

Executive Summary

Unifying swarm agents and spatial storage over a problem-knowledge-solution space

SSCCS Initiative

Overview

One coordinate space, one protocol, multiple directions, no orchestration.

As a heterogeneous system, neXus is a state fabric unifying swarm agents and spatial storage through contract-governed protocol over an immutable problem-knowledge-solution space. ¹ Any backend that can store an append-only record, a stateful record, and a read-only record becomes one: no graph database, no specialized index layer, and no hosted platform are required. Its core, nex, behaves like a lightweight universal hub, attaching an embedded filesystem, an enterprise database, or an object store as a fully functional coordination space. The same binary runs on Wasm, edge nodes, portable devices, blockchain runtimes, and bare-metal containers.

What makes neXus different

Storage is a tactical choice rather than an architectural constraint. Producers run nex on their own infrastructure, under their own terms, while verified knowledge compounds stigmatically in the shared store; each producer inherits the contributions of others and contributes back.

Backbone for AI agent architectures

From the perspective of AI agent architectures, neXus is backbone infrastructure that provides three fabrics in one.

- A semantic fabric records the research process itself: every Fact carries the Intent that proposed it and the evidence that validated it, forming a replayable audit trail, while retrieval and gap detection run on accumulated facts at zero marginal inference cost.
- A governance fabric, the contract layer, gates what may be written, evaluates constraints, and keeps an evidence chain, so agent behavior is bounded and auditable.
- A storage router fabric lets agents, editors, and verification engines coordinate through one shared interface regardless of which backend holds the state.

The industrial utility follows directly: accountable collaboration, since every conclusion is traceable to its evidence; affordable operation, since routine agent coordination costs no LLM inference; and sovereignty, since organizations run the fabric on their own infrastructure without lock-in.

¹A blackboard is a shared coordination space: agents write immutable Facts, claim Intents, and read Hints through one interface, following the blackboard architectural pattern.

The problem: knowledge systems tax every query twice

Every knowledge system today carries two recurring costs that have become invisible. Record identity is hash-based: a UUID or a SHA-256 digest names a record without saying anything about what the record is. Retrieval leans on LLM inference: text is embedded into vectors, scanned through an index, and a model is invoked for every question. The industry response has been larger indexes and more model calls, compensating for a structural inefficiency rather than eliminating it.

A conventional knowledge graph stores static entity-relationship triplets. It captures what is known, but not how it was established. A conclusion cannot be audited back to the hypothesis that generated it or the experiment that confirmed it. The graph is a static snapshot that cannot be replayed, and every query pays for the snapshot without recovering the reasoning.

A structural identity with a replayable trace

neXus replaces hash-based identity with the Tagma coordinate space ². A CoordId is a structural identifier: a path of six coordinates by default, each drawn from 11,172 valid values, giving a 2×10^{24} address space, and CoordId<20> reaches SHA-256 scale. The six axes are application-defined: time_hi, time_lo, entity kind (Fact, Intent, or Hint), origin, creator, and serial. A hash-based identifier carries no information about what a record is, while a CoordId encodes what kind of record it is, who created it, and when. Identity is a coordinate, so identity is queryable: a filter on origin or creator is an axis projection over a coordinate tree.

The data model is FIH. Fact is an immutable, validated observation and the output of a concluded Intent. Intent is a proposed exploration with a strict lifecycle: submit, claim, heartbeat, conclude. Hint is an injected, read-only constraint bounding admissible actions. **A Fact is a unit of verified progress: the Intent that proposed it, the Hint that bounded it, and the evidence that validated it travel with the Fact.** The blackboard is a queryable, replayable computational trace, and each conclusion can be audited to the experiment that confirmed it.

FihStorage realizes this model as a structural index over the Tagma coordinate space. After the restructure, record identity

²synTagma is the coordinate-space primitive behind neXus identity. Repository: github.com/ssccsorg/syntagma, documentation: docs.ssccs.org/projects/syntagma/

is a full-injective $\text{CoordId} < 20 >$ that carries all 256 bits of identity, and a content-hash guard rejects id collisions at commit time. Any backend that stores an append-only record, a stateful record, and a read-only record is sufficient, from SQLite to object storage. The same binary runs on Wasm, edge nodes, portable devices, blockchain runtimes, and bare-metal containers.

The platform around this substrate is organized in five layers: a hybrid knowledge graph engine, an artifact ingestion pipeline, an agentic research loop coordinated by stigmergy, an on-policy learning loop, and contract governance. Every layer reads and writes the same FIH store.

