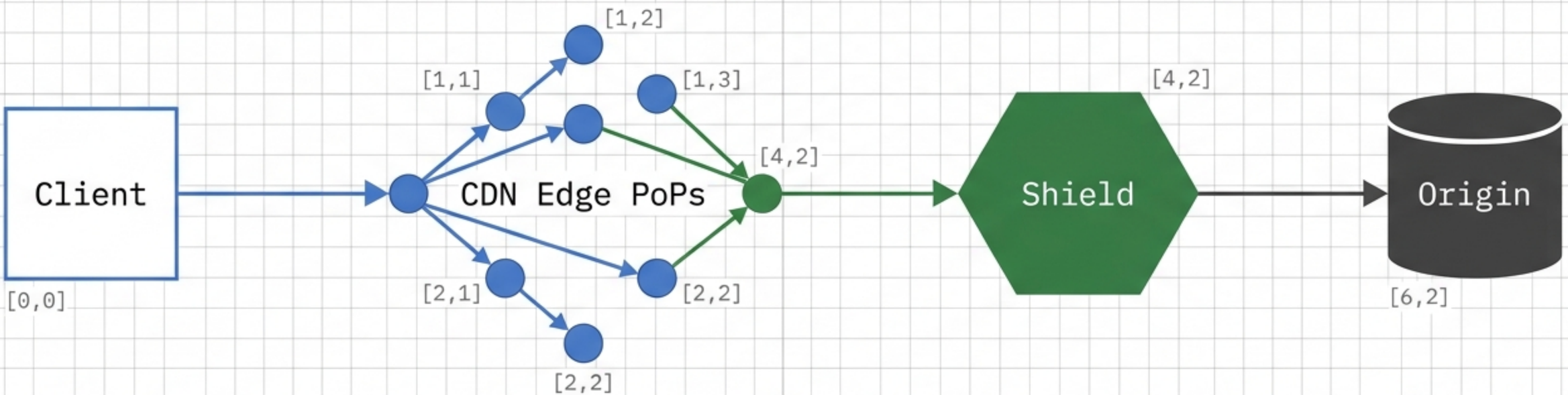
Web Architecture — S02E02

CDN Internals, RFC 9111 HTTP Caching, Origin Shielding & Cache Invalidation

An Engineering Blueprint for Backend, Frontend & Platform Teams.



Gemini Notebook

The Two Spheres of Caching

Private Caches



Dedicated to a **single user** (browser memory/disk).

Safe for personalized content.

Directive: `Cache-Control: private`

Protects the **network path** between the individual client and the edge.



Shared Caches



Positioned **between client and server** (CDN Edge PoPs, Reverse Proxies).

Shared across all users. High risk for data leakage if misconfigured.

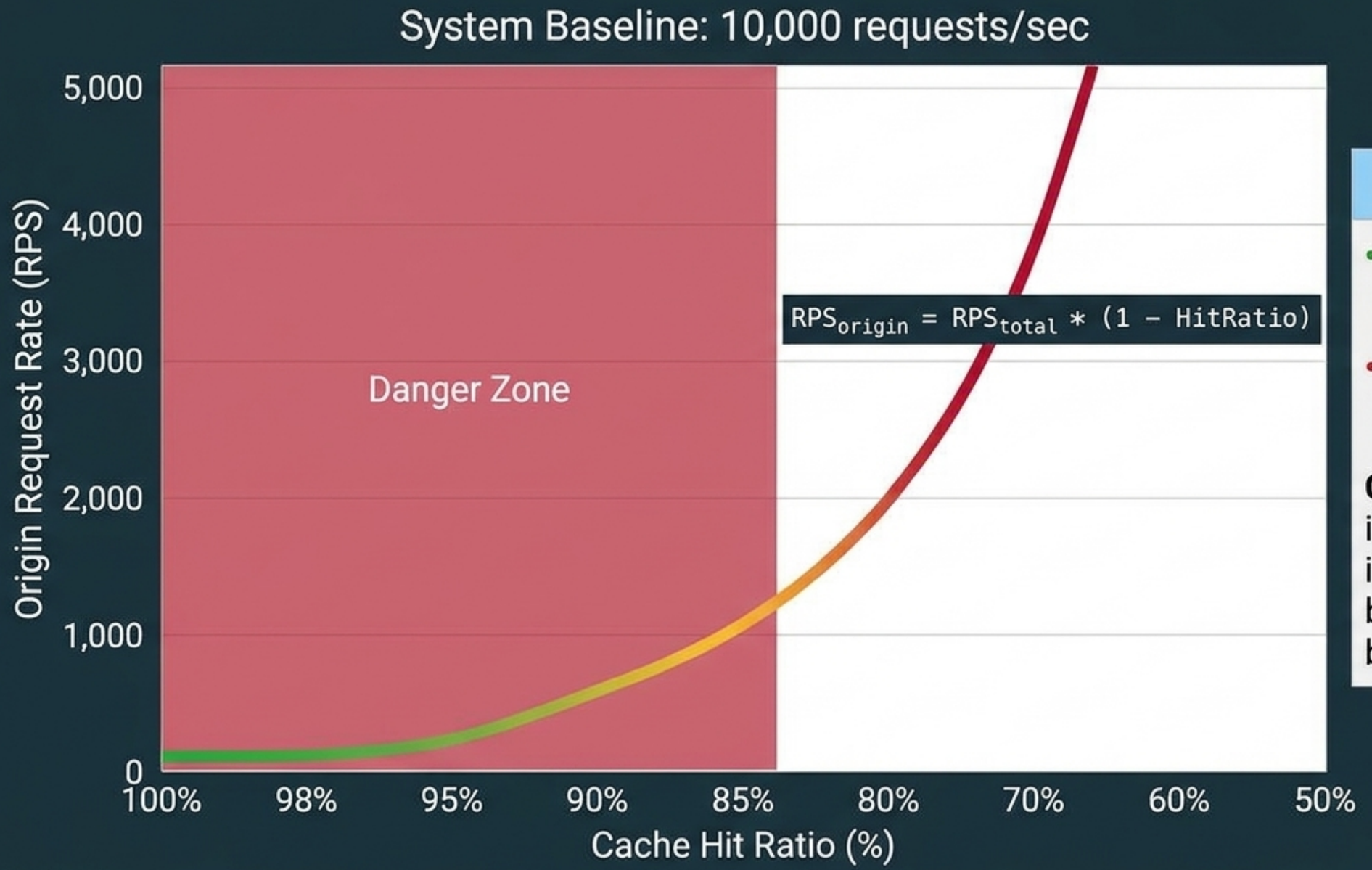
Directive: `Cache-Control: public`

Protects the **origin server** by offloading repeated identical requests.



