# SEAMKV: EFFICIENT CONSTRUCTION OF FULL-PROMPT KV FACTORS UNDER CHUNKED PREFILL

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

## ABSTRACT

Long-context LLM inference faces growing KV-cache memory and increasingly expensive long-prompt prefill. KV-cache compression reduces resident state, while chunked prefill processes prompts incrementally; they compose poorly when the compact representation itself must be built from the full prompt. In XKV-style cross-layer low-rank sharing, each factor depends on all prompt tokens but is produced by a local layer group. Under chunk-major prefill, factor completion is tied to the final chunk, causing dense K/V from multiple groups to overlap near prompt end. We present SEAMKV, which processes all prompt chunks through each factor-producing group in turn before advancing deeper. It accumulates factor statistics during forward execution and completes the remaining prompt-global products over bounded source blocks. After the group has produced all required rows and its last prefill use has passed, SEAMKV replaces dense K/V with the finished factor while preserving the same full-prompt factorization target. Relative to a matched chunk-major control, SEAMKV reduces per-GPU peak memory by **9.5–20.8%** across 32K–128K, while cache-ready time is lower at every tested length; at 128K, the peak number of simultaneously live dense groups falls from **8 to 1**. Matched interventions show that eager reclamation and production-time accumulation do not explain the memory gap; the reduction appears only when execution advances producer completion. State accounting links this earlier completion to shorter dense-state residency, while factor-finalization CUDA time differs by at most 0.33%. In exact arithmetic, the schedules compute the same full-prompt operator; under BF16, their paired RULER and LongBench results remain closely matched. The effect transfers to a per-layer full-prompt key-factor geometry, reducing peak memory by **13.6–22.6%**; all 120 paired examples produce identical token sequences under SEAMKV and the matched control. Prompt-global factorization therefore need not imply prompt-end construction.

## 1 INTRODUCTION

Large language models (LLMs) increasingly rely on long context to process large documents, codebases, and multi-turn interactions (Llama Team, 2024; Gemini Team, 2024; Yang et al., 2025). This capability comes with a substantial inference cost. During autoregressive generation, each Transformer layer caches the attention keys and values of previous tokens so that they do not have to be recomputed; the resulting KV cache grows with context length, model depth, and the number of concurrent requests. Long prompts also make the prefill stage itself expensive because many tokens must be processed before decoding can begin. Long-context serving therefore faces two related pressures: how much state must remain resident, and how a long prompt is executed.

Two lines of work address these pressures from complementary directions. KV-cache compression reduces the state retained for future attention through token selection, quantization, cross-layer sharing, or low-rank representations (Zhang et al., 2023; Li et al., 2024; Liu et al., 2024b;a; Chang et al., 2025; Sun et al., 2025). Chunked prefill instead divides a long prompt into bounded token chunks and processes them incrementally, improving execution granularity and making long prompts easier to schedule (Agrawal et al., 2024). In principle, a serving system should be able to use both: chunk the prompt during prefill, then decode from a compact KV representation.



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

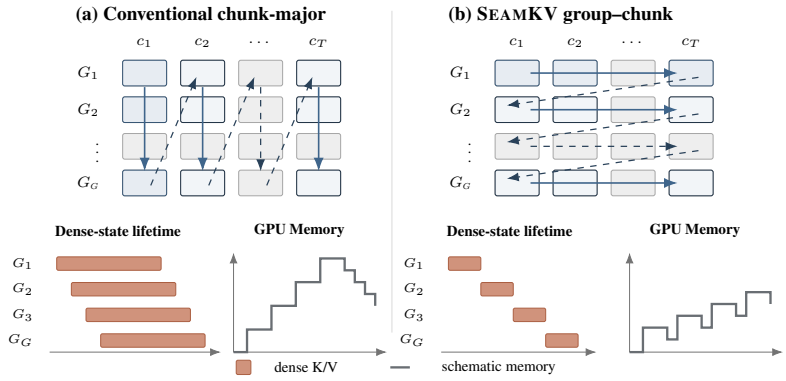


Figure 1: **Execution order and dense-state lifetimes under chunked prefill.** (a) Chunk-major execution bunches producer completion near the prompt tail, leaving dense K/V from multiple groups live concurrently. (b) SeamKV processes the full prompt one group at a time, enabling progressive dense-to-factor replacement. Blue cells and arrows show execution order, orange bars show the lifetime of live dense K/V for each group, and the gray step line shows a schematic GPU-memory trajectory rather than a measured trace.

This composition difficulty is not uniform across KV-compression methods. Token-selection and eviction methods operate on token-level KV states once produced: their compact cache is formed by retaining or dropping those states rather than by completing a shared prompt-wide representation (Zhang et al., 2023; Li et al., 2024). The composition changes when the compact state itself must instead be constructed from request-specific source rows spanning the prompt. We study this case through xKV (Chang et al., 2026). xKV partitions adjacent Transformer layers into *factorization groups* $G_g$ and jointly compresses the K and V states produced by each group into shared low-rank factors. Every prompt token contributes source rows, so each factor is full-prompt in token scope; yet all of its source rows are produced while the prompt crosses that factor's own group, making source production local in model depth. This full-prompt but producer-local structure is what couples factor construction to the execution order of chunked prefill.

Consider conventional chunk-major prefill. Prompt chunk $c_1$ passes through all factorization groups before $c_2$ begins, then $c_2$ passes through all groups, and so on. A factor for $G_g$ cannot be complete until the final prompt chunk has crossed that group. As a result, the completion points of many factors are pushed toward the traversal of the final chunk. Until a compact factor is available and can legally replace its source state, the corresponding dense K/V must remain live for prefill. Dense histories from multiple groups therefore overlap near the prompt tail, increasing peak memory even though the eventual decode representation is compressed. Figure 1 illustrates this mismatch.

A natural first remedy is to keep the standard chunk-major order but finalize and release each group's dense K/V as soon as replacement becomes legal. In matched controls, this reduces Peak/GPU by only 0.66%/0.93% at 64K/128K because producer completion remains near the prompt tail. Adding production-time factor-statistic accumulation reduces construction time by 14.2%/18.3%, yet still does not reduce peak. The fully matched chunk-major control (CM) includes both changes and serves as the causal reference to SeamKV. The hidden bottleneck is therefore not delayed reclamation or factor arithmetic: chunk-major execution makes the complete factor source arrive too late.

The central observation is that *full-prompt scope does not imply prompt-end construction*. A factor may depend on every prompt token even though its complete source is produced at an intermediate depth of the model. Once the final prompt chunk exits $G_g$, deeper groups cannot contribute additional rows to that factor's source; we refer to this event as *producer completion*. In the primary xKV geometry, producer completion coincides with the last prefill use of the group's dense K/V. More generally, construction readiness and replacement eligibility need not coincide. The key distinction is between representation scope and construction timing: a factor can remain full-prompt in scope even when its complete source becomes available before the prompt finishes traversing the model.

