Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Tamarin (HSQ): Reversible Hierarchical KV-Cache Compression for LLM Inference

Alikhan Bazakov

info@siamang-team.org

Siamang Labs

Kirill Kiselev

Siamang Labs

Method Article

Keywords: KV-cache compression, large language model inference, reversible quantization, hierarchical attention routing, equivalence testing, long-context retrieval

Posted Date: July 10th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10297225/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare potential competing interests as follows: Tamarin is developed by Siamang Labs LLC and released under Apache 2.0. Alikhan Bazakov and Kirill Kiselev are officers of Siamang Labs LLC; this affiliation and development role are the relevant competing interests for this work.

Tamarin (HSQ): Reversible Hierarchical KV-Cache Compression for LLM Inference

Alikhan Bazakov, Chief Technology Officer
Kirill Kiselev, Chief Executive Officer
Siamang Labs LLC
Correspondence: info@siamang-team.org

v0.1 preprint, July 2026

Technical name. Tamarin is the public name of the technology; HSQ (Hierarchical Semantic Quantization) is the technical name used in the codebase and in the method sections of this paper.
Code. Beta HSQ package under Apache 2.0: `https://github.com/Siamang-Labs/tamarin`.

Abstract

The key–value (KV) cache is the dominant GPU-memory consumer in long-context LLM inference. At 32K tokens, a Qwen3-8B model holds roughly 4.5 GiB of fp16 KV cache against about 5 GiB of int4 weights, and the cache grows linearly with context. Existing remedies either quantize every cache entry, which is safe but bounded, or evict/merge tokens permanently, which can destroy information that becomes relevant only after the future query is known. We present Tamarin (HSQ), a three-tier reversible compression scheme: a small learned reader routes tokens into a non-summarized 4-bit focal tier (L1), a 3-bit tier of learned group-summary vectors (L2), and a CPU-RAM archive of all compressed originals stored at 4-bit precision (L3). L2 vectors are routing keys, not content replacements: at decode time a per-head router scores L2 groups, fetches selected originals from L3, and attends to the restored tokens before the softmax. Compression is therefore an index, not a deletion.

On the Qwen3 family (0.6B–14B, fp16/int8/int4 weights, 2K–32K contexts), Tamarin reduces the GPU-resident KV cache by 12–28× at a perplexity overhead of 0.1–0.4% on 4B–14B models, token-level agreement of 96–97% with the uncompressed baseline, and binary needle retrieval statistically equivalent to baseline under TOST at $n = 500$ per depth cell. At 32K, Qwen3-8B-int4 attains 500/500 retrieval parity with zero discordant pairs. Unlike eviction-family methods, the measured quality cost decreases with model size, and 4-bit weight quantization composes with Tamarin at no measurable cost on models of at least 4B parameters. We report a paired, per-cell statistical methodology with TOST equivalence for binary retrieval and one-sided non-inferiority margins for perplexity, plus replication runs, the measured applicability boundary, and failure modes including one quantization-fragile checkpoint and one model-family effect.

Keywords: KV-cache compression; large language model inference; reversible quantization; hierarchical attention routing; equivalence testing; long-context retrieval.

1 Introduction

1.1 The problem

Autoregressive transformer decoding attends over a per-layer, per-head cache of past keys and values. For batch size B, n_{layers} layers, n_{kv} KV heads, head dimension d_{head}, context length L, and dtype width b_{dtype} bytes, the fp cache size is

$$\text{KV bytes} = 2\, B\, n_{\text{layers}}\, n_{\text{kv}}\, d_{\text{head}}\, L\, b_{\text{dtype}}. \tag{1}$$

Unless stated otherwise, all memory figures use binary units (MiB/GiB) and report $B = 1$. For Qwen3-8B (36 layers, 8 KV heads, $d_{\text{head}} = 128$, fp16), this is approximately 144 MiB per 1K tokens, 4.5 GiB at 32K, and 18 GiB at 128K. Model weights can be quantized once; the KV cache grows with every token. In the long-context regime, the cache rather than the weights often determines whether a workload fits on a GPU.

1.2 The dichotomy in prior work

Existing approaches fall into two families with a shared blind spot. *Cache quantization* compresses all entries to 2–4 bits (Liu et al., 2024; Hooper et al., 2024). It is effective and relatively safe, but it is bounded: 4-bit storage gives about 4× over fp16, and 2-bit storage can degrade quality. *Eviction and merging* methods, including StreamingLLM, H2O, SnapKV, TOVA, Scissorhands, and CaM, drop or fuse tokens judged unimportant (Xiao et al., 2024; Zhang et al., 2023; Li et al., 2024; Oren et al., 2024; Liu et al., 2023; Zhang et al., 2024). Their ratios can be high, but the decision is final: a token evicted at prefill cannot be recovered at decode step 900 when the user finally asks about it. Retrieval-style workloads expose this irreversibility directly.

The blind spot is that token importance is a function of the future query, which an eviction method must predict at compression time.

1.3 Our approach: compress as an index, not as a deletion

Tamarin keeps three synchronized representations of the context: non-summarized quantized entries for a small important set (L1), learned two-vector summaries per group of 8–16 tokens (L2), and the 4-bit-precision originals of every compressed token in CPU RAM (L3). During decoding, an attention router scores L2 summaries per KV head and pulls the originals of selected groups back into the attention window before the softmax. The L2 tier therefore does not need to reconstruct content; it needs to rank groups well enough that the right originals return.

This is the central falsifiable bet. Disabling the return path (routing budget $\beta = 0$) collapses needle retrieval to 0/100 and inflates perplexity by 2.59× (Section 6.8). The summaries carry routing structure; content lives in L3. Throughout, *reversible* means that every compressed token remains addressable and can be returned to the attention window; it does not mean bit-exact numerical reconstruction, because L3 stores originals at 4-bit precision.

