# NO PASSAGE IS AN ISLAND: ORPHANED KV CACHES, PHANTOM PEAKS, AND WARM ENCODING FOR RETRIEVAL-AUGMENTED GENERATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Retrieval-Augmented Generation improves Large Language Models by enabling document-grounded generation, however it requires recomputing the per-passage prefill at each retrieval time. As such, the systems cache each passage's key–value (KV) states once and assemble the retrieved ones at query time. Current methods reduce this overhead by reusing the per-passage KV cache of prior prefills, so that generation can skip the prefill of retrieved passages. The assembled cache, however, reduces prediction quality, and prior systems repair it online by recomputing roughly a quarter of the tokens on every query. In this work, we take a different approach. We identify where the error comes from and remove most of it offline. We find that position is not the problem, but rather that a passage encoded alone is encoded cold, meaning that its keys are wrong in the low-frequency rotary bands, exactly where the question attends. The question's attention therefore collapses and lands on phantom peaks, tokens that the mismatched question favours and the fresh prefill does not. We introduce Raftel, which works by giving the phantom peaks their true keys, so the tokens to recompute are the ones the mismatched question attends, not the ones the true question would. We therefore warm the passages instead of repairing the cache. Each passage is encoded once against the stored caches of its most similar passages and iterated twice, which removes most phantom peaks before any question arrives. We evaluate our method on LongBench and RULER and find that the bare warm cache matches the prior state of the art's best operating point at $6.6\times$ the speed of full prefill.

## 1 INTRODUCTION

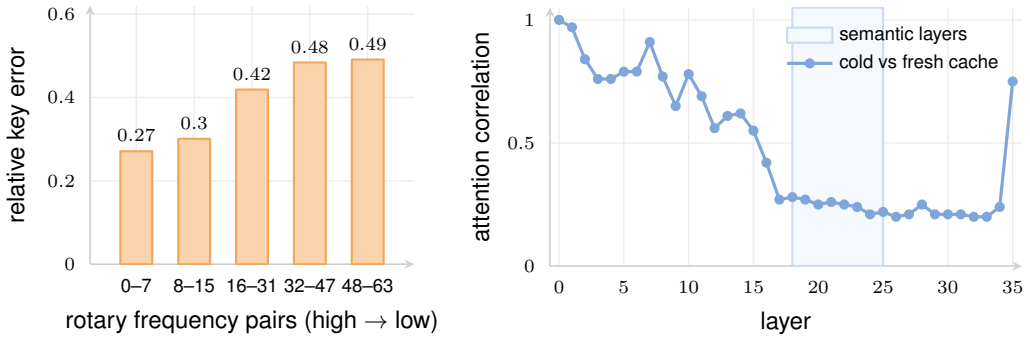
In Retrieval-Augmented Generation (RAG) (Lewis et al., 2020), the same passages are prefilled on every query, thus repeated prefill dominates the time to first token (TTFT) at long context. Caching each passage's KV states once and assembling the retrieved KV passages at query time removes that prefill overhead. However, a passage encoded on its own does not contain the KV entries the model would produce inside the recomputed context, leading to inaccurate responses. The resulting accuracy gap is well known and varies widely with model and data, from a few points to a total collapse.

Existing solutions (Yao et al., 2024; Wang et al., 2026; Yan et al., 2026; Liu et al., 2026a) repair the assembled cache online by recomputing a selected fraction of the tokens. This introduces three major problems. The repair is paid on every query. Currently, the recompute budget cannot be reduced without losing accuracy. The selection rules are heuristics with no account of what they actually repair, so the question-driven rules differ significantly and offer little interpretability. Trained alternatives (Yang et al., 2025) fine-tune the model or add trained link tokens. These approaches require retraining for each model and prior evaluations show limited transfer across architectures.

Raftel reduces the cache mismatch before online repair. Each passage is encoded with related context available before any question arrives. The analysis identifies a mechanism behind this mismatch. We call the per-passage KV states encoded alone a *cold* cache and the KV states from prefilling the full prompt at once a *fresh* cache. The key error of a cold cache concentrates in the low-frequency rotary bands, while the question's attention diverges from fresh prefill in the middle-to-late layers. The question's attention therefore collapses onto *phantom peaks*, tokens that the fresh cache ignores.



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.




(a) Relative error by frequency pair of cold vs fresh keys at the semantic layer.

(b) Per-layer correlation between the question's attention on the cold and the fresh cache.

Figure 1: Observation 1 and Observation 2. (a) The key error is largest in the low-frequency pairs. (b) The question's attention diverges from the fresh cache before and across the layer band used for selection.

The analysis also yields a rule for any system of this kind. Recompute the tokens the mismatched question attends, not the ones the true question would. In Raftel, we produce *warm* passages that remove most phantom peaks offline. A question-attention selection at the layers where attention collapses covers the rest online.

To warm a passage, Raftel encodes each passage offline against the stored caches of a small set of similar passages, chosen by passage-to-passage overlap or by adjacency in its document, without any query. Raftel iterates this for a few rounds, each time forwarding only the passage's own tokens. Online, the retrieved warm caches are relocated and concatenated, the question is run once over them, a small set of document tokens the question attends most at the profiled layers is recomputed at full depth, and decoding starts. Under the benchmark neighbourhood protocol, the bare warm cache matches the best published operating point at a fraction of its TTFT. Moreover, with a small recompute, its accuracy is not statistically distinguishable from fresh prefill in the main paired comparison.

Our contributions are as follows.

1. **A diagnosis of reused caches.** We call the keys of a passage encoded alone an *orphaned cache* and show that their error concentrates in the low-frequency rotary bands while question attention diverges in the middle-to-late (semantic) layers. The question's attention therefore collapses onto phantom peaks, tokens the fresh cache ignores. Rotary relocation is exact, so placement is not the cause of the error.
2. **An objective for online repair.** Recompute the tokens the mismatched question attends, rather than the ones the true question needs. The restoration should not aim to reproduce the fresh cache's attention pattern. The best repairs lower that correlation while raising accuracy. This account helps explain the ordering and budget dependence of the published selection rules and why removal or down-weighting without correcting the retained values can be insufficient.
3. **A new method, Raftel.** A training-free, offline encoding of each passage against the stored caches of its query-blind neighbours, iterated for a few rounds once per corpus chunk, with storage and the online path unchanged.
4. **An extensive evaluation.** Against fresh prefill, the cold cache, and the selection rules of CacheBlend, ProphetKV, and QCFuse, on three multi-hop benchmarks with isolated TTFT. Under the benchmark neighbourhood protocol, the bare warm cache matches the best prior operating point at $6.6\times$ the speed of fresh prefill. A 15% recompute is within noise at $2.1\times$ the latency.
5. **An ablation study.** We evaluate how relatedness, neighbourhood size, warm-encoding rounds, corpus source, and same-document adjacency affect the warm cache.

## 2 Analysis: Orphaned Caches and Phantom Peaks

We state three problems behind the accuracy loss of a reused cache and the consequence each implies for any repair. A *fresh* cache is the cache obtained from prefilling the whole prompt. A *cold* cache is the cache assembled from per-passage KV states, then placed at the passage's position in the prompt. Prior work measures the accuracy loss of reusing a cold cache against a fresh prefill. We use Qwen3-8B on HotpotQA throughout this section. The problem is most visible in this model, making it a good example for the visualizations. Section 4.3 shows the same pattern on additional models.

**Problem 1. A passage cached alone provides the wrong keys.**

**Observation 1** (Orphaned keys)**.** *The keys of a passage encoded alone differ from the full prompt key version. We notice that the difference is largest in low-frequency rotary pairs, which consequently are the relevant pairs for the query attention.*

A transformer (Vaswani et al., 2017) computes the key of a token autoregressively, from everything before it. Thus, a passage encoded alone lacks information from the preceding context. As rotary position embeddings are composable, relocation of a key to the correct position is exact. The error is therefore a context-encoding error, not a placement error. Rotary embeddings split every key at position $p$ into $d/2$ frequency pairs. Pair $i$ is rotated by its angle $\theta_i p$, so the key is

$$k_p = \big(R(\theta_1 p)\, k^{(1)},\, \ldots,\, R(\theta_{d/2} p)\, k^{(d/2)}\big),$$

where $R(\theta)$ is the $2{\times}2$ rotation matrix. The per-pair error between the cold and fresh keys is $\Delta k^{(i)} = k^{(i)}_{\text{cold}} - k^{(i)}_{\text{fresh}}$. Figure 1a shows that $\|\Delta k^{(i)}\|$ grows as $\theta_i$ decreases, concentrating the error in the low-frequency pairs. Moreover, we can decompose the dot product of query and keys by frequency pairs

$$q^\top k = \sum_i {q^{(i)}}^\top R(\theta_i p)\, k^{(i)}\,.$$

The measured error is largest in the low-frequency pairs where $\|\Delta k^{(i)}\|$ is large. The softmax amplifies these dot-product errors, shifting the attention mass.

