UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

DÉPARTEMENT DES SCIENCES ADMINISTRATIVES

WORKING PAPER NO. 30

Beyond Text: Cache Compression and Knowledge Transfer in Multi-Agent Systems

A Reproducible Study of Fidelity, Communication Cost, and Repeated Handoffs

Simon-Pierre Boucher

Département des sciences administratives
Université du Québec en Outaouais

simon-pierre.boucher@uqo.ca

Gatineau – Pavillon Alexandre-Taché
283, boulevard Alexandre-Taché
Gatineau, Québec, Canada J9A 1L8

October 2026

Version 1.0

Abstract

Abstract to be written after the results.

Keywords: KV cache compression; cache reuse; multi-agent systems; knowledge transfer; prompt compression; structured evidence; reproducibility; Apple Silicon; MLX

JEL Classification: C63, C88, D83, L86, O33

Contents

1 Introduction **5**

2 Related work **6**

3 Problem formulation and taxonomy **7**

3.1 Protocol notation 7

3.2 Accounting 8

3.3 Taxonomy 9

3.4 Research questions and registered hypotheses 9

4 Methods **11**

4.1 Track A: text, structured messages and evidence 11

4.2 Track B: within-model KV compression and reuse 12

4.3 Track C: cross-checkpoint transfer 13

4.4 Track D: hosted receivers 13

5 Experimental protocol **14**

5.1 Models, hardware and runtimes 14

5.2 Task families and splits 14

5.3 Cache correctness before performance 15

5.4 Pilot, freeze and campaign 16

5.5 Statistics 16

6 Results **16**

6.1 Single-handoff compression frontier (E1) 16

6.2 Two-node reuse and locality (E2/E6) 20

6.3 Hosted receivers (Track D) 21

7 Adaptive representation and placement (E8) **22**

8 Failure analysis and limitations **24**

9 Reproducibility and execution disclosure **25**

10 Conclusion **26**

Appendices **29**

A Registered hypotheses **29**

B Source audit and unexecuted methods **30**

C Correctness tests **30**

D Prompts **31**

E Pseudocode **32**

F Per-attempt record schema **32**

G Additional tables **33**

1. Introduction

Agents built on large language models rarely work alone. A reader agent summarizes a filing for an analyst agent; a crawler hands extracted facts to an aggregator; a long document is processed once and then queried many times by different workers. Every such handoff chooses a *representation* for the information that crosses the boundary between two model executions: the raw text, a shortened text, a prose summary, a structured record with provenance, a pointer to shared memory, or—when the two executions share a checkpoint—the attention state (the key–value, or KV, cache) that the first model built while reading. A growing literature proposes to compress these states (Liu et al., 2024; Li et al., 2024; Cai et al., 2024; Zhang et al., 2023; Xiao et al., 2024; Zweiger et al., 2026), to reuse them across requests, and, most recently, to translate them between different checkpoints or model families so that a receiver can "inherit" a sender's reading without re-reading the source (Fu et al., 2026; Zou et al., 2026; Alabdulkareem et al., 2026).

The engineering appeal is clear: a receiver that loads a cache skips the prefill of the document, and a compressed cache shrinks memory. The scientific question is less clear. Compression methods are evaluated with different models, different information-access assumptions (does the compressor see the question?), different budgets (tokens, positions, bits, bytes), and different measurement boundaries (kernel time versus end-to-end time, resident tensors versus serialized payloads). Text baselines are often weak or absent. Cache sizes are usually quoted in positions or bits per value rather than in the bytes that actually cross a network, and the receiver side of a handoff—deserialization, loading, fresh-query processing—is seldom counted. Nor is the behaviour of these representations studied over *repeated* handoffs, where small distortions compound.

This paper asks a single question under a common measurement harness: *under which task, model, cache, network and reuse conditions does each communication representation preserve useful information at the lowest end-to-end resource cost?* We answer it empirically, with experiments executed from a single research operator (Claude Code, an agentic coding assistant) on commodity Apple-Silicon hardware, using the MLX runtime, whose KV caches are ordinary tensors that can be inspected, sliced, quantized, serialized and loaded back. All quality numbers come from real model generations; all byte counts are measured on serialized payloads; all latencies are wall-clock spans with device synchronization; and the complete per-attempt record of every trial is released.

Contributions.

1. An **experimentally grounded taxonomy** of communication representations (Table 2) that separates text compression, KV compression, cache reuse, cache transfer and parametric transfer, and records for each method its question-access regime, its compatibility requirements and whether it composes across sources.

2. A **common harness** (`wp30`) with a validated cache adapter layer (identity round trip, prefix equivalence, position preservation, packed-byte accounting, manifest-checked loading), a per-attempt record schema, deterministic scorers, evaluator-isolated answer keys, and document-level cluster bootstrap statistics.
3. **Quality/resource trade-off maps** for twenty-two conditions on two 3B-parameter model families and three task families (synthetic private-fact ledgers, public long-document QA, executable tasks), under question-blind and question-aware regimes, with measured bytes, latency decompositions and peak memory.
4. A study of **repeated handoffs** (1 to 8 transfers) with an evaluator-only fact ledger that measures retention, unsupported additions and the cost of metered repair, and a **fan-in** study with conflicting evidence.
5. **Genuine two-node cache transfer** measurements on a 1 GbE LAN against remote text prefill and local recompute, with a measured break-even model.
6. A **same-family learned KV mapper** (Qwen2.5-1.5B $\rightarrow$ Qwen2.5-3B) trained on train-split caches only, evaluated against text handoffs and an identity-transfer negative control; and a **development-fitted adaptive router** evaluated on held-out data against the best development-selected static method.

Several results are negative or conditional, and we report them as such. Cross-family cache transfer was *not* executed: the only published cross-family method with code (Alabdulkareem et al., 2026) has no released checkpoint and no licence, and our budget did not permit training a mapper of its size; the study is therefore a *partial* transfer study in that respect (Section 8). We also do not claim that any receiver *learned* anything: every effect measured here is contextual, i.e., it vanishes when the message or cache is removed.

2. Related work

Text and prompt compression. Shortening the text a receiver reads is the oldest form of inter-agent compression. Beyond truncation and extractive selection, learned compressors drop tokens with a classifier trained on distilled data (Pan et al., 2024); LLMLingua-2 is task-agnostic by construction, which places it in our *question-blind* regime. Abstractive summaries and structured extraction are the dominant representations in deployed agent frameworks; we treat them as first-class baselines rather than as foils, tuning their prompts on development data only.

KV-cache quantization, eviction and compaction. KIVI (Liu et al., 2024) quantizes keys per channel and values per token to 2 bits with a full-precision residual window, motivated by outlier channels in keys. Eviction methods select positions to keep: H_2O (Zhang et al., 2023) accumulates

attention mass during prefill; SnapKV (Li et al., 2024) scores prefix positions by the attention of an observation window at the end of the prompt, pooled and selected per head; PyramidKV (Cai et al., 2024) allocates decreasing budgets to higher layers; StreamingLLM (Xiao et al., 2024) keeps attention sinks and a recent window and *re-assigns* positions within the cache. Attention Matching (Zweiger et al., 2026) compacts a cache into fewer positions by solving for keys and values that reproduce attention outputs on reference queries. Our audit (Appendix B) records for each method whether its selection observes the question: SnapKV- and PyramidKV-style selection are question-aware by construction when the prompt ends with the question, which matters for the comparison with reusable, question-blind messages.

Cache reuse and transfer. Prefix caching reuses an identical token prefix within one checkpoint; it is exposed by local runtimes (Apple ML Explore contributors, 2026; Hugging Face, 2026) and, as a billing abstraction, by hosted APIs (Anthropic, 2026). Transfer across checkpoints requires a mapping: Cache-to-Cache (Fu et al., 2026) trains a fuser that injects a sharer's cache into a receiver that still prefills the source; LatentMAS (Zou et al., 2026) exchanges full caches and latent thoughts among agents that share the same architecture; SwaT (Alabdulkareem et al., 2026) trains a cross-family translator and combines it with compaction, but releases no checkpoint. Concurrent work on cross-model cache bridging (see the audited list in `literature/sources.csv`) is active; we make no novelty claim about shared latent spaces and run only a same-family linear mapper as a bounded, fully disclosed experiment.

Evaluation. LongBench (Bai et al., 2024) supplies public long-document QA with F1 scoring; RULER (Hsieh et al., 2024) generates controlled retrieval tasks. We combine both ideas: public subsets (QASPER, 2WikiMultihopQA) for external validity, and a new procedural generator of private facts (identifiers, amounts, dates, negations, timestamped revisions, unanswerable questions) so that communication dependence can be tested without pre-training contamination.

3. Problem formulation and taxonomy

3.1. Protocol notation

Let d be a source document, q a downstream query, $y^\star$ its reference answer, S the sender model and R the receiver model. A communication method c under a resource setting b and a question-access policy a produces a message

$$m = E_{c,b}(d; S, a), \qquad \hat{y} = G_R\big(q, D_c(m), \mathscr{A}\big), \tag{1}$$

Table 1. Terminology used in this paper.

Term	Operational meaning	Error avoided
Text compression	Shorter text consumed by a receiver	Calling it KV-cache compression
KV compression	Quantization/selection/transformation of attention tensors	Measuring only a shorter prompt
Cache reuse	Reuse of a compatible previously computed prefix state	Assuming one family ⇒ compatible
Cache transfer	Moving/mapping cache state for a receiver	Sending a summary and calling it latent transfer
Knowledge transfer	Better access to task information via a message	Claiming a frozen receiver learned a skill
Latent communication	An internal representation crosses executions	Calling retrieval embeddings a KV state
Shared memory	Addressable external records under a protocol	Treating an identifier as free content access
Prefill-free handoff	Receiver does not prefill the transferred document	Claiming the fresh query needs no processing
Evidence traceability	Output linked to a resolvable source span	Treating self-reported confidence as proof

where D_c denotes decoding, loading or mapping at the receiver and $\mathscr{A}$ is the explicitly allowed artifact or retrieval interface (empty for all methods except A6, the reference-plus-retrieval method, and the repairable chains of Section **??**). Equation (1) is notation for a protocol, not a claim that methods share an implementation: for text methods m is a string the receiver prefills; for cache methods m is a set of tensors installed as the receiver's attention state, and only the fresh query tokens are prefilled.

Question access. Under the *question-blind* regime the sender observes d and a generic instruction but not q; the message is reusable across the questions of a document. Under the *question-aware* regime both sides may use q. The two regimes are reported separately throughout; a selection method that observes the question (SnapKV-style) is never compared as if it were blind.

Terminology. We follow a fixed vocabulary (Table 1). In particular, *text compression* is a shorter text consumed by a receiver; *KV compression* is quantization, selection or transformation of actual attention tensors; *cache reuse* reuses a compatible, previously computed prefix state; *cache transfer* moves or maps a state for a receiver; and *knowledge transfer* means improved access to task-relevant information through a defined message—never permanent learning.

