BYTEBUDGET: OPTIMIZING WEIGHT–KV PRECISION ALLOCATION FOR CPU INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive decoding on CPUs is memory-bandwidth-bound; generating each token requires streaming model weights, whose traffic grows with the context length. However, the quantization methods typically optimize weights or KV caches separately, and a fixed precision policy cannot account for changes in context length, hardware characteristics, or kernel efficiency. We introduce BYTEBUDGET, a hardware-and-context-aware planner that converts a decode-latency target into a calibrated effective-traffic budget and jointly allocates precision across weight blocks and a KV cache to minimize estimated quality loss. The framework consists of three components: a format-aware CPU cost model, modular quality-loss curves supplied by off-the-shelf quantizers, and a discrete budgeted optimizer. For practical deployment, we use an efficient marginal-gain planner to produce mixed-precision weight and KV cache configurations, with a dynamic-programming oracle and lagrangian lower bound to audit solution quality. Since KV cache traffic grows with context length while weight traffic remains roughly constant, BYTEBUDGET adapts precision allocation changing deployment conditions. Our framework integrates the quantization methods with CPU-based LLM decoding kernels and allocates precision across layers to satisfy deployment-specific latency constraints, rather than a uniform bit-width.

1 INTRODUCTION

Single-request large-language model (LLM) decoding operates under the stringent runtime constraints. Unlike prompt processing, which exposes substantial matrix–matrix parallelism, each autoregressive step at batch size one degenerates much of the dense computation into matrix–vector products and repeatedly accesses model state. Because processors execute arithmetic faster than data can be supplied from external memory, token latency can therefore be governed more by bytes transferred than by nominal floating-point operations (Pope et al., 2023; Williams et al., 2009; Yao et al., 2024). This regime on the CPUs processing matters for private, offline, and edge device inference, where CPUs are widely available, but memory bandwidth is limited.

Quantization tackles this bottleneck from two directions. Weight post-training quantization reduces the static parameters accessed at every decoding step (Dettmers et al., 2022; Frantar et al., 2023; Lin et al., 2024; Wenqi et al., 2024). Weight–activation methods further reshape or rotate distributions to support lower arithmetic precision (Xiao et al., 2023; Ashkboos et al., 2024; Liu et al., 2025; Zeng et al., 2025; Zhao et al., 2026). Separately, KV cache quantization lowers a state term that grows with the active context (Liu et al., 2024; Hooper et al., 2024; Zhao et al., 2024; Lin et al., 2025). These advances establish strong quantizers and kernels, but they do not answer by themselves a deployment question: *given a CPU, a target context, and a latency service-level objective, where should a finite traffic budget be spent?*

The answer is context dependent. Lowering the precision of a weight block changes an approximately context-independent traffic term, whereas changing the KV format changes a term multiplied by the current context length. A policy selected at one length can therefore be dominated at another. The answer is also hardware dependent: nominally equal byte reductions need not yield equal time reductions when formats differ in metadata, dequantization, packing, cache behavior, or kernel maturity. Recent serving studies show that theoretical compression may fail to become measured speedup unless the runtime path is designed and supervised accordingly (Lin et al., 2025; Gao et al.,

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.

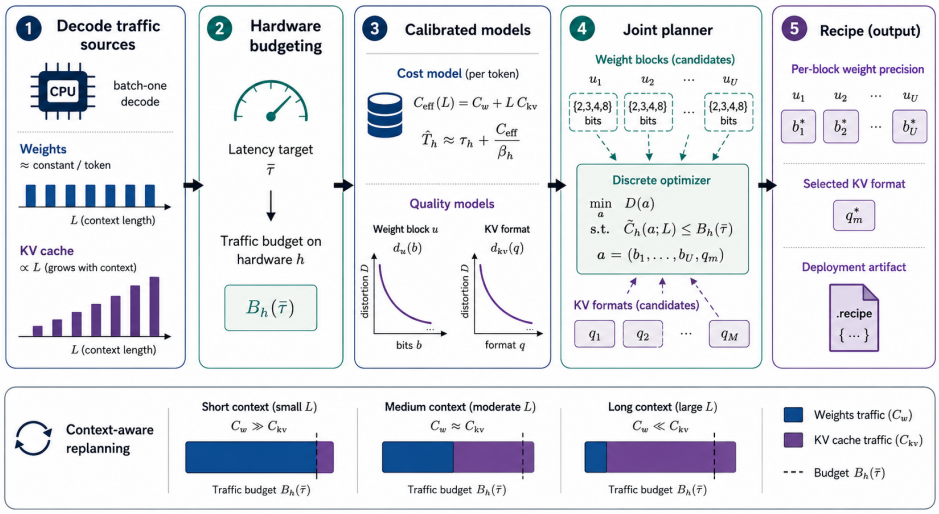


Figure 1: **Overview of ByteBudget**. ByteBudget jointly allocates precision to model weights and the KV cache for bandwidth-bound, batch-one CPU decoding. Weight traffic remains approximately constant with context length, whereas KV-cache traffic grows with the active context. A hardware-calibrated latency target is first converted into an effective traffic budget, while model-dependent quality curves characterize the distortion associated with each candidate weight precision and KV format. ByteBudget then solves a discrete budgeted allocation problem to select per-block weight precisions together with a KV-cache format. As the context length changes, the relative traffic contribution of the KV cache changes accordingly, motivating context-aware replanning under the same hardware-specific latency constraint.

2025). Consequently, "minimize model size" and "minimize CPU decode latency" are not equivalent objectives.

We introduce BYTEBUDGET, a context-aware precision planner for bandwidth-bound CPU decoding. BYTEBUDGET represents a quantization choice by its *effective traffic* on the target processor, not merely its nominal bit width. It estimates modular quality-loss curves for quantizable weight units and candidate KV formats, solves a multiple-choice budgeted allocation problem, and emits a deployment recipe that can be materialized as one mixed-precision model plus a runtime KV configuration. The framework is deliberately orthogonal to the underlying quantizer: reconstruction error, output error, Hessian-weighted error, or task-level calibration can supply the quality curves, while any backend with measurable storage and latency can supply the cost curves.

Figure 1 summarizes the straight-through workflow. We focus on dense decoder-only models, low-batch CPU decoding, and a fixed planning context or context distribution. Rather than introducing a new scalar quantizer or KV-cache kernel, BYTEBUDGET provides a hardware- and context-aware allocation layer that combines existing quantization options under an explicit latency constraint.

Our contributions are:

- We formalize joint weight/KV precision selection as latency-constrained risk minimization under a context-dependent, format-aware traffic model.
- We derive a latency-to-byte-budget reduction, a crossover diagnostic, and an exact context-scaling identity for KV upgrade densities while stating the conditions under which no monotone final-precision conclusion follows.
- We combine a practical marginal-gain planner with a rounded-cost dynamic-programming oracle and a separable Lagrangian theorem that supplies a global lower bound and an exact optimality certificate for the additive surrogate.
- We specify an end-to-end evaluation that separates predictor accuracy, allocation quality, kernel realization, and portability; all pending measurements are marked in red.

