Scaling the EOS namespace

Georgios Bitzes, on behalf of the EOS team

Workshop on Performance and Scalability of Storage Systems, Frankfurt, 2017

What is EOS?

- Distributed storage system developed since 2010 at CERN
- Today: several production instances, **224 PB** of disk storage, **1.4 billion files**

Number of Files

1438 M

Number of Directories

114 M

Total Space

224 PB

Write Throughput

9.99 GBps

Read Throughput

29.7 GBps

Current Readers

37.7 K

Current Writers

9.5 K

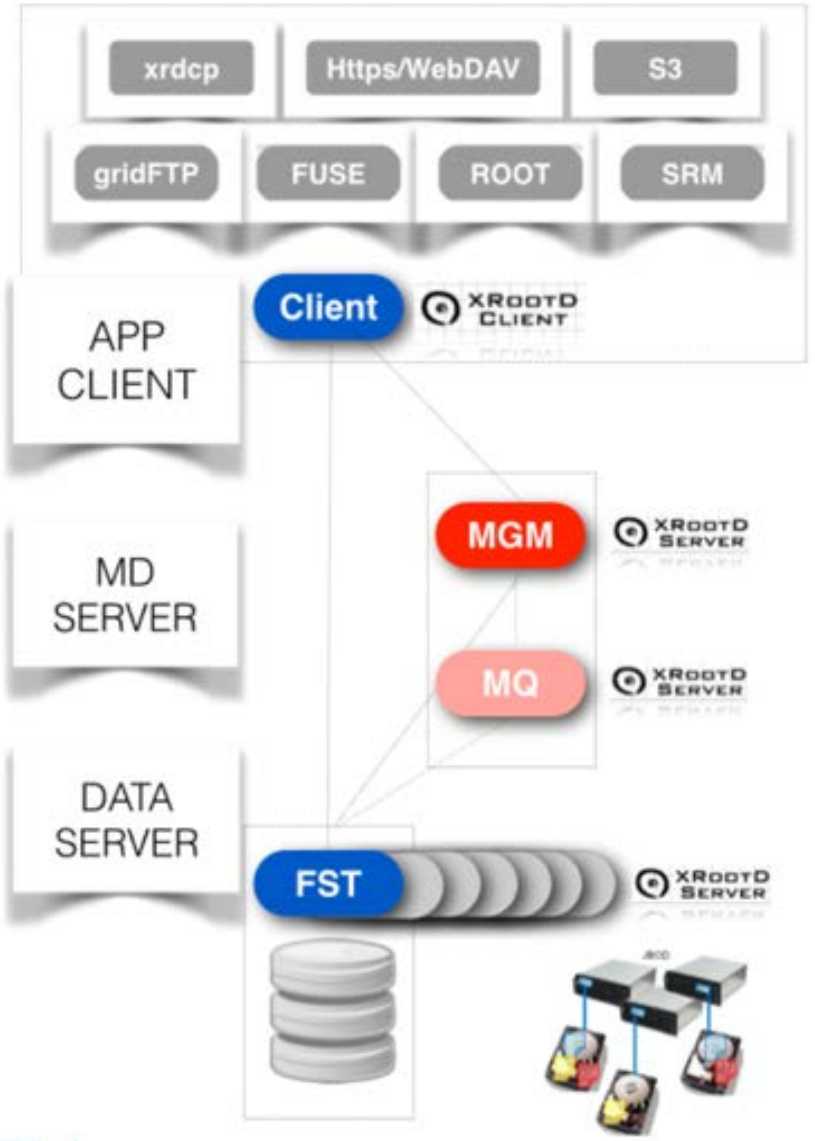
Free Space

94.12 PB

IOPS

Architecture

- File Storage Nodes **(FST)**: Management of physical disks, file serving
- Metadata Servers **(MGM)**: Namespace + client redirection to FSTs
- Message Queue **(MQ)**: Inter-cluster communication, heartbeats, configuration changes



The namespace subsystem

- EOS presents **one single** namespace to files it manages
  - … even though they are typically spread across **hundreds** of disk servers and **thousands** of physical disks
- Handles file permissions, metadata, quota accounting, **mapping** between logical filenames and physical locations

Sample file entry

Logical filename: /eos/somedir/filename

Inode number: 134563

Parent directory inode: 1234, referring to /eos/somedir

Size: 19183 bytes

File layout: 2 replicas

- **Physical replica 1:** Filesystem #23, referring to fst-1.cern.ch:/mnt34/
- **Physical replica 2:** Filesystem #45, referring to fst-2.cern.ch:/mnt11/

Checksum: md5-567c100888518c1163b3462993de7d47

In-memory namespace implementation

- The MGM always holds the **entire** namespace in-memory. Each file / directory entry allocates up to 1kb as a C++ structure in memory.

- Linear on-disk **changelogs** to track all namespace changes
  - file additions, metadata changes, quota tracking, physical location migrations …
  - One for files, one for directories

- The in-memory contents are **reconstructed** on reboot by replaying the changelogs

In-memory namespace implementation (2)

- The good
  - It's **fast**. Lookups done with a Google dense hashmap, no I/O.
  - Reliable and **proven**.

