghese. Efficient fair queueing using deficit round robin. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, COMM95, pp. 231–242. ACM, October 1995. doi: 10.1145/217382.217453. URL http://dx.doi.org/10.1145/217382.217453.

Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Josep Torrellas, and Esha Choukse. DynamoLLM: Designing LLM inference clusters for performance and energy efficiency, March 2025. URL http://dx.doi.org/10.1109/hpca61900.2025.00102.

Yinghao Tang, Tingfeng Lan, Xiuqi Huang, Hui Lu, and Wei Chen. Scorpio: Serving the right requests at the right time for heterogeneous SLOs in LLM inference. *arXiv preprint arXiv:2505.23022*, 2025.

L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. pp. 2130–2132 vol.4, 1990. doi: 10.1109/cdc.1990.204000. URL http://dx.doi.org/10.1109/cdc.1990.204000.

Yuxin Wang, Yuhan Chen, Zeyu Li, Xueze Kang, Yuchu Fang, Yeju Zhou, Yang Zheng, Zhenheng Tang, Xin He, Rui Guo, Xin Wang, Qiang Wang, Amelie Chi Zhou, and Xiaowen Chu. Burstgpt: A real-world workload dataset to optimize LLM serving systems. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pp. 5831–5841. ACM, August 2025. doi: 10.1145/3711896.3737413. URL http://dx.doi.org/10.1145/3711896.3737413.

Yifan Wu, Roshan Shariff, Tor Lattimore, and Csaba Szepesvári. Conservative bandits. In *International conference on machine learning*, pp. 1254–1262. PMLR, 2016.

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 (OSDI 22)*, pp. 521–538, 2022.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX symposium on operating systems design and implementation (OSDI 24)*, pp. 193–210, 2024.

A Additional Experiments and Proofs

The appendix first gives the algorithm and large-queue construction ablation (Appendices A.1 and A.2), then the matched-time grid comparison (Appendix A.5). The serving evidence groups the six-window policy comparison, cross-model execution, sustained backlog, and computation reuse (Appendices A.9, A.10, A.11, and A.12). Subsequent sections analyze reserve-adapted baselines, selector ablations, family and packing gaps, and robustness. Mathematical certificates and proofs follow the experimental evidence. Each comparison retains its own cohort and timing boundary.

Unless a caption specifies a narrower comparison, bold in method-comparison tables marks the best observed mean among policies sharing the stated constraints, including ties. Teal identifies TERRA-CP. Bold indicates a point estimate, not statistical significance. Paired intervals are reported separately. Configuration, margin-assessment, and state-diagnostic tables are not rankings.

A.1 Algorithm and implementation settings

In the fractional simulator, each round commits one complete allocation. We construct candidates, check the welfare reserve, and select an allocation as follows. Sequential admission to the serving engine additionally checks the reserve after each accepted prefix (Appendix A.7).

TERRA-CP: hybrid search for one allocation round

Input: active requests $\mathcal{A}_t$, all workload balances Q^t, service targets ρ, capacity C_t, welfare reserve B_t, and budget K.

1. Compute burden, full-bundle value, and affine coefficients using Equations (5) and (3).
2. For uncapped search or fewer than 256 active requests, form the explicit list of zero, distinct positive crossings, adjacent-interval midpoints, and a tail point, retaining evenly spaced indices if capped. Otherwise, find at most $K - 2$ approximate pair-crossing quantiles by inversion counting and add zero and a tail point (Appendix A.2).
3. At each retained weight, rank requests by score with the same fixed tie keys. Scan the order, clip demand to residual capacity, and allocate only if the resulting service fraction meets the minimum. Deduplicate allocations.
4. Pack the reported-value-density greedy order, skipping nonpositive values, and include it as G_t. Compute $r_t = \mathrm{res}_\gamma(\widehat{W}_t(G_t))$.
5. Retain candidates with $B_t + \widehat{W}_t(X) - r_t \geq 0$. Select the largest $(-M_t(X), P_t(X), \widehat{W}_t(X), \mathrm{sig}(X))$.
6. Commit X_t, set $B_{t+1} = B_t + \widehat{W}_t(X_t) - r_t$, and update active balances by $Q_i^{t+1} = [Q_i^t + \rho_i - a_i^t]_+$ using realized service. Inactive balances remain unchanged.

Table 3: Default fractional experiment and implementation settings.

Setting	Value
Search budget and welfare floor	$K = 64$, $\gamma = 0.90$
Burden mixture and minimum positive service	$\omega = 0.5$, $a_{\min} = 0.20$
Workloads and rounds per trajectory	$N = 100$, $T = 1000$
Coefficient representation	12 significant decimal digits, half-even rounding
Score crossings, priorities, and reserve	80-digit decimal arithmetic
Resource and service-target denominator floor	10^{-9}
Full-bundle value normalization floor	10^{-12}
Grid range in the matched-time frontier	zero and positive weights in $[10^{-8}, 10^{8}]$

Ties use a deterministic key derived from the common round seed and request identity. In explicit enumeration, connected equal-score requests at a recorded crossing are assigned the same score before applying that key. The tail point is the final crossing plus the larger of one and its magnitude, or one without positive crossings. Appendix A.2 gives the implicit branch's tail and tie handling. Packing and realized resource measurements use floating-point arithmetic. Appendix A.17 compares the resulting uncapped allocation sets with independent rational-arithmetic enumeration.

Scheduler timing covers score construction, candidate-weight construction, packing, deduplication, reserve checks, and selection inside the allocation call. It excludes trace generation, post-allocation accounting, and LLM execution. Matched-grid studies report elapsed ms/round, while published-policy and search-budget studies report process CPU ms/round. The large-queue study additionally normalizes elapsed time by eligible-request occurrences, in ms/request evaluation, with raw round costs alongside it. Fixed-state replay timings use ms/call. For serving, policy time is total recorded elapsed decision time divided by completed requests, in ms/completed request. Callback contributions use this same completed-request denominator. Ratios compare the same clock and denominator within each experiment, without pooling hardware environments.

A.2 Hybrid candidate construction and large-queue evaluation

TERRA-CP combines two ways to construct candidate weights. Explicit enumeration is inexpensive for short queues, but examines every request pair even when only 64 weights will be retained. For larger queues, implicit search locates crossing quantiles through order-inversion counts. The connection between geometric selection and ordering is classical (Cole et al., 1989). Both branches feed the same packing rule, debt selector, and cumulative welfare reserve.

Counting without enumerating. Let m count positive pairwise crossings with multiplicity, and let d count distinct positive crossing values. The order immediately to the right of zero sorts by decreasing (α_i, β_i), followed by the fixed tie key. For a query $\nu > 0$, sorting by score and breaking equal scores by increasing slope gives the order immediately before ν. Its inversion count relative to the initial order equals the number of crossings in $(0, \nu)$. Decreasing-slope ties instead include crossings at ν, giving the count in $(0, \nu]$. Parallel and coincident lines contribute no positive crossing. Comparing the initial order with decreasing (β_i, α_i) similarly gives m. A sort and inversion count require $O(n \log n)$ operations, without materializing the pair list.

Locating candidate weights. We distribute at most $K - 2$ target ranks across the m pair events, retaining their first and last rank when the budget permits. These ranks share geometric bisection queries inside positive bounds derived from coefficient gaps. We stop a branch after 24 bisection levels or when its rank bracket has width at most $\lceil m/[2(K-2)] \rceil$, retain its midpoint, and remove duplicate weights. Zero and a weight above every positive crossing complete the candidate list. With $K = 2$, only these two endpoints remain. If there are no positive crossings, we use zero and one. Pair-event quantiles count coincident crossings with multiplicity, unlike the explicit cap over distinct crossings and interval representatives. Both rules retain the greedy reference and use the same capacity and welfare checks, but can choose different allocations.

Construction cost. Explicit construction scans $\binom{n}{2}$ pairs, stores m positive events, and sorts d distinct crossing values. Under unit-cost arithmetic and expected constant-time dictionary operations, its cost is $O(n^2 + m + d \log(d+1))$. Implicit construction takes $O((1+q)n \log n)$, where $q \le (K-2)L$ is the number of distinct count queries and $L = 24$ is the bisection limit. Sharing queries and stopping by rank width can make q much smaller than this bound. Each branch additionally sorts and packs at most K candidate orders. The uncapped mode always uses explicit enumeration and retains the coverage of Lemma 1.

Routing and numerical implementation. For capped search, TERRA-CP uses explicit construction below 256 active requests and implicit construction otherwise. We fixed this hardware-dependent threshold using three development timing states before the fixed-state and full-trajectory evaluations. The default budget is 64, the bisection limit is 24, and the rank divisor is 2. Counts distinguish open from closed intervals. When finite-precision evaluation produces equal scores, the implicit branch refines their ordering using the exact supplied decimal coefficients before applying the fixed request tie key. The enumeration-only ablation uses explicit construction at every queue

size. The main-text 100-workload and six-tenant serving comparisons exercise TERRA-CP's small-queue branch.

Large-queue evaluation and construction ablation. We compare TERRA-CP with its enumeration-only ablation and Dense136 at 1,000 workloads, with capacity scaled proportionally, the same five workload classes, minimum service fraction 0.2, and welfare floor 0.90. Six states from two development trajectories select among budgets 32, 48, 64, and 96 using elapsed allocation time alone. After one warmup and three rotated timing repeats per state, budget 64 is closest to Dense136 within the prescribed 10% band, with geometric time ratio 1.079. We freeze it before five new paired 200-round trajectories. Policies run serially in rotated order on one logical core of an Intel Xeon Silver 4410Y under Linux and Python 3.10. Complete allocation-call timings exclude state copying, post-allocation accounting, and output. Each policy evolves its own service and reserve balances under identical external inputs.

For each trajectory, we divide summed elapsed allocation time by summed active-request counts, then average these five ratios equally. Candidate-construction time uses the same denominator. There are 408,728 active-request occurrences per policy across the 1,000 evaluated rounds, with 317 to 515 active requests per round. A request counted again in a later round contributes another evaluation; this denominator is neither distinct requests nor completed requests.

Table 4: Large-queue construction ablation and fixed-grid reference on five paired 200-round trajectories at 1,000 workloads. Time is elapsed ms/request evaluation: a request contributes once each round it is active, not once per completion. Bold compares the two construction variants' time columns. Service means and their paired differences quantify the accompanying quality changes. All policies retain at least 90% realized welfare relative to greedy.

Policy	Allocation time ↓	Construction time ↓	Deficit/round ↓	Minimum fulfillment (%) ↑
Construction ablation: both variants use a 64-weight cap				
TERRA-CP(64)	**0.7076**	**0.2665**	0.06564	44.02
Enumeration only	1.0633	0.6857	0.06536	44.16
Fixed-grid reference				
Dense136	0.6114	n/a	0.06864	41.24

The corresponding mean elapsed times per complete allocation round are 289.194 ms for TERRA-CP, 434.604 ms for enumeration only, and 249.895 ms for Dense136. Per-request normalization does not change the underlying measurements or make this a serving-throughput result.

TERRA-CP reduces mean allocation time by 33.5% relative to its enumeration-only ablation, with savings on all five trajectories (Table 4). The paired geometric time ratio is 0.665, with 95% interval [0.656, 0.675]. Savings are concentrated in candidate construction, whose mean falls by 61.1%. Both configurations retain 63.69 weights per round on average, so the reduction does not come from a smaller weight budget. Maximum terminal deficit matches in four trajectories and rises in one, giving a mean change of 0.000278 per round, with paired 95% interval $[-0.000494, 0.001051]$. Minimum fulfillment changes by -0.139 percentage points, with interval $[-0.378, 0.099]$. All 1,000 TERRA-CP calls use implicit construction and satisfy the rank-width tolerance and the capacity and welfare checks.

Relative to Dense136, mean deficit is 4.4% lower and minimum fulfillment is 2.78 percentage points higher. Both improve on three trajectories and tie on two. Their paired differences are -0.00300 deficit per round (95% interval $[-0.00640, 0.00040]$) and 2.78 fulfillment points ($[-0.43, 5.98]$). The elapsed-time ratio is 1.157, with 90% interval [1.144, 1.170], outside the prescribed [0.90, 1.10] equivalence band. This large-queue comparison therefore measures the remaining quality-cost tradeoff, separately from the matched-time 100-workload result. Queue-size-dependent candidate construction reduces large-queue search cost while retaining the same packing and welfare-reserve design.

A.3 CONTROLLED SIMULATOR INSTANTIATION

The simulator isolates candidate search under heterogeneous resource demands, service targets, and request values. It samples five profiles with the mixture probabilities and parameter ranges in Table 5. The cloud, local-strong, and local-weak profiles vary resource demand, value, quality, and throughput together. The bandwidth-limited profile raises bandwidth demand and network delay. The latency-sensitive profile assigns delay a larger utility penalty. Parameters are drawn uniformly from the profile-specific ranges and retained throughout the trajectory. Each round draws active workloads with periodically varying Bernoulli probabilities. Capacity combines a periodic component with Gaussian jitter. Both sequences are generated before the policy runs, giving paired policies identical external conditions but separate debt accounts. Trace-derived arrivals and model execution are evaluated separately.

To separate allocation inputs from realized outcomes, the controlled simulator assigns each active workload a data-generating type

$$\theta_i = (v_i, c_i, d_i, \rho_i, q_i, \tau_i, s_i, \kappa_i, \mu_i, \chi_i, L_i^{\text{base}}, L_i^{\text{net}}).$$

Here v_i, c_i are value and base participation cost, χ_i is a workload-specific participation-cost multiplier, $d_i = (d_i^g, d_i^k, d_i^\ell)$ is GPU-time, token, and bandwidth demand, ρ_i is the active-round service target, and q_i, τ_i, s_i are quality, latency sensitivity, and the response-time target. The parameter κ_i controls quality saturation with allocated tokens, μ_i is the token-processing rate, and L_i^{base} and L_i^{net} are baseline and network-latency terms. The service-level objective (SLO), s_i, is used by serving guards.

Table 5: Fractional-workload sampling ranges. GPU and bandwidth demands use simulator resource units, token demand uses tokens, μ uses tokens/s, and latency uses seconds. Values v, c use configured utility units, τ uses utility units/s, and κ uses inverse tokens. The service target ρ is in service units per active round. Quality q, cost multiplier χ, and base activity probability p are dimensionless.

Parameter	Cloud	Local strong	Local weak	Bandwidth limited	Latency sensitive
Mixture	0.20	0.25	0.25	0.15	0.15
v	[18, 30]	[12, 22]	[8, 16]	[10, 18]	[12, 24]
c	[0.20, 0.35]	[0.12, 0.24]	[0.04, 0.12]	[0.07, 0.17]	[0.08, 0.18]
d^g	[1.00, 1.80]	[0.65, 1.25]	[0.35, 0.90]	[0.45, 1.05]	[0.50, 1.15]
d^k	[900, 1900]	[650, 1500]	[450, 1150]	[550, 1300]	[500, 1150]
d^ℓ	[80, 170]	[40, 115]	[20, 75]	[110, 240]	[45, 125]
ρ	[0.38, 0.58]	[0.34, 0.52]	[0.28, 0.46]	[0.30, 0.48]	[0.32, 0.52]
q	[1.05, 1.25]	[0.88, 1.04]	[0.62, 0.82]	[0.70, 0.92]	[0.82, 1.00]
τ	[0.18, 0.35]	[0.35, 0.65]	[0.35, 0.75]	[0.45, 0.85]	[0.95, 1.55]
$10^3\kappa$	[2.0, 3.5]	[2.0, 4.5]	[2.5, 5.5]	[2.2, 4.8]	[2.2, 4.8]
μ	[250, 420]	[95, 170]	[35, 70]	[55, 100]	[75, 135]
L^{base}	[0.05, 0.11]	[0.11, 0.22]	[0.22, 0.42]	[0.18, 0.34]	[0.04, 0.10]
L^{net}	[0.03, 0.08]	[0.04, 0.10]	[0.04, 0.12]	[0.18, 0.35]	[0.02, 0.06]
χ	[0.90, 1.20]	[0.55, 0.85]	[0.25, 0.55]	[0.35, 0.65]	[0.35, 0.75]
p	[0.36, 0.52]	[0.34, 0.50]	[0.30, 0.48]	[0.30, 0.46]	[0.32, 0.50]

In the released code, κ, μ, χ, p correspond to `beta`, `throughput`, `local_cost`, and `activity_prob`, respectively. The demand fields are `gpu`, `tokens`, and `bandwidth`.