2 PROBLEM FORMULATION

2.1 QUANTIZATION DECISIONS

Let $\mathcal{U}$ be a set of quantizable weight units. A unit may be a linear layer, an attention sub-block, an MLP sub-block, or any deployment-aligned group. Unit $u \in \mathcal{U}$ has a finite option set $\mathcal{B}_u$. Selecting $b_u \in \mathcal{B}_u$ determines a packed storage cost $c_u(b_u) > 0$ and a nonnegative quality-loss estimate $d_{u,r}(b_u)$ under a declared quality-calibration regime r comprising data, task mixture, and context distribution. The notation b_u denotes a discrete format option and need not be a literal integer bit width.

The KV cache is represented by a finite format set $\mathcal{Q}$. A choice $q \in \mathcal{Q}$ may jointly specify key and value precision, quantization granularity, residual windows, outlier channels, and a backend kernel. It contributes $k(q)$ stored bytes per cached token and a quality-loss estimate $d_{\mathrm{kv},r}(q)$. This abstraction includes uniform KV precision as a special case and can represent asymmetric key/value policies motivated by recent analysis (Hariri et al., 2026). We use one global KV unit to match the current runtime; a backend with per-layer key/value formats is represented by adding one allocatable KV unit per independently configurable cache tensor.

2.2 FROM PHYSICAL BYTES TO EFFECTIVE TRAFFIC

For a current context length L, an ideal byte-count model is

$$\mathcal{C}(\mathbf{b}, q; L) \coloneqq c_{\mathrm{fix}} + \sum_{u \in \mathcal{U}} c_u(b_u) + Lk(q), \tag{1}$$

where c_{fix} accounts for state that is not optimized. Equation (1) captures the central asymmetry: weight traffic is paid once per generated token, while the attended KV state scales with L.

Physical size alone can mispredict latency. We therefore define hardware- and format-aware *effective traffic*

$$\widetilde{\mathcal{C}}_h(\mathbf{b}, q; L) \coloneqq c_{0,h} + \sum_{u \in \mathcal{U}} \rho_h^{\mathrm{w}}(b_u) c_u(b_u) + L\rho_h^{\mathrm{kv}}(q)k(q), \tag{2}$$

where h indexes a processor/runtime configuration and each nonnegative factor ρ converts physical bytes into time-equivalent bytes by absorbing systematic effects such as metadata, dequantization, cache reuse, and kernel efficiency. A byte-faithful implementation has $\rho = 1$. Latency observations identify only the ratios ρ/β_h, so declared reference weight and KV formats are anchored at $\rho = 1$; β_h is then the corresponding reference effective bandwidth. If shape-dependent residuals remain, formats may be stratified into a small, pre-declared set of kernel-shape classes rather than assigning an unconstrained coefficient to every layer. We model steady-state decoding latency as

$$\widehat{\mathcal{T}}_h(\mathbf{b}, q; L) \coloneqq \tau_h + \frac{\widetilde{\mathcal{C}}_h(\mathbf{b}, q; L)}{\beta_h}, \tag{3}$$

with intercept $\tau_h \geq 0$ and calibrated effective bandwidth $\beta_h > 0$. The simple roofline used by the reference prototype is the special case $\tau_h = 0$ and $\rho = 1$.

Assumption 2.1 (Target regime)**.** The requested operating points are in a low-batch decode regime for which the residual latency after a fixed intercept is approximately affine in effective memory traffic. We test rather than assume this approximation in Section 5.

2.3 QUALITY SURROGATE

Let $\mathcal{R}_r(\mathbf{b}, q)$ denote the true downstream risk of a quantized configuration in regime r. Measuring it for every joint assignment is combinatorial, so BYTEBUDGET uses the additive surrogate

$$\widehat{\mathcal{D}}_r(\mathbf{b}, q) \coloneqq \sum_{u \in \mathcal{U}} \omega_u d_{u,r}(b_u) + \omega_{\mathrm{kv}} d_{\mathrm{kv},r}(q), \tag{4}$$

where $\omega_u, \omega_{\mathrm{kv}} \geq 0$ normalize heterogeneous metrics. We fix r for one planning problem and suppress it below; planning for another context distribution may reuse the cost table but must

validate or re-estimate the quality curves. Curves may be obtained from weight reconstruction, activation-weighted output reconstruction, Hessian approximations, blockwise distillation, or held-out task loss. Quantization-as-perturbation analyses motivate treating the magnitude and placement of quantization noise as first-order predictors while warning that uniform error and clipping can be misleading (Gong et al., 2024). We therefore evaluate several estimators rather than asserting that one proxy is universally calibrated.

The prototype's calibration-free weight curve is a parameter-weighted relative Frobenius reconstruction error. For a module m inside unit u,

$$d_u(b) \coloneqq \frac{1}{\sum_{m\in u} n_m} \sum_{m\in u} n_m \frac{\|W_m - Q_b(W_m)\|_F^2}{\|W_m\|_F^2 + \varepsilon}, \tag{5}$$

where n_m is the number of parameters. This requires no forward pass, but it is only a proxy; every reported KV curve and the strongest weight curves in the final experiments will be measured on calibration data.

2.4 Latency-constrained allocation

Given a latency target $\bar{\tau}$, Equations (2)–(3) yield the effective traffic budget

$$B_h(\bar{\tau}) \coloneqq \beta_h(\bar{\tau} - \tau_h). \tag{6}$$

Let $(\mathbf{b}^{\min}, q^{\min})$ select the minimum-effective-cost option of every unit and define $B_{\min,h}(L) \coloneqq \widetilde{\mathcal{C}}_h(\mathbf{b}^{\min}, q^{\min}; L)$. When $B_h(\bar{\tau}) < B_{\min,h}(L)$, no candidate assignment is feasible. Otherwise, ByteBudget solves

$$\min_{\mathbf{b}\in\prod_{u\in\mathcal{U}}\mathcal{B}_u,\ q\in\mathcal{Q}} \widehat{\mathcal{D}}(\mathbf{b}, q) \quad \text{subject to} \quad \widetilde{\mathcal{C}}_h(\mathbf{b}, q; L) \le B_h(\bar{\tau}). \tag{7}$$

This is a multiple-choice knapsack problem with one selected option per unit (Kellerer et al., 2004). Because $\beta_h > 0$, the constraint in Equation (7) is equivalent to the predicted-latency constraint $\widehat{\mathcal{T}}_h(\mathbf{b}, q; L) \le \bar{\tau}$.

