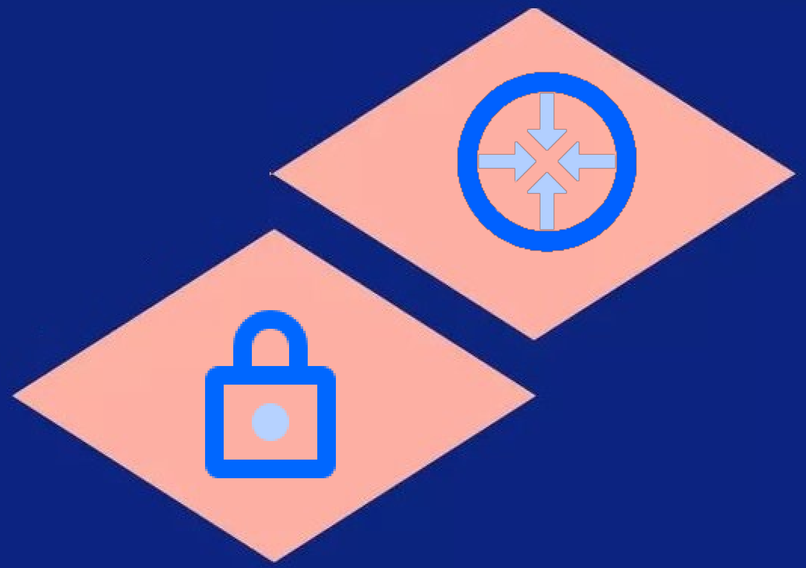
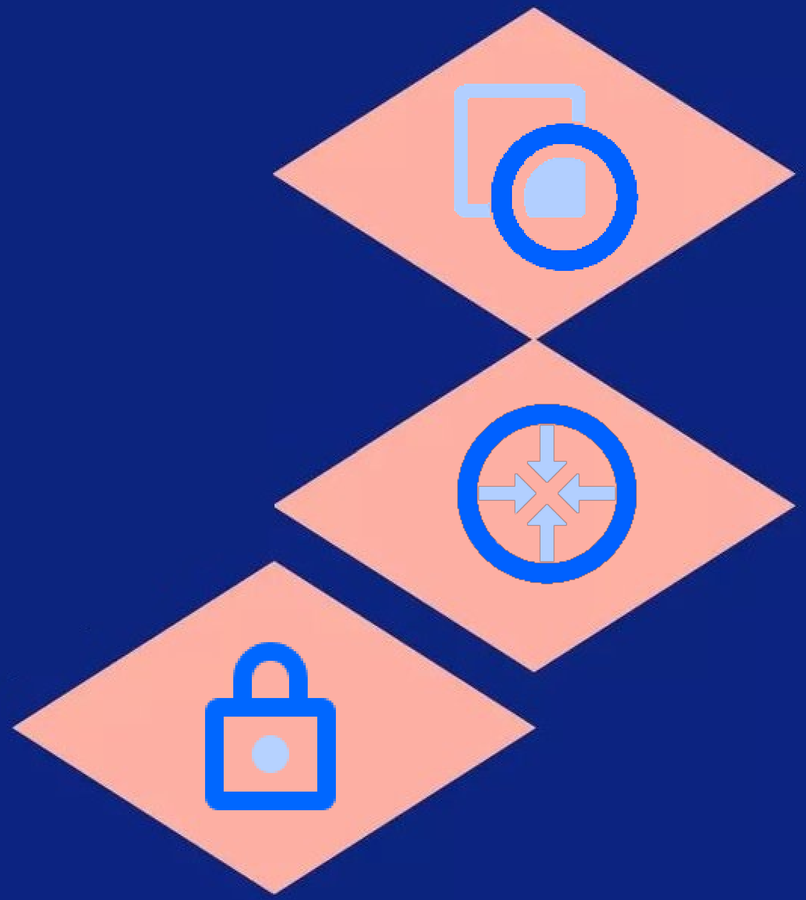
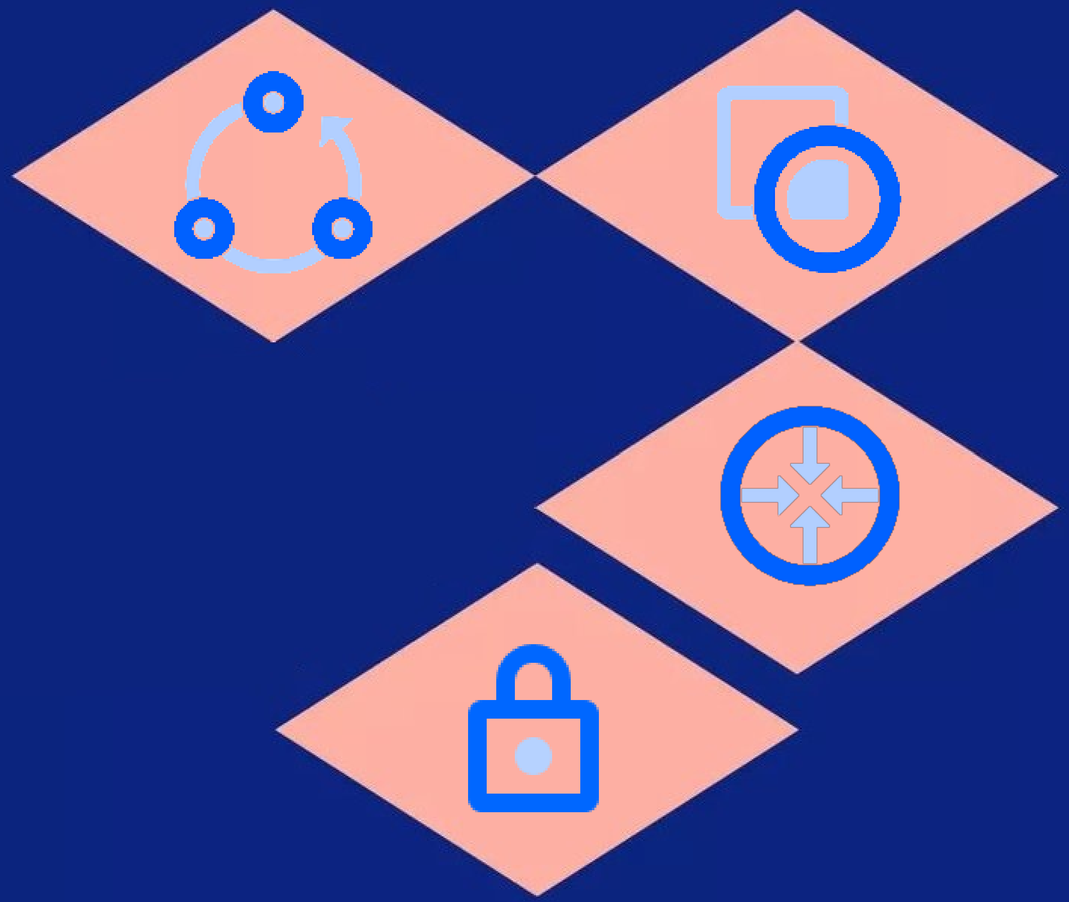
The Magic Pocket
How Dropbox is handling large data sets