This observation motivates SeamKV, which changes when each group receives the full prompt without changing what its factor represents. It executes all prompt chunks through one factorization

group in causal token order before advancing to the next group, exposing producer completion progressively. Along this traversal, SEAMKV accumulates row-additive factor statistics during source production; once producer completion is reached, it performs the remaining bounded prompt-global completion work and installs the same full-prompt factor target when replacement becomes legal.

We evaluate this explanation with matched controls and interventions. Relative to CM, SEAMKV reduces Peak/GPU by 9.5–20.8% from 32K to 128K; at 128K, the peak number of simultaneously live dense groups falls from 8 to 1. Matched interventions rule out reclamation and accumulation alone, isolating producer-oriented execution as the decisive change. The same completion–residency effect transfers to a SHADOWKV-aligned second factor geometry.

**Contributions.**

- We identify a lifecycle mismatch in composing request-specific full-prompt factors with conventional chunked prefill: chunk-major execution bunches producer-completion events near the prompt tail, extending overlap among dense K/V histories.
- We introduce SEAMKV, a producer-oriented construction path for full-prompt factors under chunked prefill. It sends the full prompt through each factor-producing group, advances factor work as source rows are produced, and replaces dense K/V progressively while preserving the same full-prompt target.
- We isolate producer-oriented execution as the decisive change in XKV; state accounting and delayed-installation checks connect earlier producer completion to shorter dense-state residency. We reproduce the same completion–residency relation end-to-end under a second factor geometry.

## 2 PROBLEM FORMULATION

For factorization group $G_g$, let the K- or V-derived source over an $N$-token prompt be

$$X_g = \begin{bmatrix} X_{g,1} \\ \vdots \\ X_{g,T} \end{bmatrix} \in \mathbb{R}^{N \times D_g}, \qquad X_g \approx U_{g,r} B_{g,r}, \tag{1}$$

where $T$ counts prompt chunks, $r$ is the target rank, and $X_{g,c}$ contains rows from chunk $c$ crossing $G_g$. In our primary XKV configuration, four adjacent Transformer layers form one group; K and V are factorized separately, and all prompt rows contribute to one request-specific shared low-rank representation per cache type (Chang et al., 2026). Thus the factor is full-prompt in token scope, while its source production is local to $G_g$ in model depth.

Let $C_g$ denote *producer completion*, when the final row required by $X_g$ has been produced, and $L_g$ the last prefill use of the corresponding dense source. The earliest boundary allowed by these two constraints is $B_g = \max(C_g, L_g)$. Let $M_g$ denote installation of the completed factor; correct execution requires $M_g \succeq B_g$. In the primary XKV geometry, $C_g = L_g = B_g$ when the final prompt chunk exits $G_g$. If instead $L_g > C_g$, factor construction may complete after $C_g$, but dense-state replacement must wait until $L_g$; the exploitable residency reduction is therefore limited by the later last-use boundary. Full-prompt scope does not require $B_g$, or $M_g$, to wait for the prompt to traverse deeper groups. Under conventional chunk-major prefill, however, every $C_g$ lies on the final-chunk traversal, so completion events bunch near the prompt tail and dense histories from multiple groups overlap before replacement.

The primary low-rank construction uses randomized subspace iteration (Halko et al., 2011). With $s = r + p$ ($p$: oversampling dimension) and fixed projection $\Omega \in \mathbb{R}^{D_g \times s}$, its initial right-space statistic is

$$Z_0 = X_g^\top X_g \Omega = \sum_{c=1}^{T} X_{g,c}^\top (X_{g,c} \Omega). \tag{2}$$

This statistic is additive across row blocks and can therefore be accumulated as chunks are produced. The complete-prompt refinement and core steps follow randomized, pass-efficient low-rank routines (Halko et al., 2011; Bjarkason, 2019). Their matrix products decompose over row blocks; SEAMKV exploits this structure to evaluate the prompt-wide products from bounded source views after $C_g$.

# 3 SEAMKV

## 3.1 PRODUCER-BOUNDARY CONSTRUCTION

To expose producer completion progressively under chunked prefill, let $v_{g,c}$ denote executing prompt chunk $c$ through factorization group $G_g$. A legal execution preserves prompt order within a group and hidden-state flow across groups, $v_{g,c} \prec v_{g,c+1}$ and $v_{g,c} \prec v_{g+1,c}$. Subject to these dependencies, SEAMKV uses the order

$$\pi_{\text{Seam}} = (v_{1,1}, \ldots, v_{1,T},\ v_{2,1}, \ldots, v_{2,T},\ \ldots,\ v_{G,1}, \ldots, v_{G,T}), \tag{3}$$

in which all prompt chunks pass through factorization group $G_g$ before execution advances to $G_{g+1}$.

Under this order, $C_g = \text{finish}(v_{g,T})$ is reached as soon as the final prompt chunk exits $G_g$, before execution advances to deeper groups. In the primary XKV geometry, the same boundary is also the last prefill use of the group's dense source, so $L_g = C_g$. The runtime can then finish the globally dependent factor work, install the compact factor, and retire the dense source before deeper groups are processed. The factor still depends on all $N$ prompt rows, while its construction lifetime now follows the source that defines it.

The runtime carries one prompt-wide hidden-state buffer $H[1:T]$ between groups, where $H[c]$ stores the hidden-state block for prompt chunk $c$ at the current group boundary. The entries are updated in place as chunks pass through $G_g$. While $G_g$ is active, its dense causal K/V remains available to all prompt chunks; after the compact factor is installed, execution advances to $G_{g+1}$ with the hidden outputs needed by the deeper group and the factor retained for decode. In serving experiments, group–chunk tiles also define scheduler yield boundaries.

## 3.2 ACCUMULATE, COMPLETE, AND REPLACE

**Accumulate: update additive statistics during production.** When chunk $c$ produces source rows $X_{g,c}$, SEAMKV immediately folds its term into Eq. 2:

$$Z_0 \leftarrow Z_0 + X_{g,c}^\top (X_{g,c}\Omega). \tag{4}$$

Every chunk contributes to the same full-prompt statistic. This places the initial additive pass on the production path instead of rereading those rows later.

**Complete: finish globally dependent work after producer completion.** Once $C_g$ is reached, all source rows for $G_g$ are available. The remaining operators are prompt-global in semantics but blockwise in physical source access. Let $X_{g,b}$ denote bounded source blocks, each holding at most $S$ rows. For refinement iterate $R_j$,

$$Z_{j+1} = X_g^\top (X_g R_j) = \sum_b X_{g,b}^\top (X_{g,b} R_j), \tag{5}$$

with $R_{j+1} = \text{qr}(Z_{j+1})$ at each refinement. After $q$ refinements, range and core construction use the same bounded row decomposition,

$$Y_b = X_{g,b} R_q, \qquad Q = \text{qr}(Y), \qquad C_{\text{core}} = Q^\top X_g = \sum_b Q_b^\top X_{g,b}, \tag{6}$$

where $Y$ stacks the blocks $Y_b$ and $Q_b$ is the aligned row block of $Q$. Writing the rank-$r$ truncation of the small core as $C_{\text{core}} \approx \widetilde{U}_r \Sigma_r V_r^\top$, we set $U_{g,r} = Q\widetilde{U}_r$ and $B_{g,r} = \Sigma_r V_r^\top$, yielding the compact factor $F_g = (U_{g,r}, B_{g,r})$. The operator still covers all $N$ prompt rows; only each physical source view is bounded.