For certification, fix h, L, and a budget B. Let $\mathcal{J} \coloneqq \mathcal{U} \cup \{\mathrm{kv}\}$, let $\mathcal{A}_u \coloneqq \mathcal{B}_u$ and $\mathcal{A}_{\mathrm{kv}} \coloneqq \mathcal{Q}$, and absorb the weights in Equation (4) into per-unit losses ℓ_j. Define $c_{u,L}(b) \coloneqq \rho_h^{\mathrm{w}}(b) c_u(b)$, $c_{\mathrm{kv},L}(q) \coloneqq L\rho_h^{\mathrm{kv}}(q)k(q)$, $\mathcal{D}(\mathbf{a}) \coloneqq \sum_{j\in\mathcal{J}} \ell_j(a_j)$, and $\mathcal{C}_L(\mathbf{a}) \coloneqq c_{0,h} + \sum_{j\in\mathcal{J}} c_{j,L}(a_j)$. The globally optimal surrogate value is

$$\mathcal{D}^\star(B, L) \coloneqq \min_{\mathbf{a}\in\prod_{j\in\mathcal{J}}\mathcal{A}_j:\, \mathcal{C}_L(\mathbf{a})\le B} \mathcal{D}(\mathbf{a}). \tag{8}$$

Theorem 2.2 (Separable Lagrangian certificate)**.** *Assume $B \ge B_{\min,h}(L)$ and define, for $\lambda \in \mathbb{R}_{\ge 0}$,*

$$\Phi_{B,L}(\lambda) \coloneqq \lambda(c_{0,h} - B) + \sum_{j\in\mathcal{J}} \min_{a\in\mathcal{A}_j} \{\ell_j(a) + \lambda c_{j,L}(a)\}. \tag{9}$$

Let $\mathcal{S}_L$ contain 0 and every nonnegative intersection point of two nonidentical affine option functions $\ell_j(a) + \lambda c_{j,L}(a)$ and $\ell_j(a') + \lambda c_{j,L}(a')$ belonging to the same unit j. Then the following statements hold.

- *(i) $\Phi_{B,L}$ is continuous, concave, and piecewise affine on $\mathbb{R}_{\ge 0}$, and $\max_{\lambda\ge 0}\Phi_{B,L}(\lambda) = \max_{\lambda\in\mathcal{S}_L}\Phi_{B,L}(\lambda)$. In particular, $|\mathcal{S}_L| \le 1 + \sum_{j\in\mathcal{J}} \binom{|\mathcal{A}_j|}{2}$ and $\underline{\mathcal{D}}(B, L) \coloneqq \max_{\lambda\ge 0}\Phi_{B,L}(\lambda)$ is computable by a finite search.*
- *(ii) Every feasible assignment $\widehat{\mathbf{a}}$ satisfies $0 \le \mathcal{D}(\widehat{\mathbf{a}}) - \mathcal{D}^\star(B, L) \le \mathcal{D}(\widehat{\mathbf{a}}) - \underline{\mathcal{D}}(B, L)$.*
- *(iii) If a feasible assignment $\widehat{\mathbf{a}}$ and a multiplier $\lambda^\star \ge 0$ satisfy $\widehat{a}_j \in \arg\min_{a\in\mathcal{A}_j}\{\ell_j(a) + \lambda^\star c_{j,L}(a)\}$ for every $j \in \mathcal{J}$ and $\lambda^\star(B - \mathcal{C}_L(\widehat{\mathbf{a}})) = 0$, then $\widehat{\mathbf{a}}$ is globally optimal and $\mathcal{D}(\widehat{\mathbf{a}}) = \mathcal{D}^\star(B, L) = \Phi_{B,L}(\lambda^\star)$.*

Theorem 2.2 does not assume convex option curves or correctness of the greedy heuristic. Its finite breakpoint set can be enumerated directly from the option table; each evaluation of $\Phi_{B,L}$ then decomposes into independent per-unit scans. A zero certified gap proves global optimality for the

additive surrogate. A nonzero gap remains a valid upper bound on suboptimality but need not be tight because the primal problem is discrete. The complete proof is given in Appendix B.

A diagnostic crossover length for a reference assignment $(\mathbf{b}, q)$ is

$$L_h^\star(\mathbf{b}, q) \coloneqq \frac{\sum_{u \in \mathcal{U}} \rho_h^{\mathrm{w}}(b_u) c_u(b_u)}{\rho_h^{\mathrm{kv}}(q) k(q)}. \tag{10}$$

At $L = L_h^\star$, predicted weight and KV traffic are equal. We use this quantity to interpret regimes, not as a substitute for solving Equation (7).

3 The ByteBudget planner

3.1 Cost calibration

For each target processor/runtime pair, we benchmark a small calibration design spanning context lengths, directly supported formats, and a limited set of mixed-format recipes. Packed tensor accounting supplies $c_u(b)$, while runtime cache accounting supplies $k(q)$. After anchoring reference factors as described above, the remaining parameters in Equation (3) are fit on the calibration split under nonnegativity and regularization constraints. We retain held-out configurations for testing extrapolation across lengths and interpolation across formats. This separation is necessary: fitting and evaluating on the same uniform-precision points would make a linear bytes–latency relation appear stronger than it is.

When only a short-context reference run is available, the prototype estimates β_h from measured token rate times streamed weight bytes. The full evaluation additionally fits τ_h and format factors ρ, compares nested models, and chooses the simplest specification whose held-out error is not materially worse than richer alternatives.

3.2 Curve construction and Pareto pruning

For every unit, we obtain pairs $(c_u(b), d_u(b))$. We remove any dominated option b: option b is dominated if another option b' has $c_u(b') \le c_u(b)$ and $d_u(b') \le d_u(b)$ with at least one strict inequality. We then order the remaining options by increasing effective cost,

$$\tilde{c}_{u,0} < \tilde{c}_{u,1} < \cdots < \tilde{c}_{u,m_u}, \qquad d_{u,0} > d_{u,1} > \cdots > d_{u,m_u}, \tag{11}$$

where $\tilde{c}$ includes the appropriate ρ factor. Equal-cost or equal-loss options are merged deterministically. The KV unit is processed identically, except that each incremental cost is multiplied by L.

This representation avoids two ambiguities. First, a larger nominal bit width is not assumed to be more expensive if its packing or kernel differs. Second, a measured quality curve need not be monotone before pruning; nonmonotone points are either dominated or retained only when they represent a genuine cost–quality trade-off.

3.3 Marginal-gain allocation

Each adjacent transition defines an upgrade edge $e \coloneqq (u, j)$ with

$$\Delta c_e \coloneqq \tilde{c}_{u,j+1} - \tilde{c}_{u,j} > 0, \qquad \Delta d_e \coloneqq d_{u,j} - d_{u,j+1} > 0, \tag{12}$$

and density $\eta_e \coloneqq \Delta d_e / \Delta c_e$. Starting from the minimum-cost option of every unit, the implemented planner repeatedly takes the available budget-feasible edge with largest density. Only the next edge of each unit is available, enforcing precedence between precision levels.