**Problem 2. The question attends to the wrong tokens.**

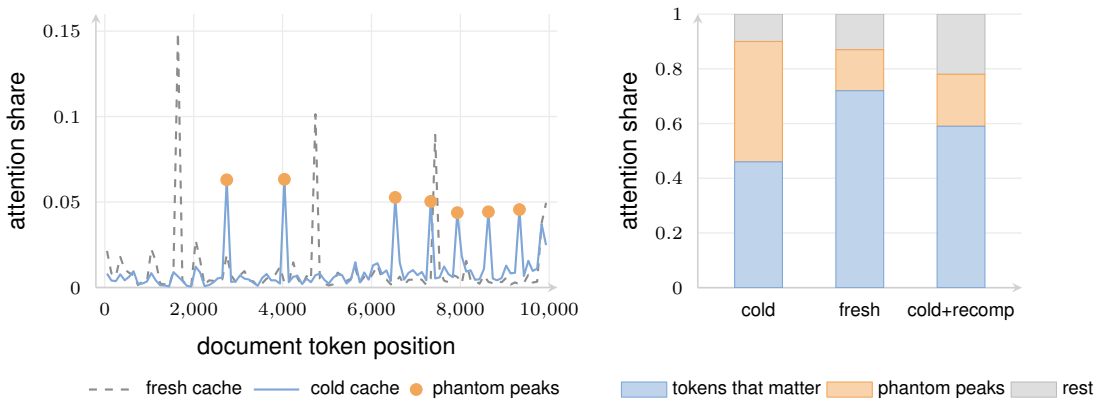
**Observation 2** (Phantom peaks)**.** *On the cold cache the question's attention diverges from the fresh cache starting at the middle layers. It leaves the tokens that hold the answer and concentrates on a few unrelated tokens, which we call* phantom peaks*.*

Figure 1b compares layerwise the attention the question pays to each token on the two caches. The correlation is high on the first layers and diverges from the middle layers onward. Two oracle interventions isolate the keys as the decisive component in this setting. Providing the fresh $q$ vectors while keeping the cold keys changes nothing, while providing the fresh keys restores the accuracy completely. From Figure 2a, we find that on the fresh cache the attention concentrates on the few positions, and that those positions receive almost no attention mass on the cold cache. Figure 2b generalizes on a large set of questions, showing that the phantom peaks take almost half of the question's attention mass. Attention concentrating on a few tokens that do not hold the answer has also been observed in streaming (Xiao et al., 2024), but those attention sinks are inherent to the model, while phantom peaks are artifacts of the cold cache and vanish on the fresh one.

**Problem 3. A repair has to remove the phantom peaks.**

**Observation 3** (Repair target)**.** *Recomputing the tokens the cold question attends restores accuracy. Recomputing the tokens the fresh question would attend does not.*

In prior methods the main course of action is the online repair of the cold cache. Methods recompute a subset of tokens with the assembled cache as context. The natural assumption is that the tokens which matter for the answer should be recomputed. We find this not to be the case. Recomputing the quarter of tokens the fresh question attends most leaves accuracy at the level of the cold cache. Recomputing the quarter the cold question attends most recovers half of the gap and moves the attention back to the tokens that matter (Figure 2b, third bar). A phantom peak left in place continues to misdirect the attention, so a recomputed answer token surrounded by phantom peaks is not read. Only the question's own attention on the cold cache reveals where the phantom peaks are.




(a) Phantom peaks for a single question.

(b) Average over a set of questions.

Figure 2: (a) For a single question on the fresh cache the attention sits on the few positions that hold the answer (dashed). On the cold cache those positions get almost nothing and the mass moves to unrelated tokens (orange dots). (b) The phantom peaks take almost half of the attention. Recomputing the tokens the cold question attends gives most of it back to the tokens that matter.

**Consequence. Matching the fresh attention is the wrong objective.**

**Observation 4** (Inverse correlation)**.** *Among caches that are repaired only partly, accuracy does not grow with the similarity of the question's attention to the fresh cache. The best restored caches have a lower similarity to the fresh cache than worse ones.*

A natural score for a repaired cache is the correlation of the question's attention with the fresh cache. The warm cache of Section 3 breaks the pattern. It answers better than a cold cache with a large recompute while its attention correlates less with the fresh cache (as we also show Figure 9 of the Appendix). The explanation is the one from Observation 3. Every token that is not recomputed keeps wrong values, so any attention it receives is an error. A good partial cache therefore puts less attention on those tokens than the fresh cache does, and thus the warm cache must correlate less with the fresh cache the better it gets. The recompute budget cannot be judged by attention similarity, and repairs that only remove or down-weight tokens without recomputing struggle in restoration, since the attention then moves to other tokens with wrong values.

**What follows for a fix.** (i) The keys in the middle-to-late (semantic) layer bands where attention diverges should be corrected with information from other passages. (ii) Whatever stays uncorrected should receive less attention than in a fresh prefill. (iii) If the cache is repaired online, the tokens to recompute are the phantom peaks, and only the question's own attention on the cold cache reveals them.

## 3 Raftel: No Passage Is an Island

Raftel finds that encoding a passage by itself creates KV cache orphans, but encoded together with its neighbours it becomes the passage the model would have seen inside its full context. We provide an overview of the core concept outlined in Figure 3.

The analysis prescribes three fixes and Raftel implements each in turn. The keys of the semantic layers must be corrected with information from other passages, which Raftel does by encoding each passage offline against its neighbours' stored caches. Whatever stays uncorrected should receive less attention than in a fresh prefill, and the warm encoding removes most phantom peaks so the remaining tokens draw less attention than they would in a cold cache. The tokens to recompute online are the phantom peaks themselves, which Raftel finds by running the question once over the assembled warm cache and recomputing the tokens it attends most at the profiled layers.

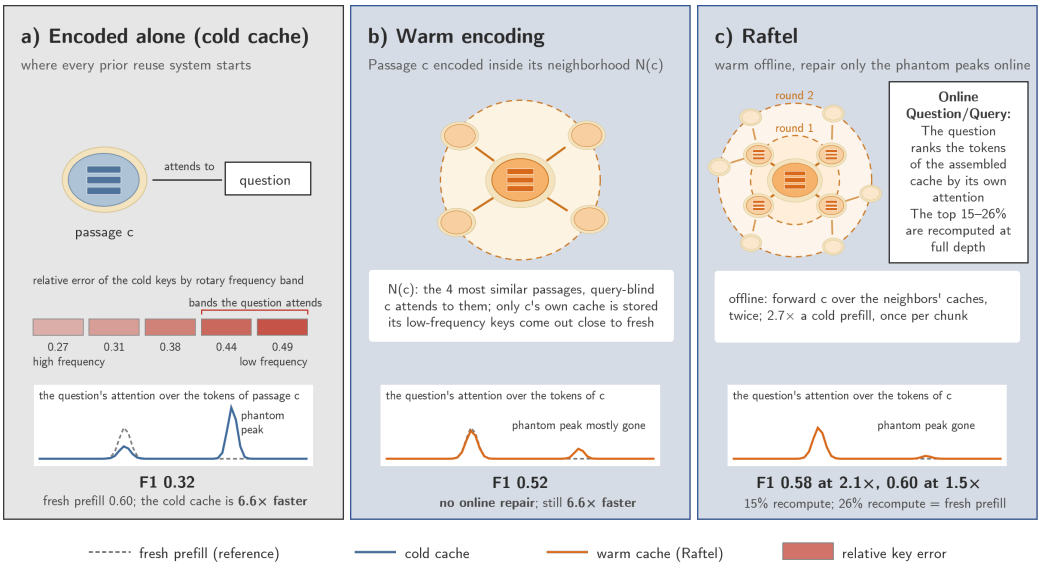



Figure 3: Overview of Raftel by the example of Qwen3-8B on HotpotQA with a fresh prefill F1 0.602. (a) A passage encoded alone contains key errors concentrated in low-frequency rotary bands, and the question's attention diverges in the middle-to-late layers. The question's attention lands on phantom peaks and the answer degrades. (b) Encoding the passage inside a query-blind neighbourhood of its four most similar passages removes most of that error before any question arrives with no online repair. (c) Raftel reaches the warm state through the neighbours' stored caches (two rounds, offline) and recomputes online only the tokens the question itself attends.