The input used for allocation, $\widehat{\theta}_i = (\widehat{v}_i, \widehat{d}_i, \widehat{\tau}_i)$, replaces value, demand, and latency sensitivity for scoring. The service target and SLO are administrator-set. The simulator permits $\rho_i = 0$, while the serving bridge requires $\rho_i \in (0, 1]$. We evaluate outcomes using the data-generating type. Thus a report deviation changes the allocation input, but not the underlying preference. Values are administrator priorities rather than private tenant bids.

All three resources are required components of a fractional bundle. True and reported demands in this instantiation satisfy $d_{ir}, \widehat{d}_{ir} \geq \epsilon$, where $\epsilon = 10^{-9}$. Ranked greedy clips each reported bundle to

residual capacity and requires a minimum reported service fraction of 0.2. Realized service is

$$a_i(x) = \max\{0, \min\{1, \min_r x_{ir}/\max(d_{ir}, \epsilon)\}\}.$$

For allocated tokens, GPU time, and bandwidth (k_i, g_i, ℓ_i), the output-quality model is

$$q_i^{\text{out}} = q_i(1 - e^{-\kappa_i k_i}) \min\{1, g_i/\max(d_i^g, \epsilon)\} \min\{1, \ell_i/\max(d_i^\ell, \epsilon)\}.$$

For a positive allocation, latency and utility are

$$L_i = L_i^{\text{base}} + \frac{k_i}{\max(\mu_i, \epsilon)} + L_i^{\text{net}} \left(1 + \left[1 - \frac{\ell_i}{\max(d_i^\ell, \epsilon)}\right]_+\right), \qquad U_i = v_i q_i^{\text{out}} - \tau_i L_i - c_i \chi_i a_i(x).$$

For no allocation, meaning no component above ϵ, the implementation sets $q_i^{\text{out}} = L_i = a_i = U_i = 0$. Reported utility substitutes only $\widehat{v}_i, \widehat{\tau}_i, \widehat{d}_i$ in these definitions. Realized service is evaluated under data-generating demand after allocation. The serving bridge instead credits whole-request first admission rather than fractional resource-metered service. Completion service is measured separately against the common external reference in Appendix A.8.

A.4 Operating-point selection

We select a welfare floor of 0.90 during development because it gives the largest improvements in maximum cumulative deficit and minimum fulfillment, while preserving realized-welfare retention and Jain's fairness index of fulfillment (Jain et al., 1998). Their prespecified allowed losses are 0.20 percentage points and 0.01, respectively. At floor 0.85, the fairness differences from equal-budget Dense64 are less conclusive. Increasing congestion also changes the minimum-fulfillment comparison. We evaluate the selected floor on new trajectories and separately vary workload and welfare floor below.

A.5 Matched-runtime fixed-grid frontier

To compare candidate placement at similar scheduler cost, we use 30 paired trajectories with 100 workloads, 1,000 rounds, and a 0.90 welfare floor. Each fixed grid contains zero and logarithmically spaced positive weights from 10^{-8} through 10^{8}. We fix Dense136 before this 30-pair run as the comparator targeting TERRA-CP(64)'s measured elapsed allocation time. The arms run sequentially in counterbalanced order.

Table 6: Absolute outcomes and paired comparisons at matched scheduling time. Each study uses 30 paired trajectories. Bold marks the best observed mean within each upper block. The lower block gives geometric elapsed-time ratios (TERRA-CP/comparator) and service differences (TERRA-CP minus comparator), with paired 95% intervals. Dense136 and Adaptive88 meet the $[0.90, 1.10]$ time-equivalence criterion. Adaptive88 uses an independent cohort, with its own TERRA-CP means.

Policy	Elapsed (ms/round) ↓	Deficit / round ↓	Min. fulfillment (%) ↑
TERRA-CP(64)	45.57	**0.04064**	**68.29**
Dense64	**23.55**	0.05194	61.28
Dense136	45.92	0.04745	63.64
Dense192	63.65	0.04615	64.41
Independent adaptive-grid study			
TERRA-CP(64)	45.69	**0.03604**	**72.53**
Adaptive88	**44.73**	0.03883	70.48

Comparator	Metric	Difference or ratio	95% CI
Dense64	Elapsed-time ratio	1.935	$[1.918, 1.953]$
	$\Delta \max D_+/T \downarrow$	-0.01130	$[-0.01284, -0.00977]$
	$\Delta \min \phi \uparrow$	$+0.0701$	$[+0.0567, +0.0835]$
Dense136	Elapsed-time ratio	0.993	$[0.983, 1.003]$
	$\Delta \max D_+/T \downarrow$	-0.00682	$[-0.00796, -0.00568]$
	$\Delta \min \phi \uparrow$	$+0.0465$	$[+0.0378, +0.0553]$
Dense192	Elapsed-time ratio	0.716	$[0.710, 0.722]$
	$\Delta \max D_+/T \downarrow$	-0.00551	$[-0.00649, -0.00454]$
	$\Delta \min \phi \uparrow$	$+0.0388$	$[+0.0326, +0.0449]$
Adaptive88	Elapsed-time ratio	1.022	$[1.010, 1.033]$
	$\Delta \max D_+/T \downarrow$	-0.00280	$[-0.00347, -0.00212]$
	$\Delta \min \phi \uparrow$	$+0.0205$	$[+0.0149, +0.0261]$

TERRA-CP(64) improves maximum deficit and minimum fulfillment in all 30 pairs against Dense136. Its realized-welfare retention changes by -0.0103 percentage points (95% interval $[-0.0128, -0.0078]$). Jain's index of fulfillment changes by $+0.000326$ ($[-0.000781, +0.001433]$). Both remain within the prespecified preservation margins. Including the separate greedy reference, TERRA-CP(64) evaluates 64.94 proposals per round versus Dense136's 137.00, yet produces 49.43 versus 14.11 unique allocations and 13.22 versus 5.08 reserve-feasible candidates. Both candidate-count differences favor TERRA-CP(64) in every trajectory.

A.6 Adaptive-grid calibration

Runtime-matched queue-adaptive control. Using scheduler time alone on five development trajectories, we selected Adaptive88 from grid sizes $72, 80, 88, 96$. Its TERRA-CP(64)/Adaptive88 runtime ratio was 1.011 in development, inside the prespecified $[0.95, 1.05]$ gate, and the grid size was then frozen. Adaptive88 derives its logarithmic endpoints from each queue's current positive crossings. It otherwise shares TERRA-CP(64)'s packer, welfare reserve, debt selector, and exogenous randomness, although their visited states can diverge.

Figure 4 reports the independent 30-pair fairness and runtime comparison. Realized-welfare retention changes by -0.000047 (95% interval $[-0.000066, -0.000027]$), and fulfillment Jain by $+0.001837$ ($[0.001276, 0.002398]$). Both meet their preservation margins. Each policy evolves on its own queue and debt states.

A.7 Request-prefix welfare reserve

The waiting bridge scans physically feasible pending requests and adjusts base value for waiting:

$$\widehat{V}_i^t = \widehat{V}_i^{\text{base}} - \widehat{\tau}_i w_i^t,$$

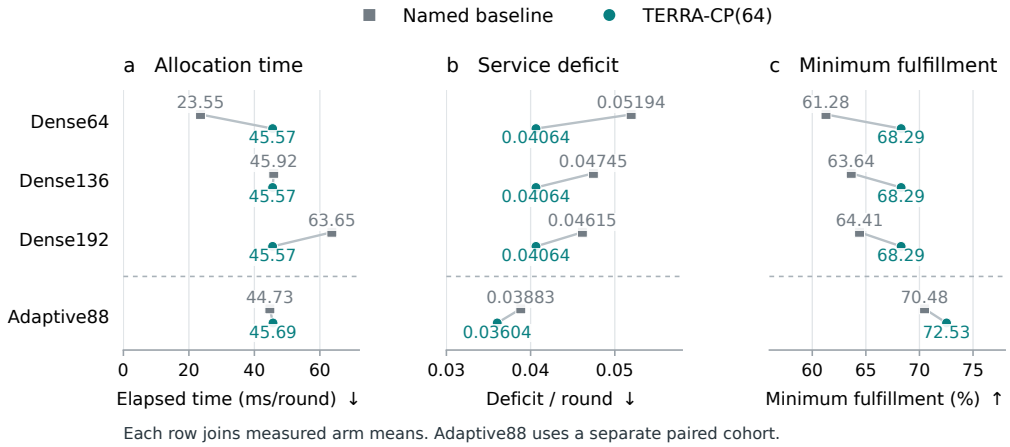


Figure 4: Both measured policies in every matched-time comparison. Teal circles show TERRA-CP(64), and gray squares show the named comparator, with a line joining their absolute means. Lower time and deficit are preferable, while higher fulfillment is better. Fixed-grid comparisons share one 30-trajectory cohort. Adaptive88 uses a separate 30-trajectory cohort and its own TERRA-CP reference. Table 6 gives the paired intervals.

where w_i^t is waiting time. It applies the signed greedy reserve in Proposition 7.

Because an engine may admit only a prefix, the waiting bridge checks the reserve throughout candidate and reference sequences $\pi_X = (x_1, \ldots, x_m)$ and $\pi_G = (g_1, \ldots, g_n)$, requiring

$$\min_{1 \le k \le \max(m,n)} \left\{ B_t + \widehat{W}_t(\pi_X^{1:k}) - \mathrm{res}_\gamma(\widehat{W}_t(\pi_G^{1:k})) \right\} \ge 0, \tag{9}$$

with each cumulative sequence held constant after it ends. This includes the reference tail when the candidate sequence is shorter. If the engine commits only a prefix, the reserve update charges the matched committed prefixes of the candidate and greedy sequences. Uncommitted requests remain queued for reconsideration.

A.8 Completion service on a common clock

Let $A_i(t)$ count tenant i's prescribed arrivals and $C_i(t)$ count successful complete HTTP responses received by time t. Each completed request contributes one unit, including multiple responses from one tenant in an arrival epoch. Completion records response delivery, not semantic quality.

For every tenant with a prescribed arrival, we fix a calibrated reference rate r_i before comparing policies and define

$$R_i(0) = 0, \qquad \dot{R}_i(t) = r_i \, \mathbf{1}\{R_i(t) < A_i(t)\}.$$

The reference stops when caught up with offered demand and accrues no idle credit. Neither its rate nor its path depends on policy queues, admissions, or completions. Each experiment specifies its calibrated rates and an observation horizon H fixed from the offered workload, rather than either policy's finish time.

The time-average largest tenant shortfall is

$$\bar{D} = \frac{1}{H} \int_0^H \max_i [R_i(t) - C_i(t)]_+ \, dt,$$

measured in requests. We integrate the piecewise-affine maximum at reference changes, line crossings, and response completions. Unlike deadline goodput, this measures accumulated tenant shortfall over time, not individual deadline misses. The six-window comparison in Appendix A.9 and the search-reuse ablation in Appendix A.12 state their own rates and horizons.

A.9 COMPARISON WITH TOKEN AND LEARNED RANKINGS

Workload and execution. We compare four welfare-constrained policies on six Azure 2024 arrival windows (Stojkovic et al., 2025), with three conversational and three code windows. Each window offers 216 requests from six tenants using the same 36 prompts from LongBench (Bai et al., 2024) and HumanEval (Chen et al., 2021), balanced across QA, summarization, and code generation. The prompts are excluded from ranker training. The two offered rates are 0.9 and 1.2 times a previously measured FCFS capacity of 1.8143 requests/s, giving arrival spans of 131.672 and 98.754 s. We fix the six windows, prompt assignments, and loads before execution, rescale source timestamps uniformly, and round them to 1 ms.

The 48 runs use Qwen2.5-7B-Instruct and vLLM 0.8.5 with eight execution slots. The server uses the V0 engine (`VLLM_USE_V1=0`) without prefix caching. Three RTX 6000 Ada GPU workers execute disjoint window-load blocks. All four policies within a block use the same GPU, CPU affinity, and server configuration. Workers are pinned to disjoint sets of eight Xeon Silver 4410Y CPU cores, with one numerical-library thread per process. Each policy starts a fresh server. Cyclically shifting the order TERRA-CP, Dense64, guarded VTC, guarded LTR across the twelve blocks puts every policy in each execution position three times. The context and batch token budgets are both 4,096 tokens. Generation ends at EOS, with safety caps of 256 output tokens for QA and 1,024 for summarization and code. All 10,368 requests finish without errors or cap termination and complete by the fixed observation horizon, the last prescribed arrival plus 60 s. The common external completion reference accrues at 0.2419026320 requests/s per tenant while below that tenant's cumulative prescribed arrivals (Appendix A.8).

Comparable admission rules. Dense64 changes only TERRA-CP's weight search. Guarded VTC orders eligible requests by weighted observed-token service, and guarded LTR orders them by predicted output-length rank. All four methods admit at most one waiting head per tenant in each decision and enforce the same cumulative 90% reported-welfare reserve, signed-reference convention, and greedy fallback. All four skip ranking and candidate construction when no slot is free, while preserving admission-account updates. VTC also preserves token-counter lifting. They use the same completion-event and queue instrumentation. No run violates the reported-welfare reserve.

The LTR adapter uses the OPT-125m ranker and ListMLE objective of Fu et al. (2024). We initialize it from the authors' checkpoint trained on LMSYS prompts with Llama-3-8B outputs, then adapt it to Qwen using 298 training prompts and five fixed epochs, learning rate 2×10^{-5}, and ranking lists of 32 prompts. Training labels are natural-EOS Qwen output lengths with the same task caps. The 18 development prompts and 36 serving prompts are excluded from training by identity and exact text. The development set measures ranking quality without selecting an epoch. On these 18 development prompts, the final ranker orders 72.37% of the 152 unequal-length pairs correctly. The 36 serving prompts have cached scores prepared in 23.23 s on four CPU threads. Admission reuses these scores without observing test output lengths. We report this preparation separately from request response and decision time.

Table 7: Whole-request serving under the same 90% reported-welfare reserve and admission rules. All metrics are lower-is-better. Bold marks the lowest mean at each load, not statistical significance. Teal identifies TERRA-CP. Entries average six windows, including window-specific P95s. Policy time sums elapsed decision time per completed request.

Load	Policy	Deficit (requests)	Response (s)	P95 (s)	Policy time (ms/completed request)
Moderate	**TERRA-CP(64)**	**0.1129**	**4.736**	10.708	1.011
	Dense64	0.1162	4.762	**10.702**	2.174
	Guarded VTC	0.1277	4.757	10.822	0.681
	Guarded LTR	0.1199	4.747	10.823	**0.651**
High	**TERRA-CP(64)**	**0.5608**	**8.833**	28.537	1.847
	Dense64	0.9464	9.020	31.046	2.942
	Guarded VTC	0.5773	9.282	**27.932**	1.483
	Guarded LTR	1.0589	8.898	30.940	**1.363**

Service outcomes. TERRA-CP has the lowest observed mean completion deficit and response time at both loads. The paired intervals in Table 8 support an 11.6% deficit reduction at moderate load and a 4.85% response reduction at high load relative to guarded VTC. The remaining deficit, mean-response, and P95 comparisons have intervals including zero.

Table 7 weights the six windows equally at each load. Mean P95 is the average of window-specific P95 response times. Policy time sums the admission policy's elapsed time over all recorded calls, including calls without a new admission opportunity, then divides by the 216 completed requests. It excludes predictor preparation, queue parsing, token-service observation, admission commit, audit I/O, and the engine's own scheduling work. It is not time per candidate search or the process-CPU time in the fractional study. Paired 95% bootstrap intervals use 10,000 resamples of the six whole windows, with the same window draws for both loads. The window, rather than an individual request or callback, is the resampling unit.

