Subquadratic Models and KV Compression on Apple Silicon: An Axis-Dependent On-Device Efficiency Frontier

Anonymous authors

Abstract

Subquadratic sequence models (SSMs) and KV-cache compression both promise cheaper long-context inference, but almost all of the evidence comes from datacenter GPUs. We benchmark both on Apple Silicon under matched parameter and memory budgets, and two results run against the textbook intuition. First, quantizing the KV cache on the MLX native path lowers decode throughput rather than raising it: the dequantization step appears to cost more per token than the smaller cache saves. That direction is clean and position-independent at 8k, while at longer context the magnitude is confounded by thermal throttling under the long prefill, so we report the direction and not a stable number. Second, the SSM's prefill advantage arrives near 64k rather than early, far later than an $O(T^2)$-versus-$O(T)$ reading predicts, because chunked attention amortizes the quadratic term. Across a ~7B 4-bit triad in one MLX harness (dense Falcon3-7B-Instruct, a widely-used Mamba-2 SSM, and a same-lineage Mamba-1 control that isolates architecture from training corpus), the efficiency frontier is axis-dependent with no single crossover: the SSM takes decode first (between 4k and 8k), then memory (near 21k), then prefill (a single-run estimate near 64k), while below that the dense transformer leads on all three. And it gets cheaper exactly where it gets lossier: across 16k to 32k the off-the-shelf SSM retains only 47 to 63% of single-fact recalls (losing 37 to 53%) while the dense model stays lossless. A same-lineage Mamba-1 control locates an architectural component to that loss (significant at 32k, not at 16k), but the larger off-the-shelf gap is model-specific, so the recall of the SSM you would actually download is a per-model question the architecture label does not answer. On device the subquadratic win is narrow, kernel-dependent (two Mamba kernels differ 9.3x in prefill at 32k), and gated by recall. Code, run logs, and result tables will be released publicly.

1 Introduction

Large language models increasingly run on-device, on laptops and phones rather than datacenter GPUs, where a single pool of unified memory must hold the weights, the KV cache, and the activations, and where latency and energy budgets are tight. Long context makes this acute: attention costs $O(T^2)$ to prefill and its KV cache grows $O(T)$ during decoding. Two remedies dominate the literature. Subquadratic architectures (state-space models such as Mamba and Mamba-2, and SSM-attention hybrids) replace the growing KV cache with a fixed-size recurrent state. KV-cache compression (low-bit quantization) shrinks the cache an attention model must carry. Both are validated almost entirely on A100/B200-class hardware, and whether either helps on a unified-memory mobile SoC is unclear: its binding constraints (memory bandwidth, a quantized 4-bit weight path, and the maturity of the available kernels) differ from the datacenter.

We ask a narrow, practical question: at matched parameter and memory budgets on Apple Silicon, when, if ever, does a subquadratic architecture or a compressed KV cache actually move the latency, quality, and memory frontier for a local long-context workload? Using one uniform MLX measurement harness over a ~7B 4-bit triad (dense Falcon3, Mamba-2 Codestral, and a same-lineage Mamba-1 control), we characterize prefill, decode, memory, and long-context recall from 512 to 131,072 tokens.

Our contributions are four. First, the result we least expected, on the titular lever: on the MLX native quantized-KV path, quantizing the cache *lowers* decode throughput rather than raising it, because the per-step dequantization appears to cost more than the smaller footprint saves. The direction is stable and position-independent at 8k; the long-context magnitude is thermally confounded under the long prefill, so we report it as directional. 8-bit is dominated by 4-bit on memory and quality, and 2-bit is a recall cliff, so KV quantization here is a memory-for-speed trade, not a free lunch. Second, an on-device efficiency crossover map whose most counterintuitive feature is a late prefill crossover: the dense-vs-SSM frontier is axis-dependent, and the SSM overtakes the dense model one axis at a time as context grows, on decode between a measured 4k and 8k, on memory near 21k, and on prefill only near 64k, far later than an $O(T^2)$-versus-$O(T)$ reading predicts. The absolute crossover lengths are specific to this M5 Max and 4-bit setting; the portable claim is the axis ordering (decode first, then memory, then prefill), which follows from the mechanism rather than the device. The map is also stack-sensitive: two Mamba kernels differ 9.3x in prefill at 32k. Third, a confound-controlled quality result: dense attention is lossless on our recall probe while an off-the-shelf SSM degrades at long context, and a same-lineage Mamba-1 control establishes an architectural penalty that is significant at 32k but not at 16k, while showing the larger off-the-shelf gap is inflated by model-specific factors, so we report deployment guidance plus a bounded, Mamba-1-specific architectural claim rather than a clean architectural law. Fourth, the apparatus itself: a matched-budget, dual-frontier benchmark (iso-params and iso-memory, reported separately) with pre-specified hypotheses and kill-criteria, and a harness and run logs to be released publicly.

We offer no new quantizer and no new architecture. The KV-quant space is crowded (§2) and the SSM recall deficit is already known (§2). What this paper adds is a careful on-device measurement of both.

2 Related work

KV-cache quantization. A large and active line of work compresses the attention KV cache to relieve the $O(T)$ memory of long-context decoding. KIVI (Liu et al., 2024) quantizes keys per-channel and values per-token to 2 bits, tuning-free; KVQuant (Hooper et al., 2024) uses pre-RoPE per-channel key quantization with non-uniform, dense-and-sparse codebooks to reach multi-million-token contexts; subsequent work pushes into sub-bit and anchor-token-aware regimes. These methods are validated almost entirely on datacenter GPUs and are evaluated primarily on quality at fixed memory; the *latency* consequence of dequantization on a unified-memory mobile SoC is not their focus. A different class of KV compression evicts or sparsifies tokens rather than quantizing them (StreamingLLM, Xiao et al., 2024; H2O, Zhang et al., 2023), which avoids the per-step dequantization our decode-penalty finding turns on; we study the quantization path that MLX exposes natively, so the penalty we report is scoped to quantization-based compression, not eviction-based compression. KV quantization is also a lever for attention caches only; it does not apply to a pure SSM, which has no KV cache (its analogue would be recurrent-state quantization, which we do not study). We make this asymmetry explicit and compare architecture-plus-native-compression as deployable bundles.

Subquadratic / state-space models. Mamba (Gu & Dao, 2024) and Mamba-2 / SSD (Dao & Gu, 2024) give linear-time sequence models with a fixed-size recurrent state, and hybrids such as Falcon-H1 (Technology Innovation Institute, 2025) and Jamba (Lieber et al., 2024) interleave SSM and attention layers to recover recall. A complementary theoretical and empirical literature explains *why* fixed-state models trail attention at retrieval: transformers copy and do associative recall better than SSMs (Jelassi et al., 2024), recall is the axis along which efficient architectures most visibly lag (Arora et al., 2024), and the fixed state imposes expressivity limits (Merrill et al., 2024). We treat these as established priors and position our recall measurement as an on-device confirmation, not a discovery.

