SCALE-KV: STATE-AUGMENTED CROSS-SCALE KV CACHE REUSE FOR LONG-CONTEXT INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context inference and context handoff in multi-agent collaboration incur substantial context-processing costs. Cross-scale key–value (KV) cache reuse allows a large model to reuse context computation performed by a smaller model, reducing repeated large-model prefilling and redundant computation after model handoff. However, structural and representational differences across model scales can degrade generation quality, while additional computation for quality recovery may offset the resulting speedup. We propose **SCALE-KV**, a state-augmented framework for cross-scale KV cache reuse. SCALE-KV establishes cross-model correspondence through a fixed base mapping and learns compact state-augmented residuals from the source KV cache and intermediate states. The cache converter and low-rank target-reader adaptation are jointly optimized for generation while both backbones remain frozen. At inference time, the source model prefills the full context once, after which the target directly processes questions and generates answers from the converted prefix without repeating full-context prefilling. Across three question-answering evaluations with 8–16K contexts, SCALE-KV remains within **3.53 F1 points** of native large-model inference. Including source-model prefilling, SCALE-KV achieves up to $1.66\times$ time-to-first-token (TTFT) speedup for fresh requests; when the source cache and required intermediate states are available, it achieves up to $6.49\times$ TTFT speedup for handoffs.

1 INTRODUCTION

Transformer-based language models (Vaswani et al., 2017) have become the foundation of retrieval-augmented generation (Lewis et al., 2020) and long-document question answering (Bai et al., 2024; 2025). As tasks become increasingly demanding, models of different scales increasingly share and process the same documents, dialogue histories, and intermediate results. Although text provides a common interface across models, the computational states produced from the same context remain model specific: differences in weights, layer structures, and internal representations prevent direct reuse of key–value (KV) caches. Consequently, a receiving model often has to prefill the same context again even after another model has already processed it. For long contexts and large receiving models, eliminating such redundant context processing is therefore a key efficiency challenge.

Existing approaches primarily follow two routes. Input-selection methods reduce the context processed by the target, for example using a lightweight model to estimate token importance (Liu et al., 2025). Although these methods reduce target-side prefilling, they may discard relevant evidence; more importantly, retained context still requires target-model encoding. An alternative is to predict or convert cross-model caches, constructing target-compatible states from computation already performed by the source (Horton et al., 2024; Chen et al., 2026; Heo et al., 2026). By reusing source-side computation, these methods reduce redundant target prefilling and enable a natural division in which a smaller model processes the document while a larger model performs generation.

However, existing cross-scale cache reuse methods still struggle to preserve large-model answer quality while maintaining end-to-end efficiency. Dense Latent Communication transfers states across heterogeneous agents through cache alignment and generation-oriented training (Chen et al., 2026); however, without access to the original input, some of its small-to-large settings still suffer substantial quality degradation and even fail to outperform the source model on several tasks, preventing them from consistently preserving the quality advantage of the larger model. Quality

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.

preservation under long-context settings also remains insufficiently explored. Heo et al. exploit linear correspondences within model families to enable low-cost cache handoff through closed-form mappings (Heo et al., 2026); however, such mappings can still degrade task quality for some model pairs, while conversion-only speedups do not capture the full cost including source-model prefilling. Therefore, preserving target-model answer quality while controlling end-to-end reuse costs remains a key challenge for cross-scale cache reuse.

As illustrated in Figure 1, cross-scale KV cache reuse faces two coupled challenges. The first is **generation-quality preservation**. Differences in layer structure and internal representations make it difficult for a large model to directly exploit contextual information stored in a smaller model's cache. Matching cache dimensions or reducing numerical reconstruction error does not necessarily preserve generation quality (Heo et al., 2026; Qu et al., 2026); moreover, retrieving and integrating evidence distributed across long contexts places even greater demands on cache adaptation (Liu et al., 2024a; Hsieh et al., 2024; Yen et al., 2025; Bai et al., 2025). The first challenge is therefore to bridge cross-model differences so that the large model can maintain answer quality close to native inference without reprocessing the full context. The second is the **trade-off between quality recovery and acceleration**. Insufficient correction may leave the converted cache unable to support accurate generation, whereas more expensive cache adaptation or target-side compensation can erode the efficiency gains from reuse (Heo et al., 2026; Qu et al., 2026). The second challenge is therefore to recover generation quality while keeping correction costs within the computation saved by avoiding large-model prefilling, so that the potential savings translate into actual first-token latency reduction.

To address these challenges, we propose **SCALE-KV**, a state-augmented framework for cross-scale KV cache reuse. The core idea is to construct a generation-ready context cache for the large model from the cache and intermediate states already produced by the small model. Specifically, as shown in Figure 2, the small model processes the full context once, after which the converter constructs a reusable large-model prefix and the adapted target processes subsequent questions and generates answers from this prefix. To preserve generation quality, source KV and intermediate states are used for residual cache correction, while generation-oriented training jointly optimizes the converter and low-rank target-reader adaptation with both model backbones frozen. To control cost, the base mapping performs the main conversion, while compact residuals use states produced by the same source-model forward pass to correct the remaining mismatch, avoiding repeated full-context target prefilling. The resulting document-only prefix can be reused across subsequent questions and provides a natural state-reuse interface for context handoff in multi-agent collaboration. We evaluate long-document question answering with Qwen3 model pairs (Yang et al., 2025), jointly measuring answer quality and end-to-end time to first token. Across three 8–16K question-answering evaluations, SCALE-KV remains within 3.53 F1 points of native large-model inference. In paired evaluations of the main configuration, it achieves 1.64–$1.66\times$ TTFT speedups for fresh requests and 5.38–$6.49\times$ for handoffs with source states ready. Our contributions are summarized as follows.

1. **Analysis and insight.** We show that adapted small-model document states can support answer quality beyond source-only inference, motivating a cross-scale division between document encoding and target-model generation.
2. **SCALE-KV.** We jointly adapt state-augmented low-rank cache correction and target reading around a fixed base mapping, reusing small-model document computation as a reusable target prefix without full-context target prefilling.
3. **Quality and efficiency evaluation.** Experiments demonstrate useful quality–latency gains in long-document QA, with task, context-length, source-size, and component comparisons explaining the benefits of the cross-scale reuse path.

2 Related Work

Long-context inference. Input compression retains useful text before target-model processing: LLMLingua studies compression budgets and information preservation (Jiang et al., 2023; 2024; Pan et al., 2024), and Speculative Prefill uses a lightweight model to guide token selection (Liu et al., 2025). LazyLLM, SwiftKV, and FastKV reduce computation across tokens, layers, or execution stages (Fu et al.; Qiao et al., 2025; Jo et al., 2026). Complementary approaches quantize caches (Liu et al., 2024c; Hooper et al., 2024), allocate retention budgets (Zhang et al., 2023; Li et al., 2024; Cai et al., 2024), or sparsify attention reads (Tang et al., 2024). PagedAttention and CacheGen address

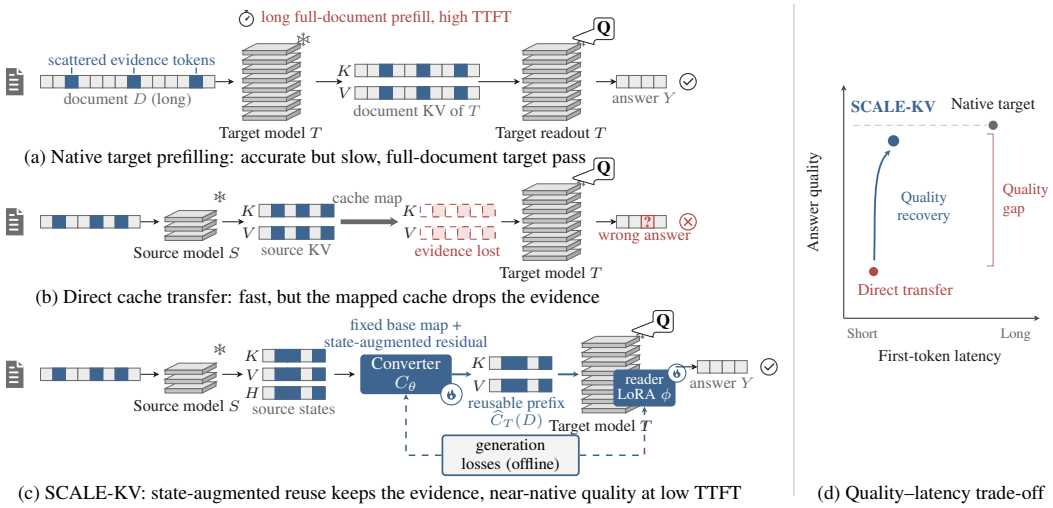

Figure 1: Motivation and challenges: native prefilling, quality loss in direct transfer, and state-augmented reuse. The right-hand panels illustrate the conceptual quality–latency trade-off.

cache management and transfer (Kwon et al., 2023; Liu et al., 2024b), while disaggregation and chunked-prefill scheduling improve serving efficiency (Zhong et al., 2024; Agrawal et al., 2024; Patel et al., 2024). Our focus is the computation shared across model boundaries: full-document processing by a small model replaces repeated prefilling by a larger one.

Cross-model cache reuse. Within-model reuse exploits shared prefixes or reusable prompt modules (Gim et al., 2024; Zheng et al., 2024); chunk-level reuse must restore missing dependencies (Yao et al., 2025; Yang et al., 2026). Across models, the reader must additionally accommodate a different representation. DroidSpeak combines layer reuse with selective recomputation for compatible models (Liu et al., 2026); SCD uses compact semantic encoding, low-rank reconstruction, and sparse patching for state transfer (Ma et al., 2026). Cross-scale methods include learned KV prediction (Horton et al., 2024), generation-trained latent communication (Chen et al., 2026), closed-form mappings after positional alignment (Heo et al., 2026), and head-local, attention-sensitive calibration (Qu et al., 2026). SCALE-KV combines source-state inputs, low-rank residual correction around a fixed base map, and jointly trained target-reader LoRA. Its learned interface coordinates cache construction with question-dependent reading rather than adapting the cache in isolation. The converter depends on the document alone, so subsequent questions reuse the same prefix.