Gemini Notebook

The Danger of Cache Addiction



Impact Analysis

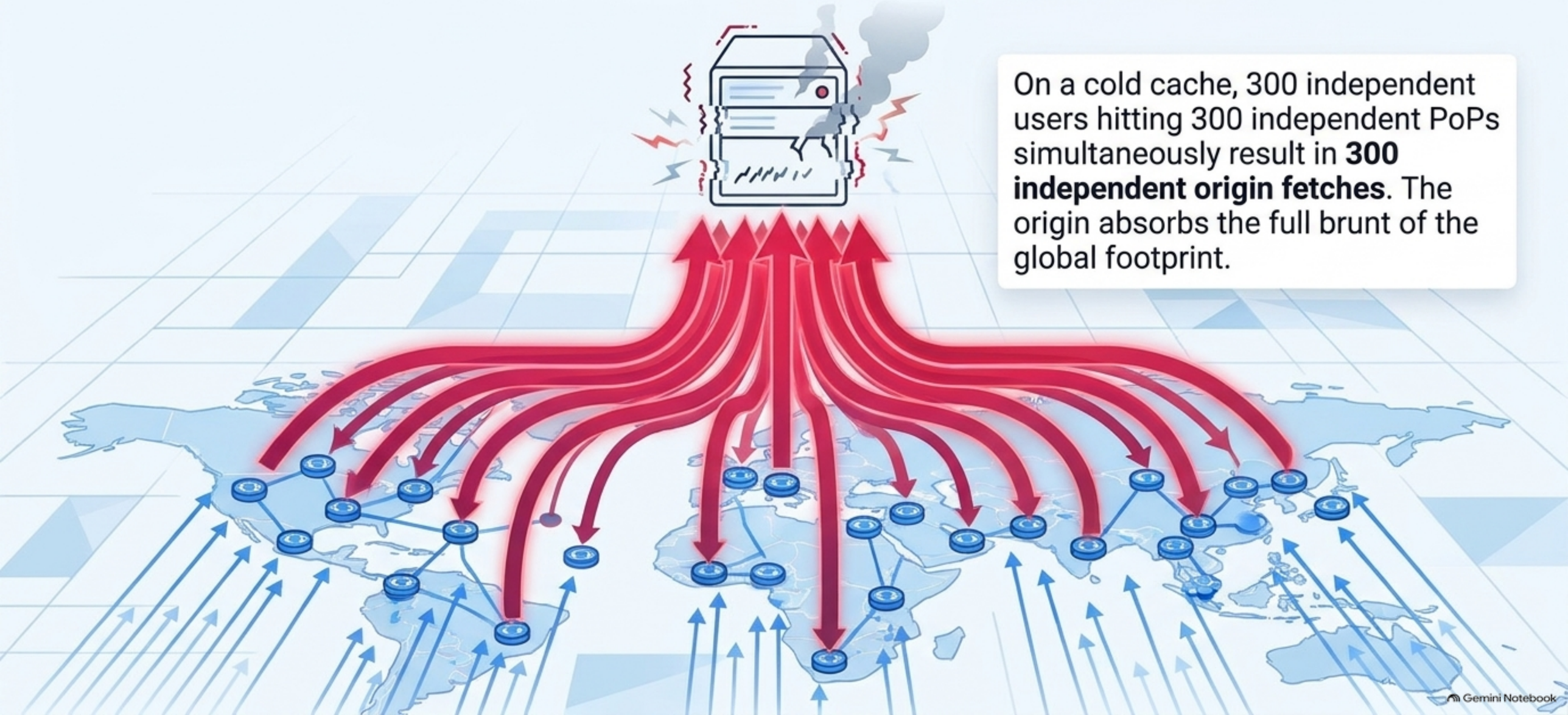
- 98% Hit Rate = 200 Origin RPS (System Stable)
- 80% Hit Rate = 2,000 Origin RPS (System Meltdown)

Conclusion: A mere 18% drop in hit rate causes a 10x surge in origin load, instantly browning out undersized backends.

Gemini Notebook

The Cold Miss Topology Problem

A flat CDN topology does not protect against global cold misses.

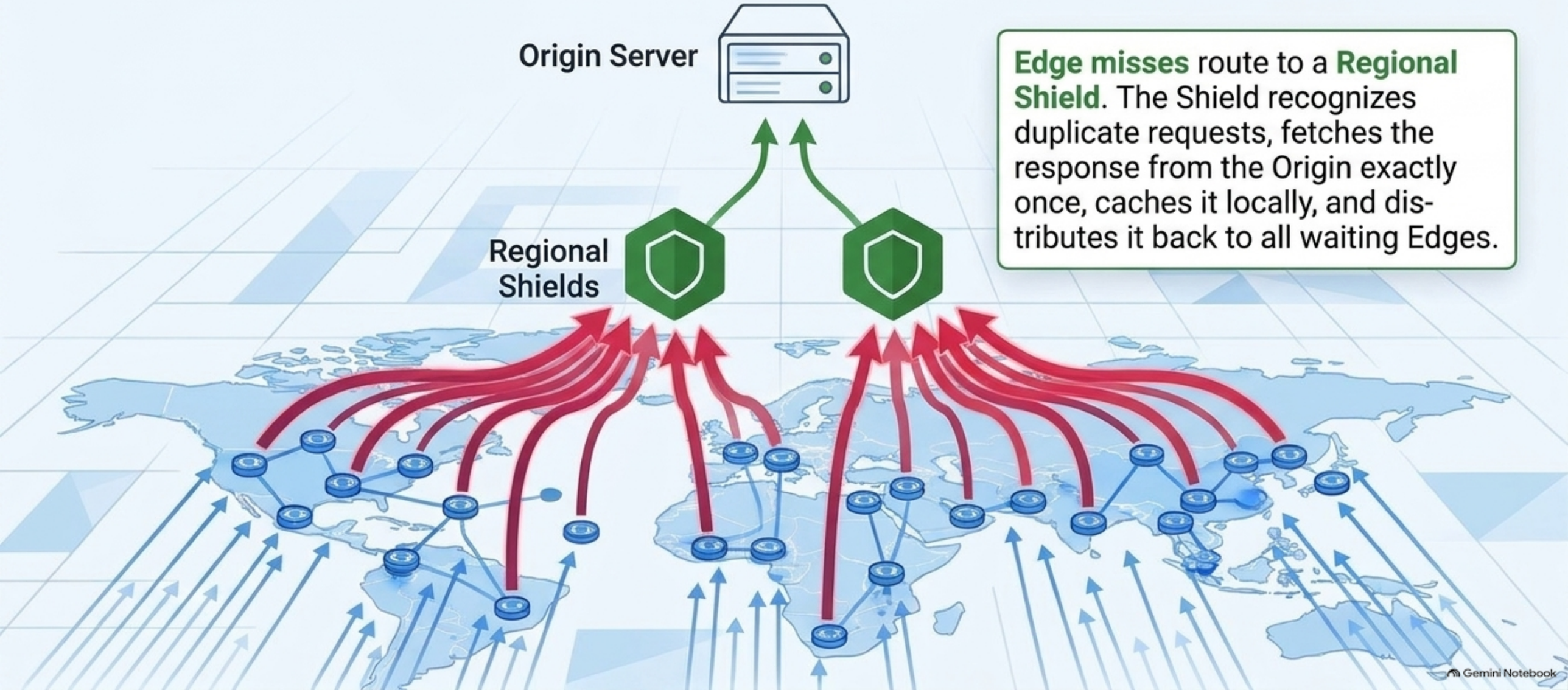


On a cold cache, 300 independent users hitting 300 independent PoPs simultaneously result in **300 independent origin fetches**. The origin absorbs the full brunt of the global footprint.

Gemini Notebook

Tiered Caching & Origin Shielding

Protecting the origin from edge-scale traffic geometry.

