ONE COMPRESSOR DOES NOT FIT ALL

Anonymous authors
Paper under double-blind review

ABSTRACT

The key–value (KV) cache dominates the memory cost of long-context inference, and many methods compress it. They are usually compared on one metric: how much task accuracy survives at a given compression ratio. Because compression is done for system efficiency, this comparison is underdetermined. Holding the model, GPU, prompts and execution stack fixed, we measure nine KV-cache configurations jointly on task score, GPU memory and latency across three benchmarks and two model scales. No configuration dominates, and the disagreement reaches the memory axis itself: attention-based eviction cuts decode-phase memory by 25% while raising whole-inference peak memory by 4.5%. Every LongBench task admits five to seven Pareto-optimal configurations, and a hindsight per-prompt oracle raises balanced utility from 0.668 to 0.714, with 87.7% of this gap left below task type. We then measure what a selector can capture from the prompt alone. Cross-fitted estimators show that the available request descriptions saturate at task granularity and that most per-prompt preference is tied to the executing model, while about 60% of the headroom concentrates in a detectable top decile. **KVRoute** is built on that structure: a detector over 46 tokenizer-derived features decides whether to route, a guarded ranker decides how, and refusal returns the best fixed compressor. Across three pre-specified transfer cohorts, it improves evaluation utility over FullKV by 2.82 points on average, with up to 38.2% less peak memory. This mean is dominated by ultra-long-context LongBench-v2, whose absolute 3B accuracy is below chance and which we therefore interpret as a resource-stress cohort rather than evidence of model competence. KVRoute limits its severe degradation to 1.6% of requests in its worst cohort, below every fixed compressor (4.5–72.4%), and decides in 14–30 ms on RULER and LongBench. Under this FullKV-relative utility and risk definition, dynamic selection combines compression efficiency with a low severe-loss rate.

1 INTRODUCTION

During autoregressive decoding, a transformer caches the key and value vectors of every processed token. This key–value (KV) cache accelerates generation, but it grows linearly with sequence length and dominates memory in long-context inference, often exceeding the model weights (Kwon et al., 2023; Liu et al., 2024b). Many training-free methods compress it: they evict tokens by position or attention (Li et al., 2024; Xiao et al., 2024; Cai et al., 2025; Devoto et al., 2024; Godey et al., 2025), quantize cache entries (Liu et al., 2024b; Hooper et al., 2024), prune layers (Zhang et al., 2024b), or merge layers (Liu et al., 2024a; Yang et al., 2024). Standard benchmarks compare them along one dimension: *how much task accuracy survives at a target compression ratio*.

Because compression is done for system efficiency, a single axis leaves the comparison underdetermined. A compressor must be judged jointly on task score, GPU memory and realized latency, and memory itself has two operational meanings: whole-inference peak memory (M_{peak}) governs admission and provisioning, decode-phase memory (M_{dec}) governs residency. Standard practice nevertheless commits to the compressor with the best benchmark average, although averages cannot separate algorithms once instance distributions vary (Wolpert & Macready, 1997; Wolpert, 1996) — and whether a discarded token matters depends on where the evidence lies. Compressor choice is therefore an *instance-level algorithm selection* problem (Rice, 1976; Kotthoff, 2014).

Evaluating nine configurations across three benchmarks, two model scales and three compression levels ($433{,}857$ matched triples; Fig. 1) establishes *persistent empirical Pareto non-domination*. The

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

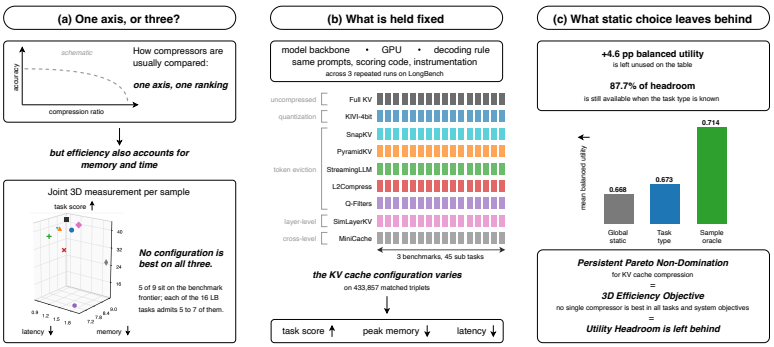


Figure 1: **Overview of the study.** **(a)** Joint 3D evaluation (↑ score, ↓ memory, ↓ latency): no compressor dominates. **(b)** Nine configurations, three benchmarks and three compression levels yield matched per-prompt triples. **(c)** 87.7% of the global-to-oracle gap lies below task type.

disagreement reaches the memory axis itself: observation-window eviction (SnapKV, PyramidKV) cuts decode-phase memory by about 25% while *raising* peak memory by 4.5%, replacing rather than reordering the non-dominated set. Every LongBench task admits five to seven Pareto-optimal options, and 87.7% of the utility headroom left by a global static choice lies below task type.

Exploiting this headroom before prefill is a representation problem: *what can a pre-inference view of the prompt encode about execution?* Prompt-side conditioning saturates at task granularity, task-level descriptors recover at most 20.2% of the headroom, and per-prompt preferences are tied to the executing model, because contrastive sentence encoders (Reimers & Gurevych, 2019; Wang et al., 2020; 2024) group requests by *subject matter* while compression sensitivity depends on *execution dynamics* such as evidence placement. The opportunity is nevertheless concentrated: the top decile of requests holds about 60% of the headroom and is detectable at task granularity (AUC 0.75–0.79).

We build **KVRoute** on this structure. A Stage-1 detector decides whether a request carries enough opportunity or risk to be routed at all; Stage 2 ranks actions under calibrated utility and quality guards. Our contributions are:

- **A controlled 3D benchmark protocol** over 433,857 matched triples, establishing persistent Pareto non-domination, the prefill/decode memory inversion, and the size of instance-level headroom (Sec. 3).
- **Attainability bounds** that measure the reachable part of this headroom instead of assuming it: the request descriptions we can form saturate at task granularity, per-prompt preference is largely model-bound, and the opportunity concentrates in a detectable decile (Sec. 4).
- **KVRoute**, a two-stage selector combining an opportunity-and-risk detector, 46 tokenizer features of question–context overlap in place of a sentence encoder (2.2–2.8× faster decisions, 14–30 ms median), and a *safe fallback*: refusals return the deployment's best fixed compressor rather than the uncompressed cache (Secs. 5–6).
- **A risk-budgeted evaluation against the complete static catalog.** KVRoute improves on the uncompressed cache by 2.82 utility points on average across the transfer cohorts while degrading at most 1.6% of requests severely, a lower tail than every compressor in the catalog (4.5–72.4%), making it the best admissible policy for severe-loss budgets between 1.6% and 4.5%. No selector, ours included, raises mean utility above the best fixed compressor (Sec. 6.2).

2 Related Work

KV-cache compression. Grouped-query attention (Shazeer, 2019; Ainslie et al., 2023) and paged memory (Kwon et al., 2023) shrink the cache by a constant factor, but it still grows linearly. Training-free compression spans five families: *token eviction* (Zhang et al., 2023; Xiao et al., 2024; Li et al., 2024; Chari & Van Durme, 2025; Goel et al., 2025; Liu et al., 2025b), *quantization* (Liu et al., 2024b; Hooper et al., 2024; Ashkboos et al., 2024; Yao et al., 2025), *layer reduction* (Zhang et al., 2024b), *cross-layer merging* (Liu et al., 2024a; Yang et al., 2024) and *low-rank factorization* (Singhania et al., 2024; Zhang et al., 2024a; Chang et al., 2026); adaptive variants reallocate budget across heads

or layers *during* the forward pass (Feng et al., 2025; Fei et al., 2026; Sun et al., 2026). Closest in spirit are systems that make compression workload-aware: DynamicKV adapts layer budgets to the task (Zhou et al., 2025), and serving systems profile a quality–memory–latency surface and control it online (Liu et al., 2026). We instead select *between* independently implemented compressors *before* execution, when no forward-pass state is observable. Every mechanism stays usable as released, and the design question shifts: what matters most is not which compressor a selector proposes, but which action it serves when it declines, since that action runs on most requests.

Multi-metric system audits. Audits show that kernel overheads can cancel memory savings (Gao et al., 2025), that eviction degrades reasoning non-uniformly (Liu et al., 2025a), and that time-to-first-token trades off with throughput (Yuan et al., 2024; Agrawal & Mayer, 2026; Gokhale et al., 2025). We evaluate reference implementations in one harness (Devoto et al., 2025) and separate M_{peak} from M_{dec}, showing that the memory accounting alone changes the Pareto frontier.

Algorithm selection, model routing and dynamic compute. Mapping instances to solvers with cheap features is the classical algorithm selection problem (Rice, 1976; Kotthoff, 2014; Xu et al., 2008), usually scored by the share of the hindsight gap a selector closes; we test whether that gap is reachable at all. Two neighbouring lines solve different problems. *Model routers* such as RouteLLM (Ong et al., 2024) dispatch a query across different backbones to trade monetary cost against a single quality axis. *Dynamic compute* changes execution inside one network — mixture-of-experts gating (Shazeer et al., 2017; Fedus et al., 2022), adaptive depth (Raposo et al., 2024), token pruning (Fu et al., 2024), speculative decoding (Leviathan et al., 2023) — trained into the model and acting below the request level. A KV-cache router keeps the backbone fixed and chooses one cache policy per request, before prefill, trading accuracy, peak memory and latency jointly, so neither line supplies an applicable action space.

Pre-inference request representations. Sentence encoders give cheap request vectors (Reimers & Gurevych, 2019; Gao et al., 2021; Wang et al., 2020; 2024; Nussbaum et al., 2024) and task embeddings transfer decisions across tasks (Achille et al., 2019), but their contrastive objectives place requests about the same subject together, whereas compression sensitivity depends on execution dynamics like evidence placement and question–context overlap, motivating KVRoute's features.

3 Empirical Diagnosis: Persistent Pareto Non-Domination

3.1 Evaluation Framework and Experimental Setup

Measurement and memory accountings. Let f be a pretrained model and $\mathcal{A} = \{a_0, \dots, a_K\}$ a catalog of KV-cache actions, a_0 being the uncompressed baseline (*FullKV*). Executing a on prompt x under batch-1 isolation yields a strictly paired triple

$$\big(A(x,a),\, M(x,a),\, L(x,a)\big) \in [0,100] \times \mathbb{R}^+ \times \mathbb{R}^+, \tag{1}$$

with A the official task score (↑), M peak GPU allocation in GiB (↓) and $L = L_{\text{prefill}} + L_{\text{decode}}$ realized latency in seconds (↓, CUDA-event synchronized). We instrument both phases,

$$M_{\text{peak}}(x,a) = \max\big(M_{\text{prefill}},\, M_{\text{decode}}\big), \qquad M_{\text{dec}}(x,a) = M_{\text{decode}}, \tag{2}$$

where M_{peak} bounds admission and provisioning and M_{dec} proxies sequence residency during decoding (App. B.1).

Pareto dominance and utility scalarization. Action a *dominates* a' ($a \succ a'$) iff it is at least as good on all three axes and strictly better on one; the non-dominated set is the *Pareto frontier* $\mathbb{P} \subseteq \mathcal{A}$, and a universally superior compressor would give $|\mathbb{P}| = 1$. To model operational priorities, each metric is min–max rescaled to $[0,1]$ against the extrema of its task t over all prompts and actions and weighted by λ on the simplex Δ^2:

$$\hat{A} = \frac{A - A_t^{\min}}{A_t^{\max} - A_t^{\min}}, \quad \hat{M} = 1 - \frac{M - M_t^{\min}}{M_t^{\max} - M_t^{\min}}, \quad \hat{L} = 1 - \frac{L - L_t^{\min}}{L_t^{\max} - L_t^{\min}}, \tag{3}$$

$$U_\lambda(x,a) = \lambda_A \hat{A}(x,a) + \lambda_M \hat{M}(x,a) + \lambda_L \hat{L}(x,a), \tag{4}$$

Table 1: **No configuration is best on all three axes.** LongBench at $\ell = L_3$, sorted by balanced utility U_λ (Eq. 4, $\times 100$). M_{peak}: whole-inference peak; M_{dec}: decode-phase peak; saved/added: relative change of means vs. FullKV; Par: tasks (of 16) on which the row is non-dominated, the two reference policies scored against the nine fixed configurations.

Configuration	$A \uparrow$	$M_{\text{peak}} \downarrow$	saved%	$M_{\text{dec}} \downarrow$	saved%	$L \downarrow$	added%	$\text{Par}_{\text{pk}} \uparrow$	$\text{Par}_{\text{dec}} \uparrow$	$U_\lambda \uparrow$
Sample oracle† ($a^\star(x)$)	47.6	8.12	+12.4	6.90	+17.7	1.774	-1.2	16	16	71.4
Task-type† ($a_{\text{type}}(g)$)	41.2	8.32	+10.3	6.79	+19.0	1.901	+5.9	16	11	67.3
StreamingLLM (a_{global})	38.0	8.22	+11.3	6.89	+17.8	1.807	+0.7	16	13	**66.8**
KIVI-4bit	42.4	8.26	+10.9	6.72	+19.7	3.125	+74.1	15	11	64.6
FullKV	**42.5**	9.27	+0.0	8.38	+0.0	**1.795**	+0.0	16	16	64.5
SimLayerKV	41.7	9.03	+2.6	8.21	+2.0	2.639	+47.0	14	8	62.1
SnapKV	40.0	9.69	-4.5	**6.26**	+25.3	1.959	+9.1	3	16	60.7
PyramidKV	40.4	9.71	-4.7	6.28	+25.1	2.027	+12.9	2	13	60.7
L2Compress	30.3	8.76	+5.5	7.52	+10.3	2.161	+20.4	9	6	60.6
Q-Filters	11.2	**7.35**	+20.7	6.35	+24.2	3.610	+101.1	16	4	52.8
MiniCache	26.2	9.30	-0.3	8.49	-1.4	3.822	+112.9	0	1	49.6

† marks retrospective, undeployable reference ceilings.

with $\lambda = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ defining *balanced utility*. Three retrospective granularities give diagnostic bounds: a **global static** action a_{global}, a **task-type** action $a_{\text{type}}(g)$ per task family g, and a hindsight **sample oracle** $a^\star(x) = \arg\max_a U_\lambda(x, a)$.

Workloads and action pool. The primary benchmark is LongBench (Bai et al., 2024) (16 tasks, 3,200 prompts, 3 runs) on Llama-3.2-3B-Instruct (Meta, 2024b) and Llama-3.1-8B-Instruct (Meta, 2024a); RULER-4096 (Hsieh et al., 2024) and LongBench-v2 (Bai et al., 2025) add other context regimes. All executions use `float16`, greedy decoding and batch-1 isolation on one NVIDIA H100 80GB GPU, with reference implementations in one harness (Devoto et al., 2025); L_3 is the most aggressive of three compression levels (75% evicted, or a 1,024-token budget). The pool $\mathcal{A}$ holds nine configurations from five families: FullKV, quantization (KIVI-4bit), attention or positional eviction (SnapKV, PyramidKV, StreamingLLM, L2Compress, Q-Filters), layer reduction (SimLayerKV) and cross-layer merging (MiniCache) (App. A.1). Across $\ell \in \{L_1, L_2, L_3\}$ this yields 433,857 matched triples (Tab. 4).

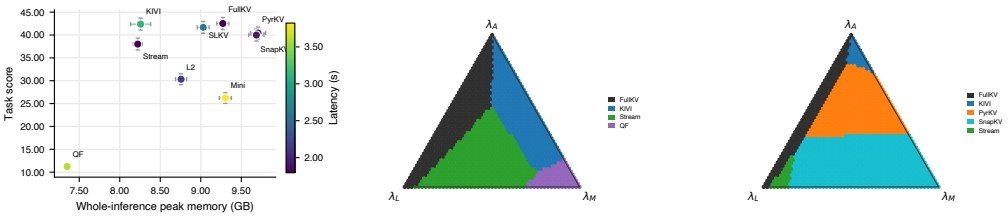
3.2 Multi-Dimensional Trade-Offs and Non-Domination

Evaluated jointly on score, peak allocation and latency, no configuration dominates: five of nine lie on the M_{peak} frontier (Tab. 1). FullKV has top accuracy (42.5) and fastest execution (1.795 s) at 9.27 GiB. KIVI-4bit matches its accuracy (42.4; equivalent within ± 0.5 points by TOST at a margin smaller than FullKV's gap to every other configuration, App. B.5) while cutting peak allocation by 10.9% and decode allocation by 19.7% at +74.1% latency. StreamingLLM saves 11.3% peak memory at +0.7% latency but forfeits 4.5 points, and Q-Filters holds the memory minimum on all 16 tasks (20.7%) and stays non-dominated despite a score of 11.2.

Accuracy alone fails to separate candidates: the top two differ significantly on 1 of 16 tasks, whereas system contrasts resolve at $p < 10^{-7}$. Per task, five to seven configurations are non-dominated (mean 5.69), and the balanced-utility winner differs from the most accurate on 13 of 16 tasks. Non-domination is robust to operating points, latency cost models and bootstrapping (App. B.3–B.5).

3.3 The Prefill/Decode Memory Accounting Inversion

Whole-inference peak allocation is prefill-dominated (98.2% of prompts): it measures the largest transient created while *constructing* the cache, not the size retained. Quantization and positional eviction compress during prefill and reduce both quantities, whereas observation-window methods (SnapKV, PyramidKV) score attention over the entire prompt and materialize intermediate buffers before discarding tokens. They therefore *increase* whole-inference peak memory by 4.5% and 4.7% while cutting decode-phase memory by 25.3% and 25.1%, the largest decode reductions above the weight floor. Changing the axis from M_{peak} to M_{dec} *replaces* the frontier: SnapKV and PyramidKV join, Q-Filters drops off, and seven of nine configurations are Pareto-optimal under at least one accounting. Reporting a single memory scalar leaves compression evaluations underdetermined.



(a) Joint 3D trade-off. (b) Winner under M_{peak}. (c) Winner under M_{dec}.

Figure 2: **3D trade-offs and preference sweeps on LongBench (3B). (a)** Score (↑) vs. peak memory (↓), latency (↓) in color. **(b, c)** Optimal configuration across $\lambda \in \Delta^2$: switching the memory accounting replaces the winning partition.

3.4 Deployment Priority and Simplex Partitioning

Sweeping λ over Δ^2 shows the optimal choice shifting with operational goals (Fig. 2). Under M_{peak} four configurations claim regional wins — StreamingLLM (38.9%), KIVI-4bit (28.0%), FullKV (26.4%), Q-Filters (6.7%) — with StreamingLLM winning the balanced point ($U_\lambda = 0.668$). Under M_{dec} the simplex is re-partitioned: SnapKV (50.2%) and PyramidKV (32.8%) capture 83% of the preference space. Tasks also disagree with each other: across 99.3% of weightings, constituent tasks select multiple distinct optima (mean 3.07; App. C.2).

3.5 Instance-Level Headroom and Physical Drivers

Retrospective granularities show how much universal static selection forfeits: global static choice reaches $U_\lambda = 0.668$, coarse task type 0.673 (+0.0056, CI $[0.0028, 0.0086]$), the hindsight sample oracle 0.714 (+0.0457, $[0.0429, 0.0487]$). Thus 87.7% **of the available utility headroom remains uncaptured beneath coarse task type**. No compressor is optimal on a majority of prompts (StreamingLLM 34.1%, KIVI-4bit 27.1%), and all nine win on specific subsets (App. C.3).

Physical drivers. The axes follow uncoupled properties of the input, all three varying prompt by prompt. *Memory* scales with context volume and prefill transients: from the shortest to the longest prompt quartile, StreamingLLM's decode savings grow from 7.7% to 26.3%, while SnapKV moves from a 9.1% saving to a 12.9% penalty. *Latency* follows the intervention locus: prefill-time eviction is nearly free (+0.7%), per-step work is not (+74.1%). *Accuracy* follows evidence placement: positional eviction is best on local code (`lcc`: 39.2 vs. 35.6) but collapses on mid-prompt retrieval (`multifieldqa_en`: 30.8 vs. 52.4), where attention eviction holds (51.4).

Replication. Multi-membered frontiers recur on every other cohort: RULER-4096, LongBench-v2, and LongBench at 8B (App. C.4, App. C.5). The share of headroom below task type is 87.7% on LongBench 3B, 93.5% on LongBench 8B, 86.3% on LongBench-v2 and 64.3% on RULER, whose synthetic tasks are the most homogeneous.

4 What a Prompt Can and Cannot Tell You

To exploit instance headroom online, a selector must commit before prefill, seeing only the prompt and the output limits. This section fixes the two quantities an operator judges such a policy on — utility against the uncompressed cache, risk against the compressors — and measures how much headroom a pre-inference signal can reach.

4.1 Evaluation Utility, Baselines and Risk

For prompt x and action $a \in \mathcal{A}$, the evaluation utility is

$$U(x,a) = \tfrac{1}{3}\left[\frac{A(x,a)}{100} + \text{clip}\left(1 - \frac{L(x,a)-L_{\min}}{L_{\max}-L_{\min}}\right) + \text{clip}\left(1 - \frac{M(x,a)-M_{\min}}{M_{\max}-M_{\min}}\right)\right], \tag{5}$$

with $M = M_{\text{peak}}$ and clipping to $[0, 1]$; differences are reported in *points* (10^{-2}). It differs from Eq. 4 in three ways: task scores enter on their native scale, out-of-range values are clipped, and the bounds belong to a cohort rather than a task. We compute $L_{\min}, L_{\max}, M_{\min}, M_{\max}$ once per cohort over all nine actions and apply them identically to every policy. They are retrospective, hence *evaluation* utility: a fixed yardstick applied after the fact, not computed online (selectors fit targets under bounds frozen on their training fold; App. D.3). Decision cost is added to L, giving *net* utility.

Two baselines, one per axis. A compression policy is asked for two things at once. *Utility* is measured against FullKV, what a cautious deployment runs today. *Risk* is measured against the compressors, as the *severe degradation rate* — the share of requests losing more than 5 task-score points relative to FullKV — which every compressor has. We also track the strongest single compressor $a_{\text{ref}} = \arg\max_a \mathbb{E}[U(x, a)]$, which is **KIVI-4bit** on all four cohorts (under Eq. 4's task-wise bounds the balanced winner is StreamingLLM; App. D.1). It matters twice over: it is the bar a selector must clear to justify itself as a compressor, beating FullKV by 0.35, 0.54, 5.70 and 10.51 points on LongBench 3B, LongBench 8B, RULER and LongBench-v2, and it is the action a selector should return when it declines, since reverting to FullKV forfeits that margin on every refused request.

Routing headroom. The hindsight per-request policy exceeds a_{ref} by

$$H = \mathbb{E}\big[\max_a U(x, a)\big] - \mathbb{E}\big[U(x, a_{\text{ref}})\big], \tag{6}$$

i.e. 3.17, 2.53, 2.85 and 5.04 points. Measured from FullKV instead, the same policy gains 3.52, 3.07, 8.55 and 15.55 points; Section 6.2 reports what fraction a deployable selector captures.

4.2 Reachability Estimators

The hindsight oracle reads realized outcomes, so variation in the best action that is not a function of the request is unreachable, however good the predictor. Three cross-fitted estimators bound the reachable part (App. D.2). The **conditioning ladder** maps each request to a group through a descriptor d (length decile, task) and allows one action per group:

$$\text{Rec}(d) = \frac{V(d) - \mathbb{E}[U(x, a_{\text{ref}})]}{H} \times 100\%, \qquad V(d) = \mathbb{E}\big[U(x, a_{d(x)})\big], \tag{7}$$

where each group's action is fitted on four folds and scored on the fifth, folds split by context hash. Grouping by nothing recovers 0%, grouping by the request itself 100%; where the curve flattens is where finer description stops paying. **Noise-corrected hindsight** selects $a^\star(x)$ on one LongBench run and scores it on the others (the correction changes H by well under 1%; App. B.4). The **cross-scale check** asks whether 8B per-prompt optima predict 3B optima on identical prompts.

4.3 Findings: Saturation, Concentration, and Where the Risk Sits

Conditioning saturates at task granularity. With five group-respecting folds, length deciles recover 5.8% and 6.4% of H on the two LongBench cohorts and nothing elsewhere, task types reach 11.2%, 7.8% and 20.2% on LongBench 3B, 8B and RULER, and the ground-truth task label, which a served request does not carry, 9.0%, 15.4% and 20.2%. No descriptor exceeds 20.2% of H, and refining past the task loses ground (2.4% and 5.3% for task $\times$ length decile) as groups become too small to estimate preference (Tab. 24). The claim is narrow: *the descriptors we can currently form* run out of information at task granularity, not prompt-side conditioning in general. We attribute the ceiling to the representation rather than predictor capacity, since contrastive encoders (Reimers & Gurevych, 2019; Wang et al., 2020) cluster text by subject matter while compression sensitivity depends on execution dynamics; a learned representation would test that (App. F).

Preferences are model-bound. Routing the 3B model with the 8B per-prompt optima recovers -12.9% of H, i.e. 0.41 points *below* the best fixed action (App. D.2): preference is a prompt–model interaction, not a property of the prompt. Restricted to the high-opportunity minority the transfer is positive, so the structure exists but does not survive averaging over requests with nothing at stake.

Opportunity concentrates in a detectable decile. Deviating from a_{ref} only on the top decile recovers 57.7–65.4% of H on all four cohorts, several times what any descriptor reaches, and that decile is easier to locate than to act on: separable at task granularity at AUC 0.75–0.79, against near-chance from input length.

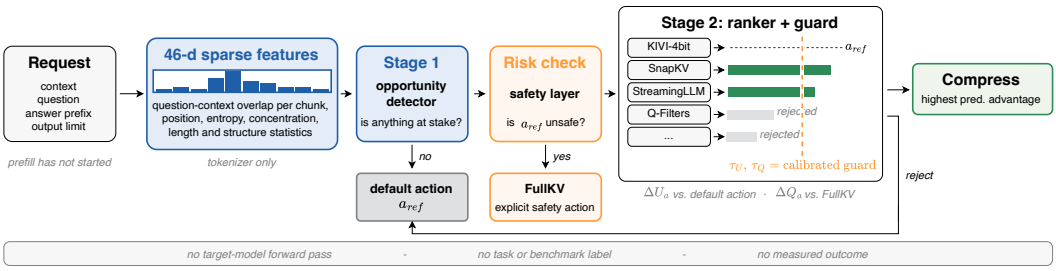


Figure 3: **KVRoute decides whether a request is worth routing before deciding how.** A Stage-1 detector over 46 tokenizer features gates the ranker; low-opportunity requests immediately receive a_{ref} (KIVI-4bit). Routed requests pass a default-risk check and calibrated guards.

Risk is concentrated too, and asymmetric. Compression failures are not graceful: a misapplied compressor costs more than 5 task-score points on 4.5% to 72.4% of requests in its worst cohort, and even a_{ref}, the mildest in the pool, fails this way on 4.5%. A ranking mistake is cheap by comparison (3.20 of nine actions lie within 0.01 utility of the optimum on a typical LongBench request; App. B.4). The expensive error is not picking the second-best action but missing a request where compression breaks.

Design implications. A prompt reveals *where* compression matters, not *which* compressor to use, and a wrong deviation costs far more than a missed one. KVRoute follows all three findings: detect high-opportunity and high-risk requests before ranking, measure execution-relevant overlap instead of semantic similarity, and treat the refusal action and the quality floor as calibrated safety decisions rather than by-products of the ranker.

5 KVROUTE: ADAPTIVE PRE-INFERENCE KV-CACHE SELECTION

KVRoute (Fig. 3) turns the findings of Section 4 into an online selector, and it is built for both axes at once: every stage is a filter that must be passed before the policy is allowed to deviate from a safe action, so utility is gained only where the evidence supports it. Cache allocation happens before prefill, so KVRoute decides before the first forward pass, seeing only prompt text, token counts and generation limits, with no access to attention maps, hidden states or outcomes.

46-d execution representation. KVRoute uses only the target model's tokenizer. Context tokens are split into ten equal, position-ordered chunks, and for each chunk we compute the fraction of question tokens it contains. The 46 features are these 10 overlap values, 8 shape statistics of their profile (maximum, entropy, argmax position, tail-minus-head concentration, ...), 3 head/tail proxies, 8 length and budget terms, 7 question-complexity and 9 context-structure indicators, and a model index (App. D.3); extraction is linear in context length.

Stage 1: opportunity and risk detector. A gradient-boosted classifier predicts membership in the union of (i) the top decile of headroom over a_{ref} and (ii) requests on which a_{ref} loses more than 5 score points against FullKV, the second set keeping the detector from being blind where the default itself is risky. Detector-negative requests get a_{ref} immediately.

Stage 2: guarded ranker with default-risk check. For detector-positive requests, one multi-output regressor predicts for every action

$$\Delta U_a(x) = U(x,a) - U(x,a_{\text{ref}}), \qquad \Delta A_a(x) = A(x,a) - A(x,a_0), \tag{8}$$

the net utility gain over the default and the absolute quality change against FullKV, which fixes a constant quality floor. First, a *default-risk check* selects FullKV explicitly if $\widehat{\Delta A}_{a_{\text{ref}}} < -\tau_{\text{risk}}$. Otherwise, $a^* = \arg\max_a \widehat{\Delta U}_a$ is served only if

$$\widehat{\Delta U}_{a^*}(x) \geq \tau_U \qquad \text{and} \qquad \widehat{\Delta A}_{a^*}(x) \geq -\tau_A. \tag{9}$$

Table 2: **Utility–risk comparison on the three transfer cohorts.** ΔU: net utility vs. FullKV in points, router cost charged. Sev.: % of requests losing >5 score points vs. FullKV; Worst: maximum over cohorts. Router: median latency on RULER. In-sample LongBench 3B in Tab. 33.

	RULER 3B		LongBench-v2 3B		LongBench 8B		Worst	Router
Policy	$\Delta U \uparrow$	Sev.↓	$\Delta U \uparrow$	Sev.↓	$\Delta U \uparrow$	Sev.↓	Sev.↓	ms↓
Fixed actions								
FullKV (a_0)	0.00	0.0	0.00	0.0	0.00	0.0	0.0	–
KIVI-4bit (a_{ref})	**+5.70**	1.3	**+10.51**	1.8	**+0.54**	4.5	4.5	–
SimLayerKV	-1.75	13.7	+4.35	2.2	-0.59	10.8	13.7	–
StreamingLLM	-10.46	62.6	+6.71	2.4	-0.14	25.8	62.6	–
SnapKV	-13.18	18.1	-2.68	4.0	-1.62	18.1	18.1	–
PyramidKV	-17.30	25.8	-2.40	2.8	-1.52	17.2	25.8	–
L2Compress	-20.55	66.2	+2.30	12.1	-1.40	30.9	66.2	–
Q-Filters	-17.67	53.0	-1.84	13.3	-5.62	45.8	53.0	–
MiniCache	-27.99	72.4	-8.78	15.5	-3.10	31.6	72.4	–
Selectors								
Random	-11.45	34.8	-0.04	7.8	-1.55	19.9	34.8	–
OutcomeForest	+2.41	1.4	+3.96	1.0	+0.77	4.9	4.9	48.3
OutcomeForest+Guard	+2.42	1.2	+3.76	1.0	+0.65	2.7	2.7	50.7
KVRoute-RankOnly	+1.04	0.4	-1.60	**0.0**	+0.19	3.1	3.1	33.9
KVRoute-Encoder	+0.27	0.2	+0.21	0.2	+0.07	2.8	2.8	38.6
KVRoute	-0.22	**0.0**	+8.59	1.6	+0.10	**1.4**	**1.6**	**13.8**

Default-action anchoring. Requests failing guards, detector-negative requests, and malformed inputs fall back to a_{ref}, preserving the margin of Section 4.1 on the majority of unrouted requests. FullKV is served only when the default-risk check chooses it, as one of the nine catalog actions: the uncompressed cache is a deliberate safety action, not where the policy lands by accident.

Safety calibration. Thresholds (τ_U, τ_A, τ_{risk}, detector quantile) are calibrated on out-of-fold predictions: settings are allowed if mean quality loss against FullKV is ≤ 0.5 points and $\leq 1\%$ of requests lose more than 5; among these we maximize net utility, then minimize severe loss (App. D.3).

6 Routing Experiments

6.1 Experimental Setup

Every learned selector is trained on the $3{,}200$ LongBench 3B requests, with grouped out-of-fold predictions used only for model selection and calibration; the final selector is refit on all and frozen (App. A.2), so its LongBench 3B replay is in-sample and appears only in the appendix. The main comparison uses three transfer cohorts no selector saw: RULER-4096 and LongBench-v2 bring unseen prompts, tasks and context lengths, LongBench 8B the outcomes of an unseen model (Tab. 5).

Policies compared (App. D.3). Routing is judged against *every* fixed action, not against FullKV alone. RANDOM samples uniformly; OUTCOMEFOREST predicts quality, memory and latency from sentence embeddings and +GUARD adds a quality guard, both falling back to FullKV; KVROUTE-RANKONLY ranks and guards over a $1{,}936$-d MiniLM representation (Wang et al., 2020), also falling back to FullKV; KVROUTE-ENCODER uses that representation with the a_{ref} fallback, isolating the representation. Router latency is charged in net utility, and risk is the *severe degradation rate*: the share of requests losing more than 5 score points against FullKV.

6.2 Main Results

Against the uncompressed cache, routing delivers. Every guarded selector improves on FullKV on most cohorts, and KVRoute does so while touching the fewest requests: $+8.59$ points on LongBench-v2, $+0.10$ on LongBench 8B and -0.22 on RULER, a mean of $+2.82$ (Tab. 2), i.e. 55% of the hindsight oracle's improvement over FullKV on LongBench-v2 and 3% on LongBench 8B — concentrated, like the headroom itself, where the choice is expensive. LongBench-v2 is the extreme-

context regime where routing matters most: FullKV peaks above 31 GiB, and KVRoute serves a_{ref} on 90.9% of requests for 38.2% less peak memory, about $1.7\times$ the next best selector. Every policy replays the same measured outcome matrix, so that comparison is exact; what the cohort does not support is an absolute reading of the 3B backbone's accuracy (App. F).

Against the compressors, KVRoute has the mildest tail. Mean utility and tail risk order the catalog differently. a_{ref} has the highest mean utility on RULER and LongBench-v2, and the other seven compressors buy utility with severe-loss rates of 13.7–72.4% in their worst cohort. KVRoute's worst cohort is 1.6%: lower than every compressor, a_{ref} at 4.5% included, and every other learned selector (2.7–4.9%). It keeps compression's savings where evidence says they are safe and reverts to the uncompressed cache where it does not. The only selector margins above a_{ref} are three on LongBench 8B, at most 0.23 points and all from policies at or above 2.7% severe loss; KVRoute stays 0.45, 1.93 and 5.92 points below it.