3.2. Accounting

For a dense cache with uniform layers the stored tensor size is

$$M_{KV} = 2BLTH_{KV}d_h s, \tag{2}$$

with batch B, layers L, stored positions T, KV heads H_{KV}, head dimension d_h and s bytes per scalar. We use (2) only as a consistency check: the harness measures stored tensor bytes from the arrays themselves (which, for the packed quantized caches, include the uint32 words, the per-group scales and biases) and measures the serialized payload from the file written to disk, including the compatibility manifest. Two ratios are reported with explicit denominators,

$$CR_{\text{payload}} = \frac{\text{reference serialized payload bytes}}{\text{method serialized payload bytes}}, \qquad CR_{KV} = \frac{\text{native cache tensor bytes}}{\text{stored cache tensor and metadata bytes}}, \tag{3}$$

and a text-token ratio is never treated as a cache-position ratio.

Timing boundaries. Each trial records separate wall-clock spans with device synchronization before and after (`mx.synchronize`): sender prefill and message generation (*encode*); document prefill into the receiver cache (*doc prefill*); compression or selection; serialization and transport codec; deserialization and cache installation (*load*); fresh query prefill up to the first content token (TTFT); decoding; and, for metered methods, retrieval. For a serial workflow,

$$T_{\text{total}} = T_{\text{queue}} + T_{\text{read}} + T_{\text{sender}} + T_{\text{encode}} + T_{\text{serialize}} + T_{\text{transport}} + T_{\text{load}} + T_{\text{receiver}} + T_{\text{tools}}. \tag{4}$$

Three scenarios are assembled from the spans: *cold* (no reusable state exists: build + compress + load + receiver), *warm per-query* (the state exists: load + receiver), and *amortized over N queries*,

$$\bar{C}(N) = \frac{C_{\text{training}} + C_{\text{setup}} + C_{\text{document}}}{N} + C_{\text{query}}, \tag{5}$$

where C_{training} is zero for all training-free methods and is reported separately for the learned mapper. Components that a hosted provider does not expose are recorded as missing, not as zero.

3.3. Taxonomy

Table 2 lists every method with its representation, question-access regime, training requirement, compatibility requirement, composability across sources, and its measured status in this study. Methods that were audited but not executed appear with the reason.

3.4. Research questions and registered hypotheses

Seven research questions (RQ1–RQ7) and ten falsifiable hypotheses were registered before any held-out outcome was examined (`protocol/hypotheses.md`; reproduced in Appendix A). In brief: RQ1 compression frontier; RQ2 transfer compatibility; RQ3 whether rankings change across resource denominators; RQ4 repeated handoffs; RQ5 structured evidence versus prose; RQ6 locality

Table 2. Method taxonomy and coverage. "Measured" = executed in WP30 with real model inference; the last rows are audited methods that were not executed, with the reason.

id	representation	question_access	training	compatibility	composable	status_in_wp30
A0	none (task only)	–	none	any	–	measured
A1	raw text	blind	none	any receiver	concatenation	measured
A2	text head/tail	blind	none	any	concatenation	measured
A3	extractive sentences (BM25/centrality)	blind or aware	none	any	concatenation	measured
A4	abstractive summary	blind or aware	none (prompted sender)	any	concatenation	measured
A5	typed facts + source spans + unknowns (JSON)	blind or aware	none (prompted sender)	any	record merge	measured
A6	reference index + metered retrieval	blind index, aware retrieval	none	any + source access	shared memory	measured
A7	raw text, zstd transport codec	blind	none	any	concatenation	measured
A8	LLMLingua-2 token dropping	blind	pretrained compressor (official)	any	concatenation	measured
B0	native bf16 KV cache	blind	none	same checkpoint	no	measured
B1	serialized KV cache (manifest-checked)	blind	none	same checkpoint + revision	no	measured (process + 2-node)
B2	packed quantized KV (8/4/2 bit)	blind	none	same checkpoint	no	measured
B3	sink+recent retention (StreamingLLM-inspired)	blind	none	same checkpoint	no	measured
B4	random retention (control)	blind	none	same checkpoint	no	measured
B5	attention-informed retention (SnapKV rule)	aware (blind variant tested)	none	same checkpoint	no	measured
B6	pyramid layer budgets (PyramidKV rule)	aware	none	same checkpoint	no	measured
B9	selection then packed quantization	aware	none	same checkpoint	no	measured
C0	sender cache fed as receiver cache	blind	none	incompatible (negative control)	no	measured (control)
C2	linear KV mapper (ridge, un-rotated K/V)	blind	ridge on train caches	trained pair (Qwen 1.5B→3B)	no	measured
C3	mapper + attention selection	aware	ridge on train caches	trained pair	no	measured
–	SwaT cross-family mapper	blind	300M mapper (not public)	Granite→SmolLM	no	not run: no checkpoint/licence
–	Attention-Matching compaction	blind	none (optimization)	same model	no	not run: CUDA/vLLM code; MLX port out of scope
–	C2C fuser	blind	released fusers (Qwen3)	Qwen3 receivers	no	not run: CUDA; receiver still prefills source
–	H2O eviction	cumulative	none	same checkpoint	no	not run: requires attention scores during whole prefill
–	KIVI per-channel key quantization	blind	none	same checkpoint	no	not run: CUDA kernels; motivation confirmed by B2 Qwen result

versus moving the cache; RQ7 adaptive routing. A primary confirmatory family of six paired comparisons is Holm-corrected; all other comparisons are exploratory.

4. Methods

All methods are implemented in one Python package (`src/wp30`) on top of `mlx-lm` 0.32.2 (Apple ML Explore contributors, 2026), whose KV cache for a layer is a pair of tensors $K, V \in \mathbb{R}^{B \times H_{KV} \times T \times d_h}$ with keys stored *after* rotary position embedding (RoPE) and a logical offset that drives the positions of new tokens. We distinguish *official* code (run as released), *faithful re-implementations* (same rule and defaults, re-coded for MLX), *adapted*, *inspired* and *custom* baselines; the label of every method is in Table 2 and in `literature/method_notes.md`.

4.1. Track A: text, structured messages and evidence

A0 no message.
The receiver gets the question only (prior/guessing floor).

A1 full raw context.
The document is placed in the receiver prompt; receiver prefill is counted.

A2 head/tail truncation.
Fixed rule (half the token budget from the head, half from the tail, under the receiver tokenizer).

A3 extractive selection.
Deterministic sentence selection under a token budget: BM25 of the question over sentences (question-aware) or a centrality score weighted by the density of digits and identifiers (question-blind).

A4 abstractive summary.
Generated by the sender with a fixed prompt and a hard output cap equal to the budget; the prompt asks for verbatim identifiers, numbers, dates and negations and, in the blind regime, for everything concrete because the questions are unknown.

A5 structured evidence.
The sender emits JSON facts {entity, attribute, value, quote} plus optional `notes` and `unknowns`; each quote is resolved to a character span of the source, and unresolved quotes are counted. The receiver sees values and spans. Truncated JSON is repaired at the last complete object.

A6 references plus retrieval.
The message is A5 *without values*: a reference index {entity, attribute, span}. At answer time the receiver ranks references by BM25 of the question and fetches at most six spans (±40/160 characters) under a 2 000-byte budget; every lookup and fetched byte is recorded and added to the message bytes.

A7 lossless transport codec.
The raw text is compressed with Zstandard (level 3) for transport and decompressed at the receiver; the receiver prefill is unchanged. Codec times are measured.

A8 learned prompt compression.
LLMLingua-2 (Pan et al., 2024), official package (`llmlingua` 0.2.2) with the released `xlm-roberta-large-me` compressor, rate = budget/source tokens, run on the Metal backend of PyTorch (fp32). It is question-agnostic by construction.

A4 and A5 use the same sender checkpoint, the same budget and prompts of comparable length; both prompts were tuned on development data only. Receiver prompts are identical across methods except for the label of the material slot (Appendix D).

4.2. Track B: within-model KV compression and reuse

Every Track-B condition manipulates the receiver's real cache tensors and runs receiver inference from the resulting state. The receiver prefix is the chat-templated system prompt plus the document; the fresh input is the question and the assistant header, so that the cache boundary is identical for all conditions (*no duplicated boundary token* is a tested property).

B0 native cache.
The dense bf16 cache, branched (deep-copied) for every question.

B1 serialized reuse.
B0 serialized to `safetensors` with a compatibility manifest (checkpoint revision, config, tokenizer and chat-template hashes, prefix-token hash and count, per-layer shapes/dtypes/quantization, positional semantics, runtime versions) and loaded back; the loader rejects a mismatching manifest before any model execution.

B2 packed quantization.
`mx.quantize` along the head dimension (groups of 64 channels; 32 for 2-bit), asymmetric affine with a scale and a bias per group, packed into `uint32` words, and consumed by the quantized attention kernel—so the compressed storage is the execution representation, not a transport format that is expanded before attention. Bits $\in \{8, 4, 2\}$.

B3 sink+recent.
Keep the first four positions and the most recent positions up to the budget. Unlike StreamingLLM, retained keys keep their original positions and the logical offset is preserved (new tokens see a positional gap); it is a retention *pattern*, labelled "inspired".

B4 random.
Uniformly random positions at the same budget (seeded control).

B5 attention-informed.
SnapKV's rule: per KV head, the attention of an observation window onto prefix positions

(softmax, averaged over the window's queries and over the query heads of the GQA group), 1-D max-pooled with kernel 7, top-k per head, window always kept. The window is the fresh question (aware) or a generic 32-token instruction (blind). Window queries are computed in a separate pass over a copy of the cache, whose cost is recorded as compression time, so the main attention kernel stays fused.

B6 pyramid budgets.
PyramidKV's per-layer budget rule (verified against the KVCache-Factory code) with B5 scoring inside each layer; because layers retain different lengths, the fresh query is processed token by token (recorded).

B9 selection then quantization.
B5 followed by B2 (8-bit).

Selection produces a *sparse cache*: a subclass that stores only retained positions (per-head gathers are allowed), keeps the logical offset, and supplies an explicit boolean mask when several fresh tokens are appended (retained positions visible to all, causal among the new tokens). Correctness is established before any comparison (Section 5.3).