On-device / MLX. Apple's MLX (Hannun et al., 2023) provides unified-memory inference on Apple Silicon, where the same physical memory holds weights, KV cache, and activations, so memory pressure and bandwidth, not just FLOPs, govern feasibility. Prior MLX benchmarks characterize single models but, to our knowledge, none jointly study the subquadratic-architecture and KV-compression axes under a matched-budget protocol.

Long-context evaluation. RULER (Hsieh et al., 2024) and the broader needle-in-a-haystack line, including LongBench (Bai et al., 2024), stress long-context recall at datacenter scale. We adapt the NIAH protocol to

device-reachable lengths and to an exact-match single-key continuation probe that removes output-format and instruction-following confounds, so that storage capacity is measured rather than generation ability.

3 Method: matched-budget dual frontier

A configuration is a tuple c = (model, architecture, kv_bits, T, needle-depth), and for each one we measure four quantities: prefill latency, decode throughput, peak memory, and NIAH recall accuracy. The central design choice is how to "match" budgets across architectures that scale differently. A pure SSM carries an $O(1)$ recurrent state; a dense transformer carries an $O(T)$ KV cache. No single scalar budget is fair to both, so we never collapse the comparison into one "matched budget" number. Instead we report two frontiers separately: an iso-parameter frontier (all models are ~7B at 4-bit) and an iso-memory frontier (configurations compared at equal peak unified memory). KV-cache quantization is a lever for attention caches only; its pure-SSM analogue would be recurrent-state quantization, which we do not study. We make that asymmetry explicit and compare each architecture together with its native compression as a deployable bundle, rather than pretending the two levers are interchangeable.

All measurements run through one MLX harness. We place explicit `mx.eval` barriers so the lazy graph is forced at known points, warm up before timing, and prefill long prompts in fixed-size chunks so the prefill graph stays bounded (the uncontrolled version exhausts Metal's allocator even at T=512; see §5.1). Peak memory is the global peak from `mx.get_peak_memory`, reset at the start of each configuration's run. Absolute peaks differ modestly between the efficiency-sweep harness (§5.1) and the KV-quant harness (§5.3), so we compare memory only within a harness; the SSM's fixed state is independent of the prefill path and reads the same ~10.1 GB in both, which is what makes the cross-architecture memory comparison in §5.4 valid. Decode is greedy and measured on the production `generate_step` path over a warmed, fixed-length output window; a short or unwarmed decode window under-reports steady-state throughput, an artifact we hit early (our first decode pass was biased ~2x low) and corrected by warming each context length and validating against `mlx_lm.generate`'s own throughput counter. Timing metrics aggregate over run-repeats; quality metrics aggregate over trials that each query a randomly chosen stored key.

The model triad is fixed throughout. The dense baseline is `Falcon3-7B-Instruct-4bit` (Technology Innovation Institute, 2024). The subquadratic model is `Mamba-Codestral-7B-v0.1-4bit`, a Mamba-2 SSM from a separate Mistral *code* lineage and the SSM a practitioner would actually download at this scale. The confound control is `Falcon3-Mamba-7B-Instruct` (Zuo et al., 2024) (Mamba-1), which shares the dense model's Falcon3 lineage and instruct-tuning and differs from it primarily in architecture (with a smaller recurrent state than Mamba-2, ~4.5 vs ~10 GB peak, a sub-confound we note but cannot resolve with two SSM models). The contrast between the two SSMs and the dense model is what lets us partially separate "the architecture" from "the particular model," enough to establish an architectural penalty at long context, not enough to fully decompose the larger off-the-shelf gap (§5.2). We also attempted a hybrid, `Falcon-H1-7B-Instruct-4bit`, as a "does adding attention back rescue recall" datapoint, but it scored 0.0 (none of the 8 needles recalled) on the multi-needle probe already at 2k tokens, before long context is even in play. Because that floor is on the harder list-all-K variant (which confounds storage with generation, §5.2), we could not get a fair recall reading from it, so we exclude it and say so plainly (§5.2).

4 Hypotheses and kill-criteria

We pre-specified three hypotheses and the conditions that would kill the paper, so the results below read as verdicts rather than post-hoc stories.

H1 predicted a context range where the subquadratic architecture Pareto-improves on dense. It is supported, but only axis-by-axis: the SSM wins decode beyond the 4k-to-8k crossover and memory beyond ~21k, while dense wins prefill until roughly 64k. The win is also gated by quality: in the very range where the off-the-shelf SSM gets cheaper, its recall degrades sharply (§5.2); the same-lineage Mamba-1 control is statistically indistinguishable from dense at 16k but significantly lossier at 32k, locating an architectural component to the loss at long context. So the Pareto "win" is cheaper-but-lossier inference for the deployable model, not a strict domination.

H2 predicted that moderate KV-quantization is roughly quality-neutral while aggressive quantization breaks. It is supported on quality (4-bit and 8-bit hold recall at 1.00, and 2-bit is the cliff), but §5.3 adds a twist we did not anticipate: on this MLX path quantization *costs* decode throughput rather than saving it. The direction is stable and position-independent at 8k; at 16k/32k the sweep shows the same ordering but the magnitude is thermally confounded (and 16k is unverified by the diagnostic). Parity holds on quality, not on speed.

H3, the negative-result branch (dense-plus-KV stays competitive to the device limit on every axis), was not triggered, though the axis that breaks it is decode, not memory: quantizing the dense cache costs decode speed and only deepens the SSM's decode lead, while on memory 4-bit-KV dense stays under the SSM line through 32k (the uncompressed fp16-KV dense is the configuration beaten on memory, near 21k), so compressing the dense cache postpones the memory crossover beyond 32k rather than eliminating it (§5.4). Our three kill-criteria were dense infeasibility forcing censored cells (Kill A), thermal drift above 20% corrupting timing (Kill B), and the absence of any Pareto gain, which would have turned this into a purely negative report (Kill C). Kill B's 20% threshold was in fact met for the KV-quant 32k decode cell (the cold-to-warm swing is ~26%, §5.3); we handle that by demoting that cell to a directional result and not building a numeric magnitude claim on it, so the finding is scoped rather than invalidated. None of the three fired in a way that invalidates the program.

5 Results

5.1 Efficiency frontier

We swept prefill latency, peak memory, and steady-state decode throughput from 512 to 131,072 tokens for the dense transformer and the Mamba-2 SSM, all 4-bit on the M5 Max. Prefill and memory come from the efficiency sweep (Table 1); decode we re-measured for both models on the production `generate_step` path over a warmed window (Table 2), because the sweep's tiny single-token decode window was biased ~2x low. All three axes are measured through one harness, which lets us pin each crossover directly.

Table 1. Prefill latency and peak memory vs context length. Dense 512 to 32k cells are n=3; dense 64k/128k are n=2; the Mamba-2 cells are single runs (n=1). Its memory is flat by construction and reproducible, but its prefill standard deviations are unquantified.

