CACHEMORPH: ASYMMETRIC KV TRANSFER FOR EFFICIENT MODEL SWITCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-model LLM systems often move a request between models, and routers switch models as load changes. Each switch has a hidden cost: the receiving model must re-prefill a shared context that may be tens of thousands of tokens long. Reusing the source model's key–value (KV) cache could remove this cost. However, same-model reuse methods do not directly support such switches, while cross-model cache alignment can still incur substantial accuracy losses. Aligning KV representations alone does not ensure accurate receiver outputs. We show why this gap matters: key errors change where attention reads, value errors change the content it retrieves, and query heads sharing KV respond differently to the same errors. Based on these insights, we introduce CacheMorph, a framework that enables the receiver to skip prefill of shared history by translating KV caches across models. Its asymmetric mapper predicts keys and values separately, conditioning each value on its mapped key. A lightweight, query-adaptive reader then adjusts how each head accesses the shared cache. Both components are trained to match the frozen receiver's native-cache output distribution. A streaming pipeline overlaps cache transfer and mapping with ongoing computation. CacheMorph improves accuracy over prior cross-model KV reuse methods, especially across model families. At 32K tokens, it achieves up to a $4.1\times$ time-to-first-token speedup over receiver re-prefill. In RAG cascades, it delivers approximately $2\times$ speedup and over 50% lower estimated token cost while retaining over 90% of re-prefill accuracy.

1 INTRODUCTION

Model routing assigns simple requests to inexpensive models, escalates demanding stages to stronger models, and switches models with serving load. Each switch carries a hidden cost: a receiving model B must prefill history already processed by a source model A when their context states cannot be reused. In RAG, agent workflows, and multi-turn dialogue (Figure 1), shared prefixes often span thousands to tens of thousands of tokens, so repeated prefilling erodes routing gains.

KV reuse can eliminate repeated prefill, but existing approaches have distinct constraints (Table 1). CacheBlend and RelayCaching selectively recompute caches to repair context discrepancies within the same model (Yao et al., 2025; Geng et al., 2026). Cache-to-Cache learns cross-model projections but

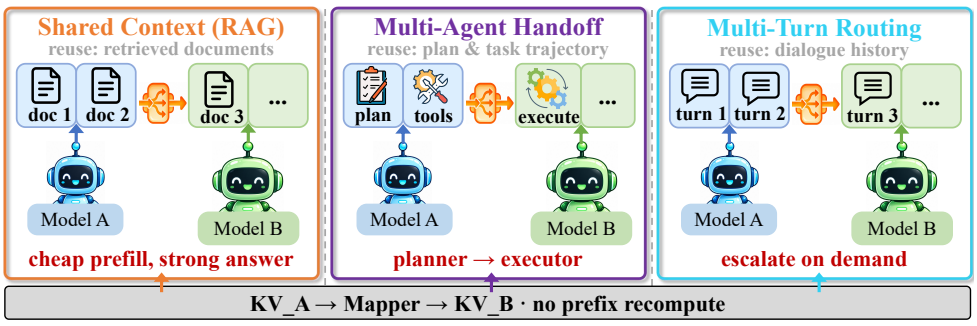


Figure 1: KV reuse at model-switching boundaries in RAG, agent workflows, and multi-turn dialogue.

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.

Table 1: **Capabilities of representative KV-reuse methods for model switching.**

Method	Cross-scale	Cross-family	Unequal KV heads	Skip prefill	No-training
CacheBlend (Yao et al., 2025)	×	×	×	◐	✓
KVCOMM (Ye et al., 2025)	×	×	×	◐	✓
RelayCaching (Geng et al., 2026)	×	×	×	◐	✓
Ridge (Heo et al., 2026)	✓	×	×	✓	✓
CacheBridge (Qu et al., 2026)	✓	×	×	✓	✓
C2C (Fu et al., 2026)	✓	✓	✓	×	×
MoT (woo Lee et al., 2026)	✓	✓	✓	◐	×
CacheMorph	✓	✓	✓	✓	×

✓: yes; ×: no; ◐: partial reuse. No-training: no gradient optimization; calibration may still be required.

fuses them with native receiver prefix KV, retaining receiver-side prefilling (Fu et al., 2026). Closed-form linear calibration directly constructs receiver caches (Heo et al., 2026; Qu et al., 2026). Its objective fits cache representations, including attention-sensitivity-weighted residuals in CacheBridge, rather than directly supervising how the receiver consumes transferred caches during continuation.

We propose CacheMorph, which is a high-performance framework for cross-model KV reuse. Our analysis shows that key errors change where attention reads and value errors change the content it retrieves, and query heads sharing KV respond differently to the same errors. An asymmetric *mapper* uses independent K/V source-feature routes and conditions value prediction on mapped keys. A low-rank *reader* combines group-shared and head-specific query corrections to adapt cache access. Output supervision through the frozen receiver trains these components, with auxiliary losses that distinguish addressing and content errors during mapper training. A streaming pipeline overlaps cache transfer, mapping, and target consumption to reduce handoff cost.

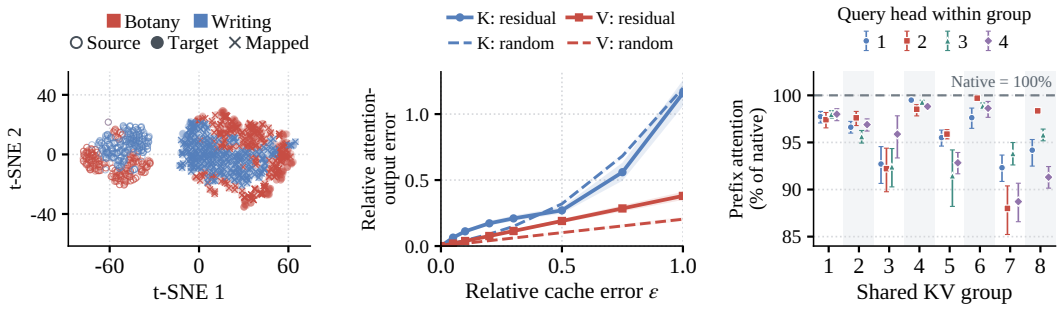
Experiments show that CacheMorph consistently outperforms prior cache-transfer methods, with especially strong gains across model families. It achieves up to a $4.1\times$ TTFT speedup at 32K prefixes and reduces latency and estimated token cost in RAG cascades and math agents while largely preserving accuracy.

Our main contributions are as follows:

- **KV asymmetry and consumption-aware transfer.** We identify that key and value errors affect attention differently and query heads consume the transferred KV cache differently. Building on these findings, we develop an asymmetric mapper with key-conditioned value prediction and a query-adaptive reader, both trained to preserve the receiver's downstream behavior.
- **Backend optimization and large-scale evaluation.** We develop a dependency-aware streaming backend that overlaps cache transfer and mapping with model computation. Across six model pairs, 14 tasks, and routed RAG and reasoning workflows, extensive experiments demonstrate consistent gains in transfer quality and end-to-end efficiency.

2 Related Work

Same-model context reuse. vLLM and SGLang manage reusable prefixes through paging and radix indexing; ChunkAttention and Hydragen accelerate shared-prefix attention, while Mooncake organizes cache-centric serving (Kwon et al., 2023; Zheng et al., 2024; Ye et al., 2024; Juravsky et al., 2024; Qin et al., 2025). Modular caching extends reuse to non-prefix segments through position handling, independent encoding, or learned links (Gim et al., 2024; Ma et al., 2025; Yang et al., 2025b;a). CacheBlend and EPIC selectively recompute tokens, while KVCOMM and RelayCaching repair context-dependent differences in multi-agent pipelines (Yao et al., 2025; Hu et al., 2025; Ye et al., 2025; Geng et al., 2026). These methods reuse states within a common model parameterization rather than translate between independently parameterized models.



(a) KV correspondence (b) K/V sensitivity (c) Head heterogeneity

Figure 2: Motivation diagnostics for Qwen3-8B → Llama-3.1-8B-Instruct at layer 17: (a) KV correspondence; (b) K/V sensitivity; (c) query-head heterogeneity. See Appendix D.

Cross-model KV transfer. Within a shared backbone, reuse exploits selective recomputation, restricted adapter effects, layer selection (Liu et al., 2026; Greenewald et al., 2025; Shi et al., 2026), or shared prefill modules (Woo et al., 2026a;b). SCD learns compact codes and sparse patches for weight-mismatched models with a common architecture (Ma et al., 2026). Cache-to-Cache fuses projected KV with receiver-native caches, and LCF compresses this communication channel (Fu et al., 2026; Rossi et al., 2026). Shared-space alignment trains per-model adapters (Dery et al., 2026); MoT combines token-dependent translators with target-trajectory supervision (woo Lee et al., 2026). IAM transfers attention patterns rather than KV states (Zhao et al., 2025). Closest to our setting, Heo et al. (2026) fit per-head ridge maps and relate downstream retention to attention-sensitive residuals. CacheBridge restricts source features and calibrates with receiver attention sensitivity (Qu et al., 2026). These results support broader evidence of linear representation transfer (Huh et al., 2024; Chen et al., 2025; Bello et al., 2025). Unlike these methods, CacheMorph combines asymmetric K/V translation with query-dependent reading: group-shared and head-specific corrections adapt queries while preserving shared physical KV.

3 Motivation: Asymmetric Errors and Head-Specific Consumption

Cross-model KV reuse can be formulated as regression: fit a closed-form affine map or train an MLP on paired caches to approximate the receiver's native KV. Yet low reconstruction error need not imply accurate attention. Three questions remain: *where* the mapping places the cache, *where* residual errors are amplified, and *whether* different query heads can use the transferred cache equally well.

Where does the mapping place the cache? Figure 2(a) compares two contexts with distinct topics in a shared t-SNE embedding. Affine-mapped source KV lies near the corresponding native target clouds, while the two topics remain separated. This qualitative alignment supports cross-family cache mapping and directs the next diagnostics toward the remaining *residual errors*.

Where are residual errors amplified? To test whether error norm alone determines functional impact, we perturb native receiver caches in external K or V along affine-transfer residuals measured on held-out documents. The relative perturbation norms are matched, with the current query, local KV, mask, and positions all fixed. Figure 2(b) measures relative attention-output error after the original output projection. At relative cache error 0.2, K and V residuals yield mean output errors of 0.172 and 0.076, respectively, a roughly $2.3\times$ difference. Random perturbations of the same magnitude produce different responses from the residual directions. Functional impact thus depends on the error's *direction and role*, beyond its norm: key errors alter position selection through query projection, whereas value errors directly enter the weighted content. This distinction motivates separate mapping branches and functional losses for addressing and content in Section 4.2.