If $B \ge B_{\min}$, Algorithm 1 is feasible because it applies an edge only after checking $C + \Delta c_e \le B$. Let $M \coloneqq \sum_{u \in \mathcal{U}} m_u$ be the number of possible upgrades. Each edge is inserted and extracted at most once, and the heap contains at most one available edge per unit; hence a heap implementation uses $O((M + |\mathcal{U}|) \log(|\mathcal{U}| + 1))$ time and $O(|\mathcal{U}|)$ auxiliary memory.

The density rule is intuitive but not globally exact for arbitrary heterogeneous upgrade costs. A locally efficient large upgrade can preclude a combination of lower-density upgrades with greater total benefit. We therefore make no unconditional optimality claim. Section 3.5 compares the heuristic with a discretized dynamic-programming oracle and the global lower bound from Theorem 2.2.

Algorithm 1 Precedence-constrained marginal-gain planner

Require: Pareto-pruned units with ordered options; budget $B \geq B_{\min}$
initialize every unit at option index 0; set total cost $C \leftarrow B_{\min}$
insert the first edge of each unit into a max-heap keyed by η_e
while the heap is nonempty **do**
  extract the maximum-density edge $e = (u, j)$
  if $C + \Delta c_e > B$ **then**
    discard e and continue ▷ it can never become feasible later
  end if
  apply e; update $C \leftarrow C + \Delta c_e$
  expose and insert unit u's next edge, if any
end while
return selected weight options and KV format

3.4 Why context changes the plan

For a fixed KV transition $q_j \to q_{j+1}$, let $\Delta k > 0$ be the per-cached-token byte increase. Holding quality regime r fixed, let $\Delta d_{\mathrm{kv},r} > 0$ be its estimated quality gain. Its upgrade density at accounting context L is

$$\eta_{\mathrm{kv}}(L; r) \coloneqq \frac{\Delta d_{\mathrm{kv},r}}{L \rho_h^{\mathrm{kv}} \Delta k}. \tag{13}$$

For fixed r and a fixed KV transition with positive cost, Equation (13) gives $\eta_{\mathrm{kv}}(aL; r) = \eta_{\mathrm{kv}}(L; r)/a$ for every $a > 0$, while weight-upgrade costs are independent of L in Equation (2). If quality is re-estimated at each length, then

$$\frac{\eta_{\mathrm{kv}}(L_2; r_{L_2})}{\eta_{\mathrm{kv}}(L_1; r_{L_1})} = \frac{\Delta d_{\mathrm{kv},r_{L_2}}}{\Delta d_{\mathrm{kv},r_{L_1}}} \frac{L_1}{L_2}. \tag{14}$$

Thus, context can reorder KV and weight upgrades, but neither the cost model nor the prototype implies a universal monotone rule for the final KV precision.

3.5 Dynamic-programming oracle and dual certificate

For experiments, we discretize the variable effective costs by a resolution $\delta > 0$. Enumerate the units, including KV, by $j \in \{1, \ldots, J\}$, use the losses ℓ_j and costs $c_{j,L}$ from Equation (8), and define the residual budget $R \coloneqq B - c_{0,h}$, rounded option costs $\bar{c}_j(a) \coloneqq \lceil c_{j,L}(a)/\delta \rceil$, and integer capacity $Z \coloneqq \lfloor R/\delta \rfloor$. The recurrence is

$$F_j(z) \coloneqq \min_{a \in \mathcal{A}_j : \bar{c}_j(a) \leq z} \big\{ F_{j-1}(z - \bar{c}_j(a)) + \ell_j(a) \big\}, \qquad F_0(z) \coloneqq 0. \tag{15}$$

The minimum of an empty set in Equation (15) is defined as $+\infty$. This yields the optimum of the rounded-cost surrogate in $O(Z \sum_{j=1}^{J} |\mathcal{A}_j|)$ time. If the returned options satisfy $\sum_j \bar{c}_j(a_j) \leq Z$, then $\mathcal{C}_L(\mathbf{a}) \leq c_{0,h} + \delta \sum_j \bar{c}_j(a_j) \leq c_{0,h} + \delta Z \leq B$. Rounding may reject an assignment that is feasible before discretization, so we sweep δ and do not treat the rounded optimum as an exact solution of the original real-valued problem.

Theorem 2.2 supplies a complementary certificate without cost discretization. For each greedy plan, we report the rounded-DP gap $\widehat{\mathcal{D}}_{\mathrm{greedy}} - \widehat{\mathcal{D}}_{\mathrm{DP}}$, the certified gap $\widehat{\mathcal{D}}_{\mathrm{greedy}} - \underline{\mathcal{D}}(B, L)$, and the downstream gap after materialization. The last quantity is essential because a smaller additive surrogate need not translate into a better task score when cross-unit interactions are significant.

3.6 Recipe materialization and runtime audit

The planner emits per-unit weight formats and a KV format. The prototype expands attention- and MLP-block decisions to per-module quantization overrides, applies an existing weight quantizer, exports one mixed-precision model, and selects runtime key/value cache types. The scientific claim is about planning, not a particular file format. Therefore every evaluated configuration must pass

three audits: (i) the exported tensor format matches the plan, (ii) measured model and KV bytes match accounting within tolerance, and (iii) the runtime executes a native path rather than silently dequantizing to another format.

The current exporter directly packs only a subset of quantizer/format combinations; unsupported options can fall back through dequantization and requantization. Such points are excluded from the primary planner comparison unless the fallback is itself the declared baseline. This rule prevents an exporter artifact from being misattributed to the allocation algorithm.

4 Related work

LLM post-training quantization. Weight-only PTQ methods reduce storage and memory traffic using second-order error compensation, activation-aware scaling, calibrated transformations, or vector codebooks (Frantar et al., 2023; Lin et al., 2024; Wenqi et al., 2024; Chee et al., 2023; Tseng et al., 2024). Weight–activation quantization addresses activation outliers through smoothing, training-free reconstruction, learned parameters, rotations, or distribution correction (Xiao et al., 2023; Yao et al., 2022; Liu et al., 2023; Ashkboos et al., 2024; Liu et al., 2025; Zeng et al., 2025). Perturbation-based analysis shows that error placement and clipping matter beyond aggregate magnitude (Gong et al., 2024); SpecQuant studies spectral decomposition and channel-adaptive truncation (Zhao et al., 2026). These works primarily improve the quality of individual quantized operators or formats. ByteBudget treats their outputs as candidate quality–cost points and decides how to combine them under a CPU latency budget.

