The Cheapest Computation Is the One You Never Perform:

Lessons from Scaling Enterprise Graph Systems

William Liang | Principal Engineer | Adobe

Adobe

laptops)

Agenda

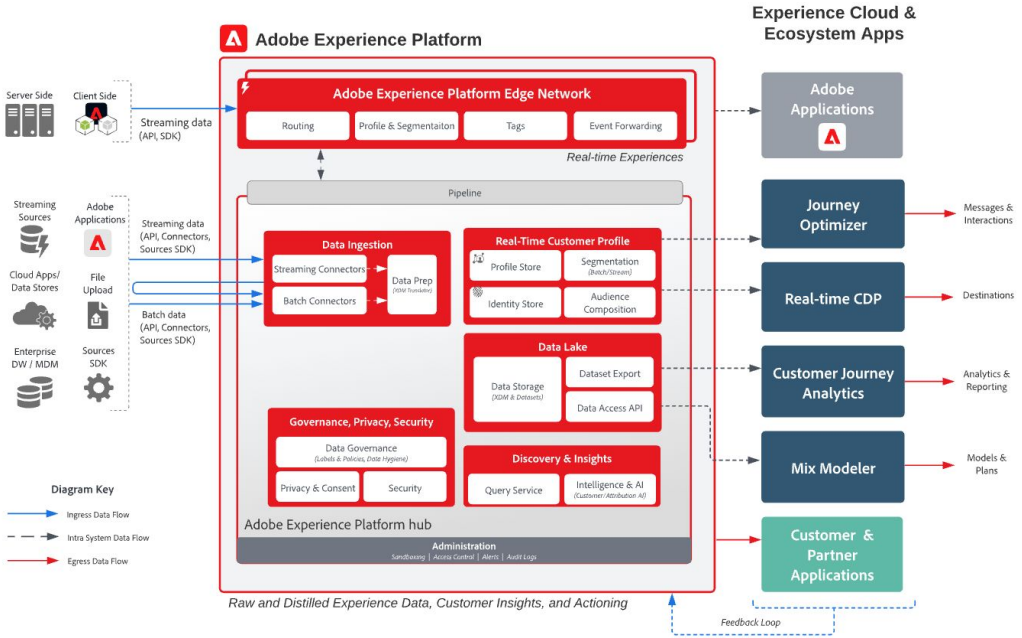
The Scale Problem:	Why graph systems become computationally expensive at scale
The Wrong Assumption:	Why scaling hardware and migrating to a graph DB both failed
Graph Complexity Defined:	Size, density, and connectivity
The Paradigm Shift:	Lean nodes, intelligent pruning, and workload rebalancing
The New Bottleneck:	Write amplification, serialization, and latency
Work Smarter Not Harder:	Achieving sub-100ms latency
The Next Frontier:	Unlocking Agentic Workflows

Adobe

laptops)

Adobe Experience Platform: Data Governance & Policy Enforcement Service

- AEP powers Adobe Experience Cloud applications
  - Journey Optimizer
  - Real-Time CDP
  - Customer Journey Analytics
- Data Governance and Management Framework
  - Classification and label data
  - Define data usage policy
  - Ensure data usage is compliant



laptops)

Adobe Experience Platform at Scale

Stat	Label	Source
1 Trillion+	Experiences powered per year	Adobe Summit 2026 press release
35 Trillion	Segment evaluations per day	Q1 FY26 earnings / Summit 2026
70 Billion	Profile activations per day	Q1 FY26 earnings
<100ms	Experience delivery latency	Adobe Tech Blog 2024