4.3. Track C: cross-checkpoint transfer

Level 1 of the compatibility ladder (same checkpoint, separate processes and separate nodes) is B1 executed through a TCP worker (Section 6.2). Level 2 (same family, different size) uses a *linear KV mapper*: for each receiver layer ℓ_r of Qwen2.5-3B we align a sender layer $\ell_s = \mathrm{round}(\ell_r\,(L_s-1)/(L_r-1))$ of Qwen2.5-1.5B (both have two KV heads of dimension 128 and identical RoPE parameters, and they share the tokenizer, so sender and receiver caches are token-aligned), un-rotate the sender keys with their known positions, and fit by ridge regression a map from $[K_s, V_s] \in \mathbb{R}^{512}$ (plus bias) to $[K_r, V_r] \in \mathbb{R}^{512}$ on train-split caches only; mapped keys are re-rotated. The loss is reconstruction only ($\lambda_{\text{task}} = \lambda_{\text{distill}} = 0$); the mapper has $36 \times 513 \times 512 \approx 9.5$M parameters and is saved with a manifest (training documents, tokens, seconds, held-out reconstruction error). C0 feeds the sender cache into the receiver *without* mapping (shapes coincide): an incompatible-state negative control. C3 applies B5 selection to the mapped cache (mapping then compaction). This is not SwaT (Alabdulkareem et al., 2026), which trains a ∼300M mapper with generation and distillation terms across families; no SwaT checkpoint or licence was available (Appendix B). Level 3 (cross-family) was therefore not executed.

4.4. Track D: hosted receivers

A0–A6 messages built by the local sender are answered by two hosted models (Anthropic `claude-haiku-4-5`; OpenAI `gpt-5-nano`) through their documented APIs, with streaming to measure the time to the

Table 3. Checkpoints, geometry and measured prefill (ms, laptop). Cache bytes per token are the dense bf16 formula (2) with $B = 1$.

alias	checkpoint	rev.	L	H_kv	H_q	d_h	max pos	cache B/token	prefill 8k	prefill 16k
qwen3b	mlx-community/Qwen2.5-3B-Instruct-bf16	bec1ad658e38	36	2	16	128	32768	36864	968	2286
llama3b	mlx-community/Llama-3.2-3B-Instruct-bf16	6d88ba43024f	28	8	24	128	131072	114688	1007	2449
qwen1b5	mlx-community/Qwen2.5-1.5B-Instruct-bf16	4ae77cb209f0	28	2	12	128	32768	28672	494	1201

first content token, provider usage fields, documented prompt caching (Anthropic `cache_control`; OpenAI automatic prefix caching) and an estimated cost from the published price tables. The provider's cached-token field is reported as billing information, never as "KV bytes saved".

5. Experimental protocol

5.1. Models, hardware and runtimes

Table 3 lists the checkpoints (all ungated `mlx-community` conversions with bf16 weights, so weight quantization is not a confound), their attention geometry and measured prefill times. The local node is a MacBook Pro with an Apple M5 Max (18 cores, 48 GiB unified memory, macOS 27.0), Python 3.12.13, `mlx` 0.32.4, `mlx-lm` 0.32.2. Two-node experiments use two Mac Studio M3 Ultra nodes of the author's cluster (M3U96a as sender, M3U96b as receiver worker; 96 GB each; identical `mlx`/`mlx-lm` pins and checkpoint revisions) connected by a wired LAN measured at 940 Mbit/s (iperf3, four streams) with 1.2 ms RTT. Both cluster nodes concurrently hosted a CPU-light workload from another study (load average ≈ 1.8 on 32 cores); timing on the laptop was taken with no other GPU process.

5.2. Task families and splits

Synthetic private-fact ledgers.

A deterministic generator (seeded) writes a fictional capital-projects register: six projects per document with invented names, random identifiers (`XX-00X-0000`), amounts and currencies, dates, sites, suppliers, approvals or rejections, cancelled-versus-postponed statuses, timestamped revisions with an explicit precedence rule, a list of "proposals under review" with no attributes (distractors with similar names), and boilerplate filler to reach 2 000, 8 000 or 16 000 target tokens. Fifteen questions per document cover single-fact retrieval at varying positions, exact copying, multi-hop joins, negation, revision/precedence, temporal order, aggregation, unit conversion and two unanswerable questions (correct action: `unknown`). An evaluator-only fact ledger (62 atomic facts per document) and the answer key are stored apart from the agent-visible text. The facts are not inferable from pre-training. A procedurally generated ledger is not a real

Table 4. Datasets, independent units (documents), questions and median length by split.

family	split	documents	questions	median_chars
exec	dev	30	30	7847
exec	test	40	40	7817
exec	train	20	20	7866
public_2wikimqa	dev	56	56	25,419
public_2wikimqa	test	144	144	25,488
public_hotpotqa	dev	61	61	65,814
public_hotpotqa	test	139	139	60,944
public_qasper	dev	50	50	19,844
public_qasper	test	150	150	23,154
synthetic	dev	30	450	29,225
synthetic	test	40	600	29,203
synthetic	train	40	600	29,196

filing.

Public long-document QA.

LongBench v1 (Bai et al., 2024) subsets QASPER (Dasigi et al., 2021) and 2WikiMultihopQA (Ho et al., 2020) (raw `data.zip`; the dataset script is no longer supported by the `datasets` library), official F1 with the LongBench normalization. Documents are used in full. We run *subsets* (the first 40 test documents of each subset in manifest order) and publish the identifiers. These texts may appear in pre-training corpora; they serve external validity, not the contamination-free test of communication dependence.

Executable tasks.

A generated engineering ticket specifies six business rules for a function `net_revenue(rows)` together with a buggy implementation missing two rules; the agent must output the corrected function, which is executed in an isolated `python -I` subprocess against five hidden test cases (exact float comparison). Agents never see the tests.

Splits are by document (train/dev/test seeds for the generated families; a hash rule for the public subsets), with exact-duplicate checks across splits (none found). Train is used for mapper fitting and prompt tuning, dev for method selection and router fitting, test for the locked campaign (Table 4). Answer keys live in `data/generated/keys` and are read only by the scorer; a leakage test asserts that agent-visible files contain no answers, facts or tests.

5.3. Cache correctness before performance

Twelve tests run a real 1.5B checkpoint (Appendix C). Identity round trip (serialize, load) and 100 % retention through the sparse adapter reproduce the native next-token logits *exactly*. Cached-prefix-plus-query and full prefill agree within a registered distributional tolerance: MLX kernels are not bitwise deterministic across prefill chunkings even in float32 (we measured a maximum logit

difference of 0.10 between a one-chunk and a token-by-token prefill, with $\max|\Delta p| \approx 7 \times 10^{-7}$ and $\mathrm{KL} \approx 3 \times 10^{-8}$), so equivalence is defined as $\mathrm{KL} < 10^{-5}$, $\max|\Delta p| < 10^{-4}$ and identical greedy top-1 in float32 ($\mathrm{KL} < 10^{-3}$ in bf16). The sparse-cache mask path agrees with token-by-token appends; preserving original positions differs measurably from relabelling them; packed storage has the intended bit width (uint32 words, ratios 1.9/3.6/6.4 for 8/4/2 bits on a 412-token prefix including scales and biases); the loader rejects a wrong prefix hash or a different checkpoint; branches do not share mutable state; and the no-message control does not know the private answer.

5.4. Pilot, freeze and campaign

A development pilot (`E1_pilot`: 16 documents, 31 conditions, two model pairs) calibrated budgets, revealed implementation failures and estimated variance and run time. The confirmatory protocol was then frozen (`protocol/frozen.json`, hash `404c52b34d964693`) with the final hypotheses, prompts, budgets, scorers and analysis plan, before any test-split item was evaluated. The core single-handoff campaign (`E1_core`) covers 160 test documents (40 per family), 22 conditions and two model pairs (13 600 scheduled trials); the handoff, fan-in, transfer, network and hosted campaigns are described with their results.

5.5. Statistics

The unit is a (document, question, condition, pair) trial; documents are the independent clusters. Means and paired differences on identical instances are reported with document-level cluster-bootstrap 95 % intervals (2 000 replicates). Six registered primary comparisons are Holm-corrected; everything else is exploratory. Quality is accuracy (synthetic, executable) or F1 (public); failed or invalid attempts count as 0 in the operational quality and are also tabulated by category. Latency is reported as p50/p95 with sample counts, from the spans defined in Section 3.

6. Results

All numbers in this section are computed from the released per-attempt records by `src/wp30/analysis/analyze.` every table cell and plotted point resolves to run identifiers through `results/provenance.json`. Quality is accuracy for the synthetic and executable families and LongBench F1 for the public subsets; intervals are 95 % document-level cluster-bootstrap intervals; test statistics follow the registered plan (Section 5). Scheduled core trials: 13,280.

6.1. Single-handoff compression frontier (E1)

Table 5 and Figure 1 summarize the core campaign: 160 test documents, 22 conditions, two model pairs, 13 280 scheduled trials, all executed; 74 attempts ended as *invalid output* (empty or non-text generations, all from 4-bit Qwen caches and random eviction) and 240 as *implementation error* (LLMLingua-2 on the Llama pair: the cluster node lacked the PyTorch runtime; those trials were retried on the laptop and both attempts are kept, Appendix G). Failures count as zero quality.

Lossless representations are interchangeable; nothing else is. Full text (A1), zstd-compressed text (A7), the native cache (B0), the serialized and reloaded cache (B1) and the 8-bit packed cache (B2-8) give the same answers within noise on every family and pair (T2: B2-8 vs B0 +0.3 points, Holm $p = 0.302$; T6: A1 vs B0 $p = 0.302$; $n = 640$ paired instances each). This is the expected consistency result—the cache *is* the document as the receiver read it—and it is also the only region of the frontier where representation choice is free of quality cost. The A7 result confirms H8: the codec shrinks the text payload $\approx 3.5\times$ (29,208 bytes raw for an average synthetic document) with no effect on quality or receiver time, while on bf16 caches zstd saves $\approx 20\,\%$ (Section 6.2).

Packed 4-bit quantization is model-dependent (H2 confirmed). At 4 bits with 64-element groups, Llama-3.2-3B keeps 98 % accuracy on private facts and 100 % on executable tasks, whereas Qwen2.5-3B collapses to 0 % and 0 % (T3, both pairs pooled: -42.5 points, Holm $p = 0.000$). The failure is total (garbled tokens, 29 empty generations), it reproduces in the native `mlx-lm` path, and it is the per-token affine quantization of Qwen's outlier key channels that KIVI's per-channel scheme was designed to avoid. Eight bits are safe for both models; halving the cache is therefore free for both, quartering it is free for one.

