TECHNICAL WHITEPAPER

Project Acumen

Universal Safe Query Compilation Framework

Empirical Performance Characterization Across Two Hardware Tiers

---

Empirical Performance Characterization

Reddy and Reddy Family Office (i.f.)

Registered Office: GIFT City, IFSC, Gujarat

Benchmark Date: 18 July 2026 | Build: `21e4d9f` | Report Date: 23 July 2026

Companion Documents

Empirical Performance Analysis Report (18 Jul 2026)

Safety-Kernel Empirical Analysis Report (19 Jul 2026)

Strategic Technology Assessment — Financial Infrastructure Applications

1 Abstract

This whitepaper presents a controlled, instrumented characterization of the Acumen FHIR Query Translation Engine — a Universal Safe Query Compilation Framework integrating a pre-execution Safety Kernel, an encapsulated Cryptographic Integrity Engine (SHA-256 hash chains, Merkle trees, ECDSA P-256), and a schema-aware Dynamic Indexing Engine. All quantitative claims derive from three independent measurement instruments applied on 18 July 2026 against build `21e4d9f` on two hardware tiers: (i) a production-scale cloud VM (AMD EPYC-Genoa, 15.2 GB, Linux) serving a 6.46-million-document clinical corpus, and (ii) a consumer laptop (AMD Ryzen 7 3750H, 5.9 GB, Windows 11) serving a 77-thousand-document corpus (83.6× smaller).

Principal measured results: indexed-search p95 latency of 24–83 ms across both tiers with P99/P50 tail ratios of 1.10–1.30×; per-query server CPU cost flat in concurrency (≈ 12–17 ms on cloud hardware from 1 to 50 concurrent users); fixed allocation footprint (GC pause ≈ 1 ms per query at every load level); logarithmic data scaling (83.6× corpus increase moves worst-case indexed p95 by only 2.4×); 100 % clean absorption of 50 302 labelled adversarial requests with zero false positives; and full write-surface execution with atomic ECDSA-signed, hash-chained audit entries at 21–113 ms p95. Known limitations — including seven guard-passthrough patterns, a persistent VM heap-state transition, single-thread binding at ≈ 135–147 req/s per process, and tier-dependent rejection economics — are reported with the same evidentiary weight as the positive results.

Every numerical claim is traceable to a cited benchmark JSON artefact and is reproducible via the harnesses documented in the companion Empirical Performance Analysis and Safety-Kernel reports.

2 Introduction and Problem Statement

Modern data-intensive systems that must simultaneously satisfy cryptographic non-repudiation, bounded computational cost, and high query throughput confront a set of structural constraints. Distributed-ledger architectures achieve mathematical integrity at the expense of throughput and latency; conventional relational engines achieve scale at the expense of intrinsic integrity guarantees and admit unbounded pathological queries. The resulting engineering gap is particularly acute in regulated domains (healthcare FHIR, financial transaction ledgers, sovereign identity systems) where both auditability and predictable resource consumption are non-negotiable.

Project Acumen addresses this gap by embedding three orthogonal mechanisms inside a single query-compilation pipeline:

1. **Safety Kernel** — pre-execution admission control that rejects pathological or malicious inputs before any database work is performed;
2. **Cryptographic Integrity Engine** — mandatory SHA-256 hash-chained, ECDSA P-256-signed `ResourceHistory` entries committed atomically with every mutation;
3. **Dynamic Indexing Engine** — schema-aware, priority-based index lifecycle that adapts to evolving search-parameter definitions without manual DBA intervention.

This whitepaper confines itself to the empirical performance and safety evidence. Architectural claims beyond the measured record are deferred to the companion Strategic Technology Assessment.

3 Test Environment and Instrumentation

3.1 Hardware Tiers and Corpora

Both tiers executed the identical server build (`21e4d9f`), identical harness configuration, and identical attack/valid taxonomy on 18 July 2026. Resident document counts were obtained by full `countDocuments()` scans of the live databases.

Table 1: Test environment specifications.

Component	Cloud VM	Laptop (QUASAR)
CPU	AMD EPYC-Genoa (8 vCPU)	AMD Ryzen 7 3750H (8 cores)
RAM	15.2 GB	5.9 GB
OS / Node.js	Linux / v22.23.1	Windows 11 / v22.19.0
Storage Engine	MongoDB replica set (rs0)	MongoDB replica set (rs0)
Clinical-core docs	6 458 565 (83.6×)	77 211

3.2 Measurement Instruments

Three mutually independent instruments were employed; the report never mixes their outputs:

- **HTTP-observable metrics (k6)** — latency percentiles, throughput, error rate, SLA pass/fail.
- **External server-process CPU sampling** — `/proc/[pid]/stat` (Linux) or `Get-Process TotalProcessorTime` (Windows), attributed against server-side query counts.
- **In-process telemetry** — V8 `PerformanceObserver` for GC events/pauses, `monitorEventLoopDelay`, `process.resourceUsage()` for context switches and I/O.