Which heads bear these errors? Figure 2(c) replaces the external prefix keys with the frozen affine prediction, retains native queries and local keys, and measures the total attention weight each query head assigns to the prefix. The four heads in KV group 8 retain 94.2%, 98.4%, 95.8%, and 91.3% of their native prefix attention, respectively. This 7.1-percentage-point gap suggests that a shared-cache

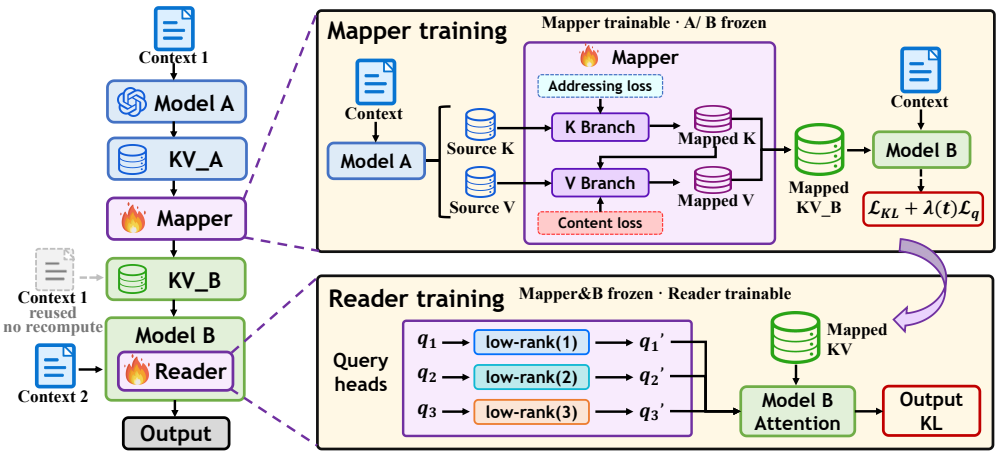


Figure 3: CacheMorph overview: asymmetric KV mapping and query-adaptive reading.

correction need not benefit every head equally. These differences motivate adapting cache access to each head's own query during reading. Future queries are unavailable when the cache is mapped.

4 METHOD

When generation switches from model A to model B, CacheMorph aims to preserve B's behavior without repeating its history prefill. A mapper translates A's cache at handoff and then a reader adapts access to that cache using B's current queries (Figure 3).

4.1 PROBLEM SETUP AND FUNCTIONAL OBJECTIVE

Given LLM models A, B, history D has already produced source cache $\mathcal{C}_A(D)$. Model B continues on new input U. Instead of prefilling D again, a mapper F_θ constructs target KV, and a reader R_ϕ adjusts how B consumes it:

$$\widehat{\mathcal{C}}_B = F_\theta(\mathcal{C}_A(D), a), \qquad \Delta\bar{q}_t = R_\phi(\bar{q}_t). \tag{1}$$

Here a aligns tokens, and R_ϕ corrects B's current pre-RoPE query $\bar{q}_t$ for external-cache reading. Layer and head indices are omitted. With target history length T_B, model B uses the mapped prefix $\widehat{\mathcal{C}}_{B,<T_B}$ and reads $U^+ = (\text{last}_B(D), U)$, replaying one boundary token before continuing generation. Online handoff needs only source KV and alignment metadata. Appendix H.2 specifies details.

Our objective is defined by the receiver's output. At prediction position t, let p_t^{nat} and $p_t^{\theta,\phi}$ be B's next-token distributions with native and transferred caches on the same token history. We train the mapper and reader parameters θ, ϕ to minimize

$$\mathcal{L}_{\text{out}} = \mathbb{E}\left[\sum_{t\in\mathcal{T}_{\text{sup}}} \rho_t \,\text{KL}\left(p_t^{\text{nat}} \| p_t^{\theta,\phi}\right)\right], \qquad \rho_t \geq 0, \quad \sum_{t\in\mathcal{T}_{\text{sup}}} \rho_t = 1. \tag{2}$$

Here $\mathcal{T}_{\text{sup}}$ is the set of supervised prediction positions, and ρ_t assigns a normalized weight to each position. The expectation is over training handoff examples, and KL denotes forward KL divergence over the full vocabulary. Appendix I describes the supervision protocol. Native B provides downstream supervision. Cache reconstruction supplies an initialization, but coordinate error does not describe how B uses the cache. We next examine that relationship through attention. For one layer, query head, and position, let d_B denote the target model's attention-head dimension. The head computes attention weights $\alpha = \text{softmax}(qK^\top/\sqrt{d_B})$ and contributes $y = \alpha V W_O$ to the attention output, where W_O is its output-projection block. Fix the query, local KV, positions, and mask. Reference $\circ$ substitutes native external KV while retaining these fixed inputs. It need not equal the fully native trajectory. Let $\mathcal{E}$ index external tokens, $\delta K = \widehat{K} - K^\circ$, and $\delta V = \widehat{V} - V^\circ$. The

change in this head's contribution, $\Delta y = \widehat{y} - y^\circ$, has the first-order expansion

$$\Delta y = \left[\underbrace{\frac{1}{\sqrt{d_B}} \sum_{i \in \mathcal{E}} \alpha_i^\circ (q \delta k_i^\top)(v_i^\circ - o^\circ)}_{\text{addressing error}} + \underbrace{\sum_{i \in \mathcal{E}} \alpha_i^\circ \delta v_i}_{\text{content error}} \right] W_O + O(\varepsilon^2), \tag{3}$$

where o° is the reference readout and $\varepsilon = \|\delta K\|_F + \|\delta V\|_F$. Queries and keys are post-RoPE. Key errors redistribute attention through query projections and value contrasts. Value errors directly change the weighted content. This motivates separate prediction branches and functional auxiliaries for keys and values.

The two effects remain coupled. With mapped-key attention $\widehat{\alpha}$ and reference values V_{all}° stacked over external and local positions, the exact attention error is

$$\Delta y = \left[(\widehat{\alpha} - \alpha^\circ) V_{\text{all}}^\circ + \widehat{\alpha}_{\mathcal{E}} (\widehat{V} - V^\circ) \right] W_O. \tag{4}$$

The content term is weighted by attention induced by $\widehat{K}$. Value prediction must therefore account for the addressing produced by the key branch. We implement this dependency by conditioning V prediction on the mapped key at the same position. Appendix C derives both identities and the resulting dependence of the value gradient on mapped keys.

4.2 Mapper: asymmetric, key-conditioned KV prediction

For each target layer and physical KV head, the K and V branches independently select source layers and heads. Each branch combines a calibrated affine prediction with a nonlinear residual; value prediction additionally conditions on the mapped key. At aligned position t, omitting the target layer and head indices,

$$\widehat{k}_t = f_\theta^K(x_t^K), \qquad \widehat{v}_t = f_\theta^V(x_t^V, \widehat{k}_t). \tag{5}$$

Here x_t^K, x_t^V collect the selected source features. For each branch $c \in \{K, V\}$, we factor the whole selected-source affine matrix as

$$z_t^c = \eta^c + (\widetilde{x}_t^c W^{c,\text{in}}) W^{c,\text{out}}, \qquad W^{c,\text{in}} \in \mathbb{R}^{D_c \times r}, \quad W^{c,\text{out}} \in \mathbb{R}^{r \times d_B}. \tag{6}$$

Here z_t^c is the affine prediction before adding the nonlinear residual, and $\eta^c \in \mathbb{R}^{d_B}$ is a learned bias. The learned factors $W^{c,\text{in}}$ and $W^{c,\text{out}}$ map the source features through an r-dimensional intermediate space to the target head, constraining the effective affine matrix to rank at most r. The row vector $\widetilde{x}_t^c$ concatenates centered, route-weighted source blocks; D_c is their total width and d_B the target head dimension. Both training and inference apply the two products directly, without materializing a dense $D_c \times d_B$ weight. Thus the affine parameter and per-token multiply-add counts decrease from $D_c d_B$ to $r(D_c + d_B)$, excluding the bias.

We initialize the factors by truncated SVD of a trained, ridge-initialized dense mapper (Heo et al., 2026), then jointly refine factors, routes, biases, and residuals. Source supports and calibration means remain fixed. The key-conditioned value branch and query-independent, positionwise mapping are preserved. Source keys are unrotated before mapping; target RoPE is applied before storage (Su et al., 2024). Appendix H.1 specifies initialization, and Appendix I describes recovery training.

Teacher and student use the frozen target with native and mapped caches on identical token histories, but generate their own internal states. The mapper minimizes output KL with early functional supervision:

$$\mathcal{L}_{\text{map}} = \mathcal{L}_{\text{out}} + \lambda_{\text{aux}}(\tau) \left(\mathcal{L}_{\text{consume}} + \lambda_K \mathcal{L}_K + \lambda_V \mathcal{L}_V \right). \tag{7}$$

Here $\mathcal{L}_{\text{map}}$ is the mapper training loss, and $\mathcal{L}_{\text{out}}$ is the output KL loss defined in Equation 2. The training step is denoted by τ; $\lambda_{\text{aux}}(\tau)$ controls the overall auxiliary-loss weight and decays to zero early in training, while λ_K and λ_V are fixed weights for the key and value losses.

$\mathcal{L}_{\text{consume}}$ aligns attention outputs after W_O on the actual trajectories. At fixed student queries and local KV, $\mathcal{L}_K$ compares native-key and mapped-key attention by JS divergence, while $\mathcal{L}_V$ compares projected value readouts under fixed reference attention. The V auxiliary stops gradients into the K branch; output and consumption losses retain coupled gradients. After the auxiliary weight reaches zero, training uses output KL alone. Appendices I–J describe the training procedure.

4.3 READER: LOW-RANK CORRECTION FOR EACH QUERY HEAD

In GQA, heads j, j' share the same key error δk_i, but their logit perturbations differ by $(q_j - q_{j'})\delta k_i^\top / \sqrt{d_B}$. An error nearly orthogonal to one query can strongly affect another. Their attention weights, reference readouts, and output projections in Equation 3 also differ. A shared cache correction therefore need not benefit every head equally.

The mapper cannot adapt to future queries. Once a query is available, however, we can adjust its external logits toward the reference: $(q_j + \Delta q_j)\widehat{k}_i^\top \approx q_j (k_i^\circ)^\top$. This supplies head-specific addressing without duplicating KV. Query changes redistribute existing values. Value prediction remains with the mapper.

To limit parameters and execution cost, we use low-rank factors (Hu et al., 2022). Suppressing layer and position indices, let $\bar{q}_j$ be head j's query after native normalization, if present, and before RoPE, and let $g(j)$ denote its KV group. The reader R_ϕ in Equation 1 computes

$$\Delta \bar{q}_j = R^{\mathrm{grp}}_{\phi, g(j)}(\bar{q}_j) + R^{\mathrm{head}}_{\phi, j}(\bar{q}_j). \tag{8}$$

Here $R^{\mathrm{grp}}_{\phi,g(j)}$ shares a transformation among heads reading the same keys, while $R^{\mathrm{head}}_{\phi,j}$ adds an individual adjustment. Both branches depend on the current query and use linear maps of ranks at most four and one, respectively, initialized to produce zero corrections. Appendix K describes reader execution and training. Each correction thus lies in an at-most-five-dimensional space, whose residual limitation is characterized in Appendix C.