Routing pays inside a risk budget. A budget makes the two axes commensurable: admit only policies whose severe-loss rate stays below B on *all three* transfer cohorts, then rank them by mean ΔU (Tab. 36). At $B \leq 1\%$ only the uncompressed cache qualifies, KVRoute's own worst cohort being 1.6%. From 1.6% to 4.5%, the worst cohort of a_{ref}, KVRoute is the best admissible policy of any kind: $+2.82$ mean points ($+8.59$, $+0.10$, -0.22) against 0.00 for the uncompressed cache, the only alternative at that risk level. Above 4.5%, a_{ref} is admissible again and wins at $+5.59$. The interval moves with the severity definition: 2.3–9.8% at >2 points, and narrowing to 1.6–1.8% at >10, but the window exists under every definition we test.

RULER: a safe failure. On RULER the default-risk check, fitted on LongBench only, fires on almost every request: KVRoute serves FullKV on 96% of them and ends at -0.22 points with one severely degraded request (0.02%): transfer error becomes forgone savings, not quality loss. Its Stage-1 detector shows the same boundary, reaching AUC 0.960 in distribution against 0.503 and 0.574 on LongBench 8B and RULER (App. E.3): what fails to transfer is the mapping from a prompt to how it will execute, exactly as Section 4.3 predicts. Recalibrating τ_{risk} on the target workload is the remedy.

Robustness and cost. Re-scoring the frozen decisions of eight policies under ten normalisations crossed with 67 weight vectors on Δ^2 leaves both halves intact: KVRoute's gain over the uncompressed cache and its tail advantage hold throughout, and no policy beats the best fixed compressor in mean utility on all three transfer cohorts under any combination. KVRoute also decides fastest (13.8 ms median on RULER, 29.5 ms on LongBench 8B, 2.2–$2.8\times$ below KVROUTE-ENCODER, 277 ms at 100k tokens), and its shortfall against the oracle is dominated by declined opportunity, the unguarded OUTCOMEFOREST's by wrong actions (App. E.2, App. E.3).

7 CONCLUSION

Single-metric evaluation of KV-cache compression is underdetermined. Jointly measured on task quality, two memory accountings and realized latency, compressors show *persistent empirical Pareto non-domination*, while 87.7% of the headroom left by a global static choice lies below task type. We then measured what a pre-inference signal can reach: prompt descriptions saturate at task granularity, preferences are model-bound, and opportunity concentrates in a detectable decile.

KVRoute operationalizes these findings as a risk-controlled compression policy. Across three prespecified transfer cohorts, it gains 2.82 evaluation-utility points over FullKV on average, with up to 38.2% less peak memory; most of this mean comes from LongBench-v2, which we treat as a resource-stress cohort because the 3B backbone scores below chance there. KVRoute limits severe degradation to 1.6% of requests in its worst cohort, below every fixed compressor (4.5–72.4%) and every other selector, and is therefore the best admissible policy for risk budgets $1.6\% < B \leq 4.5\%$. No selector beats the best fixed compressor in mean utility on all three transfer cohorts under any of ten normalisations. Per-request selection is thus supported as a way to buy compression-level efficiency at a bounded FullKV-relative tail, not yet as a mean-utility improvement; closing that gap requires representations that encode execution rather than meaning. Due to space restrictions, limitations are discussed in Appendix F.

Ethics Statement

This work studies inference-time KV-cache compression and its per-request selection for publicly released language models on public, openly licensed benchmarks (LongBench, LongBench-v2, RULER). It involves no human subjects, personal data or newly collected text, and it introduces no new compression mechanism. Reducing the memory cost of long-context inference lowers serving cost in general, which is dual-use in the ordinary sense. More specific to this work, compression changes outputs non-uniformly: a configuration that looks accuracy-neutral on average can still fail badly on a minority of requests, and an operator who monitors only aggregate quality will not see it. We therefore report the severe-degradation rate next to every aggregate routing result, and KVRoute treats it as a hard constraint during calibration rather than as a diagnostic reported afterwards. We do not evaluate safety properties such as refusal behaviour or bias; auditing is advisable before deploying compressed caches in safety-critical settings. Energy and carbon were not measured.

Reproducibility Statement

All numbers in this paper are computed from stored per-request execution logs rather than from fresh model runs. Section 3.1 and Appendix A.1 give the hardware, model, harness, every compressor setting and the compression-level mapping. Greedy decoding makes task scores bit-identical across repeated runs.

Because every request–action pair was executed in advance, every selector, reference policy and attainability estimate in this paper is a replay against a fixed matrix of $433{,}857$ measured triples: a policy emits an action per request and the corresponding recorded outcome is read. Reproducing the routing study therefore requires no target-model execution.

We supply that matrix and an executable verification of the paper's numbers. The supplementary material contains the diagnostic measurements, the per-request selector decisions with the normalised nine-action catalog, and a notebook that regenerates the tables *and asserts the published values*, failing if any recomputed figure disagrees. Two exclusions are deliberate: the cross-fitted rows of Table 24 require the fold assignment and out-of-fold predictions, which the released matrices do not contain, and Appendices D.3–E.1 specify mechanisms and hyper-parameters rather than measured rows.

Two conventions govern every reported number and are enforced in the released code. Diagnostic results use the task-wise normalisation of Eq. 4; routing results use the frozen cohort bounds of Eq. 5, and utilities under different bounds are never compared. LongBench 3B selector rows are in-sample replays of the final models, so all transfer claims rest on RULER-4096, LongBench-v2 and LongBench 8B. Appendix D.1 gives the utility and its bounds, Appendix D.2 the attainability estimators and cross-fitting, Appendix D.3 every selector with its features, hyper-parameters, calibration grid and seeds, Appendix A.2 the input allowlist and grouping rule, and Appendix E.1 the timing protocol. Bootstrap intervals use $2{,}000$ resamples (seed 2027 for the diagnosis); all selector seeds are 42, and the sentence encoder used by the baselines is pinned by revision. The selector training code and the fold assignments will be released upon publication.

AI Assistance Statement

LLMs were used solely as writing aids to assist with grammar polishing, improving prose readability, and shortening text to fit page limits. All core conceptual ideas, technical formulations, and analyses were conceived, verified, and finalized by the human authors, who take full responsibility for the contents of this manuscript.

References

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C. Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6430–6439, 2019.

Nikita Agrawal and Ruben Mayer. Benchmarking KV-cache optimizations across task quality and system performance for long-context serving, 2026. URL https://arxiv.org/abs/2607.05399.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 100213–100240. Curran Associates, Inc., 2024. doi: 10.52202/079017-3180. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/b5b939436789f76f08b9d0da5e81af7c-Paper-Conference.pdf.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.acl-long.172/.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.183. URL https://aclanthology.org/2025.acl-long.183/.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling, 2025. URL https://arxiv.org/abs/2406.02069.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Hung-Yueh Chiang, Yash Akhauri, Xilai Dai, Huiqiang Jiang, Yucheng Li, Luis Ceze, Kai-Chiang Wu, and Mohamed S. Abdelfattah. xkv: Cross-layer kv-cache compression via aligned singular vector extraction, 2026. URL https://arxiv.org/abs/2503.18893.

Vivek Chari and Benjamin Van Durme. Compactor: Calibrated query-agnostic kv cache compression with approximate leverage scores, 2025. URL https://arxiv.org/abs/2507.08143.

Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective l_2 norm-based strategy for kv cache compression, 2024. URL https://arxiv.org/abs/2406.11430.

Alessio Devoto, Maximilian Jeblick, and Simon Jégou. Expected attention: Kv cache compression by estimating attention from future queries distribution, 2025. URL https://arxiv.org/abs/2510.00636.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL http://jmlr.org/papers/v23/21-0998.html.

Chao Fei, Kaihua Liang, Hanzhi Hu, Hongcheng Guo, Jian Weng, Marco Canini, and Panos Kalnis. Exploring a layer-wise design space for kv cache eviction, 2026. URL https://arxiv.org/abs/2606.15157.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113152–113188. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a40ff56daab9f4808b1e18350c8a11ce-Paper-Conference.pdf.

Qichen Fu, Minsik Cho, Thomas Merth, Sachin Mehta, Mohammad Rastegari, and Mahyar Najibi. Lazyllm: Dynamic token pruning for efficient long context llm inference, 2024. URL https://arxiv.org/abs/2407.14057.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021. URL https://aclanthology.org/2021.emnlp-main.552/.

Wei Gao, Xinyu Zhou, Peng Sun, Tianwei Zhang, and Yonggang Wen. Rethinking key-value cache compression techniques for large language model serving. In *Proceedings of Machine Learning and Systems (MLSys)*, 2025. URL https://arxiv.org/abs/2503.24000. arXiv:2503.24000.

Nathan Godey, Alessio Devoto, Yu Zhao, Simone Scardapane, Pasquale Minervini, Éric de la Clergerie, and Benoît Sagot. Q-filters: Leveraging qk geometry for efficient kv cache compression, 2025. URL https://arxiv.org/abs/2503.02812.

Raghavv Goel, Junyoung Park, Mukul Gagrani, Dalton Jones, Matthew Morse, Harper Langston, Mingu Lee, and Chris Lott. Caote: Kv cache selection for llms via attention output error-based token eviction, 2025. URL https://arxiv.org/abs/2504.14051.

Sai Gokhale, Devleena Das, Rajeev Patwari, Ashish Sirasao, and Elliott Delaye. KV pareto: Systems-level optimization of KV cache and model compression for long context inference, 2025. URL https://arxiv.org/abs/2512.01953.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 1270–1303. Curran Associates, Inc., 2024. doi: 10.52202/079017-0040. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/028fcbcf85435d39a40c4d61b42c99a4-Paper-Conference.pdf.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling (COLM)*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Lars Kotthoff. Algorithm selection for combinatorial search problems: A survey. *AI Magazine*, 35 (3):48–60, 2014. doi: 10.1609/aimag.v35i3.2460.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970. Curran Associates, Inc., 2024. doi: 10.52202/079017-0722. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/28ab418242603e0f7323e54185d19bde-Paper-Conference.pdf.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. Minicache: Kv cache compression in depth dimension for large language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 139997–140031. Curran Associates, Inc., 2024a. doi: 10.52202/079017-4443. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/fd0705710bf01b88a60a3d479ea341d9-Paper-Conference.pdf.

Minghui Liu, Aadi Palnitkar, Tahseen Rabbani, Hyunwoo Jae, Kyle Rui Sang, Dixi Yao, Shayan Shabihi, Fuheng Zhao, Tian Li, Ce Zhang, Furong Huang, and Kunpeng Zhang. Hold onto that thought: Assessing KV cache compression on reasoning, 2025a. URL https://arxiv.org/abs/2512.12008.

Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Liuyue, Bo Li, Xuming Hu, and Xiaowen Chu. Chunkkv: Semantic-preserving kv cache compression for efficient long-context llm inference. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 28728–28778. Curran Associates, Inc., 2025b. doi: 10.52202/085713-0966. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/2987f911151b39cd3a1761e212319e8e-Paper-Conference.pdf.

Zedong Liu, Xinyang Ma, Dejun Luo, Hairui Zhao, Bing Lu, Wenjing Huang, Yida Gu, Xingchen Liu, Zheng Wei, Jinyang Liu, Dingwen Tao, and Guangming Tan. Kvserve: Service-aware kv cache compression for communication-efficient disaggregated llm serving, 2026. URL https://arxiv.org/abs/2605.13734.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache, 2024b. URL https://arxiv.org/abs/2402.02750.

Meta. Llama-3.1-8b-instruct. https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct, 2024a.

Meta. Llama-3.2-3b-instruct. https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct, 2024b.

Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed: Training a reproducible long context text embedder, 2024. URL https://arxiv.org/abs/2402.01613.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data, 2024. URL https://arxiv.org/abs/2406.18665.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models, 2024. URL https://arxiv.org/abs/2404.02258.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019. URL https://aclanthology.org/D19-1410/.

John R. Rice. The algorithm selection problem. In *Advances in Computers*, volume 15, pp. 65–118. Elsevier, 1976. doi: 10.1016/S0065-2458(08)60520-3.

Noam Shazeer. Fast transformer decoding: One write-head is all you need, 2019. URL https://arxiv.org/abs/1911.02150.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank keys for efficient sparse attention. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 16692–16723. Curran Associates, Inc., 2024. doi: 10.52202/079017-0532. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/1e027da6bec9ceb2ec37951ceeccae93-Paper-Conference.pdf.

Libo Sun, Peixiong He, Po-Wei Harn, and Xiao Qin. Moe-nd: Per-layer mixture-of-experts routing for multi-axis kv cache compression, 2026. URL https://arxiv.org/abs/2604.17695.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training, 2024. URL https://arxiv.org/abs/2212.03533.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

David H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. doi: 10.1109/4235.585893.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5e5fd18f863cbe6d8ae392a93fd271c9-Paper-Conference.pdf.

Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606, 2008. doi: 10.1613/jair.2490.

Yifei Yang, Zouying Cao, Qiguang Chen, Libo Qin, Dongjie Yang, Hai Zhao, and Zhi Chen. Kvsharer: Efficient inference via layer-wise dissimilar kv cache sharing, 2024. URL https://arxiv.org/abs/2410.18517.

Dingyu Yao, Chenxu Yang, Zhengyang Tong, Zheng Lin, Wei Liu, Jian Luan, and Weiping Wang. Vecinfer: Efficient llm inference with low-bit kv cache via outlier-suppressed vector quantization, 2025. URL https://arxiv.org/abs/2510.06175.

Jiayi Yuan, Hongyi Liu, Shaochen Zhong, Yu-Neng Chuang, Songchen Li, Guanchu Wang, Duy Le, Hongye Jin, Vipin Chaudhary, Zhaozhuo Xu, Zirui Liu, and Xia Hu. KV cache compression, but what must we give in return? a comprehensive benchmark of long context capable approaches. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4623–4648. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.findings-emnlp.266/.

Rongzhi Zhang, Kuang Wang, Liyuan Liu, Shuohang Wang, Hao Cheng, Chao Zhang, and Yelong Shen. Lorc: Low-rank compression for llms kv cache with a progressive compression strategy, 2024a. URL https://arxiv.org/abs/2410.03111.

Xuan Zhang, Cunxiao Du, Chao Du, Tianyu Pang, Wei Gao, and Min Lin. Simlayerkv: A simple framework for layer-level kv cache reduction, 2024b. URL https://arxiv.org/abs/2410.13846.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6ceefa7b15572587b78ecfcebb2827f8-Paper-Conference.pdf.

Xiabin Zhou, Wenbin Wang, Minyan Zeng, Jiaxian Guo, Xuebo Liu, Li Shen, Min Zhang, and Liang Ding. Dynamickv: Task-aware adaptive kv cache compression for long context llms, 2025. URL https://arxiv.org/abs/2412.14838.

APPENDIX

This appendix provides complete technical specifications, automated integrity checks, per-task metric breakdowns, statistical tests, reachability derivations, and model hyperparameters supporting the main text. To streamline navigation across diagnostic and algorithmic contributions, the content is organized into six consolidated parts:

Appendix	Thematic Scope	Subsections
Appendix A	**Setup & Integrity**: Hardware, harness, and routing leakage boundaries.	A.1 – A.2
Appendix B	**Diagnostic Analysis**: Memory accounting, weight floor, alternative normalizations, compression sweeps, physical stability, and statistical equivalence testing.	B.1 – B.5
Appendix C	**Workload Breakdowns**: Task taxonomies, per-task metric profiles, preference sweeps, sample heterogeneity, secondary benchmarks, and 8B scale generalization.	C.1 – C.5
Appendix D	**Routing Framework**: Evaluation utility, action catalog, reachability estimators, cross-scale transfer, and complete selector model specifications.	D.1 – D.3
Appendix E	**Extended Routing Results**: Decision overheads, component timing, exact shortfall attribution (regret decomposition), and selector calibration.	E.1 – E.3
Appendix F	**Extended Limitations**: Serving scope, batching dynamics, kernel maturity, representation bounds, and future directions.	App. F

A EXPERIMENTAL SETUP AND DATA INTEGRITY

A.1 EXPERIMENTAL PROTOCOL AND ACTION POOL

Model Backbone & Execution Hardware. All primary evaluations are executed on Llama-3.2-3B-Instruct (Meta, 2024b) in `float16` precision, hosted on a single dedicated NVIDIA H100 80GB GPU. Generation parameters follow LongBench defaults (Bai et al., 2024): decoding is greedy (`temperature` $= 0.0$, `top_p` $= 1.0$), rendering output sequences bit-identical across runs. Prompts exceeding context limits are truncated from the middle according to benchmark specifications. Dataset loading, prompt templating, and scoring metrics reuse the KVPress evaluation engine (Devoto et al., 2025). Every compressor configuration is executed as an isolated process; no KV-cache state or memory allocation is shared between distinct runs.

Artifact Fidelity & Configuration Pool. To evaluate compressors exactly as designed without introducing re-implementation artifacts, we integrate author-released reference implementations into our execution harness. Table 3 lists all nine evaluated configurations alongside their exact operational hyperparameters and indicates which expose a continuous compression level.

Compression-Level Convention. Implementations parameterize aggressiveness inconsistently, which has made operating points incomparable across papers. StreamingLLM and Q-Filters expose the fraction of the cache *evicted*; L2Compress exposes the fraction *retained*; SnapKV and PyramidKV expose a compression level in the KVPress sense. We index every configuration by a single scalar, the fraction of the KV cache removed or the number of tokens retained, and map it onto each compressor's native parameter. The two parameterizations are comparable at each level rather than only ordinally aligned: on LongBench the token budgets remove $\approx 32\%$, $\approx 57\%$ and $\approx 76\%$ of cached tokens at L_1, L_2 and L_3, against a nominal 25%, 50% and 75% for the fraction-parameterized methods (Fig. 6c). The correspondence is workload-dependent, since an absolute budget compresses less on shorter prompts; on RULER-4096 in particular the 4096-token cap is inactive and L_1 reduces to FullKV for SnapKV and PyramidKV. Under this convention larger ℓ always means more compression, and the sweeps in Appendix B.3 are monotone in accuracy and in

both memory quantities for all five swept methods. KIVI-4bit, SimLayerKV, and MiniCache expose no comparable scalar and run at their reference settings only.

Study design & measurement cohorts. Table 4 enumerates every execution cohort. The primary LongBench study contributes $3 \times 3{,}200 \times 9 = 86{,}400$ paired triples; the 8B cohort contributes an identical $86{,}400$; compression-level sweeps, RULER-4096 and LongBench-v2 contribute the remainder, for $433{,}857$ in total.

Measurement Instrumentation. For every prompt x and configuration m we record wall-clock latency using explicit CUDA event synchronization (`torch.cuda.Event`) surrounding the prefill phase (including post-prefill cache compression) and the autoregressive decoding phase, reporting $L = L_{\text{prefill}} + L_{\text{decode}}$ together with both phase components and the per-generated-token decode cost $L_{\text{decode}}/N_{\text{gen}}$. Peak GPU memory is queried via `torch.cuda.max_memory_allocated` separately for each phase, yielding M_{peak} and M_{dec} as defined in Eq. 2. Relative efficiency changes are computed as paired ratios against the uncompressed FullKV run from the identical repetition.

Routing cohorts. Sections 4–6 use the four $\ell = L_3$ cohorts on which all nine actions were executed on every request (Tab. 5). LongBench with Llama-3.2-3B-Instruct is the training cohort on which every selector is fitted; its selector results are in-sample replays of the final models (App. D.3). LongBench with Llama-3.1-8B-Instruct covers the same requests under a different model; no selector saw its outcomes, so it tests transfer across model scale on known prompts and serves the cross-scale analysis of Appendix D.2. RULER and LongBench-v2 contain prompts no selector saw and test transfer to a different task structure and to much longer contexts. LongBench request–action pairs were executed three times; the repetition index is kept, because the stability analysis of Appendix B.4 depends on it. Every cohort is dense: no measurement is imputed and no request is dropped, which is what permits replay-based evaluation.

A.2 Data Leakage Boundary and Context Grouping

Permitted inputs. A selector may observe the context, the instruction, an optional answer-prefix suffix, the requested output limit, and the input length under the target tokenizer. Nothing else exists before prefill, and the implementation enforces this with an explicit allowlist.

Prohibited inputs. The task name, the benchmark name, the reference answer, the realised output length, and every measured outcome are prohibited as selector inputs. Task labels appear only in three places, each marked where used: training sample weights, grouped analysis, and the explicitly non-deployable TaskLookup reference.

Grouping. Requests sharing an exact normalised context, a source document, a generator template, or a source family are assigned to the same group, and every split, cross-fitted estimate and bootstrap draw respects group boundaries. A random seed alone does not constitute an independent family. Target text is processed only inside a hash-only deduplicator, and where overlap is found the training-side row is removed rather than the evaluation-side row.

Table 3: **The nine evaluated configurations and the operating point each one runs at.** Settings follow the reference implementations; ℓ denotes the compression level, the fraction of the KV cache removed. The last column marks the five configurations for which ℓ is swept over $\{L_1, L_2, L_3\}$.

Configuration	Family	Operating point	Level mapping
FullKV	(uncompressed)	complete cache retained	–
KIVI-4bit	quantization	4-bit K/V, group 32, residual 32	–
SnapKV	eviction (attention)	L_3: 1024-token cap	$L_1 = 4096$; $L_2 = 2048$; $L_3 = 1024$ tokens
PyramidKV	eviction (attention)	L_3: 1024-token cap	$L_1 = 4096$; $L_2 = 2048$; $L_3 = 1024$ tokens
StreamingLLM	eviction (positional)	L_3: 75% removed, 4 sinks	$L_1 = 25\%$; $L_2 = 50\%$; $L_3 = 75\%$ removed
L2Compress	eviction (L2 norm)	L_3: 25% kept, prune after 1024	$L_1 = 75\%$; $L_2 = 50\%$; $L_3 = 25\%$ kept
Q-Filters	eviction (calibrated)	L_3: 75% removed, recent window 64	$L_1 = 25\%$; $L_2 = 50\%$; $L_3 = 75\%$ removed
SimLayerKV	layer-level reduction	lazy-layer threshold 0.9	–
MiniCache	cross-layer merging	merge threshold -1 (all eligible layers)	–

Table 4: **Complete measurement coverage.** Every execution cohort in the study. Each row contributes strictly paired triples: identical prompts run under every listed configuration.

Cohort	Benchmark	Model	ℓ	Configs	Prompts	Runs	Triples
LongBench (3B; L_3)	LongBench	3B	0.75	9	3,200	3	86,400
LongBench (3B; L_2)	LongBench	3B	0.50	5	3,200	3	48,000
LongBench (3B; L_1)	LongBench	3B	0.25	5	3,200	3	48,000
LongBench (8B; L_3)	LongBench	8B	0.75	9	3,200	3	86,400
LongBench (8B; L_2)	LongBench	8B	0.50	5	3,200	2	32,000
RULER (3B; L_3)	RULER-4096	3B	0.75	9	6,500	1	58,500
RULER (3B; L_2)	RULER-4096	3B	0.50	5	6,500	1	32,500
RULER (3B; L_1)	RULER-4096	3B	0.25	5	6,500	1	32,500
LongBench-v2 (3B; L_3)	LongBench-v2	3B	0.75	9	503	1	4,527
LongBench-v2 (3B; L_2)	LongBench-v2	3B	0.50	5	503	1	2,515
LongBench-v2 (3B; L_1)	LongBench-v2	3B	0.25	5	503	1	2,515

Total: 433,857 matched measurement triples.

Table 5: **Routing cohorts.** Every action was executed on every request. Measurements include the three LongBench repeats; repeats are not counted as independent requests. Role: how the cohort is used for selectors.

Cohort	Model	Benchmark	Role	Measurements	Requests	Actions	Tasks
LongBench 3B	3B	LongBench	training (in-sample)	86,400	3,200	9	16
LongBench 8B	8B	LongBench	scale transfer (seen prompts)	86,400	3,200	9	16
RULER 3B	3B	RULER-4096	transfer (unseen prompts)	58,500	6,500	9	13
LongBench-v2 3B	3B	LongBench-v2	transfer (unseen prompts)	4,527	503	9	16

Residual risk. Grouping by exact context and recorded provenance cannot exclude semantic near-duplicates or undocumented generator families. We regard this as a residual dependence risk rather than an eliminated one, and a near-duplicate audit at high embedding similarity across the split boundary is reported with the released artefacts.

B Diagnostic Measurement Analysis and Statistical Rigor

B.1 Memory Accounting and Weight Floor

Our instrumentation records total device allocation rather than an isolated KV-cache counter, so both M_{peak} and M_{dec} include the fixed model weights. The weight term is constant across configurations and therefore cancels in Pareto dominance, which is invariant to translation of an axis; it does, however, deflate reported percentage savings relative to the compressible component.

The floor can be established empirically. Under `float16` the 3,212,749,824-parameter backbone occupies 5.984 GiB and the 8,030,261,248-parameter backbone 14.958 GiB. The smallest allocation observed anywhere in the study is 6.020 GiB for the 3B cohorts and 15.006 GiB for the 8B cohorts, so the residual activation and cache term vanishes to within 0.05 GiB on the shortest prompts and the analytic weight figure is confirmed to better than 1%. Subtracting it isolates the cache-and-activation footprint $M - W$ that compression actually acts on (Fig. 4, third series): measured this way, decode-phase savings on LongBench reach 88.4% for SnapKV, 87.7% for PyramidKV, 84.6% for Q-Filters, 69.1% for KIVI-4bit, and 62.1% for StreamingLLM, against 7.0% for SimLayerKV and −4.9% for MiniCache.

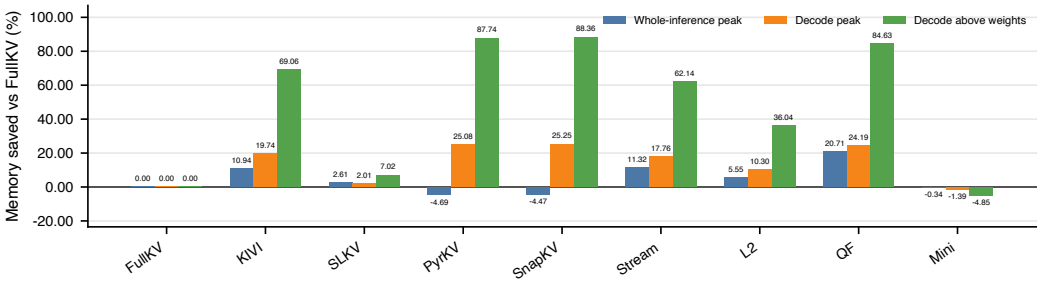


Figure 4: **Whether a compressor saves memory depends on which memory is counted.** Savings against paired FullKV on LongBench under whole-inference peak allocation, decode-phase peak allocation, and decode-phase allocation above the fixed 5.98 GiB weight floor.

Why the two accountings disagree. The prefill phase sets the peak allocation on 98.2% of LongBench prompts. A compressor's effect on M_{peak} is therefore determined by the largest transient it creates while constructing the cache, not by the cache it retains. Quantization and positional eviction compress as the cache is written and reduce both quantities. Attention-based eviction cannot: SnapKV and PyramidKV score tokens using attention computed over the full prompt, which must be materialized before anything can be discarded, so their transient exceeds the FullKV prefill transient by roughly 0.44 GiB on average and their peak allocation rises even though the retained cache shrinks by a quarter. MiniCache is a third case: it materializes merged cross-layer tensors while the source tensors are still live, and the merge yields no net reduction in either quantity (−0.3% peak, −1.4% decode), which is why it is dominated on every task.

Practical consequence. M_{peak} bounds whether a request can be served at all on a given device; M_{dec} bounds how many sequences can co-reside once generation is underway. Neither is the correct choice in general, and Section 3.3 shows they induce different Pareto frontiers and a differently partitioned preference simplex. Reporting a single memory number without naming the quantity leaves the comparison underdetermined.

B.2 Alternative Utility Normalization

Because bounds are task-wide while memory and latency scale with prompt length, $\hat{M}$ and $\hat{L}$ compress toward 1 on short prompts within a task, so utility there is weighted more toward accuracy. To verify that preference-based findings do not depend on normalization mechanics, we compare the min–max scalarization U_λ against a FullKV-relative alternative U_λ^{rel}. The relative formulation expresses accuracy, memory, and latency directly against the uncompressed baseline execution (m_0)

on prompt x:

$$\tilde{A}(x,m) = \frac{A(x,m)}{100}, \qquad \tilde{M}(x,m) = \frac{M(x,m)}{M(x,m_0)}, \qquad \tilde{L}(x,m) = \frac{L(x,m)}{L(x,m_0)}, \tag{10}$$

yielding the composite utility scalar

$$U_\lambda^{\mathrm{rel}}(x,m) = \lambda_A \tilde{A}(x,m) - \lambda_M \tilde{M}(x,m) - \lambda_L \tilde{L}(x,m), \tag{11}$$

where the negative signs enforce that lower relative memory and latency allocations yield higher utility. Because the two resource terms are ratios to FullKV, each contributing roughly $-\lambda$ at parity, U_λ^{rel} is negative for every action, FullKV included (it scores $\lambda_A - \lambda_M - \lambda_L$). The scale therefore has no meaningful zero: only differences between actions carry information, and larger (less negative) is better.

Trade-off differences across normalizations. Min–max scaling maps each metric's observed task range onto $[0, 1]$, granting equal normalized weight to the observed variation across the candidate pool. FullKV-relative scaling is unbounded above for latency and therefore penalizes large execution overheads far more aggressively: a method that is $3\times$ slower receives a $3\times$ larger latency penalty. Neither scalarization is uniquely canonical, which is why preference-free Pareto non-domination serves as our primary, scale-invariant evidence.

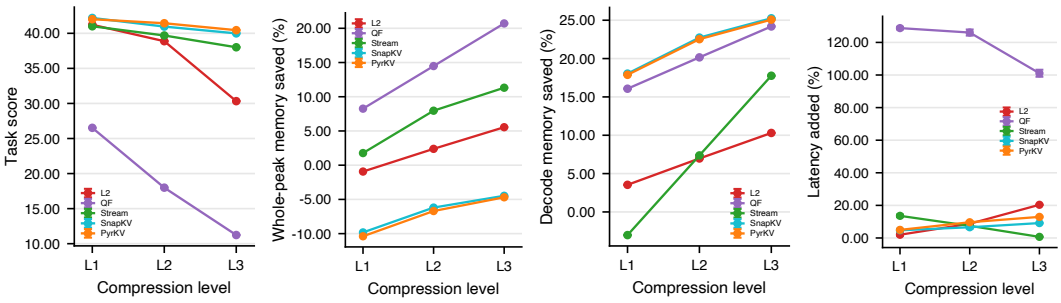
Structural invariance under U_λ^{rel}. Evaluating utility under U_λ^{rel} shifts individual winning regions but preserves every structural conclusion. Because every term is a cost relative to FullKV — the accuracy term is a loss and the two resource terms are ratios entering with negative signs — U_λ^{rel} is negative for every action except FullKV itself, and larger (less negative) values are better; the scale carries no zero point beyond the uncompressed cache. At the balanced weighting, FullKV becomes the overall winner ($U_\lambda^{\mathrm{rel}} = -0.525$) instead of StreamingLLM (-0.542), and the simplex area shares shift to FullKV (63.3%), StreamingLLM (27.8%), KIVI-4bit (7.8%), and Q-Filters (1.1%) — still four distinct regional winners. Across 99.8% of deployment weightings tasks disagree on the optimal compressor, selecting a mean of 2.83 distinct task-level winners per weighting. The choice-granularity ladder scales as $-0.525 \rightarrow -0.513 \rightarrow -0.457$, leaving 82.6% of available utility headroom uncaptured beneath the task-category level, against 87.7% under U_λ.

B.3 Complete Compression-Level Sweeps

Table 6: **Raising the compression ratio trades accuracy for memory monotonically, but not latency.** LongBench means for the five compressors exposing a continuous compression level ℓ (fraction of the cache removed). Memory and latency are paired % changes against FullKV; positive memory means a saving.

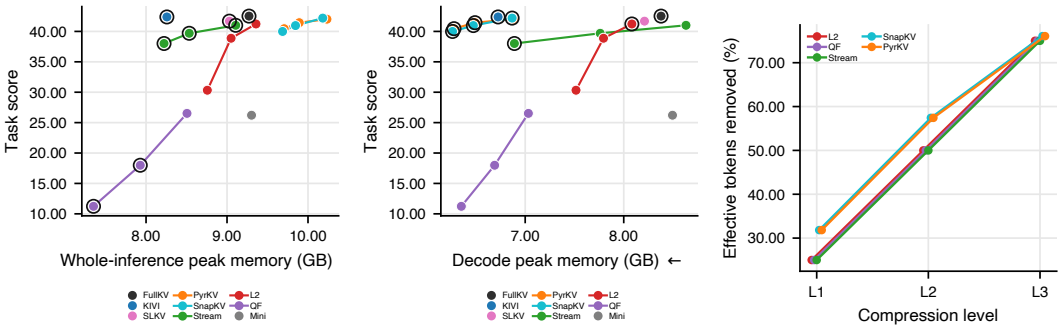
	$\ell = L_1$				$\ell = L_2$				$\ell = L_3$			
Configuration	A	M_{pk}	M_{dec}	L	A	M_{pk}	M_{dec}	L	A	M_{pk}	M_{dec}	L
L2Compress	41.2	-0.9	+3.5	+2.0	38.9	+2.4	+7.0	+8.7	30.3	+5.5	+10.3	+20.4
Q-Filters	26.5	+8.3	+16.1	+128.8	18.0	+14.5	+20.2	+126.0	11.2	+20.7	+24.2	+101.1
StreamingLLM	41.0	+1.8	-3.0	+13.6	39.7	+8.0	+7.4	+7.5	38.0	+11.3	+17.8	+0.7
SnapKV	42.2	-9.8	+18.0	+4.7	41.0	-6.2	+22.7	+6.6	40.0	-4.5	+25.3	+9.1
PyramidKV	42.0	-10.4	+17.9	+5.0	41.4	-6.7	+22.5	+9.6	40.5	-4.7	+25.1	+12.9

To verify that Pareto non-domination is robust rather than an artifact of single operating-point selection, we evaluate compression levels $\ell \in \{L_1, L_2, L_3\}$ for the five compressors that expose a continuous ratio. Table 6 presents raw metric values across all settings and Figure 5 the corresponding metric trajectories.



(a) Task score (↑). (b) Peak memory (↓). (c) Decode memory (↓). (d) Latency (↓).