Table 8: Absolute outcomes and paired differences over the six fixed windows. Both policy means precede the TERRA-CP(64) minus comparator difference and its 95% paired bootstrap interval. Negative differences favor TERRA-CP for all four endpoints. All comparators share the reported-welfare reserve, head-only eligibility, and zero-slot shortcut.

Control	**TERRA-CP(64)**	Control mean	Δ	95% CI
Moderate, Deficit (requests)				
Dense64	0.1129	0.1162	-0.0032	$[-0.0095, +0.0035]$
Guarded VTC	0.1129	0.1277	-0.0148	$[-0.0266, -0.0031]$
Guarded LTR	0.1129	0.1199	-0.0069	$[-0.0145, +0.0002]$
Moderate, Response (s)				
Dense64	4.736	4.762	-0.026	$[-0.095, +0.053]$
Guarded VTC	4.736	4.757	-0.020	$[-0.105, +0.064]$
Guarded LTR	4.736	4.747	-0.010	$[-0.078, +0.063]$
Moderate, P95 response (s)				
Dense64	10.708	10.702	$+0.005$	$[-0.161, +0.228]$
Guarded VTC	10.708	10.822	-0.115	$[-0.416, +0.152]$
Guarded LTR	10.708	10.823	-0.115	$[-0.416, +0.219]$

Table 8 (continued)

Control	TERRA-CP(64)	Control mean	Δ	95% CI
Moderate, Policy time (ms/completed req.)				
Dense64	1.011	2.174	−1.163	[−1.253, −1.080]
Guarded VTC	1.011	0.681	+0.330	[+0.276, +0.392]
Guarded LTR	1.011	0.651	+0.360	[+0.297, +0.430]
High, Deficit (requests)				
Dense64	0.5608	0.9464	−0.3856	[−1.0404, +0.0830]
Guarded VTC	0.5608	0.5773	−0.0165	[−0.3784, +0.3574]
Guarded LTR	0.5608	1.0589	−0.4981	[−1.2121, +0.0316]
High, Response (s)				
Dense64	8.833	9.020	−0.187	[−0.615, +0.238]
Guarded VTC	8.833	9.282	−0.450	[−0.942, −0.079]
Guarded LTR	8.833	8.898	−0.066	[−0.304, +0.107]
High, P95 response (s)				
Dense64	28.537	31.046	−2.508	[−8.566, +1.973]
Guarded VTC	28.537	27.932	+0.605	[−1.394, +2.210]
Guarded LTR	28.537	30.940	−2.403	[−8.811, +1.828]
High, Policy time (ms/completed req.)				
Dense64	1.847	2.942	−1.095	[−1.202, −0.956]
Guarded VTC	1.847	1.483	+0.364	[+0.281, +0.445]
Guarded LTR	1.847	1.363	+0.484	[+0.412, +0.548]

Table 9: Supplemental outcomes under the same reported-welfare reserve, eligibility, and zero-slot shortcut, averaged equally over six windows. Lower policy time and higher goodput and welfare are preferable. Bold marks the best observed mean at each load, including ties, not statistical significance. Realized welfare uses configured utility units, summed over all 216 requests in each run before averaging windows. It subtracts response-time penalties from request value, so it can be negative even when the admission-time reported-welfare reserve is satisfied.

Load	Policy	Policy time (ms/ completed req.)	Deadline goodput	Realized welfare
Moderate	TERRA-CP(64)	1.011	**1.0000**	**-35.359**
	Dense64	2.174	**1.0000**	-37.554
	Guarded VTC	0.681	**1.0000**	-37.448
	Guarded LTR	**0.651**	**1.0000**	-36.392
High	TERRA-CP(64)	1.847	**0.9985**	**-424.723**
	Dense64	2.942	0.9961	-444.545
	Guarded VTC	1.483	0.9977	-470.614
	Guarded LTR	**1.363**	0.9969	-433.850

Where the response difference occurs. We decompose response at the first scheduler callback that admits each request. At high load, TERRA-CP's mean pre-admission wait is 4.883 s versus 5.310 s for guarded VTC, a difference of -0.427 s with paired 95% interval $[-0.899, -0.081]$. This accounts for 94.9% of the mean-response reduction. The post-admission means are 3.949 and 3.972 s, with difference -0.023 s and interval $[-0.051, +0.006]$. At moderate load, both component intervals include zero. The high-load difference is concentrated before admission, consistent with admission decisions changing waiting time. This boundary includes client and transport delay before the admitting callback; its later component includes callback completion and response return, not only GPU work.

Choices left by packing and the reserve. On TERRA-CP's own trajectories, eligible heads average 1.88 at moderate load and 2.98 at high load, versus 1.91 and 2.94 for Dense64. Distinct admission prefixes average 1.41 and 1.99 for TERRA-CP, versus 1.41 and 1.91 for Dense64. The reserve leaves 1.12 and 1.21 prefixes for TERRA-CP, versus 1.13 and 1.18 for Dense64. These values average within-window means equally. Prefix counts average candidate searches, while eligible-head counts average callbacks with a free slot and at least one eligible request. Prefixes distinguish ordered admitted-request tuples. No TERRA-CP search reaches the 64-weight cap. These trajectory statistics describe how few alternatives reach the selector, without isolating search effects on a common state. Figure 7 gives the separate controlled packing comparison.

Search cost and service accounting. With the common zero-slot shortcut, TERRA-CP costs 1.011 and 1.847 ms/completed request at moderate and high load, versus 2.174 and 2.942 for Dense64, reductions of 53.5% and 37.2%. Guarded VTC costs 0.681 and 1.483 ms, so TERRA-CP adds 0.330 and 0.364 ms, with paired intervals $[0.276, 0.392]$ and $[0.281, 0.445]$. Guarded LTR is also cheaper than TERRA-CP at both loads.

The internal service account credits first admission within each prescribed arrival epoch. Completion deficit instead evaluates finished requests on the common external clock, while realized welfare includes the response-time penalties accumulated during execution.

A.9.1 Forty-eight-tenant scale extension

We compare TERRA-CP(64), Dense64, and guarded VTC on six additional, nonoverlapping Azure conversation-arrival windows. Each supplies 1,646 requests across 48 tenants, assigned cyclically, over 755.584 s (12.6 minutes) of replay. The offered rate is 1.2 times calibrated capacity, matching the main study's high-load setting. We reuse the 36-prompt pool and repeat the six value and latency profiles, with admission targets $\rho_i = 1/48$. The total external completion-reference rate is unchanged and divided equally across tenants, giving 0.0302378290 requests/s per tenant. The model, vLLM engine, token budgets, generation caps, and disabled prefix caching follow Appendix A.9.

Each policy retains its ranking, packing, admission-account updates, 90% reported-welfare reserve, head-only eligibility, and zero-slot rule from the six-tenant comparison. For these longer runs, we use client completion records and omit per-decode full-queue snapshots. All 18 runs start fresh servers and execute serially on one RTX 6000 Ada GPU, with each of the six policy-order permutations used once. Within each window, all policies receive the same prescribed request stream and use identical generation settings with temperature zero. Client concurrency is 2,048 so it does not gate the 1,646 arrivals. Completion deficit uses the common horizon $H = 815.584$ s, while response and policy cost include all eventual completions. All 29,628 requests finish without errors, output-cap termination, or reserve violations. TERRA-CP considers 12.21 eligible heads per search on average across windows, with a recorded maximum of 35. The 64-weight cap truncates the representative list in 73.4% of candidate searches (equal-window mean). Appendix A.19 examines separate fractional states, where TERRA-CP(64) usually matches uncapped search's maximum projected debt despite frequent truncation.

Table 10: Forty-eight-tenant execution over six windows. Panel (a) gives equal-window means, with lower values preferable and the lowest mean in bold. P95 is averaged across windows. The absolute deficit scale depends on the per-tenant completion-reference rates, arrival stream, and observation horizon, which differ from the six-tenant study. Panel (b) gives paired estimates and 95% intervals. Deficit and cost use CP-minus-control differences and paired t intervals. Response and P95 use geometric mean CP/control ratios, with t intervals on log ratios transformed back. All intervals use five degrees of freedom.

(a) Absolute outcomes

Policy	Deficit (requests)	Mean response (s)	Mean P95 (s)	Policy time (ms/completed request)
TERRA-CP(64)	9.0392	54.260	371.198	9.555
Dense64	9.1462	56.572	388.544	9.577
Guarded VTC	**9.0073**	**50.762**	**352.039**	**5.940**

(b) Paired comparisons, TERRA-CP(64) relative to each control

Endpoint	Dense64	Guarded VTC
Deficit difference (requests)	-0.107 $[-0.935, +0.721]$	$+0.032$ $[-1.208, +1.272]$
Mean-response ratio	0.954 $[0.858, 1.060]$	1.058 $[0.952, 1.177]$
P95-response ratio	0.951 $[0.854, 1.059]$	1.049 $[0.952, 1.155]$
Policy-time difference (ms/completed request)	-0.022 $[-0.533, +0.489]$	$+3.615$ $[2.891, 4.339]$

This extension evaluates the same admission rules with more tenants and longer request streams. TERRA-CP has lower observed response and deficit than Dense64, although their paired intervals include no difference. Guarded VTC uses less policy time, and the service comparisons do not establish an advantage over it.

A.10 Cross-model short-output execution

We evaluate cross-model execution on Qwen2.5-7B and Phi-3-mini with a fixed 32-token output cap, complementing the natural-EOS workloads. Weighted response is the primary endpoint, with prespecified margins for deadline goodput, fulfillment Jain, and timing diagnostics. Both policies use the same welfare reserve. These real HTTP and model-serving runs control waiting admission, with prompts capped at 1,500 characters. Their service target clock advances at policy-specific admission opportunities, so controller debt does not measure completed service on a shared clock.

We weight response time by each request's assigned latency sensitivity:

$$L_{\tau,p} = \frac{\sum_r \tau_r t_{r,p}}{\sum_r \tau_r}, \qquad \Delta L_\tau = L_{\tau,\mathrm{CP}} - L_{\tau,\mathrm{dense64}}.$$

Here, τ_r is the configured latency-sensitivity weight, and $t_{r,p}$ is response time under policy p.

We select load profiles using FCFS on two calibration scenarios before evaluating either TERRA-CP(64) or Dense64, then compare 12 paired scenarios per model with balanced execution order and 1,646 BurstGPT-derived arrivals (Wang et al., 2025) per policy, paired with prompts from nonoverlapping LongBench slices. Qwen2.5-7B is the primary model, while Phi-3-mini is a separate architecture-transfer test. The scenario pair is the inference unit. The primary latency margin is 0.10 s, or 3.3% of the shortest 3 s prompt-family SLO. We also assess service through completion, deadline goodput, and fulfillment Jain. Timing criteria cover P95 response and submission delay.

Table 11: Cross-model outcomes and prespecified margins on 12 paired scenarios per model. Absolute means weight scenarios equally. Differences are TERRA-CP(64) minus Dense64 with paired 95% intervals. Response is weighted by configured latency sensitivity. Goodput and Jain entries are fractions. The allowed changes are assessed using the paired intervals, without best-mean highlighting.

Metric	Dense64	TERRA-CP (64)	Difference 95% CI	Allowed change
Qwen2.5-7B				
Weighted response (s) ↓	5.6146	5.7646	0.1500 $[-0.0507, 0.3507]$	upper 0.10s
Deadline goodput ↑	0.7747	0.7665	-0.0083 $[-0.0175, 0.0010]$	lower -0.03
Fulfillment Jain ↑	0.8565	0.8557	-0.0008 $[-0.0137, 0.0121]$	lower -0.03
P95 response (s) ↓	12.2206	12.6645	0.4439 $[-0.6821, 1.5699]$	upper 5s band
P95 submit delay (s) ↓	0.003292	0.003125	-0.0002 $[-0.0006, 0.0003]$	$[-0.25, 0.25]$s
Phi-3-mini				
Weighted response (s) ↓	6.6939	6.7699	0.0760 $[-0.1635, 0.3155]$	upper 0.10s
Deadline goodput ↑	0.7219	0.7241	0.0023 $[-0.0071, 0.0117]$	lower -0.03
Fulfillment Jain ↑	0.8113	0.8150	0.0036 $[-0.0039, 0.0111]$	lower -0.03
P95 response (s) ↓	16.0937	16.2177	0.1240 $[-0.4024, 0.6504]$	upper 5s band
P95 submit delay (s) ↓	0.003403	0.003223	-0.0002 $[-0.0004, 0.0001]$	$[-0.25, 0.25]$s

Both models meet the prespecified preservation margins for deadline goodput and fulfillment Jain relative to Dense64 (Table 11). The paired goodput intervals have lower bounds of -0.0175 and -0.0071, within the allowed loss of 0.03. P95 response and submission delay also meet their stated margins. Mean weighted response increases by 0.1500 s for Qwen and 0.0760 s for Phi. Its intervals span zero and extend beyond the 0.10 s margin, so this primary endpoint does not establish latency non-inferiority.

Each policy completes all 1,646 requests per model. FCFS calibration selects four concurrent sequences for Qwen2.5-7B ($\overline{\text{goodput}} = 0.593$) and two for Phi-3-mini (0.504), both with 0.15 s arrival spacing and 32 output tokens. 65.7% of Qwen and 76.8% of Phi outputs reach the cap. Latency effects vary across scenarios: the median differences are 0.0048 s for Qwen and -0.0072 s for Phi, while the largest increases are 0.954 s and 1.223 s, respectively.

A.11 Serving under sustained backlog

We compare TERRA-CP(64) and Dense64 on two Azure 2023 arrival windows, at native and fourfold-compressed arrival scales. Each block offers 216 requests from six tenants, with the balanced QA, summarization, and code mix, model, and natural-EOS generation settings in Appendix A.12. The eight compared runs start fresh vLLM servers. The two windows use separate RTX 6000 Ada GPUs, and policies within a window use the same GPU in reversed orders across windows. We fix the inputs, external completion rates, and observation horizon H = last prescribed arrival + 60 s before execution. The four paired blocks describe two source windows, not four independent samples from a workload population.

TERRA-CP and Dense share the reserve, selector, and trace-arrival account epochs. Their internal accounts still depend on active queues and credit first admission, not completed generation. Both

expose one new request per tenant per search and are evaluated against the same external completion reference.

Table 12: Window-level outcomes for the two-window long-output comparison. All 216 requests in every cell meet their deadlines and finish by H, without request errors or output-cap termination. Deficit is the time-average largest completion shortfall in requests. Bold compares TERRA-CP and Dense64 within each block under the same reserve and eligibility. Response starts at prescribed arrival. Realized welfare is in configured utility units, summed over all 216 requests in each run using these response times, whereas the reserve uses admission-time reported welfare.

Window, load	Policy	Deficit ↓	Mean (s) ↓	P95 (s) ↓	Realized welfare ↑
Same welfare reserve and queue eligibility					
1, native	**TERRA-CP(64)**	**0.0417**	**2.961**	**6.781**	**129.30**
	Dense64	0.0461	3.217	7.192	105.28
1, high	**TERRA-CP(64)**	3.6517	**28.476**	**54.322**	**−2269.50**
	Dense64	**3.5454**	28.757	54.845	−2289.45
2, native	**TERRA-CP(64)**	**0.2648**	4.844	**13.710**	−50.22
	Dense64	0.5980	**4.663**	15.657	**−33.45**
2, high	**TERRA-CP(64)**	**3.4992**	**28.984**	54.847	**−2310.41**
	Dense64	3.6783	29.160	**54.779**	−2335.76

TERRA-CP improves mean response in both high-load windows, while the deficit ordering against Dense64 changes between windows. These metrics capture distinct outcomes: response averages completion times across all requests, whereas deficit tracks the tenant furthest behind its target.

The six-tenant queues never trigger the 64-weight cap. On TERRA-CP's visited states, packing leaves 1.52 to 2.25 distinct admission prefixes per search call and the prefix guard leaves 1.14 to 1.52. The small candidate set therefore comes from packing and the reserve check, rather than a truncated search.

