Active/Active NFS Server Recovery over CephFS

Jeff Layton <jlayton@redhat.com>

RADOS, CephFS and nfs-ganesha

- RADOS is distributed object store
  - Clusters of small, simple object storage daemons
  - Clients can batch up sets of read and write ops to be done atomically
  - Each object has unstructured data, xattrs, and an omap (kv store)
- CephFS is a cluster-coherent POSIX filesystem
  - Built on top of RADOS
  - Clients talk to Ceph MDSs and acquire **capabilities** (aka caps)
    - Delegated portions of inode metadata (Auth, File, Xattr, etc.)
  - Coherent POSIX locking and support for delegations
  - Kernel client, FUSE daemon based on libcephfs
- nfs-ganesha is a userland NFS server
  - Plugin support for filesystem and client recovery databases
  - Well suited to exporting userland filesystems
  - FSAL_CEPH backend uses libcephfs
  - Can also store recovery databases and configs in RADOS

State of the Ganesha + CephFS Union

- Active/Passive nfs-ganesha over CephFS support in RHCS 3.0
  - Traditional cluster using pacemaker

- Active/Active configurations mostly just work
  - Conflicting state can be handed out during recovery
    - opens, locks, delegations, layouts

- Ganesha has some clustering support (mostly for GPFS)
  - Driven by dbus commands (requires separate process to manage it)
  - Relies on splitting/merging recovery databases during externally imposed grace period (not atomic enough)
  - Server reboots during grace period could cause client failures
  - May be race windows where conflicting state could be acquired

Scale-out NFS Server over CephFS

- Stand up a cluster of nfs-ganesha servers over CephFS
  - Active/Active configuration
    - Eliminate bottlenecks and SPOFs
  - Focus on v4.1+, with eye toward eventual pNFS support
- Minimize coordination between ganesha nodes:
  - Have them run as independently as possible
    - Better scaling as node count increases
  - Self-aggregate as much as possible
    - Minimize need for external driving forces
- Ganesha is amenable to containerization
  - Store configuration and recovery info in RADOS
  - No real need for any writable local storage or elevated privileges
  - No 3rd party clustering (use RADOS to coordinate)
  - No takeover of one node’s resources by another
    - Rebuild container from scratch if something fails
    - May add this in future to handle migration

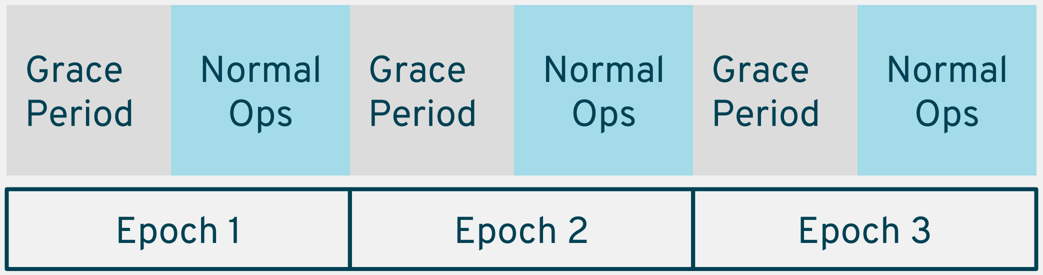
NFS Client Recovery DB

- After restart, NFS servers come up with no ephemeral state
    - Opens, locks, delegations and layouts are lost
    - Allow clients to reclaim state they held earlier for ~2 lease periods
    - Detailed state tracking on stable storage is quite expensive

- During the grace period:
    - No new state can be established, only reclaim
    - Allow reclaim only from clients present at time of crash
        - Necessary to handle certain cases involving network partitions

- Must keep stable-storage records of which clients are allowed to reclaim after a reboot
    - Prior to a client doing its first OPEN, set a stable storage record for that client if there isn't one already
    - Remove after last file is closed, or when client state expires
    - Atomically replace old client db with new just prior to ending grace