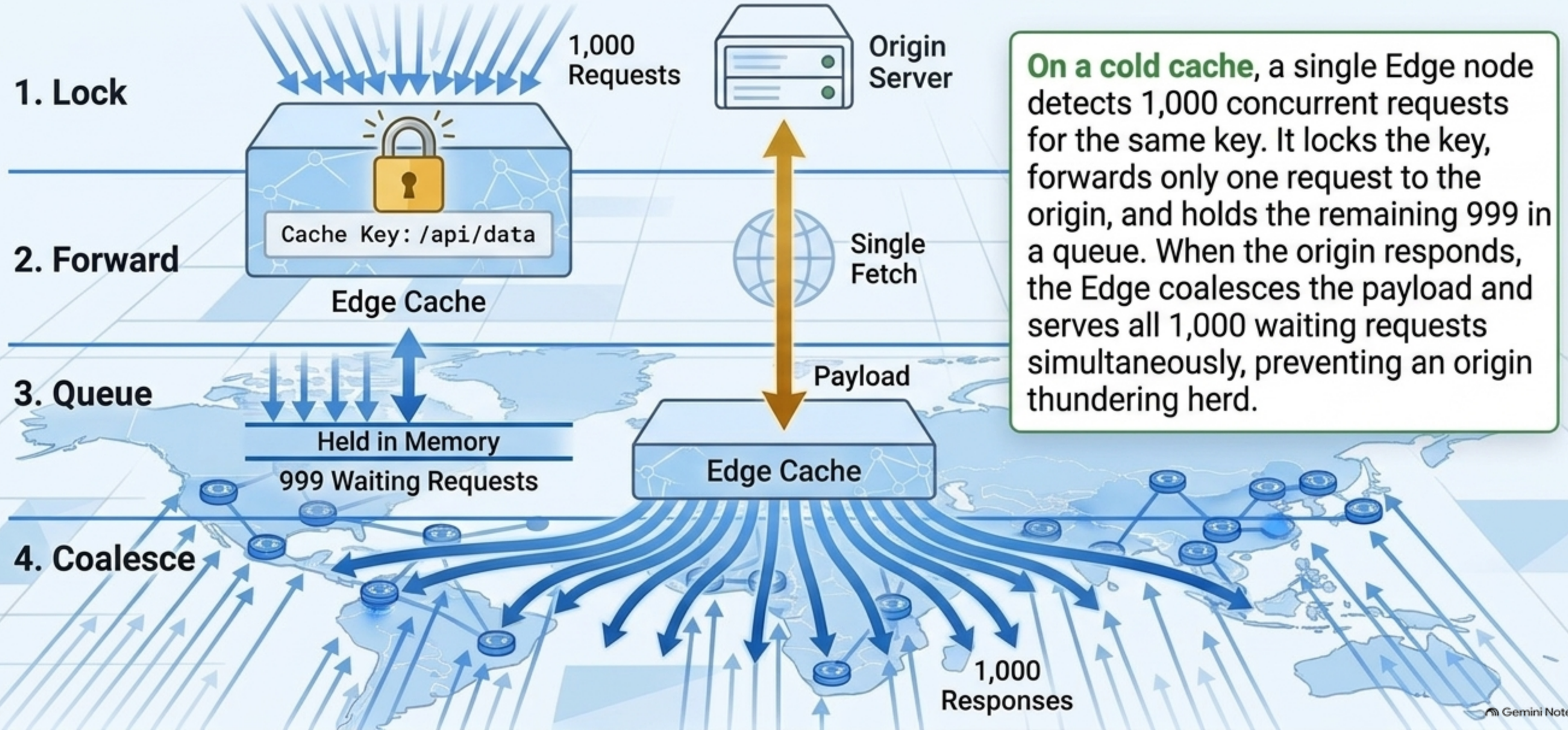


Edge misses route to a **Regional Shield**. The Shield recognizes duplicate requests, fetches the response from the Origin exactly once, caches it locally, and distributes it back to all waiting Edges.

Gemini Notebook

Request Collapsing

Thundering Herd Mitigation at a single Edge node.

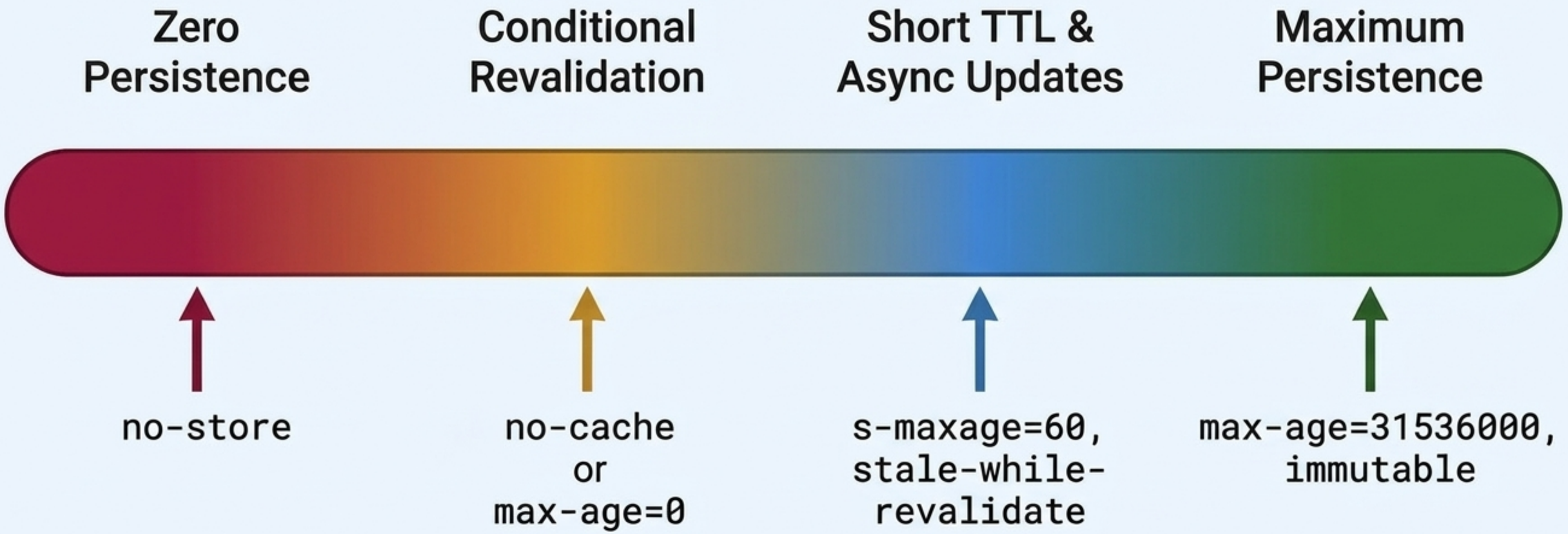


On a cold cache, a single Edge node detects 1,000 concurrent requests for the same key. It locks the key, forwards only one request to the origin, and holds the remaining 999 in a queue. When the origin responds, the Edge coalesces the payload and serves all 1,000 waiting requests simultaneously, preventing an origin thundering herd.

Gemini Notebook

The Cache-Control Spectrum

Every HTTP response must actively choose its position.



Heuristic caching (guessing based on Last-Modified) only occurs when explicit controls are missing.

Gemini Notebook

Persistence & Sharing Directives

Decoding exact RFC 9111 semantics.

`max-age=N`

Client/proxy freshness lifetime in seconds.

Example: `max-age=604800` (1 week)

`s-maxage=N`

Shared cache freshness lifetime.

Explicitly overrides `max-age` for CDNs and reverse proxies.

`private`

Restricts storage strictly to the end-user's private browser cache.

Shared caches MUST NOT store.