The corrected query is $q_j^{\mathrm{ext}} = \mathrm{RoPE}_B(\bar{q}_j + \Delta \bar{q}_j, t)$, and q_j denotes the original post-RoPE query. After applying RoPE, the corrected query scores the transferred keys, while the original query scores the local keys. A single softmax over both sets of scores produces attention weights, which aggregate the corresponding values.

$$o'_j = \mathrm{softmax}\left(\frac{[q_j^{\mathrm{ext}} \widehat{K}^\top_{g(j)},\ q_j (K^{\mathrm{loc}}_{g(j)})^\top]}{\sqrt{d_B}} \right) \begin{bmatrix} \widehat{V}_{g(j)} \\ V^{\mathrm{loc}}_{g(j)} \end{bmatrix}. \tag{9}$$

The original $W_{O,j}$ blocks project and combine the readouts. Each head can change external selection and external/local attention mass while physical KV remains shared.

4.4 DEPENDENCY-AWARE STREAMING HANDOFF

To reduce handoff overhead, the mapper's *query-independent, positionwise construction* streams each cache page once its source dependencies are ready. Transfer and mapping overlap subsequent source-prefill chunks, while final-page receiver computation overlaps mapping for later layers. Source features are reused across target layers and gathered across heads. Unrotation and alignment are fused, mapped keys receive target RoPE, and KV is written directly to final storage. Two reusable page slots bound workspace. The reader fuses its low-rank corrections with RoPE and merges external and local attention while preserving physical KV sharing under GQA. It adapts boundary replay, new inputs, and four decode forwards after the first token before native attention resumes. Appendices L and K give implementation details.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Models and tasks. The quality evaluation covers six representative directed transfers across model sizes, generations, and families. We organize 14 tasks into four categories: long-context understanding (**ContextQA**), code generation and completion (**Coding**), scientific and mathematical reasoning (**Reasoning**), and long-term memory (**Memory**). Each category retains the original task metrics.

Baselines. We compare five methods: **Text only**, which recomputes the full historical prefix on the receiver; **Relay-style**, which adapts selective recomputation from RelayCaching to cross-model cache handoff (Geng et al., 2026); **C2C-style**, an MLP mapper adapted from Cache-to-Cache (Fu et al., 2026); **Ridge**, which constructs target caches through closed-form ridge regression,

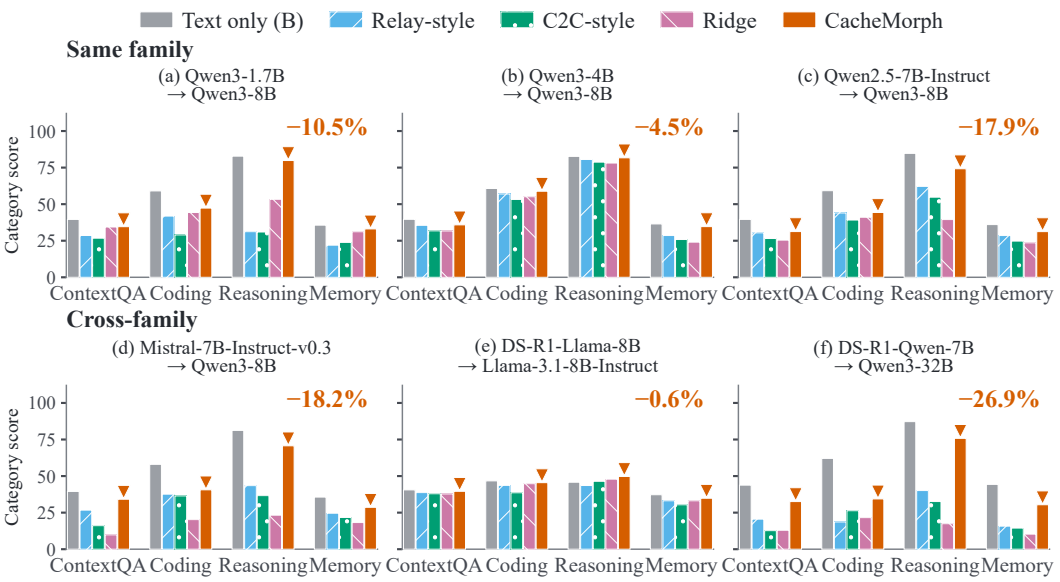


Figure 4: **Task accuracy across four task categories and six model pairs.**

drawing on the calibration approaches of Heo et al. (2026) and CacheBridge (Qu et al., 2026); and **CacheMorph**. Relay-style, C2C-style, and Ridge are variants adapted to the handoff interface used in this work. Appendix H summarizes mapper construction, alignment, and baseline implementations; Appendix L.3 specifies the timing protocol. We use two NVIDIA H200 NVL GPUs connected by NVLink, with the source and receiver deployed on separate devices.

5.2 Task quality across model transfers

Figure 4 compares four cache-transfer methods across six directed transfers and 14 tasks spanning model sizes, generations, and families. CacheMorph achieves the highest score in every model-pair/category comparison, with especially strong gains for challenging cross-family transfers, although fidelity to the native receiver remains model-dependent. Across transfers, its mean gap to Text only ranges from 0.6% to 26.9%, quantifying this remaining architecture dependence. Appendix A reports per-task results, while Section 5.5 examines how independent K/V routes, key-conditioned value prediction, and the query-adaptive reader contribute to transfer quality across model architectures and transfer settings.

5.3 Latency analysis

Figure 5 evaluates TTFT across three model-transfer directions and prefix lengths, with Text only as the receiver re-prefill baseline; Appendix L.3 provides the full protocol. CacheMorph achieves the lowest TTFT among cache-transfer methods in all tested settings and delivers 1.92–$4.12\times$ speedups over Text only at 32K prefixes, although transfer and mapping overhead outweigh the benefit at 512 tokens. As Table 2 shows, eliminating receiver-prefix recomputation and overlapping cache transfer and mapping with model computation offsets the mapping cost and yields the best end-to-end TTFT, especially at long prefixes where receiver-side recomputation dominates latency. The benefit increases with receiver prefill cost, reaching $4.12\times$ for Qwen3-32B at 32K.

5.4 End-to-end quality, latency, and cost

To evaluate practical efficiency–accuracy tradeoffs, we test CacheMorph with Qwen3-4B and Qwen3-14B on long-input RAG and long-output reasoning agents. We compare receiver re-prefill (*Text only*), cache mapping, and an all-14B workflow with native-cache reuse; Appendix F details the configurations, metrics, normalization, and workflow design for both workloads evaluated here.

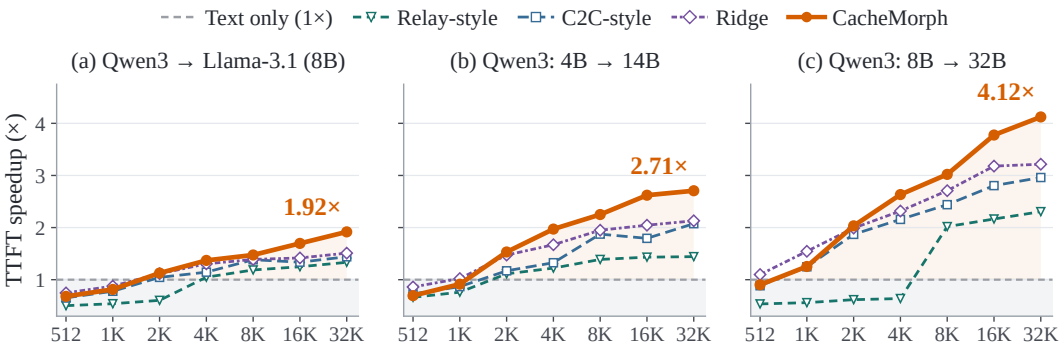


Figure 5: **TTFT speedup with KV reuse.** Mean TTFT speedup relative to Text only across target prefix lengths (tokens) for three model transfers.

Table 2: **Component latency breakdown at 32K.** Values are means over three profiles (ms).

	(a) Qwen3-8B → Llama-3.1-8B				(b) Qwen3-4B → Qwen3-14B				(c) Qwen3-8B → Qwen3-32B			
Method	Prefill	Transfer	Mapping	Overlap	Prefill	Transfer	Mapping	Overlap	Prefill	Transfer	Mapping	Overlap
Text only	1877.7	0.0	0.0	0.0	3273.0	0.0	0.0	0.0	7813.8	0.0	0.0	0.0
Relay-style	356.4	13.7	477.9	0.0	1116.5	13.7	580.0	0.0	1220.7	13.7	889.2	0.0
C2C-style	0.0	13.9	211.0	0.0	0.0	14.0	246.8	0.0	0.0	14.1	355.5	0.0
Ridge	0.0	13.8	477.8	0.0	0.0	13.8	580.7	0.0	0.0	13.8	890.7	0.0
CacheMorph	**0.0**	**13.9**	**359.9**	**322.3**	**0.0**	**15.2**	**458.7**	**334.5**	**0.0**	**15.1**	**776.5**	**558.9**

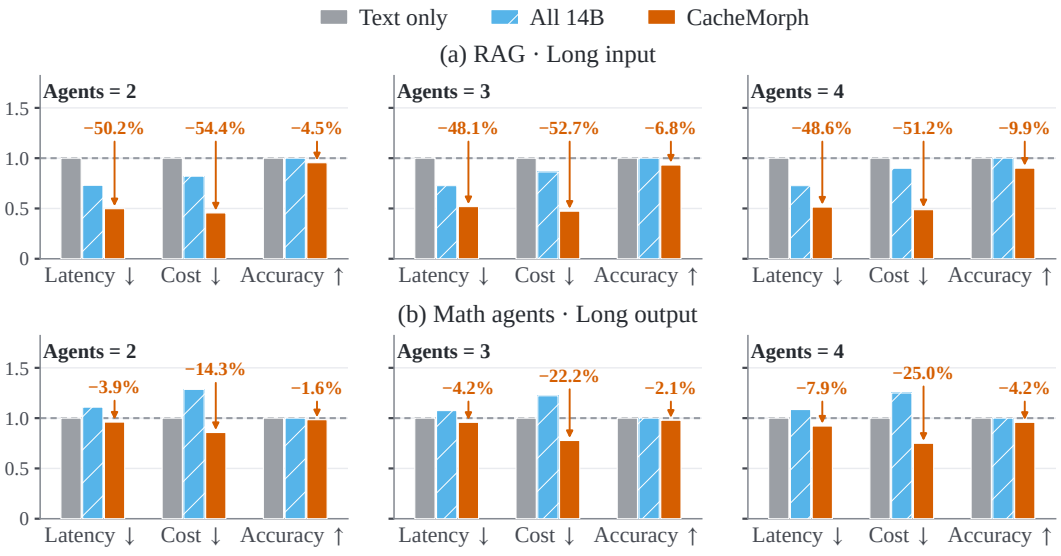


Figure 6: **Long-input and long-output workloads.** See Appendix F for details.