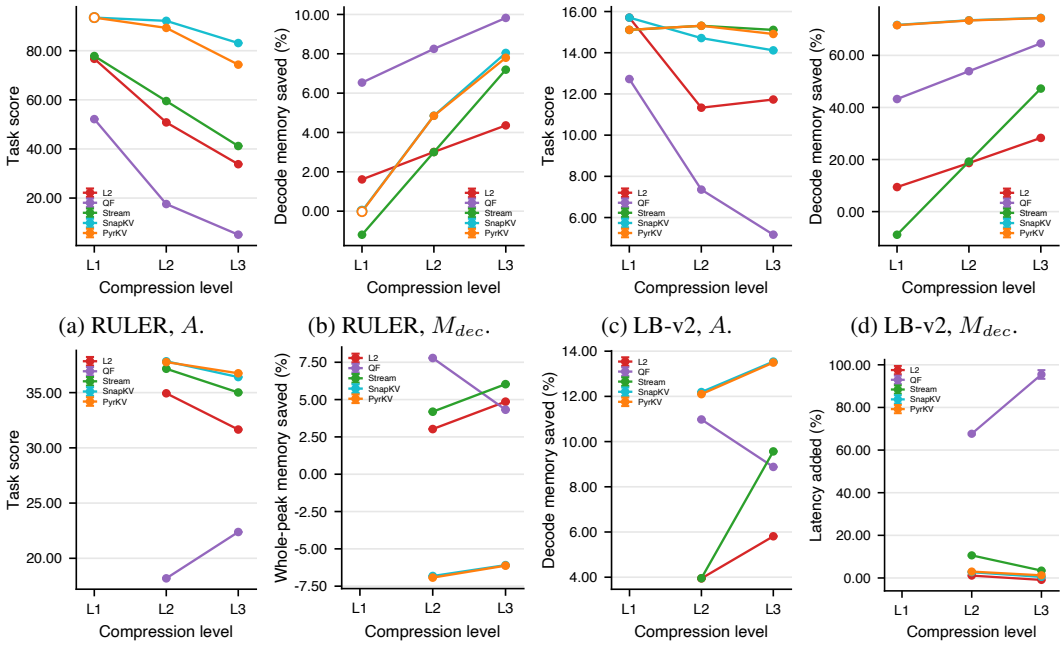
Figure 5: **Metric trajectories across compression levels $\ell \in \{L_1, L_2, L_3\}$ on LongBench.** Accuracy and both memory quantities move monotonically with ℓ for all five swept compressors; latency does not, and its direction differs by intervention locus.



(a) Peak memory. (b) Decode memory. (c) Effective compression.

Figure 6: **Sweeping the compression level populates the frontier rather than collapsing it.** All 19 LongBench operating points: four fixed configurations and five compressors at three compression levels, plotted against whole-inference peak memory **(a)** and decode-phase memory **(b)**, with lines connecting each swept compressor's three levels and outlined markers denoting non-domination (8 of 19 under peak memory, 9 of 19 under decode). **(c)** The three levels are approximately matched in realized aggressiveness across the two parameterizations, removing $\approx 32\%$, $\approx 57\%$ and $\approx 76\%$ of cached tokens on LongBench, so the token-budget and evicted-fraction methods are compared at comparable compression.

Figure 7 repeats the sweep on every other cohort for which one exists. Three findings emerge. First, **monotone parameter scaling**: increasing ℓ monotonically reduces task score and monotonically reduces both peak and decode memory for all five methods, confirming that the shared convention of Appendix A.1 is applied consistently. Second, **frontier population**: pooling the 15 swept points with the four fixed configurations yields 19 operating points, of which 8 remain Pareto-optimal under peak memory and 9 under decode memory (Tab. 7, Fig. 6); continuous ratios populate the non-dominated surface rather than collapsing it to a single winner. Third, **compression level-dependent latency trajectories**: for StreamingLLM and Q-Filters, overhead falls as ℓ rises (StreamingLLM $+13.6\%$ at $\ell = L_1 = 0.25$ down to $+0.7\%$ at $\ell = L_3 = 0.75$; Q-Filters $+128.8\%$ down to $+101.1\%$), because a smaller active cache cheapens every decoding step. For L2Compress, SnapKV, and PyramidKV the overhead rises instead ($+2.0\%$ to $+20.4\%$, $+4.7\%$ to $+9.1\%$, and $+5.0\%$ to $+12.9\%$), because their post-prefill scoring and eviction work grows with the amount discarded. Latency overhead is therefore a function of the operating point, not a static algorithm constant.



(a) RULER, A. (b) RULER, M_{dec}. (c) LB-v2, A. (d) LB-v2, M_{dec}.

(e) 8B, A. (f) 8B, M_{pk}. (g) 8B, M_{dec}. (h) 8B, L.

Figure 7: **Compression-level monotonicity reproduces on every cohort that was swept.** Metric trajectories for the five continuous-ratio compressors on RULER-4096, LongBench-v2, and the 8B LongBench cohort. 8B only sweeps L_2 and L_3.

B.4 Physical Measurement Stability and Cost Models

Because greedy decoding is deterministic, generated token sequences and downstream task scores are bit-identical across all three repeated executions of the primary LongBench cohort (28,800 matched prompt–configuration pairs per run). Peak GPU memory allocation is near-deterministic, exhibiting a maximum per-sample variation of 0.016 GiB and benchmark-level mean agreement within 2×10^{-5} GiB.

Wall-clock latency is the sole axis subject to physical run-to-run measurement noise. Benchmark-level mean latency varies by at most 0.136 s across repeats for MiniCache; at most 0.065 s for every other configuration. The induced variation in balanced utility is at most 0.024 points. Table 8 shows that StreamingLLM holds the highest balanced utility in all three independent executions. The only rank change between runs occurs between FullKV and KIVI-4bit, which are separated by at most 0.020 utility points and swap order in one of three runs; no other pair reorders. Measurement noise therefore does not affect the non-domination structure or any conclusion drawn from it.

Table 7: **Sweeping the compression ratio populates the frontier instead of collapsing it.** All 19 LongBench operating points (four fixed configurations plus five compressors at three levels), with non-domination evaluated under each memory accounting; rows are grouped by method.

Operating point	$A \uparrow$	M_{peak} saved% $\uparrow$	M_{dec} saved% $\uparrow$	L added% $\downarrow$	Pareto$_{\text{pk}}$	Pareto$_{\text{dec}}$
FullKV	42.52	0.00	0.00	0.00	✓	✓
KIVI-4bit	42.38	10.94	19.74	74.07	✓	✓
SnapKV @L_1	42.19	-9.82	18.04	4.70	–	✓
SnapKV @L_2	40.96	-6.19	22.73	6.57	–	✓
SnapKV @L_3	39.99	-4.47	25.25	9.14	–	✓
PyramidKV @L_1	42.01	-10.39	17.88	4.96	–	–
PyramidKV @L_2	41.44	-6.70	22.53	9.64	–	✓
PyramidKV @L_3	40.45	-4.69	25.08	12.94	–	✓
StreamingLLM @L_1	41.00	1.77	-3.02	13.58	✓	–
StreamingLLM @L_2	39.69	7.96	7.37	7.46	✓	–
StreamingLLM @L_3	38.01	11.32	17.76	0.68	✓	✓
L2Compress @L_1	41.22	-0.93	3.54	1.98	–	✓
L2Compress @L_2	38.88	2.39	6.97	8.72	–	–
L2Compress @L_3	30.33	5.55	10.30	20.36	–	–
Q-Filters @L_1	26.52	8.26	16.07	128.78	–	–
Q-Filters @L_2	17.99	14.48	20.16	126.05	✓	–
Q-Filters @L_3	11.24	20.71	24.19	101.08	✓	–
SimLayerKV	41.69	2.61	2.01	47.00	✓	–
MiniCache	26.22	-0.34	-1.39	112.93	–	–

Non-dominated: 8 of 19 under peak memory, 9 of 19 under decode memory.

Table 8: **Latency is the only axis with run-to-run noise, and it never changes the top-ranked configuration.** Mean end-to-end latency and balanced utility per repeated LongBench execution. Task scores are bit-identical across runs and peak memory agrees to 2×10^{-5} GiB, so all variation shown originates in wall-clock timing.

	Run 1		Run 2		Run 3	
Configuration	L	U_λ	L	U_λ	L	U_λ
FullKV	1.807	0.641	1.788	0.651	1.791	0.642
KIVI-4bit	3.131	0.640	3.089	0.660	3.154	0.636
SimLayerKV	2.640	0.615	2.622	0.631	2.654	0.615
PyramidKV	2.038	0.603	2.025	0.614	2.018	0.604
SnapKV	1.971	0.603	1.955	0.615	1.951	0.604
StreamingLLM	1.818	0.664	1.802	0.674	1.802	0.665
L2Compress	2.175	0.601	2.154	0.613	2.153	0.602
Q-Filters	3.597	0.522	3.628	0.540	3.604	0.523
MiniCache	3.772	0.489	3.908	0.509	3.787	0.490

Largest benchmark-level latency spread across runs: 0.136 s.
StreamingLLM holds the highest utility in every run.

Stability of the routing headroom. Across the three executions of every LongBench request–action combination, task score is effectively deterministic under greedy decoding: the large majority of combinations return an identical score in all three executions and the mean within-combination standard deviation is numerically zero. Peak memory varies by a negligible fraction of the standard deviation between combinations, and end-to-end latency by approximately one percent in relative terms, consistent with kernel scheduling and allocator variation rather than any change in the computation performed. Applying Eq. 14 changes the hindsight value by less than a small fraction of one percent of H, so we report H uncorrected.

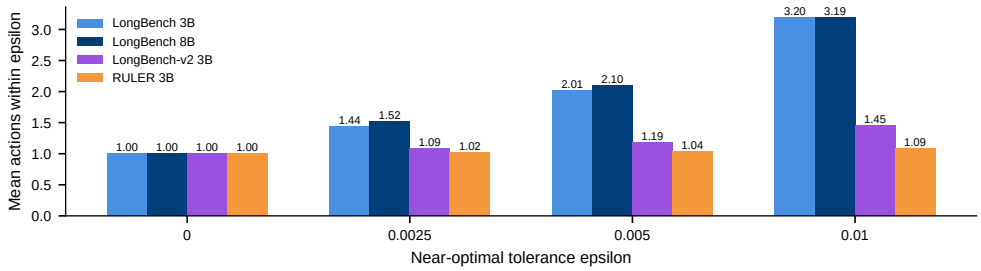
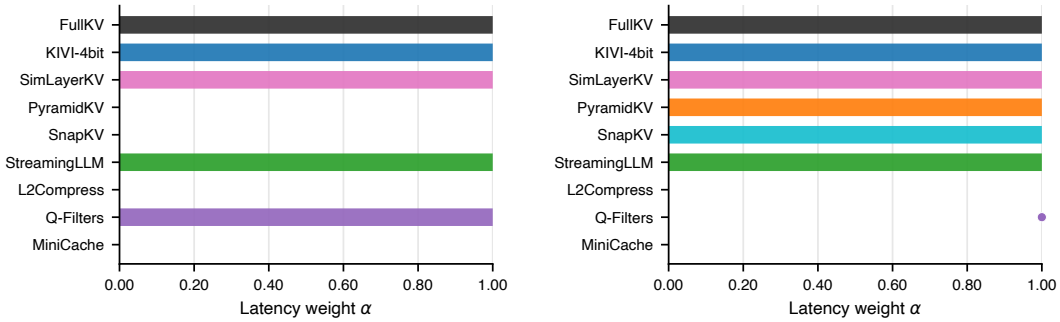


Figure 8: **On most requests several actions are nearly as good as the best one, so ranking errors are cheap and detection errors are not.** Mean number of actions whose utility falls within a given tolerance of the per-request optimum. At a tolerance of 0.01 utility, 3.20 of the nine actions qualify on LongBench 3B and 3.19 on 8B, against 1.09 on RULER, where the optimum is usually unique.

A consequence of determinism is that when two actions score within a hair of each other on a request, that closeness is a real property of the request and not measurement error. Figure 8 asks how often this happens: for a given tolerance, how many of the nine actions land within that tolerance of the best one. On LongBench roughly three do at a tolerance of 0.01 utility, meaning a selector has about three interchangeable correct answers on a typical request and loses almost nothing by choosing among them. On RULER only one does, so there the optimum is usually unique and ranking matters more. The practical implication is the same on both: the expensive mistakes are not picking the second-best action, but failing to notice that a request is one where the choice carries real cost. That asymmetry is the empirical basis for treating routing as detection before ranking.

Locus of Latency Overhead. Measuring prefill and decoding phases separately shows where runtime penalties concentrate. Across all nine configurations, mean prefill latency (including post-prefill cache compression) spans a narrow band from $0.326\,\mathrm{s}$ to $0.393\,\mathrm{s}$, a total spread of 21%. Mean autoregressive decoding latency spans $1.461\,\mathrm{s}$ to $3.457\,\mathrm{s}$, a spread of 137%. Relative to FullKV, prefill overhead reaches at most $+17.8\%$ (SnapKV) while decode overhead reaches $+136.6\%$ (MiniCache). Wall-clock overheads are therefore driven by per-step decoding arithmetic rather than by one-off prefill compression operations, which is the mechanism behind the intervention-locus account in Section 3.5.

Invariance to the Prefill/Decode Weighting. Different deployments weight the two phases differently: time-to-first-token is governed by L_{prefill} alone ($\alpha = 1$), interactive streaming by decode cost ($\alpha = 0$), and batch throughput by something in between. We therefore recompute benchmark-level Pareto membership for 41 values of $\alpha \in [0, 1]$ (Fig. 9). Under peak memory the non-dominated set is *identical* at every α: {FullKV, KIVI-4bit, SimLayerKV, StreamingLLM, Q-Filters}. Under decode memory it is constant at six members for $\alpha < 1$ and gains a seventh only in the degenerate prefill-only limit. Substituting per-generated-token decode latency $L_{\mathrm{decode}}/N_{\mathrm{gen}}$ for L entirely, which changes the axis non-affinely rather than reweighting it, expands the frontier to eight of nine configurations. No conclusion in this paper depends on the particular latency composition, and the alternatives that change anything make non-domination stronger, not weaker.



(a) Sweep over peak-memory latency.

(b) Sweep over decode-memory latency.

Figure 9: **Pareto membership is invariant to the latency cost model.** Non-dominated configurations (coloured bars) as the latency definition sweeps from pure decode cost to pure prefill cost, $L_\alpha = \alpha L_{\text{prefill}} + (1-\alpha)L_{\text{decode}}$, under each memory accounting. The frontier is constant in α under peak memory and changes only at $\alpha = 1$ under decode memory.

B.5 Statistical Hypothesis Testing and TOST Equivalence

Paired Hypothesis Testing vs. FullKV. Table 9 reports paired Wilcoxon signed-rank tests comparing each compressed configuration against the uncompressed FullKV baseline on the 3,200 identical LongBench prompts. All peak memory, decode memory, and wall-clock latency differences are statistically significant ($p < 10^{-7}$). On downstream accuracy, KIVI-4bit is the sole configuration not statistically separable from FullKV under the two-sided test ($p = 0.34$); all others degrade significantly ($p < 10^{-6}$).

Because a failure to reject is not evidence of equivalence, we additionally conduct a formal two one-sided tests (TOST) procedure on the same paired per-prompt differences (Tab. 10). We fix the equivalence margin at $\delta = \pm 0.5$ task-score points, chosen to be strictly smaller than the accuracy gap between FullKV and every other configuration in the pool (the closest being SimLayerKV at 0.83 points), so that a margin this tight cannot admit a second method by construction. KIVI-4bit passes at this margin ($p = 0.003$; mean paired difference -0.14 points, 95% prompt-clustered bootstrap CI $[-0.41, +0.10]$ over 2,000 resamples) and is the only configuration to do so at either $\delta = 0.5$ or the looser $\delta = 1.0$. TOST is computed with paired t-tests on the per-prompt differences rather than with the signed-rank variant, because the latter tests the pseudo-median, which is 0.00 for six of the eight compressed configurations even where mean accuracy falls by several points. The margin is chosen from the observed pool rather than pre-registered; we report both thresholds so that the conclusion can be read at either. We therefore describe KIVI-4bit as equivalent to FullKV within ± 0.5 score points, not merely as an unrejected null.

Table 9: **Paired Wilcoxon signed-rank tests against FullKV** on the 3,200 identical LongBench prompts, with latency averaged over the three repeated runs. Δ values are compressed minus FullKV, so a negative ΔM or ΔL means less memory or time.

	Task score		Peak memory		Decode memory		Latency	
Configuration	ΔA	p	ΔM	p	ΔM_{dec}	p	ΔL	p
KIVI-4bit	-0.14	0.337	-1.01	$< 10^{-300}$	-1.65	$< 10^{-300}$	+1.33	$< 10^{-300}$
SimLayerKV	-0.83	10^{-7}	-0.24	$< 10^{-300}$	-0.17	$< 10^{-300}$	+0.84	$< 10^{-300}$
PyramidKV	-2.07	10^{-20}	+0.43	10^{-133}	-2.10	$< 10^{-300}$	+0.23	10^{-151}
SnapKV	-2.53	10^{-31}	+0.41	10^{-125}	-2.12	$< 10^{-300}$	+0.16	10^{-123}
StreamingLLM	-4.51	10^{-47}	-1.05	$< 10^{-300}$	-1.49	$< 10^{-300}$	+0.01	10^{-66}
L2Compress	-12.19	10^{-143}	-0.51	10^{-272}	-0.86	$< 10^{-300}$	+0.37	10^{-8}
Q-Filters	-31.28	10^{-307}	-1.92	$< 10^{-300}$	-2.03	$< 10^{-300}$	+1.81	$< 10^{-300}$
MiniCache	-16.30	10^{-211}	+0.03	$< 10^{-300}$	+0.12	$< 10^{-300}$	+2.03	$< 10^{-300}$

Table 10: **Paired accuracy differences against FullKV** on the 3,200 identical LongBench prompts, with 95% prompt-clustered bootstrap CIs (2,000 resamples, seed 2027) and two one-sided tests (TOST) at equivalence margins $\delta = 0.5$ and $\delta = 1.0$ task-score points. Equivalence is declared when the CI lies strictly inside $\pm\delta$, a criterion at least as conservative as TOST at $\alpha = 0.05$.

			$\delta = 0.5$		$\delta = 1.0$	
Configuration	ΔA	95% CI	TOST p	equiv.	TOST p	equiv.
KIVI-4bit	-0.14	[-0.41, +0.10]	0.003	✓	10^{-11}	✓
SimLayerKV	-0.83	[-1.20, -0.46]	0.966	–	0.176	–
PyramidKV	-2.07	[-2.57, -1.57]	1.000	–	1.000	–
SnapKV	-2.53	[-3.06, -1.94]	1.000	–	1.000	–
StreamingLLM	-4.51	[-5.24, -3.79]	1.000	–	1.000	–
L2Compress	-12.19	[-13.19, -11.20]	1.000	–	1.000	–
Q-Filters	-31.28	[-32.72, -29.92]	1.000	–	1.000	–
MiniCache	-16.30	[-17.35, -15.25]	1.000	–	1.000	–

$\delta = 0.5$ is smaller than the accuracy gap between FullKV and every configuration other than KIVI-4bit (nearest: SimLayerKV, 0.83 points). TOST is computed with paired t-tests on the per-prompt differences.

Accuracy Rank Resolution. To determine whether accuracy differences between top configurations are resolved at all, we conduct paired Wilcoxon tests between the highest- and second-highest-scoring configuration on each individual task. The difference exceeds the paired noise floor ($p < 0.05$) on only 1 of the 16 LongBench tasks: on 15 of 16, the gap between the two best configurations is smaller than the within-task variation across prompts, while every system-axis contrast is resolved at $p < 10^{-20}$. We apply no multiplicity correction here, because the argument does not rest on rejections: correcting for 16 comparisons would only reduce the single significant task, strengthening rather than weakening the claim that accuracy alone cannot order the pool. This asymmetry in resolution between the accuracy and system axes is the statistical form of the argument for multi-metric evaluation — the axis the literature ranks on is the one its data supports least.

Prompt-Clustered Bootstrap Protocol. To compute confidence intervals for the choice-granularity utility gaps (Sec. 3.5), we execute a prompt-clustered bootstrap resampling protocol (2,000 resamples, fixed random seed 2027). Resampling is clustered by prompt unique identifier: when a prompt x is drawn into a bootstrap sample, its paired measurements across all nine configurations and all three repeated executions move together. This cluster bootstrap accounts for prompt-level sampling variance while preventing repeated runs of the same input from being counted as independent observations.

Bootstrap Stability of Frontier Membership. Pareto dominance (Sec. 3.1) is evaluated on sample means, so we verify that frontier membership is not an artifact of point estimates. Resampling the 3,200 prompts 1,000 times under the same prompt-clustered scheme, each of the five peak-memory frontier members is non-dominated in 100% of resamples and each dominated configuration in 0%; under decode memory the six members hold at 97.4–100%. Membership is therefore stable to prompt sampling as well as to the repeated-execution noise quantified in Appendix B.4.

Is the Oracle Gain Selection Bias? Taking a pointwise maximum over nine candidates invites optimizer's-curse inflation, so we bound it three ways. First, greedy decoding makes task scores bit-identical across repeated executions, so the accuracy axis contributes no measurement variance. Second, selecting per prompt on run 1 alone and evaluating the selection on runs 2–3 retains 99.4% of the in-sample oracle gain, bounding the contribution of wall-clock jitter below 1%. Third, permuting each configuration's utilities across prompts within a task family — which preserves marginals but destroys prompt–configuration pairing — yields a gain of 0.179 against the observed 0.046; because utilities are strongly correlated across prompts, pure noise-maximization would produce roughly four times the headroom we report. The measured gain is therefore not an artifact of the max operator, though it remains an in-sample ceiling over the evaluated pool rather than a recoverable target.

C Diagnostic Workload and Task-Level Breakdowns

C.1 Task-Type Categories and Granularity Definitions

The category-level choice granularity (m_{type}) conditions compressor selection on the six official LongBench task types rather than on the 16 constituent datasets individually. Using the benchmark's native taxonomy ensures an unbiased evaluation: the domain groupings are established independently of compressor outcomes, preventing post-hoc optimization. Table 11 details the composition of each task family alongside its selected compressor and sample-level winner distribution.

Table 11: **Even inside a single task family the best configuration changes from prompt to prompt.** The six official LongBench task families. "Type choice" is what the task-type granularity selects under balanced weighting; "sample winners" counts distinct configurations that are best on at least one prompt.

Task family	Tasks	Prompts	Type choice	Sample winners	Most frequent winner
Code	2	450	StreamingLLM	9	StreamingLLM (40%)
Few-shot	3	675	StreamingLLM	9	StreamingLLM (61%)
Multi-doc QA	3	600	KIVI-4bit	9	KIVI-4bit (48%)
Single-doc QA	3	430	KIVI-4bit	9	KIVI-4bit (43%)
Summarization	3	595	StreamingLLM	9	StreamingLLM (36%)
Synthetic	2	450	KIVI-4bit	6	KIVI-4bit (51%)

Two structural characteristics emerge. First, **category-level concentration**: across all six task families the category-level selector uses only two distinct configurations (StreamingLLM for Code, Few-shot, and Summarization; KIVI-4bit for both QA families and Synthetic), which explains why category-level selection yields only a modest utility gain ($+0.0056$) over static global choice. Second, **intra-family sample variance**: within every task family, between six and nine distinct configurations achieve top sample utility, and the most frequent single winner covers only 36%–61% of that family's prompts. Category-level selection therefore fails to capture the majority of the available efficiency headroom, because broad domain labels mask substantial instance-level variation in context structure and information density.

For RULER-4096 the category granularity is the 13 official subtasks; for LongBench-v2 it is the 16 official subdomains. Both taxonomies are benchmark-native and fixed before any compressor outcome is observed.

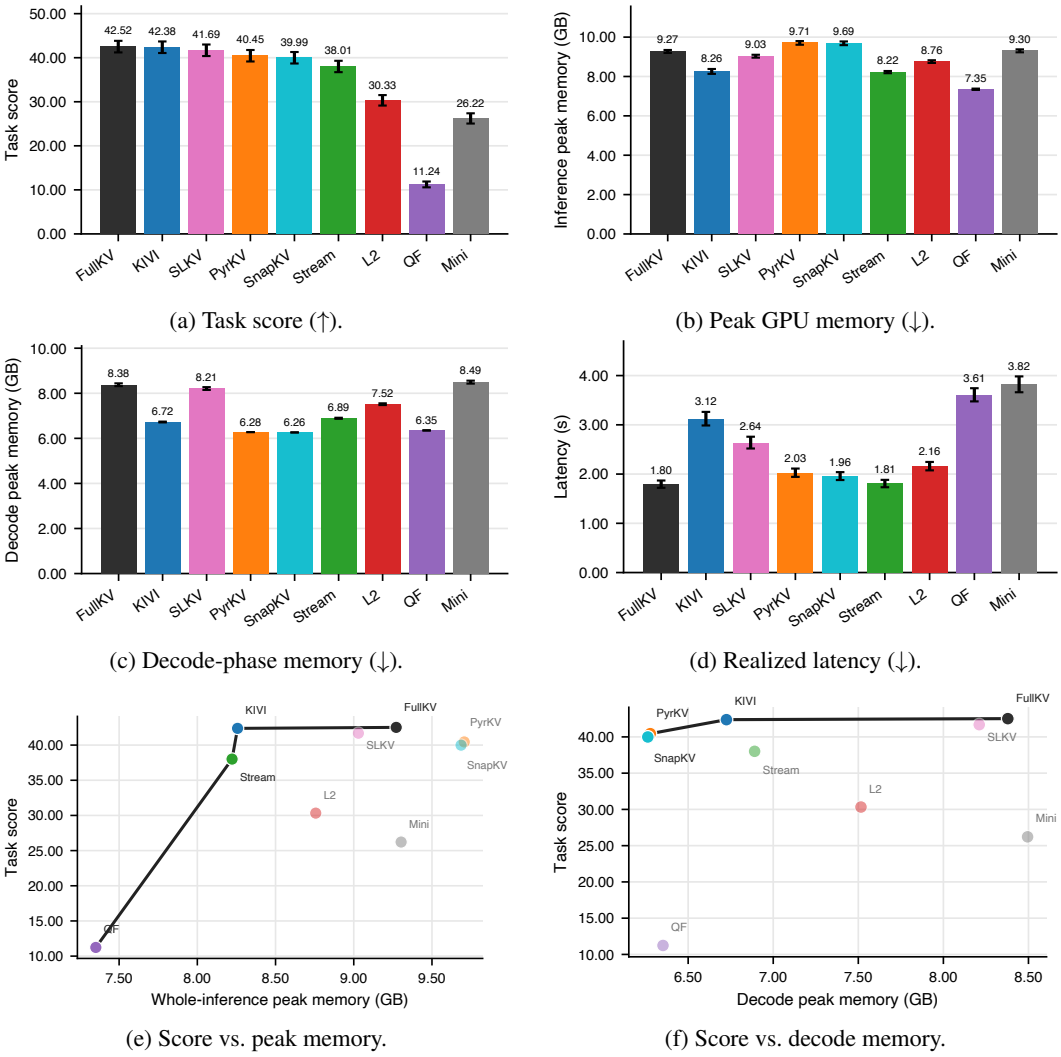
C.2 Per-Task Performance Profiles and Sweeps

This section decomposes the joint Pareto and utility results from the main text into individual measured axes across all 16 constituent LongBench tasks.

C.2.1 Benchmark-Level Means with Confidence Intervals

Figure 10 presents benchmark-level means across all measured axes. The confidence intervals expose markedly different measurement precision across axes. On the **system efficiency axes**, intervals are narrow relative to cross-configuration gaps: memory savings and latency overheads are stable, near-deterministic properties of each compressor. On the **task performance axis**, intervals are substantially wider relative to top-ranking score differences. This aligns with the paired statistical testing of Appendix B.5: the accuracy gap between the top two configurations reaches significance ($p < 0.05$) on only 1 of 16 tasks, whereas every memory difference is significant at $p < 10^{-124}$ and every latency difference at $p < 10^{-7}$. Accuracy is thus both the noisiest axis and the one the literature ranks on.

C.2.2 Per-Task Profiles Across Measured Axes



(a) Task score (↑).

(b) Peak GPU memory (↓).

(c) Decode-phase memory (↓).

(d) Realized latency (↓).

(e) Score vs. peak memory.

(f) Score vs. decode memory.

Figure 10: **Each measured axis orders the configuration pool differently.** Benchmark-level LongBench means with 95% confidence intervals, plus the two-dimensional score–memory frontiers. Evaluating in 2D hides configurations that are Pareto-optimal in 3D, and the two memory panels trace different frontiers from the same runs.

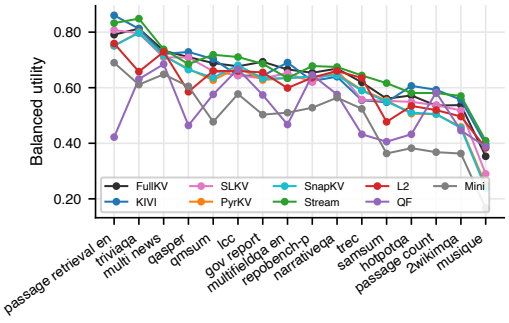
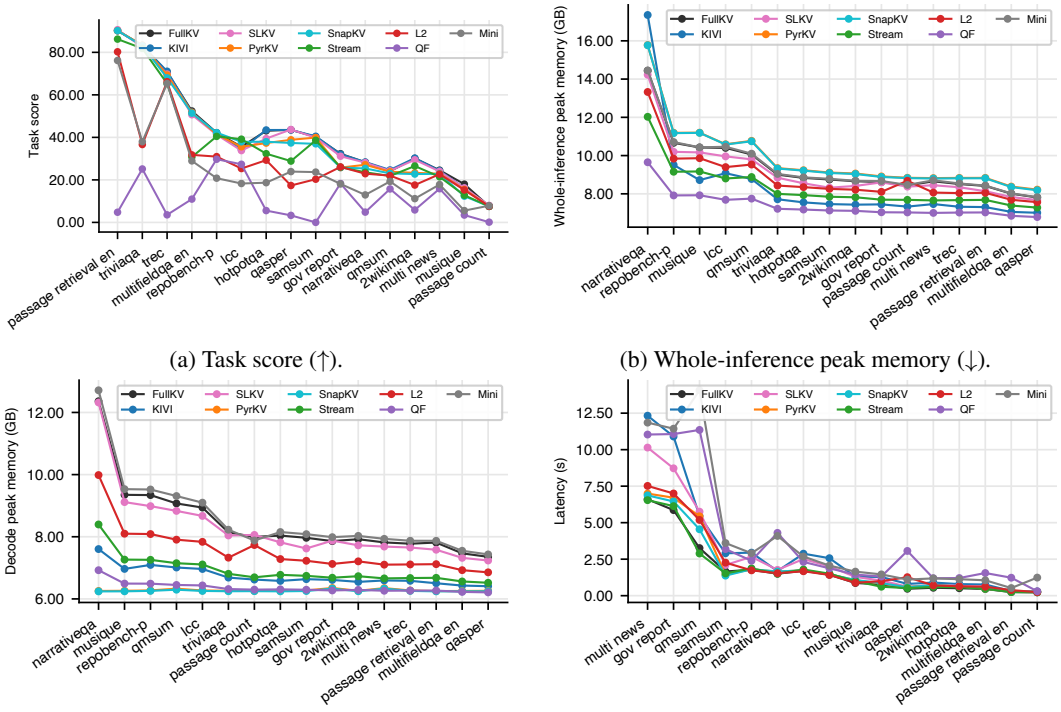


Figure 11: **No compressor holds top utility across all tasks.** Balanced utility (U_λ, ↑) per LongBench task. Every crossing of two profiles is one instance of the trade-off the diagnosis measures: the two configurations swap order between those tasks, so no ranking of the pool holds across the benchmark, and the per-task view shows the same non-domination that Sec. 3.2 reports in aggregate.



(a) Task score (↑).

(b) Whole-inference peak memory (↓).

(c) Decode-phase memory (↓).

(d) Realized latency (↓).

Figure 12: **Per-task performance profiles across the nine configurations on LongBench**, grouped by official task type. Each axis produces a distinct ordering, and the two memory panels invert the relative position of the attention-eviction methods.

Figure 11 shows the balanced-utility profiles, and Figure 12 isolates compressor behaviour across individual measured axes. On **task score**, FullKV, KIVI-4bit, and SimLayerKV form a top-performing band across most tasks, followed closely by PyramidKV and SnapKV, while Q-Filters degrades severely everywhere. On **peak memory**, the ordering is rearranged: Q-Filters achieves the lowest footprint on every task, followed by StreamingLLM and KIVI-4bit, while SnapKV and PyramidKV sit *above* FullKV. On **decode memory**, that inversion reverses and SnapKV and PyramidKV take the minimum on 12 of 16 tasks. On **latency**, a fourth distinct ranking appears, with FullKV and StreamingLLM fastest and MiniCache, Q-Filters, and KIVI-4bit incurring severe penalties. Tables 12, 13, 14, and 15 give the underlying task-level measurements.

Table 12: **Accuracy alone orders the pool differently on almost every task.** Per-task mean task score on LongBench (↑); bold marks the best value in each row. The final row weights every task equally.

Task	FullKV	KIVI	SLKV	PyrKV	SnapKV	Stream	L2	QF	Mini
2wikimqa	30.1	**30.2**	29.5	23.3	22.7	26.5	17.6	5.9	11.2
gov_report	**32.3**	32.1	31.1	25.8	25.8	26.0	26.1	18.3	18.1
hotpotqa	43.1	**43.5**	39.4	37.3	37.9	32.3	29.3	5.6	18.7
lcc	35.6	34.2	33.8	35.6	37.9	**39.2**	25.3	27.3	18.3
multi_news	**24.6**	24.4	24.0	22.9	22.8	21.4	22.7	15.8	17.8
multifieldqa_en	**52.4**	51.8	50.6	51.7	51.4	30.8	31.8	11.0	28.9
musique	**18.0**	16.1	16.2	15.5	12.3	12.8	15.2	3.4	5.5
narrativeqa	28.2	**28.4**	28.1	27.1	25.4	23.6	22.9	4.8	12.9
passage_count	7.6	7.6	**8.0**	7.6	7.6	7.6	7.6	0.1	7.9
passage_retrieval_en	90.2	90.2	**90.7**	90.2	90.2	86.2	80.3	4.8	76.2
qasper	43.5	43.5	**43.7**	38.9	37.4	28.8	17.4	3.3	23.9
qmsum	24.1	**24.7**	24.3	23.8	23.1	21.8	22.0	15.7	19.3
repobench-p	**42.2**	41.9	41.3	41.4	**42.2**	40.5	31.0	29.7	20.8
samsum	40.2	**40.5**	39.9	39.9	37.0	38.6	20.3	0.0	23.7
trec	70.7	**71.1**	69.8	69.3	68.0	65.3	66.2	3.6	65.3
triviaqa	83.3	83.1	82.5	**83.4**	83.1	81.8	36.6	25.1	38.0
Task mean	**41.6**	41.5	40.8	39.6	39.0	36.5	29.5	10.9	25.4

The summary row represents an unweighted macro-average across the 16 tasks.