1.4 Contributions

- **Architecture.** A three-tier reversible KV hierarchy with a learned token-importance reader, a per-layer cross-attention group compressor whose outputs double as routing keys, randomized scalar quantization at 4/3/4 bits (near-optimal in measured MSE relative to the rate–distortion baseline; Section 3.5), and a per-head two-pass routing attention integrated into HuggingFace generation without forking model code.
- **Validation methodology.** A paired statistical program with $n = 500$ per cell, exact McNemar tests, bootstrap confidence intervals, TOST equivalence for binary retrieval, one-sided non-inferiority margins for perplexity, Holm correction, and a retention metric for degraded baselines. Two independent replications agree to 2–3 significant figures.
- **Findings.** Tamarin quality improves with model size; weight quantization composes freely at $\geq$4B; prefill overhead amortizes at 32K; and the method has explicit, measured failure modes rather than hidden averaged failures.
- **Open-source package.** The public repository ships the beta HSQ package under Apache 2.0, with the single `HSQ` facade and model adapters described in the README. Verification scripts, evaluation harnesses, and raw per-trial materials are maintained separately and can be provided to reviewers or researchers upon reasonable request.

2 Related Work

KV quantization. KIVI and successors quantize keys and values to 2–4 bits, often with asymmetric layouts, per-channel schemes, or outlier handling (Liu et al., 2024; Hooper et al., 2024). Tamarin is orthogonal: its L1 and L3 tiers are themselves quantized stores; the additional compression comes from L2 summarization and reversible offload.

Token eviction. StreamingLLM keeps attention sinks plus a recent window (Xiao et al., 2024); H2O keeps accumulated-attention "heavy hitters" (Zhang et al., 2023); SnapKV selects from prompt-window attention at prefill (Li et al., 2024); TOVA and Scissorhands maintain a fixed online cache budget (Oren et al., 2024; Liu et al., 2023). These methods are irreversible. Tamarin borrows the sink/recent insight for its L1 floor and departs on the storage semantics.

Cache merging. CaM and KVMerger fuse to-be-evicted entries into kept entries (Zhang et al., 2024; Wang et al., 2024). Merging is softer than deletion, but still lossy and final.

Offloading and paged caches. vLLM's PagedAttention manages full KV in pages (Kwon et al., 2023); InfiniGen and related systems offload full KV to CPU and prefetch by attention prediction (Lee et al., 2024). These systems move the memory problem rather than compressing it. Tamarin uses a byte-aligned 4-bit-precision CPU archive (about 2× smaller than fp16 in RAM) and a GPU-resident learned index.

Learned importance. Learned routers and classifiers appear in token pruning and sparse attention. Tamarin uses a deliberately small reader; robustness comes from reversibility, not from perfect reader accuracy.

Table 1: Qualitative positioning of Tamarin (HSQ) against prior KV-cache families.

Method family	Ratio	Reversible	Learned components	Full content kept
KV quantization	$\sim 4\times$	Lossy in place	Some	Yes, lossy
Eviction (H2O/SnapKV)	Unbounded	No	Some	No
Merging (CaM)	Unbounded	No	Some	Partially
Offload/paged cache	GPU-shifted	Yes	Some	Yes, full size
Tamarin (HSQ)	12–28× GPU	Yes, per group	Reader + compressor	Yes, 4-bit-precision CPU archive

3 Method

3.1 Notation and overview

Consider one attention layer with n_{kv} KV heads of dimension d. After prefilling a length-L prompt, the baseline cache holds $K, V \in \mathbb{R}^{n_{\text{kv}} \times L \times d}$ per layer. Tamarin replaces this with three synchronized stores:

- **L1, non-summarized focal tier:** a token set S_1 containing sinks, an optional recent window, and tokens the reader marks important. Entries are stored as 4-bit quantized vectors at their true semantic positions.
- **L2, summary tier:** remaining tokens are partitioned into groups of size $N \in \{8, 16\}$ by importance band. Each group is compressed into $K_{\text{out}} = 2$ representative $(\hat{k}, \hat{v})$ pairs, stored at 3 bits with metadata: original positions and an L3 pointer.
- **L3, archive:** the original (k, v) of every compressed token, stored in CPU RAM at 4-bit precision.

Decoding runs the model over a short cache ordered as $[\text{L2}_{\text{low}}, \text{L2}_{\text{med}}, \text{L1}]$ at semantic positions; a registered attention implementation performs routing and prefetch.

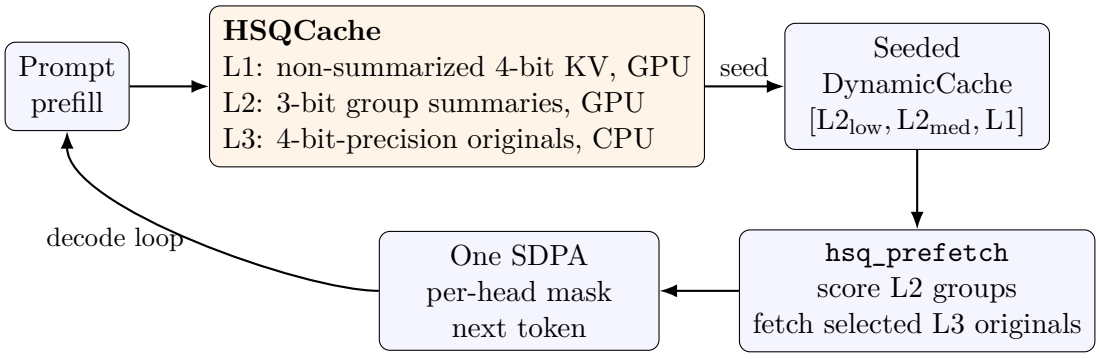

Figure 1: Tamarin treats compression as a query-time index. L2 summaries route attention; selected originals are restored from L3 before the final attention pass.