Long-input RAG. In 2–4-node MuSiQue workflows, 4B nodes process documents before a final 14B node answers the question. Compared with Text only, CacheMorph achieves 1.93–2.01× speedup and reduces token cost by 51.2–54.4%, with a relative accuracy drop below 10% (Figure 6a). Long-output Math agents. In 2–4-agent MATH-500 workflows, 4B agents generate intermediate reasoning for a final 14B verifier. CacheMorph reduces latency by 3.9–7.9% and token cost by 14.3–25.0%, with a relative accuracy drop below 5% versus Text only (Figure 6b). Gains are smaller than in RAG because long-sequence generation occupies more runtime, but both workloads show a favorable efficiency–accuracy tradeoff.

5.5 Ablation Study and Generality Analysis

Table 3 compares five mapper controls under fixed data, training budget, output KL, and reader settings. The controls replace the mapper with a matched MLP, feed source keys directly to the V branch, share K/V routes, remove K-to-V conditioning, or use reconstruction MSE. CacheMorph performs best on both tasks and transfer settings.

Table 3: **Mapper ablations on ContextQA and Reasoning.**

Category	CacheMorph	Matched MLP mapper	Source-K conditioning	Shared K/V routes	w/o $K \to V$ conditioning	Reconstruction MSE
(a) Qwen3-4B → Qwen3-8B						Scale
ContextQA	**35.85**	29.74	33.08	31.64	32.36	33.11
Reasoning	**81.70**	65.40	71.67	68.20	69.91	76.62
(b) Qwen3-8B → Llama-3.1-8B						Family
ContextQA	**32.47**	24.33	30.14	28.44	29.60	30.26
Reasoning	**74.88**	61.97	70.88	68.74	70.20	71.67

Table 4: **Reader ablations.**

Category	CacheMorph	No-reader	Group-only
(a) Qwen3-4B → Qwen3-8B			
ContextQA	**35.85**	33.29	33.89
Reasoning	**81.70**	75.03	78.83
(b) Qwen3-8B → Llama-3.1-8B			
ContextQA	**32.47**	30.17	31.10
Reasoning	**74.88**	72.31	73.69

Table 5: **Multi-hop transfer.**

Category	Text-only	$A \to C$	$A \to B \to C$
(a) Qwen3-4B → Qwen3-8B → Qwen3-14B			
ContextQA	39.68	38.27	**36.87**
Reasoning	89.02	87.36	**82.75**
(b) Mistral-7B → Qwen3-8B → Llama-3.1-8B			
ContextQA	33.67	30.95	**21.43**
Reasoning	76.87	72.33	**58.79**

With the mapper fixed, Table 4 shows quality gains from query correction and its head-specific factors relative to their respective ablations. Together, the results support both asymmetric mapping and head-specific reading. Appendix E details all controls. The reader is mapper-agnostic: across C2C-style, Ridge, and CacheMorph, fitted readers improve ContextQA by 1.12–4.65 points (Appendix E.1).

Table 5 composes two pairwise mappers through an intermediate model. Relative to direct transfer, the within-family chain loses 1.40/4.61 ContextQA/Reasoning points, whereas the cross-family chain loses 9.52/13.54. This gap indicates that mapping errors accumulate across hops, especially across model families; Appendix G gives the formal analysis.

6 Limitations and Conclusion

Limitations. Generalization of the current design remains uneven across model architectures and transfer settings. The mapper architecture could be further optimized to improve its expressiveness, robustness, and efficiency. An essential direction for future work is to integrate CacheMorph with production serving systems, specifically addressing concurrent requests, dynamic batching, and heterogeneous workloads.

Conclusion. We present CacheMorph, a framework for efficient cross-model KV reuse that preserves the receiver's downstream behavior while avoiding repeated prefill. Its asymmetric mapper models keys and values separately, its query-adaptive reader refines how the receiver consumes transferred caches, and its streaming pipeline overlaps cache transfer and mapping with model computation. Across 24 comparisons, it delivers a mean gain of 7.3 points (up to 35.7) over the best prior transfer baseline. Experiments across model transfers and routed workloads demonstrate consistent improvements in transfer quality and serving efficiency, supporting behavior-aware cache translation as a practical foundation for model switching.

Reproducibility Statement

Section 4 specifies the handoff boundary, mapper, objectives, reader, and execution pipeline. Appendix A provides per-task results; Appendix B defines the task groups and evaluation protocols; Appendices C–E give the theoretical analysis, motivation diagnostics, and ablation controls; Appendices F and G describe the routed workloads and multi-hop transfer; and Appendices H–L document implementation, training objectives, reader execution, and latency measurement. The accompanying data and scripts reproduce the reported figures and tables.

AI Use Statement

Generative AI tools, including OpenAI Codex, assisted with code development, manuscript drafting, and language polishing. The authors manually reviewed and verified all AI-assisted code and text and take full responsibility for the final content of this work.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Femi Bello, Anubrata Das, Fanzhi Zeng, Fangcong Yin, and Liu Leqi. Linear representation transferability hypothesis: Leveraging small models to steer large models, 2025. URL https://arxiv.org/abs/2506.00653.

Alan Chen, Jack Merullo, Alessandro Stolfo, and Ellie Pavlick. Transferring linear features across language models with model stitching. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Qvvy0X63Fv.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Lucio M. Dery, Zohar Yahav, Henry Prior, Qixuan Feng, Jiajun Shen, and Arthur Szlam. Latent space communication via k-v cache alignment, 2026. URL https://arxiv.org/abs/2601.06123.

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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=LeatkxrBCi.

Yingsheng Geng, Yuchong Gao, Weihong Wu, Guyue Liu, and Jiang liu. Relaycaching: Accelerating LLM collaboration via decoding KV cache reuse. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=1tbhBSXcyX.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. In P. Gibbons, G. Pekhimenko, and C. De Sa (eds.), *Proceedings of Machine Learning and Systems*, volume 6, pp. 325–338, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/a66caa1703fe34705a4368c3014c1966-Paper-Conference.pdf.

Kristjan Greenewald, Luis A. Lastras, Thomas Parnell, Vraj Shah, Lucian Popa, Giulio Zizzo, Chulaka Gunasekara, Ambrish Rawat, and David Daniel Cox. Activated loRA: Fine-tuned LLMs for intrinsics. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=S6SKa97Gm0.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

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, 2026. URL https://arxiv.org/abs/2608.03893.

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. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Junhao Hu, Wenrui Huang, Weidong Wang, Haoyi Wang, tiancheng hu, zhang qin, Hao Feng, Xusheng Chen, Yizhou Shan, and Tao Xie. EPIC: Efficient position-independent caching for serving large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=qjd3ZUiHRT.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. Position: The platonic representation hypothesis. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=BH8TYy0r6u.

Jordan Juravsky, Bradley Brown, Ryan Saul Ehrlich, Daniel Y Fu, Christopher Re, and Azalia Mirhoseini. Hydragen: High-throughput LLM inference with shared prefixes. In *Workshop on Efficient Systems for Foundation Models II @ ICML2024*, 2024. URL https://openreview.net/forum?id=Ye3ia34Fpp.

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

YUCHENG LI, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir H. Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, and Lili Qiu. SCBench: A KV cache-centric analysis of long-context methods. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=gkUyYcY1W9.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Yuhan Liu, Yuyang Huang, Jiayi Yao, Shaoting Feng, Zhuohan Gu, Kuntai Du, Hanchen Li, Yihua Cheng, Junchen Jiang, Shan Lu, Madan Musuvathi, and Esha Choukse. *DroidSpeak: KV cache sharing across fine-tuned model variants*. USENIX Association, USA, 2026. ISBN 978-1-939133-54-0.

Dongyang Ma, Yan Wang, and Tian Lan. Block-attention for efficient prefilling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7zNYY1E2fq.

Qianli Ma, Zhiqing Tang, Hanshuai Cui, Zhi Yao, and Weijia Jia. Semantic cache distillation: Efficient state transfer via reuse and selective patching. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=WLjJqhELBZ.

Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Feng Ren, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: trading more storage for less computation — a kvcache-centric architecture for serving llm chatbot. In *Proceedings of the 23rd USENIX Conference on File and Storage Technologies*, FAST '25, USA, 2025. USENIX Association. ISBN 978-1-939133-45-8.

Xingyu Qu, Siyuan Lu, Zhiyu Chen, Sheng Wang, and Tao Lin. Cachebridge: Efficient cross-model kv cache transfer, 2026. URL https://arxiv.org/abs/2609.00891.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Maximillian Rossi, Prajwal Raghunath, and Eugene Wu. Latent cache flow: Model-to-model communication without text. In *AdaptFM: Resource-Adaptive Foundation Model Inference*, 2026. URL https://openreview.net/forum?id=PvnR2LDCOs.

Xiangyu Shi, Marco Chiesa, Gerald Q. Maguire Jr., and Dejan Kostic. KVComm: Enabling efficient LLM communication through selective KV sharing. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=F7rUng23nw.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Sunghyeon Woo, Jaeeun Kil, Hoseung Kim, Minsub Kim, Joonghoon Kim, Ahreum Seo, Sungjae Lee, Minjung Jo, Jiwon Ryu, Baeseong park, Se Jung Kwon, and Dongsoo Lee. ICarus: Identical cache reuse for efficient multi-model inference. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=qrMo6R7lOS.

Sunghyeon Woo, Hoseung Kim, Sunghwan Shim, Minjung Jo, Hyunjoon Jeong, Jeongtae Lee, Joonghoon Kim, Sungjae Lee, Baeseong Park, Se Jung Kwon, and Dongsoo Lee. Prefillshare: A shared prefill module for kv reuse in multi-llm disaggregated serving, 2026b. URL https://arxiv.org/abs/2602.12029.

Jin woo Lee, Minkyung Song, Junghyun Oh, Seunghoon Han, Soyoung Park, Gwangseon Jang, and Sungsu Lim. Mixture-of-translators: Translating kv caches across heterogeneous large language models, 2026. URL https://arxiv.org/abs/2607.28979.

Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. KVLink: Accelerating large language models via efficient KV cache reuse. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=oDcAGSXZZP.

Xinyu Yang, Tianqi Chen, and Beidi Chen. APE: Faster and longer context-augmented generation via adaptive parallel encoding. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=yUC8pU508S.

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*, EuroSys '25, pp. 94–109, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400711961. doi: 10.1145/3689031.3696098. URL https://doi.org/10.1145/3689031.3696098.

Hancheng Ye, Zhengqi Gao, Mingyuan Ma, Qinsi Wang, Yuzhe Fu, Ming-Yu Chung, Yueqian Lin, Zhijian Liu, Jianyi Zhang, Danyang Zhuo, and Yiran Chen. KVCOMM: Online cross-context KV-cache communication for efficient LLM-based multi-agent systems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=yGOytgjurF.