`public`

Explicitly permits a shared cache to store the response, even if the request contained an `Authorization` header.

Gemini Notebook

Revalidation & Opt-Out Directives

`no-store`		True zero-caching. Do not write to disk/memory anywhere. Use for highly sensitive PII or transactional data.
`no-cache`		A MISNOMER. Means: You may store this, but you MUST revalidate it with the origin (via conditional request) before serving.
`must-revalidate`		Once max-age expires, the cache MUST NOT serve the stale asset under any circumstances without origin validation.

Anti-Pattern Warning

~~`Cache-Control: no-store, no-cache, max-age=0, must-revalidate`~~

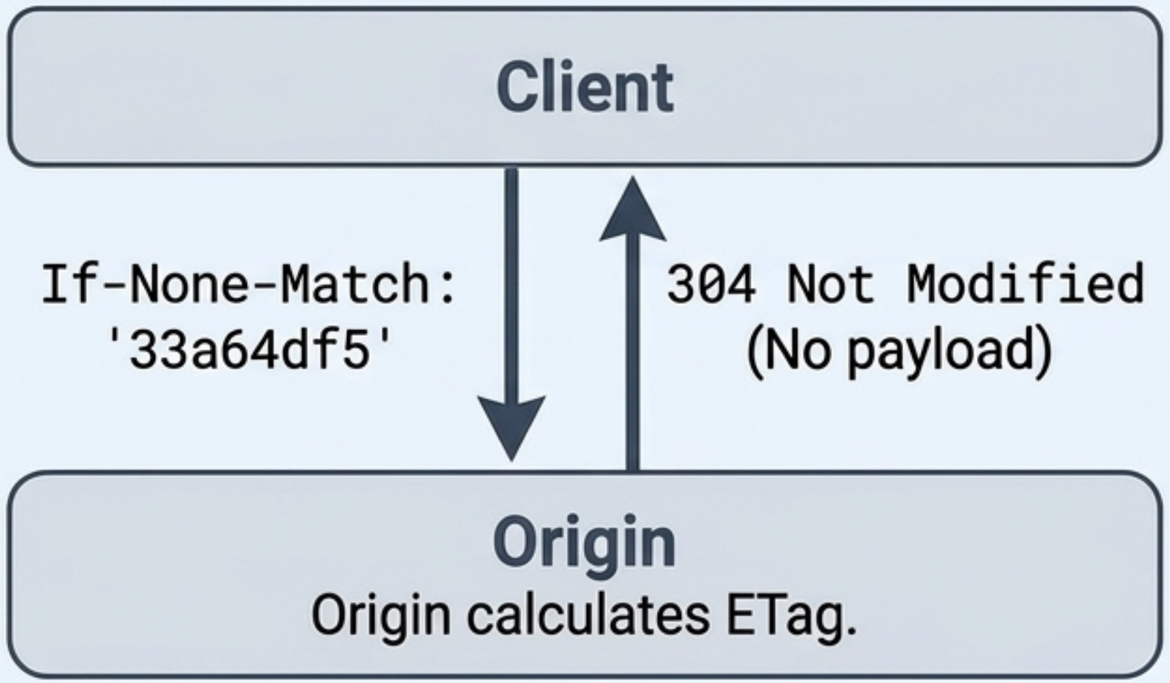
Modern browsers and CDNs only require no-store for absolute zero persistence.

Gemini Notebook

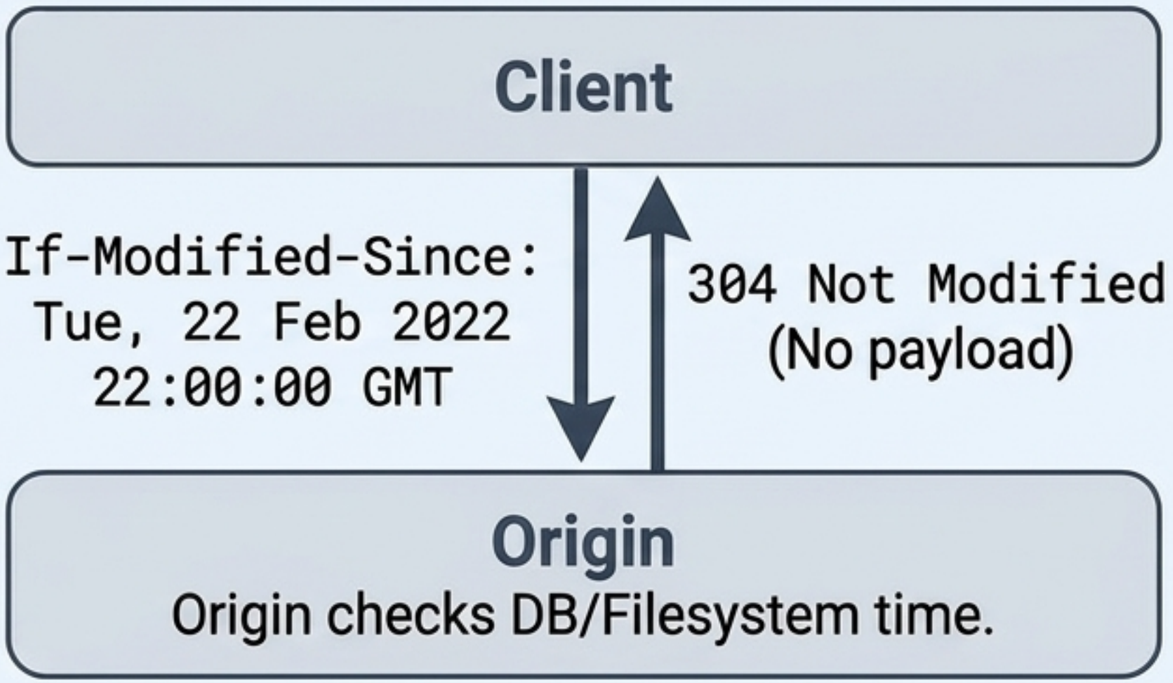
Conditional Requests

Saving bandwidth while still incurring latency.

Strong Validator (ETag)



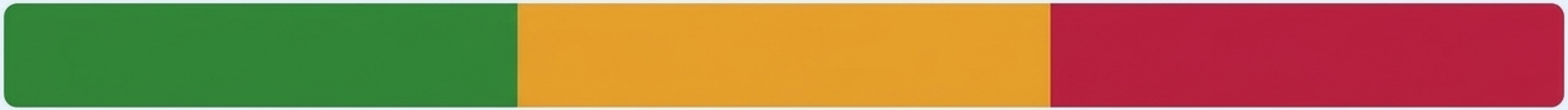
Weak Validator (Last-Modified)



Key Insight: Conditional requests eliminate payload transfer, but still incur a full network round-trip time (RTT) to the origin server.