Measured performance

All measurements come from the nexus benchmark suite, release profile, median of 10 samples, Apple M1 ³.

Query and identity

Structural prefix queries. A full walk of a 50,000-entry tree costs 540 ms. Five hundred prefix queries over the same tree cost 1.12 ms in total, 2.24 μs per query. The coordinate tree turns an $O(N)$ scan into an $O(\text{subtree})$ navigation, a 241,000x gap per query, because a prefix names a branch of the tree. A scan-based paradigm pays the full cost on every query; the coordinate tree pays only the subtree. In practice, a knowledge base that grows from 1,000 to 1,000,000 records does not increase query time; only the prefix depth changes.

Identity lookup. The same six-axis identity space resolves 10,000 lookups through a hash-based index, the existing identity paradigm, in 120.9 μs , 12.1 ns per lookup. The Tagma dense packed array resolves them in 60.8 μs , 6.08 ns per lookup, 2.0x faster than the hash index. The Tagma tree form resolves them in 533 μs , 53.3 ns per lookup; it is slower on single-record lookups because it carries prefix navigation, the $O(\text{subtree})$ path measured above. Both Tagma costs scale with coordinate depth, independent of data volume: 10^4 and 10^{24} records cost the same number of dereferences.

Filtering

Filter selectivity. Over 50,000 facts, a single-axis creator filter matching 2,500 records costs 745 μs . An origin and creator AND query matching 500 records costs 296 μs . A three-axis query, origin, creator, and time range, resolves in 83.8 μs and materializes no records. Each added dimension shrinks the candidate set before any record is loaded, and the most selective query is the cheapest. The previous model scanned the full store on every filter, with a measured ceiling of 27.8 ms at 10,000 facts ⁴; the current filters run at 50,000 facts and stay below that ceiling.

Axis hints. A query with axis hints resolves in 42 μs versus 127 μs without, on 10,000 facts. The multi-dimensional

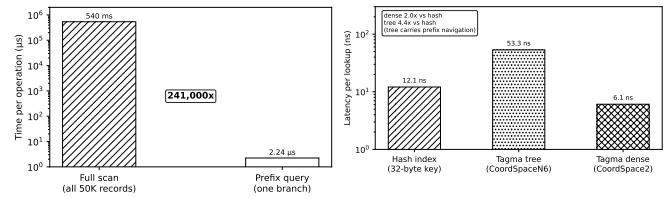


Figure 1: Query cost in a record store over 50K entries: touching every record versus navigating the named branch. Any scan-based paradigm pays the full cost on every query; the coordinate tree pays only the subtree.

Figure 2: Identity lookup in the 6-axis space (10K lookups): hash-based index versus Tagma layouts. The dense layout is 2.0x faster than the hash index; the tree layout is 4.4x slower on single-record lookups because it carries prefix navigation.

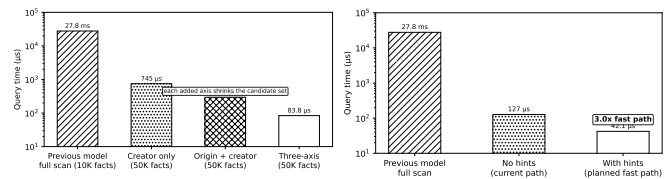


Figure 3: Filter queries: previous full-scan model (10K facts) versus current selective queries (50K facts). The previous model scanned the entire store regardless of selectivity, so the gap is conservative.

Figure 4: Filter path on 10K facts: previous full-scan model versus current predicate path and the planned prefix fast path (axis hints). Same data size throughout.

Knowledge base and write

Knowledge-base scenario. Ten thousand documents across ten projects and twenty authors: a project and author AND query resolves in 70.2 μs , a project-only query in 260 μs , and a project, author, and time-range query in 70.4 μs . Reverse lookup of intents referencing a fact resolves through the inverse index in about 1.9 μs per call, superseding the earlier full-scan path at 60.4 ms per call.

Batch write. Ten thousand facts submit and flush in 51.9 ms, 5.19 μs per fact, with a single flush boundary rather than per-record IO. The pre-Tagma implementation took 1,776 ms for the same volume, a 34x gap.