Lu Ye, Ze Tao, Yong Huang, and Yang Li. ChunkAttention: Efficient self-attention with prefix-aware KV cache and two-phase partition. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11608–11620, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.623. URL https://aclanthology.org/2024.acl-long.623/.

Yi Zhao, Zuchao Li, and Hai Zhao. IAM: Efficient inference through attention mapping between different-scale LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19522–19533, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.959. URL https://aclanthology.org/2025.acl-long.959/.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: efficient execution of structured language model programs. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

A Detailed Transfer-Quality Results

Tables 6–11 report all 14 task scores for each of the five methods across the six model pairs in Figure 4, following its panel order. Scores retain each task's original metric on a 0–100 scale and are displayed to two decimal places. Memory evaluation uses GPT-4o as an external judge.

Figure 4 averages task scores within each category. Triangles mark the highest unrounded category mean among the four cache-transfer methods, excluding Text only. Each panel's percentage is the relative change in the mean score over all 14 tasks, computed as $100\,(\bar{s}_{\text{CacheMorph}}/\bar{s}_{\text{Text only}} - 1)$, where $\bar{s}$ denotes the equal-weight task mean.

Table 6: **(a) Qwen3-1.7B → Qwen3-8B.** Task scores for Figure 4.

Task	Text only	Relay-style	C2C-style	Ridge	CacheMorph
ContextQA					
2wikimqa	43.04	34.29	29.75	34.15	35.83
gov_report	33.55	27.76	28.67	30.13	30.43
hotpotqa	59.13	42.74	39.97	51.63	50.16
musique	36.02	20.89	20.74	33.06	32.70
narrativeqa	26.09	17.17	14.42	22.90	23.88
Coding					
humaneval_plus	71.95	35.98	10.98	37.20	47.56
livecodebench	40.00	21.04	5.97	27.01	28.53
repobench-p	65.19	68.50	69.95	68.66	65.87
Reasoning					
gpqa	60.61	55.56	36.36	40.91	59.09
gsm8k	92.49	15.01	32.52	68.99	88.02
math500	95.40	23.00	24.00	50.00	92.40
Memory					
longmemeval_s_retrieval	50.00	20.00	27.00	36.60	39.60
scbench_choice_eng	50.22	41.05	40.17	51.53	54.15
scbench_qa_eng	6.73	4.76	4.36	5.31	5.47

Table 7: **(b) Qwen3-4B → Qwen3-8B.** Task scores for Figure 4.

Task	Text only	Relay-style	C2C-style	Ridge	CacheMorph
ContextQA					
2wikimqa	44.11	38.23	32.10	33.24	36.88
gov_report	33.51	31.03	28.63	29.61	32.68
hotpotqa	58.48	54.38	49.03	51.89	57.78
musique	35.76	31.54	27.47	21.68	28.04
narrativeqa	26.33	22.32	21.53	21.99	23.86
Coding					
humaneval_plus	72.56	66.46	64.63	63.41	70.73
livecodebench	42.27	36.97	30.24	36.49	39.43
repobench-p	67.34	67.46	64.78	65.36	66.38
Reasoning					
gpqa	62.12	60.10	56.57	50.51	60.10
gsm8k	92.49	90.98	90.98	93.03	91.96
math500	93.40	90.60	88.60	90.80	93.00
Memory					
longmemeval_s_retrieval	51.00	33.60	29.80	25.40	50.00
scbench_choice_eng	51.53	45.85	41.48	41.05	48.47
scbench_qa_eng	6.90	6.20	6.20	5.58	5.47

Table 8: **(c) Qwen2.5-7B-Instruct → Qwen3-8B.** Task scores for Figure 4.

Task	Text only	Relay-style	C2C-style	Ridge	CacheMorph
ContextQA					
2wikimqa	43.04	36.45	33.01	32.38	35.27
gov_report	33.57	29.05	27.57	27.92	28.92
hotpotqa	58.83	45.56	32.30	33.40	46.19
musique	36.02	19.79	20.24	13.03	23.38
narrativeqa	26.39	21.15	19.44	20.28	22.53
Coding					
humaneval_plus	71.95	42.68	36.59	35.98	41.46
livecodebench	40.47	26.07	20.28	26.45	28.53
repobench-p	65.19	63.17	60.57	60.66	62.98
Reasoning					
gpqa	60.61	35.86	30.30	27.78	52.53
gsm8k	96.97	70.58	74.15	56.41	86.50
math500	96.60	80.00	59.80	34.20	83.60
Memory					
longmemeval_s_retrieval	51.00	35.00	30.00	30.00	42.00
scbench_choice_eng	50.22	45.41	40.17	37.12	46.72
scbench_qa_eng	6.87	5.23	4.04	3.58	4.95

Table 9: **(d) Mistral-7B-Instruct-v0.3 → Qwen3-8B.** Task scores for Figure 4.

Task	Text only	Relay-style	C2C-style	Ridge	CacheMorph
ContextQA					
2wikimqa	43.54	34.24	23.98	12.87	38.85
gov_report	33.32	28.69	15.56	17.82	29.64
hotpotqa	58.48	38.89	26.50	9.80	52.59
musique	35.76	16.62	9.68	3.48	30.40
narrativeqa	26.36	15.04	4.70	5.02	19.34
Coding					
humaneval_plus	70.73	34.15	30.49	23.17	43.29
livecodebench	40.00	19.53	14.50	7.96	17.54
repobench-p	63.34	59.13	64.33	29.54	60.96
Reasoning					
gpqa	59.09	29.80	32.83	16.16	52.53
gsm8k	90.98	52.99	25.47	10.54	78.54
math500	93.60	47.60	51.60	43.00	81.00
Memory					
longmemeval_s_retrieval	50.60	24.00	18.60	15.60	30.00
scbench_choice_eng	49.34	47.60	44.98	37.55	50.66
scbench_qa_eng	6.96	2.11	1.52	1.51	5.27

Table 10: **(e) DeepSeek-R1-Distill-Llama-8B → Llama-3.1-8B-Instruct.** Task scores for Figure 4.

Task	Text only	Relay-style	C2C-style	Ridge	CacheMorph
ContextQA					
2wikimqa	48.98	46.24	43.65	42.27	47.11
gov_report	34.33	32.71	31.78	33.13	33.29
hotpotqa	58.42	55.05	53.24	53.09	54.75
musique	33.05	32.46	34.31	33.08	34.24
narrativeqa	27.94	27.06	26.58	27.32	28.53
Coding					
humaneval_plus	62.80	50.00	40.24	50.61	53.05
livecodebench	16.49	16.97	16.97	20.47	20.00
repobench-p	60.83	63.89	58.54	63.29	63.44
Reasoning					
gpqa	27.78	13.13	20.20	10.10	26.26
gsm8k	63.99	65.50	68.01	74.53	70.96
math500	45.60	52.00	51.00	59.00	52.00
Memory					
longmemeval_s_retrieval	42.60	32.00	32.00	32.60	35.60
scbench_choice_eng	59.83	58.95	49.34	58.08	59.83
scbench_qa_eng	9.28	8.38	9.79	8.63	9.10

Table 11: **(f) DeepSeek-R1-Distill-Qwen-7B → Qwen3-32B.** Task scores for Figure 4.

Task	Text only	Relay-style	C2C-style	Ridge	CacheMorph
ContextQA					
2wikimqa	55.15	21.30	22.68	20.45	38.39
gov_report	33.15	23.80	9.80	1.74	29.49
hotpotqa	60.59	30.53	16.58	19.12	46.55
musique	39.57	14.00	9.67	8.15	25.87
narrativeqa	30.55	13.00	4.69	15.17	22.51
Coding					
humaneval_plus	72.56	14.02	15.24	28.66	48.78
livecodebench	45.50	14.50	16.30	10.33	31.28
repobench-p	68.18	27.35	47.38	25.77	23.06
Reasoning					
gpqa	68.69	23.23	25.25	20.20	50.00
gsm8k	96.97	44.05	30.71	13.50	89.99
math500	96.00	53.00	41.80	18.60	87.40
Memory					
longmemeval_s_retrieval	65.60	26.60	24.60	17.80	47.00
scbench_choice_eng	59.83	16.59	15.72	10.92	39.30
scbench_qa_eng	7.50	4.03	2.94	2.14	4.87

B Task Groups and Handoff Protocols

The task taxonomy is fixed before final evaluation. We organize the 14 tasks into four categories: long-context understanding (ContextQA), coding (Coding), reasoning (Reasoning), and memory (Memory). At handoff, the native-reference and cache-transfer conditions receive identical text histories: the native reference recomputes the historical prefix's KV cache on target model B, whereas cache transfer reuses the cache converted from source model A.

Table 12: Task groups, official evaluation-set sizes, and metrics for the tasks in Figure 4. ContextQA includes five LongBench tasks; Coding includes two code-generation tasks and one code-completion task. All scores are presented on a 0–100 scale, and category scores are equal-weight means of their constituent task scores. Sample counts refer to the complete official evaluation sets.

Group	Task / version	Samples (official full set)	Metric
ContextQA	2WikiMQA (LongBench)	200	Answer F1
ContextQA	GovReport (LongBench)	200	ROUGE-L
ContextQA	HotpotQA (LongBench)	200	Answer F1
ContextQA	MuSiQue (LongBench)	200	Answer F1
ContextQA	NarrativeQA (LongBench)	200	Answer F1
Coding	HumanEval+	164	pass@1, full base and augmented tests
Coding	LiveCodeBench v6	1,055	pass@1, public and hidden tests
Coding	RepoBench-P (LongBench)	500	Code-completion edit similarity
Reasoning	GPQA Diamond	198	Multiple-choice accuracy
Reasoning	GSM8K	1,319	Final-answer accuracy
Reasoning	MATH-500	500	Final-answer accuracy
Memory	LongMemEval-S + retrieval	500	GPT-4o-judged answer accuracy
Memory	SCBench English Choice	58 groups, 229 QA turns	Per-turn multiple-choice accuracy
Memory	SCBench English QA	69 groups, 351 QA turns	Per-turn answer F1

The sizes and metrics of the five ContextQA tasks and RepoBench-P follow the official LongBench definitions (Bai et al., 2024). For SCBench, sample counts distinguish groups sharing a context from the question–answer turns within each group; task scores are aggregated over turns.

Long-context understanding. ContextQA comprises 2WikiMQA, GovReport, HotpotQA, MuSiQue, and NarrativeQA, covering multi-document question answering, narrative understanding, and long-document summarization. During evaluation, model A first reads the document and constructs its KV cache, which is handed off to model B at the end of the document. Model B then reads the question or summarization instruction and generates the final response. Neither model A's document processing nor cache mapping uses the subsequent question. In the native-reference condition, model B rereads the same document before processing the same question or summarization instruction.