Table 13: **Whole-inference peak memory is dominated by prefill, so attention-eviction methods show no saving.** Per-task mean peak GPU memory in GiB on LongBench (↓); bold marks the row minimum.

Task	FullKV	KIVI	SLKV	PyrKV	SnapKV	Stream	L2	QF	Mini
2wikimqa	8.66	7.43	8.41	9.06	9.04	7.82	8.22	**7.11**	8.68
gov_report	8.62	7.45	8.60	8.91	8.88	7.69	8.10	**7.04**	8.67
hotpotqa	8.83	7.55	8.55	9.23	9.21	7.93	8.35	**7.18**	8.86
lcc	10.40	9.08	9.96	10.59	10.58	8.80	9.40	**7.69**	10.46
multi_news	8.65	7.46	8.45	8.82	8.80	7.66	8.06	**7.00**	8.71
multifieldqa_en	8.01	7.05	7.83	8.38	8.35	7.39	7.69	**6.85**	8.02
musique	10.42	8.72	10.17	11.20	11.19	9.17	9.86	**7.93**	10.43
narrativeqa	14.44	17.35	14.23	15.78	15.77	12.03	13.33	**9.65**	14.45
passage_count	8.44	7.33	8.38	8.84	8.82	7.68	8.68	**7.03**	8.46
passage_retrieval_en	8.41	7.31	8.14	8.83	8.81	7.68	8.05	**7.03**	8.43
qasper	7.81	7.01	7.68	8.22	8.19	7.28	7.56	**6.79**	7.82
qmsum	10.06	8.77	9.80	10.77	10.75	8.88	9.53	**7.75**	10.10
repobench-p	10.67	9.51	10.19	11.19	11.17	9.15	9.84	**7.92**	10.71
samsum	8.75	7.46	8.32	9.11	9.09	7.85	8.26	**7.13**	8.78
trec	8.51	7.32	8.34	8.83	8.81	7.67	8.04	**7.02**	8.54
triviaqa	9.00	7.71	8.84	9.34	9.32	8.00	8.43	**7.22**	9.04
Task mean	9.36	8.41	9.12	9.82	9.80	8.29	8.84	**7.39**	9.39

The summary row represents an unweighted macro-average across the 16 tasks.

Table 14: **Under decode-phase memory the same pool reorders completely.** Per-task mean decode-phase peak GPU memory in GiB on LongBench (↓); bold marks the row minimum.

Task	FullKV	KIVI	SLKV	PyrKV	SnapKV	Stream	L2	QF	Mini
2wikimqa	7.92	6.55	7.72	6.26	**6.25**	6.73	7.21	6.29	8.03
gov_report	7.86	6.61	7.87	6.35	6.33	6.68	7.13	**6.27**	7.99
hotpotqa	8.04	6.58	7.82	6.26	**6.24**	6.77	7.28	6.31	8.15
lcc	8.94	6.96	8.67	6.27	**6.26**	7.11	7.84	6.43	9.10
multi_news	7.82	6.59	7.68	6.35	6.33	6.66	7.10	**6.26**	7.93
multifieldqa_en	7.47	6.43	7.32	6.26	6.24	6.56	6.92	**6.23**	7.55
musique	9.35	6.97	9.12	6.26	**6.24**	7.26	8.10	6.49	9.53
narrativeqa	12.36	7.60	12.32	6.26	**6.24**	8.40	9.99	6.92	12.71
passage_count	8.00	6.62	8.06	6.25	**6.25**	6.69	7.73	6.30	7.88
passage_retrieval_en	7.81	6.50	7.58	6.25	**6.24**	6.68	7.12	6.27	7.87
qasper	7.35	6.41	7.24	6.26	6.24	6.52	6.85	**6.21**	7.43
qmsum	9.07	7.01	8.83	6.33	**6.30**	7.15	7.91	6.45	9.31
repobench-p	9.34	7.09	8.99	6.27	**6.26**	7.26	8.09	6.49	9.52
samsum	7.97	6.64	7.62	6.27	**6.25**	6.74	7.23	6.29	8.08
trec	7.78	6.58	7.65	6.27	**6.26**	6.67	7.11	6.27	7.87
triviaqa	8.14	6.69	8.04	6.26	**6.25**	6.81	7.33	6.32	8.23
Task mean	8.45	6.74	8.28	6.28	**6.26**	6.92	7.56	6.36	8.57

The summary row represents an unweighted macro-average across the 16 tasks.

Table 15: **Latency produces a third, independent ordering.** Per-task mean end-to-end latency in seconds on LongBench (↓); bold marks the row minimum.

Task	FullKV	KIVI	SLKV	PyrKV	SnapKV	Stream	L2	QF	Mini
2wikimqa	**0.54**	0.90	0.74	0.72	0.74	0.60	0.71	1.19	1.18
gov_report	**5.86**	10.90	8.72	6.71	6.45	6.13	7.00	11.07	11.44
hotpotqa	**0.51**	0.79	0.68	0.63	0.63	0.58	0.64	1.21	1.13
lcc	1.70	2.88	2.51	1.74	1.74	1.79	**1.67**	2.31	2.68
multi_news	6.60	12.33	10.14	6.99	6.87	**6.54**	7.52	11.04	11.85
multifieldqa_en	**0.46**	0.76	0.61	0.48	0.49	0.47	0.62	1.55	1.05
musique	**0.87**	1.30	1.28	1.06	1.07	0.99	0.88	1.44	1.65
narrativeqa	**1.50**	1.69	1.77	1.64	1.66	**1.50**	1.54	4.30	4.09
passage_count	**0.23**	0.29	0.25	0.27	0.28	0.23	0.24	0.30	1.24
passage_retrieval_en	**0.25**	0.32	0.28	0.29	0.29	0.25	0.38	1.23	0.52
qasper	**0.47**	0.81	0.61	0.59	0.57	0.52	1.27	3.06	1.10
qmsum	3.26	5.46	5.75	5.44	4.55	**2.89**	5.19	11.35	14.07
repobench-p	1.77	2.95	2.66	1.81	1.81	1.87	**1.73**	2.39	2.91
samsum	1.66	2.90	2.10	1.44	**1.37**	1.53	2.26	3.19	3.60
trec	1.44	2.57	2.01	1.47	1.46	1.51	**1.42**	1.87	2.06
triviaqa	0.78	1.28	1.04	0.82	0.86	**0.61**	0.99	1.31	1.46
Task mean	**1.74**	3.01	2.57	2.01	1.93	1.75	2.13	3.68	3.88

The summary row represents an unweighted macro-average across the 16 tasks.

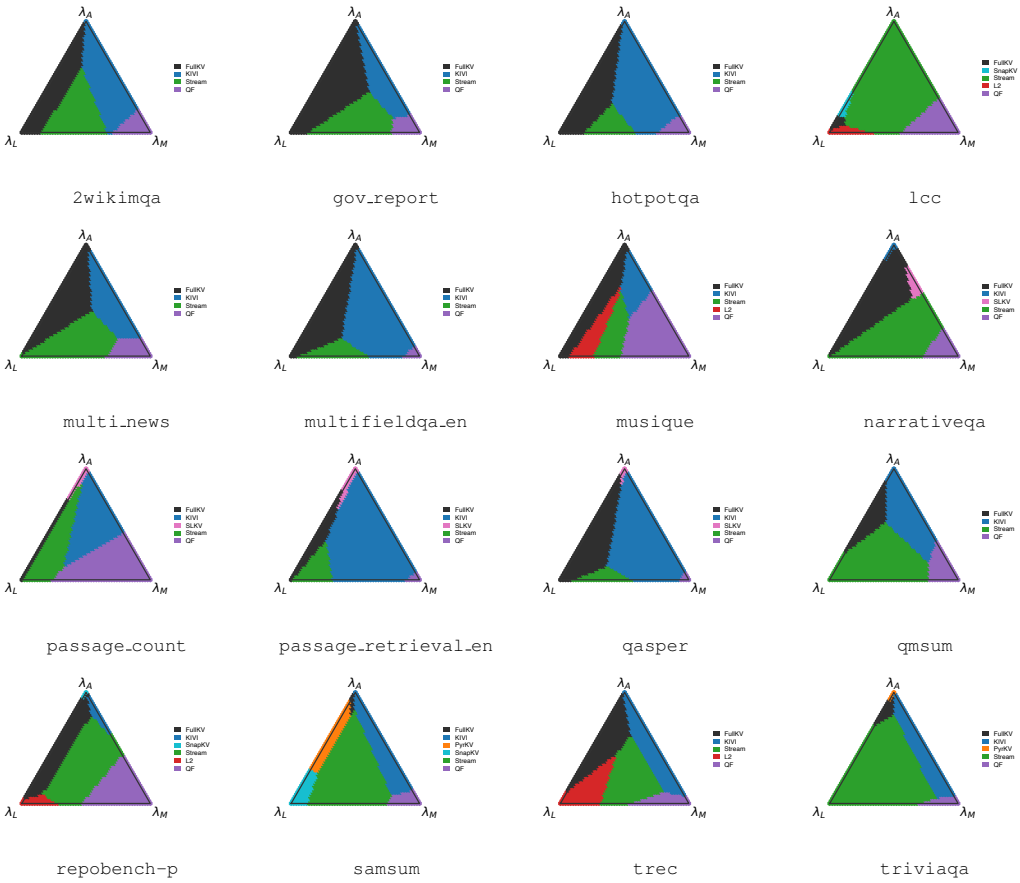


Figure 13: **Identical system priorities select different compressors depending on the task.** Per-task preference sweeps over $\Delta^2 = (\lambda_A, \lambda_M, \lambda_L)$ for all 16 LongBench tasks. Each point inside a triangle is a deployment preference; colour gives the winning configuration for that task under that preference.

C.2.3 Per-Task Preference Sweeps

Figure 13 repeats the deployment preference sweep of Section 3.4 for each individual task. The colour maps diverge sharply. On tasks where compression costs no accuracy (e.g., `lcc`), compressed configurations dominate the majority of the preference triangle. On tasks requiring global context retrieval where compression degrades accuracy (e.g., `multifieldqa_en`, `qasper`), FullKV occupies the accuracy-heavy region and yields to compressed methods only as weight shifts toward memory or latency. This task-level divergence is the mechanism behind the 99.3% disagreement figure reported in the main text.

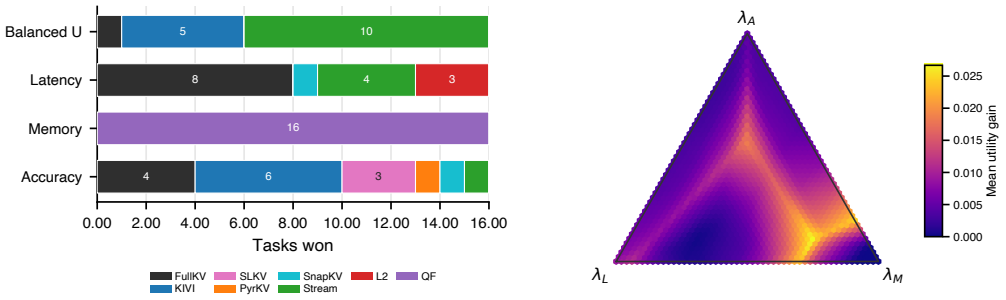
C.2.4 Per-Task Pareto Membership and Preferred Configurations

Table 16 reports per-task non-domination derived directly from raw metric triples using the dominance relation of Section 3.1. Every task admits between five and seven non-dominated configurations of nine under peak memory and between four and seven under decode memory. FullKV, StreamingLLM, and Q-Filters are Pareto-optimal on all 16 tasks under peak memory, KIVI-4bit on 15, and SimLayerKV on 14, while SnapKV and PyramidKV are non-dominated on only 3 and 2; under decode memory SnapKV rises to 16 and PyramidKV to 13 while Q-Filters falls to 4. Q-Filters attains universal membership under peak memory by occupying the absolute memory minimum rather than by offering a balanced trade-off, which is precisely the behaviour a preference-free frontier is supposed to expose rather than hide. MiniCache is the only configuration dominated on every task under peak memory, and is non-dominated on a single task under decode memory.

Table 16: **Only MiniCache is dominated on every task under peak memory, and non-dominated on a single task under decode memory.** Number of LongBench tasks (of 16) on which each configuration is non-dominated.

Configuration	Peak memory	Decode memory	Either
FullKV	16	16	16
KIVI-4bit	15	11	15
SimLayerKV	14	8	14
PyramidKV	2	13	13
SnapKV	3	16	16
StreamingLLM	16	13	16
L2Compress	9	6	9
Q-Filters	16	4	16
MiniCache	0	1	1

Per-task frontier size: 5–7 configurations (mean 5.69) under peak memory, 4–7 (mean 5.50) under decode memory.



(a) Tasks won per objective.

(b) Utility gained by conditioning on task type.

Figure 14: **Each objective induces a different ranking, and conditioning on task type recovers little of the gap. (a)** Top-performing configuration per LongBench task under each single objective and under balanced utility. **(b)** Mean utility gain of category-level over global static selection at each point of the weight simplex.

Table 17: **Six configurations are the most accurate somewhere and four the fastest somewhere, while the memory winner depends entirely on the accounting.** Preferred configuration per LongBench task under each single objective and under balanced utility.

Task	Best A	Lowest M_{peak}	Lowest M_{dec}	Lowest L	Best U_λ
2wikimqa	KIVI-4bit	Q-Filters	SnapKV	FullKV	StreamingLLM
gov_report	FullKV	Q-Filters	Q-Filters	FullKV	FullKV
hotpotqa	KIVI-4bit	Q-Filters	SnapKV	FullKV	KIVI-4bit
lcc	StreamingLLM	Q-Filters	SnapKV	L2Compress	StreamingLLM
multi_news	FullKV	Q-Filters	Q-Filters	StreamingLLM	StreamingLLM
multifieldqa_en	FullKV	Q-Filters	Q-Filters	FullKV	KIVI-4bit
musique	FullKV	Q-Filters	SnapKV	FullKV	StreamingLLM
narrativeqa	KIVI-4bit	Q-Filters	SnapKV	StreamingLLM	StreamingLLM
passage_count	SimLayerKV	Q-Filters	SnapKV	FullKV	KIVI-4bit
passage_retrieval_en	SimLayerKV	Q-Filters	SnapKV	FullKV	KIVI-4bit
qasper	SimLayerKV	Q-Filters	Q-Filters	FullKV	KIVI-4bit
qmsum	KIVI-4bit	Q-Filters	SnapKV	StreamingLLM	StreamingLLM
repobench-p	SnapKV	Q-Filters	SnapKV	L2Compress	StreamingLLM
samsum	KIVI-4bit	Q-Filters	SnapKV	SnapKV	StreamingLLM
trec	KIVI-4bit	Q-Filters	SnapKV	L2Compress	StreamingLLM
triviaqa	PyramidKV	Q-Filters	SnapKV	StreamingLLM	StreamingLLM
distinct winners	6	1	2	4	3

Table 17 and Figure 14 summarize top-performing configurations per task across individual objectives, alongside the utility that conditioning on task type recovers (Fig. 14b). Six configurations achieve top accuracy on at least one task and four minimize latency on at least one, while the memory winner is unanimous under peak allocation and splits two ways under decode allocation. This objective disagreement is pervasive: the balanced-utility winner differs from the most accurate configuration on 13 of 16 tasks, showing that optimizing purely for accuracy obscures system efficiency trade-offs.

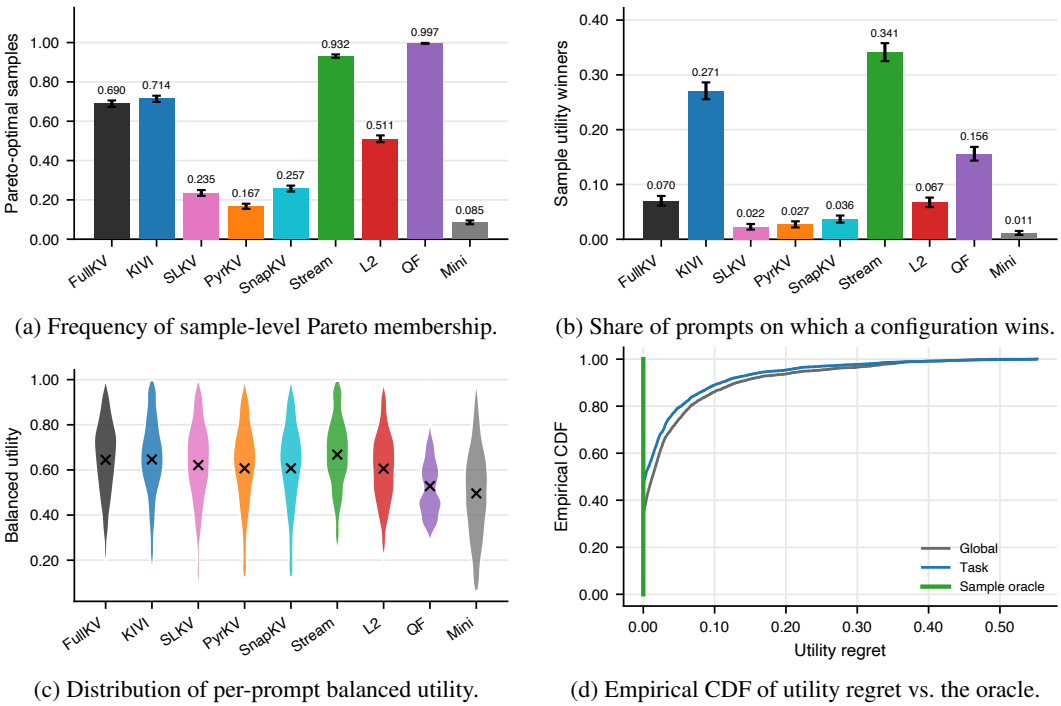
C.3 Sample-Level Heterogeneity and Regret Curves

Figure 15 presents sample-level performance distributions across all $3{,}200$ prompts. **No single dominant compressor** emerges: StreamingLLM achieves peak utility on 34.1% of prompts, KIVI-4bit on 27.1%, Q-Filters on 15.6%, FullKV on 7.0%, L2Compress on 6.7%, SnapKV on 3.6%, PyramidKV on 2.7%, SimLayerKV on 2.2%, and MiniCache on 1.1%. Every configuration is optimal on some subset of prompts, including MiniCache, which is dominated at every benchmark and task aggregate — a direct demonstration that aggregate domination does not imply instance-level irrelevance. Sample-level Pareto membership is similarly distributed: Q-Filters is non-dominated on 99.7% of individual prompts, StreamingLLM on 93.2%, KIVI-4bit on 71.4%, and FullKV on 69.0%, while MiniCache still attains membership on 8.5%.

Sample regret distributions quantify what static policies give up. Static global selection incurs a median regret of 0.013 utility points against the sample oracle, but the distribution is heavily right-tailed: the 90th percentile reaches 0.131, more than $20\times$ the entire global-to-task-type gap. Conditioning on task type reduces median regret to 0.003 but barely affects the tail, which is the distributional form of the 87.7% residual headroom reported in Section 3.5. Figure 16 resolves the same data prompt by prompt.

C.4 Complementary Context Regimes

To evaluate whether the absence of a universal best compressor holds under varying operational context regimes, we extend evaluation to RULER-4096 (Hsieh et al., 2024) and LongBench-v2 (Bai et al., 2025). These bracket the primary study across prompt length, generation length, and total time spent attending over cached context. The purpose is replication rather than new analysis: we ask whether the three structural findings of Section 3 — multi-membered frontiers, the prefill/decode inversion, and headroom below task identity — survive a change of context regime. Table 18 and Figure 17 summarize both cohorts side by side, and Appendices C.4.1–C.4.2 report each individ-



(a) Frequency of sample-level Pareto membership.

(b) Share of prompts on which a configuration wins.

(c) Distribution of per-prompt balanced utility.

(d) Empirical CDF of utility regret vs. the oracle.

Figure 15: **Sample-level analysis across** 3,200 **LongBench prompts.** Every configuration is Pareto-optimal on some prompts and optimal on some prompts, and none is optimal on a majority. Panel (d) plots the regret that each static policy incurs against the sample oracle (↓).

ually in the same form: benchmark-level means for all nine configurations under both memory accountings, per-task frontier membership, and the balanced-utility partition. Conventions follow Table 1 throughout ($\ell = L_3$, relative changes against the paired FullKV execution, U_λ at the balanced weighting), so only what is specific to each cohort is discussed below. Across both regimes, multi-membered Pareto frontiers persist.

Prompt-Length Quartile Analysis on LongBench. Before comparing cross-benchmark regimes we evaluate context-length sensitivity within LongBench by dividing its 3,200 prompts into context-length quartiles (Tab. 19, Fig. 18). Mean FullKV peak memory increases from 8.08 GiB in Q1 to 11.28 GiB in Q4. As context volume grows, relative savings expand for token eviction and quantization (StreamingLLM decode savings grow from 7.7 in Q1 to 26.3% in Q4; KIVI-4bit from 8.2% to 29.7%) while remaining flat for layer-level reduction (SimLayerKV, 1.6% to 1.8%), which removes a fixed fraction of layers irrespective of length. The peak-memory penalty of attention-based eviction also scales with length: SnapKV moves from +9.1% peak savings in Q1 to −12.9% in Q4 as its prefill scoring transient outgrows its cache saving.

C.4.1 RULER-4096: Synthetic Retrieval Stress Test

RULER-4096 evaluates synthetic needle-in-a-haystack and aggregation tasks over 6,500 prompts and 13 subtasks. Peak memory is modest (7.04 GiB FullKV), but synthetic retrieval separates compression heuristics sharply on accuracy, from 93.5 (FullKV) down to 5.1 (Q-Filters). Five of nine configurations are non-dominated under peak memory (FullKV, KIVI-4bit, SimLayerKV, StreamingLLM, Q-Filters) and six under decode memory, where SnapKV joins; per-task frontiers admit three to seven configurations (mean 4.92). The choice-granularity ladder runs 0.761 → 0.776 → 0.803, leaving 64.3% of headroom beneath task identity — the lowest residual of any cohort, because RULER's subtasks are far more homogeneous in structure than LongBench's task families. Even here, all nine configurations win at least one prompt and the most frequent winner (KIVI-4bit) covers only 61.8%. Results are given in Table 20 and Figures 19, 20, 21, 22 and 23.

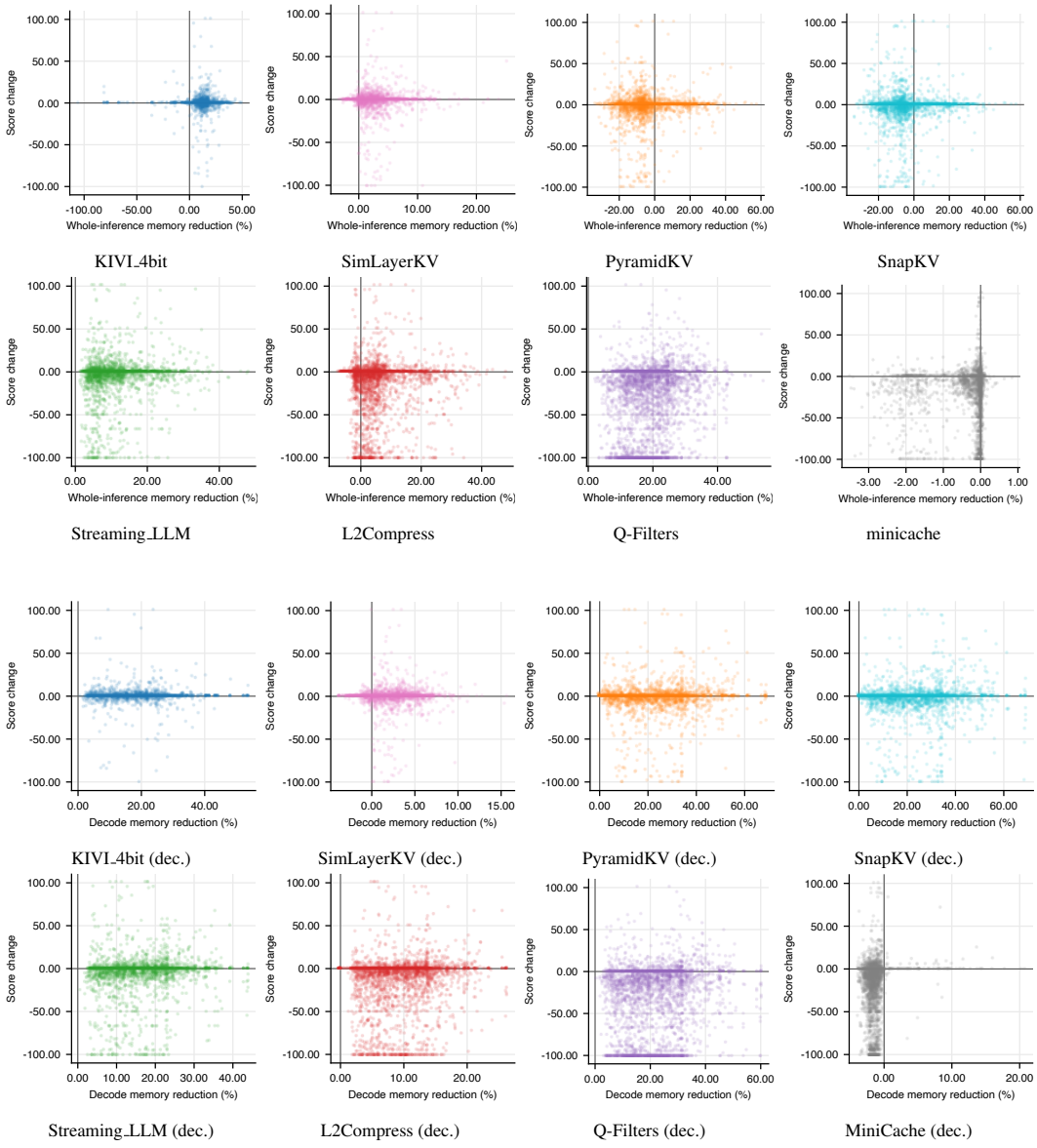
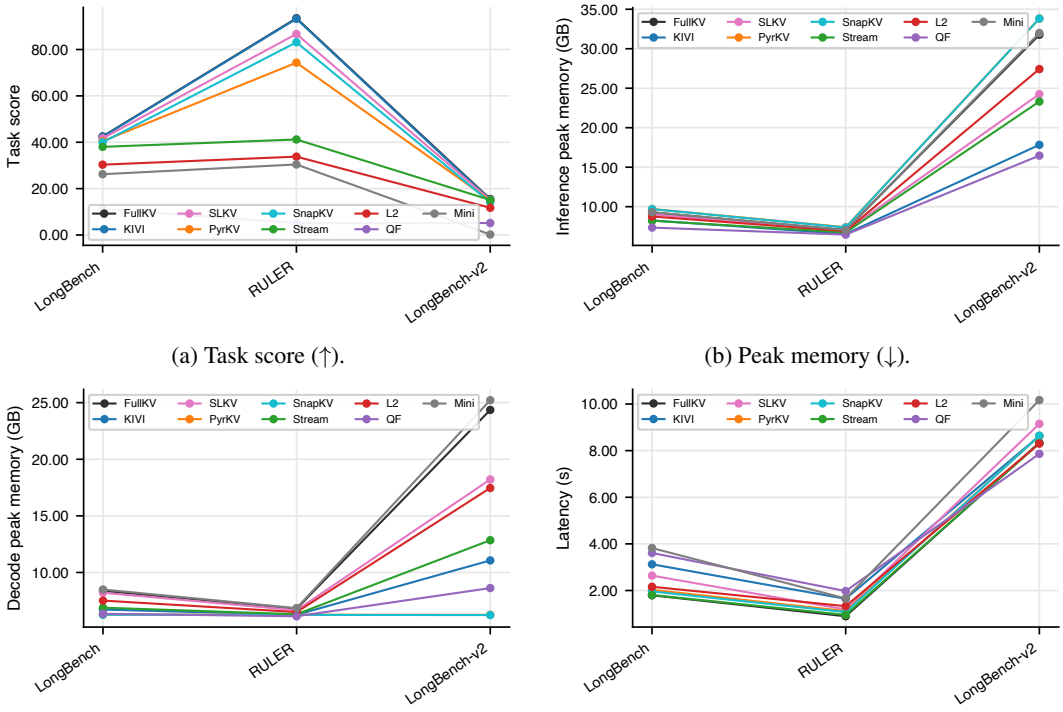


Figure 16: **Compressors that look accuracy-neutral on average introduce non-uniform per-prompt gains and losses.** Prompt-level distributions on the score-change versus memory-reduction plane relative to FullKV; rightwards is greater memory saving, upwards is accuracy improvement. Top two rows use whole-inference peak memory, bottom two rows decode-phase memory. Memory savings vary widely across prompts because of context-length variation, and score changes show a sharp peak at zero with a heavy two-sided tail.

Table 18: **Multi-membered frontiers persist across both secondary cohorts.** Two secondary benchmarks under identical conditions at $\ell = L3$; both are single-run cohorts. Memory and latency columns are paired % changes against FullKV. The last column marks non-domination under peak / decode memory.

Configuration	$A \uparrow$	$M_{\text{peak}} \downarrow$	saved%	$M_{\text{dec}} \downarrow$	saved%	$L \downarrow$	added%	Pareto
RULER-4096 (n = 6,500, 13 tasks, FullKV peak 7.04 GiB)								
FullKV	93.48	7.04	+0.0	6.80	+0.0	0.90	+0.0	✓ / ✓
KIVI-4bit	93.23	6.55	+7.0	6.26	+7.9	1.64	+83.4	✓ / ✓
SimLayerKV	86.71	6.92	+1.8	6.67	+2.0	1.19	+32.4	✓ / ✓
PyramidKV	74.32	7.40	-5.0	6.27	+7.8	1.12	+24.8	– / –
SnapKV	83.12	7.36	-4.5	6.25	+8.0	1.07	+19.9	– / ✓
StreamingLLM	41.19	6.75	+4.2	6.31	+7.2	0.95	+6.4	✓ / ✓
L2Compress	33.79	6.92	+1.8	6.50	+4.4	1.33	+48.2	– / –
Q-Filters	5.10	6.47	+8.2	6.13	+9.8	1.98	+120.5	✓ / ✓
MiniCache	30.41	7.05	-0.1	6.85	-0.7	1.66	+84.9	– / –
LongBench-v2 (n = 503, 16 subdomains, FullKV peak 31.77 GiB)								
FullKV	15.51	31.77	+0.0	24.35	+0.0	8.34	+0.0	✓ / ✓
KIVI-4bit	15.11	17.83	+43.9	11.06	+54.6	8.63	+3.5	✓ / ✓
SimLayerKV	15.11	24.25	+23.6	18.22	+25.2	9.15	+9.7	– / –
PyramidKV	14.91	33.83	-6.5	6.26	+74.3	8.63	+3.4	– / ✓
SnapKV	14.12	33.82	-6.5	6.24	+74.4	8.64	+3.6	– / ✓
StreamingLLM	15.11	23.32	+26.6	12.85	+47.2	8.32	-0.3	✓ / ✓
L2Compress	11.73	27.43	+13.7	17.46	+28.3	8.30	-0.5	✓ / ✓
Q-Filters	5.17	16.46	+48.2	8.62	+64.6	7.86	-5.7	✓ / ✓
MiniCache	0.20	31.99	-0.7	25.22	-3.5	10.17	+21.9	– / –

Llama-3.2-3B-Instruct scores below the 25% random baseline on LongBench-v2, so that cohort is read as a memory- and latency-regime probe only (Sec. 3.5).



(a) Task score (↑).

(b) Peak memory (↓).

(c) Decode memory (↓).

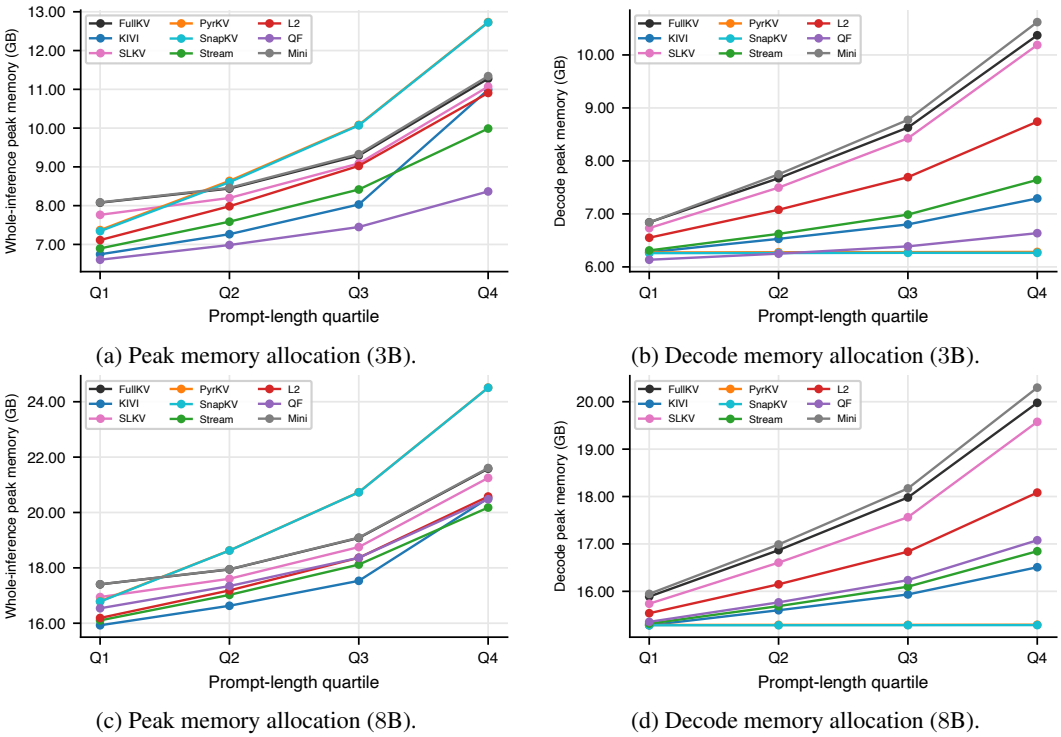
(d) Latency (↓).