³Benchmark code: github.com/ssccsorg/nexus/blob/main/benches/bench.rs

⁴Pre-Tagma baseline from the full-scan implementation, measured on the same Apple M1. Devlog 160: github.com/ssccsorg/nexus/blob/main/docs/2026-07-29-160-syntagma-integration.md

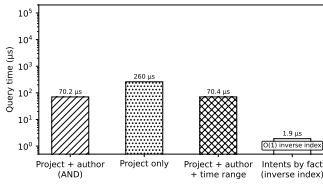


Figure 5: Knowledge-base scenario (10K documents, 10 projects, 20 authors): selective queries are sub-millisecond; reverse lookup resolves through the inverse index.

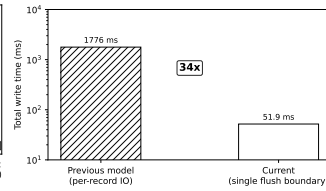


Figure 6: Batch write of 10K facts: pre-Tagma full-scan model versus current single-flush batch (34x).

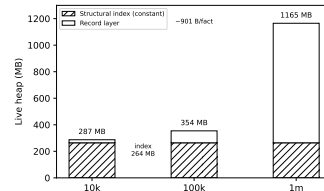


Figure 7: Flat indexing cost: the structural filter index holds a constant footprint as the record count grows. The index portion (hatched) is identical at 10k, 100k, and 1m facts over a fixed axis-combo space; only the record layer grows (~901 bytes per fact).

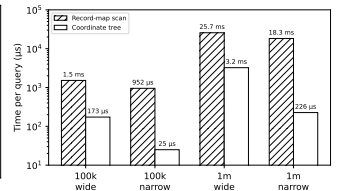


Figure 8: Multi-dimensional search at scale (origin + creator + time range): the coordinate-tree path (iter_prefix) versus the record-map scan. The gap widens as records accumulate: 9x to 38x at 100k, 8x to 81x at 1m facts.

Restructure measurements

The restructure, measured on the same Apple M1:

- Identity conflict detection at commit time dropped to sub-microsecond per call.
- Per-fact footprint dropped from ~3.4 MB to ~40 KB average.
- State reads serve from in-memory indexes; a 10,000-event stress scenario runs in ~9.0 seconds, down from 17.7.

Flat indexing cost

The deepest property of the coordinate model is now a measured one: indexing cost does not grow with the record count. Over a fixed axis-combo space, the structural filter index holds an identical footprint at 100,000 and 1,000,000 facts, about 264 MB; the record layer adds about 901 bytes per fact. Total live heap measures 287 MB at 10k facts, 354 MB at 100k, and 1,165 MB at 1m, with the index portion the same across the two larger scales.

The query side shows the same flatness. On a time-bounded three-axis filter (origin, creator, time range), the coordinate tree resolves the query 8x to 38x faster than the record-map scan at 100k facts, and 8x to 81x faster at 1m facts. The gap widens as records accumulate because the scan pays the full store on every query while the tree pays only the subtree. The same behavior appears on a real document set with the full FIH lifecycle (add, intent, claim, conclude): the tree advantage grows from 4.5x at phase 1 to 5.3x at phase 3 as the knowledge network accumulates.

The flatness boundary is axis cardinality. Each distinct axis value costs about 349 KB, one dense leaf plus one branch. The index is constant in record count and linear in distinct axis values, so application axes must stay bounded in practice: high-cardinality origins such as unique conclusion ids inflate the tree linearly ⁵.

Resource profile

The coordinate tree allocates one leaf node per written prefix, and each leaf covers 11,172 slots regardless of occupancy, a fixed cost that trades memory for predictable performance, unlike hash tables that grow unpredictably. Memory is bounded by axis cardinality, not record count; the restructure cut the per-fact footprint from ~3.4 MB to ~40 KB average. The measurements put the constant at about 264 MB over a fixed axis-combo space from 100k to 1m facts, about 901 bytes per fact for the record layer. Dense packing removes allocation entirely at a fixed memory floor and holds a flat 6 ns lookup. Graph traversal, gap detection, and reporting operate on accumulated facts at zero marginal inference cost; AI use is confined to knowledge-branch generation.

Verification by exhaustive enumeration

The coordinate space is exhaustively enumerable: 65,536 possible 16-bit values, of which exactly 11,172 are structurally valid under a closed-form composition formula. A verification harness validates every possible value against the decoder specification in milliseconds on any commodity system, and every FIH layout on these axes inherits the guarantee. Identity collisions are excluded by construction, where hash-based systems can only reduce their probability; no hash-based index can be exhaustively verified across its input space.