Gemini Notebook

The Lifespan of a Cached Object



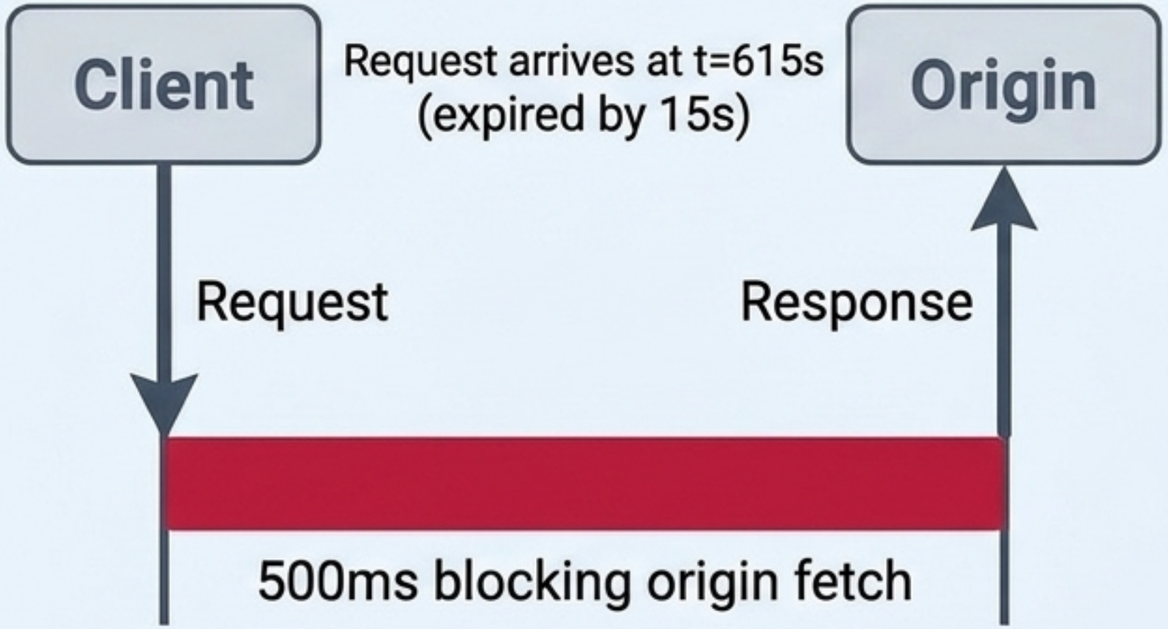
Phase 1: Fresh	Phase 2: Stale but Usable	Phase 3: Truly Stale
Age <= max-age	**max-age < Age <= (max-age + stale-window)**	**Age > (max-age + stale-window)**
Served instantly from edge cache (0ms latency).	The RFC 5861 async window. Technically expired, but edge is permitted to serve while validating.	Asset cannot be served. Next request triggers a blocking, synchronous origin fetch.

Gemini Notebook

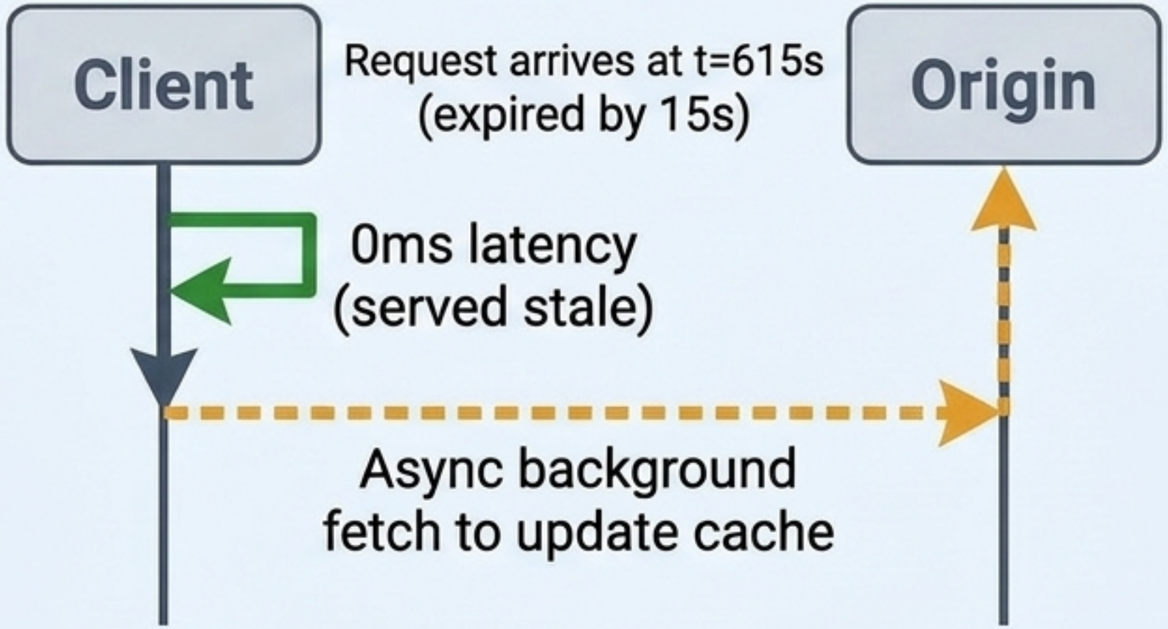
Eliminating Tail Latency

Cache-Control: max-age=600, stale-while-revalidate=30

Without stale-while-revalidate



With stale-while-revalidate

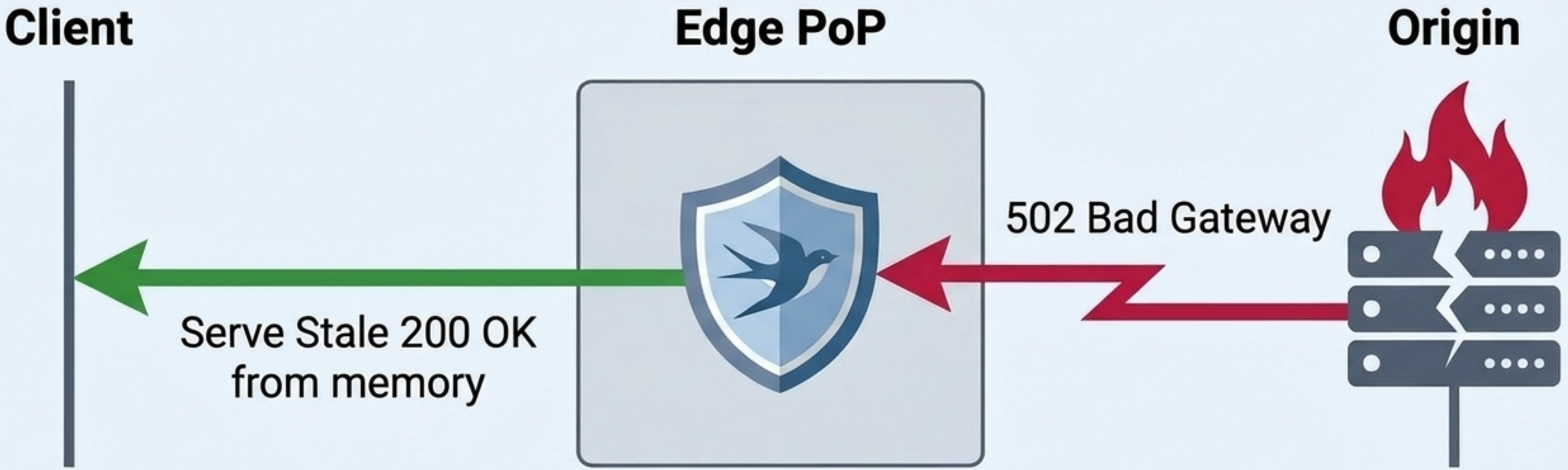


SWR completely masks backend database latency from the end-user.