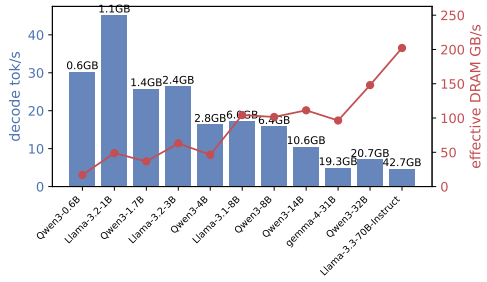
Mixed-precision allocation. Hardware-aware mixed-precision methods learn or optimize per-layer bit widths using reinforcement learning, Hessian information, or orthogonality-based criteria (Kuan et al., 2019; Dong et al., 2019; 2020; Ma et al., 2023). Their budgets are typically model size, operations, or accelerator latency, and they focus on weights and activations. ByteBudget differs by jointly charging static weight traffic and context-scaled KV traffic in the same per-token objective. It also separates allocation from sensitivity estimation so stronger quantizers can be substituted without changing the budget semantics.

KV-cache compression and serving systems. KIVI and KVQuant develop low-bit cache quantization with outlier-aware or nonuniform treatment (Liu et al., 2024; Hooper et al., 2024); Atom and QServe jointly design quantization and GPU serving kernels (Zhao et al., 2024; Lin et al., 2025). A practical reassessment demonstrates that cache compression alone may not improve end-to-end serving when kernels and output-length effects are ignored (Gao et al., 2025). More recent systems allocate precision within the KV cache or adapt model state under workload pressure: Kitty boosts sensitive key channels (Haojun et al., 2026), Quantize What Counts argues for key-favored precision (Hariri et al., 2026), and MorphServe swaps quantized layers while resizing KV capacity on GPU servers (Su et al., 2026). ByteBudget is complementary but distinct: it plans precision jointly across weight blocks and the KV representation for a specified low-batch CPU context and latency target, then audits whether the backend realizes the predicted traffic.

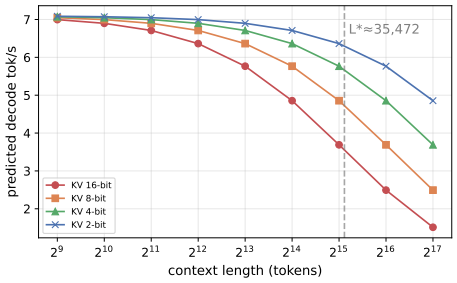
5 Experiments

Our experiments investigate following questions: (1) Does memory bandwidth impose a constraint on the decoding process in CPU-based LLMs? (2) Does KV precision cause latency in long contexts? (3) Can ByteBudget's bit-size allocation reduce a model's prediction loss? (4) What is the relationship between the quantization bit size and the reduction in prediction error for ByteBudget? (5) Is there a proportional correlation between model size and memory bandwidth?

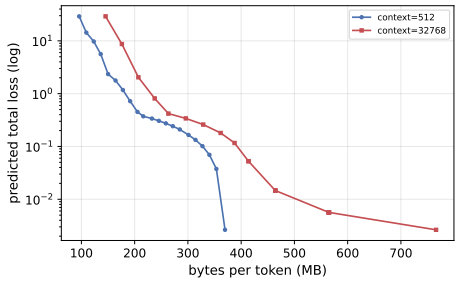
(1) Does memory bandwidth impose a constraint on the CPU-based LLM decoding process? The memory bandwidth significantly affects performance in the decoding process on CPU.The Figure2a shows the data size of each LLM model, along with its throughput and DRAM bandwidth. It can be seen that as the model's data size (bit size) increases—that is, as bandwidth increases—throughput decreases. Hence, it can be said that memory bandwidth is proportional to the latency of LLM model inference. In other words, an increase in memory bandwidth directly leads to decrease in throughput.



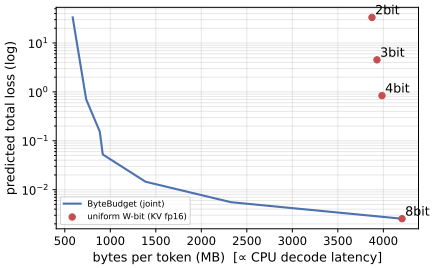
(a) Model size, throughput, and DRAM bandwidth.



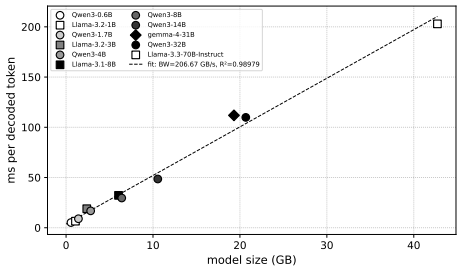
(b) Context-length dependence under different KV-cache precisions.



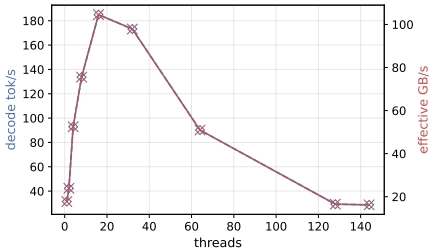
(c) Precision allocation produced by ByteBudget.



(d) Quality–traffic trade-off with ByteBudget.



(e) Relationship between decoding latency and model traffic.



(f) Throughput and memory-bandwidth scaling with CPU threads.

Figure 2: **Experimental evaluation of ByteBudget.** (a) Relationship among model size, decoding throughput, and DRAM bandwidth. (b) Effect of KV-cache precision and context length on decoding throughput. (c) Precision allocation selected by ByteBudget. (d) Quality–traffic trade-off obtained by ByteBudget compared with uniform quantization. (e) Relationship between decoding latency and model traffic. (f) Scaling of throughput and memory bandwidth with the number of CPU threads.

(2) Does KV precision cause latency in long contexts? The Figure2b illustrates the relationship between context length and throughput when the bit size used for KV cache quantization is varied.The boundary context length L^* between the weight and the KV cache, calculated using ByteBudget, is shown as the gray dashed line in the figure. This figure indicates that, for context lengths exceeding L^*, the impact of KV cache computations on overall throughput becomes significant. In particular, it can be seen that the degree of degradation increases with larger bit sizes in quantization.

(3) Can ByteBudget's bit-size allocation reduce a model's prediction loss? Figure 2c shows the relationship between the number of bytes per token and the prediction error of the LLM model when ByteBudget is applied to two different context lengths. It can be seen that as the context length increases, more bytes are required per token.ByteBudget solves an optimization problem to allocate a larger portion of the budget to the KV cache as the context length increases. Figure 2d shows the performance of the original LLM model and the simulation results when ByteBudget is applied. It

can be seen that using ByteBudget achieves a similar level of prediction error for the LLM model with fewer bytes per token.