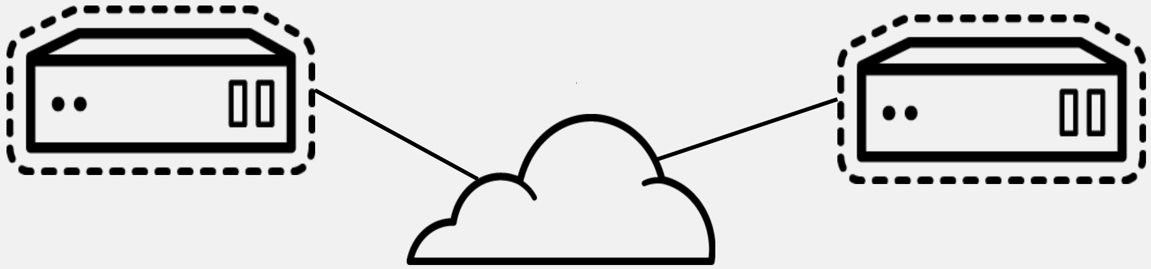
Reboot Epoch



- Consider each reboot the start of a new epoch
  - As clients perform first open (reclaim or regular), set a record for them in a database associated with current epoch
  - During grace period, allow clients that are present in the previous epoch's database to reclaim state
  - After transition from Grace to Normal operations, can delete any previous epoch's database

Parallel Server Grace Period

- Each server node must enforce grace period until it is no longer needed by any node:
  - Allow each node to join in-progress grace period, and declare when they no longer need it.
  - If node crashes while cluster wide grace period is in effect, want to allow it to rejoin if possible
- Lifting grace really two-part process:
  - Indicate that node's local grace period is over (no longer needed here)
  - Indicate when we've stopped enforcing it (no one needs it any longer)
- Epoch is now a cluster-wide property



NEED and ENFORCING flags

Each server in the cluster can represent its current state via two flags:

- **NEED (N):** does this server currently require a grace period?
  - First node to set its NEED flag declares a new grace period
  - Last node to clear its NEED flag can lift the grace period
- **ENFORCING (E):** is this node enforcing the grace period?
  - If all servers are enforcing then we know no conflicting state can be handed out (grace period is fully in force)
  - Used by a server to determine when it's safe to clear state held by previous incarnation of itself

Simple logic to allow individual hosts to make decisions about grace enforcement based on state of all servers in cluster.

RADOS Grace Period DB Design

- Single, shared grace period database object (object ID **“grace”**)
  - Used to coordinate cluster-wide grace period and recovery
  - RADOS op compounds are atomic
  - Do read op, modify, write op, and have writes assert on obj version
  - Loop and try again if assertion fails
- 2 uint64_t values:
  - **Current epoch (C)**: where should new records be stored?
  - **Recovery epoch (R):** from what epoch are we allowed to recover?
  - **R == 0** means grace period is not in force (normal operation)
  - Stored little-endian in unstructured data in the shared grace object
- An omap (kv store) to track state of each cluster node:
  - Key is node’s hostname, presence indicates cluster membership
  - Value is a byte with lower two bits used as flags (6 for later use)
    - **Need Grace (N) == 0x1**
    - **Enforcing Grace (E) == 0x2**

Parallel Server Recovery DBs

- Nodes can die at any time, so we still must have an atomic switch from old database to new
- Separate recovery DB for each host/epoch tuple
- Same format as traditional rados_kv recovery DB
- When we go from normal operation to grace without allowing recovery, must recreate DB with new epoch
- Delete old ones when recovery from them is no longer possible
  - when R==0 and DB is not for current value of C

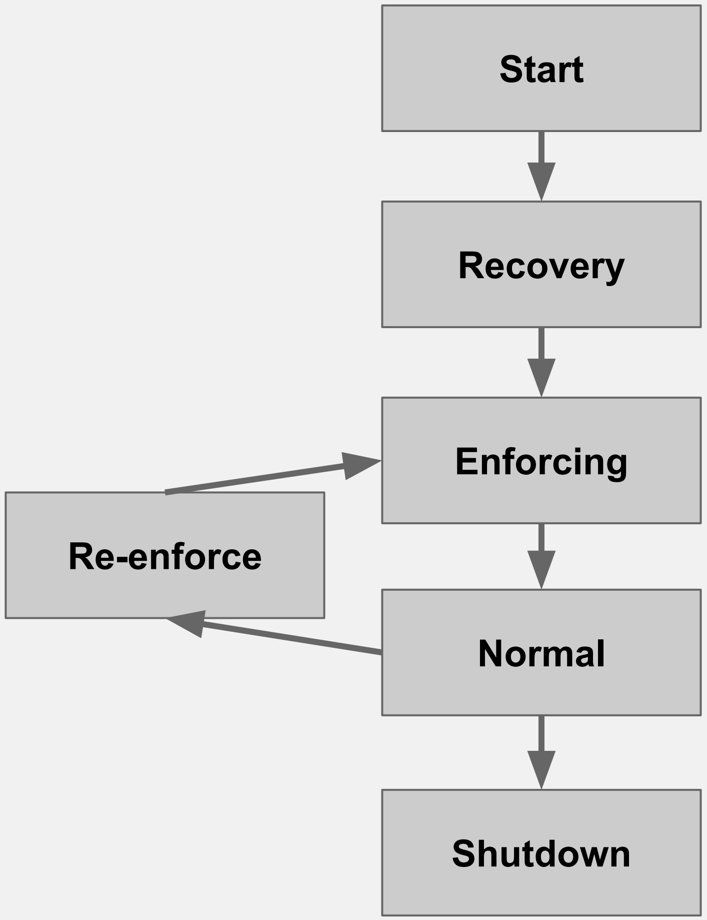
```
rec-CCCCCCCCCCCCCCCC:NNNNNNN...

C = Current Epoch (hex uint64_t)
N = NodeID (opaque string)
```

