Petabyte Scale Out Rocks

CEPH as Replacement for Openstack's Swift and Co.

Dr. Udo Seidel

Manager of Linux Strategy @ Amadeus

useidel@amadeus.com

Agenda

- Background
- Architecture of CEPH
- CEPH Storage
- OpenStack
- Dream Team: CEPH and Openstack
- Summary

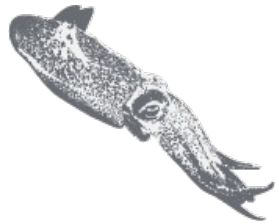
Background

Me :-)

- Teacher of mathematics and physics
- PhD in experimental physics
- Started with Linux in 1996
- Linux/UNIX trainer
- Solution engineer in HPC and CAx environment
- Head of the Linux Strategy team @Amadeus

CEPH – What?

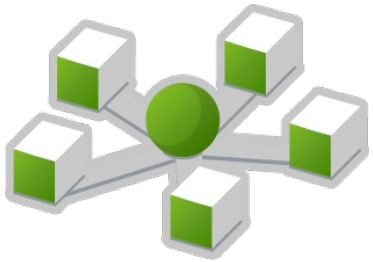
- So-called parallel distributed cluster file system
- Started as part of PhD studies at UCSC
- Public announcement in 2006 at 7th OSDI
- File system shipped with Linux kernel since 2.6.34
- Name derived from pet octopus – cephalopods



ceph

Shared File Systems – Short Intro

- Multiple server access the same data
- Different approaches
  - Network based, e.g. NFS, CIFS
  - Clustered
    - Shared disk, e.g. CXFS, CFS, GFS(2), OCFS2
    - Distributed parallel, e.g. Lustre .. and CEPH



Architecture of CEPH

CEPH and Storage

- Distributed file system => distributed storage
- Does not use traditional disks or RAID arrays
- Does use so-called OSDs
  - Object based Storage Devices
  - Intelligent disks

Object Based Storage I

- Objects of quite general nature
  - Files
  - Partitions
- ID for each storage object
- Separation of metadata operation and storing file data
- HA not covered at all
- Object based Storage Devices

