
whamcloud

LUG 2012
Austin, TX
April 2012

ORIon

- Alexey Zhuravlev & Johann Lombardi
  Lustre Engineers
  Whamcloud, Inc.

Agenda

- Orion Overview
- Quota in Orion

What ORIon project is

- Stands for OSD Restructuring Initiative
- Provide a single storage abstraction
- OSD API introduced in Lustre 2.0
- OSD API only used for MDS metadata operations
- Redundant APIs left for OSS, MDS object ops, llog
- Facilitate different backend storage systems

OSD API and Services

- OSD API becomes rich
- Enough to implement OSS, MDS, MGS
- All components use OSD API:
  - MDD, OFD, MGS, llog, Changelogs
- Old APIs can be removed to simplify code:
  - LVFS, many OBD methods
- Well defined MDS stack
  - OSD Proxy (OSP) uses OSD API to interface to OSTs
  - OSP isolates network RPCs from MDD layer
  - Simplifies error-prone code

OSD API: Benefits

- Easier to exploit new backend storage system
  - ZFS well underway today
  - Btrfs discussed for the future, when stable
- Able to interface with non-filesystem backends?
- Easier code
- Semantically clear object API
  - modules resolve specific problems internally
  - less efforts to become Lustre developer
  - more development from the community

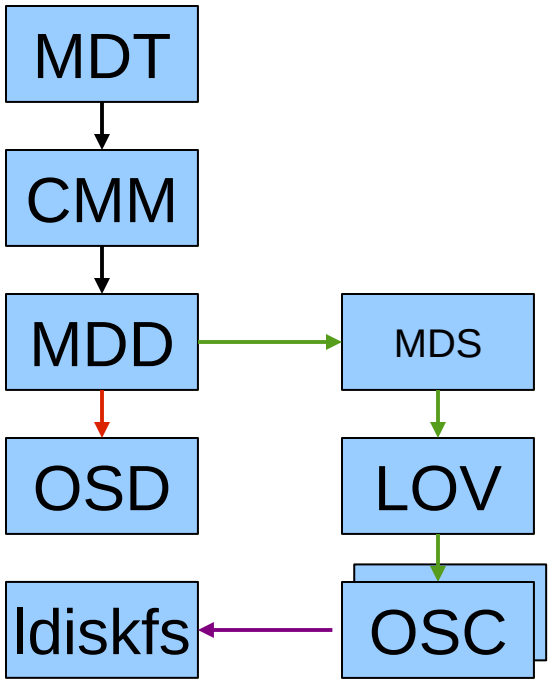
OSD API: changes in details

- 2-stage transactions:
  - Declare, execute
  - Inspired by ZFS
  - Allows to get rid of magical credits in the code
  - Stackable
- Methods to manipulate data:
  - 0-copy IO
  - Punch (truncate)
  - Caching is hidden by specific OSD
- Commit callbacks
  - Per transaction

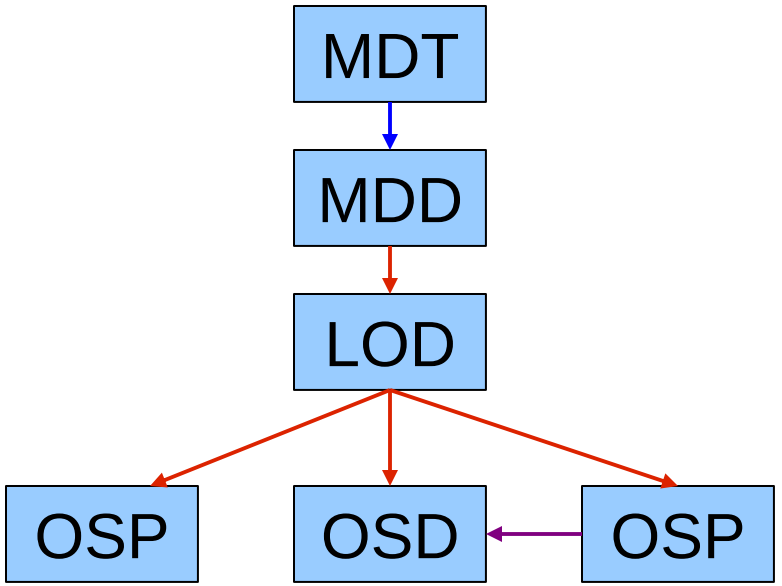
OST objects are destroyed by MDS

- In 1.8/2.0 destroy sent by the clients to OSTs
- Lots of plumbing needed for distributed transactions
  - Vulnerable to double failures
- Can result in file without objects after some failures
  - No data loss (user really deleted file), but annoying
- In ORIon destroy sent by MDS to OSTs
  - Really atomic (commit on MDS first)
  - In batches
- Explores OSD API for distributed operations
  - Next step is DNE Phase 1

