p4-cache

Tiered storage for Perforce.

Production-grade. Audited. Transparent.

Cut Perforce storage costs by 60–80%.

Without changing a single developer workflow.

The problem

Your depot grows. Premium storage doesn't get cheaper.

Every Perforce admin running a depot above a few terabytes knows the storage cost curve. Active projects expand. Historical projects accumulate. Sunsetted titles never quite get archived. The whole depot sits on enterprise SAN or premium cloud SSD because that's what `p4d` needs to serve `p4 sync` at speed — and most of it hasn't been read in months.

The standard workarounds all cost something:

- **`+X` archive filetypes** push the work onto the admin. Per-class decisions, manual restoration, a second depot to manage.
- **`p4 obliterate`** trades storage cost for irreversible deletion. Most admins won't run it without a signed-off retention review, and the time to get one usually exceeds the cost of just buying more disk.
- **NetApp / Pure / Isilon refresh.** Every 4–5 years, write another seven-figure check for the same architecture that's been getting slower-relative-to-the-depot for a decade.
- **Bigger cloud premium SSD.** Scales linearly with depot growth at $900–1,500/TB/year. Painful at 30 TB; brutal at 300 TB.
- **Hand-rolled blobfuse2 or s3fs setups.** Work until the engineer who built them leaves, the manifest gets out of sync with the bucket, or a quiet content-hash mismatch corrupts a sync six months from now.

Why this gets harder every quarter

5–15% of any Perforce depot is read in any 30-day window. The other 85–95% sits on premium storage because some of it might be needed quickly, and `p4d` doesn't distinguish between blocks hit constantly and blocks not

touched since 2019. The depot is uniformly priced: paying the rate of the worst-case access pattern, applied to the entire depot.

This is the cost gap that p4-cache exploits.

The product

How p4-cache works

p4-cache is a userspace daemon plus an LD_PRELOAD shim (`libp4shim.so`). Together they tier cold depot files to your chosen backend and rehydrate them on demand — transparently to `p4d`.

On startup. The daemon manifests the depot, identifies cold files, and uploads them to the configured backend. A redb-backed manifest tracks every file's state (clean, dirty, uploading) at all times. There's no "is this file in the bucket?" guessing — the daemon knows.

At read time. `p4d` runs with `libp4shim.so` preloaded. When `p4d` opens a file that's been tiered, the shim intercepts the read, fetches it from the backend via the daemon, and serves it to `p4d`. Large files use parallel range GETs — one shared driver in `storage/mod.rs` handles the orchestration so every backend (S3/Azure/GCS) gets the same atomic temp-rename + post-rename `assert_regular_file` + parent-fsync semantics. BLAKE3-256 content hashes are verified inline during download — a corrupted or tampered file fails the restore rather than reaching `p4d`. Local cache hits skip the daemon round-trip entirely.

At write time. The daemon's fanotify watcher detects depot writes (new submits, integrations, depot file modifications) and queues them for upload. Writers are UID-allowlisted; events from outside the allowlist are dropped at the kernel boundary.

Hot tier: bring your own NVMe

p4-cache doesn't bundle hardware and doesn't care where the NVMe comes from:

- **Existing servers** with NVMe you already have. Point p4-cache at the local mount.
- **New commodity hardware.** A 2U server with 100+ TB of enterprise NVMe runs $20–40K capex. Five-year fully-loaded TCO lands well under $200/TB/year.
- **Cloud NVMe instances** (Azure Lsv3, AWS i4i, GCP n2-highmem with local SSD). Supported.

Cold tier: any of the major options

- **Azure Blob Storage** — SharedKey, SAS, or Azure SDK default credential chain (managed identity, CLI login, environment).
- **AWS S3** — static credentials or SDK default chain. SSE-S3 encryption. S3-compatible services (MinIO, Wasabi) via endpoint pinning.
- **Google Cloud Storage** — service account JSON or Application Default Credentials.
- **NFS** — for hybrid deployments or shops with existing on-prem secondary storage.

Primary + secondary backends. Configure both, and the daemon falls back to the secondary on `NotFound` from the primary. Useful during cloud migrations.

Two ways customers buy p4-cache

Same product. Different framing. Different commercial structure.

Cloud studio

5–50 TB depot on cloud Perforce. Pain is the monthly cloud bill, asked about quarterly. Typical buyer: Perforce admin or DevOps lead with VP Engineering approval.

Typical 80 TB deployment	With p4-cache	Annual savings
$192,000/yr (Azure Premium)	$92,000/yr	**$100,000 · ROI in 8 months**

Enterprise displacement

200 TB – 2+ PB depot on NetApp / Pure / Isilon. Pain is the refresh quote on the CIO's desk. Typical buyer: storage architect with Director / VP Infrastructure sponsorship.

Typical 1 PB deployment	With p4-cache	Annual savings
$4,500,000/yr (NetApp)	$1,054,000/yr	**$3,446,000 + avoided refresh**

Five-year picture

For a multi-petabyte Perforce shop facing a refresh decision, p4-cache changes the math entirely:

Path	5-year TCO	What it requires
Refresh the array	**$7–15M**	*Status quo CapEx pulse*
FabricPool to S3	**$5–9M**	*Stay on NetApp forever*
p4-cache + commodity NVMe + blob	**$2–4M**	*New license, no refresh*

Production-grade, with citations

This is the pillar most year-1 commercial products cannot claim. p4-cache can.

