Abstracting the Cloud: The Evolution of Dropbox's Object Store

Alok Ranjan

Software Engineering Manager

Dropbox Inc.

Introduction

Alok Ranjan

Engineering Manager, Storage Platform @ Dropbox

Education

Masters from Carnegie Mellon University

Experience

Big Switch Networks, VMware, Cisco

Focus

Storage systems, scalable infrastructure, Telemetry

The Problem: Legacy Storage Pain Points

Before Object Store (Pre-2020)

- Amazon S3 + HDFS as backbone storage
- Magic Pocket for user files only
- S3/HDFS for internal products

Major Pain Points

- Cost Inefficiency: Expensive S3 PUT/GET requests
- No Flexibility: Locked into S3 as default
- Operational Overhead: Multiple storage systems
- Limited Control: No centralized policies

> *"S3 was simply an expensive default choice"*

The Vision: Meta-Store Abstraction

What We Wanted

- Transparent Backend Routing based on cost & access patterns
- Centralized Control for all blob traffic
- Additional Features: Encryption, monitoring, retention policies

The Birth of Object Store (2020)

> *"Pedestrian name, but impact has been anything but"*

Object Store Architecture

Abstraction Layer

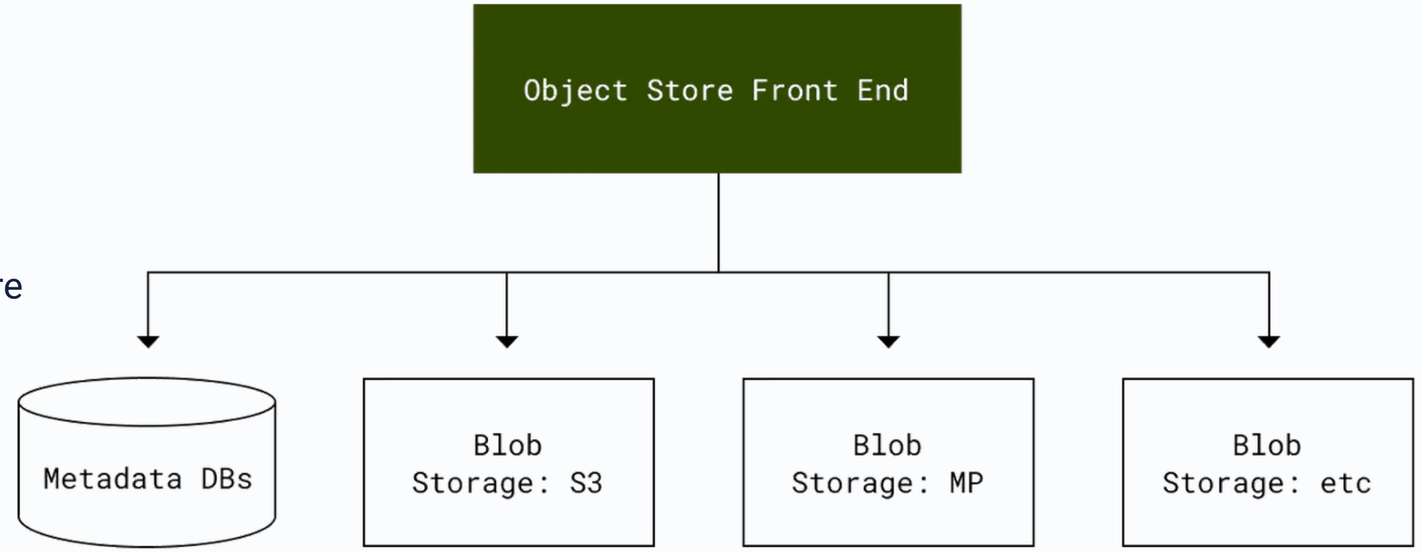
Doesn't store data directly

Smart Routing

PUTs to cheapest store, GETs to correct store

MySQL Metadata

Tracks object placements



Key Backend: Magic Pocket

- Our internal storage system
- Critical Constraint: Optimized for 4MB blobs
- Understanding Magic Pocket explains our design decisions

Magic Pocket: Our Foundation

Exabyte-Scale Blob Storage

- Customer Data Storage: All Dropbox user files
- Availability: 99.99% uptime
- Scale: Millions of queries/second, 600K+ storage drives
- Geography: Three North American regions

Magic Pocket Architecture

100+

Disks per OSD

Each Object Storage Device contains over 100 disks

2+ PB

Storage per OSD

Each OSD manages more than 2 petabytes of data

11 9s

Durability

Fifteen nines via erasure codes

Key Design Principles

- Immutable Writes: Data never changes after write
- Cold Data Optimization: Most data rarely accessed
- 4MB Blob Optimization: Sweet spot for performance

Object Store API & Structure

S3-Compatible API

- PUT, GET, DELETE, LIST operations
- Pails = containers (like S3 buckets)

Smart Backend Selection

- S3: Global regions, legacy compatibility
- Magic Pocket: Cost-efficient, high-performance
- Future backends: Ready for new technologies

