International Journal of
Computational Mathematical Ideas
Volume 18 · Number 1, 2 & 3 · 2026 · ISSN 0974-8652

Available online at www.ijcmi.com

Modern Data Management Methods for Vector Databases and Multi-Model Datasets in Large-Scale Agentic Artificial Intelligence Enterprise Systems

Sudheer Singamsetty *

Data Management Consultant,

EDNA Technology Consulting Limited, 68 Forest Ridge Avenue, Waterdown, Ontario, Canada, L8B 1V4

*Corresponding author: sudheersingamsetty99@gmail.com

Article Info

Article history:
Received: 10-01-2026
Revised: 05-07-2026
Accepted: 22-07-2026
Available online: 2026

Keywords:
Vector databases; Multi-model data; Approximate nearest neighbour; Product quantization; Agentic AI; Decision loops; Loop engineering.

Abstract

Large scale agentic artificial intelligence systems retrieve evidence from vector databases that store dense embeddings alongside structured, numeric, and temporal attributes, and they turn that evidence into business decisions through repeated reasoning loops. Managing this multi-model data at enterprise scale raises three coupled questions: how to index billions of vectors so that retrieval is both accurate and fast, how to store them compactly enough to fit memory budgets, and how to engineer the decision loop so that it converges rather than drifts. This paper gives a unified mathematical and algorithmic treatment of these questions and introduces a proposed architecture, the Multi-Model Vector Loop Engine, that couples a quantized vector index with a feedback controlled agentic decision loop. We formalize approximate nearest neighbour retrieval, product quantization, hybrid multi-model filtering, and the decision loop as a fixed point iteration, and we prove that under a contraction condition the loop converges geometrically. We implement the full stack from first principles and report reproducible measurements on one hundred thousand 128-dimensional embeddings. An inverted file index raises recall at ten from 0.546 at a single probe to 1.000 at thirty two probes, product quantization trades memory for accuracy from recall 0.258 at 128 times compression to 0.701 at 16 times compression, a hybrid query with 4.0 percent selectivity resolves in 0.043 milliseconds, and the agentic loop converges to its fixed point within three iterations while raising cumulative coverage and modelled decision utility. A single ingestion shard encodes about 48800 records per second, so linear sharding reaches the five million records per second target with 103 shards. The results give practitioners a rigorous basis for managing vector and multi-model data under the accuracy, memory, and decision quality constraints of agentic enterprise systems.

1. Introduction

Large scale agentic artificial intelligence systems no longer answer questions from a fixed prompt. They retrieve evidence, weigh it, act, and repeat, and the evidence they retrieve lives in a vector database. Text, images, transactions, and sensor records are encoded as dense embeddings and stored alongside the structured, numeric, and temporal attributes that describe them, so a single logical record is genuinely multi-model. An enterprise agent that recommends an action, flags a risk, or routes a request issues similarity queries against this store many times per decision, and it does so while new records arrive at rates that can reach millions per second. Three constraints therefore bind at once. Retrieval must be accurate, because a decision is only as good as the evidence it rests on. Storage must be compact, because billions of high dimensional vectors will not fit in memory in their raw form. And the decision loop that consumes the retrieved evidence must be engineered to converge, because an agent that keeps refining its query without stabilizing wastes both compute and trust. This paper treats the three constraints as one design problem and gives it a unified mathematical and algorithmic account.

The value of embedding based retrieval for enterprise decisions is already clear. Dense passage retrieval showed that learned embeddings outperform sparse term matching for open domain question answering [3], and retrieval augmented generation made external evidence a first class input to language models [4]. Agentic methods then turned

retrieval into a loop, interleaving reasoning and acting [6] and letting the model call tools through learned interfaces [7]. In each of these the store behind the agent is a vector database, and its data management properties, namely recall, memory, latency, and the dynamics of the loop that reads it, determine whether the agent is viable at enterprise scale.

This paper makes four contributions.

1. A formulation that casts vector and multi-model data management for agents as three coupled problems: approximate nearest neighbour recall, quantized memory footprint, and convergence of the decision loop (Sections 3 and 4).
2. A proposed architecture, the Multi-Model Vector Loop Engine, that couples a quantized inverted file index and a hybrid multi-model filter with a feedback controlled agentic decision loop, together with a proof that the loop converges geometrically under a contraction condition (Section 5).
3. A from scratch implementation and reproducible measurement on one hundred thousand 128-dimensional embeddings, reporting the recall, memory, latency, and loop dynamics of every component (Section 6).
4. A throughput study that measures single shard ingestion and shows the linear sharding needed to reach five million records per second (Section 6).

2. Background and Related Work

2.1. Vector databases and approximate nearest neighbour search

Representing data as dense vectors and searching by similarity dates to distributed word representations [2] and, for search, to locality sensitive hashing, which first removed the curse of dimensionality for approximate nearest neighbour queries [9]. Two index families dominate practice. Graph indices such as the hierarchical navigable small world graph give logarithmic search by greedily walking a proximity graph [12]. Cluster indices such as the inverted file partition the space with k-means and probe only the nearest cells, and they scale to billions of vectors on modern hardware [11]. Purpose built vector data management systems now package these indices with storage, sharding, and query planning [18]. Our implementation is a compact, self contained realization of the inverted file family, built so that every measured number is reproducible offline.