2.0/2.1



2.X



MD API + LLOG

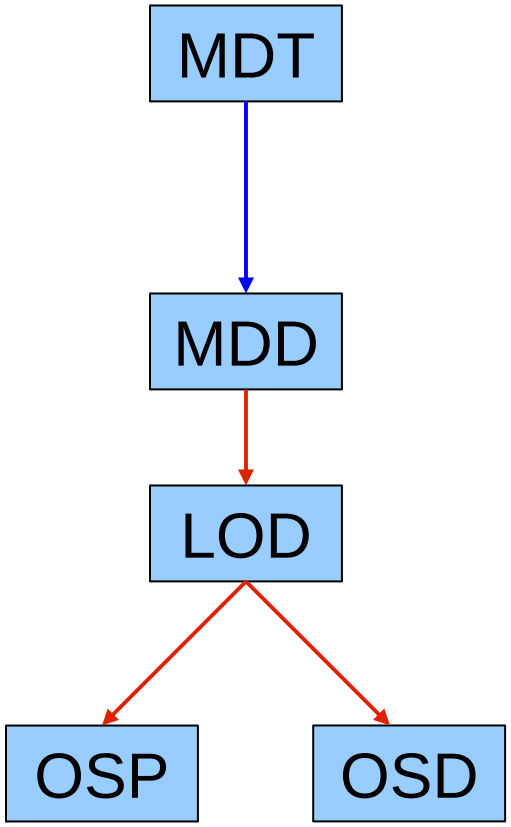
MD API

LLOG

OBD API + LLOG

OSD API

2.x model of MDS stack



Lustre protocol:
Complex operations (RPCs) from file ops:
REINT_OPEN :=
create + open + getattr + getxattr(LOV)

Posix file operations from primitives (updates): Create := object create + insert

Striping, access to striped objects, updates within transaction

Access to single object, updates within transaction
OSD – local objs, OSP – remote objs