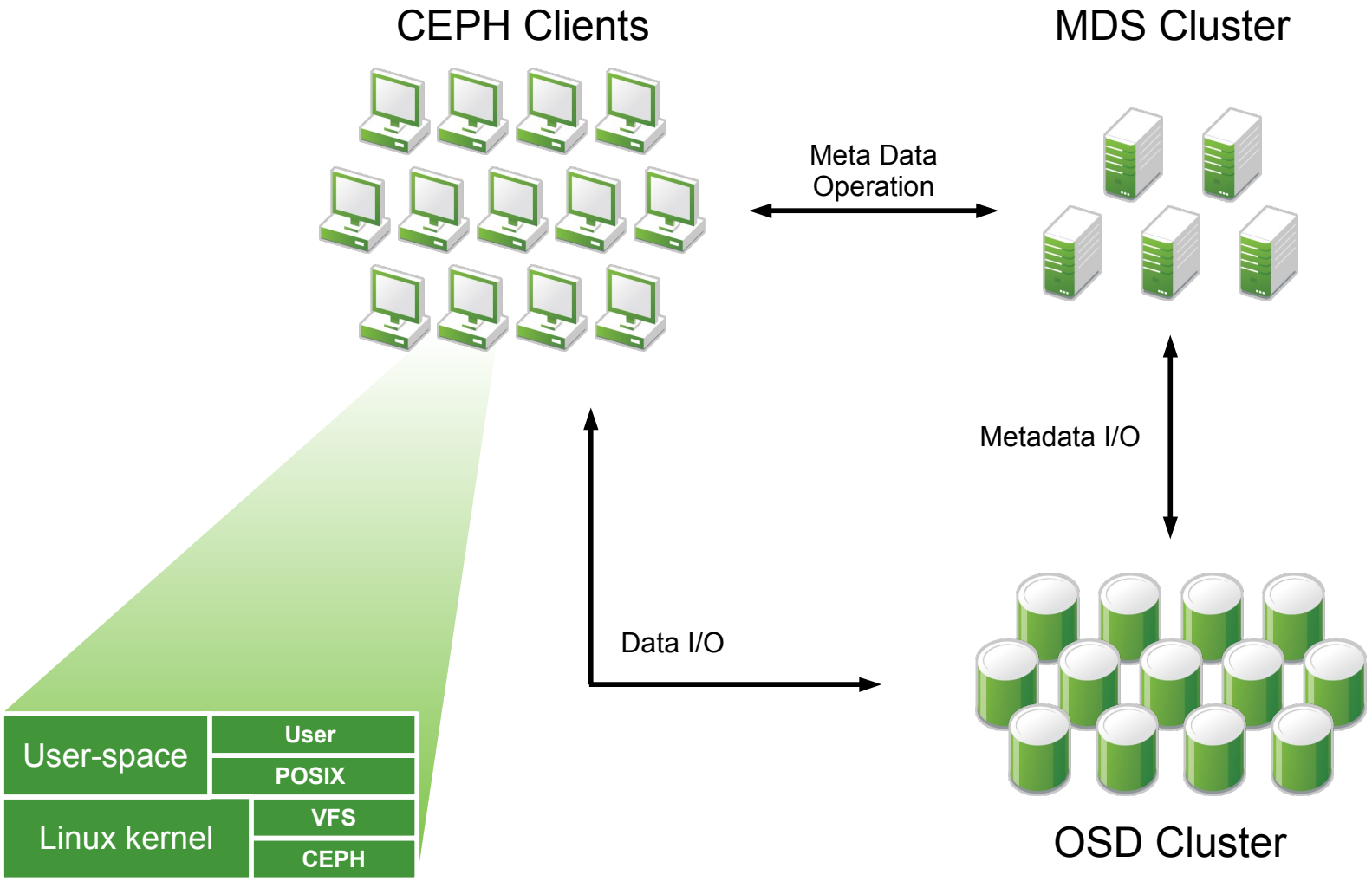
Object Based Storage II

- OSD software implementation
  - Usual an additional layer between between computer and storage
  - Presents object-based file system to the computer
  - Use a “normal” file system to store data on the storage
  - Delivered as part of CEPH
- File systems: Lustre, EXOFS

CEPH – The Full Architecture I

- 4 Components
  - Object-based Storage Devices
    - Any computer
    - Form a cluster (redundancy and load balancing)
  - Metadata Servers
    - Any computer
    - Form a cluster (redundancy and load balancing)
  - Cluster Monitors
    - Any computer
  - Clients ;-)

CEPH – The Full Architecture II



CEPH Storage

CEPH and OSD

- Userland implementation
- Any computer can act as OSD
- Uses XFS or BTRFS as native file system
  - Since 2009
  - Before self-developed EBOFS
  - Provides functions of OSD-2 standard
    - Copy-on-write
    - Snapshots
- No redundancy on disk or even computer level

CEPH and OSD – File Systems

- XFS strongly recommended
- BTRFS planned as default
  - Non-default configuration for mkfs
- EXT4 possible
  - XATTR (size) is key -> EXT4 less recommended
  - Leveldb to bridge existing gaps

OSD Failure Approach

- Any OSD expected to fail
- New OSD dynamically added/integrated
- Data distributed and replicated
- Redistribution of data after change in OSD landscape