**Replace: install the factor before releasing dense K/V.** After truncation, SEAMKV places $F_g$ in the same representation the XKV decode path reads. Only then is the dense K/V for $G_g$ released. The factorization target, rank, and decode representation are unchanged; earlier producer completion creates the opportunity for earlier replacement, which is realized only after the factor is installed at a legal boundary.

Algorithm A.1 gives the executable per-group procedure and bounded-replay interface for Eqs. 5–6.

### 3.3 State trade-off and implementation

**State trade-off.** Earlier per-group replacement reduces overlap among dense K/V from different groups but requires a hidden-state buffer covering all prompt tokens between groups, factors that become resident earlier, and active factorization state. A simple logical accounting model makes the lever explicit:

$$M_{\text{ctx}}(t) \approx n_D(t)M_D + n_F(t)M_F + M_H(t) + M_C(t), \tag{7}$$

where $M_D$ and $M_F$ denote the per-group dense-state and compact-factor footprints, $n_D(t)$ and $n_F(t)$ count how many are simultaneously live, and $M_H(t)$ and $M_C(t)$ denote the hidden-state-buffer and construction-state footprints, respectively. Equation 7 models logical state lifetime; CUDA allocator peak is measured separately. SeamKV primarily reduces $n_D(t)$ by moving producer completion earlier and installing each completed factor at the earliest legal boundary. Rereading in blocks caps each prompt-wide source view at $S \times D_g$, while the narrow $N \times s$ range/basis state remains prompt-length dependent. SeamKV reduces peak memory when the dense-state overlap it avoids exceeds the overheads from the hidden-state buffer, earlier factor residency, and construction state. Longer contexts amplify the cost of overlapping dense histories, whereas larger factor ranks increase the factor/basis terms, yielding a break-even regime between avoided dense-state overlap and added compact-state residency.

**Implementation.** Bounded replay reconstructs the same pre-RoPE K source used by the factor target (V needs no inverse rotation), so production-time accumulation and replayed products evaluate the same full-prompt operator in exact arithmetic; RoPE/replay details are given in Appendix A.1.

## 4 Evaluation

We evaluate system scaling and utility, the completion–residency mechanism, target fidelity and transfer, and serving consequences. We separate practical comparison from causal attribution: xKV is the one-shot practical reference, while the CM fixes the factor target and construction procedure and retains chunk-major execution.

### 4.1 Experimental setup

**System and controls.** Primary experiments use Llama-3.1-8B-Instruct in BF16 on 2×RTX4090 with a fixed 22-GiB/GPU budget and no CPU/disk offload; Appendix A gives the common factorization configuration and repetition policy. xKV is the frozen one-shot practical reference. The CM retains chunk-major execution but otherwise matches SeamKV in factor target, backend, accumulation, bounded replay, ranks, seed, and earliest-legal installation/reclamation, leaving schedule order and its induced lifetimes as the causal difference.

**Metrics and protocols.** *Peak/GPU* is the maximum CUDA allocation on either GPU; *cache-ready latency* runs from prompt start until compressed KV is decode-ready. Practical quality compares Full Attention, MiniCache, FINCH, xKV, and SeamKV on RULER-10 (Hsieh et al., 2024) with 96 examples/task. Diagnostic, transfer, and serving protocols are specified with their experiments and in the appendix.

### 4.2 Long-context system scaling and utility

**Memory scaling.** Relative to the CM, SeamKV's Peak/GPU reduction grows from 9.46% at 32K to 20.78% at 128K, while the absolute saving grows from 1.08 to 4.17 GiB/GPU (Fig. 2). xKV exceeds the fixed budget at 128K. Against the CM, SeamKV also reduces cache-ready latency, from 6.61% at 32K to 0.98% at 128K. The practical one-shot xKV reference retains lower cache-ready latency at 64K and 96K; exact endpoints are in Appendix Table 4.

**Long-context utility.** On RULER-10 (Table 1), SeamKV remains close to xKV at 32K/64K and, at 128K, completes all 960 examples with a 72.44 average versus 64.19 for FINCH; Full Attention, MiniCache, and xKV exceed the fixed budget. Here OOM refers to the quality-generation workload, not Fig. 2's cache-construction measurement.

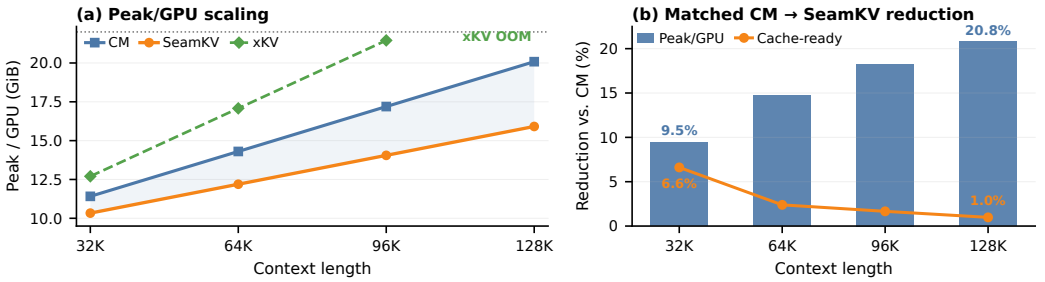


Figure 2: **Long-context system scaling.** (a) Peak/GPU across context lengths for xKV, CM, and SeamKV under the fixed memory budget. (b) Relative CM→SeamKV changes in Peak/GPU and cache-ready latency; OOM marks budget exceedance.

Table 1: **Complete RULER-10 quality matrix on Llama-3.1-8B-Instruct (96 examples/task).** Fixed tested configurations; Full Attention is uncompressed. Bold marks the best successful compressed result per context (ties included); OOM is measured under the fixed budget.