Figure 17: **The value of compression moves with the context regime, not with the algorithm alone.** Relative performance across the three benchmarks, ordered by FullKV peak memory.

Table 19: **Relative memory savings grow with context length for eviction and quantization, but shrink for layer-level reduction.** LongBench split into prompt-length quartiles. A is task score; M columns are % memory saved against the paired FullKV run within the same quartile.

	Q1 (shortest)			Q2			Q3			Q4 (longest)		
Configuration	A	M_{pk}	M_{dec}	A	M_{pk}	M_{dec}	A	M_{pk}	M_{dec}	A	M_{pk}	M_{dec}
FullKV	47.3	+0.0	+0.0	43.5	+0.0	+0.0	43.9	+0.0	+0.0	35.4	+0.0	+0.0
KIVI-4bit	47.0	+16.5	+8.2	43.4	+13.9	+14.9	43.9	+13.6	+21.2	35.2	+2.5	+29.7
SimLayerKV	45.7	+3.9	+1.6	42.7	+2.9	+2.3	43.6	+2.2	+2.3	34.8	+1.8	+1.8
PyramidKV	45.3	+8.8	+8.3	41.9	-2.3	+18.2	42.2	-8.5	+27.2	32.3	-12.9	+39.4
SnapKV	44.6	+9.1	+8.5	42.0	-1.9	+18.4	42.0	-8.4	+27.4	31.4	-12.9	+39.6
StreamingLLM	40.7	+14.6	+7.7	39.7	+10.1	+13.7	40.1	+9.4	+19.0	31.6	+11.4	+26.3
L2Compress	31.1	+11.9	+4.2	31.0	+5.4	+7.8	31.8	+2.9	+10.8	27.4	+3.3	+15.7
Q-Filters	13.0	+18.2	+10.3	10.0	+17.3	+18.6	10.6	+19.8	+26.0	11.4	+25.8	+36.0
MiniCache	35.5	-0.1	+0.0	26.6	-0.2	-0.9	25.7	-0.4	-1.7	17.1	-0.6	-2.4

Quartile boundaries: 111–3602–6207–9460–36418 prompt tokens. FullKV mean peak memory rises from 8.08 GiB (Q1) to 11.28 GiB (Q4).



(a) Peak memory allocation (3B).

(b) Decode memory allocation (3B).

(c) Peak memory allocation (8B).

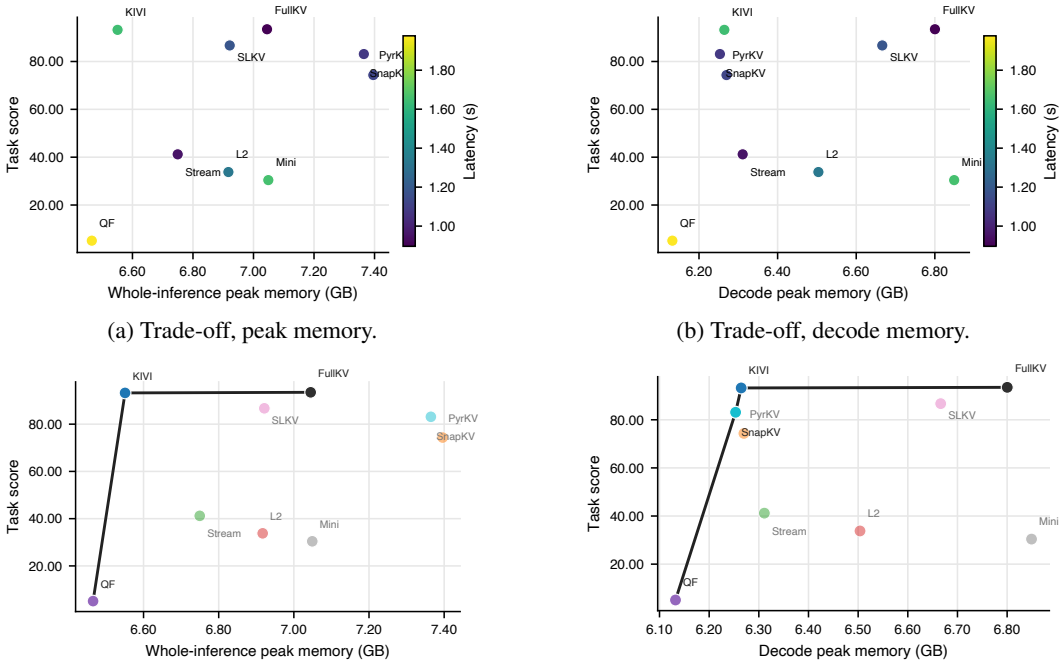
(d) Decode memory allocation (8B).

Figure 18: **Absolute GPU memory allocations scale with prompt length at method-dependent rates, identically at both scales.** Allocated GPU memory in GiB (↓) across LongBench prompt-length quartiles; the relative savings are given in Table 19 (Q1 shortest, Q4 longest).

Table 20: **RULER-4096: the frontier is multi-membered and splits by memory accounting.** $n = 6{,}500$ prompts, 13 tasks, single run. Columns follow Tab. 1; Par counts non-dominated tasks (of 13).

Configuration	$A \uparrow$	$M_{\text{peak}} \downarrow$	saved%	$M_{\text{dec}} \downarrow$	saved%	$L \downarrow$	added%	$\text{Par}_{\text{pk}} \uparrow$	$\text{Par}_{\text{dec}} \uparrow$	$U_\lambda \uparrow$
KIVI-4bit	93.23	6.55	+7.0	6.26	+7.9	1.64	+83.4	13	10	0.761
FullKV	93.48	7.04	+0.0	6.80	+0.0	0.90	+0.0	12	11	0.695
SimLayerKV	86.71	6.92	+1.8	6.67	+2.0	1.19	+32.4	8	3	0.678
StreamingLLM	41.19	6.75	+4.2	6.31	+7.2	0.95	+6.4	10	6	0.611
SnapKV	83.12	7.36	-4.5	6.25	+8.0	1.07	+19.9	2	12	0.533
L2Compress	33.79	6.92	+1.8	6.50	+4.4	1.33	+48.2	3	1	0.491
PyramidKV	74.32	7.40	-5.0	6.27	+7.8	1.12	+24.8	3	7	0.489
Q-Filters	5.10	6.47	+8.2	6.13	+9.8	1.98	+120.5	13	13	0.462
MiniCache	30.41	7.05	-0.1	6.85	-0.7	1.66	+84.9	0	0	0.392

Benchmark-level frontier under peak memory: FullKV, KIVI-4bit, SimLayerKV, StreamingLLM, Q-Filters. Per-task frontier size 3–7 (mean 4.92). Choice granularities: global 0.761 (KIVI-4bit) → task type 0.776 → sample oracle 0.803, leaving 64.3% of headroom beneath task identity. RULER-4096 separates the pool sharply on accuracy while leaving memory differences small.



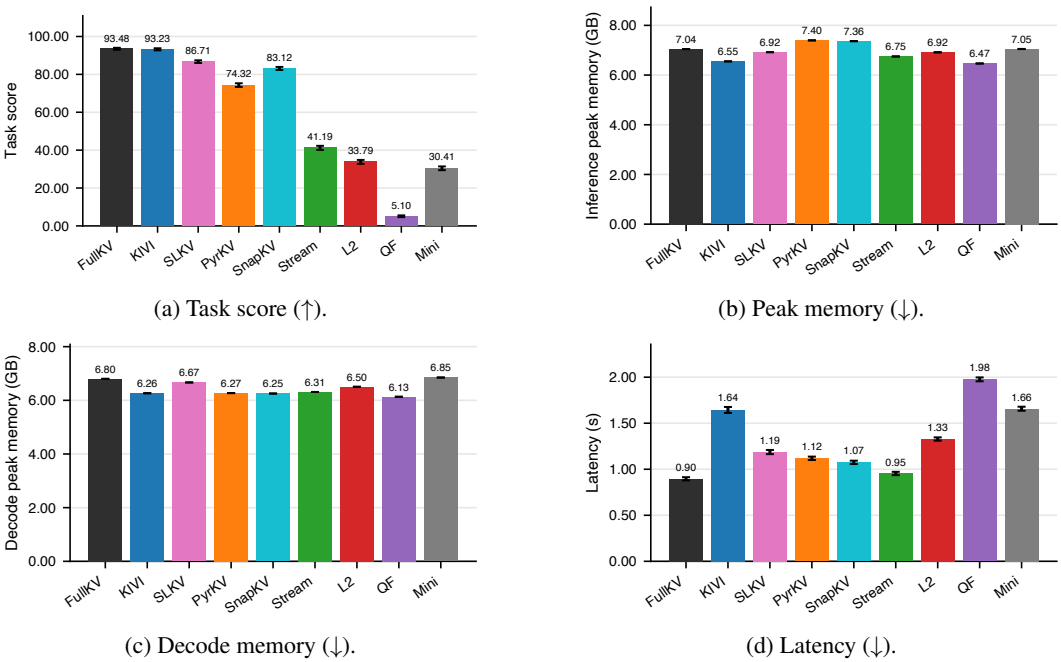
(a) Trade-off, peak memory.

(b) Trade-off, decode memory.

(c) 2D frontier, peak memory.

(d) 2D frontier, decode memory.

Figure 19: **RULER-4096: the frontier is multi-membered under both memory accountings.** Score (↑) against memory (↓) with latency in colour, and the corresponding two-dimensional frontiers. Configurations dominated in 2D can still be non-dominated once latency enters.



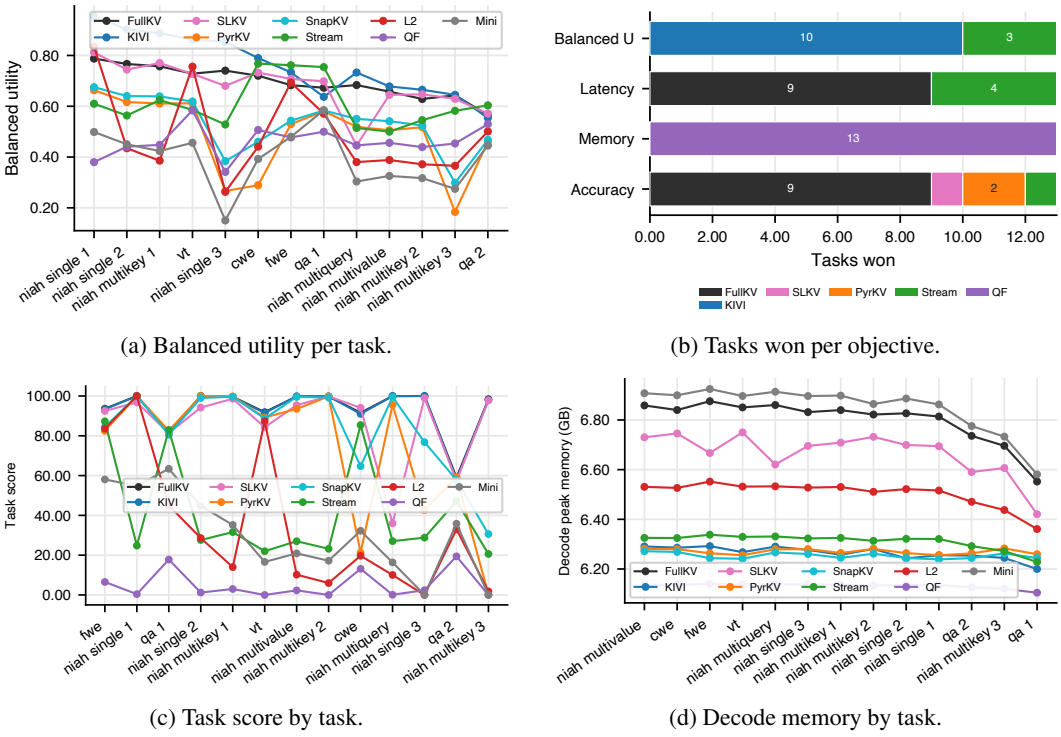
(a) Task score (↑).

(b) Peak memory (↓).

(c) Decode memory (↓).

(d) Latency (↓).

Figure 20: **RULER-4096: each measured axis orders the pool differently.** Benchmark-level means with 95% confidence intervals. Intervals are narrow on the system axes and wide on accuracy, matching the pattern reported for LongBench in App. C.2.



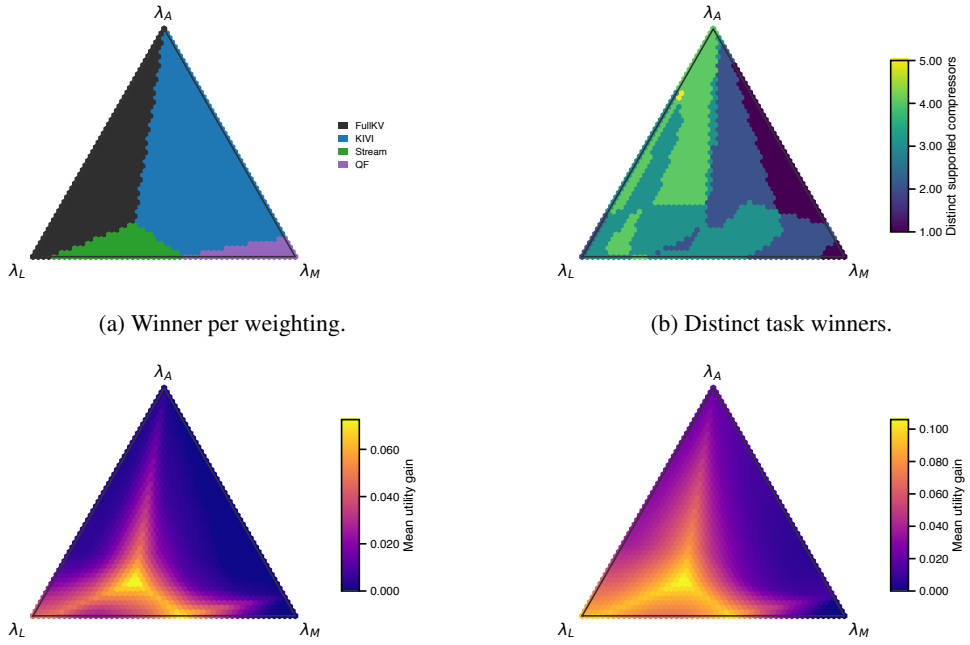
(a) Balanced utility per task.

(b) Tasks won per objective.

(c) Task score by task.

(d) Decode memory by task.

Figure 21: **RULER-4096: intersecting per-task profiles and disagreeing objectives.** Utility and raw-metric profiles across the constituent tasks, with the tasks-won breakdown per single objective.



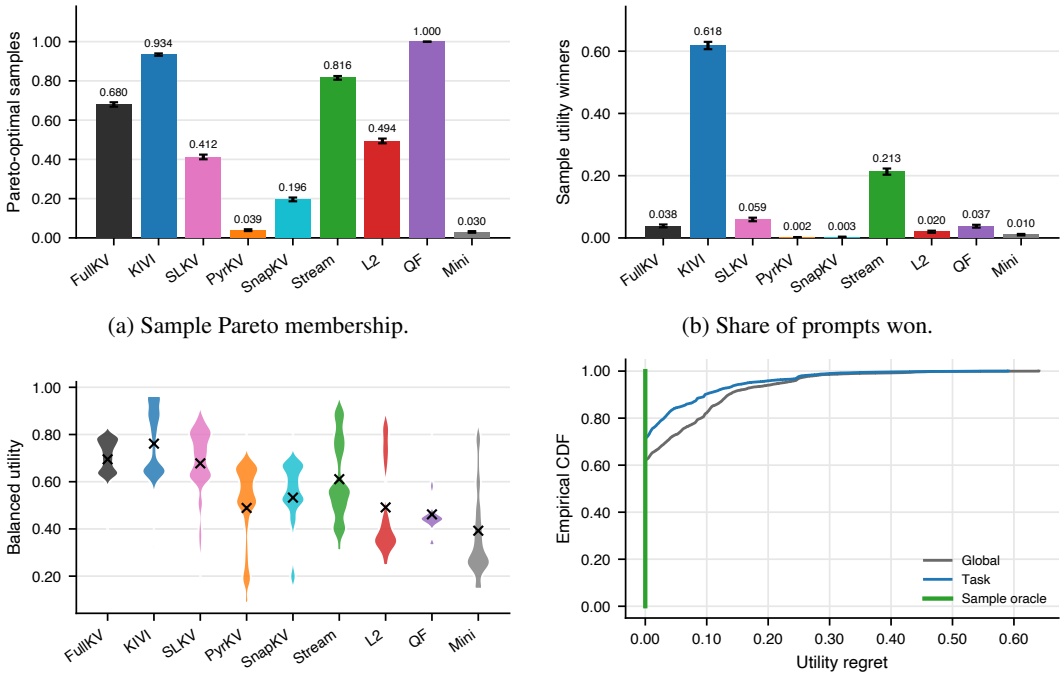
(a) Winner per weighting.

(b) Distinct task winners.

(c) Gain: task type over global.

(d) Gain: oracle over global.

Figure 22: **RULER-4096: preference sweeps over the weight simplex Δ^2.** Winning configuration and task-level disagreement at every deployment weighting, together with the utility recovered by category-level and instance-level selection.



(a) Sample Pareto membership.

(b) Share of prompts won.

(c) Per-prompt utility.

(d) Regret ECDF vs. oracle.

Figure 23: **RULER-4096: instance-level heterogeneity persists.** Sample-level Pareto membership, winner shares, per-prompt utility distributions, and the empirical CDF of utility regret against the sample oracle (↓).

C.4.2 LONGBENCH-V2: ULTRA-LONG CONTEXT REGIME

LongBench-v2 evaluates multiple-choice comprehension over 503 prompts spanning 16 subdomains at extreme context lengths (31.77 GiB FullKV peak). Memory savings expand dramatically: KIVI-4bit cuts peak memory by 43.9% and decode memory by 54.6% at $+3.5\%$ latency, and Q-Filters reaches 48.2% and 64.6%. The divergence between the two accountings is the widest in the study: per-task frontiers admit one to five configurations under peak memory but three to seven under decode memory.

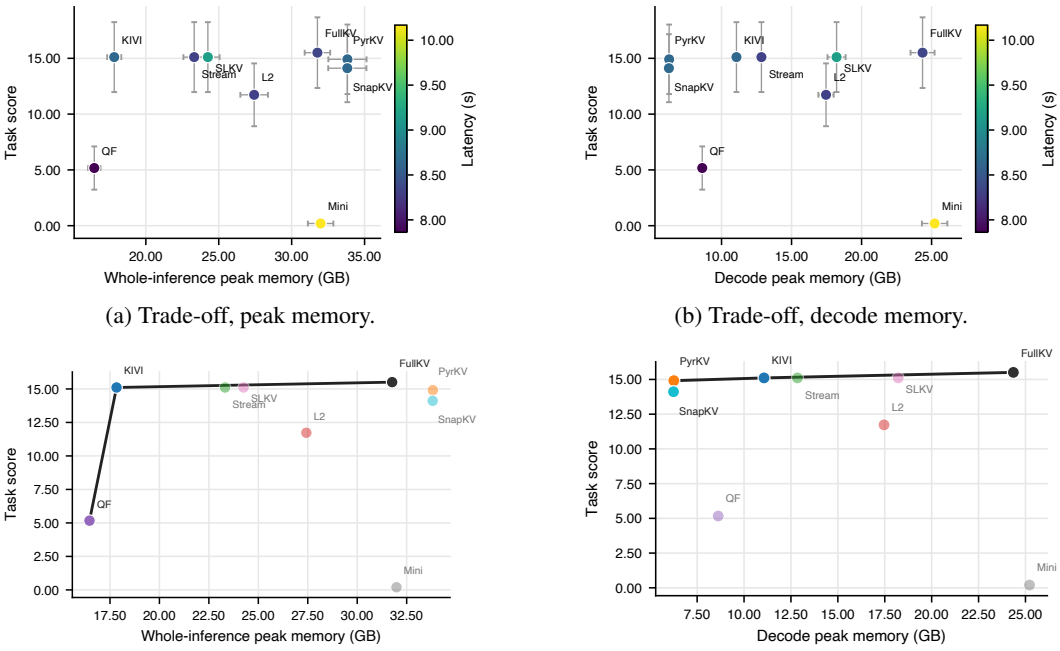
Accuracy on this cohort is uninformative, and the caveat propagates. The scoring format is four-way multiple choice, giving a 25% random baseline, and Llama-3.2-3B-Instruct attains 15.5% with FullKV, with every configuration between 0.2% and 15.5%. Because $\hat{A}$ enters Eq. 4 with equal weight, composite utility here scales sub-chance variation as though it were task competence. The U_λ column, the simplex partition and the 86.3% residual headroom in Table 21 are therefore read as memory- and latency-driven: Q-Filters attaining the highest balanced utility (0.514) reflects its memory and latency minima rather than a quality advantage, and we do not rank compressors by accuracy here. While accuracy sits at floor level for a 3B model, the cohort is a valid empirical probe of ultra-long-context memory and latency dynamics (31.8 GiB allocations), which is exactly the regime in which the selector results of Section 6.2 are the most consequential: there KVRoute successfully triggers high-compression routing, serving a_{ref} on 90.9% of requests for 38.2% less peak memory. That comparison is unaffected by the accuracy floor, because every policy is replayed against the same measured outcome matrix and therefore scored on identical per-request outcomes — the replay compares decisions, not the backbone's competence.

The lower bound of one non-dominated configuration is a direct consequence, and deserves comment since a singleton frontier is by construction a counterexample to this paper's central claim. It occurs on exactly one of the sixteen subdomains, `Legal` ($n = 33$), where Q-Filters simultaneously attains the highest score (12.1), the lowest peak memory (12.4 GiB) and the lowest latency (3.67 s). The accuracy component of that dominance is noise: 12.1 points on 33 prompts against a 25-point random baseline, so the ordering the score axis induces there is arbitrary. The singleton also does not survive the change of memory accounting — under decode memory the same subdomain admits three non-dominated configurations. We report it rather than suppress it, but it is an artifact of a sub-chance accuracy axis on a small subdomain, not evidence of a dominating compressor. Results are given in Table 21 and Figures 24, 25, 26, 27 and 28.

Table 21: **LongBench-v2: the frontier is multi-membered and splits by memory accounting.** $n = 503$ prompts, 16 subdomains, single run. Columns follow Tab. 1; Par counts non-dominated tasks (of 16).

Configuration	$A \uparrow$	$M_{\text{peak}} \downarrow$	saved%	$M_{\text{dec}} \downarrow$	saved%	$L \downarrow$	added%	$\text{Par}_{\text{pk}} \uparrow$	$\text{Par}_{\text{dec}} \uparrow$	$U_\lambda \uparrow$
Q-Filters	5.17	16.46	+48.2	8.62	+64.6	7.86	-5.7	16	16	0.514
KIVI-4bit	15.11	17.83	+43.9	11.06	+54.6	8.63	+3.5	13	11	0.511
StreamingLLM	15.11	23.32	+26.6	12.85	+47.2	8.32	-0.3	15	15	0.469
SimLayerKV	15.11	24.25	+23.6	18.22	+25.2	9.15	+9.7	3	2	0.435
L2Compress	11.73	27.43	+13.7	17.46	+28.3	8.30	-0.5	10	10	0.420
FullKV	15.51	31.77	+0.0	24.35	+0.0	8.34	+0.0	3	3	0.393
PyramidKV	14.91	33.83	-6.5	6.26	+74.3	8.63	+3.4	0	13	0.362
SnapKV	14.12	33.82	-6.5	6.24	+74.4	8.64	+3.6	0	16	0.359
MiniCache	0.20	31.99	-0.7	25.22	-3.5	10.17	+21.9	0	0	0.286

Benchmark-level frontier under peak memory: FullKV, KIVI-4bit, StreamingLLM, L2Compress, Q-Filters. Per-task frontier size 1–5 (mean 3.75). Choice granularities: global 0.514 (Q-Filters) $\rightarrow$ task type $0.523 \rightarrow$ sample oracle 0.580, leaving 86.3% of headroom beneath task identity. Accuracy on this cohort lies below the 25% random baseline.



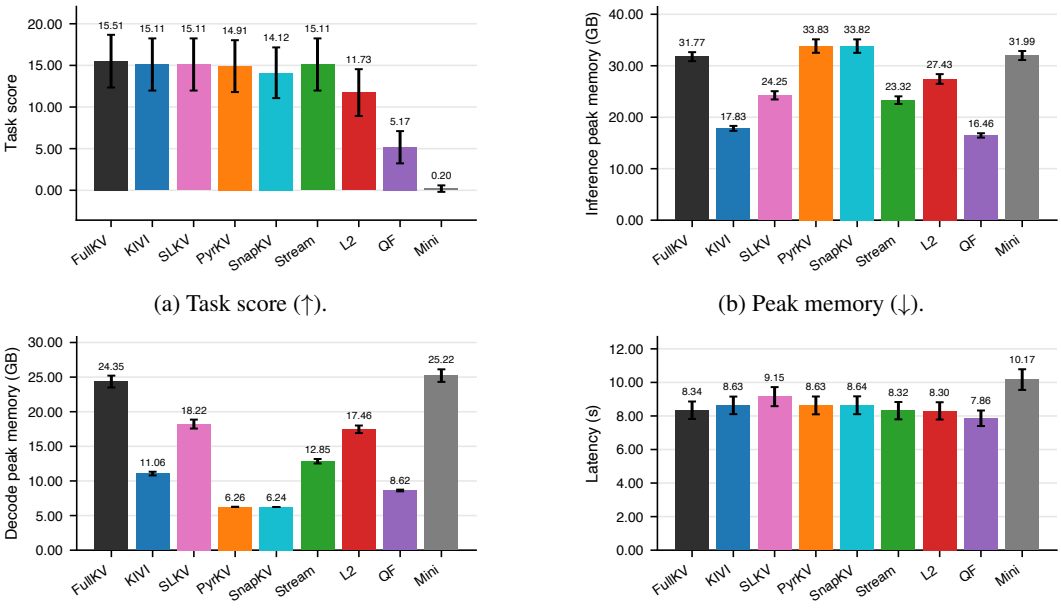
(a) Trade-off, peak memory.

(b) Trade-off, decode memory.

(c) 2D frontier, peak memory.

(d) 2D frontier, decode memory.

Figure 24: **LongBench-v2: the frontier is multi-membered under both memory accountings.** Score (↑) against memory (↓) with latency in colour, and the corresponding two-dimensional frontiers. Configurations dominated in 2D can still be non-dominated once latency enters.



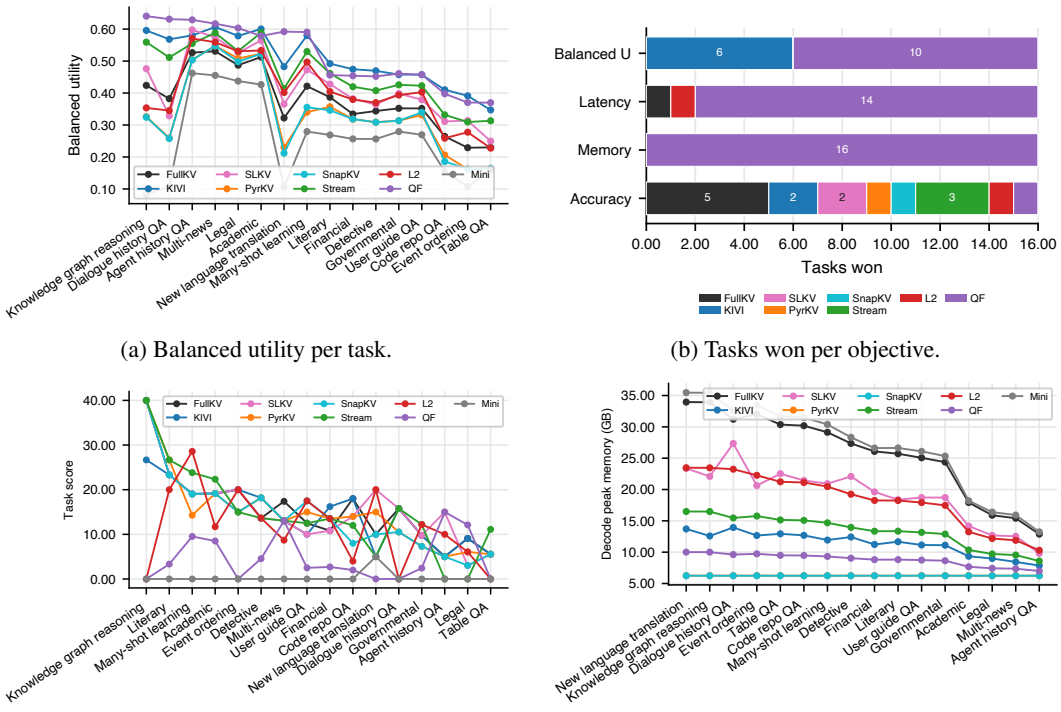
(a) Task score (↑).

(b) Peak memory (↓).

(c) Decode memory (↓).

(d) Latency (↓).

Figure 25: **LongBench-v2: each measured axis orders the pool differently.** Benchmark-level means with 95% confidence intervals. Intervals are narrow on the system axes and wide on accuracy, matching the pattern reported for LongBench in App. C.2.



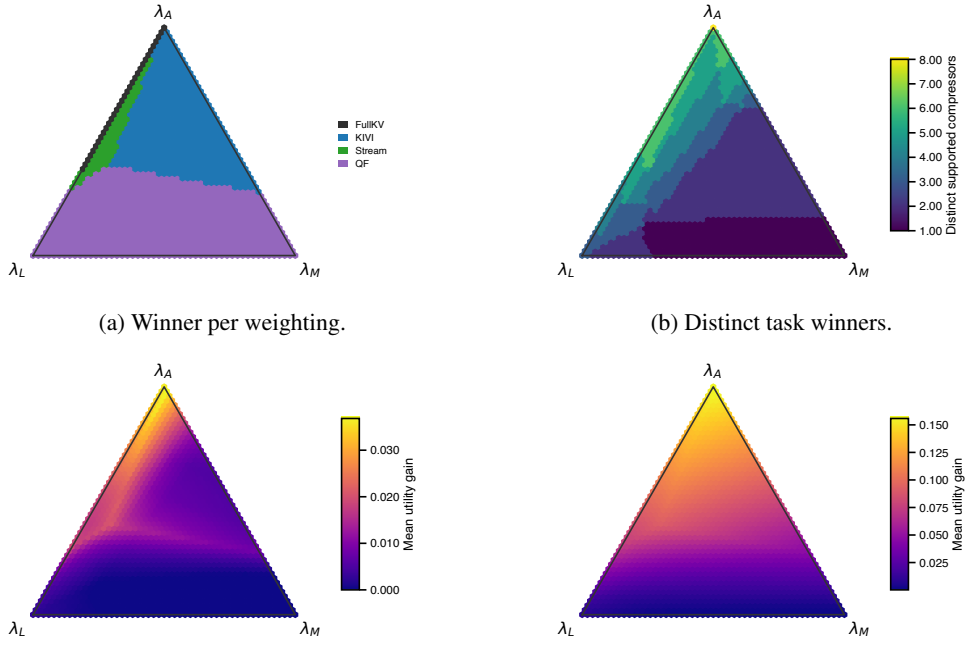
(a) Balanced utility per task.

(b) Tasks won per objective.

(c) Task score by task.

(d) Decode memory by task.

Figure 26: **LongBench-v2: intersecting per-task profiles and disagreeing objectives.** Utility and raw-metric profiles across the constituent tasks, with the tasks-won breakdown per single objective.



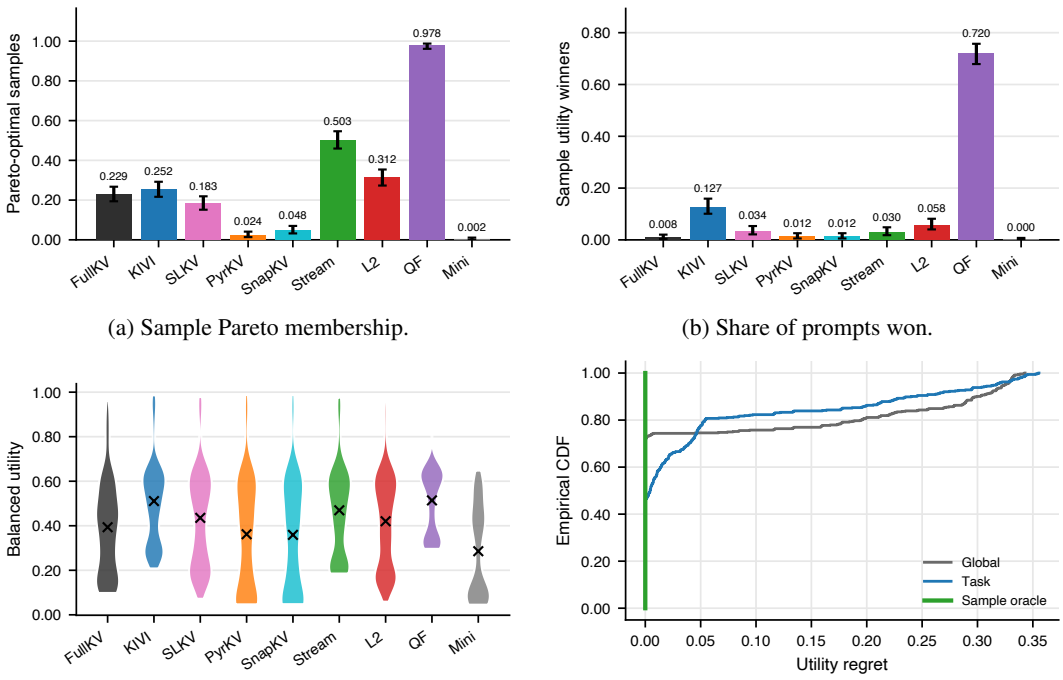
(a) Winner per weighting.

(b) Distinct task winners.

(c) Gain: task type over global.

(d) Gain: oracle over global.

Figure 27: **LongBench-v2: preference sweeps over the weight simplex Δ^2.** Winning configuration and task-level disagreement at every deployment weighting, together with the utility recovered by category-level and instance-level selection.



(a) Sample Pareto membership.

(b) Share of prompts won.

(c) Per-prompt utility.

(d) Regret ECDF vs. oracle.

Figure 28: **LongBench-v2: instance-level heterogeneity persists.** Sample-level Pareto membership, winner shares, per-prompt utility distributions, and the empirical CDF of utility regret against the sample oracle (↓).