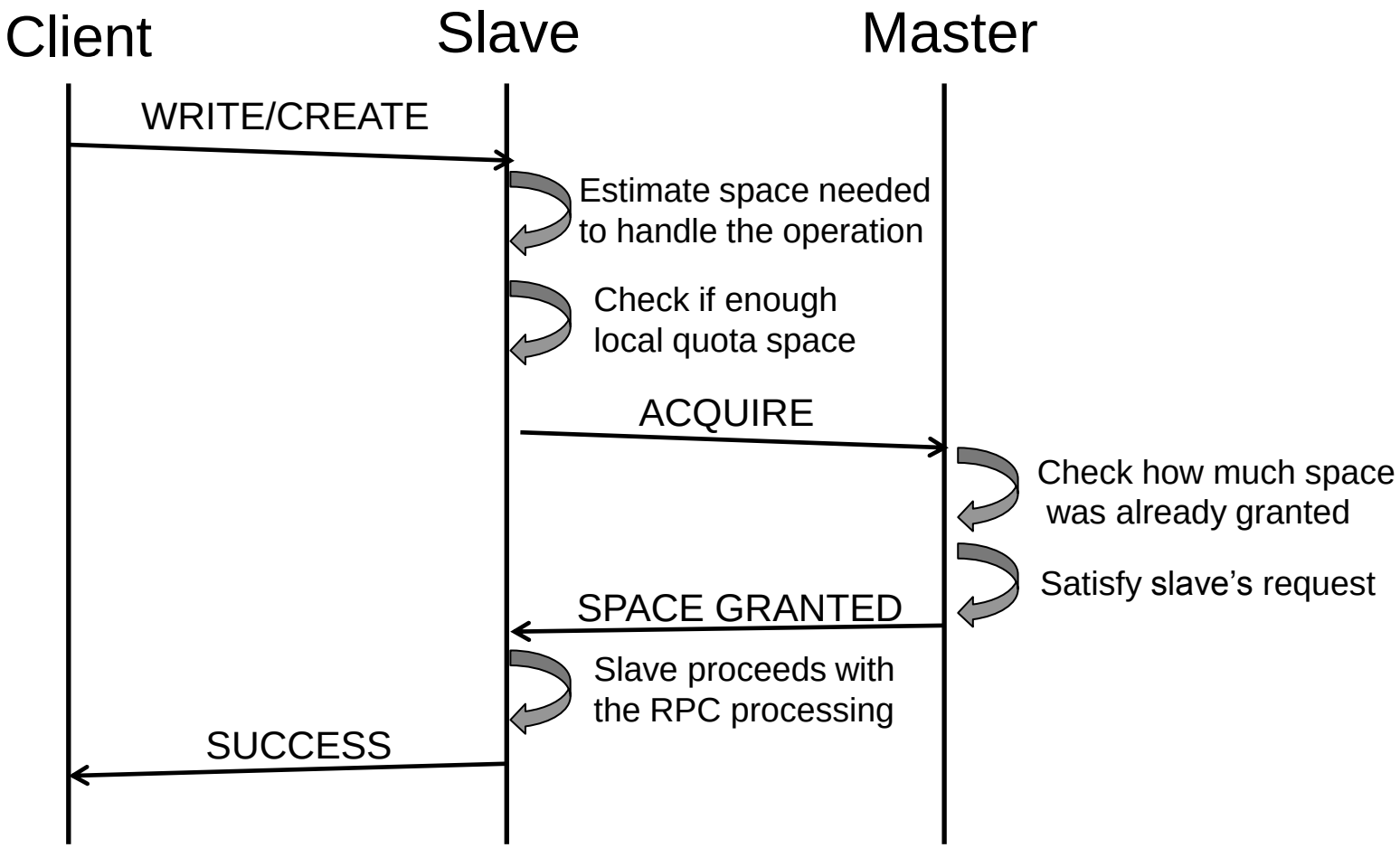
Agenda

- Orion Overview
- Quota in Orion

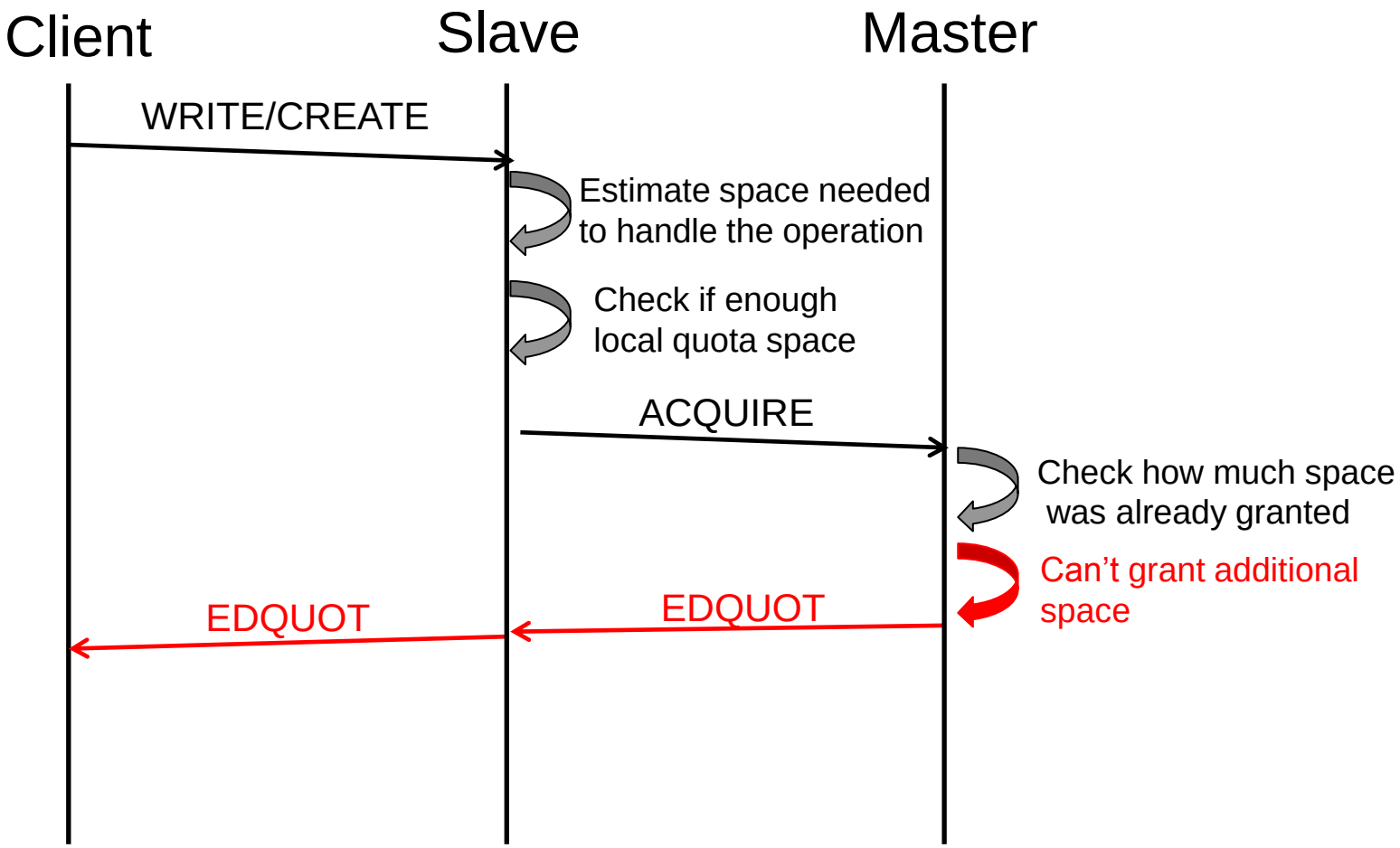
Quota Requirements

- Prevent a single entity from consuming all the filesystem resources
  - An entity can be a user or a group, but could also be a directory
  - Resources are inodes (MDTs) and blocks (OSTs)
- Ability to enforce both block and inode quota
  - Hard & soft limits are supported
- Quota master hold the cluster wide limits
  - Guarantee that global quota limits are not exceeded
  - Grant quota space to slaves
- Quota slaves
  - All OSTs & MDTs
  - Track on-disk usage and acquire/release quota space from master
  - Return EDQUOT when quota space is exhausted

Request Processing with Quota (1/2)



Request Processing with Quota (2/2)



New Quota Design in Orion

- Independent of the backend filesystem
- Quota commands can be run with missing slaves
- Efficient handling of OST addition
- Quota enforcement on/off managed globally at the filesystem level
- Add support for multiple MDTs (aka DNE)
- Allow per-pool quota in the future
- Allow per-directory quota in the future

Architecture Primer

- Slave->Master connection
  - No need to track reverse MDT import any more
  - With a real connection, slaves can now enqueue locks …
- Leverage the proven scalability of our Distributed Lock Manager (aka DLM)
  - Master uses regular lock callbacks (aka AST) to revoke quota space granted to slaves
- Master tracks on-disk quota space distribution
  - Master aware of how much space is granted to each slave
