Architecting Conversational Data Systems for Stateless LLM APIs

The Hydration Proxy Pattern

Joseph Axisa
josephaxisa@google.com
Google Cloud
California, USA

Abstract

As enterprise platforms transition to conversational reasoning interfaces, the stateless nature of high-performance LLM APIs creates a fundamental architectural gap. While statelessness enables massive horizontal scalability for AI providers, it forces the consuming application to manage the entire burden of conversational state and semantic memory. This work identifies the *Hydration Proxy Pattern*, a three-tier architecture synthesized from the operational requirements of production-grade analytical platforms. By decoupling session persistence from the reasoning engine, the framework ensures platform sovereignty over conversational data while enabling secure, multi-stage semantic grounding. We further propose the *Context Stabilization Mandate* to resolve the tension between sovereign state management and infrastructure-level KV caching.

1 The Challenge: The Stateless Paradox

The integration of LLMs into professional data environments is complicated by the *Stateless Paradox*. Analytical exploration is an inherently iterative process, relying on the continuity of previous queries, visualization states, and complex reasoning steps. However, to maximize horizontal scale and multi-tenant isolation, AI providers architect reasoning engines as stateless functions [8]. This places the burden of "memory" entirely on the application tier, requiring a *Full-History Round-Trip* model where the complete conversation history and metadata must be re-transmitted with every query.

In practice, this model creates three primary areas of operational friction:

- **The Governance Gap:** When organizations rely on centrally-managed session models (where the AI vendor stores history), they lose sovereign control over sensitive conversational logs. This restricts the organization's ability to enforce granular, platform-level security policies or adhere to regional data residency laws.
- **The Caching Conflict:** Modern inference infrastructure utilizes KV (prefix) caching to reduce latency. Naive "hydration" of requests often includes rotating security tokens or dynamic metadata at the start of a prompt, which effectively "busts" the provider-side cache and increases token costs.
- **Semantic Fragmentation:** Precise analytical reasoning requires high-fidelity grounding in data schemas. Passing raw metadata from a client-side wrapper is both insecure and inefficient, particularly for large-scale enterprise schemas with thousands of fields.

2 The Architectural Framework: The Hydration Proxy

To resolve these frictions, we present a three-tier architecture synthesized from the requirements of production-grade analytical systems. This framework establishes a clear separation between the UI, the security gateway, and the persistence layer.

- **Orchestration Layer (Frontend):** Serves as the primary entry point, maintaining the immediate UI state and reassembling the chronological message history required for each request.
- **Hydration Proxy (Secure Gateway):** Acts as a specialized backend bridge that "hydrates" raw client requests with authoritative platform context. This tier is responsible for the dynamic injection of security credentials and governing metadata (e.g., schemas) that cannot reside on the client tier.
- **Hybrid Persistence Layer:** Employs a dual-storage strategy that decouples session management from conversational content, ensuring the system can scale to handle dense, high-volume interaction logs.

3 State Management: Decoupled Persistence

Conversational history in analytical systems is information-dense, containing large JSON payloads with query results and visualization metadata. We identify a decoupled strategy that ensures high-throughput access without compromising database performance:

- **Relational Ledger (Metadata Tier):** A relational database maintains the structured "skeleton" of the session. This ledger tracks session ownership, the chronological sequence of messages, and unique pointers to the full conversation data. This allows the system to perform fast lookups and audit session activity without loading massive text blobs.
- **Object Storage Tier (Content Tier):** High-volume, unstructured payloads are offloaded to an encrypted object storage service. This prevents relational database bloat and ensures that the system can scale as users generate thousands of data-dense conversational turns.
- **Full-History Reassembly:** Upon every user interaction, the proxy uses the relational ledger to locate the relevant content in object storage. It reassembles the sanitized history, ensuring the reasoning engine has the persistent context required to resolve follow-up questions.

Table 1: Comparison of Architectural Archetypes

Archetype	State Ownership	Model Agnostic	Grounding Strategy
Naive Wrapper	Shared between client and shared server state.	Yes	Static schema injection; prone to exhaustion.
Centrally-Managed	Offloaded to the AI vendor's proprietary cloud.	No	Naive RAG managed by provider infrastructure.
Ecosystem-Coupled	Tethered to a specific data platform or ecosystem.	Limited	Tightly-bound to vendor-specific metadata.
Self-Hosted	Resident on private organizational infrastructure.	Yes	Requires external state and grounding management.
Hydration Proxy Pattern	**Sovereign ownership within organization storage.**	**Yes**	**Cache-Aware Multi-Stage Expansion.**

4 Comparative Systems Analysis

We evaluate the architectural trade-offs of this pattern against prevailing industry archetypes:

Naive Stateless Wrappers: Common in initial prototypes, this involves utilizing standard stateless endpoints, such as the OpenAI Chat Completions API [7]. While application-tier memory libraries attempt to manage context recall via client-side summarization loops, they are prone to severe context bloat and introduce security vulnerabilities by exposing raw platform metadata and governance boundaries directly to the client tier [4].

