PHOTOFINISH: ADAPTIVE PRECISION FOR EXACT LANGUAGE MODEL SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Sampling a token is a race with a single winner, yet output projection reads every weight at a fixed precision. PhotoFinish makes precision follow the race. A low-bit scan bounds token scores under fixed Gumbel perturbations, and an exact evaluation raises the threshold for remaining contenders. Only rows that can still win receive additional precision. Valid bounds certify each exclusion, preserving the same sampled token as a full scan. We characterize the minimum information required to determine the winner and analyze when another precision stage repays its cost. The analysis gives a tight worst-case guarantee for a reference access policy and explains why shrinking the candidate set need not reduce runtime. Across three large-vocabulary language models at small decoding batches, PhotoFinish outperforms the fastest of three evaluated full-scan samplers, including FlashSampling, with sampling-head speedups up to 2.61× and lower complete GPU decode-step latency during natural generation. Matched INT8 controls isolate the benefit of selective four-bit reads. PhotoFinish opens a path to exact sampling in which precision is allocated to the decision being made. Open-source code and reproduction materials accompany the submission.

1 INTRODUCTION

Sampling a token can require reading hundreds of megabytes of output weights to return a single token ID. At small decoding batches, this traffic makes the output layer sensitive to memory bandwidth. FlashSampling fuses projection and sampling to avoid materializing logits (Ruiz et al., 2026), but still reads every weight at a fixed precision. PhotoFinish addresses the precision of those reads: *how much information is needed to determine this sample?*

Gumbel-max makes the opportunity explicit. Draw the random perturbations first, then find the winner of the resulting race. Once a token's upper score bound falls below an attained score, its remaining weight bits cannot change the outcome. Close contenders need a sharper comparison; clearly trailing tokens can leave the race. This turns precision into a resource allocated to the realized sampling decision.

PhotoFinish implements this precision-on-demand principle (Figure 1). A packed low-bit scan bounds every token's score. One exact evaluation establishes an incumbent, eliminating rows that can no longer win. Unresolved rows receive either the remaining low bits or the original BF16 weights. Every exclusion has a valid certificate, and every route retains the same random draw. The result is the same sampled token as the full-scan reference.

Our analysis separates the information needed to settle a race from the cost of obtaining it. Minimum interval certificates provide an instance-specific access benchmark, and a reference policy achieves a tight worst-case guarantee in a two-level model with optional refinement. A conditional analysis explains a second tradeoff: additional precision can remove candidates yet cost more than it saves. Under symmetry, refinement is worthwhile over an intermediate band of candidate counts. These results motivate measuring complete execution costs when choosing a GPU route.

On three large-vocabulary language models at batches one and four, PhotoFinish outperforms the fastest of three evaluated full-scan samplers, including FlashSampling, with head speedups up to $2.61\times$ and complete GPU decode-step reductions of 1.45–4.42%.

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.

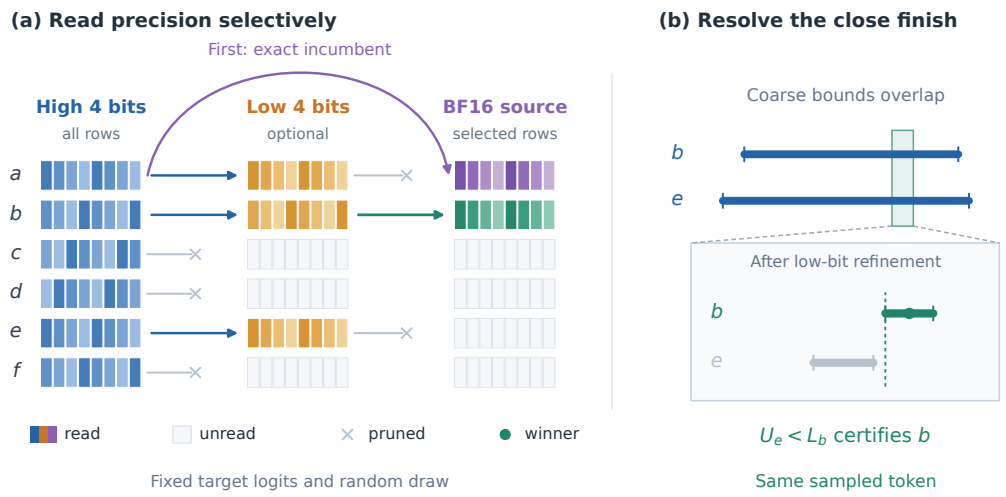

Figure 1. ***Precision follows the unresolved competition.*** *An illustrative GPU low-bit route with one refinement tile. A BF16 read of a establishes an incumbent. The compacted list retains a, b, e, which all receive low-bit reads; only b survives to final BF16. Colored cells are read; pale cells remain unread. Refined bounds satisfy $U_e < L_b$. Target logits and the Gumbel draw stay fixed.*

2 Exact Sampling with Partial Precision

PhotoFinish combines a cheap global scan with selective precision reads. Algorithm 1 summarizes the decision process; the interval construction below makes each exclusion valid.

Target. Let $W \in \mathbb{R}^{V \times D}$ contain the output weights for V tokens, w_i be its ith row, $h \in \mathbb{R}^D$ a hidden state, and $T > 0$ the temperature. For logits $z_i = w_i^\top h/T$ and independent standard Gumbels g_i,

$$w = \arg\max_i (z_i + g_i), \qquad \mathbb{P}(w = i) = \frac{\exp z_i}{\sum_j \exp z_j}. \tag{1}$$

We use this classical representation (Maddison et al., 2014). Our computational target is the winner for the *same* hidden state and random realization. The finite-precision reference used in experiments is specified in Section 5.

Bounds and exclusions. Suppose a cheap representation provides $L_i \le z_i \le U_i$. Define

$$B = \max_j (L_j + g_j), \qquad C = \{i : U_i + g_i \ge B\}. \tag{2}$$

For $i \notin C$, $z_i + g_i < B \le \max_j (z_j + g_j)$, so the token cannot be the winner. This argument is deterministic once the random draw is fixed. Low model probability alone is never a reason to exclude a token: a sufficiently large Gumbel perturbation can make it competitive.

An exact incumbent. The initial lower-bound maximum may be loose. We select $a = \arg\max_i (c_i + g_i)$ using coarse scores c_i, evaluate z_a with the reference arithmetic, and raise B to $\max\{B, z_a + g_a\}$. The anchor a need not be the true winner. It supplies an attained score, and the resulting threshold still cannot exceed the winning score. Its benefit is to remove candidates before their expensive rows are read.

Precision as an action. Intersecting a refinement with the old interval preserves all accumulated information. Newly available lower bounds or exact scores can raise the incumbent and eliminate more rows. The choice of reads may depend on the observed competition; correctness follows from the bounds used to exclude candidates.

Algorithm 1 PhotoFinish: precision follows the unresolved race

Require: Weights W, hidden state h, temperature T, fixed noise g, refinement policy π
1: Scan low-bit weights to obtain coarse scores c_i and valid intervals $[L_i, U_i]$
2: $a \leftarrow \arg\max_i(c_i + g_i)$; evaluate z_a exactly
3: $B \leftarrow \max\{\max_i(L_i + g_i), z_a + g_a\}$; $C \leftarrow \{i : U_i + g_i \geq B\}$
4: **while** $|C| > 1$ and some score in C is not exact **do**
5: Use π to choose further precision reads for unresolved rows in C
6: Intersect refined intervals; raise B using valid lower bounds or exact scores
7: $C \leftarrow \{i \in C : U_i + g_i \geq B\}$
8: **end while**
9: **return** the sole contender, or $\arg\max_{i \in C}(z_i + g_i)$ when all are exact

Proposition 1 (Adaptive refinement preserves the sample)**.** *Fix a hidden state and a random realization. If all intervals contain the target scores, every exclusion uses a valid incumbent, and refinement eventually certifies a winner or evaluates all remaining candidates exactly, the returned token equals the full-scan winner. This holds for any observation-dependent choice of refinements.*

Proof. The incumbent never exceeds the true maximum. Consequently the winner's upper bound never falls below it, so the winner is retained after every action. A final exact maximum or a valid winner certificate returns that winner. □

A fresh key is drawn for each new sampling event. Overflow and numerical recovery reuse the event's key and execute the full reference scan. Drawing new noise after an observation-dependent overflow would define a different selection process.

Constructing the intervals. For a reconstructed weight row $\widetilde{w}_i$, residual $r_i = w_i - \widetilde{w}_i$, and positive diagonal metric Λ,

$$|r_i^\top h| \leq \sqrt{h^\top \Lambda h}\sqrt{r_i^\top \Lambda^{-1} r_i}. \tag{3}$$

The residual norm is precomputed; the query norm is shared across rows. Any positive Λ gives a bound for every query. Calibration can tighten it without making correctness depend on the calibration distribution. We use inverse coordinate RMS from development states and add floating-point allowances for the interval arithmetic.

The packed representation splits each signed INT8 code as $Q = 16P + L$, where $P \in [-8, 7]$ and $L \in [0, 15]$. The first scan uses the high nibble with center $16P + 7.5$. A refinement can read L to obtain the full INT8 reconstruction, or skip it and read the original BF16 row. Full-precision weights remain resident. The method saves online traffic at the cost of additional low-bit storage; it still scans the full vocabulary at low precision.