Multi-agent communication. AutoGen, CAMEL, MetaGPT, and ChatDev exchange natural language, code, or structured messages (Wu et al., 2023; Li et al., 2023; Hong et al., 2024; Qian et al., 2024). Interaction pruning reduces redundant exchanges (Zhang et al., 2025); internal-state communication instead seeks to reuse sender computation. C2C projects and fuses source caches (Fu et al., 2026); LatentMAS shares latent reasoning states (Zou et al., 2025), Thought Communication extracts latent factors (Zheng et al., 2026), and StateBridge aligns hidden states (Peng et al., 2026). KVCOMM adjusts prefix-dependent cache offsets for overlapping contexts (Ye et al., 2026). We enable small-to-large document-state handoff through a cache interface optimized for the receiving model's answers and first-token latency.

3 Problem Formulation and Key Observation

3.1 Cross-Scale Document-State Reuse

Let D be a document, Q a question, and $Y = (y_1, \ldots, y_m)$ its answer. A frozen source model S produces document KV cache $C_S(D)$ and reusable intermediate states $H_S(D)$. A converter with parameters θ constructs a prefix cache for a larger target model T:

$$\widehat{C}_T(D) = C_\theta\big(C_S(D), H_S(D)\big). \tag{1}$$

The target uses this cache to process the question and generate the answer. With reader-adaptation parameters ϕ, the reuse path defines

$$p_{\theta,\phi}(Y \mid D, Q) = \prod_{t=1}^{m} p_{T,\phi}\big(y_t \mid \widehat{C}_T(D), Q, y_{<t}\big). \tag{2}$$

The converter depends only on D: one converted document prefix can serve multiple questions supplied later. The target computes each question and answer suffix from this prefix, without prefilling D again. We consider models with aligned document-token positions and compatible rotary positional conventions, while allowing different layer counts and representation dimensions. Online inputs are limited to the current document's source states, learned adaptation parameters, and ordinary target suffix computation. Native target document states and teacher distributions are used only for offline training or reference evaluation.

3.2 Reuse Value of Source States

A small model's standalone answer combines document encoding, question understanding, and generation. Cross-scale reuse separates these roles: the source supplies the document representation, and the target applies its stronger reader and generator to the adapted states. Table 1 compares this path with native source and target inference on the same tasks. On HotpotQA, FRAMES, and LongBench at 8–16K, reuse improves on native 4B by 6.20, 4.80, and 7.08 F1 points, respectively. The document is still processed by the small model, yet the resulting answers retain much of the native target's quality advantage. These comparisons establish the practical value of reusing source computation and motivate jointly learning the cache interface and target reading behavior.

3.3 Quality and Cost Objective

Let F_0 and $F(\theta, \phi)$ denote native-target and reuse answer F1 on the same evaluation set. Let $\tau(\theta, \phi)$ be mean time to first token (TTFT). Our design objective is low latency subject to a tolerated absolute F1 loss ϵ on the 0–100 scale:

$$\min_{\theta,\phi} \ \tau(\theta, \phi) \quad \text{subject to} \quad F_0 - F(\theta, \phi) \leq \epsilon. \tag{3}$$

The objective places answer quality and first-token latency on the same execution path. The generation losses in Section 4.2 train the converter and reader toward this goal. For a fresh request, latency includes source prefilling, state retention, cache conversion, and target question processing through the first token. A ready-state handoff begins with source KV and the required hidden states available, and includes subsequent state access, conversion, and target processing. These definitions distinguish the cost of constructing a reusable document state from the incremental cost of consuming one.

4 SCALE-KV: State-Augmented Cache Reuse

Figure 2 summarizes SCALE-KV. A fixed base mapping establishes the initial cross-model correspondence; state-augmented low-rank residuals and the target reader learn the resulting cache interface together. The source processes the complete document once, and the target computes only question and answer suffixes. Offline generation supervision couples cache construction with target reading, while online execution uses the reusable document prefix.

4.1 Base Mapping and State-Augmented Correction

Position-aligned mapping. For each target layer, we select a source layer by relative depth. In the 36-layer source and 40-layer target configuration, target layer l reads source layer $\mathrm{round}(35l/39)$. Source keys are mapped after removing source rotary position embeddings (RoPE), then target RoPE is restored at the original document positions. The base key mapping is fitted offline from paired source and target states and remains frozen during joint training. The value branch uses a fixed QR basis and decoder together with a trainable affine readout. Appendix A specifies the recorded dimensions, initialization, and optimization settings.

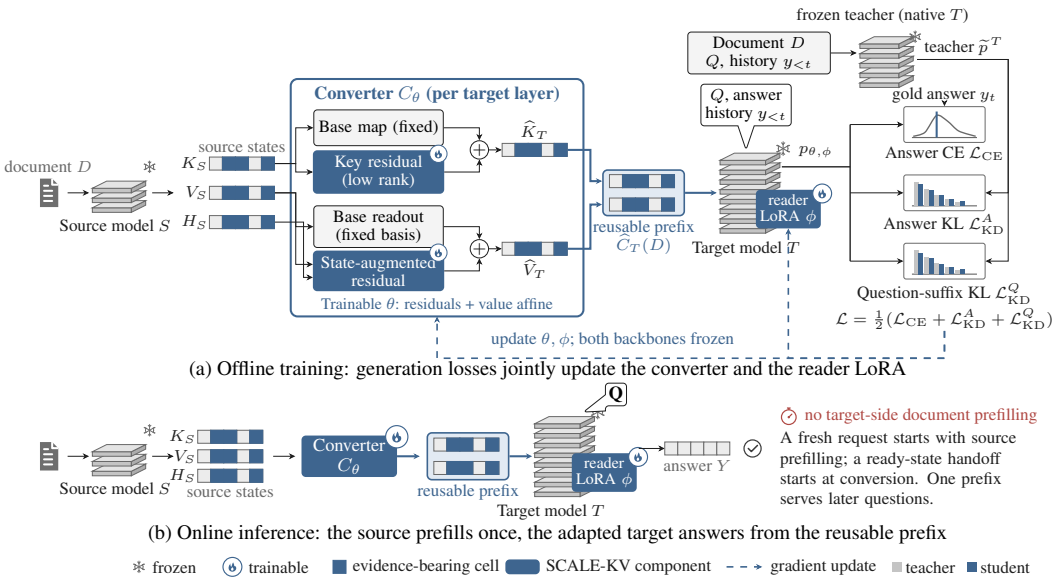

Figure 2: SCALE-KV. (a) Generation losses jointly update the converter and reader LoRA (dashed arrows); model backbones remain frozen. (b) The reusable document prefix supports target-side question answering.

Key correction. The key branch adds a low-rank nonlinear residual to the base map in the unrotated content space. Normalized source keys pass through two linear projections with a SiLU activation between them. The output projection is zero-initialized, so the residual does not perturb the base conversion at the start of training. The main key residual takes normalized source keys as its input.

Value-state augmentation. The value branch combines source values with the source layer's attention-input hidden states from the same document pass. Both inputs enter a shared low-rank correction, trained jointly with the target reader. For one aligned layer, the value update is

$$\widehat{V} = V_{\text{base}} + \mathrm{SiLU}(\bar{V}_s A_V + \bar{H}_s A_H + b)B. \tag{4}$$

Here V_{base} includes the fixed basis/decoder and trainable affine readout; $\bar{V}_s$ and $\bar{H}_s$ are normalized source values and hidden states. The projections A_V, A_H combine both inputs in a shared low-rank space, and B maps back to target value width. Normalization statistics remain fixed, and B is zero-initialized. The value residual can therefore use both the source model's value representation and its intermediate features while learning corrections around the base mapping. The resulting keys and values preserve the original document positions and prefix masks, providing a complete cache interface for target generation.

4.2 Generation-Oriented Joint Training

The purpose of conversion is accurate target generation, rather than numerical cache matching alone. A frozen native target teacher processes the document, question, and answer history offline. The reuse path predicts outputs at the same suffix positions from the converted document cache. Let $\mathcal{A}$ and $\mathcal{Q}$ index valid answer and question-suffix prediction positions. We jointly optimize

$$\mathcal{L} = \tfrac{1}{2}\mathcal{L}_{\mathrm{CE}} + \tfrac{1}{2}\mathcal{L}^A_{\mathrm{KD}} + \tfrac{1}{2}\mathcal{L}^Q_{\mathrm{KD}}, \tag{5}$$

where each term is normalized over its own positions:

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{|\mathcal{A}|}\sum_{t\in\mathcal{A}} \log p_t(y_t), \qquad \mathcal{L}^U_{\mathrm{KD}} = \frac{1}{|\mathcal{U}|}\sum_{t\in\mathcal{U}} D_{\mathrm{KL}}(\widetilde{p}^T_t \| \widetilde{p}_t), \quad U \in \{A, Q\}. \tag{6}$$

Cross-entropy anchors predictions to the reference answer. Answer distillation transfers the teacher's generation behavior, while question-suffix distillation trains the target to read converted prefixes when processing the question. The two KL terms operate on next-token output distributions at their respective positions. For storage efficiency, $\widetilde{p}^T_t$ retains the teacher's top-256 tokens and aggregates the remaining probability mass into one tail bucket; the student uses the same partition. Cross-entropy uses the full vocabulary. Together, these losses supervise cache construction through the answer-generation behavior that the cache must support.

Adapting the target reader. The target's attention query, key, value, and output projections receive rank-16 LoRA (Hu et al., 2021). These parameters act on the question and answer positions actually computed online. Gradients pass through the target to the converter, jointly optimizing key residuals, the value affine readout, value-state residuals, and reader LoRA. The source backbone, target base weights, and fixed base maps remain frozen; the teacher uses the native target without reader LoRA.

Training shares a converted prefix across QA pairs and uses 600 short-context updates followed by 900 mixed-length updates. Teacher forcing is used for optimization; free generation measures answer quality. Appendix A.7 details the schedule and valid-position masks.

