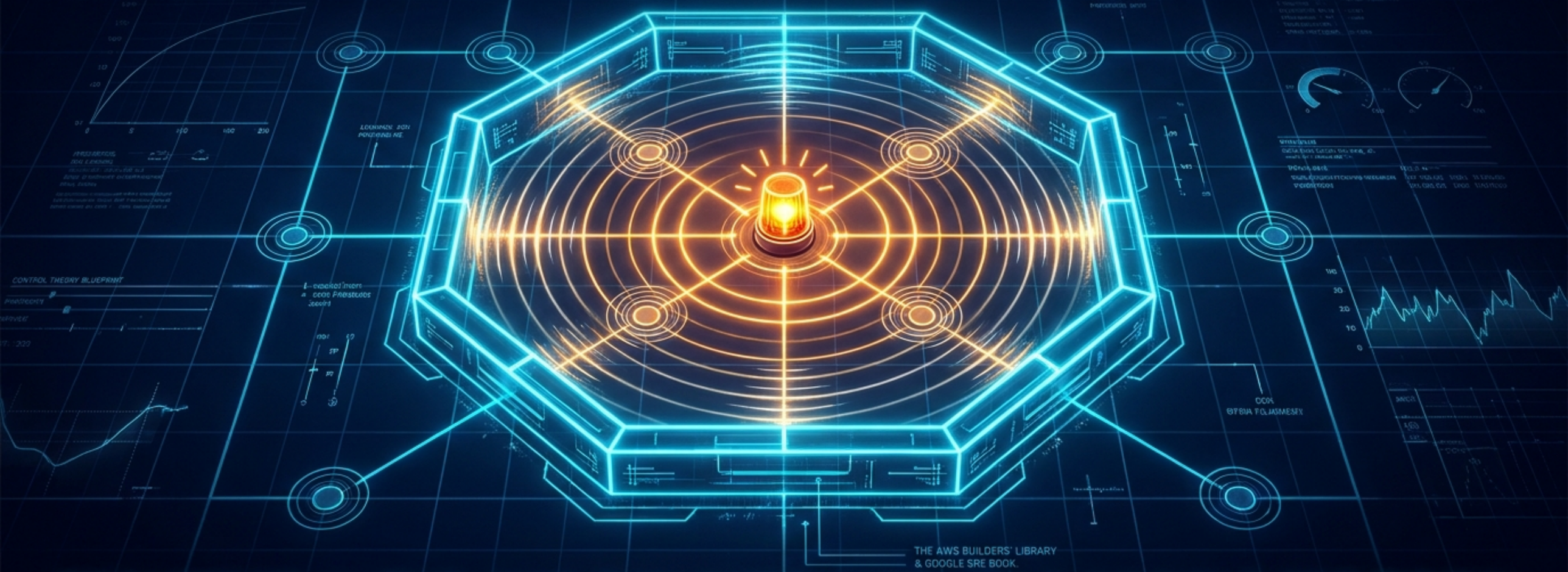
WEB ARCHITECTURE — S02E06

DISTRIBUTED RESILIENCE, OVERLOAD CONTROL, RETRIES WITH JITTER, SHUFFLE SHARDING & STOPPING CASCADING FAILURES.

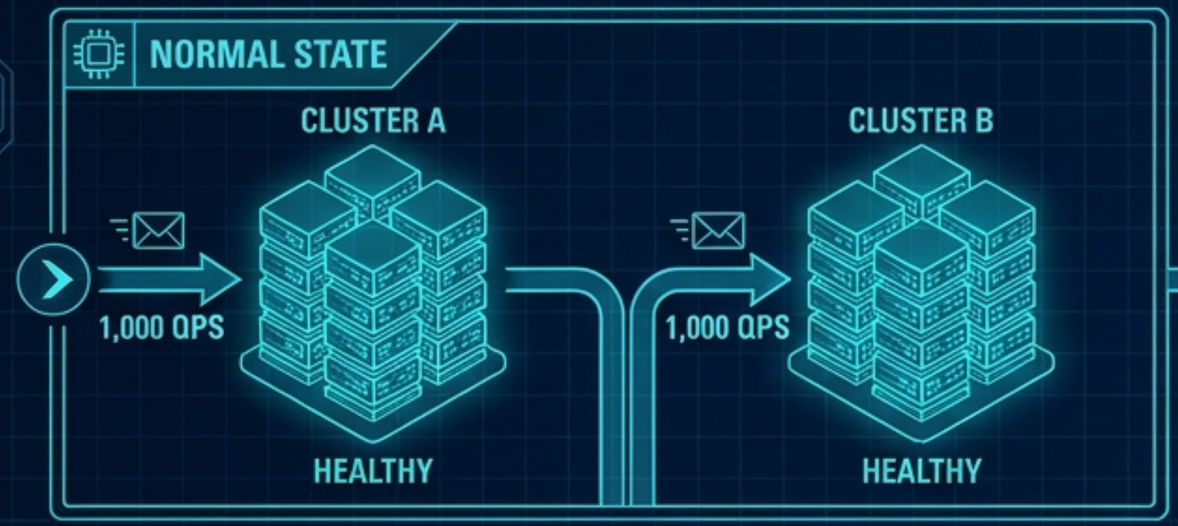


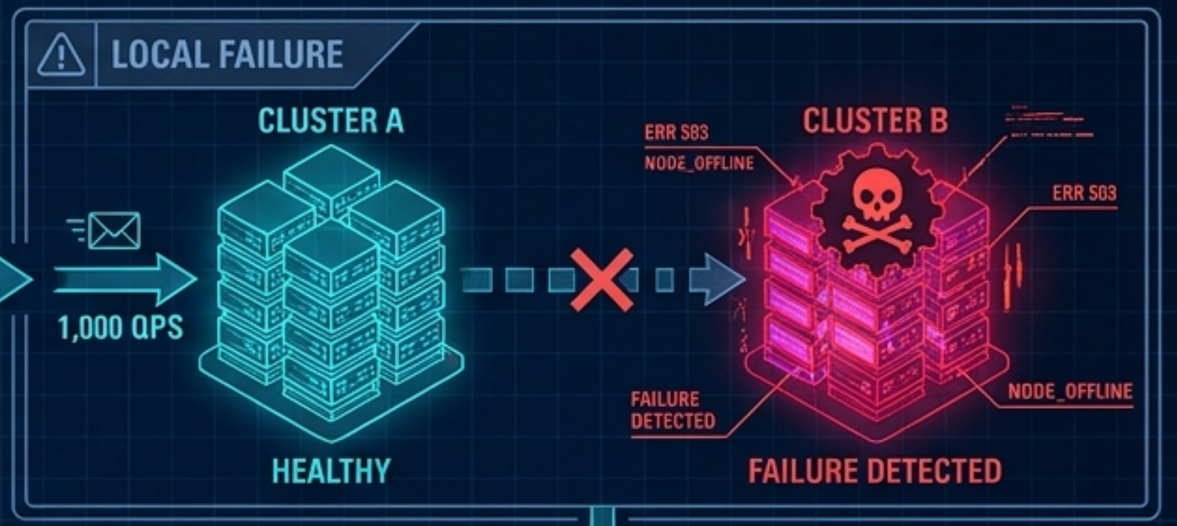
GROUNDED IN THE AWS BUILDERS' LIBRARY & GOOGLE SRE BOOK.

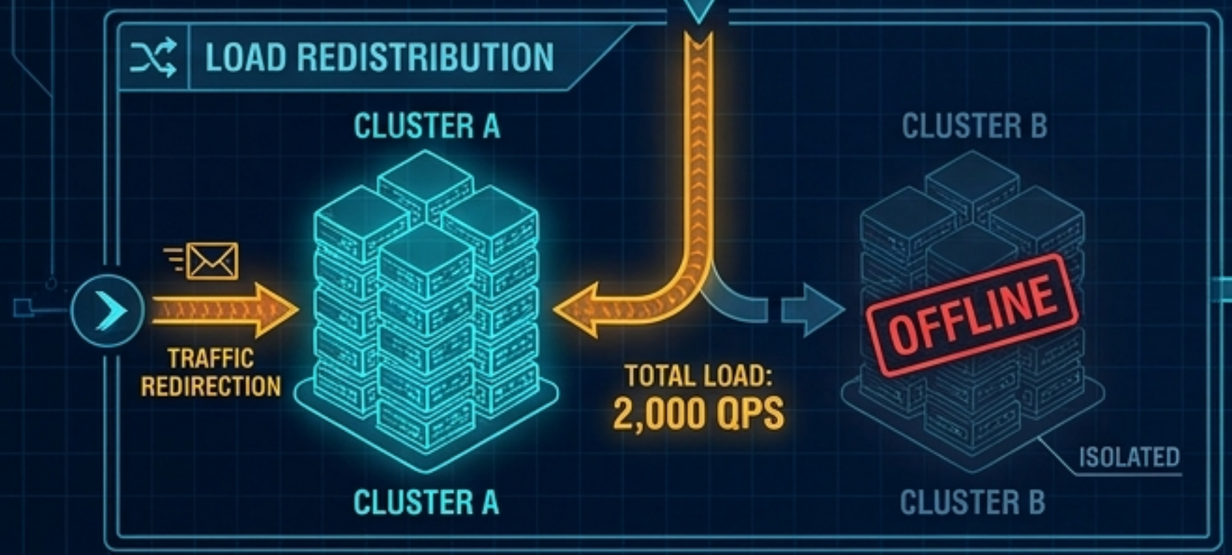
Gemini Notebook

The Anatomy of a Cascading Failure

A cascading failure is a positive feedback loop. Local failures cause load redistribution, instantly transforming a localized hardware fault into fleet-wide resource exhaustion and global unavailability.