Eviction without the question is destructive; eviction with the question is nearly free—for retrieval, not for code. Keeping a quarter of the positions by the sink-plus-recent rule (B3) leaves 20 % (Qwen) and 18 % (Llama) on private facts, close to the share of facts that happen to sit in the retained window; random retention (B4) is at zero and produces incoherent output on both models. The SnapKV-style rule with the question as observation window (B5-aware) recovers 78 % (Qwen) and 98 % (Llama) at the same 1/4 budget (T4 vs B3: +43.9 points, Holm $p = 0.000$, H3 confirmed in its first half), and the pyramid allocation (B6) and the selection-plus-8-bit composition (B9) behave like B5. With a generic window instead of the question (B5-blind) the same algorithm falls to 12 %/19 %: attention-based selection is a *question-aware* method and must be compared as one (H7). On the executable family every eviction method, aware or not, scores zero on both models: a code specification cannot be quartered, because every rule in the ticket matters for the

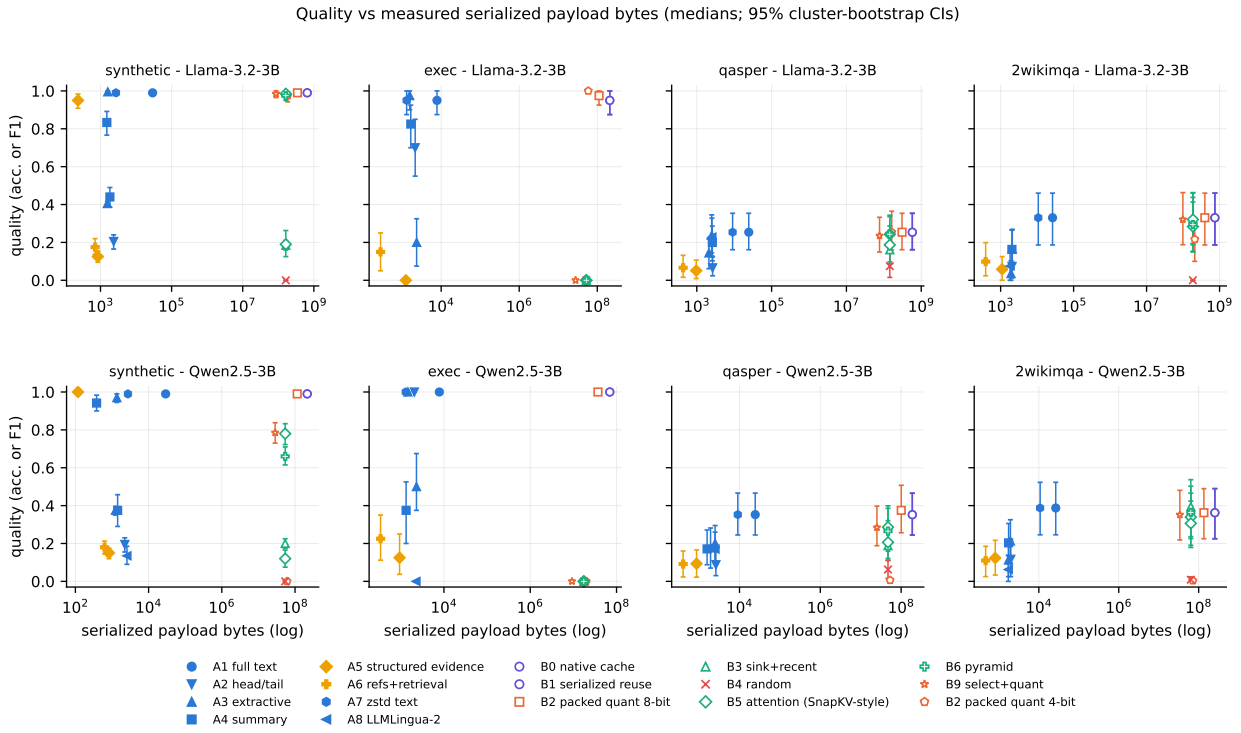


Figure 1. Quality against measured serialized payload bytes (log scale), by task family (columns) and model pair (rows). Each point is one condition (median payload, mean quality, 95 % cluster-bootstrap CI). Text methods cluster at 10^2–10^4 bytes, cache methods at 10^7–10^9 bytes: a 3B-parameter cache of a document is three to four orders of magnitude larger than the document. Question-aware and question-blind conditions are both shown here; Appendix figures separate them.

Table 5. Primary comparison: quality (%, with 95 % cluster-bootstrap CI) per condition, family and pair. Synthetic = private-fact accuracy (5 questions per document: single fact, exact copy, multi-hop), exec = hidden-test pass (all 5 tests), QASPER/2Wiki = LongBench F1. "blind"/"aware" = question access of the compressor. Latency and bytes for the synthetic family in Table 6.

condition	synthetic Qwen	synthetic Llama	exec Qwen	exec Llama	qasper Qwen	qasper Llama	2wikimqa Qwen	2wikimqa Llama
A0	0 [0,0]	0 [0,0]	0 [0,0]	0 [0,0]	4 [1,10]	5 [1,12]	11 [4,21]	6 [0,14]
A1	99 [98,100]	99 [98,100]	100 [100,100]	95 [88,100]	35 [24,47]	25 [15,36]	39 [25,54]	33 [20,47]
A2 512	20 [16,23]	20 [17,24]	100 [100,100]	70 [55,85]	9 [3,16]	7 [2,13]	12 [2,22]	7 [1,15]
A3 512 aware	97 [94,99]	100 [98,100]	50 [35,65]	20 [8,32]	20 [11,29]	24 [15,35]	21 [11,33]	16 [7,27]
A3 512 blind	38 [36,40]	40 [40,42]	100 [100,100]	98 [92,100]	18 [8,29]	14 [7,23]	11 [2,21]	3 [0,10]
A4 512 aware	94 [89,98]	83 [77,90]	–	–	–	–	–	–
A4 512 blind	38 [29,46]	44 [38,50]	38 [22,52]	82 [70,92]	17 [9,27]	20 [11,30]	20 [10,32]	16 [7,27]
A5 512 aware	100 [100,100]	95 [91,98]	–	–	–	–	–	–
A5 512 blind	15 [12,18]	12 [9,16]	12 [2,22]	0 [0,0]	9 [2,18]	5 [1,11]	12 [4,23]	6 [0,13]
A6 512 blind	18 [14,22]	18 [14,22]	22 [10,35]	15 [5,28]	9 [2,17]	7 [2,13]	11 [2,22]	10 [2,19]
A7 zstd	99 [98,100]	99 [98,100]	100 [100,100]	95 [88,100]	35 [24,47]	25 [15,36]	39 [25,54]	33 [20,47]
A8 512 blind	14 [9,18]	0 [0,0]	0 [0,0]	0 [0,0]	17 [10,25]	23 [13,33]	6 [0,14]	8 [0,18]
B0	99 [98,100]	99 [98,100]	100 [100,100]	95 [88,100]	35 [24,47]	25 [15,36]	36 [22,50]	33 [20,47]
B1	99 [98,100]	99 [98,100]	100 [100,100]	95 [88,100]	35 [24,47]	25 [15,36]	36 [22,50]	33 [20,47]
B2 bits=4	0 [0,0]	98 [94,100]	0 [0,0]	100 [100,100]	1 [0,2]	25 [15,36]	0 [0,1]	22 [10,34]
B2 bits=8	99 [98,100]	99 [98,100]	100 [100,100]	98 [92,100]	38 [25,50]	25 [15,36]	36 [22,50]	33 [20,47]
B3 1/4	20 [18,23]	18 [16,21]	0 [0,0]	0 [0,0]	19 [10,29]	16 [9,25]	39 [25,55]	32 [19,46]
B4 1/4	0 [0,0]	0 [0,0]	0 [0,0]	0 [0,0]	6 [2,13]	7 [1,16]	1 [0,2]	0 [0,0]
B5 1/4 aware	78 [72,83]	98 [96,100]	0 [0,0]	0 [0,0]	29 [19,40]	24 [15,34]	34 [20,48]	32 [19,46]
B5 1/4 blind	12 [8,17]	19 [12,26]	0 [0,0]	0 [0,0]	21 [11,32]	19 [10,29]	31 [18,45]	28 [16,42]
B6 1/4 aware	66 [61,70]	98 [95,100]	0 [0,0]	0 [0,0]	27 [16,38]	24 [15,34]	36 [23,51]	29 [16,44]
B9 bits=8 1/4 att aware	78 [73,84]	98 [96,100]	0 [0,0]	0 [0,0]	28 [18,40]	24 [14,34]	35 [22,49]	32 [19,46]

hidden tests. On the public QA subsets eviction costs little (the answers are short spans), but the F1 ceiling of 3B receivers is low (A1 $\approx$ 0.25–0.39).

Text compression at a 512-token budget: summaries beat structured evidence, extraction beats both when the question is known. Question-blind summaries (A4) retain 38 % (Qwen) and 44 % (Llama) of the private-fact answers; structured JSON evidence with spans (A5) only 15 %/12 % (T1: -27.0 points, Holm $p = 0.000$, $n = 400$). H1's expectation that typed facts would protect identifiers and numbers is *rejected* at this budget: the sender spends its 512 output tokens on exact quotes (provenance) and emits $\approx$8 facts out of 62 per document, so most answers are simply absent from the message; the receiver answers "unknown" (correctly abstaining) rather than hallucinating. References plus metered retrieval (A6) repair little (18 %/18 %) because the reference index inherits the same coverage. Deterministic extraction (A3) is the strongest cheap method: blind centrality selection keeps 38 %, and with the question (BM25) 97 %/100 %—indistinguishable from full text at 1/15 of the bytes. Head/tail truncation (A2) keeps exactly the facts that fall in the kept windows (20 %). On the executable family the ordering changes: blind extractive selection keeps the whole specification (100 %/98 %) because the rules are dense in digits and identifiers, and the Llama summary preserves the rules better than the Qwen summary (82 vs 38 %). LLMLingua-2 (A8) at the same rate removes the very tokens the questions ask for (14 % on private facts, 0 on code) while being competitive on QASPER (17 vs 35 % F1 for full text): a compressor trained on meeting transcripts is tuned for paraphrasable content, not for exact values.

Cost accounting changes the ranking (RQ3). Table 6 gives, for the synthetic family, the median serialized bytes and the cold and warm per-query latencies assembled from the recorded spans. In bytes, every text method beats every cache method by 10^3–10^4. In cold latency on the laptop (Qwen pair), full text costs 951 ms per question (prefill of the whole document each time), the native cache 1,761 ms (document prefill once plus the fresh question), and a blind summary 7,450 ms—the sender's generation dominates. In the warm, reused regime the ranking inverts: the cache answers a new question in 187 ms and the 8-bit cache in 174 ms, the summary in 127 ms (it is short), while full text still pays 951 ms. Peak memory follows bytes: the dense cache of an 8k document adds $\approx$0.3 GB (Qwen) to $\approx$0.9 GB (Llama, eight KV heads) to the receiver's footprint, 8-bit packing halves it, and text methods add nothing beyond the prompt. No single denominator ranks the methods the same way; the frontier maps (Figures 1 and 2) are the result.

6.2. Two-node reuse and locality (E2/E6)

Figure 4 and Table 7 report genuine two-node measurements: the sender node (M3U96a) prefills a test document with Qwen2.5-3B, serializes the cache with its manifest, optionally quantizes or

Table 6. Resource accounting, synthetic family (medians over trials). Cold = build + compress/serialize + load + receiver; warm = load + receiver (document state already built); bytes = serialized payload including manifest. Qwen pair on the laptop (M5 Max); Llama pair on M3U96a (M3 Ultra) for this family—compare within a pair only.