4.3 Online Execution and Cost

Algorithm 1 gives the reuse path. A fresh request starts with source document prefilling; a ready-state handoff skips that step only when both source KV and the required hidden states are already available. After conversion, the source does not participate in token-by-token decoding. The adapted target processes the question and appends its own suffix KV during answer generation. Additional questions can start from independent views of the same converted document prefix without inheriting prior answers.

Algorithm 1 SCALE-KV inference

Input: D, Q; models S, T; converter θ, reader ϕ; optional source states $\mathcal{Z}_S$.
Output: Answer Y.

if $\mathcal{Z}_S$ unavailable: $\mathcal{Z}_S \leftarrow \mathrm{Prefill}_S(D)$.
for each target layer ℓ **do**
  $(K_s, V_s, H_s) \leftarrow \mathcal{Z}_S[j(\ell)]$.
  $K_s^0 \leftarrow \mathrm{RoPE}_S^{-1}(K_s)$.
  $(K_b^0, V_b) \leftarrow \mathrm{Base}_\ell(K_s^0, V_s)$.
  $\widehat{K}_\ell \leftarrow \mathrm{RoPE}_T(K_b^0 + \mathcal{R}_{K,\ell}(K_s^0))$.
  $\widehat{V}_\ell \leftarrow V_b + \mathcal{R}_{V,\ell}(V_s, H_s)$.
end for
$\widehat{C}_T(D) \leftarrow \mathrm{Assemble}(\{\widehat{K}_\ell, \widehat{V}_\ell\}_\ell, D)$.
return $\mathrm{Generate}_{T,\phi}(Q; \widehat{C}_T(D))$.

$j(\ell)$ pairs relative depths. Base and residual maps use θ; residual inputs use fixed normalization. Assembly preserves document positions and masks.

At fixed mapping dimensions and residual ranks, conversion operates independently over document positions and scales linearly with document length. A frozen base mapping supplies the initial correspondence, and the residual rank controls the learned correction cost. The converted cache has the target model's full prefix shape, so the receiver uses its ordinary question-processing and decoding path. Source states are obtained during the document pass and retained for conversion. For efficient base conversion, the split-bf16 implementation uses two low-precision matrix products with FP32 accumulation; Appendix B.7 reports its measured quality and latency. This execution structure replaces a target document transformer pass with position-wise state conversion.

5 Experiments

We evaluate cross-scale cache reuse through answer quality, first-token latency, and execution-stage cost. Source-model scaling and multi-question reuse examine how the same state interface supports different inference workloads; controlled component comparisons explain the roles of generation supervision, target-reader adaptation, and source-state inputs.

5.1 Experimental Setup

Models and evaluation. The main configuration uses Qwen3-4B as source and Qwen3-14B as target (Yang et al., 2025), with both backbones frozen. We evaluate HotpotQA (Yang et al., 2018), FRAMES (Krishna et al., 2025), and English LongBench QA (Bai et al., 2024) at 8–16K and 16–32K, plus LV-Eval (Yuan et al., 2024) at 16–32K. Appendix A gives test-set statistics and implementation settings.

Baselines and metrics. Native 4B and 14B provide source-only and full-target references without task-specific adaptation. We compare closed-form mapping (Heo et al., 2026), CacheBridge-HL (Qu et al., 2026), linear KV mapping with reader LoRA (A2), selective native recomputation, and text handoff. A1/A2 use the same generation objectives and reader-adaptation permissions as the controlled SCALE-KV comparisons. The text baseline generates question-independent document

Table 1: Answer quality of 4B-to-14B reuse (F1 ↑). Full evaluation panels; configurations are specified in Appendix A.9.

	8–16K contexts			16–32K contexts			
Method	HP	FR	LB	HP	FR	LB	LV
Native 4B	54.93	32.20	42.30	50.96	34.46	36.36	38.57
Native 14B	63.79	38.25	52.91	58.76	42.01	40.53	42.02
Closed-form map (Heo et al., 2026)	36.92	19.98	37.76	32.71	26.89	26.92	21.89
CacheBridge-HL (Qu et al., 2026)	21.44	11.43	19.02	23.83	12.50	16.10	7.17
Linear KV + reader LoRA (A2)	61.13	35.79	47.63	53.65	39.39	38.08	32.75
Selective recompute	59.81	30.91	48.46	53.61	36.90	36.47	30.56
Text handoff (1024)	30.98	25.87	41.89	25.81	27.26	26.79	16.27
SCALE-KV	61.13	37.00	49.38	52.78	37.30	40.65	34.57

HP: HotpotQA; FR: FRAMES; LB: LongBench; LV: LV-Eval.

notes with a fixed 1,024-token budget; the target reads the notes and question. All methods receive the same evaluation inputs and use greedy decoding with at most 64 answer tokens. Quality is normalized complete-output token F1. Appendix A.3 specifies the baseline configurations.

All timed methods use the same test requests, one warmed A800, and Hugging Face/SDPA. Table 2 pairs SCALE-KV quality and TTFT: fresh requests include source prefilling, and handoffs start with resident source KV and required hidden states. Speedups are medians of per-question TTFT ratios. Appendix B.7 reports the numerical-implementation comparison.

5.2 Main Results and the Quality–Efficiency Trade-off

Answer quality. Table 1 shows that SCALE-KV preserves much of the native target's answering capability. On the three 8–16K panels, gains over native 4B are 6.20, 4.80, and 7.08 F1 points; gaps to native 14B are 2.66, 1.25, and 3.53. Relative to the linear-plus-reader baseline A2, SCALE-KV matches HP8 at the displayed precision and raises FRAMES and LongBench F1 by 1.21 and 1.75 points. It also outperforms closed-form mapping, CacheBridge-HL, and the 1,024-token text handoff on every panel. These comparisons establish a practical alternative to target-side document re-encoding.

Longer contexts. At 16–32K, SCALE-KV improves over native 4B by 1.82, 2.84, and 4.29 F1 on HotpotQA, FRAMES, and LongBench. LongBench reaches 40.65 F1 versus native 14B's 40.53. Against A2, SCALE-KV is higher on LongBench and LV-Eval by 2.57 and 1.82 points, while A2 is higher on HotpotQA and FRAMES by 0.87 and 2.09. Table 1 gives every panel; Appendix B.6 compares the variants.

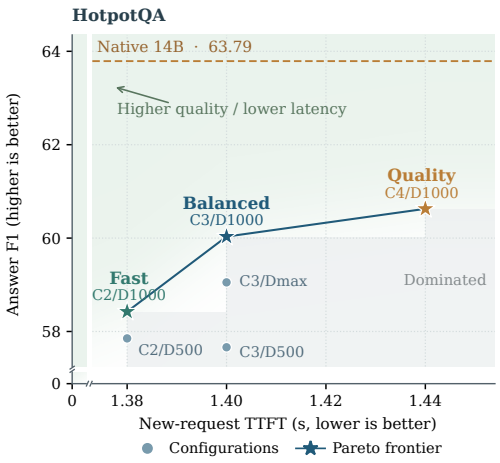

Figure 3: Measured capacity–quality trade-offs on HotpotQA.

Figure 3 examines six capacity–data configurations. C2, C3, and C4 use K/V ranks 16/32, 32/64, and 64/128; D specifies the training-context count, with Dmax corresponding to 1,941 contexts. At 1,000 contexts, these capacities achieve 58.42/60.03/60.63 F1 at measured fresh-request TTFTs of 1.38/1.40/1.44 seconds. Moving from C2 to C3 gains 1.61 F1 for about 0.02 seconds, while C4 provides a higher-quality operating choice at a modest additional conversion cost. The three starred configurations are non-dominated among the six evaluated settings. The native-14B reference line shows the remaining quality gap, and axis breaks expose the local configuration differences. This comparison makes the capacity trade-off explicit: the converter can allocate more computation to quality recovery while retaining the source-prefill/target-generation division.

Table 2: Paired quality and TTFT speedups of SCALE-KV. N14: native 14B; E/H: fresh/handoff. Settings: Appendix A.4.

	F1		Speedup (×)	
Set	N14	SCALE-KV	E	H
HP8	66.24	60.53	1.66	5.63
FR8	42.06	39.27	1.65	5.38
FR16	46.49	41.14	1.64	6.49

SCALE-KV TTFT (ms)	E	H
HP8 (327 questions)	1400	416
FR8 (59 questions)	1216	373
FR16 (64 questions)	2631	655

Table 3: Source-size extension: 1.7B→14B F1. 8/16 denote 8–16K/16–32K.

Set	1.7B	14B	SCALE-KV
HP8	38.96	63.79	51.28
FR8	23.96	38.25	30.47
LB8	30.61	52.91	35.96
HP16	36.87	58.76	43.26
FR16	22.30	42.01	33.58
LB16	16.73	40.53	35.96
LV16	27.69	42.02	22.24

Complete results: Appendix B.1.

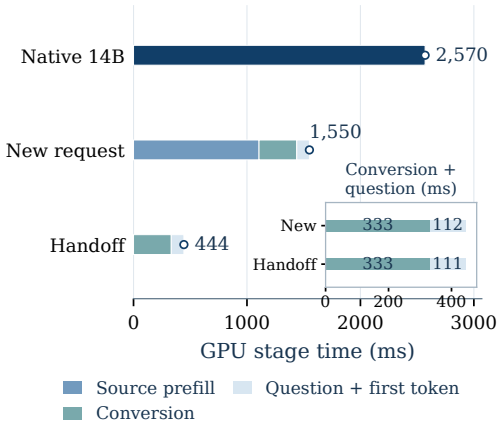

Figure 4: GPU-stage costs over 8–32K, with an enlarged view of conversion and target-suffix processing.

