OpenSearchCon INDIA

100M Logs a Day

Performance Engineering an OpenSearch Observability Platform for Kubernetes

Sravanthi Naga, Pegasystems

100 Million Logs a Day

Performance engineering an OpenSearch observability platform for Kubernetes.

> INITIATING TIER-0 PROTOCOL // OPENSEARCHCON

Sravanthi Naga

OpenSearchCon INDIA

- **Senior Engineering Manager**, Performance & Resiliency at **Pegasystems**
- Kubernetes, OpenSearch Evangelist
- Performance Tuning & An Observability nerd – loves tinkering with systems
- Cloud Native Hyderabad (CNCF) Volunteer
- Hobbies: Reading bed-time stories to my kid

LinkedIn

GitHub

Agenda

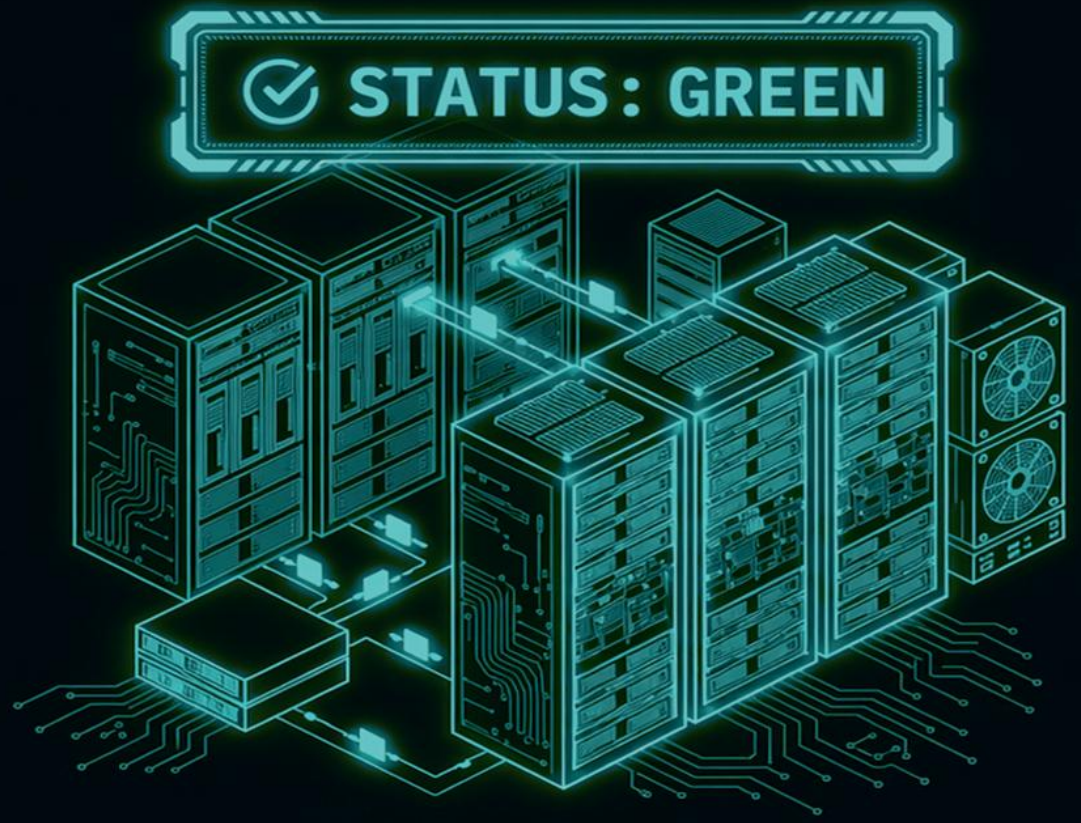
- *Problem Statement*
- *OpenSearch Internal Metrics*
- *Why systems break at scale?*
- *Solution Approach / Controls*
- *Return on Engineering & Key Benefits*
- *Key Conclusion*

OpenSearchCon INDIA

Problem Statement

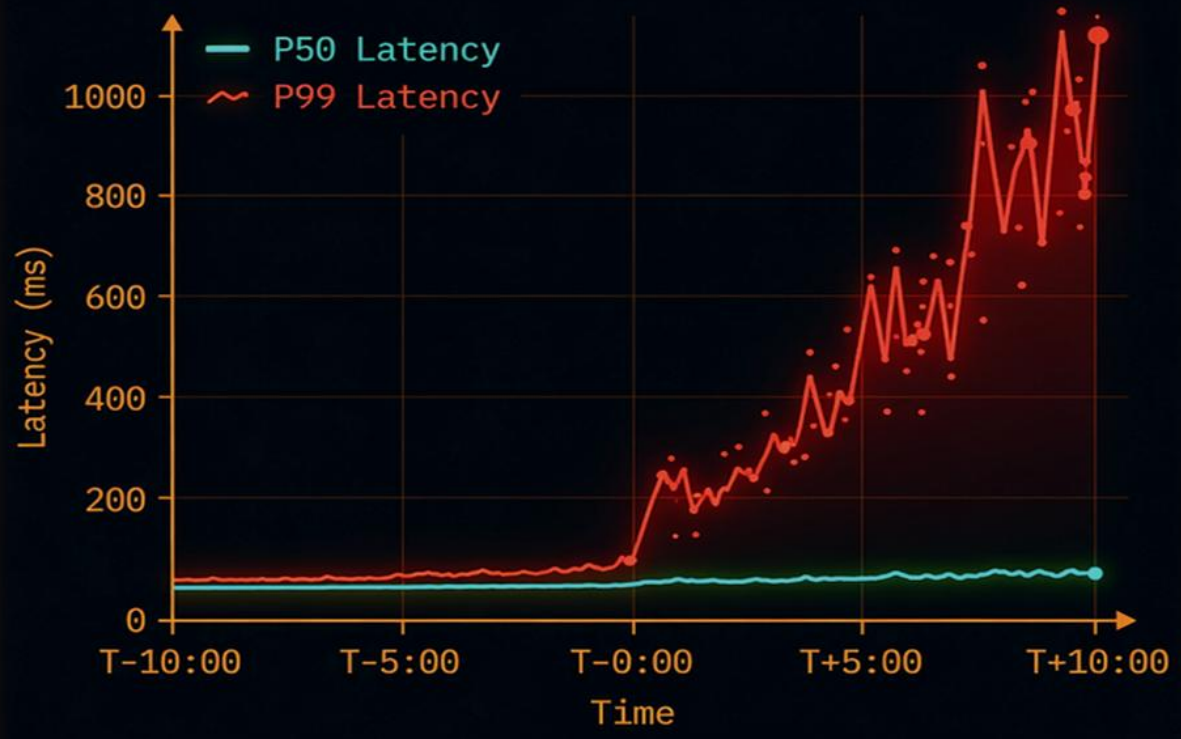
Observability was built to protect production. At scale, it became the bottleneck

The Illusion



CPU, memory, and disk metrics within normal thresholds. All nodes responding.

The Reality

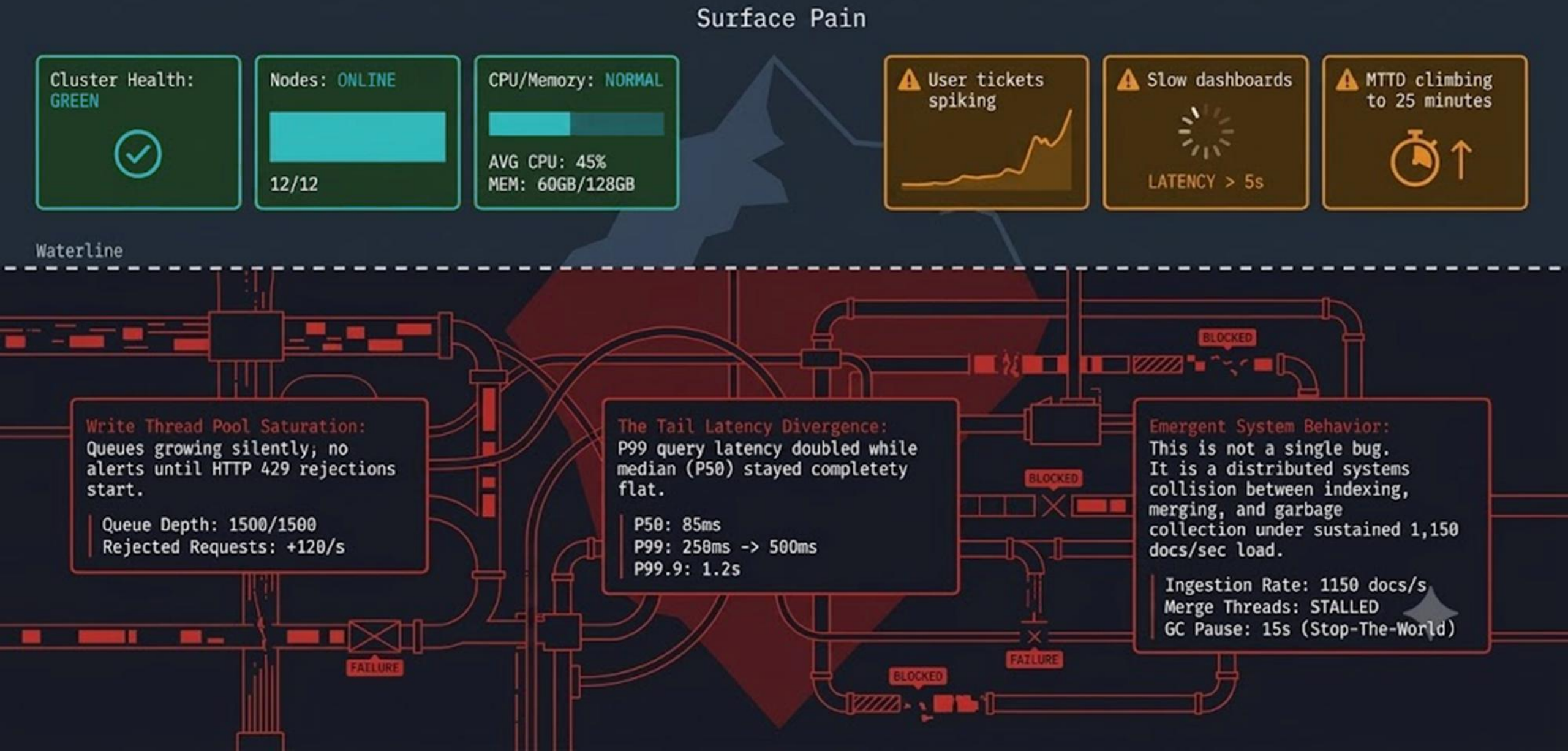


[WARN] Write thread pool queue grew silently.