3.2 Memory accounting

Units and conventions: all memory figures in this paper use binary units (MiB/GiB), and Equation 1 reports $B = 1$ unless stated otherwise. Storage cost separates *precision* from *layout*. A d-dimensional vector quantized at b bits carries db bits of index payload plus one fp32 norm, but only the GPU tiers store those indices packed:

$$\text{fp16 baseline} : 2 \cdot 16d \text{ bits per token per KV pair}, \tag{2}$$

$$\text{L1 (GPU, } b = 4, \text{ packed)} : 2 \cdot (4d + 32) \text{ bits per token}, \tag{3}$$

$$\text{L2 (GPU, } b = 3, \text{ packed)} : \frac{2K_{\text{out}}(3d + 32) + 64}{N} \text{ bits per source token}, \tag{4}$$

$$\text{L3 (CPU, } b = 4 \text{ precision, byte aligned)} : 2 \cdot (8d + 32) \text{ bits per token}. \tag{5}$$

The L2 equation counts the packed KV payload plus one 64-bit group pointer. Per-group position arrays (N original positions plus K_{out} emitted positions, stored as int32) and Python object overhead are excluded from the KVx accounting throughout the paper; they are small relative to the reported GPU totals.

L3 deliberately stores its 4-bit-precision indices one per uint8 coordinate rather than packed. Prefetch (Section 3.6) gathers rows by token position, and byte alignment makes each token row directly addressable by offset, avoiding sub-byte realignment on the hot path. The CPU archive is therefore about 2× smaller than the fp16 cache it replaces, not the $\sim 3.8\times$ that fully packed 4-bit storage would give. Packing L3 is a roadmap item that would halve CPU RAM again.

For $d = 128$, mixed $N = 8/16$ grouping, and the default reader operating point, the GPU-resident cache is about 23× smaller than fp16 at 8K. The ratio increases with context because the fixed L1 component amortizes. The measured Qwen3-8B footprint is consistent with the formulas above; for example, L3 at 8K is $2(8 \cdot 128 + 32)$ bits × 8 heads × 36 layers × $\sim$ 7.9K archived tokens, or about 585 MiB.

Table 2: Measured KV footprint for Qwen3-8B under Tamarin versus an fp16 baseline.

Context	Full fp16 KV	Tamarin GPU	Tamarin CPU (L3)	KVx
1K	144 MiB	11.5 MiB	73 MiB	12.6×
4K	576 MiB	29.5 MiB	293 MiB	19.5×
8K	1152 MiB	50.5 MiB	585 MiB	22.8×
32K	4.5 GiB	172 MiB	2.29 GiB	26.8×

A more aggressive preset ($N = 16/32$) reaches 36× at unchanged measured quality.

3.3 Reader: learned token importance

The reader is a small MLP over hidden states from an early decoder layer (default hook layer 3), which are already computed during prefill:

$$p_i = \sigma\Big(\text{MLP}(h_i^{(3)})\Big) \in [0, 1]. \tag{6}$$

With thresholds $(\tau_{\text{hi}}, \tau_{\text{lo}}) = (0.95, 0.5)$, tokens with $p_i \geq \tau_{\text{hi}}$ go to L1, tokens with $\tau_{\text{lo}} \leq p_i < \tau_{\text{hi}}$ go to medium groups ($N = 8$), and tokens with $p_i < \tau_{\text{lo}}$ go to low groups ($N = 16$). Attention sinks and an optional recent window always go to L1.

The reader is trained in minutes by distilling an attention-derived importance signal from generic text. Crucially, mistakes are recoverable: a misclassified token lands in L2/L3 and remains reachable by routing. The reader shapes cost, not the feasibility of recall.

3.4 Compressor: group summaries as routing keys

Each group $(K_g, V_g) \in \mathbb{R}^{N \times d}$ is compressed per layer by a Perceiver-style cross-attention pooler (Jaegle et al., 2021). Two learned queries attend over the group and emit $(\hat{k}_j, \hat{v}_j)$ for $j \in \{1, 2\}$. Groups are formed over a tier's token subsequence rather than contiguous spans. This "across gaps" rule increased measured compression from 5.7× to 21.0× at unchanged quality by avoiding fragmentation at tier boundaries.

Training distills full attention from teacher queries q: the summary attention is matched to full-group attention, and a small token-recovery auxiliary is added. The base model is not fine-tuned. At the default operating point, the compressor is not asked to preserve content; it is asked to produce 3-bit keys that rank well under the model's query projections.

3.5 Quantization: TurboQuant

All tiers use randomized scalar quantization. A fixed random orthonormal rotation Π decorrelates coordinates; each coordinate is quantized by a b-bit Lloyd–Max scalar quantizer with centroids fit to the unit Gaussian; and the vector norm is stored separately:

$$x \mapsto \left(Q_b(\Pi x / \max(\|x\|_2, \varepsilon)), \|x\|_2\right), \qquad \text{packed storage} = db + 32 \text{ bits}. \tag{7}$$

The ε guard covers the measure-zero all-zero-vector case; as described in Section 3.2, L3 stores its indices byte-aligned rather than packed. Rotation makes coordinates approximately Gaussian across layers and heads, so one centroid table serves all stores; measured MSE is within 2.7× of the rate–distortion bound at $b = 3$–4. Nearest-centroid assignment uses bucketization over centroid midpoints, and dequantization happens on GPU after a uint8+norm transfer. For the packed GPU tiers this payload is up to $\sim 4\times$ smaller than fp16; for L3 prefetch, the byte-aligned 4-bit-precision indices give a $\sim 2\times$ smaller PCIe payload (Section 3.2). L1 and L3 use 4-bit precision because they carry content; L2 uses 3 bits because it carries routing keys. The implementation additionally protects the first and last four layers with one extra bit, reflecting their measured sensitivity.