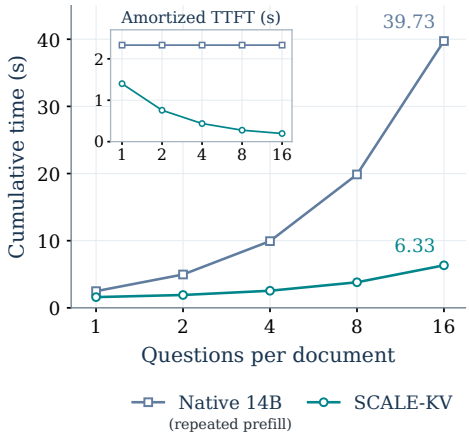

Figure 5: Multi-question generation time and amortized TTFT (inset), with repeated native-14B prefilling as reference.

5.3 Efficiency and Cost Breakdown

Overall efficiency. The paired SCALE-KV evaluations achieve 1.64–1.66× fresh-request and 5.38–6.49× handoff speedups (Table 2), showing that the saved target document pass outweighs source-prefill and conversion cost. The split-bf16 implementation of SCALE-KV reaches peak speedups of 1.80×/9.61× (Appendix B.7).

Cost breakdown. Figure 4 separates source prefilling (1.106 s), conversion/assembly (0.333 s), and target-suffix processing (0.112 s). The fresh-request stage total is 1.550 s versus 2.570 s for native 14B; handoff takes 0.444 s. Appendix B.7 gives the length-specific GPU profiles.

Comparison with randomized model selection. At SCALE-KV's mean fresh-request TTFT, a request-independent mixture of native 4B and 14B yields expected F1 of 59.12, 33.54, and 41.06 on HP8, FR8, and FR16. SCALE-KV reaches 60.53, 39.27, and 41.14: gains of 1.41, 5.73, and 0.08 points at matched average latency. Appendix D.6 derives this analytical reference from the paired measurements.

5.4 Source-Model Scaling

With a 1.7B source, SCALE-KV reaches 51.28/30.47 F1 on HP8/FR8 versus native 1.7B's 38.96/23.96 (Table 3). Its optimized implementation reaches peak fresh/handoff speedups of 3.34×/9.85×. Appendix B.1 specifies this configuration.

Table 4: Controlled comparisons under the C3 recipe (8–16K). Converter counts are trainable parameters, excluding reader LoRA.

Configuration	Converter (M)	HP F1	FR F1	Macro F1
SCALE-KV / C3	19.706	59.05	36.39	47.72
KV-only (A1)	19.709	59.84	35.62	47.73
Linear KV + reader LoRA (A2)	83.968	61.13	35.79	48.46
No reader LoRA, retrained (A3)	19.706	58.46	34.51	46.49
No answer CE (A4)	19.706	55.54	35.98	45.76

5.5 Amortizing Document-State Construction

SCALE-KV constructs one document prefix and answers each question from an independent suffix (Figure 5). Across 1–16 questions, its amortized TTFT falls from 1.400 to 0.196 s. At 16 questions, measured total generation takes 6.331 s versus 39.734 s for native 14B with repeated full prefilling (Appendix A.6).

5.6 Ablations of Components and Training Objectives

Table 4 isolates component changes under the shared C3 control recipe. Its normalization is refitted jointly for the controls; Table 1 uses SCALE-KV's fixed checkpoint normalization. A1 matches converter capacity, and A2 learns linear KV maps with reader LoRA. A3 retrains without reader LoRA; A4 removes answer CE with both KL weights fixed. Other data members and update settings are shared (Appendix A.9).

Generation supervision and target reading. Answer CE raises HP8/FR8 F1 by 3.52/0.41 points over A4, with the answer-KL and question-suffix-KL weights fixed. Joint reader adaptation raises the corresponding scores by 0.59/1.87 points over the retrained A3 converter. Both components improve the observed means on both tasks. Under the additional HPZ recipe, retraining without reader LoRA reduces F1 from 61.01/33.49 to 58.92/31.35. These comparisons support optimizing cache construction through generation and adapting how the target consumes that cache.

Source-state inputs and parameter efficiency. C3 provides a state-augmented interface with 19.706M trainable converter parameters, 76.5% fewer than A2's 83.968M. Including reader LoRA gives 40.678M versus 104.940M trainable adaptation parameters, a 61.2% reduction. C3 reaches 47.72 Macro F1, compared with 47.73 for capacity-matched A1 and 48.46 for A2. The low-rank structure therefore reduces the trainable parameter budget while supporting source-state inputs and jointly adapted reading. Appendix C.3 gives the paired intervals.

Capacity and data budget. The 11 completed capacity–data configurations in Appendix B.5 provide a broader view of the compact converter. With 1,000 contexts and a fixed update budget, increasing K/V ranks from 16/32 to 32/64 raises Macro F1 from 47.10 to 48.42; ranks 64/128 reach 48.59. Within C3, expanding the training subset from 100 to 1,000 contexts raises HP8/FR8 F1 from 55.00/33.86 to 60.03/36.80. These comparisons identify practical operating choices for allocating training coverage and residual capacity while preserving document-only conversion and the frozen model backbones. The paired intervals in Appendix C.3 complement these point estimates.

6 Conclusion

SCALE-KV reuses small-model document computation through a state-augmented low-rank converter and a jointly adapted target reader. A reusable document-only prefix lets the target answer subsequent questions without repeating full-document prefilling. Long-document QA demonstrates gains over source-only answering, while paired evaluations show lower first-token latency on fresh requests and ready-state handoffs. The controlled results highlight generation supervision, reader adaptation, and a compact trainable cache interface.

AI USE STATEMENT

Generative AI tools were used to assist with research brainstorming, methodological discussion, code development, and the preparation and revision of the manuscript and figures. All AI-assisted research ideas, implementations, experimental designs, analyses, and written content were independently reviewed and verified by the authors. Experimental results and conclusions were derived from author-conducted experiments and analyses. The authors take full responsibility for the final content of this work.

REFERENCES

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming {Throughput-Latency} tradeoff in {LLM} inference with {Sarathi-Serve}. In *18th USENIX symposium on operating systems design and implementation (OSDI 24)*, pp. 117–134, 2024.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 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, 2024.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, et al. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, 2025.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling, 2024. *URL https://arxiv. org/abs/2406.02069*, 2024.

Siyi Chen, Xiaoyan Zhang, Meng Wu, Jonathan Tremblay, Valts Blukis, Stan Birchfield, Rene Vidal, Alvaro Velasquez, Sijia Liu, and Qing Qu. See what i see, know what i think: Dense latent communication across heterogeneous agents. *arXiv preprint arXiv:2606.13594*, 2026.

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

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models. In *International Conference on Learning Representations*, volume 2026, pp. 43130–43158, 2026.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. *Proceedings of Machine Learning and Systems*, 6:325–338, 2024.

Taekyung Heo, Rasoul Shafipour, Ritchie Zhao, Maximilian Golub, Mohammad Mahdi Kamani, Ritika Borkar, Makesh Tarun Chandran, Pantea Zardoshti, and Bita Darvish Rouhani. Cross-model kv cache transfer in llm families: A closed-form linear mapping for prefill reuse. *arXiv preprint arXiv:2608.03893*, 2026.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

Maxwell Horton, Qingqing Cao, Chenfan Sun, Yanzi Jin, Sachin Mehta, Mohammad Rastegari, and Moin Nabi. Kv prediction for improved time to first token. *arXiv preprint arXiv:2410.08391*, 2024.

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.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 13358–13376, 2023.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, 2024.

Dongwon Jo, Jiwon Song, Yulhwa Kim, and Jae-Joon Kim. Fastkv: Decoupling of context reduction and kv cache compression for prefill-decoding acceleration. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 32167–32186, 2026.

Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4745–4759, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Jingyu Liu, Beidi Chen, and Ce Zhang. Speculative prefill: Turbocharging ttft with lightweight and training-free token importance estimation. *arXiv preprint arXiv:2502.02789*, 2025.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024a.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, et al. Cachegen: Kv cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pp. 38–56, 2024b.

Yuhan Liu, Yuyang Huang, Jiayi Yao, Shaoting Feng, Zhuohan Gu, Kuntai Du, Hanchen Li, Yihua Cheng, Junchen Jiang, Shan Lu, et al. {DroidSpeak}:{KV} cache sharing across fine-tuned model variants. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*, pp. 319–338, 2026.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024c.

Qianli Ma, Zhiqing Tang, Hanshuai Cui, Zhi Yao, and Weijia Jia. Semantic cache distillation: Efficient state transfer via reuse and selective patching. *arXiv preprint arXiv:2606.07684*, 2026.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, et al. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981, 2024.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 118–132. IEEE, 2024.

Yanwen Peng, Delvin Ce Zhang, Xi Wang, and Nikolaos Aletras. Statebridge: Training-free hidden-state alignment for latent communication in llm multi-agent systems. *arXiv preprint arXiv:2608.13317*, 2026.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 15174–15186, 2024.

Aurick Qiao, Zhewei Yao, Samyam Rajbhandari, and Yuxiong He. Swiftkv: Fast prefill-optimized inference with knowledge-preserving model transformation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25745–25764, 2025.

Xingyu Qu, Siyuan Lu, Zhiyu Chen, Sheng Wang, and Tao Lin. Cachebridge: Efficient cross-model kv cache transfer. *arXiv preprint arXiv:2609.00891*, 2026.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation, 2023. *URL https://arxiv. org/abs/2308.08155*, 2308, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. Kvlink: Accelerating large language models via efficient kv cache reuse. *Advances in Neural Information Processing Systems*, 38: 133797–133824, 2026.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

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. In *Proceedings of the twentieth European conference on computer systems*, pp. 94–109, 2025.

Hancheng Ye, Zhengqi Gao, Mingyuan Ma, Qinsi Wang, Yuzhe Fu, Ming-Yu Chung, Yueqian Lin, Zhijian Liu, Jianyi Zhang, Danyang Zhuo, et al. Kvcomm: Online cross-context kv-cache communication for efficient llm-based multi-agent systems. *Advances in Neural Information Processing Systems*, 38:17882–17928, 2026.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. Helmet: How to evaluate long-context models effectively and thoroughly. In *International Conference on Learning Representations*, volume 2025, pp. 98914–98965, 2025.