	Qwen2.5-3B (laptop)			Llama-3.2-3B (M3U96a)		
Condition	bytes	cold ms	warm ms	bytes	cold ms	warm ms
A1 full text	29,208	951	951	29,208	2,862	2,862
A4 summary (blind)	1,424	7,450	127	1,838	18,977	444
A5 structured (blind)	827	8,157	83	842	18,999	344
A7 zstd text	2,726	1,032	1,032	2,726	2,857	2,857
B0 native cache	215,559,973	1,761	187	651,268,854	3,254	393
B1 serialized reuse	215,560,001	1,249	211	651,268,882	3,352	505
B2 8-bit	114,541,977	1,376	174	346,006,984	3,147	358
B3 sink+recent 1/4	53,892,924	1,106	122	162,812,727	3,009	278
B5 attention 1/4 (aware)	53,892,964	1,236	136	162,812,766	3,260	292
B6 pyramid 1/4 (aware)	53,892,854	1,639	540	162,812,673	3,923	962

Table 7. Two-node transfer (medians of 18 runs; ms unless stated). "Send" is the sender-side socket time; "load" is manifest check plus deserialization at the worker; the round trip includes the worker's fresh-query prefill and 32 decoded tokens. 4-bit rows transfer a cache that yields quality 0 for Qwen2.5 (Section 6.1) and are shown only for byte and time accounting.

Context class	Representation	payload (MB)	send	load	remote prefill	round trip
2k	text → remote prefill	0.01	–	–	1,160	1,567
2k	B1 bf16 cache	72	583	14	0	1,141
2k	B2 8-bit cache	38	293	10	0	1,052
8k	text → remote prefill	0.03	–	–	3,241	3,667
8k	B1 bf16 cache	215	1,806	40	0	2,667
8k	B2 8-bit cache	115	942	25	0	1,676
16k (≈11k tokens)	text → remote prefill	0.06	–	–	7,043	7,553
16k	B1 bf16 cache	406	3,434	73	0	4,453
16k	B2 8-bit cache	216	1,804	43	0	2,682

compresses it, and sends it over the LAN to a receiver worker (M3U96b) that checks the manifest, installs the cache, prefills only the fresh question and decodes the answer; the alternative sends the token ids and lets the worker prefill the whole document. Each point is the median of 18 runs (6 test documents $\times$ 3 repeats, after one warm-up on both ends); the effective transport rate was 0.92 Gb/s on a link that iperf3 measured at 0.94 Gb/s.

Three observations follow. First, the serialized bf16 cache is three to four orders of magnitude larger than the text it replaces (215 MB versus 29 kB at 8k tokens), and zstd recovers only ≈20 % of it (high-entropy payload; H8), while packed 8-bit quantization halves it. Second, on this link and with this receiver, moving the cache is nevertheless *faster* than re-reading the text at every context length we measured: at 8k tokens the bf16 transfer answers in 2,667 ms against 3,667 ms for remote prefill, and the 8-bit cache in 1,676 ms, because the M3 Ultra receiver prefills Qwen2.5-3B at ≈0.44 ms per token (3,241 ms for 8k) whereas 1 GbE moves the per-token cache (≈26 kB bf16, ≈14 kB at 8 bits) in ≈0.23 and ≈0.12 ms. The break-even therefore depends on the ratio of receiver prefill time to link time per token, which the right panel makes explicit: a faster receiver (the laptop

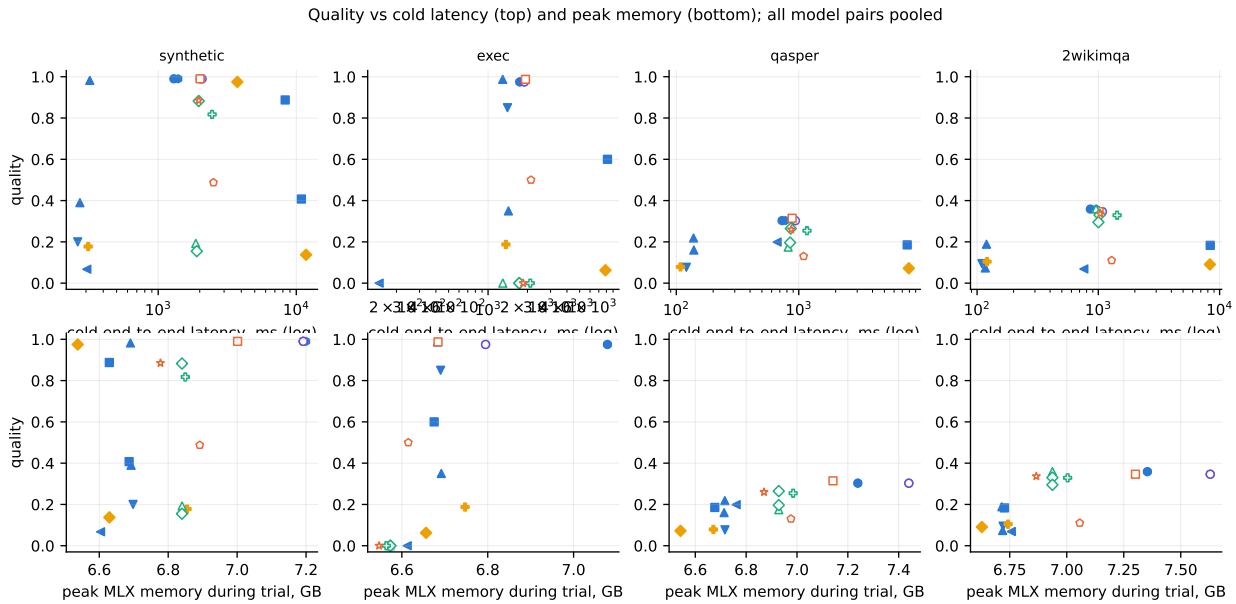


Figure 2. Quality against cold end-to-end latency (top) and peak MLX memory during the trial (bottom), by family, both pairs pooled.

prefills 8k tokens in ≈1 s) or a slower link would move the crossing to longer contexts, and a 10 Gb/s link would make transfer dominant everywhere. Third, the cost of building the cache is paid by the sender and is counted once per document: when the cache is reused for N queries at the receiver, the per-query cost is the fresh-query prefill plus decoding (≈40 ms load and ≈190 ms to the first token at 8k), while the text alternative pays the full prefill on each cold query unless the receiver keeps its own cache—at which point the two strategies converge and the question becomes where to schedule the work. Equation (5) with the measured components gives the break-even reuse count for any bandwidth; these are measured points on one link, not a universal bound.

6.3. Hosted receivers (Track D)

Table 8 reports the same text messages, built by the local Qwen2.5-3B sender, answered by two hosted models through their documented APIs on the first eight test documents of three families (5 questions per synthetic document; 672 trials; no failures; estimated spend 0.53 USD at the published list prices). Both hosted receivers answer every private-fact question correctly from the full document (A1), confirming that the synthetic facts are recoverable from the text; from the 512-token blind summary (A4) they recover only 28–30 % of the answers, and from the structured JSON (A5) 12.5 %—the same ordering, and nearly the same levels, as the local 3B receivers (Section 6.1). The bottleneck is therefore the sender's lossy message, not the receiver's capacity: a frontier receiver cannot recover a number that the message does not contain, and the reference-plus-retrieval variant (A6) does not repair it under the 2 000-byte retrieval budget. On QASPER the hosted models score

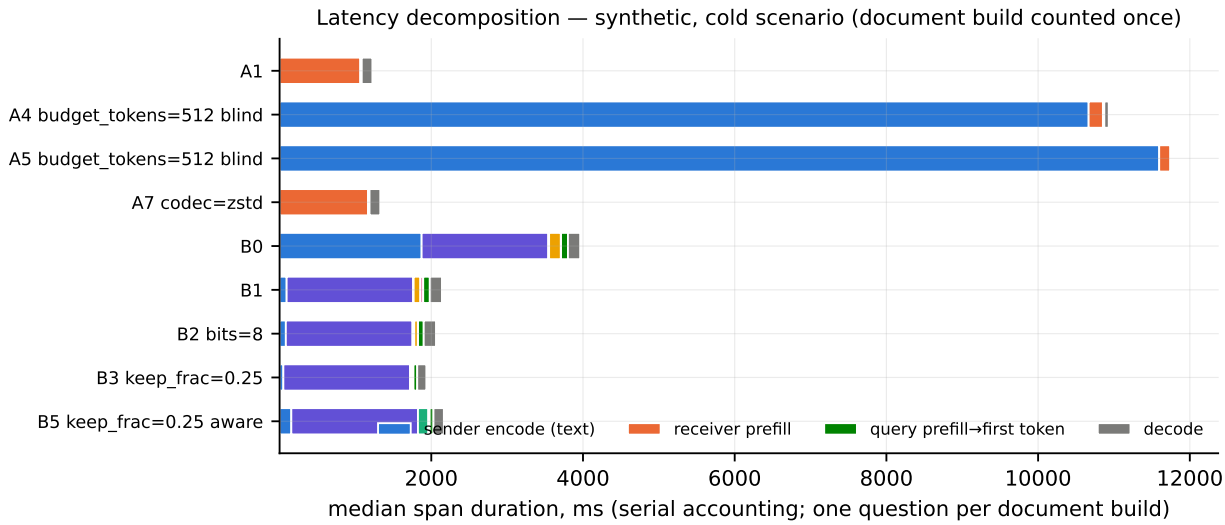


Figure 3. Latency decomposition (medians of recorded spans, synthetic family, cold scenario; document build counted once). Sender generation dominates the text methods that need a model; document prefill dominates the cache methods; selection and serialization are minor; the pyramid cache pays a token-by-token query prefill.

higher F1 than the local receivers from the same material (0.48 versus 0.35 for A1), i.e., receiver capacity matters when the material is complete.

Provider prefix caching is a billing object with a minimum size. With explicit cache breakpoints, Anthropic reported `cache_read_input_tokens` $\approx 5\,000$ on the second and later questions of the 8k-token documents and the per-query estimated cost fell from 6.3 to 2.2 m$, but *nothing* was cached for the 2k-token documents: the documented minimum cacheable prefix for this model is 4 096 tokens, so a 2 000-token document never qualifies. OpenAI's automatic caching reported cached tokens on 2k and 8k documents alike. Measured TTFT did not change with caching at this size (medians 0.72 versus 0.75 s on the 8k documents). None of these fields reveals cache bytes, precision or residency; they are reported as usage accounting only.

