DYNAMICKV: PREDICTIVE AND REVERSIBLE KV-CACHE PRECISION ALLOCATION FOR AGENTIC LLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent workflows and long-chain reasoning continually retain and access growing histories, making the KV cache a major GPU memory cost during inference. Existing KV quantization methods typically determine precision when historical KV states enter the cache or reduce bit widths unidirectionally during generation. Analysis of real generation traces shows that earlier KV states can be reused after a period of limited attention; moreover, the effect of demoting the same KV region on subsequent inference can increase or decrease across generation stages. To anticipate changes in future demand and adapt quantization under a limited budget, we introduce DynamicKV, a predictive and reversible KV precision allocation method. DynamicKV partitions historical KV states into regions aligned with underlying storage blocks and periodically estimates two signals: π, a score for significant next-interval attention, and d, a score for significant damage from demotion to low-bit precision. Their product, πd, serves as a gated ranking score for dynamic precision allocation under a fixed proportional GPU KV-byte budget. Each region retains a low-bit base on the GPU and a separable residual in host memory. When the region becomes important again, activating the residual restores an approximately high-bit representation, enabling repeated demotion and restoration.On agent workflow tasks (τ^2-bench and BFCL V4), DynamicKV improves success rates by 5.0–10.5 percentage points over the strongest compression baseline for each model. On AIME 2024/2025 and MATH-500, it improves accuracy by up to 3.3 percentage points over the strongest compression baselines. On a single A100-80GB GPU, DynamicKV supports $2.5\times$ the concurrency of BF16 at 32K and improves aggregate output throughput by 4.7–44.3% at sequence lengths of 16K–32K.

Keywords: KV-cache quantization; mixed precision; long-chain reasoning; LLM agents; predictive control.

1 INTRODUCTION

Agent workflows and long-chain-of-thought (CoT) reasoning are extending large language model inference from short, single-turn completion to long-horizon processes involving sustained reasoning, tool interactions, and feedback-driven revision (Yao et al., 2023; DeepSeek-AI, 2025). Models such as Kimi K2.5, the Qwen3.5 series, and Gemini 3.1 Pro reflect this trend (Kimi Team, 2026; Qwen Team, 2026; Google DeepMind, 2026). Gemini 3.1 Pro supports outputs of up to 64K tokens and an input context window of 1M tokens. In these settings, subsequent decisions repeatedly access a growing token history. To avoid recomputation, autoregressive Transformers cache the keys and values of each historical token at every layer for subsequent attention computation (Vaswani et al., 2017). Without eviction or compression, KV-cache capacity therefore grows linearly with the total number of historical tokens retained across active requests. This pressure remains substantial even with multi-query or grouped-query attention (Shazeer, 2019; Ainslie et al., 2023): the KV cache often exceeds the model weights in size. In our experimental setting, Qwen3-8B requires 54 GiB of KV storage at a batch size of 12 and a context length of 32K, more than three times the size of its weights. The KV cache is thus a deployment bottleneck for long-horizon reasoning and agent workflows.

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.

Existing work primarily reduces this cost through KV-cache compression. One effective approach exploits heterogeneity in KV states to choose quantization schemes, retaining more history in low-bit representations. KIVI, MiKV, and ChanMix use differences in key/value distributions, token importance, and channel-wise quantization sensitivity, respectively, to choose quantization granularity or allocate precision (Liu et al., 2024; Yang et al., 2024; Liao & Wen, 2026). These studies show that a uniform bit width is not optimal. However, they typically determine precision when KV states are cached or first compressed, with limited subsequent adaptation to the inference stage. Recent work therefore introduces runtime dynamics into KV management: PM-KVQ, TRIM-KV, and ThinKV progressively reduce precision or remove low-value KV states according to memory pressure or changing historical importance during generation (Bui et al., 2026; Liu et al., 2026; Ramachandran et al., 2026). These adjustments mainly involve unidirectional transitions such as demotion, decay, or deletion. Yet earlier states are not necessarily less important in long-horizon agent inference. Tool results and environmental feedback continually add information, confirm action outcomes, or change task states; subsequent decisions often combine this information with earlier user constraints, tool rules, and prior decisions. Similarly, mathematical reasoning can return to earlier content to verify intermediate conclusions, reuse derivations, or revise a solution path. Unidirectional precision reduction or eviction may not accommodate this later reuse. This raises a central challenge: how can we anticipate changes in future demand for historical KV states and dynamically adapt compression under a limited budget?

To address this challenge, we analyze real generation traces from long-chain mathematical reasoning and agent workflows and make three observations. First, attention to historical KV states need not decrease monotonically over time. We measure attention allocated to historical KV segments at different inference stages and find that some earlier segments regain significant attention after periods of limited use. Second, the sensitivity of a historical KV segment to low-bit quantization also changes during inference. The same demotion can have little effect on subsequent predictions at one stage and cause substantial damage at another. Third, attention demand and low-bit demotion risk are only weakly correlated: highly attended segments do not fully coincide with those at high risk of damage. Precision management under a limited KV budget must therefore consider both signals and allow precision restoration for states with high demand and high risk.

These observations motivate DynamicKV, a predictive and reversible precision allocation method for non-monotonic KV-cache demand. DynamicKV partitions historical KV states into regions and periodically reallocates their precision at predefined intervals. A future-demand score uses attention history available up to the current boundary to predict whether a region will receive significant attention in the next interval. A demotion-risk score estimates the potential effect of reducing that region's bit width on subsequent generation at the current stage. Their product forms a demand-gated risk ranking score. After subtracting the low-bit base cost from the GPU KV-byte budget, the allocator considers regions in descending order of score per incremental upgrade byte and restores those that fit within the remaining budget. To make these adjustments reversible, DynamicKV stores each region's low-bit base and recoverable residual separately. The same historical region can thus repeatedly change precision according to predicted allocation decisions and recover an approximately high-bit representation when needed.

Our main contributions are as follows:

1. Analysis of real long-CoT and agent traces shows that both future demand for historical KV states and low-bit quantization risk vary non-monotonically during inference and are only weakly correlated, exposing limitations of static or unidirectional compression in long-horizon reasoning and agent workflows.
2. We propose DynamicKV, which separately predicts next-interval demand and low-precision degradation risk for each region and dynamically allocates precision under a fixed proportional GPU KV-byte budget using actual byte costs. A low-bit base and separable residual enable repeated demotion and restoration of historical KV states.
3. At only 28.3% of BF16 KV storage, On agent workflow tasks (τ^2-bench and BFCL V4), DynamicKV improves success rates by 5.0–10.5 percentage points over the strongest compression baseline for each model. It achieves or ties the best compressed result on both AIME datasets. On a single A100-80GB GPU, it supports $2.5\times$ the concurrency of a BF16 KV cache at 32K and improves aggregate output throughput by 4.7–44.3% over BF16 at sequence lengths of 16K–32K.

