

Persistent Memory Mechanisms and Storage Architecture Design for Agent-based Information Systems

Shefeng Li

Syracuse University, Syracuse, New York, 13244, USA

Keywords: Intelligent agent information system; persistent memory; version state; storage architecture; long-term memory retrieval

Abstract: To address the problem of expired memories still participating in current retrievals after explicit state updates in long-running agents, while historical versions need to be continuously retained, this study proposes a version validity-driven persistent memory dual-storage mechanism, VPM-DS. This method maintains the current/historical state using an entity-attribute version chain. Through constraints of current state uniqueness, historical integrity, and temporal monotonicity, memory validity is mapped to retrieval participation eligibility, and a relationship model between the historical working set and retrieval latency is established. This study validates the mechanism based on Memory Remembering, LongMemEval-Cleaned Knowledge Update, and continuous write loads. Under quarterly conditions, VPM-DS only improves FAMA by 0.3 percentage points compared to State-aware Single-Store; under 9k conditions and a 30% historical version ratio, P95 decreases by 10.8%. The results show that version identification contributes the main semantic gains, and the independent value of dual storage is mainly reflected in the control of the active working set under high historical loads.

1 Introduction

Large language model-driven agents are shifting from single-question-answering tools to information systems capable of performing tasks across sessions, maintaining user states, and calling external resources. Memory is responsible for state preservation, history recovery, and subsequent task support. In customer service agents, task execution agents, and user profiling agents, address, authorization status, responsible person, and task stage may directly affect tool call parameters or business actions. Once the old state is recalled, it will not only affect the answer text but may also change subsequent system operations. Existing research regards memory formation, updating, retrieval, and forgetting as important links in the continuous operation of LLM agents [1], and external persistent memory has also been used to alleviate the constraints of limited context on long-term interactions [2]. When the running cycle extends from a single round to several weeks or even months, the questions that the system needs to answer also shift from "whether to save history" to "which history still represents the current state".

User facts, task stages, and business entities in a personalized agent are constantly changing. Existing research has maintained the personalized state across rounds through long-term fact updates [3], while methods such as SAGE reduce the information burden caused by long history through selective memory management [4]. Studies on shared persistent memory have shown that long-running conversational agents also need to handle cross-user context sharing, source tracking, and persistent memory governance [5]. External SQL memory further illustrates that language models can explicitly write interaction facts into an independent database and retrieve them on demand in subsequent tasks [6]. However, persistent storage does not equate to continuous participation in default retrieval. If a user's work location changes sequentially from A to B and C, the three versions may still maintain a high degree of similarity in the vector space; a unified index will continuously expose the old state, while directly deleting the old version will impair auditing and historical traceability.

Historical information selection studies have shown that expanding the historical scale does not automatically improve retrieval quality [7]. When new and old evidence appear simultaneously, the model also needs to determine which evidence is still valid [8]. Time knowledge studies also show that dynamic facts have clear evolutionary relationships and cannot be simply attributed to uniform time decay [9]. Therefore, for long-running agent information systems, the more important question is whether to still enter high-frequency current state queries after memory failure, rather than whether history still needs to be saved.

To address this issue, this study investigates the version-aware Persistent Memory with Dual Store (VPM-DS) mechanism, a persistent memory organization mechanism driven by version validity. The study only addresses entity states with explicit single-valued substitution relationships, not extending to multi-valued facts, implicit causal updates, or general temporal reasoning. This study sets three research questions: RQ1, can explicit version management improve current state consistency and suppress expired memories? RQ2, when version states are consistent, can Current/Historical physical separation produce storage and retrieval benefits independent of version judgments? RQ3, how do these benefits change with historical scale and the proportion of historical versions? This study does not introduce new inference modules; instead, it defines memory validity as eligibility for retrieval participation and analyzes its source of influence and applicable boundaries through state invariants, working ensemble models, and controlled ablation analysis.

2 Research Methods

2.1 Research Object, System Requirements, and Data

This study focuses on long-running agent information systems with continuously evolving states and requiring historical traceability. The system must simultaneously meet three requirements: current business queries prioritize using valid states; replaced states cannot be permanently deleted due to failure; and version relationships can be recovered from persistent metadata after storage migration anomalies. The first two correspond to current-state consistency and historical auditability, respectively, while the last corresponds to recoverable persistence.

