Design_Document

Status	Done

DESIGN DOCUMENT

1. Purpose & Scope

This design document explains how the AVID-FP Object Store turns the AVID-FP research protocol into a production-grade, container-native storage service. It targets (i) developers who will read/extend the code, (ii) operators who will deploy and monitor clusters, and (iii) reviewers who need a precise map from requirements to implementation. Topics covered:

- Functional & non-functional requirements
- High-level and component-level architecture
- Data and metadata layouts
- Control-flow (write/read) state machines
- Persistence, recovery, and GC strategies
- Configuration hierarchy
- Security model
- Performance and scalability considerations
- Extensibility hooks

2. Requirements Summary

Category	Requirement
Durability	Recover the object if ≥ m of n shards survive
Integrity	Detect any tampering with probability ≤ 2^{-64}
Fault tolerance	Liveness & safety under ≤ $f = n - m$ Byzantine nodes
Throughput	≥ 100 MB s^{-1} aggregated writes for 1 GiB objects (3-of-5)
Latency	Retrieve overhead < 10 % vs. raw copy

Observability	Prometheus metrics + Grafana dashboard out-of-box
Operability	One-command compose up, rolling upgrade, GC, snapshot
Portability	Run on any x86/ARM host that supports Docker ≥ 24

Non-goals: encryption at rest, dynamic membership, and WAN optimisation.