Gemini Notebook

High-Availability Fallback

Cache-Control: max-age=600, stale-if-error=1200

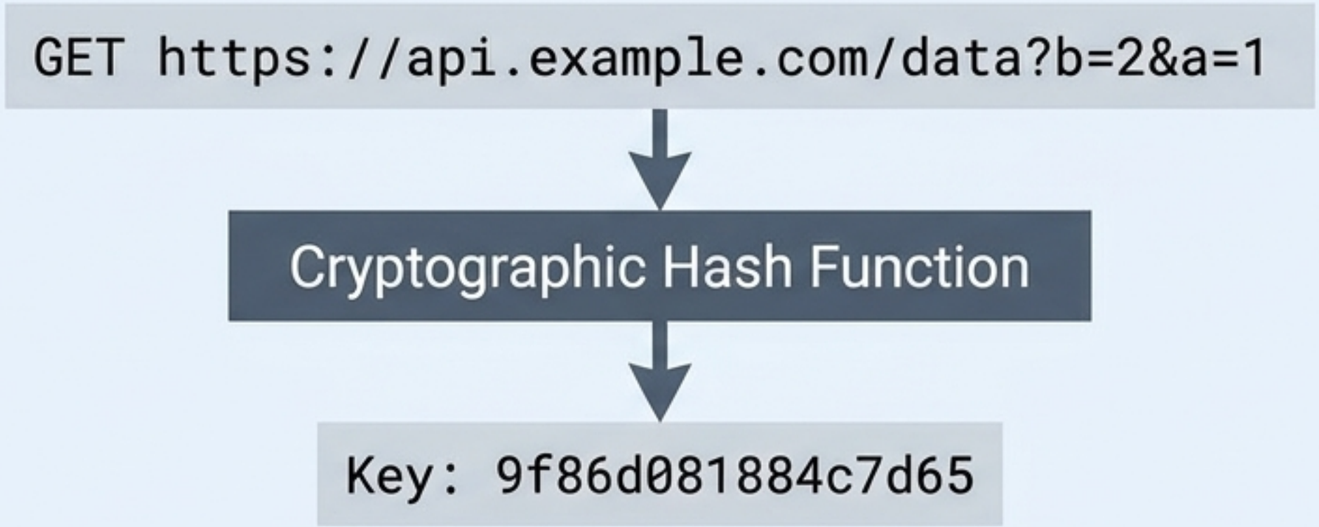


Preserves frontend uptime and user experience while backend engineering recovers the database.

Gemini Notebook

Anatomy of a Composite Cache Key

Cache Key = Hash(Method + Scheme + Host + Path + [Sorted Query Params] + [Vary Headers])



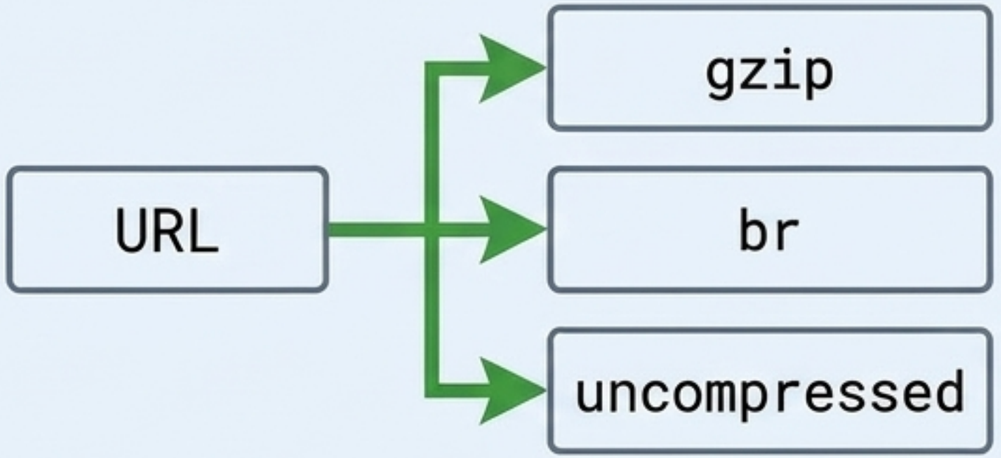
The cache key is an exact, unforgiving hash. Microscopic variations create completely distinct, empty objects (Cold Misses).

Gemini Notebook

The Vary Header & Cache Fragmentation

Efficient

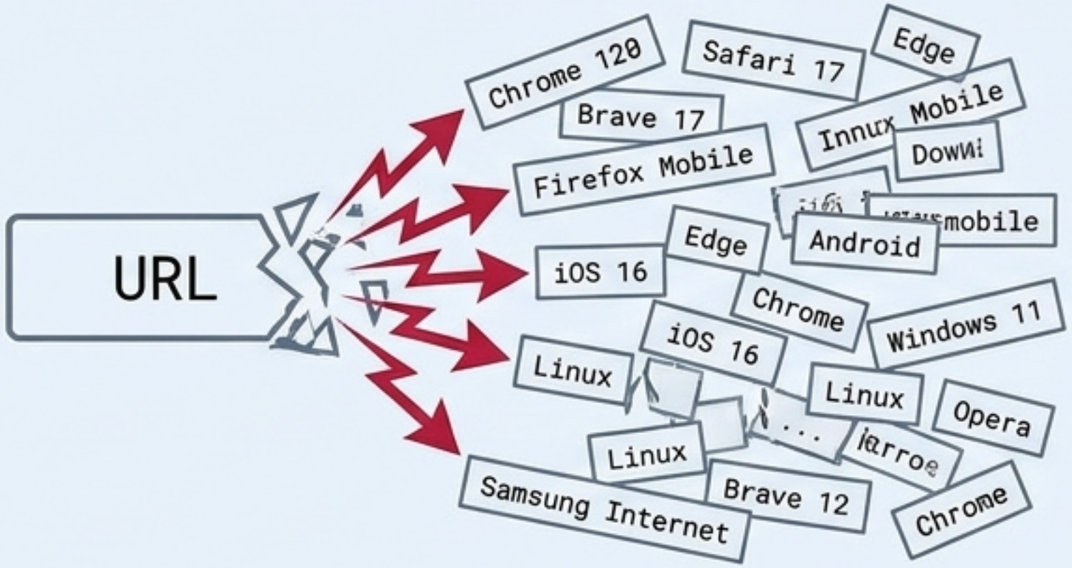
`Vary: Accept-Encoding`



Highly efficient.
Maintains high hit ratio.