The main experiment uses the Memory Remembering task. Weekly and monthly tests each contain 5 questions per persona, while quarterly tests contain 10 questions per persona, for a total of 200 Remembering questions. Two of the 10 persona classes are fixed for development, and the remaining eight classes constitute the official test sets of 40, 40, and 80 questions respectively. External validation uses only a subset of LongMemEval-Cleaned Knowledge Updates to avoid temporal reasoning altering the research object. The system experiment uses 9170 memory events from MemDialogue v2 and extracts verifiable state update pairs of the same entity from publicly available records to construct a continuous write load.

All memory is processed according to event time. Qwen3-4B-Instruct-2507 uniformly extracts five fields: entity, relation, value, event_time, and source_id ; when the same session contains multiple independent facts, it is split into multiple records. The extraction results and Qwen3-Embedding-0.6B vectors are generated only once and cached, and all methods read the same input. The development set and test set are isolated by person; test answers and gold evidence do not participate in state construction.

Continuous write loads also maintain three control conditions: the total amount of memory is consistent across different historical versions; the event order is consistent across all methods within the same workload seed; and new UPDATE events only replace publicly verifiable state pairs under the same state key, without rewriting the text content. This allows latency variations in system experiments to be primarily attributed to the composition of active indexes, rather than systemic changes in the total number of records, text length, or event order.

2.2 Version Validity Mechanism

A single memory is represented as

$$m_i = (id_i, e_i, r_i, v_i, t_i, c_i, z_i) \quad (1)$$

Here, e_i , r_i , v_i and represent the entity, attribute, and state value, respectively; t_i represents the event time; c_i stores the original evidence; and represents the version state. $z_i \in \{\text{current, historical}\}$ A version sequence is established for the state key. $k = (e, r)$

$$V_k(t) = \{m_k^{(1)}, m_k^{(2)}, \dots, m_k^{(n)}\}.$$

VPM-DS maintains three invariants for single-valued states: Any state key can have at most one current version.

$$\sum_{m \in V_k(t)} \mathbf{1}[z_m = \text{current}] \leq 1. \quad (2)$$

If a new version $m^{(j+1)}$ replaces the old version $m^{(j)}$, the old version must be retained in the Historical set and satisfy the following conditions:

$$\text{predecessor}(m^{(j+1)}) = m^{(j)}. \quad (3)$$

The same version chain must also maintain monotonic event times. Late data can be inserted into the historical sequence, but cannot overwrite current records with later event times. These three constraints correspond to current state consistency, historical auditability, and version chain recoverability, respectively.

The system first normalizes entity aliases and attribute names. If identical state keys exist, the new record directly enters the relation judgment; otherwise, it only retrieves three semantic candidates from the Current set of the same subject. τ_v Once the candidate similarity reaches a threshold, it uses Qwen3-4B to determine SAME, UPDATE, or INDEPENDENT. For SAME, only the source is merged, and the index continues to encode the normalized entity+relation+value ; for UPDATE, the old record is transferred to historical, and the new record is set as current; for INDEPENDENT, a new state group is created.

240 candidate pairs were stratified and extracted from the development data. Two annotators independently judged the data, and the gold label (Cohen's) was formed after discussion of disagreements $\kappa = 0.86$. The threshold sensitivity is shown in Table 1.

Table 1. Sensitivity of Candidate Thresholds in Versions

τ_v	Macro-F1	UPDATE-F1	False merger rate / %	Candidate false negative rate / %
0.76	0.795	0.731	11.3	4.8
0.79	0.819	0.756	8.1	6.2
0.82	0.832	0.771	5.9	8.0
0.85	0.824	0.759	4.4	10.9
0.88	0.801	0.726	3.1	14.7

$\tau_v = 0.82$ A high overall F1 score was achieved, so this threshold was fixed during the testing phase. Under this condition, the F1 scores for SAME, UPDATE, and INDEPENDENT were 0.85, 0.77, and 0.86, respectively. Approximately 36% of version errors were UPDATE misclassified as SAME, of which about 71% resulted in the old value remaining in the Current state; SAME misclassified as UPDATE accounted for about 24%, mainly adding unnecessary historical versions, with a relatively smaller direct impact on the current value. The impacts of the two types of errors are not the same; this difference is used to explain subsequent error propagation.