Reasoning. Reasoning comprises GPQA, GSM8K, and MATH-500, covering scientific reasoning, grade-school mathematical word problems, and more challenging mathematics, respectively (Rein et al., 2023; Cobbe et al., 2021; Hendrycks et al., 2021). The category score is the equal-weight mean of answer accuracy on the three tasks.

GPQA accuracy is computed from the model's final selected option. For GSM8K and MATH-500, model A first reads the problem and generates a partial reasoning trace. At a prespecified position independent of the answer outcome, the cache is handed off to model B, which continues reasoning and produces the final answer. In the native-reference condition, model B rereads the same problem and A-generated history before completing the solution. Incorrect reasoning and histories that already

contain an answer are retained without outcome-based sample filtering. Scoring uses model B's final answer.

Coding. Coding comprises HumanEval+, LiveCodeBench, and RepoBench-P. HumanEval+ and LiveCodeBench evaluate program correctness by executing test cases, using EvalPlus's base and augmented tests (Liu et al., 2023) and LiveCodeBench's public and hidden tests, respectively, and report pass@1. RepoBench-P uses the LongBench completion prompt and scorer, holds the supplied cross-file context fixed, and measures edit similarity between generated and reference code.

For generated-code handoff, model A first generates a partial program from the problem statement. At a prespecified token boundary, its cache is handed off to model B, which completes the program. Evaluation executes tests on the complete program generated across both stages. In the native-reference condition, model B rereads the same problem statement and partial program before continuing generation, ensuring identical text histories before handoff. For RepoBench-P, model A reads the code context, and model B completes the target code after receiving the cache. Test cases and reference solutions are used only for scoring and are not inputs to the mapper. The Coding score is the equal-weight mean of the three task scores.

Memory. Memory comprises LongMemEval-S retrieval, SCBench English Choice, and SCBench English QA, evaluating the retention and use of historical information after a model switch.

In the LongMemEval-S retrieval setting, content relevant to the current question is first retrieved from the interaction history, with identical retrieval results used by all methods. Model A reads the retrieved historical prefix; after receiving the cache, model B reads the question and generates a response. Responses are evaluated with the official judging prompts, using GPT-4o as an external judge of answer correctness.

In the SCBench setting (LI et al., 2025), model A first reads the shared long-text context and hands off its cache to model B before the first question–answer turn. Model B then processes the questions sequentially, retaining its previously generated answers and cache across turns. Subsequent turns neither receive reference answers nor remap the initial context. English Choice uses per-turn accuracy, while English QA uses per-turn answer F1; results are also reported separately for each turn. The Memory score is the equal-weight mean of the three task scores.

C Attention Error and Reader Capacity

C.1 From cache error to attention-output error

Fix a layer, query head, and query position. Let $\mathcal{I} = \mathcal{E} \cup \mathcal{L}$ be the disjoint union of visible external and local positions; masked positions are excluded. Superscript $\circ$ substitutes native target external KV while holding the query, local KV, positions, and mask fixed. This is a local counterfactual, not necessarily the fully native teacher trajectory: earlier cache errors can already have changed the query and local states. Vectors are rows, and keys and queries are post-RoPE. Set $\delta k_i = \delta v_i = 0$ for $i \in \mathcal{L}$.

Consider the path $k_i(\xi) = k_i^\circ + \xi \delta k_i$, $v_i(\xi) = v_i^\circ + \xi \delta v_i$. Its logits satisfy $s_i(\xi) = s_i^\circ + \xi \delta s_i$, where $s_i^\circ = q(k_i^\circ)^\top / \sqrt{d_B}$ and $\delta s_i = q \delta k_i^\top / \sqrt{d_B}$. Differentiating the softmax numerator and denominator at $\xi = 0$ gives

$$\dot{\alpha}_i := \left. \frac{d\alpha_i(\xi)}{d\xi} \right|_{\xi=0} = \alpha_i^\circ \delta s_i - \alpha_i^\circ \sum_{r \in \mathcal{I}} \alpha_r^\circ \delta s_r. \tag{10}$$

Although local logits are unchanged, their attention weights can change through the shared denominator. For $o^\circ = \sum_{i \in \mathcal{I}} \alpha_i^\circ v_i^\circ$, the product rule yields

$$\begin{aligned} \dot{o} &= \sum_{i \in \mathcal{I}} \dot{\alpha}_i v_i^\circ + \sum_{i \in \mathcal{I}} \alpha_i^\circ \delta v_i \\ &= \sum_{i \in \mathcal{I}} \alpha_i^\circ \delta s_i (v_i^\circ - o^\circ) + \sum_{i \in \mathcal{I}} \alpha_i^\circ \delta v_i \\ &= \sum_{i \in \mathcal{E}} \alpha_i^\circ \frac{q \delta k_i^\top}{\sqrt{d_B}} (v_i^\circ - o^\circ) + \sum_{i \in \mathcal{E}} \alpha_i^\circ \delta v_i. \end{aligned} \tag{11}$$

Softmax is smooth on the fixed visible set, with bounded second derivatives in a neighborhood of the reference. Hence $\Delta y = \dot{o} W_O + O(\varepsilon^2)$, giving Equation 3. Here $\varepsilon = \|\delta K\|_F + \|\delta V\|_F$; the local remainder constant depends on the fixed reference tensors, query, output projection, and visible set. At fixed target positions, orthogonal RoPE preserves the Frobenius norm of the corresponding pre-RoPE key perturbation.

The key term depends on query direction and value contrast, while the value term directly changes weighted content. Equal cache-error norms therefore need not produce equal output errors. If all visible values coincide, the key term vanishes. The expansion explains functional asymmetry without establishing a universal K/V sensitivity ordering or a particular width allocation.

C.2 Why the value objective depends on keys

Let $\widehat{\alpha}$ use mapped external keys and the same local keys. Stacking values over $\mathcal{I}$, add and subtract $\widehat{\alpha} V^{\circ}_{\mathrm{all}}$:

$$\begin{aligned} \Delta y &= (\widehat{\alpha}\widehat{V}_{\mathrm{all}} - \alpha^{\circ} V^{\circ}_{\mathrm{all}}) W_O \\ &= \left[(\widehat{\alpha} - \alpha^{\circ}) V^{\circ}_{\mathrm{all}} + \widehat{\alpha}_{\mathcal{E}} (\widehat{V} - V^{\circ}) \right] W_O. \end{aligned} \tag{12}$$

This recovers Equation 4 exactly, without a small-error assumption. Holding $\widehat{K}$ fixed, a variation $d\widehat{V}$ changes the output by $d\widehat{y} = \widehat{\alpha}_{\mathcal{E}}\, d\widehat{V}\, W_O$. For $\mathcal{E}_{\mathrm{attn}} = \frac{1}{2}\|\Delta y\|_2^2$, applying $d\mathcal{E}_{\mathrm{attn}} = \langle \Delta y, d\widehat{y} \rangle$ gives

$$\nabla_{\widehat{V}} \mathcal{E}_{\mathrm{attn}} = \widehat{\alpha}_{\mathcal{E}}^{\top} \Delta y\, W_O^{\top}. \tag{13}$$

Each external value receives the output residual projected through $W_O^{\top}$, scaled by its attention weight. Both quantities depend on $\widehat{K}$. When KV is shared, gradients from the consuming query heads and positions add, coupling value supervision to their different queries.

This dependence involves the complete attention distribution; tokenwise conditioning on $\widehat{k}_t$ is a restricted architectural choice, not the global solution implied by the gradient. It motivates coordinating K and V prediction, but does not establish superiority over directly supplying source-K features to the V branch. Equation 13 is a partial derivative with respect to values; joint mapper training also propagates gradients through their key-conditioned prediction. The output KL in Equation 2 supervises the full receiver trajectory beyond this local squared-error analysis.

C.3 Correction space of the reader

For head j, factor the group-shared and head-specific linear maps in Equation 8 into down and up matrices. Concatenate the down factors as $D_j = [D^{\mathrm{grp}}_{g(j)}, D^{\mathrm{head}}_j] \in \mathbb{R}^{d_B \times 5}$ and stack the up factors as $U_j = [U^{\mathrm{grp}}_{g(j)}; U^{\mathrm{head}}_j] \in \mathbb{R}^{5 \times d_B}$. The combined correction is $\Delta \bar{q}_{j,t} = \bar{q}_{j,t} D_j U_j$, hence

$$\Delta \bar{q}^{\top}_{j,t} \in \mathcal{U}_j := \mathrm{col}(U_j^{\top}), \qquad r_j := \dim \mathcal{U}_j \le \min(d_B, 5). \tag{14}$$

Choose an orthonormal basis $Q^{\mathrm{span}}_j \in \mathbb{R}^{d_B \times r_j}$, so $\Delta \bar{q}^{\top}_{j,t} = Q^{\mathrm{span}}_j c_{j,t}$. RoPE gives $q^{\top}_{j,t} = R_t \bar{q}^{\top}_{j,t}$ and rotates the correction by the same R_t.

For $n = |\mathcal{E}_t| > 0$, let $\Pi_t = I_n - \mathbf{1}\mathbf{1}^{\top}/n$. Two finite external-logit vectors induce the same conditional softmax exactly when their difference is constant, equivalently when their centered difference is zero. With $g = g(j)$ and external key matrices only, define

$$e^K_{j,t} = \frac{\Pi_t (K^{\circ}_g - \widehat{K}_g) q^{\top}_{j,t}}{\sqrt{d_B}}, \qquad J_{j,t} = \frac{\Pi_t \widehat{K}_g R_t Q^{\mathrm{span}}_j}{\sqrt{d_B}}. \tag{15}$$

The corrected external logits are $\widehat{K}_g (q^{\top}_{j,t} + R_t Q^{\mathrm{span}}_j c_{j,t})/\sqrt{d_B}$. The centered reference-minus-corrected residual is therefore $e^K_{j,t} - J_{j,t} c_{j,t}$, and $\mathrm{rank}(J_{j,t}) \le \min(n-1, r_j)$.

To expose the representational limit, temporarily allow arbitrary coefficients c. Writing $P_J = JJ^{\dagger}$ for the orthogonal projector onto $\mathrm{col}(J)$, with indices suppressed, gives

$$\|e - Jc\|_2^2 = \|(I - P_J)e\|_2^2 + \|P_J e - Jc\|_2^2.$$

The two components are orthogonal, and $c = J^{\dagger}e$ makes the second zero. Consequently,

$$\|e^K_{j,t} - J_{j,t}c_{j,t}\|_2^2 \geq \min_c \|e^K_{j,t} - J_{j,t}c\|_2^2 = \|(I - J_{j,t}J^{\dagger}_{j,t})e^K_{j,t}\|_2^2. \tag{16}$$

The relaxed minimum is zero exactly when $e^K_{j,t} \in \mathrm{col}(J_{j,t})$. The actual reader is more constrained: its coefficients are generated from $\bar{q}_{j,t}$ by fixed learned factors, rather than chosen independently for each residual. Even unrestricted query corrections cannot remove components outside $\mathrm{col}(\Pi_t \widehat{K}_g)$. This characterizes conditional-attention capacity, not a guarantee of downstream recovery or sufficient rank five.