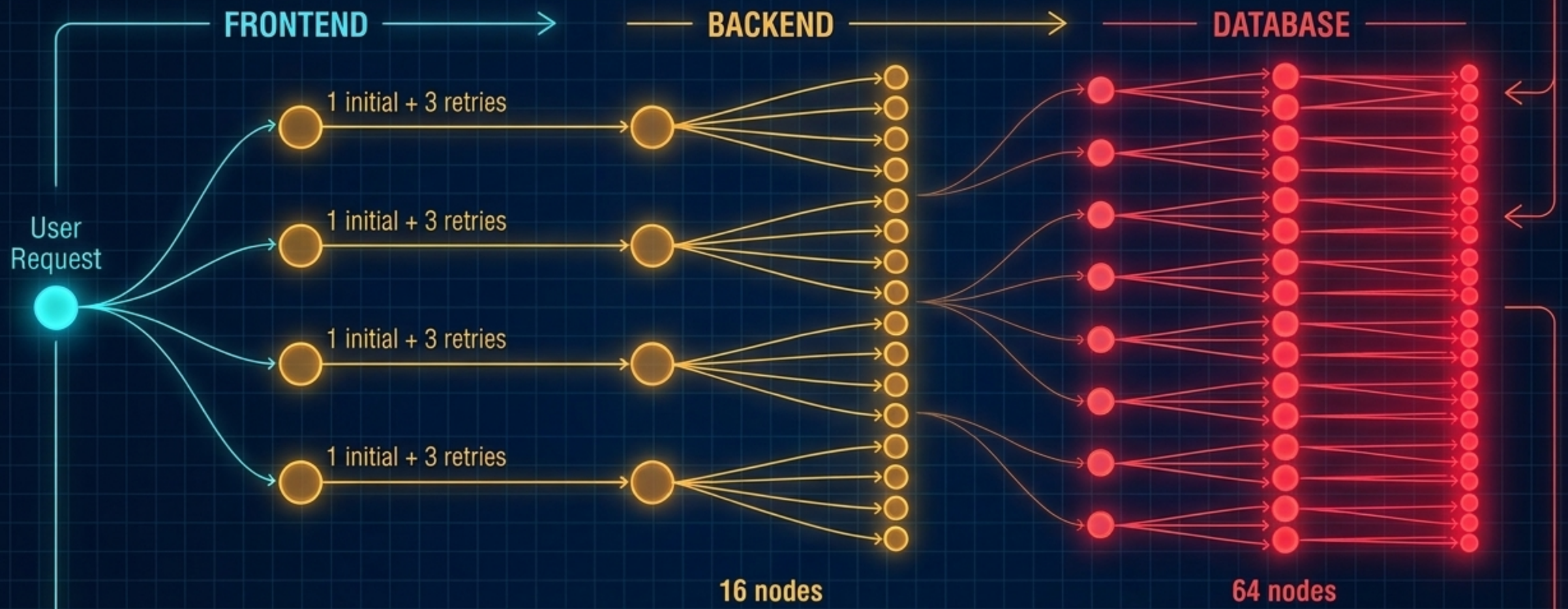




Gemini Notebook

Retry Amplification: The k^N Explosion

Retries are selfish. When independent microservice layers each configure 3 retries naively, a single transient failure amplifies multiplicatively across N hops. You DDoSed your own database.