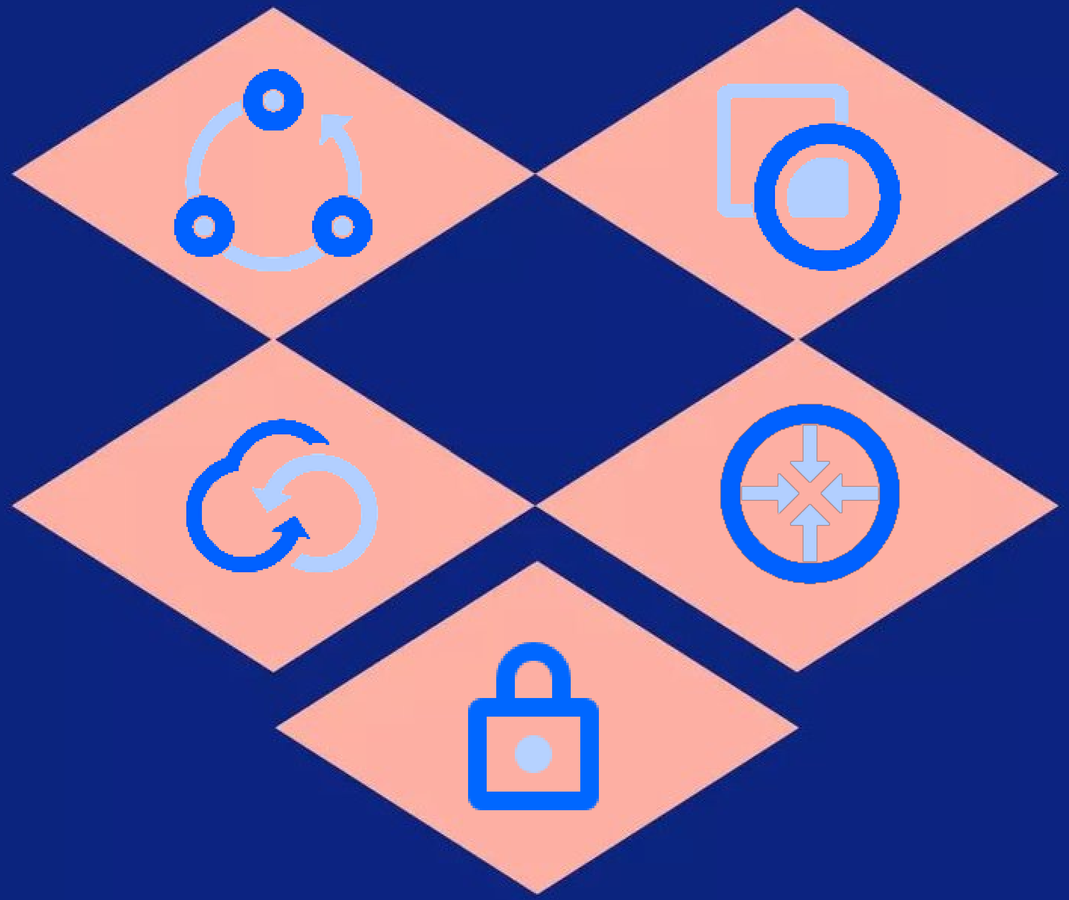
Martin Meusburger
Said Babayev

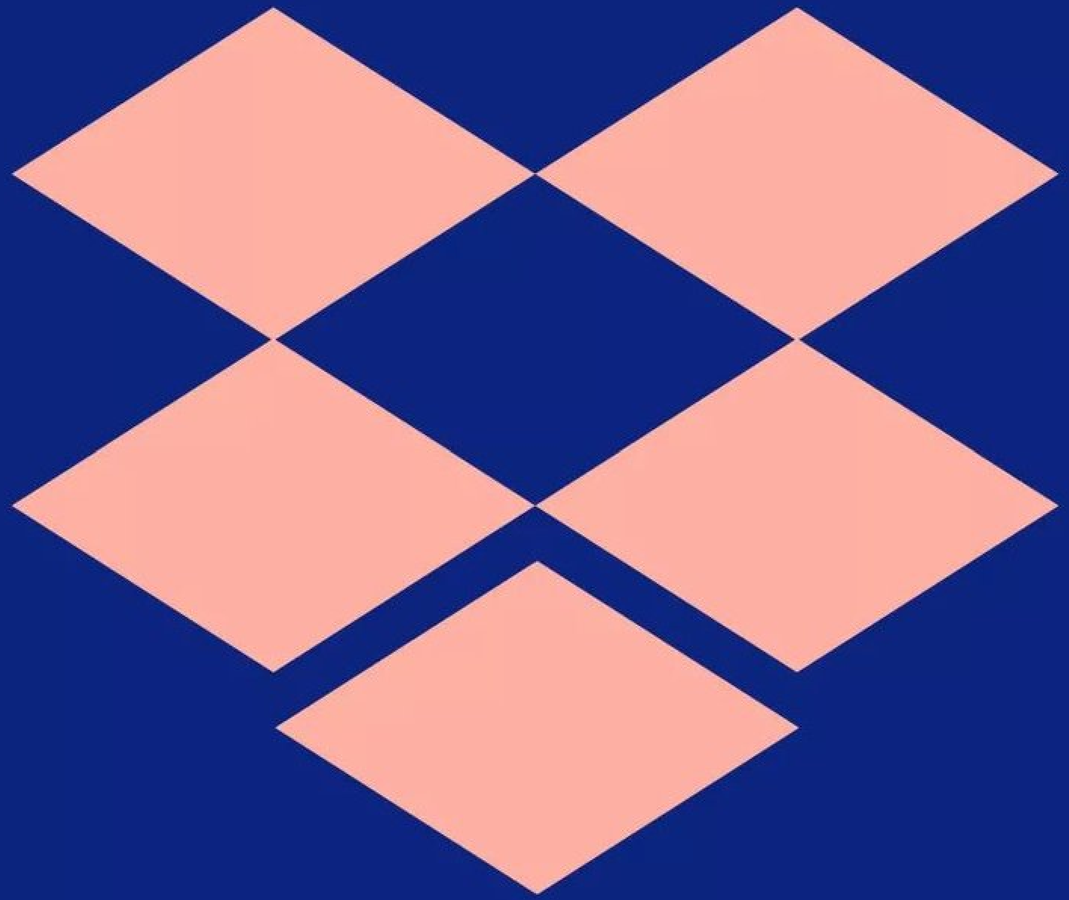
30.01.2019



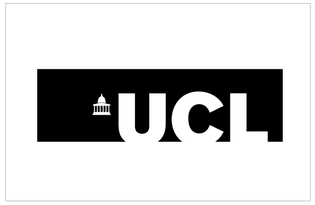


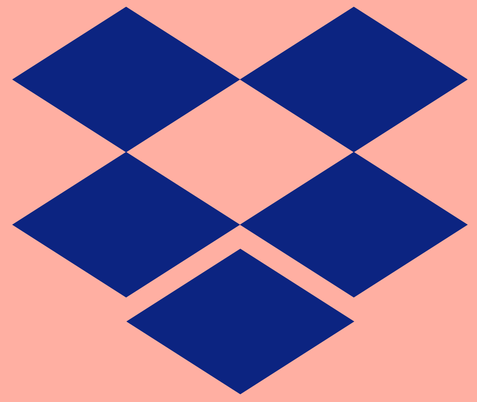






6000+ Universities & Research Centres trust Dropbox





Magic Pocket

About me

- Site Reliability Engineer at Dropbox since Jan 2017
- Teams: Storage, Production Services
- Notable Projects: OS migration, Autoallocator, Storage disks remediation