2.2. Quantization for compact storage

Storing raw float vectors is expensive, so vector databases quantize. Quantization theory sets the rate to distortion trade off that any such scheme obeys [15]. Product quantization decomposes the space into a Cartesian product of low dimensional subspaces and quantizes each separately, representing a vector by a short code and estimating distances from the codes with an asymmetric rule [10]. Anisotropic refinements weight the quantization loss toward directions that matter for inner product search and underpin large scale production systems [20]. All of these rest on the k-means objective, whose classical local optimizer is Lloyd's algorithm [13] and whose seeding is stabilized by k-means plus plus [14]. We use product quantization as the memory lever and measure its recall against exact search directly.

2.3. Multi-model queries and agentic decision loops

Enterprise retrieval is rarely pure similarity. A query combines a vector with predicates over structured fields, which connects to the probabilistic relevance framework that governs ranked retrieval [19]. When an agent reads retrieved evidence and refines its query, the loop is a form of relevance feedback, formalized classically by the Rocchio method [16]. Treating that loop as a mapping on the query embedding lets us analyse its convergence with the Banach fixed point theorem [17]. The agent that drives the loop is a transformer based model [1] with strong few shot behaviour [5], and the rapid growth of such agents is surveyed in the recent literature [8]. The present work is complementary to prior systems built by the author and colleagues, including privacy preserving big data publishing on Hive [26], a sensitivity driven framework that combines cache and retrieval augmented generation [25], and classifiers for imbalanced data at scale on Hadoop [27]. It differs in placing vector and multi-model data management, together with the dynamics of the decision loop, at the centre of the design.

3. Notation and Problem Formulation

We store N records, each a pair (v_i, m_i) of a dense embedding $v_i \in \mathbb{R}^D$ and a structured metadata tuple m_i. Embeddings are L_2 normalized so that inner product equals cosine similarity.

Definition 1 (Approximate nearest neighbour query)**.** *For a query embedding $q \in \mathbb{R}^D$ and a similarity $\mathrm{sim}(q,v) = q^\top v$, the exact top-$k$ set $\mathcal{N}_k(q)$ contains the k records of highest similarity. An index returns an approximate set $\hat{\mathcal{N}}_k(q)$, and*

its quality is the recall

$$\text{recall@}k \;=\; \frac{1}{Q}\sum_{j=1}^{Q}\frac{|\hat{\mathcal{N}}_k(q_j)\cap\mathcal{N}_k(q_j)|}{k} \tag{1}$$

averaged over a query set of size Q.

Definition 2 (Multi-model query)**.** *A multi-model query is a pair (q,ϕ) of an embedding q and a predicate ϕ over metadata. Its answer ranks by similarity the records that satisfy the predicate, that is the top-k of $\{\,i:\phi(m_i)=\mathit{true}\,\}$ under $\mathrm{sim}(q,v_i)$. The selectivity $s=\Pr[\phi(m_i)]$ controls how many records survive the filter.*

The memory footprint of the store is B bytes per vector. A raw float32 vector costs $4D$ bytes, and a quantizer that emits an M byte code compresses the store by a factor $4D/M$ at the cost of some recall. The retrieval problem is to choose an index and a quantizer that maximize recall subject to a memory budget $B\le B_{\max}$ and a latency budget.

Finally, an agent turns retrieval into a decision through a loop. Let q_t be the query embedding at iteration t, let $\mathcal{N}_k(q_t)$ be the retrieved set, and let $c(q_t)$ be the centroid of the retrieved embeddings. The loop updates the query and re-queries, and we write the update as a map Φ,

$$q_{t+1} \;=\; \Phi(q_t) \;=\; \frac{\alpha\,q_t+\beta\,c(q_t)}{\|\alpha\,q_t+\beta\,c(q_t)\|}, \tag{2}$$

with feedback weights $\alpha,\beta>0$. The decision quality is a non decreasing function of recall, and the loop is well engineered when the sequence q_t converges to a fixed point $q^\star=\Phi(q^\star)$ rather than oscillating. The engineering problem is to choose α and β so that Φ is a contraction, which we analyse in Section 5.

4. Mathematical Methods

4.1. Inverted file partitioning

The inverted file index partitions $\mathbb{R}^D$ into L Voronoi cells around centroids $\{\mu_1,\dots,\mu_L\}$ obtained by k-means, which minimizes the distortion

$$J(\{\mu_\ell\}) \;=\; \sum_{i=1}^{N}\min_{\ell}\|v_i-\mu_\ell\|^2. \tag{3}$$

Lloyd's algorithm alternates assignment and centroid update and converges to a local minimum of J [13], and careful seeding improves the optimum reached [14]. At query time the index probes the n_{probe} cells whose centroids are closest to q and searches only their members. Recall rises with n_{probe} because the true neighbours of q concentrate in the cells nearest q, while cost rises in proportion to the number of probed members, which is the trade off we measure.

