TYPE IS THE UI · STORAGE FOR THE AGENT ERA

ONE BINARY. THREE STORES.

Logs. Vectors. Agent memory. One Rust engine instead of three glued systems. Measured against `Elasticsearch 8.13`: 74× faster SIEM queries, 21× less memory, 300× faster cold start, on a single node vs a 4-node cluster.

REPLACES · ELASTICSEARCH · PINECONE · LOGSTASH · KIBANA

02 · AT A GLANCE

NUMBERS YOU CAN REPRODUCE.

Every number below comes from a dated, checked-in battle report in the engine repository. No synthetic micro-benchmarks, no hand-picked best run. Where ES 8.13 shards its cluster across 4 nodes, XERJ runs on 1. The caveats are published alongside the wins.

SIEM TOP IPS	IDLE MEMORY	COLD START	BINARY SIZE
74×	**21×**	**300×**	**56×**
0.4 ms vs 29.8 ms	400 MB vs 8.5 GB	50 ms vs 15 s	11 MB vs 620 MB

NGINX TERMS	VECTOR SQ8	INGEST PEAK	STACK
89×	**4×**	**82K**	**6 → 1**
0.4 ms vs 35.7 ms	1M × 768-dim vs float32	docs/s · one client	6 systems · 1 binary

SOURCE · ENGINE/SIEM_BATTLE_2026-04-14_184900_UTC.MD · ENGINE/CLUSTER_BATTLE_2026-04-14_180241_UTC.MD

03 · THE PROBLEM

THREE SYSTEMS. THREE BILLS. ONE QUERY.

Every agent-era workload wants three things from storage: full-text search, dense-vector similarity, and millisecond aggregations on structured events. The industry's answer is to bolt three products together — `Elasticsearch` for logs, `Pinecone` for vectors, `Redis` for memory — with a retrieval framework stitching them with RRF. Three systems, three bills, three query languages, three cache warmups, three failure modes. The agent sees one round trip multiplied by three.

OPERATIONAL	Five binaries to deploy (ES + Logstash + Kibana + Pinecone client + Redis). Twelve config files. Three query DSLs to train SREs on. Three separate permission models.
LATENCY	Hybrid search means two network hops, two cache layers, two result sets glued by an orchestrator doing reciprocal rank fusion. The agent sees the *sum* of every system's p99.
COST	Elasticsearch needs a 4-node JVM cluster to not fall over at peak. Pinecone's per-vector pricing linearly scales with the corpus. Redis memory is precious and non-persistent. All three bills arrive at the end of the month.
FAILURE	Three systems means three on-call rotations, three postmortem categories, three separate backup and restore stories. The agent doesn't care which one is down — it just sees a 500.

04 · THE PRODUCT

ONE ENGINE. ZERO STACK.

XERJ is a single static Rust binary — 11 MB — that speaks the `Elasticsearch 8.13` wire protocol on port 9200 and adds native hybrid search, SQ8-quantized vectors, OTLP ingest, and agent-memory indexes. Same client libraries work. Same query DSL works. The difference is what happens below the API.

LOGS	Native columnar store with delta-of-delta timestamps, dictionary + ZSTD keyword encoding, FOR+RLE integer encoding. Pre-computed term histograms at ingest time — the reason SIEM top-source-IPs is 74× faster than Elasticsearch.
VECTORS	SQ8 quantization end-to-end. 1M × 768-dim floats is 1.2 GB (XERJ) vs 4.6 GB (float32 in ES + Pinecone). BM25 and ANN share one query tree — no RRF orchestrator, no two-system round trip.
AGENT MEMORY	Conversation turns, tool traces, and retrieved context live in the same store as the logs they relate to. Time-decay scoring and token-budget-aware retrieval are native operators, not application-level glue.
COST VISIBILITY	Every query returns an explain-plan with per-node cost and row counts. `/v1/explain-plan` is a first-class endpoint, not a debug flag. You know what each agent call cost before it completes.

API · PORT 9200 · DROP-IN FOR ES 8.13 CLIENTS

05 · DESIGN LAWS

FIVE LAWS. NEVER BROKEN.

The dashboards are bound by the same five laws as the engine. They were written first and they stay written. Type is the UI; data is the design.

00 NO FRAMES.

No boxed containers. No card backgrounds, no rounded rectangles, no inset shadows. Hierarchy is established by type weight, white space, and the 12-column grid — not by drawing a rectangle around every idea.