(4) What is the relationship between the quantization bit size and the reduction in prediction loss for ByteBudget? The Figure2d illustrates the effect of ByteBudget on the inference accuracy of LLMs.First, the scattered data points in the figure represent the actual measured inference accuracy of Llama 3.2 1B quantized at 2-bit, 3-bit, 4-bit, and 8-bit. The solid-line plots show the error in Llama 3.2 1B when ByteBudget is applied, plotted by varying the number of bytes per token.As mentioned earlier, the number of bytes per token is directly proportional to the decoding latency on the CPU. It is evident that assigning a budget via ByteBudget results in a significant improvement in prediction error compared to the original quantized model.

(5) Is there a proportional correlation between model size and memory bandwidth? Figure 2e shows, as a dashed line, the regression model that predicts the model size of LLM models and the latency of the decoding process. In addition, we plotted the actual measured values of the data size and decoding latency for each LLM model. These results show that decoding latency increases as the model's data size increases. In this context, an increase in the model's data size is equivalent to an increase in the model's parameter size. Therefore, it can be said that decoding latency increases as the model's parameter size increases.Figure 2f shows the relationship between the number of threads, throughput, and memory bandwidth on a CPU. These results suggest that applying ByteBudget allows for good performance when running LLMs with model sizes up to 10B using about 16 threads.

6 Scope and limitations

Additive quality is an approximation. Equation (4) ignores interactions among quantized units. Error can compound across layers, and weight and KV perturbations can interact through attention. We treat the surrogate as a planner score rather than a theorem about downstream risk. The rounded-cost DP is exact only for its discretized additive surrogate, while Theorem 2.2 bounds the optimum of the original additive surrogate. Single-unit correlations, cross-intervention tests, and downstream evaluation determine whether a lower surrogate loss is meaningful.

A planned format is not a realized kernel. Nominal low precision is useful only when the runtime stores and processes that format efficiently. This is why ByteBudget includes format factors and a materialization audit. Results obtained through fallback conversion are not evidence for a native arbitrary-bit CPU kernel. Conversely, the planner can benefit from improved kernels by updating the measured cost table without changing the optimization formulation.

Support for MoE Models We applied ByteBudget to dense LLM models, conducted various experiments, and obtained favorable results. However, LLM models adopting the MoE (Mixure of Experts) architecture have been announced in recent years. The adoption of it has made it possible to run LLM models that were previously difficult to execute in general computing environments.ByteBudget is not currently designed for MoE. As a future challenge, we aim to enhance ByteBudget by adding MoE-specific processing capabilities, thereby enabling users to run LLM models with larger parameter sizes in their own environments.

7 Conclusion

We formulated bandwidth-bound CPU LLM decoding as a context-dependent allocation of effective traffic between heterogeneous weight blocks and the KV cache. ByteBudget converts a latency target into a calibrated budget, combines modular quality curves with format-aware costs, and returns a deployable mixed-precision recipe. Its central scientific question is not whether lower precision reduces nominal size, but whether the right bytes are removed at the right context and whether the runtime realizes that reduction. The completed experiments will determine the validity range of the cost model, the benefit of joint planning, and the gap between the efficient greedy solver, the rounded-cost dynamic-programming oracle, and the Lagrangian lower bound. More broadly, the formulation provides a common interface through which future quantizers and CPU kernels can be compared under explicit latency constraints rather than isolated bit-width labels.

AI USE STATEMENT

Generative AI tools were used to assist with editing the manuscript for clarity and readability and with writing and editing software code. All AI-assisted text and code were reviewed by the authors, and the code was tested for correctness. Generative AI was not used to develop the research ideas, theoretical formulation, mathematical claims, experimental methodology, or interpretation of the results. The authors take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Implementation details and experimental settings are provided in the main text and supplementary material. Source code required to reproduce the experiments will also be provided as supplementary material.

ETHICS STATEMENT

This work does not involve human subjects or personally identifiable information, and we are not aware of specific ethical concerns arising from this study.

REFERENCES

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-3180.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M. De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0196.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/c3ba4962c05c49636d4c6206a97e9c8a-Abstract-Conference.html.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 293–302, 2019.

Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ-V2: Hessian aware trace-weighted quantization of neural networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.17323.

Wei Gao, Zhou Xinyu, Sun Peng, Zhang Tianwei, and Wen Yonggang. Rethinking key-value cache compression techniques for large language model serving. In *Proceedings of Machine Learning and Systems*, volume 7, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/hash/26289c647c6828e862e271ca3c490486-Abstract-Conference.html.

Zhuocheng Gong, Jiahao Liu, Jingang Wang, Xunliang Cai, Dongyan Zhao, and Rui Yan. What makes quantization for large language models hard? an empirical study from the lens of perturbation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18082–18089, 2024. doi: 10.1609/aaai.v38i16.29765.

Xia Haojun, wu Xiaoxia, Li Jisen, Wu Tsai-chuan, Wang Junxiong, Li Jue Wang, Chenxi, Singhal Aman, Shah Alay, Dilipbhai, Ariyak Alpay, Zhuang Donglin, Zhou Zhongzhu, Athiwaratkun Ben, Zheng Zhen, and Song Shuaiwen. Kitty: Accurate and efficient 2-bit KV cache quantization with dynamic channel-wise precision boost. In *Proceedings of Machine Learning and Systems*, volume 8, 2026. URL https://proceedings.mlsys.org/paper_files/paper/2026/hash/e4d8d1b5120be349d3fff8878650cf45-Abstract-Conference.html.

Mohsen Hariri, Alan Luo, Weicong Chen, Tianyi Zhang, Qifan Wang, Xiaotian Han, and Vipin Chaudhary. Quantize what counts: More for keys, less for values. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 26393–26420. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1314.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0040.

Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Springer, 2004. doi: 10.1007/978-3-540-24777-7.

Wang Kuan, Liu Zhijian, Lin Yujun, Lin Ji, and Han Song. HAQ: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8612–8620, 2019.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. QServe: W4A8KV4 quantization and system co-design for efficient LLM serving. In *Proceedings of Machine Learning and Systems*, volume 7, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/hash/fbe2b2f74a2ece8070d8fb073717bda6-Abstract-Conference.html.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. LLM-QAT: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: LLM quantization with learned rotations. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/e5b1c0d4866f72393c522c8a00eed4eb-Abstract-Conference.html.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32332–32344. PMLR, 2024. URL https://proceedings.mlr.press/v235/liu24bz.html.

Yuexiao Ma, Taisong Jin, Xiawu Zheng, Yan Wang, Huixia Li, Yongjian Wu, Guannan Jiang, Wei Zhang, and Rongrong Ji. OMPQ: Orthogonal mixed precision quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 9029–9037, 2023. doi: 10.1609/aaai.v37i7.26084.