The sensitivity results did not show a monotonic advantage. Lower thresholds can reduce missed candidate detections, but may mistakenly merge records with similar semantics but different attributes into the same version chain; as the threshold increases, the false merging rate decreases, but true UPDATES are more likely to be excluded in the candidate stage. Taking the fine-grained supplement from "likes coffee" to "prefers light roast coffee" as an example, there is already a judgment boundary between SAME and UPDATE in this type of text. Therefore, 0.82 is only used as a development set selection under the current data and embedding configuration, and should not be interpreted as a universal threshold.

2.3 Effectiveness-Driven Dual Storage and Cost Model

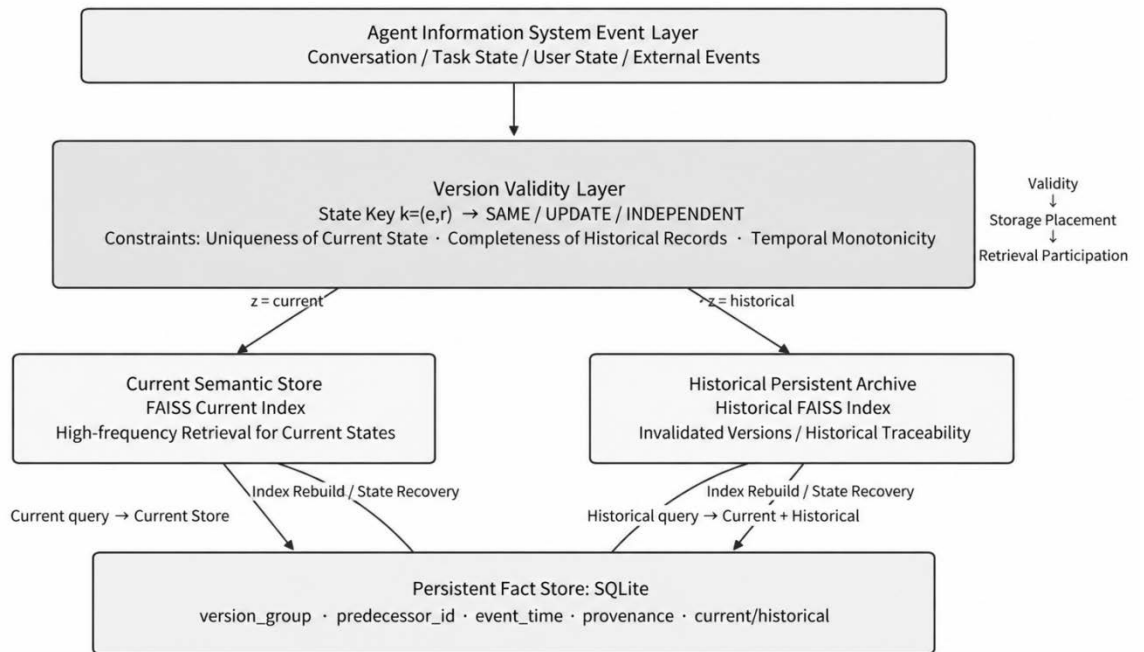


Figure 1. VPM-DS version validity-driven persistent memory and dual-storage architecture

VPM-DS does not treat the Current Store and Historical Archive as two independent storage memories, but rather organizes them into different retrieval views of a unified version fact, as shown in Figure 1. The version validity layer determines the current/historical state based on the entity-attribute version chain; SQLite stores the complete version fact and serves as the source of persistent facts; the Current Semantic Store only maintains high-frequency retrieval indexes for currently valid records, and the Historical Persistent Archive stores replaced versions and their retrieval structures. If a derived index malfunctions, the system can rebuild the corresponding FAISS index based on the version status in SQLite. In this way, state validity simultaneously constrains the storage location and default retrieval scope, and historical persistence no longer depends on older versions continuing to reside in the Current index.

Assuming total persistent memory at any given time is denoted as M_t , VPM-DS divides it into .

$$M_t = C_t \cup H_t, \quad C_t \cap H_t = \emptyset, \quad (4)$$

This C_t includes the Current Semantic Store and H_t the Historical Persistent Archive. Both types of records store text, version relationships, and source information in SQLite, and each maintains a FAISS IndexIDMap2 (IndexFlatIP) . SQLite serves as the persistent source of version state facts; if FAISS updates are interrupted, the system can rebuild the index based on the current/historical state in SQLite.