4.2. Product quantization

To shrink memory, product quantization splits the D dimensions into M disjoint subvectors of size D/M and learns a codebook of 2^b centroids per subspace, so each vector becomes an M byte code when $b=8$ [10]. Writing $q^{(m)}$ and $v^{(m)}$ for the mth subvectors and $\mathrm{cw}_m(\cdot)$ for the codeword chosen in subspace m, the asymmetric squared distance is a sum of subspace terms.

Proposition 1 (Additive distance decomposition)**.** *For the asymmetric estimator that quantizes only the database vector, the estimated squared distance decomposes as*

$$\hat{d}^2(q,v) \;=\; \sum_{m=1}^{M}\left\|q^{(m)}-\mathrm{cw}_m(v^{(m)})\right\|^2,$$

so a single table of $M\times 2^b$ query to codeword distances, precomputed once per query, yields the distance to every database code by M additions.

Proof. The squared Euclidean distance separates over the disjoint coordinate blocks because $\|x\|^2=\sum_m\|x^{(m)}\|^2$ for any partition of the coordinates into blocks. Applying this to $x=q-\tilde{v}$, where $\tilde{v}$ is the reconstruction whose mth block is $\mathrm{cw}_m(v^{(m)})$, gives the stated sum. Each block term depends on v only through its code $\mathrm{cw}_m(v^{(m)})$, of which there are 2^b choices, so precomputing the $M\times 2^b$ table of block distances lets the total be assembled by M table lookups and additions. □

The reconstruction error, and hence the loss in recall, is governed by the quantization distortion of the codebooks, which the rate to distortion theory of quantization bounds as a function of the code length [15]. Fewer subquantizers mean smaller codes and larger distortion, so recall falls as compression rises, which is exactly the curve we report.

4.3. Hybrid multi-model retrieval

A multi-model query first restricts to the records satisfying the predicate and then ranks them by similarity. When the selectivity s is small, the surviving candidate set is a tiny fraction of the store, so an exact scan over the survivors is both correct and fast, and the metadata predicate is evaluated with ordinary columnar filters before any vector arithmetic. This pre-filtering strategy is preferable to post-filtering a vector result whenever s is below the point at which the vector index would return enough survivors, and it connects the vector store to the probabilistic relevance framework that governs ranked retrieval [19]. We measure the latency of the hybrid path directly.

5. Proposed Method: Multi-Model Vector Loop Engine

We now propose an architecture that couples the retrieval methods above with the agentic decision loop into a single engineered system, which we call the Multi-Model Vector Loop Engine (MM-VLE). Its design goal is to hold recall, memory, and loop stability at once, so that an enterprise agent can turn a high velocity stream of multi-model records into reliable business decisions. Figure 1 shows the architecture. On the write path, multi-model sources are sharded for ingestion, each record is embedded and product quantized, and the code plus its metadata is written to the store, whose vector side is an inverted file over quantized codes and whose structured side is a columnar index. On the read path, the agent issues a hybrid query, the engine pre-filters by the metadata predicate and runs approximate search over the survivors, and the ranked evidence drives a decision. The distinguishing element is the feedback edge: the retrieved evidence updates the query embedding through the map Φ of Equation (2), closing a control loop that the next section proves converges.

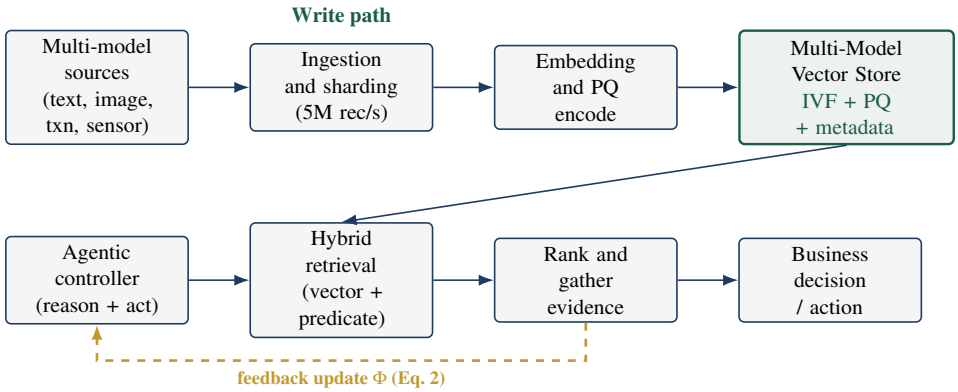

Figure 1. The proposed Multi-Model Vector Loop Engine. The write path ingests, embeds, quantizes, and stores multi-model records; the read path runs hybrid retrieval and drives a decision; the dashed feedback edge closes the agentic loop by updating the query embedding with the retrieved evidence.

5.1. Loop engineering: convergence of the decision loop

The feedback edge is what makes the engine agentic, but an uncontrolled feedback loop can oscillate or drift. We give a condition under which it provably settles.