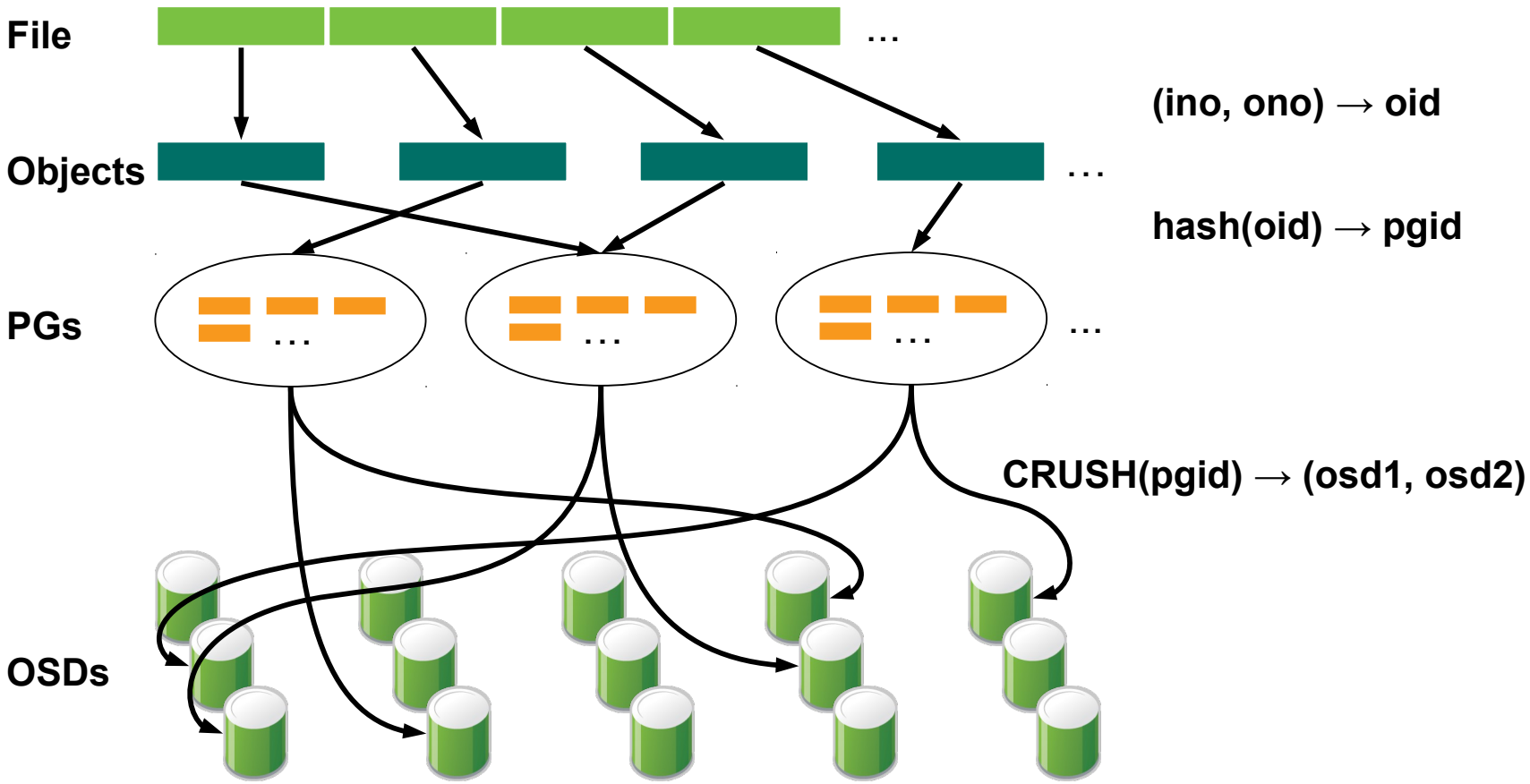
Data Distribution

- File stripped
- File pieces mapped to object IDs
- Assignment of so-called placement group to object ID
  - Via hash function
  - Placement group (PG): logical container of storage objects
- Calculation of list of OSDs out of PG
  - CRUSH algorithm

CRUSH I

- Controlled Replication Under Scalable Hashing (CRUSH)
- Considers several information
  - Cluster setup/design
  - Actual cluster landscape/map
  - Placement rules
- Pseudo random -> quasi statistical distribution
- Cannot cope with hot spots
- Clients, MDS and OSD can calculate object location

CRUSH II


File
Objects
PGs
OSDs
(ino, ono) → oid
hash(oid) → pgid
CRUSH(pgid) → (osd1, osd2)

Data Replication

- N-way replication
  - N OSDs per placement group
  - OSDs in different failure domains
  - First non-failed OSD in PG -> primary
- Read and write to primary only
  - Writes forwarded by primary to replica OSDs
  - Final write commit after all writes on replica OSD
- Replication traffic within OSD network

CEPH clustermap I

- **Objects:** computers and containers
- **Container:** bucket for computers or containers
- Each object has ID and weight
- Maps physical conditions
  - Rack location
  - Fire cells

CEPH clustermap II

- Reflects data rules
  - Number of copies
  - Placement of copies
- Updated version sent to OSDs
  - OSDs distribute cluster map within OSD cluster
  - OSD re-calculates via CRUSH PG membership
    - Data responsibilities
    - Order: primary or replica
  - New I/O accepted after information sync