**Offline warm encoding.** Without the future queries, we assign every passage $c$ in the corpus $k{=}4$ neighbours $N(c)$, from the passages most similar to it by stop-word-filtered token overlap, or, when the corpus is a set of documents, the chunks adjacent to it in its document. No passage is encoded alone. The stored caches of $N(c)$ are concatenated in front of it and the passage's own tokens are forwarded through all layers attending to that context. Only the keys and values of the current passage are stored in its pre-rotary structure. This costs one forward of the passage tokens, not of the neighbours', which are computed fully offline. The storage is identical to a cold cache, the same number of KV entries per passage with different values. A second round, against the neighbours' round-one caches, propagates warmth one step further over the neighbour graph. In the Appendix, an ablation in Figure 7 shows that two rounds suffice and a third adds essentially nothing. The two rounds together cost $2.7\times$ a cold prefill, paid once per chunk. The marginal offline cost over a cold prefill is $0.13\,\mathrm{s}$ per chunk, while the online savings per query are $0.53\,\mathrm{s}$ for twenty chunks, so a passage reused in a handful of queries already amortizes the warming. We find that related neighbours are what matters. The passage needs the information the neighbours state, not their vocabulary.

**Online path.** The retrieved warm caches are relocated to their positions and concatenated after the system prefix. The question is run once over the assembled cache. At each layer $\ell$ and head $h$, the

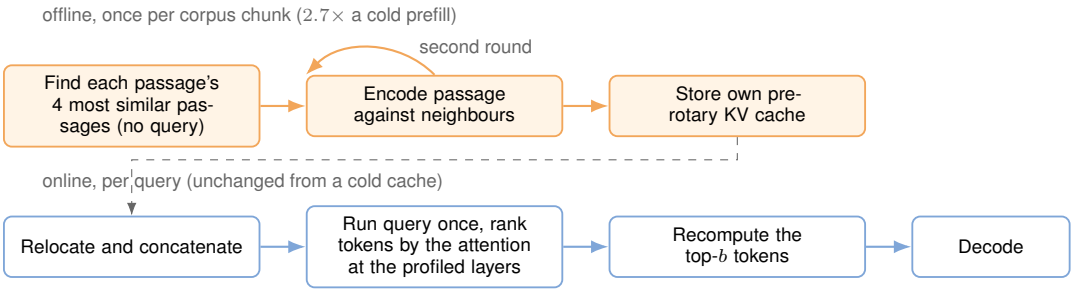



Figure 4: Raftel Pipeline. We encode each passage against its neighbours' stored caches. The online path is the standard assemble–select–recompute–decode pipeline.

**Algorithm 1** Raftel

**Require:** corpus $\mathcal{C}$, neighbour count $k$, budget $b$, profiled layers $L$
**Ensure:** warm KV cache per passage, online answer

**Offline (once per corpus chunk, $2.7\times$ cold prefill)**
**for** each passage $c \in \mathcal{C}$ **do**
  $N(c) \leftarrow k$ most similar passages by overlap or adjacency
  **for** round $r = 1, 2$ **do**
    Prepend round-$(r-1)$ caches of $N(c)$, forward $c$ through all layers
  **end for**
  Store $c$'s keys (pre-rotary) and values ▷ same size as cold cache
**end for**

**Online (per query)**
Relocate and concatenate the retrieved warm caches
Run the question once over the assembled cache
$s_t \leftarrow \frac{1}{|Q||H||L|} \sum_{q,h,\ell} A^{(\ell)}_{q,h}(t)$ ▷ selection score
Recompute top-$b$ tokens by $s_t$ through all layers
Decode

attention from question token $q$ to document token $t$ is

$$A^{(\ell)}_{q,h}(t) = \mathrm{softmax}\big(q^{\top}_{\ell,h} k_{\ell,h}(\cdot)/\sqrt{d_h}\big)_t$$

where $q_{\ell,h}$ and $k_{\ell,h}(t)$ are the query and key vectors and $d_h$ is the head dimension. At the profiled layers, the layers where attention collapses, each document token $t$ receives a selection score

$$s_t = \frac{1}{|Q|\,|H|\,|L|} \sum_{q \in Q} \sum_{h \in H} \sum_{\ell \in L} A^{(\ell)}_{q,h}(t),$$

where $Q$ is the set of question tokens, $H$ the heads, and $L$ the profiled layers. The top-$b$ document tokens by $s_i$ are recomputed through all layers against the assembled cache.

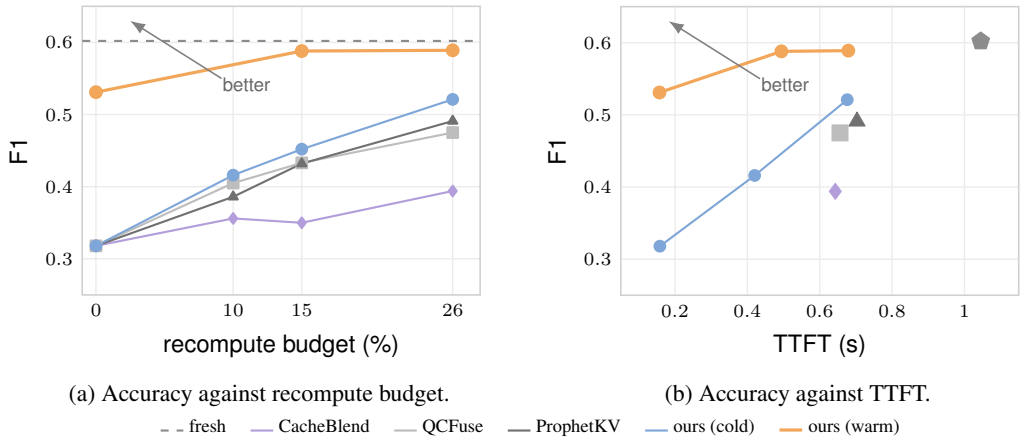
This question-driven selection belongs to the family introduced by ProphetKV (Wang et al., 2026) and QCFuse (Yan et al., 2026). Our variant reads the attention at the layers the analysis singles out, over the full cache. We do not claim the rule itself, only its placement and the explanation of why it works. Section 4.2 sweeps the layer set and finds this band best on both caches. The selection pass costs a few percent of the TTFT and the recompute is the main cost.

## 4 Evaluation

In this section, we summarize key results, showing the relevance of our analysis, and the efficacy of our method. Due to limitations in length, we provide much of our analysis in the Appendix.

**Models, and baselines.** We evaluate on a set of models: Qwen3-8B, Qwen3-14B, Qwen3-32B (Yang et al., 2024), Llama-3.1-8B, Mistral-7B-v0.3, GLM-4-9B-0414 (partial, interleaved rotary), Falcon3-10B. All runs use bf16 on one 80 GB state-of-the-art accelerator card. We compare against a fresh prefill, the cold cache, and the three published methods CacheBlend (Yao et al., 2024), ProphetKV (Wang et al., 2026), and QCFuse (Yan et al., 2026).

**Data, and metric.** We follow state-of-the-art (Yan et al., 2026) evaluation on datasets and metrics. LongBench (Bai et al., 2024) multi-hop question answering (HotpotQA, MuSiQue (Trivedi et al., 2022), 2WikiMQA (Ho et al., 2020)) in the retrieval-serialised form of Yan et al. (2026): each question comes with twenty retrieved passages of $\approx$500 tokens ($\approx$10k document tokens), scored by LongBench F1 with greedy decoding of 48 tokens. We also evaluate on eight RULER (Hsieh et al., 2024) tasks at 16k tokens split into 32 chunks of 512 tokens.




(a) Accuracy against recompute budget.

(b) Accuracy against TTFT.

Figure 5: We visualize the frontier against the three published methods on HotpotQA. The three published rules on the cold cache stay below 0.50 at any budget and just our selection reaches 0.52 at 26%. The bare warm cache already matches that best cold point at a sixth of its time, and the warm cache with a 15% recompute is within 0.015 of fresh prefill at 2.1× its speed.