3 How much information settles the race?

A sampler can often identify the winner before knowing every score. To measure how much information this decision requires, we ask for the cheapest set of precision reads that rules out every alternative winner. This gives an instance-specific comparator for an access policy.

Interval access model. Each row has nested levels $[L_i^{(k)}, U_i^{(k)}]$, ending in its exact logit. An access reveals the requested refinement; unobserved answers are fixed in advance. Apart from the intervals, the algorithm has no information coupling possible completions of different rows. The initial coarse scan is already available. Let $C_i(k)$ denote cumulative additional access cost at level k, with $C_i(0) = 0$. We write strict comparisons for continuous noise; lexicographic score/token-ID comparisons handle finite-precision ties.

Theorem 2 (Minimum interval certificate)**.** *The current levels certify token j for every consistent completion if and only if*

$$L_j^{(k_j)} + g_j > \max_{i \neq j}(U_i^{(k_i)} + g_i). \tag{4}$$

For the actual winner w, set $\ell_i(k) = \min\{\ell : U_i^{(\ell)} + g_i < L_w^{(k)} + g_w\}$, with infeasible choices assigned infinite cost. The minimum certificate cost is

$$C_{\mathrm{cert}} = \min_k \left[C_w(k) + \sum_{i \neq w} C_i(\ell_i(k)) \right]. \tag{5}$$

The first condition is sufficient by interval separation. If it fails, the rectangular information model permits a completion in which another row wins. For a fixed winner level, every other row must reach the indicated exclusion level, giving Eq. (5). The minimizing combination is an offline comparator: an online policy is not given future intervals or the winner. In particular, the winning logit itself need not be evaluated at full precision.

A competitive reference policy. Certificates connect precision allocation to selection under interval uncertainty. Witness-set methods identify updates that any successful policy must make (Gupta et al., 2011); local-ratio analysis handles weighted, repeated interval queries (Halldórsson & de Lima, 2021). We use these tools to charge a policy's reads against the information required by an optimal certificate.

In a chain model, each row advances one level at a time. Until a certificate exists, take the rows with the two largest current perturbed upper bounds. The next updates of the non-exact rows form a witness: every feasible certificate must include at least one of them. A weighted policy reduces the residual prices of the witness updates by their common minimum, performs updates whose price reaches zero, and retains the residual prices of deferred updates. The resulting cost is at most $2\mathrm{OPT}_{\mathrm{chain}}$; Appendix A gives the charging argument.

Theorem 3 (Tight two-level competitive ratio)**.** *After the initial scan, let a row admit a low-bit refinement of cost ℓ and a full-row read of cost f, where $\ell/f = 1/4$. A full read may skip the low-bit refinement; reading it after refinement still costs f. Against arbitrary valid nested interval answers, the reference policy has cost at most $\frac{5}{2}\mathrm{OPT}_{\mathrm{skip}}$. No deterministic interval-access policy has a smaller pure multiplicative worst-case ratio.*

The upper bound follows by inserting a low-bit read before each skipped full read in an optimal certificate: $\mathrm{OPT}_{\mathrm{chain}} \leq (1 + \ell/f)\mathrm{OPT}_{\mathrm{skip}}$. The lower bound uses two rows whose intervals leave a choice between revealing a cheap decisive interval and paying for full values. The three possible positions of the first full read force a ratio of at least $5/2$ (Appendix A). For output rows, the $1:4$ ratio is the requested payload of a low nibble versus BF16, $D/2$ versus $2D$ bytes. Metadata, cache lines, sorting, and kernel launches are outside this cost model. The theorem concerns the reference access policy, not the latency of the GPU policy.

4 The value of another precision stage

The cheapest certificate tells us what would suffice if future refinements were known. Choosing the next read online also requires estimating what it will accomplish. The first screening pass provides evidence about that value: its survivors are selected by the same random competition that the next stage must resolve. Treating them as fresh independent samples loses this information.

One fixed incumbent. Let c_i be fixed coarse logits, $e_i = z_i - c_i$, $q = \mathrm{softmax}(c)$, and $A = \arg\max_i(c_i + g_i)$. After evaluating $A = a$, hold the incumbent at $z_a + g_a$. At refinement level b, let N_b count non-anchor rows whose upper perturbed score exceeds this incumbent. Define

$$\tau = \Big(\sum_j e^{c_j}\Big) e^{-\max_j(c_j+g_j)}, \qquad \eta_{i,a}^{(b)} = q_i \big[e^{U_i^{(b)} - c_i - e_a} - 1\big]_+. \tag{6}$$

The exponential-race representation underlying Gumbel sampling (Maddison et al., 2014) yields the following coupled description.

Proposition 4 (Shared residuals across refinements)**.** *The variable $\tau \sim \mathrm{Exp}(1)$ is independent of A. Conditional on $A = a$ and $\tau = t$, there are independent $E_i \sim \mathrm{Exp}(1)$ such that the candidate indicator at every level is*

$$I_i^{(b)} = \mathbf{1}\{E_i \leq \eta_{i,a}^{(b)} t\}, \qquad i \neq a. \tag{7}$$

The same E_i is used at every level of a row.

The shared τ couples the surviving rows: they are independent conditional on (a, t) but generally correlated after integrating out t. For two levels, write $u_i = \eta_{i,a}^{(0)}$ and $v_i = \eta_{i,a}^{(1)}$. Appendix B derives their joint count law. Throughout this analysis, scores and interval families are fixed, and expectations concern Gumbel randomness conditioned on the stated observations.

A count can encode both size and difficulty. To expose the decision structure, suppose the $M = V - 1$ non-anchor rows have $u_i = \alpha > 0$ and $v_i = \rho\alpha$, $0 \le \rho \le 1$. This symmetry assumption jointly constrains the coarse probabilities, interval slack, and anchor error.

Theorem 5 (Conditional work and a band of refinement)**.** *Let $H = M + 1/\alpha$ and observe $N_0 = m$. Then $e^{-\alpha\tau} \mid (A = a, N_0 = m) \sim \mathrm{Beta}(H - m, m + 1)$ and, for $m \ge 1$,*

$$s_m := \mathbb{E}[N_1 \mid A = a, N_0 = m] = H\left[1 - \frac{\mathrm{B}(H - m + \rho, m)}{\mathrm{B}(H - m, m)}\right], \quad s_0 = 0. \tag{8}$$

The sequence s_m is discretely convex. If a full row costs d, a refinement costs r per row plus setup γ, the advantage of refinement,

$$D_m = dm - (\gamma + rm + ds_m), \tag{9}$$

is discretely concave. Among these two actions, using only the observed count, the counts for which refinement is strictly better form an integer interval or the empty set.

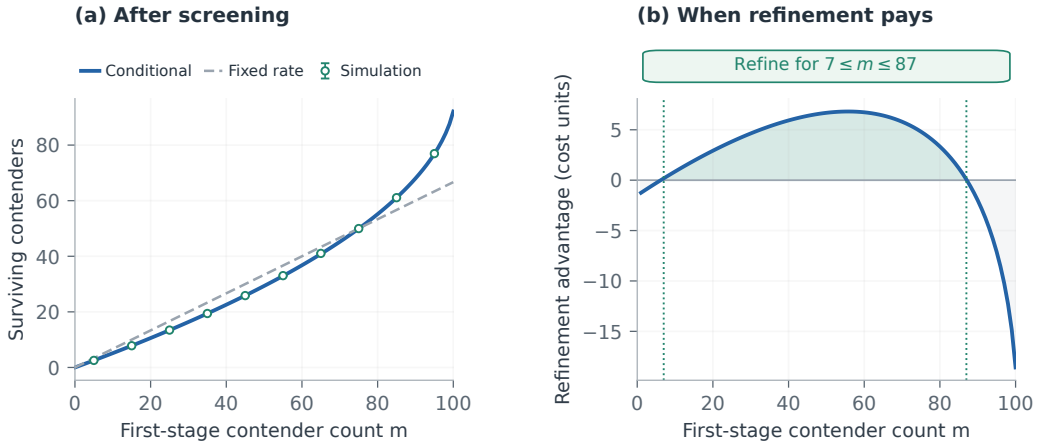

Figure 2. ***Refinement can be worthwhile only in the middle.*** *Symmetric model with $M = 100$, $\alpha = 1$, $\rho = 1/2$, $d = 1$, $r = 1/4$, $\gamma = 1.5$. Left: conditional predictions, the unconditional-rate estimate, and conditional Monte Carlo means from 300,000 draws (every tenth count shown, bars ± 2 standard errors). Right: refinement is preferable precisely for $7 \le m \le 87$. Costs are relative model units, not GPU measurements.*

Figure 2 gives a realizable example. Small candidate sets cannot amortize the extra stage. Large observed sets shift the posterior toward a harder competition, so another low-bit stage removes fewer of them. At $m = 100$, the expected remaining count is 92.08, not the 66.67 predicted by a constant survival fraction. The reference anchor can be wrong; final exact evaluation of the remaining candidates still recovers the winner.

