The Viability of Ceph

David Bonnie, Christopher Hoffman, Dominic Manno

June 4th, 2015

Overview

- **What is Ceph?**
  - Architecture
  - Components
  - Interfaces
  - Internals
- **Object storage**
- **Erasure coding basics**
- **Why is this important?**

What is Ceph?

- **Open source, object based storage**
- **Software based solution**
- **Runs on commodity hardware**
- **Scalable to exabyte and beyond**
- **No single point of failure**
- **Makes use of advanced data reliability techniques (replication, erasure)**
  - Different erasure profiles
  - Availability of multiple erasure algorithms
- **Many different interfaces (RADOS, RBD, CephFS)**

Ceph Architecture

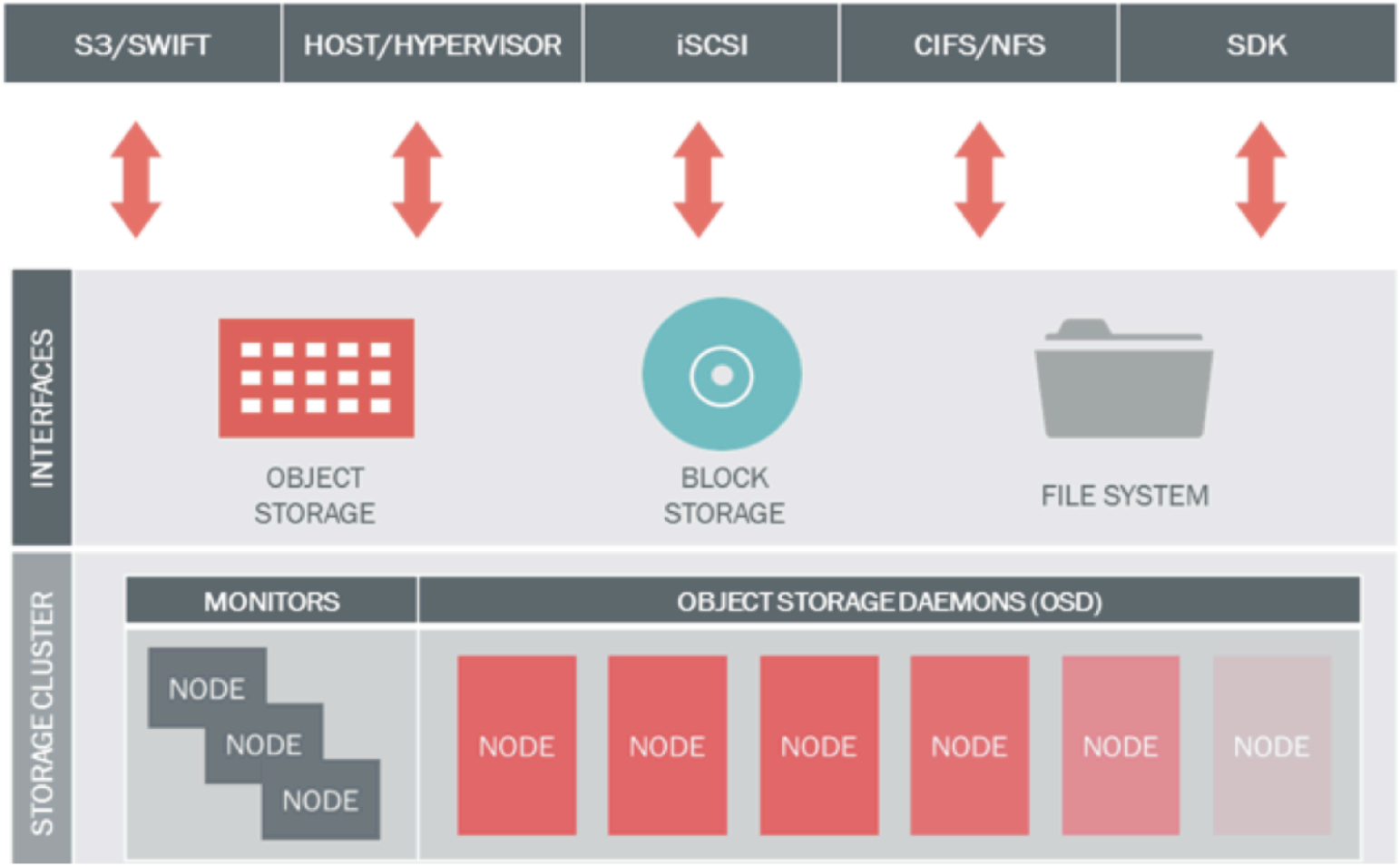


Image by Inktank

Ceph Components

- **Object Storage Daemons (OSDs)**
  - Store data objects
  - Manage replication
  - Data recovery
- **Ceph Mon (monitors)**
  - Receive state information from other components
  - Updates maps' state of different components to accomplish this
    - Monitor, OSD, Placement Group (PG), and CRUSH

Ceph Interfaces

- **Ceph Raw Block Device (RBD)**
  - Provides block storage to clients
- **Ceph Rados Gateway (RGW)**
  - Object gateway
  - Interface with RESTful (Amazon S3, OpenStack swift)
- **CephFS (file system)**
  - Parallel file system
  - Requires additional component, Ceph Metadata Server (MDS)
    - Controls metadata of CephFS to prevent load on OSDs