Let the total number of searchable versions be N , the number of historical versions be H , and the historical version ratio be $\rho_h = H/N$, then the size of the Current working set is .

$$N_c = N - H = N(1 - \rho_h). \quad (5)$$

This study uses IndexFlatIP for exact search. With a fixed embedding dimension, the vector scoring cost can be approximated as increasing linearly with the index size. The current query cost for a State-aware Single-Store is expressed as...

$$T_s = \alpha + \beta N + C_f, \quad (6)$$

This C_f represents the cost of state filtering and candidate replacement. VPM-DS only searches the Current set, therefore...

$$T_D = \alpha + \beta N(1 - \rho_h) + C_d, \quad (7)$$

C_d This represents the fixed overhead incurred by dual-storage metadata and access paths. The difference between the two is...

$$T_s - T_D = \beta N \rho_h + C_f - C_d. \quad (8)$$

generates latency benefits when the historical working set is sufficient to cover the additional management costs. This relationship does not presuppose that VPM-DS is always dominant, but is used to determine its applicable boundaries. $N\rho_h$

State-aware Single-Store and VPM-DS use the exact same version judgment results and ultimately return the Top-10 to the Reader. Single-Store first retrieves the Top-50 from the unified index, then filters the current records; if there are fewer than 10 records, it continues to retrieve in batches of 50 until 10 current records are obtained or the index is exhausted. The latency of all retrievals is included in P95. VPM-DS, on the other hand, directly performs the same similarity retrieval on the Current index. Thus, the core difference between the two methods lies only in whether historical records continue to participate in the current search space.

2.4 Control Methods and Experimental Protocol

The experiment setup includes Recent-K, Flat Persistent, Static Time-decay, VPM-DS –

Version , State-aware Single-Store, and Full VPM-DS. – Version disables SAME/UPDATE state transitions, keeping all records active; – Store , or State-aware Single-Store, retains version state without physical separation. Static Time-decay uses...

$$\text{Score}(q, m) = \text{Sim}(q, m)e^{-\lambda_d \Delta t},$$

on Δt a daily basis, λ_d is selected as 0.02 d^{-1} from $\{0.005, 0.01, 0.02, 0.05, 0.10\}$ in the development set.

Memora uses FAMA, Memory Presence Accuracy (MPA), and Forgetting Absence Accuracy (FAA) for evaluation. FAMA is calculated according to the official definition:

$$FAMA = \max[0, MPA - \lambda_f(1 - FAA)]. \quad (9)$$

LongMemEval only reports Evidence Recall@10 and MRR for Knowledge Update. The system experiments were set up with four historical scales (1k, 3k, 5k, and 9k) and three actual historical version ratios (5%, 15%, and 30%). Ten independent workload seeds were generated for each combination, and each seed was timed five times after two rounds of warm-up. Semantic differences were calculated using 95% CI through 10,000 paired bootstrap iterations, and paired comparisons were performed using the Wilcoxon test and Holm correction. The system metrics also report rank-biserial correlation as the effect size.

Total Persistent Bytes is defined as the sum of the sizes of the SQLite, Current FAISS, and Historical FAISS files, with a WAL checkpoint performed before measurement. An additional 100 migration interruption injections are performed, randomly stopping the process after the SQLite state transaction is committed but before the FAISS modification is complete, to verify the version state recovery capability. This test only evaluates consistency and is not included in the normal P95 statistics.

3. Research Results

3.1 Version Recognition and Long-Term Memory Semantic Representation

The Memora Remembering results are shown in Table 2. After canceling version conversion, Quarterly FAMA was 28.2, only slightly higher than Flat Persistent's 27.6; after adding version status, State-aware Single-Store improved to 31.3. VPM-DS further reached 31.6, but the additional increment was only 0.3 percentage points. Numerically, the improvement after adding version status is significantly greater than the changes brought about by subsequent physical database sharding.

Table 2 Memora Remembering master results and ablation results

method	FAMA-W	FAMA-M	FAMA-Q	MPA-Q/%	FAA-Q/%
Recent-K	50.6	30.4	26.7	49.3	54.8
Flat Persistent	54.4	30.0	27.6	50.4	54.3
Static Time-decay	55.7	32.1	29.0	51.1	55.8
VPM-DS-Version	54.8	30.7	28.2	50.9	54.9
State-aware Single-Store	58.0	34.8	31.3	52.6	57.3
VPM-DS	58.3	35.1	31.6	52.9	57.6

Under quarterly conditions, the FAMA difference between VPM-DS and State-aware Single-

Store was 0.3 percentage points, with a 95% CI of [-1.3, 1.8], a Holm-corrected $p=0.67$, and a rank-biserial correlation of 0.06; the difference compared to Flat Persistent was 4.0 percentage points, with a 95% CI of [0.6, 7.4], a Holm-corrected $p=0.021$, and an effect size of 0.31. The former comparison did not show an independent significant semantic advantage, while the latter comparison showed that version state management brought a more significant overall improvement.

