EXACT VOCABULARY DECODING FROM FAR MEMORY WITH GROUPWISE RESIDUAL CERTIFICATES

Anonymous authors
Paper under double-blind review

ABSTRACT

An autoregressive language model scores every vocabulary row but returns one token. When the output head resides outside accelerator memory, a dense projection must read the full matrix at each step. We show how to preserve the exact decision while retrieving only rows whose upper bounds can beat one exactly evaluated row. Our decoder keeps grouped integer representations and residual error bounds on the accelerator, while the original bfloat16 (BF16) rows remain in host memory. Integer dot products and outward rounding give intervals that contain the correctly rounded BF16 scores. A tie-aware screening rule selects the rows to retrieve and preserves greedy and fixed-counter Gumbel decisions. With groups of 512 coordinates, the resident representation occupies 51.2% of a dense BF16 output head and requires exact evaluation of only 0.005% to 2.190% of rows on average across four primary models, with no failure in 10,240 decisions. Six-model tests and paired generation paths also show no token mismatch. Request time falls by 21.2% to 77.8% against the same canonical dense decoder. These results show that exact offloaded decoding can use sparse row retrieval in place of full-matrix transfer.

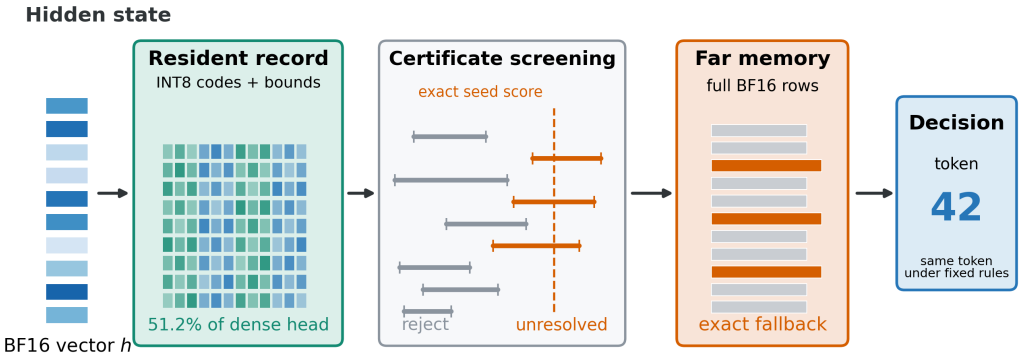


Figure 1: Exact decoding can fetch only rows that remain able to win. Grouped 8-bit integer (INT8) codes and outward residual bounds stay on the accelerator. After one seed row is scored exactly, intervals reject dominated rows and full BF16 rows are fetched only for unresolved candidates. The fixed tie rule and counter then return the same token as the dense reference. Group width 512 (G512) occupies 51.2% of a dense BF16 head when the hidden width is divisible by 512.

1 INTRODUCTION

An autoregressive language model may score more than 100,000 output rows to return one token (Grattafiori et al., 2024; Yang et al., 2025; OLMo Team et al., 2025). Even after transformer blocks are quantized, the output head can remain a large dense tensor, which has made it an active target for compression and retrieval (Tranheden et al., 2026; Kocabay et al., 2026). Weight tying removes a duplicate matrix (Press & Wolf, 2017; Inan et al., 2017), but every decoding step still applies the

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.

full projection. Moving the head to host memory releases high-bandwidth memory (HBM), yet a dense exact decoder must then read the full matrix for every token. Which original rows must cross the memory boundary to return exactly the same token?

We study this question with the full output matrix in *far memory*, meaning memory outside HBM. In our experiments, far memory is pinned host storage, as shown in Figure 1. A compact certificate stays resident, while an original row crosses the boundary only when it remains able to win. Because the original bfloat16 (BF16) row remains authoritative, exact offload becomes a retrieval problem: resident evidence determines which rows must be read.

Prior work reduces large-vocabulary cost through sampled or adaptive softmax (Jean et al., 2015; Grave et al., 2017), task-specific vocabulary selection with on-demand loading (Zhang et al., 2026), compressed draft heads (Zhao et al., 2025; Plaksin et al., 2026), and quantized or retrieval-based heads (Shao et al., 2025; Tranheden et al., 2026; Kocabay et al., 2026). Certified sub-vocabularies preserve local top-k selection (Liu et al., 2025), and fused kernels avoid materializing local logits during exact sampling (Ruiz et al., 2026). Preserving task quality or a local selection rule alone does not certify a remote row's exclusion. Exact far-memory decoding requires that evidence before the original row is retrieved. An executable finite-precision interval provides it.

One exactly evaluated row provides a threshold for exclusion. Any row whose certified upper endpoint cannot beat this seed score can be discarded without retrieval. We construct each interval from an integer center and an outward residual radius. Independent coordinate groups use their own scales, INT8 codes, and reconstruction-error norms. Groupwise Cauchy bounds contain the canonical score, and the remaining BF16 rows use the same kernel as the dense reference.

Coordinate grouping controls the balance between resident metadata and remote traffic. For fixed reconstructions, groupwise norm products sum to no more than the corresponding global product, while independent scales can further reduce reconstruction error. Smaller groups can tighten the intervals at the cost of more metadata. For hidden width d and group width g, the resident fraction is $(d + 12\lceil d/g \rceil)/(2d)$. We use G512 as the primary operating point and test widths 256 and 1,024 separately.

Exact remote retrieval is sparse in our experiments. G512 stores 51.2% of a dense BF16 head and evaluates only 0.005% to 2.190% of rows exactly on average. A centroid-radius control under the same exact-decoding contract retains 99.77% to 99.89% of rows, and a lower-resident 5-bit control (W5) retains 4.7 to 29.7 times as many. G512 preserves all 10,240 primary decisions, extends to six models and paired generation paths, and lowers canonical request time by 21.2% to 77.8%. Measurements against stock kernels and at the full-request capacity boundary then locate the remaining systems cost.

2 Related Work

2.1 Large vocabularies and reduced output heads

Large-vocabulary methods reduce how many output rows an expensive path must score. Jean et al. (2015) approximate training normalization by sampling, while adaptive softmax uses frequency-based clusters to reduce expected computation (Grave et al., 2017). VocabTailor combines embedding offload with static and dynamic head-vocabulary selection for on-demand loading (Zhang et al., 2026). FR-Spec and SlimSpec reduce speculative draft cost and retain a full verifier (Zhao et al., 2025; Plaksin et al., 2026). These methods target task quality or speculative acceptance. We preserve a fixed full-vocabulary decision when the original rows reside outside HBM.

Head compression replaces the dense projection with a cheaper representation. VQ-Logits uses a vector-quantized codebook (Shao et al., 2025). ARCHead combines a quantized low-rank core with grouped residual correction (Kocabay et al., 2026), while FlashHead combines clustering, multi-probe retrieval, and selective quantization for model-level speed (Tranheden et al., 2026). General low-precision inference similarly changes the arithmetic path (Dettmers et al., 2022). These systems evaluate the quality or speed of the replacement head. Our resident reconstruction has a different role: it certifies exclusion, while the original BF16 rows remain authoritative.

2.2 CERTIFIED SELECTION AND EXACT SAMPLING

Exact selection asks which candidates can be removed without changing the maximizer. Exact maximum inner-product search (MIPS) uses geometric indices for this purpose (Abuzaid et al., 2019). CSV-Decode applies centroid-radius bounds to certify a dynamic sub-vocabulary for exact top-k selection and also analyzes approximate softmax (Liu et al., 2025). In our setting, the index occupies resident memory and each surviving row incurs a far-memory fetch. The interval must therefore be compact, selective, and valid for the executable finite-precision score.

Exact sampling extends the selection problem to perturbed scores. A* sampling and perturbed nearest-neighbor methods use Gumbel perturbations under stated probabilistic models (Maddison et al., 2014; Mussmann et al., 2017). FlashSampling fuses exact sampling with a tiled local head and avoids writing full logits to HBM (Ruiz et al., 2026). We fix the random word, operation order, and tie rule, so monotone intervals can reproduce the same finite-counter decision when rows are remote.

2.3 MEMORY-AWARE INFERENCE AND NUMERICAL IDENTITY

Memory placement determines whether a component saving improves the full request. FlexGen schedules tensor classes across memory tiers (Sheng et al., 2023), vLLM pages key-value memory (Kwon et al., 2023), and FlashAttention reduces attention data movement (Dao et al., 2022). We therefore report output-head latency and full-request capacity separately. Numerical identity is equally important because floating-point reduction order can change a near-boundary output (Yuan et al., 2025). A canonical reference gives the theorem a fixed decision, while stock BF16 supplies the practical performance comparison.

3 EXACT FAR-MEMORY DECODING

3.1 SETTING AND RESOURCE BOUNDARY

The decoder relies on one invariant: every resident interval contains the reference score of its remote row. It reads one promising row exactly and discards each row whose upper endpoint cannot beat that score. Every unresolved row uses the same kernel as the dense reference. This contract is sufficient for exact selection under the tie rule below.

Let a decoder produce a BF16 hidden state $h \in \mathbb{B}^d$, where $\mathbb{B}$ is the set of finite BF16 values. The output head is $W \in \mathbb{B}^{V \times d}$ with row w_i for token i. Full rows reside in host memory. The accelerator stores a row-separable record R_i and may stage selected rows. We count $R = (R_1, \ldots, R_V)$ as resident output-head state. Host copies, transient buffers, the backbone, and the key-value cache are separate. Appendix F specifies tied-embedding retrieval. All six evaluated heads are bias-free, as detailed in Appendix C.2.

