OpenCQL: An Exploration of Structured Context Engineering

Investigating Deterministic Control Planes for Probabilistic Models

Author: Saurabh Gupta
Status: Request for Comment (RFC)
Date: February 2026

1. Abstract

As Large Language Models (LLM) integrate deeper into enterprise workflows, the management of "Context"—the prompt data retrieved from vector databases—remains an open challenge. Current "Linear RAG" approaches often suffer from context dilution and "Context Collapse."

This paper **investigates the potential** of using Domain-Specific Languages (DSLs) to structure this process. Specifically, we explore **OpenCQL (Context Query Language)**, a prototype grammar that adapts SQL primitives (JOIN, GROUP BY) to the semantic domain. Our primary inquiry is whether enforcing a "Semantic MapReduce" topology via strict query syntax can improve the auditability and separation of concerns in recursive reasoning tasks.

2. The Hypothesis

We posit that the "Hallucination" problem in RAG is often a "Topology" problem. When we ask an LLM to answer a question based on 50 mixed documents, we are asking it to perform retrieval, filtering, reasoning, and synthesis in a single pass.

Hypothesis: By enforcing a strict "Map-Reduce" topology at the query level, we can isolate reasoning errors.

- **Map Phase:** Partition context by metadata (e.g., GROUP BY Department).
- **Reduce Phase:** Synthesize independent conclusions.

3. Methodology: Semantic MapReduce

OpenCQL implements a custom runtime that interprets SQL-like syntax not as data retrieval, but as **Reasoning Orchestration**.

3.1 The GROUP BY Experiment

```
SELECT risk_assessment
FROM gpt-4
JOIN KNOWLEDGE (source='audit_logs')
GROUP BY compliance_domain ('GDPR', 'SOC2', 'HIPAA')
```

The runtime intercepts this query and spawns three parallel "Agents," each with a context window restricted *only* to documents matching their specific domain. This prevents "GDPR" rules from confusing the "HIPAA" analysis.

4. Preliminary Architecture

The prototype consists of three core components:

1. **Compiler:** A Lark-based parser that validates query intent.
2. **Vector Abstraction Layer:** A mock interface for vector DB connectivity.
3. **Orchestrator:** A Python runtime managing async execution of 'Map' agents.

5. A Note on AI-Augmented Research

This project was developed as an experiment in **AI-Native Engineering**. The conceptual architecture, grammar design, and reasoning topology were architected by the author to address specific governance gaps observed in industry. The implementation details (Python boilerplate, AST generation) were accelerated using Generative AI tools. This methodology allowed for rapid prototyping of the DSL in under 48 hours.

6. Invitation to Collaborate

This is not a finished product; it is a proposal. We invite the community to fork the repository and test the efficacy of "Structured Context" vs. "Standard RAG" in their own pipelines.