3. Architecture Overview

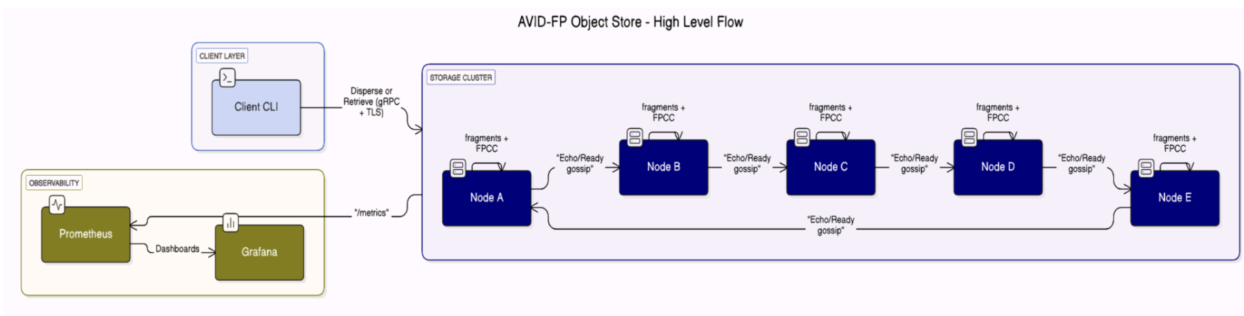


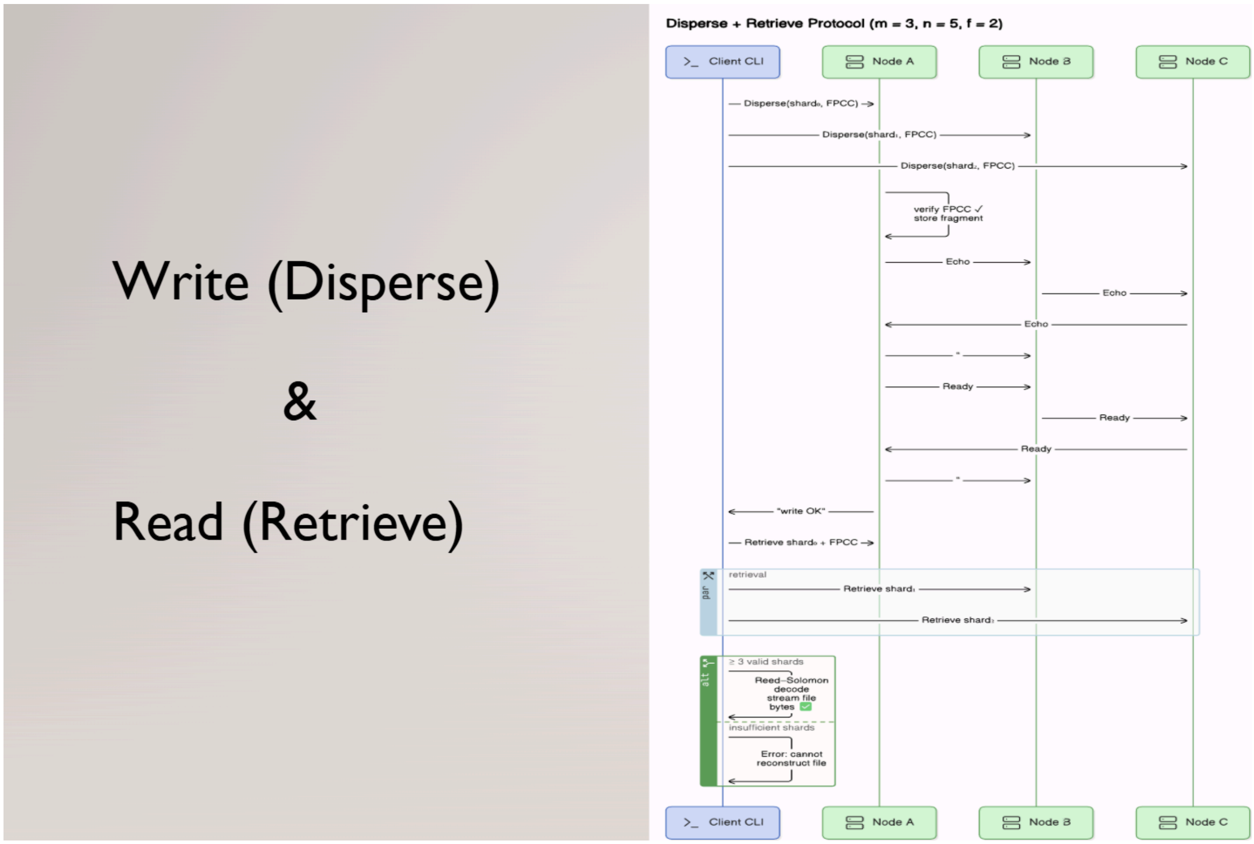
Fig - High-Level Design

Stateless clients compute shards and metadata; *stateful nodes* persist fragments and orchestrate quorums. No coordinator or external DB is needed.

4. Key Data Structures

Structure	Description	Stored In
Shard	Binary fragment (data or parity)	Filesystem
FPCC	`{hashes[], fps[], seed}` – per-object cross-checksum	BoltDB (`fpccs`)
Quorum receipts	`echoSeen` , `readySeen` keyed by `object	peer
Meta	`created: time.Time` for GC	BoltDB (`meta`)

5. Control Flows



5.1 Disperse (Write)

1. **Client**
   - Encode object → n shards (`pkg/erasure`).
   - Compute SHA-256 + 64-bit FP (`pkg/fingerprint`).
   - Broadcast `DisperseRequest` to every node.
2. **Node**
   - Verify hash & FP; atomic-write shard; persist FPCC if first fragment.
   - Mark `echoSeen[self]`; batch commit; gossip `Echo`.
   - On ≥ $m + f$ Echoes → gossip `Ready`; on ≥ 2f + 1 Readies → commit, reply `ok` to client.
3. **Client** succeeds when every shard index has an `ok` response.

5.2 Retrieve (Read)

1. Client fetches shard 0 from any node to obtain FPCC.
2. Downloads fragments until *m* pass hash + FP checks.
3. RS-decodes, trims padding, writes output.
4. Aborts if < *m* good shards before list exhausted.