T	Dense prefill	Dense mem	Mamba-2 prefill	Mamba-2 mem
512	0.6s	4.88GB	1.8s	10.06GB
2k	2.3s	5.21GB	5.6s	10.14GB
8k	11.0s	6.44GB	21.2s	10.14GB
32k	67.1s	11.68GB	80.7s	10.14GB
64k	153.9s	18.86GB	154.9s	10.14GB
128k	510.8s	33.22GB	313.5s	10.14GB

Dense prefill scales super-linearly (8k to 32k is 6.1x latency for 4x tokens, a two-point slope of about $T^{1.3}$ on this hardware), the $O(T^2)$ attention cost partly amortized by chunked prefill on the MLX path, while the SSM is roughly linear but starts from a larger constant. The two prefill curves meet near 64k (153.9s vs 154.9s), but both sides of that crossing are sparse: the dense point is n=2 with an observed sd of ~4.8s and the Mamba-2 point is a single run (n=1) with unquantified variance, so the 1.0s margin sits inside the combined uncertainty and we read this as a single-run crossover near 64k rather than a tightly pinned tie. Beyond it the SSM is faster (128k: 313.5s vs 510.8s). Memory tells the opposite, simpler story: the dense KV cache grows monotonically (4.88 to 33.2GB) while the SSM's recurrent state is flat at ~10.1GB regardless of T, so the dense model crosses above the SSM in memory between the 8k and 32k samples. No length was sampled between those two, so the crossover is interpolated, not measured: log-linear interpolation of the sweep's global peak (Table 1) places it near 21k, and linear-in-T interpolation near 25k. We report ~21k, the figure value, and note it is an interpolation with a roughly 21k to 25k band. The KV-quant harness reads

dense fp16 memory lower at the same length (9.21 vs 11.68 GB at 32k, §5.3), which would place the crossover later still, so ~21k is the earliest, worst-case-for-dense estimate and the axis ordering is the portable claim.

Table 2. Steady-state decode throughput (tok/s), production `generate_step` path, warmed 64-token window, n=3 to 5. Both models measured through the identical harness, so the columns are directly comparable.

T	Dense decode	Mamba-2 decode
1k	88.5 ± 1.5	82.5 ± 2.0
2k	97.6 ± 2.5	87.5 ± 1.2
4k	96.6 ± 2.4	86.6 ± 0.5
8k	64.9 ± 2.6	98.8 ± 0.3
16k	44.6 ± 2.5	96.9 ± 1.2
32k	30.2 ± 2.5	75.0 ± 6.9
64k	19.5 ± 4.2	67.9 ± 0.3

Dense decode falls as $O(T)$: every generated token attends over a longer KV cache, so throughput drops monotonically from 97.6 tok/s at 2k to 19.5 at 64k. We measure this on the production `generate_step` path over a warmed 64-token window, and the rates agree with `mlx_lm.generate`'s own verbose throughput counter at 8k, which is how we validated the harness. A short or unwarmed decode window under-reports steady-state throughput, so the efficiency sweep's single-token decode samples were biased low by roughly 2x; that is why both columns here are re-measured on the warmed window rather than read off the sweep.

The two decode curves cross between 4k and 8k (interpolated; we did not sample a midpoint, so the precise crossover length is unresolved within that interval). Below it, dense is actually faster (96.6 vs 86.6 at 4k): its KV cache is still small, and the Mamba-2 kernel carries a larger fixed per-step overhead. Above it, the SSM wins at every length we tested (98.8 vs 64.9 at 8k, up to 67.9 vs 19.5 at 64k). The mechanism is the SSM's fixed-size recurrent state: its peak memory is flat at ~10.1 GB across every T (Table 1), so per-step work does not grow with context the way the dense $O(T)$ KV scan does. Two caveats apply to the long-context SSM numbers. First, the SSM column is not perfectly flat (98.8 at 8k down to 67.9 at 64k); the within-cell 32k trials decay monotonically (83.2 to 67.5 tok/s across five runs) with peak memory pinned at ~10.1 GB, a thermal-drift signature, not per-step cost growing with T. The decode bench sweeps lengths in sequence, so by the long-context cells the chip has been under sustained load (the 32k and 64k prefills alone run ~81 s and ~155 s), and the dense column shows the same within-cell decline at 64k; the flat memory is what rules out per-step cost growth as the cause. We therefore read the $O(1)$-per-step cost as the mechanism and report the measured throughput as a lower bound on the steady-state SSM rate at 32k/64k. Second, we state the SSM's margin as measured (roughly 1.5x at 8k to 3.5x at 64k) and do not lean on its *growth* as an architectural claim, since the thermal suppression at 64k makes the true ratio an underestimate.

The headline for this section is the map, not any one crossover: the axes cross in a fixed order as context grows, the SSM taking decode first (between 4k and 8k), then memory (near 21k), then prefill (near 64k), while below that range the dense transformer wins all three. The map is also stack-sensitive: the two Mamba implementations we measured differ by **9.3x** in prefill at 32k (Mamba-1 at 751s vs Mamba-2 at 80.7s), so on device the kernel one happens to have matters as much as the O(.) on paper.

5.2 Quality vs context: SSM recall degrades at long context

We use a RULER-style NIAH probe. A single-needle (one-fact) version saturates at 100% for both models and is not discriminative, so our primary probe is a clean single-key query: store K=8 facts in a T-token haystack, then query one of them via pure continuation ("...The magic number for {key} is") and score by exact match. This removes output-format and instruction-following confounds; a list-all-K variant, by contrast, floors weak instruct models and confounds storage with generation. We ran n=30 independent prefills per cell at the discriminating lengths (16k and 32k), each trial querying a random stored key.

(a) Single-key recall vs context (n=30 at 16k/32k; n≤10 elsewhere).

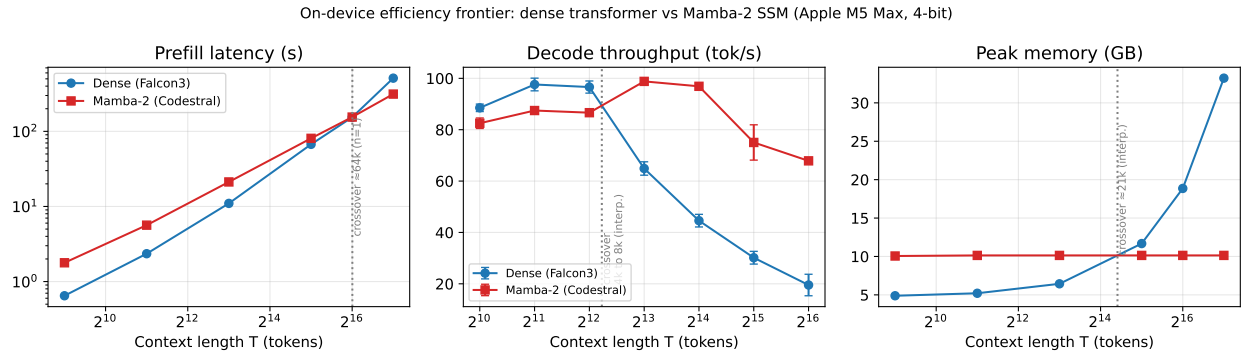

Figure 1: On-device efficiency frontier, dense transformer vs Mamba-2 SSM (M5 Max, 4-bit). All three panels are clean `generate_step`/sweep measurements through one harness. The SSM overtakes the dense model on decode between 4k and 8k, on memory near 21k, and on prefill near 64k; below that range the dense transformer leads on every axis.