| | | Needle-in-a-Haystack | | | | | | QA | | VT | FWE | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ctx. | Method | N-S1 | N-S2 | MK1 | MK2 | MQ | MV | QA1 | QA2 | | | ↑ |
| 32K | Full Attention | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 99.22 | 84.38 | 55.21 | 99.58 | 93.40 | 93.18 |
| | MiniCache | 68.75 | 81.25 | 67.71 | 17.71 | 36.98 | 26.82 | 58.33 | 39.58 | 49.79 | 54.17 | 50.11 |
| | FINCH | **100.00** | **100.00** | 91.67 | 17.71 | **97.92** | **96.61** | 65.62 | 44.79 | **99.38** | 71.87 | 78.56 |
| | xKV | **100.00** | 92.71 | 96.88 | **98.96** | 93.75 | 95.05 | 72.92 | 50.00 | 97.92 | 90.63 | 88.88 |
| | SeamKV | **100.00** | 93.75 | **100.00** | **98.96** | 91.93 | 95.83 | **76.04** | **51.04** | 97.08 | **90.97** | **89.56** |
| 64K | Full Attention | 100.00 | 100.00 | 97.92 | 100.00 | 100.00 | 99.74 | 80.21 | 52.08 | 98.12 | 84.03 | 91.21 |
| | MiniCache | 63.54 | 62.50 | 40.62 | 19.79 | 19.53 | 13.80 | 55.21 | 34.38 | 38.33 | 39.58 | 38.73 |
| | FINCH | **100.00** | 97.92 | 92.71 | 18.75 | **93.75** | **94.27** | 41.67 | 40.62 | **99.38** | 52.43 | 73.15 |
| | xKV | **100.00** | **98.96** | **97.92** | **96.88** | 91.67 | 89.84 | 73.96 | **48.96** | 93.12 | **83.68** | **87.50** |
| | SeamKV | **100.00** | 97.92 | 96.88 | **96.88** | 92.71 | 84.64 | **76.04** | **48.96** | 92.50 | 81.25 | 86.78 |
| 128K | Full Attention | | | | | OOM | | | | | | |
| | MiniCache | | | | | OOM | | | | | | |
| | FINCH | **100.00** | **93.75** | 84.38 | 6.25 | 80.47 | **72.92** | 40.62 | 36.46 | **99.58** | 27.43 | 64.19 |
| | xKV | | | | | OOM | | | | | | |
| | SeamKV | 98.96 | 92.71 | **93.75** | **84.38** | **82.55** | 68.75 | **69.79** | **37.50** | 46.04 | **50.00** | **72.44** |

### 4.3 Causal mechanism: producer completion and residency

We separate three explanations for the matched memory gap: earlier reclamation after late completion, cheaper factor work from accumulation, and earlier producer completion. Within the matched diagnostic chain, A is the chunk-major construction path; B adds earliest-legal finalization/release; C additionally adds production-time accumulation and is the matched CM control; D replaces chunk-major order with SeamKV's producer-oriented execution.

**Endpoint elimination.** Figure 3 isolates one change per transition. A→B changes Peak/GPU by only 0.66/0.93% at 64K/128K, showing that earliest-legal reclamation alone has little effect. B→C reduces construction time by 14.16/18.29% but does not reduce peak memory. Only C→D produces a substantial Peak/GPU reduction of 14.75/20.78%, while matched factor-finalization CUDA time differs by at most 0.33% (Appendix A.10). Thus neither earlier reclamation nor cheaper factor work explains the endpoint memory gain; the decisive change is producer-oriented execution and the earlier producer completion it induces.

**State accounting.** The C→D endpoint change shortens dense-state residency rather than changing factor size. At 128K, peak live dense groups fall from 8 to 1 and logical dense K/V from 16.0 to 2.0 GiB, while SeamKV adds a 1.0-GiB hidden-state buffer and leaves factor, sketch, and bounded-

replay state matched. Measured physical Peak/GPU falls from 20.084 to 15.912 GiB. Logical state is not an allocator-peak decomposition: allocator peak also includes weights, workspaces, placement, fragmentation, and kernel transients; Appendix Table 8 reports the full measured accounting.

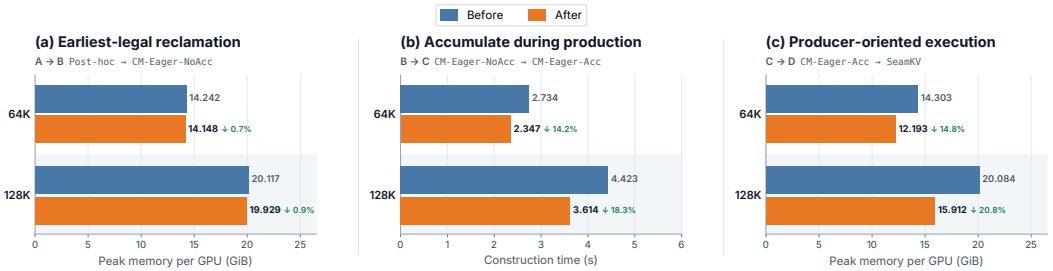


Figure 3: **Matched causal endpoint decomposition.** Panels isolate earliest-legal reclamation (A→B), production-time accumulation (B→C), and producer-oriented execution (C→D); bars show the measured 64K/128K endpoints. B→C reduces construction time without lowering Peak/GPU, while C→D delivers the substantial Peak/GPU reduction reported in the text.

### 4.4 Target preservation and transfer

**Target fidelity.** The schedules evaluate the same full-prompt operator in exact arithmetic; Table 2C–D audits BF16 execution. Paired RULER is dominated by ties (78/80 at 64K; 79/80 at 128K), while the 128K loss interval and the LongBench-6 (Bai et al., 2024) interval both include zero. Represented-matrix deviation is at most 0.141% and mean basis overlap at least 0.999991. Thus the intended target is preserved up to finite-precision differences, not bitwise-identical factors or sequences; exact-sequence details are in Appendix A.8.

Table 2: **Target preservation and transfer.** Panels A–B test transfer across factor geometry and model family; Panels C–D audit paired downstream and forward-pass numerical fidelity.

**A. SHADOWKV-aligned per-layer key representation**

| Ctx. | Peak (GiB) | | Peak red. | Ready (s) | | Live dense layers | Factor Δ | Exact gen. |
|---|---|---|---|---|---|---|---|---|
| | CM | SEAMKV | (%) | CM | SEAMKV | | (%) | |
| 32K | 11.243 | 9.708 | **13.65** | 9.295 | 9.069 | **32→1** | +0.358 | **60/60** |
| 64K | 14.131 | 10.937 | **22.60** | 26.821 | 26.035 | **32→1** | −0.322 | **60/60** |

**B. Matched 32K replication across model families**

| Model | Peak (GiB) | | Peak red. | Ready (s) | | Ready red. |
|---|---|---|---|---|---|---|
| | CM | SEAMKV | (%) | CM | SEAMKV | (%) |
| Llama-3.1-8B-Instruct | 11.413 | 10.333 | 9.46 | 9.898 | 9.244 | 6.61 |
| Mistral-7B-Instruct-v0.2 | 9.601 | 8.656 | 9.84 | 9.972 | 9.282 | 6.92 |

**C. Paired downstream fidelity**

| Benchmark | Ctx. | $n$ | Score | | Δ | 95% CI | W/T/L |
|---|---|---|---|---|---|---|---|
| | | | CM | SEAMKV | | | |
| RULER | 64K | 80 | 82.71 | 84.27 | +1.56 | [0.00, +6.25] | 2/78/0 |
| RULER | 128K | 80 | 72.19 | 71.77 | −0.42 | [−1.67, 0.00] | 0/79/1 |
| LongBench-6 | – | 300 | 45.165 | 44.803 | −0.362 | [−1.591, +0.278] | 23/249/28 |

**D. Forward-pass numerical fidelity**

| Ctx. | Relative matrix deviation ↓ | Mean basis overlap ↑ |
|---|---|---|
| 32K | 0.141% | 0.999994 |
| 64K | 0.136% | 0.999994 |
| 96K | 0.132% | 0.999996 |
| 128K | 0.135% | 0.999991 |

**Factor-geometry transfer.** To test transfer beyond XKV's grouped factorization, a SHADOWKV-aligned port uses per-layer full-prompt pre-RoPE key factors with landmark retrieval and selective reconstruction (Sun et al., 2025). The matched paths share the rank-96 target, factorization/retrieval backend, value policy, and decode path; only producer order and induced lifetimes differ. Peak/GPU falls by 13.65%/22.60% at 32K/64K, producer completion and release advance together, factorization CUDA time remains matched, and all 120 paired examples generate identical token sequences (Table 2A). This reproduces the completion–residency relation under the distinct factor geometry and sparse-reconstruction decode path.