4 Indexed Query Latency

Core indexed-search scenarios were measured after 20 warmup iterations (100 measured iterations each). Despite the VM serving an 83.6× larger corpus, p95 latencies remain in the same order of magnitude.

Table 2: Indexed-search p95 latency. Sources: `benchmark-2026-07-18T10-20-50-{ssd,hdd}.json` (laptop), `benchmark-2026-07-18T13-58-39.json` (VM).

Scenario (p95)	Laptop SSD	Laptop HDD	VM NVMe
`STRING_SEARCH_EXACT`	34.13 ms	27.87 ms	83.07 ms
`TOKEN_SEARCH_SINGLE`	73.10 ms	47.03 ms	31.77 ms
`TOKEN_SEARCH_OBSERVATION`	44.04 ms	36.62 ms	24.45 ms
`DATE_RANGE_SEARCH`	53.80 ms	44.87 ms	30.37 ms

4.1 Tail-Latency Ratios

P99/P50 ratios on indexed operations are bounded between 1.10× and 1.30× — substantially tighter than the 5–10× ratios typical of unmanaged database-backed services. Bounded query complexity is the structural cause.

4.2 Logarithmic Data Scaling

An 83.6× increase in clinical-core documents raises the worst-case indexed-search p95 by only 2.43× (`STRING_SEARCH_EXACT`) and reduces latency on tightly-indexed token and date scenarios. This is the expected signature of B-tree index utilization, formally $O(\log n)$ rather than $O(n)$.

Deep-offset pagination exhibits cost invariance: `PAGINATED_PAGE_N` p50 equals `PAGINATED_PAGE_1` p50 to within 2 % on the production-scale corpus.

5 Concurrency Model and Per-Query Server CPU

A fixed-workload concurrency sweep (identical valid-only query mix, 1 → 50 concurrent users, 20 s warmup + 90 s measured window per level) isolates the effect of concurrency from workload composition. Per-query CPU is measured against the server process itself.

5.1 Flat Per-Query Cost

On both tiers, per-query server CPU is essentially invariant with concurrency:

- **Laptop:** 23.2–26.9 ms across the full 1–50 range (14.7 % spread; fitted power-law exponent $|\alpha| \leq 0.055$ at 95 % confidence).
- **VM:** flat within each heap state — 11.7–13.5 ms (low-GC, 1–10 users) and 17.0–17.2 ms (high-GC, 15–50 users; 1.2 % spread).

Concurrency neither amortizes nor super-linearly inflates the CPU cost of a query. The engine is single-JavaScript-thread-bound; throughput saturates at ≈ 57–63 req/s (laptop) and ≈ 135–147 req/s (VM). Beyond the knee, additional concurrent users purchase only queueing latency.

5.2 Energy Model

Under a linear power model $P = P_{\text{idle}} + (P_{\max} - P_{\text{idle}}) \cdot U$ with an active-power assumption of 8 W:

$$E_{\text{query}}^{\text{VM}} \approx 8\,\text{W} \times (12.1\text{–}17.2) \times 10^{-3}\,\text{s} \approx 97\text{–}138\,\text{mJ}, \tag{1}$$

$$E_{\text{query}}^{\text{Laptop}} \approx 8\,\text{W} \times 25 \times 10^{-3}\,\text{s} \approx 200\,\text{mJ}. \tag{2}$$

Because per-query CPU is flat, per-query energy is likewise flat in concurrency (subject only to the discrete VM heap-state transition). These figures are modelled from measured CPU time, not RAPL; they quantify the genuine resource cost of the translation-and-safety pipeline.

6 In-Process Runtime Telemetry

6.1 Fixed Allocation Footprint

GC events per query (0.20–0.23 laptop; 0.29–0.33 VM) and GC pause per query (0.92–1.26 ms laptop; 0.80–1.00 ms VM) are constant across a 50× concurrency range. Collection work scales linearly with query volume, not with load level. GC accounts for 4–7 % of per-query CPU — bounded overhead,

not a scaling risk. Peak RSS exhibits no growth trend across eight consecutive 90-second saturation windows on either tier.