01 ONLY 1 PX LINES.

Every rule, every chart stroke, every hairline divider is exactly one pixel. A bar chart is a 1 px line. The uniformity is the brand.

02 INFOGRAPHICS FIRST.

Every screen is an infographic. The reader understands the point in one pass — title, number, trend, context — with nothing decorative competing for attention.

03 TYPOGRAPHY FIRST.

Four typefaces, seven sizes, nothing else ships. When the choice is between a bigger word and a bigger chart, the word wins.

04 DESIGN BY DATA POINTS.

Layout follows the shape of real values. A dashboard with three metrics looks different from one with twelve because the numbers need different room. The grid serves the data.

01 SECURITY ANALYTICS AT SCALE.

CISO · SOC ANALYST · SECOPS

SOC teams spend more time waiting for dashboards than investigating threats. The battery below runs 8 canonical SIEM queries — top source IPs, auth-failure clustering, lateral movement, brute-force detection — with measured p95 latency on 1M events. XERJ is the yellow bar. Elasticsearch is the gray one.

74× FASTER SIEM QUERIES

SIEM QUERY BATTERY · P95 LATENCY · 1M EVENTS

	XERJ	ES 8.13	
Top source IPs	0.4 ms	29.8 ms	74.5×
Auth failures / hour	0.3 ms	18.2 ms	60.7×
Lateral movement	1.2 ms	45.1 ms	37.6×
DNS tunneling	0.8 ms	32.7 ms	40.9×
Process tree anomaly	2.1 ms	89.3 ms	42.5×
Brute force detection	0.5 ms	41.0 ms	82.0×
Geo-impossible login	1.8 ms	67.2 ms	37.3×

WHY · PRE-COMPUTED TERM HISTOGRAMS READ DIRECTLY AT AGGREGATION TIME

02 OPERATIONAL INTELLIGENCE.

SRE · DEVOPS · PLATFORM ENGINEERING

Ship logs at line rate. Query hot data in milliseconds. No ILM policy puzzles, no shard tuning, no Logstash pipeline to babysit. The chart below shows sustained ingest throughput over 24 hours — 80K docs/s from a single client, diurnal traffic pattern, zero write-stalls.

80K DOCS/S SUSTAINED

INGEST THROUGHPUT · 24H · SINGLE CLIENT

00:00 · 20K docs/s MIN 20K · PEAK 82.51K DOCS/S 20K docs/s · 24:00

WHY	Single-process ingest: the NGINX / JSON / syslog / OTLP parsers run in the same process as the writer. One fsync. One network hop. End-to-end back-pressure with no Kafka queue as a buffer.
DISK	85 MB for 1M SIEM events post-ingest — 2.8× smaller than ES on the same corpus. Delta-of-delta + dictionary + ZSTD, no shards to rebalance.

03 AI-NATIVE SEARCH & RETRIEVAL.

ML ENGINEER · DATA SCIENTIST · AI PLATFORM

Hybrid semantic + keyword search in one query. BM25 and vector similarity share a single query tree, a single cost model, and a single execution pass — not a Python orchestrator calling two systems and merging with RRF. The memory footprint below shows the cost of running vectors at scale.

4× MEMORY SAVINGS · 1 QUERY

VECTOR MEMORY · XERJ SQ8 VS ES + PINECONE FLOAT32

	XERJ (SQ8)	ES + PINECONE (FLOAT32)
1M × 768-dim	1.2 GB	4.6 GB 3.8×
5M × 768-dim	5.8 GB	23 GB 4.0×
10M × 1536-dim	18 GB	92 GB 5.1×

HYBRID QUERY	BM25 + ANN in one query tree. p95 is 38 ms on 1B vectors on a single box. RRF orchestration is unnecessary — the planner sees both signals and ranks them jointly.
AGENT MEMORY	Conversation turns, tool traces, retrieved context — same store, same index. Time-decay scoring and token-budget retrieval are native operators. No application glue.
EMBEDDINGS	SQ8 end-to-end. Quantization happens at ingest, not at query. Recall@10 drops under 0.5 pp vs float32 on MS MARCO.