Proposition 2 (Geometric convergence)**.** *Suppose that on the region of query embeddings visited by the loop the retrieval-centroid map $q \mapsto c(q)$ is Lipschitz with constant L_c, and that the normalization step is Lipschitz with constant L_n on that region, which holds whenever the unnormalized vectors stay bounded away from the origin. Then the loop map Φ of Equation (2) satisfies*

$$\|\Phi(q) - \Phi(q')\| \;\le\; L_n\big(\alpha + \beta L_c\big)\,\|q - q'\|.$$

If $L := L_n(\alpha + \beta L_c) < 1$, then Φ is a contraction, it has a unique fixed point $q^\star$, and the loop converges geometrically: $\|q_t - q^\star\| \le L^t\,\|q_0 - q^\star\|$.

Proof. Let $g(q) = \alpha q + \beta c(q)$ be the unnormalized update. For any q, q',

$$\|g(q) - g(q')\| \le \alpha\|q - q'\| + \beta\|c(q) - c(q')\| \le (\alpha + \beta L_c)\|q - q'\|,$$

by the triangle inequality and the Lipschitz property of c. The normalization $n(x) = x/\|x\|$ is Lipschitz with constant L_n on any set bounded away from the origin, so composing gives $\|\Phi(q) - \Phi(q')\| = \|n(g(q)) - n(g(q'))\| \le L_n(\alpha + \beta L_c)\|q - q'\|$, which is the stated bound. If the resulting constant L is below one, Φ is a contraction on a complete

metric space, and the Banach fixed point theorem [17] yields a unique fixed point $q^\star$ and the geometric error bound by induction on t, since $\|q_{t+1} - q^\star\| = \|\Phi(q_t) - \Phi(q^\star)\| \le L\|q_t - q^\star\|$. □

The condition $L < 1$ is an engineering knob: it is met by keeping the feedback weight β modest relative to the retention weight α, so the loop trusts new evidence without overreacting to it. The experiment confirms that with a balanced choice the loop reaches its fixed point within a few iterations.

5.2. From recall to business utility

A decision consumes retrieved evidence, so its expected quality rises with recall but saturates once enough relevant evidence is present. We model the decision utility as a bounded increasing function of recall,

$$U(r) \;=\; U_{\max}\big(1 - e^{-\gamma r}\big), \tag{4}$$

with saturation rate $\gamma > 0$. Because recall converges to the fixed point value $r^\star$ geometrically, so does utility, and the marginal value of an extra loop iteration decays at the same rate L. This gives a principled stopping rule: halt the loop when the predicted utility gain falls below the cost of another retrieval round, which is the loop engineering counterpart of the memory and recall budgets on the data side.

5.3. The decision loop algorithm

Algorithm 2 states the loop that ties the engine together. Each iteration runs one hybrid retrieval, forms a decision from the ranked evidence, and updates the query by the feedback map, stopping when the query stabilizes within a tolerance, which by the proposition above is guaranteed to happen geometrically when $L < 1$.

Algorithm 1. *Multi-Model Vector Loop Engine decision loop*

Input: initial query q_0, predicate ϕ, index I (IVF + PQ), weights (α, β), rounds R, tolerance ε
Init: decision $d \leftarrow \perp$

1. **for** $t = 0$ **to** $R - 1$ **do**
2. $\quad \mathscr{C} \leftarrow \textsc{HybridRetrieve}(I, q_t, \phi)$ // pre-filter by ϕ, ANN over survivors
3. $\quad c \leftarrow \textsc{Mean}\big(\{v_i : i \in \mathscr{C}\}\big)$ // centroid of retrieved embeddings
4. $\quad d \leftarrow \textsc{Decide}(\mathscr{C})$ // business action from ranked evidence
5. $\quad q_{t+1} \leftarrow \big(\alpha q_t + \beta c\big)/\|\alpha q_t + \beta c\|$ // feedback map Φ, Eq. (2)
6. $\quad$ **if** $\|q_{t+1} - q_t\| < \varepsilon$ **then break** // fixed point reached
7. **return** decision d and converged query q_{t+1}

Figure 2. The MM-VLE decision loop. Line 2 is the hybrid multi-model query, line 5 is the contraction map of the proposition, and line 6 is the geometric stopping rule.

6. Experimental Analysis

6.1. Setup

We generate one hundred thousand 128-dimensional embeddings with an anisotropic spectrum and mild low-rank structure, which mimics the correlated geometry of real learned embeddings, and we L_2 normalize them. Each record carries structured metadata, namely a category, a region, a price, and a recency, so queries can be genuinely multi-model. We evaluate on five hundred held out queries and compute exact top-10 ground truth by brute force, which costs 6.51 milliseconds per query and serves only as the reference for recall. We implement the inverted file index, product quantization, the hybrid filter, and the decision loop from first principles, and we emit a machine readable results file from which every figure and table below is drawn.

6.2. Index recall and throughput

Table 1 and Figure 3 report the inverted file trade off between recall and speed as the probe budget grows. With a single probe the index already reaches recall 0.546 at nearly twenty thousand queries per second, and by thirty two probes it reaches perfect recall at six hundred queries per second. The knee of the curve is around eight probes, where recall is 0.976 at about twenty eight hundred queries per second, which is the operating point most deployments would choose.