Resident head storage is one term in the complete request. If H_g is the resident certificate size, $B(b, n)$ is backbone and key-value memory for batch b and context n, and $S_g(b, n)$ is method staging, then

$$M_g(b, n) = B(b, n) + H_g + S_g(b, n). \tag{1}$$

A smaller H_g increases full-request capacity only when its saving exceeds the relevant change in $B + S_g$ near the capacity boundary. This relation motivates measuring capacity directly instead of inferring it from output-head storage.

3.2 CANONICAL SCORE AND DECISION

Exactness requires a numerical object that is independent of an unspecified parallel reduction. We define the canonical score as

$$z_i = \mathcal{C}(h, w_i) = \mathrm{RN}_{\mathrm{BF16}} \left(\sum_{j=1}^{d} h_j w_{ij} \right), \tag{2}$$

where BF16 operands are interpreted as exact real numbers, the sum inside parentheses is exact, and $\mathrm{RN}_{\mathrm{BF16}}$ performs one round-to-nearest-even operation. The implementation evaluates this definition with exponent-binned integer accumulation. Appendix C gives the finite construction and overflow checks.

Greedy decoding returns the smallest token index among maximizers of z_i. For sampling at temperature $T > 0$, a counter keyed by the state, step, and token produces a fixed finite Gumbel value g_i. The processed score is

$$x_i = P_{T,i}(z_i) = \mathrm{RN}_{64}(\mathrm{RN}_{64}(z_i/T) + g_i)\,. \tag{3}$$

Here, RN_{64} is round-to-nearest-even binary64 rounding. The same smallest-index rule resolves finite ties. We call the selected greedy or processed-score token the *canonical decision*. All exactness results use this program. Stock matrix multiplication and multinomial sampling define separate numerical programs.

3.3 Screening contract

A valid record produces an interval $[L_i, U_i]$ containing the relevant canonical or processed score. Screening discards row i only after another row has an exact score that dominates U_i under the tie rule. Every remaining row is fetched and evaluated with Equation 2. Invalid metadata, integer overflow, or failed containment stops execution rather than returning an approximate token.

4 Groupwise Residual Certificates

4.1 Integer centers and residual radii

Partition the hidden coordinates into disjoint groups $G_1, \ldots, G_m$ of width at most g. Within group r, quantize the hidden state and row independently as

$$\widehat{h}_r = a_r p_r, \qquad \widehat{w}_{ir} = s_{ir} q_{ir}, \tag{4}$$

where p_r and q_{ir} are signed INT8 vectors and a_r, s_{ir} are finite 32-bit floating-point (FP32) max-absolute scales. Nearest-even rounding and clipping to $[-127, 127]$ give the codes. Appendix C.2 fixes the complete rule. The exact real-valued center for row i is

$$c_i = \sum_{r=1}^{m} a_r s_{ir} (p_r^\top q_{ir}). \tag{5}$$

Each integer dot product accumulates in a 32-bit integer (INT32). The implementation checks $|G_r| 127^2 < 2^{31}$ for every real group, including a shorter final group. It evaluates Equation 5 in 64-bit floating point (FP64). We denote the resulting executable center by $\widetilde{c}_i$.

The certificate stores two outward norms for each weight group. Define

$$e_{ir} = \|w_{ir} - \widehat{w}_{ir}\|_2, \quad r_{ir} = \|\widehat{w}_{ir}\|_2, \quad e_{h,r} = \|h_r - \widehat{h}_r\|_2. \tag{6}$$

The three stored FP32 values are (s_{ir}, e_{ir}, r_{ir}). Hidden quantities are computed once per state. Expanding $h_r^\top w_{ir} - \widehat{h}_r^\top \widehat{w}_{ir}$ and applying Cauchy-Schwarz in each group gives

$$|h^\top w_i - \widetilde{c}_i| \le \sum_{r=1}^{m} (\|h_r\|_2 e_{ir} + e_{h,r} r_{ir}) + \eta_i^{\mathrm{fin}}. \tag{7}$$

The finite term η_i^{fin} upper-bounds the FP64 evaluation error in $\widetilde{c}_i$. Weight residual and reconstruction norms are stored as upward-rounded FP32 values, and hidden norms receive the same treatment once per state. The nonnegative radius reduction and endpoint operations are also rounded outward. Appendix C specifies the operation counts and directed conversions.

4.2 Tighter bounds from coordinate groups

Coordinate groups tighten the real-valued certificate for fixed reconstructions. Cauchy-Schwarz across the group index gives

$$\sum_r \|h_r\|_2 e_{ir} \le \|h\|_2 \left(\sum_r e_{ir}^2 \right)^{1/2}, \tag{8}$$

and an analogous inequality holds for the hidden residual term. The groupwise radius is therefore no larger than its global counterpart for the same reconstruction, and independent scales can further reduce reconstruction error. Changing g also changes the reconstruction and finite rounding terms, so the resulting candidate traffic is measured empirically.

4.3 Finite endpoints and sampling

Let Δ_i denote the outward nonnegative right side of Equation 7. Directed operations form real-score endpoints $\widetilde{c}_i - \Delta_i$ and $\widetilde{c}_i + \Delta_i$, which are mapped outward through the monotone canonical BF16 rounding function. Equation 3 is also monotone in its score argument for $T > 0$. Applying the same finite operations to both endpoints therefore contains x_i for the fixed counter value used by both paths.

Proposition 1 (Interval containment)**.** *If the stored norms upper-bound their exact residual norms and every finite operation is rounded outward as specified, then the interval produced for row i contains its canonical score. The transformed interval also contains the fixed finite-counter score in Equation 3.*

The proof follows directly from Equation 7, monotonicity of BF16 rounding, and monotonicity of $P_{T,i}$. Appendix D states the finite assumptions and proof in full.

4.4 Tie-aware exact-row retrieval

The screening algorithm first computes all intervals in vocabulary chunks and chooses $b = \min \arg\max_i L_i$. It fetches row b and obtains its exact canonical or processed score s_b. A row with index below b remains a candidate when $U_i \geq s_b$. A row above b remains a candidate when $U_i > s_b$. Row b is always included. All candidates are fetched in one or more capacity-bounded batches, scored by Equation 2, and reduced with the fixed tie rule.

Proposition 2 (Decision and path identity)**.** *Assume interval containment and exact candidate evaluation. The screened decision equals the canonical full-vocabulary decision. Applying this statement at every autoregressive step with the same counter state yields identical token paths by induction.*

The strict and non-strict inequalities are necessary for smallest-index ties. Any excluded row is provably unable to beat the exact seed, while the final exact reduction covers every row that remains able to do so. Appendix D gives the one-step proof and path induction.

4.5 Resident layout and execution

Each row stores d INT8 code bytes and three FP32 values for each of $m = \lceil d/g \rceil$ groups. Its resident fraction relative to a dense BF16 row is

$$\rho_g(d) = \frac{d + 12\lceil d/g \rceil}{2d}. \tag{9}$$

For dimensions divisible by 512, G512 uses 0.51171875 of the dense head. A shorter final group is charged in full for its three metadata values but uses its true width in all norm and overflow calculations.

The implementation uses a two-pass chunked screen. The first pass finds the lower-bound anchor, and the second emits candidates once its exact score is known. Deterministic per-row intervals and counters make the decision invariant to chunk boundaries. Appendix E gives the complete algorithm and execution rules.

5 Evaluation

5.1 Evaluation design

The evaluation connects exactness to system cost. We measure token identity, row traffic, and group sensitivity, then resident memory, canonical and stock latency, and full-request capacity.

G512 is the primary operating point. W5 is a lower-resident width-128 control that uses 33.59% of a dense head, with a packed 5-bit center and wider candidate sets. Group widths 256 and 1,024, denoted G256 and G1024, test the storage-tightness trade-off. Dense canonical, stock BF16, full-head transfer, static sub-vocabularies, and a centroid-radius screen provide complementary controls. Only methods with certified fallback contribute to the exactness claim.

5.2 Models, states, and decoding modes

The primary matrix covers Qwen3 0.6B and 1.7B (Yang et al., 2025), SmolLM2 1.7B (Allal et al., 2025), and OLMo 2 1B (OLMo Team et al., 2025). Five workload types supply natural text, code, mathematics, multilingual text, and tool-use prompts. Sources include WikiText-103 (Merity et al., 2017), HumanEval (Chen et al., 2021), Grade School Math 8K (GSM8K) (Cobbe et al., 2021), XQuAD (Artetxe et al., 2020), and Hermes Function Calling V1 (Nous Research, 2024). We sample 128 hidden states per model and workload. Greedy decoding and finite-counter Gumbel sampling at temperatures 0.2, 0.7, and 1.0 produce $4 \times 5 \times 128 \times 4 = 10{,}240$ primary state-mode rows.

External validity adds Gemma 2 2B (Gemma Team et al., 2024) and Llama 3.1 8B (Grattafiori et al., 2024). Across all six models, vocabulary sizes span 49,152 to 256,000 and hidden widths span 1,024 to 4,096. The set includes both tied and untied output heads. Each model uses 60 decode prompts. A separate path test pairs Qwen3 1.7B, Gemma 2 2B, and Llama 3.1 8B with contexts 128, 512, and 2,048, then compares prefill and eight generated steps. Group sensitivity uses the same three models, 128 natural-text states, four modes, and three group widths, for 4,608 state-mode rows.

5.3 Metrics and timing protocol

We test correctness at four levels: interval containment, containment of every fetched score, repeated-kernel identity, and final token identity. Candidate fraction is the number of full BF16 output rows evaluated exactly, including the seed, divided by V. For tied models, decode and request timing also include input-row gather and transfer, although candidate fractions count output rows only. Resident fraction follows the packed-byte expression in Equation 9. We report per-state containment slack alongside aggregate failure counts.