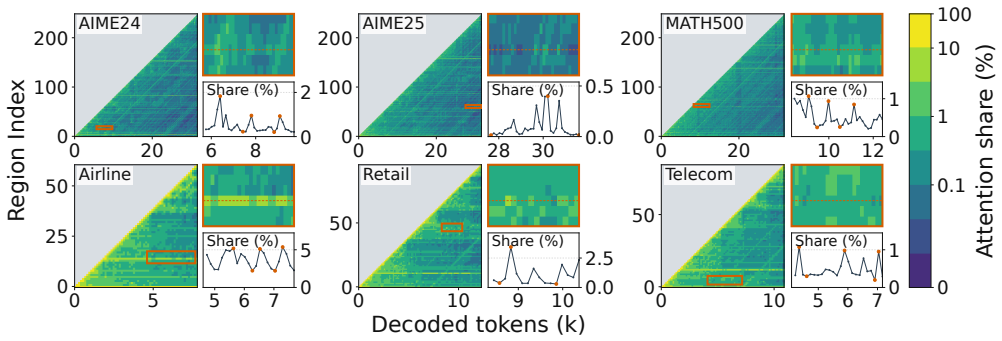

Figure 1: Attention dynamics of historical regions in math (top) and τ^2-bench agent (bottom) traces; orange boxes mark regions that regain attention after low-demand intervals.

2 Related Work

KV-cache compression. KV-cache compression primarily comprises eviction and quantization. H2O, StreamingLLM, FastGen, and SnapKV select historical tokens using attention or positional signals; PyramidKV additionally allocates retention across layers (Zhang et al., 2023; Xiao et al., 2024; Ge et al., 2024; Li et al., 2024; Cai et al., 2024). Eviction reduces storage but removes information that may be needed later. Another line of work retains more history through low-bit representations. KIVI exploits key/value distribution differences, KVQuant addresses representation and outlier structure, MiKV uses token importance, and ChanMix assigns precision by channel sensitivity (Liu et al., 2024; Hooper et al., 2024; Yang et al., 2024; Liao & Wen, 2026). These methods reveal heterogeneous compressibility across tokens, layers, and channels, showing that uniform bit widths are not optimal. Their precision configurations, however, are mainly determined by current importance, fixed representational properties, or offline calibration.

Temporal and predictive KV-cache management. Recent work models temporal variation in KV demand. PM-KVQ unidirectionally reduces bit widths as memory pressure increases (Liu et al., 2026); TRIM-KV and ThinKV compress low-value history through decaying retention scores or progressive eviction (Bui et al., 2026; Ramachandran et al., 2026). Although these methods introduce temporal dynamics, their main state changes remain progressive demotion, score decay, or deletion. Other work uses future-demand prediction or runtime changes in importance for KV management. LookaheadKV predicts during prefill which prompt KV states will be needed by future responses (Ahn et al., 2026). QEvict further adjusts windows among full-precision, quantized, and evicted tiers according to accumulated attention (Garg et al., 2026). In contrast, DynamicKV predicts next-interval demand and precision-reduction risk at every boundary rather than routing solely by observed importance, while reversibly reallocating the precision of historical regions.

3 Motivation

To examine how attention to historical KV states changes during inference and whether precision allocation can track these changes, we analyze real Qwen3-8B traces from mathematical reasoning and τ^2-bench. We partition historical context into regions aligned with underlying KV blocks, each containing at most 128 tokens, and measure their attention scores and low-bit quantization damage throughout their lifetimes. The analysis yields three observations.

Observation 1: demand for historical regions can rise again after cooling. In Figure 1, early and intermediate regions regain attention after low-demand intervals. Creation time or monotonic decay alone therefore cannot determine which history needs protection throughout a trajectory.

Observation 2: demotion damage varies non-monotonically over time. For each target region p in agent traces, we demote it from 8 to 4 bits at different checkpoints while keeping all other states

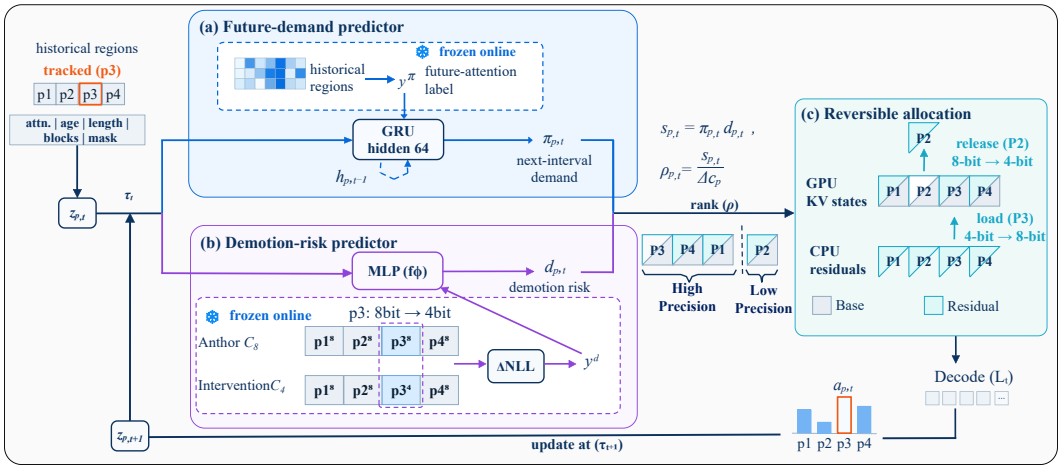

Figure 3: DynamicKV overview.

fixed. Its next-interval demotion loss is

$$\Delta_{p,t}^{4\leftarrow 8} = \max_{q\in I_t} \left[\ell_q^{p\leftarrow 4} - \ell_q^{8}\right]_+, \tag{1}$$

where the two negative log-likelihoods evaluate reference token q under the 8-bit reference and single-region 4-bit conditions, respectively. Figure 2 shows that the same region can alternate between low- and high-damage stages. Permanently retaining high precision wastes GPU memory during insensitive intervals, whereas permanent demotion risks errors at critical stages. Precision allocation should therefore jointly consider demotion risk and near-future demand.

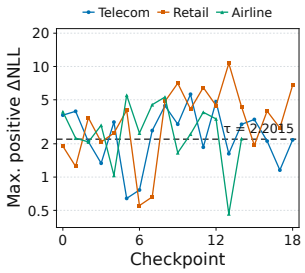

Figure 2: Stage-dependent demotion damage of representative regions in τ^2-bench traces.

Observation 3: future demand and demotion damage are not interchangeable signals. Finally, we compare next-interval demand scores with low-bit demotion risk. On the τ^2-bench validation set, the scores have Pearson correlation $r = -0.151$ and Spearman correlation $\rho = -0.38$. This distinction directly affects precision allocation under a limited budget. Demand alone may protect regions adequately represented at low precision; risk alone may protect regions unlikely to be used next. Effective allocation requires both conditions: the region is likely to participate in attention in the next decision interval, and its low-bit representation may damage subsequent generation.

Together, these observations imply three design requirements: predict next-interval demand for historical regions, independently estimate low-bit demotion risk, and support precision restoration. DynamicKV captures the first two signals as π and d, and uses a low-bit base with a recoverable residual to enable reversible precision changes, yielding dynamic precision allocation under a fixed proportional GPU KV-byte budget.