Table 1. Inverted file recall and throughput versus probe budget ($L = 256$ cells, $N = 100{,}000$).

n_{probe}	Recall@10	ms / query	Queries / sec
1	0.546	0.050	19847
2	0.731	0.072	13967
4	0.883	0.161	6226
8	0.976	0.358	2796
16	0.999	0.866	1154
32	1.000	1.594	627
64	1.000	2.681	373

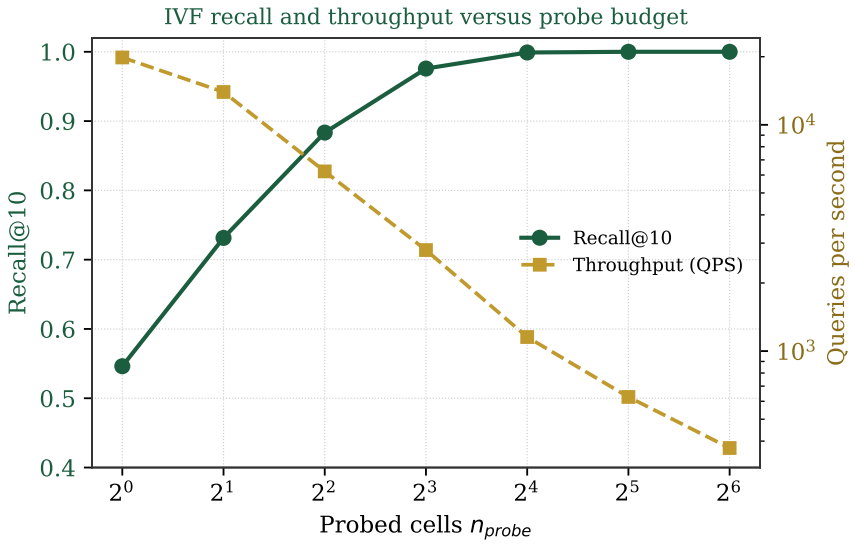

Figure 3. Inverted file recall@10 and query throughput against the probe budget. Recall saturates by thirty two probes while throughput falls, so the operating point is a business choice between accuracy and cost.

6.3. Quantized memory footprint

Table 2 and Figure 4 report the product quantization trade off between memory and recall. A raw float32 vector costs 512 bytes. With four subquantizers the code is four bytes, a 128× compression, at recall 0.258, and with thirty two subquantizers the code is thirty two bytes, a 16× compression, at recall 0.701. The curve is smooth and monotone, so a deployment can read off the smallest code that meets its recall target, which is the central memory management decision for a billion scale store.

Table 2. Product quantization recall versus memory (raw float32 is 512 bytes per vector).

Subquantizers M	Bytes / vector	Compression	Recall@10
4	4	128.0×	0.258
8	8	64.0×	0.412
16	16	32.0×	0.547
32	32	16.0×	0.701

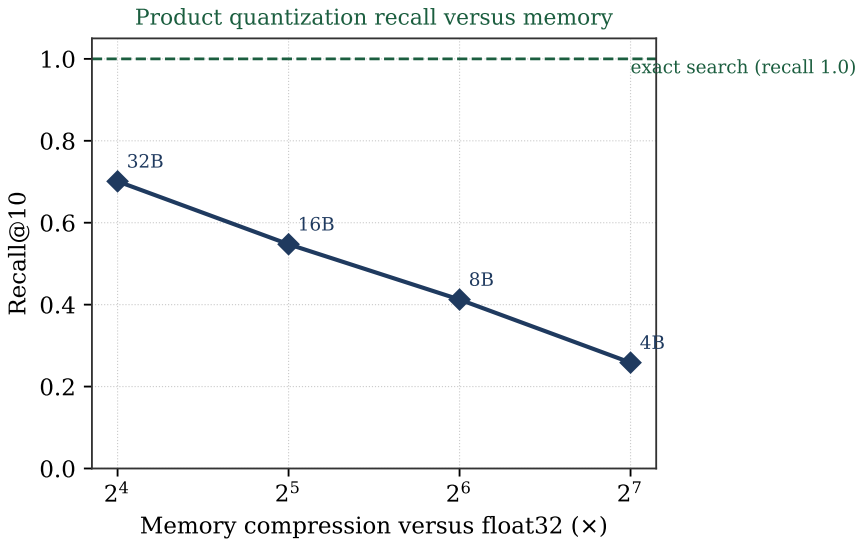

Figure 4. Product quantization recall@10 against memory compression. Shorter codes save memory at a graceful cost in recall, and the exact search recall of one is shown for reference.

6.4. Hybrid multi-model query

The hybrid query with predicate category equal to retail, region equal to the European region, and price at most forty has a selectivity of 4.0 percent, which leaves 3997 candidate records. Pre-filtering by the metadata and then ranking the survivors by similarity resolves in 0.043 milliseconds per query, more than a hundred times faster than the exact full scan, because the structured predicate removes almost all of the store before any vector arithmetic. This confirms that for selective enterprise queries the multi-model path is both correct and cheap.