For heterogeneous intervals, identities matter as well as counts, and the convexity conclusion need not hold. Raising the incumbent during refinement also changes the conditioning. Our GPU policy uses development-calibrated thresholds rather than evaluating this posterior online. The result explains why a fixed survival-rate estimate can misprice another stage and identifies conditions under which count-only decisions have a precise structure. Proofs, a heterogeneity counterexample, and additional precision-depth results appear in Appendix B.

5 GPU REALIZATION

We implement PhotoFinish in Triton (Tillet et al., 2019). The first scan reads packed high nibbles and forms score intervals. A separate exact evaluation establishes the anchor threshold. The compacted list retains the anchor, so the low-bit pass can revisit it before pruning. Candidate compaction, optional low-nibble reads, BF16 refinement, and final reduction remain on the device. Branching uses device-resident counts and is compatible with CUDA graph replay.

Choosing a stage. At batch one, the scan uses vector kernels; batch four shares coarse weight access through Tensor Core tiles. A low-bit stage adds launches and intermediate traffic even when it reduces candidate counts. In requested weight bytes, refining K_4 rows and then reading K_8 full rows saves payload over direct full refinement only if

$$(D/2)K_4 + 2DK_8 < 2DK_4, \quad \text{or} \quad K_8 < \tfrac{3}{4}K_4. \tag{10}$$

Runtime also depends on layout, synchronization, and cache state. We therefore select among INT8 screening, direct high-nibble-to-BF16 refinement, and candidate-count thresholds on development prompts, then freeze the choices before confirmation. The selected rules use direct four-bit screening for both Qwen models; Llama refines low bits when $K_4 > 1024$ at batch one or $K_4 > 64$ at batch four; TinyLlama uses INT8 paths.

Numerical target and storage. The full-scan reference reads BF16 hidden states and output weights, uses a fixed FP32 reduction and finite-precision Gumbel mapping, and applies a deterministic token-ID tie rule. The exact row path uses the same operations. Intervals include residual bounds, path-specific arithmetic allowances, norm margins, and directed rounding. Appendix D.9 derives the score enclosure, including temperature division, and states the Tensor Core arithmetic assumption. Unusable intervals trigger conservative retention or same-key full-scan recovery. We separately test sample agreement with this operational reference and with cuBLAS FP32 logits.

The original output matrix stays resident. A high-nibble representation adds approximately $VD/2$ bytes plus metadata; a low-nibble plane adds another $VD/2 + 8V$ bytes. The frozen Llama, Qwen3 0.6B, Qwen3 1.7B, and TinyLlama paths add 253.0, 75.9, 150.1, and 62.9 MiB, respectively. Construction peaks, head workspace, and preprocessing time are measured separately in Appendix D.8.

6 EXPERIMENTS

The experiments ask three questions: **(1)** does selective precision preserve the sampled token, **(2)** does it accelerate the complete head and decoding step, and **(3)** which precision reads justify their cost? Coupled sampling checks answer the first, full-scan comparisons the second, and matched controls and access experiments the third.

Models and workloads. The models are Llama 3.2 1B, Qwen3 0.6B, Qwen3 1.7B, and TinyLlama 1.1B, with vocabulary sizes 128,256, 151,936, 151,936, and 32,000. Output heads are BF16; the Llama checkpoint has an FP8 body and the others have BF16 bodies. The main study uses one RTX 5090 D with 32 GB memory and 96 MiB L2, PyTorch 2.13.0, and Triton 3.7.1. Hardware and model revisions are listed in Appendix D.

Public inference inputs come from the 80 first-turn MT-Bench questions (Zheng et al., 2023). The head workload retains each last prefill state and up to 16 generation states, stopping at EOS: 5,420 sampling states across models. Calibration and policy selection use separate development prompts. MT-Bench supplies inputs, not quality scores.

Natural generation uses the first question per category at B1 and the first four at B4: 32 unique prompts and 40 slots per model across eight categories. Complete chat inputs are retained. Requests stop at EOS or 64 output tokens, including the prefill sample. Finished slots use PAD, with their remaining static-batch work charged.

Baselines and timing. The full-scan comparators are a native fused reference, cuBLAS with FP32 logits plus Gumbel selection, and FlashSampling (Ruiz et al., 2026). INT8 controls include the original sampler, a cached-norm implementation, and a matched variant sharing the four-bit path's

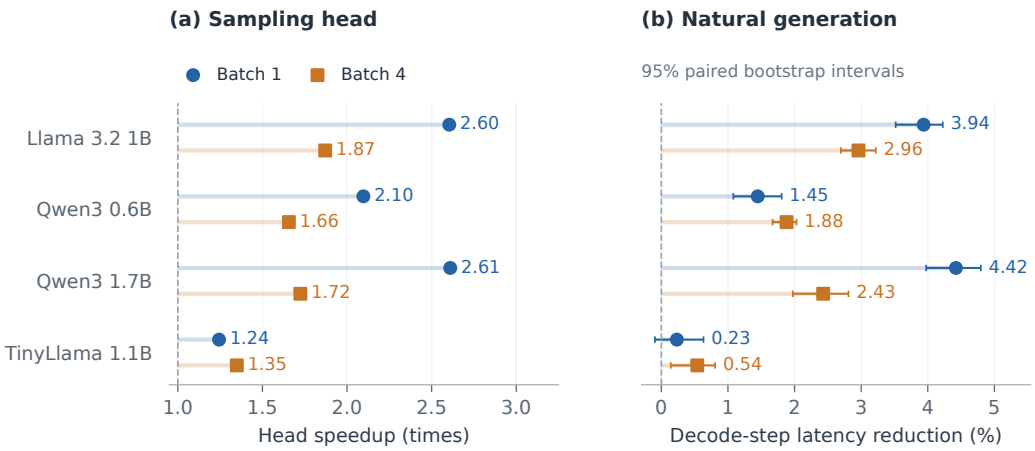

Figure 3. ***Head acceleration carries through to decoding.*** *Left: cold-cache complete sampling-head speedup over the fastest measured full scan. Right: natural, EOS-aware GPU decode-step reductions with 95% paired bootstrap intervals over request groups and trials. Large-vocabulary models benefit at both batch sizes; TinyLlama's full-step gains are below one percent.*

Table 1. Complete sampling-head latency.

Model	B	Full scan	Opt. INT8	Matched INT8	PhotoFinish	Gain
Llama 3.2 1B	1	336.38	202.38	208.04	129.17	2.60×
Llama 3.2 1B	4	345.16	198.81	205.09	184.49	1.87×
Qwen3 0.6B	1	215.11	151.74	156.79	102.60	2.10×
Qwen3 0.6B	4	214.37	139.70	144.89	129.44	1.66×
Qwen3 1.7B	1	394.89	251.18	243.21	151.31	2.61×
Qwen3 1.7B	4	404.48	313.26	241.01	234.60	1.72×
TinyLlama 1.1B	1	91.49	67.81	–	73.57	1.24×
TinyLlama 1.1B	4	91.02	67.48	–	67.50	1.35×

Notes. *Times are in µs. Full scan is the fastest of the three full-scan comparators. Opt. INT8 caches query norms; matched INT8 additionally matches the metric and exact-anchor pipeline of direct four-bit screening. The selected TinyLlama paths already use INT8.*

metric, anchor procedure, query statistics, capacities, tiles, and exact kernel. The complete head includes compaction and fallback. Each model uses 32 state starts, three trials, and ten replays per trial. A separate buffer three times L2 is touched before timing; cache clearing is excluded. We average the per-state median trial means.

Natural-generation timing uses native Hugging Face operations captured in CUDA graphs, including the model body, KV updates, sampling, and static-batch work. Six providers are measured in 12 balanced cyclic and reverse orders, including two independent captures of the selected policy. Total GPU duration is divided by total batch decode steps. Prefill and the first sample are excluded from this latency metric. A common saved trajectory supplies the next input so every provider faces identical model work, including FlashSampling's different random-index layout.

Paired bootstrap intervals resample request groups and timing trials, retain provider pairing and decode-step weighting, and reselect the fastest full-scan baseline. They describe variation within each recorded process. We additionally repeat Llama B4 and Qwen3 1.7B B1 in independent processes (Appendix D.8).

Table 2. Natural-generation GPU decode steps.

Model	Batch	Full scan	PhotoFinish	Reduction [95% interval]	Repeat
Llama 3.2 1B	1	4986.48	4790.13	3.94 [3.52, 4.23]	0.13
Llama 3.2 1B	4	5320.03	5162.58	2.96 [2.69, 3.22]	0.12
Qwen3 0.6B	1	3454.71	3404.72	1.45 [1.08, 1.81]	0.21
Qwen3 0.6B	4	4574.18	4488.19	1.88 [1.67, 2.03]	0.02
Qwen3 1.7B	1	5085.95	4861.02	4.42 [3.98, 4.80]	0.02
Qwen3 1.7B	4	6890.66	6723.26	2.43 [1.97, 2.81]	0.01
TinyLlama 1.1B	1	2978.72	2971.79	0.23 [-0.10, 0.63]	0.10
TinyLlama 1.1B	4	4063.14	4041.23	0.54 [0.14, 0.81]	0.06

Notes. *Times are in μs per batch step. Reductions and repeat differences are percentages; brackets give 95% paired bootstrap intervals. Repeat is the absolute difference between two captures of the same policy.*

RQ1: Does selective precision preserve the sample? All selected-policy outputs agree with the coupled full-scan reference. The public-head study covers 216,800 distinct state/temperature/key events under normal execution, forced-capacity fallback, and a cuBLAS cross-check. Normal-capacity checks have no overflows. The tests exercise 169,751 forced final-stage fallbacks and 54,400 first-stage fallbacks on Llama. Natural generation adds 9,807 sampling events, including 160 prefill and 9,647 live decode events, with no coupled mismatches.