C.4 External mass and value redistribution

Assume both external and local visible sets are nonempty. Define their partition functions $Z_{\mathcal{E}} = \sum_{i\in\mathcal{E}} \exp(s_i)$ and $Z_{\mathcal{L}} = \sum_{i\in\mathcal{L}} \exp(s_i)$, external mass $m = Z_{\mathcal{E}}/(Z_{\mathcal{E}} + Z_{\mathcal{L}})$, and conditional distributions $\pi^{\mathrm{ext}} = \mathrm{softmax}(s_{\mathcal{E}})$, $\pi^{\mathrm{loc}} = \mathrm{softmax}(s_{\mathcal{L}})$. Joint attention factors as $\alpha = (m\pi^{\mathrm{ext}}, (1-m)\pi^{\mathrm{loc}})$.

Let α° use native external keys and α' use mapped keys with the reader. Under the fixed-state assumptions above, their local logits and π^{loc} coincide. Substituting the factorization into $\sum_i \alpha^{\circ}_i \log(\alpha^{\circ}_i/\alpha'_i)$ separates region selection from selection within each region:

$$\mathrm{KL}(\alpha^{\circ}\|\alpha') = \mathrm{KL}(\mathrm{Bern}(m^{\circ})\|\,\mathrm{Bern}(m')) + m^{\circ}\,\mathrm{KL}(\pi^{\circ,\mathrm{ext}}\|\pi'^{\mathrm{ext}}). \tag{17}$$

The local conditional KL term vanishes. Centered external-logit matching addresses the second term, but not the first: adding a constant to external logits preserves their conditional softmax while changing $Z_{\mathcal{E}}$ and hence m. The reader can change both through its query correction, without a separate mass gate. If no local position is visible, $m = 1$ and only the conditional KL remains; if no external position is visible, no external correction is needed.

Finally, for fixed local readout $o^{\mathrm{loc}} = \pi^{\mathrm{loc}}V^{\mathrm{loc}}$, the corrected readout is

$$o' = m'\pi'^{\mathrm{ext}}\widehat{V} + (1-m')o^{\mathrm{loc}} \in \mathrm{conv}\big(\{\widehat{v}_i : i \in \mathcal{E}\} \cup \{o^{\mathrm{loc}}\}\big)\,. \tag{18}$$

This is an outer bound: the query parameterization need not realize every convex combination. The reader redistributes available content, whereas the mapper determines that content. These local results delimit their roles; they do not prescribe the reader's activation window. Window length and task-level benefits require the controlled evaluations in Appendix E.

D Motivation Diagnostic Setup

Setup for Figure 2. The diagnostics use Qwen3-8B → Llama-3.1-8B-Instruct. The two models have similar parameter counts but belong to different model families, providing a setting for examining cross-family cache correspondence and transfer errors at a comparable model scale while reducing confounding from differences in model size.

We use layer 17 of the receiver, Llama-3.1-8B-Instruct, as a fixed intermediate observation point. This layer lies near the middle of its 32-layer network. We select the layer based on network depth, without screening diagnostic outcomes. All three panels in Figure 2 use the same layer pair, allowing cache correspondence, K/V sensitivity, and differences across attention heads to be examined under consistent conditions.

The affine map is fitted on 400 FineWeb-Edu calibration documents, then frozen and evaluated on documents disjoint from the fitting data. Panels (b,c) use 32 held-out documents; shaded bands and error bars show 95% confidence intervals obtained by bootstrap resampling at the document level.

E Ablation Comparison Controls

Representative directions. The mapper and reader ablations in Tables 3 and 4 use Qwen3-4B → Qwen3-8B and Qwen3-8B → Llama-3.1-8B-Instruct, reporting ContextQA and Reasoning for each direction.

Main-table interpretation. Table 3 reports mapper interventions on the full configuration. All mapper variants receive the same prescribed reader-training procedure.

The shared-route control ties source support and routing but retains separate K/V outputs. Reconstruction MSE replaces only the K/V functional auxiliary terms, preserving output KL and consumption supervision; its coefficients are calibrated on development data. Reader variants in Table 4 freeze the same mapper. The matched-budget group-only control keeps per-head query inputs while sharing adaptation factors, so it is not a single correction broadcast to all heads.

Mapper controls. The matched MLP control replaces the structured mapper with an ordinary nonlinear MLP, using the same selected source information, output-KL supervision, training budget, and reader configuration under matched active-parameter/compute budgets. Readers are trained separately for their corresponding mappers using the same procedure.

The Source-K conditioning control replaces the mapped-key input to the V branch with a separately learned projection of the same selected source-K blocks, with matched active capacity. It retains the K prediction branch and the remaining V computation. Together with removing K-to-V conditioning, this distinguishes the benefit of source-K information from the benefit of representing it through predicted target keys.

Mapper generality through multi-hop transfer. Table 5 evaluates mapper generality beyond a single handoff using two chains: Qwen3-4B $\rightarrow$ Qwen3-8B $\rightarrow$ Qwen3-14B and Mistral-7B $\rightarrow$ Qwen3-8B $\rightarrow$ Llama-3.1-8B. Within each chain, A, B, and C denote the first, intermediate, and final models.

The table compares Text-only, direct $A \rightarrow C$ transfer, and sequential $A \rightarrow B \rightarrow C$ transfer on ContextQA and Reasoning. Sequential transfer bridges the endpoints using two pairwise mappers, $A \rightarrow B$ and $B \rightarrow C$. The comparison measures the additional quality loss from a second mapping step within and across model families.

E.1 Generality of the Reader

We evaluate the reader independently of CacheMorph's mapper by applying it to C2C-style, Ridge, and CacheMorph. This comparison tests whether query-dependent reading remains effective with alternative mapping methods.

Reader. We use Qwen3-4B $\rightarrow$ Qwen3-8B and Qwen3-8B $\rightarrow$ Llama-3.1-8B-Instruct, matching the component ablations. Table 13 compares ContextQA scores before and after fitting a group/head reader to each frozen mapper. Readers are trained separately using the same data and reader budget.

Table 13: **Reader across mappers.** ContextQA scores (↑); gains in score points.

Mapper	No reader	+ Reader score ↑	Gain Δ
(a) Q3-4B → Q3-8B			
C2C-style	31.75	**32.87**	**+1.12**
Ridge	31.68	**33.26**	**+1.58**
CacheMorph	33.29	**35.85**	**+2.56**
(b) Q3-8B → L3.1-8B			
C2C-style	24.52	**28.40**	**+3.88**
Ridge	22.47	**27.12**	**+4.65**
CacheMorph	30.17	**32.47**	**+2.30**

Frozen mapper; separately trained group/head reader. Same data and reader budget.

F ROUTED WORKLOADS AND PERFORMANCE REPORTING

Figure 6: metrics and normalization. The first row (a) uses aggregated LongBench MuSiQue results; the second row (b) uses aggregated MATH-500 results. Columns correspond to 2, 3, and 4 agents. Latency is mean end-to-end time in seconds. Quality is accuracy for both RAG and Math agents, on a 0–100 scale.

For both RAG and Math agents, cost denotes the mean reference token cost per request in US cents. No currency conversion or further division by sample count is applied. Both workloads use a fixed model-specific reference cache-read rate of 20% of the corresponding input rate, including valid native-cache reuse by the all-14B baseline. The cost estimates exclude unpriced mapper, transfer, and tool fees.

For each workload, metric, and agent count, the bar height is the method mean divided by the corresponding text-only mean. Reductions in latency and cost are beneficial; reductions in quality indicate a loss. For RAG with 2, 3, and 4 agents, latency decreases by 50.2%, 48.1%, and 48.6%; cost by 54.4%, 52.7%, and 51.2%; and accuracy by 4.5%, 6.8%, and 9.9% relatively.

For Math agents at the same counts, latency decreases by 3.9%, 4.2%, and 7.9%; cost by 14.3%, 22.2%, and 25.0%; and accuracy by 1.6%, 2.1%, and 4.2% relatively, corresponding to absolute decreases of 1.5, 2.0, and 4.0 percentage points.

RAG and reasoning workflows. In the RAG setting, documents are randomly partitioned according to the agent count and assigned to different preceding agents, which prefill their assigned documents to construct KV caches. The final agent receives the transferred KV caches, reads the question, and generates the answer. The reasoning-agent setting uses AutoGen's multi-agent framework. Its role graph is Solver→Verifier for two agents, Solver→Code Executor→Verifier for three, and Solver→Code Executor→Critic→Verifier for four.

All earlier roles run on Qwen3-4B, and the final verifier runs on Qwen3-14B. The all-14B reference runs the same roles on 14B. Two-agent runs contain no code-execution role. Earlier roles have a 2,048-token generation cap, and the verifier has a 4,096-token cap.

G MULTI-HOP CACHE TRANSFER AND ERROR PROPAGATION

The multi-hop comparison in Table 5 examines whether trained pairwise mappers can support transfer through an intermediate model. Let $F_{A\to B}$ and $F_{B\to C}$ denote the two directed cache transformations, including their respective alignment operations. For a fixed history D, their composition constructs

$$\widehat{\mathcal{C}}_B(D) = F_{A\to B}(\mathcal{C}_A(D)), \qquad \widehat{\mathcal{C}}_C^{\text{chain}}(D) = F_{B\to C}(\widehat{\mathcal{C}}_B(D)). \tag{19}$$

The final receiver C consumes this cache when processing new input. The direct-transfer reference instead uses $F_{A\to C}(\mathcal{C}_A(D))$, while Text-only reconstructs the prefix with native receiver prefill. Composition therefore extends the set of reachable model pairs through existing trained edges without requiring a direct $A \to C$ mapper. Each edge retains its own mapping parameters.

Compatibility at the intermediate interface. The first mapper's output must satisfy the second mapper's source-cache interface: token identities, positions, layer and head organization, and rotary coordinates must refer to the same intermediate prefix. Following Section 4.2, each hop removes its source RoPE, applies its own token alignment and KV mapping, and applies the next receiver's RoPE. The intermediate cache is expressed in model B's coordinates, so the two endpoint models can differ in architecture and tokenization. In Equation 19, B serves as an intermediate cache representation; the next hop directly consumes the first mapper's prediction. The following analysis holds the history and alignment operators fixed.