Timing separates output-head execution, eight-step decode, and the complete request. Head benchmarks use three warm-ups and eight measured repeats; request benchmarks use eight repeats with bracketed dense references. Appendix F specifies their warm-up schedules. We use 2,000 paired bootstrap resamples to obtain a one-sided 95% upper bound for each primary time ratio. Full-request capacity is the largest batch that completes one warm-up and three measured prefill-plus-eight-step requests at contexts 128, 512, and 2,048. All comparisons use one accelerator. The supplement provides model identities, raw observations, and reconstruction scripts.

6 Results

6.1 Exactness on primary states

G512 preserves every tested primary decision. All 10,240 decisions have finite, nonnegative containment slack and no containment, repeatability, or winner failure. Separate decode checks also return identical tokens. W5 is also exact, whereas center-only and static-vocabulary controls produce mismatches. The contrast shows that the outward intervals and exact fallback are necessary to retain the canonical decision.

6.2 Sparse exact-row retrieval

Row-level residuals make exact retrieval sparse. Table 1 and Figure 2 show that G512 evaluates only 0.005% to 2.190% of vocabulary rows exactly on average. It reduces candidate density relative to W5 by factors of 4.7, 5.1, 5.2, and 29.7, whereas an exact 512-centroid radius control retains 99.77% to 99.89% of rows. The per-row certificate therefore provides useful exclusion that a global cluster radius does not capture in this setting. Tail behavior remains model dependent: the per-model 95th percentile (P95) spans 0.014% to 9.969%. Appendix Tables 9 and 10 report maxima, workload breakdowns, and persistence across modes.

Table 1: G512 preserves every primary decision while reducing exact-row evaluation on all four models. Candidate and resident entries are fractions of vocabulary rows and dense-head bytes, respectively. Each model contributes 2,560 decisions. Request-time ranges cover greedy decoding and sampling at $T = 0.7, 1.0$.

Model	W5 cand. ↓	G512 cand. ↓	G512 resident	Req./canonical ↓	Mismatch
Qwen3 0.6B	0.024%	0.005%	51.17%	[0.230, 0.239]	0
Qwen3 1.7B	0.124%	0.024%	51.17%	[0.222, 0.231]	0
SmolLM2 1.7B	11.493%	2.190%	51.17%	[0.593, 0.788]	0
OLMo 2 1B	35.327%	1.188%	51.17%	[0.263, 0.356]	0

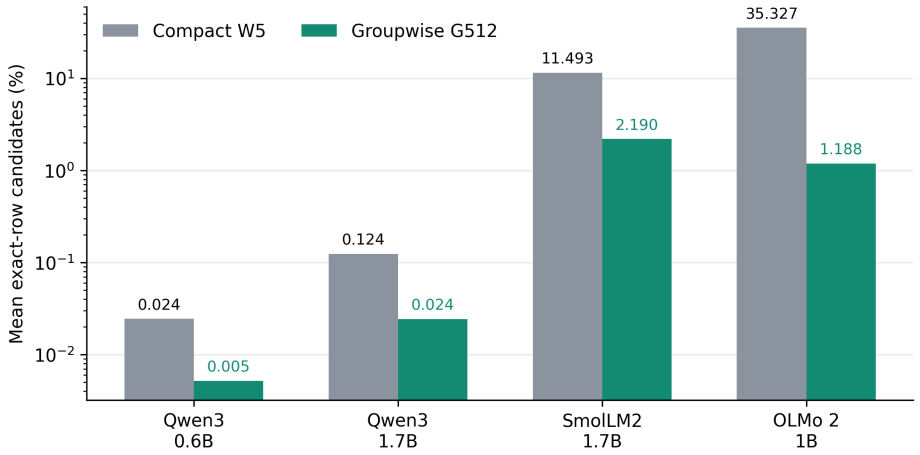


Figure 2: Mean exact-row candidate fractions across 2,560 decisions per model. G512 reduces remote BF16 retrieval by factors of 4.7 to 29.7 relative to W5, while using a larger resident record. The vertical axis is logarithmic.

6.3 Exactness across models and generated paths

G512 preserves the same decision across model families and generated paths. All 15,360 six-model decisions and decoded tokens from 360 prompts match their references. Mean candidate fractions range from 0.0052% on Qwen3 0.6B to 2.8120% on Gemma 2 2B (Appendix Table 13). Nine batched model-context tests also match every prefill decision and 79,432 subsequent token comparisons, with eight steps per sequence. Each selected token feeds the next step.

6.4 Group width and the storage-traffic trade-off

Group width controls a measurable storage-traffic trade-off. Increasing g from 256 to 1,024 reduces resident storage by 1.6 to 1.8 percentage points of dense-head bytes, while candidate sets generally widen. The effect is mild for Llama, visible for Qwen, and large for Gemma, whose mean candidate fraction rises from 0.070% at G256 to 5.248% at G1024. All 4,608 sensitivity decisions remain exact. G512 provides a useful middle operating point across the three models, and Appendix Table 15 reports the complete tails.

6.5 G512 accelerates canonical decoding

Against the same exact numerical contract, G512 lowers request time on all four primary models. The ranges in Table 1 correspond to reductions of 21.2% to 77.8%. Every one-sided 95% upper bound is below one, including $T = 0.2$ (Appendix Table 17). This comparison isolates the benefit of avoiding a correctly rounded full-vocabulary evaluation while preserving the decision on both sides.

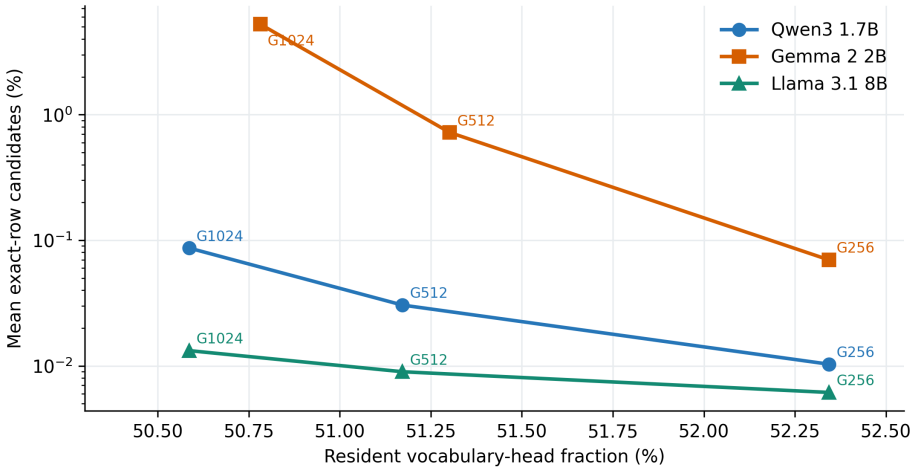


Figure 3: Resident storage and mean candidate fraction at three group widths. Smaller groups usually tighten the certificate at the cost of more metadata; Gemma shows the strongest sensitivity. Equation 9 includes its shorter final group.

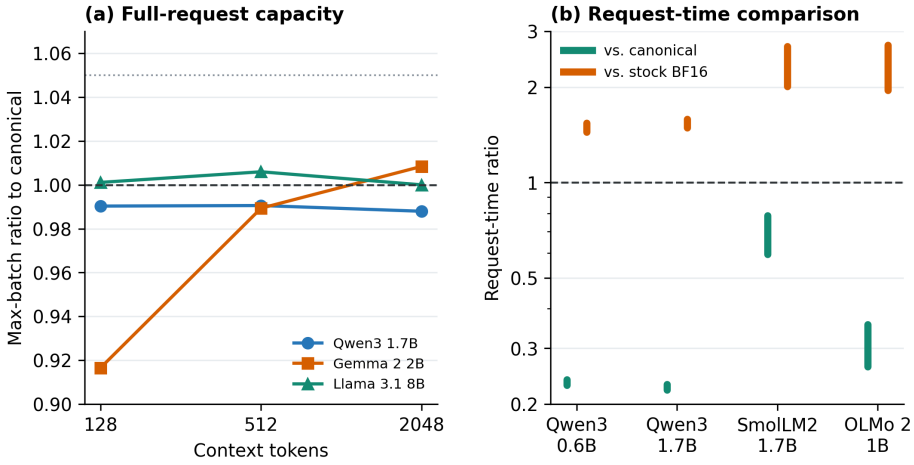


Figure 4: Complete-request capacity and latency. (a) A 49% resident-head reduction yields capacity ratios of 0.917 to 1.008 across all nine settings. The dotted line marks 1.05. (b) Request-time ranges across greedy decoding and sampling at $T = 0.7, 1.0$ show gains against canonical exact decoding and the remaining cost relative to stock BF16.

These gains combine exact decisions with about half of the resident head. Stock BF16 remains faster: the unfused path takes 1.44 to 2.71 times its request time, and its optimized reduction tree defines a different numerical program. The next systems target is the vocabulary-wide screen, where fusion could reduce the remaining overhead.

G512 also removes approximately 49% of resident output-head bytes, saving 290 to 548 MiB on the three capacity models. Full-request capacity does not improve consistently: G512-to-canonical maximum-batch ratios span 0.917 to 1.008 across all nine model-context pairs. Backbone, key-value, and staging terms dominate Equation 1 at the measured frontier. Appendix Table 18 gives every context and batch. The result shows when head compression affects capacity: the head must be material at the complete-request boundary.

7 Discussion and Limitations

7.1 Exactness as a retrieval contract

Exact output-head offload is a selective-retrieval problem. ARCHead packs an approximate head, whereas FlashHead uses clustered retrieval (Kocabay et al., 2026; Tranheden et al., 2026). Certified vocabulary pruning and exact inner-product search preserve a local winner (Liu et al., 2025; Abuzaid et al., 2019). Our authoritative BF16 rows are remote, so resident centers and intervals determine which rows need to be transferred.