4 Method

Overview. DynamicKV reallocates historical KV precision using two predictions: significant next-interval attention $\pi_{p,t}$ and demotion risk $d_{p,t}$. Their product $s_{p,t} = \pi_{p,t} d_{p,t}$ is a demand-weighted ranking surrogate. A persistent GPU base and recoverable host residual allow repeated demotion and restoration; the allocator selects upgrades using predicted scores and actual incremental byte costs.

4.1 Regions and Decision Intervals

Boundaries $\tau_0 < \cdots < \tau_T$ define $I_t = (\tau_t, \tau_{t+1}]$. They occur at user, assistant, and tool-result event endings, with fixed-length checkpoints inside long events. Decisions at τ_t use only available history, and the selected precision states remain fixed within the interval.

Regions contain at most $R_{\max}$ contiguous tokens from one event and align with physical KV blocks. Regions are prediction/allocation units; blocks are storage/migration units. Event endings or checkpoints close the current block; subsequent tokens start a new block. Let $\mathcal{P}_t$ contain regions created by τ_t, with n_p tokens in region p. Every region retains a low-bit base; binary $x_{p,t}$ selects base-only (Cold-R) or base-plus-residual storage (Hot).

4.2 Predicting Next-Interval Demand

Offline supervision measures next-interval attention on complete reference trajectories. Let $\mathcal{K}_p$ contain region positions, Q_t contain the queries included in the interval statistics, and $\alpha_{\ell hqk}$ denote attention from query q to position k in layer ℓ and head h. The target and label are

$$A^{\text{next}}_{p,t} = \frac{1}{|Q_t|} \sum_{q \in Q_t} \sum_{\ell=1}^{L} \sum_{h=1}^{H} \sum_{k \in \mathcal{K}_p} \alpha_{\ell hqk}, \qquad y^{\pi}_{p,t} = \mathbb{1}[A^{\text{next}}_{p,t} > \tau_A]. \tag{2}$$

Query averaging normalizes for interval length; region attention is summed without length normalization because precision acts on the whole region. Each model's threshold τ_A is calibrated on training trajectories and frozen before validation and testing. Incomplete next intervals receive no labels.

Future attention is unavailable online. Let $a_{p,t-1}$ be the same attention aggregate measured over the completed interval I_{t-1}. The predictor uses

$$\mathbf{z}_{p,t} = \text{Norm}\big(\log(1 + a_{p,t-1}), \text{age}_{p,t}, n_p, g_p, m_{p,t}\big)\,, \tag{3}$$

where age counts elapsed decision intervals, $g_p = \lceil n_p/b \rceil$ counts physical blocks of size b, and $m_{p,t}$ indicates a valid previous attention observation. Normalization uses training statistics.

A single-layer GRU with 64-dimensional hidden states processes each region across successive decision intervals:

$$\mathbf{h}_{p,t} = \text{GRU}_\theta(\mathbf{z}_{p,t}, \mathbf{h}_{p,t-1}), \qquad \pi_{p,t} = \sigma(\mathbf{w}_\pi^\top \mathbf{h}_{p,t} + b_\pi). \tag{4}$$

Here σ is sigmoid. Regions at the same boundary are batched, not temporal steps. New regions use zero hidden states and a missing-observation mask. Attention from I_t updates the predictor only after τ_{t+1}, never its own interval's decision.

4.3 Estimating Counterfactual Demotion Risk

A frozen BF16 model generates reference trajectories. At selected training boundaries, paired teacher-forced replays compare anchor C_8, with all historical regions at approximately 8-bit precision, against $C_4(p)$, which demotes only region p to 4-bit. Parameters, input tokens, other regions, and the reference continuation remain fixed, isolating demotion damage rather than differences between independently generated continuations.

For reference-token negative log-likelihood ℓ_q, define

$$\Delta^{4\leftarrow 8}_{p,t} = \max_{q \in Q_t} [\ell_q^{C_4(p)} - \ell_q^{C_8}]_+, \qquad y^{d}_{p,t} = \mathbb{1}[\Delta^{4\leftarrow 8}_{p,t} > \tau_d], \tag{5}$$

where $[u]_+ = \max(u, 0)$. The maximum preserves localized, large predictive damage. The threshold τ_d corresponds to a predefined high-risk training quantile and is frozen before validation and testing. Longer intervals can be replayed in chunks while retaining the same maximum. Supervision uses this NLL difference; top-1 match and rank drop are diagnostic only.

A lightweight MLP predicts $d_{p,t} = \sigma(f_\phi(\mathbf{z}_{p,t}))$ from the same causal features, supervised by damage rather than demand labels. Here $d_{p,t}$ is a score for exceeding the damage threshold. Updated attention and age make risk time-dependent. The product score prioritizes regions likely to receive attention and vulnerable to demotion.

4.4 REVERSIBLE RESIDUAL REFINEMENT

Increasing storage bit width cannot recover information discarded during demotion. For original KV states X_p, let $\mathcal{Q}_b$ denote quantization to b bits followed by dequantization. We retain

$$B_p = \mathcal{Q}_4(X_p), \qquad R_p = \mathcal{Q}_4(\mathcal{Q}_8(X_p) - B_p), \qquad \widehat{X}_{p,t} = B_p + x_{p,t} R_p. \tag{6}$$

The residual approximates the base-to-8-bit difference in 4 bits. In COLD-R, the GPU stores only the base and a host sidecar retains the residual. In HOT, loading and adding the residual gives an approximately high-bit representation. Demotion releases only the GPU residual copy, enabling repeated restoration without regenerating residuals or recomputing historical KV.

4.5 BYTE-BUDGETED ALLOCATION

The budget coefficient β is relative to the bytes required to store the current history uniformly in INT8, denoted M_t^{INT8}. Let $c_4(n_p)$ and $c_{B+R}(n_p)$ be actual GPU byte costs under the storage layout. The base cost is $M_t^{\mathrm{base}} = \sum_{p \in \mathcal{P}_t} c_4(n_p)$, and upgrading region p costs $\Delta c_p = c_{B+R}(n_p) - c_4(n_p)$. After reserving every base, the remaining budget is $M_t^{\mathrm{up}} = \beta M_t^{\mathrm{INT8}} - M_t^{\mathrm{base}}$.

The allocation objective and the score per incremental byte are

$$\begin{aligned} \max_{\{x_{p,t}\}} \quad & \sum_{p \in \mathcal{P}_t} x_{p,t} s_{p,t} \\ \text{s.t.} \quad & \sum_{p \in \mathcal{P}_t} x_{p,t} \Delta c_p \le M_t^{\mathrm{up}}, \qquad x_{p,t} \in \{0,1\}. \end{aligned} \tag{7}$$

We approximate this objective greedily using

$$\rho_{p,t} = \frac{s_{p,t}}{\Delta c_p}. \tag{8}$$

Regions are checked in descending order, and residuals are activated whenever they fit within the remaining capacity. The allocator checks actual GPU bytes and breaks ties by fixed region identifiers. Equal upgrade costs make this ordering equivalent to score ordering; unequal costs require the byte-normalized score.

