Adaptive and Constraint-Aware Data Placement for S3-Compatible Storage

Marcell Feher
Chocolate Cloud ApS

ML4ECS Workshop, 26/01/2026, Krakow

Distributed, Provider-Independent Object Storage

System Baseline

- S3-compatible object storage
- Per-object pipeline
  - Sharding
  - Compression
  - AES-256 encryption
  - Erasure coding (*k-of-n*) with redundancy
- Fragments distributed
  - Multiple cloud providers
  - Multiple regions
  - Optional on-prem backends

Benefits

- No single provider holds reconstructable data
- Resilience to provider and regional failures
- Strong data sovereignty guarantees

Limitations (pre-MLSysOps)

- Storage Policies are static and manually defined
  - Policy: EC parameters and explicit backend placement
- One policy per bucket, applied uniformly to all objects

Technical problem to solve in MLSysOps:
How to adapt data placement when **traffic patterns, constraints or objectives change**?

Online Storage Policy Migration

Initial Contribution

- Introduced bucket-wide policy migration
- Users can update:
  - Erasure coding parameters (redundancy)
  - Backend provider/region assignments
- System migrates existing objects to the new policy
- Online process (no bucket downtime)
- Transparent for users and applications
- Deterministic final state

Why this matters

- Decouples data lifetime from policy lifetime
- Enabled policy evolution without re-upload
- Serves as a **building block for automation**

Policy migration turns static configuration into a controllable operation

Intent-Driven Storage Policies

High-level interface

- Users specify objectives, not placement
- Optimize for:
  - Download latency
  - Egress cost
  - Proximity to users
- Constraints supported:
  - Geo-fence (allowed countries)
  - GDPR mode (no provider stores 100% data)
  - Metro-level outage resilience
  - Max cost ($/TB)
  - Min speed

Key shift in approach

- Explicit placement → constraint satisfaction
- Policy generation is an optimization problem

Results

- System generates a concrete storage policy, not the user
- Policy is valid if all constraints are satisfied

Traffic-Aware, Closed-Loop Adaptation

Observability inputs

- Per-bucket access telemetry
  - Download volume
  - Request origin distribution
  - Serving gateway location
- Storage backend performance
  - Measured throughput from real downloads
  - Per gateway-backend pair

Control loop

- Periodic evaluation of recent traffic
- Solve for optimal policy under constraints
- Compare with current policy
- If different → trigger migration

Characteristics

- Feedback-driven (not rule-based)
- Autonomous but bounded by user constraints
- Migration is the actuation mechanism

MLSysOps spirit

- Applies control loops to storage, not just compute
- Treats data placement as a managed system
- Integrates monitoring → decision → actuation

DEMO

Current Challenges

Better Policy Generation

- Forecast bucket traffic (instead of looking at the recent past)
- Change redundancy when traffic volume changes and constraints allow
- Consider current policy and cost of the migration itself
- Introduce user-defined cost limits for migrations
- Additional controls (e.g., add green energy to the optimization mix)

Finer granularity

- Object or sub-object level traffic profiling, policy eval and migrations

Thank You!