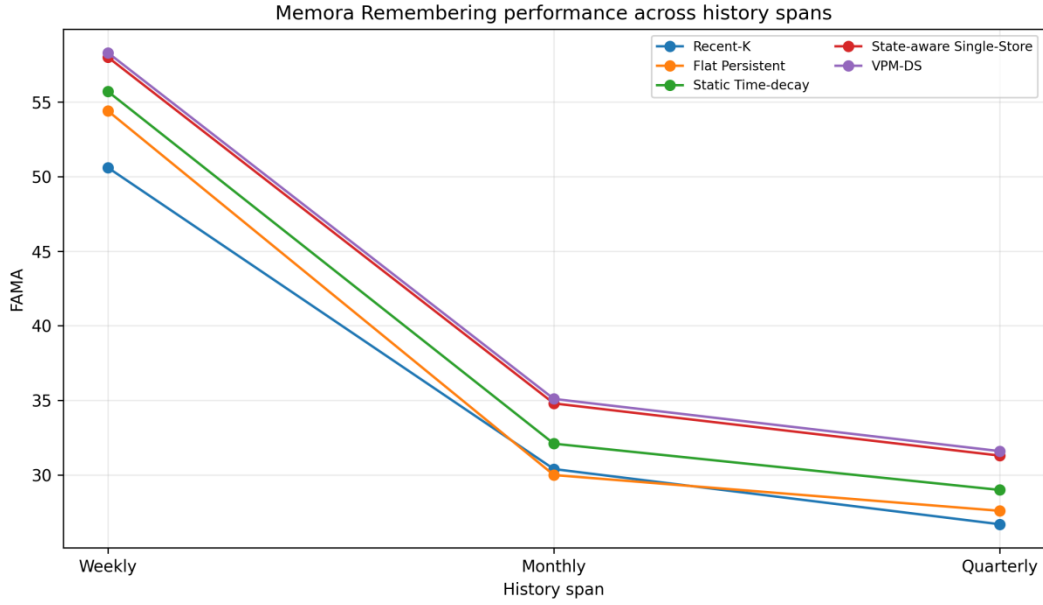


Figure 2. FAMA changes of Memora Remembering across different historical spans

As shown in Figure 2, in LongMemEval-Cleaned Knowledge Update, the Evidence Recall@10 for State-aware Single-Store and VPM-DS are 0.803 and 0.808, respectively, with a difference of 0.005 and a 95% CI of [-0.013, 0.022]. After Holm correction, $p=0.58$, and the rank-biserial correlation is 0.07; the corresponding MRRs are 0.552 and 0.556. External data also did not amplify the semantic benefits of dual storage, and the results are consistent with Memora.

Version error diagnostics show that UPDATE→SAME has the greatest impact on the current query because the old version will incorrectly remain in the Current state. While SAME→UPDATE causes unnecessary history migration, it usually retains a semantically equivalent new current version. The propagation consequences of the two types of errors are not the same, which explains why UPDATE's F1 score is lower than the other two types and requires more attention.

3.2 System Contributions and Applicability Boundaries of Dual Storage

Table 3 compares only State-aware Single-Store and VPM-DS using the exact same version labels to observe the impact of physical storage organization separately.

Table 3 System performance under different historical scales and historical version ratios

HistoryScale	ρ_h	Single-Store P95/ms	VPM-DS P95/ms	P95 change / %	PersistentBytes changes
1k	5%	3.50	3.65	+4.3	+1.2%
1k	15%	3.40	3.35	-1.5	+1.1%
1k	30%	3.60	3.68	+2.2	+1.3%
3k	5%	5.30	5.20	-1.9	+0.8%