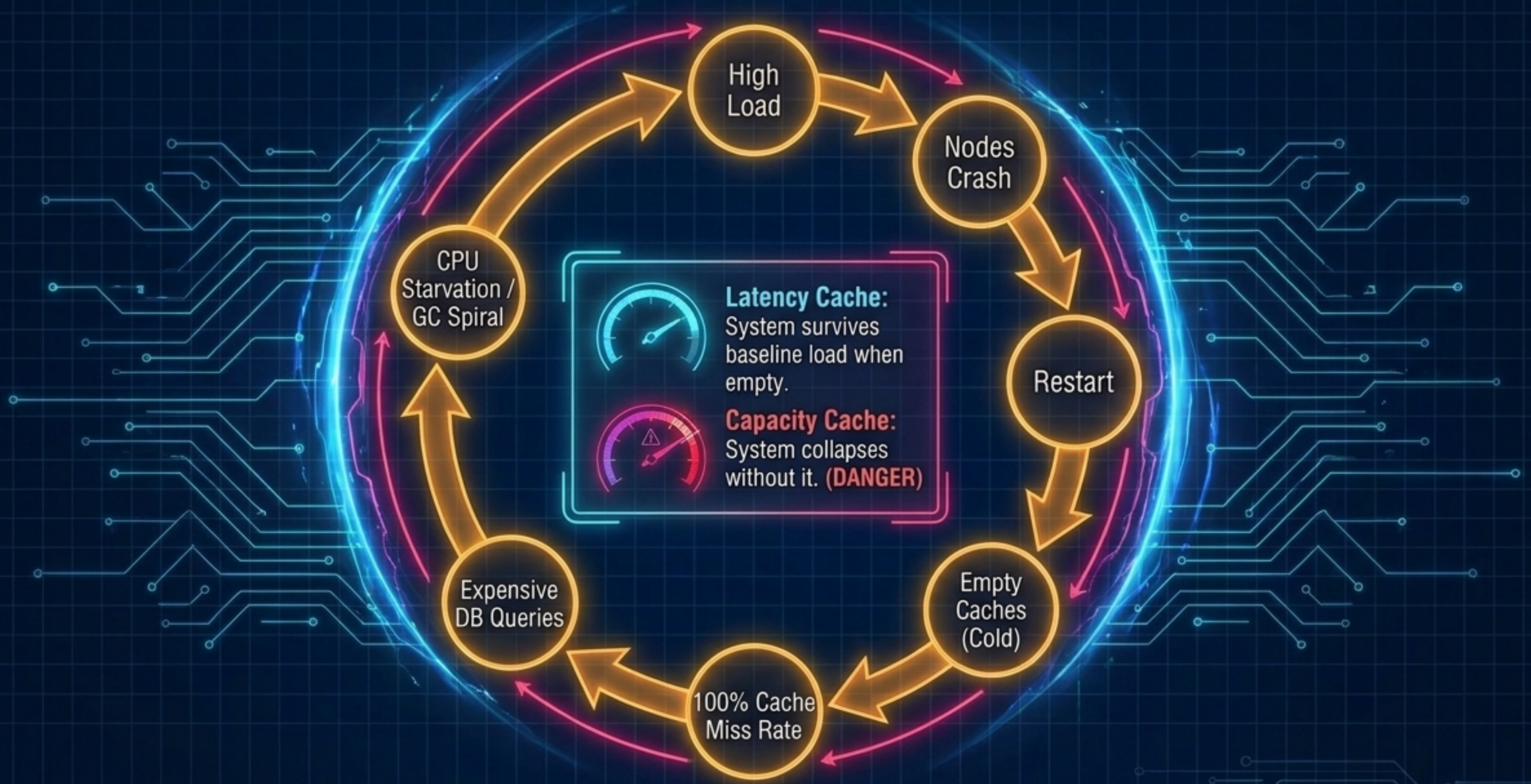


4^3 = 64 simultaneous DB queries from 1 user action

Gemini Notebook

Metastable States & The Cold-Cache Death Spiral

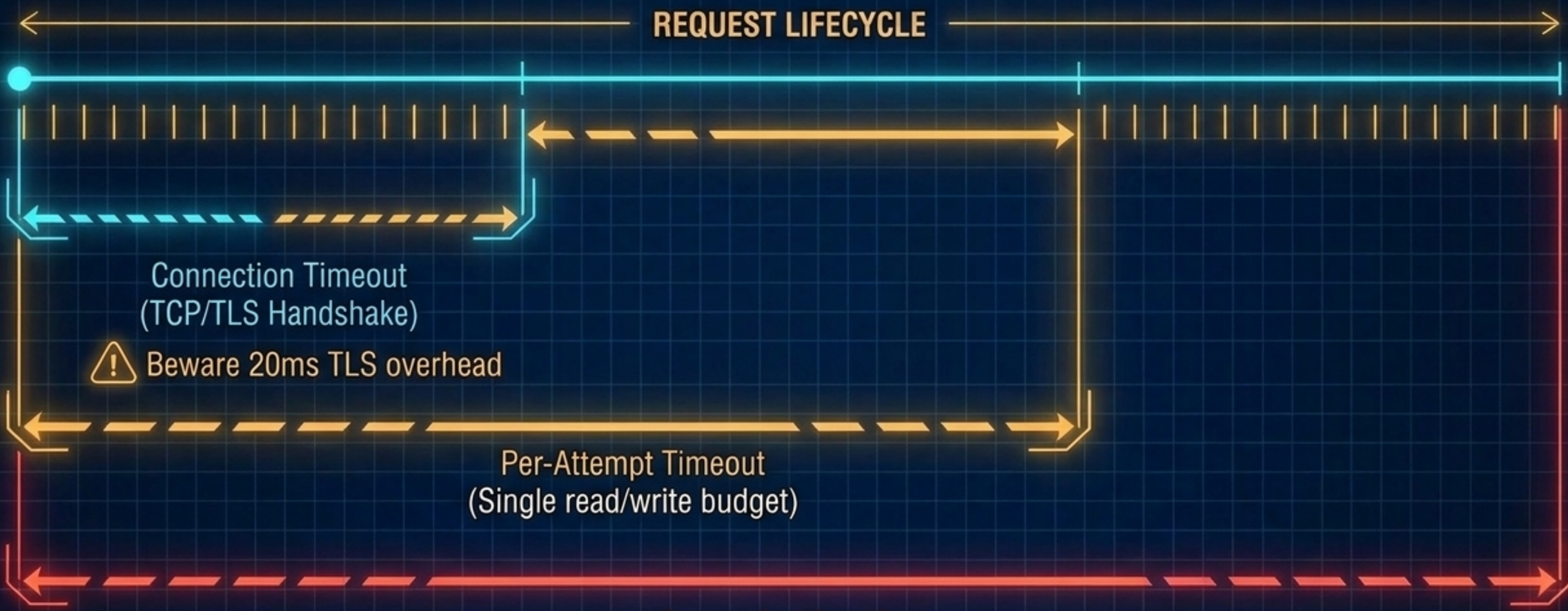
A system is metastable if it cannot recover autonomously after the initial fault resolves. If your architecture relies on a capacity cache rather than a latency cache, newly restarted nodes will instantly crash under the CPU weight of 100% cache misses.



Gemini Notebook

The Three Layers of Timeouts

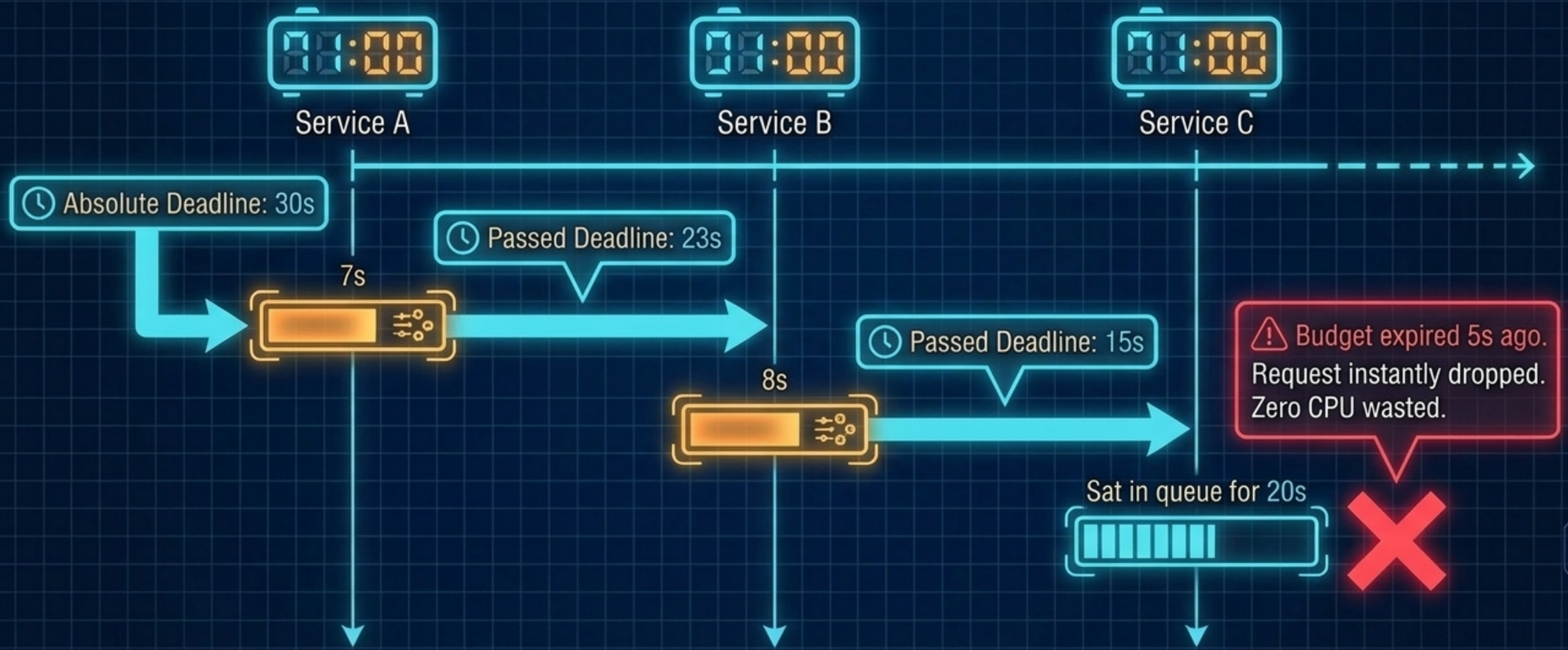
Without strict timeouts, threads block infinitely on dropped packets. This silent wait exhausts connection pools and memory.
Warning: Ensure your connection timeout accounts for TLS handshake overhead (avoid the SO_RCVTIMEO trap).



Gemini Notebook

Deadline Propagation & Canceling Wasted Work

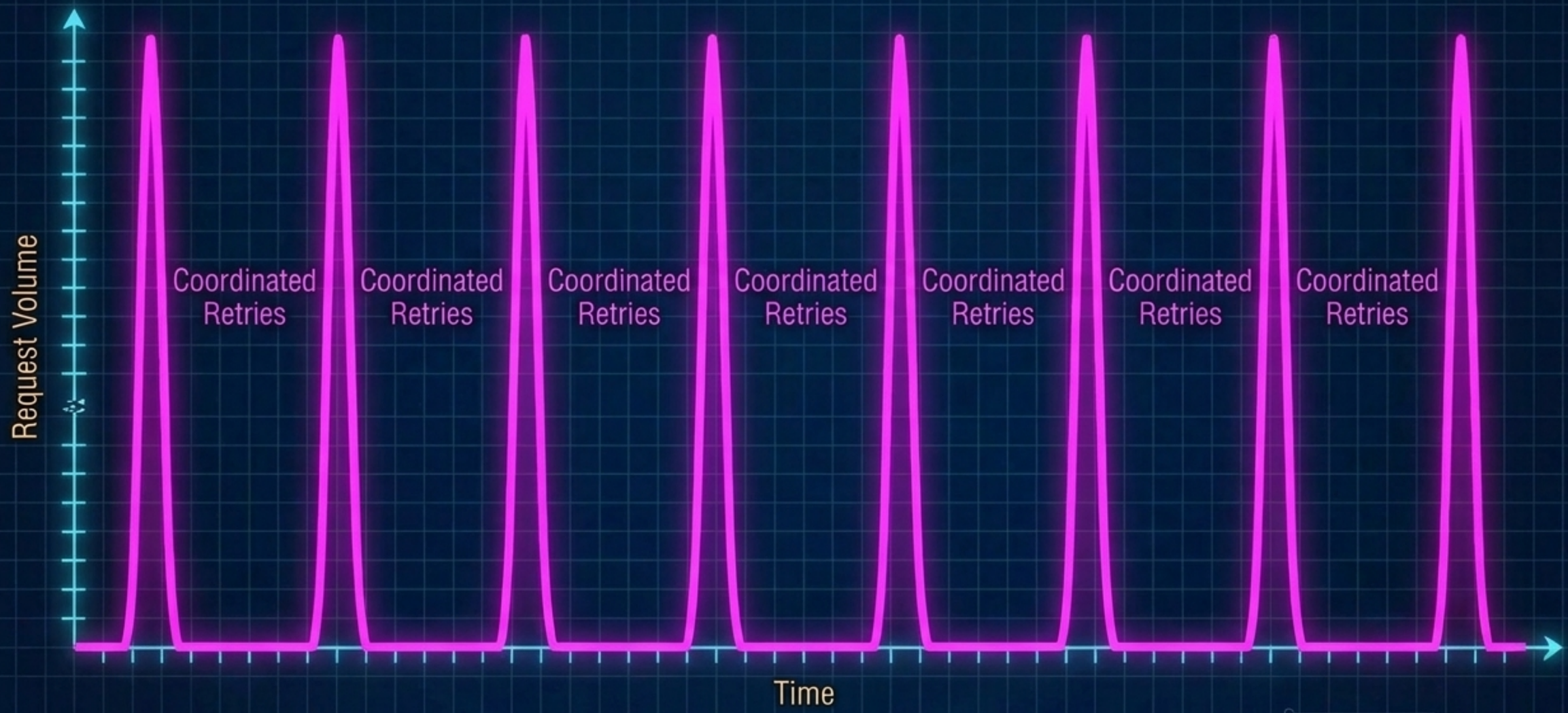
You don’t get credit for late assignments. Absolute deadlines must propagate via headers. If the client has already given up, downstream nodes must immediately shed the doomed work to save CPU.