A.12 Exact reuse in vLLM

We test whether search reuse reduces controller cost during whole-request execution by comparing two implementations of TERRA-CP(64). Both use the same ranking, integer token-and-slot packing, reserve prefix checks, and admission-account updates. Unlike the grid comparisons, we change only the computation, not the policy's decision rule. Each selector call discards its caches before the next callback.

The workload reuses an Azure 2023 arrival window (Patel et al., 2024) with 216 requests distributed evenly across six tenants. Each tenant receives 12 question-answering, 12 summarization, and 12 code-generation requests from a fixed pool of 12 prompts drawn from LongBench (Bai et al., 2024) and HumanEval (Chen et al., 2021). We stretch native interarrival times by four for low load and compress them by four for high load. The arrival window, assignments, and three load factors were fixed during first-come, first-served (FCFS) calibration. Each load has one pair with packing reuse disabled or enabled. Their execution order alternates across loads. All runs start a fresh Qwen2.5-7B-Instruct server with vLLM 0.8.5 on one RTX 6000 Ada GPU, eight sequence slots, and a 4,096-token budget. Generation uses temperature zero, natural end-of-sequence (EOS) termination, and output caps of 256 tokens for QA and 1,024 for summarization and code. Completion service uses the fixed external reference in Appendix A.8, with equal calibrated rates of 0.3791 requests/s per tenant and observation horizons of 337.36, 129.34, and 77.335 s for low, native, and high load.

All 1,296 requests complete without request errors or output-cap termination. On all 1,259 recorded candidate states, the selectors with reuse disabled and enabled return identical results, including candidate labels, deferred order, reserve prefix checks, and secondary objectives. Three alternating timing pairs per state on an Intel Xeon Silver 4410Y server with Python 3.10 give a 10.6% aggregate reduction in CPU time per replayed selector call, including its diagnostics. CPU savings

are 6.7 to 6.9% on low-load trajectories and 11.7 to 12.1% at high load. Input reconstruction and equality checks are outside timing. The end-to-end measurements in Table 13 instead follow each implementation's own evolving queue.

Table 13: Whole-request execution with packing reuse disabled or enabled under the same TERRA-CP(64) policy. Each load offers the same 216 requests to both implementations. Response time includes admission waiting. The final columns give the time-averaged largest completion-service deficit across tenants, in requests. Lower is better. These are three distinct load pairs, not repeated estimates of one workload.

	Mean response (s)		P95 response (s)		Completion deficit	
Load	No reuse	Reuse	No reuse	Reuse	No reuse	Reuse
Low	1.887	**1.881**	3.745	**3.697**	0.384	**0.382**
Native	3.257	**3.238**	7.457	**7.337**	0.041	**0.040**
High	**28.078**	28.133	**53.672**	53.898	**3.061**	3.383

Queueing and completion order. At high load, requests spend a mean 25.44 s before first admission out of 28.08 s total response time with reuse disabled. The corresponding reuse times are 25.49 s and 28.13 s. Thus about 91% of response time precedes admission, whereas matched-state selector CPU time is about 0.28 ms/call with reuse disabled and 0.25 ms/call with reuse enabled. The saving is too small to directly remove this backlog. The controller repairs debt at admission, whereas the external metric credits service at completion. Search reuse preserves this admission objective. Its completion-service effect therefore depends on queue evolution and execution timing, even when its decisions agree at every fixed state.

At high load, one tenant determines the maximum deficit for 30.25 s instead of 8.41 s. An early summarization request is admitted 18.24 s later despite its post-admission duration falling from 4.64 to 4.50 s. Faster completions for other tenants offset this delay in mean latency but not in the maximum deficit. Total output differs by only 13 tokens out of roughly 19,900, and finish time by 0.008 s. The logs locate a redistribution of completion times, but do not isolate the first cause of the admission-order divergence.

A.13 Comparison with reserve-adapted allocation policies

We implement weighted dominant-share progressive filling following DRF (Ghodsi et al., 2011). Each active workload has a fixed-proportion resource-demand vector capped at one full bundle, with weight ρ_i. Within a round, we increase unfrozen weighted dominant shares together until a demand cap or resource bottleneck is reached. Workloads not using an exhausted resource continue to receive service. Resources are released between rounds, so this allocation concerns current resource shares rather than accumulated service deficits.

Two adaptations align DRF with the evaluation constraints. To satisfy the minimum positive service fraction of 0.20, we remove the smallest below-floor fraction and refill until the allocation is feasible, with a fixed per-round tie order. Banked DRF then selects this allocation whenever it passes the shared exact welfare reserve, and uses the common greedy allocation otherwise. The added floor and reserve are adaptations, not properties of the original DRF guarantee.

VTC instead prioritizes the workload with the smallest weighted token-service counter (Sheng et al., 2024). We adapt this rule to fractional rounds by charging token service from the allocation actually delivered, divided by the workload's target weight. Newly active workloads are lifted to the active counter floor. The estimated prefill fraction is 0.35, with input and output token weights of 1 and 2. Banked VTC accepts its proposal if the common reserve allows it and otherwise selects greedy, charging the chosen allocation in either case. These round-level adaptations use the same trajectories as DRF, rather than the per-token events used by the original serving scheduler.

Table 14: Published allocation policies on twelve fixed fractional trajectories. Bold marks each column's best mean among policies meeting the 90% welfare requirement. Teal identifies ours. Realized welfare is shown as a percentage of greedy. CPU time is per allocation round on the same machine, with VTC measured in a separate pass.

Policy	Max. deficit / round ↓	Min. fulfillment (%) ↑	Realized welfare ↑ (% of greedy)	CPU time ↓ (ms/round)
Policies meeting the 90% welfare requirement				
TERRA-CP(64)	**0.03792**	**69.50**	90.005	29.63
TERRA-CP(8)	0.04080	67.05	90.007	12.42
Banked DPP	0.05343	66.76	90.276	2.79
Banked DRF	0.15131	19.82	90.040	2.45
Banked VTC	0.12868	37.28	90.038	2.59
Greedy	0.21275	0.00	**100.000**	**0.54**

All methods use the twelve fixed external trajectories of the budget study, without additional tuning. DPP retains its independently selected $V_{\mathrm{DPP}} = 1$. The service-floor repair removes 0.220% of active allocations and affects 6.325% of rounds. Banked DRF retains its proposal in 20.55% of rounds. Relative to banked DRF, TERRA-CP(8) and TERRA-CP(64) reduce maximum deficit by 0.11051 (95% paired interval [0.09788, 0.12315]) and 0.11339 ([0.10077, 0.12601]), respectively. The corresponding minimum-fulfillment gains are 47.24 and 49.68 percentage points. Relative to DPP, TERRA-CP(8) and TERRA-CP(64) reduce deficit by 0.01263 ([0.00610, 0.01916]) and 0.01551 ([0.00866, 0.02236]), respectively. These comparisons use paired t intervals across trajectories. The runs use an Intel Xeon W-2235 under Windows 11 and Python 3.12, with policy order rotated across trajectories. Process-CPU timing includes allocation construction and the reserve check, but excludes trace generation and post-allocation observation.

Banked VTC retains its token-priority proposal in 22.85% of rounds. TERRA-CP(8) and TERRA-CP(64) reduce its deficit by 0.08788 (95% paired interval [0.07288, 0.10288]) and 0.09076 ([0.07600, 0.10551]). Both also improve minimum fulfillment on all twelve trajectories. VTC timings come from a separate pass on the same machine, using the same allocation-call boundary.

A.14 DPP CALIBRATION AND SELECTOR ABLATION

We use banked DPP to compare TERRA-CP(64) with a debt-weighted scheduler that does not search multiple weights. It shares the fractional workload, reflected service accounts, resource packer, and 0.90 welfare reserve. It ranks full bundles by $(Q_i\widehat{s}_i + V_{\mathrm{DPP}}\widehat{V}_i)/b_i$, skips nonpositive scores, and packs one order. Among that allocation and the greedy reference satisfying the reserve, it maximizes $\sum_i Q_i\widehat{a}_i(X) + V_{\mathrm{DPP}}\widehat{W}(X)$, breaking ties by welfare and then the allocation signature. This is a density-packing approximation to the DPP subproblem, using raw debt and value without TERRA-CP's normalizations or maximum-debt selector.

We choose V_{DPP} from $\{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100, 1000\}$ on five development trajectories, minimizing mean maximum cumulative deficit per round, then maximizing minimum fulfillment and preferring smaller V_{DPP}. The resulting $V_{\mathrm{DPP}} = 1$ is fixed for the held-out comparisons in Appendices A.13 and A.15.

Reducing repeated candidate-evaluation work. Within each allocation call, we cache request-level utility, service fraction, tie keys, affine coefficients, and debt terms that are shared by its candidates. The cache is discarded at the next decision. Packing, candidate weights, arithmetic and summation order, reserve checks, and selection remain unchanged. On 100 fixed states from five fractional TERRA-CP(64) trajectories, 900 timed pairs of uncached and cached calls give identical allocations and post-call states. Mean allocation CPU time falls from 46.20 to 29.58 ms/call for TERRA-CP(64) and from 49.01 to 33.54 ms/call for Dense136. DPP remains at 2.45 ms/call. Three counterbalanced repetitions include cache setup but exclude state restoration and equality checks. These Xeon W-2235 measurements concern per-decision evaluation reuse. The additional

order-level optimization is evaluated separately in Appendix A.26. The budget comparison gives all methods this same evaluation cache.

Isolating the repayment tie-break. To test the contribution of the repayment tie-break, we remove only $P_t(X)$ from TERRA-CP(64)'s lexicographic selector on 12 independent held-out trajectories with 100 workloads and 1,000 rounds. This cohort is separate from the 12-trajectory budget and published-policy comparison. It now selects by $(-M_t(X), \widehat{W}_t(X), \mathrm{sig}(X))$, retaining the same candidates, packer, reserve, debt update, and external inputs. The intervention is fixed before examining these outcomes. The two arms share each external trace but maintain their own debt and reserve states. Greedy on each trace supplies the realized-welfare denominator.

Table 15: Repayment tie-break ablation over 12 paired traces. Differences are TERRA-CP(64) minus TERRA-CP(64) without repayment, with paired 95% t intervals.

Metric	No repayment	**TERRA-CP(64)**	Difference [95% CI]
Maximum deficit / round ↓	0.03761	**0.03683**	-0.00079 $[-0.00459, +0.00302]$
Minimum fulfillment (%) ↑	68.67	**71.83**	$+3.17$ $[-0.16, +6.49]$
Realized-welfare retention ↑	**0.90039**	0.90001	-0.00038 $[-0.00070, -0.00006]$

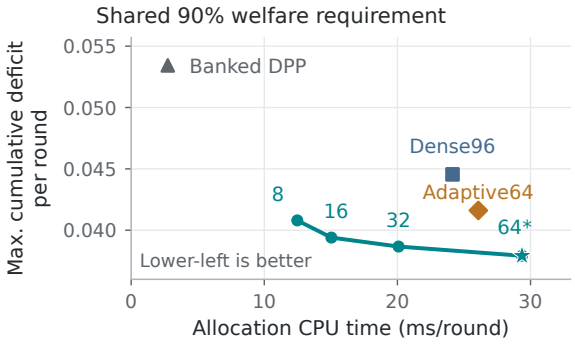
Adding the repayment tie-break raises mean minimum fulfillment by 3.17 percentage points and lowers mean deficit by 0.00079 per round, while realized-welfare retention falls by 0.038 percentage points. Maximum deficit improves in seven pairs and minimum fulfillment in nine, with both fairness intervals including zero. Realized-welfare retention is lower in all twelve pairs. This ablation measures the service and welfare tradeoff induced by the secondary selector, separately from the candidate-search comparisons.

A.15 Allocation quality under smaller search budgets

To measure the tradeoff between search cost and allocation quality, we evaluate TERRA-CP(8), TERRA-CP(16), TERRA-CP(32), and TERRA-CP(64) on 12 paired trajectories with 100 workloads, 1000 rounds, minimum positive service 0.20, and a 0.90 welfare reserve. Every policy receives the same external trace and maintains its own allocation history. TERRA-CP, fixed-grid, adaptive-grid, and banked DPP implementations share the request-level calculation reuse in Appendix A.14. Caching leaves each policy's packing, numerical precision, and selector unchanged.

We choose the budget on three 500-round development trajectories, using the smallest TERRA-CP budget that retains at least 80% of TERRA-CP(64)'s mean deficit improvement over DPP and has realized-welfare retention at most 0.002 below TERRA-CP(64). Only TERRA-CP(64) meets this rule. Dense96 and Adaptive64 have the closest development CPU costs in their respective budget pools. They are not presumed runtime-equivalent. Banked DPP uses $V_{\mathrm{DPP}} = 1$, fixed by the independent calibration in Appendix A.14. All four TERRA-CP budgets are specified before evaluation and are reported in Table 16. TERRA-CP(16) is a point on this complete curve, not the development-selected configuration.

Figure 5 plots the quality-cost curve. Dense96 uses 96 fixed weights, and Adaptive64 uses 64 queue-adaptive weights. Timings come from a separate pass from Table 14. Table 16 and the paired intervals below retain the full service and welfare comparison.



Policy	Deficit	ms/round
TERRA-CP(8)	0.04080	12.48
TERRA-CP(16)	0.03940	15.04
TERRA-CP(32)	0.03867	20.09
TERRA-CP(64)*	**0.03792**	29.36
Dense96	0.04455	24.14
Adaptive64	0.04161	26.08
Banked DPP	0.05343	**2.75**

* Default budget. Labels on the curve give K.

Figure 5: Service quality and computation under the same 90% welfare requirement. Means over 12 paired fractional trajectories, with lower-left preferable. All four TERRA-CP budgets were fixed before evaluation. The table gives absolute values from this timing cohort.

Table 16: The complete budget curve on 12 independent fractional trajectories. Entries are means. Lower deficit and CPU are preferable. Higher fulfillment and realized-welfare retention are preferable. Retention is relative to the independently evolving greedy policy on the same trace. Dense96 and Adaptive64 use 96 fixed and 64 queue-adaptive weights, respectively.

Policy	Max. deficit / round	Min. fulfillment (%)	Realized welfare (% of greedy)	CPU time (ms/round)
TERRA-CP(8)	0.04080	67.05	90.007	12.48
TERRA-CP(16)	0.03940	68.27	90.004	15.04
TERRA-CP(32)	0.03867	68.93	90.004	20.09
TERRA-CP(64)	**0.03792**	**69.50**	90.005	29.36
Dense96	0.04455	64.40	90.011	24.14
Adaptive64	0.04161	66.65	90.010	26.08
Banked DPP	0.05343	66.76	90.276	2.75
Greedy	0.21275	0.00	**100.000**	**0.61**

For budget K, define the retained mean benefit as $(\bar{D}_{\mathrm{DPP}} - \bar{D}_K)/(\bar{D}_{\mathrm{DPP}} - \bar{D}_{64})$, where $\bar{D}$ is mean maximum cumulative deficit per round. It is 81.4%, 90.4%, and 95.2% for TERRA-CP(8), TERRA-CP(16), and TERRA-CP(32). Their CPU savings relative to TERRA-CP(64) are 57.5%, 48.8%, and 31.6%.

TERRA-CP(16)'s deficit reductions relative to Dense96 and Adaptive64 are 0.005145 (95% paired interval [0.003606, 0.006684]) and 0.002212 ([0.001203, 0.003220]). Its fulfillment increases are 3.86 percentage points ([2.69, 5.04]) and 1.62 points ([0.79, 2.45]). Relative to DPP, its deficit reduction is 0.01402 ([0.00745, 0.02060]). Its fulfillment difference is 1.51 points ([-3.82, 6.83]). Realized-welfare retention changes by -0.272 points ([-0.681, 0.137]) relative to DPP, -0.0068 points ([-0.0134, -0.0002]) relative to Dense96, and -0.0065 points ([-0.0106, -0.0025]) relative to Adaptive64. Intervals are endpoint-wise paired t intervals over trajectories, not over allocation rounds.

