SDC
STORAGE DEVELOPER CONFERENCE
SNIA SANTA CLARA, 2014

NFS-Ganesha for Clustered NAS

Poornima Gupte
IBM

Venkateswararao Jujjuri
IBM

Outline

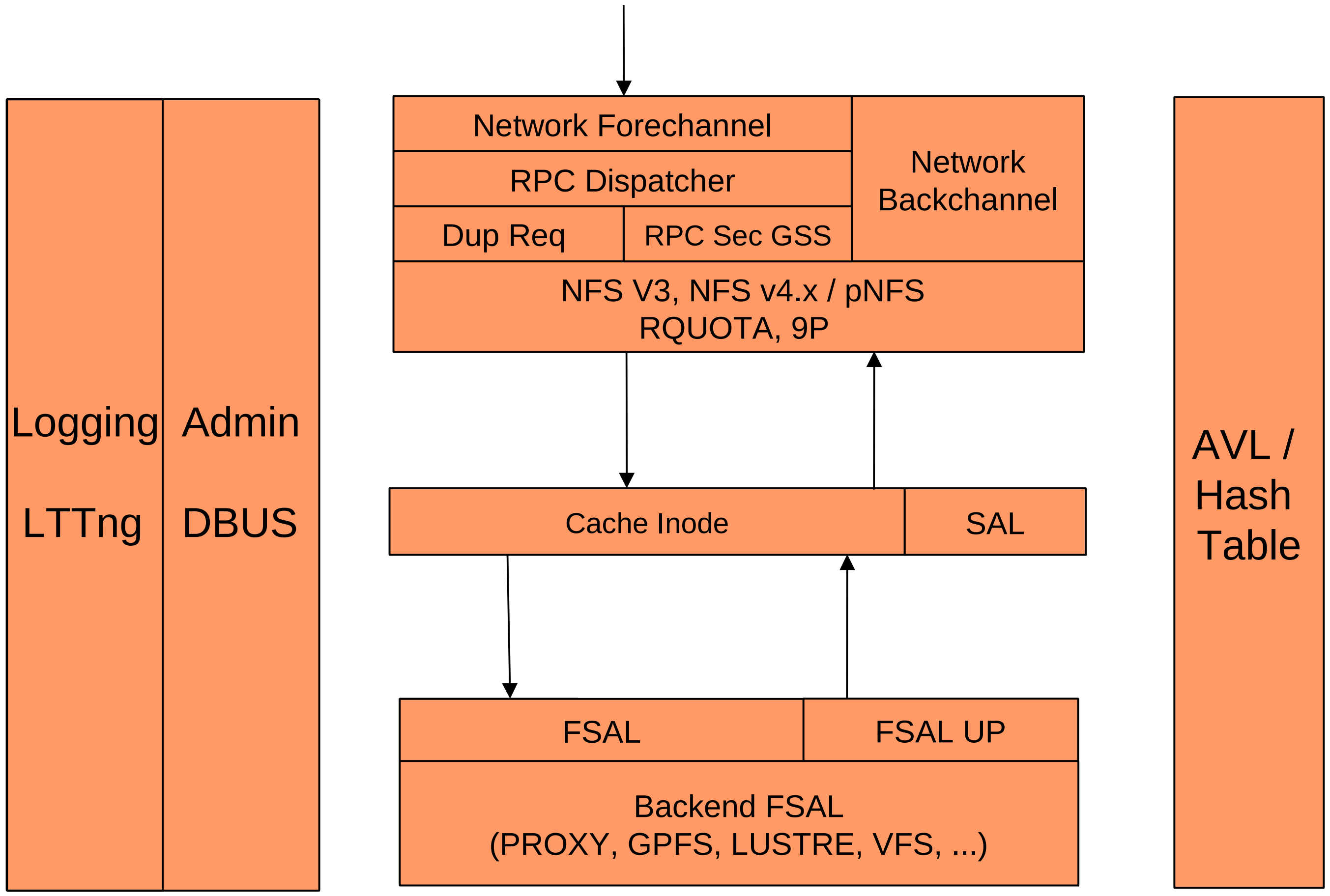
- What is NFS-Ganesha ?
- Enterprise NAS Requirements
- NFS-Ganesha Architecture
- Clustering with NFS-Ganesha
- What is CMAL ?
- Use Cases of CMAL
  - Clustered Duplicate Reply Cache (cDRC)
  - Recovery and Failover

What is NFS-Ganesha?

- User Level implementation of NFS Server
  - Supports v3, v4, v4.1, pNFS, v4.2
- Can serve multiple types of file systems at the same time
- Can manage huge meta-data cache
- Provision to exploit file system specific features
- Can act as proxy server to export a remote NFS server.
- Cluster Manager agnostic

- Small but growing community.
- Active participants – IBM, Panasas, Redhat, CohortFS (LinuxBox), CEA

Modules in NFS-Ganesha



Modular Architecture

- RPC Layer: Uses libntirpc
- FSAL: File System Abstraction Layer, provides an API to generically address the exported namespace.
- Cache Inode: manages the metadata cache for FSAL. It is designed to scale to millions of entries
- Log Management: One API for internal logging (in files) and remote logging (via syslog API). Support for LLTng tracing.
- AVL Trees/ Hash Tables:This layer is widely used to build various cache, including the meta-data cache.
- FSAL UP: provides the daemon a way to be notified by the FSAL that changes have been made to the underlying FS outside of Ganesha. This information is used to invalidate or update the Cache inode.