Table 1: We provide a comparison with state-of-the-art methods on the HotpotQA dataset, to show the benefit of Rafteland also ablate our key findings. The bare warm cache equals the cold cache with a 26% recompute at a sixth of its TTFT

| cache | online step | TTFT ↓ | vs fresh | F1 ↑ | Δ vs fresh [95% CI] |
|---|---|---|---|---|---|
| fresh prefill | — | 1.05 s | 1.0× | 0.602 | — |
| cold | none | 0.15 s | 6.6× | 0.318 | −0.284 [−0.371, −0.201] |
| cold | CacheBlend, recompute 26% | 0.67 s | 1.6× | 0.394 | −0.208 [−0.281, −0.137] |
| cold | QCFuse, recompute 26% | 0.66 s | 1.6× | 0.475 | −0.127 [−0.190, −0.070] |
| cold | ProphetKV, recompute 26% | 0.70 s | 1.5× | 0.491 | −0.111 [−0.174, −0.055] |
| cold | **(ours)**, recompute 26% | 0.68 s | 1.5× | 0.521 | −0.081 [−0.141, −0.026] |
| Raftel (warm, 2 rounds) | none **(ours)** | **0.15 s** | **6.6**× | **0.531** | −0.071 [−0.124, −0.023] |
| Raftel | recompute 15% **(ours)** | **0.50 s** | **2.1**× | **0.588** | −0.015 [−0.044, +0.009] |
| Raftel | recompute 26% **(ours)** | 0.68 s | 1.5× | **0.589** | −0.013 [−0.041, +0.008] |
| Raftel, oracle neighbourhood (all 19 other passages) | none **(ours)** | 0.15 s | 6.6× | 0.541 | — |

**Neighbours.** Each passage's $k$=4 neighbours are determined once by stop-word-filtered token overlap within the corpus, before any query is served. In the benchmark setting the corpus is the set of passages that the benchmark supplies, so the neighbours of a passage are the other passages topically closest to it, which tend to be the ones retrieved alongside it. The choice itself never uses the query.

### 4.1 Main results

Table 1 shows that without any online work the warm cache matches the prior state of the art's best point (a cold cache with 26% recompute) at 6.6× the speed of fresh prefill. With a 15% recompute it is within 0.015 of fresh prefill at 2.1×, and 26% adds nothing more. The last row is the ceiling of the approach: warmth with the whole retrieved set answers as well as a fresh prefill of the same questions without any online work, so the room between the cold cache and that row is entirely how much related context a passage saw when it was encoded.

### 4.2 Three benchmarks

Figure 6 additionally shows the main configurations on MuSiQue and 2WikiMQA. The deviation rule is below the question-driven rule, the bare warm cache is at or above the cold cache with 26% recompute, and the warm cache with a 15% recompute is within noise of fresh prefill. The size of the cold-start gap varies (from 0.12 on 2WikiMQA to 0.28 on HotpotQA), and on 2WikiMQA the bare warm cache is already the best point (0.424 against 0.380 for fresh prefill), so the recompute has nothing left to repair there. Additional benchmark results on the QCFuse data release and CacheBlend's own setting are in Appendix F. The additional benchmarks show that the warm cache generally improves the accuracy–cost trade-off, although the strongest cold-cache selection rule varies by benchmark and model.

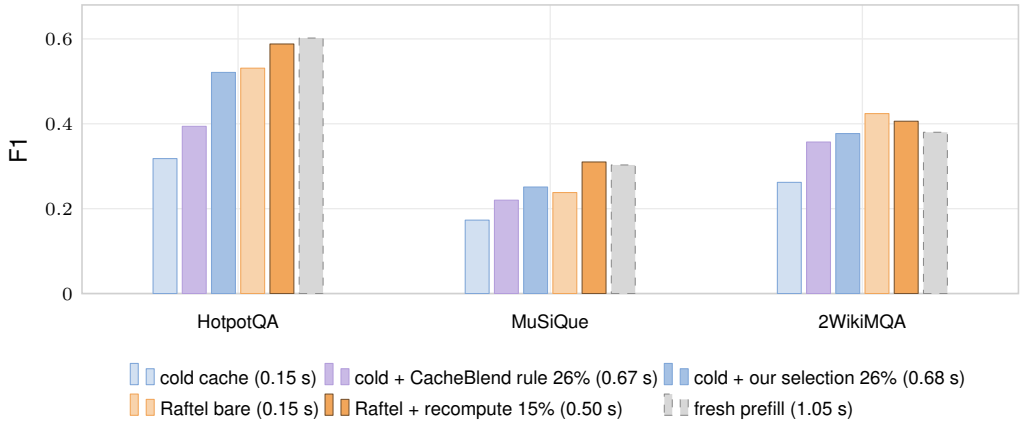



Figure 6: Three LongBench multi-hop benchmarks under the same QCFuse (Yan et al., 2026) protocol. The bare warm cache is at or above the cold cache with 26% recompute on every benchmark at a sixth of its TTFT. With a 15% recompute it is close to fresh prefill on all three.

Table 2: Other model families and sizes under the same harness. A passage encoded alone loses F1 on every model, question-driven selection beats the deviation rule at the same budget on every model, and the warm cache with a 26% recompute is within noise of fresh prefill on all of them.

| | | | cold cache, recompute 26% | | | | Raftel | | |
|---|---|---|---|---|---|---|---|---|---|
| model | fresh | cold | CacheBlend | QCFuse | ProphetKV | ours **(ours)** | bare **(ours)** | +15% **(ours)** | +26% **(ours)** |
| Qwen3-8B | 0.553 | 0.273 | 0.343 | 0.415 | 0.436 | **0.478** | 0.444 | **0.537** | 0.524 |
| Qwen3-14B | 0.547 | 0.256 | 0.384 | 0.475 | 0.467 | **0.508** | 0.483 | 0.523 | **0.546** |
| GLM-4-9B-0414 | 0.471 | 0.252 | 0.337 | 0.379 | 0.361 | **0.381** | 0.341 | 0.428 | **0.454** |
| Falcon3-10B | 0.517 | 0.236 | 0.398 | **0.490** | 0.429 | 0.454 | 0.476 | 0.492 | **0.505** |
| Llama-3.1-8B | 0.472 | 0.287 | 0.427 | 0.438 | **0.448** | 0.446 | 0.432 | 0.459 | **0.479** |
| Mistral-7B-v0.3 | 0.459 | 0.287 | 0.366 | 0.425 | 0.407 | **0.434** | 0.412 | 0.435 | **0.449** |
| Qwen3-32B | 0.530 | 0.308 | 0.414 | 0.457 | 0.461 | **0.535** | 0.491 | **0.549** | 0.536 |
| average | 0.465 | 0.255 | 0.353 | 0.405 | 0.395 | 0.425 | 0.408 | 0.450 | 0.459 |

### 4.3 Cross-model results

Table 2 shows HotpotQA comparison on other model families with each model's own profiled layers. The three findings transfer. A passage encoded alone loses 0.17–0.29 F1 on Qwen3-14B, Qwen3-32B, Llama-3.1-8B and Mistral-7B, question-driven selection beats the deviation rule at the same budget, and the warm cache with 26% recompute ties fresh prefill on Llama, Mistral and Qwen3-14B. The rules are closer to each other on Llama than on Qwen3, where the layer placement matters most. Table 3 gives the time to first token for the same models. The speed-ups are properties of the budget, not of the model, and grow with model size as prefill grows faster than the fixed part of the reuse path. Table 4 shows results on RULER, where the tasks separate into three kinds. On needle tasks the cold cache loses half of the accuracy and every question-driven rule, including ours, restores it. The bare warm cache recovers most of the gap with no online work. On the two question-answering tasks the bare warm cache is at or above fresh prefill, as on the LongBench benchmarks. Additional cross-model results including the 32B model on all three benchmarks and RULER on Qwen3-32B are in Appendix G.

## 5 Related Work

Existing KV reuse methods differ mainly in how the selection rule chooses which values to recompute. CacheBlend (Yao et al., 2024) concatenates per-chunk caches and recomputes the tokens whose first-layer KV entries deviate most between the cold and the recomputed cache, and was the first to show the gains that cache reuse can achieve. EPIC (Hu et al., 2024) pre-chooses which components of the KV cache to recompute, observing that tokens at chunk boundaries carry the largest deviation. Both methods work by mainly comparing the current and old KV cache, thus are often termed query-agnostic. Our analysis shows an inverse relation between the fresh cache's attention and the cold cache's, which explains why deviation-based rules lose accuracy as the budget shrinks.