All configurations run serially on an Intel Xeon W-2235, with policy order rotated across trajectories. Process-CPU time includes the allocation call, per-decision reuse setup, and score-crossing construction. It excludes trace generation and post-allocation observation. Elapsed scheduler means are 15.15, 29.60, and 2.76 ms/round for TERRA-CP(16), TERRA-CP(64), and DPP. This cohort is separate from the repayment-selector ablation in Appendix A.14.

A.16 DEBT-WEIGHTED SCORES AND THE ADMISSION OBJECTIVE

TERRA-CP separates proposing allocations from choosing one. The ranking score generates candidates that trade debt repayment against value. The selector compares their maximum projected debt under the welfare requirement. A quadratic potential motivates debt-weighted service in the ranking. Let $L(Q) = \frac{1}{2}\sum_i Q_i^2$ and, for an active workload, set $u_i = \rho_i - \widehat{a}_i^t(X) \in [-1, 1]$. Since $[Q_i^t + u_i]_+^2 \le (Q_i^t + u_i)^2$,

$$\frac{[Q_i^t + u_i]_+^2 - (Q_i^t)^2}{2} \le Q_i^t \rho_i - Q_i^t \widehat{a}_i^t(X) + \frac{u_i^2}{2}.$$

Inactive coordinates are unchanged. Summing and using $u_i^2 \le 1$ gives

$$L(\widehat{Q}^{t+1}(X)) - L(Q^t) \le \frac{|\mathcal{A}_t|}{2} + \sum_{i \in \mathcal{A}_t} Q_i^t \rho_i - \sum_{i \in \mathcal{A}_t} Q_i^t \widehat{a}_i^t(X). \tag{10}$$

At a given queue state, the first two terms are fixed. More debt-weighted service therefore lowers this bound. Combining that service term with normalized value motivates

$$S_\nu(X) = \sum_{i \in \mathcal{A}_t} Q_i^t \widehat{a}_i^t(X) + \nu \frac{\widehat{W}_t(X)}{s_t}, \qquad s_t = \max\{\sigma_t, 10^{-12}\}.$$

For full-bundle indicators $z_i \in \{0, 1\}$, additive full-bundle value, and $\widehat{a}_i^t(X) = z_i \widehat{s}_i^t$, this becomes $S_\nu(X) = \sum_i z_i (Q_i^t \widehat{s}_i^t + \nu \widehat{V}_i^t / s_t)$. Dividing each coefficient by b_i^t yields Equation (3). The division and multi-resource greedy packing are heuristics for turning these priorities into feasible candidates. They do not maximize S_ν in general. For partial allocations, the nonlinear utility above is not a fixed full-bundle value multiplied by a service fraction. We therefore evaluate each packed candidate using its allocation-specific service and utility.

Thus ν explores candidate orders, while the reserve enforces welfare. Appendix A.23 tests the default resource weight $\omega = 0.5$ while keeping the welfare reference fixed.

A.17 CHECKING SEARCH COVERAGE AND ALLOCATION QUALITY

We check whether enumeration covers its ranking family and how close that family is to the best feasible allocation. For coverage, we compare uncapped search with independent rational-arithmetic enumeration on 1,000 states. Eligible-set size has median 6 (range 1 to 13). Positive crossing count has median 2 and maximum 29. The complete list has at most 60 representatives, so $K = 64$ never binds. The two implementations produce exactly the same allocation sets and positive-event counts on every state. Both use the 12-significant-digit canonical coefficients and the same floating-point packer. This checks enumeration after canonicalization, not the rounding of the input scores.

We separately enumerate all resource- and reserve-feasible binary subsets. This oracle evaluates the restriction imposed by ranked-greedy packing, rather than numerical implementation. Table 17 shows that $K \ge 16$ adds no primary cap loss in this sample, although the complete family does not always attain the binary optimum. At $K = 8$, truncation increases the primary gap on two states, by 0.0000619 on average over all states.

Table 17: Small-state comparison with the exact binary-allocation optimum. "Primary optimal" means attaining the smallest maximum projected debt. "Lex. optimum" also matches the subsequent objectives in Equation (8). Intervals are 95% half-widths over five trajectory means. The final row applies to every tested budget $K = 16, 32, 64, 128, \infty$.

Comparison	Primary optimal (%)	Lex. optimum (%)	Oracle gap
TERRA-CP(8)	83.6 ± 2.2	69.6 ± 12.4	0.010 ± 0.005
TERRA-CP(16) to uncapped	83.8 ± 2.1	69.8 ± 12.5	0.0100 ± 0.0055

The uncapped family gap has mean 0.009970 ± 0.005487, pooled 95th and 99th percentiles 0.060440 and 0.144157, and observed maximum 0.812706. Figure 6 groups the states by eligible-set size (287,

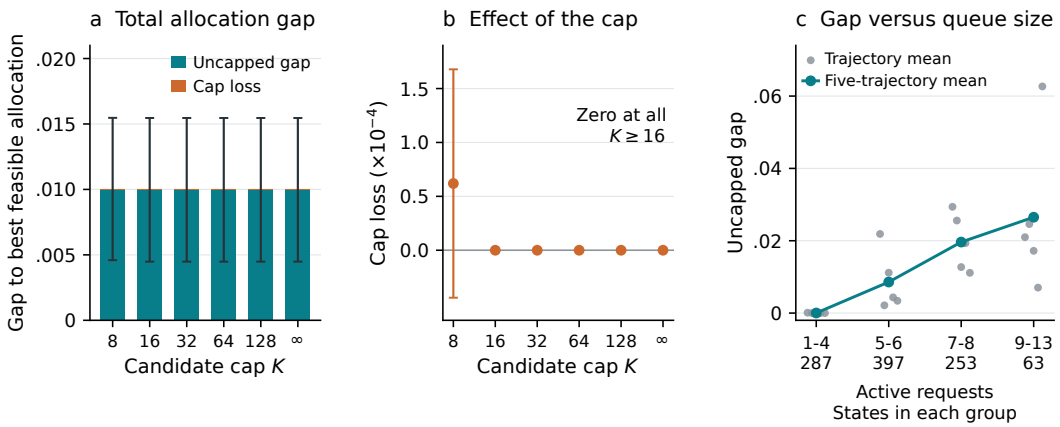


Figure 6: Separating two sources of excess maximum debt. (a) The gap to the best feasible allocation combines the uncapped family gap (teal) and cap loss (orange). (b) Cap loss on its own scale. It is zero for $K \geq 16$ on these 1,000 binary states. Bars in (a,b) are 95% t intervals over five trajectory means. (c) Gray points are trajectory means, teal points average them, and counts give states per group. Grouping is descriptive.

397, 253, and 63 states). Group means weight the five trajectories equally. This descriptive grouping does not isolate set size from other queue conditions.

A matched-grid check reconstructs the same five uncapped-driven trajectories. Dense64 uses zero and 63 log-spaced weights in $[10^{-6}, 10^{6}]$, with the same coefficients, binary packer, reference, reserve, and selector. It produces 2.114 distinct allocations per state versus 2.120 for TERRA-CP(64), yet Dense64, TERRA-CP(64), and uncapped search have identical primary objectives on all 1,000 states. All three attain the binary primary optimum in $83.8 \pm 2.1\%$ of states and the full lexicographic optimum in $69.8 \pm 12.5\%$. Thus the additional orders do not improve maximum projected debt on these whole-request states.

A.18 Optimization under the actual allocation constraints

The certificate in Appendix A.29 relaxes the reserve and minimum service. We restore both to compare the same 100 states with optimization beyond the ranking family. For complete bundles, binary variables y_i select requests. We minimize M subject to

$$\sum_i \widehat{d}_{ir} y_i \leq C_{tr}, \qquad \sum_i \widehat{V}_i y_i \geq r_t - B_t, \qquad M \geq Q_i^t + \rho_i - y_i,$$

with $M \geq 0$ and every inactive debt as an additional lower bound. This mixed-integer program (MIP) searches all resource- and reserve-feasible subsets, not just those produced by ranking. Each packing mode recomputes its own greedy reference. The formulation matches the simulator's atomic allocation. The serving bridge additionally checks every accepted prefix.

For fractional service, binary indicators enforce zero service or at least 0.20, and service equals the minimum of the three allocated resource ratios. The nonlinear utility contains the product of GPU ratio, bandwidth ratio, and exponential token quality. We retain the original latency and service costs and relax only these nonlinear products using McCormick envelopes and quality-function tangents at $0, 0.2, \ldots, 1$. Every originally feasible allocation remains feasible in this relaxed problem, so its optimum is a valid lower bound on maximum debt. We separately check whether a returned allocation also satisfies the original nonlinear welfare requirement.

Table 18: Comparison beyond the ranking family under reserve and service constraints. Numerical primary certification uses tolerance 10^{-6}. Full-bundle gaps are to the solved subset optimum. Fractional entries are gap upper bounds from the outer relaxation.

Allocation	Quantity	TERRA-CP(64)	Dense128
Full bundle	Primary optimal (states)	**59 / 100**	53 / 100
	Mean optimality gap	**0.21728**	0.26861
Fractional	Primary certified (states)	**60 / 100**	51 / 100
	Mean gap upper bound	**0.23317**	0.29609

HiGHS solves all 100 full-bundle programs within five seconds each, and all recovered subsets pass the original utility, capacity, and strict reserve checks. TERRA-CP(64) reaches the subset optimum more often than Dense128, but its remaining gap shows where changing the packing family could help. Of the fractional programs, 98 finish within the same limit. We retain the two timeouts and use valid solver lower bounds for all 100. Ten returned allocations satisfy the original reserve check. For the other 90 states, we use only the solver's lower bound, not its allocation, in the comparison. The mean TERRA-CP(64) gap bound tightens from 0.23466 to 0.23317 and remains unchanged on the 18 active-limited states. The modest tightening, together with the limited feasibility of the returned allocations, points to residual slack in the nonlinear relaxation.

A.19 When a search cap changes the decision

We measure how much allocation diversity is lost at $K = 64$, then whether that loss changes maximum debt. We first compare TERRA-CP(64) with uncapped enumeration on 100 matched fractional states. Eligible-set size has median 41.5 (range 26 to 56), while positive crossings have median 702.5 and maximum 1,300. The complete list contains 1398.4 ± 114.3 representatives on average, and $K = 64$ binds in 95 states. TERRA-CP(64) retains a fraction 0.489 ± 0.038 of the 108.3 ± 12.2 distinct uncapped allocations and selects the same allocation in 0.580 ± 0.094 of states. Uncapped search achieves 3.539 ± 2.854 more normalized repayment. This sample has no maximum-debt gap above 10^{-12}. The 30,000-state comparison below resolves less frequent losses. We separate the larger sample by what limits debt repair. In inactive-floor states, an inactive workload's debt sets a lower bound that current admissions cannot reduce. In active-limited states, the limiting workload is active but can receive at most one unit of service.

Table 19: TERRA-CP(64) matches the uncapped maximum debt in 98.62% of 30,000 matched states at $N = 100$. Rates average within-trajectory rates over 30 trajectories, whereas state counts are pooled. Gaps larger than 10^{-12} occur in 1.22% and are concentrated in active-limited states. Nested budgets add representatives to the same diagnostic sequence. They are distinct from the independently spaced production TERRA-CP(128) and TERRA-CP(256) policies. The inactive-floor group includes the 30 initial zero-floor ties.

Statewise comparison	Mean [95% CI]	States
Same maximum projected debt	98.623% [98.411, 98.836]	29,587/30,000
Maximum-debt gap above tolerance	1.220% [1.024, 1.416]	366/30,000
Gap in active-limited states	3.146% [2.649, 3.643]	264/8,782
Gap in inactive-floor states	0.455% [0.311, 0.599]	102/21,218
K64 gaps recovered by nested budget 128	50.536% [44.036, 57.037]	184/366
K64 gaps recovered by nested budget 256	77.933% [72.980, 82.886]	288/366

The 30,000-state comparison shows where the cap matters. When an inactive account's debt fixes the lowest achievable maximum debt, above-tolerance gaps occur in 0.455% of states. When active requests determine that bound, the rate rises to 3.146%. Adding representatives from the same ordered diagnostic sequence recovers 50.5% of TERRA-CP(64)'s above-tolerance failures at nested budget 128 and 77.9% at nested budget 256.

In a separate 30-trajectory budget comparison, mean terminal cumulative deficit was 45.672 for TERRA-CP(64), 45.113 for TERRA-CP(128), and 44.947 for TERRA-CP(256). Minimum fulfillment was 63.609%, 64.119%, and 64.108%, respectively. Moving from 64 to 128 representatives reduced deficit by 0.559 (95% interval $[0.247, 0.872]$) and increased fulfillment by 0.51 percentage points ($[0.29, 0.73]$). Moving from 128 to 256 changed deficit by -0.166 ($[-0.451, 0.120]$) and fulfillment by -0.01 percentage points ($[-0.24, 0.22]$). These production budgets use nonnested representative sets.

A.20 Full-bundle robustness

To test whether search gains persist when requests must be served whole, we evaluate ten strata covering the baseline, offered load, tenant count, service target concentration, Markov load modulation, prediction error, correlation between value and demand, value tails, output length, and two prespecified interactions, for 24 configurations in total. Each tenant maintains a first-in, first-out request queue. Only its first waiting request is eligible for admission, which reserves the complete GPU, token, and bandwidth bundle. We measure service over 128 externally specified epochs shared by all policies. We drain requests remaining after this 128-epoch measurement window to completion, excluding their later service from external-horizon metrics.

Thirty-two sampled workloads each cover 24 configurations and eight policies. For each workload and endpoint, TERRA-CP(64)-minus-Dense64 differences are averaged equally within each stratum and then equally across the ten strata. Thus the inference unit is the sampled workload ($n = 32$), not the 768 combinations of workloads and configurations. Intervals use 20,000 cluster-bootstrap resamples, and exact two-sided sign tests are Holm-adjusted over the three primary metrics.

Table 20: Whole-request outcomes across 24 configurations. Means weight configurations equally within each of ten strata, then weight strata and the 32 sampled workloads equally. Differences are TERRA-CP(64) minus Dense64, with paired 95% cluster-bootstrap intervals. Favorable counts workloads with an improvement. Fulfillment is the uncapped delivered-service-to-target ratio, so it can exceed one.

Endpoint	Dense64	**TERRA-CP(64)**
Reflected debt / $T \downarrow$	0.022070	**0.021727**
Active wait / $T \downarrow$	**0.370028**	0.370593
Minimum fulfillment $\uparrow$	**2.034496**	2.032796

Endpoint	Difference 95% CI	Favorable	Holm p
Reflected debt / $T \downarrow$	-0.000343 $[-0.001111, +0.000282]$	16/32	1.000
Active wait / $T \downarrow$	$+0.000566$ $[-0.003208, +0.004053]$	14/32	1.000
Minimum fulfillment $\uparrow$	-0.001700 $[-0.018820, +0.015230]$	16/32	1.000

Mean differences are close to zero, with all three intervals spanning zero (Table 20). A fractional feasibility check uses each policy's realized head-of-line queue. Targets are feasible on 611 of 768 TERRA-CP(64) paths and 618 of 768 Dense64 paths. The remaining 157 and 150 paths require positive target slack. This check describes each policy's path under divisible service, rather than a common integral schedule.

A.21 Why whole-request packing reduces search gains

We ask whether richer rankings still produce more choices when requests must be served whole. We reconstruct the first 191 rounds of each of the first five TERRA-CP(64) trajectories in the saved fractional certificate study and sample rounds $0, 10, \ldots, 190$, giving 100 fixed states. All 955 reconstructed TERRA-CP(64) pre-decision outcomes and 100 sampled Dense128 outcomes match the

saved objective values, welfare, and candidate counts. Counterfactual allocations do not drive the reconstructed trajectory.

At each sampled state, we hold requests, debts, reserve balance, capacities, affine coefficients, and tie keys fixed, changing the minimum positive service fraction from the existing 0.20 to full-bundle admission at 1.0. Each mode recomputes the greedy reference with its own packer, so the welfare guard has the same meaning within that mode. Both packing modes are evaluated at the same states visited by the fractional policy, isolating how indivisibility changes the available choices.