HistoryScale	ρ_h	Single-Store P95/ms	VPM-DSP95/ms	P95 change / %	PersistentBytes changes
3k	15%	5.20	5.17	-0.6	+0.8%
3k	30%	5.40	5.25	-2.8	+0.9%
5k	5%	7.60	7.72	+1.6	+0.6%
5k	15%	7.50	7.08	-5.6	+0.6%
5k	30%	7.70	7.05	-8.4	+0.7%
9k	5%	12.90	12.75	-1.2	+0.4%
9k	15%	12.70	12.35	-2.8	+0.5%
9k	30%	13.00	11.60	-10.8	+0.5%

At low historical version ratios, VPM-DS did not show stable gains, with the 1k/5% and 5k/5% conditions even showing slightly slower returns. Under the 5k/30% condition, the P95 decreased by 0.65 ms, with a 95% CI of [-1.12, -0.21], a Holm-corrected $p=0.041$, and a rank-biserial correlation of 0.62. Under the 9k/30% condition, the decrease was 1.40 ms, with a 95% CI of [-2.08, -0.78], a corrected $p=0.014$, and an effect size of 0.78. The decrease direction was consistent with the 9k/15% condition, but the corrected $p=0.11$, still not reaching a significant level. Overall, the time-delay gains only became clear when both the historical size and historical version ratio were high.

Regressing the P95 gains of the 12 workloads yields $\dots N\rho_h$

$$\Delta T_{95} = 0.000532N\rho_h - 0.153,$$

$R^2 = 0.91$ According to the model, the average net benefit range only begins after a historical work set reaches 288 records. However, under certain low-scale conditions, fixed costs and operational noise can still alter the results; therefore, 288 records are more suitable as an empirical boundary in the current experiment than a definitive threshold.

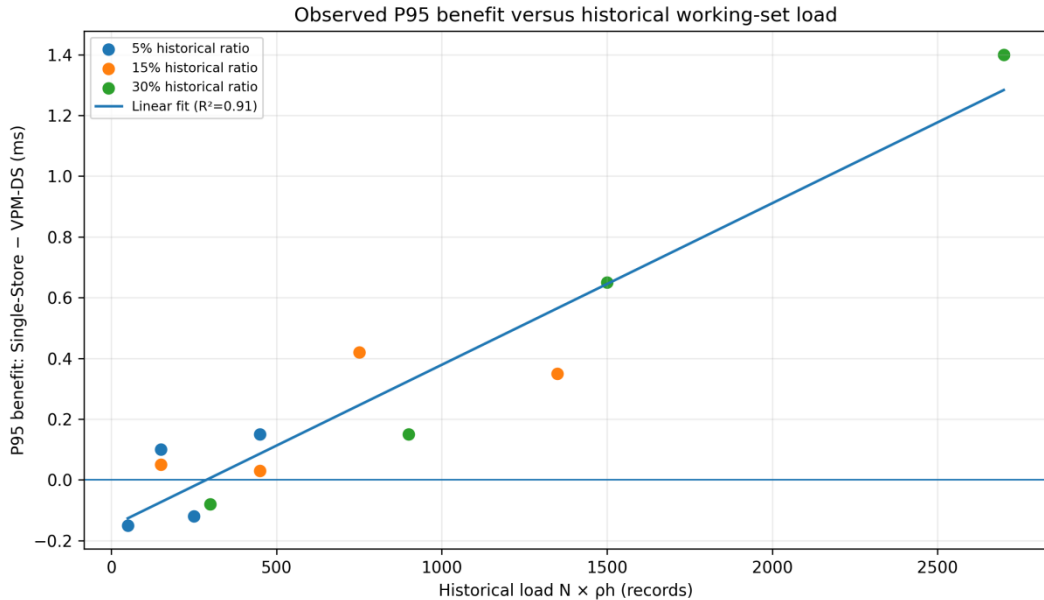


Figure 3. Relationship between historical working sets and P95 revenue

Compared to State-aware Single-Store, VPM-DS only increases the persistent space by about 0.4%–1.3%, mainly due to the second index header and the detached ID mapping; the total vector size remains unchanged. In fault injection, 96 out of 100 interruptions were recovered through

incremental index updates, and 4 triggered full FAISS rebuilds; after recovery, both Current uniqueness and version chain integrity were restored. This indicates that SQLite, as a version fact source, can provide a recovery basis for derived vector indexes, but this result does not necessarily mean that concurrent writes or distributed consistency have been verified.