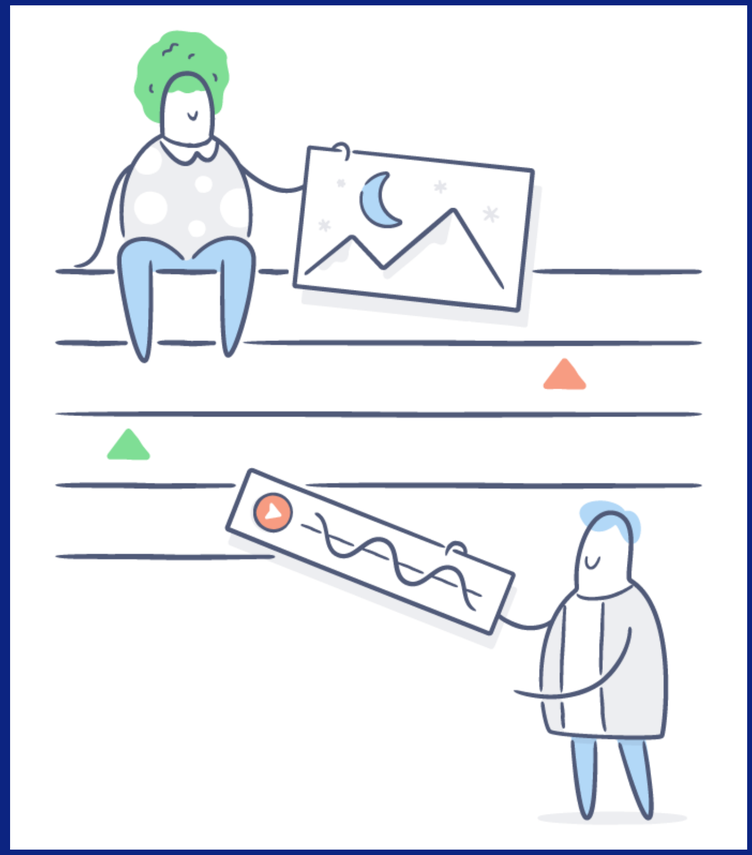
Agenda

Intro

Magic Pocket Architecture Overview

Object Storage Devices

Innovative Hardware Solutions



Immutable

content-addressable

block storage system

aka Magic Pocket

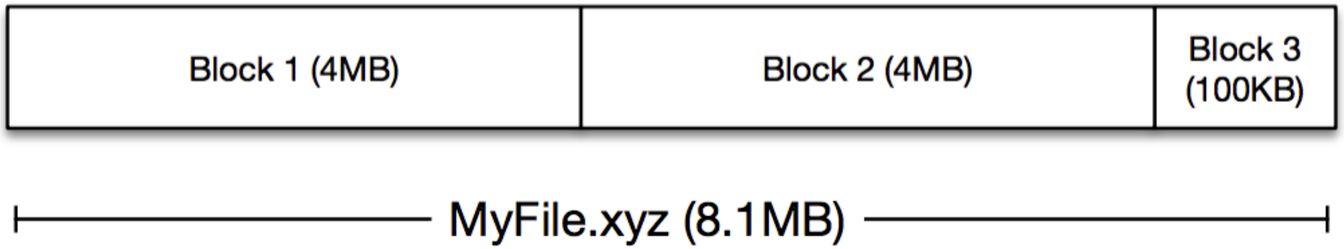
Scale

- 1, 000 PB+ user block data
- 3+ geographic regions
- 600+ million users
- 300k business customers
- *12k+ OSD machines*
- *650k+ disks*
- Basejump!

Magic Pocket

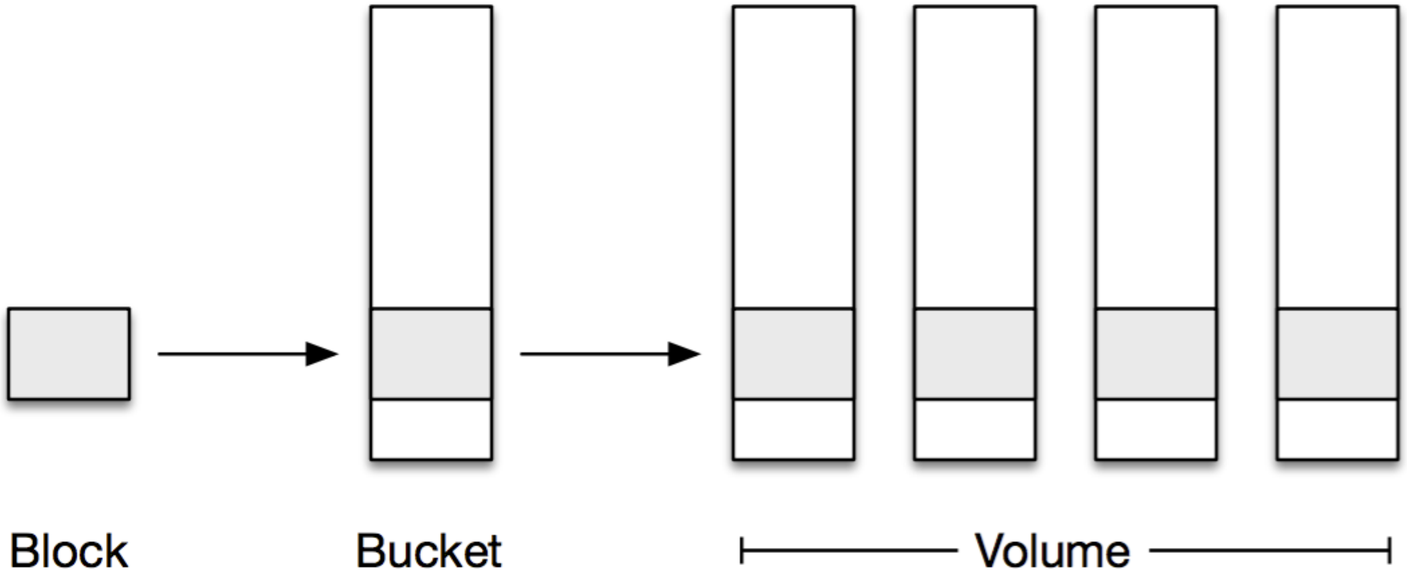
- File content and metadata about files and users
- Data is encrypted at rest with durability over ten 9s
- Durability is the top priority for MP
- Stored blocks are immutable for simplicity

