

Ceph

A file system a little bit different

Udo Seidel

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

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

Ceph and storage

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

Storage – looking back

- Not very intelligent
- Simple and well documented interface, e.g. SCSI standard
- Storage management outside the disks

Storage – these days

- Storage hardware powerful => Re-define: tasks of storage hardware and attached computer
- Shift of responsibilities towards storage
 - Block allocation
 - Space management
- Storage objects instead of blocks
 - Extension of interface -> OSD standard

Object Based Storage I

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

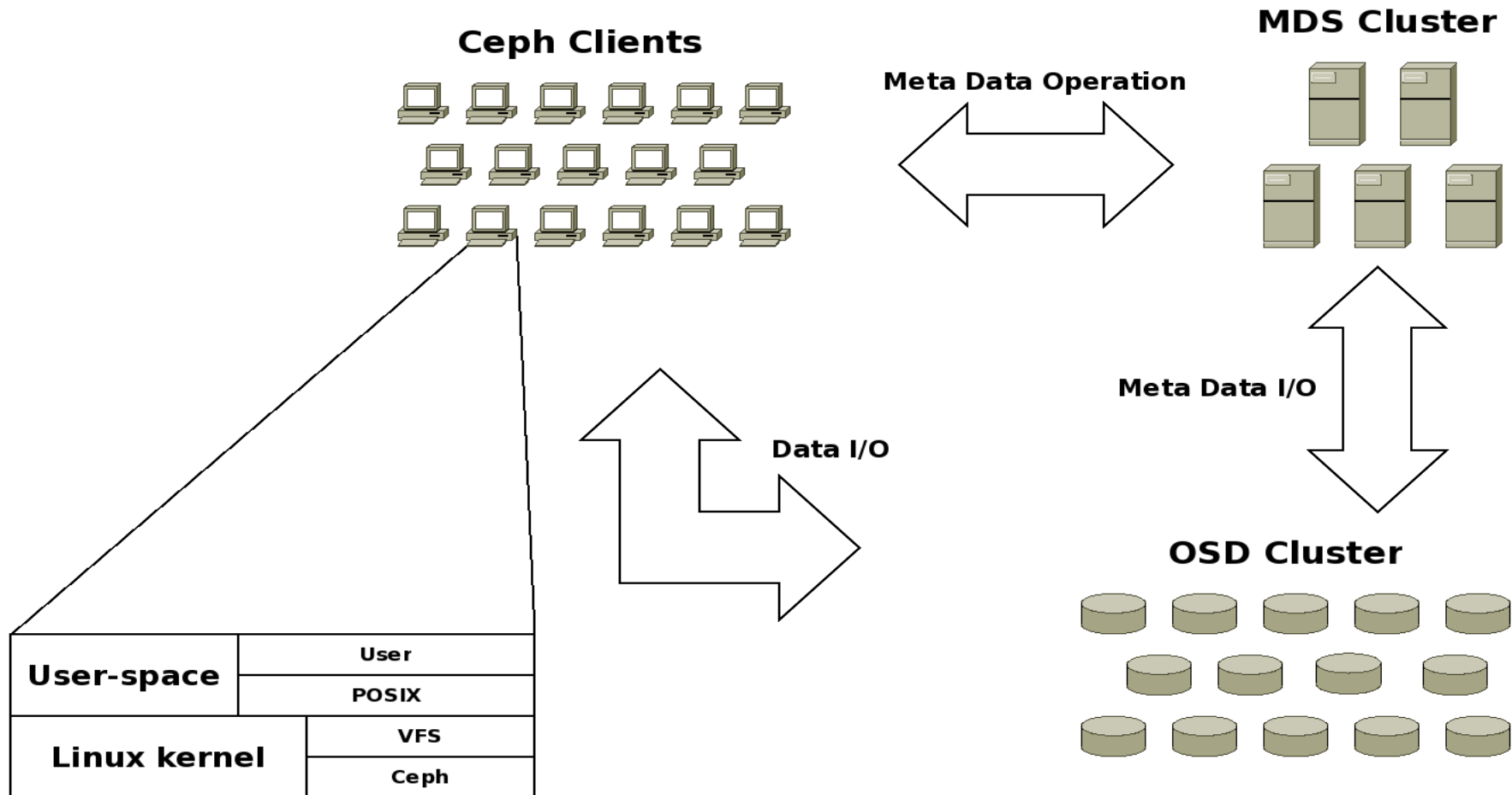
Object Based Storage II

- 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 architecture I

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

Ceph – the architecture II



Ceph client view

- The 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

Ceph and OSD

- User land implementation
- Any computer can act as OSD
- Uses 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