C.5 Model Scale Generalization (8B Backbone Cohort)

To examine whether three-dimensional Pareto non-domination generalizes to larger parameter architectures, we execute three end-to-end repeated runs of the primary $\ell = L_3 = 0.75$ setting on Llama-3.1-8B-Instruct (86,400 matched measurement triples), plus a $\ell = L_2 = 0.50$ cohort over two runs. The cohort is reported with the same five-block pattern used for the complementary benchmarks in Appendix C.4.

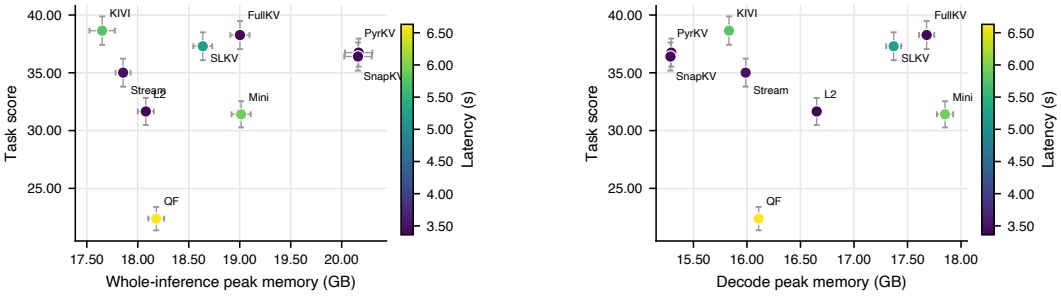
Three findings emerge. First, **frontier persistence and ranking correlation**: at 8B, multi-membered Pareto frontiers persist across all workloads, with tasks admitting between three and seven non-dominated configurations (mean 5.06). StreamingLLM maintains top global balanced utility ($U = 0.646$), and per-configuration utility rankings correlate with the 3B cohort at Spearman $\rho = 0.70$ ($p = 0.036$). Second, **the memory-accounting split reproduces**: SnapKV and PyramidKV again increase peak allocation (-6.1% savings each) while cutting decode memory by 13.5%, and the benchmark frontier again differs between accountings (five members under peak, six under decode). Third, **L2Compress performance shift**: at 8B, L2Compress rises from seventh to second in overall balanced utility ($U = 0.632$), incurring less relative accuracy degradation while executing marginally faster than FullKV (-0.9% latency). Category-level selection recovers $+0.0022$ utility over static global choice, leaving 93.5% of available headroom ($m^\star = 0.681$) uncaptured beneath the task-category level. Instance-level utility headroom is therefore a property of KV-cache compression that holds and strengthens across both evaluated parameter scales.

The 8B cohort is also the sharpest demonstration that the third axis is load-bearing. In the accuracy-versus-peak-memory plane alone — the projection the compression literature reports — KIVI-4bit attains both the highest score (38.65) and the lowest peak allocation (17.65 GiB), dominating all eight alternatives, and the frontier collapses to a single configuration (Fig. 29c). It is only once realized latency enters that five configurations become non-dominated. The same 2D projection on 3B admits four of nine against five in 3D, so the gap between the two-dimensional and three-dimensional views widens with scale rather than closing. KIVI-4bit's accuracy advantage at 8B is $+0.38$ points with a 95% prompt-clustered bootstrap CI of $[+0.08, +0.69]$, excluding zero; the paired signed-rank test does not resolve it ($p = 0.41$), since the median per-prompt difference is zero and the advantage lives in the tail. The 2D singleton is therefore supported on mean accuracy, and we do not claim it at the level of the typical prompt. Full results are given in Table 22 and Figures 29, 30, 31, 32 and 33; Figure 35 compares the two scales directly.

Table 22: **LongBench (Llama-3.1-8B): the frontier is multi-membered and splits by memory accounting.** $n = 3{,}200$ prompts, 16 tasks, three runs. Columns follow Tab. 1; Par counts non-dominated tasks (of 16).

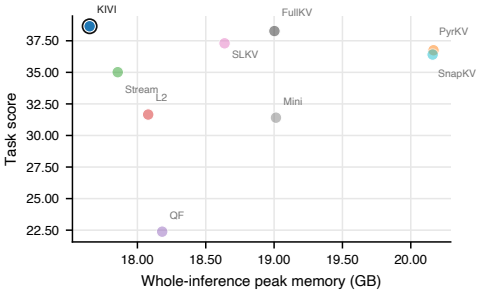
Configuration	$A \uparrow$	$M_{\text{peak}} \downarrow$	saved%	$M_{\text{dec}} \downarrow$	saved%	$L \downarrow$	added%	$\text{Par}_{\text{pk}} \uparrow$	$\text{Par}_{\text{dec}} \uparrow$	$U_\lambda \uparrow$
StreamingLLM	35.01	17.86	+6.0	15.99	+9.6	3.51	+3.5	16	7	0.646
L2Compress	31.66	18.08	+4.9	16.65	+5.8	3.36	-0.9	15	14	0.632
FullKV	38.27	19.00	+0.0	17.68	+0.0	3.39	+0.0	15	14	0.630
KIVI-4bit	38.65	17.65	+7.1	15.83	+10.4	5.76	+69.7	15	13	0.615
SimLayerKV	37.30	18.64	+1.9	17.37	+1.7	5.21	+53.7	10	4	0.592
PyramidKV	36.75	20.17	-6.1	15.29	+13.5	3.43	+1.3	4	12	0.583
SnapKV	36.41	20.16	-6.1	15.29	+13.5	3.41	+0.5	5	16	0.582
MiniCache	31.41	19.01	-0.1	17.85	-1.0	5.92	+74.5	0	0	0.539
Q-Filters	22.38	18.18	+4.3	16.11	+8.9	6.63	+95.5	1	0	0.518

Per-configuration utility rankings correlate with the 3B cohort at Spearman $\rho = 0.70$ ($p = 0.036$).

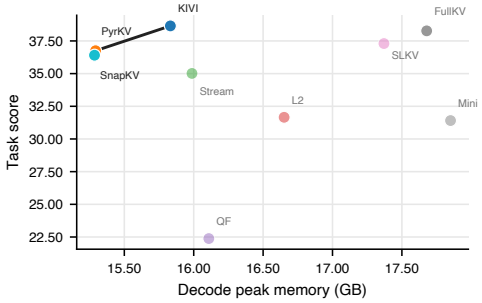


(a) Trade-off, peak memory.

(b) Trade-off, decode memory.

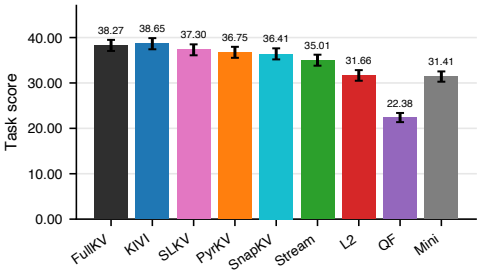


(c) 2D frontier, peak memory.

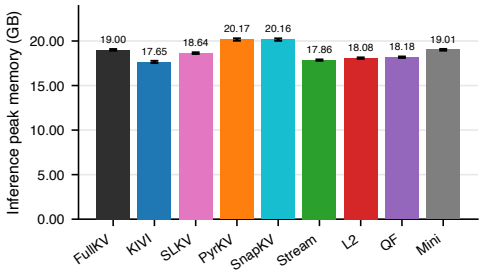


(d) 2D frontier, decode memory.

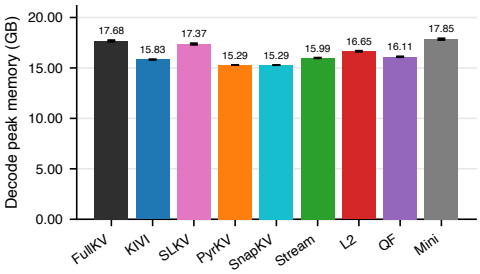
Figure 29: **LongBench, Llama-3.1-8B: the third axis is what makes the frontier multi-membered.** Score (↑) against memory (↓) with latency in colour **(a, b)**, and the corresponding two-dimensional frontiers **(c, d)**. Under accuracy versus peak memory alone, KIVI-4bit dominates all eight alternatives and the 2D frontier collapses to a single point **(c)**; adding realized latency expands it to five members. Under decode memory the 2D frontier already admits three **(d)**, and the 3D frontier six.



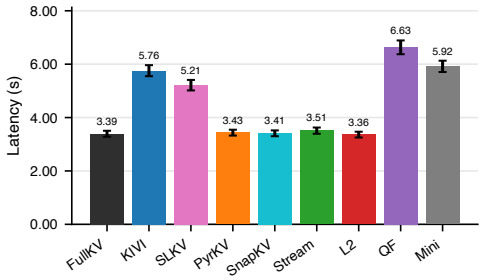
(a) Task score (↑).



(b) Peak memory (↓).

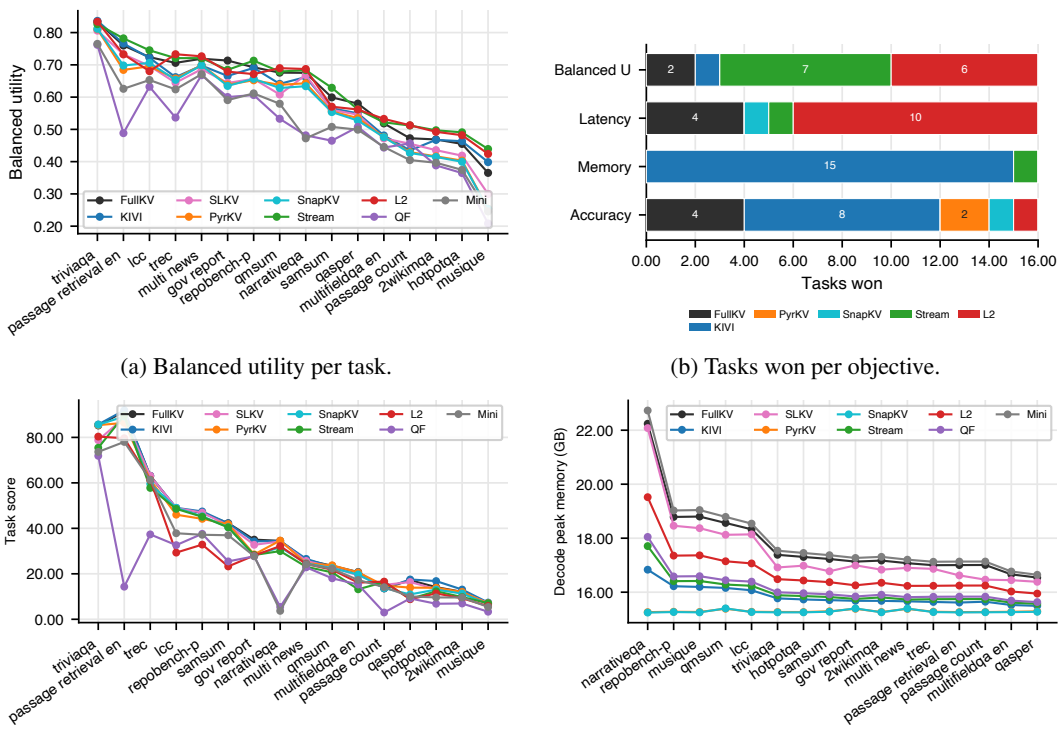


(c) Decode memory (↓).



(d) Latency (↓).

Figure 30: **LongBench, Llama-3.1-8B: each measured axis orders the pool differently.** Benchmark-level means with 95% confidence intervals. Intervals are narrow on the system axes and wide on accuracy, matching the pattern reported for LongBench in App. C.2.



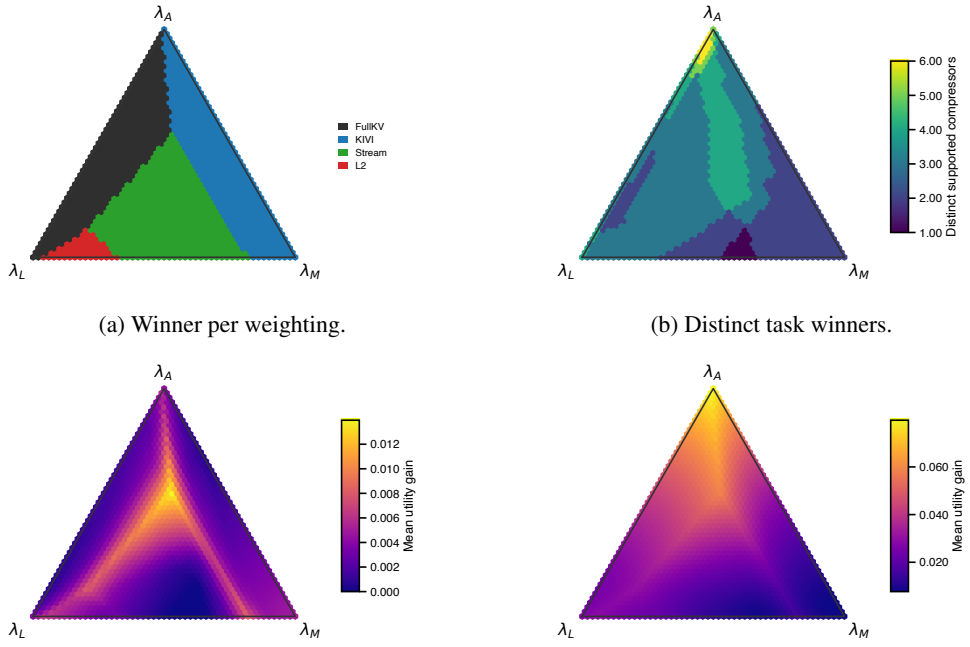
(a) Balanced utility per task.

(b) Tasks won per objective.

(c) Task score by task.

(d) Decode memory by task.

Figure 31: **LongBench, Llama-3.1-8B: intersecting per-task profiles and disagreeing objectives.** Utility and raw-metric profiles across the constituent tasks, with the tasks-won breakdown per single objective.



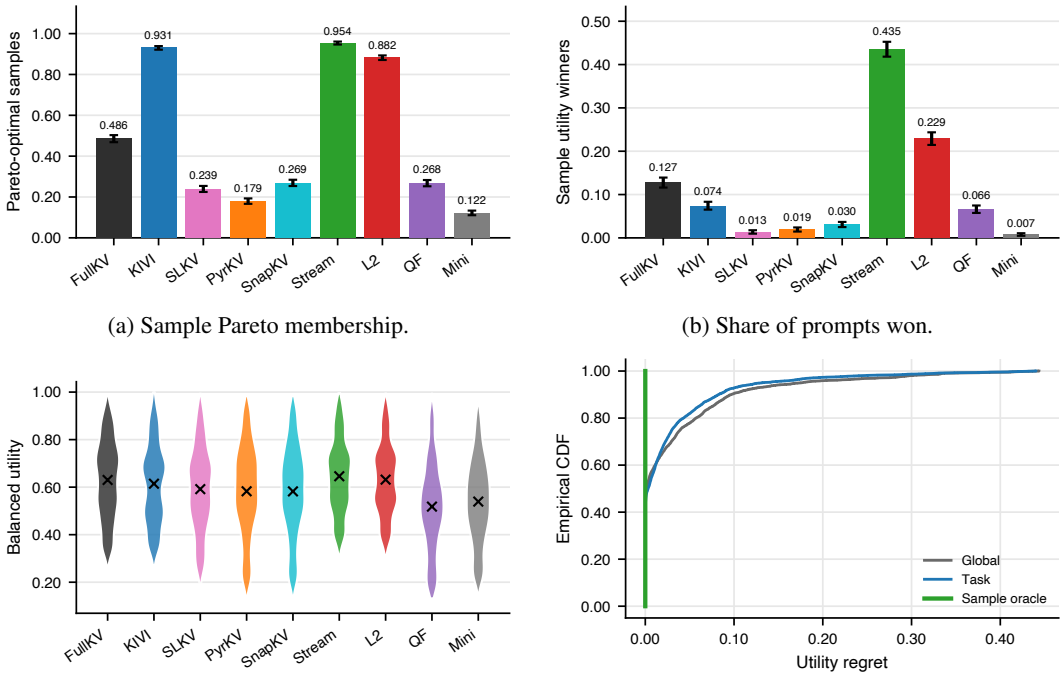
(a) Winner per weighting.

(b) Distinct task winners.

(c) Gain: task type over global.

(d) Gain: oracle over global.

Figure 32: **LongBench, Llama-3.1-8B: preference sweeps over the weight simplex Δ^2.** Winning configuration and task-level disagreement at every deployment weighting, together with the utility recovered by category-level and instance-level selection.



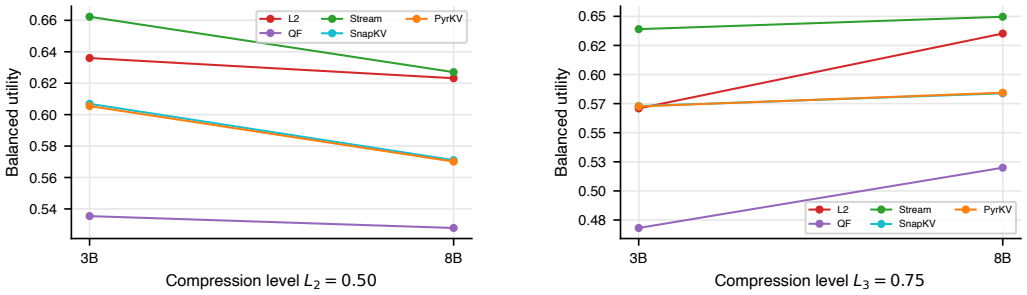
(a) Sample Pareto membership.

(b) Share of prompts won.

(c) Per-prompt utility.

(d) Regret ECDF vs. oracle.

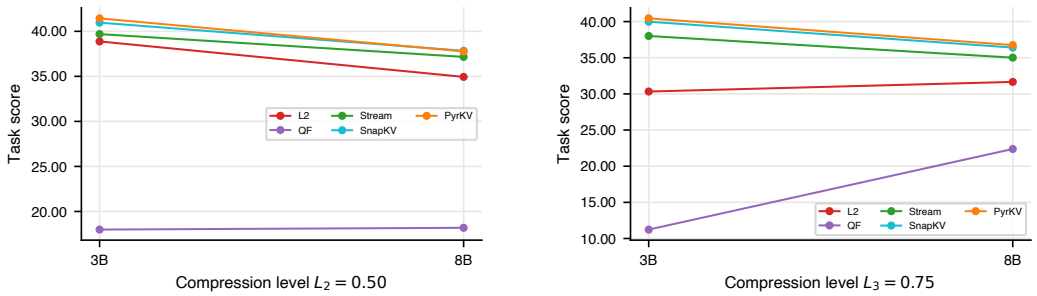
Figure 33: **LongBench, Llama-3.1-8B: instance-level heterogeneity persists.** Sample-level Pareto membership, winner shares, per-prompt utility distributions, and the empirical CDF of utility regret against the sample oracle (↓).



(a) Utility. (b) Utility.

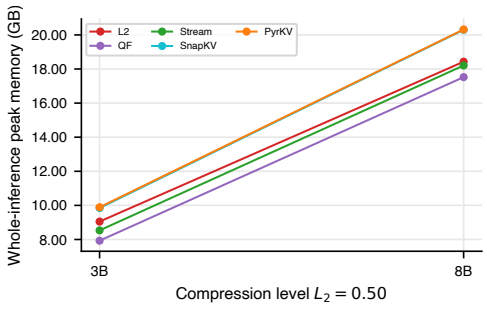
Figure 34: **8B gains utility on higher compression compared to 3B**, while both scales show the similar global configuration ranking on the utility axis. Direct 3B versus 8B comparison on LongBench at both compression levels for which an 8B cohort exists.

Figure 36 compares balanced utility per configuration across all cohorts, and the granularity ladder behind it is summarised numerically here: the residual headroom beneath task identity is 87.7% on LongBench, 64.3% on RULER-4096, 86.3% on LongBench-v2, and 93.5% on LongBench at 8B. The pooled simplex maps of Figures 36b and 36c show that both the recoverable gain and the number of supported frontier members vary continuously with the deployment weighting, and Figure 36a places the cohorts side by side. Absolute utility levels are not comparable across cohorts because the min–max bounds of Eq. 4 are computed within each pool, but the ladder's shape is consistent everywhere: conditioning on coarse task identity recovers a small fraction of what instance-level choice would.

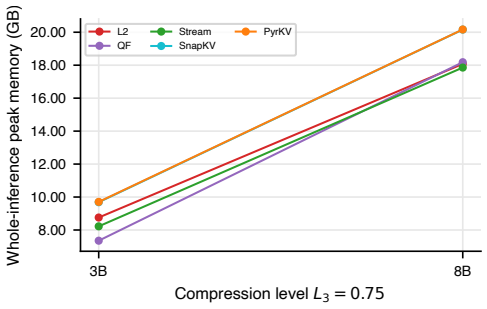


(a) Task score.

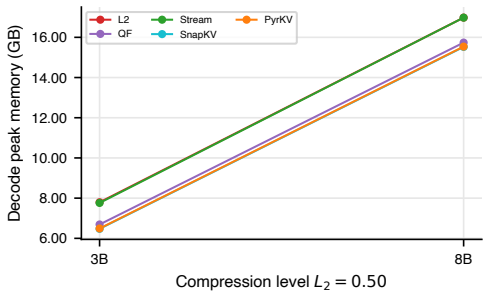
(b) Task score.



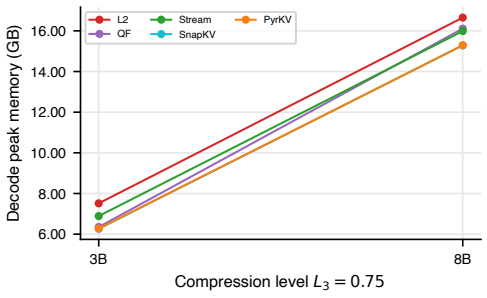
(c) Peak memory.



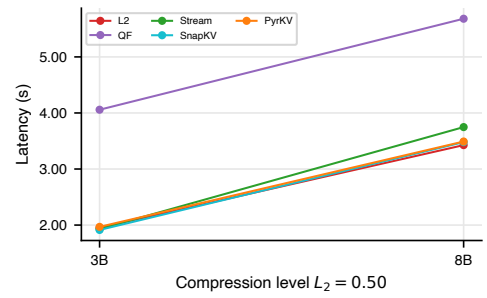
(d) Peak memory.



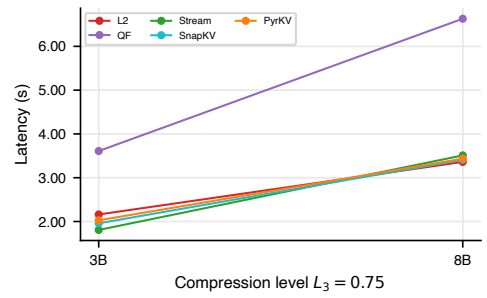
(e) Decode memory.



(f) Decode memory.

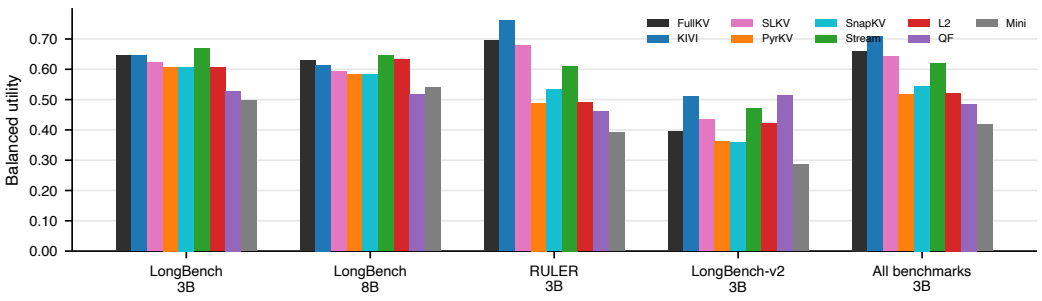


(g) Latency.

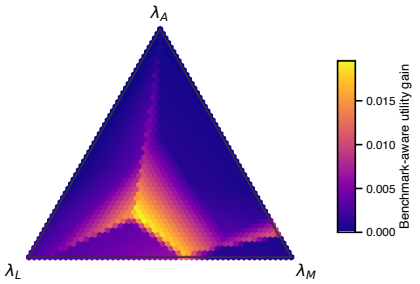


(h) Latency.

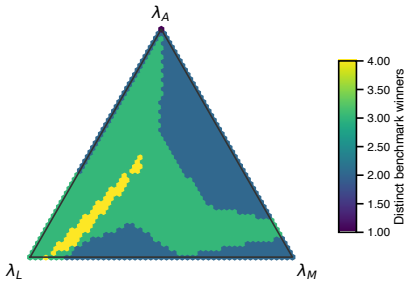
Figure 35: **Both scales agree on the top global configuration and on the overall trade-off structure.** Direct 3B versus 8B comparison on LongBench at both compression levels for which an 8B cohort exists.



(a) Balanced utility across cohorts.



(b) Utility-gain heatmap.



(c) Frontier members per weighting.

Figure 36: **No configuration holds top utility across cohorts, and the recoverable gain varies continuously with deployment priority.** Per-configuration balanced utility in each cohort (a), the utility gained by moving from global static to instance-level selection at each deployment weighting (b), and the number of distinct configurations that win across benchmarks at each weighting (c).

D Routing Framework, Estimators, and Selector Specifications

D.1 Deployable Utility and Action Catalog

Table 23: **KIVI-4bit is the highest-utility action on all four cohorts, and its margin over FullKV exceeds the entire routing headroom on two of them.** Measured nine-action catalog at $\ell = L_3$. U: mean evaluation utility (Eq. 5), which differs from the diagnostic U_λ of Tab. 1; "best for": share of requests on which the action is the retrospective oracle choice.

Cohort	Action	A	M_{peak} (GiB)	L (s)	U	Best for
LongBench 3B	SimLayerKV	41.69	9.03	2.64	0.770	3.4%
LongBench 3B	FullKV	42.52	9.27	1.80	0.774	6.8%
LongBench 3B	KIVI-4bit	42.38	8.26	3.12	0.778	32.2%
LongBench 3B	L2Compress	30.33	8.76	2.16	0.737	6.0%
LongBench 3B	MiniCache	26.22	9.30	3.82	0.712	3.2%
LongBench 3B	PyramidKV	40.45	9.71	2.03	0.763	3.5%
LongBench 3B	Q-Filters	11.24	7.35	3.61	0.680	20.8%
LongBench 3B	SnapKV	39.99	9.69	1.96	0.762	4.3%
LongBench 3B	StreamingLLM	38.01	8.22	1.81	0.768	19.6%
LongBench 8B	SimLayerKV	37.30	18.64	5.21	0.744	3.4%
LongBench 8B	FullKV	38.27	19.00	3.39	0.750	8.3%
LongBench 8B	KIVI-4bit	38.65	17.65	5.76	0.756	37.4%
LongBench 8B	L2Compress	31.66	18.08	3.36	0.736	11.3%
LongBench 8B	MiniCache	31.41	19.01	5.92	0.719	3.6%
LongBench 8B	PyramidKV	36.75	20.17	3.43	0.735	2.9%
LongBench 8B	Q-Filters	22.38	18.18	6.63	0.694	6.6%
LongBench 8B	SnapKV	36.41	20.16	3.41	0.734	3.8%
LongBench 8B	StreamingLLM	35.01	17.86	3.51	0.749	22.7%
RULER 3B	SimLayerKV	86.71	6.92	1.19	0.681	5.4%
RULER 3B	FullKV	93.48	7.04	0.90	0.698	16.2%
RULER 3B	KIVI-4bit	93.23	6.55	1.64	0.755	56.9%
RULER 3B	L2Compress	33.79	6.92	1.33	0.493	1.1%
RULER 3B	MiniCache	30.41	7.05	1.66	0.418	0.5%
RULER 3B	PyramidKV	74.32	7.40	1.12	0.525	0.2%
RULER 3B	Q-Filters	5.10	6.47	1.98	0.453	5.5%
RULER 3B	SnapKV	83.12	7.36	1.07	0.566	0.4%
RULER 3B	StreamingLLM	41.19	6.75	0.95	0.593	13.8%
LongBench-v2 3B	SimLayerKV	15.11	24.25	9.15	0.413	4.0%
LongBench-v2 3B	FullKV	15.51	31.77	8.34	0.369	0.6%
LongBench-v2 3B	KIVI-4bit	15.11	17.83	8.63	0.474	12.9%
LongBench-v2 3B	L2Compress	11.73	27.43	8.30	0.392	5.4%
LongBench-v2 3B	MiniCache	0.20	31.99	10.17	0.281	0.2%
LongBench-v2 3B	PyramidKV	14.91	33.83	8.63	0.345	1.2%
LongBench-v2 3B	Q-Filters	5.17	16.46	7.86	0.467	71.8%
LongBench-v2 3B	SnapKV	14.12	33.82	8.64	0.343	1.2%
LongBench-v2 3B	StreamingLLM	15.11	23.32	8.32	0.436	2.8%

Figure 37 gives the retrospective oracle's action distribution per cohort, and Table 23 the mean outcomes behind it. An action is one compression method together with one complete setting of its parameters; the same method at a different retention budget occupies a different point in the accuracy–memory–latency space and is therefore a different action. KIVI-4bit is the highest mean-utility action — the default action a_{ref} — on all four cohorts.

Degenerate requests. On 14.2% of LongBench requests no action scores above zero. On these requests the resource terms determine the utility maximiser: KIVI-4BIT wins 74.4% of the time, while the requests collectively contribute only 1.6% of H. Restricting to the 2,600 requests where actions differ in task score raises H from 3.17 to 3.83 points (20.9%) and leaves the qualitative conclusion intact: the cross-fitted task rung recovers 10.4% of headroom and the top decile holds 59.4%.

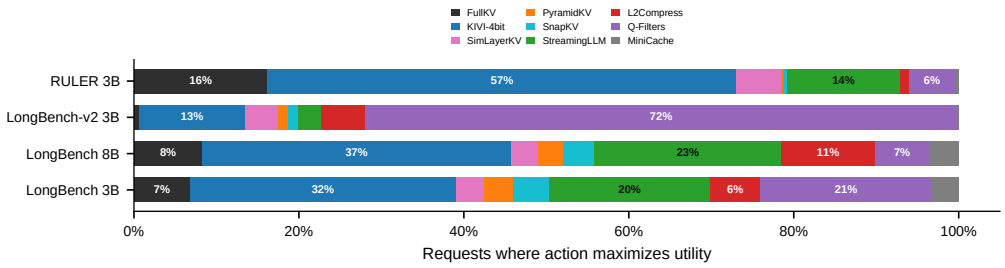


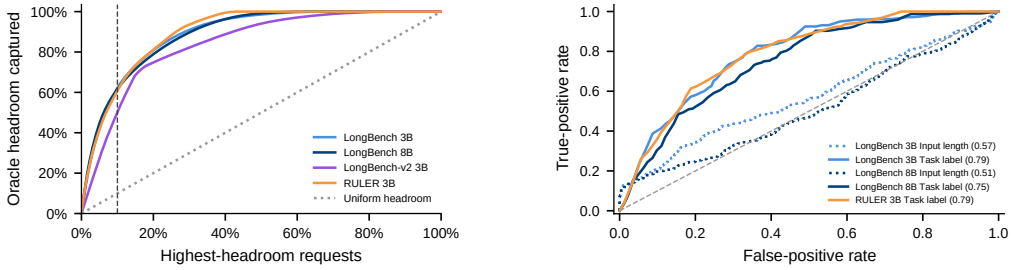
Figure 37: **No action dominates and the uncompressed cache is rarely optimal, which is what makes per-request routing worth considering.** Share of requests for which each action maximises utility, per cohort. FullKV wins 0.6–16.2% of requests and KIVI-4bit 12.9–56.9%; on LongBench-v2 Q-Filters wins 71.8%, mostly on degenerate requests.

Frozen bounds and weighting. The normalisation bounds of Eq. 5 are the extreme values observed across all measured actions on the fitting population. They are computed once, frozen before any evaluation, and recorded with an identifier that accompanies every reported utility; reporting a utility without its bound identifier is an error the evaluation code rejects.

The three components do not occupy their ranges equally. The bounds span the full measured population while the action pool occupies a narrow band within it, particularly on memory, so a proportionally large memory saving corresponds to a small absolute utility change. This is why Section 6.2 reports severe-loss rates and memory separately in addition to utility, and why utility deltas should not be compared across studies that normalise differently. It is also why the balanced winner differs from the task-normalised utility of Eq. 4 (StreamingLLM) and the evaluation utility (KIVI-4bit): the frozen bounds span the whole population, so the latency cost of quantization weighs less.

We repeat the headroom and attainability analysis under a quality-priority and a latency-priority weighting. The ordering of actions by mean utility changes, but the qualitative conclusions of Section 4.3 — early saturation, weak cross-scale transfer, concentration of the opportunity — hold under all three. A selector calibrated under one weighting must be recalibrated before being reported under another, and we never report the frozen artefact under a weighting it was not calibrated to.

D.2 Reachability Estimators and Cross-Scale Transfer



(a) Headroom concentration.

(b) Top-decile detectability.

Figure 38: **Prompts reveal where compression matters, not which action to use. (a)** Cumulative share of H held by the highest-headroom requests. **(b)** Detecting top-decile membership from the task label against input length. The conditioning ladder itself is in Tab. 24.

What this section measures, in plain terms. A hindsight policy that picks the best action for every request sets a ceiling, but part of that ceiling exists only because hindsight sees the outcome.

We want the part a selector could reach from the request alone. The construction is a thought experiment made measurable: suppose a selector could see nothing about a request except one coarse fact — its length band, or which task it belongs to. It would then have to serve the same action to every request sharing that fact, and the best it could do is the action with the highest average utility inside each group. Computing that value for progressively finer facts traces a curve from "know nothing" (0% of the headroom) to "know the request itself" (100%). Where the curve stops rising is where finer knowledge of the prompt stops buying anything, and that is the ceiling we report. Two safeguards keep it honest: each group's action is chosen on four folds and scored on a fifth, so a description with many tiny groups cannot win by memorising them, and the folds respect context groups so near-duplicate requests never straddle the split.