6.2 Emergent VM Heap-State Transition

Between the 10- and 15-user windows the VM runtime entered a persistent high-GC operating point: major-GC activity rose $\approx 7\times$, peak RSS fell 34 %, and per-query CPU stepped from ≈ 12.1 ms to ≈ 17.2 ms. Client-visible throughput changed by only -4 % because the additional collection work ran on background GC threads. The transition exhibited hysteresis — it did not revert when load returned to a single user. Capacity planning on the VM must budget the high-GC state.

6.3 Latency Under Load Is Event-Loop Queueing

Mean event-loop delay measured inside the server rises monotonically with concurrency (28 → 304 ms laptop; 22 → 166 ms VM) and tracks mean request latency in near-constant proportion. Requests spend their added time waiting for the single JavaScript thread, not executing more slowly once scheduled.

7 Safety Kernel — Adversarial Absorption

A purpose-built, fully labelled adversarial taxonomy (chain-depth 12–25, value-count 700, malformed/operator-injection) was driven against live servers on both tiers, interleaved with valid control traffic and a sustained attack-storm phase. Every request carried its expected outcome; the rejection population is therefore classified by construction.

7.1 Absorption Correctness

Table 3: Absorption correctness. Source: `coreCorrectness` in `kernel-benchmark-2026-07-18T12-29-22.json` (VM) and `...T14-00-48.json` (laptop).

Metric	Cloud VM	Laptop
Hostile requests	26 838	23 464
Absorbed (clean 4xx + OO)	26 838 (100 %)	23 464 (100 %)
Hard leaks (5xx / timeout)	0	0
False positives (valid blocked)	0	0
Valid control queries served	2 501	2 177

7.2 Cost Asymmetry (Tier-Dependent)

On server-grade hardware a rejection costs 3.3–3.8 ms median against an 18.5 ms valid baseline — 4.9–5.6× cheaper to refuse than to serve. On the consumer tier the asymmetry inverts (rejection $\approx 1.15\times$ the valid baseline). The favourable economics case rests exclusively on data-centre deployment.

7.3 Traffic Isolation Under Attack Storm

Under sustained concurrent attack, cloud-tier valid traffic showed no measurable median degradation (13.0 ms under attack vs 18.5 ms quiet baseline). Consumer-tier valid traffic degraded 3.74×. Server

Table 4: Cost asymmetry per tier. Ratio = valid median / rejection median (values > 1 mean rejection is cheaper).

Class	VM Med.	VM Ratio	Lap. Med.	Lap. Ratio
Valid baseline	18.52 ms	1.00×	9.97 ms	1.00×
chain-depth reject	3.33 ms	5.56× cheaper	11.37 ms	0.88×
value-count reject	3.79 ms	4.89× cheaper	11.78 ms	0.85×
malformed reject	3.44 ms	5.39× cheaper	11.42 ms	0.87×

heap peaked at 92.3 % (cloud) / 96.2 % (consumer). Correct rejection is therefore not denial-of-service immunity; upstream edge defence remains necessary against volumetric attack.

7.4 Guard Gaps (Un-Absorbed Inputs)

Gap probes document the perimeter boundary. Four (cloud) / five (consumer) candidate patterns pass the guards and execute as legal-but-expensive queries. On the production-scale corpus one `include/revinclude` fan-out class ran to the 20.0-second client timeout — a genuine DoS vector whose cost scales with corpus size. These patterns are excluded from all absorption figures and remain open optimization targets.

8 Write Path and Cryptographic Integrity

Every mutation (create, conditional create/update/delete, transaction bundles, batches) is executed through a single versioning gateway that records an ECDSA P-256-signed, SHA-256 hash-chained `ResourceHistory` entry as part of the same ACID transaction. A failure to sign or record history rolls the resource mutation back; a resource never persists without its audit entry.

Integrity envelope (verified against the live database):

$$H_n = \text{SHA-256}(\text{StableJSON}(\text{State}_n) \parallel H_{n-1}) \tag{3}$$

with $H_0 = \text{SHA-256}(\text{StableJSON}(\text{State}_0))$. Any modification of version k invalidates the chain for all subsequent versions. The full write surface executes at 21–113 ms p95 on the cloud tier; history reads at 7–16 ms p95. Twelve of twelve transaction-rollback scenarios (including concurrent conflict detection and circular-dependency rejection) pass.