Data Model



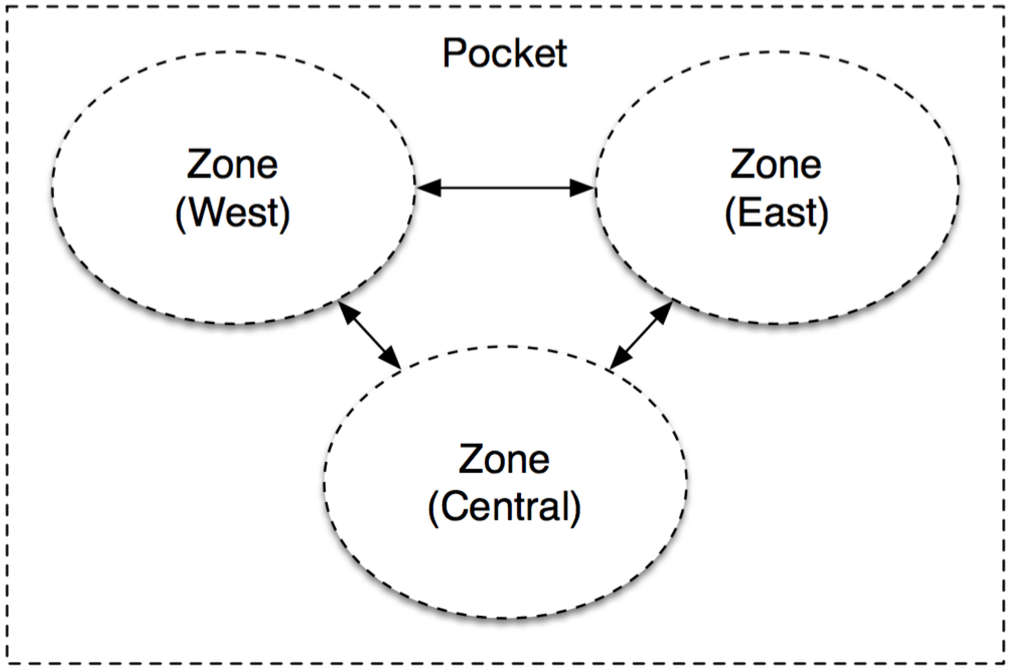
- Files are split in up to 4MB blocks
- Block key or hash is SHA-256 of the block
- Keys are stored in sharded MySQL databases
- MP stores compressed & encrypted blocks

Buckets



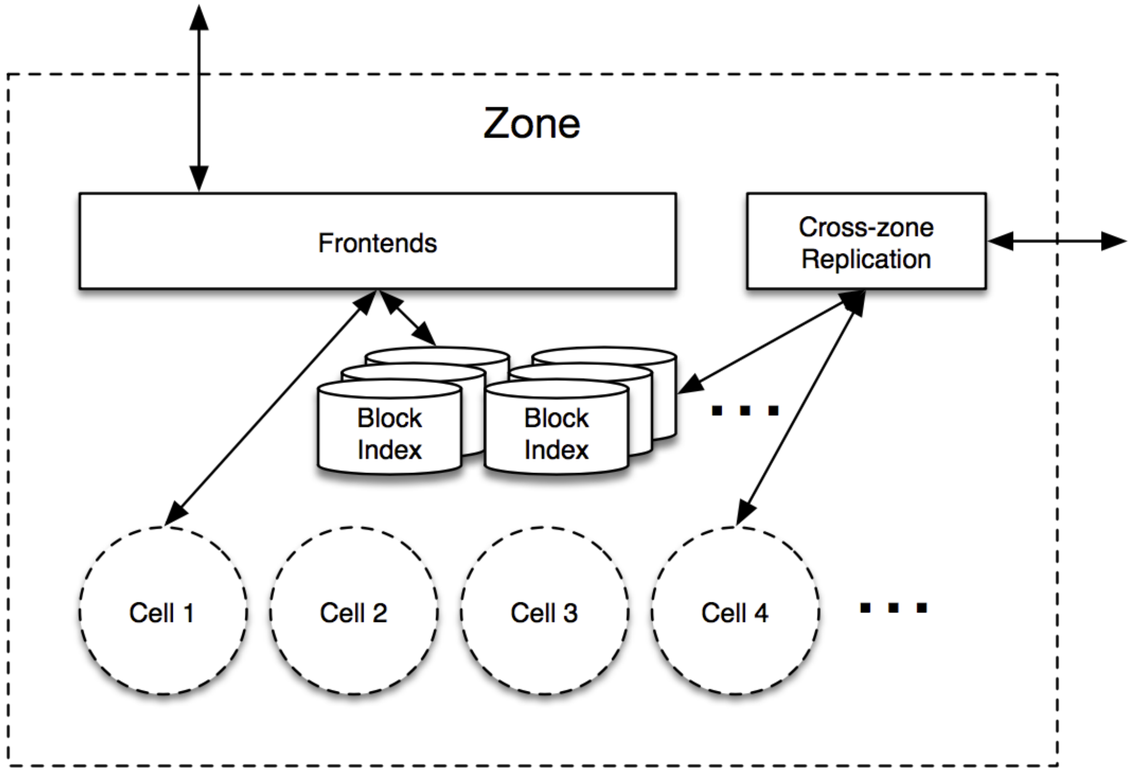
- Blocks are aggregated into 1GB logical buckets
- Buckets are replicated across multiple machines
- Replicated buckets are grouped in volumes
- Only a small number of volumes are open for writes
- Volumes are then erasure coded for efficiency