Engineering maturity

- **~36.5K lines of Rust across six workspace crates** (`p4cache` daemon, `p4shim` LD_PRELOAD, `shared` types/codec, `license` envelope verification, `license-tool` vendor signer, `integrity-stamp` artifact stamper). 286 tests across 31 files — roughly one test per 128 lines of production code.
- **Three independent code audits** completed: quality, security, and verify-binary verification. All findings tracked in repo `*-REMEDIATION.md` files.
- **Zero open CRITICAL findings** today, where CRITICAL is defined as data loss, RCE, auth bypass, secret leak, persisted state corruption, or known-exploit CVE in a direct dependency. **Zero open HIGH engineering findings** — the three previously-tracked items (integrity-failed staging-path RAII, fanotify-overflow recovery covering tracked files, verifier argv hardening) are all closed with pinning tests in the suite. Two remaining HIGH supply-chain advisories are transitive deps pinned by upstream SDKs (waiting on upstream bumps; ignored in `rust/.cargo/audit.toml` with documented rationale).
- **Compile-time defenses** against entire classes of bugs (Rust edition 2024): every unsafe operation requires an explicit unsafe block; holding mutex guards across await boundaries is a compile error; silent integer casts produce warnings.

- **CI gates** on `cargo audit` (`--deny warnings`) and `cargo deny` (advisories, bans, licenses, sources).
- **Fuzz harnesses** on the FETCH protocol parser and the Perforce journal parser. Run in CI smoke mode on every push.

Defenses by category

Process isolation. `CAP_SYS_ADMIN` is dropped from the daemon's effective and permitted capability sets after fanotify initialization. The shim's panic strategy is `unwind` (not `abort`) so its FFI guards actually catch panics; a Rust panic in shim code falls through to real libc rather than crashing `p4d`.

Authentication boundaries. Three independent UID allowlists, enforced at the kernel boundary: shim socket (`SO_PEERCRED`), audit-log socket (`SCM_CREDENTIALS`), fanotify writer. Rejections counted in Prometheus. No process outside the allowlist can connect to the shim socket or inject audit events.

Data integrity. BLAKE3-256 content hashing, end-to-end. Every uploaded file's hash is stored in the manifest. Backends write into an engine-owned staging path; the atomic rename onto the live depot path happens only after the content-hash check passes. The `StagingPathGuard` RAII unlinks the staging file on every error path, so rejected bytes never land at the live path. The manifest codec is versioned with tripwire-safe downgrade behavior.

Forensic watermark. Every cold-tier write carries a deployment-identifying watermark in object metadata and (where the SDK supports it) in the cloud `User-Agent`. A leaked or exfiltrated blob is traceable to the licensed deployment that produced it. Soft and hard managed-capacity ceilings surface as Prometheus metrics for operator alerting.

Audit trail. Every depot file read is recorded to Elasticsearch or PostgreSQL with TLS (custom CA supported), deduplication, batching, and optional on-disk spool for outage resilience. Two distinct drop boundaries (pre-spool queue overflow vs. post-spool spool overflow), both observable in Prometheus. No silent drops.

> ***What this means for your security review***
>
> *The Security & Compliance Brief is the foundational document for an enterprise security review. It cites specific ADRs and audit findings, documents the threat model, and enumerates defenses by category. Most year-1 products require weeks of security-team investigation before approval. p4-cache provides the answers up front. Available under NDA.*

Pricing & next steps

Pricing

Per-TB managed capacity. No per-seat licensing. No feature gating. Free 60-day evaluation — feature-complete, capacity-limited (500 GB hot / 5 TB cold).

Depot size	Annual list	Buyer profile
5 TB	**$7,000**	*Admin / lead approval*
30 TB	**$29,000**	*VP Engineering signoff*
100 TB	**$89,000**	*Director / VP Infrastructure*
500 TB	**$219,000**	*Enterprise — typically ELA*
2 PB	**$519,000 list / ELA in practice**	*Array-displacement scale*

Multi-year ELAs available for deployments above 500 TB. Typical structure: 3-year commitment, paid migration services ($50K–$250K depending on scope), reference rights, committed growth bands.

Perpetual license available for procurement environments that prefer CapEx accounting. Priced at 3× annual list plus 18% annual maintenance.

Next steps

1. **Run the TCO calculator** at rcjacksonconsulting.com. Anchored to your actual baseline (cloud, on-prem, or NetApp). Returns a number you can show your CFO in 30 seconds.
2. **Request the briefs** for your technical and security teams: Architecture Deep-Dive, Security & Compliance Brief, Engineering Maturity Brief. Available under NDA.
3. **Schedule a 30-minute call** to discuss your storage baseline, refresh timeline, and proof-of-concept scope.
4. **Run the POC** for 2–4 weeks (cloud studio) or 4–8 weeks (enterprise). Measure on your hardware, your access patterns, your real depot.

Calculate your savings · Request an evaluation · Book a technical call

rusty@rcjacksonconsulting.com *rcjacksonconsulting.com*