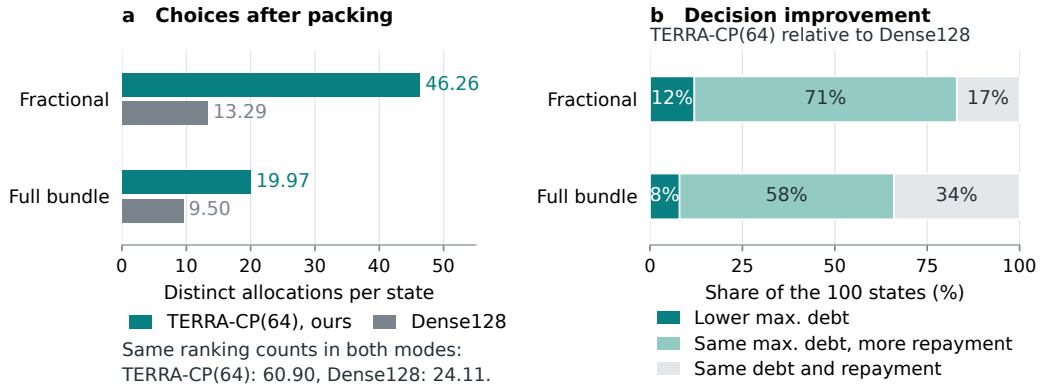


Figure 7: Whole-request packing reduces choices even when rankings are unchanged. Both modes use the same 100 fractional-policy states. (a) Distinct allocations before the reserve check. (b) TERRA-CP(64)'s selected allocation versus Dense128: lower maximum debt, more repayment at equal debt, or equality on both (tolerance 10^{-9}). Each mode uses its own greedy reference.

Table 21: Whole-request packing merges candidate allocations despite unchanged ranking diversity. Means and counts cover the same 100 states. Objective ties use a 10^{-9} tolerance. All selected allocations satisfy the corresponding capacity and welfare constraints.

Quantity	Fractional	Full bundle
TERRA-CP(64) distinct orders, mean	60.90	60.90
Dense128 distinct orders, mean	24.11	24.11
TERRA-CP(64) distinct generated allocations, mean	46.26	19.97
Dense128 distinct generated allocations, mean	13.29	9.50
TERRA-CP(64)-exclusive reserve-feasible allocations, mean	9.37	2.84
States with different selected allocations	89/100	66/100
TERRA-CP(64) primary better / equal / worse	12 / 88 / 0	8 / 92 / 0
TERRA-CP(64) repayment better / equal / worse on primary ties	71 / 17 / 0	58 / 34 / 0

Full-bundle admission merges more rankings into the same allocation, reducing search opportunities. The remaining improvements in maximum debt and repayment show that indivisibility alone does not remove the benefit in these states. The serving experiments also differ from the fractional simulator in queue size, service accounting, and subsequent trajectory.

Search on native whole-request trajectories. We examine all 128 external epochs in each of the first five evaluation trajectories generated by whole-request TERRA-CP(64) admission. The 640 states contain a median of 14 eligible requests (maximum 32). TERRA-CP(64) advances each queue and reserve. Dense64 is evaluated on the same states with the same packing rule and service accounts. These probes leave the original admission and completion transcripts unchanged.

Table 22: Whole-request queues offer few distinct reserve-safe choices. Means cover 640 external-epoch states on five existing TERRA-CP(64) paths. Rankings and allocations include the common reference. Allocation identity ignores request order. Dense64 is evaluated on the TERRA-CP(64) state, not its own trajectory.

Quantity	**TERRA-CP(64)**	Dense64
Distinct orders	42.09	14.98
Distinct packed allocations	6.09	5.07
Distinct reserve-safe allocations	1.33	1.29
Exclusive packed allocations	1.02	0.002
Exclusive reserve-safe allocations	0.034	0

TERRA-CP(64) has an exclusive packed allocation in 330 states, but an exclusive reserve-safe allocation in only 19. It selects an exclusive alternative in 11 states, improving projected maximum debt in five and repayment at the same maximum debt in six. The selected request sets agree in the remaining 629 states. None has a worse TERRA-CP(64) primary objective. The 11 changed decisions are spread across all five paths. Ranking diversity therefore passes through two filters before it can affect service: orders must yield different packed allocations, and the welfare reserve must permit those alternatives.

A.22 Candidate search and selector interaction

To separate the roles of proposing and choosing allocations, we cross candidate generation with the selection objective on the 100 fixed states in Appendix A.21. Each family uses the same fractional packer, 0.20 minimum service, greedy reference, and 0.90 welfare reserve. We generate each candidate list once and apply either Equation (8) or the raw DPP objective $\sum_i Q_i^t \widehat{a}_i^t(X) + V_{\mathrm{DPP}} \widehat{W}_t(X)$, with the previously selected $V_{\mathrm{DPP}} = 1$. The families are TERRA-CP and Dense at budgets 16 and 64, and the tuned DPP ranking plus the common greedy reference. Equal budgets isolate the components here. The trajectory studies measure CPU tradeoffs.

Table 23: Candidate search and selection on the same 100 states. Dense(K) and TERRA-CP(K) share the budget and selector in each row. Absolute entries are mean maximum projected debt M. Differences are TERRA-CP minus Dense. Better, tied, and worse count states relative to the equal-budget grid.

K	Selector	Dense M	**TERRA-CP** M	ΔM	Better / tied / worse
16	Maximum debt	6.64218	6.54676	−0.09542	21 / 79 / 0
16	DPP objective	6.64453	6.64334	−0.00119	1 / 99 / 0
64	Maximum debt	6.61502	6.52695	−0.08807	21 / 79 / 0
64	DPP objective	6.64782	6.64334	−0.00448	2 / 98 / 0

TERRA-CP(16) offers 4.61 distinct reserve-feasible allocations per state, versus 1.93 for Dense16. At budget 64 these counts are 13.27 and 3.56. With the maximum-debt selector, TERRA-CP chooses an allocation absent from the equal-budget grid in 91 states at either budget. For the candidate set formed by the DPP order and greedy reference, changing the selector changes 25 allocations but leaves maximum projected debt unchanged. In 36 states, only one candidate satisfies the welfare reserve. The selector can exploit a useful alternative only if candidate generation supplies it.

The two objectives make different use of welfare surplus. Within TERRA-CP(16)'s candidate set, the maximum-debt selector lowers mean reported welfare from 353.49 to 326.03 relative to DPP selection while satisfying the same reserve. Conversely, holding the DPP objective fixed, TERRA-CP(64) improves that objective over Dense64 in 29 states, ties in 69, and worsens it in two.

A.23 RESOURCE-NORMALIZATION SENSITIVITY

To test whether the search advantage depends on resource normalization, we vary only the ranking mixture on the same 100 states in Equation (5). The greedy reference retains its default $\omega = 0.5$, so each comparison faces exactly the same welfare requirement. TERRA-CP(64) and Dense128 retain their original budgets and packer. These probes do not advance the trajectory or retune runtime.

Table 24: Resource-normalization sensitivity on the same 100 states. Absolute entries are mean maximum projected debt M. Differences are TERRA-CP(64) minus Dense128. Better, tied, and worse count states relative to Dense128. Repayment wins use only maximum-debt ties.

ω	Dense128 M	**TERRA-CP(64)** M	ΔM
0.25	6.56811	6.51192	-0.05619
0.50	6.58987	6.52695	-0.06292
1.00	6.58403	6.56075	-0.02329

ω	Better / tied / worse	Repayment wins in ties
0.25	15 / 84 / 1	64 / 84
0.50	12 / 88 / 0	71 / 88
1.00	3 / 97 / 0	76 / 97

At $\omega = 1$, the score ignores demand beyond the bottleneck resource. The mean primary improvement is smaller, although repayment still improves in 76 primary ties. Relative to the default, TERRA-CP(64) changes its allocation in 94 states at $\omega = 0.25$ and 97 at $\omega = 1$. The mean one-step advantage persists across these weights, while the selected allocations remain sensitive to normalization. Maximum debt captures only part of that decision change.

A.24 SENSITIVITY TO WORKLOAD AND WELFARE REQUIREMENTS

To test sensitivity to the operating point, we use the fixed 128-weight grid Dense128 and cross the default workload and a more congested shifted workload with welfare floors 0.88, 0.90, and 0.92. We average effects equally across these six settings within each paired trajectory. Relative to Dense128, every seed-level aggregate favors TERRA-CP(64) on both maximum cumulative deficit per round and minimum fulfillment. The setting-specific debt intervals favor TERRA-CP(64) in five of six cases, while the default workload accounts for the clearest minimum-fulfillment gains.

Against equal-budget Dense64, the corresponding aggregate differences are -0.004174 for maximum cumulative deficit per round and $+0.01696$ for minimum fulfillment. Under the congested workload at floor 0.92, the minimum-fulfillment effect reverses to favor Dense64.

Table 25: Absolute outcomes across two workloads and three welfare floors. Means weight the six settings equally within each of 30 paired trajectories. Differences are TERRA-CP(64) minus Dense128. The upper panel gives paired 95% intervals for equal-setting effects. The lower panel retains Bonferroni-adjusted 95% intervals over the six settings within each endpoint. Fulfillment and retention differences are percentage points. Preservation margins are -0.01 for Jain and -0.20 pp for retention.

Metric	Dense128	**TERRA-CP(64)**	Difference 95% CI
Deficit / round ↓	0.10352	**0.09994**	-0.00358 $[-0.00415, -0.00301]$
Minimum fulfillment (%) ↑	34.185	**35.702**	$+1.517$ $[+1.135, +1.900]$
Fulfillment Jain index ↑	0.753942	**0.754512**	$+0.000570$ $[-0.000025, +0.001164]$
Realized-welfare retention (%) ↑	**90.0144**	90.0023	-0.0121 $[-0.0137, -0.0105]$

Workload	Welfare floor	Δ deficit / round [95% CI]	Δ fulfillment (pp) [95% CI]
Default	0.88	-0.00495 $[-0.00661, -0.00329]$	$+3.624$ $[+2.432, +4.817]$
Default	0.90	-0.00601 $[-0.00724, -0.00478]$	$+3.939$ $[+3.086, +4.793]$
Default	0.92	-0.00487 $[-0.00674, -0.00300]$	$+2.113$ $[+0.880, +3.346]$
Shifted	0.88	-0.00256 $[-0.00446, -0.00066]$	$+0.225$ $[-0.933, +1.383]$
Shifted	0.90	-0.00269 $[-0.00431, -0.00107]$	-0.104 $[-0.886, +0.678]$
Shifted	0.92	-0.00040 $[-0.00221, +0.00142]$	-0.695 $[-1.542, +0.152]$

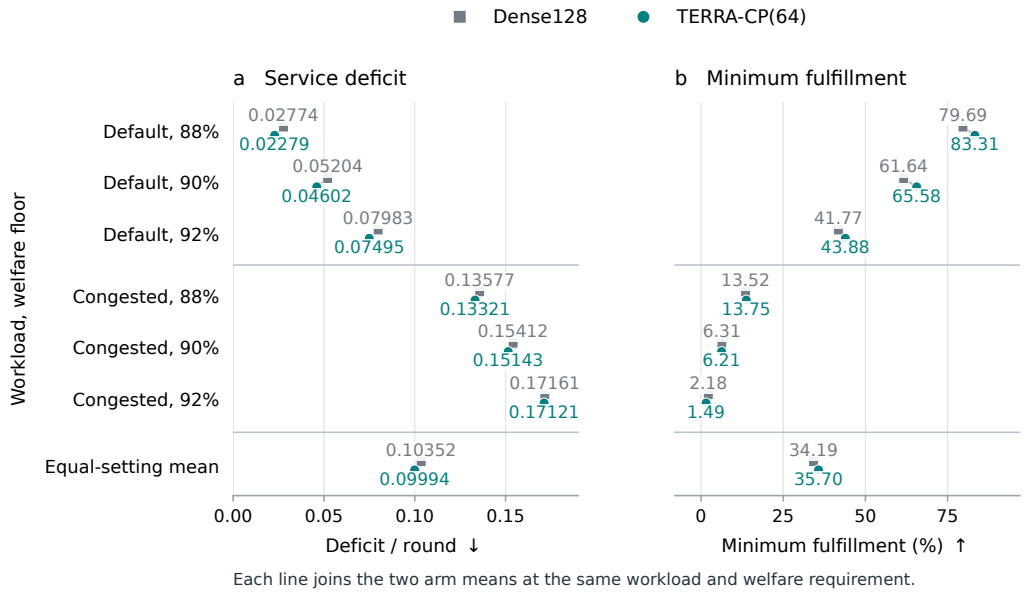


Figure 8: Absolute service outcomes across workload and welfare-floor settings. Gray squares show Dense128, and teal circles show TERRA-CP(64). Lines connect means over 30 paired trajectories within each setting. The last row averages the six settings equally. TERRA-CP(64) lowers mean deficit in every setting, with larger fulfillment gains in the default workload.

A.25 Fractional workload feasibility

We use a continuous relaxation to test whether resource capacity alone can support every tenant's aggregate active-round service target over the finite horizon. It seeks a policy-independent proportional allocation. For active tenant i, let $x_{it} \in [0, 1]$ be the common fraction of its reported three-resource bundle served in round t. The optimization minimizes nonnegative service target slack subject to the capacity constraint for every resource in every round and $\sum_{t:i\in A_t} x_{it} + s_i \geq \rho_i h_i$.

All 70 sampled trajectories have zero shortfall in this relaxation. Maximum resource utilization is 0.719 in ten development trajectories, 0.739 in ten post-selection trajectories, 0.775 in 30 independent Dense128-comparison trajectories, and 0.901 in 20 shifted trajectories. The relaxation does not enforce the simulator's 0.20 minimum positive fraction: 10,871 of 2,970,777 positive variables fall below 0.20. It also omits the nonlinear welfare-reserve constraints along a schedule. Resource capacity is therefore sufficient in the relaxed model. Appendix A.18 adds reserve and minimum-service constraints at individual states, rather than certifying the feasibility of every target along a full trajectory.

A.26 Exact reuse across candidate orders

Many changes in request order leave the packed allocation unchanged. We reuse the unchanged work without removing any retained weight or changing the selector. The packer's state consists of its remaining resource vector and an exhaustion flag. Serving a request updates that state. Skipping a request leaves it unchanged. For consecutive orders, we retain their common prefix and identify the changed block before their common suffix.

Proposition 3 (Exact packing reuse)**.** *Fix the requests, capacity, minimum service fraction, and ranked-greedy packer of Section 4. Suppose two orders differ only by permuting a block. If the old scan serves no request in that block, the two orders produce identical allocations. Otherwise, replay may start at the block's beginning. Its unchanged suffix may be reused whenever the replay reaches exactly the old residual resource vector and exhaustion flag at that suffix.*

Proof. The common prefix produces the same state. If the changed block contains only skipped requests, each was tested at the same unchanged residual capacity, or after exhaustion. Its admission outcome is therefore unchanged under any permutation of the block. The common suffix then starts in the same state and produces the same allocation. In the second case, each suffix transition depends only on its request and incoming packing state, not on which preceding requests received service. Equality of that state and the suffix order proves equality of all subsequent transitions by induction. The new assignments before the suffix are retained. □

The implementation compares actual residual vectors exactly, so reuse does not assume that floating-point subtractions commute. Candidate dictionaries, labels, and multiplicities are preserved. Consequently, all coordinates of Equation (8), the chosen allocation, and the reserve update are identical to fresh packing. This includes repayment and the final allocation tie-break, not only the maximum-debt objective.

We also restrict crossing-component construction to the requests appearing in crossing pairs. All other requests are singletons whose scores remain unchanged. Each component retains the same minimum-ID score anchor. At every retained weight, we evaluate the Decimal scores, apply these event corrections, and sort the previous order using the fixed score and tie keys. Input position resolves any remaining tie, preserving stable sorting. No intermediate swap orders are added to the candidate family.