9 Known Limitations and Optimization Targets

The empirical record is explicit about its own boundaries:

1. **Guard passthroughs** — seven candidate attack patterns (including production-scale fan-out to 20 s) remain unabsorbed.
2. **Instrumentation coverage** — only ≈ 41 % of rejections are attributed to named server-side counters; the remainder occur at the uncounted validation layer.
3. **Single-thread binding** — per-process ceiling ≈ 135–147 req/s (cloud); horizontal fleets are required for higher throughput.

4. **VM heap-state transition** — persistent high-GC regime raises per-query CPU $\approx 42\,\%$ under sustained load.
5. **Stable SLA breaches** — terminology-expansion cascades (up to $\approx 8\,$s on production corpus) and two date-prefix operators remain optimization targets.
6. **Tier-dependent rejection economics** — cost asymmetry favourable only on server-grade hardware.

10 Mathematical Cost Model Summary

The measured behaviour admits a compact formal model:

- **Per-query CPU:** $C(q) = C_0(\text{hardware, heap-state})$ — independent of concurrency for admitted queries.
- **Latency under load:** $L(n) = C(q) + Q(n)$ where $Q(n)$ is event-loop queueing delay, measured to rise linearly with concurrent users beyond the per-core ceiling.
- **Data scaling:** $L_{\text{indexed}}(N) = O(\log N)$ — empirically confirmed by $83.6\times$ corpus increase producing $\leq 2.4\times$ p95 change.
- **Admission control:** any query with complexity score $\Theta > \Theta_{\text{safe}}$ or depth > 11 is rejected in $< 100\,$ms before database contact.
- **Integrity:** every committed mutation satisfies $H_n = \text{SHA-256}(\text{StableJSON}(S_n) \parallel H_{n-1})$ and carries a valid ECDSA-P256 signature under the active `keyId`.

Infrastructure dimensioning therefore reduces to two measured constants: Peak QPS $\times\, C(q) \times$ safety margin, with horizontal process count set by the measured per-process ceiling.

11 Conclusion

On a controlled, fully instrumented workload spanning two hardware tiers and an $83.6\times$ corpus-size differential, the Acumen engine demonstrates:

1. Predictable, low indexed latency (p95 24–83 ms, tail ratios 1.10–1.30×) with logarithmic data scaling;
2. Concurrency-invariant per-query server CPU ($\approx$ 12–17 ms cloud, $\approx$ 23–27 ms laptop) and a fixed allocation footprint ($\approx$ 1 ms GC pause per query);
3. $100\,\%$ absorption of 50 302 labelled adversarial requests with zero false positives and favourable rejection economics on server-grade hardware;
4. Atomic cryptographic integrity on every mutation at native write latency (21–113 ms p95).

These results are not theoretical projections. They are the output of reproducible harnesses (`npm run benchmark`, `npm run benchmark:kernel`) executed against live servers on 18 July 2026. The same evidentiary standard that establishes the positive results also records the guard gaps, the

VM heap-state transition, the single-thread ceiling, and the tier-dependent cost asymmetry. Any subsequent engineering or valuation claim must remain anchored to this measured record.

12 References

[1] Acumen Engineering Directorate. *Empirical Performance Analysis Report — Load-Test and Server-Instrumentation Evidence Across Two Hardware Tiers.* Benchmark date 18 July 2026, build `21e4d9f`.

[2] Acumen Engineering Directorate. *Safety-Kernel Empirical Analysis Report — Adversarial, Classified Absorption Measurement Across Two Hardware Tiers.* 19 July 2026.

[3] Acumen Engineering Directorate. *Strategic Technology Assessment: Project Acumen — Financial Infrastructure Applications.* 19 July 2026.

[4] Acumen Engineering Directorate. *Strategic Appraisal: Project Acumen — Measured-Function Basis.* 19 July 2026.

[5] Möbius, C., Dargie, W. & Schill, A. Power Consumption Estimation Models for Processors, Virtual Machines, and Servers. *IEEE TPDS* 25(6), 2013.

[6] Khan, K.N. et al. RAPL in Action: Experiences in Using RAPL for Power Measurements. *ACM TACO* 3(2), 2018.

[7] Harizopoulos, S. et al. OLTP Through the Looking Glass. *SIGMOD* 2008.

[8] Neumann, T. Efficiently Compiling Efficient Query Plans for Modern Hardware. *PVLDB* 4(9), 2011.

[9] Diaconu, C. et al. Hekaton: SQL Server's Memory-Optimized OLTP Engine. *SIGMOD* 2013.

— End of Technical Whitepaper —

Reddy and Reddy Family Office (i.f.) • GIFT City, IFSC, Gujarat