T	Dense (attn)	Mamba-2 SSM	Dense vs SSM (Fisher exact)
2k	1.00	1.00	saturated
8k	1.00	0.80	n=10
16k	**1.00** (30/30)	**0.47** (14/30)	$\mathbf{p} \approx 2 \times 10^{-6}$
32k	**1.00** (30/30)	**0.63** (19/30)	$\mathbf{p} \approx 3 \times 10^{-4}$

The Wilson 95% CIs at 16k are cleanly separated: dense [0.89, 1.00] versus Mamba-2 [0.30, 0.64]. Dense is lossless at every length, since its KV cache is exact, while the Mamba-2 SSM drops to ~0.5 to 0.6 at long context. An earlier n=10 read gave 0.70/0.80 at 16k/32k; those were optimistic small-sample draws, and at n=30 the gap is both larger and significant, so we report the n=30 numbers.

(b) The model-family confound, controlled, and what it does to the architectural reading. Dense (Falcon3-7B-Instruct, TII) and Mamba-Codestral (a Mistral code-base model) differ in lineage and instruct-tuning, not only architecture. As a control we add Falcon3-Mamba-7B-Instruct: same TII Falcon3 lineage and instruct-tuning as the dense model, differing only in architecture (Mamba-1, a pure SSM). We run the control at both probe lengths (the Mamba-1 control at 32k is n=20 rather than 30 because its per-timestep kernel makes each 32k trial the most expensive in the study; the other cells are n=30). The reduced count only costs power, and the effect survives it: 14/20 is significantly below dense 30/30 (Fisher $p = 2.4 \times 10^{-3}$), and a binomial 95% interval on 0.70 already excludes the dense 1.00, so the attribution does not rest on the n=20 versus n=30 difference:

Model	16k recall	Fisher vs dense	32k recall	Fisher vs dense
Dense (Falcon3 attn)	30/30 = **1.00**	(self)	30/30 = **1.00**	(self)
Falcon3-Mamba-1 (same lineage)	26/30 = **0.87**	p = 0.11 (NS)	14/20 = **0.70**	$p = 2.4 \times 10^{-3}$
Codestral Mamba-2 (Mistral code base)	14/30 = **0.47**	$p = 1.9 \times 10^{-6}$	19/30 = 0.63	$p = 3 \times 10^{-4}$

The control supports a partial architectural attribution, with two guardrails on how far it reaches. The existence of an architectural recall penalty is established at long context: at 32k the same-lineage Mamba-1, which differs from dense essentially only in architecture, is at 0.70, significantly below dense (Fisher $p = 2.4 \times 10^{-3}$). At 16k the same control is at 0.87, not significantly different from dense ($p = 0.11$), so the 16k result is consistent with no architectural penalty at that length; the penalty is established at 32k only. We are careful not to over-read this as the penalty growing with length: "significant at 32k, not at 16k" is a difference-in-significance, not a significant difference. The direct within-control growth test, Mamba-1 16k to 32k (0.87 to 0.70), is itself non-significant (Fisher $p = 0.17$), so we claim emergence of detectability, not a measured growth rate. One scope limit follows from the control's identity: because it is a Mamba-1 model, what is established is an architectural penalty for a Mamba-1 fixed-state SSM. It does not transfer automatically to the Mamba-2 model a practitioner would actually deploy, whose own long-context recall we measure directly (§5.2a) but, with only two SSM models, cannot cleanly decompose into architecture versus the model-specific factors below.

The attribution stays partial because the raw Codestral gap is inflated by model-specific factors. At 16k the two SSMs differ sharply (0.87 vs 0.47, Fisher $p = 2.2 \times 10^{-3}$), so most of Codestral's 16k deficit is not architectural; the like-for-like control holds at 0.87 there. With only two SSM models the Codestral gap confounds at least four differences at once: training corpus (general vs code), instruct-tuned vs base, Mamba-1 vs Mamba-2 (itself an architectural change), and state size (~4.5 vs ~10 GB peak), so its magnitude is unidentifiable. One of these four cuts against a pure capacity reading: Codestral's Mamba-2 carries the *larger* recurrent state of the two SSMs (~10 vs ~4.5 GB peak), which a capacity argument would expect to help recall, yet it is the lossier model, so state size does not explain its deficit and the corpus and tuning differences are the more likely drivers. (At 32k the two SSMs are statistically indistinguishable, 0.70 vs 0.63, Fisher $p = 0.76$; we report this as a fact at each length, not as a convergence trend, since Codestral's 0.47 to 0.63 synthetic bounce does not survive on real prose, §5.2d.) What is supported: dense is lossless at both lengths; a like-for-like SSM carries an architectural penalty established at 32k; and the off-the-shelf SSM one would actually download is lossier still, by an amount this two-model design cannot fully attribute. We therefore frame the quality axis as deployment guidance plus a partial architectural attribution, not a clean architectural verdict.

(c) The failure mode is silent confabulation. Every SSM miss is a correctly-formatted but wrong 5-digit number (e.g. 70911→44444, 52351→16053), never garbage or a format break; one 16k miss even preserved the last four digits (42915→12915), consistent with lossy fixed-state compression rather than wholesale dropout. The SSM fails confidently and silently, with no reliability signal, which is the most dangerous mode for deployment.

(d) The deficit is not an artifact of synthetic filler: it reproduces on natural prose. The probe above embeds the needles in neutral repeated filler, which invites the objection that the SSM is only failing on degenerate, low-entropy context. We therefore re-ran the identical exact-match probe with the haystack drawn from real scientific text (PubMed article bodies) instead of filler, n=20 per cell. The dense model stays perfectly lossless on real prose (1.00 = 20/20 at both 16k and 32k), which validates the probe: real scientific prose is not simply hard for everyone, so a low score is a model property, not a probe artifact. The off-the-shelf Mamba-2 (Codestral), by contrast, recalls 0.35 at 16k (7/20, Wilson 95% CI $\approx$ [0.18, 0.57]) and 0.20 at 32k (4/20, $\approx$ [0.08, 0.42]) on real prose, with the same silent-confabulation signature (e.g. 85616→27583, 46578→53378). The dense-versus-SSM gap on real prose is therefore both large and clean (1.00 vs 0.20 at 32k, Fisher $p \approx 1.5 \times 10^{-7}$). Two further points. The deficit is at least as severe on real prose as on filler (0.35 vs 0.47 at 16k, overlapping CIs), so it is a fixed-state capacity effect, not a filler-specific quirk. And on real prose the degradation is monotonic in length (0.35→0.20 from 16k to 32k), unlike the slightly non-monotonic synthetic series (0.47→0.63); we do not test whether that synthetic non-monotonicity is haystack-specific or sampling noise, but the monotonic real-prose trend is what a fixed-state bottleneck predicts: more context, less reliable single-fact recall. Peak memory is the same flat ~10.1 GB, confirming the real-text haystack exercises the same recurrent state.