For n requests, let m be the number of retained weights and R the total number of request positions replayed. Feasibility checks are bounded by R, instead of mn. A changed block containing only skipped requests needs no new checks, and reaching the old state at a common suffix avoids checking that suffix. Finding these blocks and emitting candidates still takes up to $O(mn)$ work. Full score evaluation and worst-case sorting remain $O(mn)$ and $O(mn \log n)$, respectively. The saving depends on which parts of the packing scan change, rather than only on the number of distinct allocations.

Computational evaluation. We measure complete allocation calls on 40 states from four fractional TERRA-CP(64) trajectories, with ten states per trajectory (Table 26). We apply both packing rules to each state, so the full-bundle measurements isolate packing cost rather than describe native whole-request trajectories. The implementation was fixed on a separate development trajectory, with a prespecified requirement of at least 20% aggregate CPU savings over the uncapped packing-reuse reference before evaluation.

The two implementations agree exactly on all 109,456 candidate comparisons and 480 measured paired calls, including selector counters and reserve updates. For complete search, mean CPU time falls from 168.10 to 126.17 ms/call under fractional packing and from 177.86 to 128.13 ms/call under full-bundle packing, reductions of 24.9% and 28.0% (Table 26). The corresponding elapsed-time reductions are 24.9% and 28.7%, with savings on all four source trajectories under both rules. The timed call includes representative construction, score evaluation, sorting, packing, signatures, reserve checks, selection, and cache disposal. Inputs and correctness checks are outside timing. Methods run serially in counterbalanced order after warm-up on an Intel Xeon W-2235 CPU, with Windows 11 and Python 3.12.2.

Of 52,336 uncapped target orders per packing rule, 26,128 repeat the previous order. A further 10,708 fractional and 10,544 full-bundle targets change only a block of skipped requests, avoiding a fresh packing scan despite a different order. The combined optimization also includes sparse event processing and adaptive sorting. TERRA-CP(64) retains more widely separated orders and offers fewer opportunities for reuse than complete search. These fixed-state measurements isolate computation. Trajectory timings are reported with their corresponding experiments.

A.27 Service accounting and workload inputs

The fractional VTC control charges the token service delivered by each allocation, whereas guarded VTC in the serving study accumulates observed prefill and output-token service. Both adapt service accounting to admission rather than replace the engine's decode scheduler, so they do not inherit the continuous-batching guarantee of in-engine VTC (Sheng et al., 2024).

Table 26: Exact reuse lowers the cost of complete search. Mean allocation CPU time in ms/call over 40 fixed states, with three serial repetitions. The range is the smallest and largest reduction among the four source trajectories, not a confidence interval. The TERRA-CP(64) reference reuses candidate-independent evaluations. The uncapped reference additionally reuses packing prefixes and suffixes.

Budget	Packing	Reference CPU (ms/call)	Optimized CPU (ms/call)	CPU reduction	Reduction range
64	Fractional	26.43	**25.26**	4.4%	$[-6.1, 11.1]\%$
64	Full bundle	29.17	**26.30**	9.8%	$[0.0, 21.1]\%$
Uncapped	Fractional	168.10	**126.17**	24.9%	$[20.3, 27.1]\%$
Uncapped	Full bundle	177.86	**128.13**	28.0%	$[24.9, 31.0]\%$

For the distinction between reflected and cumulative debt, consider a workload with target 0.5, active only in rounds 1 and 3, receiving service 1 and 0. The active-round increments $(-0.5, 0.5)$ sum to zero, but their maximum suffix sum is 0.5. Thus cumulative deficit is zero while reflected debt is 0.5. Inactive rounds leave it unchanged (Lemma 2).

The artifact retains the Azure and BurstGPT trace windows, prompt-source rows, task instructions, and preprocessing settings. LongBench (Bai et al., 2024) and HumanEval (Chen et al., 2021) provide the prompt sources for the serving workloads. Values, service targets, SLOs, latency sensitivities, and workload classes are administrator-configured inputs. Prompt-pool construction uses whitespace-based input estimates before vLLM tokenization; execution logs record actual token counts. The serving subsections specify the workload-dependent arrival scales, output limits, and calibration.

A.28 Tenant-controlled reports

The main allocation experiments use administrator-assigned priorities. We also test fixed deviations in reported value, demand, and latency sensitivity, evaluating outcomes under the unchanged underlying types. Ten paired trajectories with 100 workloads and 1,000 rounds compare a fixed 20% deviating population with truthful reports. A separate ten-trajectory study changes one target's report at a time, using ten targets per trajectory. Value inflation yields mean realized-utility gains of 650.8 ± 252.4 and 699.4 ± 163.2, respectively, with 67% and 72% of targets benefiting. Intervals use trajectory-level means. The other three fixed deviations are not profitable on average. These tests are not a best-response search. Proposition 6 gives a two-workload counterexample, so the reported-welfare reserve is not an incentive-compatibility guarantee.

A.29 Bounds on one-step allocation quality

Search loss versus family restriction. Let $M_K = M_t(X_K)$ and $M_\infty = M_t(X_\infty)$. Under Lemma 1's exact-arithmetic assumptions, with identical service predictions, guard, and allocation domain, the guarded capped family is contained in the uncapped family, which lies in the full feasible set $\mathcal{H}_t$ from Equation (2). Consequently,

$$M_K - M_t^* = \underbrace{M_K - M_\infty}_{\text{cap loss}} + \underbrace{M_\infty - M_t^*}_{\text{family and packing gap}} . \tag{11}$$

Both terms are nonnegative. Searching the complete family eliminates cap loss. A binary oracle measures the remaining family and packing gap for full-bundle candidates. For fractional service, the state-specific lower bound $L_t \le M_t^*$ below gives

$$0 \le M_K - M_\infty \le \min\{1, M_K - L_t\}, \qquad 0 \le M_K - M_t^* \le M_K - L_t.$$

The difference $M_K - L_t$ bounds the remaining one-step improvement available beyond the retained candidates. Equality certifies the smallest maximum debt, while repayment and welfare can still distinguish tied allocations. We compute these bounds offline by relaxing the welfare and minimum-service constraints, so the calculation also covers allocations outside the ranking family.

Proposition 4 (Reserve-relaxed projected-debt certificate)**.** *Use the minimum fulfilled reported-resource fraction as predicted service. Fix a round and let $\mathcal{I}$ contain the workload identities whose debts enter M_t. For active request i, define*

$$q_i = Q_i^t + \rho_i, \qquad c_{ir} = \max\{\widehat{d}_{ir}^t, \epsilon\}, \qquad \bar{s}_i = \min\left\{1, \min_r \frac{\widehat{d}_{ir}^t}{c_{ir}}\right\}.$$

Consider the continuous program

$$\begin{aligned} L_t = \min_{z,s} \quad & z \\ \text{s.t.} \quad & 0 \le s_i \le \bar{s}_i && i \in \mathcal{A}_t, \\ & \sum_{i \in \mathcal{A}_t} c_{ir} s_i \le C_{tr} && r \in \mathcal{R}, \\ & z \ge q_i - s_i && i \in \mathcal{A}_t, \\ & z \ge Q_j^t && j \in \mathcal{I} \setminus \mathcal{A}_t, \\ & z \ge 0. \end{aligned} \tag{12}$$

It exactly minimizes projected maximum debt over the continuous reported-service domain after relaxing the welfare-reserve guard and minimum positive service. If M_t^{valid} is the infimum after restoring those restrictions but before restricting the candidate family, then every controller-valid allocation X satisfies

$$L_t \le M_t^{\text{valid}} \le M_t(X), \qquad 0 \le M_t(X) - M_t^{\text{valid}} \le M_t(X) - L_t.$$

Thus $M_t(X) = L_t$ certifies the smallest maximum projected debt after one round over all continuous capacity-feasible reported allocations.

For any resource-feasible allocation x, set

$$s_i = \max\left\{0, \min\left\{1, \min_r \frac{x_{ir}}{c_{ir}}\right\}\right\}.$$

Then $c_{ir} s_i \le x_{ir}$, $s_i \le \bar{s}_i$, and the service vector is feasible in Equation (12). Conversely, every feasible service vector is realized in the relaxed domain by $x_{ir} = c_{ir} s_i$. Its upper bound gives $x_{ir} \le \widehat{d}_{ir}^t$. This proves the proposition. For computation, let

$$I_t = \max\left(\{Q_j^t : j \notin \mathcal{A}_t\} \cup \{0\}\right), \quad U_t = \max\left(\{q_i - \bar{s}_i : i \in \mathcal{A}_t\} \cup \{0\}\right)$$

and

$$\lambda_{tr} = \inf\left\{z \ge 0 : \sum_{i \in \mathcal{A}_t} c_{ir}[q_i - z]_+ \le C_{tr}\right\}.$$

Maximum debt z requires at least $[q_i - z]_+$ service per active request. Inactive debt I_t, service limits U_t, and resource capacities imply $L_t = \max\{I_t, U_t, \max_r \lambda_{tr}\}$. Canonical decimal inputs make all breakpoints rational, so these finite water-filling pieces are solved exactly.

Repayment when maximum debt is optimal. When $M_t(X) = L_t$, we bound repayment without increasing maximum debt. Define mandatory service $\ell_i = [q_i - L_t]_+$, weight $w_i = Q_i^t / \max\{\rho_i, \epsilon\}$, and residual capacity $R_{tr} = C_{tr} - \sum_i c_{ir}\ell_i$. For each resource, maximize additional repayment as a single-resource fractional knapsack:

$$\overline{P}_{tr} = \sum_i w_i \ell_i + \max_{\substack{0 \le y_i \le \bar{s}_i - \ell_i \\ \sum_i c_{ir} y_i \le R_{tr}}} \sum_i w_i y_i.$$

With $\overline{P}_{t0} = \sum_i w_i \bar{s}_i$, set $\overline{P}_t = \min\{\overline{P}_{t0}, \min_r \overline{P}_{tr}\}$. Every multi-resource primary-optimal service vector is feasible for each single-resource relaxation, so $P_t(X) \le \overline{P}_t$. When $M_t(X) = L_t$ and $P_t(X) = \overline{P}_t$, equality certifies the first two coordinates in this exact model. A positive difference only bounds possible repayment improvement: the relaxation can violate the reserve or use positive service below 0.2.

Matched-state certificate evaluation. We evaluate the certificate on all 30,000 rounds of 30 TERRA-CP(64) trajectories and evaluate Dense128 at the same pre-decision states. A positive inactive-debt floor determined the bound in 21,188 states. 30 initial states were all-zero ties. In the remaining 8,782 active-limited states, the per-request service upper bound U_t determined the bound. No resource floor attained L_t at a positive value.

We rationalize logged float allocations and evaluate them with a $C_{tr} + 10^{-9}$ capacity envelope. The largest observed capacity overage is 1.39×10^{-12} for TERRA-CP(64) and 1.417×10^{-12} for Dense128, so the certificates apply to the reconstructed rational model.

Table 27: One-step optimality certificates on the same 30,000 states. A certificate establishes minimum maximum debt. The two-objective certificate also establishes maximum repayment at that debt level. Fractions average 30 within-trajectory rates, with 95% t-interval half-widths. Active-limited rows use that subset, and other fractions use all states.

Metric	TERRA-CP(64)	Dense128
Inactive debt sets the bound	0.707 ± 0.028	
Service cap sets bound, active-limited	1.000 ± 0.000	
Resource capacity sets bound, active-limited	0.000 ± 0.000	
Maximum-debt optimality certified ↑	$\mathbf{0.689} \pm 0.018$	0.610 ± 0.021
Debt optimality certified, active-limited ↑	$\mathbf{0.291} \pm 0.043$	0.098 ± 0.032
Mean bound on excess maximum debt ↓	$\mathbf{0.2462} \pm 0.0193$	0.3195 ± 0.0211
Debt and repayment optimality certified	$\mathbf{0.027} \pm 0.011$	$\mathbf{0.027} \pm 0.011$
Both certified, active-limited	$\mathbf{0.011} \pm 0.004$	$\mathbf{0.011} \pm 0.004$

Across seed summaries, TERRA-CP(64) exceeds Dense128's active-limited maximum-debt certificate rate by 0.19321 ± 0.02256. Its matched-state difference in one-step maximum projected debt is -0.07331 ± 0.01070, where lower is better. The exact joint debt-and-repayment certificate holds in 796/30,000 states overall and in 84/8,782 active-limited states for either policy.

Repayment among equal-maximum-debt allocations. Equal maximum debt need not mean equal repayment or the same allocation. We stratify the matched-state records by the source of the lower bound: an inactive account that cannot be served now, or an active account whose unit-service limit sets the bound. Table 28 compares these three levels of agreement within the post hoc strata. In the positive inactive-floor stratum, TERRA-CP(64) attains the lower bound in 18,196 of 21,188 states and Dense128 in 17,528. Thus an inactive floor does not guarantee that a candidate attains the bound.

Table 28: Maximum projected debt and decision agreement on the same TERRA-CP(64)-visited states. Absolute means and differences use pooled states within each stratum. Differences are TERRA-CP(64) minus Dense128. M denotes maximum projected debt, P normalized repayment, and X the allocation. The lower panel retains the pooled agreement counts.

State stratum	Dense128 M	TERRA-CP(64) M	ΔM
All states	25.07796	25.00465	−0.07331
Positive inactive floor	23.88790	23.87206	−0.01584
Active-limited	28.03318	27.82097	−0.21221
Zero-floor ties	0.48998	0.48998	0.00000

State stratum	n	Same M	Same (M, P)	Same X
All states	30,000	27,001	2,465	1,950
Positive inactive floor	21,188	20,316	1,972	1,513
Active-limited	8,782	6,655	463	407
Zero-floor ties	30	30	30	30

The active-limited states show a larger improvement in maximum debt: -0.2131 ± 0.0270, compared with -0.0151 ± 0.0051 for the positive inactive-floor states (means of 30 within-trajectory effects, with 95% interval half-widths). TERRA-CP(64) has higher maximum debt in eight of the 30,000 states. When maximum debt ties, repayment distinguishes the methods in 24,536 states: TERRA-CP(64) repays more in 89.99% of these ties and less in 0.89%. The mean difference is 92.00 ± 18.94 over trajectory means. Within inactive-floor and active-limited ties, the favorable proportions are 89.44% and 92.07%, respectively. Thus equal maximum debt can leave substantial differences in how other accounts are repaid. The separate selector intervention in Table 15 tests whether this component changes long-run service outcomes.

A.30 Additional proof details

Proof of Lemma 1. We cover constant-order intervals and handle their crossing points separately. For each pair,

$$f_i^t(\nu) - f_j^t(\nu) = (\alpha_i^t - \alpha_j^t) + \nu(\beta_i^t - \beta_j^t)$$

is affine. Its sign, and hence the complete request order, is constant on every open interval without a positive pairwise crossing. One representative therefore suffices for each interval. At each crossing, TERRA-CP applies the same fixed tie keys as direct evaluation. Coincident lines keep that order everywhere. If a positive crossing exists, we also sample between zero and the first crossing, and above the last crossing to cover the unbounded interval. Because $\mathcal{P}_t$ is deterministic and reads only the total order at the fixed round state, equal orders give identical allocations. With at most $\binom{n}{2}$ distinct positive crossings, zero, the crossing points, all bounded open intervals, and the tail require at most $n(n-1)+2$ weights.

Proof of Proposition 2. Capping only removes feasible allocations, so its optimum cannot improve. Since $\widehat{a}_i(X) \in [0,1]$, reflection is one-Lipschitz in service:

$$|[Q_i + \rho_i - \widehat{a}_i(X)]_+ - [Q_i + \rho_i - \widehat{a}_i(Y)]_+| \le 1.$$

Taking maxima gives $0 \le M(X_K) - M(X_\infty) \le 1$.