The comparison holds hidden states and target arithmetic fixed. Model-body arithmetic can itself change the sampled token; Appendix D reports cache and full-prefix interventions that isolate this effect.

RQ2: Does the gain survive complete execution? In the main study, PhotoFinish beats the fastest evaluated full scan at all six large-vocabulary operating points: head speedups are 2.10–$2.61\times$ at B1 and 1.66–$1.87\times$ at B4 (Table 1). The gains survive model-body and KV work, reducing full GPU decode-step latency by 1.45–4.42% (Table 2). The selected paths also outperform optimized INT8. TinyLlama's smaller output head leaves little full-step benefit.

Native-platform confirmation. On H800 PCIe with native FlashSampling scheduling, the unchanged Qwen3 1.7B policy yields $1.34\times$ head speedup and 0.71% lower decode-step latency at B1. B4 is effectively tied, and its matched INT8 head is faster (Appendix D.3).

RQ3: Which precision reads justify their cost? *Four-bit screening provides the clearest precision gain at B1.* Under the same INT8 parent representation, metric, anchor procedure, and exact kernel, direct four-bit screening reduces head latency by 34.57–37.75% at B1 and 2.71–10.69% at B4 (Figure 4). The anchor's identity can differ with the coarse scores. The B4 intervals include zero for Llama and Qwen3 1.7B: selective exact reads already give INT8 much of the benefit there.

Candidate reduction alone does not determine the best route. On Llama B1, always reading the low bits reduces mean final candidates from $2{,}304$ to 1.31, yet head latency changes from $130.03\,\mu$s to $130.39\,\mu$s. At B4, the same refinement reduces latency from $199.27\,\mu$s to $184.97\,\mu$s. The additional information pays only when its acquisition cost is recovered. Dynamic stage choice adds a smaller increment over the best fixed four-bit route: 0.66% at B1 and 0.26% at B4 (Table 5). The main acceleration comes from selective precision reads; the conditional theory explains why another stage needs a cost-sensitive decision.

Certificates expose how little additional information can suffice. Across 128 model-derived interval races, the sequential reference policy's aggregate additional payload is within 0.014–0.426% of the optimal skip-enabled certificate; per-event maxima range from 1.000291 to 1.230769. This measures information access under FP64 intervals, separately from GPU scheduling. Exhaustive checks cover 300 synthetic instances and 273,000 level vectors; a 36-state decision-tree search recovers the $5/2$ lower bound. The conditional formulas are checked with 3.6 million Gumbel events and 48 integration comparisons (Appendix D).

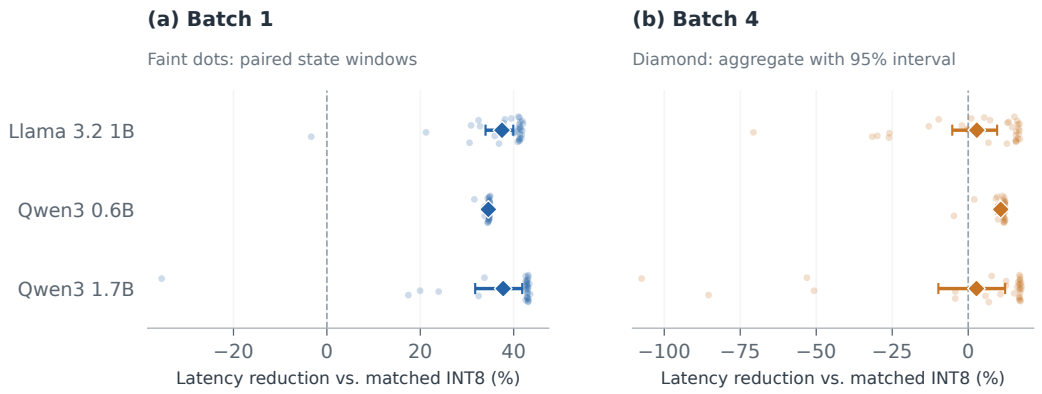

Figure 4. ***Four-bit screening under a matched setup.*** *Direct four-bit-to-BF16 refinement versus INT8 with the same metric, query statistics, anchor procedure, capacity, tiles, and exact kernel. Faint dots show reductions for the 32 paired state windows; diamonds show the aggregate reduction with 95% paired bootstrap intervals. Positive values favor four-bit screening. Every online head stage is included.*

7 Related work

Sampling as optimization. A* Sampling (Maddison et al., 2014) uses bounds and delayed evaluation in a Gumbel process. Mussmann et al. (2017) combine random perturbations with precomputed maximum-inner-product search. These works establish that sampling need not evaluate every alternative fully. PhotoFinish studies the precision of those evaluations: valid intervals permit staged weight reads while retaining the same realized winner.

Efficient output layers and memory traffic. FlashSampling (Ruiz et al., 2026) fuses full projection with tile-wise Gumbel maxima, eliminating logits materialization while retaining fixed-precision weight reads. FlashHead (Tranheden et al., 2026) combines balanced clustering, multiprobe retrieval, and quantization; its sampling path selects clusters before sampling tokens within them. PhotoFinish preserves the full-vocabulary Gumbel race and excludes rows only when their bounds certify that they cannot win. Its target is selective output-weight precision traffic, complementing the broader IO-aware approach exemplified by FlashAttention (Dao et al., 2022).

Queries under uncertainty. Witness-set and local-ratio tools provide the basis for our reference policy (Gupta et al., 2011; Halldórsson & de Lima, 2021). Our two-level result accounts for the option to skip a precision stage and gives a matching lower bound. The conditional analysis then addresses the value of a read after the random screening outcome is observed. Together, these results connect the information required for an exact sampling decision to the cost of revealing it.

8 Discussion

PhotoFinish makes precision follow the sampling decision. Certificates identify when further reads cannot change the winner, enabling lower weight traffic and latency even after logits materialization has been eliminated.

The evaluation covers full-vocabulary categorical sampling at small batches, with additional resident storage for low-bit copies. Query statistics and hardware determine which reads pay off. Top-p, beam search, complete log-probabilities, serving queues, and tensor parallelism are directions for extending the certificate approach.

Code availability and reproducibility. The anonymous supplementary archive contains the sampler, verification scripts, frozen experiment records, and a CPU-only paper build. Original code and records are provided under Apache-2.0. Proofs and experimental protocols appear in the appendix.

AI ASSISTANCE

The author developed the research direction, core sampling mechanism, theoretical derivations, overall software design, and manuscript. Codex and ChatGPT were used as assistants for literature search, exploration of candidate derivations and proof checking, code implementation and debugging, experiment design and execution, and result analysis. They also assisted with manuscript drafting and editing, synthetic development prompts and numerical test cases, and figures. The MT-Bench evaluation questions came from the cited public source.

Checks of AI-assisted outputs included proof review, numerical experiments, exhaustive certificate tests, and comparisons with dense sampling references. The author takes responsibility for the final paper, code, and results.

REFERENCES

Pierre Blanchard, Nicholas J. Higham, Florent Lopez, Theo Mary, and Srikara Pranesh. Mixed precision block fused multiply-add: Error analysis and application to GPU tensor cores. *SIAM Journal on Scientific Computing*, 42(3):C124–C141, 2020. doi: 10.1137/19M1289546. URL https://epubs.siam.org/doi/10.1137/19M1289546.

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. URL https://arxiv.org/abs/2205.14135.

Manoj Gupta, Yogish Sabharwal, and Sandeep Sen. The update complexity of selection and related problems. In *Foundations of Software Technology and Theoretical Computer Science*, pp. 325–338, 2011. doi: 10.4230/LIPIcs.FSTTCS.2011.325. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.FSTTCS.2011.325.

Magnús M. Halldórsson and Murilo S. de Lima. Query-competitive sorting with uncertainty. *Theoretical Computer Science*, 867:50–67, 2021. doi: 10.1016/j.tcs.2021.03.021. URL https://arxiv.org/abs/2012.09475.

Chris J. Maddison, Daniel Tarlow, and Tom Minka. A* sampling. In *Advances in Neural Information Processing Systems*, 2014. URL https://arxiv.org/abs/1411.0030.

Stephen Mussmann, Daniel Levy, and Stefano Ermon. Fast amortized inference and learning in log-linear models with randomly perturbed nearest neighbor search. In *Uncertainty in Artificial Intelligence*, 2017. URL https://arxiv.org/abs/1707.03372.