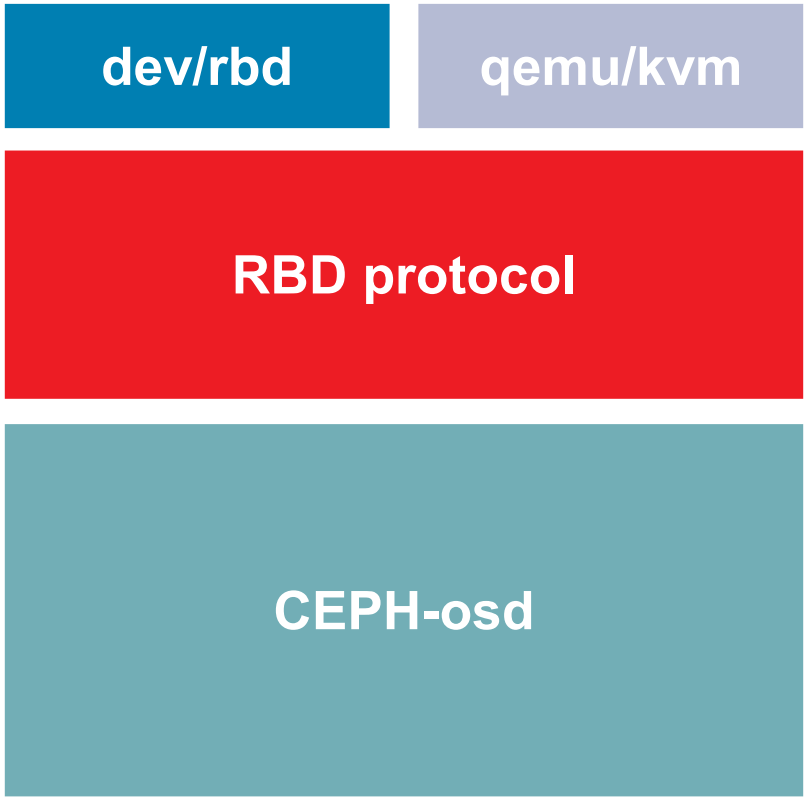
CEPH - RADOS

- Reliable Autonomic Distributed Object Storage (RADOS)
- Direct access to OSD cluster via *librados*
  - Supported: C, C++, Java, Python, Ruby, PHP
- Drop/skip of POSIX layer (CEPHFS) on top
- Visible to all CEPH members => shared storage

CEPH Block Device

- Aka RADOS Block Device (RBD)
- Second part of the kernel code (since 2.6.37)
- RADOS storage exposed as block device (*dev/rbd*)
  - qemu/KVM storage driver via *librados/librdb*
- Alternative to:
  - Shared SAN/iSCSI for HA environments
  - Storage HA solutions for qemu/KVM and Xen
- Shipped with SUSE® Cloud

RADOS – A Part of the Picture



CEPH Object Gateway (RGW)

- Aka RADOS Gateway
- RESTful API
  - Amazon S3 -> s3 tools work
  - OpenStack SWIFT interface
- Proxy HTTP to RADOS
- Tested with apache and lighthttpd

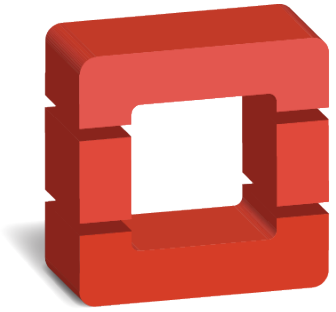
CEPH – Takeaways

- Scalable
- Flexible configuration
- No SPOF, but based on commodity hardware
- Accessible via different interfaces