7. Adaptive representation and placement (E8)

The router uses only deployment-observable features—task family, receiver context length (three bins), model pair and question-access regime—and a table fitted on the development pilot: for each cell and each candidate action (fifteen static conditions spanning text, evidence and cache representations) it stores the development mean quality, its lower 95 % bootstrap bound and the mean amortized latency; it picks the cheapest action whose lower bound is within $\tau = 0.9$ of the best lower bound in the cell, and falls back to the development-best static action when a cell is missing

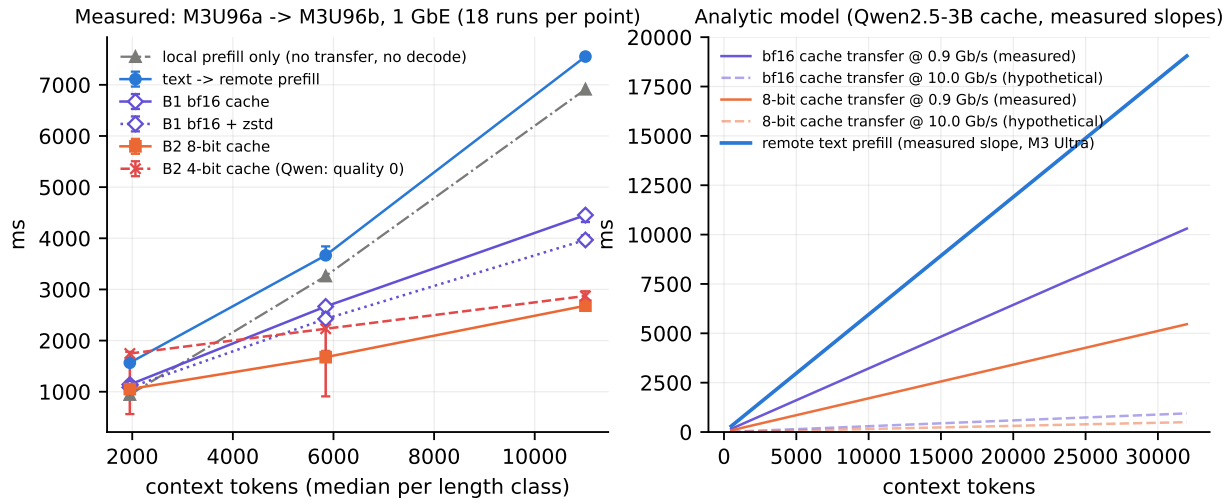


Figure 4. Reuse/network break-even. Left: measured round trip to a remote answer (medians, p25–p75 bars) for moving the native bf16 cache (B1), the zstd-compressed cache, the 8-bit and 4-bit packed caches (B2), versus sending the text and prefilling remotely; the grey line is the sender's own local prefill for scale. Right: analytic transfer time from the measured bandwidth (solid) and a hypothetical 10 Gb/s link (dashed) against the measured remote prefill slope. Qwen2.5-3B; synthetic test documents; M3U96a → M3U96b.

(Appendix E). The table was frozen before any test item was evaluated; on test, the chosen action's outcome is read from the full-factorial core records of that instance (an off-policy evaluation that is exact because every action was run on every instance), and the router's own lookup time (≈1 ms) is added. Figure 5 and Table 9 compare it with the development-best static action, full context, the native cache, the question-aware attention cache and the infeasible oracle that picks the best action per instance.

In the question-blind regime the router matches the quality of the development-best static policy (paired difference -1.4 points, 95 % CI [-3.0, +0.2], $p = 0.081$) at half its amortized latency (-934 ms, CI [-1,157, -697]): it learned on development data that the native cache (B0) and the 8-bit cache are as accurate as full text for the private-fact and executable families and cheaper once the document is reused, and that nothing beats full text on the public QA subsets. In the question-aware regime the router does not help: its quality equals the development-best static action (-0.3 points, CI [-2.6, +2.2]) but its cost is *higher* (+1,512 ms), because in several cells it selects attention-selected caches whose per-query compression cost on the test documents exceeded the development estimate, and because the cheap aware extractive baseline (A3) was already near the oracle. The fallback rate was 23.8 % (context bins unseen on dev). Taken together, H9 is confirmed for the aware regime and rejected for the blind regime, with the caveat that the gain is a latency gain at equal quality and that it depends on reuse: at $N = 1$ the cache build cost is not amortized and full text would be preferred.

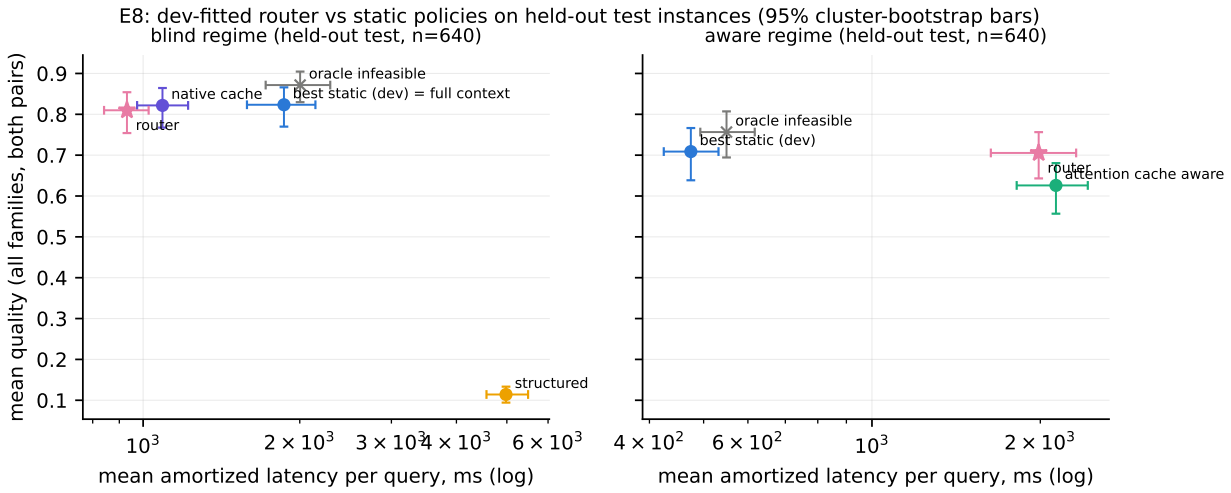


Figure 5. Dev-fitted router against static policies on held-out test instances (both pairs, all families; $n = 640$ per regime). The oracle is an infeasible upper reference.

8. Failure analysis and limitations

Contextual, not parametric. Every effect reported here disappears when the message or cache is removed (A0). No receiver acquired knowledge; frozen models were given better or worse *access* to information. Claims about "learning" are out of scope.

Local white-box versus hosted black-box. Cache experiments require a runtime that exposes attention tensors (MLX here). Hosted APIs expose a billing abstraction (cached input tokens) whose precision, layout and residency are undisclosed; Track D therefore contains text methods only and its cached-token fields are never translated into bytes.

Same checkpoint versus cross-family. All cache reuse and transfer results concern identical checkpoints and revisions (Track B, two-node B1) or a same-family size ladder with a linear mapper (Track C). Cross-family cache transfer was not executed: the only published method with code (SwaT) released no checkpoint and no licence, and training a mapper of its size exceeded our budget. The paper's title names knowledge transfer in general; its *cache*-transfer evidence is same-family only.

Query-aware versus reusable compression. Attention-based selection is query-dependent; its question-blind variant (generic window) is a different method with much lower quality (Section 6). Reusable (blind) messages and query-specific (aware) messages are reported in separate panels and should not be ranked against each other.

Translated-cache distribution shift. The mapper was fitted on native sender caches; chains of mapped caches (feeding a mapped cache into a further mapping) were not run, so no closure property is claimed.

Budgets are measured but not universal. Byte counts include manifests and quantization metadata; latencies include serialization, loading and fresh-query processing; memory peaks are MLX allocator peaks. They are specific to `mlx-lm` 0.32.2 kernels on one M5 Max and two M3 Ultra machines; a fused eviction kernel or a different attention backend would change the compression and TTFT components. We compared methods on a common backend; we did not compare against vendor-optimized kernels.

Network and reuse. The two-node link is a 1 GbE LAN with 1.2 ms RTT; break-even positions scale with bandwidth and with the receiver's prefill speed, both of which we report so that the analytic curves can be re-drawn. Queue delays were not imposed; the cluster nodes carried a light concurrent CPU workload from another study.

Contamination and synthetic facts. The public QA texts may appear in pre-training corpora; the communication-dependence conclusions rest on the synthetic private-fact family, whose facts are new by construction but whose templates are narrower than real documents. The generated ledger is not a filing; the executable tasks are small.

Scoring. All scoring is deterministic (exact/numeric/yes-no/abstention/F1/hidden tests); no language-model judge and no human annotation were used. The fact-ledger retention metric is string containment of normalized values, which under-counts paraphrased retention and over-counts accidental matches of short values.

Coverage. Two 3B model families, three task families, one sender budget (512 tokens) in the core, one selection ratio (1/4) for the registered comparisons. Audited but unexecuted methods are listed with reasons in Table 2 and Appendix B; their absence is a coverage gap, not evidence about them.

Auditability and security of latent messages. A cache is not human-readable; our manifests bind it to a checkpoint revision, tokenizer, chat template, prefix-token hash and document id, and the loader rejects mismatches, which protects against the wrong-document handle. Structured evidence with spans is auditable but depends on the integrity of the source it points to; retrieved passages are treated as data, never as instructions, and the public benchmark text never enters the operator's command channel.

9. Reproducibility and execution disclosure

How the study was executed. All code, experiments, analyses and this manuscript were produced by Claude Code (Anthropic; model `claude-fable-5-1`) acting as research operator under a written brief from the author, through actual terminal commands on the author's machines: environment inspection, dependency pinning, implementation, launching inference, reading exit codes and logs, diagnosing failures, re-running affected trials, analysis and compilation. The author provided the brief, the hardware, API credentials already present in sibling projects, and no mid-run intervention; dated engineering decisions are in `DECISIONS.md`. Models under test are the experimental subjects; the operator's own inference state was never an experimental interface. This disclosure describes the process; it is not a claim that autonomous research is superior to any alternative.

Defects found and handled. (i) The `mlx-lm` cache API changed between the system-installed version (0.31.3) and the pinned environment (0.32.2); adapters were rewritten for the pinned API before any experiment. (ii) The receiver prefix boundary was initially computed from the prefix text alone and did not match the tokenization of the full prompt; it is now the common token prefix with a canonical question, and a mismatch raises. (iii) Heterogeneous retained lengths (B6) broke the shared attention mask; the fresh query is processed token by token for such caches. (iv) The first executable-task specification (seven rules) was unsolvable for 3B receivers even with the full ticket; it was simplified on development data before any test item was run (D-010). (v) Running three GPU jobs concurrently exhausted unified memory and slowed the pilot by an order of magnitude; GPU jobs were serialized.