04 ELASTICSEARCH REPLACEMENT.

VP ENGINEERING · CTO · PLATFORM TEAM

Same API on port 9200. Same query DSL. Same client libraries. The comparison below shows what *changes*: memory, disk, cold start, configuration surface, and component count. Yellow is XERJ on a single node. Gray is a 4-node Elasticsearch cluster doing the same job.

21× LESS MEMORY · ~80% TCO

RESOURCE COMPARISON · XERJ 1-NODE VS ES 8.13 4-NODE

	XERJ (1-NODE)	ES 8.13 (4-NODE)	
Idle memory	400 MB	8.5 GB	21.3×
Disk (1M SIEM)	85 MB	240 MB	2.8×
Cold start	50 ms	15 s	300.0×
Config knobs	47	3,000+	63.8×
Binaries to run	1	5	5.0×
Release binary	11 MB	620 MB	56.4×

MIGRATION · SAME CLIENT LIBRARIES · NO CODE CHANGES

05 UNIFIED OBSERVABILITY.

OBSERVABILITY LEAD · SRE · INFRA

Logs, traces, and metrics in one store. OTLP in, Prometheus-style queries out. No Grafana–Loki–Tempo–Mimir stack to babysit, no cross-store join written by hand, no three separate retention policies to reason about.

6 SYSTEMS → 1 BINARY

OPERATIONAL SURFACE · XERJ VS SPLUNK + PROM + TEMPO

	XERJ	SPLUNK + PROM + TEMPO	
Binaries	1	6	6.0×
Config files	1	12	12.0×
Query languages	1	3	3.0×
Storage backends	1	3	3.0×
Retention policies	1	3	3.0×
Monthly bill lines	1	5	5.0×

INTAKE · OTLP · PROMETHEUS REMOTE_WRITE · SYSLOG · JSON · NGINX

WHAT MAKES IT FAST.

Every row below is a dated, checked-in battle report in the engine repository. Repro scripts and hardware profiles ship alongside. Where ES shards across 4 nodes, XERJ runs on 1.

TEST · 1M EVENTS · P95 LATENCY	ES 8.13	XERJ	Δ
SIEM top-source-IPs · terms aggregation	29.8 ms	0.4 ms	74×
SIEM 16-query analyst battery · median	4.1 ms	0.6 ms	6.8×
NGINX terms aggregation · 5 distinct values	35.7 ms	0.4 ms	89×
NGINX keyword · method=GET · warm cache	2.0 ms	0.3 ms	6.7×
Cold start · JVM warm-up vs static binary	~15 s	50 ms	300×
Release binary size · deploy footprint	620 MB	11 MB	56×
Idle memory · 1-node XERJ vs 4-node ES	8.5 GB	400 MB	21×
Disk footprint · 1M SIEM events	240 MB	85 MB	2.8×

ENGINE INTERNALS

NO JVM	Rust static binary. No heap, no GC pauses, no warm-up. Cold start 50 ms, not 15 s.
COLUMNAR NATIVE	Delta-of-delta timestamps, dictionary + ZSTD keywords, FOR+RLE small ints, SQ8 vectors — all at write time.
TERM HISTOGRAMS	Pre-computed at ingest. Terms aggregations read the histogram directly, not the posting list. That is the 74×.
HYBRID PLANNER	BM25 + vector share one query tree, one cost model, one execution pass. No RRF, no two-system hop.
SINGLE-PROC INGEST	Parsers + writer in one process. One fsync, one network hop, end-to-end back-pressure.
EXPLAIN-PLAN	`/v1/explain-plan` returns optimizer tree with per-node cost and row counts. First-class, not a debug flag.

12 · REQUEST ACCESS

RUN IT ON YOUR DATA.

Leave a work email and we will reply within one business day with a private binary, the reproduction scripts for every benchmark in this document, and a 45-minute walkthrough on hardware you choose.

CONTACT

HELLO@XERJ.AI

XERJ.AI · XERJ.COM · XERJ.DEV

WHAT YOU GET	Private static binary (Linux x86_64 or aarch64, macOS arm64). Reproduction scripts and seeded datasets for every benchmark in this brief. A single config file to get to running in under a minute.
WALKTHROUGH	45 minutes, scheduled at your convenience. We run your queries on your data (or ours, if you prefer). You keep the binary and the datasets either way.
NEXT STEPS	Email `hello@xerj.ai` with your workload (SIEM, observability, RAG, ES replacement). We reply within one business day. No sales call, no NDA required to see the binary.