OpenStack

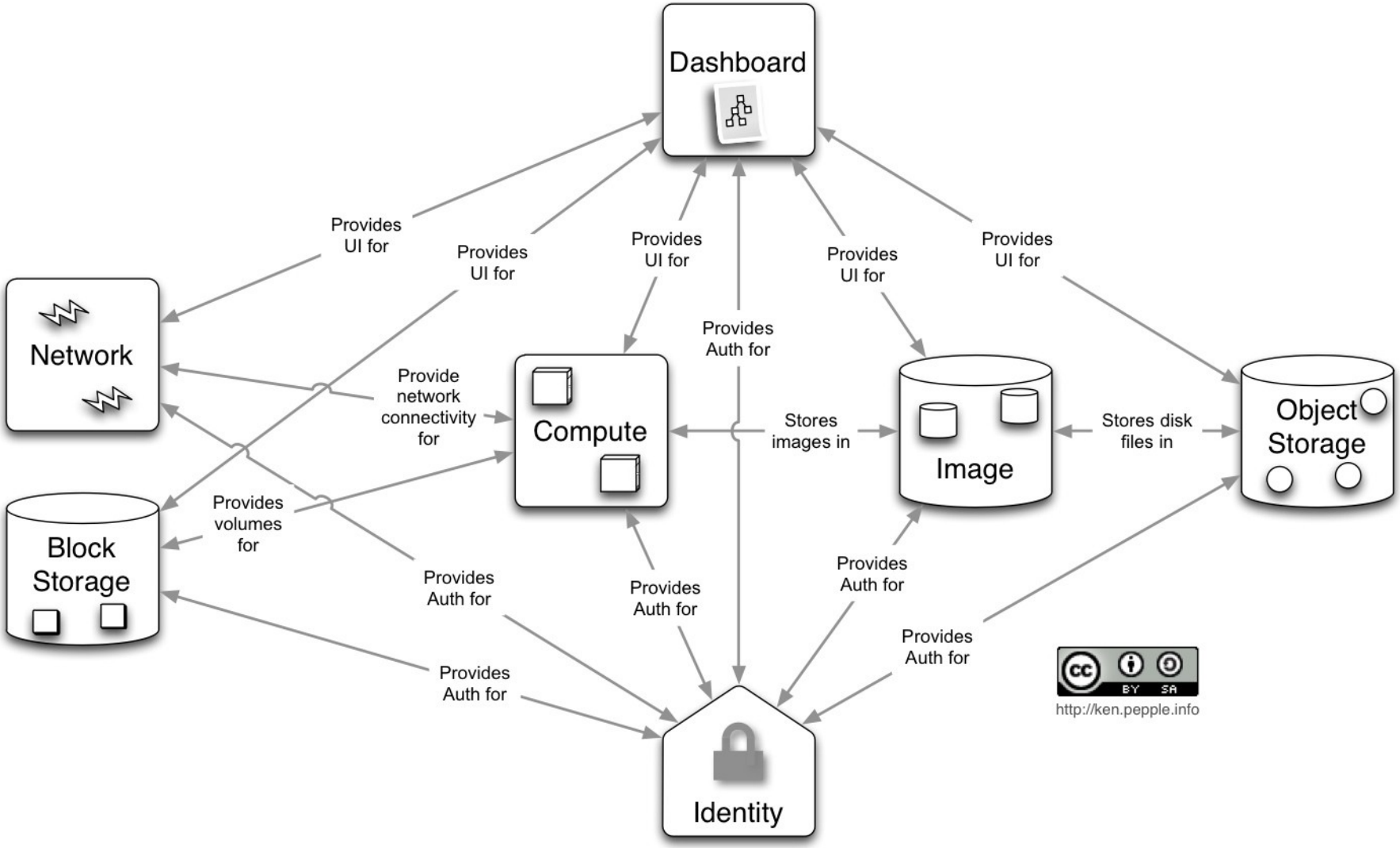
What?

- Infrastructure as as Service (IaaS)
- 'Open source version' of AWS :-)
- New versions every 6 months
  - Current (2013.1.4) is called Havana
  - Previous (2013.1.3) is called Grizzly
- Managed by OpenStack Foundation



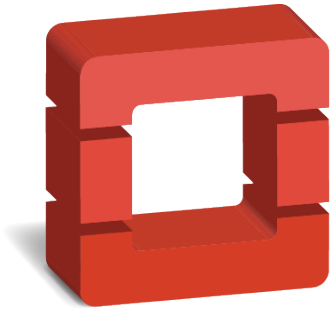
openstack™

OpenStack Architecture



OpenStack Components

- Keystone - identity
- ***Glance*** *- image*
- Nova - compute
- ***Cinder*** *- block storage*
- ***Swift*** *- object storage*
- Quantum - network
- Horizon - dashboard



openstack™

About Glance

- There since almost the beginning
- OpenStack's Image store
  - Server
  - Disk
- Several formats possible
  - Raw
  - Qcow2
  - ...
- Shared/non-shared

About Cinder

- OpenStack's block storage
- Quite new
  - Part of nova before
  - Separated since Folsom
- For compute instances
- Managed by OpenStack users
- Several storage back-ends possible

About Swift

- Since the beginnig :-)
- Replace Amazon S3 -> cloud storage
  - Scalable
  - Redundant
- OpenStack's object store
  - Proxy
  - Object
  - Container
  - Account
  - Auth

Dream Team: CEPH and OpenStack

Why CEPH in the first place?