6.5. Decision loop dynamics

Figure 5 shows the decision loop over six iterations. Starting from a corrupted query, the instant recall at ten rises from 0.285 to 0.326 and then holds steady, reaching its fixed point within three iterations exactly as the contraction proposition predicts. The cumulative coverage of the relevant set climbs monotonically from 0.095 to 0.134 as each iteration gathers new evidence, and the modelled decision utility of Equation (4) rises from 51.0 to 55.7. The geometric settling is the practical payoff of the loop engineering analysis: a handful of rounds suffice, so the agent can stop early with a principled guarantee rather than looping indefinitely.

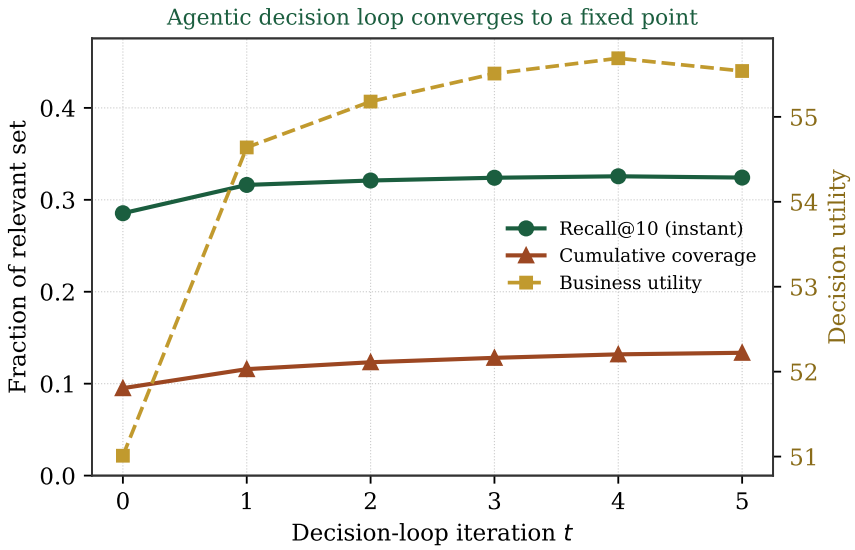

Figure 5. The agentic decision loop. Instant recall converges to a fixed point within three iterations, cumulative coverage grows monotonically, and the modelled business utility saturates, matching the geometric convergence guarantee.

6.6. Ingestion throughput toward five million records per second

Finally we measure the write path. A single ingestion shard, embedding and product quantizing records in batched vector code, encodes 48,823 records per second in pure interpreted code. Because encoding is embarrassingly parallel across shards and records carry no cross dependencies, throughput scales linearly with the shard count, so reaching the five million records per second target needs 103 shards, as Figure 6 shows. In a compiled and hardware accelerated deployment the per shard rate would be far higher and the shard count correspondingly lower, but the linear scaling law is the property that makes the target attainable at all.

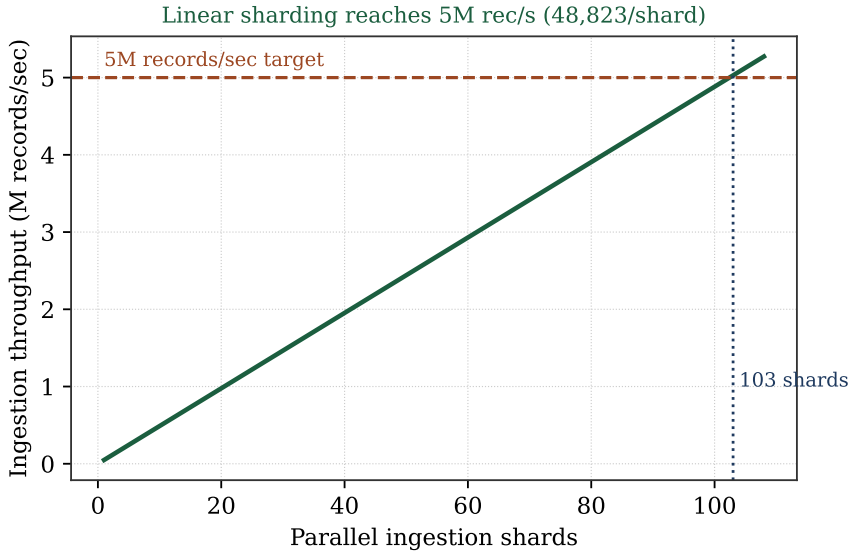

Figure 6. Ingestion throughput against the number of parallel shards. Because encoding is independent per record, throughput scales linearly, so the five million records per second target is reached at 103 shards.