NVIDIA. *Parallel Thread Execution ISA*. NVIDIA, 2026. URL https://docs.nvidia.com/cuda/parallel-thread-execution/index.html.

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. URL https://arxiv.org/abs/2603.15854.

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, 2019. URL https://www.eecs.harvard.edu/~htk/publication/2019-mapl-tillet-kung-cox.pdf.

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. URL https://arxiv.org/abs/2603.14591.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2306.05685.

A CERTIFICATE ACCESS PROOFS

A.1 MINIMUM CERTIFICATES

Fix the Gumbel variables. If Eq. (4) holds, the lower score of j exceeds every other possible score. Conversely, if another row i has $U_i + g_i \geq L_j + g_j$, the rectangular interval model permits setting i to its upper endpoint and j to its lower endpoint. Thus the available information does not certify j (with equality resolved by the declared tie order). This establishes necessity and sufficiency.

For the actual winner w, fix a revealed level k for its interval. Any valid certificate with this winner level must reveal every other row to at least $\ell_i(k)$, so its cost is at least the corresponding term in Eq. (5). Conversely, revealing exactly these levels forms a certificate. Minimizing over the winner's level establishes the formula. No assertion is made that an online policy knows the minimizing levels in advance.

The model excludes additional constraints between rows. A matrix algorithm that exploits shared structure or an encoding that reveals multiple rows in one operation has a different information model. This is why we use the certificate as an access comparator rather than a lower bound on arbitrary GPU computation.

A.2 WEIGHTED WITNESSES

We now absorb g_i into the interval endpoints. In the chain model, an atom $e = (i, k)$ advances row i from level $k - 1$ to k and has a fixed positive cost c_e. A feasible certificate consists of query prefixes for the rows.

Before termination, let p, q have the largest and second-largest current upper endpoints in the total score order. Remove exact rows from this pair and form W from the next atom of each remaining row. The set is nonempty: if p were exact, its attained score would dominate all other upper endpoints and certify it.

Witness property. Every feasible certificate from the initial state intersects W. Suppose otherwise, and combine such a certificate with the algorithm's current history by taking the more-refined prefix of each row. Neither interval in the pair changes. Nontermination implies that p's lower endpoint is below q's upper endpoint, while q's lower endpoint is below p's upper endpoint. No other row can have a lower endpoint above p's upper endpoint. Thus the combined history cannot certify a winner. It must, however, preserve the alleged certificate, a contradiction.

Policy and charging. Initially each atom has residual cost c_e. For a witness W, subtract y_W, the smallest residual cost in W, from all its members and reveal every atom reaching zero. An atom that temporarily leaves a witness retains its residual cost. Each iteration advances at least one row, so the finite-level procedure terminates.

The reductions satisfy $\sum_{W: e \in W} y_W \leq c_e$. For any feasible certificate F, the witness property gives

$$\sum_{e \in F} c_e \geq \sum_{W} |F \cap W| y_W \geq \sum_{W} y_W.$$

Let S be the set of atoms actually executed. Each is executed only when its residual cost reaches zero, and $|W| \leq 2$, so

$$C_{\text{alg}} = \sum_{W} |S \cap W| y_W \leq 2 \sum_{W} y_W \leq 2\text{OPT}_{\text{chain}}.$$

This is a local-ratio/witness argument in the tradition of Gupta et al. (2011); Halldórsson & de Lima (2021). The quantity $\sum_W y_W$ is also a checkable lower bound along the run.

A.3 SKIPPING A LEVEL: UPPER AND LOWER BOUNDS

In the two-level model, a low read costs ℓ, a full read costs f, and reading low then full costs $\ell + f$. Inserting the omitted low reads into any skip-enabled optimal certificate yields a chain

certificate with cost at most $(1 + \ell/f)\mathrm{OPT}_{\mathrm{skip}}$. Therefore the reference chain policy costs at most $2(1 + \ell/f)\mathrm{OPT}_{\mathrm{skip}}$, which is $5/2$ for $\ell/f = 1/4$.

For the matching lower bound, consider initial perturbed-score intervals

$$I_0 = [0, 3], \quad I_1 = [1, 4],$$

and possible low-read replies

$$J_0 = [0.1, 2.9], \quad J_1 = [1.1, 3.9], \qquad N_0 = [0.4, 0.6], \quad N_1 = [3.4, 3.6].$$

Fix a deterministic policy. Until its first full read, answer any low read with the corresponding J_i. These intervals cannot certify a winner, so a correct policy eventually makes a full read.

1. If that row has not been read at low precision, give it its extreme value: 0.5 for row 0 or 3.5 for row 1, with the unobserved low reply fixed to N_i. Give the other row value 2. The offline certificate reads only N_i, costing ℓ, while the policy has spent at least f. The ratio is at least 4.
2. If that row has been read low but the other has not, give the first full read value 2. Give the other row its extreme value and low reply N_i. Again $\mathrm{OPT}_{\mathrm{skip}} = \ell$, whereas the policy has spent at least $\ell + f$.
3. If both rows have been read low, give the first full read value 2 and the other row its extreme value. The latter's observed J_i permits values on either side of 2, so the other full read is necessary. The policy spends at least $2\ell + 2f$. Reading only the extreme row in full would certify the answer, costing f, and the two actual low replies do not suffice. The ratio is at least $5/2$.

These cases cover all deterministic choices. After selecting a case, all replies and values can be fixed as a complete static instance producing the same observed history. The construction does not require queries to change the underlying values. A full read that also reveals its row's low information does not avoid the cost already paid. The lower bound permits all valid nested intervals; it is not a claim that the particular adversarial replies arise from every BF16 quantizer.

B Conditional refinement and precision depth

B.1 Exponential residuals

Define $Y_i = (\sum_j e^{c_j})e^{-(c_i+g_i)}$. The Y_i are independent exponentials with rates q_i, and $\sum_i q_i = 1$. Consequently

$$\mathbb{P}(A = a, \tau \in dt) = q_a e^{-t}\, dt.$$

Given $A = a$ and $\tau = t$, the memoryless property gives $Y_i = t + E_i/q_i$ for $i \neq a$, where the E_i are independent standard exponentials. The candidate condition is

$$U_i^{(b)} + g_i \geq z_a + g_a \iff E_i \leq q_i t \left(e^{U_i^{(b)} - c_i - e_a} - 1\right).$$

A nonpositive right side gives probability zero, establishing Proposition 4. The same residual for a row must be used at successive levels.

For two levels, let $u_i = \eta_{i,a}^{(0)}$ and $v_i = \eta_{i,a}^{(1)}$. Each non-anchor row is either excluded initially, excluded after refinement, or retained through both. Their conditional probabilities are $e^{-u_i t}$, $e^{-v_i t} - e^{-u_i t}$, and $1 - e^{-v_i t}$. Multiplying the generating terms and integrating over t gives

$$\mathbb{E}[x^{N_0} y^{N_1} \mid A = a] = \int_0^\infty e^{-t} \prod_{i \neq a} \left[e^{-u_i t} + x(e^{-v_i t} - e^{-u_i t}) + xy(1 - e^{-v_i t})\right] dt. \tag{11}$$

In particular,

$$\mathbb{E}[N_b \mid A = a] = \sum_{i \neq a} \frac{\eta_{i,a}^{(b)}}{1 + \eta_{i,a}^{(b)}}.$$

Two same-level indicators with parameters u, v have covariance, after integrating out τ,

$$\mathrm{Cov}(I_i, I_j \mid A = a) = \frac{uv}{(1+u)(1+v)(1+u+v)}.$$

Thus marginally independent treatment would discard a source of positive dependence.

B.2 THE POSTERIOR AFTER OBSERVING A SET

If C is the observed set of first-stage non-anchor survivors, Bayes' rule gives

$$f(t \mid A = a, C) \;\propto\; e^{-(1+\sum_{i\notin C\cup\{a\}} u_i)t} \prod_{i\in C}(1 - e^{-u_i t}), \qquad t > 0.$$

For $i \in C$ and fixed t,

$$\mathbb{P}(I_i^{(1)} = 1 \mid A = a, C, \tau = t) = \frac{1 - e^{-v_i t}}{1 - e^{-u_i t}}.$$

The actual conditional mean integrates this expression against the posterior. Candidate identities generally matter. These distributions condition on the stated statistics for fixed logits and interval families; using additional observed scores or read history requires updating the conditional information.

B.3 PROOF OF THE COUNT FORMULA AND CONVEXITY

Under symmetry, $X = e^{-\alpha\tau}$ initially has distribution $\mathrm{Beta}(1/\alpha, 1)$. Since $N_0 \mid X \sim \mathrm{Binomial}(M, 1 - X)$, its posterior is $\mathrm{Beta}(H - m, m + 1)$. Conditional on X and m,

$$N_1 \sim \mathrm{Binomial}\left(m, \frac{1 - X^\rho}{1 - X}\right).$$

Integrating the mean gives

$$s_m = \frac{m[\mathrm{B}(H - m, m) - \mathrm{B}(H - m + \rho, m)]}{\mathrm{B}(H - m, m + 1)},$$

which simplifies to Eq. (8). The count distribution itself is

$$\mathbb{P}(N_0 = m \mid A = a) = \binom{M}{m} \frac{1}{\alpha} \mathrm{B}(H - m, m + 1).$$

For $\alpha = 1$ it is uniform on $\{0, \ldots, M\}$.