Per-row uncertainty supplies the selectivity. A cluster radius covers every member, so one distant row can leave a cluster unresolved. G512 stores a reconstruction and outward error norm per row and group, reducing mean exact retrieval to at most 2.190% across the four primary models while using about half of the head. The tails identify hard states.

The rule also covers fixed-counter sampling. FlashSampling retains exact categorical behavior with a fused local head (Ruiz et al., 2026), while our intervals reduce full-row evaluation when rows are remote. A fused implementation could combine both properties under the same numerical and counter rules.

7.2 From component gains to serving gains

The system measurements show what must change for component gains to become serving gains. FlexGen coordinates memory tiers, vLLM changes a context-scaling cache term, and FlashAttention reduces attention input/output (I/O) (Sheng et al., 2023; Kwon et al., 2023; Dao et al., 2022). G512 reduces H_g and adds staging while leaving backbone and key-value memory intact. Those terms absorb its 290 to 548 MiB head saving at the measured capacity frontier.

Stock latency reveals a second bottleneck. Low-precision dense inference succeeds when cheaper arithmetic offsets conversion and kernel costs (Dettmers et al., 2022). G512 still performs $O(Vd)$ INT8 center work and $O(Vm)$ bound work for $m = \lceil d/g \rceil$. Sparse candidates remove most host-row transfer and exact fallback, but the vocabulary-wide screen remains. Appendix H.1 gives the operation and traffic accounting.

These findings define the operating regime. Exact remote selection is useful when head residency constrains deployment, path identity matters, and candidate tails fit the far-memory budget (Sheng et al., 2023; Abuzaid et al., 2019; Ruiz et al., 2026). G512 preserves path identity on every tested decision and keeps mean candidate traffic sparse, although the reported tails remain model dependent. Stock gains require lower screening overhead, while capacity gains require the head to move the complete-request frontier.

7.3 Scope

The exactness claim covers the correctly rounded BF16 score, greedy decoding, and the specified finite-counter program with row-separable records. Stock reductions and unspecified top-p, top-k, or multinomial implementations define different numerical programs (Yuan et al., 2025). The proof covers any completed deterministic trajectory that satisfies this arithmetic contract. Our experiments span six models, five workloads, three contexts, and eight generated steps on one accelerator.

Deployment may add costs from continuous batching, other interconnects, sharding, or network-attached memory. Candidate tails also vary by model and workload, and coarser groups widen them on Gemma. These factors determine deployment value without changing the exactness result.

8 Conclusion

Exact output-head offload need not transfer a full matrix to reproduce its decision. Across four primary models, G512 uses 51.2% of a dense BF16 head and evaluates only 0.005% to 2.190% of rows exactly on average. It preserves every tested decision across six models and paired generation paths. This selective retrieval lowers canonical request time by 21.2% to 77.8%. Lower screening cost is the next requirement for competing with stock arithmetic.

AI USE STATEMENT

Generative AI tools supported mathematical formalization, experimental code checking, and manuscript polishing. The authors developed the theoretical framework and experimental design, interpreted the results, and verified the citations, equations, code, figures, insights, claims, and prose. The authors take responsibility for all content.

REPRODUCIBILITY STATEMENT

The assumptions and algorithm appear in Sections 3 and 4, with complete proofs in Appendices C and D. Appendix F specifies models, workloads, modes, timing, memory accounting, and failure handling. The Supplementary Material includes the complete experimental code as a self-contained Colab program, raw result records, and scripts that regenerate every empirical table and figure.

REFERENCES

Firas Abuzaid, Geet Sethi, Peter Bailis, and Matei Zaharia. To index or not to index: Optimizing exact maximum inner product search. In *2019 IEEE 35th International Conference on Data Engineering*, 2019.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, et al. SmolLM2: When smol goes big: Data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, 2022.

Gemma Team, Morgane Riviere, Shreya Pathak, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Édouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. Efficient softmax approximation for GPUs. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. In *International Conference on Learning Representations*, 2017.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 2015.

Şuayp Talha Kocabay, Talha Rüzgar Akkuş, and Kamer Ali Yüksel. ARCHead: Activation-metric residual correction for large language model output heads. *arXiv preprint arXiv:2608.02703*, 2026.

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

Dong Liu, Shu Wang, Yanxuan Yu, Haisheng Wang, and Ben Lengerich. CSV-Decode: Certifiable sub-vocabulary decoding for efficient large language model inference. *arXiv preprint arXiv:2511.21702*, 2025.

Chris J. Maddison, Daniel Tarlow, and Tom Minka. A* sampling. In *Advances in Neural Information Processing Systems*, 2014.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Stephen Mussmann, Daniel Levy, and Stefano Ermon. Fast amortized inference and learning in log-linear models with randomly perturbed nearest neighbor search. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*, 2017.

Nous Research. Hermes function calling v1. Hugging Face dataset, 2024. URL `https://huggingface.co/datasets/NousResearch/hermes-function-calling-v1`.

OLMo Team, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, et al. 2 OLMo 2 furious. *arXiv preprint arXiv:2501.00656*, 2025.

Anton Plaksin, Sergei Krutikov, Sergei Skvortsov, and Alexander Samarin. SlimSpec: Low-rank draft LM-head for accelerated speculative decoding. *arXiv preprint arXiv:2605.10453*, 2026.

Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, 2017.

Tomas Ruiz, Zhen Qin, Yifan Zhang, Xuyang Shen, Yiran Zhong, and Mengdi Wang. FlashSampling: Fast and memory-efficient exact sampling. *arXiv preprint arXiv:2603.15854*, 2026.

Jintian Shao, Hongyi Huang, Jiayi Wu, Yiming Cheng, Zhiyu Wu, You Shan, and Mingkai Zheng. VQ-Logits: Compressing the output bottleneck of large language models via vector quantized logits. *arXiv preprint arXiv:2505.10202*, 2025.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. FlexGen: High-throughput generative inference of large language models with a single GPU. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Wilhelm Tranheden, Shahnawaz Ahmed, Devdatt Dubhashi, Jonna Matthiesen, and Hannes von Essen. FlashHead: Efficient drop-in replacement for the classification head in language model inference. *arXiv preprint arXiv:2603.14591*, 2026.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wentian Zhao, Kun Wan, Jing Shi, Xia Hu, and Zirui Liu. Understanding and mitigating numerical sources of nondeterminism in LLM inference. In *Advances in Neural Information Processing Systems*, 2025. doi: 10.52202/085713-5653.

Hanling Zhang, Yayu Zhou, Tongcheng Fang, Zhihang Yuan, Guohao Dai, Wanli Ouyang, and Yu Wang. VocabTailor: Dynamic vocabulary selection for downstream tasks in small language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 28453–28471, 2026. doi: 10.18653/v1/2026.findings-acl.1418.

Weilin Zhao, Tengyu Pan, Xu Han, Yudi Zhang, Ao Sun, Yuxiang Huang, Kaihuo Zhang, Weilun Zhao, Yuxuan Li, Jie Zhou, Hao Zhou, Jianyong Wang, Zhiyuan Liu, and Maosong Sun. FR-Spec: Accelerating large-vocabulary language models via frequency-ranked speculative sampling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3909–3921, 2025. doi: 10.18653/v1/2025.acl-long.198.

A Appendix Contents

The appendix follows the claim order of the main paper. Appendix B fixes notation. Appendix C gives the finite numerical contract, including the deterministic quantizers and bias scope. Appendix D proves interval containment, decision identity, path identity, and the layout law. Appendix E states the screening algorithm and gives a tie-aware example. Appendix F records models, workloads, the W5 control, metrics, and execution controls. Appendices G and H provide expanded correctness, variability, sensitivity, cost, timing, capacity, and lifecycle results. Appendix I compares the contract with nearby output-head systems. Appendix J describes the executable reproduction program and result records.

Table 2: Map from main-paper claims to supporting appendix material

Main claim	Supporting material
Canonical numerical identity	Appendix C; Propositions 3 and 4
Deterministic G512 construction	Appendix C.2; Equation 11
Exact interval screening	Appendix D; Algorithm 4
G512 resident fraction	Proposition 5; Tables 15 and 6
Cross-model token identity	Tables 13 and 14
Candidate-rate variability	Tables 9, 10, and 15
Full-request systems boundary	Appendix H; Tables 18 and 19

B Notation and Scope

Table 3 collects the main symbols used in the method. A *state* is one decoder hidden vector, and a *state-mode row* pairs that vector with one decision mode. A *candidate* is a vocabulary row that is read in full bfloat16 (BF16) form and evaluated by the canonical score. A *certificate record* is the accelerator-resident collection of 8-bit integer (INT8) codes and outward metadata for one row. The record is used only for screening. We use 32-bit floating-point (FP32), 64-bit floating-point (FP64), and 32-bit integer (INT32) arithmetic. G256, G512, and G1024 denote group widths 256, 512, and 1,024, while W5 denotes the 5-bit lower-resident control.

The theorem preserves the decision of one executable reference, including its arithmetic, counter, and tie rule. The empirical evaluation compares screened and dense implementations of that program on fixed states and trajectories. This pathwise statement differs from equality in distribution under an ideal continuous random variable.