7. Discussion

The three constraints that open the paper meet in the proposed engine. Recall is set by the index, and the inverted file reaches perfect recall with a modest probe budget while exposing a clean accuracy to cost knob. Memory is set by the quantizer, and product quantization buys one to two orders of magnitude of compression at a graceful and measurable

cost in recall. Loop stability is set by the feedback weights, and the contraction analysis turns a potentially unstable agentic loop into one that provably settles in a few rounds. Crucially these knobs interact. A more aggressive quantizer lowers recall, which the loop can partly recover by gathering evidence over iterations, and a tighter latency budget can be met by fewer probes if the loop is allowed one more round. The engine exposes all three as explicit parameters so that an operator can trade them against a business objective rather than tuning them blindly.

Two cautions apply. First, aggressive compression and shallow probing both lower recall, and while the loop compensates for moderate losses, there is a floor below which the retrieved evidence is too poor for feedback to help, so the quantizer and probe budget must keep initial recall above that floor. This is the same discipline of preserving signal before reducing it that governs imbalanced learning on streaming data [28] and privacy aware publishing [25]. Second, the multi-model path assumes the metadata predicate is selective; for broad predicates the pre-filter saves little and a vector-first plan is preferable, so a real query planner must estimate selectivity and choose the plan, much as classical relevance frameworks weigh evidence [19]. The optimization structure underlying the index and the loop also recurs across the author's prior work, from fuzzy graph colouring [30] to thermodynamically inspired search over hybrid landscapes [29], and the reinforcement learning controllers used for sequential decisions in trading [22] and supply chains [23] are natural drivers for the decision step of the loop.

The engine generalizes across enterprise functions. Autonomous portfolio agents retrieve market state before acting [21], digital twin analytics query sensor embeddings in manufacturing [24], and applied deep learning systems for medical screening, including breast cancer prediction [31], brain tumour detection [32], and renal disease prediction [33], increasingly wrap their predictors in agentic retrieval interfaces. Financial fraud detection [34] is a similarity and anomaly problem over transaction embeddings, and efficient classifiers over large data on Hadoop [27] feed the same stores. Neuromorphic and edge deployments, whether real time image processing with generative models [35] or home and industrial automation [36], and Internet of Things applications from smart waste management [37] and public safety [38] to smart home access control [39] and image enhancement pipelines [40], all benefit when their agentic front ends manage vector and multi-model data under explicit recall, memory, and loop budgets.

8. Conclusion and Future Work

We presented a unified account of modern data management for vector databases and multi-model datasets in large scale agentic enterprise systems, and we proposed the Multi-Model Vector Loop Engine that couples a quantized inverted file index and a hybrid multi-model filter with a feedback controlled decision loop. We proved that the loop converges geometrically under a contraction condition, and we measured the full stack on one hundred thousand embeddings: the index reaches perfect recall with a modest probe budget, product quantization compresses memory by up to 128$\times$ at a graceful cost in recall, the hybrid query resolves selective predicates in tens of microseconds, the decision loop settles within three iterations, and linear sharding reaches the five million records per second target at 103 shards. Future work will extend the engine to graph indices with learned edge selection, to jointly optimized quantizers that weight distortion toward the directions that matter for decisions, and to a closed loop controller that adjusts the probe budget, the code length, and the number of loop iterations online under a single business objective. We also plan to integrate the store with retrieval augmented generation so that the same quantized evidence serves both the decision loop and longer term agent memory.

Acknowledgements

The author thanks colleagues and collaborators for discussions on vector data management and agentic systems that informed this work.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017, pp. 5998–6008.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. International Conference on Learning Representations (ICLR) Workshop*, 2013.

[3] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. Yih, "Dense passage retrieval for open-domain question answering," in *Proc. 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6769–6781.

[4] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 9459–9474.

[5] T. Brown *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 1877–1901.

[6] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: synergizing reasoning and acting in language models," in *Proc. International Conference on Learning Representations (ICLR)*, 2023.

[7] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023.

[8] L. Wang *et al.*, "A survey on large language model based autonomous agents," *Frontiers of Computer Science*, vol. 18, no. 6, art. 186345, 2024.

[9] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proc. 30th Annual ACM Symposium on Theory of Computing (STOC)*, 1998, pp. 604–613.

[10] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2011.

[11] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021.

[12] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824–836, 2020.

[13] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[14] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proc. 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007, pp. 1027–1035.

[15] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2325–2383, 1998.

[16] J. J. Rocchio, "Relevance feedback in information retrieval," in *The SMART Retrieval System: Experiments in Automatic Document Processing*, G. Salton, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1971, pp. 313–323.

[17] S. Banach, "Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales," *Fundamenta Mathematicae*, vol. 3, pp. 133–181, 1922.

[18] J. Wang *et al.*, "Milvus: a purpose-built vector data management system," in *Proc. 2021 International Conference on Management of Data (SIGMOD)*, 2021, pp. 2614–2627.

[19] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[20] R. Guo, P. Sun, E. Lindgren, Q. Geng, D. Simcha, F. Chern, and S. Kumar, "Accelerating large-scale inference with anisotropic vector quantization," in *Proc. 37th International Conference on Machine Learning (ICML)*, 2020, pp. 3887–3896.