This is a proportional budget that grows with stored history, not a time-invariant byte cap. Within an interval, existing precision assignments remain fixed and newly allocated blocks add low-bit bases. Let $\mathcal{G}_t(u)$ be the blocks added by time u in I_t. Their contributions are

$$\begin{aligned} M_t^{\mathrm{GPU}}(u) &= M_t^{\mathrm{base}} + \sum_{p \in \mathcal{P}_t} x_{p,t} \Delta c_p + \sum_{g \in \mathcal{G}_t(u)} c_4(g), \\ M_t^{\mathrm{target}}(u) &= \beta \left(M_t^{\mathrm{INT8}} + \sum_{g \in \mathcal{G}_t(u)} c_8(g) \right). \end{aligned} \tag{9}$$

If the boundary allocation is feasible and each new block satisfies $c_4(g) \le \beta c_8(g)$, these additions preserve the proportional constraint, where $c_4(g)$ and $c_8(g)$ are its actual low-bit and INT8 byte costs. Otherwise, corresponding bytes must be reserved or allocation updated earlier.

4.6 TRAINING AND ONLINE EXECUTION

Only the predictors are trained; the language model remains frozen. Complete BF16 reference trajectories record boundaries, region mappings, and interval attention. Splits are task-disjoint, and agent examples sharing a task template stay together so that related regions and interventions cannot cross splits. All KV conditions use the same checkpoint, prompts, chat template, tokenizer, and generation settings. Training fits predictors and thresholds; validation selects architectures and allocation hyperparameters; testing is reserved for final evaluation. The two predictors are optimized independently.

Demand training uses weighted binary cross-entropy with continuous attention as auxiliary supervision. A softplus head on the GRU state predicts $\widehat{A}^{\text{next}}_{p,t} = \text{softplus}(\mathbf{w}_A^\top \mathbf{h}_{p,t} + b_A)$. The independently optimized objectives are

$$\mathcal{L}_\pi = \mathcal{L}_{\text{WBCE}}(y^\pi_{p,t}, \pi_{p,t}) + \lambda_A \,\text{Huber}\Big(\log(1 + A^{\text{next}}_{p,t}), \log(1 + \widehat{A}^{\text{next}}_{p,t})\Big), \tag{10}$$

$$\mathcal{L}_d = \mathcal{L}_{\text{WBCE}}(y^d_{p,t}, d_{p,t}). \tag{11}$$

Class weights follow training-set proportions; Huber loss limits the influence of examples with extremely large attention mass on the auxiliary regression term. Only the classification output enters allocation, not the auxiliary attention prediction. Normalization statistics and label thresholds use training data. Predictor widths, the auxiliary weight, region cap, checkpoint interval, and budget parameters are selected on validation data and frozen before testing.

Online, attention statistics accumulate by region. At each boundary, the system summarizes the completed interval, updates features and GRU states, predicts both scores, and selects the next hot set. It releases GPU residual copies for departing regions before loading residuals for new hot regions. Attention then uses the base with or without its residual until the next boundary. With fixed predictor widths, prediction over $|\mathcal{P}_t|$ regions takes $O(|\mathcal{P}_t|)$ time and sorting takes $O(|\mathcal{P}_t| \log |\mathcal{P}_t|)$.

5 Experiments

5.1 Settings

Models and datasets. We use Qwen3-8B (Yang et al., 2025), Qwen3.5-27B (Qwen Team, 2026), and DeepSeek-R1-Distill-Qwen-7B (DeepSeek-AI, 2025). For hybrid Qwen3.5-27B, management applies to standard KV states in Gated Attention layers. Each model has separate, deployment-frozen predictors for mathematics and agent workflows. Mathematical evaluation uses AIME 2024 and 2025 (30 problems each) and MATH-500, a 500-problem MATH test subset (Hendrycks et al., 2021; Lightman et al., 2024), reporting final-answer accuracy. Agent evaluation uses τ^2-bench (Barres et al., 2025) and BFCL V4 (Patil et al., 2025), with benchmark environments, user simulators, and official closed-loop task success.

Mathematical training problems are stratified by category and difficulty from the MATH training split (Hendrycks et al., 2021), using seed 42. All trajectories and interventions from one problem remain together; models and ablations share problem-ID splits. Agent training uses the benchmark training split. Test data are excluded from training and hyperparameter selection.

Baselines and resources. KIVI uses channel-wise key and token-wise value quantization (4/4) (Liu et al., 2024); MiKV retains important tokens at higher precision (4/16) (Yang et al., 2024); PM-KVQ progressively lowers precision with block-level memory management (Liu et al., 2026). Their GPU KV costs are 26.6%, 41.3%, and 26.6% of BF16, versus 28.3% for DynamicKV. The component ablations use a common budget. Quality runs use one A100-80GB GPU, batch size 1, closed-loop task-by-task serving, and at most 32,768 output tokens. DynamicKV uses 4-bit bases and residual upgrades targeting approximately 8-bit representations. Efficiency runs fix each method's batch at its maximum stable 32K value.

5.2 Long-Form Reasoning

Figure 4 shows seven best or tied-best compressed results for DynamicKV. Qwen3-8B has the most consistent gains: final accuracy improves over each dataset's strongest compression baseline by 3.3, 3.3, and 1.0 percentage points on AIME 2024, AIME 2025, and MATH-500, respectively, and by 16.7–27.6 points over MiKV. Uniform quantization does not distinguish region-level demotion risk, while creation-time importance cannot track later demand; predicting both signals and reallocating precision preserves historical states relevant to subsequent reasoning. On Qwen3.5-27B, DynamicKV exceeds PM-KVQ and MiKV by 30.0 points on AIME 2024. In this more low-bit-tolerant model, it matches the quality of compression methods better suited to the model–task combination at a near-INT4 budget.

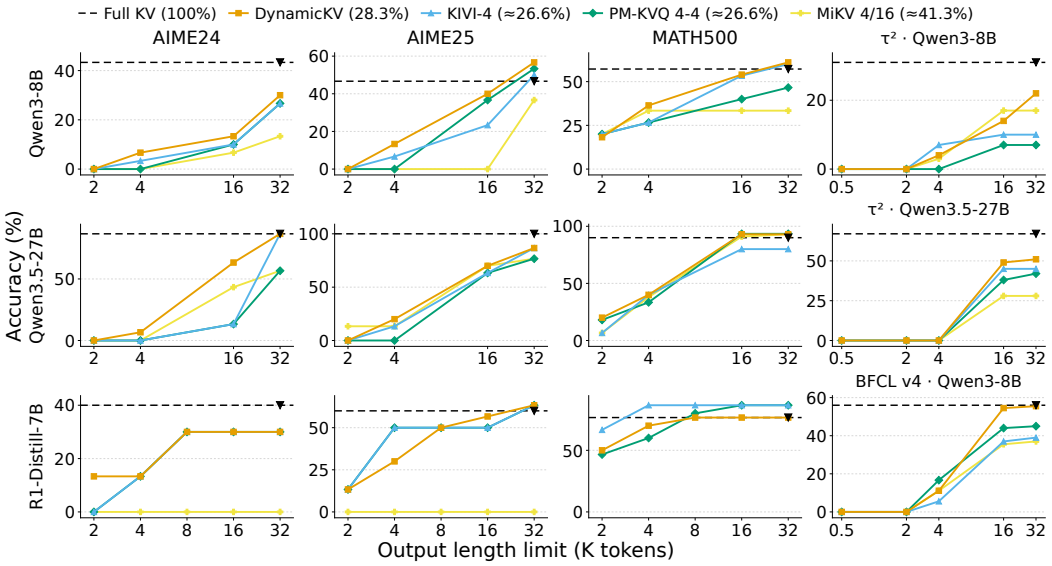

Figure 4: Task performance under different output length limits on agentic and math datasets.