NVIDIA Corporation. Nvidia grace cpu superchip whitepaper. Technical report, NVIDIA Corporation, 2024. URL https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-cpu-superchip.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems*, volume 5, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/hash/c4be71ab8d24cdfb45e3d06dbfca2780-Abstract-mlsys2023.html.

Zhaoyuan Su, Zeyu Zhang, Tingfeng Lan, Zirui Wang, Haiying Shen, Juncheng Yang, and Yue Cheng. Morphserve: Efficient and workload-aware LLM serving via runtime quantized layer swapping and KV cache resizing. In *Proceedings of Machine Learning and Systems*, volume 8, 2026. URL https://proceedings.mlsys.org/paper_files/paper/2026/hash/8144a9d62e506af0fcdeac0e456b2710-Abstract-Conference.html.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 48630–48656. PMLR, 2024. URL https://proceedings.mlr.press/v235/tseng24a.html.

Shao Wenqi, Chen Mengzhao, Zhang Zhaoyang, Xu Peng, Zhao Lirui, Li Zhiqian, Zhang Kaipeng, Peng Gao, Qiao Yu, and Luo Ping. Omniquant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/c6483c8a68083af3383f91ee0dc6db95-Abstract-Conference.html.

Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009. doi: 10.1145/1498765.1498785.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-1970.

Zhewei Yao, Xiaoxia Wu, Cheng Li, Stephen Youn, and Yuxiong He. Exploring post-training quantization in LLMs from comprehensive study to low rank compensation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19377–19385, 2024. doi: 10.1609/aaai.v38i17.29908.

Chao Zeng, Songwei Liu, Yusheng Xie, Hong Liu, Xiaojian Wang, Miao Wei, Shu Yang, Fangmin Chen, and Xing Mei. ABQ-LLM: Arbitrary-bit quantized inference acceleration for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 22299–22307, 2025. doi: 10.1609/aaai.v39i21.34385.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate LLM serving. In *Proceedings of Machine Learning and Systems*, volume 6, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/5edb57c05c81d04beb716ef1d542fe9e-Abstract-Conference.html.

Zhixiong Zhao, Fangxin Liu, Junjie Wang, Chenyang Guan, Zongwu Wang, Li Jiang, and Haibing Guan. Specquant: Spectral decomposition and adaptive truncation for ultra-low-bit LLMs quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 28786–28794, 2026. doi: 10.1609/aaai.v40i34.40112.

A Experimental Conditions

We ran the experiment under the following conditions. For detailed information about the NVIDIA Grace CPU, see (NVIDIA Corporation, 2024).

Table 1: Experimental Conditions

Item	Value
CPU	NVIDIA Grace Neoverse-V2 (72 Core 72 Threads) 2 Sockets
DRAM	LPDDR5X 500GB
Quantization format	QEP + Q4_0
Performance measurement cicles	5(2 warm up cicles)
Benchmark	Salesforce / wikitext
Quantization Framework	OneCompression v1.3.3
Inference Framework	Llama.cpp v0.3.0

Table 2: Model Families

Series	Parameter Size
Qwen/Qwen3	0.6B
Qwen/Qwen3	1.7B
Qwen/Qwen3	4B
Qwen/Qwen3	8B
Qwen/Qwen3	14B
Qwen/Qwen3	32B
meta-llama/Llama-3.2	1B
meta-llama/Llama-3.2	3B
meta-llama/Llama-3.1	8B
meta-llama/Llama-3.3	70B
google/gemma-4	31B

B Proofs and optimization details

B.1 Proof of Theorem 2.2

Proof. Because every option set $\mathcal{A}_j$ is finite and nonempty, the product set $\prod_{j\in\mathcal{J}}\mathcal{A}_j$ is finite. Moreover, $B \geq B_{\min,h}(L)$ makes the assignment that selects a minimum-cost option for every unit feasible. Hence the feasible set in Equation (8) is nonempty and finite, so $\mathcal{D}^\star(B, L)$ is attained.

Part (i). For each $j \in \mathcal{J}$ and $a \in \mathcal{A}_j$, define $f_{j,a}(\lambda) := \ell_j(a) + \lambda c_{j,L}(a)$ and $\phi_j(\lambda) := \min_{a\in\mathcal{A}_j} f_{j,a}(\lambda)$. For $\lambda_1, \lambda_2 \geq 0$ and $\theta \in [0, 1]$, affine linearity gives

$$\phi_j(\theta\lambda_1 + (1-\theta)\lambda_2) = \min_{a\in\mathcal{A}_j}\{\theta f_{j,a}(\lambda_1) + (1-\theta)f_{j,a}(\lambda_2)\} \geq \theta\phi_j(\lambda_1) + (1-\theta)\phi_j(\lambda_2).$$

Thus ϕ_j is concave. Since $\mathcal{A}_j$ is finite, ϕ_j is the pointwise minimum of finitely many continuous affine functions; consequently, it is continuous and piecewise affine. Equation (9) writes $\Phi_{B,L}$ as an affine function plus the finite sum $\sum_{j\in\mathcal{J}}\phi_j$, so $\Phi_{B,L}$ is also continuous, concave, and piecewise affine.

We next identify all intervals on which $\Phi_{B,L}$ is affine. For a fixed unit j, the difference $f_{j,a} - f_{j,a'}$ of two nonidentical option functions is a nonzero affine function and therefore has at most one zero. Every nonnegative zero of this form belongs to $\mathcal{S}_L$ by definition. If two option functions are identical, they have the same value for every λ and can be treated as one function without changing ϕ_j. Let I be any connected component of $[0, \infty) \setminus \mathcal{S}_L$. For every pair of nonidentical option functions of the same unit, the sign of their difference is constant on I; otherwise, continuity would give a zero in I, contradicting the definition of I. Hence the relative ordering of all option functions is fixed on I, and

ϕ_j coincides on I with a single affine function. Therefore $\Phi_{B,L}$ is affine on every such component. Since each unordered pair contributes at most one nonnegative intersection point,

$$|\mathcal{S}_L| \le 1 + \sum_{j\in\mathcal{J}} \binom{|\mathcal{A}_j|}{2}.$$

It remains to show that the maximum cannot escape to infinity. For each $j \in \mathcal{J}$, let $c_j^{\min} := \min_{a\in\mathcal{A}_j} c_{j,L}(a)$ and choose

$$a_j^0 \in \arg\min\left\{\ell_j(a) : a \in \mathcal{A}_j,\ c_{j,L}(a) = c_j^{\min}\right\}.$$

Define

$$\Lambda_j := \max\left(\{0\} \cup \left\{\frac{\ell_j(a_j^0) - \ell_j(a)}{c_{j,L}(a) - c_j^{\min}} : a \in \mathcal{A}_j,\ c_{j,L}(a) > c_j^{\min}\right\}\right), \qquad \Lambda := \max_{j\in\mathcal{J}} \Lambda_j.$$

For every $\lambda \ge \Lambda_j$ and every higher-cost option a, the definition of Λ_j implies $\ell_j(a_j^0) + \lambda c_j^{\min} \le \ell_j(a) + \lambda c_{j,L}(a)$. The same inequality holds for equal-cost options by the choice of a_j^0. Hence a_j^0 minimizes the jth Lagrangian term for every $\lambda \ge \Lambda$. With $C_{\min} := c_{0,h} + \sum_{j\in\mathcal{J}} c_j^{\min} = B_{\min,h}(L)$, we obtain

$$\Phi_{B,L}(\lambda) = \sum_{j\in\mathcal{J}} \ell_j(a_j^0) + \lambda(C_{\min} - B), \qquad \lambda \ge \Lambda.$$

Because $B \ge C_{\min}$, this eventual affine representation has nonpositive slope.