Boundaries

neXus does not replace cryptographic primitives. A content hash remains for evidence chains and content integrity; the coordinate axes provide ordering, not preimage resistance. Record identity is semantic: CoordId is derived from entity kind, origin, creator, and content, so an id describes what it names. The two compose: a SHA-256 digest encoded as coordinates stays verifiable and becomes queryable by axis. The

⁵Multi-dimensional search benchmark, issue 179: github.com/ssccsorg/nexus/issues/179; devlog 179 in the nexus repository docs folder.

benchmark suite reports the current predicate path and the structural fast path side by side, so the reader sees the measured ceiling and the measured headroom.

Application domains

Research documentation. The research process is recorded as FIH: hypotheses as Intents, validated results as Facts, scope as Hints. Conclusions are auditable to the experiments that confirmed them.

Agent coordination. Peers interact only through the shared space, leaving and reading traces. There is no privileged orchestrator; roles are defined by what a peer reads and writes.

Serverless and edge. Any append-only backend is sufficient, so the same binary deploys on Wasm, edge nodes, portable devices, blockchain runtimes, and bare-metal containers.

Knowledge accumulation. Each producer solving a problem contributes verified knowledge back to the shared store, and the store compounds recursively across agents, experiments, projects, and ecosystems.

Market position

The agentic AI market is moving from experimentation to production scale, and analyst projections put it on a path from about USD 19 billion in 2026 to more than USD 200 billion by the early 2030s. The dominant cloud providers are converging on the same conclusion: production agents need a runtime layer with durable execution, memory, and governance. Google's open-source Agent Executor, Microsoft's execution containers, and LangGraph 1.0 all target this gap.

neXus addresses the layer beneath these runtimes. Where current memory solutions depend on knowledge graphs and vector indexes that grow costlier with data volume, neXus uses coordinate arithmetic for index-free lookup, keeps every conclusion in a replayable trace governed by a contract layer, and runs on any storage backend. The result is what the market identifies as the missing piece: auditable agent memory that stays fast as data grows, without vendor lock-in.

Conclusion

neXus is a substrate: a shared coordinate space where agents record, retrieve, and verify knowledge without indexing overhead, vendor lock-in, or inference cost on every query. The hub pattern makes it deployable anywhere, the FIH model makes it auditable, and the coordinate space makes it fast.

Latest developments

- **Flat indexing cost measured:** the structural filter index holds a constant footprint as records grow; multi-axis queries via the coordinate tree run 8-81x faster than the record-map scan at 1m facts, with the gap widening on accumulation.

- **Restructured record layer:** record identity is now a full-injective 256-bit coordinate; structural queries run over a compact coordinate index; content-hash conflict detection hardens the record layer.
- **Semantic identity:** record ids are derived from domain meaning (entity kind, origin, creator, content) instead of opaque hashes, so identity is self-describing.
- **Consolidated storage engine:** all storage paths now run on one shared engine; the last proprietary backend dependency was removed.
- **Runtime separation:** the coordination server is a standalone binary with a formalized protocol, and the daemon is a pure process supervisor.

Roadmap

The roadmap follows a single principle: make the substrate robust first, then expand the surfaces that sit on it.

- Wire the benchmark-validated multi-dimensional search path into the production read path.
- Formalization of the coordinate system: how raw input becomes a Fact, how queries project back to results, and temporal semantics.
- A native query language over the coordinate model, replacing graph-style query syntax.
- Contract layer extensions: declarative specifications, composable governance rules, and durable evidence trails.
- Data-at-rest protection: encryption, tiered storage, and hardware acceleration.
- A research loop: AI-assisted hypothesis generation grounded in accumulated facts, with experimental feedback.
- Attack-scenario verification of the storage interface.
- Agent-layer stigmergy: pressure fields and trace decay.
- Editor integration through a standard agent protocol surface.

Status

neXus is an open-source project (Apache 2.0) under the SS-CCS Initiative. The Rust workspace is available on GitHub ([ssccsorg/nexus](https://github.com/ssccsorg/nexus)) with a unified benchmark suite and an end-to-end verification harness. All storage paths run on one shared engine. The governance layer (write admission, constraint evaluation, evidence chains) is implemented, and the coordination server is a standalone binary with a formalized protocol, supervised by a dedicated daemon. Gap and contradiction detectors record findings as idempotent Facts. For inquiries, demos, or partnership discussions: contact@ssccs.org