For tightness, we construct an omitted allocation with much lower debt. Choose rational $0 < A < \min\{1/5, \delta/4\}$, set $\varepsilon = A/(4K^2)$, and index $K+1$ requests by $i = 0, \dots, K$. Let $C = \widehat{d}_i = (1,1,1)$, so $b_i = 2$, and define

$$Q_i = 2(A - \varepsilon i^2), \qquad \widehat{V}_i = i/K.$$

Then $\sigma = 1$, $\alpha_i = A - \varepsilon i^2$, and $\beta_i = i/(2K)$. Completing the square in $f_i(\nu) = A - \varepsilon i^2 + \nu i/(2K)$ shows that each request is the unique maximizer on a nonempty open interval, in the order $0, 1, \dots, K$. For $i < j$, their crossing is $2K\varepsilon(i+j)$, so the distinct positive levels are exactly $2K\varepsilon s$, $s = 1, \dots, 2K-1$. The complete weight sequence therefore has $4K$ elements. For TERRA-CP(64)'s 12-significant-digit coefficient representation, take $K = 64$, $A = 16384 \cdot 10^{-p}$, and $\varepsilon = 10^{-p}$ for sufficiently large $p \ge 6$. Then $\alpha_i = (16384 - i^2)10^{-p}$ and $\beta_i = i/128$. 12-significant-digit canonicalization changes neither coefficient.

Both capped rules choose at most K weights from the affine coefficients and include zero and a tail point. Each full demand consumes all capacity, so each allocation serves one request. The endpoints serve requests 0 and K. The additional greedy reference serves request K, duplicating the tail. Thus at most K single-request allocations remain and some interior request m is absent. At fixed debts and full-bundle service credits, neither weight rule depends on the service targets chosen below. To make this omission costly without changing the retained weights, let $H = 8K$ and define

$$\rho_i = Q_i/H \ (i \ne m), \qquad \Delta = \sum_{i \ne m} \rho_i, \qquad \rho_m = 1 - \Delta, \qquad \gamma = 0.$$

Then $\sum_i \rho_i = 1$, $\Delta \le A/4 < Q_m$, and all service targets lie in $(0,1]$. All allocations have nonnegative welfare and pass the zero-reserve guard. Serving the omitted request gives

$$M(X_\infty) = 2A(1 + 1/H).$$

Serving any retained request $j \ne m$ leaves request m at $1 + Q_m - \Delta$. All other projected debts are smaller. Thus

$$M(X_K) = 1 + Q_m - \Delta, \qquad Q_m > 3A/2.$$

Since $2/H \le 1/8$, the additive gap exceeds $1 - 7A/8 > 1 - \delta$. Also $H \ge 16$ and $Q_m > \Delta$, so the ratio exceeds $1/[2A(1 + 1/H)] > 1/\delta$.

We now realize this state from zero debt with fixed service targets. First, make only workload m active in a history slot, set its reported utility identically to zero for every allocation in that slot, and set capacity to $(1 - \Delta - Q_m)\widehat{d}_m$. The affine candidate serves fraction $1 - \Delta - Q_m > 0.2$, while the greedy reference is empty. Serving leaves debt Q_m. Leaving the request unserved would produce debt $\rho_m > Q_m$. The selector therefore serves it, and the welfare reserve remains zero. In H later zero-capacity slots, activate every $i \ne m$. Each accrues $H\rho_i = Q_i$, while inactive m retains its debt. The displayed slot may use different requests and reports for the same workload identities, as the round model allows. All displayed quantities and the history are rational.

Proof of Proposition 1. We place a useful request order between grid weights, then realize a state that selects it. Because $\mathcal{G}$ is finite, choose rational $0 < a < b < 1/2$ with $[a, b] \cap \mathcal{G} = \varnothing$. For a one-slot witness, use $C = (1, 1, 1)$, identical reported demands $\widehat{d}_i = (0.9, 0.1, 0.1)$, and $\omega = 0.5$, giving $b_i = 1$. After packing one bundle, the next request's clipped GPU fraction is $1/9 < 0.2$, so the minimum-service rule rejects it. Define

$$(\alpha_1, \beta_1) = ((a + b)/2, 0), \qquad (\alpha_2, \beta_2) = (b/2, 1/2), \qquad (\alpha_3, \beta_3) = (0, 1).$$

Request 1 is uniquely highest below a, request 2 inside (a, b), and request 3 above b. Thus no grid point ranks request 2 first, while the critical construction evaluates an interior representative. With three requests the sequence has at most eight weights, so TERRA-CP(64) does not truncate it. Greedy serves request 3, which has the highest reported value.

We next specify the target debts and service targets. Let $q_1 = (a + b)/2$, $q_2 = b/2$, and choose an integer h such that $q_1/h < (b - a)/4$. Take $\widehat{s}_i = 1$ and fixed service targets

$$(\rho_1, \rho_2, \rho_3) = (q_1/h, q_2, 0).$$

We first fund the welfare reserve at zero debt. If $\gamma \le 1/2$, no funding round is needed. If $1/2 < \gamma < 1$, make only request 3 active in one sufficiently provisioned round and give it full-bundle reported value

$$V = \frac{\gamma - 1/2}{1 - \gamma}.$$

Both greedy and TERRA-CP serve it, so the reserve becomes $(1 - \gamma)V = \gamma - 1/2$, while $\rho_3 = 0$ keeps every debt at zero. For $\gamma = 1$, use one funding round with $C = (1, 1, 1)$, demands

$$d_1 = (0.8, 0.2, 0.2), \quad d_2 = (0.2, 0.8, 0.2), \quad d_3 = (0.85, 0.85, 0.85),$$

and full-bundle reported values $(7/4, 7/4, 3)$. Their burdens are $(1, 1, 1.7)$, so reported-value density ranks request 3 first. Its residual capacity gives requests 1 and 2 clipped fraction $0.15/0.8 = 0.1875 < 0.2$, and greedy welfare is 3. At $\nu = 0$, fixed tie keys order requests 1, 2, then 3. Requests 1 and 2 both fit fully and give welfare $7/2$. This candidate is feasible under the atomic guard, has projected maximum debt zero rather than $\max\{q_1/h, q_2\} > 0$, and is therefore selected. Its service clears both service target increments, while the reserve becomes $1/2$. The construction uses the same three workload identities. Funding and displayed slots may contain different requests and reports, while service targets remain fixed.

After funding, use h zero-capacity rounds immediately before the displayed slot to accrue the target debts. Make request 1 active in all h, request 2 active only in the last, and request 3 newly active in the displayed slot. Starting from zero debt gives $(Q_1, Q_2, Q_3) = (q_1, q_2, 0)$, hence the stated affine intercepts. Set zero reported latency and participation cost and scale reported values so the full-bundle surpluses are $(0, 1/2, 1)$. Then $\sigma = 1$ and the stated slopes follow.

With all three requests active in the displayed slot, serving request 2 gives

$$M_2 = q_1 + q_1/h,$$

because its own pre-service projected debt is $2q_2 = b < 1$ and one unit of service clears it. Serving request 1 or request 3 leaves request 2 at debt b, and all other projected debts are smaller, so $M_1 = M_3 = b$. Thus

$$M_1 - M_2 = M_3 - M_2 = b - q_1 - q_1/h > (b - a)/4 > 0.$$

The funding history leaves the current welfare reserve at $B = \max\{0, \gamma - 1/2\}$. The greedy reserve is $r = \gamma$, so request 2 has slack $B + 1/2 - \gamma \geq 0$, and greedy request 3 has slack $B + 1 - \gamma \geq 0$. Every grid allocation serves request 1 or 3. Among its reserve-feasible candidates plus greedy, projected maximum debt is therefore b. TERRA-CP(64) contains request 2 and selects it uniquely on the first lexicographic coordinate. All displayed-slot demands, coefficients, debts, service targets, and reported utilities are rational. The reserve is nonnegative for every $\gamma \in [0, 1]$, depends on γ, and need not be rational.

Theorem 2 (Positive-scale equivariance)**.** *In exact arithmetic, replace the entire reported-utility function by $\widehat{U}_i'^t(X) = c\widehat{U}_i^t(X)$ for every round, request, and allocation, where $c > 0$. Assume $\sigma_t \geq \delta$ and $c\sigma_t \geq \delta$ at every round, where $\delta = 10^{-12}$, so the normalization floor is inactive in both runs. Leave all other inputs fixed and initialize $B_0' = cB_0$. TERRA-CP retains the same candidate set, allocation, and debt trajectory, while $B_t' = cB_t$.*

The implementation evaluates utilities in binary floating point before Decimal event handling. The conclusion applies when evaluated utilities remain exactly proportional and the floor and canonicalization branches are identical.

Proof of Theorem 2. Normalization preserves rankings. Positive scaling preserves feasibility and selection. The active utility scale becomes $\sigma_t' = c\sigma_t$, hence

$$\frac{c\widehat{V}_i^t}{c\sigma_t b_i^t} = \beta_i^t.$$

Debt intercepts, pairwise crossings, and deterministic subsampling are unchanged. The active count and normalized coefficients are also unchanged, so TERRA-CP chooses the same construction branch, inversion counts, and quantile weights. Candidate welfare, greedy reference welfare, and the reserve scale by c. Thus reserve feasibility is preserved. The first two selector coordinates are unchanged and the positive scaling of its welfare coordinate preserves all remaining comparisons. Equation (7) gives $B_{t+1}' = cB_{t+1}$. Induction proves the claim.

Lemma 2 (Active-round debt reflection and accounting)**.** *Let $t_{i,1} < \cdots < t_{i,h}$ be workload i's $h = h_i(T)$ active rounds before T, $y_{i,m} = \rho_i - a_i^{t_{i,m}}$, and $D_{i,k} = \sum_{m=1}^{k} y_{i,m}$, with $D_{i,0} = 0$. If $Q_i^0 = 0$, then*

$$Q_i^T = \max_{0 \leq k \leq h} \sum_{m=k+1}^{h} y_{i,m} = D_{i,h} - \min_{0 \leq k \leq h} D_{i,k}.$$

For arbitrary $Q_i^0 \geq 0$,

$$\sum_{t<T : i \in \mathcal{A}_t} a_i^t \geq \rho_i h_i(T) + Q_i^0 - Q_i^T.$$

Proof of Lemma 2. We ignore inactive rounds, which leave debt unchanged, and unfold the reflection over the h active rounds. Write $Q_{m+1} = [Q_m + y_{i,m+1}]_+$. For $h = 0$, both sides of the suffix identity are zero. If the identity holds after h active rounds, then

$$Q_{h+1} = \max\left\{0, \max_{0 \leq k \leq h} \sum_{m=k+1}^{h} y_{i,m} + y_{i,h+1}\right\} = \max_{0 \leq k \leq h+1} \sum_{m=k+1}^{h+1} y_{i,m}.$$

Induction proves the suffix identity. Since $\sum_{m=k+1}^{h} y_{i,m} = D_{i,h} - D_{i,k}$, maximizing over k gives the cumulative-deficit form in the lemma. For arbitrary nonnegative initial debt, the service bound follows because $[x]_+ \geq x$ on each active round gives $Q_i^{t+1} \geq Q_i^t + \rho_i - a_i^t$. Rearranging and summing over active rounds telescopes the debt terms.

Proof of Theorem 1. We first show that greedy remains feasible, then sum reserve updates. For an atomic round, greedy is included and has slack

$$B_t + \widehat{W}_t(G_t) - \mathrm{res}_\gamma\left(\widehat{W}_t(G_t)\right),$$

equal to $B_t + (1-\gamma)\widehat{W}_t(G_t) \geq 0$ when the reference welfare is nonnegative and to $B_t \geq 0$ otherwise. Thus the feasible set is nonempty. The selected allocation's whole-allocation guard is exactly $B_{t+1} \geq 0$. Induction from $B_0 = 0$ proves reserve nonnegativity, and summing

$$B_{t+1} - B_t = \widehat{W}_t(X_t) - r_t$$

through $T-1$ gives the cumulative welfare bound.

For sequential admission, let $\pi_X = (x_1, \ldots, x_m)$ and $\pi_G = (g_1, \ldots, g_n)$, extending cumulative rewards constantly after a sequence ends. For $k \leq n$, the greedy prefix slack is

$$B_t + \widehat{W}_t(\pi_G^{1:k}) - \mathrm{res}_\gamma\left(\widehat{W}_t(\pi_G^{1:k})\right),$$

equal to $B_t + (1-\gamma)\widehat{W}_t(\pi_G^{1:k}) \geq 0$ when the prefix is nonnegative, and to $B_t \geq 0$ otherwise. For $k > n$, it stays constant, so greedy passes through $\max(m, n)$. Every selected prefix has nonnegative slack, and the final horizon certifies the complete planned pair. If only a prefix is committed, the matched-prefix update makes its post-commit reserve this same slack, so summing the recurrence gives the same cumulative bound.

A stronger planned-sequence check. Checking through $\max(m, n)$ rejects planned tails that would violate the reserve. This certifies the full planned pair. A partial commit needs only the matched-prefix check, since neither uncommitted tail is charged.

Proposition 5 (Projected-debt optimization over the affine family)**.** *Under the exact-arithmetic assumptions of Lemma 1, uncapped TERRA-CP at a fixed state exactly optimizes Equation* (8) *over every reserve-feasible ranked-greedy allocation induced by any $\nu \geq 0$, together with the greedy reference. With a candidate cap, it optimizes only over the retained deduplicated subset. In either case, the selected allocation satisfies $M_t(X_t) \leq M_t(G_t)$.*

Without capping, Lemma 1 covers every allocation reachable by the affine family. The greedy reference remains feasible, so minimizing maximum projected debt over the candidate set cannot give a larger value than $M_t(G_t)$. Different allocations create different future states, so this is not sample-path dominance over greedy, dense-grid MaxWeight, or VTC.

Proposition 6 (Incentive boundary)**.** *TERRA-CP is not strategy-proof. There is a one-round two-agent single-slot instance in which an unserved agent gains realized utility, evaluated at its unchanged underlying type, by inflating reported value.*

We let reported welfare break a debt-objective tie. Give both agents equal positive service target, debt, and non-value type, with positive utility from service at their true types. Fix the $\nu = 0$ tie key against agent 1 and choose $\gamma = 0$. If agent 1 truthfully reports the lower value, agent 2 is top-ranked for every $\nu > 0$, so agent 1's allocation need not enter the truthful candidate family and agent 1 is unserved. If agent 1 reports above agent 2, it is top-ranked for $\nu > 0$, so its allocation is generated. Serving either agent is reserve-feasible and ties the first two selector coordinates. The inflated reported welfare then selects agent 1, which gains its positive unchanged-type utility.

Proposition 7 (Non-idling candidate under signed serving welfare)**.** *Consider the waiting-admission bridge in which the greedy reference and every ranked candidate scan feasible pending requests without discarding requests whose waiting-adjusted reported value is nonpositive. If $B_t \geq 0$ and at least one pending request fits within the available positive slot and token capacities, then the selector returns a nonempty reserve-feasible admission.*

Proof of Proposition 7. Each scan skips only infeasible requests before its first admission, so an initially fitting request makes every candidate, including greedy, nonempty. A greedy prefix has slack $B_t + (1-\gamma)\widehat{W}_t \geq 0$ when $\widehat{W}_t \geq 0$, and $B_t \geq 0$ otherwise. Greedy therefore supplies a nonempty prefix-safe candidate, and the selected scan is nonempty.

Critical-family event count. For $n = |\mathcal{A}_t|$ and $c_t \leq \binom{n}{2}$ distinct positive crossings, Lemma 1's construction uses at most $2c_t + 2 \leq n(n-1) + 2$ representatives before deduplication. Thus $c_t \leq 31$ prevents truncation in the enumeration branch of TERRA-CP(64), and $n \leq 8$ suffices under the hybrid routing rule. Under unit-cost arithmetic, constructing and sorting all crossings costs $O(n^2 \log n)$, and packing K representatives costs $O(Kn \log n)$. These bounds exclude arithmetic bit complexity. Appendix A.2 gives the implicit-construction bound.

Debt accounting corollary. For $h_i(T) > 0$, Lemma 2 gives $h_i(T)^{-1} \sum_{t<T:i\in\mathcal{A}_t} a_i^t \geq \rho_i - (Q_i^T - Q_i^0)/h_i(T)$. If $h_i(T) \to \infty$ and $Q_i^T = o(h_i(T))$, asymptotic average active-round service is at least ρ_i. This sublinear-debt condition must be established separately and can fail when aggregate targets exceed feasible capacity.