Sort the distinct points of $\mathcal{S}_L$ as $0 = s_0 < s_1 < \cdots < s_K$. On each bounded interval $[s_i, s_{i+1}]$, the function $\Phi_{B,L}$ is affine on (s_i, s_{i+1}) and continuous at the endpoints, so it agrees on the closed interval with the continuous extension of that affine function and attains its maximum at an endpoint. On the final component (s_K, ∞), $\Phi_{B,L}$ is affine. Its slope equals $C_{\min} - B \le 0$ on the nonempty tail $[\max\{s_K, \Lambda\}, \infty)$ and, being constant on the whole final component, is therefore nonpositive throughout (s_K, ∞). Continuity then implies that the maximum on $[s_K, \infty)$ is attained at s_K. Consequently,

$$\max_{\lambda\ge 0} \Phi_{B,L}(\lambda) = \max_{\lambda\in\mathcal{S}_L} \Phi_{B,L}(\lambda),$$

which proves part (i), including the finite-search claim.

Part (ii). Fix $\lambda \ge 0$ and any feasible assignment $\mathbf{a}$. Evaluating every inner minimum in Equation (9) at a_j gives

$$\Phi_{B,L}(\lambda) \le \lambda(c_{0,h} - B) + \sum_{j\in\mathcal{J}} (\ell_j(a_j) + \lambda c_{j,L}(a_j)) = \mathcal{D}(\mathbf{a}) + \lambda(\mathcal{C}_L(\mathbf{a}) - B) \le \mathcal{D}(\mathbf{a}),$$

where the last inequality uses $\lambda \ge 0$ and feasibility, namely $\mathcal{C}_L(\mathbf{a}) \le B$. Since this holds for every feasible $\mathbf{a}$, $\Phi_{B,L}(\lambda) \le \mathcal{D}^\star(B, L)$ for every $\lambda \ge 0$. Maximizing over λ yields $\underline{\mathcal{D}}(B, L) \le \mathcal{D}^\star(B, L)$. For any feasible $\widehat{\mathbf{a}}$, primal optimality gives $\mathcal{D}^\star(B, L) \le \mathcal{D}(\widehat{\mathbf{a}})$. Subtracting these inequalities from $\mathcal{D}(\widehat{\mathbf{a}})$ proves

$$0 \le \mathcal{D}(\widehat{\mathbf{a}}) - \mathcal{D}^\star(B, L) \le \mathcal{D}(\widehat{\mathbf{a}}) - \underline{\mathcal{D}}(B, L).$$

Part (iii). Suppose that a feasible assignment $\widehat{\mathbf{a}}$ and $\lambda^\star \ge 0$ satisfy the stated per-unit minimizing conditions and complementary slackness. The minimizing conditions allow every inner minimum in Equation (9) to be evaluated at $\widehat{a}_j$, while complementary slackness gives $\lambda^\star(\mathcal{C}_L(\widehat{\mathbf{a}}) - B) = 0$. Therefore

$$\Phi_{B,L}(\lambda^\star) = \mathcal{D}(\widehat{\mathbf{a}}) + \lambda^\star(\mathcal{C}_L(\widehat{\mathbf{a}}) - B) = \mathcal{D}(\widehat{\mathbf{a}}).$$

By weak duality, $\Phi_{B,L}(\lambda^\star) \le \mathcal{D}^\star(B, L)$, whereas feasibility of $\widehat{\mathbf{a}}$ gives $\mathcal{D}^\star(B, L) \le \mathcal{D}(\widehat{\mathbf{a}})$. The displayed equality places the same quantity at both ends, so both inequalities are equalities. Thus $\widehat{\mathbf{a}}$ is globally optimal and $\mathcal{D}(\widehat{\mathbf{a}}) = \mathcal{D}^\star(B, L) = \Phi_{B,L}(\lambda^\star)$. □

B.2 CONTEXT-SCALING IDENTITY

Fix quality-calibration regime r. Then $\Delta d_{\mathrm{kv},r}$ and Δk are constants for a fixed transition, while its effective per-step incremental traffic is $L\rho_h^{\mathrm{kv}}\Delta k$. Substitution into Equation (13) gives $\eta_{\mathrm{kv}}(aL; r) = \eta_{\mathrm{kv}}(L; r)/a$ for every $a > 0$. If quality is recalibrated with context, division of the two densities gives Equation (14). Weight upgrade costs have no factor L in Equation (2), although their measured quality gains may change when the calibration regime changes.

B.3 WHY DENSITY GREEDY IS NOT CLAIMED OPTIMAL

The production solver is a precedence-constrained density heuristic. General density greedy is not exact for heterogeneous discrete costs: a high-density upgrade can consume enough budget to block a combination of lower-density upgrades whose total gain is larger. The rounded-cost DP in Equation (15) and the lower bound in Theorem 2.2 therefore serve different audit roles. The former gives the optimum of a conservative discretized instance, whereas the latter bounds the optimum of the original additive surrogate without discretizing costs.

B.4 FEASIBILITY OF THE ROUNDED-COST ORACLE

If the DP returns options satisfying $\sum_j \bar{c}_j(a_j) \le Z$, then $\mathcal{C}_L(\mathbf{a}) = c_{0,h} + \sum_j c_{j,L}(a_j) \le c_{0,h} + \delta \sum_j \bar{c}_j(a_j) \le c_{0,h} + \delta Z \le B$. Thus ceiling option costs cannot create a budget violation; they can only reject assignments that would fit before rounding. The resolution sweep in RQ5 quantifies this conservatism.

B.5 BUDGET NORMALIZATION

For user-facing frontier sweeps, a normalized budget parameter $s \in [0, 1]$ may be mapped to

$$B_h(s; L) \coloneqq B_{\min,h}(L) + s\,(B_{\max,h}(L) - B_{\min,h}(L))\,. \tag{16}$$

This parameterization is convenient for visualization but is not a hardware service-level objective. Headline results use measured latency targets and Equation (6).