COMPILED DOCUMENT MEMORY FOR LLMS

Compile your documents once. Query them with zero reprocessing.

LATCH compiles each document into a fixed-size, deterministic memory artifact — once. At query time, the runtime answers from compiled memory: zero source tokens reprocessed, no embeddings, no index, no retrieval pipeline. It ships as a sealed, self-hosted Docker image, so your documents never leave your hardware — provably, even air-gapped.

1 · Compile
Document → deterministic memory tensor. 347 SEC filings in ~50 s on one A100.

→

2 · Store & Ship
Hash-validated `.latch` files; passphrase-protected bundles. 5.4 GB holds the full corpus.

→

3 · Query
Memory reloads in 2 ms. Deterministic generation: temperature 0, no sampling.

0
SOURCE TOKENS REPROCESSED PER QUERY — 1.65M-TOKEN CORPUS, MEASURED

~50 s
TO COMPILE 347 SEC FILINGS — ONE A100 80GB

0.79 s
TIME-TO-FIRST-TOKEN AT CORPUS SCALE — 347 FILINGS, ROUTED

~1 min
FULL-CORPUS RECOMPILE WHEN CHANGING BASE MODELS

Deterministic & Verifiable

Same input, same artifact. Hash-validated files and temperature-0 generation — answers you can reproduce for an auditor, months later.

Zero-Reprocessing Economics

Retrieval stacks re-embed, re-rank, and re-read on every query — 5.5–8.8 s before generation even starts on our measured baseline. LATCH reads compiled memory.

Self-Hosted & Portable

Sealed Docker image on your GPU. Exportable artifact bundles. No phone-home, no cloud dependency, air-gap capable.

Built to be measured, not marketed. Every number above comes from a logged run on dated hardware, published alongside the ones we lose (decode throughput currently trails a vLLM baseline, ~27 vs ~44 tok/s). Full tables: `codynamicslab.com/benchmarks.html`

The three scenarios that follow show what this looks like inside real organizations — a small firm, an enterprise compliance team, and a software platform. They are **modeled scenarios with every assumption stated**, built on the measured figures above, not customer case studies. Check our math; that's the point.

MODELED SCENARIO — ASSUMPTIONS STATED BELOW

The law firm that couldn't use AI at all

Dana is managing partner at an 8-person patent-prosecution boutique. Every AI tool she evaluated sends client documents to someone's cloud — an instant no under confidentiality obligations, and several clients' outside-counsel guidelines prohibit it outright. The alternative — building a private retrieval stack — means hiring or contracting someone to stand up and *maintain* an embedding service, a vector database, a reranker, and a model server. A firm with no IT department was never going to do that. So the firm's answer to AI has been: "we don't."

With LATCH: one sealed Docker image on a single GPU workstation in the office. Dana's paralegal compiles fifteen years of office actions, prior-art references, and client disclosures — once. Associates query it all day. Nothing leaves the building, provably: the box doesn't even need internet access.

The math — every assumption on the table

Assumption	Value
Attorneys actively using it	`4 of 6`
Time each now spends digging through past matters	`~3 hrs / week`
Value of recovered time (deliberately below billable rate)	`$200 / hr`
Working weeks	`46 / yr`
Avoided alternative: contracted private RAG build + upkeep	`$30–60K + ~$15K/yr`

Recovered capacity: 4 × 3 hrs × 46 wks × $200 ≈ $110K/yr — against an illustrative ~$15K/yr license and one workstation. The avoided build matters just as much: the *cheapest* private alternative costs more to construct than three years of LATCH.

> *"We're the only firm our size answering prior-art questions in seconds — and I can tell any client, in writing, that their documents never left this office."*
>
> — what Dana gets to say

Modeled scenario, not a customer case study. Performance figures are measured (see page 1); staffing, hours, and the $15K license figure are stated assumptions — substitute your own numbers. Corpus-scale figures extrapolate from a 347-document / 1.65M-token measured benchmark.