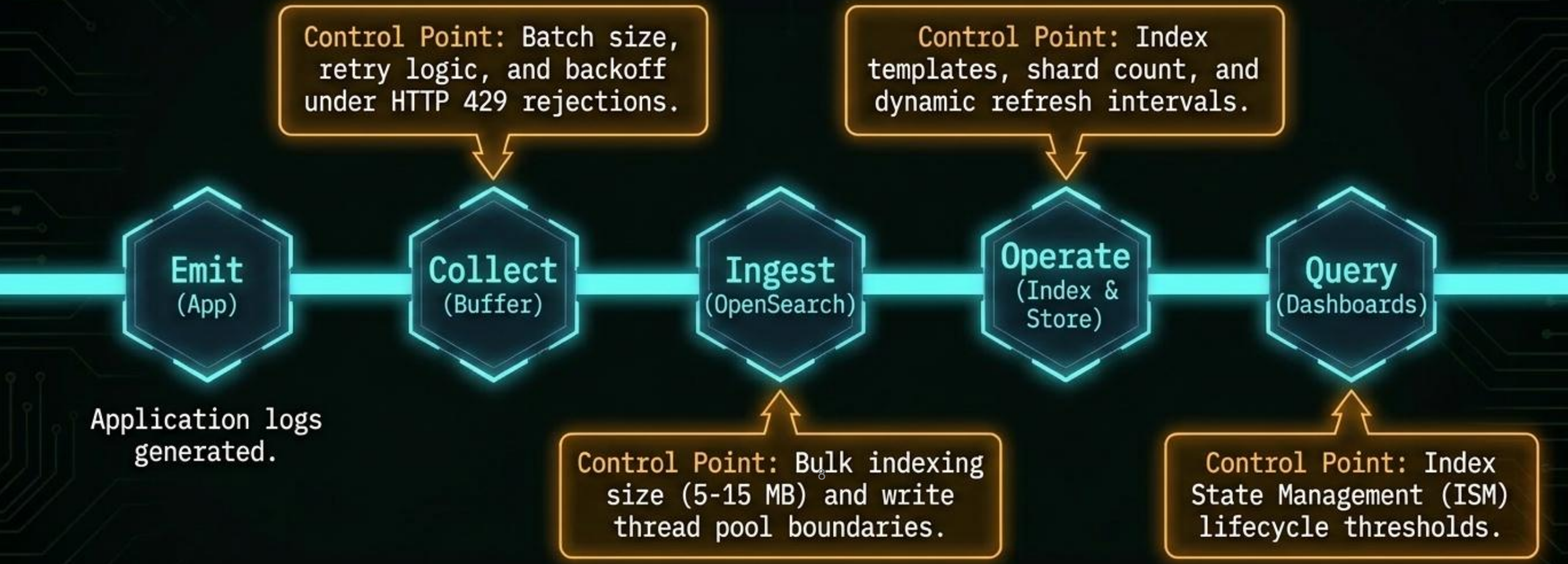
[FAIL] No alerts fired until HTTP 429 rejections cascaded.

[TIME] Mean Time to Diagnosis (MTTD) climbed to 25 minutes of manual cluster inspection.

Everything looked fine on the dashboard until the platform inverted



The End-to-End Telemetry Pipeline



Key Takeaway: Performance issues can originate anywhere in this chain. Optimizing only one component yields diminishing returns.

OpenSearchCon INDIA

OpenSearch Internal Mechanics

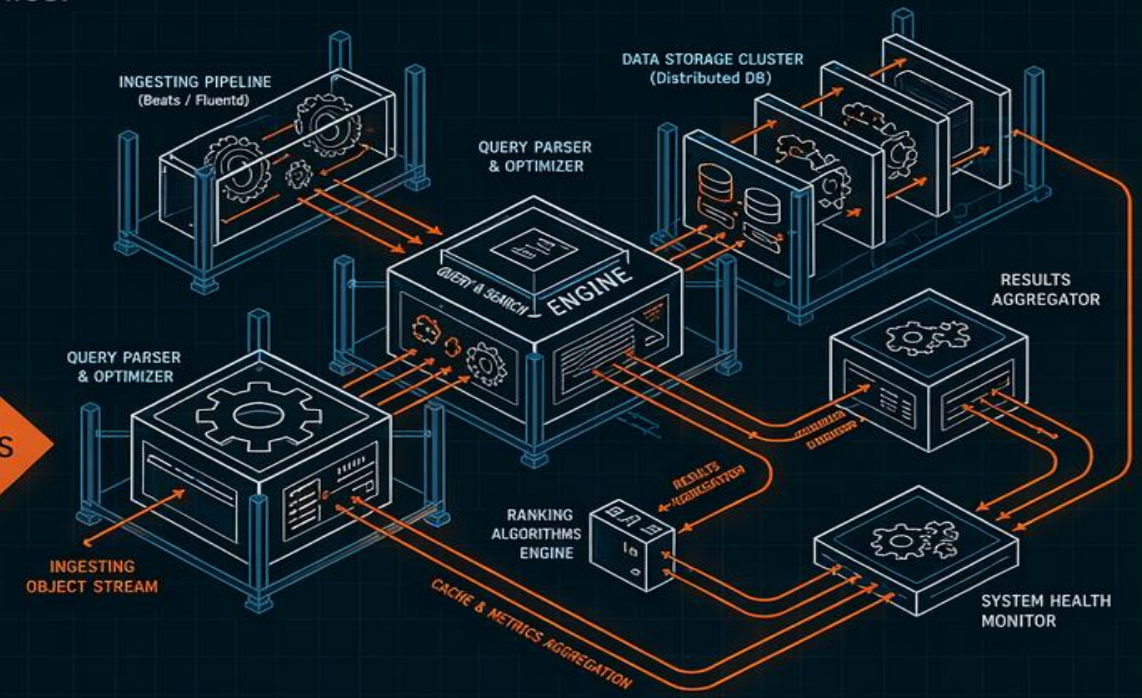
LOAD_MODULE: true

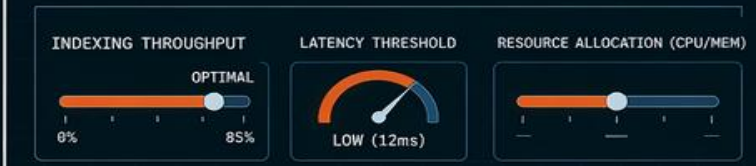
Handling **100** million logs per day requires understanding the complete ecosystem.

Troubleshooting at scale often looks like a game of whack-a-mole—fixing isolated errors as they appear.
However, true optimization requires a mental model of the low-level mechanics.

```
2024-05-27T11:31:21.460Z ERROR [core:34561] : Connection timed out: to="shard-03"
2024-05-27T11:31:21.460Z ERROR [core:34201] : Query parse failed: unexpected token near "partner"
2024-05-27T11:31:21.460Z WARN  [cluster:master] : High latency detected on query parser
2024-05-27T11:31:21.460Z INFO  [replication:sync] : Replica catch-up ceceed bias 1278
2024-05-27T11:31:21.462Z INFOG [node:engine1206] : Batch processing started for job 90912
2024-05-27T11:31:21.463Z DBRUG [index:lucene]  : /usr/local/lucene/core/segments_9 loaded
2024-05-27T11:31:21.463Z DEBUG [perf:mem]    : Buffer utilization: 68%
2024-05-27T11:31:21.463Z
2024-05-27T11:31:21.462Z ERROR [core:34561] : Connection timed out: to="shard-03"
2024-05-27T11:31:21.463Z WARN  [cluster:coor[] Connection retry attempt 3/5
2024-05-27T11:31:21.463Z ERROR [search:exec] : Circuit breaker detected high query parser
2024-05-27T11:31:21.463Z WARN  [cluster:master] : High latency detected on auth service
2024-05-27T11:31:21.460Z ERROR [core:angio1206] : Segment merge completed
2024-05-27T11:31:21.463Z EBUOG [io:flush]    : Flushed catch-up clock /var/lib/opensearch/translog
2024-05-27T11:31:21.463Z DEBUG [replication:sync] : Replica catch-up completed in 1178ms
2024-05-27T11:31:21.462Z ERROR [core:34561] : Connection timed out: to="shard-03"
2024-05-27T11:31:21.463Z WARN  [cluster:coord] : Candidate node demoted: high cpu
2024-05-27T11:31:21.463Z ERROR [search:exec] : Circuit breaker detected high query parser
2024-05-27T11:31:21.462Z INFO  [batch:indexer] : Batch indexing completed
2024-05-27T11:31:21.463Z DEBUG [node:engine1206] : Batch processing started for job 90912
2024-05-27T11:31:21.489Z INFO  [index:lucene] : Successfully merged segments_10
2024-05-27T11:31:21.483Z INFO  [cache:query] : Query cache hit ratio: 79%
2024-05-27T11:31:21.482Z DEBUG [perf:mem]    : Buffer utilization: 72%
2024-05-27T11:31:21.479Z
2024-05-27T11:31:21.429Z CRITICAL [disk:alert] : Detecting free space is < 10% on /data/nodes/0
2024-05-27T11:31:21.483Z ERROR [snapshot]    : Snapshot creation failed: not enough disk space
2024-05-27T11:31:21.483Z ERROR [alloc:disk] : Relocating shards due to low disk watermark
2024-05-27T11:31:21.483Z WARN  [aluster:master] : Routing table updated
2024-05-27T11:31:21.483Z WARN  [alloc:disk] : Moved shard [logs-2024.05.27][2] to node-04
2024-05-27T11:31:21.483Z ERROR [search:queue]: Rejected execution: queue capacity exceeded
2024-05-27T11:31:21.480Z CRICAL [jvm:heap]    : High heap usage: 92%
2024-05-27T11:31:21.483Z INFO  [node:engine1206] : Garbage collection completed (45ms)
2024-05-27T11:31:21.480Z INFOG [batch:indexer] : Batch processing started for job 90912
2024-05-27T11:31:21.495Z DEBUG [io:merge]  : Merge throttle active: 20MB/s
2024-05-27T11:31:21.498Z INFO  [coord:query] : Search request completed in 185ms
2024-05-27T11:31:21.499Z
2024-05-27T11:31:21.489Z ERROR [core:34561] : Connection timed out: to="shard-03"
2024-05-27T11:31:21.483Z ERROR [cluster:coord] : No healthy node available
2024-05-27T11:31:21.483Z WARN  [search:exec] : Fallback to cached results for query parser
2024-05-27T11:31:21.480Z WARN  [cluster:master] : High latency detected to query parser
2024-05-27T11:31:21.499Z ERPOR [replication:sync] : Replica catch-up completed in 1310ms
2024-05-27T11:31:21.493Z DEBOG [node:engine1206] : Batch proceotion: queeated job 90912
2024-05-27T11:31:21.493Z INFO  [index:lucene] : /usr/local/lucene/core/segments_11 loaded
2024-05-27T11:31:21.489Z DEBUG [io:flush]  : Translog fsync completed
2024-05-27T11:31:21.499Z WARN  [jvm:gc]    : GC overhead detected
2024-05-27T11:31:21.490Z ERROR [core:34561] : Connection tiimd out: to="shard-03"
2024-05-27T11:31:21.490Z CRITICAL [disk:alert] : Disk usage exceeded 95% on /data/nodes/0
2024-05-27T11:31:21.493Z ERROR [search:exec] : Circuit breaker tripped
2024-05-27T11:31:21.493Z INFO  [batch:indexer] : Batch processing started for job 90912
2024-05-27T11:31:21.493Z ERROR [core:345681] : Connection timed out: to="shard-03"
2024-05-27T11:31:21.490Z DEBUG [node:engine1206] : Batch poceessing started for job 90912
2024-05-27T11:31:21.490Z ERROR [core:34501] : Connection timed out: to="shard-03"
2024-05-27T11:31:21.493Z INFO  [perf:cpu]    : CPU utilization: 87%
2024-05-27T11:31:21.590Z ERROR [orc:ch401] : OutOfMemoryError: Java heap space
```