The actual reduction of the Current index does not perfectly align with the nominal historical version ratio. SAME relationships do not generate new historical vectors, and some candidates may remain in the Current index due to version misidentification. Therefore, the historical version ratio is more suitable for describing the intensity of state evolution in a workload; what truly affects retrieval costs is the number of vectors that ultimately enter the Current index. This also explains why, under the same conditions ρ_h , different seeds can still exhibit some P95 fluctuations.

4. Discussion

The research results indicate that the main contradiction in long-term state memory arises first from "validity judgment," rather than the number of databases. There is a significant difference between Version ablation and State-aware Single-Store, while the FAMA difference between Single-Store and VPM-DS is minimal. Version state determines which facts represent the current world, and physical separation increases the cost of searching these facts. When an UPDATE is misjudged as SAME, the error directly enters the Current working set, and dual storage cannot remedy an already erroneous state judgment. Therefore, improving the stability of version resolution is more likely to yield semantic benefits than continuing to add memory layers.

The value of the storage architecture is mainly reflected in accessing the working set, rather than inference capability. Equation (8) and Figure 3 together show that the system benefits are mainly N_{p_h} driven by the historical working set. When the history is short or the proportion of historical versions is low, the management overhead of dual indexes is similar to that of the current working set reduction, and VPM-DS may not be superior. As the historical size and the proportion of historical versions increase simultaneously, the unified index still needs to perform vector scoring on a large number of invalid versions, while VPM-DS only processes the current subset, and the P95 gap gradually widens. This phenomenon is consistent with existing research on the impact of index size and retrieval configuration on the performance of vector databases [10].

This result also explains why the total storage space did not decrease. VPM-DS never deletes the historical version, thus retaining historical auditing, state tracking, and task replay capabilities; it reduces the number of active vectors on the current query path. Therefore, Persistent Memory and Storage Architecture form a clear system division of labor: version state maintains current consistency, the Historical Archive supports historical auditing, and the SQLite fact source and FAISS reconstruction mechanism ensure the recoverability of derived indexes. This design is closer to state governance in long-term agent information systems than simply adjusting RAG retrieval parameters.

From an information systems perspective, this division of labor also limits the appropriate use of the method. The Current Store is not a new fact database, but rather a frequently accessed view of the complete persistent state; the Historical Archive is not a cold data deletion area, but rather a long-term fact layer required for auditing and recovery. Only when both share the same version of the fact source can retrieval efficiency optimization not compromise state traceability. This is also the main system implication that distinguishes VPM-DS from ordinary hot and cold data tiering.

The boundaries of this study are relatively clear. The method only applies to single-value, explicit state substitution, and does not cover aggregate facts, implicit causal changes, and multi-hop temporal reasoning; the current/historical query intent is given by the benchmark or explicit temporal conditions, so the real Query Router has not yet been verified. The system experiment is

still based on 9k-level memory and a single-process Colab environment, and does not handle concurrent writes by multiple agents. The approximate linear relationship of equations (6)-(8) is based on the exact scan of IndexFlatIP; if approximate indexes such as HNSW or IVF are used instead, the cost function and benefit boundary need to be re-estimated. Therefore, the conclusions of this study are only applicable to agent information systems with continuous evolution of explicit states, frequent current queries, and gradual accumulation of historical versions, and cannot be directly extended to general long-term memory schemes.

5. Conclusion

RQ1 results indicate that state validity judgment is the main bottleneck for the correctness of long-term agent memory, with UPDATE misjudgments particularly prone to directly polluting the Current working set. RQ2 results show that when version labels are consistent, Current/Historical separation adds almost no semantic benefit; its main function is to shrink the current retrieval working set while preserving historical auditing and recovery capabilities. RQ3 results further show that this benefit is influenced by both historical scale and the proportion of historical versions, and may not be able to offset the dual-storage management costs under low historical load. Based on this boundary, VPM-DS is more suitable for agent information systems with explicitly single-valued states that continuously evolve, frequent current queries, and the need for historical tracing and recovery.