Table 3: Notation

Symbol	Meaning
V, d	Vocabulary size and hidden dimension
h, w_i	BF16 hidden state and BF16 output row for token i
$\mathcal{C}(h, w_i)$	Correctly rounded canonical BF16 dot product
z_i, x_i	Canonical score and finite processed score for row i
$T, g_i, P_{T,i}$	Temperature, fixed Gumbel value, and score processor
G_r, g, m	Coordinate group, maximum group width, and number of groups
a_r, s_{ir}	Hidden and weight scales for group r
p_r, q_{ir}	Signed INT8 hidden and weight codes
$c_i, \widetilde{c}_i$	Exact real center and its executable FP64 evaluation
$e_{h,r}, e_{ir}, r_{ir}$	Hidden residual norm, weight residual norm, and reconstructed-row norm
$\Delta_i, \eta_i^{\mathrm{fin}}$	Outward score radius and its finite-arithmetic term
$[L_i, U_i]$	Outward interval for a canonical or processed score
b, s_b	Lower-bound anchor and its exact canonical or processed score
$\mathcal{S}$	Set of rows evaluated exactly
H_g, S_g, B	Resident certificate, method staging, and backbone plus key-value memory
$\rho_g(d)$	Resident vocabulary-head fraction at width g

C Complete Numerical Contract

C.1 Finite BF16 operands

Every hidden and weight operand is a finite BF16 bit pattern. A nonzero finite BF16 value can be written as $\sigma m 2^q$, where $\sigma \in \{-1, 1\}$, m is an integer significand, and q is an integer exponent. The product of two operands is therefore an integer times a power of two. The canonical dot product groups products by exponent, accumulates each bin in a signed integer, propagates carries, and rounds the resulting exact magnitude once to BF16 with ties to even. Zero sign, subnormal output, and overflow are handled explicitly.

For every evaluated hidden width, the implementation checks an integer upper bound before launching the kernel. The exponent-bin accumulator and the groupwise INT32 center both stop if their proved ranges are insufficient. The canonical dense reference and every exact candidate use the same kernel and output-bit decoder.

Proposition 3 (Finite canonical score)**.** *For finite BF16 vectors h, w_i whose checked integer bounds hold, the exponent-binned kernel returns Equation 2 independently of thread scheduling and candidate batch shape.*

Proof. Each product has an exact integer-power-of-two representation. Integer addition within an exponent bin is associative because overflow is excluded by the checked bound. Carry propagation yields the unique signed binary expansion of the exact real sum. The final guard, round, and sticky bits implement round-to-nearest-even, including subnormal and signed-zero cases. Candidate batching only partitions independent row sums and therefore cannot change any bin or final rounding decision. □

C.2 Deterministic group quantizers and bias scope

The G512 record is constructed without calibration data, clipping search, or an outlier heuristic. For every row i and real coordinate group G_r, let ν_{32} be the smallest positive normal FP32 value. The stored scale and signed code are

$$s_{ir} = \max\left\{ \mathrm{RN}_{32}\left(\frac{\max_{j \in G_r} |w_{ij}|}{127} \right), \nu_{32} \right\}, \tag{10}$$

$$q_{ij} = \mathrm{clip}_{[-127,127]}(\mathrm{RN}_{\mathbb{Z}}(w_{ij}/s_{ir})), \qquad j \in G_r, \tag{11}$$

where RN_{32} is nearest-even FP32 rounding and $\mathrm{RN}_{\mathbb{Z}}$ is nearest-integer rounding with even ties. The reconstruction is the exact real value of the stored FP32 scale times the integer code. A shorter

final group uses only its real coordinates when selecting the scale and computing norms. The online hidden quantizer uses the same rule with a_r and p_r in place of s_{ir} and q_{ir}.

The three stored FP32 values are fully determined by the row. They are the scale s_{ir}, the upward residual norm e_{ir}, and the upward reconstruction norm r_{ir}. Both norms are first guarded in FP64 and then moved to the adjacent FP32 value toward $+\infty$. This construction explains the 12 metadata bytes per group in Equation 9. It also makes the offline record reproducible from the BF16 head alone.

All six evaluated output heads are bias-free, and Equation 2 therefore contains no bias term. A biased contract can define

$$z_i^{\beta} = \mathrm{RN}_{\mathrm{BF16}}\left(\sum_j h_j w_{ij} + \beta_i\right), \tag{12}$$

where the finite bias β_i has a fixed scalar format and is interpreted exactly before the final rounding. Storing that scalar in the resident row record permits directed addition of β_i to both real endpoints before the existing BF16 map. The containment proof then applies unchanged. If the bias uses B_β bytes, the resident fraction relative to a dense biased row becomes $(d + 12m + B_\beta)/(2d + B_\beta)$, and the complete head adds VB_β resident bytes. A reference that rounds the dot product before adding the bias requires the endpoint program to follow that order instead. This extension specifies the arithmetic and storage effects. Its candidate and runtime effects are not measured by the six bias-free heads.

C.3 Outward metadata and interval construction

The finite interval follows a fixed binary64 construction. Let $u = 2^{-53}$ and $\gamma_k = ku/(1 - ku)$. The implementation computes γ_k in 80-digit decimal arithmetic and rounds it upward to FP64. Original coordinates are BF16, and reconstructed coordinates are integer multiples of FP32 scales. Any nonzero coordinate or reconstruction residual therefore has magnitude at least 2^{-149}, so a nonzero product entering these reductions is at least 2^{-298}. Even its product with the smallest relevant γ_k remains above the smallest normal FP64 value. Thus, the standard relative-error model applies without an underflow exception. For a nonnegative reduction whose evaluation uses at most k floating-point operations, it maps the computed value $\widetilde{s}$ to

$$\mathrm{Up}\left(\frac{\widetilde{s}}{1 - \gamma_k}\right), \tag{13}$$

where Up returns the next FP64 value toward $+\infty$. The corresponding Down operation moves toward $-\infty$. The executed contract uses round-to-nearest-even binary64 arithmetic with gradual underflow. Non-finite results stop execution.

Stored scales are interpreted as exact finite FP32 values and therefore define the reconstructions in Equation 4. A group norm is computed from FP64 squares and a sum guarded by Equation 13 with $k = 2|G_r|$, followed by an upward square root. Casting this norm to FP32 and then taking the adjacent FP32 value toward $+\infty$ gives the stored residual or reconstruction norm. Hidden group norms and residual norms use the same procedure at inference time.

The executable center has a separate roundoff bound. Let $d_{ir} = p_r^{\top} q_{ir}$, which is exact in INT32, and let m be the number of groups. The FP64 program uses two scale multiplications and one accumulation per group. It forms an outward upper bound A_i on $\sum_r |d_{ir} a_r s_{ir}|$ and sets

$$\eta_i^{\mathrm{fin}} = \mathrm{Up}(\gamma_{3m} A_i)\,. \tag{14}$$

The geometric terms in Equation 7 are nonnegative. Their two products per group and subsequent reductions use Equation 13 with the conservative count $k = 4m$. A final upward addition combines this geometric radius with Equation 14.

Endpoint construction completes the outward map. The implementation applies Down to $\widetilde{c}_i - \Delta_i$ and Up to $\widetilde{c}_i + \Delta_i$. Each endpoint is cast to nearest BF16 and moved by one adjacent BF16 value if that cast points inward. The evaluation checks real-score containment, fetched-score containment, center repeatability, and final winner dominance. A negative or non-finite slack value stops execution. These checks test the executed assumptions on the evaluated states; the guarantee itself follows from the finite construction above.

C.4 FINITE COUNTER-BASED SAMPLING

For each state, decoding step, mode, and global token index, a logical counter produces the same finite random word in dense and screened execution. The word maps deterministically to a finite Gumbel value g_i. Temperature division and Gumbel addition follow Equation 3 in 64-bit floating-point (FP64) arithmetic. The endpoint implementation applies downward rounding to the lower path and upward rounding to the upper path after both operations.

Proposition 4 (Processed-score containment)**.** *If $z_i \in [L_i, U_i]$ and $T > 0$, the directed endpoint program contains $P_{T,i}(z_i)$ for the same finite g_i.*

Proof. Division by positive T and addition of fixed g_i are monotone. Correctly directed rounding after each operation preserves the lower and upper inequalities. The composed finite program therefore contains the processed point. □

The result is pathwise for the specified finite counter. It does not equate the discrete counter generator with an ideal continuous Gumbel distribution and does not cover rejection samplers, top-k, or nucleus sampling.

D PROOFS OF SCREENING AND LAYOUT

D.1 GROUPWISE REAL-SCORE RADIUS

Within group r, add and subtract $h_r^\top \widehat{w}_{ir}$:

$$h_r^\top w_{ir} - \widehat{h}_r^\top \widehat{w}_{ir} = h_r^\top (w_{ir} - \widehat{w}_{ir}) \tag{15}$$

$$+ (h_r - \widehat{h}_r)^\top \widehat{w}_{ir}. \tag{16}$$

Cauchy-Schwarz bounds the absolute value by $\|h_r\|_2 e_{ir} + e_{h,r} r_{ir}$. Summing over disjoint groups and adding the finite outward term proves Equation 7. The result also covers a final group shorter than g because all vectors and norms use its true coordinate set.

For comparison with a global radius, apply Cauchy-Schwarz once more over the group index:

$$\sum_r \|h_r\|_2 e_{ir} \le \left(\sum_r \|h_r\|_2^2 \right)^{1/2} \left(\sum_r e_{ir}^2 \right)^{1/2}, \tag{17}$$

$$\sum_r e_{h,r} r_{ir} \le \left(\sum_r e_{h,r}^2 \right)^{1/2} \left(\sum_r r_{ir}^2 \right)^{1/2}. \tag{18}$$

Thus grouping cannot loosen the real-valued bound when the reconstruction is fixed. Quantizer scales and finite arithmetic change with g, so empirical candidate ordering still requires measurement.