Tao Yuan, Xuefei Ning, Dong Zhou, Zhijie Yang, Shiyao Li, Minghui Zhuang, Zheyue Tan, Zhuyu Yao, Dahua Lin, Boxun Li, et al. Lv-eval: A balanced long-context benchmark with 5 length levels up to 256k. *arXiv preprint arXiv:2402.05136*, 2024.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *International Conference on Learning Representations*, volume 2025, pp. 75389–75428, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in neural information processing systems*, 36:34661–34710, 2023.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought communication in multiagent collaboration. *Advances in Neural Information Processing Systems*, 38:123389–123418, 2026.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. {DistServe}: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX symposium on operating systems design and implementation (OSDI 24)*, pp. 193–210, 2024.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, et al. Latent collaboration in multi-agent systems. *arXiv preprint arXiv:2511.20639*, 2025.

CONTENTS OF TECHNICAL APPENDICES

A Implementation and Evaluation Details 15
- A.1 Main Configuration 15
- A.2 Test Panels and Sample Counts 15
- A.3 Baseline Implementations and Scoring 15
- A.4 Timing and Paired Quality–Latency Evaluation 16
- A.5 Evaluation Scope 16
- A.6 Protocol for Multiple Questions per Document 16
- A.7 Training Objective and Converter Details 16
- A.8 Online Execution Details 17
- A.9 Controlled Configurations 17

B Supplementary Experimental Results 18
- B.1 1.7B Source Results 18
- B.2 Training Update Cost 18
- B.3 Quality Across All Seven Evaluation Sets 18
- B.4 Component Ablations and Controlled Length Probes 18
- B.5 Adaptation Capacity and Training Data 19
- B.6 Training Recipes and Longer Contexts 19
- B.7 Timing Records and Numerical Implementation 20

C Score Differences and Sensitivity Analysis 22
- C.1 Paired Evaluation 22
- C.2 Score Aggregation 22
- C.3 Score Differences for Ablation and Tuning 22

D Supplementary Quality and Compute Measurements 23
- D.1 Ablation and Data-Size Results 23
- D.2 Paired Quality and Timing 23
- D.3 Training Costs of Ablation and Tuning Configurations 24
- D.4 Online Process Memory 24
- D.5 Text-Handoff Budget Sensitivity 24
- D.6 Matched-Latency Randomized Routing 25

A Implementation and Evaluation Details

A.1 Main Configuration

SCALE-KV uses K/V residual ranks 32/64 and reader-LoRA rank 16. Table A1 summarizes the architecture and optimization settings. The fixed base mapping initializes the cross-model correspondence. The affine value readout and residuals are subsequently optimized together with the target reader through the generation objective.

Table A1: Main implementation and training configuration.

Item	Main configuration
Models	Qwen3-4B→14B; 36 source and 40 target layers; KV width 1024
Layer pairing	Source layer $\mathrm{round}(35l/39)$ for target layer $l = 0, \ldots, 39$
Key branch	Remove RoPE; frozen affine map plus rank-32 SiLU residual; restore target RoPE
Value branch	Fixed QR basis/decoder, trainable affine readout, rank-64 residual from source values and hidden states
Initialization	Zero terminal K/V residual matrices; value readout initialized from pretraining
Target reader	Rank-16 LoRA on q/k/v/o; scale 1; 20,971,520 parameters
Training objective	Answer CE, answer KL, and question-suffix KL, each weighted 0.5; teacher top-256 plus tail bucket
Update schedule	Four contexts per update, up to three QA pairs each; 600 short-context then 900 mixed-length updates
Optimizer	AdamW, 20 warmup steps, cosine decay, parameter-group gradient clipping
Generation	Greedy, at most 64 new tokens; training answers truncated to 63 tokens plus a stopping token
Precision	bf16 backbones; FP32/FP64 reference conversion; split-bf16 acceleration

A.2 Test Panels and Sample Counts

Table A2 lists the seven evaluation panels. Questions sharing one document form a context cluster. We report both question counts, which determine question-mean F1, and cluster counts, which describe the number of shared document contexts.

Table A2: Evaluation sets and sample counts.

Evaluation set	Questions	Clusters	Construction
HotpotQA 8–16K	651	100	Packed HotpotQA dev
FRAMES 8–16K	191	176	FRAMES articles
LongBench 8–16K	97	74	English LongBench QA
HotpotQA 16–32K	619	50	Packed HotpotQA dev
FRAMES 16–32K	269	226	FRAMES articles
LongBench 16–32K	84	57	English LongBench QA
LV-Eval 16–32K	347	233	LV-Eval English 16k

A.3 Baseline Implementations and Scoring

All baselines are evaluated on the same test panels with common decoding and scoring settings. Closed-form mapping uses fitted cross-model correspondences; CacheBridge-HL is the head-local mapping configuration evaluated in Table 1 (Qu et al., 2026). A2 learns layerwise linear KV maps with rank-16 reader LoRA, under the controlled generation objective. Selective recomputation computes the first target layer fully, selects 10% of tokens by state deviation, and recomputes their subsequent target states.

The text baseline uses native 4B to generate document notes without seeing the question, and native 14B to answer from the notes and question. The primary note budget is fixed at 1,024 tokens; Appendix D.5 reports 512/2,048-token settings. Notes are reused only for

matching document and configuration fingerprints. Table 1 includes complete panels for all eight distinct methods; A2 is listed once.

All methods use greedy decoding with at most 64 answer tokens. We apply SQuAD-style normalization to the complete output and take the maximum token F1 over its reference answers on a 0–100 scale. The LongBench scorer uses the last-row-per-ID reference lookup. The full-panel and paired timing sample counts are listed in Tables A2 and 2. In the paired runs, decoding stops on token ID 151643 or 151645, or after 64 new tokens. The target receives the document prefix before the question suffix; each new question begins from that prefix without previous answer states.

A.4 Timing and Paired Quality–Latency Evaluation

Paired evaluation uses one NVIDIA A800-SXM4-80GB with both models and the converter resident. The recorded environment is Python 3.12.12, PyTorch 2.10.0+cu128, Transformers 5.3.0, and HF SDPA. Models run in bf16; reference conversion uses FP32 key affine operations, FP64 value projection, and FP32 reader LoRA. The paired SCALE-KV checkpoint has SHA-256 prefix `1de3b0fd4234`. Batch size and concurrency are both one, after three distinct warm-up contexts.

HP8, FR8, and FR16 each contain 50 sampled context clusters, retaining 327, 59, and 64 questions. Each request has three timing repetitions. TTFT is the mean of per-question repetition medians; speedup is the median of per-question native-14B/SCALE-KV ratios. The outer clock is device-synchronized and starts with token IDs ready; tokenization is measured separately. E includes source prefilling for SCALE-KV. H begins with its source KV/hidden states resident, or with the native 4B prefix resident; native 14B constructs its prefix in both scenarios. Independent process profiles record memory separately from these timing repetitions.

A.5 Evaluation Scope

The test panels cover long-document short-answer QA and document-state handoff within the Qwen3 family at 8–16K and 16–32K. Table 1 reports all seven task–length panels; paired quality and latency use the common test subsets listed in Appendix A.4.

A.6 Protocol for Multiple Questions per Document

The HP8 multi-question workload evaluates 1, 2, 4, 8, and 16 questions per document. SCALE-KV constructs and retains one converted document prefix, and each question starts from an independent suffix. The native-14B reference performs a complete document prefill for each question. Figure 5 reports cumulative generation time and document-state construction cost amortized across questions.

Cumulative generation time includes document preparation, question processing, and complete answer generation. Amortized TTFT is the sum of one-time preparation and all first-token processing costs divided by the question count. At one question, native 14B and SCALE-KV take 2.483352 and 1.599435 seconds for complete generation, with TTFTs of 2.331066 and 1.399946 seconds. At 16 questions, cumulative generation takes 39.733627 and 6.331275 seconds, and amortized TTFT is 2.331066 and 0.196216 seconds, respectively.

A.7 Training Objective and Converter Details

Equations 5–6 assign weight 0.5 to answer CE, answer KL, and question-suffix KL, each averaged over its own valid positions. Teacher and student use the same top-256 token set and aggregated tail bucket for KL; CE uses the full vocabulary. The teacher supplies native document states and predictions offline.

For target layer ℓ, the source layer is $j(\ell) = \mathrm{round}(35\ell/39)$. Let K_s^0 denote its unrotated source keys. The key branch can be written, before restoring target RoPE, as

$$\widehat{K}^0 = K_s^0 M_{K,\ell} + \mathbf{1} b_{K,\ell}^\top + \mathcal{R}_{K,\ell}(\bar{K}_s^0).$$

Here the base affine map is fixed. The trainable residual $\mathcal{R}_{K,\ell}$ applies two low-rank projections separated by SiLU; its terminal projection is zero-initialized. With $d_s = d_t = 1024$ and $r_K = 32$,

Table A3: SCALE-KV and its controlled settings. CE/A-KL/Q-KL list loss weights.

Run	Value residual input	K/V ranks	Reader rank	CE/A-KL/Q-KL
SCALE-KV	$V + H$	32/64	16	0.5/0.5/0.5
C3	$V + H$	32/64	16	0.5/0.5/0.5
A1	V	32/144	16	0.5/0.5/0.5
A2	Linear KV map	—	16	0.5/0.5/0.5
A3	$V + H$	32/64	0	0.5/0.5/0.5
A4	$V + H$	32/64	16	0/0.5/0.5

$M_{K,\ell}$ is 1024×1024, $A_{K,\ell}$ is 1024×32, and $B_{K,\ell}$ is 32×1024. Target RoPE is then applied at the original document positions.

For a document of n tokens, V_s has shape $n \times 1024$ and H_s has shape $n \times d_H$. In Equation 4, $A_V \in \mathbb{R}^{1024\times 64}$, $A_H \in \mathbb{R}^{d_H\times 64}$, $b \in \mathbb{R}^{64}$, and $B \in \mathbb{R}^{64\times 1024}$. The projected V/H features therefore share an $n \times 64$ correction space, and the output has the target value shape $n \times 1024$. The fixed QR basis/decoder and affine value readout construct V_{base}; only the affine readout and residual projections are updated. Fixed normalization statistics define the source inputs, and zero-initialized terminal residual projections preserve the base map at initialization.