R1-Distill-7B illustrates the limits of dynamic reallocation. On both AIME datasets, DynamicKV matches KIVI and PM-KVQ and exceeds MiKV by 30.0 and 63.3 points, respectively. On MATH-500, its result plateaus at 8K; KIVI also reaches its endpoint early. The main quality differences thus emerge before later reallocation has much scope to help. Benefits concentrate on long trajectories with changing historical demand and later reuse of earlier states, rather than universal changes to decoding outputs.

5.3 Agent Workflows

Agent interactions introduce settings in which user constraints, tool calls, and environmental observations become relevant again after multiple intervening events. We evaluate this setting on the test tasks of τ^2-bench and BFCL V4 using official end-to-end task success rates. Figure 4 reports that DynamicKV achieves the highest compressed success rate on both backbones. On τ^2-bench, it improves task completion by 5.0–6.0 percentage points over the strongest compression baseline; on BFCL V4, it outperforms the strongest compression baseline, PM-KVQ, by 10.5 percentage points. Most tasks for the 27B model finish within 16K tokens, whereas more 8B trajectories extend beyond 16K, making the later-stage benefits of reversible reallocation more pronounced. This advantage is consistent with agent tasks' sensitivity to critical local states: losing a user constraint, entity identifier, or key tool observation can directly cause subsequent actions to depart from the task objective.

5.4 Efficiency

On one A100-80GB GPU with Qwen3.5-27B, BF16, DynamicKV, and KIVI support 12, 30, and 32 concurrent 32K requests, respectively. Tests use the official KIVI 4/4-bit CUDA kernel (Liu et al., 2024) and our custom kernel for DynamicKV, with otherwise aligned inference engines. Table 1 shows that DynamicKV achieves 2.5× BF16 concurrency and 4.7–44.3% higher aggregate throughput than BF16 at 16K–32K. Compared with KIVI, its aggregate throughput is 5.4–10.2% lower, but at 32K it improves τ^2-bench success and MATH-500 accuracy by 6.0 and 13.0 percentage points, respectively, while tying both AIME datasets.

5.5 Ablations

Complementary signals and residual refinement. Table 2 compares future-demand prediction π, demotion risk d, the joint score πd, and residual configurations on Qwen3-8B under a fixed GPU KV budget of 28.3% of BF16. The joint score without residual refinement exceeds the better single-

Table 1: Aggregate and per-request throughput.

	16K		24K		32K	
Method	Agg.	Per req.	Agg.	Per req.	Agg.	Per req.
BF16	163.9	13.66	158.8	13.23	146.6	12.22
KIVI	263.4	8.23	215.5	6.73	162.3	5.07
DynamicKV	236.6	7.89	193.7	6.46	153.5	5.12

Table 2: Ranking and refinement ablations at 32K on Qwen3-8B. Ranking uses π, d, or πd; the final row additionally enables residual refinement.

Configuration	KV size	AIME24	AIME25	MATH-500	τ^2	BFCL V4
BF16	100%	43.33	46.67	57.2	31.00	56.00
Uniform 4-bit	26.6%	0.00	0.00	6.60	16.00	5.00
π only	28.3%	20.00	40.00	40.00	17.00	38.50
d only	28.3%	23.33	36.67	36.60	18.00	34.00
πd, no residual refinement	28.3%	26.67	46.67	51.20	20.00	44.50
DynamicKV	28.3%	30.00	56.67	61.00	22.00	55.50

signal arm by 2.0–11.2 points across five tasks (mean 5.8; Table 2). The πd no-residual arm uses direct Q8 and one-way demotion, discarding Q8 when a region is demoted. The full DynamicKV configuration gains a further 2.0–11.0 percentage points (mean 7.2). Uniform 4-bit storage has a KV-byte cost close to DynamicKV but performs substantially worse on all five evaluations. These comparisons show that neither average bit width nor cache size alone explains the full method's results.

Prediction compared with observed attention. At $\beta = 0.55$, π achieves a budgeted positive recall of 0.2133 versus 0.1829 for last-attention, a gain of 3.04 percentage points (Table 5). Under the same byte-constrained selection procedure, π therefore covers more regions with significant next-interval demand. In the end-to-end comparison with the risk and residual settings held fixed (Table 3), replacing last-attention with π increases AIME25 accuracy from 43.33% to 56.67% and MATH-500 accuracy from 36.60% to 61.00%. The offline and end-to-end results are complementary rather than proportional, supporting the value of predicting future demand instead of relying only on previous-interval attention. See Appendix B for details.

Table 3: Same-budget ranking comparison.

Configuration	AIME 2025	MATH-500
$\pi \times d$	56.67	61.00
Last-attention $\times d$	43.33	36.60
Recency	40.00	33.40
Cumulative attention	26.67	13.20

6 Conclusion

We presented DynamicKV, a predictive and reversible KV-cache precision management method for long-horizon generation and agent workflows. At each decision boundary, it separately predicts next-interval demand and demotion risk for historical regions and uses $\pi_{p,t} d_{p,t}$ to rank recoverable residuals for allocation within a fixed proportional GPU KV-byte budget. This enables repeated switching between low- and high-bit representations. At approximately 28.3% of BF16 GPU KV bytes, DynamicKV improves task success on agent workflow datasets by 5.0–10.5 percentage points over the strongest compression baseline for each evaluation, and improves long-chain mathematical reasoning by up to 3.3 percentage points. Maximum stable concurrency reaches $2.5\times$ BF16, and aggregate output throughput improves by 4.7–44.3% over BF16 at sequence lengths of 16K–32K.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology and experimental design, assist with method implementation and translation, and support the interpretation and presentation of experimental results. We also used generative AI tools to organize research materials, identify and summarize relevant literature, create or edit supporting code and scientific figures, and draft and edit portions of the manuscript. All AI-assisted outputs were reviewed by the authors. Literature references and factual claims were checked against the original sources, AI-assisted code was reviewed and tested, and reported numerical results and technical descriptions were verified against the implementation and experimental records. The authors made all final methodological and scientific decisions and take full responsibility for the final content of this work, including all text, claims, code, figures, and other artifacts produced with AI assistance.

REFERENCES

Jinwoo Ahn, Ingyu Seong, Akhil Kedia, Junhan Kim, Hyemi Jang, Kangwook Lee, and Yongkweon Jeon. LookaheadKV: Fast and accurate KV cache eviction by glimpsing into the future without generation. In *International Conference on Learning Representations*, pp. 71406–71430, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/746222c1871d3fa7a94bdacabc34e26c-Abstract-Conference.html.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://aclanthology.org/2023.emnlp-main.298/.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025. URL https://arxiv.org/abs/2506.07982v1.