Bird’s-eye view



- Each block is stored in at least two zones
- Zones are independent of each other
- Protects against natural & misconfig disasters

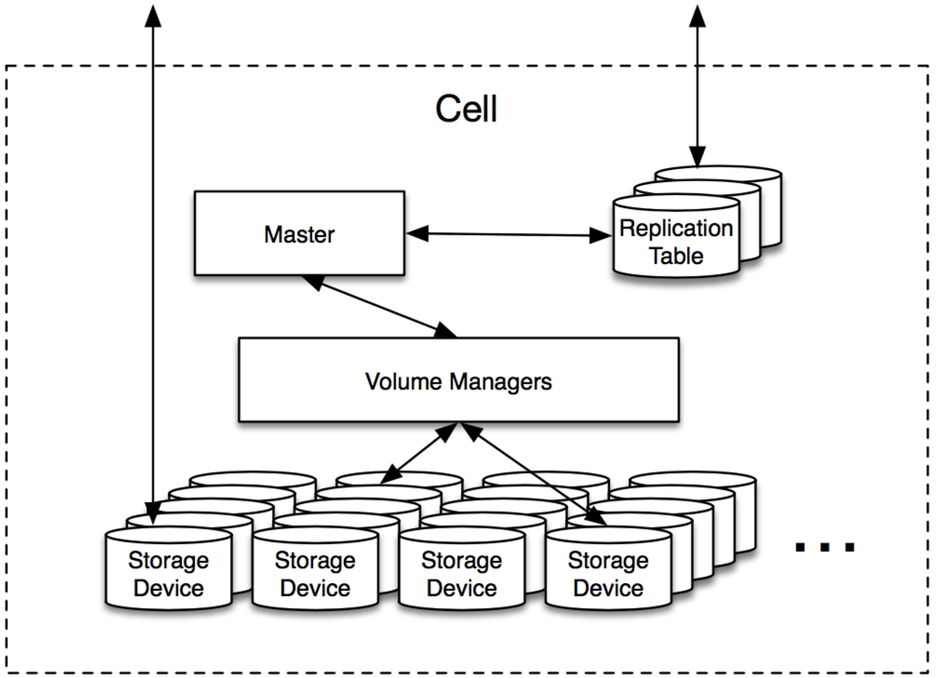
Inside a zone



hash → cell, bucket, checksum

- Frontends determine where to put or get a block
- Block Index maps blocks to buckets in cells
- Cross-zone replication for async replication
- Cells are logical storage clusters with size ~50PB