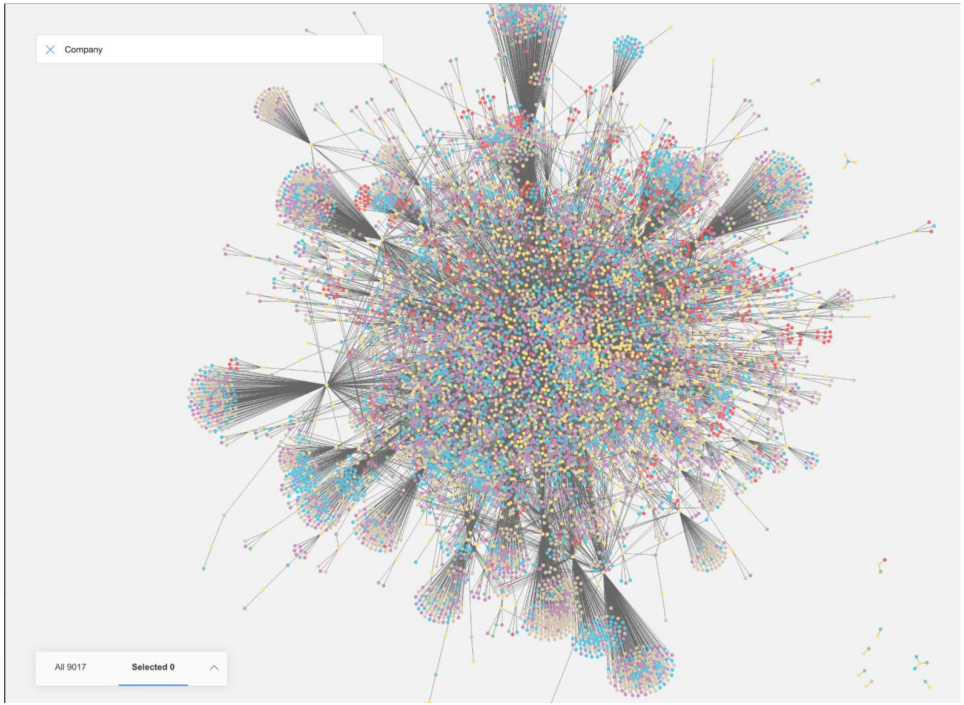
At this scale, inefficiencies don't slow you down. They take you down.

Adobe

laptops)

The Scale Problem: At scale graphs become computationally expensive

- Complex Graphs
  - High Connectivity
  - Dense Relationships
- Performance Issues
  - Throttling issues
    - 429 Too Many Requests
  - Concurrency issues
    - 409 Conflict
    - 412 Precondition Failed
    - 423 Lock
  - Latency issues
    - Latency in the order of minutes and hours

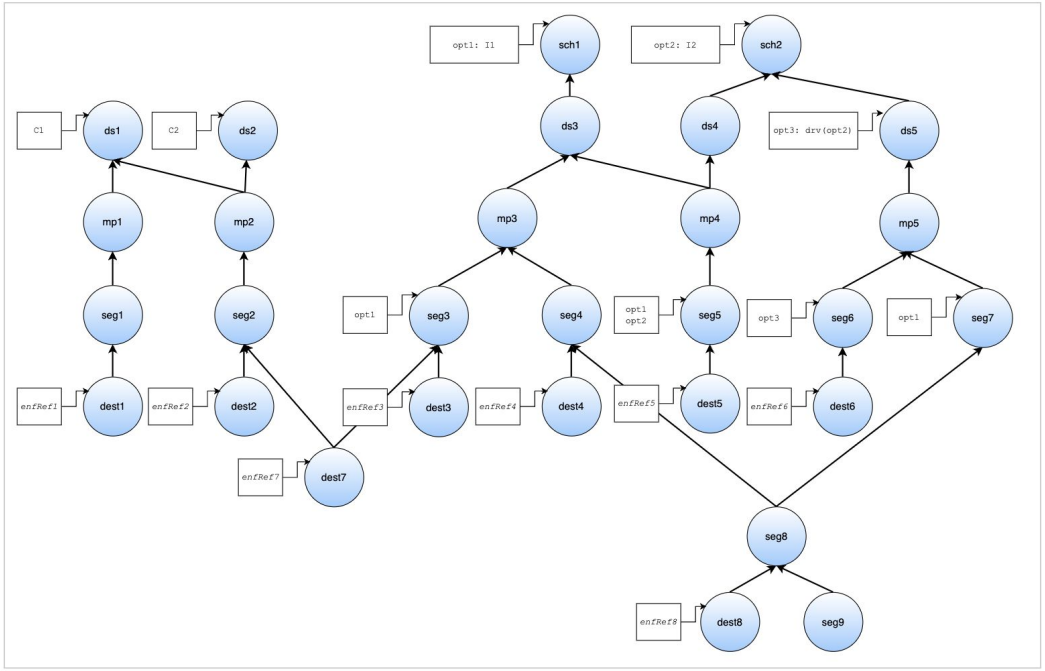


Adobe

laptops)

The Legacy Era Inefficiencies: Entity centric data model

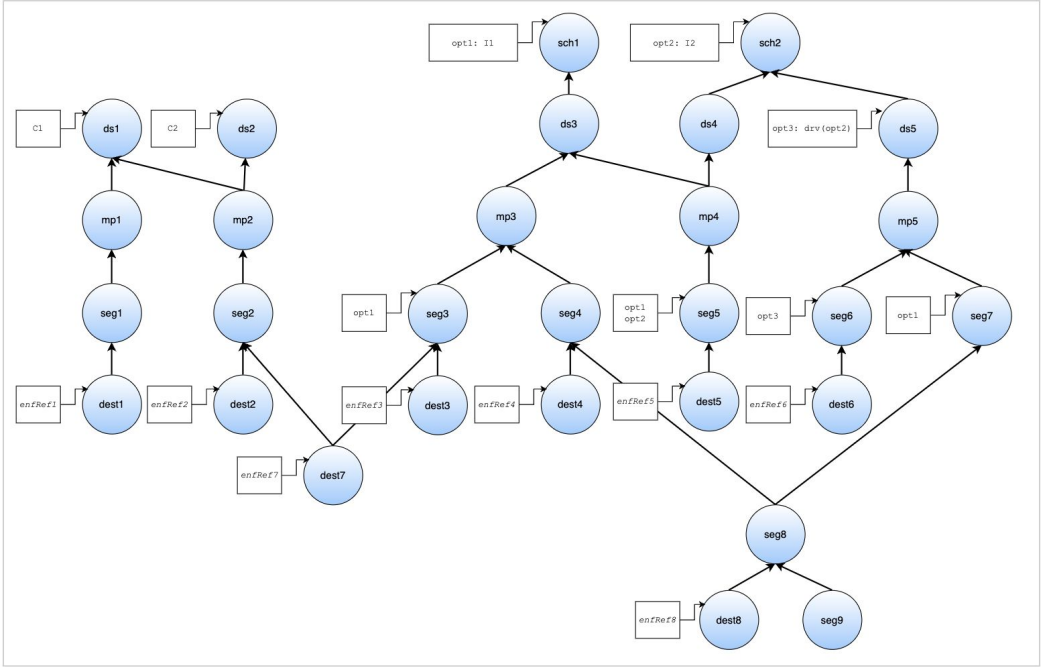
- PES data model was an entity with many properties
- Queries were entity centric
  - Optimized for the minority case
  - Full traversal was required in all cases
  - Most paths produce no labels or marketing actions
- Queries by properties were not supported
  - What labels and/or marketing actions are in-use



laptops)

The Legacy Era Inefficiencies: Brute force traversal

- Business Logic
  - Relied on exhaustive, brute-force graph traversal
- Scaling Problem
  - Performance was a direct function of graph complexity
- Breaking Point
  - High CPU utilization
  - High memory utilization
  - High latency

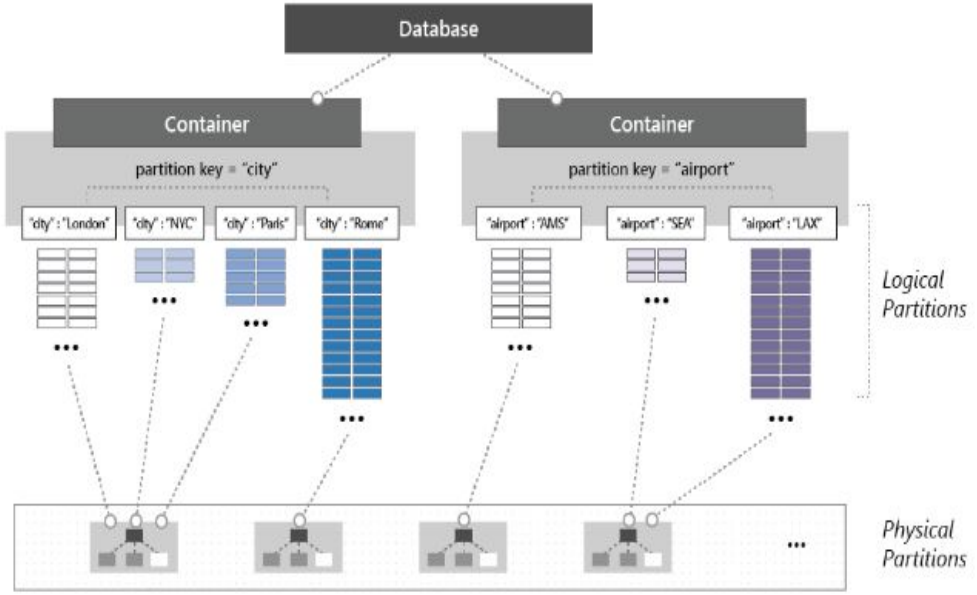


laptops)

The Wrong Assumption: Graph systems failed because graph is too complex

- Need More CPU
  - To improve latency
- Need More Memory
  - To handle larger requests
- Need More Hardware
  - To improve throughput

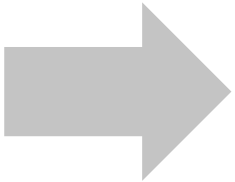
The following image shows how a physical partition hosts one or more logical partitions of a container:



Adobe

laptops)

Lift and Shift: Migrate to graph database



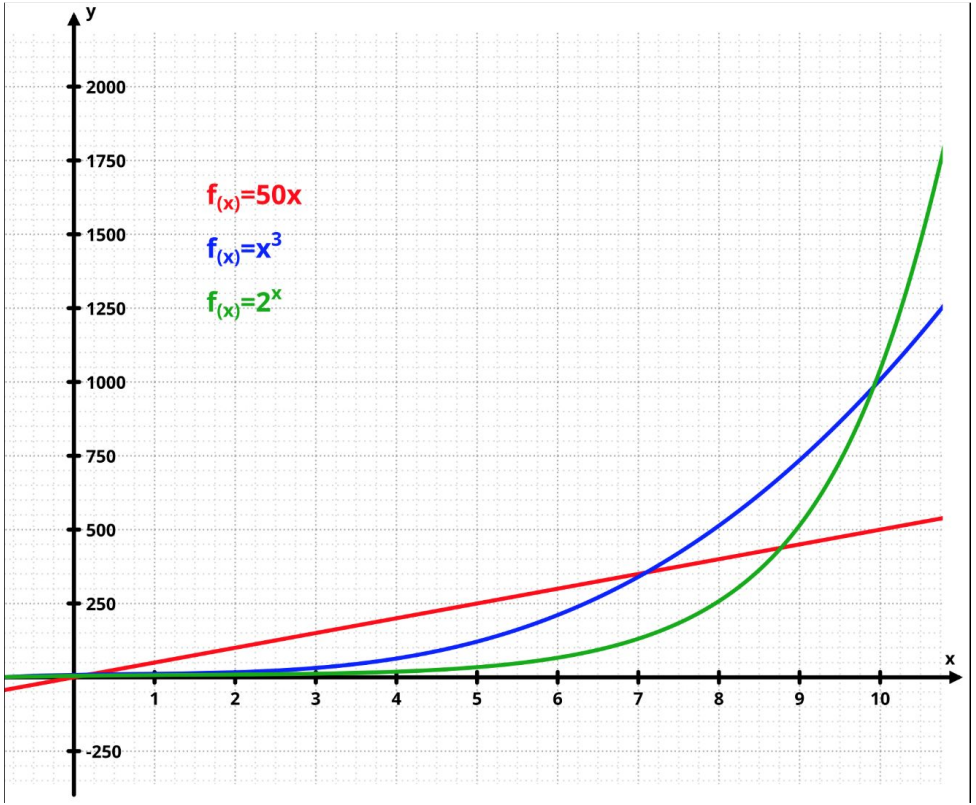
- Migrating to Graph Database
  - Did Not Address Performance and Scalability Issue
- Issues and Inefficiencies were ported forward
  - Did Not Address the Root Cause

Adobe

laptops)

The Discovery: We were solving the wrong problem

- Graph systems don’t fail because they’re too complex
- Graph systems fail because the workload is increasing exponentially
- Graph systems fail because we’re doing too much work

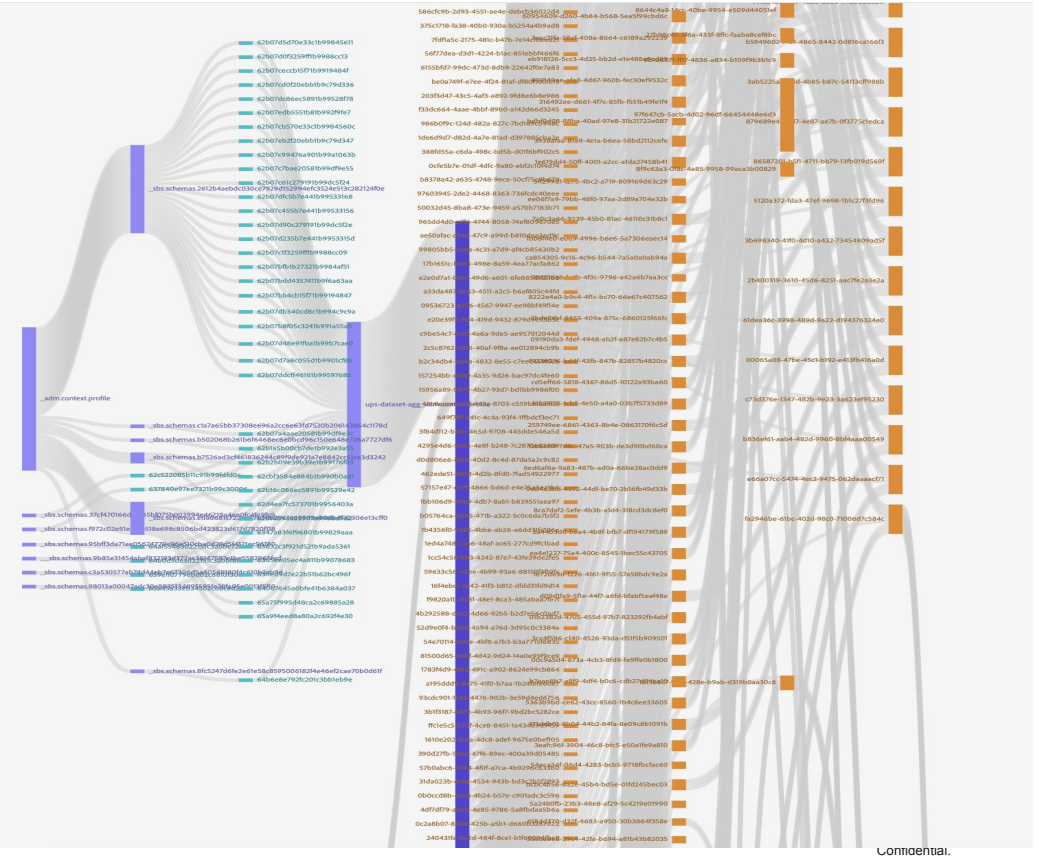


Adobe

laptops)

Defining Graph Complexity: Size, density (super nodes), and connectivity

- Graph Size
  - Node cardinality
  - Edge cardinality
- Graph Density
  - Local Density
    - "Super Nodes" that have massive fan-in or fan-out of edges
  - Global Density
    - Highly connected sub-graphs, such as every destination being related to every segment
- Graph Connectivity
  - Path cardinality
  - Path depth

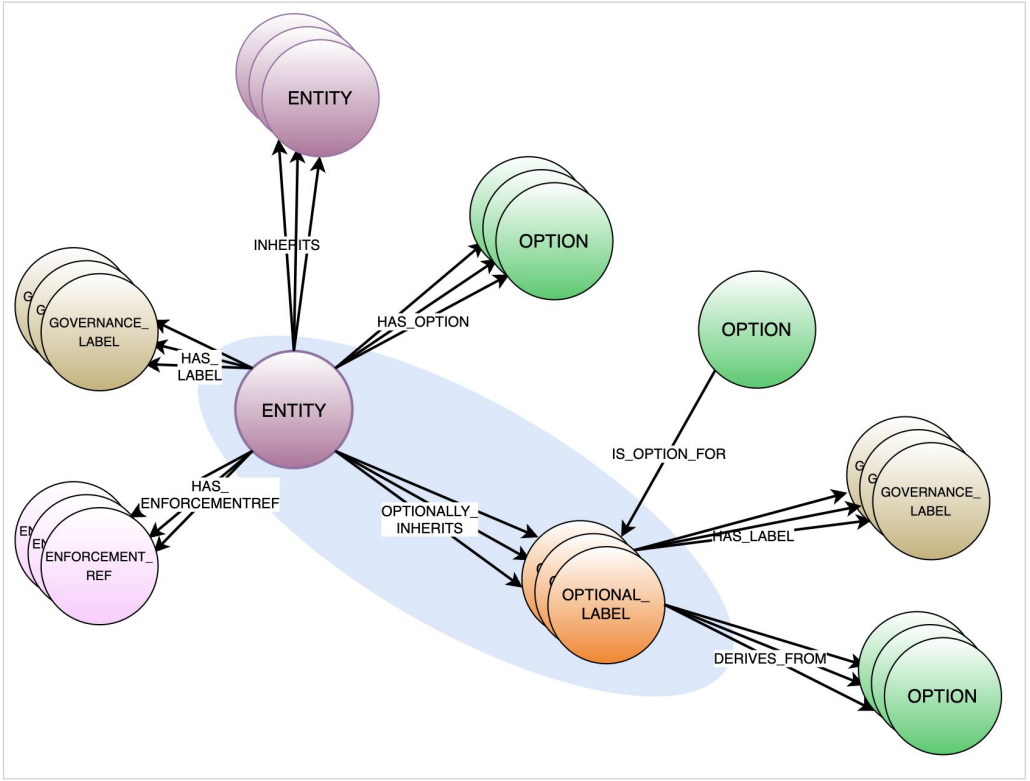


Adobe

laptops)

The Paradigm Shift: Three shifts that collapsed graph complexity

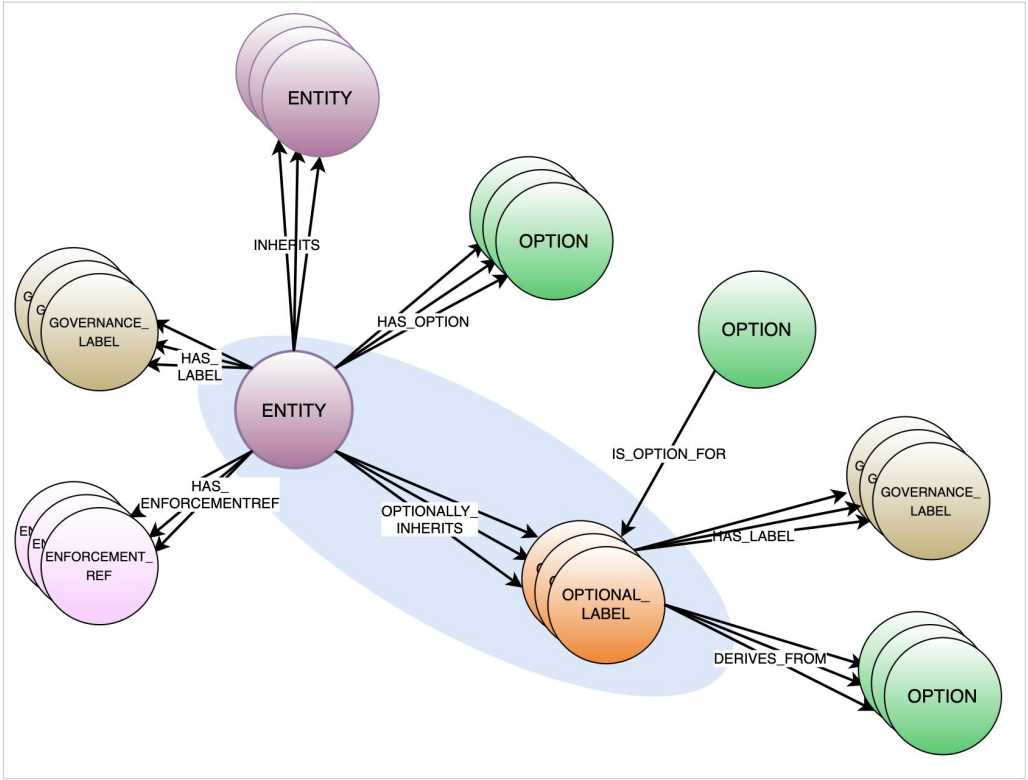
- Structural Shift (Lean Nodes)
  - The data model evolved from large (property-heavy) nodes to lean nodes
- Logic Shift (Intelligent Pruning)
  - The business logic evolved from brute-force traversal to intelligent pruning
- Workload Shift (Reads to Writes)
  - Performance constraints evolved from read latency to write maintenance



laptops)

Lean Nodes: Property centric data model

- Entity properties factored out as first-class objects
- Query are now property centric
  - Optimize for the majority case
    - Optimized for enabled policies
    - No more brute force traversal
  - Selective property updates
    - Only update changed properties
  - Separate read and write concerns
    - Load balance across availability zones

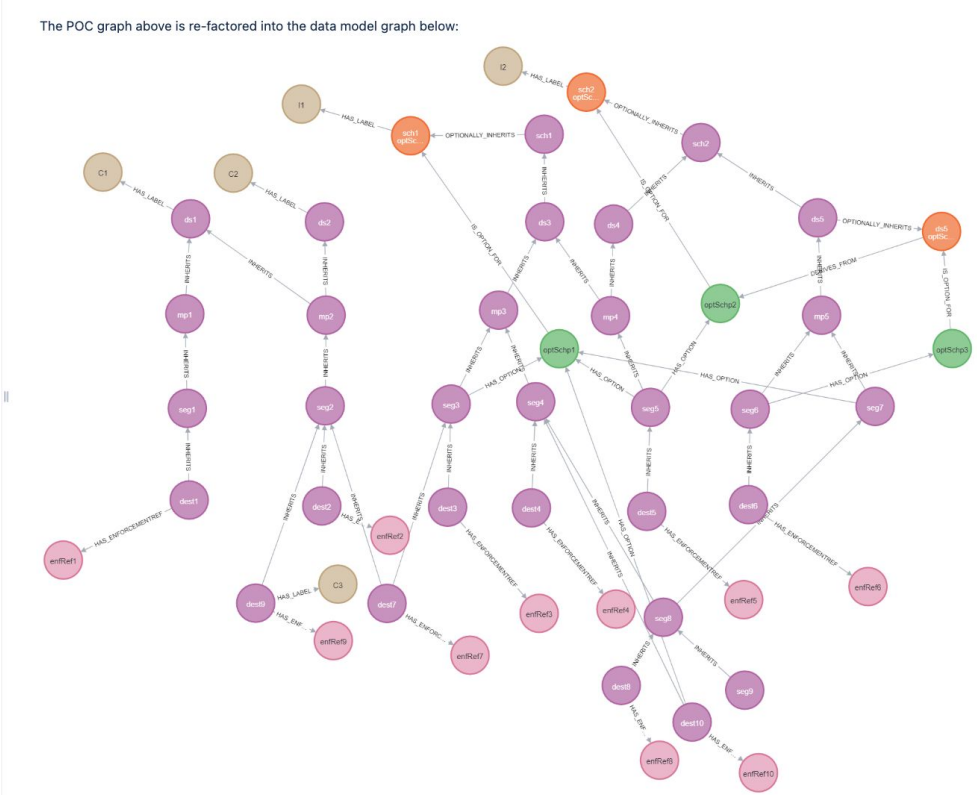


Adobe

laptops)

Surgical Search: Navigating complex graphs with intelligent pruning

- Reduce Graph Complexity to Policy Complexity
  - Workload grows linearly, not exponentially
- Read performance is now a function of the following
  - The number of enabled policies
  - The number labeled fields in schemas
    - The brown nodes
  - The number of fields in segments and audiences
    - The orange nodes
  - The number of marketing actions in destinations, journeys, etc.
    - The pink nodes

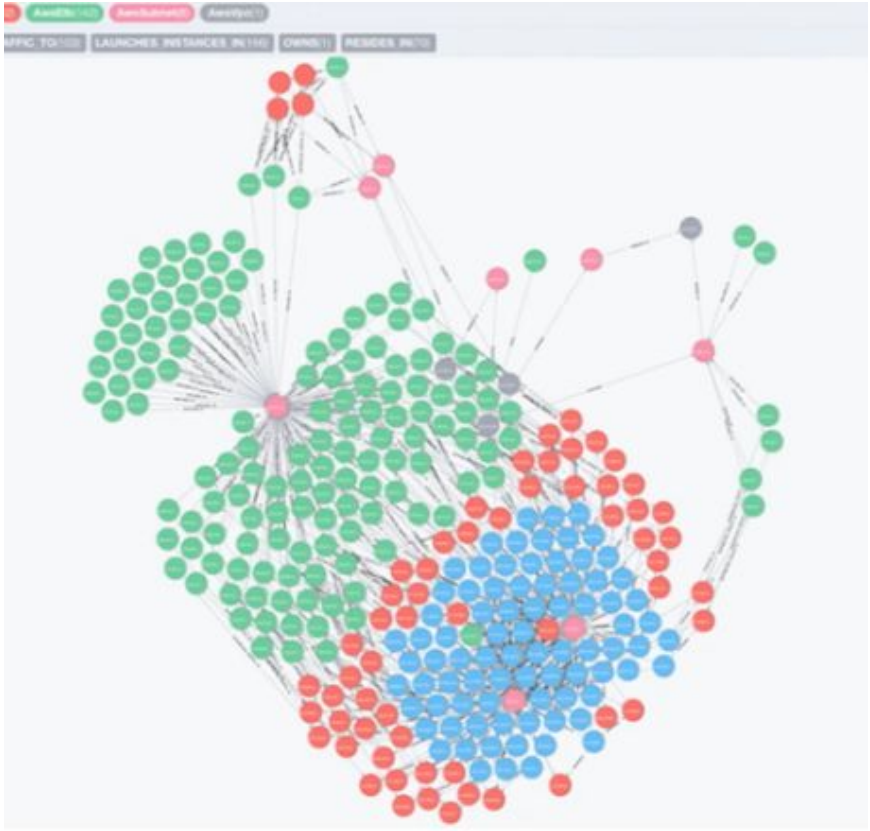


Adobe

laptops)

The New Bottleneck: Write Performance

- Write amplification (The "Fan-out" Effect)
  - Updates to a node fans out updates to its relationships and neighboring nodes
- Write serialization (The "Wait Queue")
  - Dense graphs result in a large lock footprint which results in lock contention for concurrent requests
- Write latency (The "Saturated Leader")
  - High volume and high frequency of writes results in high CPU utilization on the Leader in an availability zone

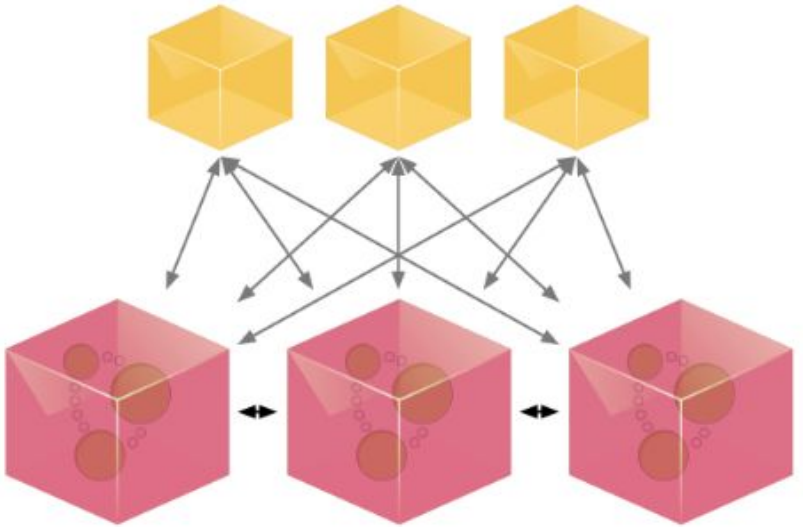


Adobe

laptops)

Optimization Wins: Refactored requests and removed unnecessary relationships

- Write amplification (The "Fan-out" Effect)
  - Eliminated unnecessary relationships
- Write serialization (The "Wait Queue")
  - Support a proper PATCH request
- Write latency (The "Saturated Leader")
  - Delineates and directs only write requests at the Leader in an availability zone

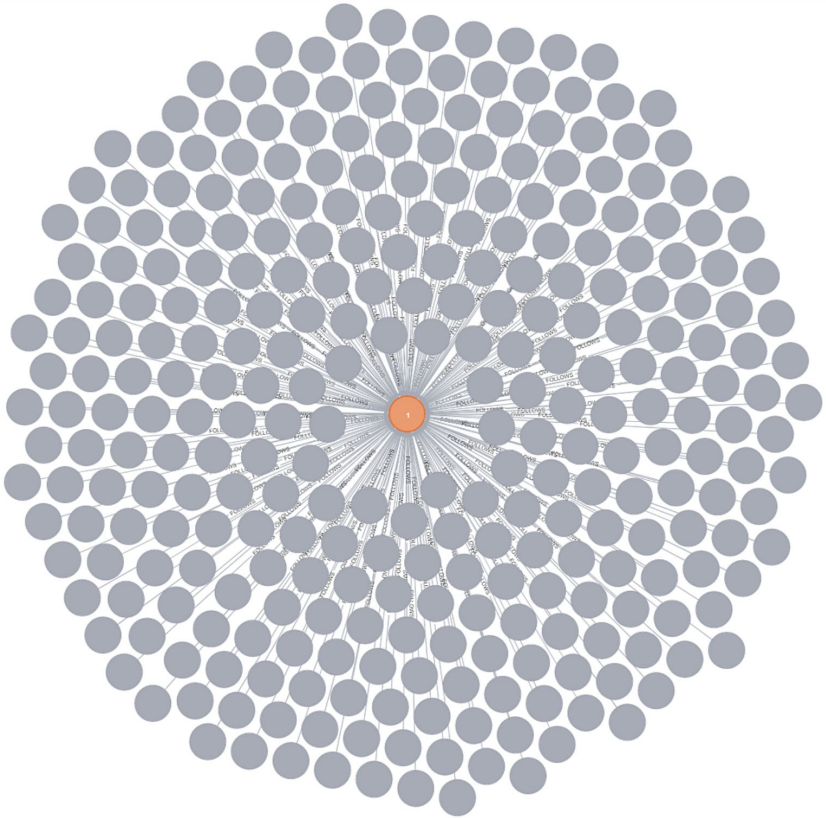


Adobe

laptops)

Work Smarter Not Harder: Graph traversal isn't about traversing faster, it's about traversing less

- Refactor to Atomic Updates
  - Shift away from batch operations to reduce lock contention and to avoid gateway timeouts
- Refactor to PATCH Updates
  - Transition to PATCH when updating nodes with large number of fields to minimize write amplification and lock footprint
- Refactor Wildcard Usage
  - Minimize wildcard properties to prevent super-nodes
- Create Logical Grouping
  - Nodes with 10s of thousands of properties should be grouped logically



An example of a super node, AKA dense node.

laptops)

Results

Metric	Before	After
P90	Seconds	100ms
P95	Mins	200ms
P99	Hours	1000ms

laptops)

The Next Frontier: AI Enablement

- Yesterday
  - Optimize graph queries.
- Today
  - Eliminate unnecessary computation.
- Tomorrow
  - Enable agentic workflows and tooling for data governance, knowledge graphs, and data lineage.

- Key Takeaway
  - AI agents amplify graph workloads. They don’t change the laws of scalability.
  - The cheapest computation is still the one you never perform.

Adobe


Adobe