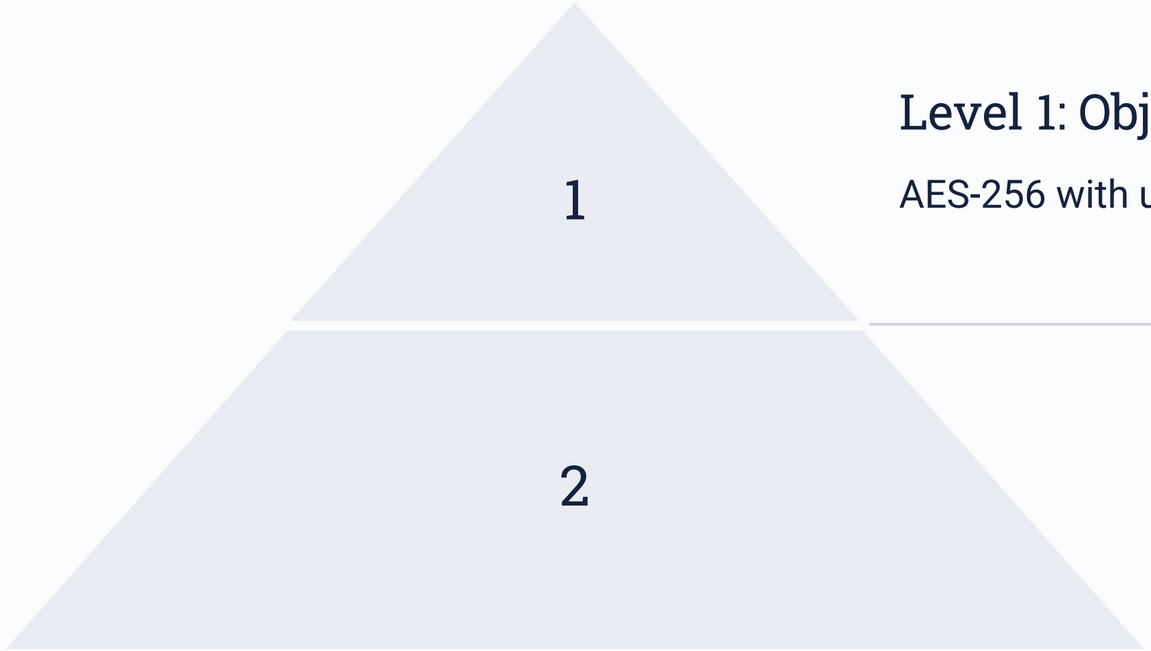
Innovation #1: Batched Writes

1. **Enqueue** PUT requests by pail
2. **Trigger** on timeout OR size threshold
3. **Concatenate** multiple objects → single blob
4. **Write** to Magic Pocket (targeting 4MB)
5. **Store metadata** with offsets

Impact

- Fewer PUT requests = major cost savings
- Optimized for Magic Pocket 4MB constraint
- Cache optimization for GETs

Innovation #2: Layered Encryption



Level 1: Object Encryption

AES-256 with unique 256-bit BEK per object

Level 2: Key Encryption

BEK encrypted with versioned KEK

Encrypted BEK stored in MySQL

Benefits

- Three-factor security: Data + BEK + KEK required
- Efficient key rotation: No object rewrites needed

Innovation #3: Crypto-Shredding for Deletions

The Challenge

- Compliance: Data must be purged within deadline
- Batching Problem: Can't batch DELETEs safely

Solution: Crypto-Shredding

- Simply wipe object metadata (including encrypted BEK)
- No KEK = No data access
- Synchronous compliance via MySQL
- Asynchronous space reclamation

Innovation #4: Object Chunking

The Problem

- Magic Pocket optimized for 4MB
- Many objects > 4MB (logs, artifacts, media)
- Original solution: Route large objects to expensive S3

Chunking Solution

- **Break large objects** into 4MB chunks
- **Store chunks** in Magic Pocket
- **Parallel reconstruction** at read time
- **Deterministic keys** : No extra metadata needed

Benefits

- All data hits Magic Pocket's sweet spot
- Millions in annual S3 cost savings

AI Era Evolution: Future Directions

Current AI Opportunities

- Natural language search
- Content summarization
- Knowledge management systems

Planned Enhancements

- **Lambda Functions**: Object writes/modifications
- **Database Backend**: Cassandra/RocksDB on Object Store
- **Third-party Integration** : Apache Pinot, Loki

SkyVault-Inspired Next-Gen Storage

- **Open-source cloud-native** key-value store
- **Compute-storage separation** using Object Store foundation
- **Target**: AI/ML workloads, feature stores, analytics

Impact & Results

Cost Savings

- **Millions saved annually** from S3 optimization
- **HDFS deprecation** = additional millions
- **Reduced API costs** via batching

Technical Achievements

- **Unified storage interface** across backends
- **Flexible routing** based on cost/requirements
- **Enhanced security** with granular encryption
- **Compliance ready** with instant deletion

Strategic Impact

- **Vendor independence**
- **Future-proof architecture** for AI workloads
- **Platform foundation** for next-gen systems

Key Takeaways

Technical Lessons

1. **Abstraction pays off** : One interface, multiple backends
2. **Batching transforms economics** : Fewer API calls = savings
3. **Right -sizing matters** : Match chunks to backend constraints
4. **Learn from open source** : SkyVault guides next-gen storage

Strategic Insights

1. **Build for flexibility** : Requirements change
2. **Measure everything** : Optimization requires monitoring
3. **Security by design** : Easier than bolting on later
4. **Think beyond current needs** : AI era reshapes storage

Questions?

Thank you!

Alok RanjanSoftware Engineering Manager, Dropbox IncStorage Platform Team

@LifeInsideDropbox

dropbox.com/jobs