SYSTEMIC DIAGNOSIS





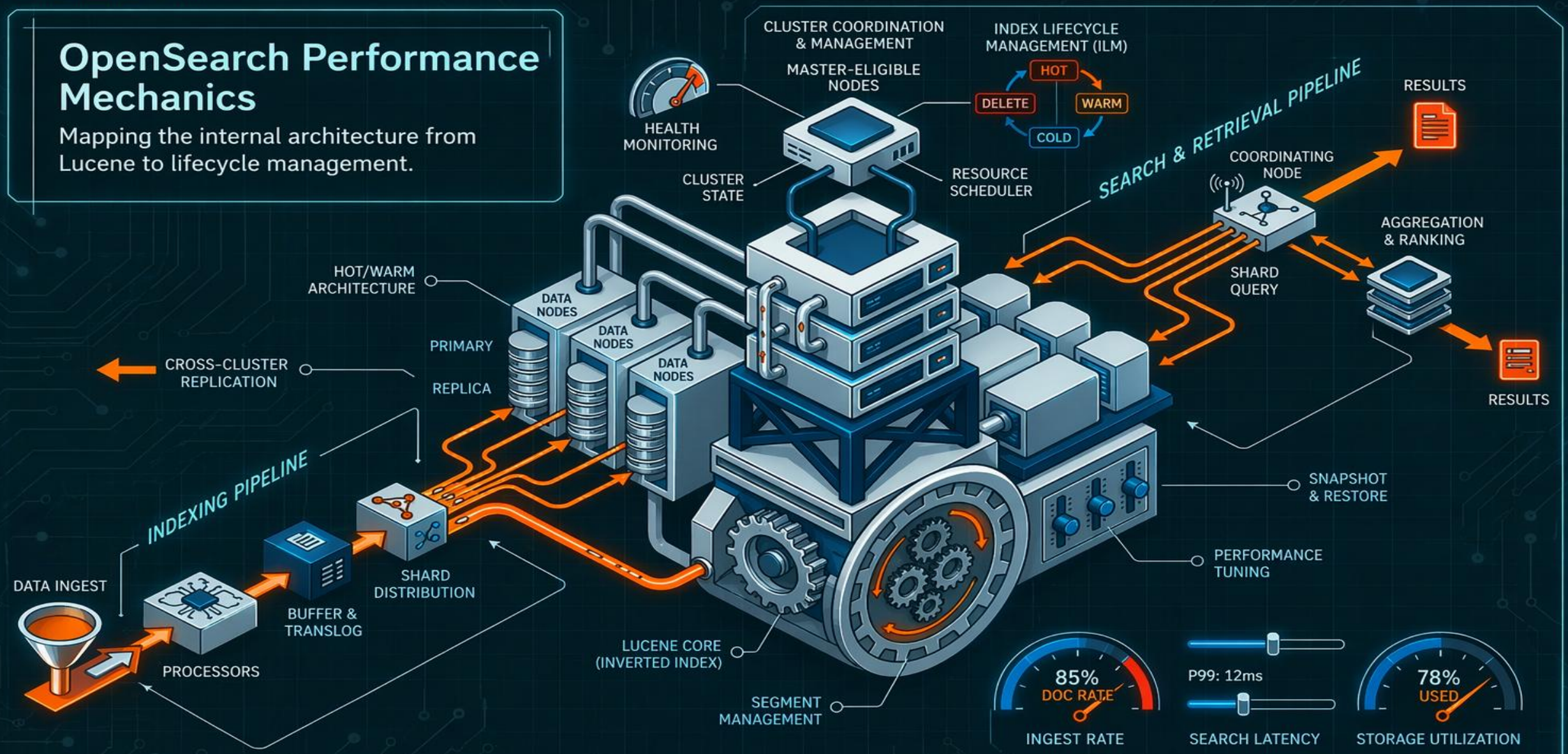
Takeaway Box

We are going to zoom in to the absolute micro-ecosystem core of the search engine, and progressively zoom out to see how tuning one knob imrpacts the entire machine.

LOAD_MODULE: true

OpenSearch Performance Mechanics

Mapping the internal architecture from Lucene to lifecycle management.



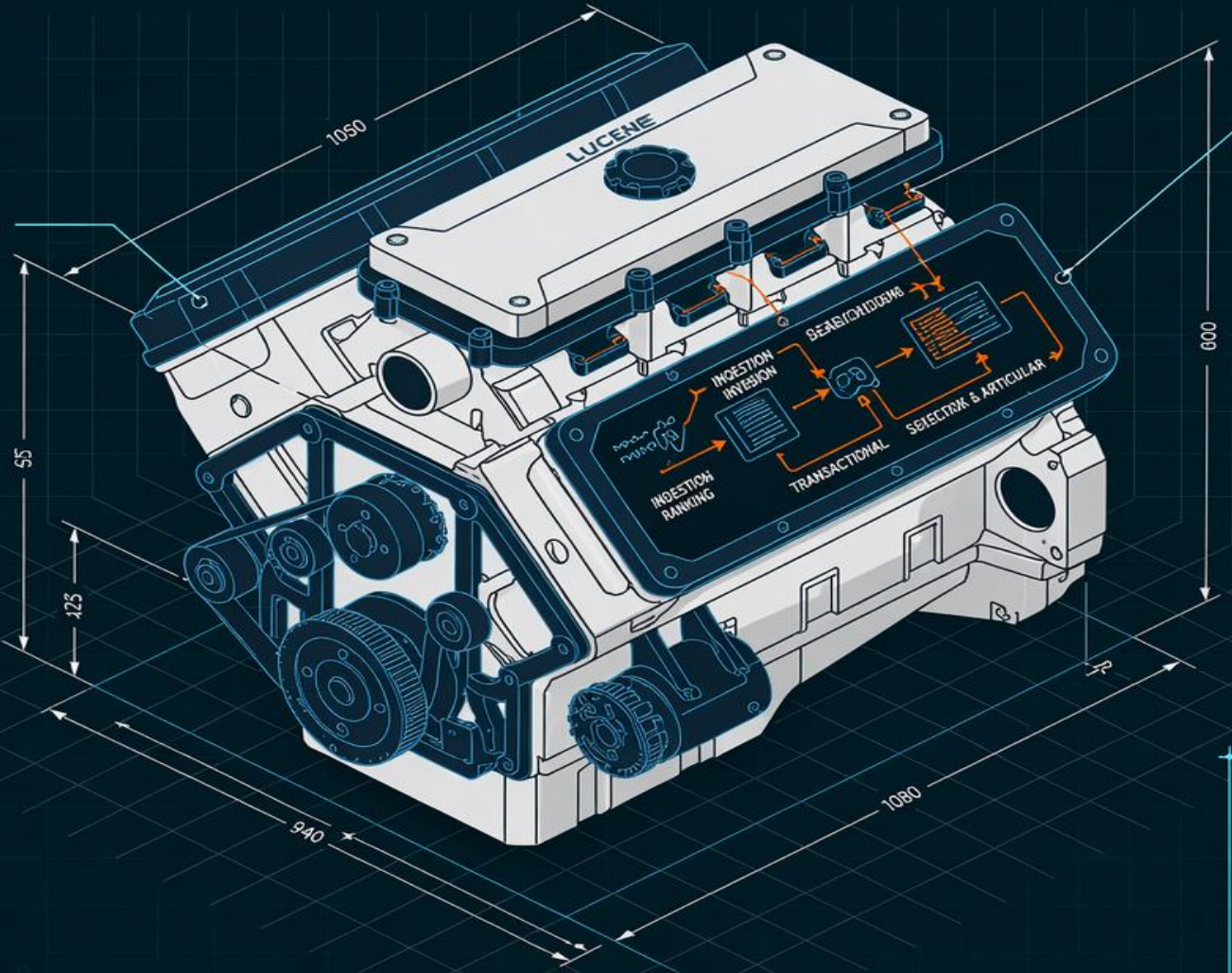
PRECISION SCALE: 1:500 UNITS

SCALE: 100M_LOGS/DAY

Lucene is the engine that does the actual search work.