- Full blown storage solution
  - Support
  - Operational model
  - Ready for 'cloud' type
- Separation of duties
- One storage solution for different storage needs

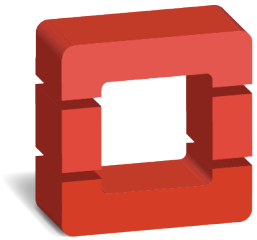
Open source!

Integration

- Full integration with RADOS since Folsom
- Two parts
  - Authentication
  - Technical access
- Both parties must be aware
  - CEPH components installed in OpenStack environment
- Independent for each of the storage components
  - Different RADOS pools

Integration – Authentication

- CEPH part
  - Key rings (ceph auth)
  - Configuration (ceph.conf)
  - For cinder and glance
- OpenStack part
  - Keystone
    - Only for Swift
    - Needs RadosGW



openstack™

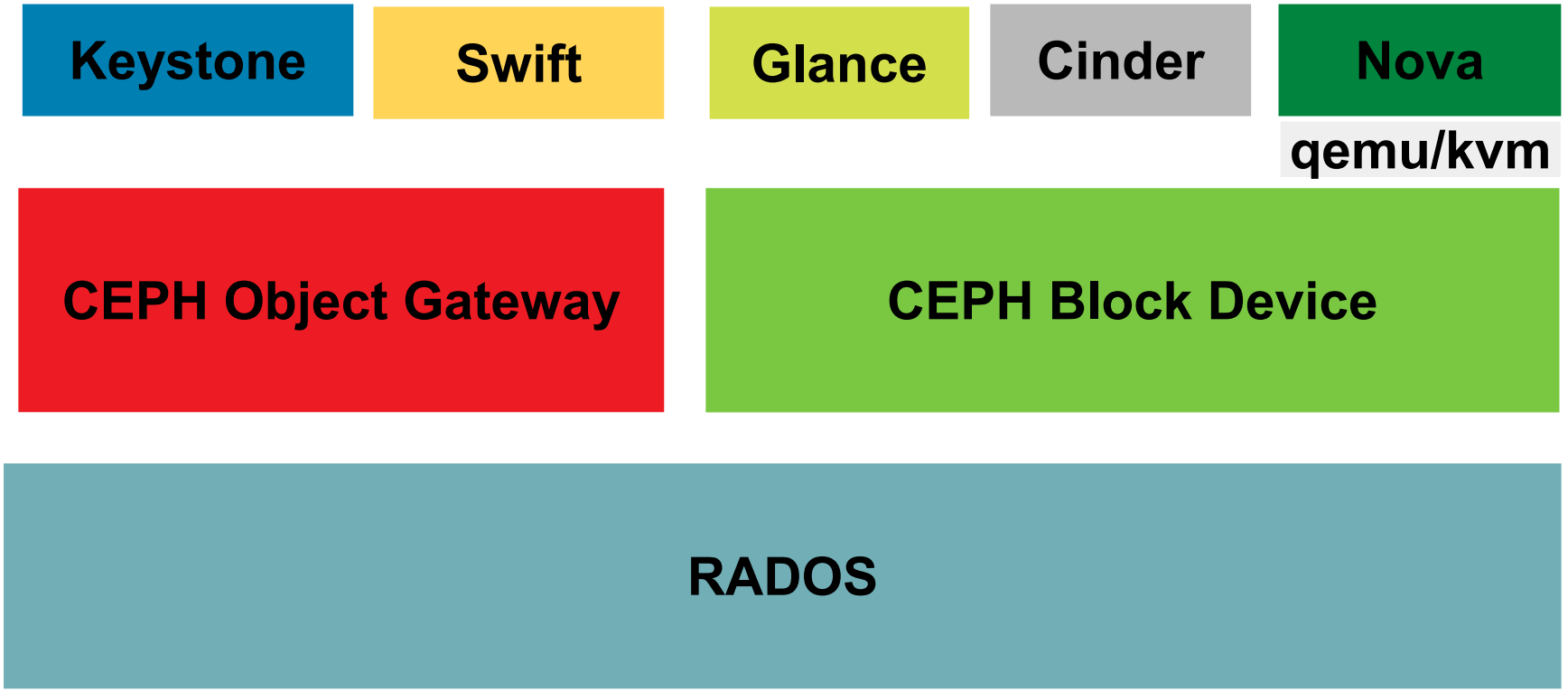
Integration – Access to RADOS/RBD I

- Via API/libraries => no CEPHFS needed
- Easy for Glance/Cinder
  - **Glance**
    - CEPH keyring configuration
    - Update of ceph.conf
    - Update of Glance API configuration
  - **Cinder**
    - CEPH keyring configuration
    - Update of ceph.conf
    - Update of Cinder API configuration

Integration – Access to RADOS/RBD II

- More work needed for Swift
- **Again: CEPHFS not needed**