Destructive

`Vary: User-Agent`



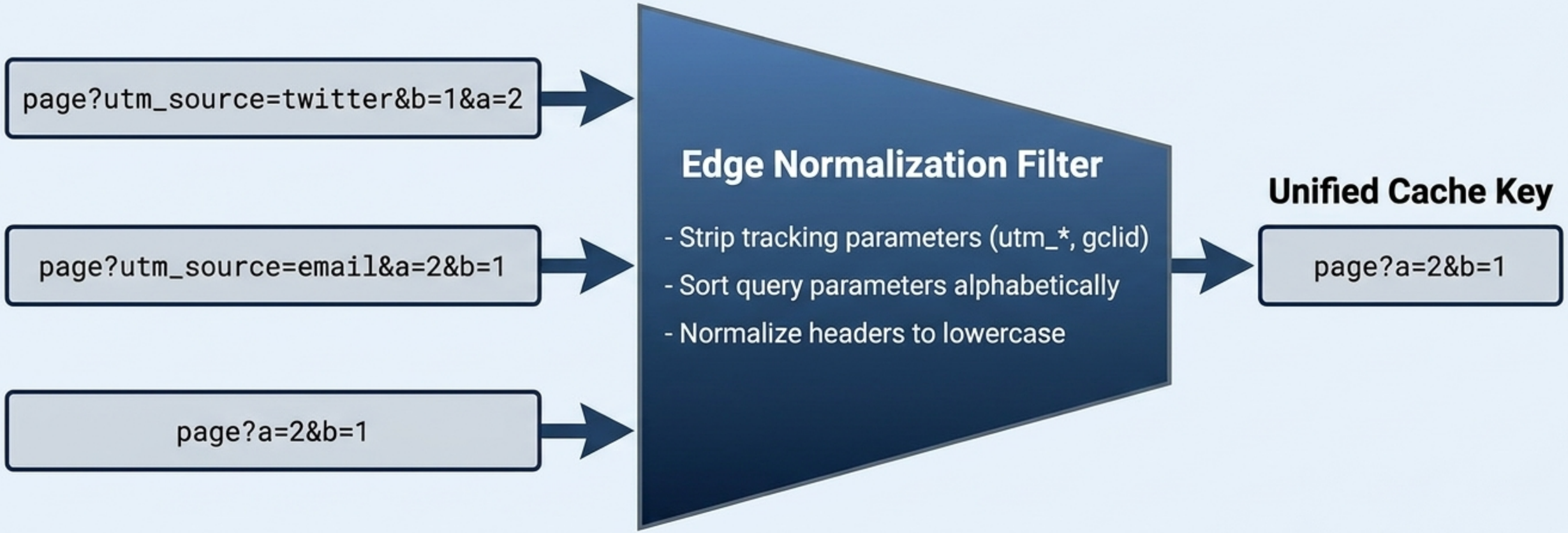
Total fragmentation.
Destroys the hit ratio.

Use feature detection in frontend code, not User-Agent sniffing on the backend.

Gemini Notebook

Cache Key Normalization Strategies

Edge compute rules rewrite requests before the hash function.

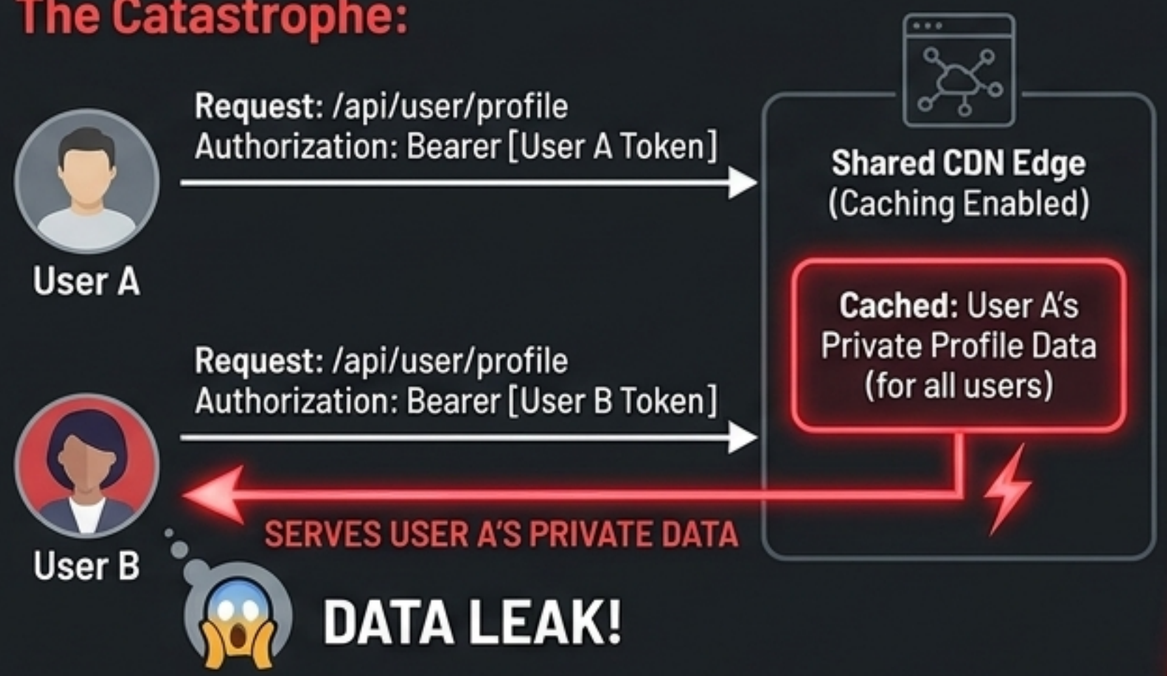


Gemini Notebook

⚠ Fatal Security Bugs: Authenticated Endpoints

Scenario: A JSON API returning personal data at `/api/user/profile` with an `Authorization: Bearer` token.

The Catastrophe:



The Fix:

Authenticated routes **MUST** explicitly return `Cache-Control: private` (for browser storage) or `no-store` (for absolute zero-persistence).

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: private
...
```

Gemini Notebook

Hard Deletion vs. Soft Invalidation

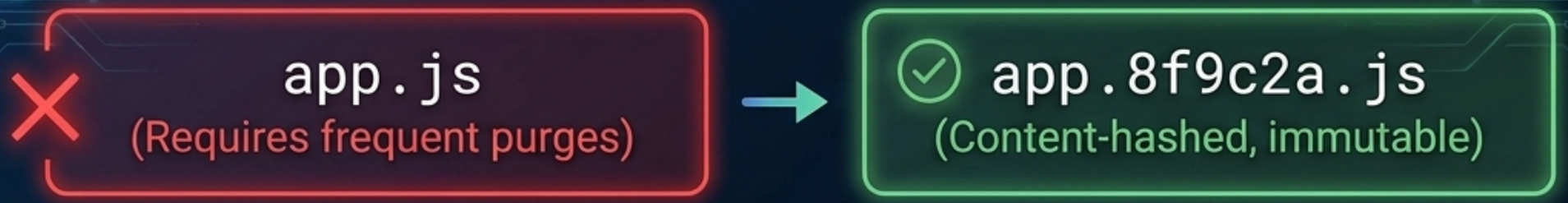
Updating cached data at scale.

	Mechanism	Consequence
Hard Delete (Purge)	Immediately wipes the asset from CDN disks.	The very next request triggers an immediate, blocking origin fetch. At scale, guarantees a **Thundering Herd** load spike.
Soft Invalidation (Mark as Stale)	Modifies object metadata, marking Age as expired, but leaves payload on disk.	Next request triggers a **background stale-while-revalidate** fetch. Zero user latency, smooth and collapsed origin load.

Gemini Notebook

Strategy 1: Immutable Cache Busting

The optimal strategy for static bundles.