Ngoc Bui, Shubham Sharma, Simran Lamba, Saumitra Mishra, and Rex Ying. Cache what lasts: Token retention for memory-bounded KV cache in LLMs. In *International Conference on Learning Representations*, pp. 120316–120342, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/c3887741ee4e655494770be95303383f-Abstract-Conference.html.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024. URL https://arxiv.org/abs/2406.02069v4.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948v2. Version 2, revised January 2026.

Ayushman Garg, Akshita Gupta, Shaswata Bhattacharya, Abhishek Gupta, Sandeep Kumar, and Manoj Kumar. QEvict: Recoverable quantized KV eviction for attention-drift-robust long-context decoding. *arXiv preprint arXiv:2608.05326*, 2026. URL https://arxiv.org/abs/2608.05326v1.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.01801v4.

Google DeepMind. Gemini 3.1 Pro Preview. Official model documentation, 2026. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.1-pro-preview. Accessed September 25, 2026.

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 *Advances in Neural Information Processing Systems*, 2021. URL https://arxiv.org/abs/2103.03874v2.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems*, 2024. URL https://papers.nips.cc/paper_files/paper/2024/hash/028fcbcf85435d39a40c4d61b42c99a4-Abstract-Conference.html.

Kimi Team. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026. URL https://arxiv.org/abs/2602.02276v2.

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. In *Advances in Neural Information Processing Systems*, 2024. URL https://papers.nips.cc/paper_files/paper/2024/hash/28ab418242603e0f7323e54185d19bde-Abstract-Conference.html.

Chengxi Liao and Zeyi Wen. Channel-aware mixed-precision quantization for efficient long-context inference. In *International Conference on Learning Representations*, pp. 52467–52483, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/55f5aa8d997a8071d0651e0009ec49a2-Abstract-Conference.html.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Tengxuan Liu, Shiyao Li, Jiayi Yang, Tianchen Zhao, Feng Zhou, Xiaohui Song, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. PM-KVQ: Progressive mixed-precision KV cache quantization for long-CoT LLMs. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/afb8caec018d3c8f6ef8b81fa52386fe-Abstract-Conference.html.

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. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://proceedings.mlr.press/v235/liu24bz.html.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5. Accessed September 25, 2026.

Akshat Ramachandran, Marina Neseem, Charbel Sakr, Rangharajan Venkatesan, Brucek Khailany, and Tushar Krishna. ThinKV: Thought-adaptive KV cache compression for efficient reasoning models. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/b2b502c3629beadda06311386d2c6f73-Abstract-Conference.html.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019. URL https://arxiv.org/abs/1911.02150v1.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. URL https://arxiv.org/abs/1706.03762v7.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.17453v4.

A. Yang, A. Li, B. Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388v1.

June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. No token left behind: Reliable KV cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*, 2024. URL https://arxiv.org/abs/2402.18096v1.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629v3.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/6ceefa7b15572587b78ecfcebb2827f8-Abstract-Conference.html.

A Per-Model Baseline Curves and Supplementary Analyses

A.1 Region-Size Sensitivity: 8B Mathematical Reasoning

Table 4 examines how the region cap affects demand prediction on the Qwen3-8B mathematical validation set. For every setting, the prediction target is defined consistently as

$$h1 = \mathbb{1}[\text{next-interval region attention mass} > \tau_A].$$

Because changing the region cap repartitions the same trajectories, the number of region–boundary examples and the positive rate also change. The 128-token row uses the canonical snapshot containing 1,137,621 examples, exactly the dataset evaluated in Appendix A.2; the remaining rows are obtained from separate repartitioning sweeps at the corresponding caps.

For the canonical 128-token configuration, we additionally report budgeted recall at $\beta = 0.55$. This metric is computed by applying the byte-constrained region-selection procedure to the validation snapshot and measuring the fraction of $h1$-positive regions selected for upgrade. Under the canonical storage layout, $\beta = 0.55$ corresponds to 28.38% of BF16 KV storage and leaves upgrade capacity roughly equivalent to 6.8% of tokens after accounting for the 4-bit base. This token-equivalent percentage is a description of the available capacity, not a theoretical upper bound on region-level recall: regions can have different lengths and incremental upgrade costs, and the available budget is enforced independently at each decision boundary.

Table 4: Sensitivity to the region cap on the Qwen3-8B mathematical validation set. Budgeted recall is reported only for the canonical 128-token configuration, for which byte-budgeted selection was evaluated at $\beta = 0.55$.

Cap (tokens)	AUPRC (π)	AUPRC (last-attn.)	Budgeted recall (π/last-attn.)	Positive rate	Rows
128 (main)	0.9295	0.9145	**0.2133/0.1829**	0.313	1,137,621
256	0.9685	0.9708	—	0.616	570,749
512	0.9911	0.9923	—	0.895	287,131
1024	0.9939	0.9942	—	0.989	132,461
4096	0.9945	0.9947	—	0.992	37,036
Full trajectory	0.9921	0.9949	—	0.994	10,447

Interpretation. Increasing the region cap substantially changes the prediction problem. Larger regions have much higher positive rates: the fraction of regions labeled as receiving significant next-interval attention increases from 31.3% at 128 tokens to approximately 99% for the largest caps. Correspondingly, AUPRC also increases, because the region-level classification problem becomes progressively less selective. The higher AUPRC at large caps therefore should not be interpreted as evidence that those granularities provide better precision allocation under a fixed GPU-byte budget.

For the canonical 128-token configuration, π achieves a measured budgeted recall of 0.2133 at $\beta = 0.55$, compared with 0.1829 for last-attention. For the other caps, we report only prediction statistics because byte-budgeted recall was not evaluated in this sweep. Thus, this analysis characterizes the effect of region granularity on the demand-prediction task rather than establishing a budget-constrained recall ordering across all region sizes. We use a 128-token cap in the main system and evaluate its budgeted ranking behavior separately in Appendix A.2.

A.2 Direct Evaluation of Demand Prediction

Offline ranking comparison. The canonical Qwen3-8B mathematical validation snapshot contains 1,137,621 rows, all matched to predictions, with a positive rate of 31.3%. The label is $h1$, with $\tau_A = 403$. This is the same dataset as the cap-128 row in Appendix A.1. Last-attention mass uses attention from the immediately preceding interval, whereas cumulative attention aggregates attention from region creation. The latter obtains an AUPRC of only 0.3852, indicating that accumulated historical importance is a poor proxy for next-interval demand. In contrast, π improves AUPRC over last-attention from 0.9145 to 0.9295.

Table 5: Offline demand-ranking evaluation. The actual byte budgets at $\beta = 0.55$ and $\beta = 0.75$ allow upgrades for approximately 6.8% and 45.7% of tokens, respectively.