D.2 TIE-AWARE CANDIDATE CORRECTNESS

Let x_i be the canonical greedy score or fixed processed sampling score, and assume $x_i \in [L_i, U_i]$. Choose $b = \min \arg\max_i L_i$ and evaluate it exactly to obtain $s_b = x_b$. Define

$$\mathcal{S} = \{b\} \cup \{i < b : U_i \ge s_b\} \cup \{i > b : U_i > s_b\}. \tag{19}$$

For an excluded $i < b$, $U_i < s_b$, hence $x_i < s_b$. For an excluded $i > b$, $U_i \le s_b$, hence $x_i \le s_b$ and token b wins any equality. No excluded row can beat b. The global smallest-index maximizer therefore lies in $\mathcal{S}$. Exact evaluation and the same tie rule over $\mathcal{S}$ return the global canonical decision.

The proof permits a conservative implementation that includes extra equality cases. Such inclusion increases traffic but cannot change the exact result. It also permits chunked interval recomputation because Equation 19 depends only on deterministic per-row endpoints, the global index, and s_b.

The same proof permits multiple exact anchors. Let A be a nonempty anchor set, let a be the smallest index attaining the largest exact score in A, and let $s_A = x_a$. Then

$$\mathcal{S}_A = A \cup \{i < a : U_i \geq s_A\} \cup \{i > a : U_i > s_A\} \tag{20}$$

contains the global canonical winner. If A includes the lower-bound anchor b, its non-anchor survivors cannot exceed those from Equation 19, because $s_A \geq s_b$. Total traffic also includes every row in A, so a stronger threshold does not guarantee fewer reads. Multi-anchor selection is therefore an exact traffic trade-off, not an additional correctness assumption.

D.3 Autoregressive path identity

At step zero, screened and dense execution receive the same prompt and therefore the same hidden state and logical counter. Candidate correctness returns the same token. Assume both paths have returned the same prefix through step t. Their deterministic model inputs, hidden state, and counter key at step $t + 1$ are equal. Candidate correctness again returns the same token. Induction proves equality for every completed step unless either implementation raises an explicit failure.

D.4 Resident layout

Proposition 5 (Resident fraction)**.** *A certificate with d INT8 codes and three FP32 metadata values for each group of maximum width g occupies fraction $\rho_g(d)$ in Equation 9 relative to a dense BF16 row.*

Proof. The codes use d bytes. There are $\lceil d/g \rceil$ real groups and each stores 12 metadata bytes. A dense BF16 row uses $2d$ bytes. Division gives the expression. A short final group still requires one scale and two norms, so the ceiling is necessary. □

E Screening Algorithm

Table 4: Tie-aware groupwise screening and exact fallback

1	Quantize the hidden state within the fixed coordinate groups; compute outward hidden norms and residual norms.
2	Stream vocabulary chunks through the INT8 center kernel; construct outward canonical or processed intervals for every row.
3	Set $b = \min \arg\max_i L_i$. Fetch full BF16 row b from host memory and compute its exact score s_b.
4	Recompute or read intervals. Emit b, each $i < b$ with $U_i \geq s_b$, and each $i > b$ with $U_i > s_b$.
5	Fetch emitted rows in capacity-bounded atomic batches. Evaluate Equation 2 and, for sampling, Equation 3.
6	Return the smallest-index exact maximizer. Stop on non-finite values, invalid intervals, overflow, missing rows, or a containment witness below zero.

The algorithm can cache first-pass endpoints when memory permits or recompute them in the second pass. The scientific output is invariant because chunk size never enters the quantizer, counter key, endpoint function, or candidate predicate. Capacity recovery commits a batch only after all its rows have completed. An interrupted batch is rerun at a measured smaller size.

E.1 A tie-aware screening example

The asymmetric candidate rule is easiest to inspect in a three-row example with one lower-index tie and one dominated higher-index row. Suppose the lower endpoints select row $b = 1$, and its exact score is $s_b = 1.9$. Table 5 shows intervals consistent with that seed. Row 0 must remain because it has a smaller index and could tie the seed. Row 2 can be excluded because it has a larger index and cannot exceed the seed. Exact evaluation of rows 0 and 1 then applies the dense reference's smallest-index rule, which selects row 0 if its score ties the seed.

Table 5: Example of the strict and non-strict candidate predicates

Row i	Interval $[L_i, U_i]$	Predicate at $s_b = 1.9$	Outcome
0	$[1.2, 1.9]$	$i < b$ and $U_i \geq s_b$	Fetch
1	$[1.6, 2.1]$	Seed row	Fetch
2	$[1.0, 1.9]$	$i > b$ and $U_i \not> s_b$	Exclude

If row 0 instead had $U_0 = 1.8$, the certificate would exclude both non-seed rows and fetch only row 1. The numbers are illustrative intervals. The executable method obtains them from the grouped construction in Appendix C.

F Experimental Protocol

F.1 Model and head properties

Table 6: Evaluated model and output-head properties

Model	Tied	Parameters	V	d	Dense head	Head/parameters
Qwen3 0.6B	Yes	0.60B	151,936	1,024	296.8 MiB	26.10%
Qwen3 1.7B	Yes	1.72B	151,936	2,048	593.5 MiB	18.08%
SmolLM2 1.7B	Yes	1.71B	49,152	2,048	192.0 MiB	5.88%
OLMo 2 1B	No	1.48B	100,352	2,048	392.0 MiB	13.84%
Gemma 2 2B	Yes	2.61B	256,000	2,304	1,125.0 MiB	22.56%
Llama 3.1 8B	No	8.03B	128,256	4,096	1,002.0 MiB	6.54%

The supplement identifies each model by its repository, resolved model revision, vocabulary shape, hidden width, data type, tied status, and output-head values. These fields are checked against the loaded model before evaluation. For a tied model, removing the output matrix also removes the shared input embedding. Exact input-token rows are then gathered from the pinned host matrix and transferred before backbone execution. For an untied model, the input embedding remains accelerator-resident. Candidate fractions count only output-decision rows, while request timings include the tied input-row gather and transfer.

F.2 Workload construction

The five workload families are natural text, code, mathematics, multilingual text, and tool use. They use the five sources named in Section 5.2, including the single-turn file from Hermes Function Calling V1 for tool-use prompts. Source documents are selected deterministically before hidden-state extraction. Primary results use 128 states from each model-workload pair. Each state is evaluated under greedy decoding and three finite-counter temperatures. Workload and state identifiers remain attached to raw rows, which allows aggregation by document cluster rather than by shuffled tensor position. Candidate percentiles use linear interpolation at position $(n-1)p$ in the sorted evaluation population.

External decode prompts use the same five content families and 60 prompts per model. The paired trajectory test uses three models and contexts 128, 512, and 2,048. Every path begins from the same tokenized prompt, compares the prefill decision, then feeds each returned token to both methods for eight steps.

F.3 Controls

The dense canonical reference evaluates every vocabulary row with Equation 2. Stock BF16 uses the framework's optimized local output projection and defines a different numerical path. Full-head offload transfers the complete BF16 head before local evaluation. The centroid-radius control uses 512 clusters, eight fixed k-means iterations, exact geometric bounds, and full-row fallback under the

same canonical greedy score. A separate diagnostic uses 2,000 clusters on 128 fixed Qwen3 0.6B natural-text states, and a length-bound diagnostic orders 4,096-row blocks by outward row norm. Both diagnostics use the canonical score and tie rule, and neither contributes a timing result. Static controls retain fixed vocabulary fractions of 5%, 15%, and 35%, may change the answer, and are reported only as approximate controls.

The primary G512 result was not selected against these controls after measurement. Group-width sensitivity changes only g among 256, 512, and 1,024, while all other numerical operations, states, modes, and tie rules stay fixed.

F.4 W5 LOWER-RESIDENT CONTROL

W5 is an exact lower-resident control, not an approximate decoder. It partitions each weight row into groups of width 128 and applies symmetric signed 5-bit quantization with $q_{\max} = 15$. Each group stores packed codes, one 16-bit floating-point (FP16) max-absolute scale, and one upward FP32 residual norm. The hidden state remains BF16. Its group center is $h_r^\top (s_{ir} q_{ir})$, and its geometric radius is $\|h_r\|_2 e_{ir}$ plus the outward finite-arithmetic term. The same canonical fallback and tie rule then evaluate every unresolved row.

For hidden width d, the W5 record uses

$$\rho_{\mathrm{W5}}(d) = \frac{\lceil 5d/8 \rceil + 6\lceil d/128 \rceil}{2d}. \tag{21}$$

When 128 divides d, this fraction is 0.3359375. W5 removes the hidden-reconstruction term and two FP32 fields used by G512, but its packed center is less hardware-aligned and its larger weight error often widens the exact set. It serves as a concrete storage-versus-traffic control rather than a candidate operating point chosen after seeing the primary results.

F.5 TIMING AND MEMORY

Head timings use batches 1, 8, 32, 64, and 128, with three warm-ups and eight measured repeats for the primary methods. Decode contains eight generated steps. Request time includes prefill, decode, and input-row host gather and transfer for tied models. G512 and W5 each receive one complete-request warm-up before eight measured repeats. Dense canonical and stock references each have eight measured repeats before and after the method runs, without a separate request warm-up. The baseline for repeat r is the mean of its before and after measurements. Paired bootstrap samples resample these matched repetition indices, with the prompt set fixed. Table 1 summarizes greedy decoding and temperatures 0.7 and 1.0; Table 17 also includes temperature 0.2.

Capacity search keeps the model and method state resident, then measures the largest batch that completes full prefill and eight decode steps. Contexts are 128, 512, and 2,048. A frontier point requires one warm-up and three measured successes. A failed batch is recorded and the unchanged request is retried below it. Device memory is probed at runtime. Model weights, temporary key-value tensors, and scratch stay local. Only compact progress and reusable scientific records are persisted.