ganesha-rados-grace

Command-line tool to manipulate grace database

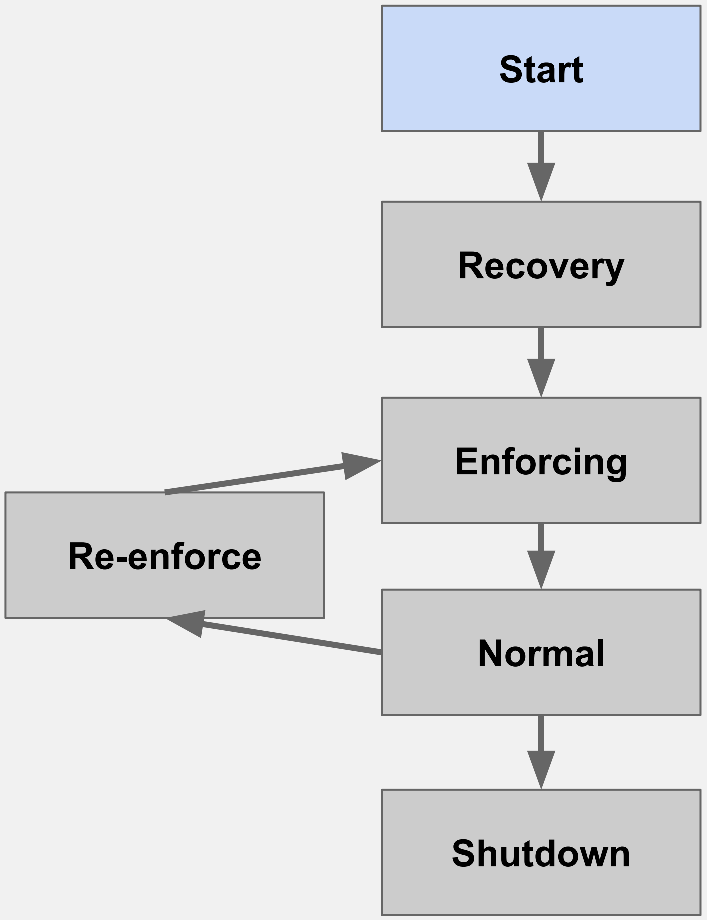
- Show current state of the grace DB
  - Current and recovery epoch values
  - Servers that are members + their flags
- Add/remove servers from cluster
- Start a new grace period or join an existing one
- Lift the grace period for one or more nodes
- Set/clear enforcing flags