  - CEPH Object Gateway
    - Webserver
    - RGW software
    - Keystone certificates
    - Does not support v2 of Swift authentication
  - Keystone authentication
    - Endlist configuration -> to RADOSGW

Integration – The Full Picture



Integration – Pitfalls

- CEPH versions not in sync
- Authentication
- CEPH Object Gateway setup

Why CEPH – Reviewed

- Previous arguments still valid :-)
- Modular usage
- High integration
- Built-in high availability
- One way to handle the different storage entities
- No need for POSIX compatible interface
- Works even with other IaaS implementations

Summary

Takeaways

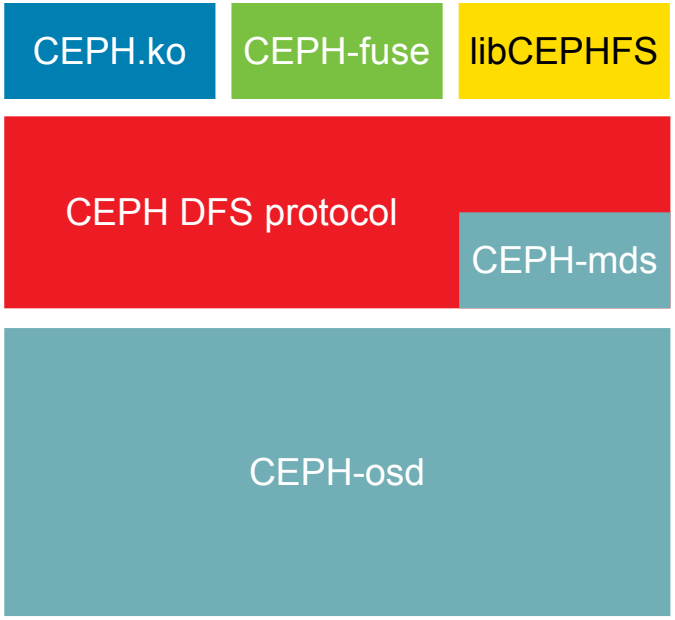
- Sophisticated storage engine
- Mature OSS distributed storage solution
  - Other use cases
  - Can be used elsewhere
- Full integration in OpenStack


SUSE®
We adapt. You succeed.

Backup Slides

CEPHFS – File System Part of CEPH

- Replacement of NFS or other DFS
- Storage just a part



CEPHFS – Client View

- The first kernel part of CEPH
- Unusual kernel implementation
  - “light” code
  - Almost no intelligence
- Communication channels
  - To MDS for meta data operation
  - To OSD to access file data