  - Allow dead OST decommissioning and better quota recovery resiliency
- Quota on/off managed on the MGS
  - enabled/disabled globally for the whole filesystem via "lctl conf_param"

Space Accounting in Orion

- ZFS **permanently** tracks per-UID/GID disk usage
  - Even when there is no quota limit enforced
  - Only #blocks and not #inodes (done in lustre itself)
- Same scheme adopted with ldiskfs
  - Quota as a new core ext4 feature
  - mkfs.lustre/mke2fs creates empty quota files
  - Usage tracking always active
  - e2fsck can now fix quota files
- End of quotacheck
- Quota on/off **only** enables/disables **enforcement**

Slave (re)Integration

- Replace existing quota recovery
  - Executed after a master or slave reboot
- Also allow slaves disconnected for a long amount of time to resynchronize quota settings
- 3 steps procedures
  - #1 slave enqueues the global quota lock
  - #2 slave fetches quota settings from the master via a bulk transfer, if needed
    - Settings for 43,520 IDs can be packed in a 1MB bulk
  - #3 slave re-acquires quota space and re-enqueues per-ID quota locks

Comparison with Today's Quota

- More friendly interface
  - global parameter to turn quota on/off
- More robust to slave failures
  - Support disconnected slaves, online slave addition/removal, dead slave decommissioning, …
- Backend agnostic
  - Operate on top of the OSD API
  - Works with both ldiskfs & ZFS
- Better integration with other components
  - Use the LDLM to manage/revoke quota space granted to slaves
  - Use FIDs to access/export quota objects

- Alexey Zhuravlev & Johann Lombardi