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

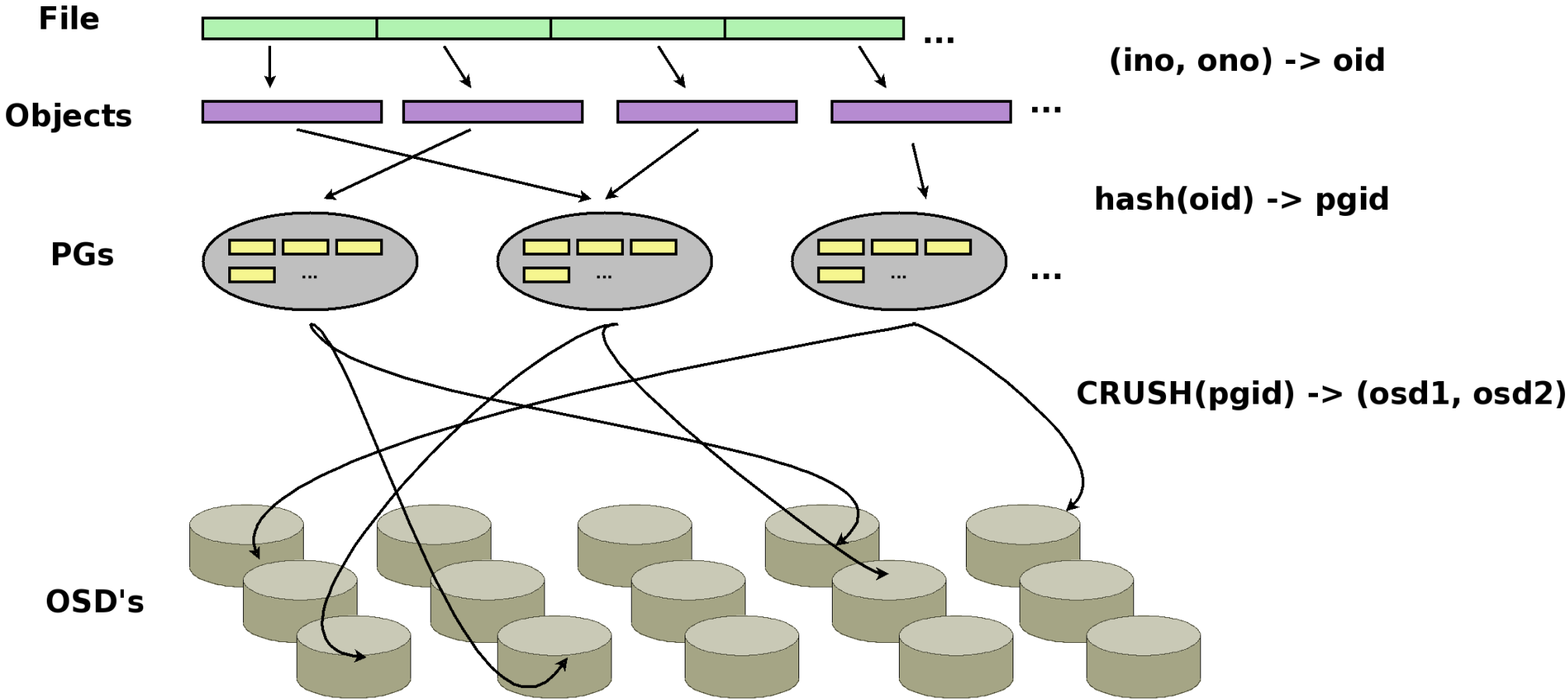
Data distribution

- 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 OSD's out of PG
 - CRUSH algorithm

CRUSH I

- Controlled Replication Under Scalable Hashing
- 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