3.6 Decode: per-head routing and reversible prefetch

The compact cache seeds the model's own DynamicCache. A registered attention implementation, `hsq_prefetch`, replaces only the attention score/softmax stage. Per decode step and KV head h:

1. **Routing pass.** Scores between query q_h and L2 summary keys produce per-group relevance. Let G denote the number of L2 groups and define $K_{\text{route}} = \lceil \beta G \rceil$. Each KV head independently

selects its top K_{route} groups, with default budget $\beta = 0.5$. Here β is a per-head routing budget, not a global fetch fraction: the fetched set is the cross-head union, capped during decode by a pool limit, so the realized fraction of groups fetched per step can exceed β when heads disagree.

2. **Fetch.** The union of selected groups is fetched from L3 through an LRU group cache on GPU as byte-aligned uint8 indices plus norms, dequantized on GPU, and RoPE-rotated to the groups' original positions (Su et al., 2024).

3. **Attention pass.** One scaled dot-product attention (SDPA) runs over [L1∪L2∪fetched originals] with a per-head additive mask. For heads that fetched a group, its L2 summary is hidden to avoid double counting; for heads that did not fetch it, the summary remains visible.

Re-routing every r steps (default $r = 4$) amortizes the routing pass. Teacher-forced scoring uses the same machinery with per-token top-k unions and a causal block mask.

Positional correctness. Seeded summaries and fetched originals live at semantic positions. Explicit position IDs keep query and key rotations in the same coordinate system; otherwise RoPE-dependent circuits can silently break. An early implementation bug of this kind cost 12× perplexity while remaining invisible on toy models, making position-ID tests a reproduction requirement for seeded-cache methods.

3.7 Implementation

Tamarin is implemented in roughly 9K lines of Python on top of HuggingFace Transformers (Wolf et al., 2020); it uses no custom CUDA and does not modify model weights. Key engineering points are validated by tests: streaming ingest compresses each layer during the prefill forward and frees raw K/V immediately; quantization is batched per layer and tier; L1/L2 are offloaded after seeding; `logits_to_keep=1` avoids storing full prefill logits; batch > 1 native decode is handled by left-padded per-sequence prefixes; and hybrid attention/state-space models keep native recurrent state while HSQ splices compressed KV into attention layers.

An adapter layer isolates model-family differences: q/k normalization, RoPE variants, fused QKV projections, NoPE layers, MoE-only layers, and hybrid attention/SSM blocks. This is important empirically: Qwen3's per-head key normalization standardizes the quantity ranked by the router and is a leading explanation for the family effect described in Section 7.5.

```
from hsq import HSQ

h = HSQ(model, tokenizer)
h.fit(dataset="english", output_dir="w")
h = HSQ.from_pretrained(model, tokenizer, "w")
out = h.generate(prompt, max_new_tokens=256)
```

4 Experimental Setup

Models. We evaluate Qwen3-0.6B, 4B, 8B, and 14B (Yang et al., 2025); weight formats include fp16, bitsandbytes int8 (Dettmers et al., 2022), and int4 nf4 with double quantization and fp16

compute (Dettmers et al., 2023). One additional non-Qwen family, Mistral-7B-v0.3, is included for failure analysis (Jiang et al., 2023).

Fitting. The reader and compressor are fitted per model for 200 steps on WikiText-2 (Merity et al., 2017), taking minutes on an A100. Base-model weights are never fine-tuned. Unless stated otherwise, the configuration is $\beta = 0.5$, group sizes 8/16 with across-gaps grouping, and bit allocation 4/3/4.

Benchmarks. (i) Needle-in-a-haystack (NIAH): a random five-digit code is planted at relative depth $d \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ inside shuffled neutral filler; a 16-token greedy answer is a hit if it contains the code verbatim. (ii) Multi-key retrieval, RULER-style (Hsieh et al., 2024): eight look-alike needles are planted at jittered depth slots; the query asks for one key, and we track both hits and confusions. (iii) Perplexity on English (WikiText-2), code (codeparrot-clean-valid), and multilingual C4 samples (Raffel et al., 2020). (iv) Token-level agreement: greedy argmax match and full-vocabulary KL(baseline$\|Tamarin$). (v) Memory and speed via phase-separated prefill/decode benchmarks.

Design. Every trial specification is run through both systems, yielding paired outcomes. Power is allocated per cell: $n = 500$ per NIAH depth, $n = 500$ per multi-key slot, and $n = 500$ paired chunks per perplexity domain; 32K contexts use $n = 100$ per depth. Six batteries share the design: 0.6B-fp16, 0.6B-int4, 4B-fp16, 4B-int4, 8B-int4, and 14B-int4. Each baseline is the same-weight model without Tamarin, isolating cache-compression cost from weight-quantization cost.

Hardware and provenance. Runs use NVIDIA A100-40GB Colab instances and Transformers 5.x. Speed comparisons are made only within a run because Colab hardware varies; paired quality metrics are hardware-independent. Raw per-trial outcomes are checkpointed to JSON and archived by the authors; verification code and raw materials can be provided upon reasonable request.

5 Statistical Methodology

We treat "no quality loss" as an equivalence claim requiring affirmative evidence, not as a failure to reject a null. The procedures are implemented dependency-free in `hsq/eval/stats.py` and unit-tested against reference values.