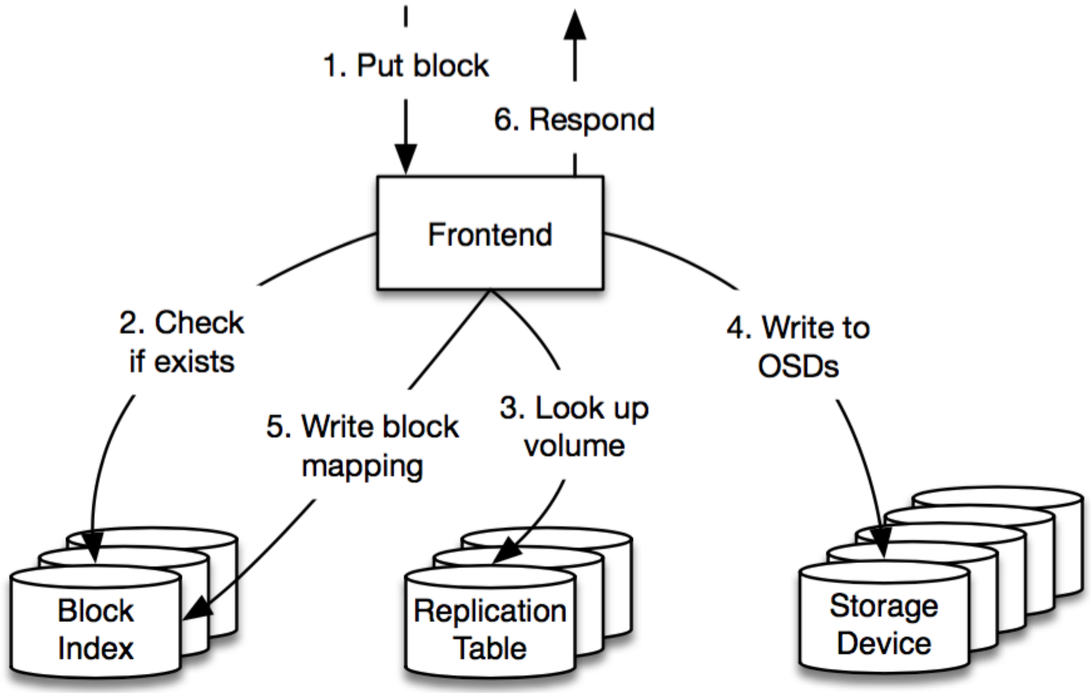
Cells



bucket → volume
volume → OSDs, open, type, generation

- OSDs store blocks of data
- Replication Table maps buckets to volumes and specific OSDs where blocks are stored
- Master is a coordinator in a given cell
- Volmgrs processes requests from masters

Protocol



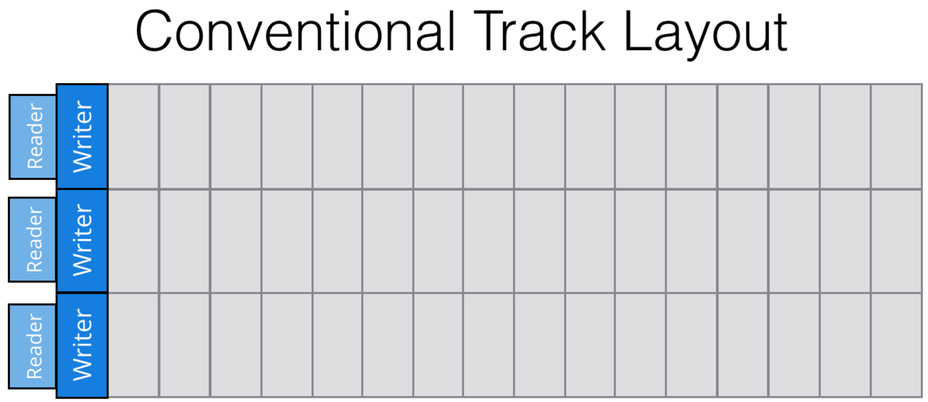
- Frontends do the most of the work
- FEs store and get information from BI and RT
- PUT operation retry may choose a different volume, cell or forward a request to another zone
- During a GET, FE consults BI & RT to find OSDs

Object Storage Devices (OSD) and next-gen disks

Object Storage Devices

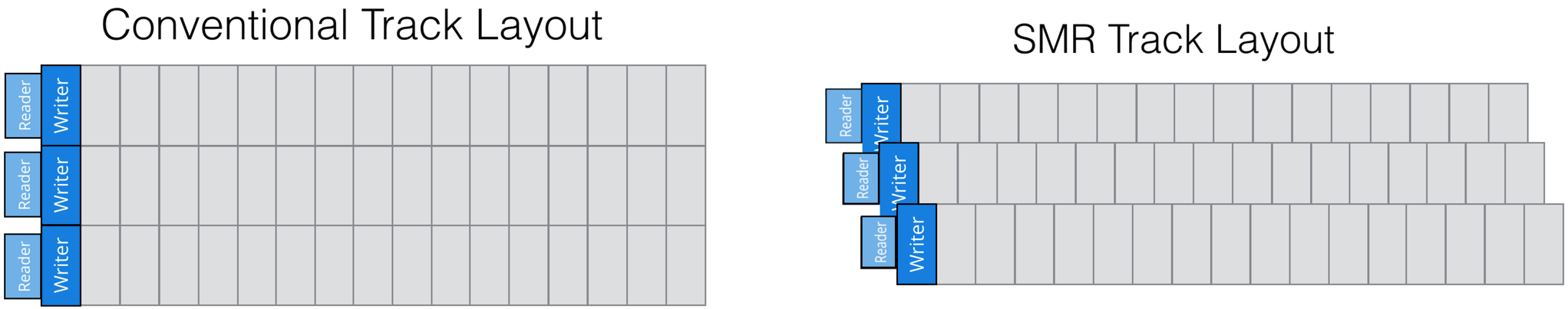
- 40 to 100 disks on each OSD machine
- Disk sizes vary from 6TB PMR to 14TB SMR
- Different hardware classes and revisions
- OSD2 has no filesystem layer

Perpendicular Magnetic Recording



- Not scalable due to physical limitations
- Reducing grain size on media leads to lower energy barriers which in turn may cause a bit flip
- High costs in order to keep data integrity and gain storage capacity

Shingled Magnetic Recording



- Sequential writes only
- Better cost in terms of density (€/GB)
- Tracks get *shingled* to the width of reader head
- Writer head is wider for better retention & readability
- Bands, zones and conventional areas

Types of SMR drive management

- Drive managed
- Host managed
- Host aware

Drive managed

- SMR drive firmware manages all IO
- Host is unaware that the disk is SMR
- Unpredictable performance
- Writes are buffered and later written sequentially
- Firmware manages zone overwriting
- Can be deployed anywhere

Host managed

- Host manages all IO
- Writes are only sequential and within a zone
- Non sequential writes fail
- Performance is predictable
- Traditional filesystems may not work in this mode
- F2FS filesystem natively supports HM mode
- ZBC/ZAC libraries to interact with disks