Troubleshooting at scale often looks like a game of whack-a-mole—fixing isolated errors as they appear. However, true optimization requires a mental model of the low-level mechanics.

Core Unit: Apache Lucene

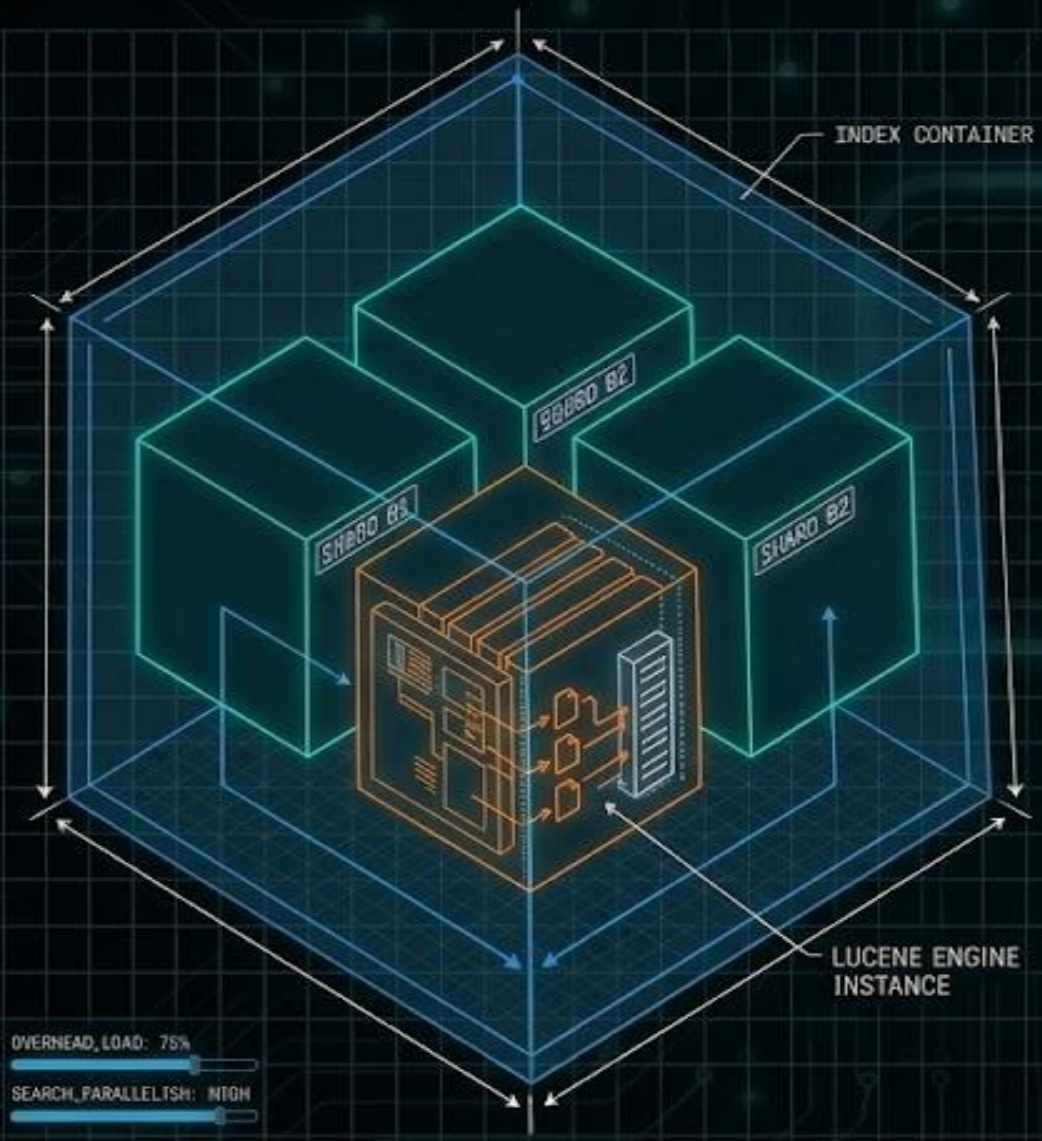


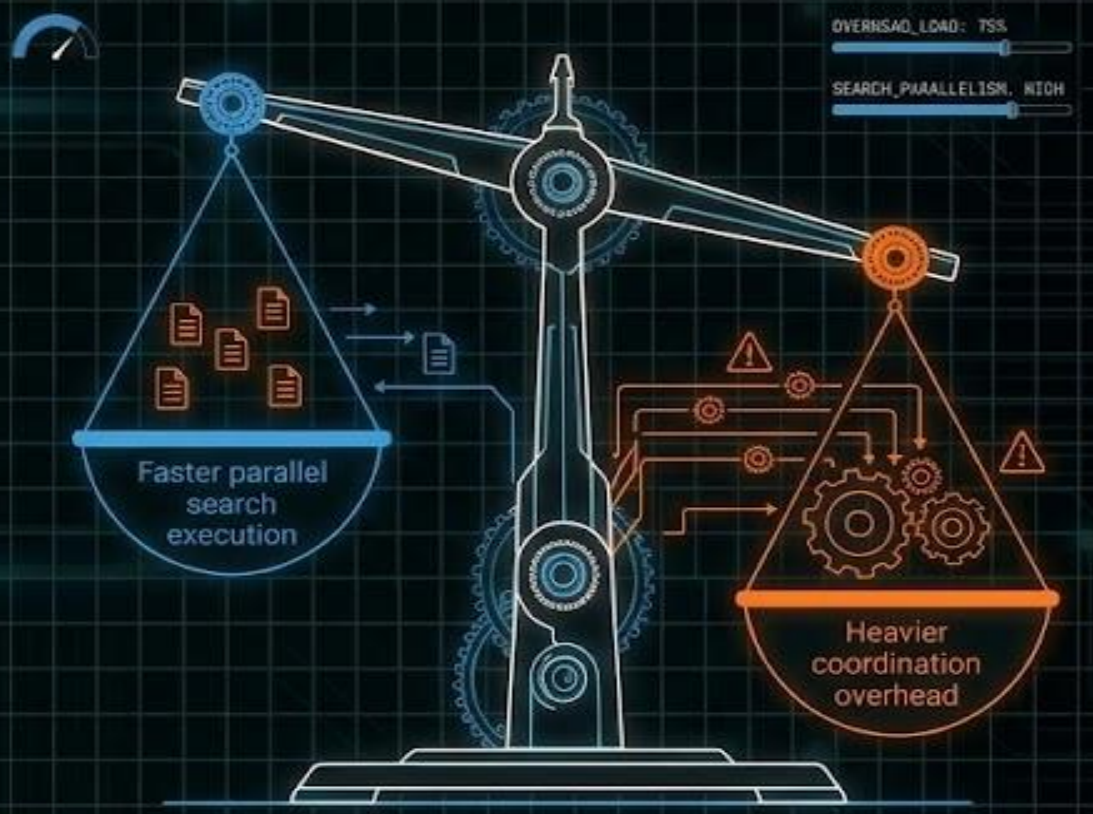
Function:
The foundational search engine powering the core search capabilities inside OpenSearch.

If OpenSearch is the chassis, the routing system, and the dashboard of a high-performance vehicle, Lucene is the internal combustion engine sitting under the hood.

Instead of one massive container, data is broken down into smaller boxes called Shards.

When managing massive amounts of information, storing everything in a single location creates an unsearchable monolith. Data is logically distributed into shards to make searching faster and highly parallel.





Core Rule

More shards = more coordination = slower queries.
The system must look in too many boxes.

Inside the shard, fast-arriving data is stored in fragments called Segments.

Mechanic: Data inside a shard is not kept as a single contiguous file. It is broken down into numerous individual segments.

Velocity Problem: When data ingestion is extremely fast (e.g., continuous log streams arriving every second), the system generates thousands of tiny segments.

ANALOGY:
It is exactly like trying to write down incoming information in 1,000 tiny notebooks instead of organizing it into 1 solid, well-indexed book.

SYSTEM METRIC:
SEGMENT COUNT > 100,000
(HIGH FRAGMENTATION WARNING)

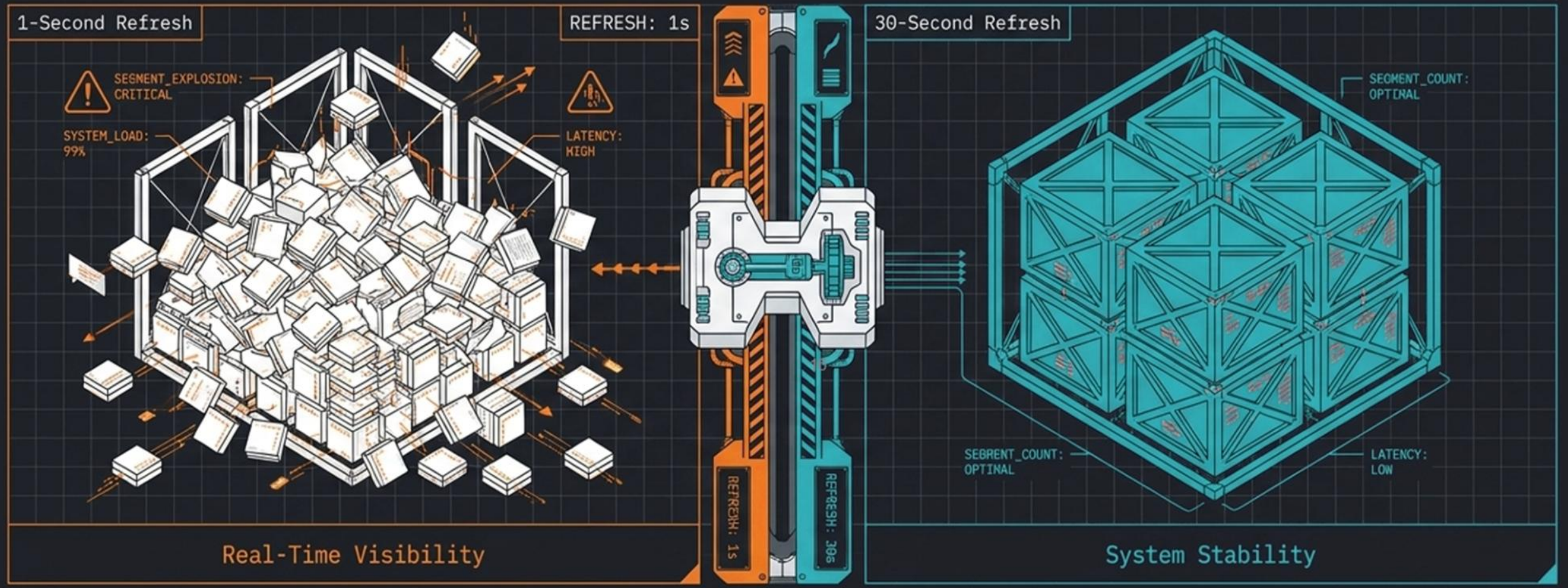
The Refresh Interval balances data visibility with segment explosion.