Clustered NFS Server Lifecycle



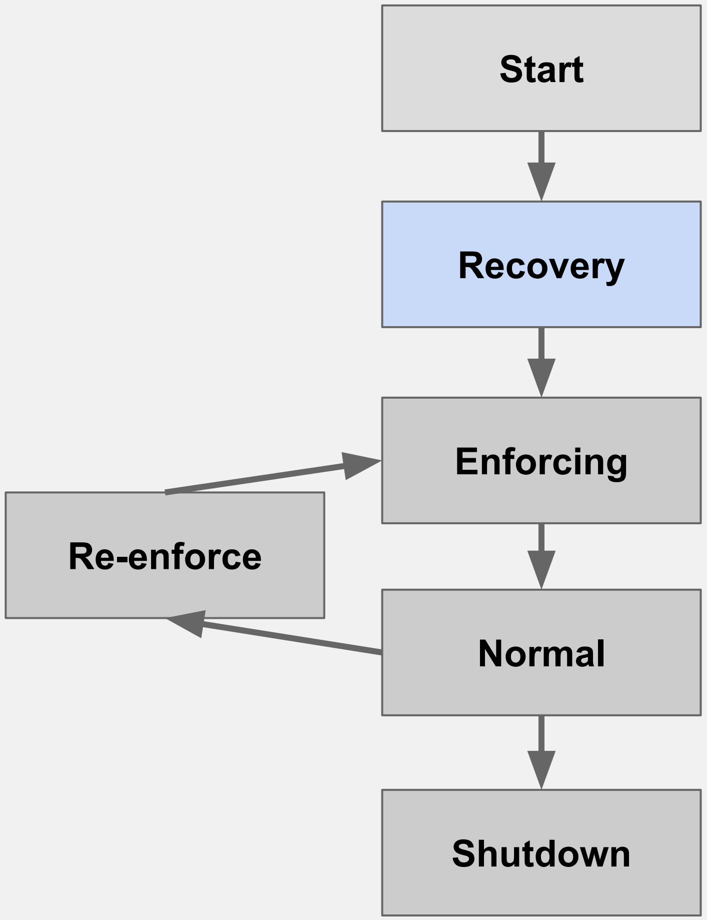
- **Start:** server comes up, waits for other nodes to start enforcement
- **Recovery:** allow clients to recover
- **Enforcing:** no longer allowing recovery, but are still enforcing the grace period
- **Normal:** normal operation. Clients can acquire new state
- **Re-enforce:** prepare to start enforcing again
- **Shutdown:** prepare cluster for eventual restart

Start State



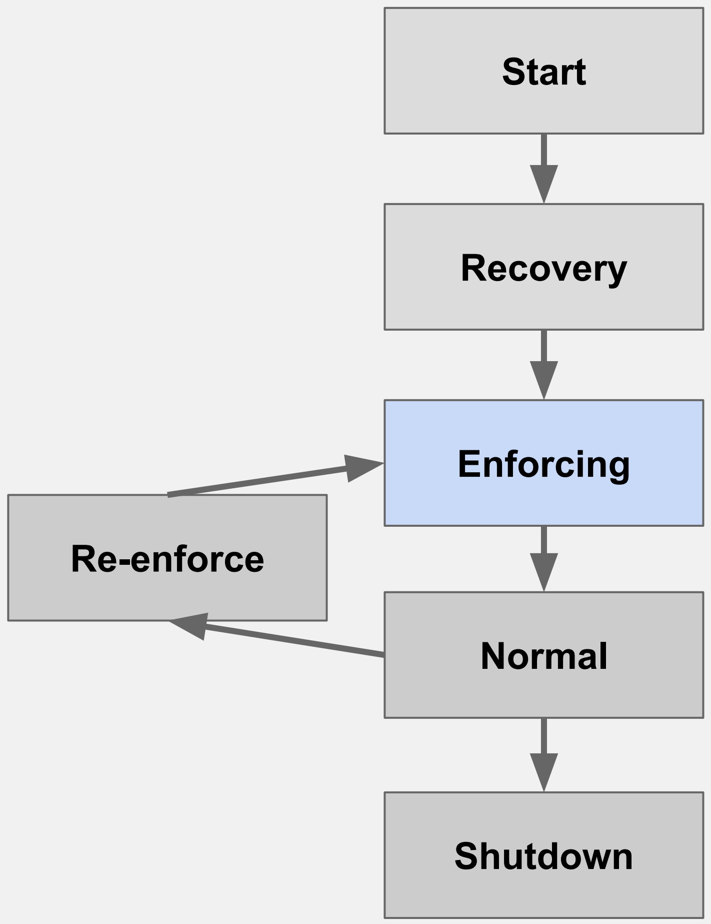
- Startup from zero for the server
- Request new grace period if one is not already active
- Ensure that **N** and **E** flags are set
- Blocking wait for all other cluster servers to set their **E** flags
- Once all hosts are enforcing, continue startup:
  - Load exports
  - Kill off any state held on Ceph MDS by previous incarnation of the server