SCALE-KV uses 600 short-context updates followed by 900 mixed-length updates, four contexts per update and up to three QA pairs per context. Stage-one peak learning rates are 10^{-4} for the key residual and value readout, and 2×10^{-4} for the value residual and reader LoRA. Stage two halves these rates. Each stage uses 20 warm-up steps and cosine decay to 0.1 of its peak. Multiple QA pairs share the converted training prefix. Rank-16 LoRA acts on target q/k/v/o projections for question and answer positions.

A.8 Online Execution Details

Algorithm 1 takes a document, a question, the frozen source and target models, trained converter and reader parameters, and optionally ready source states. Conversion writes the target KV at the original document positions and assembles matching attention masks. The target then processes the question and generates the answer with its adapted attention projections.

For another question on the same document, retain the converted document prefix and create a fresh suffix view. Fresh requests include document-state construction; handoffs begin with the required source states resident. These measurements count conversion on the request path. When the target is known before a question arrives, the same document-only converter can run in advance and retain the target prefix for later questions. At fixed dimensions and ranks, conversion is linear in document length.

A.9 Controlled Configurations

SCALE-KV denotes the cache-conversion and reader-adaptation method throughout. Table A3 defines its controlled variants. C3 uses the main optimization schedule with shared normalization refitted for each adaptation-budget condition; the main evaluations use the fixed normalization accompanying the SCALE-KV checkpoint. All controlled runs inherit the first-stage state and complete 600+900 updates with seed 20260920. The numerical-implementation control in Table B5 uses the 1,000-update reference also studied in Appendix B.4.

C2/C3/C4 use K/V residual ranks 16/32, 32/64, and 64/128. D100, D200, D500, D1000, and Dmax denote nested adaptation-context subsets, with Dmax=1,941. A1 matches C3's trainable converter count within 3,200 parameters. A2 learns the full layerwise linear KV maps and also trains the reader; it is the linear baseline in Table 1. Table D1 gives all 15 complete runs. The component labels specify structural changes, while the precision columns in Table B5 specify the numerical implementation.

B Supplementary Experimental Results

B.1 1.7B Source Results

Table B1 reports the 1.7B-to-14B reuse configuration. It improves on native 1.7B on the HotpotQA, FRAMES, and LongBench panels, with peak fresh-request and handoff speedups of $3.34\times$ and $9.85\times$. Native 1.7B reaches a peak fresh-request speedup of $5.45\times$. Table 1 reports the corresponding 4B-to-14B test results.

Table B1: Results with a 1.7B source (F1). Differences retain the recorded precision within this tuning group.

Evaluation set	Native 1.7B	SCALE-KV (1.7B→14B)	Difference
HotpotQA 8–16K	38.96	51.28	+12.33
FRAMES 8–16K	23.96	30.47	+6.51
LongBench 8–16K	30.61	35.96	+5.35
HotpotQA 16–32K	36.87	43.26	+6.38
FRAMES 16–32K	22.30	33.58	+11.28
LongBench 16–32K	16.73	35.96	+19.23
LV-Eval 16–32K	27.69	22.24	-5.45

B.2 Training Update Cost

Table B2 reports GPU time and peak memory for the parameter-update loop. Teacher generation, base-map fitting, model loading, checkpoint writing, and evaluation are outside this measurement. Per-run training and evaluation wall times appear in Table D4.

Table B2: Update-loop compute and peak training memory.

Configuration	Updates	GPU hours	Peak GiB
SCALE-KV (1000 updates)	1000	5.38	70.71
SCALE-KV (1500 updates)	1500	7.54	70.74
SCALE-KV K augmentation	1500	7.77	70.97
SCALE-KV K/V augmentation	1500	8.17	71.37

B.3 Quality Across All Seven Evaluation Sets

Figure B1 visualizes the native-4B, state-reuse, and native-14B scores from Table 1 across all seven panels. The view highlights the reuse gains on HotpotQA, FRAMES, and LongBench and the variation across task–length settings. Table C1 reports the corresponding F1 differences.

B.4 Component Ablations and Controlled Length Probes

Table B3 and Figure B2 examine reader adaptation, training objectives, residual initialization, value rank, selective recomputation, and numerical precision under the HPZ recipe. The full configuration reaches 61.01 F1 on HP8. The reader and objective comparisons explain the gains from joint generation-oriented adaptation; the remaining variants characterize alternative cache-construction choices. A separate padding probe on 33 contexts produced no consistent score trend.

Reader adaptation. Reader LoRA improves HotpotQA 8–16K F1 by 2.09 points relative to a retrained no-LoRA converter. With the trained converter held fixed, disabling LoRA at inference changes F1 from 61.01 to 56.96. Applying the same reuse-trained LoRA to native 14B prefilling gives 63.15 F1, compared with 63.79 for the unadapted native model. These settings examine reader adaptation with retraining, inference-only removal, and a native cache, respectively.

Training objectives. Under HPZ, the teacher-only HPK configuration scores 57.63 F1 on HP8; it removes answer CE and increases the answer-KL weight from 0.5 to 1. The FR8 score of 34.31

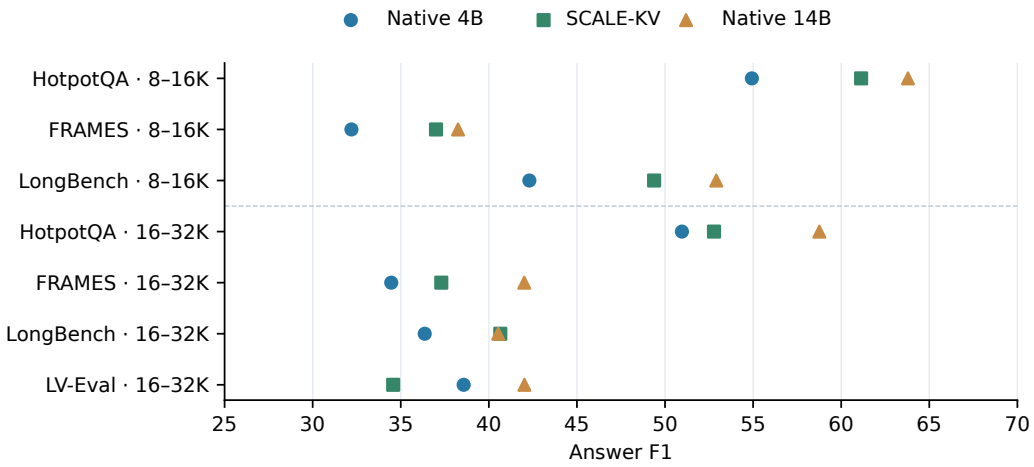

Figure B1: Measured state-reuse quality across tasks and lengths, using the scores from Table 1.

Table B3: HPZ-reference ablations on HotpotQA 8–16K. Differences are relative to HPZ.

Configuration	F1	Difference
HPZ reference	61.01	0
No reader LoRA, retrained	58.92	-2.09
LoRA disabled at inference	56.96	-4.05
Teacher-only objective, $\alpha = 0$	57.63	-3.38
No LoRA and $\alpha = 0$	56.35	-4.66
Warm-started key residual	59.57	-1.43
Value residual rank 256	60.68	-0.33
10% selective native recomputation	59.81	-1.20
split-bf16 conversion	61.12	+0.11

belongs to the separate HPKN configuration, which also removes reader LoRA. These are reported as distinct settings. The single-loss C3/A4 comparison in Table 4 instead holds both KL weights at 0.5.

Controlled recipe. C3, A1, A2, A3, and A4 share the recorded context members, update schedule, and data-specific statistics policy. A1 varies source-state input at matched converter capacity; A2 uses full linear KV maps; A3 removes reader LoRA and retrains; A4 removes answer CE. Table D1 and Table D4 give their complete quality and cost records.

B.5 Adaptation Capacity and Training Data

We evaluate eleven capacity–data configurations. C2/C3/C4 use K/V ranks 16/32, 32/64, and 64/128 at 500, 1,000, and 1,941 adaptation contexts; C3 also uses 100 and 200 contexts. Inputs, reader adaptation, loss weights, and update counts follow the common controlled recipe. Figure B3 reports the measured grid; Figure 3 presents six representative settings with measured latency.

At 1,000 contexts, HP8 F1 rises from 58.42 to 60.03 to 60.63 as capacity grows; FR8 gives 35.77, 36.80, and 36.55. C4/D1000 has the highest observed Macro F1, 48.59, among these eleven grid settings. C3 improves from 500 to 1,000 contexts by 2.37/1.76 F1 on HP8/FR8, while Dmax reaches 59.05/36.39. The grid exposes practical choices between adaptation capacity and data coverage.

B.6 Training Recipes and Longer Contexts

Figure B4 compares HPZ, B0_600, and the main HPU+ recipe across seven panels. HPU+ reaches 61.13/37.00/49.38 F1 on HP8/FR8/LB8, leading the three recipes on HP8 and LB8. Relative to HPZ, HPU+ gains about 3.5 F1 on FR8 and 3.9 on LV16. HPZ leads on HP16 and LB16, while

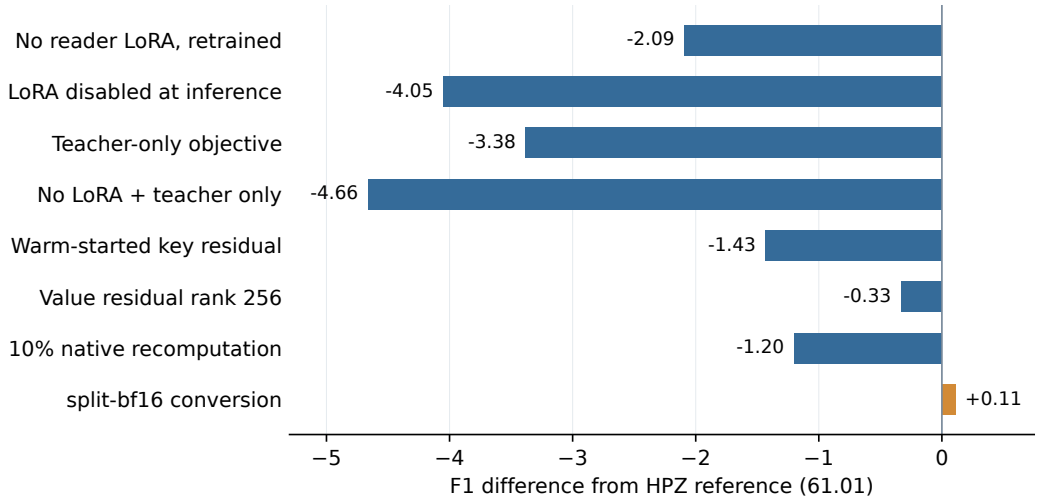

Figure B2: Measured component and objective differences on HotpotQA 8–16K, relative to the HPZ reference.