Conditioning on a descriptor. We want to know how much of the routing opportunity a selector could capture if it saw only a limited description of each request. We formalise a description as a function g that assigns each request x to a group $g(x)$ — for instance its input-length decile, or its task. A policy restricted to g cannot distinguish requests within a group, so it must commit to one action per group and apply it to every member. The best such policy picks each group's highest-average action, and its value is

$$V(g) = \mathbb{E}\big[U(x, a_{g(x)})\big], \qquad a_k = \arg\max_{a \in \mathcal{A}} \mathbb{E}\big[U(x, a) \mid g(x) = k\big]. \tag{12}$$

Here U is the per-request utility of Eq. 5, while $V(g)$ is a property of a whole *policy*: the average utility that policy achieves over the population. Two extremes bound the scale. If g puts every request in one group, the policy is the best fixed action. If g gives every request its own group, it is the hindsight policy. We therefore report $V(g)$ rescaled so those extremes are 0% and 100%:

$$\mathrm{Rec}(g) = \frac{V(g) - \mathbb{E}[U(x, a_{\mathrm{ref}})]}{H} \times 100\%. \tag{13}$$

Reporting it this way makes the quantity comparable across cohorts with different absolute headroom, and makes the point at which the curve flattens directly readable as the limit of what that family of descriptions can buy.

Cross-fitting. A finer description always looks better if we let it choose and be graded on the same requests, because each group's action is then fitted to exactly the requests it is scored on. To avoid this we split the requests into $K = 5$ folds along group boundaries (App. A.2), pick each group's action using four folds, and score it on the fifth, rotating through all five and averaging. If a group is missing from the four training folds we fall back to the globally best action. The benefit is that a description which has more groups than data to fill them now scores *worse* than a coarser one, which is the honest signal, and it is exactly what we observe for task $\times$ length decile (Tab. 24). Without cross-fitting, recovery increases with refinement automatically and the curve tells us nothing.

Noise-corrected hindsight. The hindsight policy takes a maximum over nine measured numbers per request. If those measurements carry any noise, the maximum tends to land on whichever action was measured favourably by chance, so the hindsight value is optimistic and H is inflated. Since the whole paper is about whether H is a real target, we have to rule this out rather than assume it. With R repeated executions we can: choose the action using one execution, then score that choice on the average of the others, so the noise that drove the choice cannot also inflate the score,

$$\widehat{V}^{\mathrm{corr}} = \frac{1}{R} \sum_{r=1}^{R} \mathbb{E}\Big[U^{(-r)}\big(x, \arg\max_a U^{(r)}(x, a)\big)\Big]. \tag{14}$$

The difference between the uncorrected and corrected values is the share of H attributable to measurement variability. Appendix B.4 reports it as negligible.

Intervals. All intervals are paired cluster bootstrap with $2{,}000$ resamples. We resample *group keys* rather than individual requests, so that requests sharing a source context move together and the intervals are not narrowed by treating near-duplicates as independent; task and benchmark means are recomputed inside each draw. Draws are shared across policies, so a difference between two selectors is estimated on the same resampled populations and the comparison is paired. On the $3{,}200$-request LongBench cohorts this yields intervals on mean utility of roughly ± 0.3 utility points,

and on the 503-request LongBench-v2 cohort roughly ± 1.2, which is why we treat single-cohort LongBench-v2 differences below about one point as uninformative. Intervals are descriptive and are not adjusted for multiplicity. Where a policy shows zero severe degradations on a finite sample the interval is degenerate and must not be read as evidence of zero population risk: with 503 requests, an observed rate of 0.0% is consistent with a true rate up to roughly 0.6%.

Results. Table 24 gives the ladder, and Table 25 the headroom of every task. Headroom varies by roughly an order of magnitude across tasks, so recovery figures measured over different task mixes are not comparable.

Table 24: **Prompt-side conditioning saturates at task granularity on every cohort, and refining past it loses ground.** Cross-fitted value of each information set as a percentage of that cohort's routing headroom H (Eq. 7), five group-respecting folds under the stored cohort bounds. Rows below the line are reference policies that use outcomes.

Information set	LongBench 3B	LongBench 8B	RULER 3B	LongBench-v2 3B
Best fixed (a_{ref})	0.0%	0.0%	0.0%	0.0%
Length decile	5.8%	6.4%	0.0%	0.0%
Task type	11.2%	7.8%	20.2%	0.0%
Task label	9.0%	15.4%	20.2%	1.7%
Task label × length	2.4%	5.3%	20.2%	0.0%
Top-decile oracle	65.4%	61.6%	64.6%	57.7%
Per-request oracle	100.0%	100.0%	100.0%	100.0%
Cross-model oracle	-12.9%	–	–	–

Routing headroom H: 3.17, 2.53, 2.85 and 5.04 utility points.

Table 25: **Routing headroom varies by about an order of magnitude across tasks.** Mean hindsight headroom over each cohort's best fixed action (Eq. 6) per task, in utility points, sorted within each cohort.

LongBench 3B		LongBench 8B		RULER 3B		LongBench-v2 3B	
Task	H	Task	H	Task	H	Task	H
narrativeqa	6.75	narrativeqa	5.24	niah_multikey_3	7.51	Many-shot learning	9.69
lcc	6.09	lcc	4.16	niah_multikey_2	6.46	Agent history QA	7.68
2wikimqa	5.59	repobench-p	3.78	cwe	5.04	New language translation	6.37
musique	5.41	passage_count	3.50	qa_1	3.50	Knowledge graph reasoning	6.00
hotpotqa	5.12	qmsum	2.77	qa_2	3.20	User guide QA	5.87
repobench-p	3.65	multi_news	2.48	niah_multikey_1	3.08	Financial	5.64
qasper	3.51	passage_retrieval_en	2.47	niah_multivalue	2.87	Academic	5.63
multifieldqa_en	2.82	triviaqa	2.47	niah_multiquery	2.00	Governmental	5.47
multi_news	2.40	qasper	2.44	niah_single_2	1.81	Legal	5.19
triviaqa	2.37	multifieldqa_en	2.32	fwe	1.07	Multi-news	4.79
samsum	1.94	gov_report	2.29	niah_single_3	0.38	Event ordering	4.66
gov_report	1.63	samsum	2.19	vt	0.12	Literary	4.65
passage_count	1.58	hotpotqa	1.52	niah_single_1	0.00	Code repo QA	2.91
qmsum	1.43	trec	1.41			Detective	2.88
trec	1.08	musique	1.19			Table QA	1.48
passage_retrieval_en	0.59	2wikimqa	0.64			Dialogue history QA	0.78

Cross-scale transfer. The LongBench requests were executed under both model scales using the same action pool, which permits a direct test of whether per-request compressor preference is a property of the request.

Agreement. We compare the per-request best-action assignments at the two scales. Agreement exceeds the rate expected by chance given the two marginal distributions, but only modestly, and expressed as a chance-corrected coefficient the value is low. Restricting the comparison to within-task pairs — which removes the component attributable to task identity — reduces it further, indicating that most of the agreement that does exist is task-level rather than request-level.

Transfer as a policy. Using the 8B best-action assignment to route the 3B model across all requests reaches 0.773575 mean utility against 0.777678 for the best fixed action and 0.809373 for the per-request oracle, i.e. -12.9% of H (-0.41 points), worse than the fixed default, because on the large majority of requests, where several actions are near-equivalent, the transferred preference contributes noise. Applied only to the high-opportunity minority it recovers a positive share, and the 8B per-request opportunity is itself weakly predictive of which 3B requests carry high opportunity.

Interpretation. Both halves matter. The within-task component of per-request preference is largely specific to the model executing the request, which bounds what a prompt-only representation can transfer across scales and explains why a selector fitted at one scale should not be assumed to survive a change of scale. But the restricted result establishes a positive lower bound on prompt-conditional attainability obtained without any prompt representation at all, which is direct evidence that structure beyond the task level exists and is in principle learnable.

Selectors across scales. Table 26 compares each selector's decisions on the identical requests at both scales. All learned selectors except KVRoute make the same decision at both scales on every request, because their features do not depend on the model; KVRoute's model index changes 2.5% of its decisions. Every learned selector gains less at 8B than at 3B except KVRoute, whose gain is small at both scales.

Table 26: **Paired LongBench selector behavior under direct 3B-to-8B transfer.** Action agreement and utility changes use identical request identifiers at both model scales.

Policy	Paired requests	Action agreement	ΔU_{net} 3B	ΔU_{net} 8B
KVRoute	3200	97.5%	+0.03	+0.10
KVRoute-Encoder	3200	100.0%	+1.25	+0.07
KVRoute-RankOnly	3200	100.0%	+1.30	+0.19
KVRoute-RankOnly-AS	3200	100.0%	+2.28	+0.29
KVRoute-RankOnly-Fixed	3200	100.0%	+1.97	+0.57
OutcomeForest	3200	100.0%	+1.05	+0.77
OutcomeForest+Guard	3200	100.0%	+0.90	+0.65
Random	3200	10.5%	-1.97	-1.55

D.3 Complete Selector Models and Hyperparameters

All selectors choose among the nine actions of Appendix D.1, observe only the fields permitted by Appendix A.2, and emit one action per request. A decision selects an action identifier and its frozen native settings; no selector synthesises compressor settings at inference time.

Shared protocol. Every learned selector is trained on the same $3{,}200$ LongBench training requests with all nine measured actions; RULER, LongBench-v2 and all LongBench 8B outcomes are excluded from fitting, model selection and calibration. Evaluation proceeds in three stages, which we keep distinct:

1. *Internal grouped out-of-fold prediction* on the training requests, used only to choose hyperparameters, guards, thresholds and fallback policies. Each request is predicted by a fold model that saw neither it nor its context group. The out-of-fold estimates quoted below are of this kind.
2. *Final-model replay.* The selected configuration is refit on all $3{,}200$ requests, frozen, and replayed on those same requests. These LongBench 3B rows (Tab. 33) are in-sample and are not held-out results.
3. *Frozen transfer.* The same frozen selector is applied without refitting to RULER-4096 ($6{,}500$ requests) and LongBench-v2 (503), whose prompts it never saw, and to LongBench 8B, which reuses the training prompts with outcomes of a model it never saw.

Connected groups are formed from context hashes together with any available document, template, source-family or supplied group identifiers, and three-fold splitting assigns whole groups while balancing benchmark and task strata. Latency and memory bounds are fitted only on the relevant

training fold. Training sample weights give equal mass to each benchmark and to each task within a benchmark, which is distinct from the equal weights inside the utility.

Hardware. All selector development ran on single-node Databricks compute with one NVIDIA Tesla T4 (Azure `Standard_NC4as_T4_v3`, Databricks Runtime 17.3). The node provides 28 GiB of host memory; the T4 itself has 16,384 MiB of GPU memory, so the correct description is one Tesla T4 on a 28 GiB host-memory node. The MiniLM encoder runs on the T4; the tree heads and ONNX operators run on CPU with one fitting thread. Replay records confirm one Tesla T4, CUDA 12.6, Python 3.12.3 and Runtime 17.3 for all five evaluated selectors. Compressor outcomes were measured separately on H100 hardware.

D.3.1 KVROUTE

KVRoute is a two-stage, execution-aware selector. Its reference action is the default action a_{ref} (KIVI-4bit), recorded in the freeze as `reference_action="KIVI-4BIT"` and `fullkv_fallback=false`. Every ordinary rejection, detector-negative decision, deadline violation, tokenizer exception and malformed prediction returns the default action. FullKV remains one of the nine rankable actions and can be selected deliberately by the default-risk policy, but it is never an implicit fallback.

Stage 1. A CPU `HistGradientBoostingClassifier` with 100 boosting iterations, at most 15 leaves, learning rate 0.08, L2 regularisation 1.0 and seed 42. Its positive target is the union of requests in the top decile of hindsight utility headroom above the default action and requests where the default loses more than five task-score points against FullKV. The second term is necessary because the unconditional default carries a 4.352% source severe-loss rate, already above the final 1% routed-policy constraint. The frozen positive-label rate is 13.188% and the frozen detector threshold is 0.040208443.

Stage 2. One multi-output `ExtraTreesRegressor` with 64 trees, `min_samples_leaf`$=$ 20, `max_features`$=0.5$, one CPU thread and seed 42, predicting in a single call nine default-relative utility advantages and nine FullKV-relative quality drops. The default's utility output and FullKV's quality-drop output are reset to exact zero constants after prediction. For detector-positive requests the default-risk threshold is applied first; if the predicted FullKV-relative quality loss of the default is unsafe, FullKV is selected explicitly. Otherwise the maximum predicted default-relative utility is proposed and accepted only if the utility and quality thresholds pass. A 50 ms end-to-end deadline is checked after extraction, detection and ranking, failing closed to the default.

Training and calibration. Training uses the 3,200 LongBench source requests, 2,881 groups, all nine actions and three-fold grouped cross-fitting. RULER, LongBench-v2 and LongBench 8B outcomes are excluded from feature fitting, detector fitting, ranker fitting and calibration. Calibration searches detector-score quantiles, utility thresholds $\{0, 0.0025, 0.005, 0.01\}$, FullKV-relative quality-loss thresholds $\{0, 0.5, 1, 2\}$ and default-risk thresholds derived from out-of-fold prediction quantiles. Feasibility requires mean task-score change against FullKV of at least -0.5 points and severe-loss rate below 1%. The frozen policy uses detector threshold 0.040208443, utility threshold 0.0, quality-drop threshold 0.0 and default-risk threshold -1.525385257. Its grouped source out-of-fold estimate is mean utility 0.764998, mean quality change -0.0245 points, severe-loss rate 0.656%, detector fire rate 75% and explicit FullKV safety-action rate 75%. Final model fitting took 0.66 s.

Observed behaviour and its failure mode. The high source-side FullKV safety-action rate is the origin of the RULER result in Section 6.2. Because the default-risk head is fitted on LongBench alone, it generalises poorly: on LongBench-v2 it fires rarely and KVRoute serves the default on 91% of requests, whereas on RULER it fires almost always and 96% of requests are routed to FullKV despite the default costing only 0.25 task-score points there. We report this rather than retune, because retuning on a held-out cohort would invalidate it as held out.

D.3.2 KVROUTE-ENCODER

KVROUTE-ENCODER tests whether the refusal action, rather than ranking capacity, explains poor cross-benchmark recovery. It retains all nine actions and FullKV as the absolute quality-safety reference, but replaces FullKV as the ordinary fallback with a robust non-FullKV action selected from source-training outcomes only; RULER, LongBench-v2 and LongBench 8B are excluded from fallback selection, feature selection, forest fitting and guard calibration. The fallback is selected independently inside every grouped source fold and once on the complete source population, maximising worst-cell source utility among non-FullKV actions; **KIVI-4bit was selected in all three folds and in the complete-source fit**.

The shared 32-tree `ExtraTreesRegressor` predicts utility differences relative to the learned fallback and quality differences relative to FullKV, with squared error, `min_samples_leaf`$= 20$, `max_features`$= 0.25$, one CPU thread and seed 42. Two predeclared representations were compared under identical groups and settings: the $1{,}936$-dimensional six-view MiniLM encoding, and a $1{,}964$-dimensional *combined* representation adding 28 bounded structural features. The combined representation was selected; the semantic-only candidate had no feasible guard under the frozen search. The frozen guard is `min_gain` $= 0.0$ and `max_quality_loss` $= 0.0$. Rejection returns KIVI-4bit when that fallback is itself predicted quality-safe, and FullKV otherwise. Grouped source out-of-fold result: net utility 0.760623, gain over the fallback 0.005547, mean quality change $+0.0287$ points, severe-loss rate 0.886%.

This selector is the cleanest control on the fallback question. Changing the fallback alone, while keeping the semantic encoder and the rank-then-guard architecture, gives $+1.25$ points on LongBench 3B against $+1.30$ for KVROUTE-RANKONLY; it is worse on LongBench 8B and RULER and better only on LongBench-v2 ($+0.21$ vs. -1.60), far below KVRoute's $+8.59$. The fallback matters, but not in isolation from the detector-first architecture.

Table 27: **What each policy sees and predicts.** All selectors choose among the nine actions of App. D.1, observe only the fields permitted by App. A.2, and emit one action per request; none synthesises compressor settings at inference time. Guards, fallbacks and fitting status are in Tab. 28.

Selector	Representation	Predictor and targets
KVRoute	46 tokenizer features, no neural encoder	Stage 1: `HistGradientBoosting` classifier over the routing-worthy target. Stage 2: one multi-output `ExtraTrees` predicting $9 \times \Delta U$ against a_{ref} and $9 \times \Delta A$ against FullKV
KVROUTE-ENCODER	1,964-d: six-view MiniLM (1,936-d) plus 28 bounded structural features	Multi-output `ExtraTrees`: ΔU against the learned fallback, ΔA against FullKV
KVROUTE-RANKONLY	1,936-d six-view MiniLM	One shared multi-output `ExtraTrees`: ΔU and ΔA, both against FullKV
-RANKONLY-AS	as RANKONLY	One `ExtraTrees` per non-constant action/target coordinate: action-specific partitions, no statistical sharing
-RANKONLY-FIXED	as RANKONLY	Multi-output `ExtraTrees`: ΔU and ΔA for nine actions, exact token counts under the target tokenizer
OUTCOMEFOREST	128 PCA components of the semantic features plus three standardised runtime features	Multi-output `ExtraTrees` predicting *absolute* quality, peak memory and latency for nine actions (27 outputs)
OUTCOMEFOREST+GUARD	as OUTCOMEFOREST	the same frozen predictor, unchanged
RANDOM	none	none

Table 28: **What each policy does with its predictions, and what was actually fitted.** Not every selector is an independently trained model: the last column distinguishes fitted predictors from reused ones and searched guards from frozen ones. Every fit uses the 3,200 LongBench 3B source requests and seed 42; no other cohort enters any fitting, calibration or selection step.

Selector	Guard	Fallback	Fitting status
KVRoute	Default-risk check τ_{risk} first, then τ_U and τ_A; all searched on grouped out-of-fold predictions	a_{ref} (KIVI-4bit); FullKV only when the risk check selects it	Both stages fitted here; thresholds searched under the feasibility constraints
KVROUTE-ENCODER	Fixed at gain ≥ 0 and quality loss ≤ 0	Learned non-FullKV action per fold (KIVI-4bit in all three and on the full source); FullKV if that action is predicted unsafe	Predictor fitted here; the fallback is selected inside every grouped fold, which is what isolates the fallback question
KVROUTE-RANKONLY	(τ_U, τ_A) searched under cross-fitting	FullKV	Predictor fitted here; guard from the same searched family as -AS
-RANKONLY-AS	identical search	FullKV	Separately fitted predictor, same guard family
-RANKONLY-FIXED	Fixed at gain ≥ 0, quality loss ≤ 2 points; never searched	FullKV	Separately fitted predictor under an unsearched guard, which is what distinguishes it from KVRoute
OUTCOMEFOREST	None; serves the maximum predicted balanced utility	FullKV	Predictor not retrained: a frozen earlier model re-exported to ONNX, accepted only at action agreement ≥ 0.999 and prediction error $\leq 10^{-3}$
OUTCOMEFOREST+GUARD	Margin ≥ 0 over FullKV and quality loss ≤ 2.0 points, calibrated on grouped out-of-fold predictions	FullKV	Nothing refitted; only the acceptance policy is calibrated, so the pair isolates guarding rather than learning
RANDOM	none	–	Nothing fitted, calibrated or thresholded

Table 29: **Frozen hyper-parameters and thresholds.** All trees use squared error, one CPU fitting thread and seed 42. Folds respect the grouping rule of App. A.2.

Selector	Tree settings	Frozen thresholds	Folds / groups
KVRoute	Stage 1: 100 boosting iterations, ≤ 15 leaves, learning rate 0.08, L2 1.0. Stage 2: 64 trees, `min_samples_leaf` 20, `max_features` 0.5	detector 0.040208; $\tau_U = 0$; $\tau_A = 0$; $\tau_{\text{risk}} = -1.525385$; positive-label rate 13.19%	3 / 2,881
KVROUTE-ENCODER	32 trees, leaf 20, `max_features` 0.25	gain 0.0, quality loss 0.0	3 / 2,881
KVROUTE-RANKONLY	32 trees, leaf 20, `max_features` 0.25, no depth limit, gap weighting off	$(\tau_U, \tau_A) = (0.0025,\ 2.0)$	3, nested
-RANKONLY-AS	as above, one forest per output coordinate	$(\tau_U, \tau_A) = (0.0025,\ 2.0)$	3, nested
-RANKONLY-FIXED	64 trees, maximum depth 20, leaf 10, `max_features` $= \sqrt{\cdot}$	gain ≥ 0, quality loss ≤ 2 (no search)	3, nested
OUTCOMEFOREST	32 trees, leaf 20, `max_features` 0.25, no depth limit, 27 outputs	none	inherited
OUTCOMEFOREST+GUARD	predictor unchanged	searched over 3 weightings, 5 quality-loss limits and 4 margin thresholds	inherited

D.3.3 INTERPRETATION LIMITS

Router timing is isolated from target-model serving and is not co-resident end-to-end latency. Allocated GPU peak memory is not a direct KV-cache byte measurement. OUTCOMEFOREST and OUTCOMEFOREST+GUARD inherit an earlier training population and frozen objective. Every selector is fitted on LongBench 3B alone; no cohort other than LongBench 3B enters any fitting, calibration or selection step. LongBench 8B is prompt-only transfer of selectors frozen on 3B-side information: no selector observes 8B outcomes during routing or calibration.

E Extended Routing Results and Shortfall Analysis

E.1 Decision Overhead and Component Latency

Table 30: **KVRoute is the cheapest selector on every cohort because it makes no encoder pass and skips the ranker on detector-negative requests.** Pooled warm router latency quantiles; raw timing rounds are pooled rather than averaging request-level quantiles. LongBench-v2 shows where the sub-50 ms budget is not met.

Cohort	Policy	p50 (ms)	p95 (ms)	p99 (ms)	observations
LongBench 3B	KVRoute	29.8	67.9	107.5	32,000
LongBench 3B	KVRoute-Encoder	72.1	163.5	238.8	32,000
LongBench 3B	KVRoute-RankOnly	68.5	162.0	236.9	32,000
LongBench 3B	KVRoute-RankOnly-AS	95.7	204.6	308.0	32,000
LongBench 3B	KVRoute-RankOnly-Fixed	76.5	183.6	290.3	32,000
LongBench 3B	OutcomeForest	80.6	161.5	227.3	32,000
LongBench 3B	OutcomeForest+Guard	79.8	161.5	224.1	32,000
LongBench 8B	KVRoute	29.5	67.7	106.9	32,000
LongBench 8B	KVRoute-Encoder	72.4	164.7	237.7	32,000
LongBench 8B	KVRoute-RankOnly	69.8	165.5	240.5	32,000
LongBench 8B	KVRoute-RankOnly-AS	95.4	204.3	308.6	32,000
LongBench 8B	KVRoute-RankOnly-Fixed	75.5	182.2	292.2	32,000
LongBench 8B	OutcomeForest	86.7	177.2	246.4	32,000
LongBench 8B	OutcomeForest+Guard	80.0	160.8	221.9	32,000
RULER 3B	KVRoute	13.8	15.8	16.7	65,000
RULER 3B	KVRoute-Encoder	38.6	42.5	44.2	65,000
RULER 3B	KVRoute-RankOnly	33.9	37.5	39.2	65,000
RULER 3B	KVRoute-RankOnly-AS	55.6	60.4	62.9	65,000
RULER 3B	KVRoute-RankOnly-Fixed	37.6	42.4	44.9	65,000
RULER 3B	OutcomeForest	48.3	84.0	100.1	65,000
RULER 3B	OutcomeForest+Guard	50.7	86.8	103.2	65,000
LongBench-v2 3B	KVRoute	276.8	2667.3	8684.8	5,030
LongBench-v2 3B	KVRoute-Encoder	608.7	5540.1	17548.9	5,030
LongBench-v2 3B	KVRoute-RankOnly	615.9	5824.8	19263.7	5,030
LongBench-v2 3B	KVRoute-RankOnly-AS	617.0	5579.9	16639.1	5,030
LongBench-v2 3B	KVRoute-RankOnly-Fixed	706.8	5680.3	17258.6	5,030
LongBench-v2 3B	OutcomeForest	408.2	3078.2	9239.5	5,030
LongBench-v2 3B	OutcomeForest+Guard	413.2	3054.9	9253.8	5,030

Selector latency is measured on raw requests with five warm-up rounds followed by ten timed rounds, in isolation from target-model serving; timing rounds are never counted as repeated quality outcomes, and selector cost is charged to request latency in every net-utility figure in the main text. Table 30 pools raw timing rounds rather than averaging request-level quantiles. KVRoute is the cheapest learned selector on every cohort, and KVRoute-RankOnly-AS the most expensive, which is the expected cost of replacing one shared forest call with one call per output. The LongBench-v2 rows show the regime where routing stops being free: for the encoder-based selectors the median rises to 408–707 ms and p_{99} to 9–19 s, because encoder cost scales with input length while the head does not. KVRoute's tokenizer features also grow with length, to a median of 277 ms and a p_{99} of 8.7 s. Table 31 splits the mean latency by component: the encoder pass accounts for almost all of the difference, and KVRoute's feature extraction is the only component that grows with context length.

E.2 Shortfall Attribution and Regret Decomposition

Table 31: **Mean warm router component latency in milliseconds.** Components follow each frozen adapter's native instrumentation, so a blank cell means the adapter does not time that stage separately rather than that the stage is free: OUTCOMEFOREST+GUARD applies its guard inside the head call, and KVRoute reports one fused figure for feature extraction, detection and ranking.

Cohort	Policy	Features	Head	Guard	Format	Other	Total
LongBench 3B	KVRoute	–	–	–	–	33.48	33.48
LongBench 3B	KVRoute-Encoder	79.33	1.28	0.04	0.01	0.25	80.92
LongBench 3B	KVRoute-RankOnly	75.53	1.36	0.03	0.17	0.25	77.34
LongBench 3B	KVRoute-RankOnly-AS	86.14	20.13	0.03	0.22	0.28	106.80
LongBench 3B	KVRoute-RankOnly-Fixed	83.76	0.30	0.29	0.18	2.85	87.38
LongBench 3B	OutcomeForest	85.19	0.36	–	–	3.78	89.33
LongBench 3B	OutcomeForest+Guard	84.91	0.13	–	–	3.70	88.74
LongBench 8B	KVRoute	–	–	–	–	33.16	33.16
LongBench 8B	KVRoute-Encoder	79.45	1.28	0.04	0.01	0.25	81.03
LongBench 8B	KVRoute-RankOnly	77.00	1.36	0.03	0.17	0.26	78.81
LongBench 8B	KVRoute-RankOnly-AS	85.70	20.16	0.03	0.22	0.28	106.40
LongBench 8B	KVRoute-RankOnly-Fixed	83.11	0.31	0.29	0.18	2.86	86.75
LongBench 8B	OutcomeForest	92.14	0.35	–	–	3.77	96.26
LongBench 8B	OutcomeForest+Guard	84.87	0.13	–	–	3.67	88.66
RULER 3B	KVRoute	–	–	–	–	13.78	13.78
RULER 3B	KVRoute-Encoder	36.86	1.29	0.04	0.01	0.25	38.45
RULER 3B	KVRoute-RankOnly	31.88	1.34	0.03	0.17	0.25	33.68
RULER 3B	KVRoute-RankOnly-AS	34.90	19.86	0.03	0.19	0.27	55.24
RULER 3B	KVRoute-RankOnly-Fixed	33.83	0.28	0.28	0.17	2.77	37.33
RULER 3B	OutcomeForest	49.79	0.35	–	–	3.72	53.86
RULER 3B	OutcomeForest+Guard	52.97	0.13	–	–	3.89	56.99
LongBench-v2 3B	KVRoute	–	–	–	–	669.60	669.60
LongBench-v2 3B	KVRoute-Encoder	1409.51	1.30	0.04	0.01	0.25	1411.11
LongBench-v2 3B	KVRoute-RankOnly	1487.59	1.42	0.05	0.17	0.26	1489.49
LongBench-v2 3B	KVRoute-RankOnly-AS	1387.57	19.69	0.03	0.19	0.25	1407.73
LongBench-v2 3B	KVRoute-RankOnly-Fixed	1467.79	0.32	0.28	0.17	2.85	1471.40
LongBench-v2 3B	OutcomeForest	802.34	0.28	–	–	2.88	805.49
LongBench-v2 3B	OutcomeForest+Guard	817.36	0.10	–	–	2.76	820.21

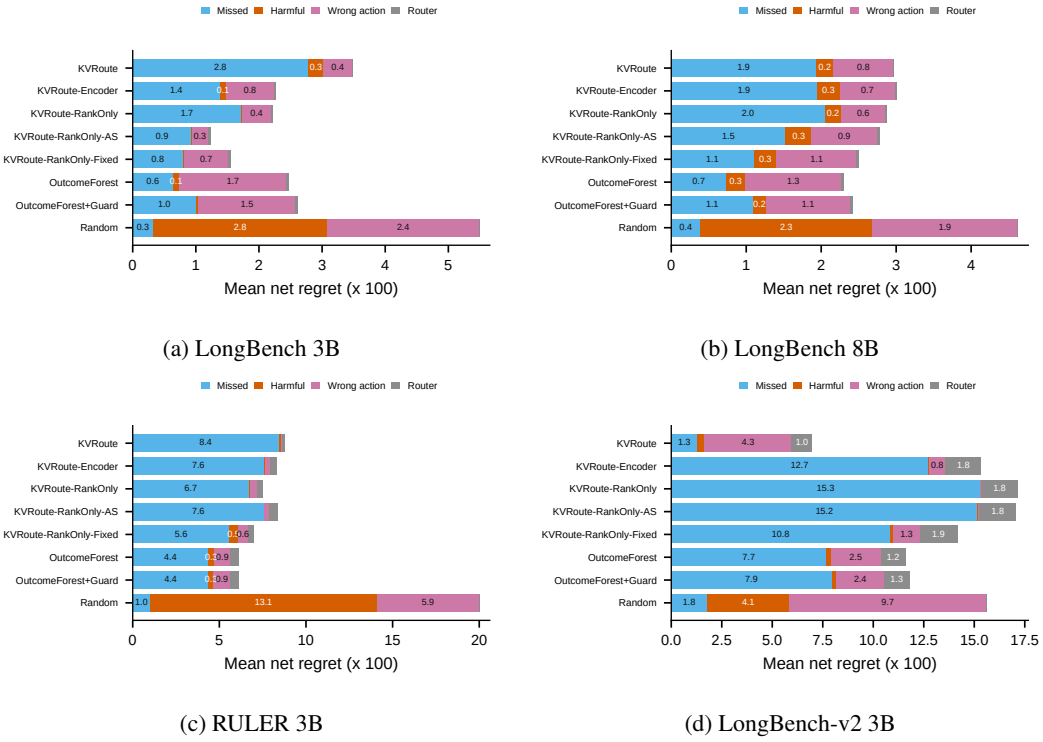
Because the measurement matrix is dense, the shortfall of any selector π against the hindsight policy decomposes exactly. Writing $o(x) = \max_a U(x, a)$, $f(x) = U(x, a_0)$ for the uncompressed utility and $s(x) = U(x, \pi(x))$ for the realised utility,

$$\underbrace{o - s}_{\text{shortfall}} = \underbrace{\mathbb{1}[\pi = a_0](o - f)}_{\text{declined a worthwhile request}} + \underbrace{\mathbb{1}[\pi \neq a_0](f - s)^+}_{\text{compressed and harmed}} + \underbrace{\mathbb{1}[\pi \neq a_0]\big(o - \max(f, s)\big)}_{\text{compressed, wrong action}}, \tag{15}$$

with the selector's runtime cost accounting for the remainder of the net shortfall. Here a_0 is FullKV for every selector, including KVRoute, so its default (KIVI-4bit) decisions count as compressed decisions. Its diagnostic value is that it distinguishes a selector that is too cautious from one that is miscalibrated. The three terms are exhaustive and mutually exclusive: every point of shortfall is assigned to exactly one of declining a worthwhile request, compressing a harmful one, or compressing correctly but choosing a worse action than the best available, with the selector's own runtime accounting for the remainder of the net figure. A guard tuned to avoid harm converts tail risk into declined opportunity, so a cautious selector shows a shortfall dominated by the first term. Table 32 shows this for KVRoute on LongBench 8B and RULER, and on the in-sample LongBench 3B replay. On LongBench-v2 its shortfall is mostly wrong action, because it serves KIVI-4bit where Q-Filters is usually the per-request optimum. OUTCOMEFOREST, which has no guard, loses mostly to wrong actions on both LongBench cohorts; on RULER every selector loses mostly by declining, because each serves FullKV on at least 56% of requests.

Table 32: **Where each selector loses utility.** Mean regret against the hindsight policy in utility points, split into missed (served FullKV although a better action existed), harmful (compressed and lost against FullKV), wrong action, and router overhead (Eq. 15).

Cohort	Policy	Missed	Harmful	Wrong action	Overhead
LongBench 3B	KVRoute	2.78	0.25	0.44	0.01
LongBench 3B	KVRoute-Encoder	1.38	0.10	0.75	0.03
LongBench 3B	KVRoute-RankOnly	1.72	0.02	0.45	0.03
LongBench 3B	KVRoute-RankOnly-AS	0.93	0.00	0.26	0.04
LongBench 3B	KVRoute-RankOnly-Fixed	0.80	0.02	0.69	0.03
LongBench 3B	OutcomeForest	0.64	0.10	1.70	0.03
LongBench 3B	OutcomeForest+Guard	1.00	0.04	1.54	0.03
LongBench 3B	Random	0.32	2.76	2.41	0.00
LongBench 8B	KVRoute	1.93	0.23	0.80	0.01
LongBench 8B	KVRoute-Encoder	1.94	0.31	0.73	0.03
LongBench 8B	KVRoute-RankOnly	2.05	0.22	0.59	0.02
LongBench 8B	KVRoute-RankOnly-AS	1.53	0.35	0.87	0.03
LongBench 8B	KVRoute-RankOnly-Fixed	1.10	0.31	1.07	0.03
LongBench 8B	OutcomeForest	0.73	0.25	1.29	0.03
LongBench 8B	OutcomeForest+Guard	1.09	0.17	1.13	0.03
LongBench 8B	Random	0.39	2.29	1.94	0.00
RULER 3B	KVRoute	8.43	0.16	0.06	0.12
RULER 3B	KVRoute-Encoder	7.57	0.08	0.28	0.35
RULER 3B	KVRoute-RankOnly	6.71	0.05	0.45	0.30
RULER 3B	KVRoute-RankOnly-AS	7.57	0.03	0.30	0.50
RULER 3B	KVRoute-RankOnly-Fixed	5.60	0.47	0.61	0.34
RULER 3B	OutcomeForest	4.36	0.34	0.95	0.49
RULER 3B	OutcomeForest+Guard	4.37	0.31	0.94	0.51
RULER 3B	Random	1.01	13.07	5.92	0.00
LongBench-v2 3B	KVRoute	1.29	0.35	4.33	1.00
LongBench-v2 3B	KVRoute-Encoder	12.71	0.06	0.82	1.76
LongBench-v2 3B	KVRoute-RankOnly	15.29	0.00	0.07	1.81
LongBench-v2 3B	KVRoute-RankOnly-AS	15.16	0.05	0.06	1.80
LongBench-v2 3B	KVRoute-RankOnly-Fixed	10.83	0.18	1.29	1.87
LongBench-v2 3B	OutcomeForest	7.67	0.24	2.45	1.23
LongBench-v2 3B	OutcomeForest+Guard	7.95	0.24	2.35	1.26
LongBench-v2 3B	Random	1.78	4.07	9.75	0.00



(a) LongBench 3B

(b) LongBench 8B

(c) RULER 3B

(d) LongBench-v2 3B

Figure 39: **Where each selector loses utility.** Shortfall against the hindsight policy in utility points, split into missed (served FullKV although a better action existed), harmful, wrong action, and router overhead. The LongBench 3B panel is an in-sample replay on the training requests; the other three panels are frozen transfer.