Recovery State



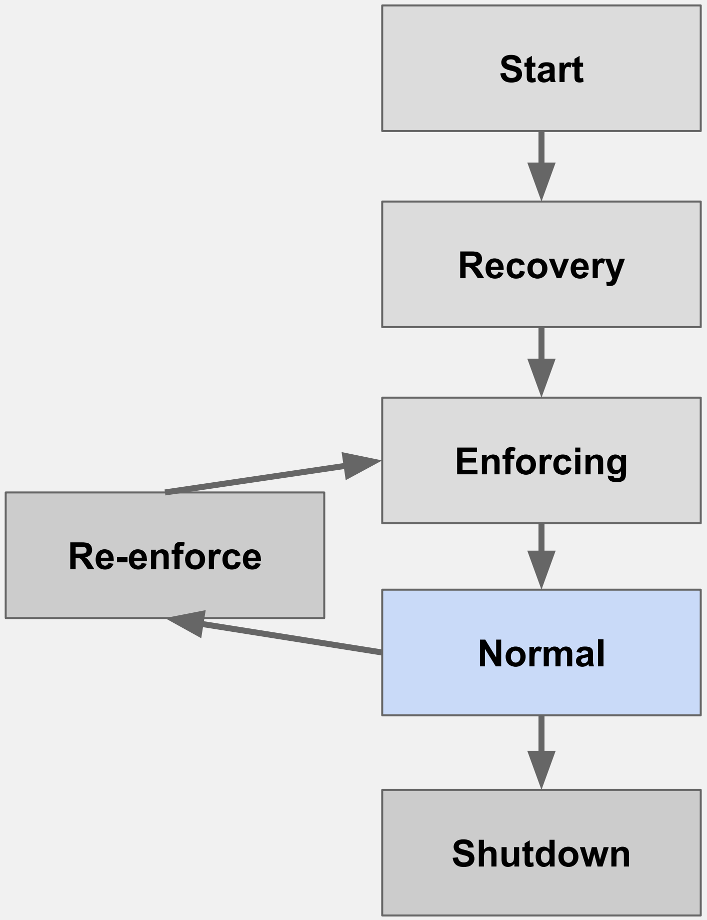
- Grace period timer starts (usually 90-120s)
- Clients are allowed to reconnect and reclaim previous state iff they are listed in recovery DB from previous epoch
    - Just like normal singleton server
- Lasts until:
    - grace period timeout
    - ...or all known clients have sent RECLAIM_COMPLETE

Enforcing State



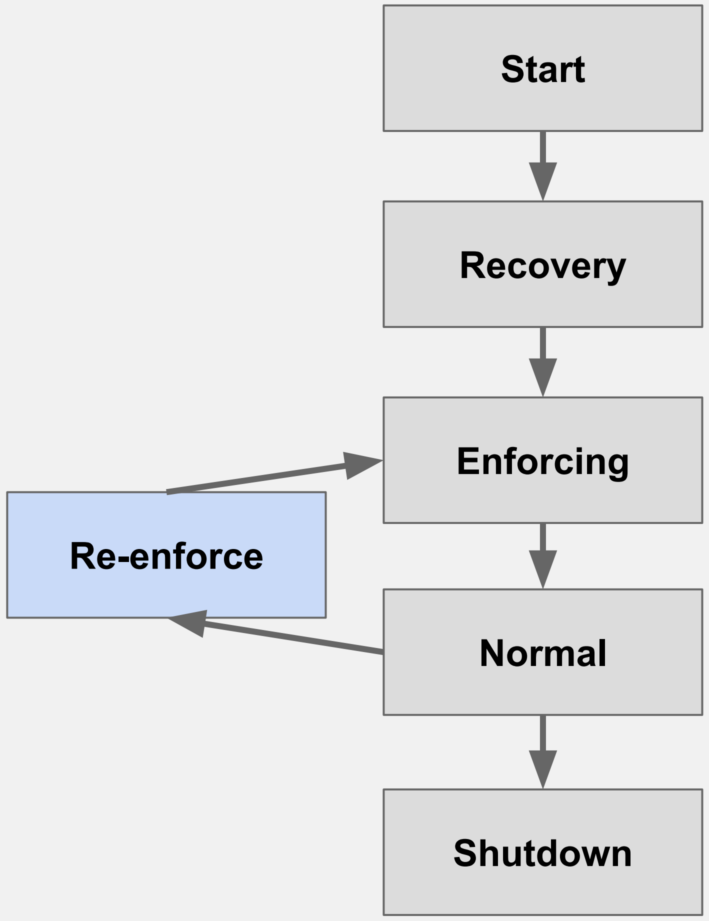
- Clear N flag, leave E set
  - If ours is last N flag, then we can fully lift the grace period (by setting R=0)
- Clean out recovery DB in memory
- “Limbo” state:
  - No reclaim allowed (reclaim DB is blank)
  - No new state can be acquired
  - Other operations can proceed
- Transition to next state when R == 0
  - At that point, no one needs grace period any longer
  - Clear **E** flag