```
Cache-Control: public, max-age=31536000, immutable
```

Mechanism

- Never purge the CDN. Set a 1-year TTL.
- When code changes, the build process generates a new file hash, updating the HTML reference.

Benefit

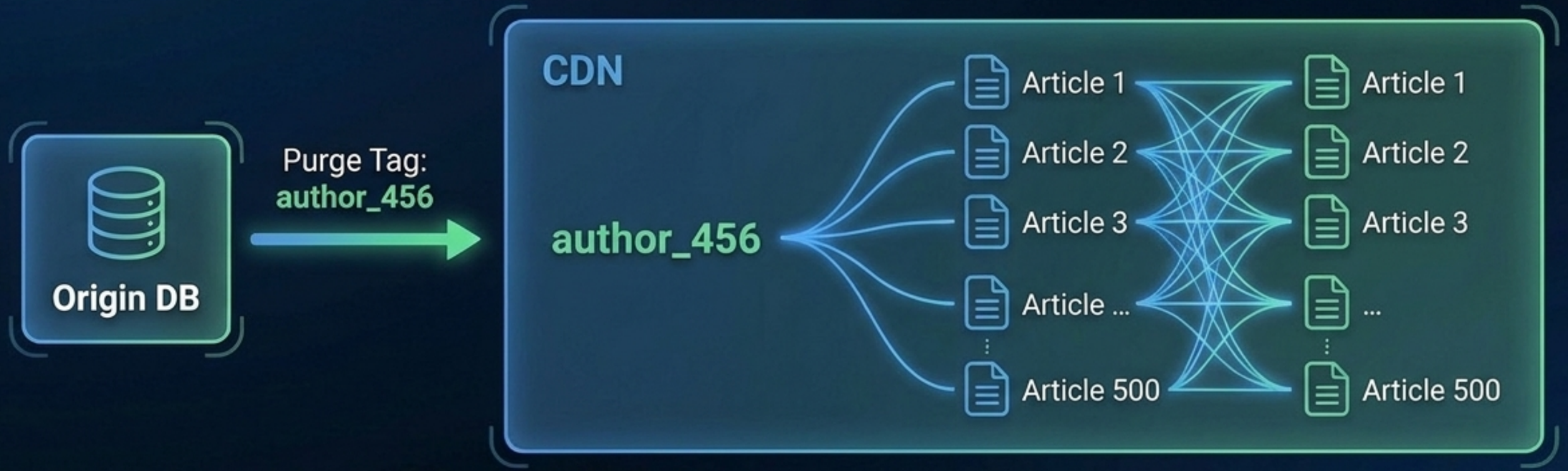
- Guarantees absolute version consistency globally.
- Zero CDN purge API calls required.

Gemini Notebook

Strategy 2: Surrogate-Key / Cache-Tags

Soft invalidation for dynamic data (APIs, SSR).

```
Surrogate-Key: article_123 author_456
```



Instead of purging by URL, purge by relational tag.
One API call instantly invalidates all 500 articles across all global PoPs.

Gemini Notebook

Blueprint: Static Assets & Media

Target: JS, CSS, Web Fonts, Images

```
Cache-Control: public, max-age=31536000, immutable
```

Rationale

- Maximize edge caching to the absolute limit.
- Rely strictly on filename hashing (e.g., logo.v2.png) for content updates.
- Requirement: The origin must ensure the file contents for a given URL never change once published.

Gemini Notebook

Blueprint: SSR HTML & Public APIs

Target: E-commerce product pages, news articles, public JSON

```
Cache-Control: public, max-age=0, s-maxage=60,
               stale-while-revalidate=86400
```

Rationale

- **max-age=0**: Ensures the browser always asks the CDN.
- **s-maxage=60**: Caches at the CDN edge for 1 minute.
- **stale-while-revalidate**: Ensures users never see latency while the CDN fetches updates asynchronously in the background.

Gemini Notebook

Blueprint: Authenticated / Private APIs

Target: Dashboards, checkout flows, user profiles

```
Strict Zero-Persistence:
Cache-Control: no-store

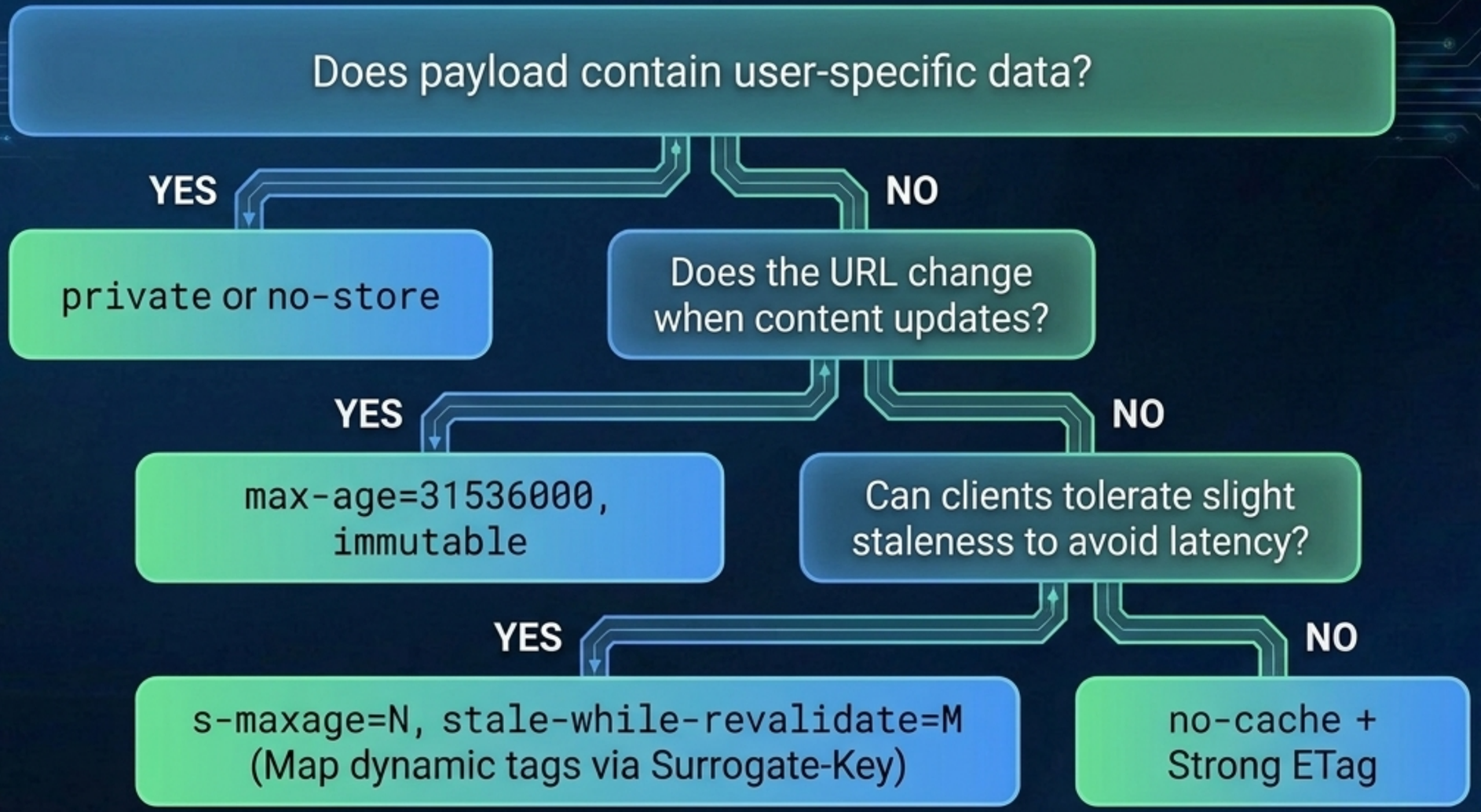
Optimized Private (if browser conditional requests are safe):
Cache-Control: private, no-cache
```

Rationale

- Absolute protection against cross-user data leakage on shared edge infrastructure.

Gemini Notebook

Synthesis: The Caching Decision Tree



An Architectural Cheat Sheet for HTTP Caching.

Gemini Notebook