Ranker	AUPRC	Recall@10%	Recall@25%	$\beta = 0.55$	$\beta = 0.75$
π (p_{h1})	**0.9295**	0.3164	**0.7311**	**0.2133**	**0.9657**
Last-attention mass	0.9145	0.3156	0.7213	0.1829	0.9472
Recency ($1/(1 + \text{age})$)	0.7045	0.2782	0.5455	0.2038	0.7735
Cumulative attention	0.3852	0.1727	0.2776	0.1375	0.3896

Under the tighter $\beta = 0.55$ budget, π achieves a budgeted recall of 0.2133, compared with 0.1829 for last-attention, an absolute gain of 3.04 percentage points. At $\beta = 0.75$, where substantially more regions can be upgraded, the corresponding recalls are 0.9657 and 0.9472. These results show that the learned predictor provides a consistent ranking advantage over directly reusing the previous interval's attention under both budget settings.

The offline ranking metric and the end-to-end task results measure different aspects of the allocation problem. Budgeted recall assigns equal credit to all $h1$-positive regions, whereas downstream quality also depends on which demanded regions are sensitive to low-precision representation. We therefore use this analysis to evaluate the future-demand predictor itself, and the same-budget end-to-end comparison in Section 5.5 to measure its effect when combined with demotion-risk ranking and residual refinement.

B Ablation Configuration and Recovery Controls

Purpose and scope. The first question is whether a residual-augmented representation contains information that is absent from the low-bit base. The second question is whether allowing a region to be promoted repeatedly after demotion improves task-level performance. The recovery-control experiment below addresses only the second question. All recovery-control arms use the same low-bit base, the same residual-augmented hot representation, the same byte budget, and the same ranking signal; they differ only in the permitted promotion history.

Common protocol. All three recovery-control arms use the same model checkpoint, generation configuration, region/block definition, task identifiers, decoding procedure, judge, and repetitions. The recovery-control arms share task identifiers without re-sampling; their τ^2 results use the same 100-task evaluation set as the main results. The configuration matrix is given in Table 6.

Initial allocation and promotion semantics. The initial hot set of the demote-only and repeatable arms is identical. After a region is demoted, the demote-only arm permanently excludes that region from subsequent promotion decisions. The repeatable arm may promote the same region again when it is selected by the controller. The frozen arm does not perform any post-initial promotion. Thus, under the configuration in Table 6, the comparison is intended to isolate promotion reversibility. Task-level results are reported in Table 7.

Interpretation of the recovery-control results. The recovery-control comparison holds the $Q4$ base, $B + R$ hot representation, host residual, ranking signal, and logical KV-byte budget fixed to examine the benefit of allowing a region to be promoted again after demotion. The corresponding task-level results are shown in Table 7.

C Algorithmic Details

Offline supervision and predictor fitting. Algorithm 1 constructs labels from complete training trajectories and paired replays. Feature normalization and label thresholds are fitted on training data; validation selects hyperparameters, and the predictors remain frozen during evaluation.

Table 6: Configuration matrix for the recovery-control ablation. The three arms use the same representation and byte budget. The only intended difference is whether a region may be promoted again after demotion.

Arm	Ranking signal	Cold/base repr.	Hot repr.	Residual location	Re-promotion rule	Shadow/ direct-Q8	Budget accounting
Frozen	πd	Q4	$B + R$	Host	No promotion after initial allocation	None	28.3% logical KV-byte budget
Demote-only	πd	Q4	$B + R$	Host	Initial hot set shared with full arm; each region promoted at most once; no return after demotion	None	28.3% logical KV-byte budget
Repeatable / full	πd	Q4	$B + R$	Host	Repeated promotion after demotion allowed	None	28.3% logical KV-byte budget

Notes. Here $B + R$ denotes reconstruction from the low-bit base and the separately stored quantized residual. Quantization metadata, padding, indexing, temporary buffers, and host-resident residuals follow the same accounting convention in all three arms.

Table 7: Task-level results for the recovery-control ablation.

Arm	AIME24	AIME25	MATH-500	τ^2	BFCL v4
Frozen	3.3	3.3	10.0	17.0	8.0
Demote-only	26.7	46.7	51.0	20.0	44.5
Repeatable / full	30.0	56.7	61.0	22.0	55.5

Algorithm 1: Offline targets and predictor fitting

Input: Complete BF16 training trajectories with boundaries and attention; frozen language model; selected training replay pairs (p, t).
Output: Frozen demand predictor θ, risk predictor ϕ, training normalization, and thresholds τ_A, τ_d.

for each complete training trajectory **do**
  Partition history into regions and decision intervals as in Section 4.1.
  for each τ_t with a complete next interval I_t **do**
    for each $p \in \mathcal{P}_t$ **do**
      Record raw causal features from I_{t-1}, region age and size, block count, and missing-observation mask.
      Measure $A^{\text{next}}_{p,t}$ from reference attention in I_t (Eq. (2)).
      if (p, t) is a selected replay pair **then**
        Teacher-force the same reference continuation with anchor C_8 and single-region demotion $C_4(p)$.
        Record $\Delta^{4\leftarrow 8}_{p,t}$ from the maximum positive NLL difference (Eq. (5)).
Fit feature normalization and calibrate τ_A, τ_d using training data only; form $y^{\pi}_{p,t}$ and, where replayed, $y^{d}_{p,t}$.
Train the per-region GRU demand head on successive boundaries with weighted BCE and auxiliary attention Huber loss (Eq. (10)).
Independently train the MLP risk head on damage-labeled pairs using weighted BCE (Eq. (11)).
Select hyperparameters on validation data; freeze normalization, thresholds, and both predictors before testing.

The replay changes only the target region's precision: the model parameters, input tokens, other regions, and teacher-forced continuation are fixed. Incomplete next intervals have no demand label, and regions without a selected replay have no risk label.

Reversible representation and migration. Algorithm 2 separates encoding a new region (or physical KV block) from switching the GPU residency of existing residuals. Host sidecars are never discarded when a region cools, so a later upgrade does not reconstruct information from the low-bit base alone.

Algorithm 2: Residual encoding and hot-set transition

Input: Newly available original KV states X_p; current GPU hot set H; selected next hot set S.
Encode(X_p), while the original KV states are available:
  $B_p \leftarrow \mathcal{Q}_4(X_p)$; $R_p \leftarrow \mathcal{Q}_4(\mathcal{Q}_8(X_p) - B_p)$ (Eq. (6)).
  Keep B_p on the GPU and R_p on the host; start p in the base-only state.
Switch(H, S) at a decision boundary:
  for $p \in H \setminus S$ **do** release the GPU copy of R_p; retain its host sidecar.
  for $p \in S \setminus H$ **do** load R_p from the host to the GPU.
  Set $H \leftarrow S$; use $B_p + R_p$ for $p \in H$, and B_p otherwise.
Return: Updated GPU hot set H; all bases and host residuals persist.

The representation $B_p + R_p$ approximates the 8-bit reference; it is not claimed to recover the original KV states exactly. Release-before-load avoids holding both the departing and arriving GPU residual sets simultaneously.

Causal budgeted online allocation. Algorithm 3 calls the representation procedure above and uses only completed-interval statistics when deciding the next hot set. The budget is proportional to current INT8 history bytes, and the greedy ordering uses actual incremental GPU byte costs.