Table 3: Time to first token evaluation by model. The first six columns keep the persistent cache on HBM, while the last two include the copy of the passages' caches from CPU. The bare cache is 5–13× faster than fresh prefill, a 15% recompute 2.0–2.7×, a 26% recompute 1.5–1.9×, and the selection pass costs a few percent regardless of the rule.

| model (layers) | fresh prefill | cold, bare | cold + ours 26% **(ours)** | ProphetKV 26% | Raftel + 15% **(ours)** | Raftel + 26% **(ours)** | Raftel bare, incl. CPU | Raftel + 15%, incl. CPU |
|---|---|---|---|---|---|---|---|---|
| Qwen3-8B (36) | 1.045 s | 0.158 s (6.6×) | 0.676 s (1.5×) | 0.703 s (1.5×) | 0.494 s (2.1×) | 0.679 s (1.5×) | 0.324 s (3.1×) | 0.660 s (1.6×) |
| Llama-3.1-8B (32) | 0.896 s | 0.134 s (6.7×) | 0.565 s (1.6×) | 0.583 s (1.5×) | 0.420 s (2.1×) | 0.563 s (1.6×) | 0.271 s (3.3×) | 0.563 s (1.6×) |
| Mistral-7B-v0.3 (32) | 1.076 s | 0.137 s (7.8×) | 0.682 s (1.6×) | 0.708 s (1.5×) | 0.476 s (2.3×) | 0.680 s (1.6×) | 0.293 s (3.7×) | 0.635 s (1.7×) |
| GLM-4-9B-0414 (40) | 1.164 s | 0.240 s (4.9×) | 0.780 s (1.5×) | 0.806 s (1.4×) | 0.591 s (2.0×) | 0.775 s (1.5×) | 0.325 s (3.6×) | 0.687 s (1.7×) |
| Falcon3-10B (40) | 1.193 s | 0.172 s (7.0×) | 0.765 s (1.6×) | 0.783 s (1.5×) | 0.548 s (2.2×) | 0.766 s (1.6×) | 0.369 s (3.2×) | 0.749 s (1.6×) |
| Qwen3-14B (40) | 1.724 s | 0.182 s (9.5×) | 0.970 s (1.8×) | 0.997 s (1.7×) | 0.688 s (2.5×) | 0.967 s (1.8×) | 0.457 s (3.8×) | 0.988 s (1.7×) |
| Qwen3-32B (64) | 4.041 s | 0.309 s (13.1×) | 2.146 s (1.9×) | 2.227 s (1.8×) | 1.488 s (2.7×) | 2.154 s (1.9×) | 0.604 s (6.7×) | 1.812 s (2.2×) |
| average | 1.505 s | 0.184 s (8.2×) | 0.895 s (1.7×) | 0.925 s (1.6×) | 0.642 s (2.3×) | 0.890 s (1.7×) | 0.378 s (4.0×) | 0.871 s (1.7×) |

Table 4: We evaluate RULER at 16k tokens with Qwen3-8B. Our configuration contains 32 chunks of 512 tokens while using the official RULER metrics and answer prefixes. All results have been produced on a 15% recompute budget.

| | fresh prefill | | cold | CacheBlend | | QCFuse | | ProphetKV | | Raftel **(ours)** | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RULER task (16k) | acc ↑ | TTFT (s) ↓ | acc ↑ | acc ↑ | TTFT (s) ↓ | acc ↑ | TTFT (s) ↓ | acc ↑ | TTFT (s) ↓ | acc ↑ | TTFT (s) ↓ |
| MQ | 99.69 | 1.80 | 45.00 | 62.19 | 0.86 | 98.12 | 0.82 | 96.25 | 0.92 | **99.06** | 0.87 |
| MV | 97.19 | 1.80 | 25.94 | 43.12 | 0.86 | 92.19 | 0.82 | 86.56 | 0.91 | **96.88** | 0.87 |
| MK-1 | 100.00 | 1.80 | 41.25 | 60.00 | 0.86 | 96.25 | 0.83 | 92.50 | 0.92 | **100.00** | 0.87 |
| QA-1 | 75.00 | 1.61 | 50.00 | 61.25 | 0.77 | **73.75** | 0.74 | 72.50 | 0.80 | 72.50 | 0.77 |
| QA-2 | 47.50 | 1.68 | 25.00 | 40.00 | 0.78 | **48.75** | 0.75 | 47.50 | 0.83 | 47.50 | 0.79 |
| FWE | 97.92 | 1.77 | 96.25 | 97.08 | 0.85 | **98.33** | 0.82 | **98.33** | 0.91 | 97.92 | 0.87 |
| CWE | 94.00 | 1.78 | 83.75 | 86.12 | 0.86 | 78.12 | 0.82 | 84.00 | 0.92 | **94.12** | 0.87 |
| VT | 99.50 | 1.81 | 54.00 | 66.00 | 0.86 | 97.00 | 0.83 | **98.00** | 0.92 | **98.00** | 0.87 |
| average | 88.85 | 1.76 | 52.65 | 64.47 | 0.84 | 85.32 | 0.80 | 84.46 | 0.89 | **88.56** | 0.85 |

ProphetKV (Wang et al., 2026), QCFuse (Yan et al., 2026), and CacheSlide (Liu et al., 2026a) move to an online, query-aware selection. ProphetKV aggregates query-to-context attention across query tokens and model layers, then recomputes the highest-scoring context tokens. QCFuse provides the query a compressed view of the retrieved context through compressed anchors and scores at a few critical layers, making selection cheaper. CacheSlide targets the shift in absolute position, using position-aware encoding to reduce the mismatch and choosing a small recompute set rotated to match the fresh prompt position. RedKnot (Liu et al., 2026b) classifies heads offline and recomputes only the global heads with full-context attention, without the question. Trained alternatives (Yang et al., 2025) fine-tune the model or add trained tokens and are orthogonal to our training-free approach.

Our analysis revisits the assumptions behind these rules. The phantom-peak account offers an explanation for the ordering and budget dependence of the token-level rules, and Raftel reduces part of the mismatch offline that these methods repair online.

## 6 Conclusion

The accuracy gap of a reused per-passage KV cache has a specific source. The keys of a passage encoded in isolation are wrong in the low-frequency rotary bands at the semantic layers, where the question's attention concentrates, and the attention collapses onto phantom peaks that the fresh cache ignores. This diagnosis yields both the selection rule and the offline remedy. The rule recomputes the tokens the mismatched question attends. Raftel warms each passage once against the stored caches of its neighbours, removing most phantom peaks before any query arrives. The bare warm cache matches the best published operating point at 6.6× the speed of fresh prefill, and a 15% recompute on the warm cache reaches within noise of fresh prefill at 2.1×.

The gap between a reused cache and fresh prefill is therefore not inherent to cache reuse. It comes from a fixable encoding error, and fixing it offline is cheaper than repairing it online at every query. This analysis suggests a promising direction for future contributions. While the recomputation budget is fixed across queries, the need for repair is bimodal. We found that most warm-cache queries already match fresh-prefill accuracy, given an appropriate neighbourhood. Learning this neighborhood structure from the corpus could extend the method to settings where related passages are not known at encoding time.

### AI USE STATEMENT

In this work, we used generative AI tools for implementing parts of the code. We have not used generative AI tools for generating research ideas, designing experiments, or analyzing experimental results, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for editing the paper to improve readability and clarity, summarization of main ideas, summarization of prior work in the related work section, suggesting a reordering of the appendix, and modifying figure code. We have reviewed all AI-assisted work. Every AI-suggested edit was checked for accuracy and consistency with our experimental results, and no technical claims or content were introduced by the AI that are not already supported by our experiments and analysis. The code base has been thoroughly tested by the researchers and the main metrics for evaluation and setup have been created by the authors themselves. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES

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, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172.

Yihua Cheng, Yuhan Liu, Jiayi Yao, Yuwei An, Xiaokun Chen, Shaoting Feng, Yuyang Huang, Samuel Shen, Kuntai Du, and Junchen Jiang. Lmcache: An efficient kv cache layer for enterprise-scale llm inference. *ArXiv*, abs/2510.09665, 2025.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. In *European Conference on Artificial Intelligence*, 2025.

Xanh Ho, 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*, pp. 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. URL https://www.aclweb.org/anthology/2020.coling-main.580.

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? *arXiv preprint arXiv:2404.06654*, 2024.

Jun-Hao Hu, Wen-Rui Huang, Weidong Wang, Haoying Wang, Tian-Cheng Hu, Qin Zhang, Hao Feng, Xusheng Chen, Yizhou Shan, and Tao Xie. Epic: Efficient position-independent caching for serving large language models. In *International Conference on Machine Learning*, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Yang Liu, Yunfei Gu, Liqiang Zhang, Chen-Tao Wu, Guangtao Xue, Jie Li, Minyi Guo, Junhao Hu, and Jie Meng. Cacheslide: Unlocking cross position-aware kv cache reuse for accelerating llm serving. In *USENIX Conference on File and Storage Technologies*, 2026a.

Yang Liu, Zhao-Kai Luo, Hua-Yi Jin, Zhi-Yong Wang, Ruo-Zhou He, Bo-Yu Wang, Guan-Jie Chen, Tao Xie, and Jun-Hao Hu. Redknot: Efficient long-context llm serving with head-aware kv reuse and segpagedattention. *ArXiv*, abs/2606.06256, 2026b.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017.