Paired binary outcomes. For each NIAH or multi-key cell, we report the hit rate with a Wilson 95% confidence interval (CI), paired contingency counts $(n_{11}, n_{10}, n_{01}, n_{00})$, and the exact binomial McNemar test on discordant pairs $b = n_{10}$ and $c = n_{01}$:

$$p = \min\left(1,\ 2 \sum_{k=\max(b,c)}^{b+c} \binom{b+c}{k} 2^{-(b+c)}\right). \tag{8}$$

We also bootstrap the rate difference by multinomial resampling of the four pair categories. Reporting uses 95% CIs; TOST uses 90% CIs. A binary cell is equivalent iff the 90% CI of the paired difference lies within ± 5 percentage points. We also report retention, $n_{11}/(n_{11} + n_{10})$, the share of baseline successes preserved by Tamarin.

Paired continuous outcomes. Perplexity uses the mean paired ΔNLL on identical tokens,

$$\Delta\text{NLL} = \text{NLL}_{\text{Tamarin}} - \text{NLL}_{\text{baseline}}, \tag{9}$$

with a percentile-bootstrap CI and a sign-flip permutation test; a paired t-test is used as a cross-check. The perplexity ratio is exp(mean ΔNLL). For NLL/PPL we use a one-sided non-inferiority criterion: non-inferior iff the 90% upper confidence bound of the ratio is below 1.10×. We reserve *equivalence* and TOST terminology for the two-sided binary retrieval margins.

Multiplicity and power. All p-values in a run form one family and are adjusted by Holm. Significance and equivalence can coexist: at $n = 500$, tiny real differences are detectable, so the declared margins carry the practical claim. At $p \approx 0.97$, $n = 500$ gives a Wilson CI of roughly ± 1.7 percentage points; paired chunks resolve 0.1–0.3% NLL shifts. Two batteries were re-run in independent sessions with re-fitted adapters, and continuous metrics replicate to 2–3 significant figures.

6 Results

6.1 Headline: the uniform int4 series

The target deployment configuration – int4-nf4 weights plus Tamarin cache – is evaluated across the model-size axis, with each row paired against its own int4 baseline.

Table 3: Headline results across the Qwen3 size axis with int4 weights and $n = 500$ per cell. Asterisks mark the quantization-fragile 14B checkpoint discussed in Section 7.3; the dagger marks the program's single non-inferiority failure.

Metric ($\beta = 0.5$)	0.6B-int4	4B-int4	8B-int4	14B-int4
NIAH @2K	98.5% (worst 96.8)	98.8% (97.6)	99.7% (99.0)	92.2%*
NIAH @8K	100.0%	100.0%	99.9%	93.2%*
Multi-key @2K/@8K	95.3/95.3%	99.7/100%	97.8/100%	99.7/100%
Confusion @2K (vs. base)	3.6% vs. 1.7%	0.15% vs. 0.05%	0.15% vs. 0.03%	N/A
PPL English/code/multilingual	1.083/1.061/1.099†	1.0013/1.0018/0.9972	1.0038/1.0028/1.0037	1.0038/1.0017/1.0026
Greedy match/KL	86.9%/0.0849	95.9%/0.0101	96.7%/0.0066	97.1%/0.0069
KVx @8K	23.4	22.7	22.8	∼23
Decode vs. baseline	47–62%	57–63%	72–73%	N/A

On the 4B and 8B models, all per-cell retrieval verdicts are TOST-equivalent: there are no retrieval holes. The 8B is the cleanest model in the program, with worst NIAH cell 99.0%. On 4B, multilingual perplexity is measured below baseline (0.9972, Holm-adjusted $p = 0.0004$), indicating that the cache-compression cost is below the measurement floor at this scale.

6.2 fp16 batteries

Qwen3-0.6B-fp16 is the worst-case fp16 model: NIAH is 98.4% [97.9, 98.9] at 2K and 99.5% [99.2, 99.7] at 8K; multi-key is 97.6% [97.0, 98.0] at 2K and 97.3% [96.7, 97.7] at 8K, with all slot cells TOST-equivalent; PPL ratios are 1.039/1.037/1.060; agreement is 88.2%, with KL 0.066. An earlier $n = 250$ multi-key run could not show equivalence; at $n = 500$ per slot this was revealed as under-power rather than a real gap.

Qwen3-4B-fp16 is effectively at parity: NIAH is 99.4% [99.1, 99.7] at 2K and 99.9% at 8K;

multi-key is 99.7% at 2K and a perfect 4000/4000 at 8K with zero confusions; PPL ratios are 1.0038/1.0052/1.0050; agreement is 96.3%, with KL 0.0083; decode throughput is 40–55% of baseline (26.3 versus 66.8 tok/s at 8K, 39.7 versus 72.7 tok/s at 1K). The fp16 baselines decode faster than the int4 baselines, so the same Tamarin overhead is a larger fraction of fp16 baseline throughput than in Table 3.

6.3 Replication

Table 4: Independent replication of two batteries in separate Colab sessions with re-fitted adapters.

Quantity	Run A	Run B
0.6B-fp16 PPL English/code/multilingual	1.038/1.034/1.062 ($n = 50$)	1.039/1.037/1.060 ($n = 500$)
0.6B-fp16 match/KL	87.4%/0.069	88.2%/0.066
4B-fp16 PPL English/code/multilingual	1.0038/1.0054/1.0050	1.0038/1.0052/1.0050
4B-fp16 match/KL	96.4%/0.0083	96.3%/0.0083

Continuous metrics replicate to 2–3 significant figures. NIAH rates vary within about 1.5 percentage points across adapter refits, which is why retrieval claims are stated with confidence intervals rather than point values alone.

6.4 Scaling law

Every quality axis improves with model size, with the 14B KL matching the 8B within noise.

