p4-cache — Executive Briefing

The opportunity

Every Perforce shop with a depot above a few terabytes pays for storage at the rate of its worst-case access pattern. In a typical environment only 5–15% of depot data is read in any 30-day window; the other 85–95% sits on premium enterprise storage — NetApp, Pure, Isilon, premium cloud SSD — paying array prices for content almost nobody touches.

p4-cache ends that. It keeps your hot working set on fast local NVMe and transparently tiers the cold bulk to cheap cloud object storage (Azure Blob, AWS S3, Google Cloud Storage) or NFS. Perforce itself is untouched: no `p4d` patches, no protocol changes, no client software, no change to how developers work. You retire the expensive array and pay object-storage prices for the 85–95% that was sitting idle.

What you save

Enterprise storage runs roughly **$1,500–3,000 per TB per year** fully loaded. p4-cache moves the cold bulk to object storage at **$24–150/TB/yr** and keeps the hot set on commodity NVMe (**~$0–300/TB/yr**, often near-zero if you reuse hardware you already own). The p4-cache license is a flat **$100 per TB per year** — a small slice of what you stop paying the array vendor.

Recurring annual savings, tiering 90% of the depot to object storage and keeping 10% hot on NVMe:

Depot	Storage today (~$1,800/TB/yr)	With p4-cache, all-in (~$210/TB/yr)	Annual savings	of which, license
50 TB	$90,000	$10,500	**$79,500 (88%)**	$5,000
100 TB	$180,000	$21,000	**$159,000 (88%)**	$10,000
250 TB	$450,000	$52,500	**$397,500 (88%)**	$25,000
500 TB	$900,000	$105,000	**$795,000 (88%)**	$50,000
1 PB	$1,800,000	$210,000	**$1,590,000 (88%)**	$100,000

All-in with p4-cache = object storage for the cold 90% + NVMe for the hot 10% + the $100/TB/yr license. The license is only ~6% of what you save; the storage displacement is the prize. Uses a conservative $1,800/TB/yr array baseline (the range is $1,500–3,000). Run the calculator against your real depot and baseline for your number.

Plus avoided refresh CapEx. Enterprise arrays carry a $2–3M refresh pulse every 4–5 years. p4-cache lets you skip that pulse — for a large deployment the avoided one-time CapEx is on the order of a full year of the recurring savings above.

Five-year picture: a 1 PB deployment

Path forward	5-year cost	What it means
Refresh the array	**$9–12M**	Status quo plus a refresh pulse in the window. Lock-in persists.
Partial tiering, stay on the array	**$5–8M**	Helps at the margin. You still own and refresh the array.
p4-cache + commodity NVMe + object storage	**$1–2M**	A small annual software license. No array. No refresh.

Your developers will not notice — and may get faster

The savings do not cost you a single second of developer time. This is measured, in an in-repo, re-runnable benchmark (500 files / 492.5 MiB, 32 concurrent readers, regenerated 2026-07-09):

- **Writes stay at local speed.** `p4 submit` lands within about 4% of plain `p4d` on every cloud backend (−1.0% to +4.2%). The durable cloud upload runs asynchronously behind the local cache, so submit returns at local-cache speed and is never gated on cloud latency.
- **Warm reads are local-disk speed.** Once a file is cached, steady-state reads run 170–190 MB/s against a 196 MB/s no-cache baseline — within ~4% — with **100% of reads served from the local cache and zero backend round-trips** (verified: cache restores did not increase, on every backend).
- **The cold miss is a one-time cost.** The first read of a file after eviction pays a single cloud round-trip, then it is cached. That is the worst case, not the steady state; we do not claim cold cloud reads match local speed.
- **Moving off network storage is architecturally faster.** If your working set lives on NAS/NFS today, putting it on a local NVMe cache removes the network filesystem from the read path — a structural speedup. (The benchmark uses a local-directory stand-in for NFS, so it demonstrates no-regression parity rather than putting a number on the network-NFS gain.)

These are single-run representative ratios from our harness, not a spec sheet — traceable to `docs/perf/backend_comparison.md`.

Proven, safe, and recoverable

p4-cache is built for environments where Perforce uptime is non-negotiable and the data cannot be lost.

- **Independently reviewed, adversarially.** A deep, multi-subsystem code and security review surfaced **31 issues** across the daemon, the shim, storage, and licensing — logic, failure-handling, and security findings. Every one was adversarially verified and fixed.

- **Disaster recovery, proven.** The cloud object store is the source of truth. If the local catalog database is ever lost, a fresh install reconstructs it from the cloud — a documented recovery path with a purpose-built rebuild tool. The durability gaps that could strand or delete recoverable data were found and closed.
- **Tested against failure, not just the happy path.** A failure-injection suite (~45 tests) proves recovery with **no data loss and no corruption** across process crash / SIGKILL, Perforce-server outage, network and backend loss, disk-full and read-only disks, and bit-level corruption. Every faulty-bytes path is refused, never served.
- **Every restored file is content-verified.** Restores are checked against a BLAKE3 content hash before they reach `p4d`; a corrupted or tampered file fails the restore. A checkpoint-verify tool offers three cost-tiered integrity levels for ongoing assurance.
- **Reversible by design.** Stop the daemon and remove the shim, and `p4d` reverts to a stock deployment. It fits inside your existing Perforce high-availability and replication — no new HA model to adopt.
- **Traceable and auditable.** Every cold-tier object carries a deployment-identifying watermark, so a leaked blob traces back to the deployment that produced it. Depot reads can be audit-logged to Elasticsearch or PostgreSQL for compliance.

Engineering rigor you can lean on. *Every change must clear a strict quality gate before it ships: an 85% test-coverage floor, a zero-warnings lint wall, a minimum-supported-compiler build, fuzzing, a performance-regression gate, and dependency vulnerability and license auditing. Release binaries ship with build-provenance attestation and checksums. Review artifacts are available under NDA.*

Commercial structure

- **Flat, per-capacity license: $100 per TB per year.** A $1,000/yr floor for the smallest deployments (≤10 TB) and a flat **$100,000/yr cap** reached at 1 PB. Simple to model, simple to approve.

Capacity	10 TB	25 TB	50 TB	100 TB	250 TB	500 TB	750 TB	1 PB
List / yr	$1,000	$2,500	$5,000	$10,000	$25,000	$50,000	$75,000	$100,000

- **Design Partner** — for our first customers: 50% off year one, reference rights.
- **Annual** — the standard subscription, priced from the table above.
- **Perpetual** — for procurement environments that prefer CapEx: 3× annual list plus 18%/yr maintenance.
- **Enterprise License Agreement** — for deployments above 1 PB, custom-priced.
- **Built-in floating trial** so you can run a real proof-of-concept against a representative slice of your own depot before you commit.

Next steps

1. Run the TCO calculator against your actual depot and storage baseline for a savings number anchored to your environment.
2. Request the architecture deep-dive and the security & recovery brief for your technical and security teams.
3. Book a 30-minute call to walk through your storage baseline, refresh timeline, and proof-of-concept scope.

rusty@rcjacksonconsulting.com *p4-cache.com*