CEPHFS Caches

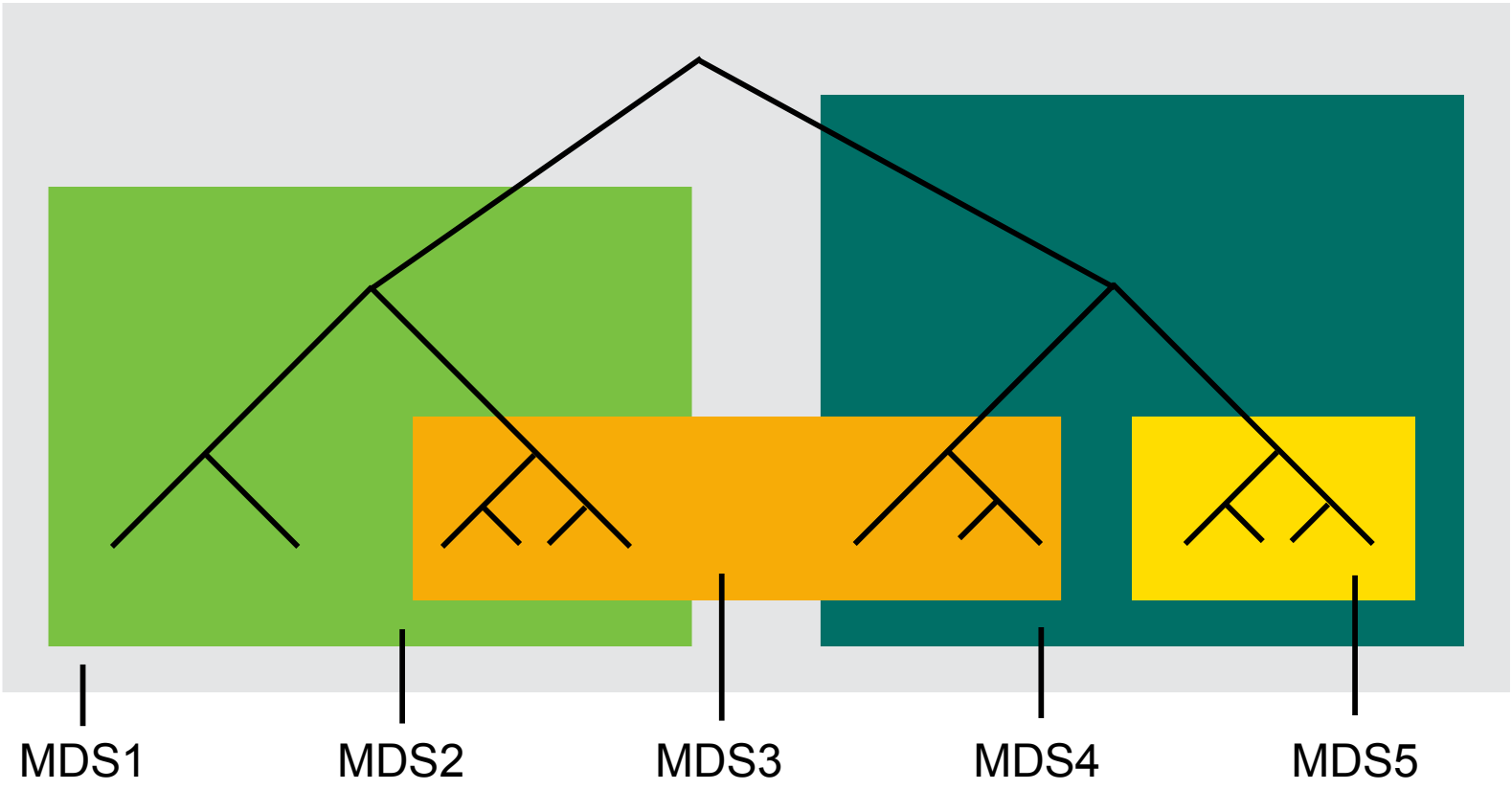
- Per design
  - OSD: Identical to access of BTRFS
  - Client: own caching
- Concurrent write access
  - Caches discarded
  - Caching disabled -> synchronous I/O
- HPC extension of POSIX I/O
  - O_LAZY

CEPH and Meta Data Server (MDS)

- MDS form a cluster
- don’t store any data
  - Data stored on OSD
  - Journaled writes with cross MDS recovery
- Change to MDS landscape
  - No data movement
  - Only management information exchange
- Partitioning of name space
  - Overlaps on purpose

Dynamic sub tree Partitioning

- Weighted subtrees per MDS
- “load” of MDS re-belanced



Meta Data Management

- Small set of meta data
  - No file allocation table
  - Object names based on inode numbers
- MDS combines operations
  - Single request for *readdir()* and *stat()*
  - *stat()* information cached

CEPH Monitors

- Status information of CEPH components critical
- First contact point for new clients
- Monitor track changes of cluster landscape
  - Update cluster map
  - Propagate information to OSDs

CEPH Storage – All In One