Shihao Wang, Jiahao Chen, Yan-Qi Pan, Hao Huang, Yichen Hao, Xiangyu Zou, Wen Xia, Wen-Tao Zhang, Chong Qiu, and Pengfei Wang. Prophetkv: User-query-driven selective recomputation for efficient kv cache reuse in retrieval-augmented generation. *ArXiv*, abs/2602.02579, 2026.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

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.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=FiM0M8gcct.

Jianxin Yan, Wangze Ni, Zhenxi Li, Jiabao Jin, Zhitao Shen, Haoyang Li, Jia Zhu, Peng Cheng, Xuemin Lin, Lei Chen, and Kui Ren. Qcfuse: Query-aware cache fusion via compressed view for efficient rag serving. *ArXiv*, abs/2606.05875, 2026.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jia-Lin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-Yang Chen, Kexin Yang, Mei Li, Min Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yunyang Wan, Yunfei Chu, Zeyu Cui, Zhen-Ru Zhang, and Zhi-Wei Fan. Qwen2 technical report. *ArXiv*, abs/2407.10671, 2024.

Jing-Bo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. Kvlink: Accelerating large language models via efficient kv cache reuse. *ArXiv*, abs/2502.16002, 2025.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. Cacheblend: Fast large language model serving for rag with cached knowledge fusion. *ACM Transactions on Computer Systems*, 2024.

Table 5: What warms a passage on a major subset of HotpotQA on Qwen3-8B with no online recompute. The fresh prefill accuracy sits at 0.553. Every variant encodes the passage after the given context and stores only the passage's own keys and values. Length does not warm a passage and vocabulary warms it a little. Related text does most of the work, and the neighbours' stored caches reach the same place without re-reading their text. The last column is the paired difference to the cold cache with a bootstrap 95% interval.

| what the passage is encoded with | context tokens | offline cost | bare F1 ↑ | Δ vs cold cache |
|---|---|---|---|---|
| nothing, the cold cache | 0 | 1× | 0.273 | — |
| 4 unrelated passages | 2130 | 5× | 0.324 | +0.051 [−0.009, +0.118] |
| a list of the neighbours' salient terms | 355 | 1.3× | 0.307 | +0.034 [−0.015, +0.094] |
| the neighbours' 6 most related sentences | 360 | 1.7× | 0.352 | +0.079 [+0.013, +0.154] |
| the same sentences with the words shuffled | 353 | 1.7× | 0.339 | +0.065 [−0.007, +0.145] |
| the neighbours' full text, $k$=1 | 500 | 2× | 0.376 | +0.103 [+0.003, +0.205] |
| the neighbours' full text, $k$=2 | 1000 | 3× | 0.407 | +0.134 [+0.034, +0.238] |
| the neighbours' full text, $k$=4 | 2002 | 5× | 0.422 | +0.149 [+0.051, +0.252] |
| the neighbours' full text, $k$=8 | 4001 | 9× | 0.438 | +0.165 [+0.068, +0.267] |
| the neighbours' stored caches, 1 round, Raftel**(ours)** | 0 | 1.3× | 0.395 | +0.122 [+0.036, +0.213] |
| the neighbours' stored caches, 2 rounds, Raftel**(ours)** | 0 | 2.7× | **0.444** | +0.171 [+0.076, +0.274] |
| the neighbours' stored caches, 3 rounds **(ours)** | 0 | 4.1× | 0.443 | +0.170 [+0.065, +0.279] |
| all 19 other passages, the oracle neighbourhood | 9427 | 20× | **0.541** | +0.268 [+0.167, +0.374] |

# A Extended Setup

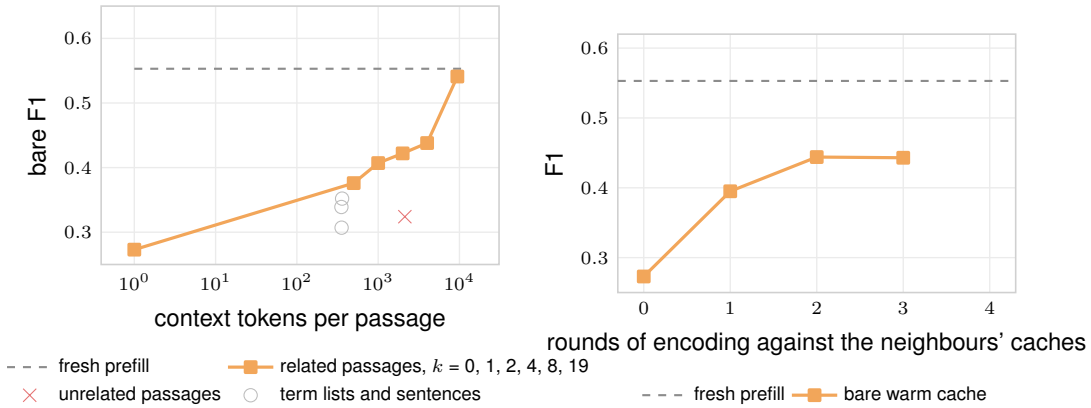
We measure the TTFT on a state-of-the-art accelerator card with 80 GB of HBM with all caches HBM-resident, similar to previous state-of-the-art methods. The timer covers assembly, selection, recompute, suffix prefill, first token and synchronisation. The initial construction of the KV cache artifacts is excluded. Two independent timing runs agree within 5%. A serving system such as vLLM (Kwon et al., 2023) or LMCache (Cheng et al., 2025) must move each passage's cache from host memory to HBM, so we also report each configuration including that copy. It adds 0.1–0.3 s to every reuse arm, as seen in Table 3, so with the copy included the bare warm cache is 3.2–6.7× faster than fresh prefill and the warm cache with a 15% recompute 1.6–2.2×.

We also measure some statistics over the totality of our work. Two logically identical implementations of our and prior state-of-the-art methods differ by ≈0.05 F1, so we report paired bootstrap 95% confidence intervals of the mean difference. Decoding is greedy, so the results are deterministic and not affected by sampling temperature.

# B Ablations

Two important comparisons relate to the neighbourhood chosen to compute the warm cache. The essential questions are what a passage is encoded with and how many rounds it takes, both of which require an in-depth comparison.

Table 5 and Figure 7 say what warms a passage. Four unrelated passages of two thousand tokens leave the cache where it was. A list of the neighbours' salient terms, and their sentences with the words shuffled, move it by less than half of what the sentences themselves do. Related text is what matters, and more of it is better, from one neighbour to eight and up to the whole retrieved set, which is the ceiling of the idea. The text also need not be recomputed or re-prefilled. Encoding the passage against the neighbours' stored caches reaches the same place as their full text in two rounds, at a fraction of the cost as seen in Table 6. However, one round is not enough, and we do not use all ground-truth passages as neighbours, as that would make the comparison trivial. In the first round the neighbours' caches are themselves cold. The second round encodes the passage against neighbours that have been warmed, and a third adds nothing. The neighbour count is not sensitive, so we use four throughout.




(a) Only related text warms a passage, and more of it helps.

(b) Encoding against the neighbours' stored caches converges in two rounds.

Figure 7: Ablations of warmth on a major subset of HotpotQA on Qwen3-8B with no online recompute. (a) Accuracy against the number of context tokens the passage was encoded with. (b) Accuracy of the warm cache against the number of rounds.

Table 6: Offline cost. A million chunks warm in about 55 accelerator-hours on one state-of-the-art accelerator card, once. A new document needs only its own chunks' rounds against the existing index.

| per corpus chunk, isolated | tokens forwarded | time | vs cold |
|---|---|---|---|
| cold prefill | 495 | 0.073 s | 1.0× |
| neighbours' full text prefilled with the passage, 4 neighbours | 2,490 | 0.262 s | 3.6× |
| Raftel, one round, assembling 4 stored caches and forwarding the passage **(ours)** | 495 | 0.099 s | 1.3× |
| Raftel, two rounds **(ours)** | 990 | 0.20 s | 2.7× |
| fresh prefill of the twenty passages, the reference | 9,900 | 1.006 s | 13.7× |

## C Neighbour Sources

The central assumption in Raftel is that related neighbours are known when a passage is encoded. This holds naturally for chunks from the same document and may hold in curated or evolving memory systems, but it is not guaranteed for an unstructured corpus.

Table 7 measures the method under the neighbourhoods a deployment has, with the neighbour choice blind to the query and the retrieved set excluded by construction where noted. Two findings. Neighbours from a large generic corpus, whether by word overlap or by an embedding index, are similar in topic but rarely contain the linked facts a multi-hop question needs, and warm the cache by 0.02 to 0.05. A chunk's own document, the neighbourhood every document-chunking pipeline provides for free, warms the cache to above fresh prefill on NarrativeQA even when the adjacent chunks that were retrieved are excluded. The benchmark's own passage set sits between the two, and the passages retrieved with a chunk, which a deployment also has for free once a query has been served, give most of the oracle's gain. The practical rule is therefore to warm each chunk with its document. Where there is no document, an index adds a little and costs nothing online.