G EXPANDED RESULTS

G.1 PRIMARY CORRECTNESS AND CANDIDATE RESULTS

Table 7: Primary G512 failure counts. Head checks cover 10,240 state-mode rows; decode checks use 60 prompts per model.

Model	Containment	Fetched interval	Repeatability	Winner/decode
Qwen3 0.6B	0	0	0	0 / 0
Qwen3 1.7B	0	0	0	0 / 0
SmolLM2 1.7B	0	0	0	0 / 0
OLMo 2 1B	0	0	0	0 / 0

The zero counts are paired with finite nonnegative slack values for every row. The smallest external-validity containment slack is 0.546875 on Qwen3 0.6B. The other model minima are 0.761719, 3.394531, 1.937500, 2.687500, and 1.021484. These values are implementation witnesses in canonical-score units, not calibrated probabilistic margins.

Table 8: Mean G512 exact-row fraction by workload, averaged over four decision modes and 128 states per cell

Model	Code	Math	Multilingual	Natural	Tool use
Qwen3 0.6B	0.0011%	0.0065%	0.0073%	0.0065%	0.0046%
Qwen3 1.7B	0.0102%	0.0027%	0.0213%	0.0305%	0.0571%
SmolLM2 1.7B	0.0167%	0.9542%	2.0153%	3.1933%	4.7690%
OLMo 2 1B	0.4960%	0.1490%	3.1231%	0.1739%	1.9992%

The workload split localizes the broad model averages. SmolLM2 rises from 0.0167% on code to 4.7690% on tool-use states, while OLMo 2 peaks on multilingual states. The two Qwen models remain below 0.058% in every workload cell. By comparison, model-wide means across the four decision modes span 0.0047%–0.0054%, 0.0225%–0.0261%, 1.8489%–2.4240%, and 0.8803%–1.3523%, respectively. In this population, workload and architecture are associated with larger candidate variation than the tested temperature.

Table 9: Primary G512 candidate tails over 2,560 state-mode rows per model. P95 and P99 denote the 95th and 99th percentiles. ρ_{mode} is the median of six pairwise Spearman correlations between candidate-rate ranks for the same 640 states under four modes.

Model	Median	Mean	P95	P99	Max	ρ_{mode}
Qwen3 0.6B	0.0033%	0.0052%	0.0145%	0.0273%	0.2389%	0.79
Qwen3 1.7B	0.0072%	0.0244%	0.0942%	0.2013%	2.0818%	0.95
SmolLM2 1.7B	0.5147%	2.1897%	9.9694%	21.7441%	47.3755%	0.91
OLMo 2 1B	0.1799%	1.1882%	5.6154%	9.8602%	46.7873%	0.93

The mean hides substantial transfer bursts on SmolLM2 and OLMo 2. The largest workload tails occur on SmolLM2 natural states, with P99 43.9715%, and OLMo 2 multilingual states, with P99 15.4967%. Candidate rankings nevertheless persist across decision modes: the median same-state rank correlation is 0.79–0.95, and the top 1% of states account for 9.9%–16.8% of candidate mass. Thus, state identity organizes the observed tail more strongly than the tested temperature alone. The recorded rows contain workload, mode, token count, and candidate count, but not hidden norms or score margins. They support this recurrence analysis without identifying a norm-based predictor.

Table 10: G512 candidate tails by workload. Each row pools 128 states under four decision modes ($n = 512$).

Model	Workload	Mean	P95	P99	Max
Qwen3 0.6B	Code	0.0011%	0.0033%	0.0039%	0.0066%
	Math	0.0065%	0.0112%	0.0132%	0.0158%
	Multilingual	0.0073%	0.0178%	0.0267%	0.0559%
	Natural	0.0065%	0.0244%	0.0564%	0.2389%
	Tool use	0.0046%	0.0118%	0.0165%	0.0217%
Qwen3 1.7B	Code	0.0102%	0.0342%	0.0677%	0.0981%
	Math	0.0027%	0.0053%	0.0072%	0.0092%
	Multilingual	0.0213%	0.1019%	0.1904%	0.3515%
	Natural	0.0305%	0.0919%	0.2206%	2.0818%
	Tool use	0.0571%	0.1574%	0.3101%	0.6608%
SmolLM2 1.7B	Code	0.0167%	0.0285%	0.0364%	0.0448%
	Math	0.9542%	2.3683%	4.2096%	6.9356%
	Multilingual	2.0153%	11.2995%	24.7677%	30.6396%
	Natural	3.1933%	17.4296%	43.9715%	47.3755%
	Tool use	4.7690%	11.8697%	17.4574%	41.0665%
OLMo 2 1B	Code	0.4960%	2.2968%	6.7109%	13.8313%
	Math	0.1490%	0.4120%	0.6428%	1.1151%
	Multilingual	3.1231%	8.8727%	15.4967%	46.7873%
	Natural	0.1739%	0.4456%	2.4650%	11.4108%
	Tool use	1.9992%	5.5279%	9.2256%	9.8075%

Table 11: Exact 512-centroid radius control on the primary greedy states

Model	Greedy states	Mean candidates	Winner mismatch
Qwen3 0.6B	640	99.890%	0
Qwen3 1.7B	640	99.873%	0
SmolLM2 1.7B	640	99.768%	0
OLMo 2 1B	640	99.818%	0

The centroid control provides almost no traffic reduction under the far-memory accounting. It isolates centroid-radius pruning under our fixed score, states, and fallback rule, rather than executing the complete CSV-Decode system. The comparison attributes the observed selectivity to row-level residual information.

We also tested whether a finer global partition closes this geometric gap. The 2,000-centroid diagnostic opens clusters in descending corrected upper-bound order, updates the exact incumbent after each cluster, and stops with the same tie rule as the canonical reference. Table 12 shows that it still reads 99.3588% of rows on average. The length-bound block diagnostic reads every row. All 128 centroid queries exceed an 80% row fraction, with a sample standard deviation of 3.00×10^{-5} in fractional units. These same-reference geometric diagnostics measure pruning selectivity without sparse-kernel timing.

Table 12: Same-reference exact-index diagnostics on 128 fixed Qwen3 0.6B natural-text states

Diagnostic	Mean candidates	P95 candidates	Winner mismatch
Corrected 2,000-centroid radius	99.3588%	99.3596%	0
Length-bound 4,096-row blocks	100.0000%	100.0000%	0

Table 13: Six-model G512 external validity

Model	Candidate	Minimum slack	Decode prompts	Token mismatch
Qwen3 0.6B	0.0052%	0.546875	60	0
Qwen3 1.7B	0.0244%	0.761719	60	0
SmolLM2 1.7B	2.1897%	3.394531	60	0
OLMo 2 1B	1.1882%	1.937500	60	0
Gemma 2 2B	2.8120%	2.687500	60	0
Llama 3.1 8B	0.0081%	1.021484	60	0

Table 14: Paired trajectory tests at three context lengths. Each batched test compares the prefill decision and eight subsequent generated tokens per sequence. Generated-token counts sum over all sequences, not the length of a single trajectory.

Model	Contexts	Batched tests	Generated tokens compared	Mismatch
Qwen3 1.7B	128, 512, 2,048	3	31,048	0
Gemma 2 2B	128, 512, 2,048	3	30,984	0
Llama 3.1 8B	128, 512, 2,048	3	17,400	0

The external and trajectory tests preserve the same exact decision contract across architectures and autoregressive feedback. The external set adds two architectures and preserves every compared token. The nine batched trajectory tests match all 9,929 prefill decisions and 79,432 subsequent token comparisons, with eight generated steps per sequence. Each selected token is fed back into its own execution. These results test implementation identity beyond the primary state population, while the proof establishes identity for every completed path that satisfies the numerical contract.

G.2 Complete group sensitivity

Table 15: Group-width sensitivity over 4,608 state-mode rows. Candidate columns report mean, 95th percentile, 99th percentile, and maximum fractions. All failure counts are zero.

Model	Group	Resident	Mean cand.	P95 cand.	P99 cand.	Max cand.
Qwen3 1.7B	256	52.344%	0.0103%	0.0316%	0.0731%	0.5733%
	512	51.172%	0.0305%	0.0919%	0.2206%	2.0818%
	1,024	50.586%	0.0867%	0.2733%	0.9249%	4.5565%
Gemma 2 2B	256	52.344%	0.0697%	0.1751%	1.4591%	5.9137%
	512	51.302%	0.7206%	1.8164%	18.0894%	39.9086%
	1,024	50.781%	5.2482%	33.7721%	56.4491%	94.0250%
Llama 3.1 8B	256	52.344%	0.0061%	0.0163%	0.0371%	0.3103%
	512	51.172%	0.0090%	0.0230%	0.0708%	0.4569%
	1,024	50.586%	0.0132%	0.0351%	0.1386%	0.5746%

The sensitivity experiment uses 128 natural-text states per model, whereas the primary means pool 640 states across five workloads. Each group-width comparison uses the same states and modes within its model. This paired design isolates the storage-tightness trade-off.

The controls expose two first-order representation choices. G256, G512, and G1024 hold INT8 precision and the deterministic max-absolute quantizer fixed while changing error localization. W5 changes code width, group width, scales, and packed center jointly. It uses fewer resident bytes but raises mean candidate density by 4.7 to 29.7 times relative to G512. Candidate tightness is therefore not determined by resident bytes alone. Learned scales, vector codebooks, and 4-bit layouts would require new certified records and paired measurements, and the present controls do not rank those designs.

The containment proof does not require equal group widths. For a fixed partition shared by all rows, Equation 7 holds and each row occupies $d+12m$ bytes for m groups. A mixed-width partition could therefore allocate more metadata to sensitive coordinate ranges. Selecting such a partition from candidate rates would add a calibration problem and could overfit workloads, while row-specific partitions would also lose the shared hidden quantizer and regular packed layout. We leave adaptive grouping unevaluated rather than selecting it from the sensitivity population.