Mechanic:

New incoming data does not immediately become searchable. The Refresh Interval dictates how often the system updates to make new data visible to users.

Consequence:

A frequent refresh prioritizes real-time visibility but aggressively creates too many segments. It is like stopping to reorganize your entire room every single second—eventually, the system slows to a crawl.

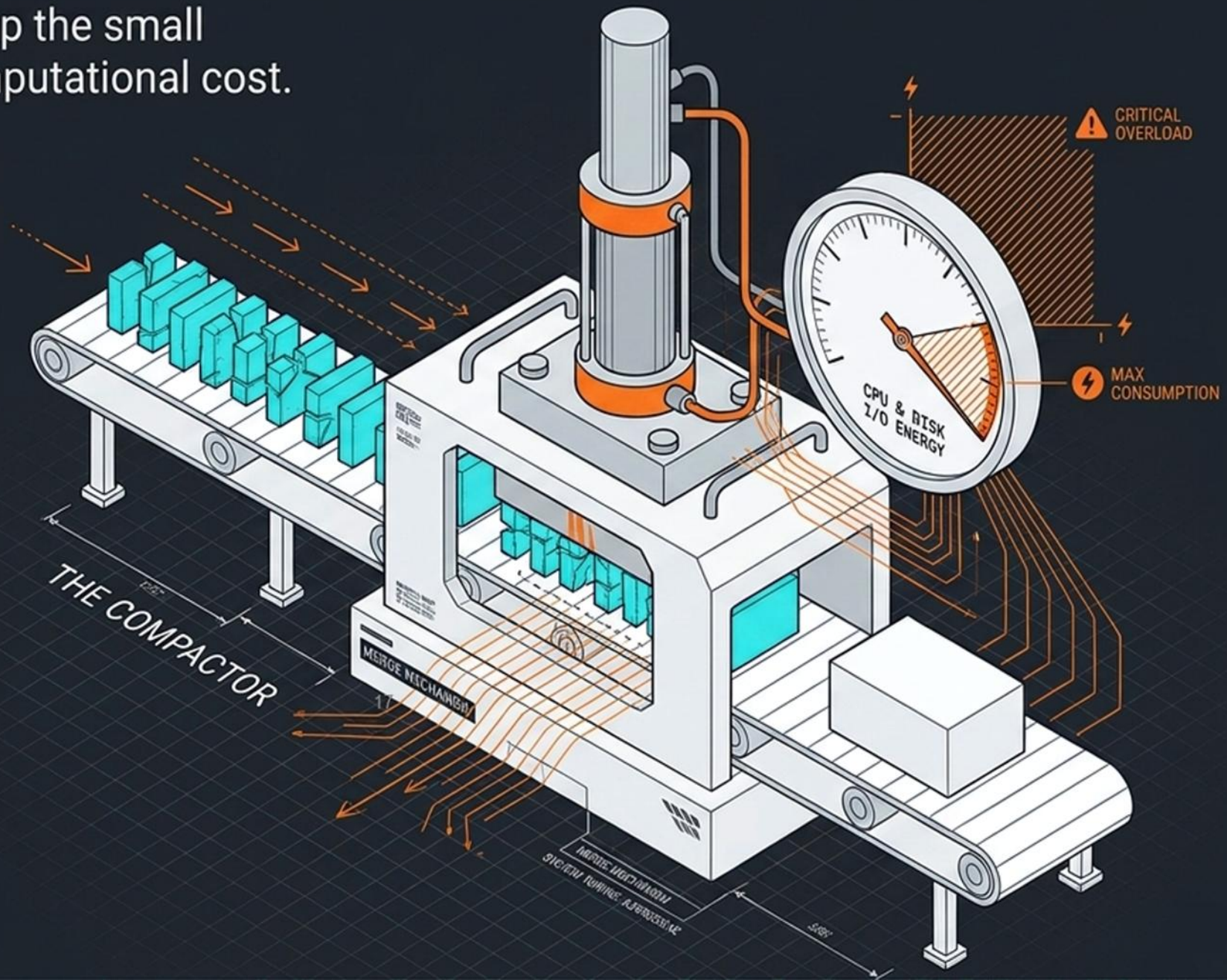


The Merge process cleans up the small segments, but at a high computational cost.

Mechanic: To prevent an abundance of tiny data pieces, a background merging process process constantly consolidates small segments into larger, more efficient ones.

Conflict: This "cleaning up of small notebooks" requires massive CPU and disk effort.

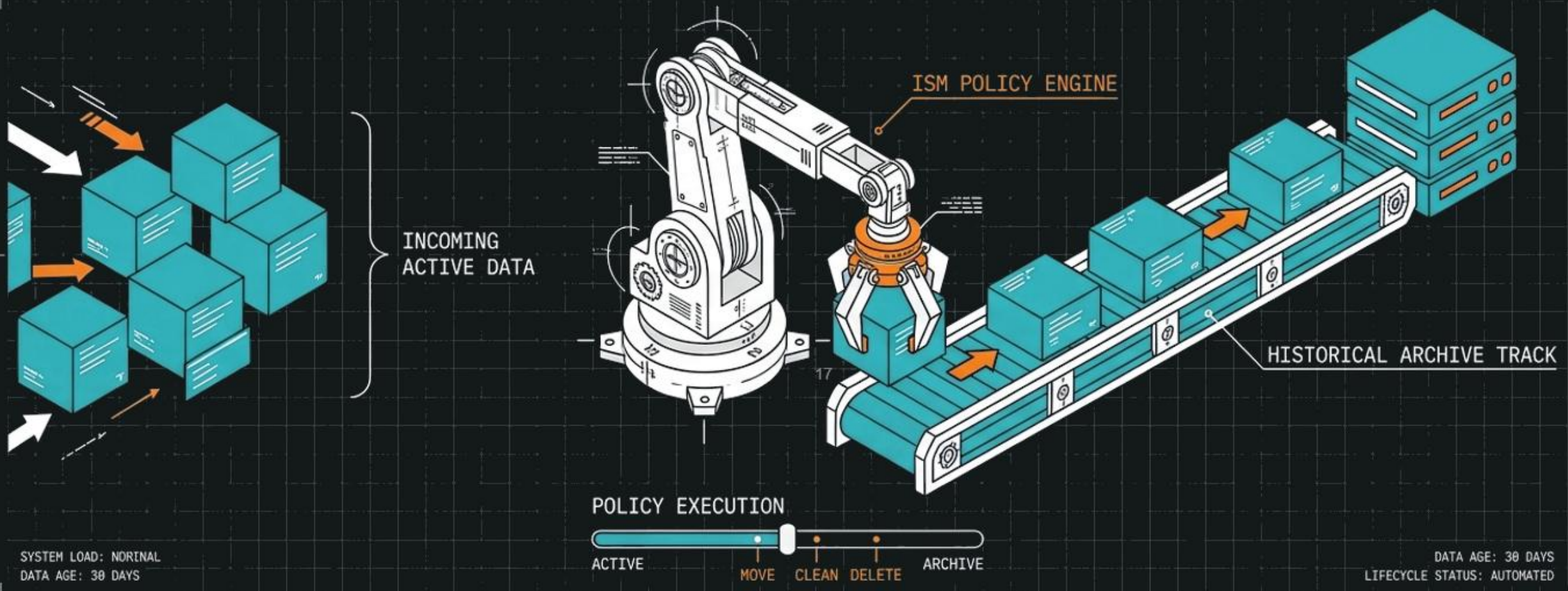
Risk: If the system is constantly forced to clean up tiny segments while a firehose of new data is still arriving, the engine reaches total compute overload.



Index State Management (ISM) acts as the automated housekeeper.

Mechanic: ISM provides automated data lifecycle management. It monitors the age and relevance of the data.

Function: Like an automated system continuously cleaning out old items and organizing new arrivals, ISM determines exactly when to move, clean, or delete data based on predefined policies.

