

OCFS2

Mark Fasheh

Oracle

What is OCFS2?

- General purpose cluster file system
 - Shared disk model
 - Symmetric architecture
 - Almost POSIX compliant
 - `fcntl(2)` locking
 - Shared writeable mmap
- Cluster stack
 - Small, suitable only for a file system

Design Principles

- Many learned from kernel community
- Avoid useless abstraction layers
 - Use VFS object life times.
 - Mimic the kernel API.
 - Keep necessary abstractions as thin layers.
- Copy good ideas
 - JBD, Ext3 directory code, group allocation
- Keep file system operations local
 - Reduces lock contention

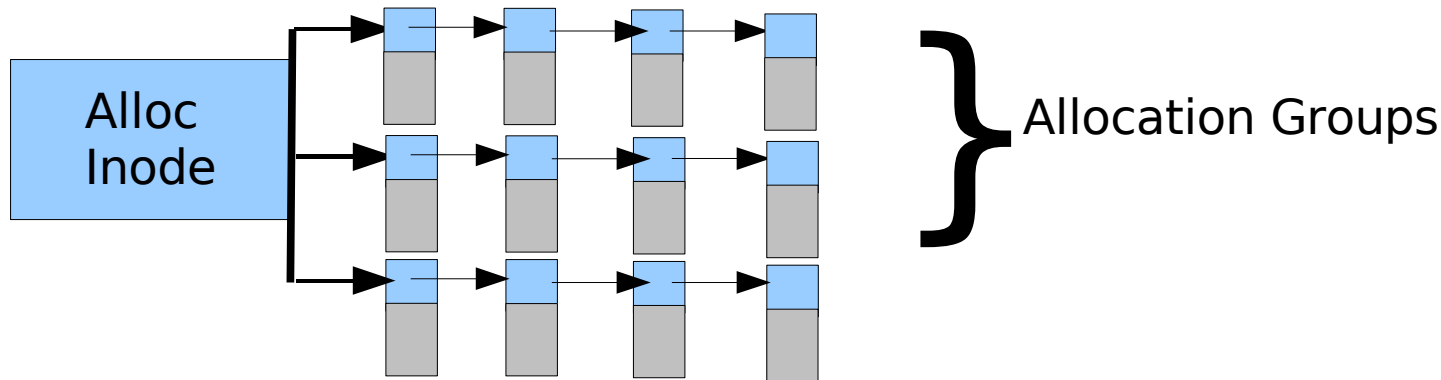
File Allocation

- Allocate meta data in blocks (.5K - 4K)
- Allocate data in clusters (4K – 1M)
- Extent based allocation
 - Requires less accounting for large files
 - Extents arranged in a b+ tree
 - Small #'s of extents stored in a list
- Directories are normal files.
 - Contain an array of records (thanks, Ext3!)
 - Indexing to be implemented soon

System Files

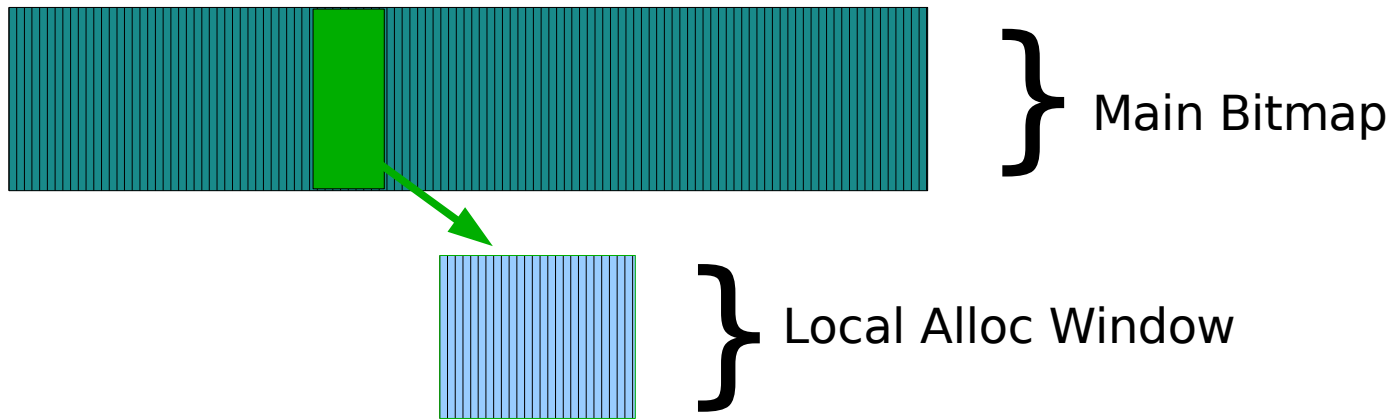
- System files reference all fs meta data
 - Linked into file system via hidden system dir
 - Allows a very dynamic layout
- Global system files
 - Generally accessible by any node at any time
- Node local system files
 - One per mounted node (each node gets a unique “*slot number*”)
 - Reduce contention on resources
 - Some are never locked by another node