6. Component Design

Component	Core APIs / Responsibilities
pkg/erasure	`New(m,n)`, `Encode([]byte)`, `Decode(shards, size)`
pkg/fingerprint	`NewRandom()`, `Eval([]byte) uint64`, `Seed()`
pkg/storage	`AtomicWrite(path, data)`, `Batcher.Put(k,v)`
cmd/server	Flag parse → config, open BoltDB, start gRPC, register Prometheus, GC, snapshot
cmd/client	CLI flag parse, codec + FP compute, shard fan-out, retry policy, retrieve orchestration
deploy/	`docker-compose.yml`, `prometheus.yml`, Grafana JSON

7. Persistence & Recovery

- **Atomicity** – Write to `<path>.tmp` then `os.Rename`.
- **Crash safety** – BoltDB is WAL-backed; commit after fsync ensures metadata durable.
- **Startup recovery** – On boot, server reloads FPCC bucket to serve reads instantly; incomplete `.tmp` files are ignored.
- **Garbage collection** – GC loop every TTL/2 deletes expired fragment dirs and BoltDB keys in one transaction.
- **Snapshot** – `snapshot /dst` copies BoltDB + fragment tree to timestamped dir; safe because writes are immutable after commit.

8. Configuration & Deployment Details

Layer	Mechanism	Example
Node config	`-config /etc/avid/config.yaml`	Ports, peers, paths
Env override	`AVID_ERASURE_DATA=4`	CI secrets, quick tuning
CLI override	`server -peers a,b,c`	Dev experiments
Build	Two-stage Dockerfile (Go builder → distroless)	Static bins, 14 MB image
Orchestration	`docker compose up -d`	5 nodes + Prom/Graf

9. Security Considerations

- **Integrity** – Combined SHA-256 + 64-bit FP; collision prob ≤ 2^{-64}.
- **Confidentiality** – BYO encryption (client-side).
- **Transport** – Optional mTLS (`tls_cert`, `tls_key`, client `tls_ca`).
- **Supply chain** – Distroless image, SBOM via `cosign sbom`.
- **Threats out-of-scope** – Physical theft post-GC, side-channel leakage, kernel exploits.

10. Performance Notes

- SIMD RS codec saturates single core at ~1.1 GB s^{-1}.
- Fingerprint Eval cost ≈ 0.13 cycles/byte (< 2 % CPU).
- Batcher reduces fsyncs to ~12 per 1 GiB object (3-of-5).
- Throughput bottleneck = SSD write bandwidth, not CPU/NIC.
- Gossip adds ~2 RTTs = 0.5 ms on LAN, 200 ms on 100 ms WAN.

---

11 Extensibility Hooks

Area	How to Swap
Erasure code	Implement `Codec` interface; register in `pkg/erasure/factory.go`.

Fingerprint	Replace `Eval()` with 128-bit GHASH; adjust FPCC proto.
Metadata KV	Replace BoltDB calls with Badger or SQLite; keep same buckets.
Transport	Inject `grpc.WithTransportCredentials(creds)` for mTLS/ALTS.
Observability	Add OpenTelemetry interceptors; spans propagate via gRPC metadata.

12 Risks & Mitigations

Risk	Mitigation
Large objects exhaust RAM	Future streaming encode/decode pipeline
Quorum traffic $O(n^2)$	Cap clusters at ≈ 10 nodes or adopt hierarchical overlay
Disk-full mid-commit	Disperse RPC returns error; client retries after operator action
Long GC pauses	GC runs in small batches, sleeps between objects

13 Conclusion

The design keeps the **trusted computing base minimal**, separates **stateless client logic from stateful storage**, and uses **well-understood libraries** for cryptography, encoding, and persistence. Echo/Ready quorums give Byzantine safety without heavyweight consensus logs; homomorphic fingerprints cut verification cost to the bone; distroless containers and Prometheus/Grafana deliver DevOps parity with modern micro-services. The resulting architecture meets academic correctness proofs *and* real-world operability demands, providing a solid foundation for future enhancements such as streaming stripes, dynamic membership, and geo-replicated clusters.