Data replication

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

Ceph caches

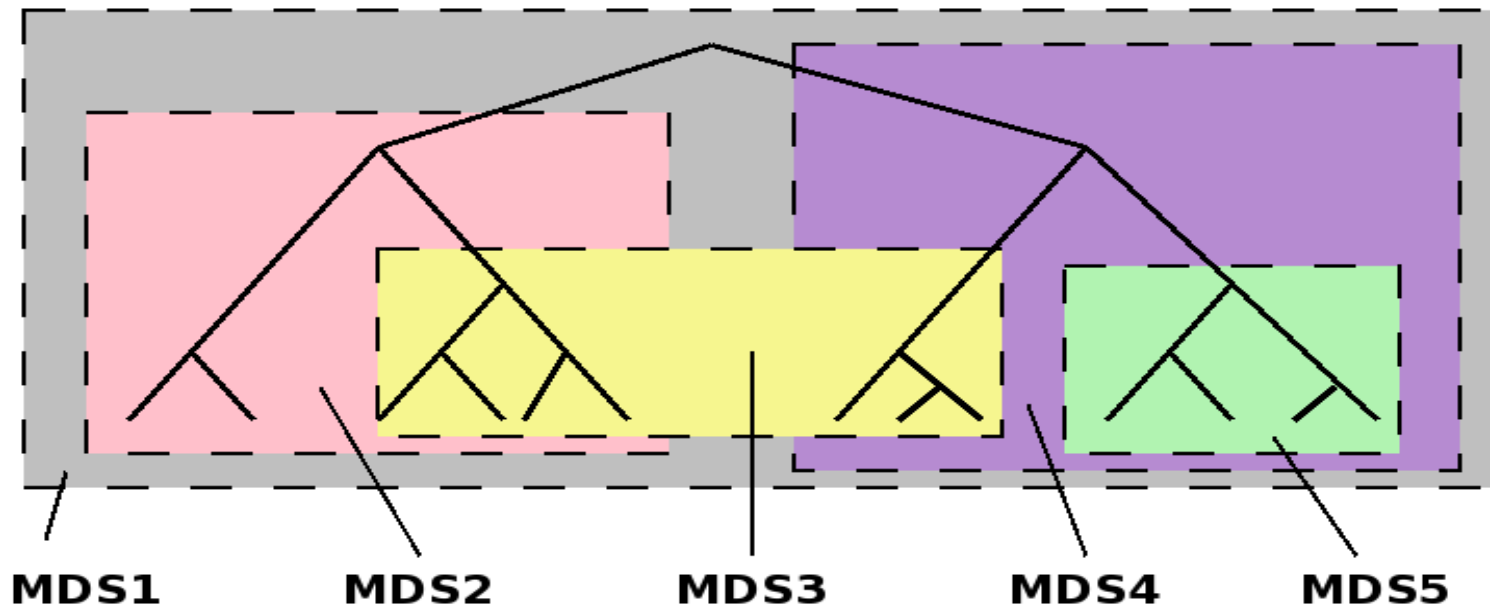
- Per design
 - OSD: Identical to access of BTRFS
 - Client: own caching
- Concurrent write access
 - Caches discarded
 - Caching disabled -> synchronous I/O
- HPC extension of POSIX I/O
 - O_LAZY

Meta Data Server

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

Dynamic subtree partitioning

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



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 cluster monitors

- Status information of Ceph components critical
- Monitor track changes of cluster landscape
 - Update cluster map
 - Propagate information to OSD's

Ceph cluster map 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 cluster map II

- Reflects data rules
 - Number of copies
 - Placement of copies
- Updated version sent to OSD's
 - OSD's 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 synch

Ceph – first steps

- A few servers
 - At least one additional disk/partition
 - Recent Linux installed
 - Ceph installed
 - Trusted ssh connections
- Ceph configuration
 - Each servers is OSD, MDS and Monitor

Summary

- Promising design/approach -> worth looking at
- High grade of parallelism -> no SPOF
- still experimental status -> not recommended for use in production
- Big installations?
 - Number of components
 - Layout

References

- <http://ceph.newdream.net>
- <http://www.ssrc.ucsc.edu/Papers/weil-sc06.pdf>

Thank you!