Algorithm 3: Online predictive precision allocation

Input: Frozen predictors θ, ϕ and normalization; budget ratio β; actual GPU byte-cost functions.
State: Region GRU states $\mathbf{h}_p$ (zero for new regions); GPU bases and host residuals; current hot set H.
for each decision boundary τ_t **do**
  Form $\mathcal{P}_t$; read attention from the completed I_{t-1}, or mark the observation missing.
  for each $p \in \mathcal{P}_t$ **do**
    Construct $\mathbf{z}_{p,t}$ from available history (Eq. (3)); update $\mathbf{h}_{p,t}$ with the GRU (Eq. (4)).
    Predict $\pi_{p,t}$ by Eq. (4) and $d_{p,t} \leftarrow \sigma(f_\phi(\mathbf{z}_{p,t}))$; set $s_{p,t} \leftarrow \pi_{p,t} d_{p,t}$.
    Set $\Delta c_p \leftarrow c_{B+R}(n_p) - c_4(n_p)$; $\rho_{p,t} \leftarrow s_{p,t}/\Delta c_p$.
  Reserve all bases: $U \leftarrow \beta M_t^{\mathrm{INT8}} - \sum_{p \in \mathcal{P}_t} c_4(n_p)$; $S \leftarrow \varnothing$.
  for p in decreasing $\rho_{p,t}$ (ties by fixed region ID) **do**
    if $\Delta c_p \le U$ **then** $S \leftarrow S \cup \{p\}$; $U \leftarrow U - \Delta c_p$.
  $H \leftarrow$ **Switch**(H, S) (Algorithm 2); release departing GPU residuals first.
  Generate I_t using $B_p + R_p$ for $p \in H$, and B_p for other existing regions; keep their assignments fixed.
  For each new KV block g, **Encode**(X_g) (Algorithm 2); keep it base-only until the next boundary.
  Accumulate attention over I_t for the decision at τ_{t+1}, never for the current decision.

The procedure assumes $\Delta c_p > 0$ and a feasible base budget. New blocks preserve the within-interval proportional constraint when $c_4(g) \le \beta c_8(g)$; otherwise bytes must be reserved or the allocation updated earlier, as in Eq. (9).

D Numerical Data for the Length Curves

D.1 Long-CoT Results

Table 8: Qwen3-8B on AIME24, AIME25, and MATH-500.

Dataset	Method	2K	4K	16K	32K
AIME24	DynamicKV	0.00	6.67	13.33	30.00
AIME24	KIVI	0.00	3.33	10.00	26.67
AIME24	PM-KVQ	0.00	0.00	10.00	26.67
AIME24	MiKV	0.00	0.00	6.67	13.33
AIME25	DynamicKV	0.00	13.33	40.00	56.67
AIME25	KIVI	0.00	6.67	23.33	50.00
AIME25	PM-KVQ	0.00	0.00	36.67	53.33
AIME25	MiKV	0.00	0.00	0.00	36.67
MATH-500	DynamicKV	18.20	36.40	54.00	61.00
MATH-500	KIVI	20.00	26.60	53.40	60.00
MATH-500	PM-KVQ	20.00	26.60	40.00	46.60
MATH-500	MiKV	20.00	33.40	33.40	33.40

Table 9: Qwen3.5-27B on AIME24, AIME25, and MATH-500.

Dataset	Method	2K	4K	16K	32K
AIME24	DynamicKV	0.00	6.67	63.33	86.67
AIME24	KIVI	0.00	0.00	13.33	86.67
AIME24	PM-KVQ	0.00	0.00	13.33	56.67
AIME24	MiKV	0.00	0.00	43.33	56.67
AIME25	DynamicKV	0.00	20.00	70.00	86.67
AIME25	KIVI	0.00	13.33	63.33	86.67
AIME25	PM-KVQ	0.00	0.00	63.33	76.67
AIME25	MiKV	13.33	13.33	70.00	76.67
MATH-500	DynamicKV	20.00	40.00	93.00	93.00
MATH-500	KIVI	6.60	40.00	80.00	80.00
MATH-500	PM-KVQ	18.00	33.40	93.40	93.40
MATH-500	MiKV	6.60	38.00	91.00	93.00

Table 10: R1-Distill-7B on AIME24, AIME25, and MATH-500.

Dataset	Method	2K	4K	8K	16K	32K
AIME24	DynamicKV	13.33	13.33	30.00	30.00	30.00
AIME24	KIVI	0.00	13.33	30.00	30.00	30.00
AIME24	PM-KVQ	0.00	13.33	30.00	30.00	30.00
AIME24	MiKV	0.00	0.00	0.00	0.00	0.00
AIME25	DynamicKV	13.33	30.00	50.00	56.67	63.33
AIME25	KIVI	13.33	50.00	50.00	50.00	63.33
AIME25	PM-KVQ	13.33	50.00	50.00	50.00	63.33
AIME25	MiKV	0.00	0.00	0.00	0.00	0.00
MATH-500	DynamicKV	50.00	70.00	76.60	76.60	76.60
MATH-500	KIVI	66.60	86.60	86.60	86.60	86.60
MATH-500	PM-KVQ	46.60	60.00	80.00	86.60	86.60
MATH-500	MiKV	—	—	—	—	—

D.2 NUMERICAL RESULTS ON τ^2-BENCH

Our τ^2-bench evaluation covers the complete official test partition of v1.0.1: all 100 test tasks (airline 20, retail 40, telecom 40). All methods use the same 100 tasks, and Table 11 reports success rates over this combined set.

Table 11: Qwen3-8B and Qwen3.5-27B on the 100-task τ^2-bench evaluation set.

Model	Method	0.5K	2K	4K	16K	32K
Qwen3-8B	DynamicKV	0.00	0.00	4.00	14.00	22.00
Qwen3-8B	KIVI	0.00	0.00	7.00	10.00	10.00
Qwen3-8B	PM-KVQ	0.00	0.00	0.00	7.00	7.00
Qwen3-8B	MiKV	0.00	0.00	3.00	17.00	17.00
Qwen3.5-27B	DynamicKV	0.00	0.00	0.00	49.00	51.00
Qwen3.5-27B	KIVI	0.00	0.00	0.00	45.00	45.00
Qwen3.5-27B	PM-KVQ	0.00	0.00	0.00	38.00	42.00
Qwen3.5-27B	MiKV	0.00	0.00	0.00	28.00	28.00

D.3 NUMERICAL RESULTS ON BFCL V4

We evaluate tasks from `multi_turn_base` and agentic tasks. All methods use the same task IDs and the official BFCL V4 evaluator. Multi-turn tasks must pass both state-based and response-based checks at every turn; agentic tasks follow their category-specific official scoring rule.

Table 12: Qwen3-8B task success (%) on the 200-task BFCL V4 agentic subset.

Method	0.5K	2K	4K	16K	32K
DynamicKV	0.00	0.00	11.00	54.50	55.50
KIVI	0.00	0.00	5.50	37.00	39.00
PM-KVQ	0.00	0.00	16.50	44.00	45.00
MiKV	0.00	0.00	11.00	35.50	37.00