Dense holds perfect single-fact recall across context, while an off-the-shelf fixed-state SSM (Codestral) loses ~37 to 53% of long-context recalls over 16k to 32k, and this degradation sets in right around the ~21k memory crossover (§5.1), well after the SSM has already taken the decode and (soon) the memory axes. For

the model you would actually download, the SSM gets cheaper exactly where it gets lossier. A like-for-like Mamba control is statistically indistinguishable from dense at 16k (0.87, NS) but significantly lossier at 32k (0.70, p=2.4×10^{-3}), so the architectural penalty is established at 32k only, while the larger off-the-shelf gap is partly model-specific. Either way the matched-budget frontier is a quality-efficiency tradeoff to verify per model, not a free win.

Hybrid note: Falcon-H1-7B-Instruct (4-bit) scored 0.0 on the list-all-K multi-needle probe already at 2k. That probe confounds storage with generation, and we did not run the lenient single-key probe (used for the other quality results) on H1 due to resource constraints, so a fair "does attention rescue recall" reading is unavailable; we exclude it and report the gap transparently rather than over-reading the 2k floor. Whether H1's attention layers rescue single-fact recall on the lenient probe is left to future work.

5.3 KV compression (the titular axis)

The headline of this section is one counterintuitive ordering: on the MLX native quantized-KV path, fp16 KV decodes fastest and quantizing the cache lowers throughput. That direction is clean and position-independent at 8k; the rest of this section is about how loosely we can pin its magnitude, which the long prefill's thermal load confounds at long context. The setup: native MLX quantized-KV (`generate_step`, `kv_bits/kv_group_size=64`, chunked prefill) on the dense model, single-needle NIAH with a stable 64-token decode window, kv_bits {16,8,4,2} crossed with T {8k,16k,32k}. Decode and memory are measured n=5 in one sequential sweep over bit levels; the 2-bit accuracy cells, the one quality result we lean on, were re-run at n=20 for tighter intervals. The decode numbers carry one caution: absolute decode throughput on this shared-memory laptop varies between separate process launches by up to ~40% with background machine load (a fresh fp16 8k read of 97.9 tok/s vs the sweep's 70.6 under identical settings), so we report one internally-consistent sweep and read only *within-sweep* ordering, not absolute rates. To check that the within-sweep ordering is trustworthy we ran an order-swap diagnostic (`src/diag_thermal_order.py`): measure fp16 and 2-bit in both orders, isolated per process. At 8k the bits effect is position-independent (fp16 97.9 cold / 97.2 warm; 2-bit 83.8 cold / 80.6 warm), about a 16% quant penalty either way, so the 8k comparison is solid in *direction*. The penalty *ratio* is itself run-variable, a further facet of the between-run variance above: the order-swap diagnostic reads ~16% while the plotted sweep reads ~26% (70.6 to 52.5) at the same length, so we treat the 8k magnitude as modest rather than a single pinned percentage. At 32k the direction-vs-magnitude split worsens. Decode is thermally throttled under the 67 s prefill (Table 1) accumulated across the cell: fp16 drops from 60.1 (cold) to 44.3 (warm), and the cold 2-bit cell itself decays within trials (33.6 to 24.0). The penalty estimate is therefore not identifiable, ranging from about 53% (cold) to 9% (warm-steady) across measurement states and bracketing the single-sweep 65%. So at long context we report direction, not magnitude. One asymmetry to flag in the diagnostic itself: each "warm" cell is measured second, after the other precision's prefill has already heated the device, so the warm fp16 and warm 2-bit readings are not thermally interchangeable; this is one more reason we lean on the cold-vs-warm span rather than any single warm cell. This harness prefixes a NIAH prompt rather than the pure filler of §5.1, which shifts absolute decode rates substantially (fp16 here reads about 9% higher at 8k, 35% higher at 16k, and 30% higher at 32k than the §5.1 decode_bench numbers at the same T), so the two harnesses are not cross-comparable cell-for-cell: we compare only within this table and read the §5.1 dense curve from its own harness. The same harness difference shifts absolute peak memory: dense fp16 reads lower here than in Table 1 (9.21 vs 11.68 GB at 32k), so memory too is compared only within a table; the SSM's ~10.1 GB is path-independent and identical across both, which licenses the §5.4 cross-architecture memory comparison.

T	metric	fp16	8-bit	4-bit	2-bit
8k	decode tps	**70.6**	61.2	63.0	52.5
	peak mem GB	5.70	5.37	5.20	5.11
	accuracy	1.00	1.00	1.00	0.60
16k	decode tps	**60.1**	41.7	42.0	37.5
	peak mem GB	6.78	6.04	5.54	5.28
	accuracy	1.00	1.00	1.00	0.80
32k	decode tps	**39.2**	13.4	16.1	13.8

T	metric	fp16	8-bit	4-bit	2-bit
	peak mem GB	9.21	7.73	6.72	6.22
	accuracy	1.00	1.00	1.00	0.65

Decode at 32k is thermally confounded under the long prefill (panel (a) and methods) and 16k is unverified by the order-swap diagnostic (we ran it only at 8k and 32k); read both decode rows as directional, not as magnitudes, and both are shown dashed in Figure 2. Memory and accuracy rows are stable.

Accuracy n: fp16/8-bit/4-bit are n=5 (single-needle, saturating at 1.00, which fixes the comparison baseline); 2-bit is n=20 (re-run; Wilson 95% CIs [0.39,0.78] / [0.58,0.92] / [0.43,0.82] at 8k/16k/32k).

(a) KV-quant is a memory-for-speed trade: it lowers decode throughput rather than raising it. fp16 KV decodes fastest at every length; quantizing the cache *lowers* throughput, the opposite of the naive "less memory means faster" intuition. We attribute this to the dequantization kernel, which runs each decode step over the KV cache and adds per-step work that the smaller footprint does not pay back in speed; we do not have kernel-level profiling to rule out an MLX implementation-specific overhead, so we state that mechanism as the most likely explanation rather than a measured one. The plotted sweep shows quant no faster than fp16 at every length; the order-swap diagnostic then tests whether that ordering is precision-driven or a measurement-order artifact, and confirms it is precision-driven at 8k (we ran the diagnostic only at 8k and 32k, so 16k is shown dashed as unverified). The *magnitude* is where we are careful. At 8k the penalty is position-independent (the order-swap shows no cold/warm effect); its magnitude is modest but itself run-variable, about 16% in the diagnostic and 26% in the plotted sweep. At long context we do not claim a precise number: the single-sweep figure (a 65% drop at 32k) conflates the quantization cost with thermal throttling under the long prefill, and across measurement states the 32k penalty ranges from about 53% (cold) to 9% (warm-steady), so it is not identifiable (§5.3 methods). The reading is directional: KV-quant costs decode speed on this stack, with the long-context magnitude thermally confounded rather than a stable 65%.