Gemini Notebook

The Thundering Herd (Unjittered Backoff)

Standard exponential backoff fails under load. If cron jobs or disconnected clients retry simultaneously, they create highly correlated traffic spikes, repeatedly crushing the recovering service.



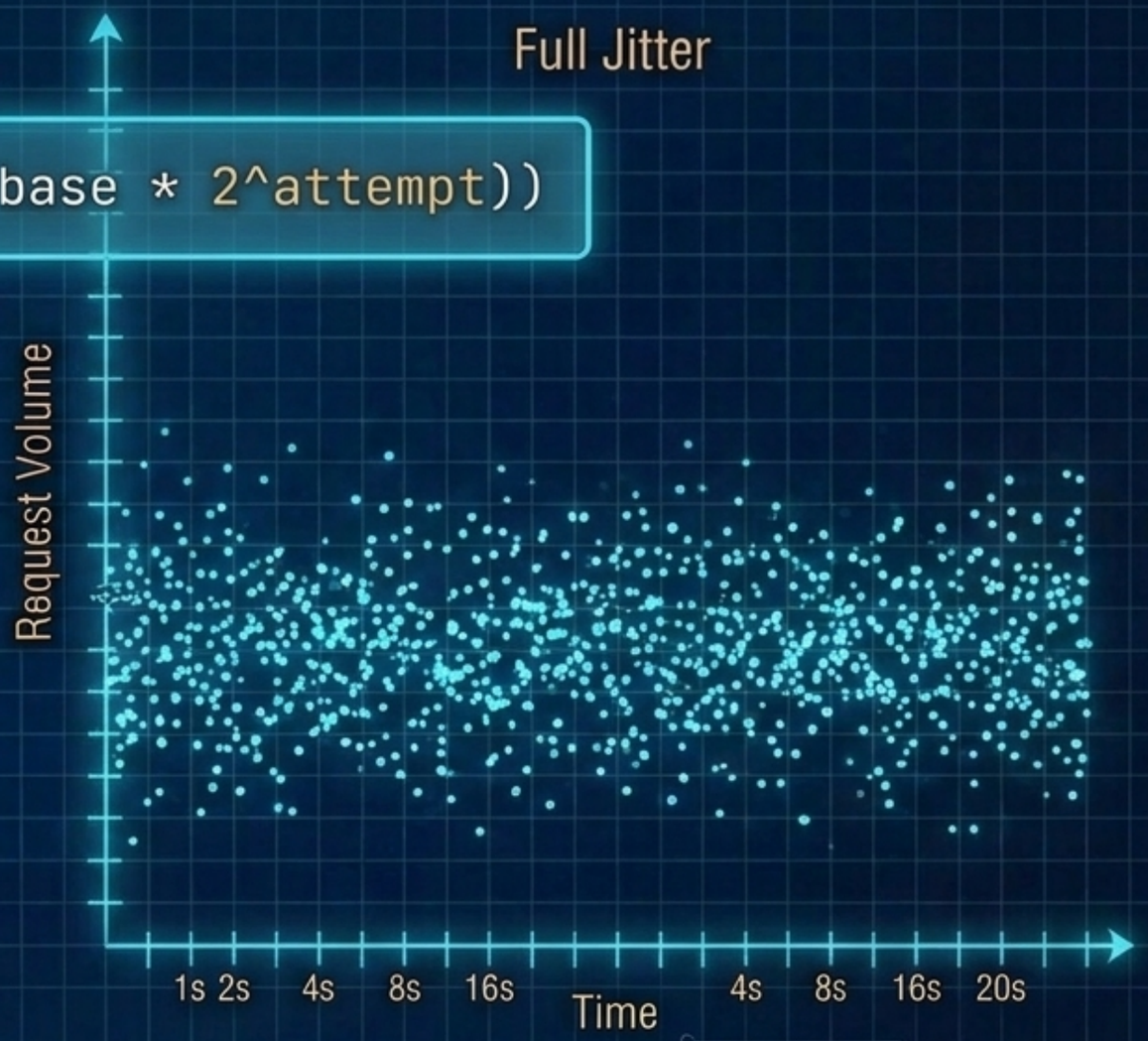
Gemini Notebook

The Math of Jitter

Jitter adds randomness to the backoff equation, converting coordinated spikes into a flat, manageable arrival rate.
Pro-tip: Seed the random function with the server IP to make jitter deterministic for debugging.

```
random(0, min(cap, base * 2^attempt))
```

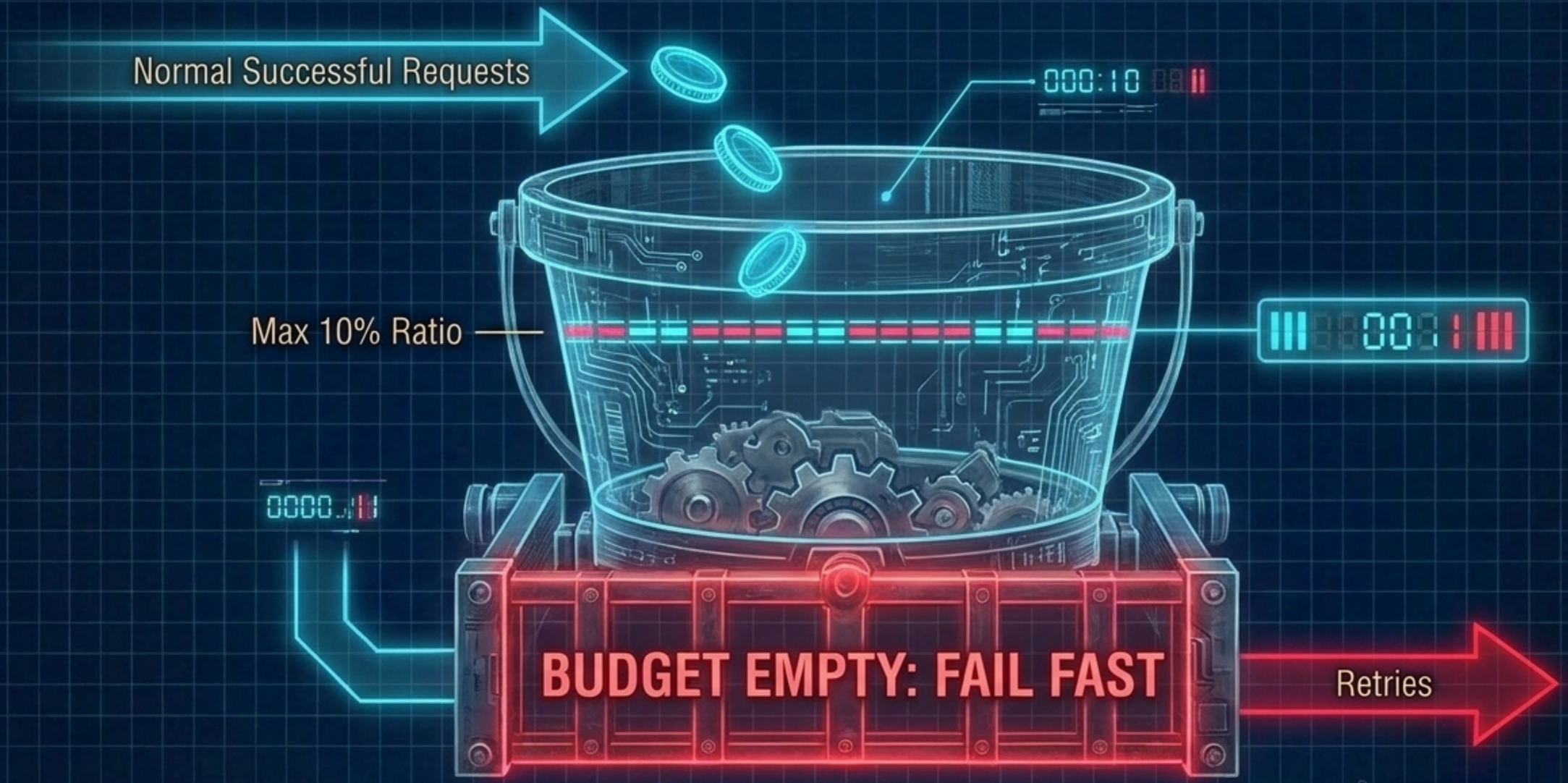




Gemini Notebook

Capping the Cost: Token-Bucket Retry Budgets

Protect the backend by limiting retries locally. A healthy client caps its retry ratio at <=10% and enforces a max of 3 attempts per request. If the token budget is exhausted, the local client fails fast.