References

- [1] Zhang Z, Dai Q, Bo X, et al. A Survey on the Memory Mechanism of Large Language Model-based Agents[J]. *ACM Transactions on Information Systems*, 2025, 43(6): Article 155, 1-47. DOI: 10.1145/3748302.
- [2] Zheng S, He K, Yang L, Xiong J. MemoryRepository for AI NPC[J]. *IEEE Access*, 2024, 12: 62581-62596. DOI: 10.1109/ACCESS.2024.3393485.
- [3] Wang K, Yan S, Yuan H, et al. Toward Interpretable and Persistent Personalization: A Memory-Augmented Agent Framework for LLM-Based Travel Planning[J]. *IEEE Access*, 2025, 13: 193125-193141. DOI: 10.1109/ACCESS.2025.3630495.
- [4] Wang, Z. (2026). Research on Operation Data Analysis and Efficiency Improvement of the New Energy Industry Based on Machine Learning. *Procedia Computer Science*, 285, 605-612.
- [5] Cui, N. (2024, December). Research on Adaptive Urban Traffic Green Wave Control Algorithm and Safety Based on Reinforcement Learning DQN-COP. In *International Conference on Big Data Analytics for Cyber-Physical System in Smart City* (pp. 379-387). Singapore: Springer Nature Singapore.
- [6] Wang, B. (2025, November). Research on Performance Improvement of Personalized Recommendation System for E-commerce Platform Based on Big Data and Collaborative Filtering Algorithm. In *International Conference on Computational Technologies for Research in Data Analytics* (pp. 36-46). Cham: Springer Nature Switzerland.
- [7] Hou, Y. (2026, March). Adaptive Upgrading of Computing Power Platforms for Large Model Workloads. In *International Conference on Computing and Communication Systems for Industrial Applications* (pp. 429-437). Cham: Springer Nature Switzerland.
- [8] Xin, H. (2026). Research on Distributed Data Pipelines and Event-Driven Architecture for Freight Visibility.
- [9] Zhang, Z. (2026). User Behavior Anomaly Identification Integrating Telemetry Data and Its Application in Consumer Finance Risk Control.

- [10] Jin, C. (2026). *Application of Data Mining Technology in Predicting User Churn and Retention Strategies in E-Commerce.*
- [11] Liu, S. (2026). *Design and Implementation of Retrieval Enhancement Generation Method for Enterprise Knowledge Base.*
- [12] Zhang, Z. (2026). *Research on Service-oriented Packaging of AI Systems and Data-driven Decision Support for Real-world Business Processes.* *Engineering Advances*, 6(3).
- [13] Zhang, Y. (2025, January). *Research on the Application and Optimization of Multi Dimensional Data Model Based on Kylin in Enterprise Technology Management.* In *International Conference on Innovative Computing* (pp. 234-241). Singapore: Springer Nature Singapore.
- [14] Qian, X. (2026, July). *Application of the Combination Model of Prophet and LSTM in Multi-Level Inventory Demand Prediction for Retail Supply Chain of CPG-Consumer Packaged Foods.* In *2026 5th International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)* (pp. 1-7). IEEE.
- [15] Jin, C. (2026). *Construction and Empirical Study of Credit Risk Assessment Index System for Small and Micro Enterprises in Commercial Banks.*
- [16] Liu, S. (2026). *Automatic Generation Method and Implementation of Web Front-End Components Based on Generative AI.*
- [17] Zhang, Z. (2026). *User Behavior Device Reliability Joint Modeling and Intelligent Risk Control Practice In Consumer Technology.*
- [18] Zhang, Y. (2026). *Exploration of Enterprise Big Data Microservice Architecture Based on Domain-Driven Design (DDD).* *Procedia Computer Science*, 282, 1994-2003.
- [19] Jingzhi Yin. (2026). *Constructing a CMBS Default Risk Sentiment Index Using Financial Text Embeddings and Evaluating Its Predictive Effectiveness.* *Engineering Advances*, 6(1), 50-54.
- [20] Fei, W., Xue, W., & Wang, B. (2025). *Research on the innovative path of rural education teaching methods from the perspective of PBE and PBL integration.* *Int. Acad. J. Humanit. Soc. Sci.*
- [21] Fei, W. (2026). *Families Of Chinese Background In Rural Alabama: Education, Immigration, And Identity.*
- [22] Wendan Xue, Wen Fei. *An Analytical Study on the Differences and Determinants of Basic Education Resource Allocation in Rural China and the United States.* *International Journal of Educational Research and Development*, 1(3), 13-17.
- [23] Boli Wang, Wen Fei. *Innovations and practices in the classroom for rural aesthetic education in the perspective of school integration.* *International Journal of Educational Research and Development*, 3(1), 36-41.
- [24] Fei, W., & Qian, J. (2026). *Application and Exploration of Reverse Instructional Design in the Teaching of the Primary School Physical Education" Walking and Playing" Unit.* *Journal of Sociology and Education*, 2(2).