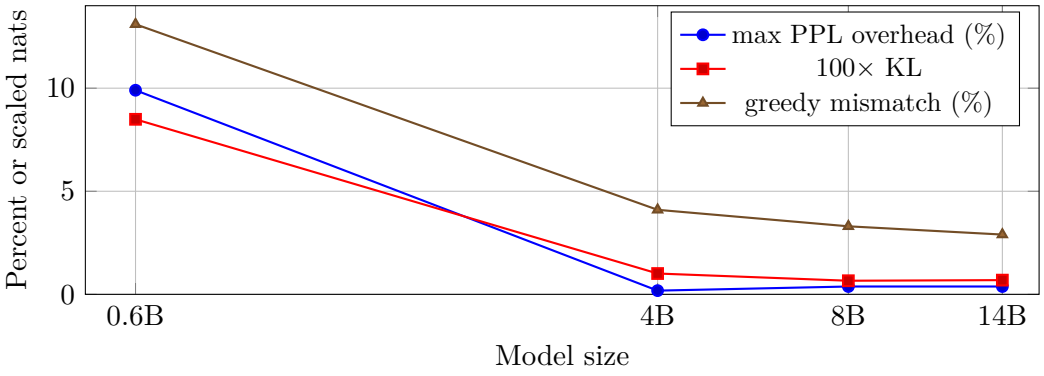

Figure 2: The measured Tamarin cost decreases sharply from 0.6B to 4B and remains near the measurement floor through 14B. PPL overhead is the maximum domain ratio minus one; KL is scaled by 100 for display.

Table 5: Quality axes versus model size for int4 weights.

Axis	0.6B	4B	8B	14B
PPL ratio range	1.061–1.099	0.997–1.002	1.003–1.004	1.002–1.004
Greedy match	86.9%	95.9%	96.7%	97.1%
KL, nats	0.085	0.0101	0.0066	0.0069
Worst NIAH cell @2K	96.8%	97.6%	99.0%	See Section 7.3

To our knowledge, eviction-family methods do not report this direction of scaling: Tamarin becomes easier to use as model capacity increases.

6.5 Weight quantization: composition, compounding, boundary

For models of at least 4B parameters, weight quantization composes freely with Tamarin. Full batteries show int4 costs indistinguishable from fp16: 4B has 0.997–1.002 int4 versus 1.004–1.005 fp16, and 8B/14B remain in the 1.002–1.004 range. An int8 run on 0.6B matches fp16 (1.040 versus 1.038).

The 0.6B model exposes noise compounding. Tamarin cost roughly doubles under int4 weights, from 1.037–1.060 in fp16 to 1.061–1.099 in int4. A refit ablation attributes the surcharge to model margin rather than adapter mismatch: re-training the reader and compressor on the quantized model reproduces the transferred-weights ratio (1.090 versus 1.092). The measured applicability boundary is 0.6B-int4 multilingual at 1.0986 [1.0926, 1.1051], the program's single non-inferiority failure under the 1.10 margin.

6.6 Long context: 32K

Table 6: Long-context 32K results on 8B and 14B int4 checkpoints.

Metric	8B-int4 @32K	14B-int4 @32K
KVx	26.8× (4.5 GiB → 172 MiB)	27.9× (5.0 GiB → 184 MiB)
Prefill vs. baseline	1.27×	1.19×
NIAH ($n = 100$/depth)	500/500 both systems; zero discordant pairs	94.6% vs. base 95.4% ($p = 0.34$; TOST-equivalent; retention 98.5%)
Decode vs. baseline	17%	18%
PPL quick pass	N/A	1.00×

Three phenomena are visible. First, prefill overhead amortizes: 2–3.4× at 2K–8K shrinks to 1.2–1.3× at 32K as baseline attention dominates. Second, retrieval at 32K is at perfect parity on the healthy 8B: the router selects the needle's group out of roughly 2700 candidates from 3-bit keys 500 times out of 500. Third, decode at 32K is the measured bottleneck at about 17–18% of baseline throughput.

6.7 Speed and peak memory

Within-run A100 measurements show prefill overhead decreasing with context and decode throughput at $\beta = 0.5$ running at 40–73% of baseline at contexts up to 8K. At 32K, decode falls to about 17–18% because routing is approximately linear in the number of groups and because a large fraction of originals may be prefetched. Prefill-phase peak VRAM is below baseline due to streaming ingest; end-to-end peak is approximately baseline because weights and transients dominate. Tamarin's present value proposition is capacity – contexts and batch slots that otherwise do not fit – not latency.

6.8 Ablations

- **No L3 return path.** With routing budget $\beta = 0$, summaries alone yield NIAH 0/100 and PPL 2.59× [2.36, 2.86]. This is the key ablation: L2 summaries are routing keys; content lives in L3.
- **Budget.** $\beta = 0.3$ loses 10–17% of needles because the top-30% cutoff is too strict. $\beta = 0.5$ reaches statistical parity and decodes faster by stabilizing cross-step group-cache hits.
- **Routing freshness.** Wider candidate pools and per-step re-routing do not improve quality at a 35% wall-clock cost; the budget, not freshness, is the binding constraint.
- **Across-gaps grouping.** Grouping over tier subsequences raises KVx from 5.7 to 21.0 at unchanged quality. Fragmentation, not quantization, was the original compression ceiling.
- **Refit on quantized models.** Refit reproduces transferred-weights cost bit-for-bit, indicating that fp16-fitted adapters transfer cleanly and that small-model int4 degradation is intrinsic noise compounding.