A 32K Mistral-7B-Instruct-v0.2 replication shows the same matched C→D trend, with Peak/GPU and cache-ready latency reduced by 9.84% and 6.92% (Table 2B). The 64K rank/group sweep identifies the tested break-even boundary: eight of nine configurations remain on the favorable side, with the highest-rank $G = 2$ configuration crossing it (Appendix Table 5). This is consistent with compact-state residency eventually offsetting the dense overlap removed by earlier replacement.

### 4.5 SERVING CONSEQUENCES

Serving can conflate responsiveness from finer scheduler yield points with capacity from shorter state lifetime. We therefore use controlled workloads that separately expose scheduler responsiveness and residency-limited capacity, followed by periodic arrivals to test sustained-load feasibility. We overlap one resident 8K decode with $N \in \{1, 2, 4\}$ incoming 64K prefills. *CM-coarse* and *CM-tile* differ only in full-model-chunk versus group–chunk-tile yielding, while *SeamKV* shares CM-tile's yield points but changes the legal tile order. Generated-token hashes match across all paths at the common successful $N = 1, 2$ points; repetition details are in Appendix A.7.

CM-coarse→CM-tile changes only yielding. It sharply improves resident P95 gaps while increasing cache-ready time by about 13–15%; the responsiveness gain therefore comes from the finer tile-level yield boundary, not producer-oriented traversal (Fig. 4). With tile-level yielding fixed, SEAMKV reduces peak memory and is the only path that completes $N = 4$ (Fig. 4). Its resident P95 gap is slightly higher than CM-tile at both matched successful points, so traversal improves capacity through shorter residency, not responsiveness.

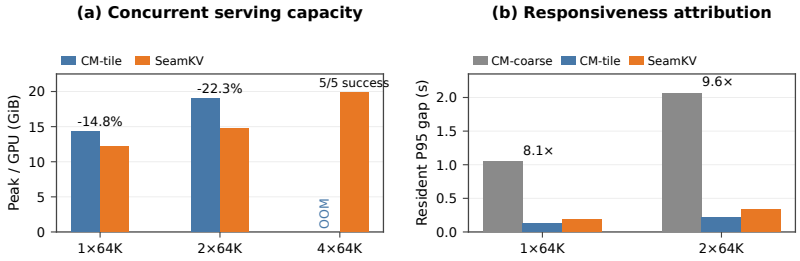


Figure 4: **Matched-yield serving.** (a) Peak/GPU under increasing 64K prefill concurrency, with CM-tile and SEAMKV sharing tile-level yield points. (b) Resident post-trigger P95 gap under coarse versus tile-level yielding; OOM marks budget exceedance.

Unlike Fig. 4's fixed-$N$ overlap, periodic 64K arrivals probe sustained-load feasibility. Both methods sustain the two lower offered loads; at 1.25× the isolated reference rate, the CM OOMs while SEAMKV completes all requests in every repeat (Table 3).

Table 3: **Controlled periodic-arrival memory boundary.** One warmup and three measured repetitions per cell on the same physical GPU pair. For the OOM row, Peak/GPU reports the observed control failure peak; no latency or peak-reduction percentage is compared between successful and OOM runs.

| Offered load | CM | SEAMKV | Peak/GPU (control / SEAMKV) |
|---|---|---|---|
| 0.50× (0.01797 req/s) | 3/3 success | 3/3 success | 14.303 / 12.193 |
| 0.85× (0.03055 req/s) | 3/3 success | 3/3 success | 14.303 / 12.193 |
| 1.25× (0.04492 req/s) | 3/3 OOM | 3/3 success | 22.825 / 16.312 |

## 5 Related Work

**KV-cache compression.** Existing methods include token-retention, eviction, and sparse-cache policies (Zhang et al., 2023; Liu et al., 2023; Adnan et al., 2024; Li et al., 2024; Xiao et al., 2024; Corallo & Papotti, 2024; Tang et al., 2025), quantization (Liu et al., 2024b; Hooper et al., 2024), cross-layer merging (Liu et al., 2024a), and low-rank projection or factorization (Chang et al., 2025; Sun et al., 2025; Chang et al., 2026). OjaKV adapts a low-rank subspace online, PuzzleKV factorizes completed cache pages, and JoLT/FlashJoLT use tensor factorization (Zhu et al., 2026; Wang et al., 2026; Krishnan & Schulz, 2026). SeamKV instead reorganizes chunked prefill around the producer boundary of request-specific prompt-global factors, enabling progressive dense-to-factor replacement without changing the compression target. xKV is the primary cross-layer factorization instance; a ShadowKV-aligned per-layer low-rank key representation with sparse reconstruction provides the second tested geometry.

**Randomized and pass-efficient low-rank approximation.** Randomized subspace iteration and pass-efficient algorithms support low-rank approximation with blocked or limited-pass matrix access (Halko et al., 2011; Bjarkason, 2019). SeamKV uses these routines for prompt-global products that remain after producer completion, evaluated from bounded source blocks. These methods address numerical construction given matrix access; SeamKV addresses when that source becomes complete during model execution.

**Chunked prefill and layer-wise scheduling.** Sarathi-Serve introduced chunked prefill and stall-free batching to interleave long prefills with ongoing decode (Agrawal et al., 2024); TAKE and QuoKA reduce active KV state or attention work during prefill (Hu et al., 2026; Jones et al., 2026), while QUEST uses query-aware page selection for long-context attention (Tang et al., 2024). PagedAttention, FlexGen, Mooncake, LayerKV, and PrefillOnly address KV memory management, offloading/disaggregation, or prefill execution (Kwon et al., 2023; Sheng et al., 2023; Qin et al., 2025; Xiong et al., 2024; Du et al., 2025). Layered Prefill reorganizes layer-group execution to reduce MoE expert-weight reloads and decode stalls (Lee et al., 2026). SeamKV instead derives its execution boundary from request-specific factor-source production and couples producer completion to dense-to-factor replacement; the two optimize different state lifecycles despite both exposing layer-group scheduling units.