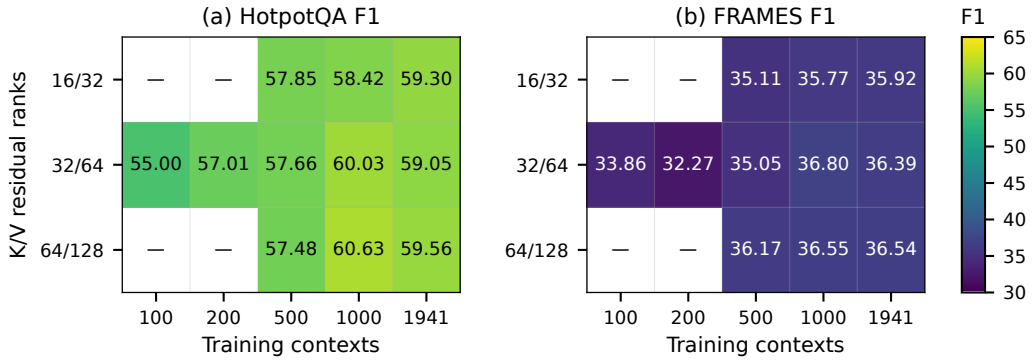

Figure B3: F1 across eleven evaluated capacity–data configurations. Dashes denote unmeasured combinations.

B0_600 leads on FR8, FR16, and LV16. The recipe comparison shows how the mixed-data adaptation distributes quality across tasks and context lengths.

At 16–32K, SCALE-KV obtains 52.78, 37.30, 40.65, and 34.57 F1 on HotpotQA, FRAMES, LongBench, and LV-Eval, respectively. Its gains over native 4B are 1.82, 2.84, and 4.29 points on the first three tasks, and LongBench reaches the native-14B quality level. Table C1 includes the complete differences for every panel. The two length bins contain different question sets.

Table B4 compares training and capacity variants with SCALE-KV. Joint key augmentation combines key rank 128, hidden-state input, and per-head gains; it improves HotpotQA F1 by 0.72 points at 8–16K and 1.09 at 16–32K. Combining key augmentation with value rank 256 reaches 62.01 F1 at 8–16K, an improvement of 0.88 over HPU+. These results identify additional cache-construction capacity as a useful axis for quality improvement.

The base SCALE-KV recipe reaches 61.13 F1 at 8–16K. Increasing value rank alone gives 53.40 at 16–32K, while joint key augmentation gives 53.87. The combined key/value variant provides the strongest short-panel score in this comparison. Each row specifies the training or cache-construction change relative to the main recipe.

B.7 Timing Records and Numerical Implementation

Split-bf16 improves the runtime of cache conversion (Table B5). In the two length bins, fresh-request speedup rises from $1.67\times/1.64\times$ with reference conversion to $1.80\times/1.75\times$ with split-bf16. With

Table B4: HotpotQA variants and F1 differences relative to SCALE-KV.

Configuration	8–16K F1	Δ	16–32K F1	Δ
SCALE-KV	61.13	+0.00	52.78	+0.00
Long-only continuation	60.56	-0.57	52.55	-0.23
Doubled long-context exposure	59.25	-1.88	52.55	-0.23
Value rank 256	59.91	-1.22	53.40	+0.63
Joint key augmentation	61.85	+0.72	53.87	+1.09
Joint key + value rank 256	62.01	+0.88	53.34	+0.57

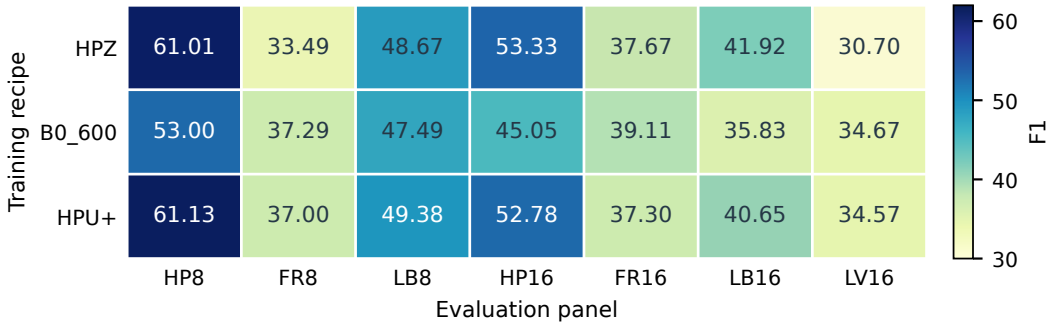

Figure B4: Training-recipe F1 across seven task–length panels. HP/FR/LB/LV denote HotpotQA/FRAMES/LongBench/LV-Eval; 8 and 16 denote 8–16K and 16–32K.

source states ready, the split-bf16 path reaches handoff speedups of $7.83\times$ and $9.61\times$. The table identifies each timed configuration and its corresponding TTFT.

Table B5: SCALE-KV numerical-implementation control: 50 documents per length bin and three repetitions. Native-4B handoff measurements appear in Table D3.

Length	Timed configuration	TTFT (s)	Fresh speedup	Handoff speedup
8–16K	Native 14B	2.506	1.00×	1.00×
8–16K	Native 4B	1.037	2.38×	—
8–16K	SCALE-KV (reference)	1.483	1.67×	5.78×
8–16K	SCALE-KV (split-bf16)	1.367	1.80×	7.83×
16–32K	Native 14B	5.545	1.00×	1.00×
16–32K	Native 4B	2.579	2.13×	—
16–32K	SCALE-KV (reference)	3.375	1.64×	7.00×
16–32K	SCALE-KV (split-bf16)	3.162	1.75×	9.61×

Split-bf16 decomposes mapping weights into two low-precision products and accumulates the results in FP32. On HotpotQA 8–16K, split-bf16 changes F1 by $+0.11$ in the 1,000-update precision control and by -0.27 for the main SCALE-KV checkpoint. The implementation accelerates conversion while retaining comparable answer scores. Appendix D.2 reports the paired quality and latency measurements.

Figure B5 decomposes the GPU-stage costs by length. Fresh-request stage totals are 1.372/2.629 seconds at 8–16K/16–32K, with conversion and assembly taking 0.304/0.504 seconds. Ready-state stage totals are 0.409/0.655 seconds, compared with native-14B prompt-stage times of 2.284/4.298 seconds. The profiles show the computation removed when document states are reused and identify source prefilling as the largest remaining fresh-request stage.

The main adaptation update loop uses 7.54 GPU hours and 70.74 GiB peak training memory. Table B2 defines the update-loop boundary; Table D4 adds per-run training and evaluation wall times. Online complete-answer times and co-resident process-memory profiles appear in Appendix D.

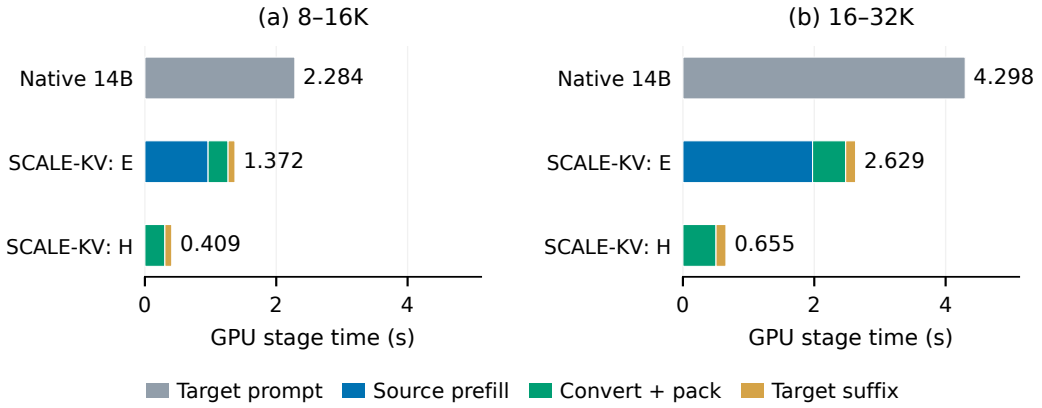

Figure B5: Recorded GPU-stage costs by context length. Native prompt processing includes the target suffix; reuse paths separate source prefilling, conversion/assembly, and target suffix processing. Bar labels give GPU-stage totals on a shared time scale; E/H denote fresh requests/handoffs.

C Score Differences and Sensitivity Analysis

C.1 Paired Evaluation

Table C1 reports question-mean F1 differences between the reuse path and each native model on the same evaluation panels. Positive differences indicate gains from the adapted source-state path. The table covers all seven task–length combinations in the main quality comparison.

Table C1: Mean F1 differences computed at the available precision of the quality table.