Propagation of cache errors. Suppress D and write $\mathcal{C}_B$ and $\mathcal{C}_C$ for the native caches of the same history. Define the first-hop residual $\delta_B = F_{A\to B}(\mathcal{C}_A) - \mathcal{C}_B$ and the second mapper's residual on native input $\epsilon_{B\to C} = F_{B\to C}(\mathcal{C}_B) - \mathcal{C}_C$. Adding and subtracting $F_{B\to C}(\mathcal{C}_B)$ gives the exact decomposition

$$\widehat{\mathcal{C}}_C^{\text{chain}} - \mathcal{C}_C = F_{B\to C}(\mathcal{C}_B + \delta_B) - F_{B\to C}(\mathcal{C}_B) + \epsilon_{B\to C}. \tag{20}$$

Thus the final residual contains both the second mapper's own error and its response to a perturbed input cache. If $F_{B\to C}$ is $L_{B\to C}$-Lipschitz on the segment between $\mathcal{C}_B$ and $\mathcal{C}_B + \delta_B$, the triangle inequality yields

$$\left\|\widehat{\mathcal{C}}_C^{\text{chain}} - \mathcal{C}_C\right\|_F \le L_{B\to C}\|\delta_B\|_F + \|\epsilon_{B\to C}\|_F, \tag{21}$$

where the cache norm includes all key and value entries. The bound is conditional on local stability of the second mapper; it does not assume $L_{B\to C} < 1$. First-hop errors may therefore be amplified, and key errors can also affect value prediction through the key-conditioned V branch. The two residual terms can reinforce or cancel each other, so this decomposition does not imply monotonic error growth on every example. Appendix C further shows that the effect of a cache residual on attention depends on the receiver's queries and value readouts; Equation 21 is a cache-space bound, not a task-accuracy guarantee.

Observed behavior and routing implications. For Qwen3-4B → Qwen3-8B → Qwen3-14B, sequential transfer reduces ContextQA and Reasoning scores by 1.40 and 4.61 points relative to direct transfer. For Mistral-7B → Qwen3-8B → Llama-3.1-8B, the corresponding reductions are 9.52 and 13.54 points. These results show that trained mappers can be composed in the tested directions, with a larger quality penalty in the cross-family chain. Multi-hop routing trades direct-edge coverage for additional transformations: an available path permits cache transfer, while its fidelity depends on the entire composition. Repeated switching consequently requires attention to cumulative quality loss as well as the cost of each handoff.

H Mapper Implementation, Alignment, and Baselines

This appendix summarizes the mapper, handoff interface, and comparison methods used in the main text.

H.1 Mapper construction and initialization

For each target layer and KV head, the mapper selects source features independently for the K and V branches. Each branch combines a low-rank affine prediction with a nonlinear residual, and learned routing weights determine the contributions of the selected source blocks. The V branch additionally uses the predicted target key, allowing value prediction to account for the addressing representation produced by the K branch. Mapping operates independently at each aligned position and does not require future receiver queries.

Aligned source and native target caches provide offline calibration data. Ridge regression initializes a dense mapper, which is then trained using the receiver's downstream behavior. Truncated SVD converts the trained affine map into low-rank factors, followed by recovery training of the mapper. Source supports and calibration means remain fixed during refinement. This procedure preserves the asymmetric K/V structure described in Section 4.2.

H.2 Token alignment and boundary replay

The source and receiver process the same text history with their respective tokenizers. Alignment associates each target token with the last source token completed by its character endpoint. This allows different tokenizations to share a causal correspondence; positions without a completed source token use a zero-cache fallback.

Source keys are unrotated at their original positions before alignment and mapping. Predicted target keys then receive the receiver's RoPE at the target positions. Values are mapped directly, and the resulting cache follows the receiver's layer and KV-head organization.

At handoff, the transferred cache covers the history preceding its final target token. The receiver replays that boundary token, processes the new input, and continues generation. For a source-generated history, the final generated token is first committed to the source cache. Native-reference and cache-transfer evaluations use the same target token history, masks, and positions: the reference reconstructs the prefix through receiver prefill, while transfer reuses the mapped source cache.

H.3 Baseline implementations

The methods in Section 5.1 receive the same historical text and subsequent input. They differ in how the receiver cache is constructed and consumed. The cache-transfer methods use the alignment conventions above; execution and timing protocols are given in Appendix L.3.

Text only. The receiver prefills the full historical text to construct its native cache before processing new input. This provides the native reference under the same history.

Relay-style. The cross-model adaptation starts from a Ridge-mapped cache and replaces cache entries through selective receiver recomputation. It combines cache reuse with native prefix computation to reduce transfer discrepancies.

C2C-style. An MLP maps aligned source features into receiver KV. The mapped cache supplies the historical prefix directly, without the native receiver-prefix cache used by the original Cache-to-Cache fusion.

Ridge. Closed-form ridge regression fits affine K/V maps from aligned native caches. The maps remain fixed during evaluation and construct the receiver prefix from source features.

CacheMorph. The asymmetric mapper constructs the receiver cache, and the group-shared and head-specific reader adapts access to it using current queries. Streaming execution overlaps cache transfer and mapping with model computation, as described in Sections 4.3–4.4.

I Mapper Supervision and Optimization

Output supervision. The mapper is trained against the receiver's native behavior on the same token history. The teacher uses native receiver KV, while the student uses the mapped cache; their internal states evolve independently. Output supervision minimizes the forward KL divergence between their next-token distributions, as defined in Equation 2. Supervised positions cover the handoff, subsequent reading, and continuation, with normalized weights balancing these stages. Both backbone models remain frozen, while gradients pass through the receiver to the mapper.

Functional auxiliary supervision. During early training, output supervision is supplemented by attention-output alignment and the separate K/V objectives in Appendix J. Attention-output alignment compares teacher and student readouts after the receiver's output projection, normalized by the native output energy. These signals guide the mapper toward caches that preserve how the receiver selects and reads historical information. Their overall weight decreases during training, after which optimization uses output KL alone.

Refinement and recovery. The calibrated dense mapper is refined with the functional objective in Section 4.2. After low-rank factorization, recovery training uses the same objective to refine the compressed mapper. Source supports and calibration means remain fixed while the mapping components are optimized jointly. Reader training follows as a separate stage with the mapper frozen, as described in Appendix K.

J Conditional Key and Value Losses

Key objective. The key auxiliary measures how mapped keys change attention addressing. At sampled reading positions, the student's queries, local keys, positions, and attention mask are held fixed. The receiver's attention distribution is computed using either native or mapped external keys, and their Jensen–Shannon divergence defines the key loss. Both distributions include external and local positions, so the objective captures changes in the selection of historical tokens and in the total attention assigned to the transferred prefix.

Value objective. The value auxiliary measures the content error under fixed reference attention. Native and mapped external values are aggregated using the same detached reference weights, then compared after the receiver's original output projection. The discrepancy is normalized by the reference readout energy. Holding attention fixed separates the effect of value prediction from changes in key-induced addressing.

Gradient isolation. For the value auxiliary, the mapped key used to condition the V branch is detached from the computation graph. This preserves the forward prediction while restricting updates to the value-producing components, including the key-conditioning projection. Output KL and attention-output alignment retain coupled gradients through both branches. The auxiliary objectives are used during offline training; online transfer requires no native receiver cache or reference attention.

K Reader Execution and Training

Query adaptation. The reader combines a group-shared low-rank transformation with a head-specific transformation of the current query, as described in Section 4.3. The shared component adapts heads that access the same KV group, while the individual component accounts for differences in their queries. Corrections are applied before RoPE and are initialized to leave the native query unchanged. The physical KV cache remains shared across heads.

Attention execution. The corrected query reads the transferred prefix, while the original query reads locally generated KV. The two attention computations are merged using their normalization factors, yielding the same result as a joint softmax over both regions. Query correction and RoPE are fused during execution, and the receiver's native normalization and causal masking conventions are preserved.

Activation and training. The reader is active during boundary replay, new-input processing, and a finite initial decoding window. Later steps resume native attention while retaining the mapped prefix and accumulated local KV. Training follows the same activation policy and supervises the receiver's output distributions both during and after the active window. The mapper and backbone models remain frozen, while gradients through local KV created during adapted reading are retained. This trains the reader to improve immediate cache access and subsequent continuation under the same history.

L Page-Streaming Handoff and Latency Measurement

The streaming execution in Section 4.4 exploits the mapper's positionwise, query-independent construction. Cache conversion can proceed as source features become available, allowing transfer and mapping to overlap with source computation. Receiver execution is released according to layer dependencies, so completed layer caches can be consumed while the remaining conversion continues.

L.1 Dependency-preserving scheduling

Source prefill proceeds in causal chunks, each of which attends to the preceding source history. Target pages are defined in the receiver's token space. Because the tokenizers may differ, a target page becomes ready when all source positions and layers required by its alignment are available. This readiness condition connects source computation to target cache construction without requiring identical token boundaries.

For each ready page, transfer precedes alignment and mapping. These operations can overlap the production of later source chunks. A receiver layer begins processing the boundary token and new input only after its full historical cache and the preceding layer's output are available. On the final page, this permits early receiver layers to execute while later layers are still being mapped. Device events enforce these dependencies, preserving causal access to the complete required prefix while exposing concurrency across stages.

L.2 Cache placement and buffer reuse

Transferred source keys are unrotated before alignment. Aligned K/V features are then reused across the target layers that consume them. Alignment is shared across heads while branch-specific mapping weights remain independent. Predicted keys receive target RoPE, and keys and values are written directly to their final receiver-cache locations. This avoids intermediate cache concatenation and installation copies.

Temporary receive and alignment storage uses double buffering. A staging slot is released only after its last mapping consumer completes; receiver attention reads the separate final cache. Buffer reuse therefore depends on completion of the computations that consume the slot, rather than completion of its transfer alone. Temporary workspace is governed by page size and mapper requirements, while persistent source and receiver caches continue to grow with history length.

The reader consumes the transferred prefix together with locally generated KV under the activation policy in Appendix K. When query adaptation ends, native attention retains both cache regions. The execution schedule thus supports continued generation without reconstructing the historical prefix.

L.3 Latency measurement and interpretation

The cold-request measurements in Section 5.3 start with source prefill and end at the receiver's first generated token. The interval includes transfer, alignment, mapping, boundary replay, and new-input processing. For a handoff with an existing source cache, source prefill lies outside the online interval. Subsequent decoding contributes to generation latency but is excluded from TTFT. Model loading, offline training, tokenization, and compilation warmup are excluded from the reported request latency.

At each context length, methods are evaluated on the same input under matched batching and generation settings. Measurements follow warmup and use repeated runs with randomized method order. Figure 5 reports the ratio of mean Text-only TTFT to each method's mean TTFT.

Table 2 separates receiver prefill or recomputation, KV transfer, mapping, and overlap. For Text only, prefill includes receiver prompt processing. Relay-style recomputation also includes selection and cache-installation overhead. Mapping includes token alignment, positional conversion, KV prediction, and target-cache writes. Overlap is computed from the active intervals of source prefill, transfer, and mapping on a common device timeline: it is their aggregate active duration minus the duration of their union, with overlapping intervals within each stage counted once. These component durations describe concurrency; end-to-end latency follows the dependency graph's critical path and cannot be recovered by simply summing them.

The main comparison evaluates each method together with its execution path.