Chain Allocators



- Group descriptor contains a bitmap, next group ptr., etc.
- Chains are self optimizing – we can move empty groups to the beginning of their chain as we allocate from them.
- Main bitmap formatted to cover entire volume at mkfs time. Units in clusters.
- Suballocators use block size units but get their groups from the main bitmap as needed.
- Block allocators grow as needed.

Local Alloc



- Node local.
- Reduces contention on main bitmap.
- All bits in window are set on main bitmap, local alloc starts clean.
- As space is used, local alloc bits are set.
- Unused bits are returned to bitmap on shutdown, recovery, or on a window move.

Truncate Log

- Similar performance goals to local alloc
- Used for deallocation
- Single block array of “truncate records”
 - Cluster Offset, Extent Length
- Flushed at specific times
 - When full
 - 2 seconds after most recent truncate
 - On demand via `sync(2)`

Journaling

- Block journaling via JBD
- Journals are node local system files
- JBD was chosen for two main reasons
 - Stability
 - Time - journaling is hard to get right!

Clustered UpToDate

- Small amount of code (550 lines)
- Mimic `buffer_uptodate()` within cluster
- Stores info using a per-inode list or rbtree
- Simple rules:
 - Meta data changes are journaled under cluster lock
 - Meta data reads have at least RO cluster lock
 - Getting a new cluster lock means you destroy the cache
 - Don't need to pin `buffer_heads` – just record block number

A Short DLM Overview

- Pared down VMS style API
 - Operations are asynchronous
 - Supports NL, PR and EX lock levels
 - Supports lock value blocks (LVB)
 - Use callbacks to indicate lock state changes
- A node can
 - Create a new lock
 - Up convert a lock to a higher level
 - Down convert a lock to a lower level
 - Drop the lock completely

DLM Glue

- FS Internal abstraction for cluster locking
- Implements *lock caching*
- Up converts and down converts locks
 - If down converting to PR, will not block read only locks
 - Likewise, will not block compatible requests during up convert
- Lifetimes left up to container objects
- Only concerned with cluster locking
- Lock specific actions via a set of callbacks

Inode Locks

- `ip_rw_lockres`
 - serializes `read(2)`, `write(2)`
- `ip_meta_lockres`
 - Protects inode meta data
 - Refreshes `struct inode` when necessary
 - Use LVB for most common inode fields
 - Might checkpoint journal before dropping EX
- `ip_data_lockres`
 - Protects inode data
 - Might sync pages before dropping EX

Messaging (“votes”)

- Very few fs messages left
 - Last few will be replaced by cluster locks
- Handle hard cases by broadcast message
 - Rename / Unlink – remove a dentry
 - Query an inode delete
 - Mount/Unmount notification
- Messages related to a given inode are serialized by the meta data lock

Node Liveness

- Disk Heartbeat
 - Final arbiter of “liveness”
- Network Keepalive
 - Majority based quorum
- Self-fencing mechanism
 - Simple, but harsh
 - Makes timing between disk and network complicated
 - Real hardware fencing in the future

Recovery

- A FS is notified, blocks meta data locking
- B DLM is notified, completes lock recovery
- C FS begins journal recovery
 - A One node wins race to lock journal
 - B Journal Recovery
 - c Clear local alloc and truncate log
 - D Unlock Journal, re-enable meta data locking
- D Recover local alloc, truncate log, and orphan dir

Questions

- Cluster File System BOF
- Today, 6PM