Ceph Internals

- **Pools**
  - Logical group for storing objects
  - Tell it how many PGs and replicas (or m coding value for erasure)
- **Placement groups (PGs)**
  - Ceph maps object -> PG
  - PGs are spread across OSDs
  - Multiple objects in a PG
  - Multiple PGs in a pool
- **CRUSH**
  - Algorithm to determine data storage location
  - Efficiently maps objects to devices
  - Decentralized distribution across storage cluster while following defined policies

Object Storage

- Application layer access
- APIs, APIs, APIs
- No directory tree, flat storage utilizing containers
- Metadata resides with object
- Very scalable

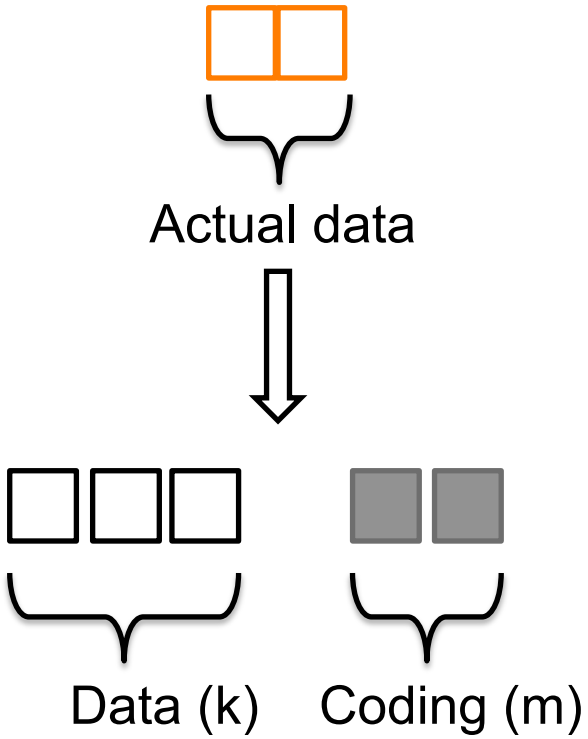
Erasure Coding Basics

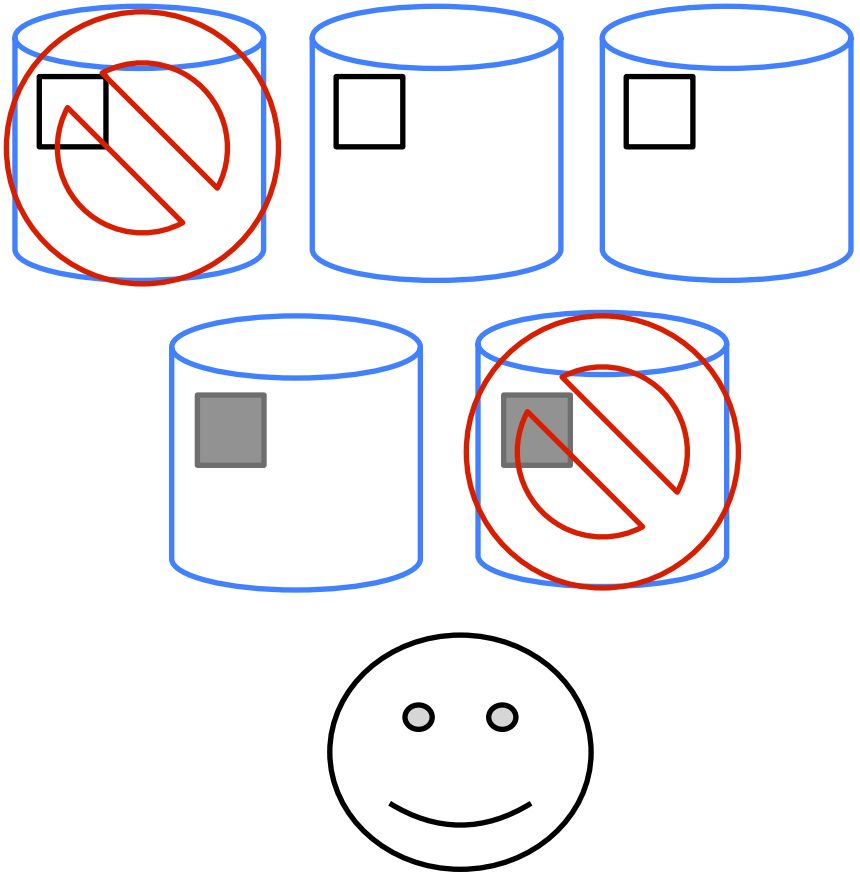
- Data fragmented into k chunks (data chunks)
- Also redundant pieces, m (coding chunks)
- End up with k+m=n total chunks
- Encoded to allow reproduction of data
- Must have x, where x ≥ k, chunks out of n available to prevent loss of access or data
- Most common error-correcting code, Reed-Solomon

Erasure Coding Basics

- Flexible configuration
- Requires less storage than replication
- Encoding rate, r = k/n where r < 1
- Required storage for file = 1/r *original file size
- Much more resilient than RAID
- RAID6 implements Reed-Solomon error correction, but you are limited to the m value (discussed earlier) of 2.

Erasure Coding Basics (very simple 3,2)





Why is this important?