- The bad

  - **Long** boot time, proportional to the number of files on an instance. For large namespaces can exceed **1hour**
  - Requires **a lot** of RAM: each file takes up around 1kb of memory on the MGM

Namespace on top of a datastore

- Requirements: **consistent** low latency, scalable, very high rate of writes per second
- EOS to replace AFS at CERN, hold network home directories
- Needs to be reasonably performant for tasks such as…
  - interactive usage
  - compiling
  - untarring archives the size of the linux kernel

Choosing a persistent datastore

- **RDBMs:** scalability issues, complicates our setup

- **Redis:** scalable and fast, but…
  - high per-entry RAM overhead
  - redis cluster can lose acknowledged writes

- **Cassandra:** scales very well, but…
  - adds significant complexity to setup & operation
  - performance / resource cost is high

Choosing a persistent datastore (2)

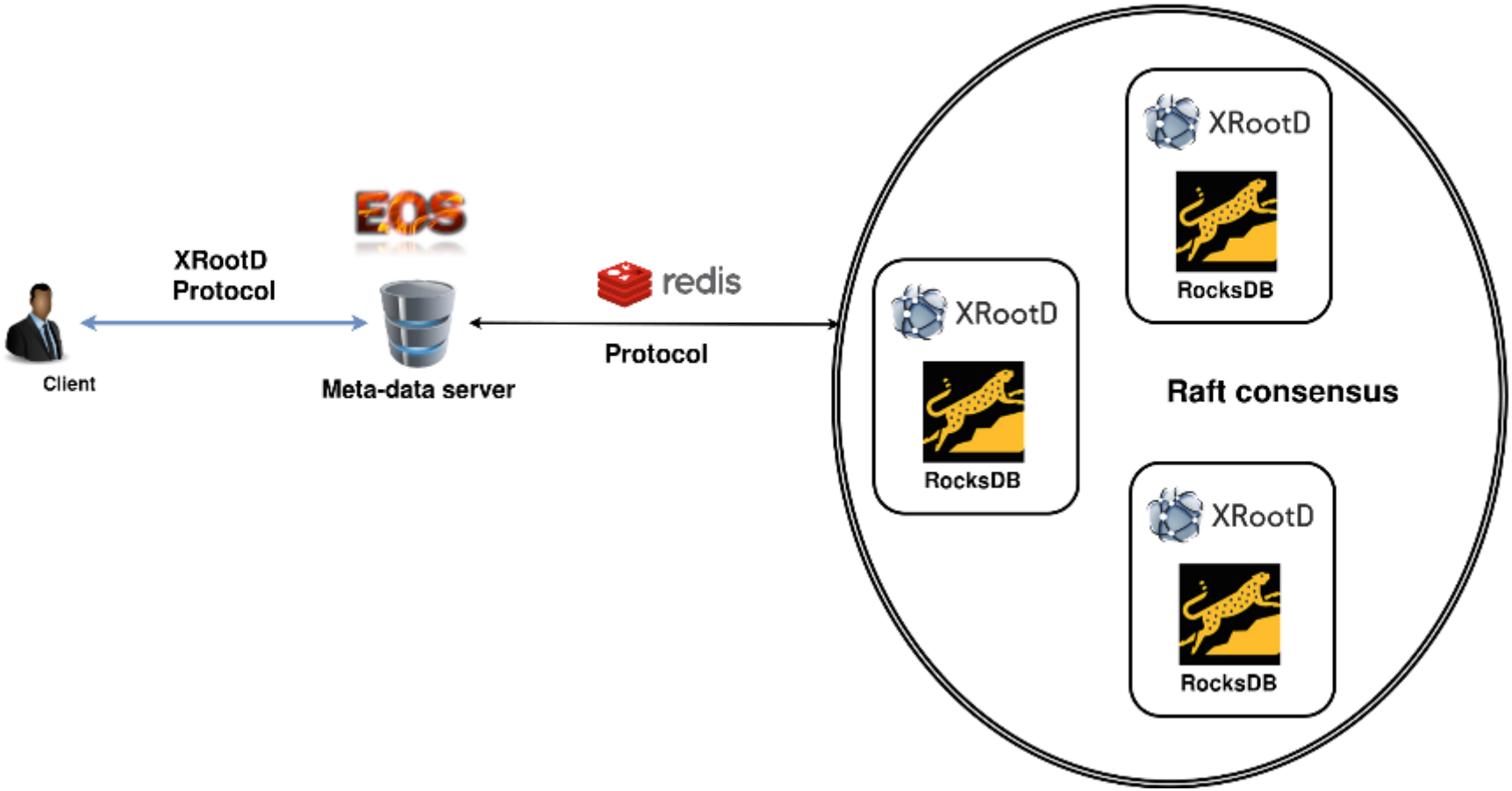
- We chose **RocksDB**: fast, embeddable KV store by Facebook

- Open source (C++), very high read-write rates, scales with number of threads

QuarkDB: a highly-available datastore

- Implement the minimum necessary to keep the system simple
  - QuarkDB runs as a plug-in to the XRootD server framework used by EOS
- A redis-like server on top of RocksDB
  - Support for a subset of the redis command-set: `HASH`, `STRING`, `SET` operations
- High availability through multiple **strongly-consistent** replicated nodes
  - Raft consensus algorithm to keep replicas in sync

Overview


EOS
XRootD
Protocol
Client
Meta-data server
redis
Protocol
XRootD
RocksDB
XRootD
RocksDB
Raft consensus
XRootD
RocksDB

Some QuarkDB numbers

- **10k** lines of C++ (including tons of tests)
- Preliminary benchmarks: peak of **100khz** 200-byte writes, **300khz** reads (non-replicated mode)
- Replicated performance currently **10–15 khz** writes – plans to improve through automatic sharding

Summary

- EOS is the current and future solution for data storage at CERN
  - physics / analysis data
  - networked filesystem, user home directories
- Current in-memory namespace implementation based on changelogs reaching its limits
- New scalable namespace on top of a persistent, highly available key-value store based on XRootD framework

Thanks

Links

https://gitlab.cern.ch/dss/eos

https://gitlab.cern.ch/eos/quarkdb

Comments, questions?

Georgios.Bitzes@cern.ch


CERN