Artifacts. The repository contains the harness, pinned lock file, data manifests and generators, the frozen protocol and its hash, every per-attempt record (`runs/raw`), sanitized logs, the mapper checkpoint and manifest, generated tables and figures with a provenance index linking each to its source records and script, a claim ledger, and the minimal reproduction profile. Checkpoints are not redistributed; their Hugging Face identifiers, revisions and weight hashes are recorded. Hosted-API trials require the provider credentials and reproduce only up to provider nondeterminism.

Reproduction commands.

```
make reproduce-analysis                          # tables + figures from runs/raw, provenance audit
python -m wp30 reproduce --profile reproduction/minimal.yaml   # re-run a fixed subset, compare greedy o
make paper                                       # compile this document
```

Greedy decoding on MLX is deterministic for an identical prefill chunking but not bitwise across chunkings (Section 5.3); the reproduction report lists identical output hashes and identical scores separately.

10. Conclusion

To be written from results.

References

Alabdulkareem, Abdulrahman, Isabel Tenison, Jun Zhang, Yiyang Zhu, Boris Katz, and Dan Gutfreund, "SwaT: Cross-Family KV Cache Transfer between Language Models," 2026. arXiv preprint, v1 (October 2026); repository Ar-Kareem/SwaT, no released checkpoint or licence (verified 2026-10-29).

Anthropic, "Prompt caching," https://platform.claude.com/docs/en/build-with-claude/prompt-caching 2026. accessed 2026-10-29.

Apple ML Explore contributors, "mlx-lm: Run LLMs with MLX," https://github.com/ml-explore/mlx-lm 2026. v0.32.2 (16 October 2026), MIT.

Bai, Yushi, 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)" 2024, pp. 3119–3137.

Cai, Zefan, 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," 2024. v4 (May 2025); code Zefan-Cai/KVCache-Factory, MIT.

Dasigi, Pradeep, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner, "A Dataset of Information-Seeking Questions and Answers Anchored in Research Papers," 2021. NAACL 2021; QASPER, CC BY 4.0.

Fu, Tianyu et al., "Cache-to-Cache: Direct Semantic Communication Between Large Language Models," in "International Conference on Learning Representations (ICLR)" 2026. arXiv:2510.03215; code thu-nics/C2C, Apache-2.0; fuser checkpoints nics-efc/C2C_Fuser.

Ho, Xanh, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa, "Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps," in "Proceedings of the 28th International Conference on Computational Linguistics (COLING)" 2020.

Hsieh, Cheng-Ping, 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. arXiv:2404.06654.

Hugging Face, "KV cache strategies," https://huggingface.co/docs/transformers/en/kv_cache 2026. Transformers v5 documentation, accessed 2026-10-29.

Li, Yuhong, 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 "Advances in Neural Information Processing Systems (NeurIPS)" 2024. arXiv:2404.14469; code FasterDecoding/SnapKV.

Liu, Zirui, 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," in "International Conference on Machine Learning (ICML)" 2024. arXiv:2402.02750.

Pan, Zhuoshi, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang, "LLMLingua-2: Data Distillation for Efficient and Faithful Task-Agnostic Prompt Compression," in "Findings of the Association for Computational Linguistics: ACL 2024" 2024. arXiv:2403.12968.

Xiao, Guangxuan, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis, "Efficient Streaming Language Models with Attention Sinks," in "International Conference on Learning Representations (ICLR)" 2024. arXiv:2309.17453.

Zhang, Zhenyu, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen, "H_2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models," in "Advances in Neural Information Processing Systems (NeurIPS)" 2023. arXiv:2306.14048.

Zou, Jiaru et al., "LatentMAS: Latent Collaboration in Multi-Agent Systems," in "International Conference on Machine Learning (ICML)" 2026. arXiv:2511.20639; code GenVerse/LatentMAS, Apache-2.0.

Zweiger, Adam, Yilun Fu, Han Guo, and Yoon Kim, "Fast KV Compaction via Attention Matching," 2026. arXiv preprint, v2 (May 2026); code adamzweiger/compaction, MIT.

Appendices

A. Registered hypotheses

```
# WP30 -- Research questions and falsifiable hypotheses (registered before any held-out outcome was exam

Registered: 2026-10-29 (Phase 1). Amendments only as dated entries at the bottom.
Test-split outcomes had **not** been examined at registration (no test split existed yet; only the E0 sm
hand-written 3-sentence memo had been executed, see `runs/logs/`).

## Research questions

- **RQ1 (compression frontier)** How do text shortening (A2-A4, A8), packed KV quantization (B2), token
  learned compaction (B7, if run) trade task quality against *measured* serialized bytes and resident ca
- **RQ2 (transfer compatibility)** How much performance is preserved by same-checkpoint reuse (B1, cross
  versus learned mappings between distinct checkpoints (Track C), relative to the receiver's own full-co
- **RQ3 (resource accounting)** Do rankings of methods under a token budget change when the budget is by
  peak memory or amortized cost over N reuses?
- **RQ4 (repeated handoffs)** How do correct-fact retention, unsupported additions and task accuracy evo
  for prose summaries vs structured evidence, in closed vs repairable chains?
- **RQ5 (evidence)** Under an equal byte budget, does structured evidence with source spans (A5) retain
  numbers, dates, negations) better than an abstractive summary (A4)? Does metered retrieval (A6) repair
- **RQ6 (locality)** When is running the next query where the cache resides cheaper than moving the cach
  as a function of context length, payload bytes, measured bandwidth and queue delay?
- **RQ7 (adaptive policy)** Can a router fitted on development data improve the held-out quality/cost fr
  development-selected static method, once routing overhead and fallbacks are counted?

## Hypotheses (each may be rejected)

- **H1** Lossy compression degrades exact identifiers, decimal numbers, negation and temporal order more
  on the synthetic private-fact family, the accuracy drop from A1 to A4 at a 512-token budget is larger
  for `exact_copy`, `negation`, and `revision` question types than for `single_fact` questions.
- **H2** Packed KV quantization is model-dependent: at 4 bits/group 64, Llama-3.2-3B retains >= 95 % of
  Qwen2.5-3B loses >= 20 percentage points (motivated by E0 smoke: Qwen2.5-1.5B produced garbage at 4-bi
  mlx_lm path and ours; Llama-3.2-3B did not). 8-bit retains >= 97 % for both.
- **H3** Sink+recent eviction (B3) at <= 1/4 retention is worse than a *question-aware* attention-inform
  at equal retained positions, but **not** reliably better than random selection (B4) on questions whose
  middle of the document.
- **H4** Serialized cache bytes exceed raw UTF-8 document bytes by >= 2 orders of magnitude for 3B model
  wins on total latency when (a) the receiver's prefill cost exceeds transport time, which on the measur
  above a break-even length we will estimate (prediction: several thousand tokens), or (b) the cache is
- **H5** Repeated prose handoffs lose facts monotonically; after 4 handoffs the retained-correct-fact ra
  70 % while A5 (structured, explicit unknowns) stays above 85 %; unsupported additions appear mainly in
```

```
- **H6** Metered retrieval (A6, repairable chains) restores most of the lost exact facts but adds fetche
  the message itself.
- **H7** Question-aware compression dominates question-blind compression at equal budget for *all* lossy
  largest for attention-based selection (which is query-dependent by construction).
- **H8** A lossless transport codec (A7, zstd) on text yields >= 2x byte savings with < 5 ms overhead bu
  latency or quality; on bf16 KV tensors the codec yields < 1.3x (high-entropy payload).
- **H9** The adaptive router (E8) will not reliably beat the best development-selected static method on
  overhead is included (we expect a null or small, uncertain gain).
- **H10** Cross-family cache transfer without a trained mapper is not possible; with a mapper trained un
  quality is below the receiver's own A4 summary baseline (we expect learned transfer to lose to text at

## Primary endpoints (frozen at Phase 7 together with the matrix)

Per task family: exact-match / F1 (LongBench normalization) or numeric correctness with tolerance; opera
serialized payload bytes; end-to-end wall time (defined boundaries, S6.1 of the brief); peak MLX memory.

## Primary confirmatory comparison family (Holm-corrected; everything else exploratory)

1. A5 vs A4 at matched byte budget (synthetic, question-blind) -- accuracy.
2. B2-8bit vs B0 -- accuracy (both models).
3. B2-4bit vs B0 -- accuracy (Qwen vs Llama interaction stated as H2).
4. B5-aware vs B3 at 1/4 retention -- accuracy.
5. A4 chain vs A5 chain at 4 handoffs -- fact retention.
6. Router vs best static -- held-out accuracy at equal or lower cost.

## Amendments

(none yet)
```

B. Source audit and unexecuted methods

Thirty-four sources were verified on 2026-10-29 from their arXiv pages, repositories and documentation (`literature/sources.csv`; notes in `literature/audit_group{1,2,3}.md`). Table 10 summarizes the core methods.

C. Correctness tests

`tests/test_cache_correctness.py` (12 tests, all passing on Qwen2.5-1.5B-Instruct-bf16): prefix equivalence in float32 and bf16 (distributional tolerance of Section 5.3, identical greedy continuation); no duplicated boundary token; identity round trip with exact logits and measured bytes

equal to (2); loader rejection of a wrong prefix hash and of a different checkpoint; 100 % retention exact (shared and per-head indices); sparse mask path versus token-by-token append; preserved versus relabelled positions differ; packed quantization bit widths and execution for 8/4/2 bits; no shared mutable state across branches; attention selection keeps the window; no-message control. `tests/test_leakage.py` (3) and `tests/test_scoring_and_data.py` (5) cover evaluator isolation, scorer behaviour and generator determinism.