E.3 Additional Selector Performance and Calibration

Table 33: **Selector outcomes relative to FullKV on all four cohorts.** ΔU_{net}: net evaluation utility change in points (Eq. 5), router latency charged; ΔA: mean task-score change; Memory saved: mean peak-memory reduction; Severe: share of requests losing more than five score points; Compressed: share of requests not served with FullKV; Router: mean decision latency; Speedup: end-to-end latency ratio against FullKV without (target) and with (net) router overhead (< 1 is slower); Router share: router overhead as a share of net latency. LongBench 3B rows replay each final selector on its own 3,200 training requests and are *not* held-out results; RULER, LongBench-v2 and LongBench 8B are frozen transfer (App. D.3). Tab. 2 is the short version for the transfer cohorts.

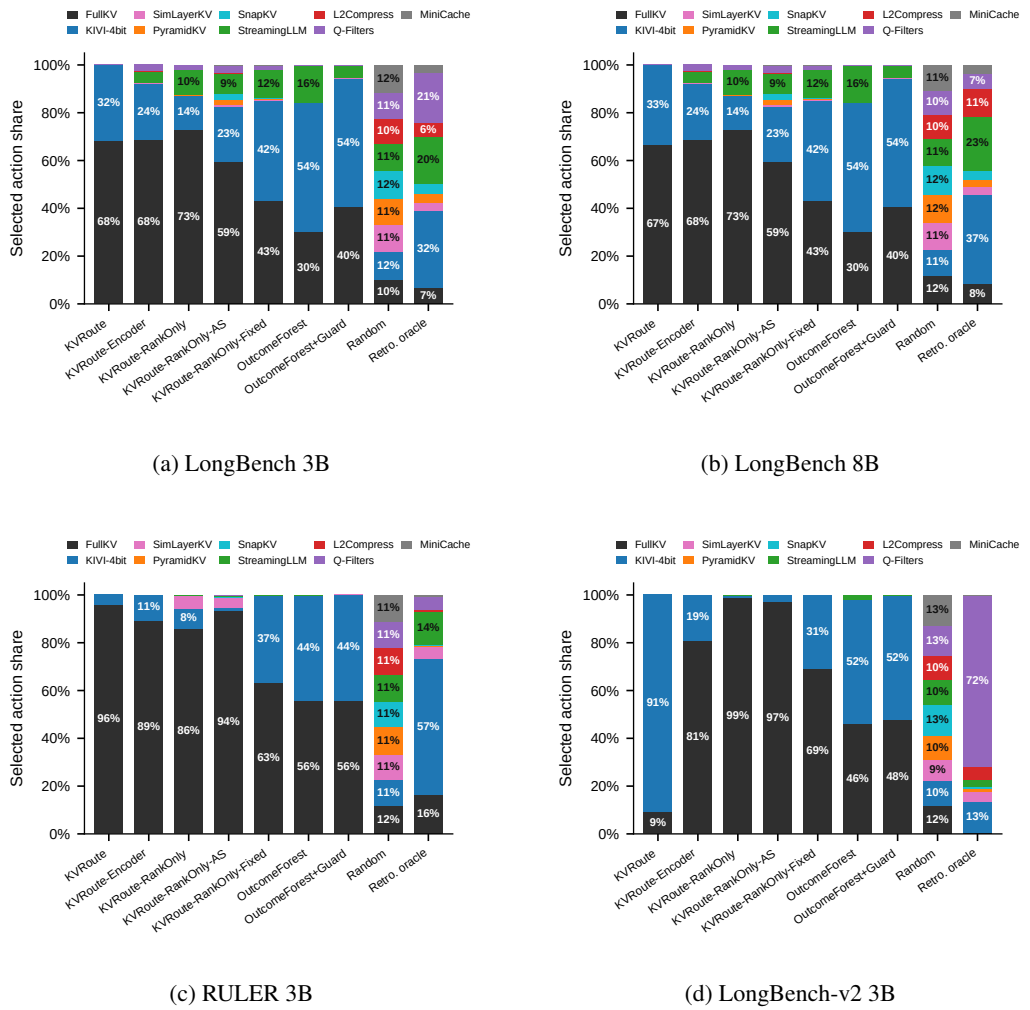
Cohort	Policy	ΔU_{net}	ΔA	Memory saved	Severe	Compressed	Router	Target speedup	Net speedup	Router share
LongBench 3B	KVRoute	+0.03	+0.17	2.3%	0.6%	31.6%	33.5 ms	0.90x	0.86x	4.3%
LongBench 3B	KVRoute-Encoder	+1.25	+3.33	3.7%	0.6%	31.6%	80.9 ms	0.95x	0.85x	9.9%
LongBench 3B	KVRoute-RankOnly	+1.30	+3.14	3.4%	0.7%	27.0%	77.3 ms	1.00x	0.90x	9.5%
LongBench 3B	KVRoute-RankOnly-AS	+2.28	+5.77	4.9%	0.1%	40.7%	106.8 ms	0.99x	0.86x	12.4%
LongBench 3B	KVRoute-RankOnly-Fixed	+1.97	+4.32	7.5%	0.8%	56.6%	87.4 ms	0.91x	0.82x	9.4%
LongBench 3B	OutcomeForest	+1.05	+1.04	9.5%	2.9%	69.9%	89.3 ms	0.84x	0.76x	9.6%
LongBench 3B	OutcomeForest+Guard	+0.90	+1.10	8.2%	0.9%	59.6%	88.7 ms	0.83x	0.75x	9.5%
LongBench 3B	Random	-1.97	-5.84	3.4%	20.3%	90.0%	0.0 ms	0.84x	0.84x	0.0%
LongBench 8B	KVRoute	+0.10	-0.08	2.0%	1.4%	33.2%	33.2 ms	0.89x	0.88x	1.5%
LongBench 8B	KVRoute-Encoder	+0.07	-0.19	2.2%	2.8%	31.6%	81.0 ms	0.91x	0.87x	3.6%
LongBench 8B	KVRoute-RankOnly	+0.19	-0.18	1.9%	3.1%	27.0%	78.8 ms	0.95x	0.91x	3.5%
LongBench 8B	KVRoute-RankOnly-AS	+0.29	-0.07	2.6%	4.3%	40.7%	106.4 ms	0.92x	0.87x	4.5%
LongBench 8B	KVRoute-RankOnly-Fixed	+0.57	+0.13	4.2%	4.4%	56.6%	86.7 ms	0.85x	0.82x	3.4%
LongBench 8B	OutcomeForest	+0.77	+0.21	5.3%	4.9%	69.9%	96.3 ms	0.81x	0.78x	3.7%
LongBench 8B	OutcomeForest+Guard	+0.65	+0.33	4.5%	2.7%	59.6%	88.7 ms	0.81x	0.78x	3.5%
LongBench 8B	Random	-1.55	-4.11	1.0%	19.9%	88.0%	0.0 ms	0.86x	0.86x	0.0%
RULER 3B	KVRoute	-0.22	-0.02	0.3%	0.0%	4.2%	13.8 ms	0.98x	0.95x	3.1%
RULER 3B	KVRoute-Encoder	+0.27	-0.02	0.7%	0.2%	10.6%	38.5 ms	0.96x	0.88x	7.8%
RULER 3B	KVRoute-RankOnly	+1.04	+0.87	0.7%	0.4%	14.1%	33.7 ms	0.97x	0.90x	6.7%
RULER 3B	KVRoute-RankOnly-AS	+0.16	+1.42	0.1%	0.0%	6.4%	55.2 ms	1.00x	0.89x	10.8%
RULER 3B	KVRoute-RankOnly-Fixed	+1.55	-0.11	2.6%	0.3%	36.8%	37.3 ms	0.85x	0.79x	6.8%
RULER 3B	OutcomeForest	+2.41	-0.34	3.0%	1.4%	44.2%	53.9 ms	0.81x	0.73x	8.9%
RULER 3B	OutcomeForest+Guard	+2.42	-0.22	3.0%	1.2%	44.0%	57.0 ms	0.81x	0.73x	9.3%
RULER 3B	Random	-11.45	-28.13	0.9%	34.8%	88.2%	0.0 ms	0.89x	0.89x	0.0%
LongBench-v2 3B	KVRoute	+8.59	-0.80	38.2%	1.6%	90.9%	669.6 ms	0.94x	0.89x	5.9%
LongBench-v2 3B	KVRoute-Encoder	+0.21	+0.40	7.7%	0.2%	18.9%	1411.1 ms	0.98x	0.87x	12.0%
LongBench-v2 3B	KVRoute-RankOnly	-1.60	+0.20	0.6%	0.0%	1.4%	1489.5 ms	1.00x	0.88x	12.3%
LongBench-v2 3B	KVRoute-RankOnly-AS	-1.51	+0.00	1.2%	0.2%	2.8%	1407.7 ms	1.00x	0.87x	12.5%
LongBench-v2 3B	KVRoute-RankOnly-Fixed	+1.39	-0.20	13.3%	0.8%	30.8%	1471.4 ms	0.97x	0.85x	12.6%
LongBench-v2 3B	OutcomeForest	+3.96	-0.20	22.4%	1.0%	54.1%	805.5 ms	0.94x	0.87x	7.8%
LongBench-v2 3B	OutcomeForest+Guard	+3.76	-0.40	22.0%	1.0%	52.5%	820.2 ms	0.94x	0.87x	8.0%
LongBench-v2 3B	Random	-0.04	-4.57	11.4%	7.8%	88.1%	0.0 ms	0.88x	0.88x	0.0%

Which actions selectors choose. Figure 40 reports the distribution of selected actions per selector against the retrospective oracle. It shows what the aggregate utility figures hide: KVRoute serves 68% of LongBench 3B requests and 96% of RULER requests uncompressed, where the oracle uses FullKV on 7% and 16%, so the guard's conservatism is visible as a concentration on a single action rather than as a spread across the catalog.

Utility against risk. Figure 41 plots mean utility gain against severe-degradation rate for every selector and cohort, which is the plane an operator actually chooses on and the one Section 6.2 argues should replace the utility ranking.

Resource savings and where risk concentrates. Figure 42 plots allocated memory saved against end-to-end speedup relative to FullKV. Every selector sits below parity on the speedup axis, because the default action is a quantisation method that trades latency for memory — a trade the scalarised utility of Eq. 5 absorbs but an operator should see explicitly. KVRoute's residual risk is not spread evenly, but it is small everywhere on the repeated cohorts: the highest task-level rate is 5.3% (`qmsum` at 8B) and most tasks are at zero, with RULER showing a single severe request in 6,500. On LongBench-v2 one subdomain reaches 13.3%, but on 15 requests, so that figure is two events rather than a rate (Tab. 37).

Per-task headroom and near-optimality. Table 25 (App. D.2) gives the routing headroom of every task. Table 34 reports the fraction of each selector's decisions falling within a given utility tolerance of the per-request optimum. Read alongside Figure 8, it separates two distinct behaviours: on LongBench the tolerance columns rise steeply, so most of KVRoute's apparent ranking errors are near-optimal choices, whereas on RULER they barely move, confirming that the optimum there is usually unique and a missed choice is a real loss.



(a) LongBench 3B

(b) LongBench 8B

(c) RULER 3B

(d) LongBench-v2 3B

Figure 40: **Guarded selectors concentrate on a narrow set of actions, which aggregate utility figures hide.** Distribution of selected actions per selector against the retrospective oracle. KVRoute serves 68% of LongBench 3B and 96% of RULER requests uncompressed, whereas the oracle uses FullKV on 7% and 16%; on LongBench-v2 it serves KIVI-4bit on 91%.

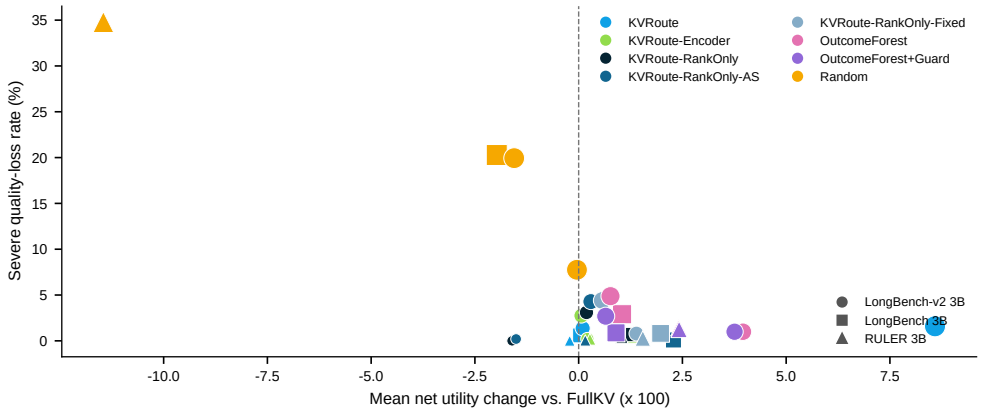
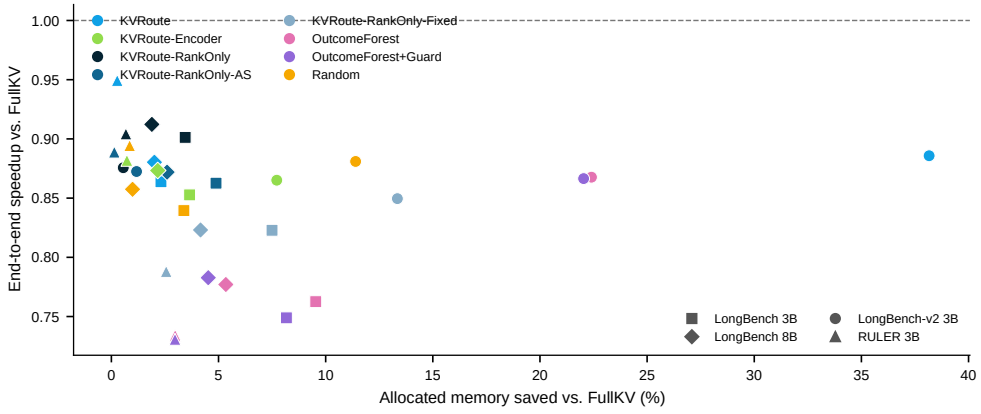


Figure 41: **Guarding buys tail safety at a modest cost in mean utility.** Mean net utility change against severe-degradation rate. RANDOM sits at 20–35% severe, while every learned selector stays below 5%; the lower-right region is where a deployable selector must sit.



(a)

Figure 42: **Memory savings come with an end-to-end slowdown.** Allocated memory saved against end-to-end speedup versus FullKV; every selector sits below parity because quantisation trades latency for memory. KVRoute's severe-loss rate per task is in Tab. 37.

Table 34: **How close selector decisions come to the retrospective oracle.** Mean rank: realized catalog rank of the chosen action (1 = best of nine) under the request-level deployable utility; Top-2: share of decisions among the two best actions; ϵ columns: share of decisions within ϵ utility of the oracle ($\epsilon = 0$ is exact oracle agreement).

Cohort	Policy	Mean rank	Top-2	$\epsilon = 0$	$\epsilon = .0025$	$\epsilon = .005$	$\epsilon = .01$
LongBench 3B	KVRoute	3.99	32.5%	19.7%	25.9%	33.1%	49.9%
LongBench 3B	KVRoute-Encoder	3.63	35.4%	21.4%	26.8%	34.2%	55.7%
LongBench 3B	KVRoute-RankOnly	3.56	37.8%	23.5%	28.7%	36.8%	56.2%
LongBench 3B	KVRoute-RankOnly-AS	2.94	51.6%	37.0%	42.5%	50.3%	69.4%
LongBench 3B	KVRoute-RankOnly-Fixed	2.53	62.2%	42.9%	51.4%	58.1%	69.1%
LongBench 3B	OutcomeForest	2.39	65.9%	44.5%	54.0%	60.0%	68.0%
LongBench 3B	OutcomeForest+Guard	2.57	61.3%	41.0%	50.2%	55.9%	63.9%
LongBench 3B	Random	4.98	23.4%	11.5%	17.0%	23.6%	39.0%
LongBench 8B	KVRoute	3.78	35.3%	22.1%	28.2%	35.3%	48.1%
LongBench 8B	KVRoute-Encoder	3.94	30.9%	17.4%	23.4%	31.2%	47.3%
LongBench 8B	KVRoute-RankOnly	3.79	33.3%	18.7%	25.1%	33.4%	48.5%
LongBench 8B	KVRoute-RankOnly-AS	3.58	38.5%	23.1%	29.9%	38.2%	53.1%
LongBench 8B	KVRoute-RankOnly-Fixed	3.00	50.3%	32.5%	41.4%	49.4%	59.6%
LongBench 8B	OutcomeForest	2.69	57.0%	38.2%	48.1%	55.6%	64.1%
LongBench 8B	OutcomeForest+Guard	2.81	53.2%	35.7%	45.1%	52.4%	60.8%
LongBench 8B	Random	5.06	21.1%	10.8%	16.2%	22.9%	34.2%
RULER 3B	KVRoute	3.28	35.4%	14.1%	14.4%	14.5%	15.1%
RULER 3B	KVRoute-Encoder	3.05	42.8%	20.6%	21.0%	21.3%	22.4%
RULER 3B	KVRoute-RankOnly	2.87	48.5%	24.8%	25.0%	25.2%	25.8%
RULER 3B	KVRoute-RankOnly-AS	3.08	42.2%	18.1%	18.4%	18.6%	19.3%
RULER 3B	KVRoute-RankOnly-Fixed	2.81	46.9%	29.4%	29.9%	30.5%	31.7%
RULER 3B	OutcomeForest	2.22	63.4%	38.7%	39.6%	40.5%	43.2%
RULER 3B	OutcomeForest+Guard	2.21	63.6%	38.8%	39.7%	40.6%	43.3%
RULER 3B	Random	5.01	21.6%	10.8%	11.2%	11.6%	12.8%
LongBench-v2 3B	KVRoute	1.92	83.5%	67.6%	68.0%	68.4%	70.8%
LongBench-v2 3B	KVRoute-Encoder	4.62	18.3%	14.3%	14.3%	14.3%	15.5%
LongBench-v2 3B	KVRoute-RankOnly	5.32	2.6%	1.4%	1.4%	1.4%	2.0%
LongBench-v2 3B	KVRoute-RankOnly-AS	5.27	3.6%	2.6%	2.6%	2.6%	3.4%
LongBench-v2 3B	KVRoute-RankOnly-Fixed	4.19	26.8%	22.5%	22.7%	22.9%	24.1%
LongBench-v2 3B	OutcomeForest	3.29	47.7%	35.8%	36.8%	37.4%	39.4%
LongBench-v2 3B	OutcomeForest+Guard	3.34	46.3%	35.8%	36.8%	37.4%	39.2%
LongBench-v2 3B	Random	5.24	18.9%	9.9%	10.3%	10.7%	13.7%

Quality-informative subset. Repeating the headroom and attainability analysis on the subset of requests where actions differ in task score — excluding the degenerate requests of Appendix D.1 — raises the measured headroom slightly and leaves every conclusion of Section 4.3 unchanged.

The RULER failure in numbers. KVRoute's default-risk check is fitted on LongBench, where the default loses more than five points on 4.4% of requests and the check selects FullKV on about 75% of out-of-fold requests. On RULER the default costs only 0.25 score points on average and is at least as good as FullKV on 71.9% of requests, yet the check selects FullKV on 96% of them. The check therefore over-predicts the default's quality loss under the shift in context length and task structure. Refitting it on several benchmarks is the most direct fix (App. F).

Sensitivity to the scalarisation. A utility conclusion can be an artifact of its normalisation, so we re-scored every policy's frozen decisions against the same nine measured outcomes under ten normalisations and a weight grid of step 0.1 on Δ^2: cohort-wide min–max (the bounds stored in the artifacts and reported in the main text), min–max over the nine catalog actions only, native task score with cohort-wide resource min–max, LongBench 3B bounds frozen and clipped on every cohort, task-wise min–max (Eq. 4), per-request min–max, percentile-clipped bounds (p_5–p_{95}), cohort z-scores mapped through a logistic, and two FullKV-relative forms (linear and log ratio). Nothing is retrained and no action is re-chosen, so the sweep isolates the objective. Across $5{,}360$ policy–normalisation–weight combinations, none beats the best fixed action on all three transfer cohorts; for KVRoute the best margin on any transfer cohort is -0.01 points (RULER, z-score with a quality-only weighting). Table 35 gives the balanced-weight slice. The per-cohort best settings are retrospective sensitivity results: selecting a normalisation or a weighting after seeing the outcome is not a deployment protocol, and we do not report it as one.

Table 35: **KVRoute's gain over the uncompressed cache survives every normalisation; its shortfall against the best fixed compressor does too.** Net utility difference in points at the balanced weighting, frozen decisions re-scored under each normalisation of Eq. 5; nothing retrained. LongBench 3B is omitted as in-sample (App. D.3).

	vs. FullKV			vs. best fixed action		
Normalisation	RULER	LB-v2	LB 8B	RULER	LB-v2	LB 8B
Cohort min–max (reported)	-0.22	+8.59	+0.10	-5.92	-1.93	-0.45
Catalog min–max	-0.21	+8.98	-0.04	-6.19	-1.77	-0.04
Native quality + cohort resources	-0.21	+8.98	-0.04	-6.19	-1.77	-0.04
Frozen source (LongBench 3B) bounds	-0.04	+10.30	-0.07	-0.04	-1.49	-0.07
Task min–max (Eq. 4)	-0.09	+10.46	-0.80	-8.00	-1.94	-2.37
Per-request min–max	-0.57	+16.70	-0.95	-3.33	-4.23	-9.29
Percentile-clipped (p_5–p_{95})	+0.04	+9.54	+0.80	-10.91	-1.94	-2.14
z-score with logistic map	+0.01	+7.28	+0.44	-7.89	-1.27	-1.36
FullKV-relative (linear)	-2.35	+7.77	-5.85	-2.35	-2.74	-5.85
FullKV-relative (log ratio)	-1.88	+12.36	-4.32	-1.88	-3.26	-4.32

What the Stage-1 detector detects. KVRoute's detector separates high-opportunity from ordinary requests at AUC 0.960 on LongBench 3B, using its own predicted probability against the target it was fitted on (top-decile headroom over a_{ref}, or a_{ref} losing more than five score points to FullKV). This covers the 90.7% of requests that reach the detector; scoring the remainder as zero gives a pipeline-level 0.807. Both figures are final-model, in-sample values. Under transfer the signal largely disappears: AUC 0.503 on LongBench 8B and 0.574 on RULER. Detection is therefore not the bottleneck in distribution, and the ceiling measured in Section 4.3 reappears inside the selector.

Risk budgets and severity definitions. Table 36 sweeps the worst-cohort severe-loss budget. Because KVRoute's own worst cohort is 1.6% and the default's is 4.5%, routing is the best admissible policy exactly on the interval between them; outside it either nothing compresses (below) or the default is admissible and better (above). The width of that interval is a property of the severity definition as well: measured at >2 score points the default's worst cohort rises to 9.8% and the interval widens, and at >10 points it falls to 1.8% and the interval narrows. All three definitions are recomputed from the per-request decisions in the released replay matrix.

Table 36: **Routing dominates only between its own tail and the default's.** A policy is admissible at budget B if its severe-loss rate stays below B on *all three* transfer cohorts; admissible policies are ranked by mean ΔU against FullKV over those cohorts (points). Severe loss is defined as >5 score points against FullKV.

	Best admissible fixed action		Best admissible selector	
Budget B	Policy	mean ΔU	Policy	mean ΔU
0.5%	FullKV	0.00	none	–
1%	FullKV	0.00	none	–
2%	FullKV	0.00	**KVRoute** (1.6%)	**+2.82**
5%	KIVI-4bit (4.5%)	+5.59	KVRoute (1.6%)	+2.82
10%	KIVI-4bit (4.5%)	+5.59	KVRoute (1.6%)	+2.82
none	KIVI-4bit	+5.59	KVRoute	+2.82

Worst-cohort severe-loss rate under three severity definitions (>2, >5, >10 score points): FullKV $0.0/0.0/0.0\%$, KIVI-4bit $9.8/4.5/1.8\%$, KVRoute $2.3/1.6/1.6\%$. The admissible window therefore runs from 2.3% to 9.8% at >2 points, from 1.6% to 4.5% at >5, and narrows to 1.6–1.8% at >10. KVRoute's per-cohort deltas behind the mean are -0.22 (RULER), $+8.59$ (LongBench-v2) and $+0.10$ (LongBench 8B).

Further tables. Table 34 gives the realised catalog rank and oracle agreement of each selector's decisions, Table 33 the memory and latency changes, Table 37 KVRoute's severe-loss rate per task, and Table 38 the calibration of chosen-action predictions.

Table 37: **Highest task-level severe quality-loss rates for KVRoute.** Severe loss means a task-score decrease greater than five percentage points relative to FullKV.

Cohort	Task	Severe	ΔU_{net}	requests
LongBench 3B	repobench-p	3.1%	-0.02	225
LongBench 3B	lcc	2.2%	-0.14	225
LongBench 3B	qmsum	1.3%	-0.02	150
LongBench 3B	musique	0.7%	-0.08	150
LongBench 3B	narrativeqa	0.7%	-2.90	150
LongBench 3B	multi_news	0.5%	-0.20	220
LongBench 3B	gov_report	0.4%	-0.19	225
LongBench 3B	triviaqa	0.4%	+0.29	225
LongBench 3B	2wikimqa	0.0%	+0.12	225
LongBench 3B	hotpotqa	0.0%	+0.18	225
LongBench 3B	multifieldqa_en	0.0%	+0.08	112
LongBench 8B	qmsum	5.3%	-0.30	150
LongBench 8B	passage_count	4.4%	+0.30	225
LongBench 8B	passage_retrieval_en	3.6%	+0.60	225
LongBench 8B	repobench-p	2.2%	+0.51	225
LongBench 8B	gov_report	1.8%	-0.20	225
LongBench 8B	lcc	1.3%	+0.24	225
LongBench 8B	narrativeqa	1.3%	-1.51	150
LongBench 8B	multifieldqa_en	0.9%	-0.00	112
LongBench 8B	qasper	0.6%	+0.05	168
LongBench 8B	2wikimqa	0.4%	-0.03	225
LongBench 8B	hotpotqa	0.4%	+0.04	225
LongBench-v2 3B	Knowledge graph reasoning	13.3%	+9.22	15
LongBench-v2 3B	New language translation	5.0%	+9.47	20
LongBench-v2 3B	Literary	3.3%	+7.75	30
LongBench-v2 3B	Legal	3.0%	+6.08	33
LongBench-v2 3B	User guide QA	2.5%	+9.15	40
LongBench-v2 3B	Code repo QA	2.0%	+10.31	50
LongBench-v2 3B	Academic	1.1%	+6.75	94
RULER 3B	niah_multikey_3	0.2%	-1.71	500

Table 38: **Chosen-action prediction calibration.** Mean absolute error and bias of the predicted utility gain and quality change for the chosen action; n_U, n_A: number of predictions. Missing cells indicate selectors whose adapters do not expose that prediction.

Cohort	Policy	Utility MAE	Utility bias	Quality MAE	Quality bias	n_U	n_A
LongBench 3B	KVRoute	0.0115	0.0042	0.2087	-0.2087	2197	2197
LongBench 3B	KVRoute-Encoder	0.0134	-0.0116	3.2699	-2.9750	3200	3200
LongBench 3B	KVRoute-RankOnly	0.0120	-0.0112	3.2372	-3.0260	3200	3200
LongBench 3B	KVRoute-RankOnly-AS	0.0182	-0.0175	5.0859	-4.8864	3200	3200
LongBench 3B	KVRoute-RankOnly-Fixed	0.0178	-0.0166	–	–	3200	0
LongBench 3B	OutcomeForest	–	–	–	–	0	0
LongBench 3B	OutcomeForest+Guard	–	–	–	–	0	0
LongBench 3B	Random	–	–	–	–	0	0
LongBench 8B	KVRoute	0.0109	0.0050	0.0212	0.0212	2144	2144
LongBench 8B	KVRoute-Encoder	0.0069	0.0003	1.5537	0.5397	3200	3200
LongBench 8B	KVRoute-RankOnly	0.0059	-0.0001	1.5197	0.2920	3200	3200
LongBench 8B	KVRoute-RankOnly-AS	0.0101	0.0025	2.6459	0.9548	3200	3200
LongBench 8B	KVRoute-RankOnly-Fixed	0.0115	-0.0025	–	–	3200	0
LongBench 8B	OutcomeForest	–	–	–	–	0	0
LongBench 8B	OutcomeForest+Guard	–	–	–	–	0	0
LongBench 8B	Random	–	–	–	–	0	0
RULER 3B	KVRoute	0.0039	-0.0037	0.0000	0.0000	6227	6227
RULER 3B	KVRoute-Encoder	0.0077	-0.0060	0.0763	0.0357	6500	6500
RULER 3B	KVRoute-RankOnly	0.0139	-0.0126	1.0197	-0.6883	6500	6500
RULER 3B	KVRoute-RankOnly-AS	0.0064	-0.0057	1.3180	-1.2001	6500	6500
RULER 3B	KVRoute-RankOnly-Fixed	0.0280	-0.0184	–	–	6500	0
RULER 3B	OutcomeForest	–	–	–	–	0	0
RULER 3B	OutcomeForest+Guard	–	–	–	–	0	0
RULER 3B	Random	–	–	–	–	0	0
LongBench-v2 3B	KVRoute	0.0024	-0.0024	0.0000	0.0000	46	46
LongBench-v2 3B	KVRoute-Encoder	0.0206	-0.0194	0.8312	-0.3576	503	503
LongBench-v2 3B	KVRoute-RankOnly	0.0020	-0.0020	0.2033	-0.1998	503	503
LongBench-v2 3B	KVRoute-RankOnly-AS	0.0037	-0.0027	0.4060	-0.0023	503	503
LongBench-v2 3B	KVRoute-RankOnly-Fixed	0.0359	-0.0323	–	–	503	0
LongBench-v2 3B	OutcomeForest	–	–	–	–	0	0
LongBench-v2 3B	OutcomeForest+Guard	–	–	–	–	0	0
LongBench-v2 3B	Random	–	–	–	–	0	0

F Extended Limitations and Future Directions

Serving-system scope. All measurements use batch-1 isolation on a dedicated GPU, which makes the prefill/decode decomposition (App. B.1) attributable to the algorithm; under continuous batching, phase boundaries are shared across requests and attribution mixes compressor and scheduler. Production serving would change absolute memory values and could reorder latency. We expect it to change constants rather than the sign of the peak/decode inversion, which follows from *where* a method computes: in our eager-mode stack, attention-based eviction materializes full-prompt buffers before discarding, while fused kernels or chunked prefill could decouple this transient from provisioning. Confirming this requires instrumenting a serving engine.

Memory instrumentation. We measure PyTorch allocations (`torch.cuda.max_memory_allocated`), not isolated KV tensors, so both accountings include weights, activations and workspaces. The constant weight term leaves Pareto dominance unchanged but dilutes relative changes, so reported percentages understate magnitudes relative to the cache itself. The weight floor is confirmed to within 1% (analytic 5.984 vs. minimum observed 6.020 GiB at 3B; 14.958 vs. 15.006 at 8B). M_{dec} is a decode-phase residency proxy, not direct KV size or achievable concurrency.

Operating points and implementations. Compression levels are swept only for the five single-parameter compressors; KIVI-4bit, SimLayerKV, and MiniCache use reference settings as they lack a comparable axis. Latency is measured for the evaluated algorithm–implementation pairs only. Differing kernel maturity across implementations prevents drawing an implementation-independent latency ranking. For the routing default a_{ref} = KIVI-4bit, the $+74.1\%$ latency reflects the realized cost of the particular KIVI reference path used by our harness — including online Triton packing and custom CUDA quantized GEMV alongside separate full-precision residual-cache operations — not an implementation-independent cost of 4-bit KV caching. Faster kernels would raise a_{ref}'s utility, widening the mean-utility gap selectors must close, but would leave its severe-loss rate and the tail rates of the frozen selector decisions unchanged. Selectors retrained on the new latencies could decide differently.

Benchmark validity. Llama-3.2-3B-Instruct scores at floor level on LongBench-v2, so we neither read absolute accuracy nor rank compressors by quality there. The cohort serves as an ultra-long-context memory-stress regime (31.8 GiB FullKV peaks): the accounting inversion is clearly visible, the selector comparison remains exact because all policies replay identical per-request outcomes, and KVRoute's gain there is a memory result, not a quality claim. This does not affect the LongBench, RULER-4096 or 8B conclusions.

Oracle and attainability. The sample oracle is an in-sample ceiling over the evaluated pool, not recoverable performance. Section 4 bounds the recoverable part for the descriptors we can compute today, not for learned representations.

Routing. KVRoute improves on FullKV in utility and on every compressor in tail risk, but beats the best fixed compressor in mean utility on no transfer cohort under any normalisation (App. E.3). Its default-risk estimate, fitted on LongBench alone, does not transfer to RULER. Thresholds are calibrated to $\leq 1\%$ severe loss out-of-fold, yet the worst transfer cohort reaches 1.6%: the target is a calibration constraint, not a guarantee, and a hard budget requires recalibration on target-workload data. All selectors are trained on the LongBench 3B requests, so their LongBench 3B results are in-sample. Selector latency is measured outside target-model serving and reaches 277 ms median at 100k tokens, requiring faster features or a deadline that fails closed to a_{ref}. Determinism relies on greedy decoding; under sampling, the noise terms of Appendix B.4 must be re-estimated. All evidence covers one model family at two scales.

Future work. (i) Fitting the default-risk estimate on multiple benchmarks; (ii) learning execution-aware request features from outcomes; (iii) adding learned-representation groupings to the conditioning ladder; (iv) 3B→8B preference transfer; (v) repeated executions beyond LongBench; (vi) serving-engine instrumentation with continuous batching; (vii) extending the formulation to vision and multimodal caches.