H Resource and Failure Analysis

H.1 Operation and traffic accounting

The certificate reduces full-row traffic but retains a vocabulary-wide screen. Let b be the state batch, $m = \lceil d/g \rceil$, and $\mathcal{S}_t$ be the exact set for state t. Table 16 separates the logical work and storage terms. These expressions do not assume a cache hit rate or an interconnect bandwidth.

Table 16: Logical work and storage for one G512 output-head invocation

Stage	Arithmetic	Logical storage or transfer
Hidden preparation	$O(bd)$ quantization and grouped norms	bd INT8 bytes and $12bm$ FP32 metadata bytes, transient
Vocabulary screen	$O(bVd)$ INT8 multiply-accumulate work and $O(bVm)$ FP64 scale, radius, and endpoint work	Reads the $V(d+12m)$-byte resident record per logical pass
Endpoint staging	No additional score arithmetic when cached	Optional two BF16 endpoint tensors, $4bV$ bytes; otherwise the screen is recomputed
Exact seed	b canonical dot products	$2bd$ BF16 row bytes cross from host memory
Remaining candidates	$O(d\sum_t(\lvert\mathcal{S}_t\rvert-1))$ canonical work	$2d\sum_t(\lvert\mathcal{S}_t\rvert-1)$ BF16 row bytes cross from host memory

Candidate fractions count the seed once. Since $\mathcal{S}_t$ includes that seed, the last two rows of Table 16 sum to $2d\sum_t|\mathcal{S}_t|$ transferred bytes. Sparse candidates reduce this traffic, while the vocabulary-wide screen remains. Endpoint caching exchanges memory for a second pass. Timing records cover complete head, decode, and request executions and retain transfer bytes, host gather time, and endpoint-cache choices. They do not separate non-overlapping times for quantization, integer centers, FP64 endpoints, and exact fallback, so the table reports logical work alongside measured total latency.

FP64 arithmetic is part of the present executable proof. Scale products, center-error bounds, non-negative radius reductions, and endpoint construction use FP64 before directed conversion to BF16. This choice makes the numerical contract explicit, but accelerators differ substantially in FP64 throughput. A 32-bit floating-point (FP32) endpoint path would require a proved error budget and a new containment test. The measured gap to stock BF16 identifies this arithmetic and the two-pass screen as optimization targets, although the current records do not separate their individual runtime shares.

H.2 Canonical and stock timing boundary

G512 reduces complete-request latency against the canonical dense reference in every primary model-mode cell. The three-mode ranges are [0.230, 0.239] for Qwen3 0.6B, [0.222, 0.231] for Qwen3 1.7B, [0.593, 0.788] for SmolLM2 1.7B, and [0.263, 0.356] for OLMo 2 1B. Relative to stock BF16, the corresponding ranges are [1.442, 1.542], [1.487, 1.584], [2.011, 2.691], and [1.951, 2.714]. These comparisons distinguish the benefit under the exact numerical contract from the remaining gap to stock execution.

The canonical improvement also holds when temperature 0.2 is included. Table 17 gives every mode separately, and all pointwise one-sided 95% upper confidence bounds remain below one. The upper bounds describe each matched comparison; the ranges above summarize variation across modes.

Table 17: Complete-request timing by decision mode. Ratios below one favor G512. The 95% upper column is the pointwise one-sided upper confidence bound for G512/canonical.

Model	Mode	G512/canonical	95% upper	G512/stock
Qwen3 0.6B	Greedy	0.239	0.301	1.542
	$T = 0.2$	0.231	0.238	1.447
	$T = 0.7$	0.230	0.237	1.442
	$T = 1$	0.231	0.239	1.455
Qwen3 1.7B	Greedy	0.231	0.286	1.584
	$T = 0.2$	0.224	0.230	1.505
	$T = 0.7$	0.222	0.229	1.495
	$T = 1$	0.222	0.229	1.487
SmolLM2 1.7B	Greedy	0.788	0.919	2.691
	$T = 0.2$	0.764	0.776	2.592
	$T = 0.7$	0.706	0.716	2.403
	$T = 1$	0.593	0.606	2.011
OLMo 2 1B	Greedy	0.356	0.364	2.714
	$T = 0.2$	0.358	0.365	2.651
	$T = 0.7$	0.306	0.313	2.274
	$T = 1$	0.263	0.271	1.951

The largest observed W5 batch moves 18.76 GiB of exact-row and residual payload, compared with 3.31 GiB for G512. W5's lower resident record is therefore paired with much larger transient traffic on hard OLMo states. These maxima depend on the measured batch schedule and complement the per-state candidate fractions: both transient traffic and resident bytes contribute to the method's resource cost.

H.3 Full-request frontiers

Table 18: Measured maximum batches for complete prefill and eight-step requests

Model	Context	Stock	Canonical	G512	G512/canonical
Qwen3 1.7B	128	3,200	3,200	3,169	0.9903
	512	636	636	630	0.9906
	2,048	83	83	82	0.9880
Gemma 2 2B	128	3,285	3,285	3,011	0.9166
	512	751	751	743	0.9893
	2,048	119	119	120	1.0084
Llama 3.1 8B	128	1,800	1,800	1,802	1.0011
	512	336	336	338	1.0060
	2,048	39	39	39	1.0000

Two G512 frontier searches encounter a memory failure immediately above the committed batch. The Qwen context-2,048 search moves from 83 to 82, and the Gemma context-512 search moves from 744 to 743. Both lower batches rerun the unchanged complete request and satisfy the warm-up and repeat contract. The failed upper attempts and recovery events remain in the raw records.

H.4 Host lifecycle

The lifecycle measurements report residual resident set size (RSS) after releasing each method's owned buffers.

Table 19: Selected host-lifecycle records after method release

Model	Method	Resident payload	Owned host payload	RSS residual
Qwen3 1.7B	Canonical	593.5 MiB	0	0
	W5	199.4 MiB	808.9 MiB	0.09 MiB
	G512	303.7 MiB	901.3 MiB	0.01 MiB
Gemma 2 2B	Canonical	1,125.0 MiB	0	0
	W5	377.9 MiB	3,807.0 MiB	0.01 MiB
	G512	577.1 MiB	1,774.2 MiB	0
Llama 3.1 8B	Canonical	1,002.0 MiB	0	0.48 MiB
	W5	336.6 MiB	1,354.7 MiB	0
	G512	512.7 MiB	1,515.9 MiB	0

G512 uses less method-owned host storage than W5 on Gemma: 1,774.2 MiB versus 3,807.0 MiB, despite its larger resident certificate. On Qwen and Llama, G512 instead owns slightly more host storage. Every selected post-release RSS residual is below 0.5 MiB. Thus, these measurements distinguish live method storage from memory retained after release. Methods run within a shared process, so the table reports owned payloads and release residuals rather than isolated peak process memory.

I Relation to Neighboring Output-Head Systems

Nearby methods optimize different output contracts, so their reported latency numbers are not interchangeable. Table 20 states the distinction at the level needed to interpret our controls.

Table 20: Analytic positioning against nearby output-head methods

Method	Main mechanism	Reported contract	Distinction in this paper
CSV-Decode (Liu et al., 2025)	Offline clusters, centroid-radius bounds, sparse exact evaluation	Exact top-k certification and ε-certified softmax	Full-row survival is counted as a host fetch under a fixed BF16 decision
ARCHead (Kocabay et al., 2026)	Quantized low-rank core, grouped low-bit residuals, activation-metric correction	Packed approximate head evaluated by perplexity, logit fidelity, and throughput	Residual metadata certifies exclusion; remote BF16 rows determine the token
FlashHead (Tranheden et al., 2026)	Balanced clusters, multiprobe retrieval, selective quantization	Training-free head replacement evaluated by output accuracy and speed	Original BF16 rows and path identity remain authoritative
FlashSampling (Ruiz et al., 2026)	Tiled head matmul with fused Gumbel reduction	Exact categorical sampling without materialized logits	The full rows are remote and the claim fixes one finite counter program
Groupwise certificate	Per-row grouped centers, outward residual intervals, exact fallback	Greedy and fixed finite-counter identity to Equation 2	Resident bytes and host row traffic are explicit, while stock speedup is not achieved

The centroid controls in Tables 11 and 12 isolate one geometric difference under our score and host-fetch accounting. They do not reproduce the optimized systems in Table 20. A latency comparison would require one hardware and model stack plus a common output contract. The available evidence supports the mechanism-level conclusion that row-level grouped uncertainty is effective for exact host-row exclusion, while approximate head replacement, clustered local selection, and fused local sampling address other performance goals.

J Reproducibility Artifacts

J.1 Supplement contents

The anonymous supplement contains one self-contained Python program for a single Google Colab cell, a run guide, and machine-readable result records. The program downloads the specified public models and datasets and computes all empirical results from those inputs. It checks numerical kernels, probes available accelerator and host capacity, and selects staging batches from measured memory. Models, transient key-value tensors, and scratch stay on local notebook storage. Only compact results and selected recovery checkpoints are mirrored to Google Drive within a fixed storage budget. Atomic completion records allow finished work to be skipped after restart, and the result archive downloads only after completion.

The completed archive contains the executed source, configuration, model and workload identities, per-state candidate and witness rows, decode token arrays, timing repetitions, capacity observations, and recovery events. The program computes the percentile and rank statistics in Tables 9 and 10 from its own observations. Separate rendering scripts read the completed report to produce the fourteen empirical tables and four figures used here. The proofs, screening example, analytic cost table, and literature comparison are supplied as manuscript source.