## 6 Conclusion

When request-specific full-prompt factors are produced locally in model depth, prompt-global scope need not force prompt-end construction. SeamKV aligns construction with that source-production boundary, enabling progressive dense-to-factor replacement without changing the target. In xKV, Peak/GPU falls by 9.5–20.8% from 32K to 128K and the 128K peak number of live dense groups from 8 to 1; matched interventions rule out eager reclamation and reduced construction time as the explanation. The same completion–residency effect transfers to a distinct per-layer key-factor geometry. Prompt-global factorization therefore need not imply prompt-end construction.

## AI Use Statement

Generative AI tools were used to assist with research-methodology discussion, implementation and debugging, experimental-design feedback, literature search, result interpretation, manuscript drafting and editing, and figure/table planning. All AI-assisted outputs were reviewed by the authors; code and numerical claims were additionally checked against executable tests, experiment outputs, or primary sources. The authors take responsibility for the final text, claims, code, and artifacts.

## Reproducibility Statement

The main text specifies the model, GPU budget, control definitions, memory and timing metrics, practical quality protocol, matched endpoint interventions, and measured state accounting. The appendix records the delayed-installation implementation check together with the common factorization,

chunk/replay, repetition, numerical-validation, paired-fidelity, component-accounting, second-model, and SHADOWKV-transfer protocols, including CUDA event timing.

## REFERENCES


Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J. Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: KV cache reduction through key tokens selection for efficient generative inference. In *MLSys*, 2024.

Amey Agrawal et al. Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve. In *OSDI*, 2024.

Yushi Bai et al. LongBench: A bilingual, multitask benchmark for long context understanding. In *ACL*, 2024.

Elvar K. Bjarkason. Pass-efficient randomized algorithms for low-rank matrix approximation using any number of views. *SIAM Journal on Scientific Computing*, 41(4):A2355–A2383, 2019.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Hung-Yueh Chiang, Yash Akhauri, Xilai Dai, Huiqiang Jiang, Yucheng Li, Luis Ceze, Kai-Chiang Wu, and Mohamed S. Abdelfattah. xKV: Cross-layer KV-cache compression via aligned singular vector extraction. In *ICML*, 2026. arXiv:2503.18893.

Chi-Chih Chang et al. Palu: KV-cache compression with low-rank projection. In *ICLR*, 2025.

Giulio Corallo and Paolo Papotti. FINCH: Prompt-guided key-value cache compression for large language models. *TACL*, 12:1517–1532, 2024.

Kuntai Du et al. PrefillOnly: An inference engine for prefill-only workloads in large language model applications. In *SOSP*, 2025.

Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024. Google; arXiv:2403.05530.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53 (2):217–288, 2011.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *NeurIPS*, 2024.

Cheng-Ping Hsieh et al. RULER: What's the real context size of your long-context language models? In *COLM*, 2024.

Long Hu, Nan Jia, Rui Wang, Jiahui Li, Qingyi Yang, Yixue Hao, Xianzhi Li, and Xiaofei Liao. TAKE: Task-aware chunked KV cache eviction for efficient long-context LLM prefill, 2026. Submitted to ICLR 2026, OpenReview.

Dalton Jones, Junyoung Park, Matthew Morse, Mingu Lee, Chris Lott, and M. Harper Langston. QuoKA: Query-oriented KV selection for efficient LLM prefill. In *ICLR*, 2026.

Rahul Krishnan and Volker Schulz. A JoLT for the KV cache: Near-lossless KV cache compression via joint tucker and JL-residual allocation for LLMs, 2026. arXiv:2607.12550.

Woosuk Kwon et al. Efficient memory management for large language model serving with PagedAttention. In *SOSP*, 2023.

Gunjun Lee, Jiwon Kim, Jaiyoung Park, Younjoo Lee, and Jung Ho Ahn. From tokens to layers: Redefining stall-free scheduling for MoE serving with layered prefill. In *MLSys*, 2026.

Yuhong Li et al. SnapKV: LLM knows what you are looking for before generation. In *NeurIPS*, 2024.

Akide Liu et al. MiniCache: KV cache compression in depth dimension for large language models. In *NeurIPS*, 2024a.

Zichang Liu et al. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *NeurIPS*, 2023.

Zirui Liu et al. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *ICML*, 2024b.

Llama Team. The Llama 3 herd of models, 2024. AI @ Meta; arXiv:2407.21783.

Ruoyu Qin et al. Mooncake: Trading more storage for less computation—a KVCache-centric architecture for serving LLM chatbot. In *FAST*, 2025.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. FlexGen: High-throughput generative inference of large language models with a single GPU. In *ICML*, 2023.

Hanshi Sun et al. ShadowKV: KV cache in shadows for high-throughput long-context LLM inference. In *ICML*, 2025.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Danning Ke, Shikuan Hong, Yiwu Yao, and Gongyi Wang. RazorAttention: Efficient KV cache compression through retrieval heads. In *ICLR*, 2025.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *ICML*, 2024.

Zizhong Wang, Jieying Wang, Zhao Zhang, and Jiajia Li. PuzzleKV: Page-wise low-rank decomposition for KV cache compression, 2026. arXiv:2608.23843.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *ICLR*, 2024.

Yi Xiong et al. LayerKV: Optimizing large language model serving with layer-wise KV cache management, 2024. arXiv:2410.00428.

An Yang et al. Qwen3 technical report, 2025. arXiv:2505.09388.

Zhenyu Zhang et al. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *NeurIPS*, 2023.

Yuxuan Zhu, David H. Yang, Mohammad Mohammadi Amiri, Keerthiram Murugesan, Tejaswini Pedapati, and Pin-Yu Chen. OjaKV: Context-aware online low-rank KV cache compression. In *Findings of ACL*, pp. 10161–10178, 2026.

# A Supplementary Evidence

**Main factorization configuration.** Unless otherwise stated, the main experiments use 2,048-token chunks, 4,096-row bounded source views, four-layer groups, K/V ranks 384/576, oversampling 16, two QR-normalized refinement steps, BF16, seed 42, and a fixed projection policy shared by the CM and SeamKV paths. Primary two-GPU runs expose exactly two GPUs and use Hugging Face `device_map=balanced` for module placement. Single-request measurements use three warmups and ten measured runs.

## A.1 Executable construction and bounded replay

Algorithm A.1 spells out the per-group procedure summarized in the main text. ForwardGroup advances hidden block $H[c]$ through $G_g$ and emits $X_{g,c}$; ReplaySource returns aligned source blocks for $G_g$ with at most $S$ rows each.

For K, ordinary attention retains post-RoPE history while the xKV factor source is pre-RoPE. Accumulation uses the pre-RoPE rows already present in the forward pass; later bounded replay reconstructs only the requested historical K rows using absolute positions and inverse RoPE. V requires no inverse rotation, and ReplaySource exposes at most $S = 4096$ aligned source rows per block. Under these fixed conventions, production-time accumulation and bounded products evaluate the same full-prompt operator in exact arithmetic.