(b) 8-bit KV has no advantage over 4-bit here. 4-bit uses strictly less memory at every length, a deterministic, reproducible win (the peak-memory numbers are identical across runs), and matches 8-bit on accuracy (both 1.00). On decode the two are comparable; the small cell-level gaps (e.g. 32k 16.1 vs 13.4) sit inside the between-run variance noted above, so we do not lean on a timing difference. The case against 8-bit rests on memory and quality parity, where it is clear: 8-bit is never preferable to 4-bit on this stack. 2-bit is the quality cliff: it breaks recall (0.60 at 8k, 0.80 at 16k, 0.65 at 32k, n=20) even on this saturating single-needle probe, for only ~7% more memory savings than 4-bit. The usable aggressive setting is 4-bit (quality-neutral, 27% less memory at 32k). At n=20 these 2-bit cells are tightly enough bounded to settle the cliff itself (Wilson 95% CIs $\approx$ [0.39, 0.78], [0.58, 0.92], [0.43, 0.82] at 8k/16k/32k), every upper bound well below the 1.00 that fp16, 8-bit, and 4-bit all hold. The three values are non-monotone in length (0.60, 0.80, 0.65) and their intervals overlap, so the data establish that 2-bit breaks recall but leave any length trend unresolved; we read it as a broken operating point, not a length curve. The memory numbers are the deterministic, reproducible part of this section; the decode direction (quant is slower) holds; the long-context decode magnitude is the one quantity we hold loosely.

(c) The iso-memory leg revises the earlier "SSM wins memory" reading. Our first efficiency sweep had the Mamba-2 SSM's flat ~10.1 GB beating dense fp16-KV beyond ~21k. But with 4-bit KV, dense at 32k uses 6.72 GB, below the SSM's ~10.1 GB, while holding recall at 1.00. So a KV-quantized dense model stays under the SSM memory line through 32k, pushing the memory crossover well beyond 32k (§5.1). The dual frontier, then: KV-quant helps dense on memory (at a decode cost) but widens the SSM's decode advantage. The harness-consistent cross-architecture comparison (§5.1, Table 2) puts dense fp16-KV decode at 30.2 tps against the SSM's 75.0 at 32k; both 32k decode rates are thermally throttled under the long prefill (we already report the SSM as a lower bound, and the same throttling depresses the dense side), so we read this as a *direction* (the SSM owns the long-context decode axis) rather than a precise 2.5x ratio. Quantizing the dense cache for memory does not reverse it: it costs dense decode speed, so it only deepens the SSM's decode lead. No single config wins on quality, memory, and decode at once, which is the three-way tradeoff the matched-budget method was built to surface.

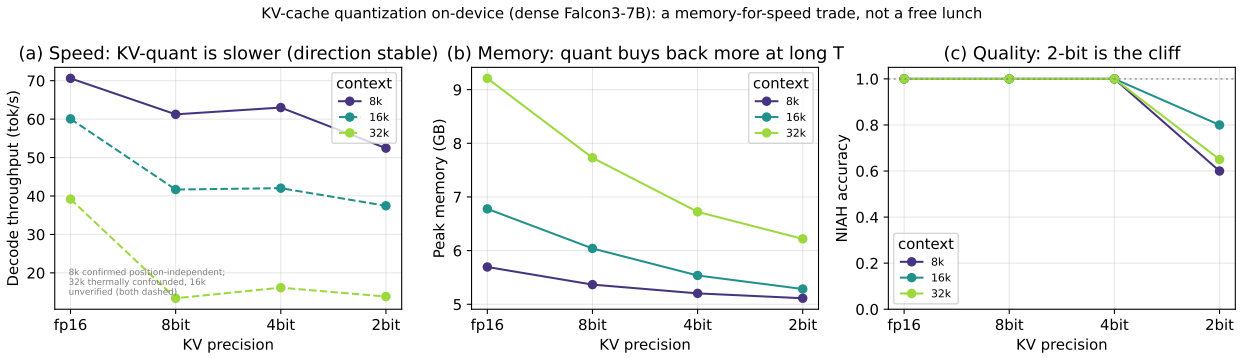

Figure 2: KV-cache quantization on-device (dense Falcon3-7B); decode and memory n=5 in one sequential sweep, 2-bit recall re-run at n=20. Quantizing the cache lowers decode throughput rather than raising it (a); the direction is stable, while the magnitude is position-independent only at 8k (modest, run-variable ~16 to 26%). The 32k magnitude is thermally confounded under the long prefill and 16k is unverified (we diagnosed only 8k and 32k), so both are shown dashed (order-swap diagnostic, §5.3). It buys back progressively more memory at long context (b), a deterministic win, and holds recall until the 2-bit cliff (c). A memory-for-speed trade, not a free lunch.

5.4 Combined Pareto: the no-free-lunch view

Putting quality, memory, and decode on one plot at T=32k (Figure 3; `master.csv`) makes the central claim concrete, and the recall axis is a single-probe comparison. The dense model's 1.00 at 32k is measured on the K=8 single-key probe at fp16-KV (§5.2, 30/30); we ran the identical probe (continuation design, seed 3, n=30) on the dense model with a 4-bit KV cache and measured recall 1.00 (30/30) at 32k, so dense 4-bit-KV is lossless on the same probe the Mamba-2 SSM's 0.63 was scored on. The dense-4-bit-versus-SSM quality comparison therefore rests on one protocol rather than a bridge between two. The §5.3 KV-quant sweep is consistent with this on the easier single-needle probe: 4-bit preserves recall while 2-bit breaks it (0.65). Memory and decode are the measured peaks/throughputs.

Quality versus memory is the strong result, and it survives the one cross-harness comparison it depends on. The dense 6.72 GB comes from the KV-quant harness and the SSM's 10.14 GB from the efficiency sweep, two harnesses whose absolute peaks differ (dense fp16 reads 9.21 vs 11.68 GB at 32k across them). What licenses the comparison is that the gap between harnesses lives entirely in the KV-cache allocation path, which a pure SSM does not have: with no KV cache, the SSM's peak is just weights plus its fixed recurrent state, and it reads the same ~10.1 GB in every harness we ran (efficiency sweep, single-key probe, real-prose probe). The dense advantage is also robust to which harness we read it from: dense 4-bit-KV at 6.72 GB beats the SSM's 10.14 GB, and even dense fp16-KV at its higher 9.21 GB still undercuts it. At 32k the dense model with 4-bit KV therefore Pareto-dominates the Mamba-2 SSM: it uses less memory (6.72 vs 10.14 GB) and retains effectively perfect recall (1.00 vs 0.63). Dense fp16-KV also out-recalls the SSM at higher memory (9.21 GB, 1.00), and dense 2-bit-KV is the only way to undercut the SSM further on memory (6.22 GB), but it pays for that in recall (0.65). So at device-reachable lengths the SSM is not on the quality-memory Pareto front; a KV-quantized dense model is both smaller and more reliable.