## D Selection Rules

Table 8 isolates the selection rule. Holding the cache and the budget fixed and changing only the rule that picks the recompute set separates the contribution of the selection rule from the warm cache contribution. Deviation-based selection finds the tokens whose KV changed most, not the tokens the question wrongly favours, and does not improve with budget. The question-driven rules differ by where the question's attention is read: uniformly over all layers (ProphetKV), over anchors at three layers (QCFuse), or at the layers where the attention collapses (ours). Ours leads at 26% by 0.04, 0.06 and 0.13 respectively. The same picture holds across the other benchmarks and models in Figure 6, Table 10 and Table 2. The QCFuse or the ProphetKV rule is ahead of ours by 0.02–0.04 on

Table 7: Warmth needs related neighbours. Neighbours from a generic corpus, by word overlap or by an embedding index, are semantically similar with mean cosine 0.73 but rarely state the linked facts a multi-hop question needs, and warm the cache by 0.02 to 0.05 only. A chunk's own document, which every RAG deployment has, warms it above fresh prefill on NarrativeQA. With a 26% recompute the embedding-index neighbours give 0.502 against 0.478 cold with 7 wins, 5 losses and 48 ties. The last three rows share the protocol of Table 5. The Wikipedia-index rows are separate runs whose cold baseline is the same 0.273.

| query-blind neighbourhood | benchmark | cold | Raftel bare | fresh prefill |
|---|---|---|---|---|
| unrelated passages of other questions | HotpotQA | 0.273 | 0.324 | 0.553 |
| word-overlap index over 400k Wikipedia paragraphs, retrieved passages excluded | HotpotQA | 0.273 | 0.294 | 0.553 |
| embedding index using BGE-small over the same 400k paragraphs, retrieved passages excluded | HotpotQA | 0.273 | 0.322 | 0.553 |
| word-overlap index over the benchmark's 3,780 passages | HotpotQA | 0.309 | 0.408 | 0.564 |
| **the chunk's own document**, 2 chunks before and after, retrieved chunks excluded **(ours)** | NarrativeQA | 0.221 | **0.278** | 0.239 |
| **the passages retrieved with it (ours)** | HotpotQA | 0.273 | **0.444** | 0.553 |
| all nineteen other retrieved passages, the oracle | HotpotQA | 0.273 | 0.541 | 0.553 |

Table 8: Four selection rules in one harness. A major subset of HotpotQA was used. Fresh prefill scores 0.553, and the cold cache 0.273. Cells marked — were not run, and the warm rows use neighbour lists built from the benchmark's own rows. CacheBlend's rule is flat in the budget, as the false-peak account predicts. The question-driven rules differ by where the question is read. Warm caches lift every rule by 0.03 to 0.13 at 26%.

| | | F1 ↑ at recompute budget | | | |
|---|---|---|---|---|---|
| cache | rule that picks the recompute set | 10% | 15% | 20% | 26% |
| cold | CacheBlend: layer-1 KV deviation | 0.301 | 0.295 | 0.295 | 0.343 |
| cold | QCFuse: anchor-conditioned question, 3 profiled layers | 0.360 | 0.400 | 0.400 | 0.415 |
| cold | ProphetKV: question attention, uniform over all layers | 0.321 | 0.387 | **0.431** | 0.436 |
| cold | question attention at the layers where it collapses **(ours)** | 0.353 | 0.399 | 0.410 | **0.478** |
| Raftel | CacheBlend rule | — | 0.444 | — | 0.475 |
| Raftel | QCFuse rule | — | 0.515 | — | 0.507 |
| Raftel | ProphetKV rule | — | 0.482 | — | 0.515 |
| Raftel | question attention at the layers where it collapses **(ours)** | **0.517** | 0.501 | — | **0.519** |

MuSiQue, 2WikiMQA and Falcon3-10B, within the noise of those runs, while the deviation rule is last on every one of them.

**Where the question is read.** The one free choice in the selection is the set of layers. Table 9 sweeps it with everything else fixed. Reading the first or the last layers recovers little, the middle of the network recovers most of what there is, and the band the analysis singles out is the best on both caches. A uniform read over all layers, which is what ProphetKV does, sits between the middle band and ours. The choice is therefore not tuned per benchmark. It follows the layers where Figure 1b shows the attention collapsing.

## E Cost of the Selection Pass

The budget is a free parameter of the trade-off and the pass that spends it is cheap. Defining the recompute set by a threshold on the question's attention (a multiple of uniform, or a share of the attention mass) instead of a fixed fraction gives the same accuracy at the same mean budget everywhere between 2% and 26%, so the budget is a genuine free parameter of the trade-off. The cheapest useful setting is the peaks alone, the tokens carrying 60% of the question's attention (a median of 82 tokens, 1.5%).

Table 9: Where to read the question's attention. A major subset of HotpotQA was used with Qwen3-8B. The cache and the budget are fixed and only the layer set of the selection changes. Bands are fractions of the depth so they transfer across models. The band we use is the best on both caches, the middle band and a uniform read over all layers follow, and reading the first or the last layers recovers little. The last column is the paired difference to our band on the cold cache with a bootstrap 95% interval.

| | | F1 ↑ | | |
|---|---|---|---|---|
| layers the question is read at | indices | cold, 26% | warm, 15% | Δ vs ours on cold |
| none, the bare cache | — | 0.273 | 0.444 | — |
| first layers, 0–15% | 1, 2, 3, 4, 5 | 0.355 | 0.448 | −0.123 [−0.210, −0.044] |
| early, 15–35% | 5, 7, 9, 11, 13 | 0.380 | 0.479 | −0.098 [−0.169, −0.035] |
| middle, 35–55% | 13, 14, 16, 18, 20 | 0.445 | 0.533 | −0.033 [−0.076, +0.000] |
| ours, 56–69% | 20, 21, 22, 23, 25 | **0.478** | **0.537** | — |
| late, 70–85% | 25, 27, 28, 29, 31 | 0.440 | 0.511 | −0.038 [−0.101, +0.020] |
| last layers, 86–99% | 31, 32, 33, 34, 35 | 0.369 | 0.477 | −0.109 [−0.193, −0.022] |
| all layers | all | 0.436 | 0.503 | −0.041 [−0.094, −0.002] |
| single layer at 61% | 22 | 0.394 | 0.518 | −0.083 [−0.154, −0.021] |

Table 10: The same comparison on the data release of QCFuse (Yan et al., 2026). All values are F1 ↑ on Qwen3-8B. The chunks are longer and fewer than in our main setting, so the cold-start gap is smaller at 0.12, 0.08 and 0.08 respectively, and the picture is the same. The bare warm cache is at or above fresh prefill on HotpotQA and MuSiQue with no online work. On MuSiQue the improvement is +0.033 with a 95% interval of [−0.023, +0.095]. Question-driven selection beats the deviation rule at the same budget, and on 2WikiMQA the QCFuse rule is the best of the cold-cache rules.

| | | | cold cache, recompute 26% | | | | Raftel warm cache | | |
|---|---|---|---|---|---|---|---|---|---|
| benchmark | fresh | cold | CacheBlend | QCFuse | ProphetKV | ours (**ours**) | bare (**ours**) | +15% (**ours**) | +26% (**ours**) |
| HotpotQA | 0.673 | 0.549 | 0.627 | 0.617 | 0.644 | **0.667** | 0.685 | **0.689** | 0.665 |
| MuSiQue | 0.219 | 0.143 | 0.156 | 0.184 | **0.203** | 0.182 | **0.252** | 0.246 | 0.245 |
| 2WikiMQA | 0.392 | 0.310 | 0.337 | **0.370** | 0.338 | 0.340 | 0.349 | 0.354 | **0.391** |

## F Additional Benchmark Results

Table 10 repeats the comparison on the data release of QCFuse, whose chunks are longer and fewer (ten of ≈1,200 tokens). The gap shrinks to 0.08–0.12, and the bare warm cache alone is at or above fresh prefill on two of the three benchmarks and within 0.04 on the third.

## G Additional Cross-Model Results

The main results use Qwen3-8B. The same experiments on Qwen3-32B, the largest model in our harness, follow.