**Algorithm A.1** Factor construction for one SEAMKV group

**Require:** group index $g$, hidden chunks $H[1:T]$, rank $r$, oversampling $p$, refinement count $q$, replay bound $S$

$s \leftarrow r + p$; sample fixed $\Omega \in \mathbb{R}^{D_g \times s}$; $Z_0 \leftarrow 0$
**for** $c = 1, \ldots, T$ **do**
  $H[c], X_{g,c} \leftarrow \text{FORWARDGROUP}(g, H[c])$
  $Z_0 \leftarrow Z_0 + X_{g,c}^\top (X_{g,c}\Omega)$ ▷ accumulate
**end for**
$R_0 \leftarrow \mathrm{qr}(Z_0)$
**for** $j = 0, \ldots, q-1$ **do**
  $Z \leftarrow 0$
  **for all** $X_b \in \text{REPLAYSOURCE}(g, S)$ **do**
    $Z \leftarrow Z + X_b^\top (X_b R_j)$
  **end for**
  $R_{j+1} \leftarrow \mathrm{qr}(Z)$
**end for**
form $Y = X_g R_q$ through bounded views; $Q \leftarrow \mathrm{qr}(Y)$
$C_{\mathrm{core}} \leftarrow 0$
**for all** $(X_b, Q_b)$ from aligned bounded views **do**
  $C_{\mathrm{core}} \leftarrow C_{\mathrm{core}} + Q_b^\top X_b$
**end for**
compute rank-$r$ truncation $\widetilde{U}_r \Sigma_r V_r^\top$
$F_g \leftarrow (Q\widetilde{U}_r, \Sigma_r V_r^\top)$; install $F_g$; retire dense K/V for $G_g$ ▷ replace
**return** $F_g$

### A.2 DELAYED-INSTALLATION IMPLEMENTATION CHECK

We retain one direct physical sanity check for the replacement path. This 64K Llama diagnostic uses the main factorization contract (four-layer groups and K/V ranks 384/576). With the producer-completion profile fixed, delaying factor installation by zero, one, two, or all remaining completion events increases peak live dense groups $1 \to 2 \to 3 \to 8$ and Peak/GPU $11.944 \to 12.725 \to 13.506 \to 14.333$ GiB. This diagnostic is not used to establish the causal mechanism; it verifies that the measured allocator peak responds when an already-complete dense state is deliberately kept resident longer.

### A.3 SYSTEM ENDPOINT MEASUREMENTS

Table 4 gives the exact single-request endpoint measurements underlying Figure 2. The xKV row is the practical one-shot reference; only the CM→SEAMKV comparison carries schedule-level attribution.

### A.4 RULER PROTOCOL NOTES

Table 1 in the main paper gives the complete task-level matrix. The public RULER/xKV task set contains two single-needle retrieval tasks (N-S1/N-S2), two multi-key tasks (MK1/MK2), multi-query and multi-value retrieval (MQ/MV), two question-answering tasks (QA1/QA2), variable tracking (VT), and frequent-word extraction (FWE). We use 96 examples per task. Quality-run OOM is specific to this generation workload and is distinct from the cache-construction measurements in Figure 2 and Table 4.

### A.5 MATCHED RANK/GROUP OPERATING BOUNDARY

The 64K sweep uses Llama-3.1-8B-Instruct on 2×RTX4090 with the main numerical contract. Each cell runs the CM and SEAMKV sequentially on the same physical pair, with one warmup and three measured repetitions. Positive ΔPeak means lower Peak/GPU for SEAMKV.

Table 4: **Exact long-context system endpoints.** The matched control and SEAMKV share the numerical target/backend, production-time accumulation, bounded replay, rank/seed, and earliest-legal reclamation. Positive $\Delta$ values denote reductions relative to the matched control.

| Ctx. | Method | Peak/GPU (GiB)↓ | ΔPeak (%)↑ | Cache-ready (s)↓ | ΔReady (%)↑ |
|---|---|---|---|---|---|
| 32K | xKV | 12.704 | – | 9.345 | – |
| | CM | 11.413 | – | 9.898 | – |
| | SEAMKV | **10.333** | **9.46** | **9.244** | **6.61** |
| 64K | xKV | 17.079 | – | 19.879 | – |
| | CM | 14.303 | – | 25.926 | – |
| | SEAMKV | **12.193** | **14.75** | 25.306 | **2.39** |
| 96K | xKV | 21.454 | – | 34.075 | – |
| | CM | 17.194 | – | 49.852 | – |
| | SEAMKV | **14.052** | **18.27** | 49.027 | **1.66** |
| 128K | xKV | OOM under fixed budget | | | |
| | CM | 20.084 | – | 83.739 | – |
| | SEAMKV | **15.912** | **20.78** | **82.921** | **0.98** |

Table 5: **Matched rank/group boundary at 64K.** Eight of nine settings reduce Peak/GPU; fine grouping plus the highest rank crosses the measured physical break-even.

| Group | K/V rank | CM Peak (GiB) | SEAMKV Peak (GiB) | ΔPeak (%) | ΔReady (%) |
|---|---|---|---|---|---|
| 2 | 256/384 | 14.244 | 11.946 | +16.13 | +4.95 |
| 2 | 384/576 | 14.288 | 12.596 | +11.84 | +4.12 |
| 2 | 768/1152 | 14.457 | 14.569 | −0.78 | +2.97 |
| 4 | 256/384 | 14.254 | 11.851 | +16.86 | +2.68 |
| 4 | 384/576 | 14.303 | 12.193 | +14.75 | +1.97 |
| 4 | 768/1152 | 14.539 | 13.276 | +8.69 | +2.58 |
| 8 | 256/384 | 14.297 | 12.590 | +11.94 | +1.57 |
| 8 | 384/576 | 14.356 | 12.786 | +10.94 | +0.73 |
| 8 | 768/1152 | 14.702 | 13.502 | +8.16 | +0.63 |

At high rank and fine grouping, earlier factor residency can offset the avoided dense-state overlap on the bottleneck device. The main $G = 4$ configuration remains favorable at all three tested ranks. This sweep characterizes the operating boundary.

### A.6 SECOND-MODEL AND SHADOWKV TRANSFER PROTOCOLS

The second-model check repeats the matched C→D comparison on Mistral-7B-Instruct-v0.2 at its native 32K context. It uses 2,048-token chunks, 4,096-row replay, four-layer groups, K/V ranks 384/576, oversampling 16, two QR-normalized refinements, BF16, seed 42, three warmups, and ten measured repetitions. The corresponding Llama and Mistral endpoints are reported in Table 2B.

**SHADOWKV transfer protocol.** The transfer experiment uses Llama-3.1-8B-Instruct on the same 2×RTX4090 class with BF16 and a 22-GiB/GPU budget. Both paths use a per-layer rank-96 full-prompt pre-RoPE key factor, 2,048-token prefill chunks, 4,096-row replay, oversampling 16, two QR-normalized refinements, seed 42, sparse budget 2,048, retrieval chunks of eight tokens, four local chunks, matched landmark/outlier construction, and the same CPU value storage and selective reconstruction path. The CM path uses chunk-major prefill and per-layer factor finalization and release at the earliest legal point on the final chunk; the Seam path processes all prompt chunks through one layer before advancing deeper. Formal system measurements use 3 warmups and 10 measured

repetitions with alternating method order. The endpoint, live-dense-layer, and exact-generation results are reported in Table 2A.