7 Analysis and Discussion

7.1 Why an index is the right abstraction

Eviction methods must predict at compression time what a future query will need. Tamarin moves that decision to query time: the router sees the actual query. The empirical signature is the error profile: no depth cell of a healthy Qwen3 model falls below 96.8% while average PPL cost remains at 0.1–0.4% for 4B–14B. The $\beta = 0$ ablation in Section 6.8 is the converse proof: remove the query-time return path and the same stored bits lose retrieval entirely.

7.2 Why quality improves with model size

Two mechanisms compound. First, larger models spread information over more heads and channels, so a two-vector group summary discards a smaller fraction of the effective rank of K_g, and the router's ranking problem becomes easier. Second, quantization perturbations become losses only when they cross decision boundaries; larger models have more margin. From 0.6B to 8B, KL falls about 13× (0.085 to 0.0066) while greedy match rises by roughly 10 percentage points, consistent with a margin mechanism.

7.3 Failure anatomy I: a quantization-fragile checkpoint

Qwen3-14B-int4 dissociates: it has best-in-series PPL ratios (1.0017–1.0038), agreement (97.1%, KL 0.0069), and multi-key retrieval (99.7–100%), yet two narrow single-needle holes: depth 0.9 at 2K reaches 80.4%, and depth 0.1 at 8K is down 7 percentage points. The baseline itself is degraded by nf4 at these cells (98.4% at 2K, 95.6% at 8K, with a lost-in-the-middle dip to 91.8%).

A 2×2 attribution design (model size by weight format) resolves the confound: the same nf4 pipeline yields zero holes on 4B and 8B, and the corresponding 4B-fp16 cell is 98.6%. The holes therefore belong to this checkpoint's quantization fragility, not to int4 or scale generically. At 32K both holes close. Retention quantifies the honest cost: 96.9% at 8K, 98.5% at 32K, and 83.4% in the worst cell. Mitigations are int8/fp16 weights or a non-zero recent window that places near-query content in L1.

7.4 Failure anatomy II: the noise-compounding law

Tamarin's cost is not additive with weight-quantization cost; it is conditional on remaining model margin. On 0.6B, fp16 ratios of 1.037–1.060 become 1.061–1.099 under int4. On 4B, ratios of 1.004–1.005 become 0.997–1.002. With the refit ablation excluding trainable-component mismatch, the operative model is multiplicative noise interaction: KV noise that a margin-rich model absorbs silently can tip already-perturbed logits over decision boundaries in a margin-poor one.

7.5 Failure anatomy III: a model-family effect

Mistral-7B-v0.3-int4, the first non-Qwen family measured, shows an inverted pattern: a perfect baseline with a Tamarin-side gap concentrated at short context and mid depths (NIAH 88.4% at 2K with a bottom of -20 percentage points at depth 0.5; 97.4% at 8K; multi-key 95.2% versus 99.6%). The same harness is clean on Qwen3, so this is a family effect.

The leading hypothesis is absence of per-head q/k normalization. Heterogeneous key norms survive compression and 3-bit quantization, making ranking by compressed keys noisier; Qwen3's key normalization standardizes exactly what the router ranks. Competing hypotheses are Qwen-tuned defaults and an int4 interaction. Until resolved, the statistically validated claim scope is Qwen3. If the normalization hypothesis is confirmed, routing-key normalization before compression/quantization is a likely portable fix for norm-free families.

7.6 What we claim and what we do not

For healthy Qwen3 4B and 8B checkpoints across fp16/int8/int4 formats and 2K–32K contexts, and for the 14B checkpoint on perplexity, token agreement, and multi-key retrieval, we claim 12–28$\times$ GPU KV reduction at 0.1–0.4% perplexity overhead, 96–97% token agreement, free composition with 4-bit weights, and prefill overhead that amortizes with context. The two narrow 14B-int4 single-needle cells are explicitly excluded from the retrieval-equivalence claim and are quantified in Section 7.3 by retention of 83.4–98.5%.

We do not claim downstream task-suite dominance, latency wins, or numerical comparisons against re-implemented eviction baselines. Our evaluation targets retrieval and language modeling under

compression; decode throughput is currently a cost; and Table 1 compares architectures qualitatively pending standardized head-to-heads.

8 Limitations

- **Family scope.** The method is statistically validated on Qwen3 0.6B–14B. Llama and Mistral adapters pass functional tests, but Mistral-7B shows the family effect in Section 7.5; non-Qwen families should be treated as experimental for retrieval-critical use.
- **Decode throughput.** Throughput is 40–73% of baseline at contexts up to 8K and 17–18% at 32K. Tamarin buys capacity, not speed.
- **Small models.** 0.6B-class models pay 4–6% PPL in fp16 and 6–10% in int4; the measured boundary is 0.6B-int4 multilingual at 1.0986 versus the 1.10 non-inferiority margin.
- **Fragile checkpoints.** Locally degraded nf4 baselines can be amplified. Retention audits, int8/fp16 weights, or a recent window mitigate the issue.
- **Context length.** Contexts up to 32K are measured. 128K+ remains extrapolation, although the memory ratio is constant by construction and the 32K distance gradient is clean.
- **Serving state.** Native-path decode is currently batch-1, with batched prefill supported. Continuous batching and multi-tenant serving are future work.
- **CPU RAM and bandwidth.** L3 consumes 0.6–2.6 GiB for 8K–32K on 8B–14B and requires transfer bandwidth on prefetch misses.
- **Per-model fitting.** Each model needs a short fit; zero-shot adapter transfer across checkpoints is untested.

9 Future Work and Roadmap

Engineering. Packed 4-bit prefetch caches should keep the decode working set below baseline at all contexts. Hierarchical or coarse-to-fine routing should break the linear-in-groups cost that drives the 32K decode gap. Batched decode and GPTQ/AWQ validation are direct engineering extensions.