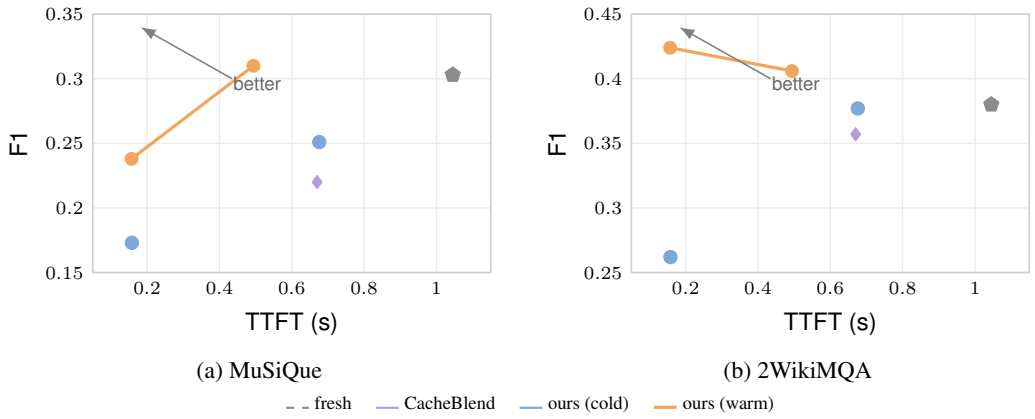
Table 11 takes the 32B model through all three benchmarks. The warm cache exceeds fresh prefill on each of them. Table 12 repeats the three RULER tasks of the 20-chunk files on Qwen3-32B. The same picture at a larger cold-start loss.

## H Reproducibility Details

Every number in the paper comes from one harness based on HuggingFace Transformers (Wolf et al., 2019) with the hyperparameters in Table 13.

## I Limitations

We find three major limitations in our approach.




(a) MuSiQue (b) 2WikiMQA

Figure 8: Accuracy against TTFT on MuSiQue and 2WikiMQA under the same protocol as Figure 5. The bare warm cache is at or above the best cold-cache point on both benchmarks at a fraction of its TTFT. On 2WikiMQA the bare warm cache already exceeds fresh prefill.

Table 11: The largest model on the three benchmarks. Qwen3-32B was used on a major subset of each benchmark, with calibration rows from another benchmark. On HotpotQA and MuSiQue our selection on the cold cache already reaches fresh prefill and the warm cache is at or above it.

| | | | cold cache, recompute 26% | | | | Raftel warm cache | | |
|---|---|---|---|---|---|---|---|---|---|
| Qwen3-32B, benchmark | fresh | cold | CacheBlend | QCFuse | ProphetKV | ours **(ours)** | bare **(ours)** | +15% **(ours)** | +26% **(ours)** |
| HotpotQA | 0.530 | 0.308 | 0.414 | 0.457 | 0.461 | **0.535** | 0.491 | **0.549** | 0.536 |
| MuSiQue | 0.448 | 0.280 | 0.347 | 0.467 | 0.377 | **0.495** | 0.366 | 0.431 | **0.438** |
| 2WikiMQA | 0.458 | 0.406 | 0.450 | 0.430 | **0.469** | 0.445 | 0.459 | 0.476 | **0.478** |

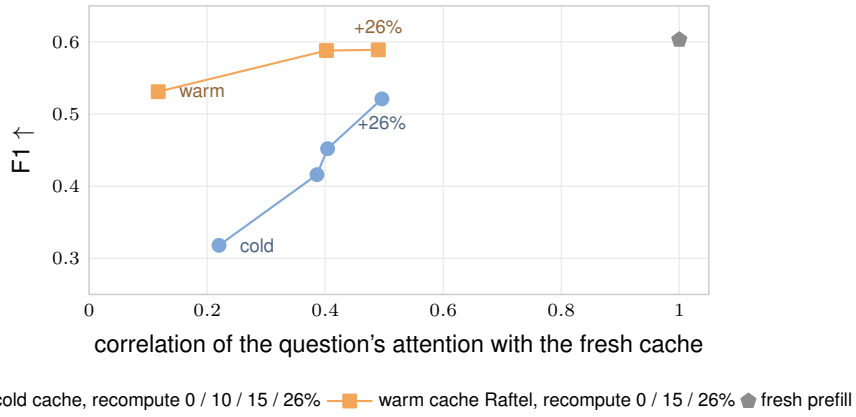



Figure 9: Observation 4. Similarity of the question's attention to the fresh cache against accuracy for the caches of this paper. Qwen3-8B was used on the HotpotQA questions of Table 1. Recompute raises both along each curve. The warm cache answers better than a cold cache with a 26% recompute while its attention correlates less with the fresh cache than the cold cache's does.

First, the neighbourhood must be related. Raftel works where the corpus supplies related neighbours. This can be chunks in the same document or an index of related chunks found by an embedding search common in modern RAG-based memory systems such as Xu et al. (2025); Chhikara et al. (2025). Without related neighbours the method is no better than a cold cache, with an additional offline overhead of 2.7× a cold prefill per chunk.

The recompute budget is a fixed hyperparameter. The need for repair is per question and bimodal (on the warm cache, 84 of 100 HotpotQA questions are already at fresh-prefill accuracy with no recompute, and an oracle that spends the budget only where needed would average under 4%), but neither the statistics of the question pass we examined (sink share, entropy, peak counts, mass

Table 12: RULER on the largest model. Qwen3-32B was used on a major subset of each RULER task, with official RULER metrics and answer prefixes. Only the three tasks with 20 chunks of $\approx$500 tokens are shown. The 32-chunk 16k-token tasks of Table 4 do not fit next to the 32B model's weights on one 80 GB accelerator card in our harness. The needle tasks repeat the 8B picture with a larger cold-start loss and a full recovery by every question-driven rule. The bare warm cache recovers most of the loss on the multi-query task and a smaller part on the multi-value task.

| | | | cold cache, recompute 26% | | | | Raftel | | |
|---|---|---|---|---|---|---|---|---|---|
| Qwen3-32B, RULER task at 16k | fresh | cold | CacheBlend | QCFuse | ProphetKV | ours (**ours**) | bare (**ours**) | +15% (**ours**) | +26% (**ours**) |
| multi-query needles | 1.000 | 0.331 | 0.794 | **1.000** | **1.000** | **1.000** | 0.988 | **1.000** | **1.000** |
| multi-value needles | 1.000 | 0.250 | 0.544 | 0.981 | **1.000** | 0.994 | 0.794 | **1.000** | **1.000** |
| variable tracking | 0.920 | 0.635 | 0.840 | 0.945 | **0.995** | 0.925 | 0.825 | 0.920 | **0.970** |
| TTFT (s) $\downarrow$ | 7.08 | 0.47 | 4.12 | 4.08 | 4.38 | 4.21 | 0.47 | 2.73 | 4.21 |

Table 13: Hyperparameters and protocol used for every number in the paper.

| setting | value |
|---|---|
| neighbours per passage | 4, by Jaccard overlap of the passages' token sets after removing the 80 most frequent tokens of another benchmark's rows |
| warm-encoding rounds | 2. A third adds nothing, see Figure 7 |
| selection layers, ours | the layers at 56, 58, 61, 64 and 69% of the depth: 20, 21, 22, 23, 25 of 36 for Qwen3-8B |
| QCFuse rule | anchors = 4 chunk-initial tokens + top 15% by key norm per chunk, scored at the three critical layers of the QCFuse release |
| ProphetKV rule | question attention averaged uniformly over all layers |
| CacheBlend rule | tokens with the largest layer-1 key–value deviation between the cold and the recomputed cache |
| recompute | the selected tokens through all layers against the assembled cache, keys and values overwritten |
| budgets | 10, 15, 26% of the document tokens. 15% is the recommended point |
| decoding | greedy, 48 new tokens; Qwen3 with an empty think block |
| metrics | LongBench F1, max over gold answers; RULER string match, all items or any item for the QA tasks |
| calibration rows | rows 100–120 of another benchmark's file, HotpotQA for MuSiQue and 2WikiMQA and MuSiQue for HotpotQA, for the stop-word list, no fitted parameters |
| timing | isolated medians of $5\times3$ runs, HBM-resident caches, host copy reported separately, one state-of-the-art accelerator card with 80 GB HBM, bf16 |

concentration) nor the agreement between the bare answer and the answer after a small recompute predicts reliably which questions need it, so a budget-free gate remains open.

Our TTFT is a research-harness measurement (no batching, no overlap of cache loading with recompute). The host-to-HBM copy is reported, throughput is not. The 32-chunk 16k-token RULER tasks do not fit next to a 32B model on one 80 GB accelerator card in our harness, so the 32B RULER results cover the 20-chunk tasks only. Results are at $\approx$10k tokens on models up to 32B. The analysis figures are from one model, and the largest model we evaluate is 32B, at $n$=40.