A separate timing audit uses one warmup and three measured repetitions per context and CUDA events rather than host wall intervals. This matters because host-side construction timers are enqueue-sensitive under layer-major execution and are retained only for provenance.

Table 6: **CUDA event timing audit for the SHADOWKV transfer.** Positive advancement means the Seam event occurs earlier. Factor $\Delta = (T_{\text{Seam}} - T_{\text{CM}})/T_{\text{CM}}$ compares summed matched factorization CUDA time; positive values mean SEAMKV is slower.

| Ctx. | Layer-0 source advance | Layer-0 release advance | Factor $\Delta$ |
|---|---|---|---|
| 32K | 6.653 s | 6.675 s | +0.358% |
| 64K | 21.654 s | 21.663 s | −0.322% |

The bounded source peak is 16 MiB in both paths, post-source passes are 96 in both paths, and factor workspace is matched (3,866,576 elements at 32K; 7,496,048 at 64K). These counters, together with the CUDA event factor times, rule out cheaper factorization as the explanation for the transfer result.

### A.7 CONTROLLED-ARRIVAL SERVING DETAILS

Group–chunk tiles define scheduler yield boundaries; resident full-depth decode work can be scheduled between yielded prefill units without changing the legal prefill order. The matched-yield test uses one warmup and five measured runs per point; at $N = 1/2$, the maximum resident gaps improve by 34.8×/39.2× from CM-coarse to CM-tile (these are maxima, whereas Fig. 4 reports resident P95 gaps). The periodic-arrival test uses twelve 64K requests with 8-token decode tails, a four-request active-set cap, the same request content across rates, and scales only inter-arrival time. The isolated matched control reference is 0.0359378 req/s. The high-load control terminates before the full stream arrives, so future scheduled requests are not counted as queued work and no latency or peak-reduction percentage is reported between successful and OOM runs. A separate scaled-exponential trace probes burst robustness: transient bursts cause the matched control to OOM at all three tested average rates once three long prefills overlap, while SEAMKV completes all 12 requests in all measured repeats. The periodic trace above defines the rate boundary; the scaled-exponential trace provides complementary burst-robustness evidence.

### A.8 PAIRED FIDELITY AUDIT DETAILS

The 64K RULER and LongBench-6 audits use one CM path from their same-input paired runs, whereas the 128K RULER audit uses a separately instantiated CM path from its own paired run. Because the reference protocols differ, these results are interpreted within-pair rather than as a cross-context trend. At 64K, the CM scores 82.71 and SEAMKV 84.27 over 80 RULER examples; the paired difference is $+1.56$ with paired 95% CI $[0.00, +6.25]$, W/T/L is 2/78/0, and exact-sequence agreement is 68.8%. At 128K, the CM scores 72.19 and SEAMKV 71.77 over 80 paired examples; $\Delta = -0.42$, paired 95% CI $[-1.67, 0.00]$, W/T/L=0/79/1, and exact-sequence agreement is 78.8%. The sole task-level score loss is on FWE. On 300 paired examples from six LongBench datasets under identical input token digests, the CM scores 45.165 and SEAMKV 44.803. The paired difference is $-0.362$ with 95% hierarchical bootstrap CI $[-1.591, +0.278]$, W/T/L=23/249/28, and exact-sequence agreement 73.3%.

### A.9 NUMERICAL FIDELITY

Forward-pass numerical audits at 32K/64K/96K/128K report maximum relative matrix deviations of 0.141%, 0.136%, 0.132%, and 0.135%, with minimum mean basis overlap 0.999994, 0.999994, 0.999996, and 0.999991. These values measure the aggregate finite-precision differences introduced by the changed accumulation and replay execution path and complement the paired downstream checks above.

### A.10 MATCHED FACTOR FINALIZATION TIMING

The top-level compression timer spans different host-side boundaries in the matched chunk-major path and SEAMKV. We therefore compare C→D factor finalization work with a matched factor finalization CUDA timer, which shows essentially unchanged finalization time.

Table 7: Matched factor finalization CUDA time in the C→D comparison.

| Ctx. | CM (s) | SEAMKV (s) | Relative difference |
|---|---|---|---|
| 64K | 1.554969 | 1.552827 | $-0.14\%$ |
| 128K | 2.857990 | 2.848518 | $-0.33\%$ |

### A.11 CONSTRUCTION STATE ACCOUNTING

Table 8 reports the measured 128K C→D lifecycle-state accounting summarized in Sec. 4.3. Factor, fixed sketch/core, and replay quantities remain matched across C→D; the hidden-state buffer is the principal added logical state. The prompt-length-dependent $N \times s$ range/basis state is reported separately in Table 9.

Table 8: **Measured 128K C→D lifecycle-state accounting.** Logical quantities are model-wide and non-additive; Peak/GPU is the maximum allocator peak across the two GPUs.

| Measured quantity | CM | SEAMKV |
|---|---|---|
| Model-wide logical dense K/V peak | 16.0 GiB | 2.0 GiB |
| Prompt-wide hidden-state buffer | 0 | 1.0 GiB |
| Live factor state | 3.867 GiB | 3.867 GiB |
| Live fixed sketch/core state | 0.030 GiB | 0.030 GiB |
| Bounded replay reads | 96 GiB | 96 GiB |
| Peak live dense groups | 8 | 1 |
| Physical Peak/GPU | 20.084 GiB | 15.912 GiB |

Table 9 summarizes the analytical construction states most relevant to bounded source residency under the fixed Llama configuration. It is not a decomposition of the measured CUDA allocator peak. The accounting uses $D = 4096$, $S = 4096$, $s_K = 400$, and $s_V = 592$ in FP32.

Table 9: **Analytical construction-state accounting.** Selected FP32 state sizes under the fixed Llama configuration.

| Ctx. | Kind | $N \times D$ source | $S \times D$ view | $N \times s$ state | $D \times s$ / $s \times D$ |
|---|---|---|---|---|---|
| 32K | K | 512 | 64 | 50 | 6.25 / 6.25 |
| | V | 512 | 64 | 74 | 9.25 / 9.25 |
| 64K | K | 1024 | 64 | 100 | 6.25 / 6.25 |
| | V | 1024 | 64 | 148 | 9.25 / 9.25 |
| 96K | K | 1536 | 64 | 150 | 6.25 / 6.25 |
| | V | 1536 | 64 | 222 | 9.25 / 9.25 |
| 128K | K | 2048 | 64 | 200 | 6.25 / 6.25 |
| | V | 2048 | 64 | 296 | 9.25 / 9.25 |

Across 32K–128K, a prompt-wide $N \times D$ source grows from 512 to 2,048 MiB, while the bounded $S \times D$ replay view stays at 64 MiB. The retained $N \times s$ state still scales with prompt length, growing from 50 to 200 MiB for K and from 74 to 296 MiB for V; the $D \times s$ and $s \times D$ states remain fixed at 6.25 MiB for K and 9.25 MiB for V. Thus the implementation bounds prompt-wide source residency without claiming context-independent factorization state.