Evaluation set	SCALE-KV minus native 4B	SCALE-KV minus native 14B
HotpotQA 8–16K	+6.20	-2.66
FRAMES 8–16K	+4.80	-1.25
LongBench 8–16K	+7.08	-3.53
HotpotQA 16–32K	+1.82	-5.98
FRAMES 16–32K	+2.84	-4.71
LongBench 16–32K	+4.29	+0.12
LV-Eval 16–32K	-4.00	-7.45

C.2 Score Aggregation

Panel F1 is the mean of per-question scores over all questions in that panel. For the two-task comparisons in Table D1, Macro F1 is the unweighted mean of HP8 and FR8 F1. Context clusters identify questions sharing a document; their counts accompany the panel statistics in Table A2. Appendix A.5 summarizes the test tasks.

C.3 Score Differences for Ablation and Tuning

We pair methods by question ID and resample complete context clusters with replacement. Each replicate computes question-mean F1 over all questions in the sampled clusters. We use 10,000 replicates, seed 20260914, lexicographically ordered cluster IDs, and NumPy's default random generator; the interval is the 2.5th–97.5th percentile. All compared checkpoints use the recorded training seed 20260920. Table C2 quantifies evaluation-sample uncertainty for those fixed checkpoints.

Table C2: Cluster-paired F1 differences and 95% percentile intervals.

Comparison (first minus second)	Set	ΔF1	95% interval
C3_D1000 – C3_D500	HP8	+2.370	[+0.20, +4.64]
C3_D1000 – C3_D500	FR8	+1.756	[-0.23, +3.98]
C3 – C3_D1000	HP8	-0.980	[-2.93, +0.94]
C3 – C3_D1000	FR8	-0.415	[-2.80, +1.90]
C3 – A1	HP8	-0.790	[-2.49, +0.83]
C3 – A1	FR8	+0.772	[-2.01, +3.44]
A2 – C3	HP8	+2.078	[-0.17, +4.17]
A2 – C3	FR8	-0.593	[-4.06, +2.85]
C3 – A3	HP8	+0.591	[-1.02, +2.24]
C3 – A3	FR8	+1.875	[-0.51, +4.43]
C3 – A4	HP8	+3.517	[+1.67, +5.45]
C3 – A4	FR8	+0.412	[-1.88, +2.65]

Table D1: Complete HP8/FR8 training and control results (0–100 scale).

Run	HP8 F1	HP8 EM	FR8 F1	FR8 EM	Macro F1
A1_Dmax	59.842	43.472	35.616	21.990	47.729
A2_Dmax	61.130	45.315	35.794	23.037	48.462
A3_Dmax	58.461	41.782	34.513	19.895	46.487
A4_Dmax	55.535	40.399	35.975	20.942	45.755
C2_D1000	58.425	42.243	35.768	19.895	47.097
C2_D500	57.852	41.321	35.115	20.419	46.483
C2_Dmax	59.296	42.550	35.918	21.990	47.607
C3_D100	55.001	38.556	33.860	19.372	44.431
C3_D1000	60.032	43.779	36.802	20.942	48.417
C3_D200	57.006	41.014	32.267	19.372	44.636
C3_D500	57.662	41.782	35.047	20.942	46.354
C3_Dmax	59.052	43.011	36.388	22.513	47.720
C4_D1000	60.627	44.393	36.553	20.942	48.590
C4_D500	57.483	41.167	36.174	21.990	46.829
C4_Dmax	59.562	42.550	36.543	23.560	48.052

D Supplementary Quality and Compute Measurements

D.1 Ablation and Data-Size Results

Table D1 reports all 15 complete training runs, each evaluated on 651 HP8 and 191 FR8 questions. Macro F1 is the unweighted mean of the two panel F1 scores. C2/C3/C4 label residual capacity; A1–A4 identify the controlled changes in Table A3.

C3 uses 19.706M trainable converter parameters, 76.5% fewer than A2's 83.968M. At Dmax, C3/A1/A2 achieve Macro F1 of 47.720/47.729/48.462. Adding reader adaptation to A3 raises Macro F1 by 1.233 points, and adding answer CE to A4 raises it by 1.964 points. Table 1 and the paired timing tables report the main SCALE-KV results.

D.2 Paired Quality and Timing

The paired timing subsets contain 50 clusters each: HP8 has 327 questions, FR8 59, and FR16 64. E denotes a fresh request beginning with the raw document; H denotes a handoff with source KV and the required hidden states ready. Native 14B constructs its own document prefix in both settings. N4 and N14 denote native 4B and 14B. F1 and exact match are measured on these timing subsets. TTFT and complete-answer time are reported in milliseconds.

Table D2: Fresh requests (E). Speedup is the median per-question TTFT ratio to native 14B.

Set	Method	F1	EM	TTFT (ms)	Complete (ms)	Speedup
HP8	N4	55.886	40.979	976.0	1123.0	2.388×
HP8	N14	66.236	48.930	2331.1	2483.4	1.000×
HP8	SCALE-KV	60.531	44.037	1399.9	1599.4	1.655×
FR8	N4	29.517	15.254	835.8	1075.0	2.425×
FR8	N14	42.061	28.814	2021.3	2265.3	1.000×
FR8	SCALE-KV	39.270	27.119	1215.7	1548.0	1.648×
FR16	N4	38.891	31.250	1964.7	2217.8	2.226×
FR16	N14	46.491	37.500	4298.5	4533.4	1.000×
FR16	SCALE-KV	41.136	31.250	2631.0	2961.3	1.642×

Table D3: Ready-state handoff (H), using the actual executed paths.

Set	Method	F1	EM	TTFT (ms)	Complete (ms)	Speedup
HP8	N4	55.315	40.367	85.2	230.7	27.669×
HP8	N14	66.236	48.930	2331.7	2482.9	1.000×
HP8	SCALE-KV	60.531	44.037	415.6	613.9	5.635×
FR8	N4	29.652	15.254	78.0	313.7	25.752×
FR8	N14	42.061	28.814	2020.8	2265.0	1.000×
FR8	SCALE-KV	39.270	27.119	373.3	702.5	5.380×
FR16	N4	38.748	31.250	128.8	404.1	33.064×
FR16	N14	46.491	37.500	4298.9	4544.5	1.000×
FR16	SCALE-KV	41.136	31.250	655.3	1007.6	6.489×

On HP8, FR8, and FR16, the reuse path achieves 60.531, 39.270, and 41.136 F1 with fresh-request speedups of $1.655\times$, $1.648\times$, and $1.642\times$. Ready-state handoff retains these scores and reaches $5.635\times$, $5.380\times$, and $6.489\times$. Tables D2–D3 include the native-model scores for each execution path.

Native 4B uses full-prompt prefilling in E and a resident document prefix in H. Tables D2–D3 report the measurements for each executed path.

D.3 Training Costs of Ablation and Tuning Configurations

Each run uses one A800. Training-stage wall time includes loading, statistics fitting, updates, and checkpoint writing, excluding evaluation and shared teacher-library/static-map preparation. Table D4 reports all 15 runs. C3 and A1 have 19.706M and 19.709M trainable converter parameters; A2 has 83.968M. Reader LoRA adds 20,971,520 trainable parameters except in A3. Including reader adaptation, C3 and A2 have approximately 40.678M and 104.940M trainable parameters, respectively; C3 reduces this combined budget by 61.2%.

D.4 Online Process Memory

Table D5 reports peak allocated process memory with the source model, target model, and converter co-resident. The values are 53.66/53.73 GiB for HP8/FR8 and 71.12 GiB for FR16, specifying the memory footprint of the evaluated execution configuration.

D.5 Text-Handoff Budget Sensitivity

The text-handoff baseline uses note budgets of 512, 1,024, and 2,048 tokens; Table 1 reports the 1,024-token configuration, and Table D6 gives all three budgets. Native 4B sees only the document when generating notes; native 14B receives the notes and question. All three budgets are evaluated on the full seven panels. Larger notes improve F1 across the panels, while SCALE-KV avoids autoregressive note generation and retains a full document-position cache interface.

Table D4: Measured training-stage wall time and peak memory.

Run	Training (hours)	Peak memory (GiB)
A1_Dmax	7.56	65.18
A2_Dmax	7.11	65.92
A3_Dmax	7.52	70.50
A4_Dmax	7.60	70.74
C2_D1000	7.60	70.10
C2_D500	7.60	70.72
C2_Dmax	7.55	70.66
C3_D100	7.53	68.89
C3_D1000	7.72	70.20
C3_D200	7.62	70.05
C3_D500	7.68	70.83
C3_Dmax	7.73	70.74
C4_D1000	7.85	70.42
C4_D500	7.89	71.04
C4_Dmax	7.83	70.90

Table D5: Peak allocated memory with co-resident models and converter.

Set	SCALE-KV peak allocated (GiB)
HP8	53.66
FR8	53.73
FR16	71.12

Table D6: Text handoff F1 under three document-note budgets.

Panel	512 tokens	1024 tokens	2048 tokens
HP8	26.87	30.98	40.21
FR8	17.34	25.87	28.09
LB8	34.42	41.89	44.86
HP16	21.90	25.81	29.43
FR16	21.29	27.26	31.35
LB16	21.44	26.79	32.36
LV16	10.67	16.27	18.94

Table D7: Randomized-routing expectation at SCALE-KV's measured mean fresh-request TTFT. p is the native-14B selection probability; ΔF1 is SCALE-KV minus the routing expectation.

Set	TTFT (ms)	p (%)	Routing F1	SCALE-KV F1	ΔF1
HP8	1400	31.28	59.124	60.531	+1.407
FR8	1216	32.05	33.537	39.270	+5.733
FR16	2631	28.55	41.061	41.136	+0.075

D.6 Matched-Latency Randomized Routing

For the paired fresh requests in Table D2, let t_4, t_{14}, t_S and F_4, F_{14}, F_S denote measured mean TTFT and F1. A request-independent router selecting native 14B with probability $p = (t_S - t_4)/(t_{14} - t_4)$ and native 4B otherwise matches SCALE-KV's mean latency. Its expected F1 is $F_R = (1-p)F_4 + pF_{14}$.

Table D7 gives analytical expectations from the unrounded measured endpoints, with both models resident and zero routing overhead. SCALE-KV gains 1.407/5.733 F1 on HP8/FR8 and reaches 41.136 versus 41.061 on FR16 at the same mean TTFT.