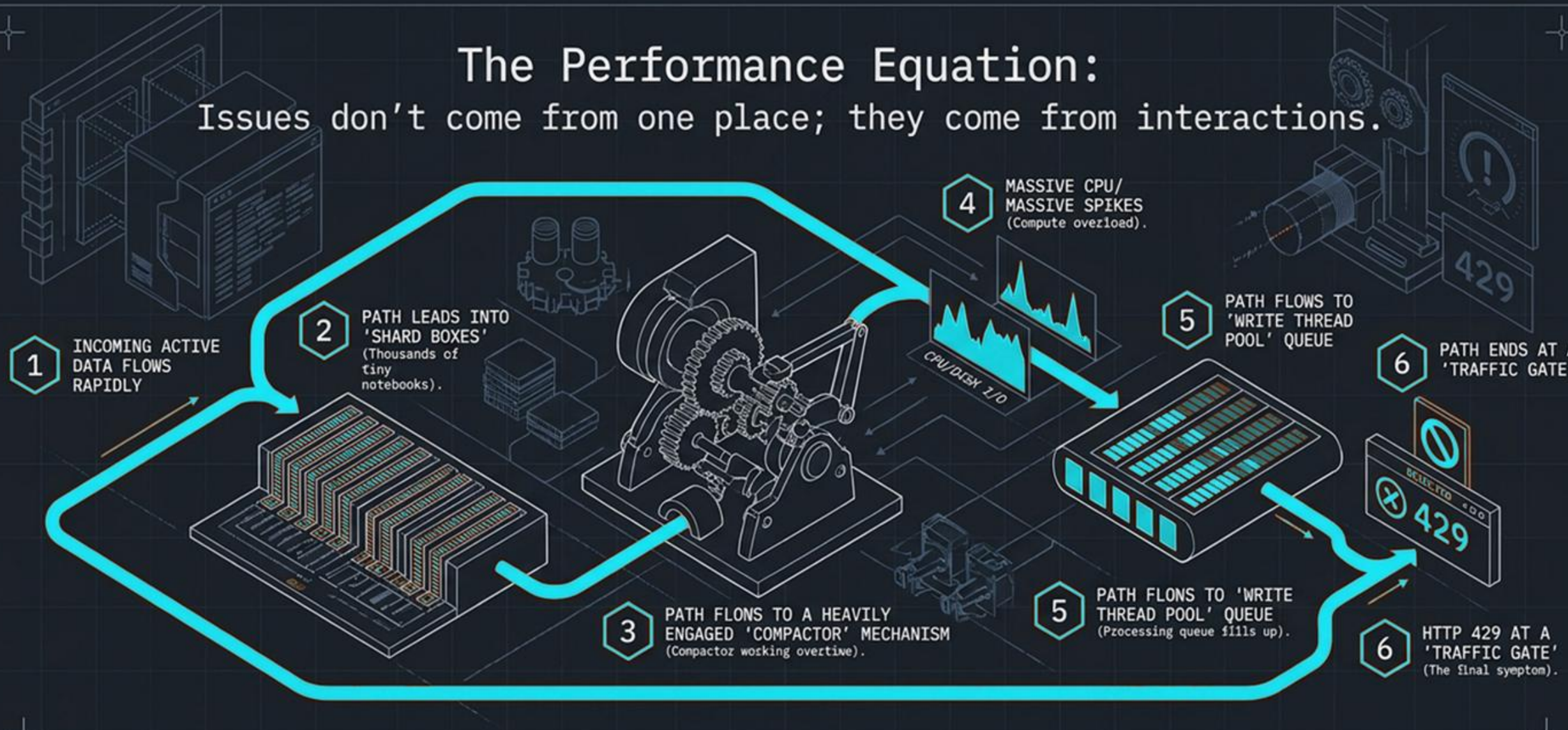


Distributing the lifecycle across Hot and Warm Tiers.

Dimension	Hot Tier	Warm Tier
Data Profile	Latest, highly active data	Older, less frequently accessed data
Read/Write Velocity	High velocity, rapid search	Slower retrieval, infrequent updates
Hardware / Cost	Fast, expensive compute/storage	Economical, high-density storage
Role	The active workspace	The accessible archive

The Performance Equation:

Issues don't come from one place; they come from interactions.



SYSTEM LOAD: CRITICAL

DATA AGE: N/A

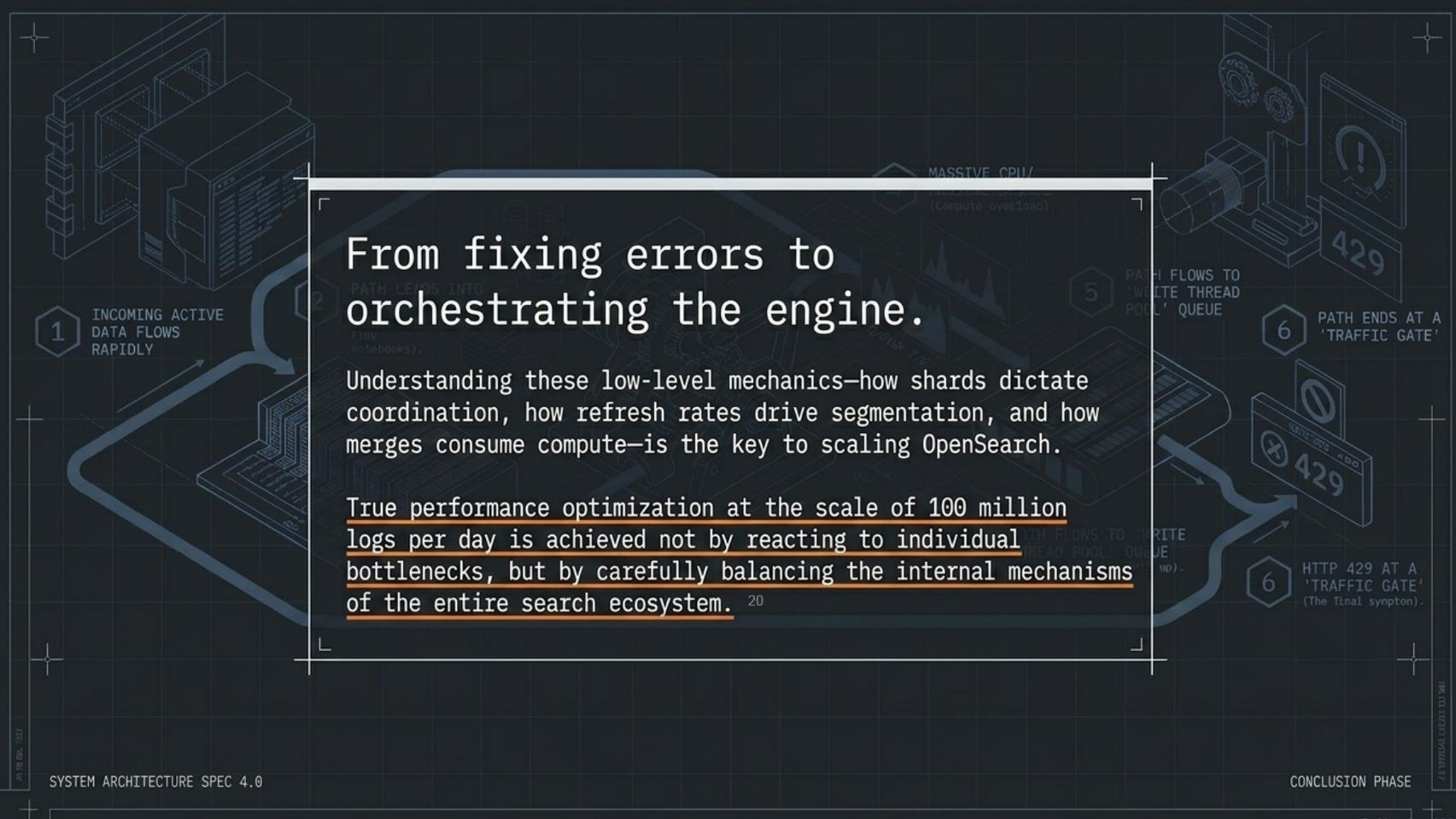
LIFECYCLE STATUS: OVERLOADED

SYNTHESIS INSIGHT: Fixing a 429 traffic error often doesn't mean increasing thread pools—it means slowing down the refresh interval.

SYSTEM LOAD: CRITICAL

DATA AGE: N/A

LIFECYCLE STATUS: OVERLOADED



From fixing errors to orchestrating the engine.

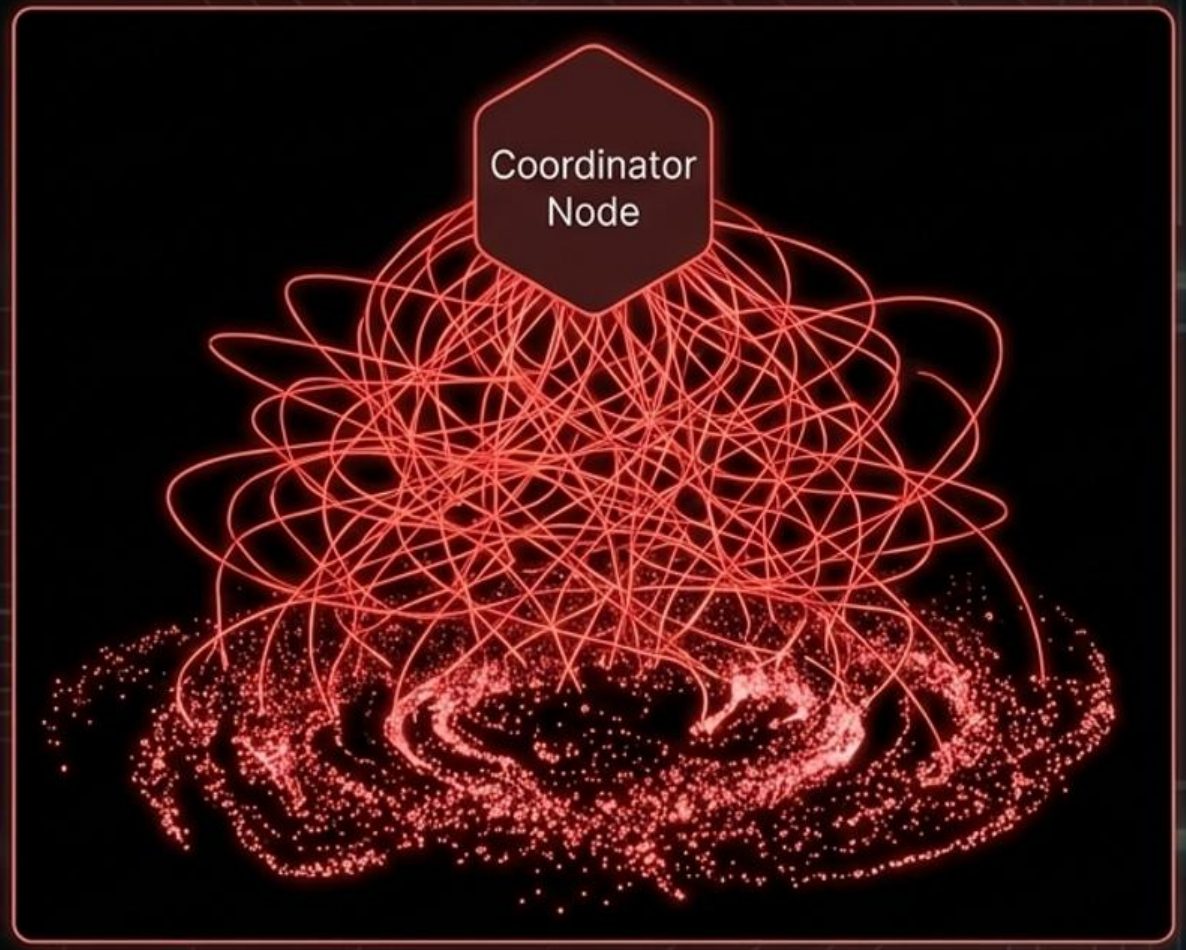
Understanding these low-level mechanics—how shards dictate coordination, how refresh rates drive segmentation, and how merges consume compute—is the key to scaling OpenSearch.

True performance optimization at the scale of 100 million logs per day is achieved not by reacting to individual bottlenecks, but by carefully balancing the internal mechanisms of the entire search ecosystem. [20]

OpenSearchCon INDIA

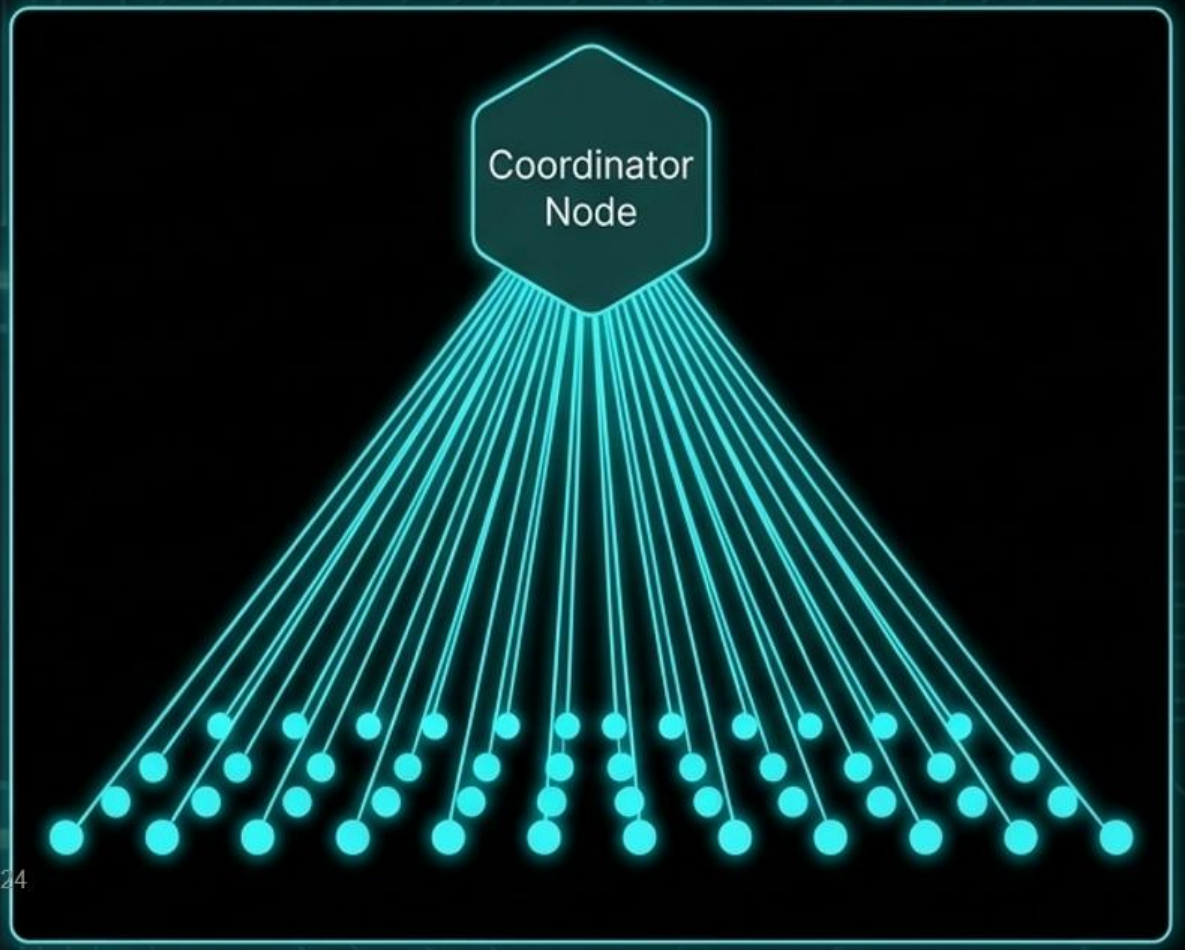
Why systems break at scale?

Root Cause 1: Shard explosion and the Fan-Out Tax.



Default Drift

- Default 1 shard/index × daily rollover = 200+ active shards.
- Math: 200 shards × 1ms coordination = 200ms added to every query before data is touched.



Right-Sized

- 30 right-sized shards/day.
- Math: Predictable fan-out, zero latency penalty.

Root Cause 2: The Merge Storm Collision.

The Mechanics:

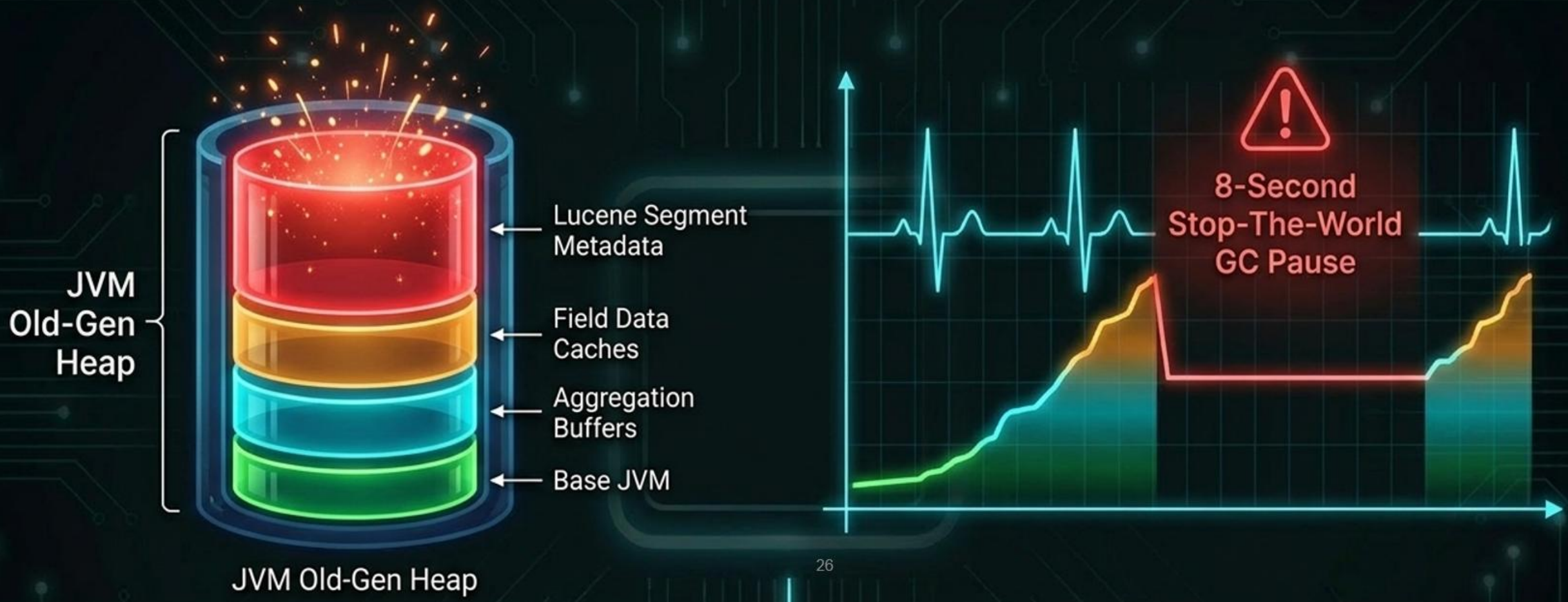
At 1,150 docs/sec, a 1s refresh creates 3,000+ segments before merges can catch up.

The Result:

Merging a 10GB segment competes for the same disk write path as bulk indexing.

Throughput drops 40-60% during merge windows, causing queue buildup and 429 rejections.

Root Cause 3: JVM Heap and GC Instability.



The Mechanics:

Unchecked segment metadata and fielddata caches fill the JVM old generation.

The Result:

When old-gen exceeds 85%, the JVM triggers a stop-the-world collection. Spikes of up to 8 seconds are not random—they repeat directly on the merge cycle.

CRISIS

The Compounded Crisis

Signal	Before tuning	Visible impact
Shard count	315 active	+200ms per query
Segment count	4,200 / index	96% merge CPU consumed
Write queue	165 queued	429 in 4 minutes
Heap usage	82%	8s GC pauses

Mean time to diagnosis 25 min → <5 min

OpenSearchCon INDIA

Solution Approach / Controls

Control 1: Right-Size Shards by Daily Ingest Volume

Target 10-50 GB per shard to balance parallelism with coordination overhead.

```
Shards/Day = Daily Ingest GB / Target Shard Size
```

```
100M docs/day (~900 bytes/doc) = 90 GB/day
90 GB / 30 GB Target = 3 Primary Shards/day
```

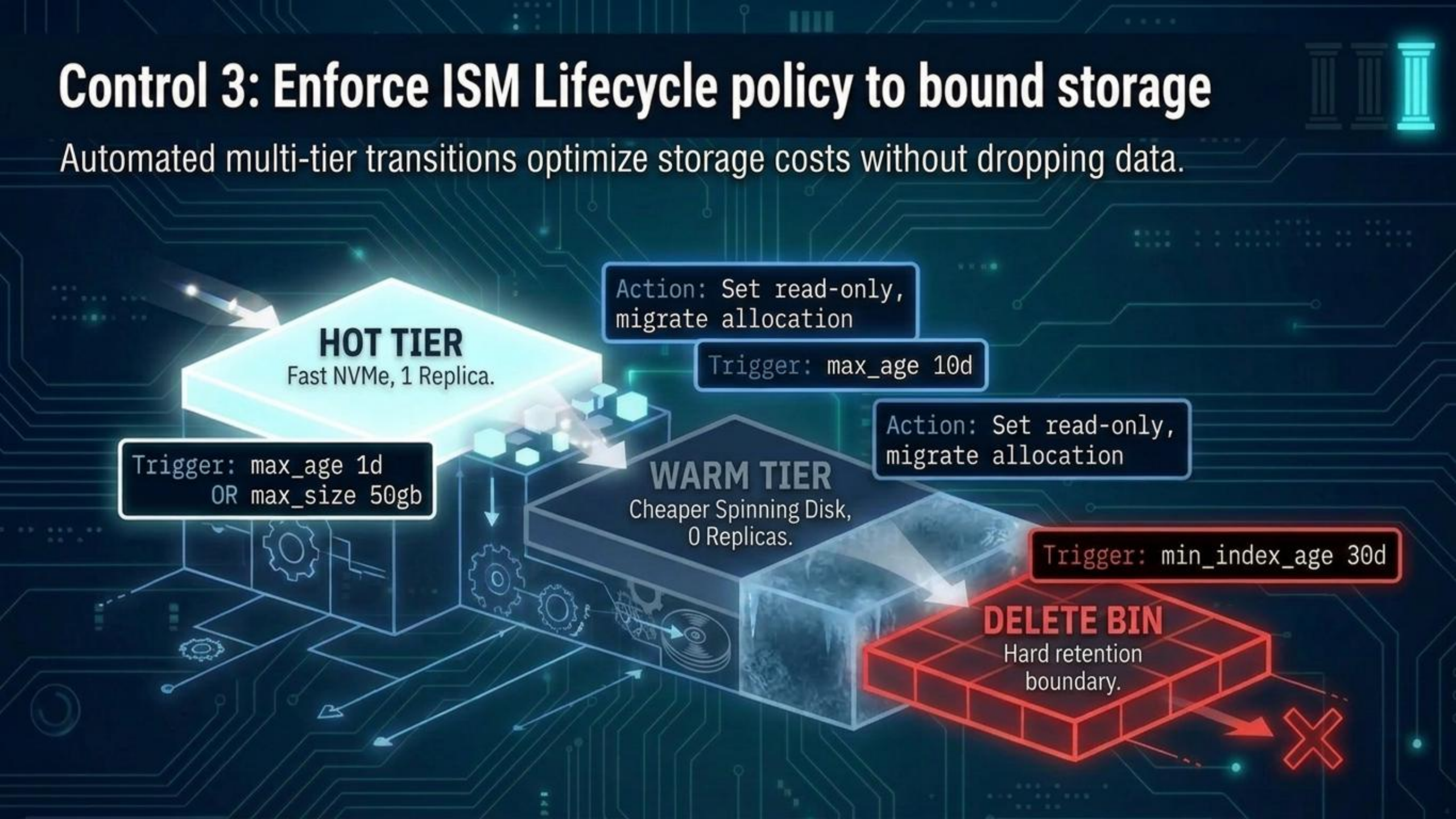
Result: 30 shards/day vs 200+ default shards.

Control 2: Constrain segment creation and bulk request overhead.

Dial 1: Bulk Sizing	Dial 2: Peak Ingest Refresh	Dial 3: Investigative Refresh
`5,000 - 10,000 docs/request`	`index.refresh_interval = 30s`	`index.refresh_interval = 1s`
Concurrent threads = CPU cores ÷ 2	Drops segment creation from 4,000/hr to 38/hr. Merge thread stays ahead.	**Restored dynamically only during active incident response.**

Control 3: Enforce ISM Lifecycle policy to bound storage

Automated multi-tier transitions optimize storage costs without dropping data.



OpenSearchCon INDIA

Demo Time!

The KPI Scorecard: Your Platform Truth Set

Dashboards lie. The cluster API does not.

KPI	Target SLO	Alert Threshold	API Command
Ingest Throughput	> 1,000 docs/sec	queue > 50	`_cat/thread_pool/write`
Rejection Rate	< 0.5%	> 1% for 5 min	`_cat/thread_pool/write?v&h=rejected`
p99 Query Latency	< 1.5 s	> 3 s	`_nodes/stats/indices`
GC Overhead	< 10% CPU	> 20% CPU	`_nodes/stats/jvm`
Segment Count	< 100/index	> 500/index	`_cat/segments/*`
ISM Transition	100% Success	Any state stuck	`_plugins/_ism/explain/*`

Capacity and Incident Runbook Triggers

Every threshold has an action. Write it down before the incident.

```
IF: queue_depth > 50 for 2m
THEN: Reduce bulk concurrency 50%,
raise refresh_interval to 60s.
```

```
IF: p99 > 2s for 10m AND
segment_count > 500
THEN: Force-merge cold indices, reduce
query fan-out.
```



```
IF: jvm_old_gen > 85%
THEN: Inspect fielddata circuit breaker,
reduce replica count on warm indices.
```

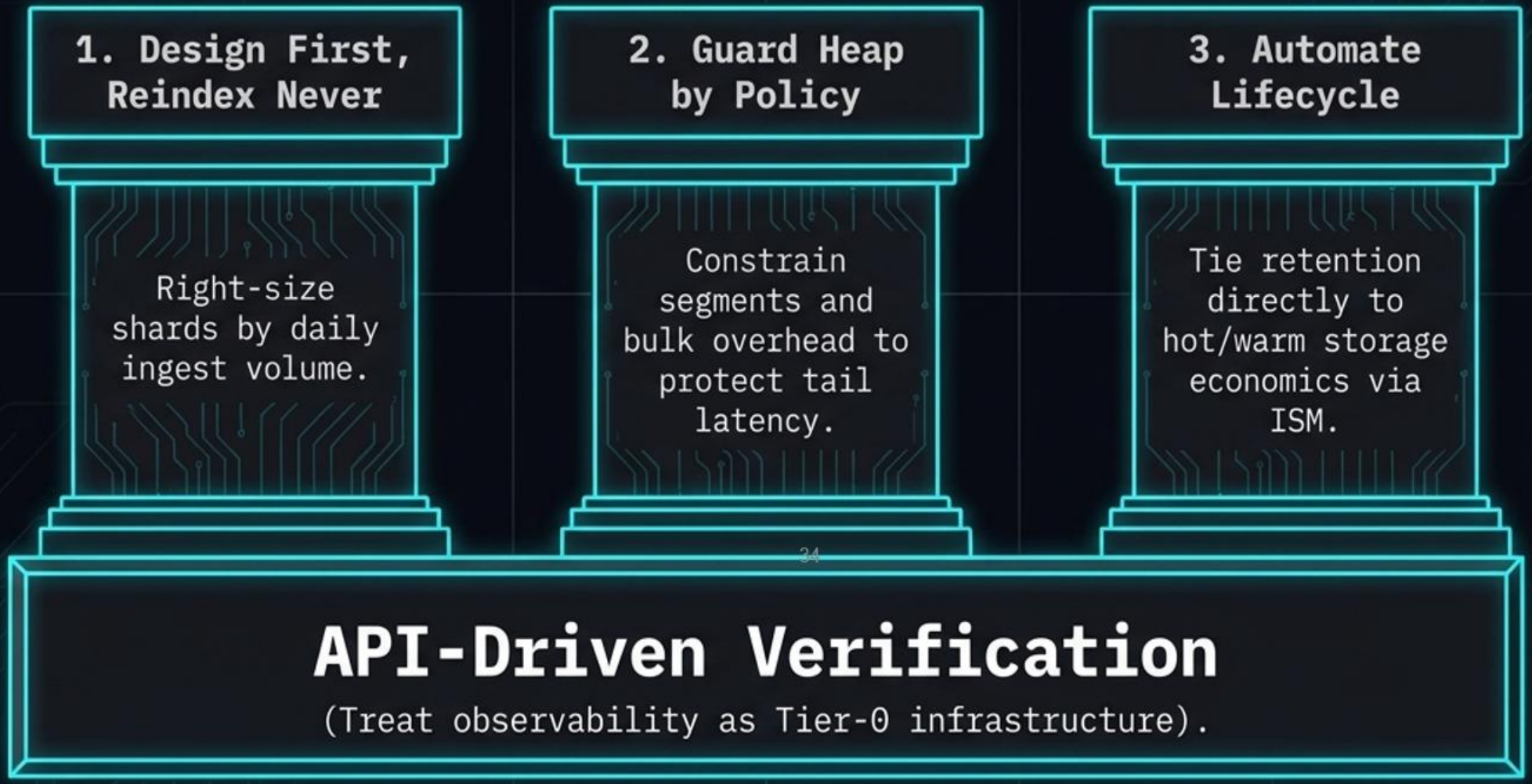
Quantifiable stability proves the return on engineering effort.

The Return on Engineering

Metric	Before	vs.	After	Control Applied
Write Rejections:	3-5% peak	➡	**< 0.5% sustained**	(Control: Bulk size + 30s refresh)
p99 Latency:	3-8s during merges	➡	**< 1.5s**	(Control: Shard sizing + Segment limits)
Segment Count:	2,000-5,000	➡	**< 100**	(Control: Force-merge on warm)
Time-to-Diagnosis:	25 min manual	➡	**< 5 min**	(Control: KPI API checks)
Storage Economics:	Unbounded	➡	**-40% on warm tier**	(Control: 1 -> 0 replicas via ISM)

> // end_playbook. cluster_status: optimal.

Scale requires a shift from configuration to distributed systems engineering.



Summary: The Scaling Equation

The Chain Reaction

Performance issues at 100M logs/day do not stem from a single configuration parameter. It is a chain reaction:

Fast Ingestion + Low Refresh Interval

- Too many Segments
- High Merge activity
- CPU & Disk exhaustion
- Write Thread Pool overflow
- HTTP 429 Errors

The Golden Rule

- Tune refresh intervals
- Size shards to 30-50GB
- Automate lifecycle transitions using ISM
- Balance hot/warm tiers to keep the system resilient!


OpenSearchCon
INDIA