Where the SSM does win is the flat profile. Its memory is constant in T and its per-step decode cost is $O(1)$, whereas dense KV (even at 4 bits) keeps growing. Dense 4-bit-KV memory rises 5.20 to 5.54 to 6.72 GB across 8k/16k/32k, and the steps are accelerating (0.34 then 1.18 GB), so it does not reach the SSM's flat 10.14 GB until somewhere beyond 32k. We do not measure that far and the three-point trend is too short to fit a curve to, so we report the postponement only as "past the ~21k crossover for fp16-KV dense", not as a pinned length. So KV-quant does not eliminate the SSM's memory advantage; it pushes it out by several times in context length. On decode throughput the SSM is already ahead by 32k, measured on a consistent harness: the warmed `generate_step` measurement (§5.1, Table 2) gives dense fp16-KV 30.2 tok/s versus the SSM's 75.0 at 32k, both for the same window. We read this as a *direction* rather than a precise ~2.5x

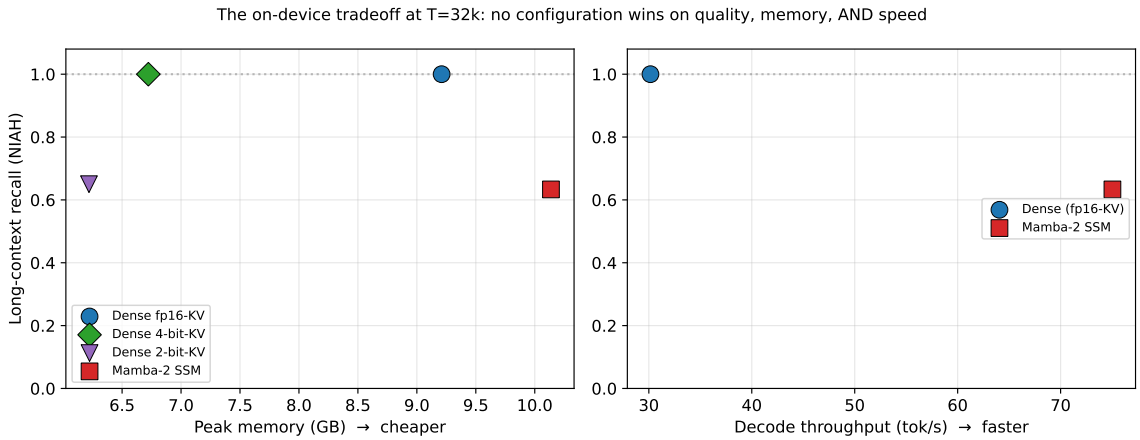

Figure 3: The on-device tradeoff at T=32k. Left: recall vs memory across KV-quant levels, where a 4-bit-KV dense model Pareto-dominates the SSM (less memory, perfect recall). Right: recall vs decode throughput, with both architectures measured on the same warmed `generate_step` harness (dense fp16-KV 30.2 vs SSM 75.0 tok/s); both 32k rates are thermally throttled under their long prefills, so we read the SSM's decode lead as a direction rather than a precise ratio. No configuration wins on quality, memory, and decode at once.

ratio: at 32k both decode rates are thermally throttled under their long prefills (we already report the SSM as a lower bound, and the dense side is depressed the same way, §5.3), so the gap is real but its exact size is not. Quantizing the dense KV for memory does not reverse it, since it costs dense decode speed too (§5.3). Either way the SSM owns the decode axis at this length, which is the point the matched-budget plot makes concrete.

No single configuration wins on quality, memory, and decode simultaneously at 32k: dense-fp16 has the best quality and worst memory, dense-4bit-KV the best quality and memory but worst decode, 2-bit buys memory at a recall cost, and the SSM has the best decode and a flat profile but the worst recall. The on-device "right answer" is workload- and length-dependent, which is exactly the decision the matched-budget dual frontier was built to inform.

6 Discussion

On a single Apple M5 Max, across a ~7B 4-bit triad, the long-context inference frontier is axis-dependent with no single crossover. The Mamba-2 SSM overtakes the dense transformer one axis at a time as context grows, on decode between 4k and 8k, on memory near ~21k against fp16-KV dense (pushed past 32k once the dense cache is 4-bit quantized, §5.4), and on prefill near ~64k (a single-run crossover, the dense side n=2 with ~4.8s sd, so read as directional rather than a tightly pinned length; §5.1). That prefill crossover arrives later than a naive $O(T^2)$-vs-$O(T)$ extrapolation would predict: dense prefill grows only about 6x over the 8k to 32k interval for a 4x token increase (a local two-point slope near $T^{1.3}$, not a fitted law), because chunked prefill amortizes the attention cost, and the SSM starts from a constant roughly 3x higher than dense at short context (1.8s vs 0.6s at 512 tokens), so dense holds the prefill lead far past where a pure-quadratic curve would have lost it. The titular lever, KV-cache quantization, is a memory-for-speed trade that costs decode speed rather than saving it (§5.3): it lets a dense model dip below the SSM's flat memory line while staying lossless, but it widens the SSM's decode advantage and never closes it. And on quality, dense attention is lossless on our recall probe at every length while an off-the-shelf SSM degrades substantially at long context, and a same-lineage Mamba control establishes an architectural penalty that becomes significant by 32k (0.70 vs dense 1.00, p=2.4×10^{-3}) while remaining non-significant at 16k. The larger off-the-shelf gap is inflated by model-specific factors the two-model design cannot fully separate (§5.2), so we frame recall as deployment guidance plus a bounded, partial architectural attribution, not a clean architectural law.

Relation to prior work. That fixed-state models trail attention on associative recall and copying is already known, but the evidence comes from datacenter GPUs (Jelassi et al., 2024; Arora et al., 2024; Merrill et al., 2024). We do not re-derive it. We measure it on device, and add a same-lineage control and significance tests. Two of our findings we have not seen reported before: the per-axis crossover map on Apple Silicon, and the KV-quant decode penalty, where quantizing the cache slows decode down instead of speeding it up on the MLX native path.