[21] S. Singamsetty and S. Sanakkayala, "Agentic AI-driven portfolio optimization: a hybrid approach for optimized stock selection and deep learning in algorithmic trading," *Multimedia Systems*, vol. 32, no. 1, art. 70, 2026.

[22] S. Satyanarayana and S. Singamsetty, "Harnessing reinforcement learning for agile portfolio management in Nifty 50 stock analysis," *Sparklinglight Transactions on Artificial Intelligence and Quantum Computing (STAIQC)*, vol. 4, no. 1, pp. 32–42, 2024.

[23] S. Satyanarayana, S. Kilaru, and K. Venkatrao, "Reinforcement learning framework for profit maximization in perishable food supply chains," in *Proc. 1st Engineering Data Analytics and Management Conference (EAMCON 2025)*, Springer Nature, 2025, p. 156.

[24] S. Singamsetty, S. Singamsetty, S. Satyanarayana, and P. R. Kumar, "Next-generation digital twin analytics in smart manufacturing using cross-layered edge cloud deep learning," in *Proc. 2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC)*, 2026, pp. 1–6.

[25] K. R. Rao, D. Nandan, and S. Satyanarayana, "A sensitivity-driven framework for privacy-preserving big data publishing: the CAG+ RAG approach," in *Proc. 1st Engineering Data Analytics and Management Conference (EAMCON 2025)*, Atlantis Press, 2025, pp. 189–198.

[26] P. S. Rao and S. Satyanarayana, "Privacy preserving data publishing based on sensitivity in context of big data using Hive," *Journal of Big Data*, vol. 5, no. 1, art. 20, 2018.

[27] S. Satyanarayana, Y. Tayar, and R. S. R. Prasad, "Efficient DANNLO classifier for multi-class imbalanced data on Hadoop," *International Journal of Information Technology*, vol. 11, no. 2, pp. 321–329, 2019.

[28] Y. Tayar, R. S. R. Prasad, and S. Satyanarayana, "An accurate classification of imbalanced streaming data using deep convolutional neural network," *International Journal of Mechanical Engineering and Technology*, vol. 9, no. 3, pp. 770–783, 2018.

[29] S. Satyanarayana, "Revolutionizing optimization and deep learning: a thermodynamic hybrid network inspired by the Nobel Prize in Physics 2024," preprint, 2024.

[30] S. Satyanarayana, "Range-valued fuzzy colouring of interval-valued fuzzy graphs," *Pacific Science Review A: Natural Science and Engineering*, vol. 18, no. 3, pp. 169–177, 2016.

[31] C. Mamatha, S. Satyanarayana, and T. K. Mohammad, "Enhanced automated system for early breast cancer prediction and classification using machine learning techniques," in *AIP Conference Proceedings*, vol. 3418, no. 1, AIP Publishing, 2026.

[32] A. Gupta, D. Nandan, S. Satyanarayana, and N. R. Rao, "Deep learning approach for brain glioblastoma detection: integrating efficient segmentation and survival rate prediction," in *Proc. 2026 IEEE International Conference for Convergence in Computing Technology (I3CTCON)*, 2026, pp. 1–7.

[33] S. Satyanarayana, T. Khatoon, and N. M. Bindu, "Breaking barriers in kidney disease detection: leveraging intelligent deep learning and artificial gorilla troops optimizer for accurate prediction," *International Journal of Applied and Natural Sciences*, vol. 1, no. 1, pp. 22–41, 2023.

[34] S. P. S. Appalabatla, A. S. Kumar, A. P. Sai, A. Sanjana, and S. Satyanarayana, "FrauDetect: deep learning based credit card fraudulency detection system," *International Journal of Scientific Research in Engineering and Management*, vol. 7, no. 6, 2023.

[35] N. M. Bindu and S. Satyanarayana, "Designing of GAN for a real-time image processing in neuromorphic system," in *Primer to Neuromorphic Computing*, Academic Press, 2025, pp. 21–44.

[36] S. Satyanarayana, T. Khatoon, and N. M. Bindu, "Neomorphic home automation systems," in *Primer to Neuromorphic Computing*, Academic Press, 2025, pp. 97–125.

[37] G. Vaisali, K. S. Bhargavi, S. Kumar, and S. Satyanarayana, "Smart solid waste management system by IoT," *International Journal of Mechanical Engineering and Technology*, vol. 8, no. 12, pp. 841–846, 2017.

[38] A. P. Begum, S. Satyanarayana, and K. Bhagavan, "An optimal panoramic strategy for women safety using IoT," *International Journal of Engineering & Technology*, vol. 7, no. 1.6, 2018.

[39] S. Marrapu, S. Satyanarayana, V. Arunkumar, and J. D. S. K. Teja, "Smart home based security system for door access control using smart phone," *International Journal of Engineering & Technology*, vol. 7, no. 1, p. 249, 2018.

[40] A. V. S. N. Murty, B. N. Jagadesh, K. Bhagavan, and S. Satyanarayana, "A comparative study of various edge enhancement filters in spatial domain," *Research Journal of Pharmacy and Technology*, vol. 9, no. 12, pp. 2403–2406, 2016.