D. Prompts

```
"""Prompt templates (frozen at Phase 7). Receiver prompts are identical across methods except for the me
SYSTEM_RECEIVER = ("You are a careful assistant. Answer the question using ONLY the provided material. "
                   "If the material does not contain the answer, reply exactly: unknown.")
SYSTEM_RECEIVER_NOMSG = ("You are a careful assistant. Answer the question. If you do not know, reply ex

def receiver_user_text(material_label: str, material: str, question: str, family: str) -> str:
    if family == "exec":
        return f"{material_label}:\\n{material}\\n\\nTask: {question}"
    fmt = "Give a short answer (a few words or a number), with no explanation."
    return f"{material_label}:\\n{material}\\n\\nQuestion: {question}\\n{fmt}"

def receiver_question_only(question: str, family: str) -> str:
    if family == "exec":
        return f"Task: {question}"
    return f"Question: {question}\\nGive a short answer (a few words or a number), with no explanation."

# Sender prompts (question-blind: no question shown; question-aware: question appended)
SUMMARY_SYSTEM = "You are a precise note-taker. You write faithful summaries that preserve exact names,
def summary_user(doc: str, budget_tokens: int, question: str | None) -> str:
    q = f"\\nThe reader will need to answer this question: {question}\\n" if question else "\\nThe reade
    return (f"Summarize the following document in at most {budget_tokens} tokens (about {int(budget_toke
            f"numbers, dates and names exactly as written. Mark explicitly what is NOT stated if relevan

EVIDENCE_SYSTEM = ("You extract structured evidence. Output ONLY a JSON object with a key \\"facts\\": a
                   "{\\"entity\\": str, \\"attribute\\": str, \\"value\\": str, \\"quote\\": str} where
                   "supporting the fact, and optionally a key \\"notes\\": a short string for anything i
                   "and a key \\"unknowns\\": list of things the document explicitly leaves unstated. Co
def evidence_user(doc: str, budget_tokens: int, question: str | None) -> str:
    q = f"\\nThe reader will need to answer this question: {question}\\n" if question else "\\nThe reade
    return f"Extract structured evidence from the document. Keep the JSON under {budget_tokens} tokens.
```

E. Pseudocode

Algorithm 1 Trial scheduler (resumable, append-only)

load config, manifest, resources; $done \leftarrow$ trial keys with status success in `runs/raw/<exp>.jsonl`
for document d in selected items **do**
  for model pair (S,R) **do**
    for condition c **do**
      for seed **do**
        $P \leftarrow$ questions of d whose trial key $\notin done$; **continue** if $P = \emptyset$
        if c is question-blind **then** build message/cache m once for d (reuse count $= |P|$)
        end if
        for question $q \in P$ **do** build m if question-aware; run receiver; score with evaluator-only key; append validated record
        end for
      end for
    end for
  end for
end for

Algorithm 2 Cache handoff pipeline (Track B/C)

pre $\leftarrow$ chat tokens of (system, document) that are a true prefix of the full (system, document, question) prompt
$C \leftarrow$ receiver.prefill(pre) ▷ dense bf16; measured doc prefill
$C' \leftarrow$ compress(C): quantize / select positions (sparse cache with logical offset $|\text{pre}|$) / map from sender
payload $\leftarrow$ serialize(C', manifest); optionally codec
at receiver: check manifest; deserialize; install; prefill fresh query tokens with retained-positions mask; decode greedily

Algorithm 3 Evidence retrieval (A6) and repairable chains

refs $\leftarrow$ A5 facts without values; rank refs by BM25(question, entity+attribute)
for ref in ranked order while lookups $<$ max and bytes $<$ budget **do** fetch source span $\pm$ context; count bytes and lookups
end for
chains: for codes mentioned in the previous message, fetch source sentences containing the code under the same meters

F. Per-attempt record schema

Each attempt is one JSON line validated against `wp30-attempt-1`: identifiers (study, experiment, trial, attempt, run group, protocol and config hashes, git commit, timestamp), task (family, document id and hash, question id, split, cluster), models (sender and receiver revisions, tokenizer hashes, backend versions, hardware id, device state, weight precision), method (id, fidelity, compression config with realized values, question access, topology, handoff index, reuse count, cache hit, compatibility hash, seed), sizes (source tokens under both tokenizers, query and message tokens,

Algorithm 4 Adaptive policy (E8)

1: on dev: for each (family, context bin, pair, access) and action a: $\hat{q}_a$, lower bootstrap bound $\underline{q}_a$, mean cost $\hat{c}_a$
2: feasible $\leftarrow \{a : \underline{q}_a \geq \tau \max_{a'} \underline{q}_{a'}\}$; choose $\arg\min_{a \in \text{feasible}} \hat{c}_a$; fallback to the dev-best static action
3: freeze; on test: look up the cell (nearest bin fallback counted), apply the chosen action, add router time

UTF-8 bytes, cache tensor bytes, serialized payload bytes, side information, fetched bytes, transport bytes), timing spans and TTFT, usage and provider cached usage, memory (process RSS, MLX active/peak, prefill and decode peaks, mapper memory), cost with source, prediction and evidence artifacts, key reference, scores, status, error code, retry link, log path, payload and output hashes.

G. Additional tables

Table 8. Hosted receivers on messages built by the local sender (test split, first 8 documents per family). Quality = accuracy (synthetic, exec) or F1 (QASPER) with 95 % cluster-bootstrap CI; cost = estimated per-query cost from provider price tables (m$ = 0.001 USD); TTFT = time to first streamed content token (Anthropic only; the OpenAI Responses call was not streamed). "+cache" = documented prompt caching (Anthropic `cache_control`; OpenAI automatic prefix caching). Provider sampling defaults (no temperature control, D-014).

model	condition	family	n	quality_ci	cost (m$)	TTFT p50 (ms)	total p50 (ms)
claude-haiku-4-5	A0	exec	8	0.00 [0.00, 0.00]	1	558	1,396
claude-haiku-4-5	A1	exec	8	1.00 [1.00, 1.00]	3	503	1,438
claude-haiku-4-5	A1 +cache	exec	8	1.00 [1.00, 1.00]	3	464	1,389
claude-haiku-4-5	A4 (512)	exec	8	0.62 [0.38, 0.88]	1	641	21,157
claude-haiku-4-5	A5 (512)	exec	8	0.50 [0.12, 0.88]	1	439	21,792
claude-haiku-4-5	A6 (512)	exec	8	0.25 [0.00, 0.50]	1	416	22,233
claude-haiku-4-5	A0	qasper	8	0.01 [0.00, 0.04]	0	608	721
claude-haiku-4-5	A1	qasper	8	0.48 [0.16, 0.80]	5	520	1,109
claude-haiku-4-5	A1 +cache	qasper	8	0.49 [0.15, 0.83]	6	709	1,168
claude-haiku-4-5	A4 (512)	qasper	8	0.07 [0.00, 0.15]	0	556	14,589
claude-haiku-4-5	A5 (512)	qasper	8	0.13 [0.00, 0.38]	0	790	25,202
claude-haiku-4-5	A6 (512)	qasper	8	0.02 [0.00, 0.06]	0	821	25,476
claude-haiku-4-5	A0	synthetic	40	0.00 [0.00, 0.00]	0	597	630
claude-haiku-4-5	A1	synthetic	40	1.00 [1.00, 1.00]	4	719	817
claude-haiku-4-5	A1 +cache	synthetic	40	1.00 [1.00, 1.00]	2	748	856
claude-haiku-4-5	A4 (512)	synthetic	40	0.28 [0.10, 0.45]	1	558	22,930
claude-haiku-4-5	A5 (512)	synthetic	40	0.12 [0.05, 0.17]	0	571	25,272
claude-haiku-4-5	A6 (512)	synthetic	40	0.12 [0.05, 0.17]	1	613	25,775
gpt-5-nano	A0	exec	8	0.00 [0.00, 0.00]	0	–	2,447
gpt-5-nano	A1	exec	8	1.00 [1.00, 1.00]	0	–	2,202
gpt-5-nano	A1 +cache	exec	8	1.00 [1.00, 1.00]	0	–	2,201
gpt-5-nano	A4 (512)	exec	8	0.38 [0.12, 0.75]	0	–	21,920
gpt-5-nano	A5 (512)	exec	8	0.25 [0.00, 0.62]	0	–	22,180
gpt-5-nano	A6 (512)	exec	8	0.12 [0.00, 0.38]	0	–	23,295
gpt-5-nano	A0	qasper	8	0.00 [0.00, 0.00]	0	–	1,881
gpt-5-nano	A1	qasper	8	0.38 [0.12, 0.67]	0	–	1,202
gpt-5-nano	A1 +cache	qasper	8	0.37 [0.12, 0.66]	0	–	1,273
gpt-5-nano	A4 (512)	qasper	8	0.22 [0.04, 0.46]	0	–	14,291
gpt-5-nano	A5 (512)	qasper	8	0.04 [0.00, 0.11]	0	–	25,297
gpt-5-nano	A6 (512)	qasper	8	0.02 [0.00, 0.05]	0	–	25,710
gpt-5-nano	A0	synthetic	40	0.00 [0.00, 0.00]	0	–	1,148
gpt-5-nano	A1	synthetic	40	1.00 [1.00, 1.00]	0	–	1,370
gpt-5-nano	A1 +cache	synthetic	40	1.00 [1.00, 1.00]	0	–	1,072
gpt-5-nano	A4 (512)	synthetic	40	0.30 [0.15, 0.45]	0	–	22,532
gpt-5-nano	A5 (512)	synthetic	40	0.12 [0.05, 0.17]	0	–	26,477
gpt-5-nano	A6 (512)	synthetic	40	0.12 [0.05, 0.17]	0	–	26,536

Table 9. Router evaluation on held-out test instances. Quality in percent with 95 % cluster-bootstrap CI; cost = mean amortized latency per query (ms, document build divided by the number of reusing questions); fallback = share of instances routed by the nearest-cell fallback.

Regime	Policy	quality (%)	95 % CI	cost (ms)	fallback (%)
blind	router	81.0	[75.4, 85.4]	931	23.8
blind	best static (dev) = full context (A1)	82.3	[77.0, 86.6]	1,865	0
blind	native cache (B0)	82.2	[76.7, 86.4]	1,089	0
blind	structured evidence (A5)	11.4	[9.4, 13.3]	4,981	0
blind	oracle (infeasible)	87.2	[83.0, 90.5]	2,003	0
aware	router	70.5	[64.3, 75.6]	1,987	23.8
aware	best static (dev) = extractive (A3-aware)	70.9	[63.9, 76.6]	474	0
aware	attention cache (B5-aware)	62.6	[55.7, 68.1]	2,133	0
aware	oracle (infeasible)	75.7	[69.4, 80.7]	549	0

Table 10. Audited core methods: availability of code and checkpoints, backend, question access, and WP30 status.

Method	Code	Checkpoint	Backend	Question access	WP30
SwaT (2610.15095)	yes (no licence)	none	torch/CUDA	blind	not run
Attention Matching (2602.16284)	MIT	n/a (optimization)	vLLM 0.11 / torch	blind	not run (MLX port out of scope)
C2C (2510.03215)	Apache-2.0	Qwen3 fusers	torch/CUDA	blind	not run
LatentMAS (2511.20639)	Apache-2.0	n/a	HF/vLLM, CUDA	blind	not run
KIVI (2402.02750)	MIT	n/a	CUDA kernels	n/a	motivation tested via B2
SnapKV (2404.14469)	Apache-2.0	n/a	torch	aware	faithful re-implementation (B5)
PyramidKV (2406.02069)	MIT	n/a	torch	aware	budget rule re-implemented (B6)
H_2O (2306.14048)	no LICENSE file	n/a	torch	cumulative	not run
StreamingLLM (2309.17453)	MIT	n/a	torch	blind	retention pattern only (B3)
LLMLingua-2 (2403.12968)	MIT	yes (HF)	torch (MPS ok)	blind	official (A8)