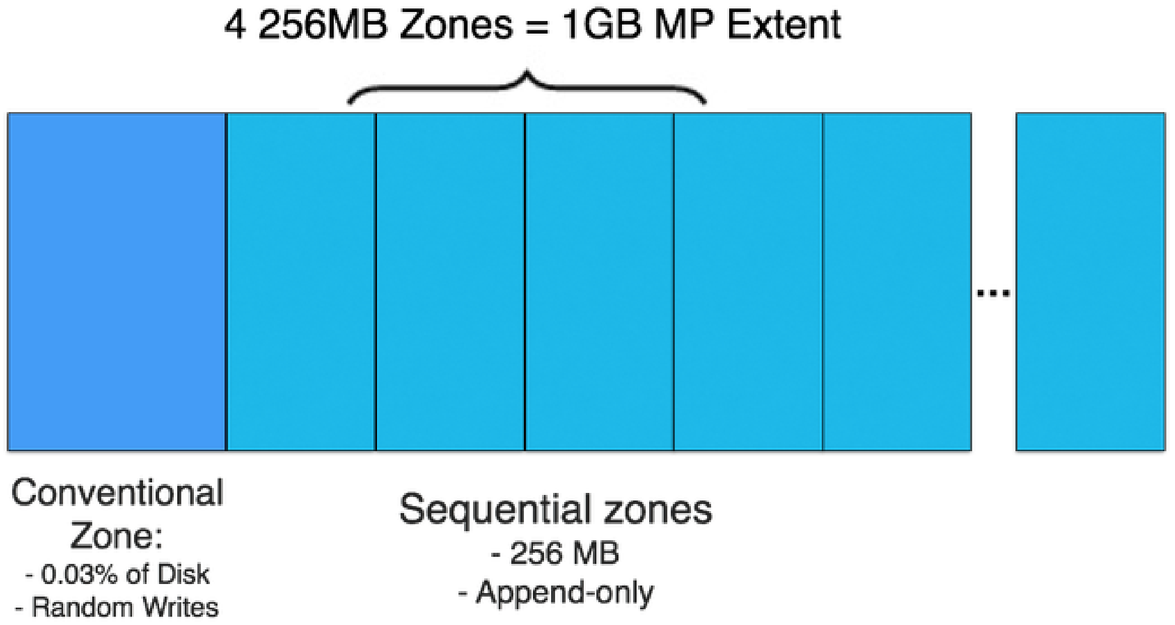
Host aware

- Combination of DM and HM modes
- Drive is self managed but also implements ZBC/ZAC
- Non sequential writes are accepted with perf impact
- Backwards compatible with traditional setups
- Suits well as an interim solution

Hardware considerations

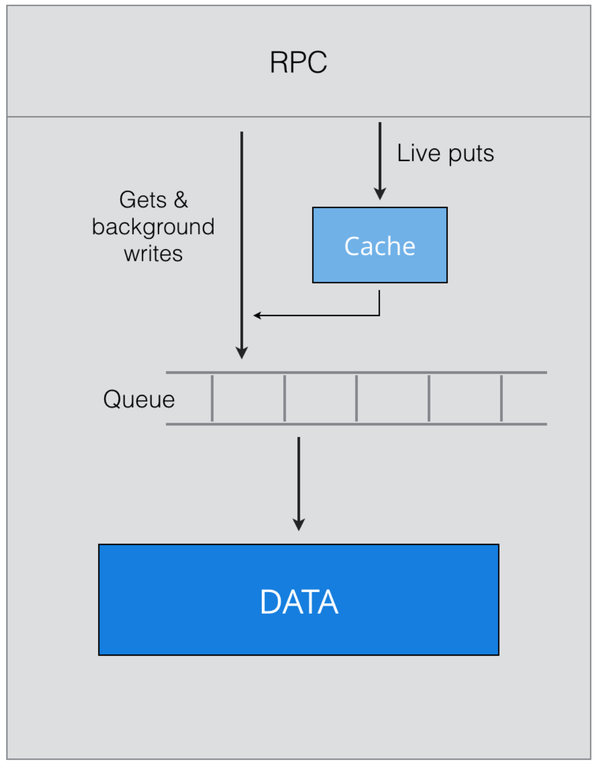
- Chassis with 100 14TB LFF disks & 8 chassis per rack
- Memory increased to 96GB to sustain the load
- CPU with 20 cores / 40 threads
- HBAs instead of RAID controllers for reliability and simplicity
- SSD disks for write caching
- 50 Gbps NIC card per chassis with 100 Gbps uplink

Software redesign



- OSD directly manages data on disk w/o a filesystem
- Libzbc to manipulate disks with ZBC/ZAC command sets
- MP filesystem metadata is stored in conventional zones
- Large batched sequential writes (~4-5 MB)
- Moved from Go to Rust for performance and control

Live traffic



- SSD disk(s) for live PUT operations
- GETs and background writes are sent to disks
- OSDs manage to align and flush data from cache to disks

Conclusion

- Magic Pocket is a heart of our business
- Optimisation of costs due to SMR storage
- Better control & performance by avoiding filesystems
- Rust language gave us full control over memory & GC
- Immutable append only architecture goes well with SMR