Centrally-Managed Session Models: Providers like the OpenAI Assistants API [1] store history and manage context windowing on proprietary infrastructure. While this minimizes implementation complexity, it presents challenges for enterprise data sovereignty and multi-cloud portability.

Ecosystem-Integrated Agent Models: Analytical platforms like Databricks Genie [2], Snowflake Cortex [3], or Gemini in BigQuery [9] utilize a tightly-coupled integration model. In these systems, the AI reasoning engine is an intrinsic component of the data warehouse ecosystem. While this enables performance, it creates significant vendor lock-in and restricts multi-cloud flexibility.

5 The Context Stabilization Mandate

A substantive design challenge of the Hydration Proxy is the tension with provider-side KV caching. Modern inference infrastructure [6] utilizes KV caching to store attention states for static prefixes. Naive hydration—injecting rotating security tokens or dynamic metadata at the prompt origin—busts this cache, increasing latency and cost.

We identify the *Linear Context Strategy* as a core requirement for production proxies. The Proxy must maintain a stable prompt structure, ensuring that repeated context remains identical for every request:

(1) **Static Prefix:** High-token platform schemas and system instructions are fixed at the prompt origin to maximize cache reuse across multiple turns.
(2) **Semi-Stable History:** Chronological logs are reassembled without mutation to preserve byte-identity across turns, ensuring the cache is only appended, never invalidated.
(3) **Volatile Suffix:** Ephemeral data, including rotating OAuth tokens, user location metadata, and the current query, are relegated to the suffix.

6 System Merits and Design Synthesis

The architectural merits of the *Hydration Proxy Pattern* are synthesized below:

- **Sovereign Persistence:** By offloading message payloads to an organization-controlled storage tier, conversational state remains platform-resident. This ensures users can resume complex reasoning context across multiple devices without involving the AI vendor's storage.
- **Governed Semantic Grounding:** The proxy tier prevents sensitive platform schemas from being exposed to the client. This allows for the dynamic injection of authoritative field-level security constraints just-in-time for the reasoning task, ensuring the model is grounded only in data the user is authorized to access.
- **Responsive Interleaving:** Through real-time stream parsing at the proxy tier, the system can separate the model's internal reasoning steps (thoughts) [5] from user-facing answers. This enables the UI to drive immediate progress states, mitigating the inherent latency of complex analytical tasks.
- **Strategic Context Scaling:** The Proxy acts as a "context governor," implementing *Recursive Summarization* and *Removable Reasoning*. By periodically compacting history or stripping high-token internal reasoning steps [5], the Proxy maintains operational reliability. While compaction causes a transient cache miss, it resets the prompt's token baseline, ensuring the session remains viable as conversational density increases.

7 Conclusion

By identifying the *Hydration Proxy Pattern* and the *Context Stabilization Mandate*, we can architect Conversational Data Systems that are scalable, secure, and provider-agnostic. This architecture allows platforms to maintain sovereign control over the "Memory" and "Governance" of the system while utilizing stateless reasoning APIs for "Intelligence." It resolves the tension between data privacy and the infrastructure-level economic necessity of prefix caching.

References

[1] OpenAI. 2024. *Assistants API Overview.* Retrieved from https://platform.openai.com/docs/assistants/overview
[2] Databricks. 2024. *Databricks Genie: AI-powered data assistant.* Retrieved from https://www.databricks.com/blog/introducing-databricks-genie
[3] Snowflake. 2024. *Snowflake Cortex Analyst.* Retrieved from https://www.snowflake.com/en/data-cloud/cortex/
[4] K. Greshake, et al. 2023. *Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Injected Prompts.* arXiv preprint arXiv:2302.12173.
[5] S. Yao, et al. 2023. *ReAct: Synergizing Reasoning and Acting in Language Models.* ICLR 2023.
[6] Meta AI. 2024. *Introducing Llama 3.* Retrieved from https://ai.meta.com/blog/meta-llama-3/
[7] OpenAI. 2024. *Chat Completions API Guide.* "The API is stateless, meaning the server does not keep any record of previous requests." Retrieved from https://platform.openai.com/docs/guides/text-generation

[8] Google DeepMind. 2024. *Gemini 1.5 Pro: Scalable Stateless Inference*. Retrieved from https://deepmind.google/technologies/gemini/

[9] Google Cloud. 2024. *Gemini in BigQuery*. Retrieved from https://cloud.google.com/bigquery/docs/gemini-introduction