Set $R_m = \mathrm{B}(H - m + \rho, m)/\mathrm{B}(H - m, m)$ for $m \geq 1$, with $R_0 = 1$. Gamma-function identities give

$$\frac{R_{m+1}}{R_m} = \frac{H - m - 1}{H - m - 1 + \rho}, \qquad d_m := s_{m+1} - s_m = \frac{H\rho R_m}{H - m - 1 + \rho}.$$

For $0 < \rho \leq 1$ and $0 \leq m \leq M - 2$,

$$\frac{d_{m+1}}{d_m} = \frac{H - m - 1}{H - m - 2 + \rho} \geq 1.$$

The denominator is positive, and the inequality is strict for $0 < \rho < 1$. At the endpoints, $s_m = 0$ for $\rho = 0$ and $s_m = m$ for $\rho = 1$. This proves discrete convexity and hence concavity of Eq. (9). The strict positive set of a concave sequence is an integer interval. At $m = 0$ there are no non-anchor candidates and no refinement cost is paid.

The example in Figure 2 is realizable with $V = 101$, $c_i = 0$, $z_a = 0$, $z_i = 0.5$ for $i \neq a$, conditioned on $A = a$. Take non-anchor upper bounds $U_i^{(0)} = \log 102$ and $U_i^{(1)} = \log 51.5$, with their negative counterparts as lower bounds. These give $\alpha = 1$ and $\rho = 1/2$. The strict refinement band is 7 through 87. Mean remaining costs under the declared affine cost model are 50.0000 for direct full refinement, 47.2799 for the best single lower threshold, and 46.3631 for the best count band.

B.4 ADDITIONAL INFORMATION AND HETEROGENEITY

If $\tau = t$ is also observed, the expected surviving count is

$$\mathbb{E}[N_1 \mid a, m, t] = m \frac{1 - e^{-\rho\alpha t}}{1 - e^{-\alpha t}}.$$

For $0 < \rho < 1$, the ratio increases strictly from ρ to 1 with t. Thus refinement is beneficial precisely when

$$\frac{1 - e^{-\rho\alpha t}}{1 - e^{-\alpha t}} < 1 - \frac{r}{d} - \frac{\gamma}{dm}.$$

This statistic separates the amount of work from a feature of the random competition. Its acquisition cost must also be paid. In the example, count and time reduce the analytical mean cost to 46.3257, compared with 46.3631 for counts alone; this is not a measured GPU improvement.

Symmetry is essential to the count-band result. For heterogeneous parameters $u = (10, 0.1)$ and $v = (5, 0.05)$, direct integration gives $s_1 = 0.9094288991$ and $s_2 = 1.5027684466 < 2s_1$. Even a common contraction ratio does not imply convexity. In another example, two rows with equal u but $v = 0$ and $v = u$ give different refinement outcomes when exactly one candidate survives, despite identical counts.

Across several levels, the history of counts is informative. With symmetric parameters $\alpha_0 \geq \cdots \geq \alpha_b$ and observations $n_0 \geq \cdots \geq n_b$, the posterior is proportional to

$$e^{-t} e^{-\alpha_0 t(M-n_0)} \prod_{j=1}^{b} (e^{-\alpha_j t} - e^{-\alpha_{j-1} t})^{n_{j-1}-n_j} (1 - e^{-\alpha_b t})^{n_b}.$$

The current count alone generally loses part of this information.

B.5 A CLOSE FINISH REQUIRES MORE PRECISION

Let $p = \mathrm{softmax}(z)$ and let Δ be the difference between the largest and second-largest true perturbed scores. For $V \geq 2$ and $x \geq 0$,

$$\mathbb{P}(\Delta > x) = \sum_w \frac{p_w}{1 + (1 - p_w)(e^x - 1)}. \tag{12}$$

To see this, use the true-score exponential race. Conditional on its winner w and normalized arrival time $t \sim \mathrm{Exp}(1)$, the waiting time to the next arrival has rate $1 - p_w$. The score gap exceeds x if this waiting time exceeds $(e^x - 1)t$. Integrating over t and then w proves Eq. (12).

Writing $G = 1 - \sum_i p_i^2$, we obtain

$$\mathbb{P}(\Delta \leq x) \leq G(e^x - 1), \qquad \mathbb{P}(\Delta \leq x) = Gx + O(x^2).$$

Suppose global refinement level b has true one-sided interval margins at most $\epsilon_b \leq \kappa 2^{-b}$. If B_* is the first level at which a certificate exists, then

$$\mathbb{P}(B_* > b) \leq \mathbb{P}(\Delta \leq 2\epsilon_b) \leq \min\{1, G(e^{2\kappa 2^{-b}} - 1)\}. \tag{13}$$

Indeed, when $\Delta > 2\epsilon_b$, the winner's lower endpoint exceeds every other upper endpoint. If $2\kappa 2^{-b_0} \leq 1$, summing the tail yields

$$\mathbb{E}[(B_* - b_0)_+] \leq 4(e-1)G\kappa 2^{-b_0} \leq 2(e-1)G.$$

This bounds extra global precision depth after a suitable starting level, not the total number of row reads. A nonzero residual floor must remain in ϵ_b; the implementation terminates at BF16 rather than assuming indefinitely shrinking quantization error.

C BATCHING EXACT READS

Consider the separate one-shot model in which every unknown row can be evaluated exactly at unit cost. Read up to B rows with the largest initial upper bounds per round and check for a certificate between rounds. Let Q be the number of row reads and R the number of rounds. Then

$$Q \leq \mathrm{OPT} + B, \qquad R \leq \lceil \mathrm{OPT}/B \rceil + 1.$$

Let K count unknown rows whose upper score is at least the winning score. Every such nonwinner needs a read, so $K \leq \mathrm{OPT} + 1$. After at most $\lceil K/B \rceil$ rounds the winner is certified, and $Q \leq B\lceil K/B \rceil \leq K + B - 1 \leq \mathrm{OPT} + B$. If a certificate already exists, no round is needed.

For all-unknown initial rows and Gumbel noise, define

$$K^* = \left| \{ i : U_i + g_i \geq \max_j (z_j + g_j) \} \right|, \qquad \mu = \mathbb{E}[K^* - 1].$$

Using the Gumbel-max identity with U_i replacing only row i's logit,

$$\mu = \sum_i \frac{e^{U_i}}{e^{U_i} + \sum_{j \neq i} e^{z_j}} - 1.$$

For a batch overhead γ and per-row cost c, $R \leq 1 + (K^* - 1)/B$ gives

$$\mathbb{E}T_B \leq (\gamma + cB)(1 + \mu/B).$$

The continuous upper bound is minimized at $B = \sqrt{\gamma\mu/c}$ when positive, followed by clipping and checking neighboring integers. This is a cost-model result. Estimating μ, maintaining upper-bound order, and actual GPU scheduling have additional costs.

D Experimental details and reproducibility

D.1 Fixed model sources

The following checkpoint revisions are used. The head weights are BF16 in every case.

Llama 3.2 1B. `RedHatAI/Llama-3.2-1B-Instruct-FP8-dynamic`, revision `e23d444f8d7da0a3e556cae44a7d3c46f127e642`; FP8 body, $V = 128256$, $D = 2048$.

Qwen3 0.6B. `Qwen/Qwen3-0.6B`, revision `c1899de289a04d12100db370d81485cdf75e47ca`; BF16 body, $V = 151936$, $D = 1024$.

Qwen3 1.7B. `Qwen/Qwen3-1.7B`, revision `70d244cc86ccca08cf5af4e1e306ecf908b1ad5e`; BF16 body, $V = 151936$, $D = 2048$.

TinyLlama 1.1B. `TinyLlama/TinyLlama-1.1B-Chat-v1.0`, revision `5243d158d6f4b356f1142ea8fd6a99cb5ac2c0e1`; BF16 body, $V = 32000$, $D = 2048$.

The GPU has 170 streaming multiprocessors and 96 MiB L2. The environment uses PyTorch 2.13.0 with CUDA 13.2, Triton 3.7.1, Transformers 5.16.1, and compressed-tensors 0.17.0. Calibration uses positive diagonal coefficients paired between the query and residual norms in Eq. (3). Six development metrics, including learned variants, were explored; they did not consistently outperform RMS, which remains the frozen rule.

D.2 Baseline compatibility

The FlashSampling implementation is pinned to source revision `1e5d8fa3eb9a3fe84cb56a1f2b9949f2e1d16b50`. The original warp-specialization path triggers a compiler assertion on the tested `sm_120`/Triton configuration. On the RTX 5090 D, we disable that scheduling option while retaining its four autotuning configurations. The H800 confirmation below preserves upstream scheduling. For changing random keys inside CUDA graphs, a local adaptation reads the key from a GPU scalar, preserving the noise expression and index mapping. The baseline's noise indexing differs from the native reference, so we align the body workload through saved input trajectories rather than require its individual samples to match.

The optimized INT8 comparator receives the same relevant coarse-scan tiling improvements as the four-bit implementation. The matched INT8 control additionally uses the same diagonal metric, query statistics, exact-anchor procedure, capacities, and exact-refinement kernel as direct four-bit screening. The current measurements requalify the numerical implementation without reselecting the development-frozen policy or retuning kernels on public inputs.

D.3 Confirmation with the native baseline on H800

We transfer the frozen Qwen3 1.7B checkpoint, hidden states, RMS metric, precision policy, and B1/B4 protocols to an H800 PCIe with 80 GB memory, 114 streaming multiprocessors, and a 350 W power limit. This environment uses driver 580.82.07, PyTorch 2.12.1 with CUDA 13.0, Triton 3.7.1, and Transformers 5.16.1. No policy or kernel parameters are retuned. FlashSampling retains all upstream autotuning configurations and enables warp specialization. Separate fixed-key and graph-resident-key checks cover 480 events with zero mismatches against its own logits and independently indexed noise. The operational-reference and compiled-instruction checks also pass.

The selected policy agrees with the coupled reference in 54,280 state/temperature/key events, including forced recovery, and 2,412 natural-generation sampling events. Natural trajectories are generated on H800; the earlier 5090 body reconstruction is not reused. Table 3 reports all three full-scan baselines. The B1 head gains $1.34\times$, with 0.71% lower full-step latency. At B4, both head and decode intervals include zero. The matched INT8 head takes $262.50\,\mu$s versus $356.42\,\mu$s for the frozen four-bit policy. Thus the B1 advantage survives native baseline scheduling, while the preferred screening precision and transfer benefit vary with the execution platform.

Table 3. Frozen-policy confirmation on H800 PCIe.

Scope	B	Native	cuBLAS	FlashSampling	PhotoFinish	Reduction [95%]
Head	1	570.65	359.84	350.67	262.13	25.25 [17.29, 30.46]
Head	4	1548.78	367.82	360.62	356.42	1.17 [-12.40, 11.51]
Decode	1	6257.56	6029.32	6000.21	5957.31	0.71 [0.22, 1.13]
Decode	4	7876.29	6755.01	6733.67	6738.49	-0.07 [-0.69, 0.46]

Notes. *Qwen3 1.7B; all latencies are in µs. Decode measures the full GPU batch step. Reduction is relative to the fastest of Native, cuBLAS, and FlashSampling, with a 95% paired bootstrap interval; the fastest baseline is reselected in each draw. PhotoFinish retains its original development-frozen policy.*

D.4 Natural requests and numerical replay

The eight categories are writing, role play, reasoning, mathematics, coding, extraction, STEM, and humanities. First-turn questions are pinned to FastChat revision `b494d0c6b4e7935f1764f8439e75da3e66beccc7`. Qwen uses its non-thinking chat-template setting. Inputs preserve complete prompts, token IDs, masks, logical lengths, random keys, active-slot flags, EOS transitions, and next inputs. Llama's template date is explicitly fixed because its default template can insert the current date.

Natural generation uses temperature one and at most 64 output tokens. The 40 request slots per model comprise 8 batch-one requests and 32 batch-four requests, with 32 distinct prompt IDs. The latency denominator counts batch decode steps; a live-token statistic separately charges padding work to real generated tokens. The first sample after prefill contributes to correctness counts but not decode-step timing.

An independent reconstruction recovers all 9,807 graph-trajectory sampling events. Ordinary DynamicCache changes 19 of the 9,647 decode samples, while uncached full-prefix execution changes 2 of 624 checked samples. These comparisons probe model-body numerical changes. For the inspected Llama and Qwen batch-one inputs, forcing mathematical SDPA removes the static/dynamic hidden-state discrepancy, whereas graph versus eager execution of the same static input already agrees. One Qwen batch-four checkpoint retains a discrepancy under mathematical SDPA. These interventions are diagnostic; no alternative body backend is credited as a sampling speedup.

Earlier synthetic fixed-work measurements repeated or truncated a chat prefix and ran past EOS to retain a fixed number of steps. They are excluded from the natural-generation results. In one Qwen 0.6B configuration, 629 of 640 historical timed slots were after the first EOS; that workload cannot substitute for natural requests.

D.5 Access simulation

Each model contributes the final prefill state of the first question from each category and four Gumbel keys, for 32 events. Every interval is checked to contain its FP64 target. The online policy sees future intervals only through a reveal interface; the offline comparator can inspect all levels and skip the low stage. All 128 policy winners agree with the FP64 reference. Three Qwen 0.6B events end with a certificate without any additional BF16 row read. The simulated access sequence is not the GPU's parallel schedule.

Table 4. Additional access cost after the coarse scan.

Model	Events	Policy mean	Optimum mean	Aggregate ratio	Maximum ratio
Llama 3.2 1B	32	2835.78	2835.38	1.000143	1.012346
Qwen3 0.6B	32	71.78	71.48	1.004263	1.230769
Qwen3 1.7B	32	402.48	402.12	1.000913	1.142857
TinyLlama 1.1B	32	2826.13	2825.73	1.000141	1.000291

Notes. *A full BF16 row has unit cost; a low-nibble row costs* $1/4$*. Intervals and targets use FP64. Aggregate ratios compare total policy and optimal-certificate costs; the maximum is the largest individual ratio.*

D.6 Executable evidence

The source includes `certificate_policy.py` for the weighted reference policy, `analyze_certificate_policy.py` for synthetic and model-interval checks, and `verify_certificate_lower_bound.py` for exact rational decision-tree enumeration. The latter checks 36 states for the two-row lower-bound construction without giving the policy a promise that inputs come from only the listed adversarial worlds.

`analyze_refinement_theory.py` compares joint-count integrals, conditional formulas, direct Gumbel simulations, and exhaustive minimum certificates. Its 48 symmetric integration checks have a maximum absolute joint-first-moment discrepancy of about 3.11×10^{-14}. A total of 3.6 million Gumbel events covers anchor-conditioned counts and gap/depth distributions. Numerical agreement supports implementation of the formulas; the theoretical claims follow from the proofs.

The GPU drivers retain the frozen policy and workload identities. `benchmark_adaptive.py` measures complete heads; `benchmark_generation.py` measures natural requests; and `verify_generation_reference.py` reconstructs saved trajectories. The manuscript's tables and vector figures are generated by `paper/build.py` from a compact measurement extract with source-file hashes. Neither manuscript construction nor plotting reruns model inference.

D.7 Paired timing uncertainty

We apply 20,000 bootstrap draws with seed 2026091402 to the frozen timing records. For natural generation, each draw samples request groups with replacement, then samples the 12 timing trials within each drawn group. Trial indices are shared by all providers. We aggregate total GPU durations and divide by the sampled groups' total decode steps, retaining the original latency estimator. The fastest of the three full-scan baselines is selected again within each draw. Table 2 reports the 2.5th and 97.5th percentiles.

For isolated heads, the estimator remains the mean of per-state-window medians of three trials. We resample trials within each window, paired across providers, and apply multinomial bootstrap weights to clusters of windows connected by shared prompt identities. Llama has 28 such clusters at batch one and 26 at batch four. This avoids treating neighboring windows from the same prompt as unrelated requests. The best fixed four-bit route is reselected within each draw.

Table 5. Incremental benefit of dynamic stage selection on Llama 3.2 1B.

Batch	vs. direct BF16	vs. always low bits	vs. best fixed route
1	0.66 [-1.05, 2.93]	0.93 [0.57, 1.28]	0.66 [-1.05, 1.07]
4	7.42 [2.83, 12.38]	0.26 [0.01, 0.59]	0.26 [0.01, 0.59]

Notes. *Sampling-head latency reductions are percentages, with 95% paired bootstrap intervals. Both fixed routes use four-bit screening and an exact incumbent; one always reads low bits before final BF16 refinement.*

These resampling intervals describe the deliberately selected prompts and paired trials within each process. Table 2 also reports the difference between independent captures of the same policy. Separate

process repeats are reported below; two process measurements are not treated as a process-level confidence interval. The analysis and frozen input hashes are provided by `summarize_uncertainty.py`.

D.8 Resource costs and independent process repeats

Table 6 measures the selected implementation above an already loaded native model. Resident bytes count unique sampler-owned CUDA storage, excluding the existing BF16 output matrix. Peak allocations are measured by the PyTorch CUDA allocator. Graph workspace is the peak additional allocation for an isolated head capture and replay, including returned outputs. It excludes model-body activations and KV caches. Construction peak is the larger extra allocation of parent INT8 quantization and representation construction. Preprocessing time includes both stages, synchronized with the device; model loading and disk IO are excluded. The current implementation forms FP64 residual arrays during construction, so its temporary peak is much larger than its persistent representation. These construction allocations are released before online sampling.

Table 6. Actual representation and head workspace costs.

Model	B	Resident MiB	Graph MiB	Build peak MiB	Build s
Llama 3.2 1B	1	252.95	1.97	17291	0.167
Llama 3.2 1B	4	252.95	7.84	17291	0.150
Qwen3 0.6B	1	75.93	2.33	10253	0.145
Qwen3 0.6B	4	75.93	9.29	10253	0.122
Qwen3 1.7B	1	150.12	2.33	20484	0.192
Qwen3 1.7B	4	150.12	9.29	20484	0.179
TinyLlama 1.1B	1	62.87	0.28	1813	0.085
TinyLlama 1.1B	4	62.87	0.99	1813	0.079

Notes. *MiB means 2^{20} bytes. Resident is additional persistent sampler storage. Graph is extra head-capture/replay allocation above the loaded model, representation, and query inputs. Build peak is additional temporary allocated memory above the loaded model. Build time is a synchronized wall measurement in seconds.*

The numerical revision retains all development-selected metrics and precision policies. Table 7 repeats only the prespecified Llama B4 and Qwen3 1.7B B1 operating points in new processes, with identical source, prompts, hidden-state starts, temperatures, keys, and sampled natural trajectories. Each process selects its fastest full-scan comparator from the same three baselines. Both process results are retained.

Table 7. Independent process repeats.

Model	B	Head first	Head repeat	Decode first	Decode repeat
Llama 3.2 1B	4	1.87×	1.87×	2.96	2.88
Qwen3 1.7B	1	2.61×	2.60×	4.42	4.54

Notes. *Head values are complete-head speedups over the fastest full scan. Decode values are full GPU decode-step latency reductions in percent. Each result includes the original within-process trial protocol; no policy or kernel retuning intervenes.*

D.9 Implementation-level score enclosures

The interval argument must cover the arithmetic used to construct the certificate. We give the error budget for the saved representation and operational reference, including temperature division. Write $u = 2^{-24}$, $\eta = 2^{-149}$, $\nu = 2^{-126}$, and $\gamma_n(v) = nv/(1 - nv)$.

Domain and arithmetic. Weights and hidden states are finite BF16 values. Row scales, diagonal metric entries, and the temperature are finite positive FP32 values. Certificates are constructed for

the stored values; changing weights, codes, scales, or the metric requires rebuilding their metadata. Set $n = 2D + 8$ and require $2nu \le 1/2$. The code checks finite endpoints and an upper bound on the absolute products plus their arithmetic error, and rejects nonzero subnormal error components. An unusable first-stage interval requests the full reference; an unusable low-bit interval retains the candidate.

The SIMT path uses FP32 round-to-nearest multiplication, addition, and reductions. BF16 products have at most 16 significant bits; products of BF16 hidden values and integer codes have at most 15. Normal products therefore fit exactly in FP32. The Tensor Core path uses a block-FMA forward-error model: a block $a + \sum_{j=1}^{b} q_j h_j$ has relative absolute-sum allowance $\gamma_{b+1}(2u)(|a| + \sum_j |q_j h_j|)$, with an additional allowance for FP32 output conversion and underflow. Chaining such blocks is analyzed as in Blanchard et al. (2020); $2u$ allows directed or truncating rounding, and the bound accommodates arbitrary summation order. This block-FMA condition is an explicit arithmetic assumption. The PTX specification leaves BF16 MMA accumulation order, rounding, and subnormal handling unspecified (NVIDIA, 2026), so a single-precision accumulator declaration alone does not establish it.

Residual and query norms. For the stored reconstruction $\widetilde{w}_i$, preprocessing forms $r_i = w_i - \widetilde{w}_i$ in FP64 and bounds $\|r_i\|_{\Lambda^{-1}}$. A shared query kernel bounds $\|h\|_\Lambda$. Both use an inflation of $1 + 8\gamma_{D+8}(2^{-53})$ before conversion to FP32 and an upward neighboring value. Finite BF16 operands and positive finite FP32 metric entries keep these FP64 intermediate products, squares, divisions, and sums within the FP64 normal range. The inflation covers residual formation, norm evaluation, and conversion, giving

$$R_i \ge \|r_i\|_{\Lambda^{-1}}, \qquad H \ge \|h\|_\Lambda.$$

The original INT8 path instead accumulates $\widehat{s} = \mathrm{fl} \sum_j h_j^2$ in FP32. With $s = \sum_j h_j^2$, normal BF16 squares are exact in FP32 and the sum obeys $|\widehat{s} - s| \le \gamma_D(u) s + D\eta$. It uses an upward square root of an upward evaluation of $(\widehat{s} + D\eta)/(1 - \gamma_D(u))$. Nonzero queries with $\widehat{s} < \nu$ recover by a full scan; an actual zero query has bound zero.

Dot products. Let $\widehat{x}_i$ be the reference dot product before temperature scaling, and $\widehat{c}_i$ the computed coarse value. For $f = 16$, $m = 7.5$, and $Q = fP + L$, the row metadata bounds the following absolute sums:

$$\begin{aligned} A_i^{(8)} &\ge \sum_j |w_{ij}| + s_i \sum_j |Q_{ij}|, \\ A_i^{(4)} &\ge \sum_j |w_{ij}| + s_i \big(f \sum_j |P_{ij}| + mD \big), \\ A_i^{(4+4)} &\ge \sum_j |w_{ij}| + s_i \big(f \sum_j |P_{ij}| + \sum_j |L_{ij}| \big). \end{aligned}$$

These quantities are also evaluated in FP64, inflated, and converted outward. Keeping the absolute high- and low-bit terms separately is necessary because their computations can incur errors even when their sum cancels.

Expanding either arithmetic path gives at most $n = 2D+8$ relative-rounding factors along a product's propagation path, including block output conversions, row scaling, and the high-bit translation. Define

$$F_i = 4n\nu(1 + A_i), \qquad E_i = HR_i + \gamma_n(2u)\|h\|_\infty A_i + F_i,$$

with $F_i = 0$ for a zero query. The absolute term covers gradual underflow in the reference and flushing in the block-FMA model. Each lost subnormal hidden input contributes at most $\nu|q_j|$; a nonzero integer code has absolute value at least one, so the corresponding part of A_i bounds both the count of nonzero products and their later scale. The translation is covered by mDs_i. Allowing up to n loss sites in each dot path and subsequent amplification at most $1/(1 - 2nu) \le 2$ yields the stated floor. The residual inequality and the two arithmetic allowances therefore give

$$|\widehat{x}_i - \widehat{c}_i| \le E_i. \tag{14}$$

The shared implementation computes every positive product and sum in E_i upward, including the constant $\gamma_n(2u)$, producing $\widehat{E}_i \ge E_i$. Its finite absolute-sum check also prevents a certificate from assuming a finite reference reduction when the accumulated magnitude may overflow.

Temperature division. The reference compiles to `div.full.f32`, whose documented error is at most two ulps (NVIDIA, 2026). Moving a rounded endpoint by one neighboring value alone is therefore insufficient. First form

$$a = \mathrm{sub}_{\downarrow}(\widehat{c}_i, \widehat{E}_i), \quad b = \mathrm{add}_{\uparrow}(\widehat{c}_i, \widehat{E}_i), \quad l = \mathrm{div}_{\downarrow}(a, T), \quad r = \mathrm{div}_{\uparrow}(b, T).$$

For $q = \max(|l|, |r|)$, evaluate $d = 4uq + 2\eta$ upward. Since $\mathrm{ulp}(x) \le 2u|x| + \eta$, the interval

$$[\mathrm{sub}_{\downarrow}(l, d), \mathrm{add}_{\uparrow}(r, d)] \tag{15}$$

contains the actual reference division for every dot product in $[a, b]$. This argument does not require the approximate division itself to be monotone. `query_bounds.py` implements Eqs. (14)–(15) with explicit directed PTX instructions without `.ftz`. In the recorded Triton build, the corresponding libdevice FP32 intrinsics acquired `.ftz`, which discards subnormal inputs or outputs; the explicit instructions preserve the needed rounding contract.

Noise, ties, and recovery. Both endpoints and the reference use the same finite-precision Gumbel value, indexed by the same key, row, and batch offset. Applying the same round-to-nearest addition is monotone and preserves enclosure. Strict exclusion retains all possible tied winners; exact reduction chooses the smallest token ID among them. These conditions connect the numerical enclosure to Proposition 1. First-stage numerical failure and capacity overflow use the same-key full scan. Invalid refinements retain the old candidate instead of excluding it.

Executable checks. `tests/test_interval_enclosure.py` checks the full two-ulp division envelope against exact rational arithmetic, including binade boundaries, intervals crossing zero, subnormal temperatures, and subnormal outputs. It separately calls the actual exact-row kernel to check every row of high-bit, low-bit, ordinary INT8, and matched INT8 intervals at B1/B4 under cancellation, residual alignment, and nonuniform metrics. `verify_numerical_contract.py` records compiled instructions and checks that the reference functions are unchanged from the original study. Compiler inspection and these tests complement the error analysis; they do not turn the block-FMA assumption into a hardware-independent theorem.

An underflow regression additionally uses two constant BF16 rows $w_0 = 2^{-60}$ and $w_1 = -2^{-60}$, $D = 2048$, hidden coordinates $h_j = 2^{-100}$, and temperature 2^{-149}. Individual reference products underflow to zero, while the coarse path can retain a nonzero subnormal by accumulating before row scaling. An earlier unguarded error budget returned token 0 for key 1, whereas the full reference returned token 1. The current implementation recovers the same-key reference. Tests also exercise normal/underflow mixtures and recovery across CUDA graph replays.