Normal State



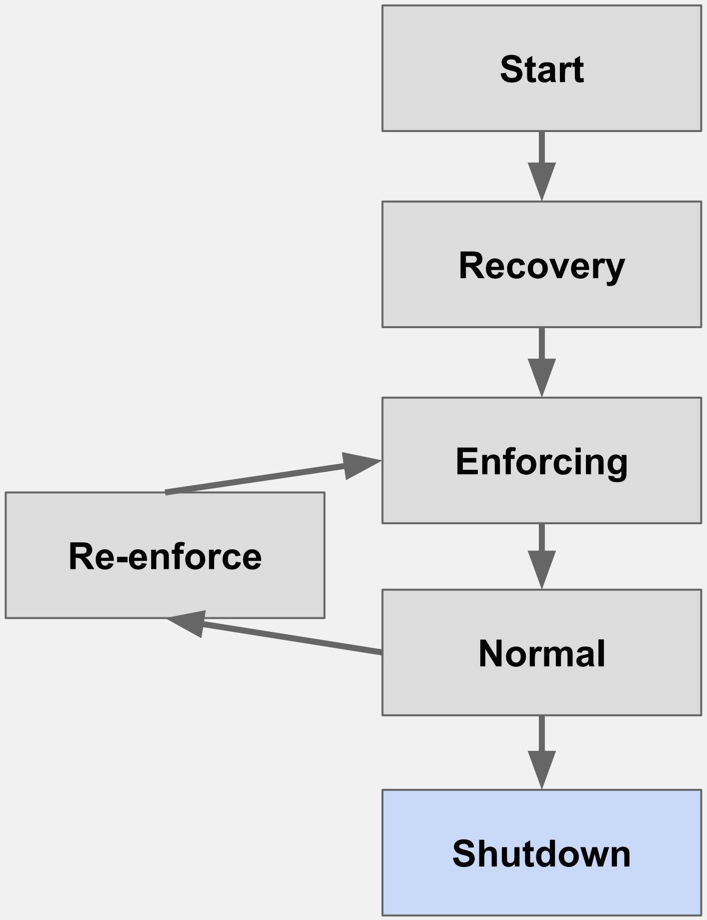
- Normal server operations
  - No recovery allowed (R==0)
  - New state can be acquired
- Represents the bulk of server time
- Ends when server shuts down
  - goto Shutdown state
- Or another node starts a new grace period
  - goto Re-enforce

Re-enforce state



- New grace period has started
  - State held by existing clients is intact
- "Drain off" any in progress operations that would be disallowed during grace
- Create a new recovery DB for the new epoch, and write records for active clients into it
- Set **E** flag in grace DB
  - No new state can be acquired
- Goto enforcing state

Shutdown state



- Can technically enter this from any previous state
- Stop processing RPCs
- Start a new grace period or join an existing one, set N+E flags
    - Only do this if we're still a member of the cluster. If this server's omap key is not there, then we don't expect to return and won't request a grace period.
- Complete shutdown
    - Ensure that state is left intact in Ceph MDS

“Sticky” grace periods

- Usually we check grace period status at the beginning of an operation and assume it will stay that way
- Convert these calls to take a reference to the current grace period status
- Only change once refcount goes to zero
- Once a request to change is submitted, stop handing out new refs until the change is done
- Ensure no conflicting state can occur once the enforcing flag is set
- Patches for ganesha currently WIP

Caveats and Future Directions

- Relies on MDS holding state for a client that has gone down
  - All nodes must be in grace before MDS times out old session
  - Declare new grace period ASAP on clean shutdown
  - Draft ceph patches to kill off old session and increase timeouts

- Grace enforcement should be refcounted (sticky grace periods)
  - Take a reference when starting a state-morphing operation
  - Stop processing new requests
  - Ensure no state morphing ops in progress before we report enforcing

- Container orchestration integration
  - nfs-ganesha-crd as part of rook project

- Redo grace DB handling with OSD class methods (nice to have)

Questions?

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos