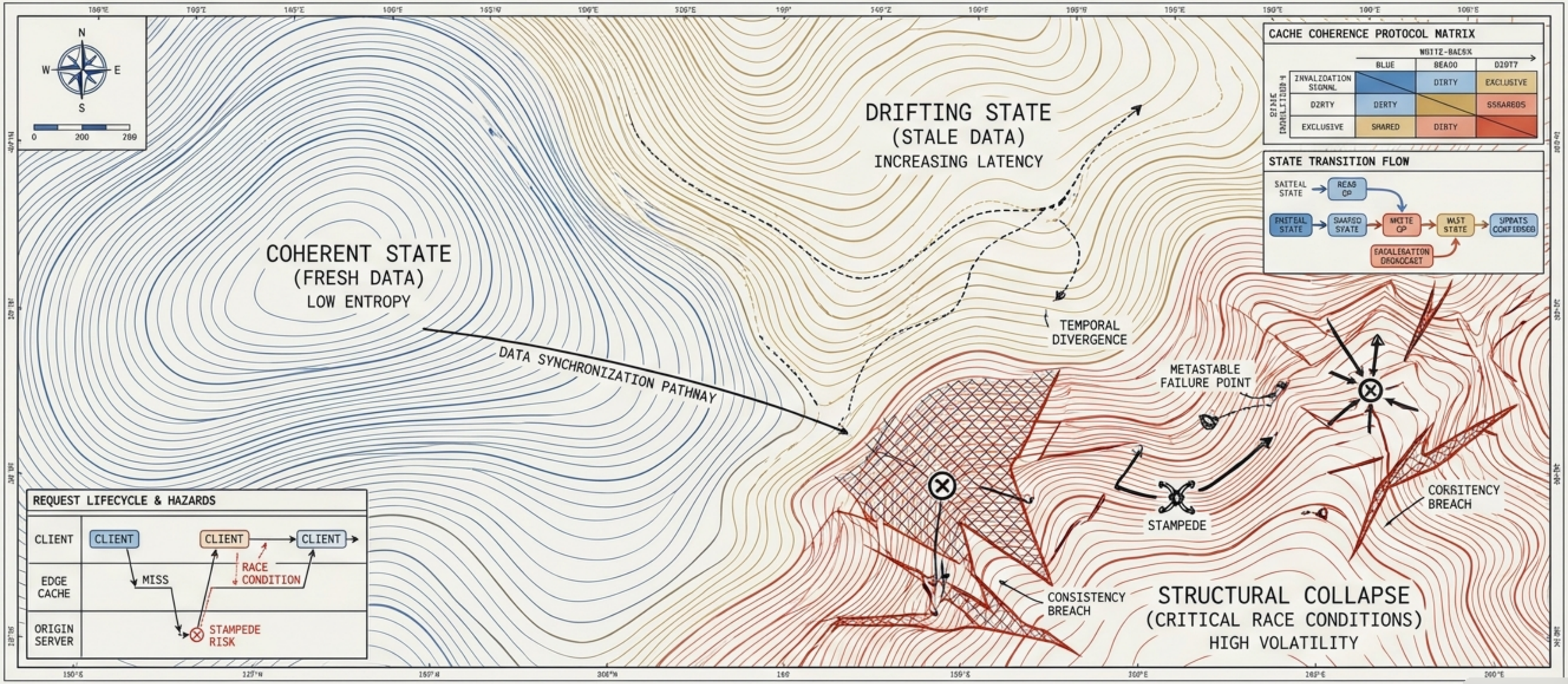
HARD PROBLEMS IN COMPUTING — S08E09: CACHE INVALIDATION

The Architecture of State, Metastable Collapse, and the Physics of Copied Truth

[01] PHYSICS [02] 6-LAYER MATRIX [03] RACES [04] STAMPEDES [05] HTTP EDGE [06] LLM TRAPS [07] AI PLAYBOOK



Gemini Notebook

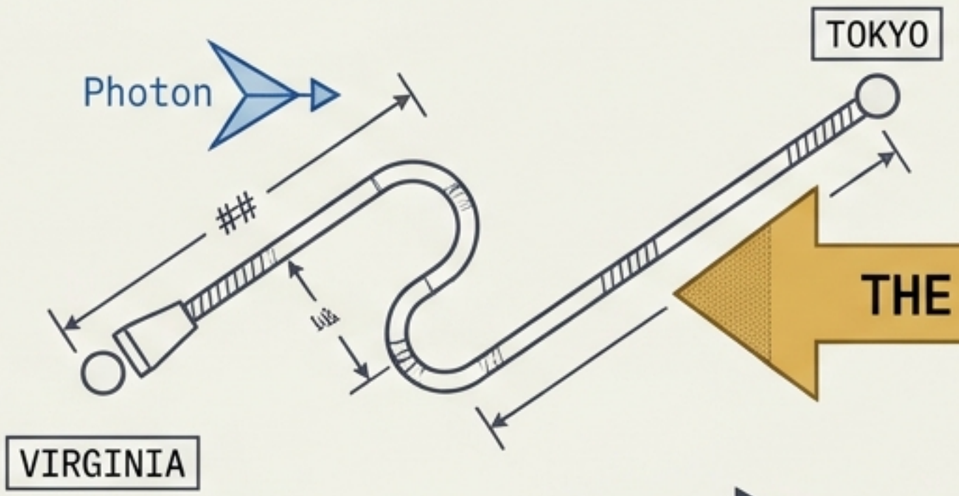
PHIL KARLTON'S MAXIM & THE REBELLION AGAINST LIGHT

Continuity Recap from S08E08: **String Encoding & Unicode**

"There are only two hard things in Computer Science: cache invalidation and naming things."

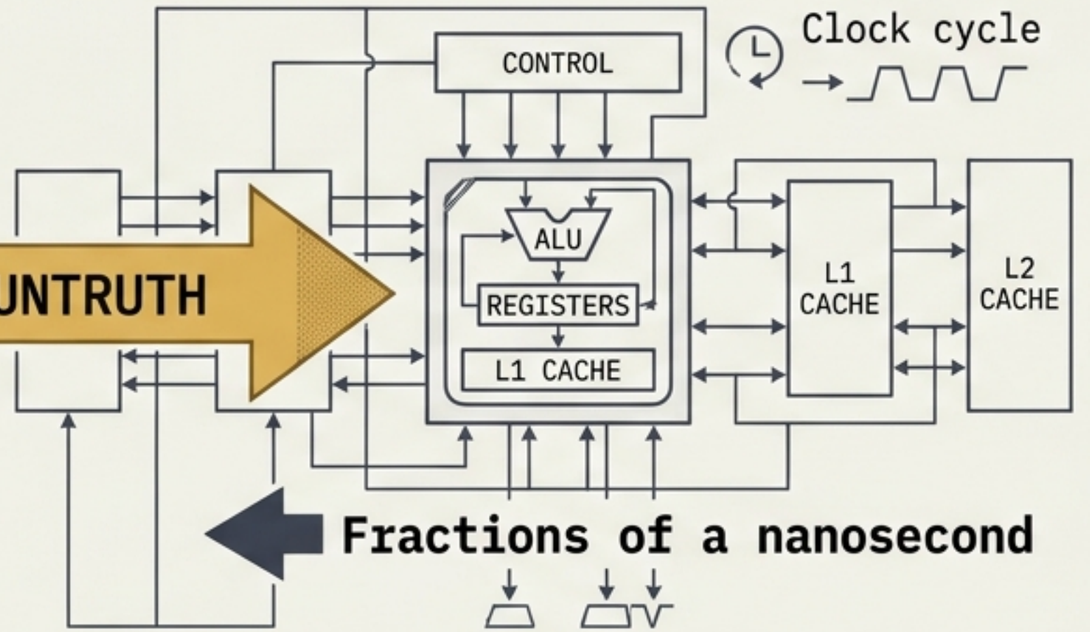
Caching is fundamentally a rebellion against the speed of light—copying state across space creates unavoidable staleness.

MACRO-SCALE GEOGRAPHIC ROUTING



100ms+ latency

MICRO-ARCHITECTURAL CPU EXECUTION



Fractions of a nanosecond

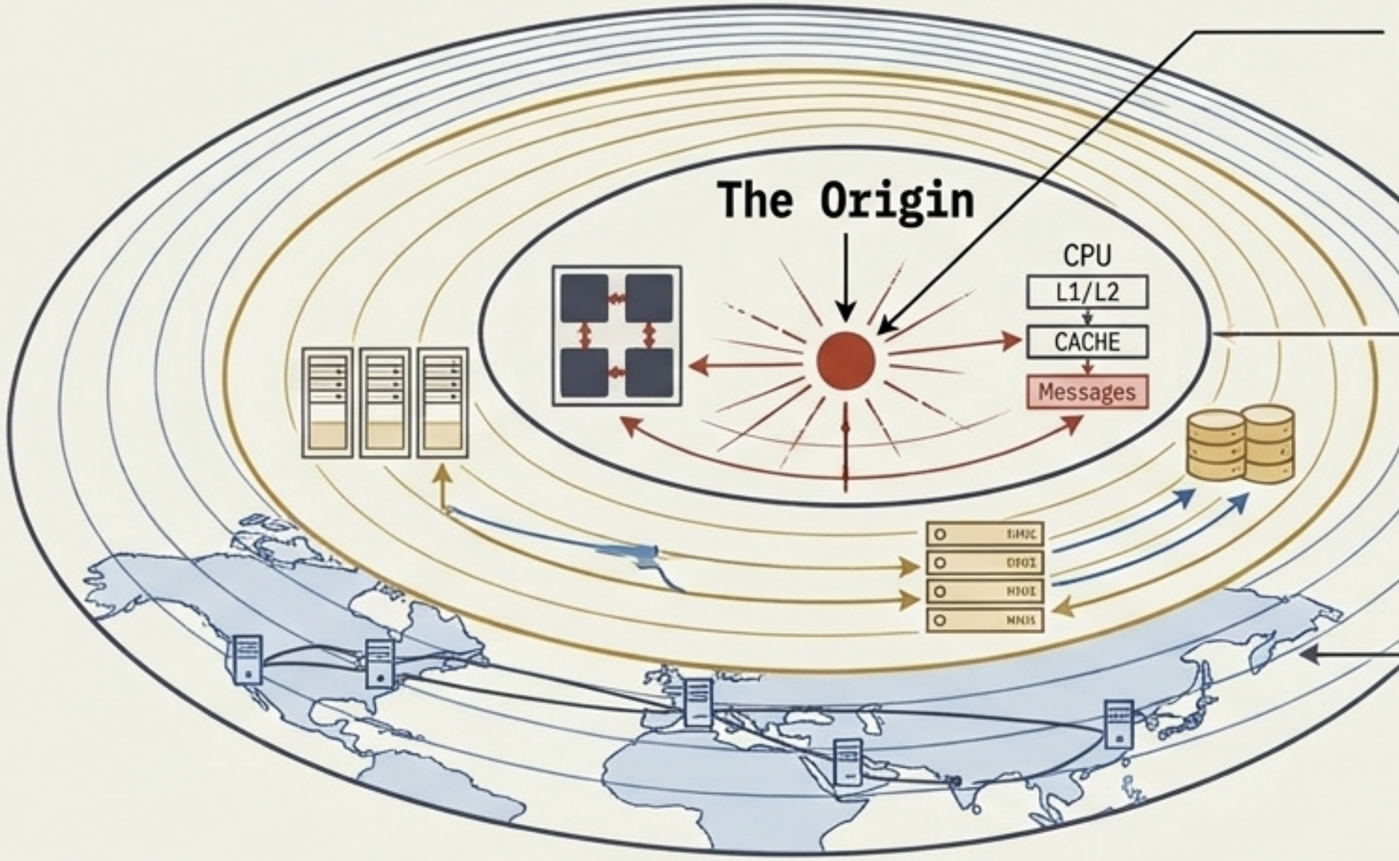
THE GAP OF UNTRUTH

Storing data close to the **CPU** buys **speed** but sacrifices **universal truth.**

Gemini Notebook

THE BLAST-RADIUS OF STALENESS

Coherence traffic couples systems across all distances. A single cache miss or invalidation ripples outward, creating coupling from silicon dies up to global CDNs.



Ring 1 (Nanometers):
CPU L1/L2
(MESI RFO messages)

Ring 2 (Meters):
Datacenter Distributed Caches
(Redis invalidation streams)

Ring 3 (Kilometers):
CDN Edge networks
(RFC 9111 HTTP semantics)

Every cached layer creates a **shadow architecture** of **invalidation traffic.**

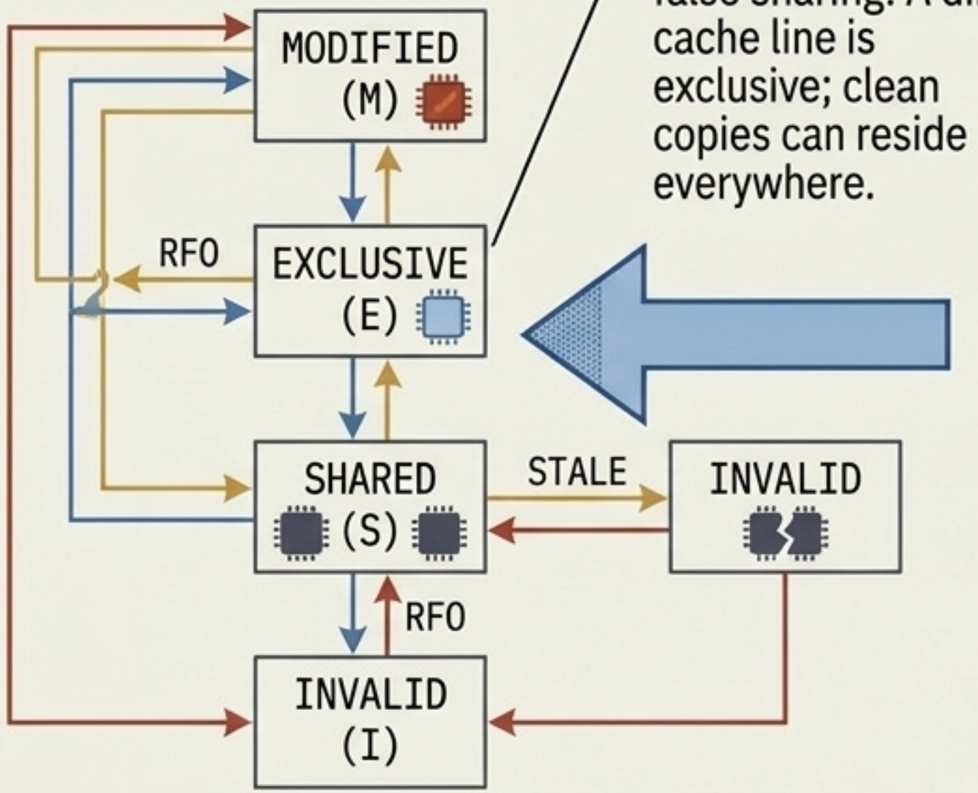
Gemini Notebook

THE FRACTAL NATURE OF CACHING

From 64-byte silicon lines to 10-megabyte edge assets, the theoretical constraints are identical.

HARDWARE LAYER

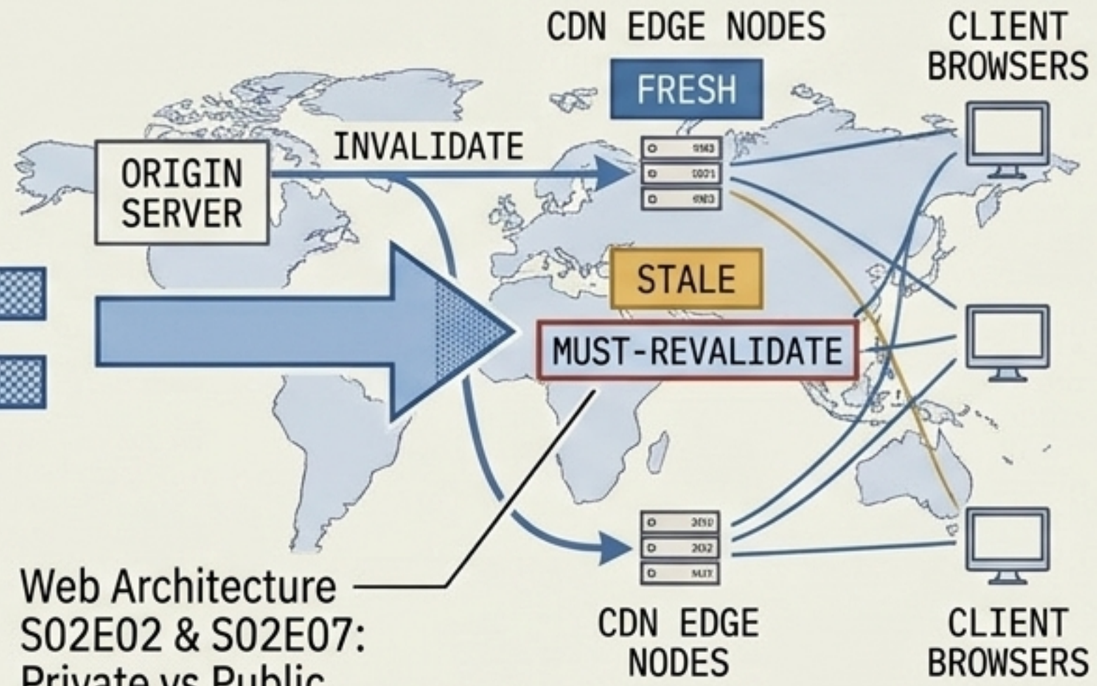
CPU MESI Coherence



Martin Thompson's false sharing: A dirty cache line is exclusive; clean copies can reside everywhere.

DISTRIBUTED LAYER

CDN Edge Invalidation



Web Architecture S02E02 & S02E07: Private vs Public cache rules.

Gemini Notebook

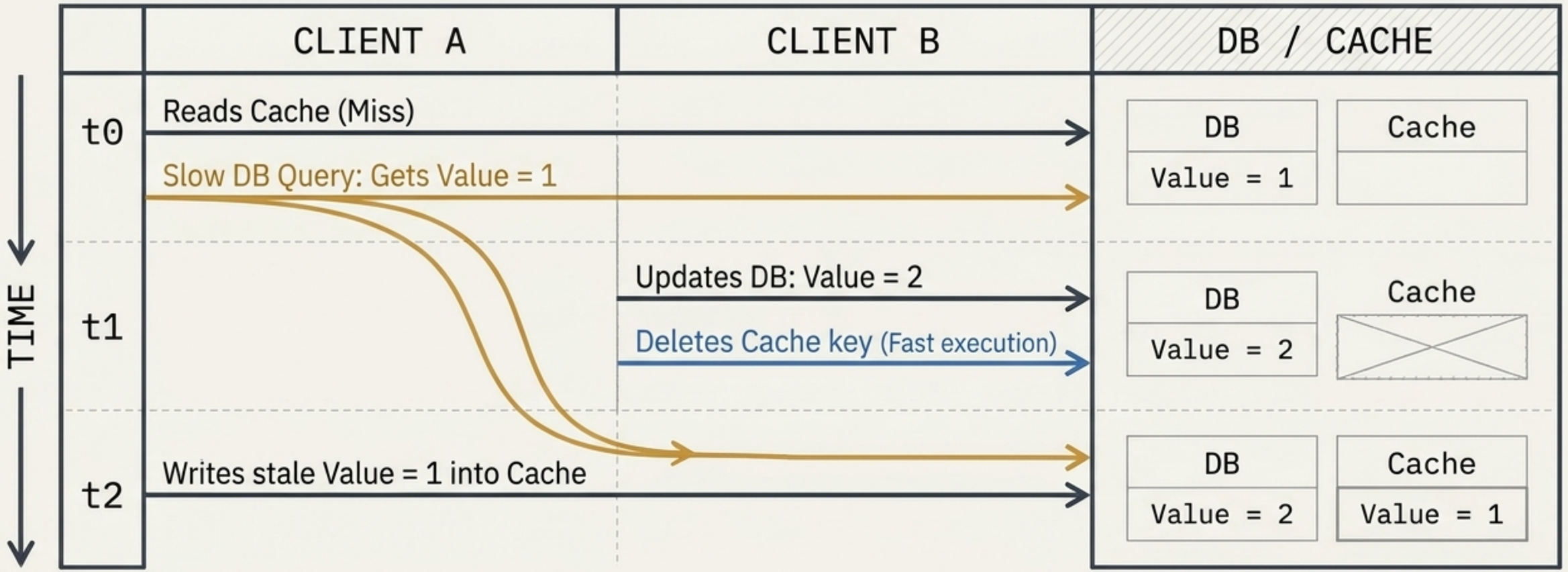
THE 6-LAYER FULL-STACK CACHE MATRIX

	1) CPU L1/L2/L3	2) OS Kernel	3) App L1 Heap	4) Distributed L2	5) CDN Edge	6) Client
Tech	MESI	Page Cache	Caffeine TinyLFU	Redis / CacheLib	RFC 9111	Service Worker
Latency	~1-10 ns	~1 µs	~100 ns	~1-10 ms	~10-50 ms	Local
Scope	Core/Die	Machine	Process	Datacenter	Global	Browser
Eviction	LRU/LFU	LRU/Clock	Window TinyLFU	LRU/TTL	TTL/Quota	LRU
Invalidation	RFO Messages	fsync/writeback	Explicit TTL	RESP3 Streams	s-maxage/must-revalidate	Clear-Site-Data

Gemini Notebook

THE CACHE-ASIDE STALE-OVERWRITE RACE

How a slow read-miss beats a write+delete, permanently corrupting the cache.



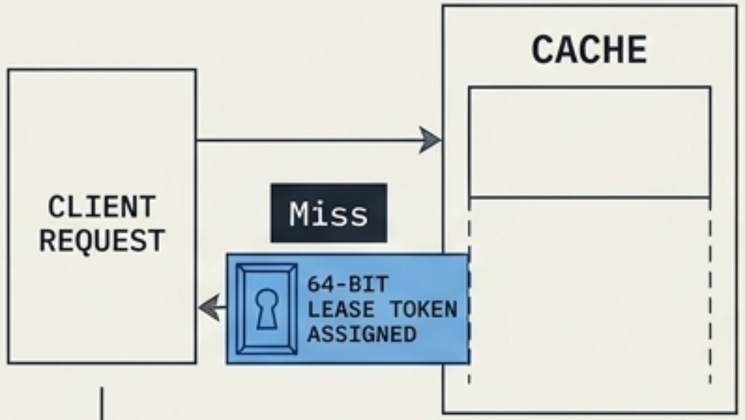
WARNING: The cache now holds Value=1 while the DB holds Value=2. The cache is **poisoned** infinitely until TTL expires.

Gemini Notebook

FACEBOOK'S NSDI 2013 LEASE TOKEN SOLUTION

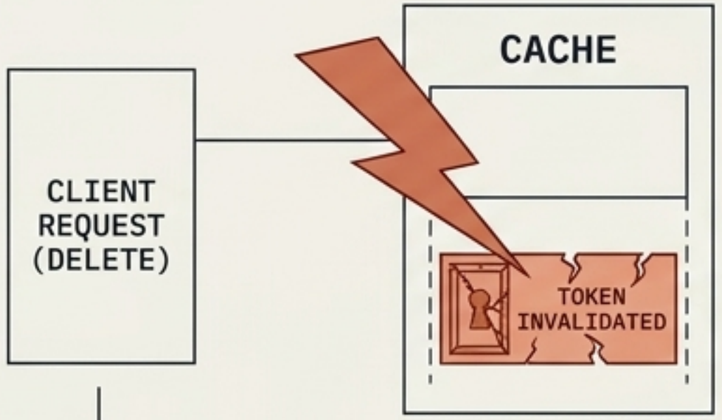
Memcached's 64-bit lease tokens prevent stale sets by acting as an optimistic concurrency lock.

STEP 1: LEASE ASSIGNMENT



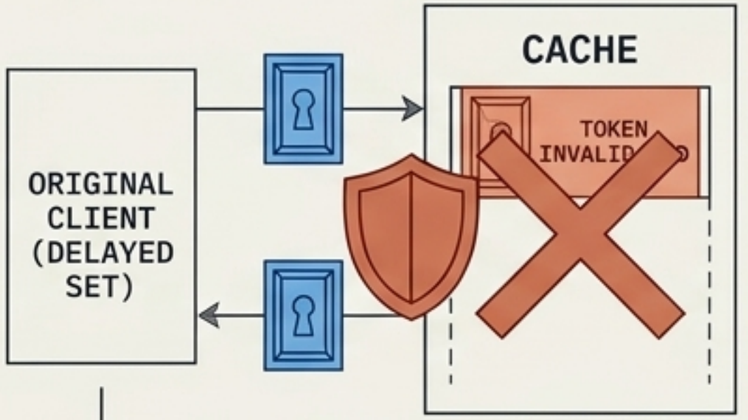
Client makes a request to an empty cache. The cache responds with a miss and a unique lease token.

STEP 2: CONCURRENT INVALIDATION



A concurrent "Delete" request from a different client invalidates the lease token at the cache layer.

STEP 3: STALE SET REJECTION



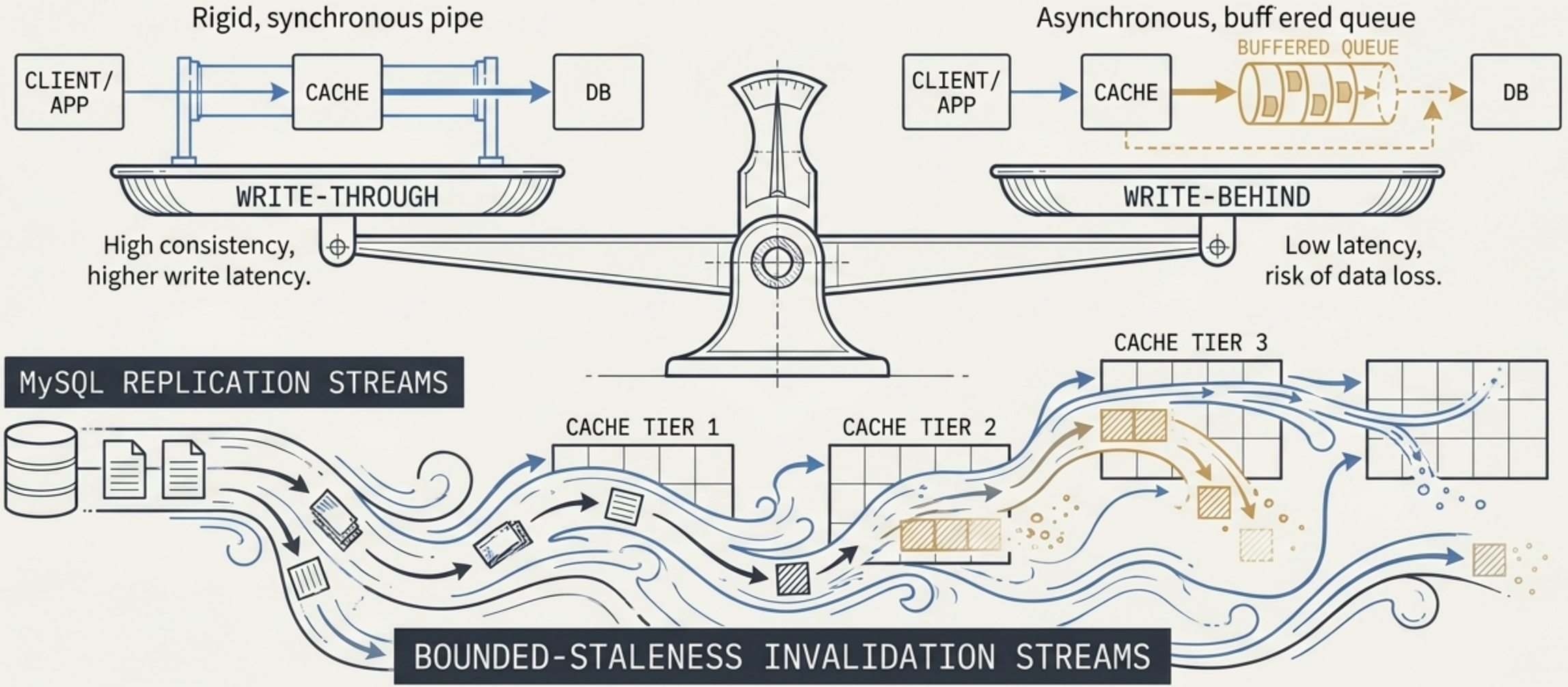
The original client attempts a delayed "Set" operation, presenting the stale token. The cache rejects the write, preventing a stale overwrite.

Arbitrating concurrent writes at the cache layer prevents stale overwrites.

Gemini Notebook

TOPOLOGY TRADE-OFFS: WRITE-THROUGH VS. WRITE-BEHIND

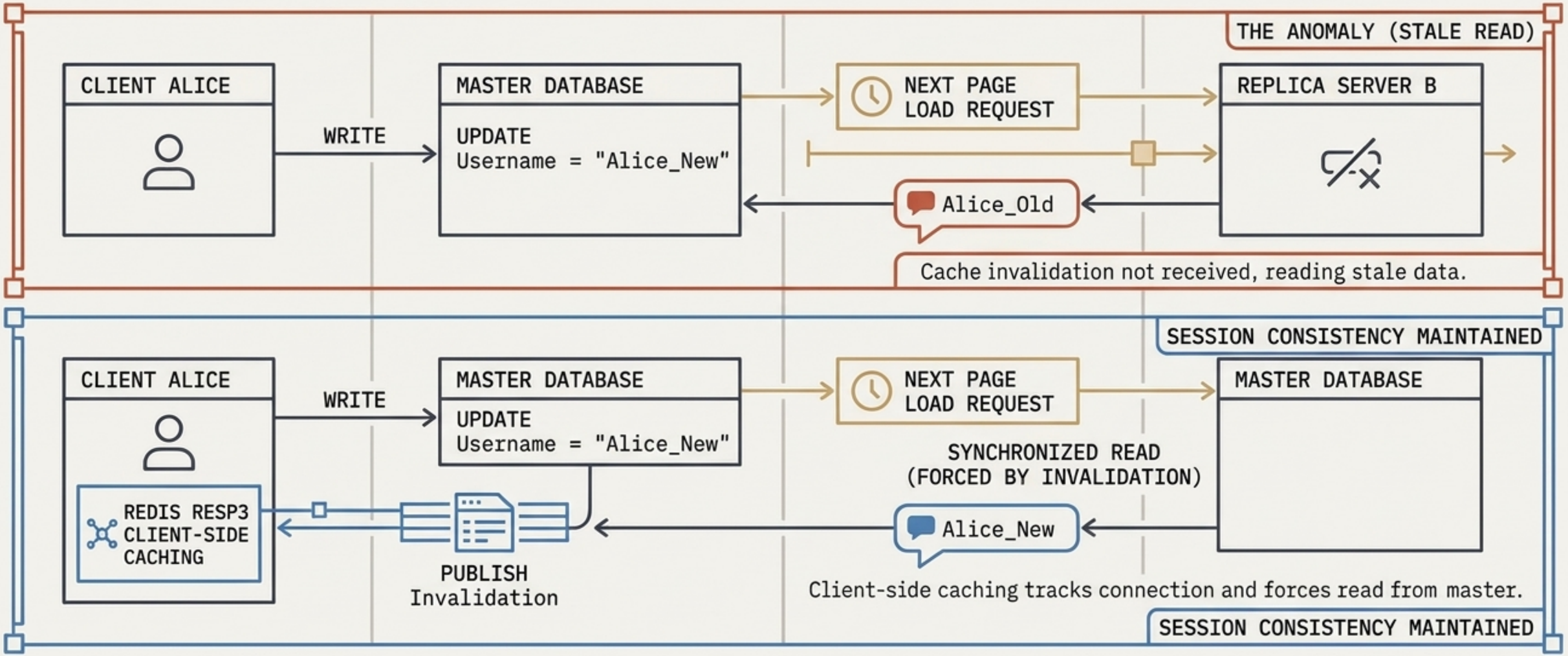
Balancing latency against consistency across structural cache topologies.



Gemini Notebook

JEPSEN ANOMALIES & SESSION CONSISTENCY

Solving the “Read-Your-Writes” anomaly where a user updates their profile but immediately sees old data.



Gemini Notebook

THE MATH OF BIMODAL COLLAPSE

Caches hide structural rot. A tiny drop in hit rate causes exponential downstream load.

DATA POINT 1:
99% HIT RATE = 1% MISS RATE
(100 DB QUERIES/SEC).



DATA POINT 2:
98% HIT RATE = 2% MISS RATE
(200 DB QUERIES/SEC).



A 1% DROP IN CACHE EFFICIENCY =

↓ X200%

DATABASE LOAD.

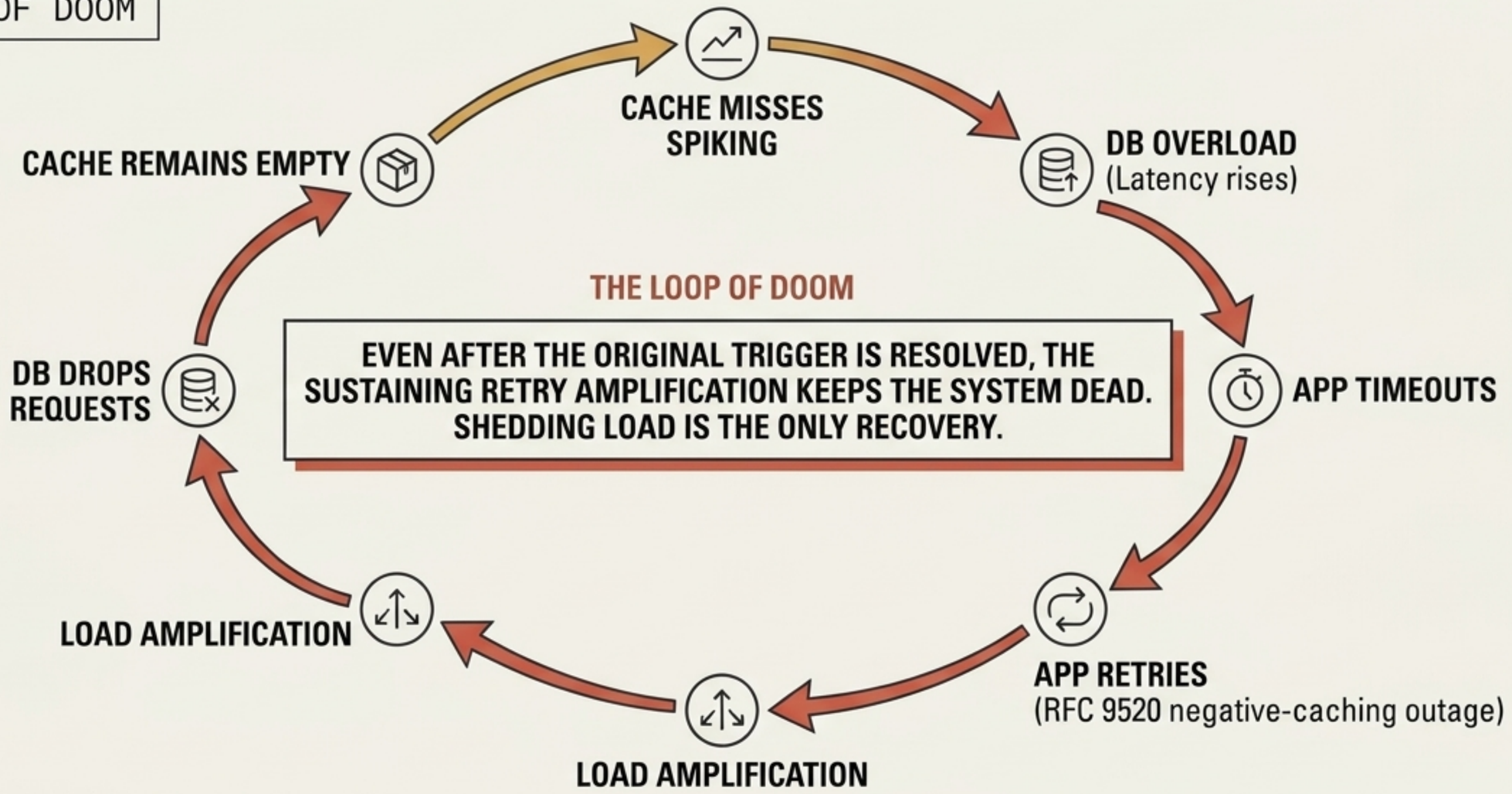
> “Load testing doesn’t catch this because caches extract cacheability—they behave better under predictable load than real-world heavy-tailed distributions.” (Marc Brooker)

Gemini Notebook

METASTABLE STAMPEDES & SUSTAINING LOOPS

Why cold caches cause total system collapse (ACM HotOS 2021). A trigger causes a cache loss, pushing the DB into unrecoverable overload.

THE LOOP OF DOOM

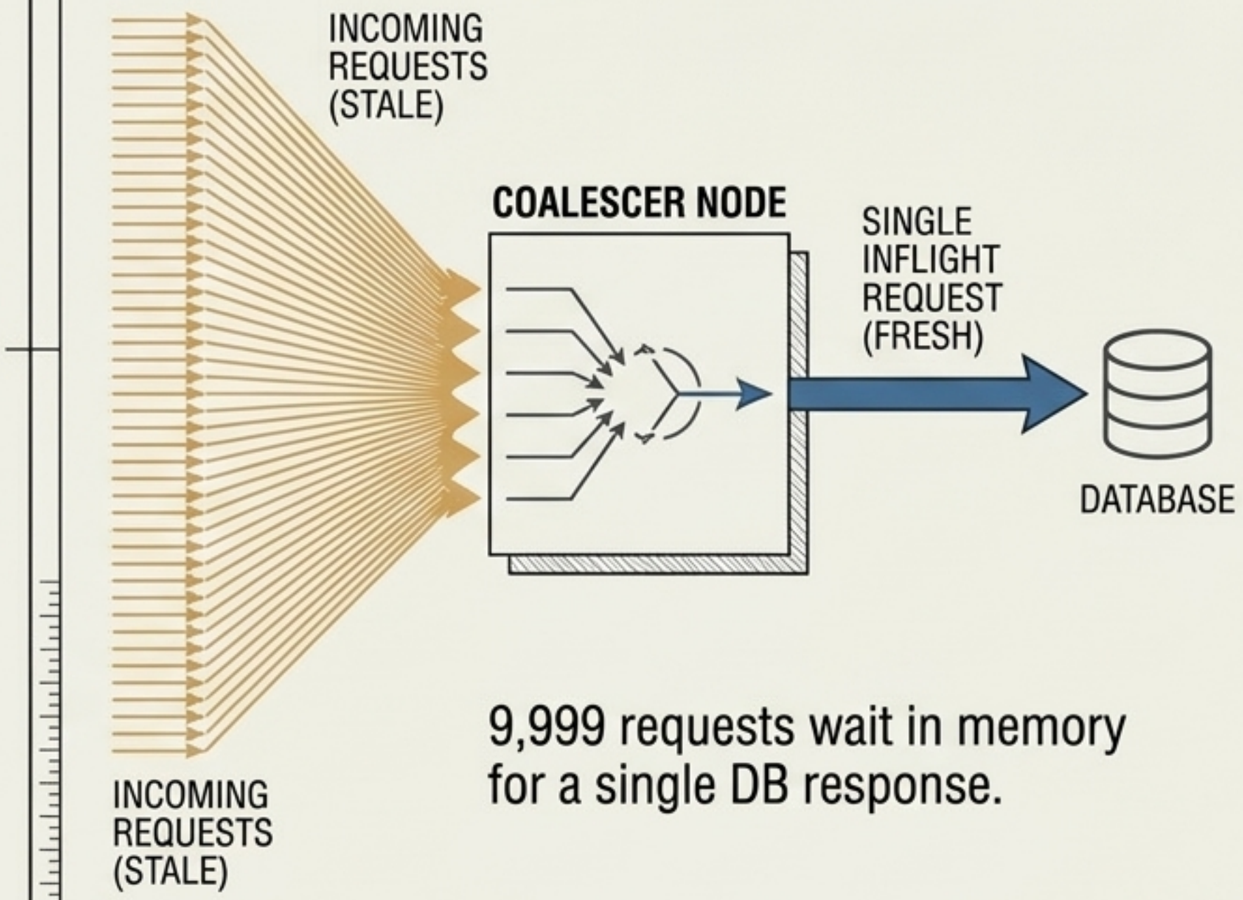


Gemini Notebook

STAMPEDE DEFENSES I: COALESCING & JITTER

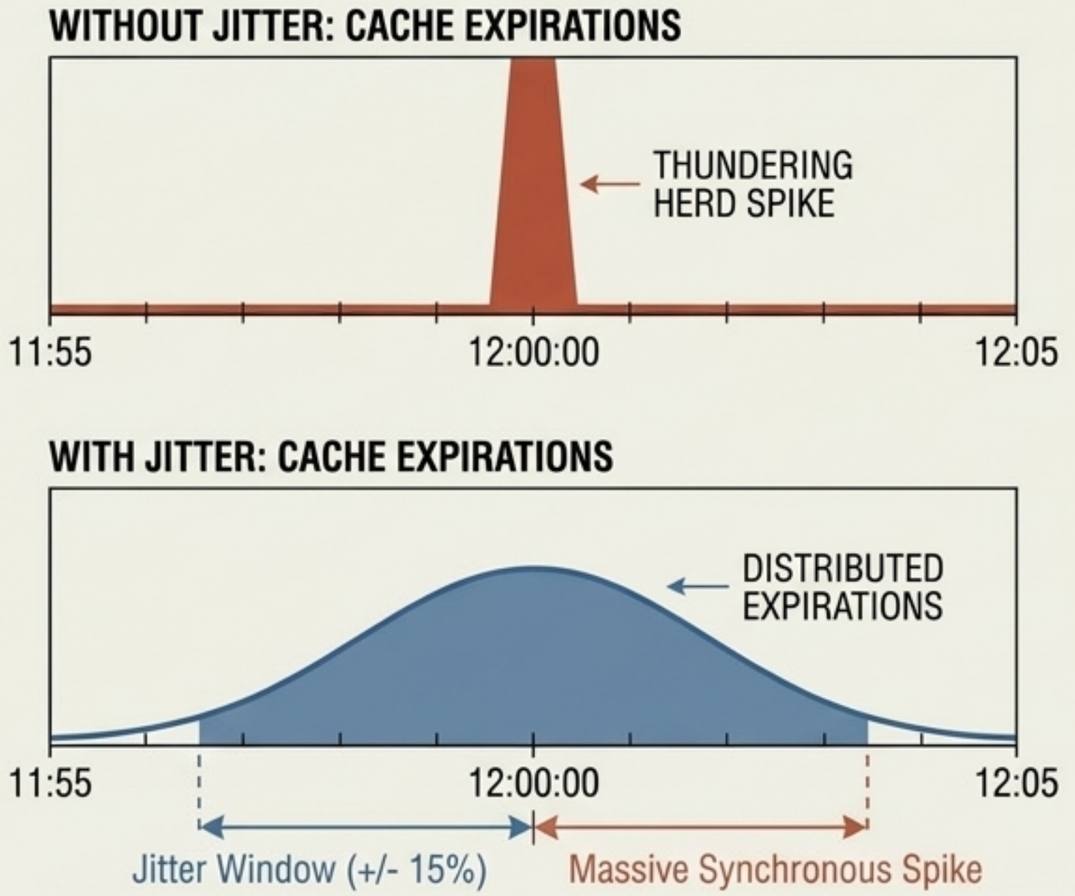
Stopping thundering herds at the application boundary.

GO SINGLEFLIGHT



9,999 requests wait in memory for a single DB response.

+/- 15% TTL Jitter

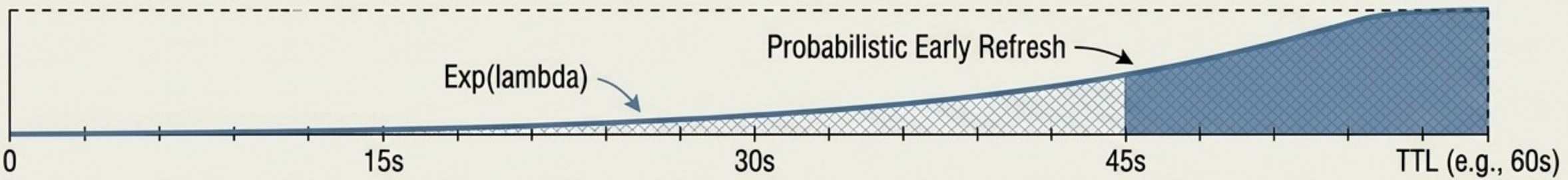


Gemini Notebook

Stampede Defenses II: XFetch & Stale-While-Revalidate

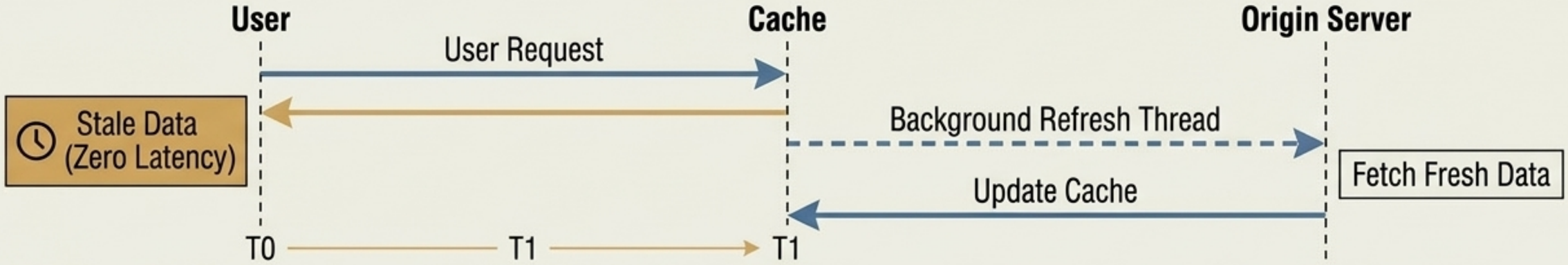
More advanced probabilistic and asynchronous techniques for preventing thundering herds, moving beyond simple coalescing and jitter.

VLDB 2015 XFetch Probabilistic Early Refresh



A request independently decides to refresh early based on the curve's probability, distributing refreshes before the hard expiration spike.

RFC 5861: stale-while-revalidate

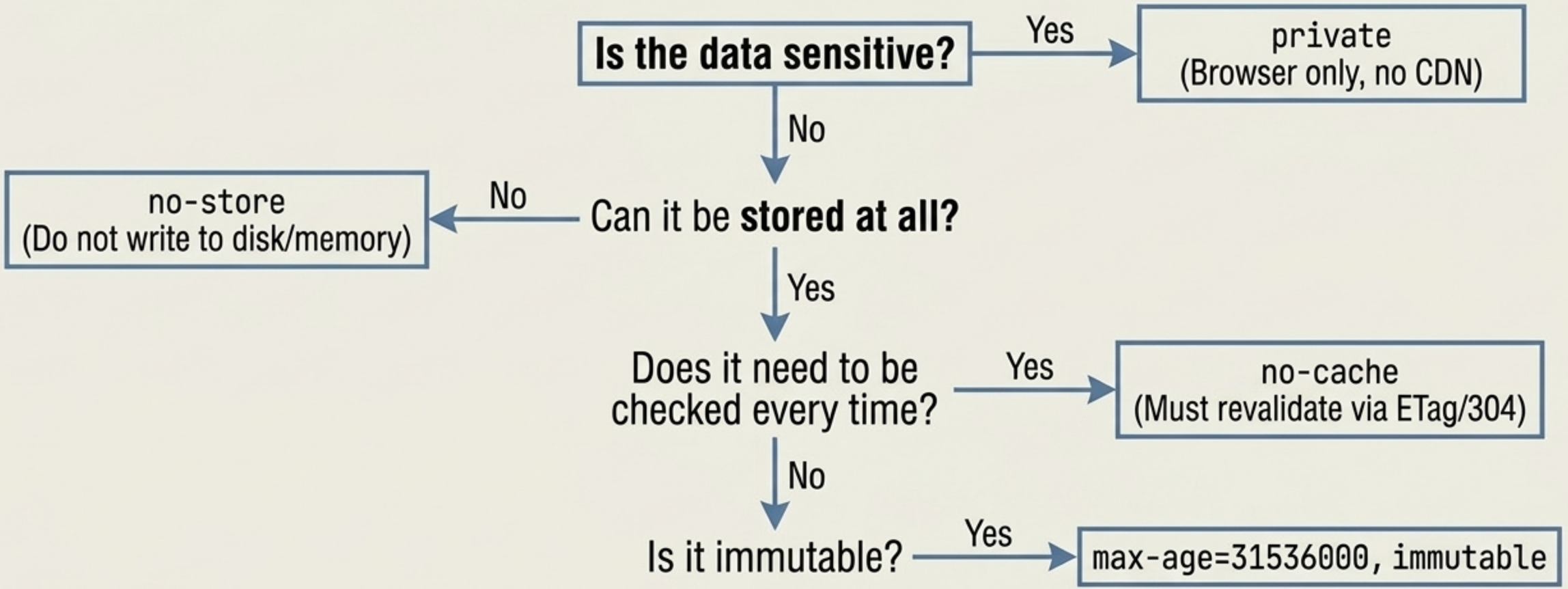


User gets immediate, albeit potentially stale, data. A background process silently refreshes the cache without blocking the user.

Gemini Notebook

HTTP Caching Semantics Demystified

Stop guessing Cache-Control headers. Follow the logic.



"no-cache" doesn't mean "don't cache"—it means "don't serve without checking first"

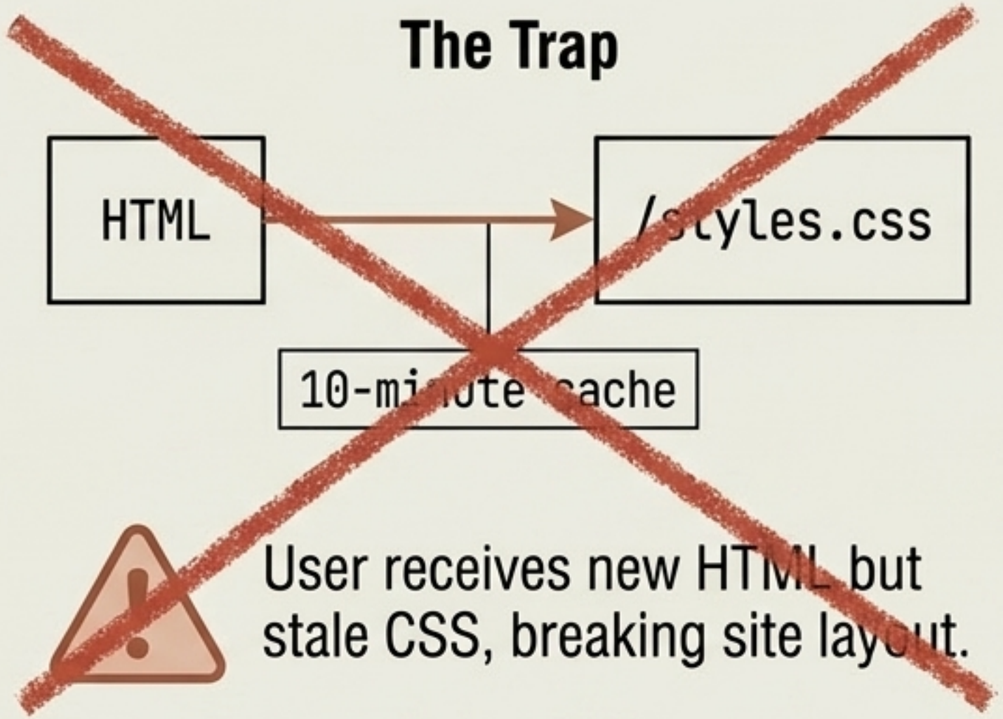
Gemini Notebook

The Immutable Hash Pattern

Why max-age on mutable content is a trap that causes racing dependencies.

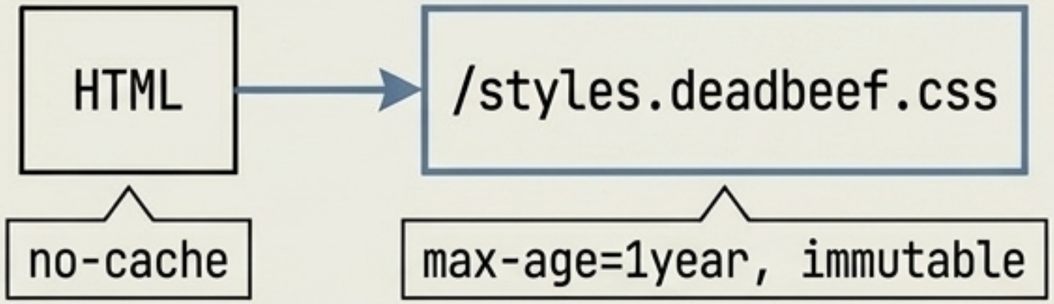
Asset Pipeline Dependency Graph

The Trap



User receives new HTML but stale CSS, breaking site layout.

The Solution



Correctly synchronizes HTML and CSS via content-based hashing.

Never modify a file in place. **Change** the content, **change the hash**, change the **URL**.

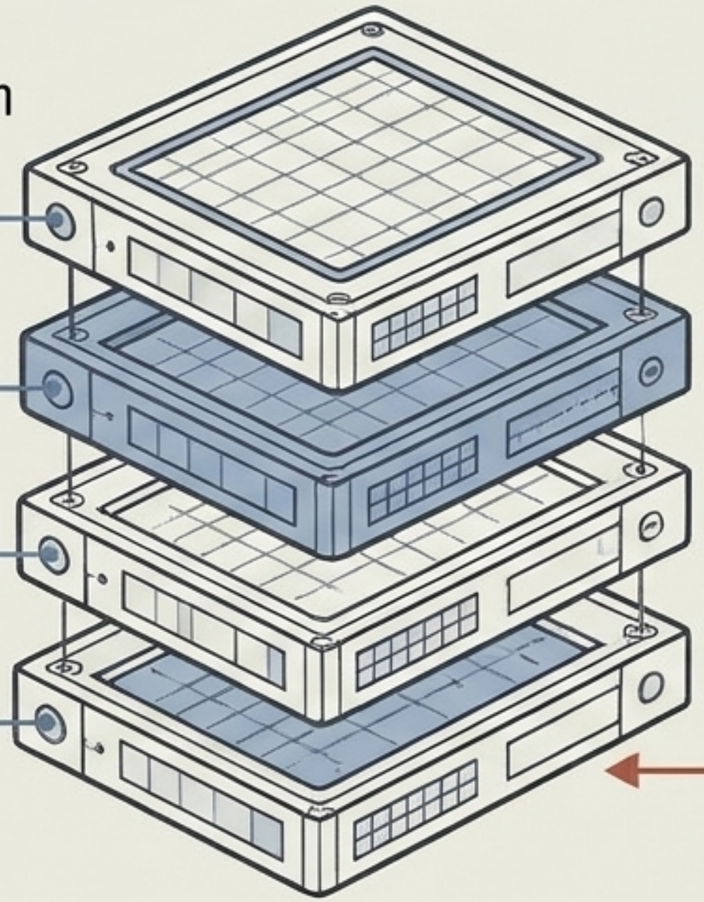
Gemini Notebook

Framework Traps & Cardinality Explosions

Modern frameworks obscure caching, leading to nested traps and memory exhaustion.

Next.js 4-layer caching structure

1. Request Memoization (React cache)
2. Data Cache (unstable_cache)
3. Full Route Cache
4. Router Cache



HTTP

Vary: User-Agent

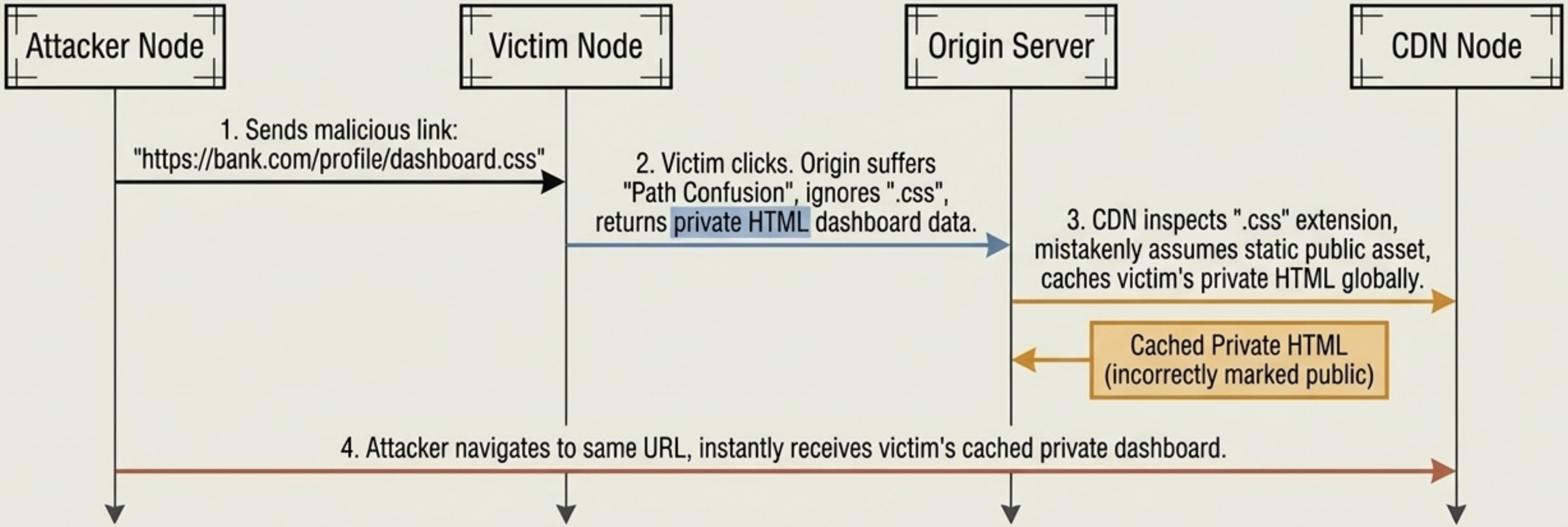
The Cardinality Trap: Varying by User-Agent creates a unique cache entry for every device, browser, and version on Earth, instantly filling and evicting the entire CDN.

Gemini Notebook

Web Cache Deception & Poisoning

How attackers force CDNs to cache and serve private victim data.

Security Attack Anatomy Sequence Diagram

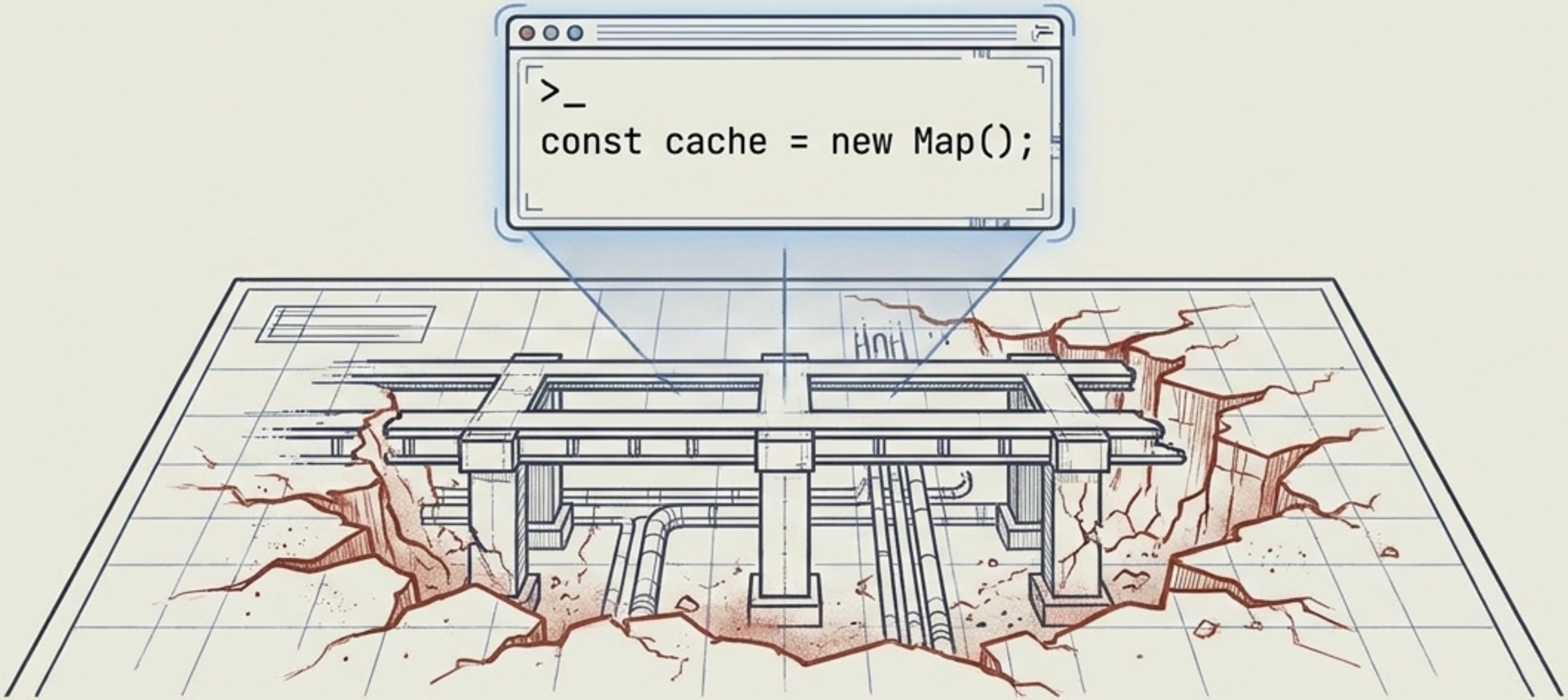


⚠ Always use “private” or “no-store” on authenticated routes, regardless of file extension.

Gemini Notebook

The Vibe-Coder Trap: "Just Add Caching"

LLMs autocomplete local optimization while causing global collapse. Default AI suggestions are almost always missing systemic safeguards.



⚠ Six production anti-patterns that survive code review but destroy systems at scale.

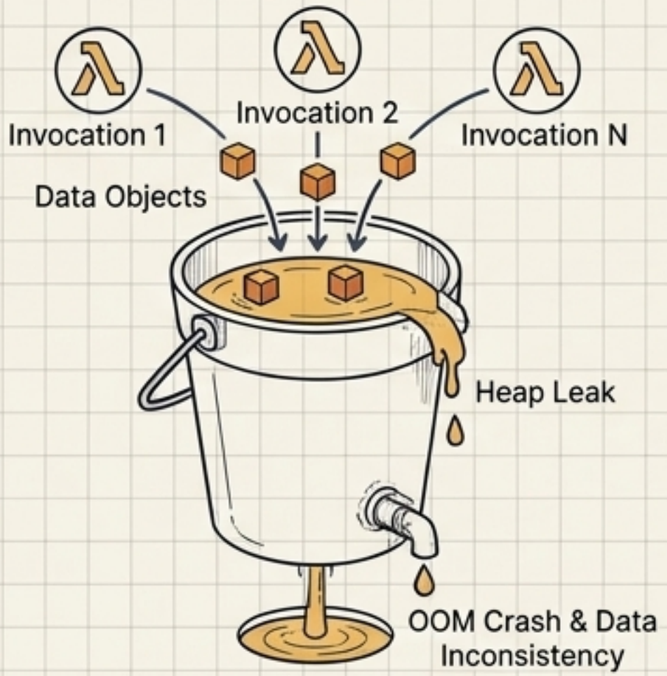
Gemini Notebook

Default Anti-Patterns: Memory & Consistency

How standard practices and AI suggestions silently erode system stability and data integrity.

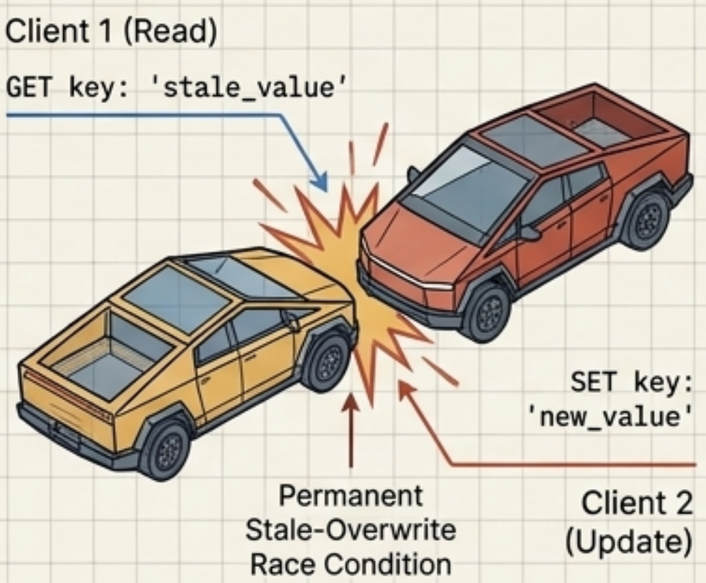
Serverless new Map()

Node.js heap leaks across warm lambda invocations, causing OOM crashes and pod split-brain data inconsistency.



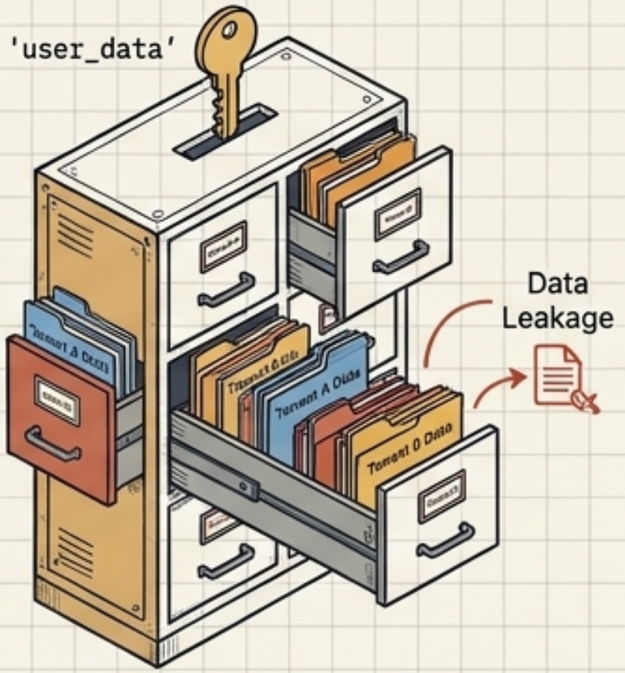
Blind SET-on-write

AI code uses `client.set(key, val)` instead of `client.del(key)` on updates, causing permanent stale-overwrite races.



Unscoped Multi-Tenant Keys

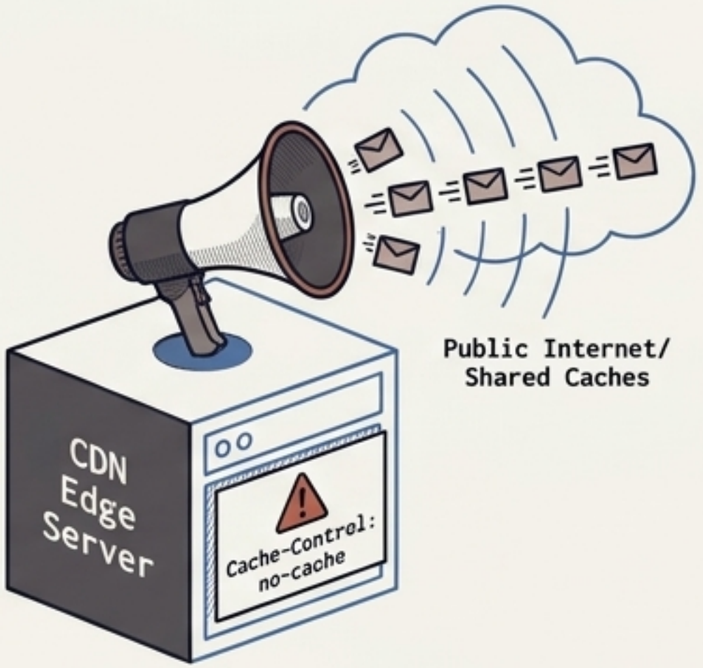
Using 'user_data' as a key instead of 'tenant_id:user_id:data', leading directly to cross-customer data leakage.



Gemini Notebook

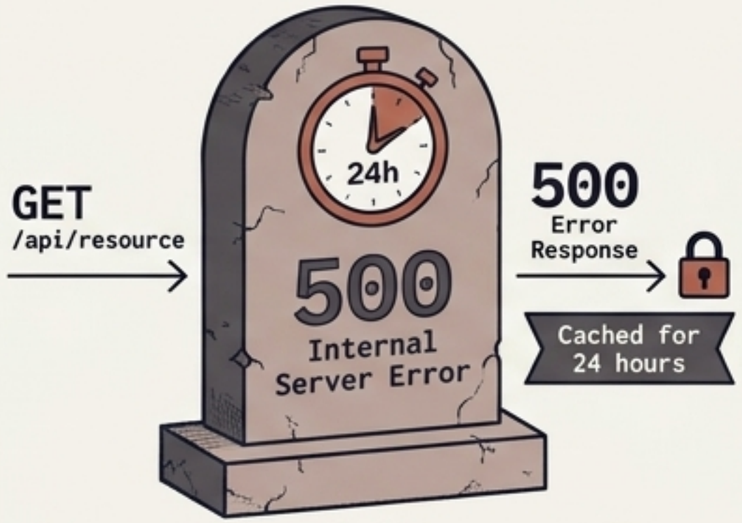
Default Anti-Patterns: HTTP & Reliability

no-cache on PII



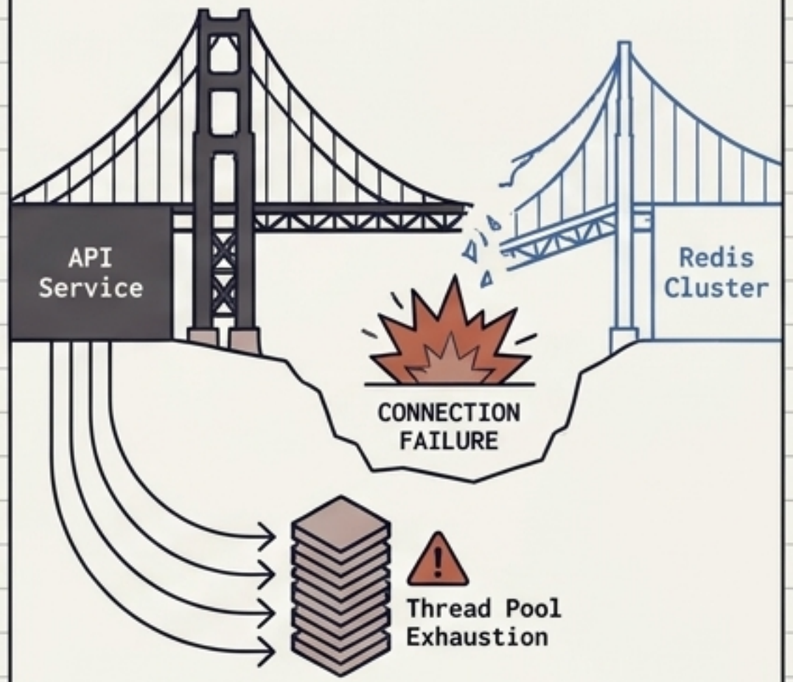
Using “no-cache” instead of “private/no-store”. The cache stores the PII and **leaks it to shared public caches**.

Caching 500s



A backend fails, the empty/error response is cached for 24 hours, **locking the entire system in a broken state**.

Hard Redis Dependency

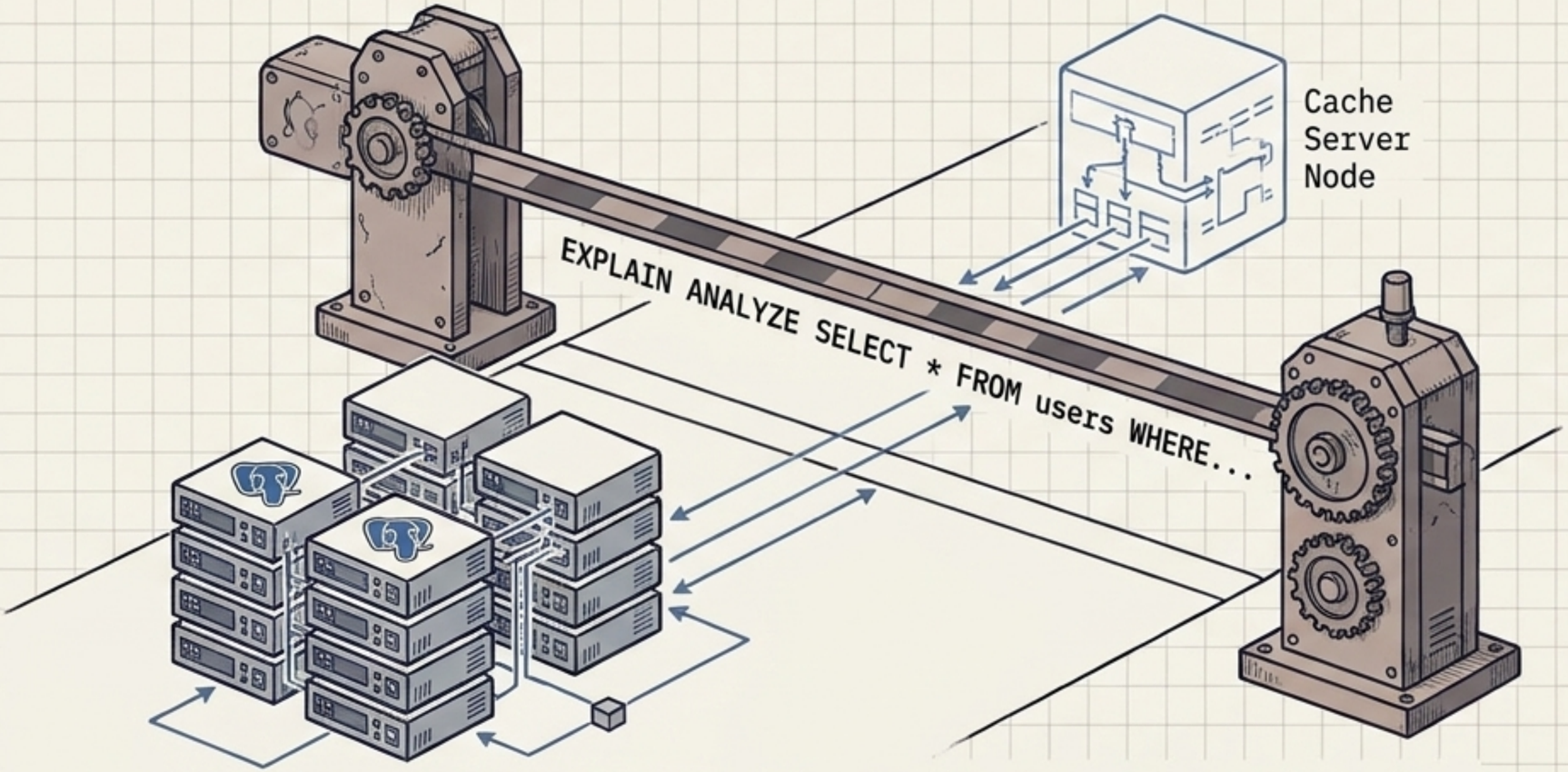


No circuit breaker wrapped around the Redis call. If Redis hangs, the **entire API thread pool exhausts in seconds**.

Gemini Notebook

The YAGNI Index Gate

You Aren't Gonna Need It. Before adding the massive complexity of distributed invalidation, prove the database actually needs it.



Gate Opening Conditions

- [ ] 1. Are you missing a composite index?
- [ ] 2. Do you have an N+1 ORM query?
- [ ] 3. Is your connection pool starved?

Only if all three are optimized, you may open the cache gate.

Gemini Notebook

The 5-Part Cache Contract (Prompt Playbook)

A copy-paste system prompt for `CLAUDE.md` or `AGENTS.md` to force LLMs to write production-grade cache layers.

```
When implementing any cache, you MUST provide:

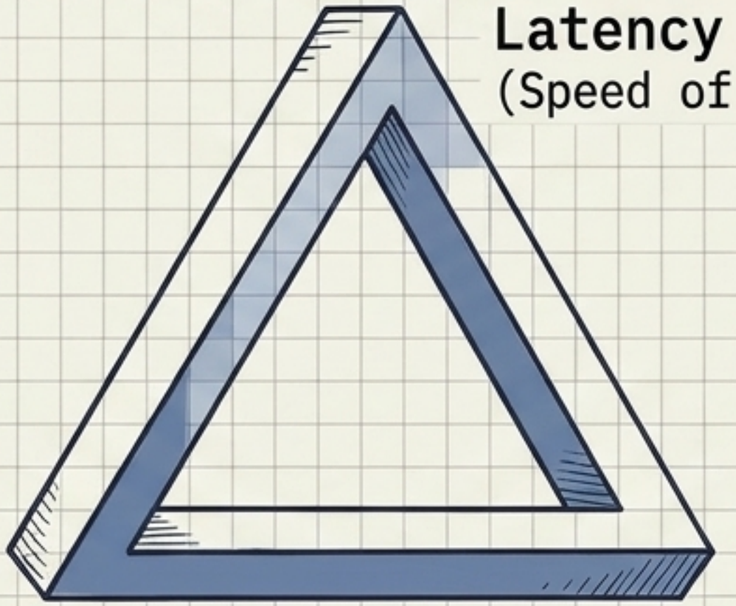
1) Eviction: Max memory limits & LRU/LFU policy.
2) Invalidation: Exact lifecycle of how the key is deleted on mutation.
3) Stampede Defense: Singleflight, jitter, or background refresh.
4) Fallback: Circuit breaker logic if the cache network fails.
5) Observability: Hit/Miss and latency metrics emitted.
```

Gemini Notebook

S08E09 Synthesis: The Triangle of Truth

Caching is not a speed feature; it is a complex distributed systems architecture that trades universal truth for localized availability.

Latency
(Speed of light constraints)



Scale
(Thundering Herds & Metastability)

Consistency
(Stale Races & Invalidation)

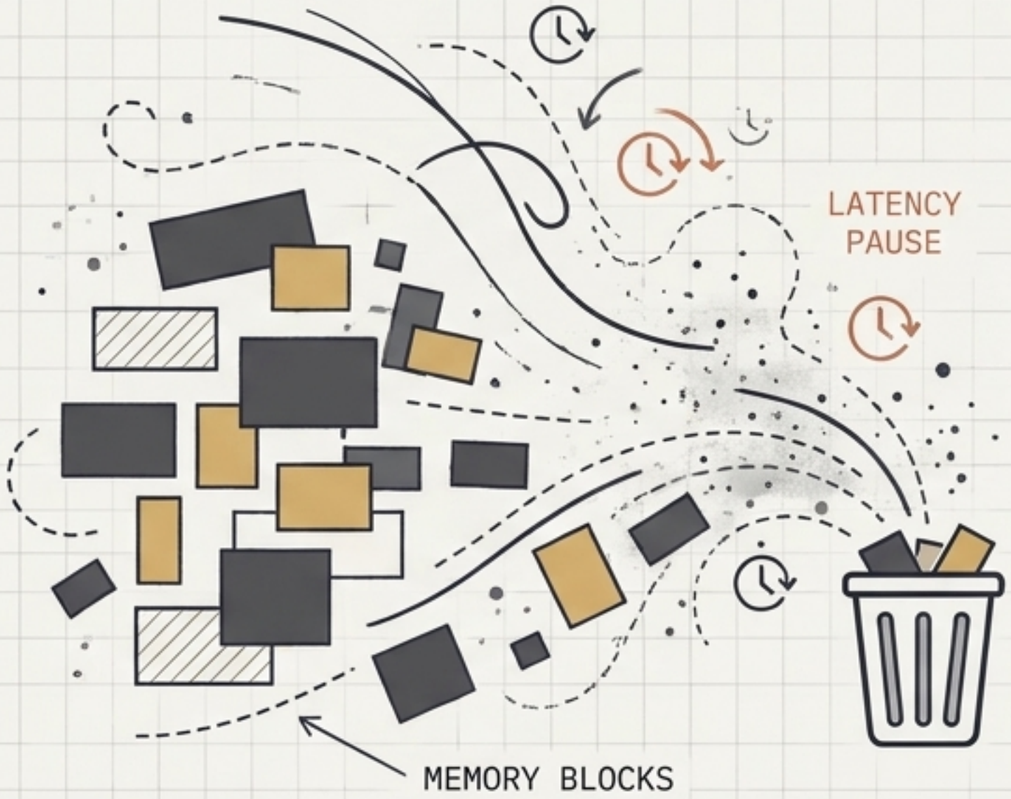
You can rebel against the physics of latency by copying data, but you must pay the tax of invalidation traffic and metastability to survive.

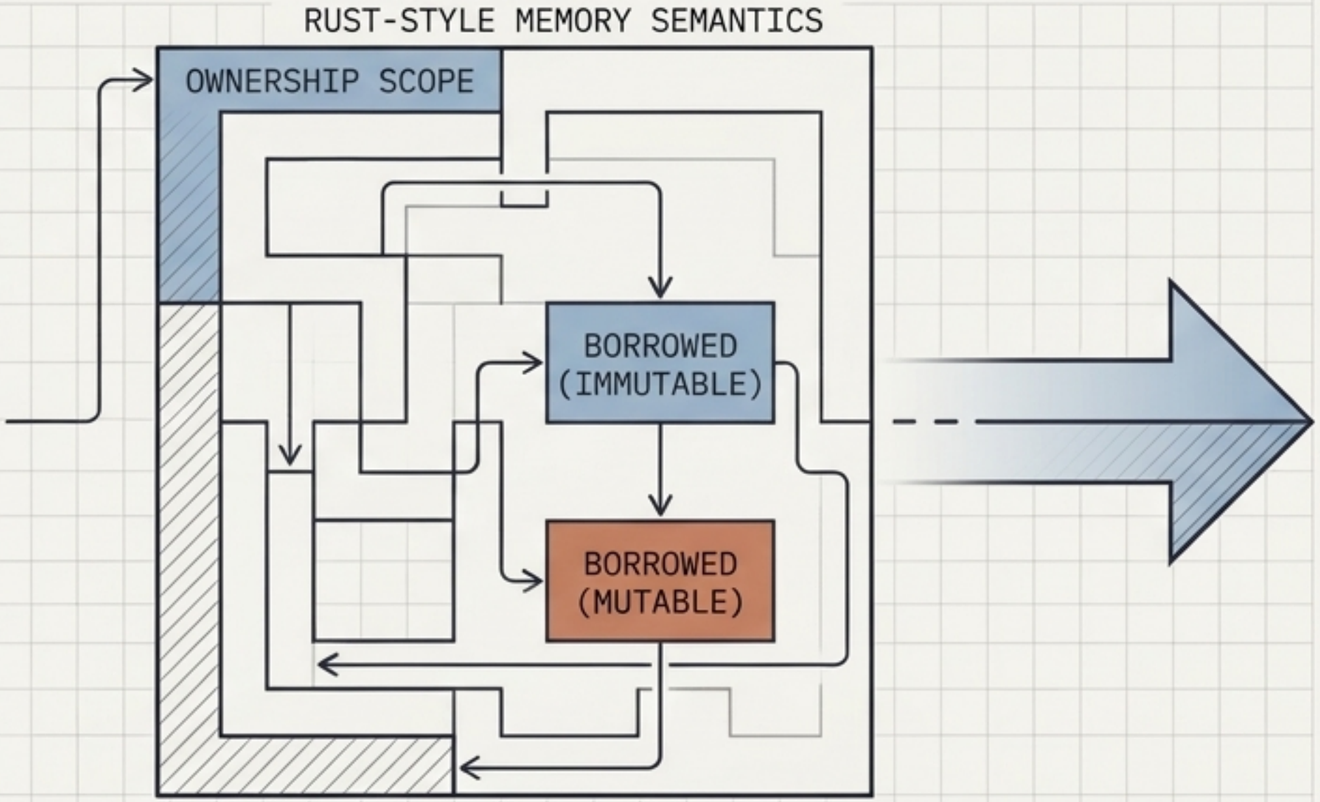
Gemini Notebook

Preview — Arc 4: Memory, State & Theoretical Limits

S08E10: Garbage Collection vs. Ownership

Unpredictable pauses, memory safety, and the limits of automated memory management.

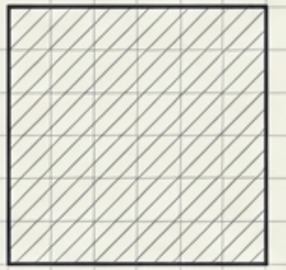




Gemini Notebook

System References & Bibliography

- HotOS ‘21: Metastable Failures in Distributed Systems.
- VLDB ‘15: Optimal Probabilistic Cache Stampede Prevention (XFetch).
- NSDI ‘13: Scaling Memcache at Facebook (Lease Tokens).
- OSDI ‘20: CacheLib - General-Purpose Caching Engine.
- RFC 9111 / 5861 / 9520: HTTP Caching Semantics & Extensions.
- Brooker (2021): Caches, Modes, and Unstable Systems.



SCAN FOR SOURCE PAPERS / DOWNLOAD LINK

Gemini Notebook