Gemini Notebook

Network Ambiguity (Timeout != Failure)

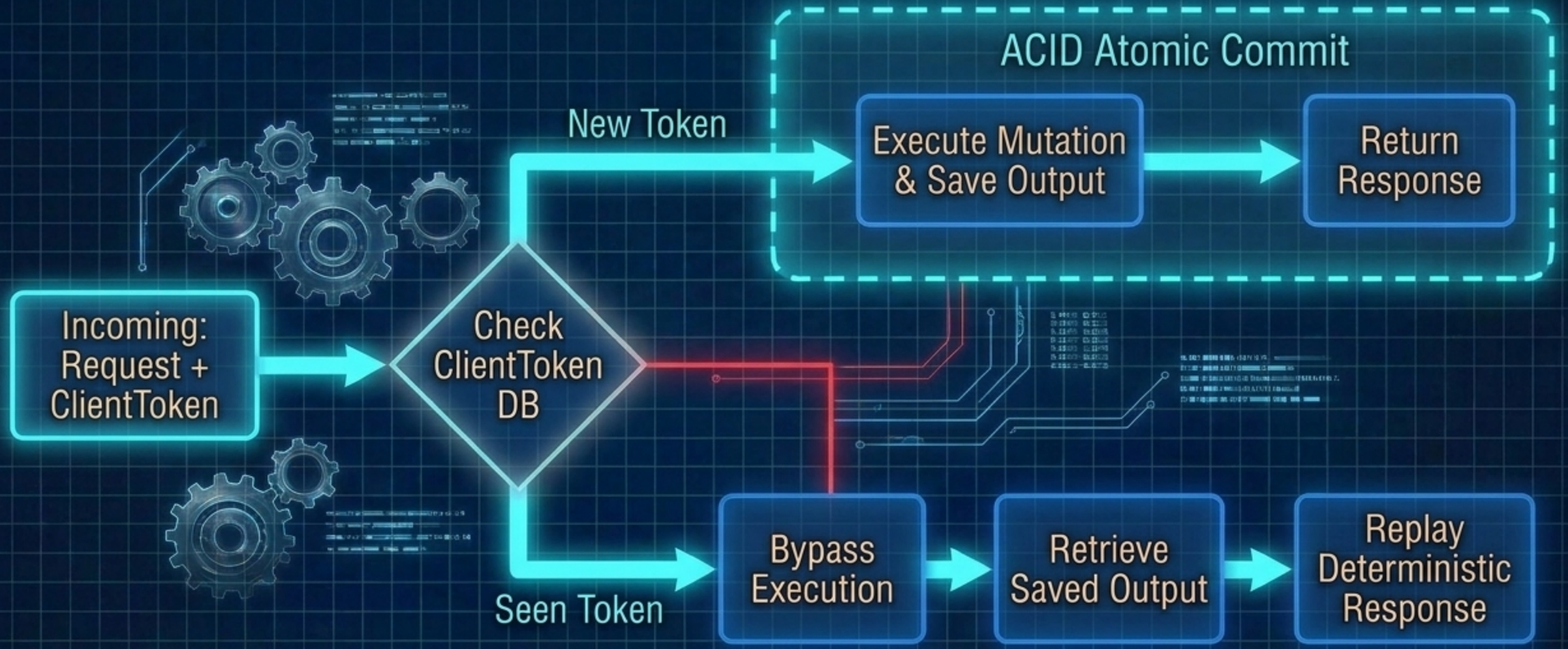
The core dilemma of distributed systems. A client timeout provides no state guarantee. Without idempotency, a blind retry of a mutation leads to catastrophic double-provisioning.



Gemini Notebook

Idempotency Tokens & Atomic State Machines

Idempotency guarantees ‘at most once’ semantics. Crucially, this requires ACID server-side compliance: the storage of the ClientToken and the actual resource mutation must commit atomically to prevent ghost resources.



Gemini Notebook

Semantic Equivalence & Edge Cases

Retries must return semantically equivalent responses to seamlessly resume client workflows.
If a retry alters parameters, reject it immediately to prevent intent collisions.

Late Arriving Request



Code 48:
Terminated

Incoming Retry

ClientToken X

Semantically
Equivalent Response:
Already Terminated
(Not an Error)

Parameter Mismatch



Code 16:
Running
(t2.micro)

Incoming Retry

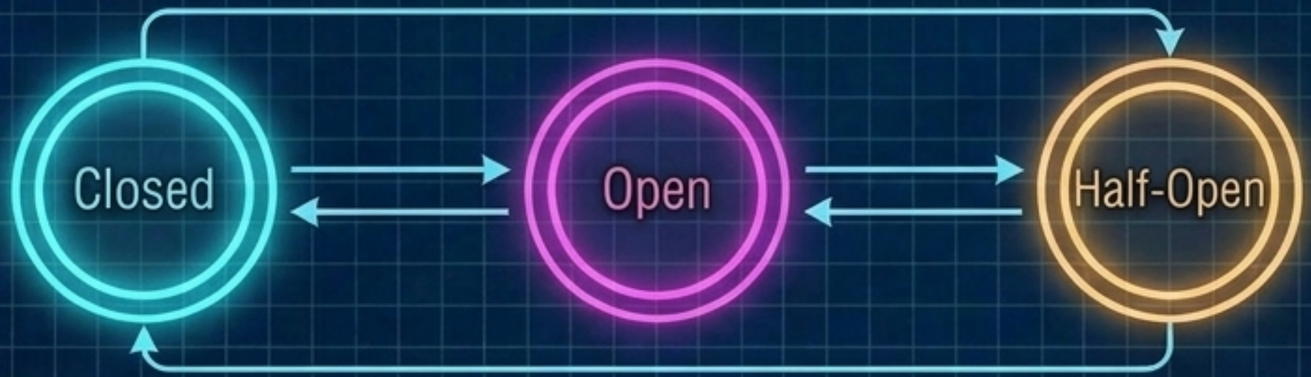
ClientToken X,
Size: m5.large

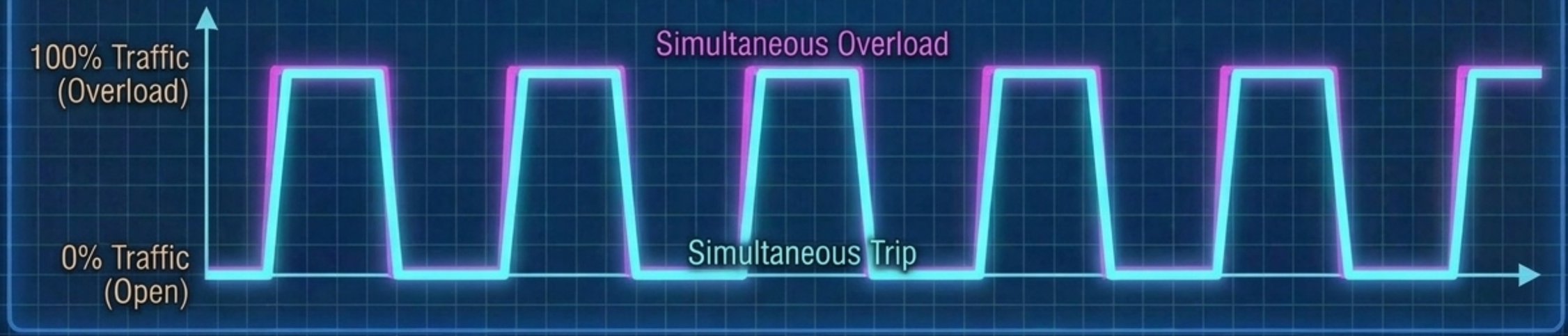
Validation Error:
Intent Collision

Gemini Notebook

The Flapping Circuit Breaker

Traditional modal circuit breakers track cluster maximum connections. However, strictly modal circuit breakers can cause violent fleet-wide flapping—where servers rapidly alternate between 0 traffic and 100% overload.





Control Theory Blueprint

Gemini Notebook

Client-Side Adaptive Throttling

Instead of modal circuit breakers, clients track local backend rejection rates. Clients probabilistically drop traffic locally before it hits the network, ensuring smooth degradation.

Live Metrics

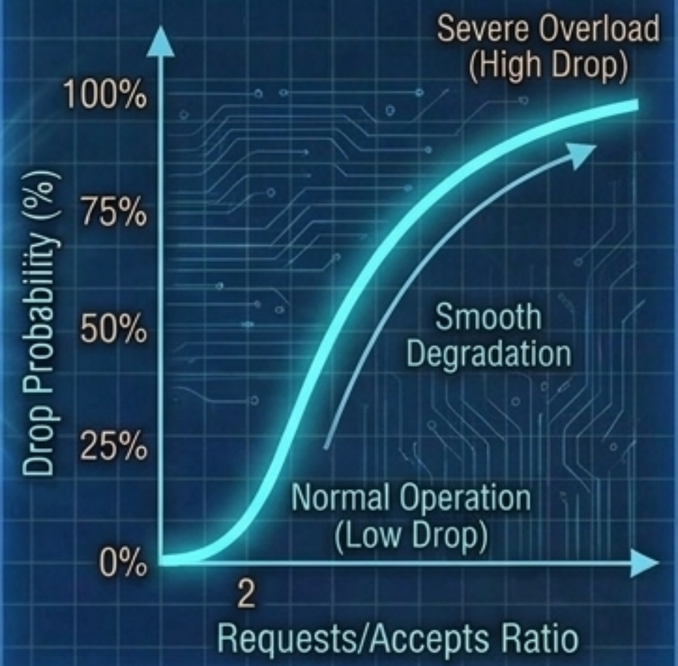
requests (attempted) = 1000

accepts (successful) = 400

$$P_{drop} = \max(0, \frac{requests - K*accepts}{requests + 1})$$

Using Google SRE Multiplier K=2

Drop Probability



Control Theory Blueprint

Gemini Notebook

Server-Side Load Shedding & QoS

If a request sits in a queue for 10 seconds, the user has already refreshed.
Switch queues to LIFO or CoDel to process fresh requests and discard stale ones.

FIFO Queue



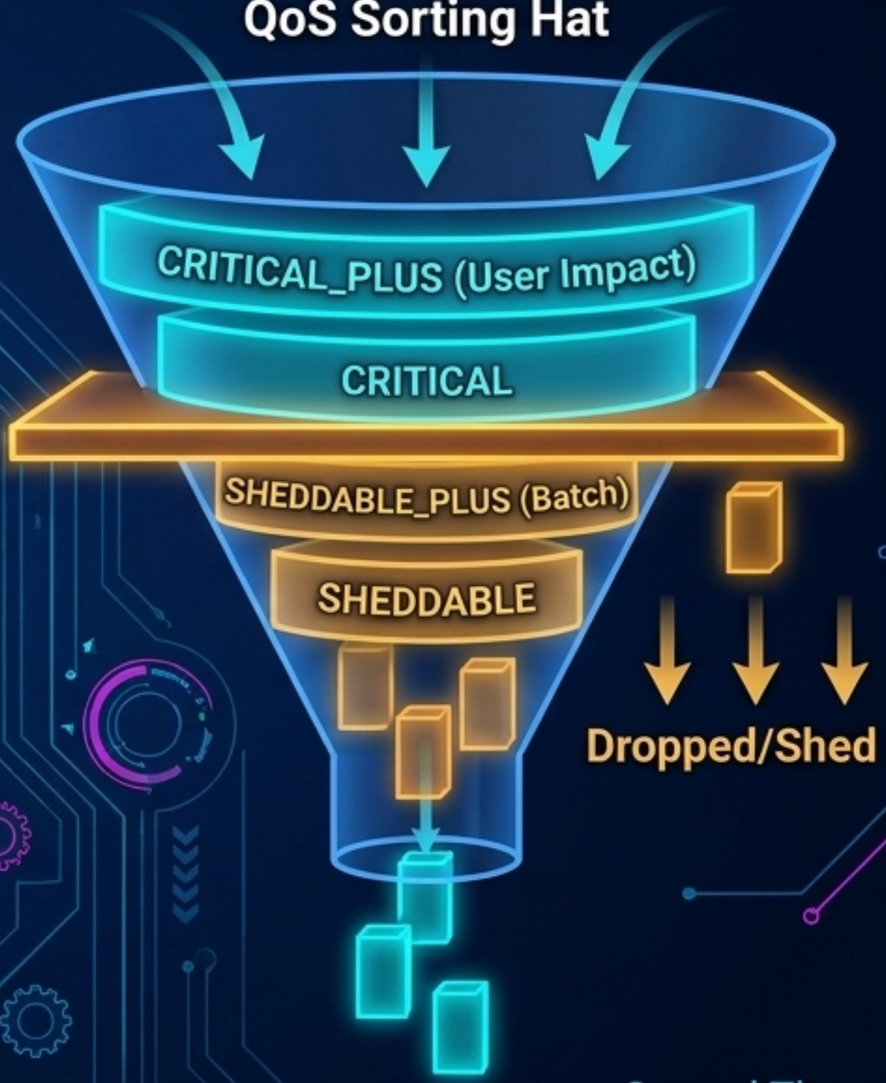
Oldest Processed First
(Blocks Fresh Requests)

LIFO Queue / CoDel



(Prioritizes Fresh, Discards Stale)

QoS Sorting Hat



Control Theory
Gemini Notebook

Diagnostic Matrix: Overload Mitigation Strategies

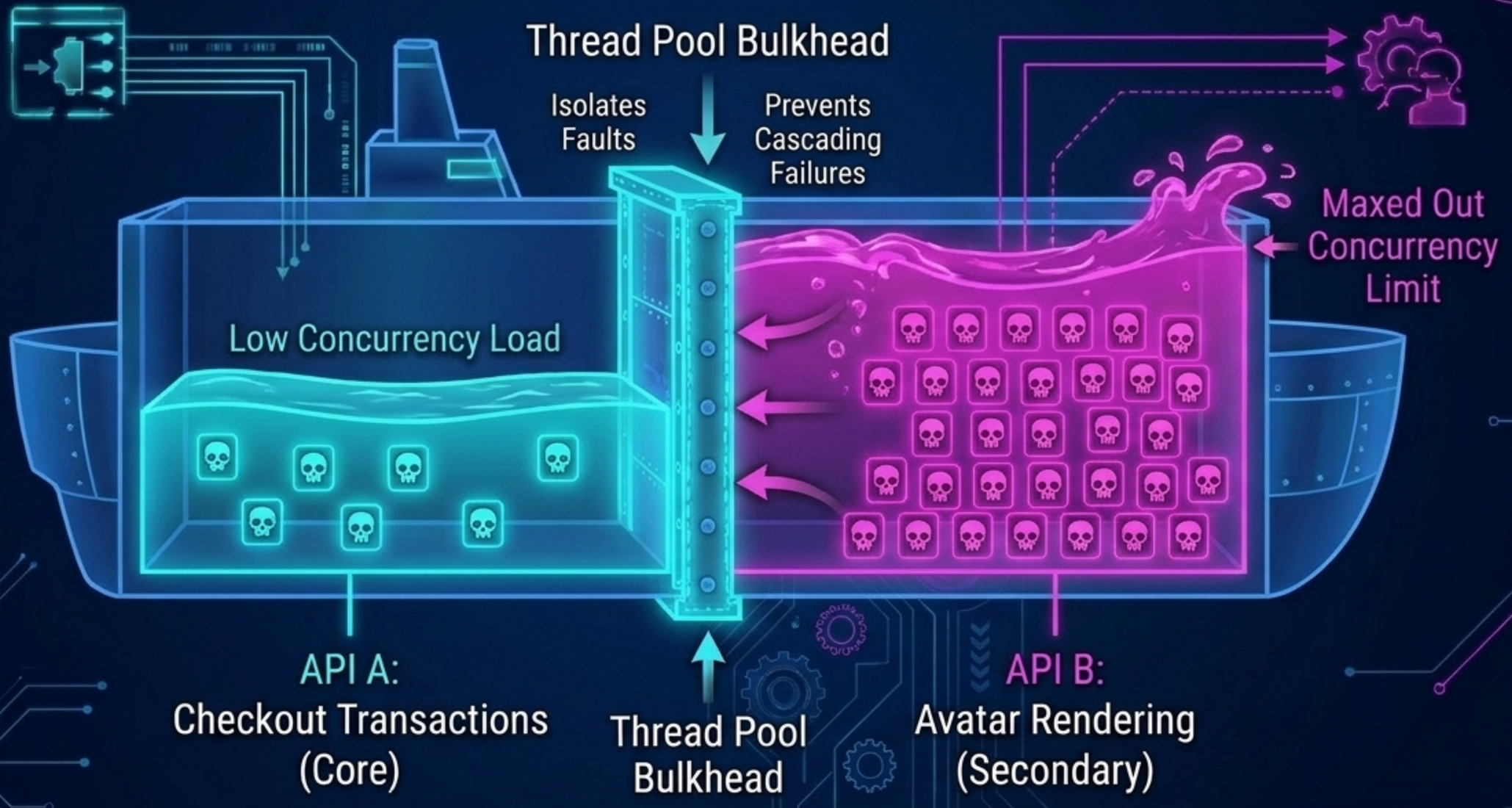
Defense in depth requires distinct mechanisms at different boundaries.
Clients throttle to preserve the network; servers shed load to preserve their CPUs.

Mechanism	Decision Locus	Trigger Metric	Primary Use Case	Flapping Risk
Circuit Breakers	Client/Proxy	Errors/Latency	Total Dependency Outage	High Risk
Adaptive Throttling	Client	Requests vs Accepts Ratio	Smooth Degraded Performance	Low Risk
Load Shedding	Server	CPU/Memory/ Concurrency	Self-Preservation & QoS	None

Gemini Notebook

Dependency Isolation (The Bulkhead Pattern)

Based on Little's Law, increased dependency latency drives up server concurrency.
Hard-allocate thread pools per dependency so a slow secondary API cannot starve core operations.



Control Theory
Gemini Notebook

Cellular Architecture & Static Stability

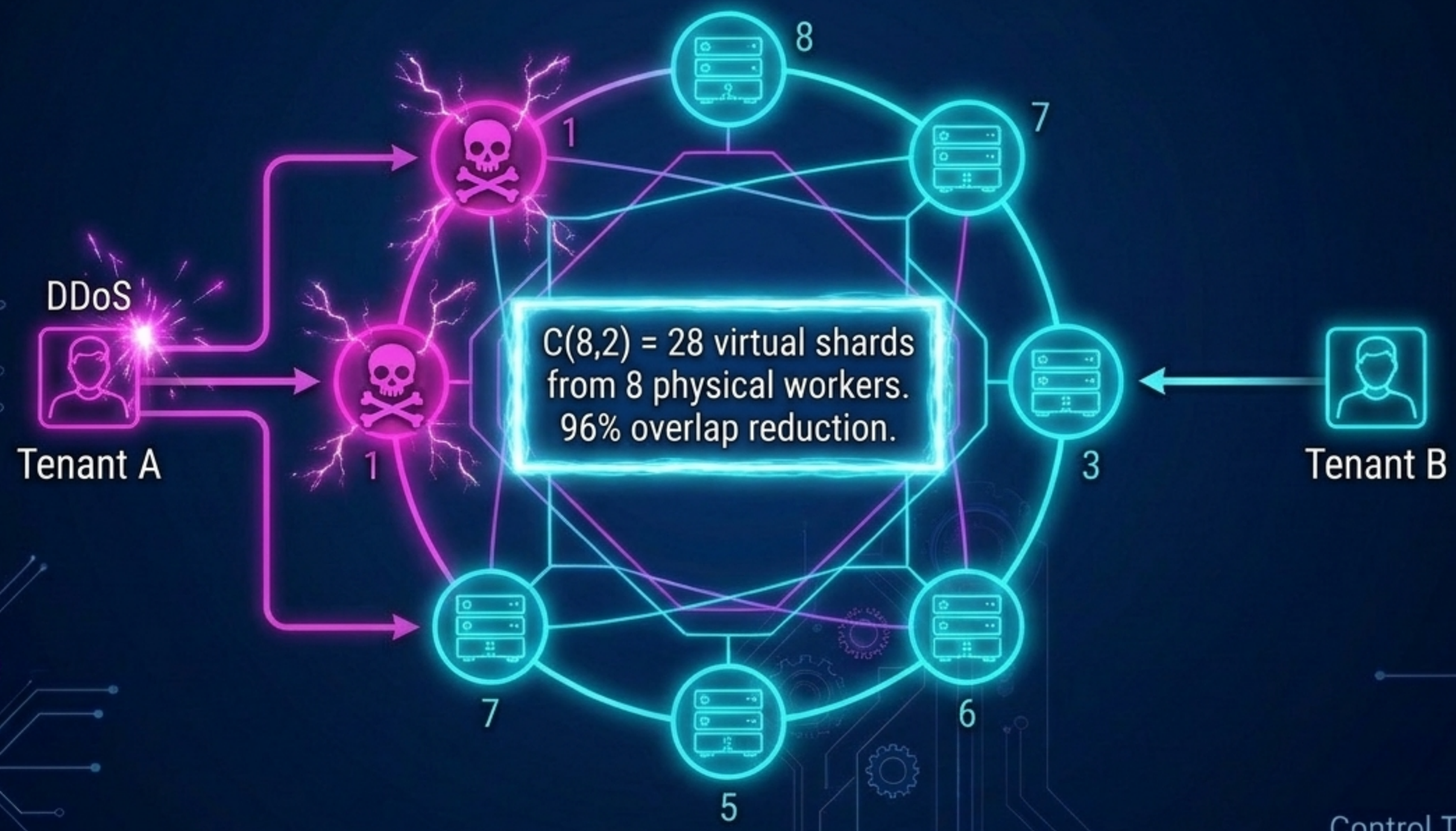
True fault tolerance requires isolating infrastructure. Architect systems into independent cells so that operator errors or bad deployments are physically contained to a fractional blast radius.



Gemini Notebook

Combinatorial Shuffle Sharding

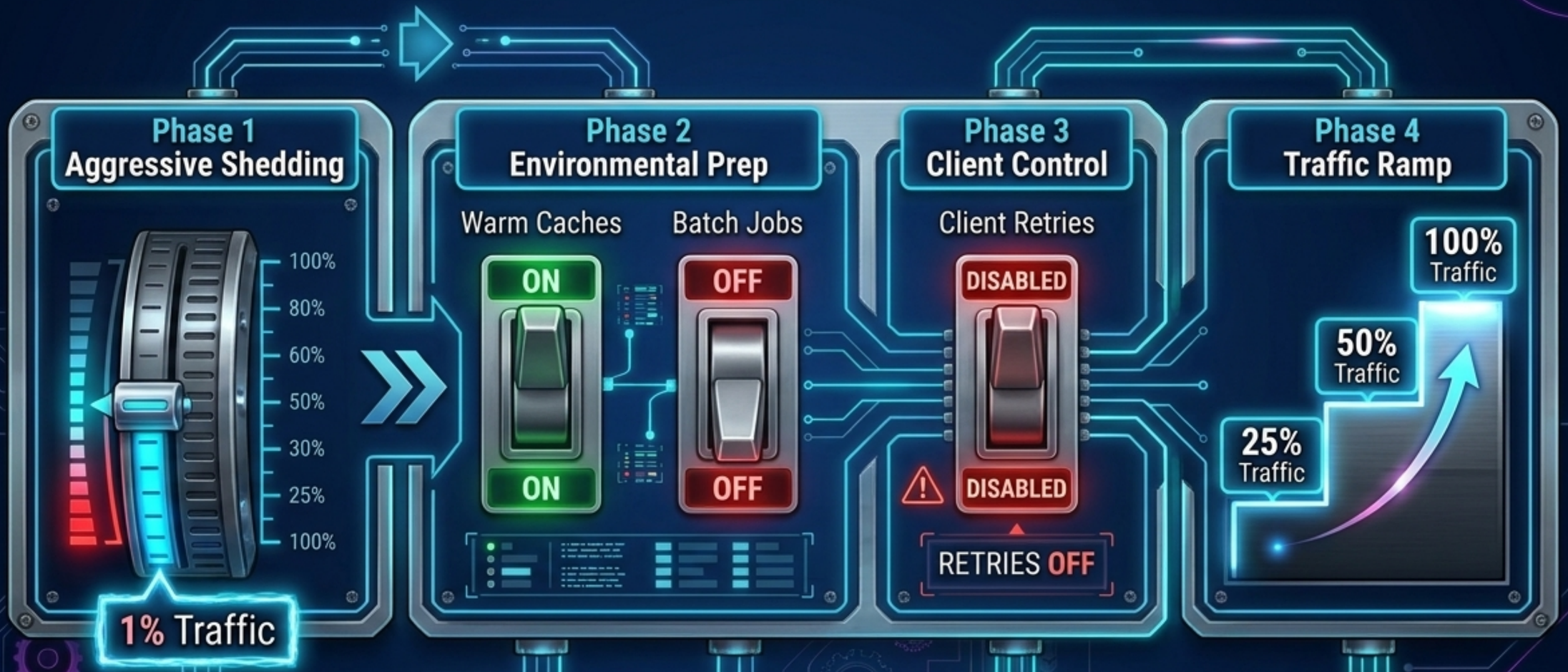
Instead of standard sharding, assign clients to overlapping virtual combinations. A catastrophic noisy neighbor can only ever overlap with a fraction of your nodes, preserving static stability.



Control Theory
Gemini Notebook

The SRE Playbook: Escaping Metastable Overload

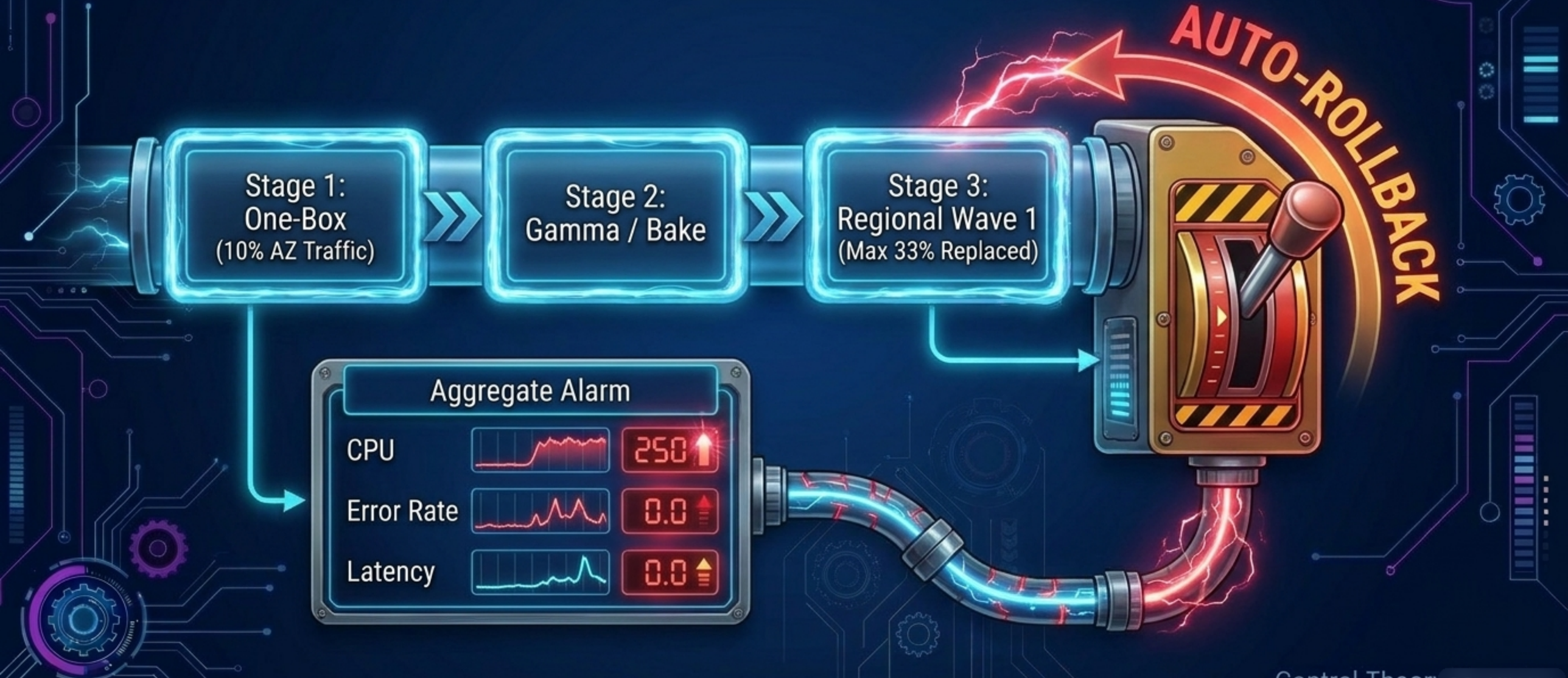
When a system enters a death spiral, you must aggressively throttle traffic below the baseline to allow caches to warm and servers to stabilize before ramping back up.



Control Theory
Gemini Notebook

Cautious Rollouts & Automated Pipelines

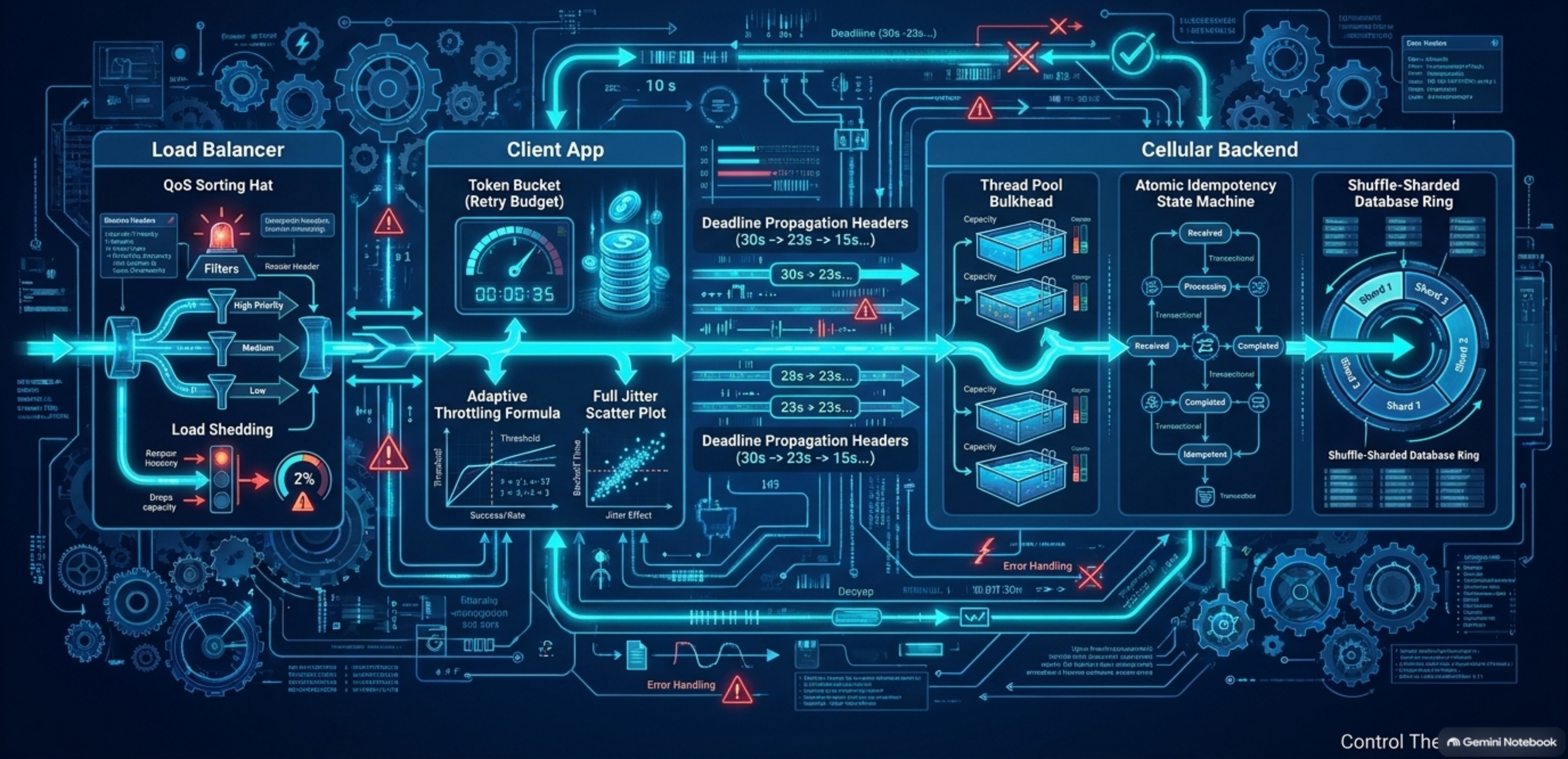
Prevent correlated failures at the source. Use staggered rolling deployments. Tie CI/CD pipelines to high-severity aggregate alarms to auto-rollback without human intervention.



Gemini Notebook

The Unified Distributed Resilience Architecture

Reliability is not a single feature; it is assuming failure at every boundary. By layering defense-in-depth, distributed systems survive local faults without triggering global cascading failure.



Control The

Gemini Notebook