Method. Routing-key normalization is the likely fix for q/k-norm-free families if the Mistral diagnosis holds. A compressor ranking loss could train summaries to rank directly under model queries; joint reader/compressor training and adaptive per-layer bit allocation are natural next steps.

Family coverage and scale. The adapter layer already covers per-layer kind classification, NoPE layers, MoE MLPs, and hybrid attention/SSM models. The validation program should be re-run family by family – Mistral and Llama first, then Gemma and Phi – with custom routing configurations where required. Larger 32B–70B models and 128K contexts are expected to be favorable regimes.

10 Conclusion

Tamarin reframes KV-cache compression as indexing: summaries route, originals answer, and no token is intentionally deleted. The measured consequences are $\sim$ 20–28$\times$ GPU-cache reduction at low perplexity cost on healthy 4B–14B settings, per-cell retrieval equivalence for healthy Qwen3 4B/8B checkpoints up to 32K, free composition with 4-bit weights at $\geq$4B, and quality that improves with model scale. The same program also surfaces boundaries: small models pay percent-level costs, one fragile checkpoint shows narrow amplified holes, one model family awaits diagnosis, and decode throughput is the price of restored originals.

The evaluation standard is part of the contribution. Paired per-cell power, declared binary-retrieval equivalence margins, declared PPL non-inferiority margins, retention against degraded baselines, and replication make the claims attackable. The public release contains the HSQ package; verification scripts, evaluation harnesses, and raw per-trial materials can be provided upon reasonable request to support independent rechecking.

Code and Data Availability

The public repository contains the beta HSQ package under Apache 2.0, with installation and quickstart documentation for the `HSQ` facade. The statistically validated claims in this paper are for Qwen3 models; adapters for other families should be treated as experimental until their validation batteries are complete. The evaluation harness, verification scripts, and raw per-trial JSON checkpoints are maintained separately from the package release and can be provided to reviewers or researchers upon reasonable request for independent rechecking.

Acknowledgments

This research was self-funded by Siamang Labs LLC, including the compute used for the experiments. Development and manuscript preparation were assisted by AI models, specifically Anthropic Claude Opus 4.7, Anthropic Claude Opus 4.8, and Anthropic Claude Fable 5, which were used for engineering support, statistical-harness design, drafting, editing, and documentation. The authors reviewed all AI-assisted material and take responsibility for the final content, analyses, and claims.

Competing Interests

Tamarin is developed by Siamang Labs LLC and released under Apache 2.0. Alikhan Bazakov and Kirill Kiselev are officers of Siamang Labs LLC; this affiliation and development role are the relevant competing interests for this work.

References

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. (2022). LLM.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. *Advances in Neural Information Processing Systems*.

Hooper, C., Kim, S., Mohammadzadeh, H., Mahoney, M. W., Shao, Y. S., Keutzer, K., and Gholami, A. (2024). KVQuant: Towards 10 million context length LLM inference with KV cache quantization. *arXiv:2401.18079*.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Yang, Z., and Ginsburg, B. (2024). RULER: What's the real context size of your long-context language models? *arXiv:2404.06654*.

Hugging Face (n.d.). codeparrot/codeparrot-clean-valid dataset. *Hugging Face Datasets*.

Jaegle, A., Gimeno, F., Brock, A., Zisserman, A., Vinyals, O., and Carreira, J. (2021). Perceiver: General perception with iterative attention. *Proceedings of the 38th International Conference on Machine Learning*.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., and others (2023). Mistral 7B. *arXiv:2310.06825*.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. (2023). Efficient memory management for large language model serving with PagedAttention. *Proceedings of the 29th Symposium on Operating Systems Principles*, 611–626.

Lee, W., Lee, J., Seo, J., and Sim, J. (2024). InfiniGen: Efficient generative inference of large language models with dynamic KV cache management. *18th USENIX Symposium on Operating Systems Design and Implementation*, 155–172.

Li, Y., Huang, Y., Yang, B., Venkitesh, B., Locatelli, A., Ye, H., Cai, T., Lewis, P., and Chen, D. (2024). SnapKV: LLM knows what you are looking for before generation. *arXiv:2404.14469*.

Liu, Z., Desai, A., Liao, F., Wang, W., Xie, V., Xu, Z., Kyrillidis, A., and Shrivastava, A. (2023). Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. *Advances in Neural Information Processing Systems*.

Liu, Z., Yuan, J., Jin, H., Zhong, S., Xu, Z., Braverman, V., Chen, B., and Hu, X. (2024). KIVI: A tuning-free asymmetric 2-bit quantization for KV cache. *arXiv:2402.02750*.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2017). Pointer sentinel mixture models. *International Conference on Learning Representations*.

Oren, M., Hassid, M., Yarden, N., Adi, Y., and Schwartz, R. (2024). Transformers are multi-state RNNs. *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 18724–18741.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Su, J., Lu, Y., Pan, S., Wen, B., and Liu, Y. (2024). RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Wang, Z., Jin, B., Yu, Z., and Zhang, M. (2024). Adaptive KV Cache Merging for LLMs on Long-Context Tasks. *arXiv:2407.08454*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and others (2020). Transformers: State-of-the-art natural language processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. (2024). Efficient streaming language models with attention sinks. *International Conference on Learning Representations*.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., and others (2025). Qwen3 technical report. *arXiv:2505.09388*.

Zhang, Y., Du, Y., Luo, G., Zhong, Y., Zhang, Z., Liu, S., and Ji, R. (2024). CaM: Cache merging for memory-efficient LLMs inference. *Proceedings of the 41st International Conference on Machine Learning*.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Re, C., Barrett, C. W., Wang, Z., and Chen, B. (2023). H2O: Heavy-Hitter Oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*.