MODELED SCENARIO — ASSUMPTIONS STATED BELOW

The compliance team the auditors signed off on

Marcus runs compliance technology at a mid-size asset manager — 60 analysts. They already have an internal RAG pilot. What's killing it isn't speed; it's that retrieval is probabilistic. The same question can pull different context on different days, and when the exam team asks *"why did your system say X on March 3rd?"*, nobody can reproduce the answer. Legal's verdict: unusable for anything an examiner touches. Meanwhile the stack itself — embedding service, vector DB, reranker, model server — quietly consumes half a platform engineer.

With LATCH: policies, regulations, and past exam letters compiled into hash-validated artifacts; temperature-0 generation. Same question, same artifact, same answer — and the artifact hash proves which version of the policy corpus produced it. When the base model is upgraded, the corpus recompiles in about a minute and before/after outputs are diffable, so re-validation is an afternoon, not a quarter.

The math — every assumption on the table

Lever	Assumption	Annual value
Ops labor	0.5 FTE platform engineer no longer operating a four-service retrieval stack ($180K loaded)	~$90K
Analyst time	375K queries/yr × ~6 s less waiting (0.79 s vs 5.5–8.8 s, measured) at $75/hr	~$47K
Exam & audit prep	~200 hrs/yr reconstructing and re-validating AI answers — eliminated by reproducibility	~$30K
Model migration	Annual model upgrade: weeks of probabilistic re-testing → days of diffable recompile	~$40K / event

Roughly $200K/yr in labor alone against an illustrative $50K enterprise license — and the unpriced line is the real one: *this is the version compliance and the examiners will actually approve.*

> *"I'm the one who got AI past audit. Every answer it has ever given is reproducible on demand."*
>
> — what Marcus gets to say

Modeled scenario, not a customer case study. Latency figures are measured on a 347-document / 1.65M-token corpus (one A100 80GB); headcount, query volume, rates, and the $50K license figure are stated assumptions — substitute your own.

MODELED SCENARIO — ASSUMPTIONS STATED BELOW

The platform that stopped running 400 databases

Priya is VP Engineering at a legal-tech SaaS with 400 law-firm tenants. Per-tenant document AI means per-tenant retrieval infrastructure: index shards kept warm for tenants who query a few times a day, plus a re-embedding pipeline every time the base model changes. Infrastructure cost scales with tenant count, not tenant usage — and her biggest prospects keep asking for on-prem, which her architecture can't ship.

With LATCH: each tenant's corpus becomes a portable artifact bundle — 5.4 GB for a 1.65M-token corpus in our measured run. Nothing stays warm: an artifact loads in ~2 ms (18.5 ms for a 19-document selection) when a query arrives. Per-tenant *infrastructure* becomes per-tenant files. And on-prem stops being a re-architecture: ship the same sealed runtime with the tenant's own bundle.

The shape of the win

Lever	What changes
Serving economics	Hundreds of always-on index workloads collapse into on-demand artifact loads (~2 ms reload, measured)
Model upgrades	Fleet-wide re-embedding pipeline → per-tenant recompile, ~1 minute per 347-document corpus on one A100
New market	On-prem and air-gapped tenants become sellable — the firms that were disqualifying the product now match its architecture

The CEO line is revenue, not savings: "we can now sell to the firms that were disqualifying us." Cost collapse is the floor; the on-prem segment is the upside.

> *"I cut our per-tenant AI infrastructure to a file format — and opened the on-prem segment sales couldn't touch."*
> — what Priya gets to say

Modeled scenario, not a customer case study. Reload, compile, and bundle figures are measured (347-document / 1.65M-token corpus, one A100 80GB); tenant counts and serving topology are stated assumptions.

See it on your documents.

A live demo takes 30 minutes — we compile a corpus while you watch.

Mike Holford · Founder & CEO, CoDynamics Lab
mike@codynamicslab.com
www.codynamicslab.com · /benchmarks.html