6.1 Limitations

1. **Single device, single scale.** One M5 Max, one ~7B parameter band, 4-bit weights. Absolute crossovers are device- and quant-specific, so we report the structure (which axis crosses where) as the portable claim.
2. **Off-the-shelf SSM kernels.** The 9.3x Mamba-1-vs-Mamba-2 prefill gap shows kernel maturity matters as much as asymptotics; our SSM numbers reflect current MLX implementations, not the architecture's limit. The gap is also partly a chunking artifact: the Mamba-1 path required chunk size 256 to stay under MLX's single-shot op limit, while Mamba-2 and dense used 512, so the 9.3x is an upper bound on the architectural gap rather than the gap a fully tuned Mamba-1 kernel would show.
3. **Decode-tps carries a thermal and measurement confound at long context, for both architectures and for KV-quant.** We measure decode on the warmed 64-token `generate_step` window, which pins the decode crossover (between 4k and 8k) cleanly (short prefill, low thermal load). But two effects make long-context decode magnitudes unreliable. First, absolute decode throughput varies up to ~40% between separate process launches with background load on this shared-memory laptop (a fresh fp16 8k read of 97.9 tok/s vs 70.6 under identical settings), so we compare only within a single sweep. Second, at 32k decode is depressed by thermal throttling accumulated during the long prefill: the SSM's 32k within-cell trials decay 83 to 67 tok/s at flat ~10.1 GB memory, and an order-swap diagnostic (`src/diag_thermal_order.py`) shows the same throttling on dense KV-quant, where position in the sweep moves dense 32k decode from 60.1 (cold) to 44.3 (warm). That is enough to confound the bits penalty: across measurement states the 32k estimate ranges from about 53% (cold) to 9% (warm-steady), bracketing the single-sweep 65%, so it is not identifiable. At 8k the same diagnostic shows no position effect (a modest, run-variable penalty). We ran the diagnostic only at 8k and 32k, so the 16k KV-quant row is unverified (shown dashed in Figure 2): its tabulated direction is consistent with the sweep but not separately confirmed against the confound. We therefore report long-context decode results as *directional* (the SSM owns the decode axis, KV-quant lowers decode) and lean on the $O(1)$-per-step mechanism (flat SSM memory) rather than precise 32k/64k magnitudes or ratios.
4. **Quality probe is single-key NIAH, now checked on real prose.** Recall is measured with an exact-match single-key probe. To guard against a filler-specific artifact we re-ran it with real PubMed prose as the haystack (§5.2d): dense stays lossless (1.00 at both lengths, validating the probe) while the SSM deficit reproduces and runs cleanly monotonic in length (0.35 at 16k, 0.20 at 32k), so it is not an artifact of degenerate filler. A broader multi-task long-document suite (summarization, QA over natural documents) is still future work; the present claim is about single-fact storage capacity.
5. **Confound only partially resolved, and the attribution is Mamba-1-specific.** The same-lineage control identifies an architectural penalty at 32k (significant) but cannot decompose the *larger* off-the-shelf gap, which still confounds training corpus, instruct-tuning (the off-the-shelf Codestral is a code base model, not instruct-tuned like the dense and control models), Mamba version, and state size. The clean architectural finding holds for a Mamba-1 fixed-state SSM; it does not transfer automatically to the deployed Mamba-2, whose recall we measure but cannot decompose. We also do not claim the penalty grows with length: the within-control 16k to 32k decline (0.87 to 0.70) is non-significant (p=0.17), so we report emergence of a detectable penalty by 32k, not a measured growth rate. We label the unidentified remainder as such rather than overclaim.
6. **Probe geometry is fixed: single needle depth and single seed.** Every recall cell places the queried needle at or near the context midpoint and uses one fixed seed (3), so the haystack realization and needle positions are held constant and only the queried key varies across the n=30

trials. The Wilson intervals and Fisher tests therefore quantify query-sampling variability given a fixed haystack, not variability across haystack draws. Needle depth is a known sensitivity axis in NIAH work, and SSM recall is typically weakest away from the midpoint, so the reported deficits are plausibly a lower bound on the worst-case SSM penalty; depth- and seed-stratified probes are future work.

7 Conclusion

For on-device long-context inference on Apple Silicon, neither "subquadratic architecture" nor "compress the KV cache" is a free win. The SSM trades recall reliability for a flat memory and $O(1)$-per-step decode profile, taking the decode axis past the 4k-to-8k crossover, memory past ~21k, and prefill near ~64k (the last a single-run, directional crossover), but an off-the-shelf SSM's long-context recall can fall far below a dense model's. A same-lineage control locates an architectural component to that loss by 32k, while the full size of the off-the-shelf gap is also model-specific, so quality is not predicted by the architecture label alone. All crossover lengths are specific to the M5 Max (128 GB unified memory, 40 GPU cores); a device with a different compute-to-bandwidth ratio will shift these thresholds, though the axis ordering is expected to be portable.

KV quantization, the technique the field reaches for first, buys memory at the cost of decode speed on this stack (a stable, position-independent penalty at 8k; the long-context magnitude is thermally confounded), and it breaks recall at 2 bits; 4-bit is the usable aggressive setting and dominates 8-bit on memory and quality.

The practical rule we draw is to choose the long-context configuration empirically, per workload and per device: use the SSM (or 4-bit-KV dense) when the memory and throughput axis dominates and recall demands are modest, and keep lossless-KV dense when multi-fact recall is the binding constraint. Code, run logs, and result tables, including the uniform MLX harness and the synthesis pipeline, will be released publicly so these crossovers can be re-measured on other devices and scales.

8 Reproducibility

Harness modules in `src/` (all `.py`): `harness`, `sweep`, `sweep_ssm`, `decode_bench`, `niah_query`, `niah_real`, `kvquant`, `diag_thermal_order`, `niah_analyze`, `synthesize`; all run logs and per-trial JSONs in `results/`. Software: MLX 0.31.2, mlx_lm 0.31.3, on macOS / Apple M5 Max (18 CPU, 40 GPU cores, 128 GB unified memory); all models are the 4-bit `mlx-community` conversions. Stack versions are load-bearing here given the 9.3x kernel gap (§5.1), so we pin them. NIAH probes use a fixed seed (3) with K=8 stored facts and the queried needle at context midpoint; KV-quant runs use seed 1. Per-cell n: prefill (dense 512 to 32k n=3, 64k/128k n=2, Mamba-2 n=1); decode (n=5 at 1k to 4k for both models, n=3 at 8k to 64k, except the Mamba-2 32k cell re-run at n=5); single-key recall (n=30 at 16k/32k, n≤10 elsewhere), plus a dense 4-bit-KV single-key cell at 32k (n=30, seed 3, the §5.2 K=8 probe seed, not the §5.3 KV-quant seed 1) on the identical probe for the §5.4 same-protocol Pareto comparison; real-prose recall (n=20); KV-quant decode/memory n=5 (one sequential sweep over bit levels), KV-quant 2-bit recall n=20.

References

Simran Arora, Sabri Eyuboglu, Aman Timalsina, et al. Zoology: Measuring and improving recall in efficient language models. *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2312.04927.

Yushi Bai, Xin Lv, Jiajie Zhang, et al. LongBench: A bilingual, multitask benchmark for long context understanding. *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. arXiv:2308.14508.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. *International Conference on Machine Learning (ICML)*, 2024. arXiv:2405.21060.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *Conference on Language Modeling (COLM)*, 2024. arXiv:2312.00752.

Awni Hannun, Jagrit Digani, Angelos Katharopoulos, and Ronan Collobert. MLX: An array framework for apple silicon, 2023. github.com/ml-explore/mlx.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, et al. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2401.18079.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, et al. RULER: What's the real context size of your long-context language models? *Conference on Language Modeling (COLM)*, 2024. arXiv:2404.06654.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.01032.

Opher Lieber, Barak Lenz, Hofit Bata, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint*, 2024. arXiv:2403.19887.

Zirui Liu, Jiayi Yuan, Hongye Jin, et al. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.02750.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. *International Conference on Machine Learning (ICML)*, 2024. arXiv:2404.08819.

Technology Innovation Institute. The falcon 3 family of open models, 2024. Model cards at huggingface.co/tiiuae and huggingface.co/mlx-community.

Technology Innovation Institute. Falcon-H1: A family of hybrid-head language models redefining efficiency and performance. *arXiv preprint*, 2025. arXiv:2507.22448.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2309.17453.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, et al. H2O: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2306.14048.

Jingwei Zuo, Maksim Velikanov, et al. Falcon mamba: The first competitive attention-free 7B language model. *arXiv preprint*, 2024. arXiv:2410.05355.