Enterprise NAS Requirements

- Reliable, Redundant and Fail-Safe through clustered configuration.
- Serve structured and unstructured data over multiple protocols
- Scalable in capacity and performance
  - Flexible to Scale-Up and/or Scale-out
  - Capable of supporting large number of clients.
- Enterprise features – tiering, De-dupe, multi-tenancy, multi-protocol etc.
- Flexible to run on various platforms and serve heterogenous sets of data
- Supports complex security configurations.
- QoS – Quality of Service

File System Abstraction Layer (FSAL)

- The FSAL provides a namespace independent API, which shields Ganesha from native namespace's API
- FSAL API is the interface between core server and backend namespace support of the FSAL
- FSAL is a dynamically loaded shared object library
- Multiple FSAL libraries can be loaded in a single Ganesha instance

FSAL

- Uses file handle interface
- Uses VFS interface or directly calls the backend filesystem's interface
- Combine multiple calls into one
- Integrated Ganesha traces with FS traces
- GPFS FSAL supports all these features

FSAL UP

- FS to FSAL interface (upcall)
- A thread per file system
  - Call into backend FS and wait for events
- All linux inode upates also notify Ganesha inode cache
- All async activity uses up-call
  - Grant / Retry locks
  - Recall delegations and pNFS layouts

Single Node NFS Server

- NFSv3
  - In general, protocol is “stateless” for filesystem operations...
  - except locking
  - Locking needs lock manager (NLM) and status manager (NSM)
  - “Grace period” on server restart (reboot) to allow clients to reclaim locks

NFSv4

- Protocol is now “stateful”, no separate ancillary protocols
- Client monitors server health with periodic “lease” renewals
- Still uses grace period for server restarts (reboots)
- Client discovers restart with stateid, and reclaims open and lock state during grace

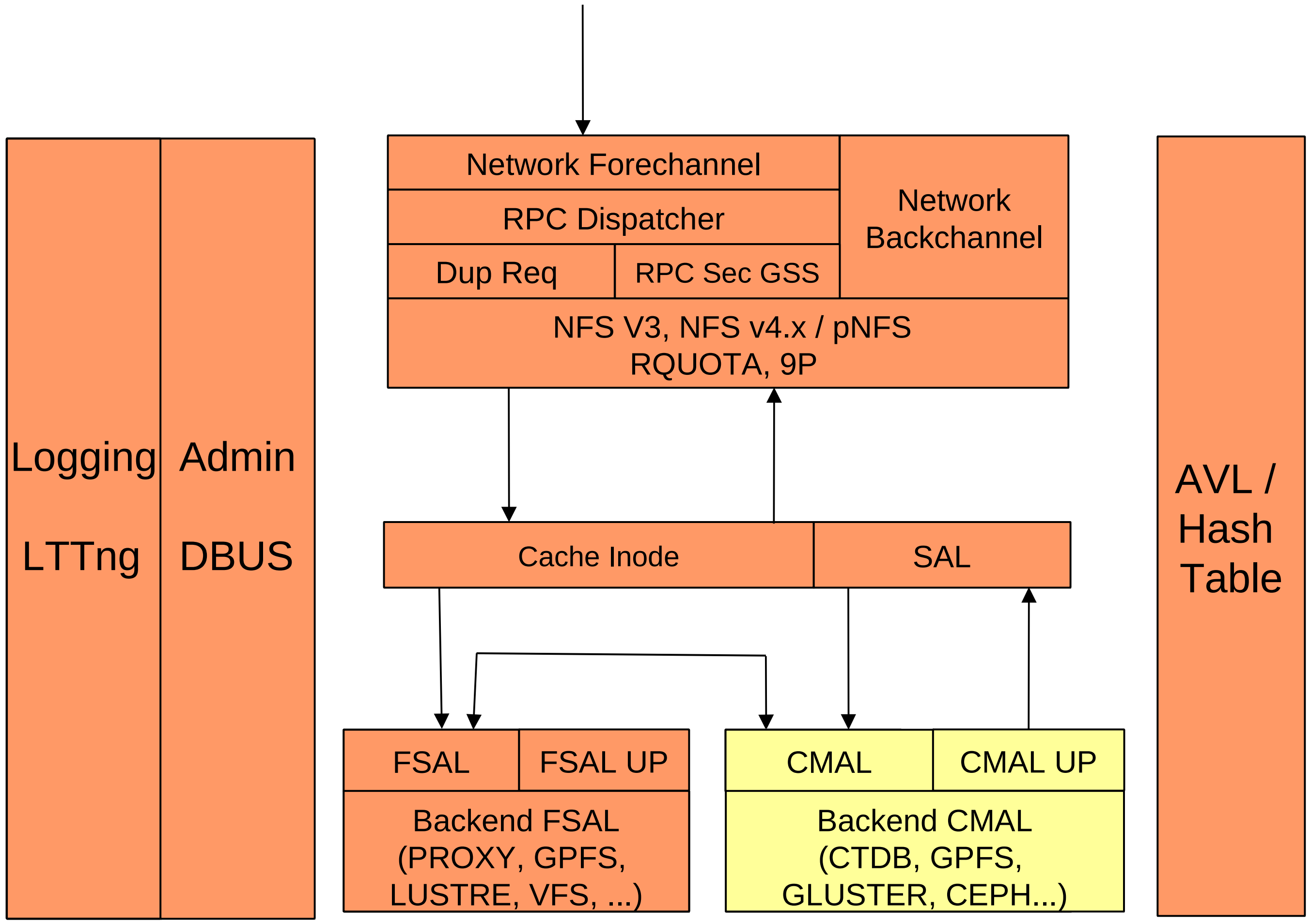
Clustered NFS Server

- Cluster wide change notifications for cache invalidations
- Co-ordinate Grace period across nodes in the cluster
- Maintain and recover lock, share reservation and delegations state
- Provide “high availability” to stateful parts of NFS
  - Share state across the cluster to allow failover
  - IP Failover in case of node failure
  - Lock recovery in case of node failure

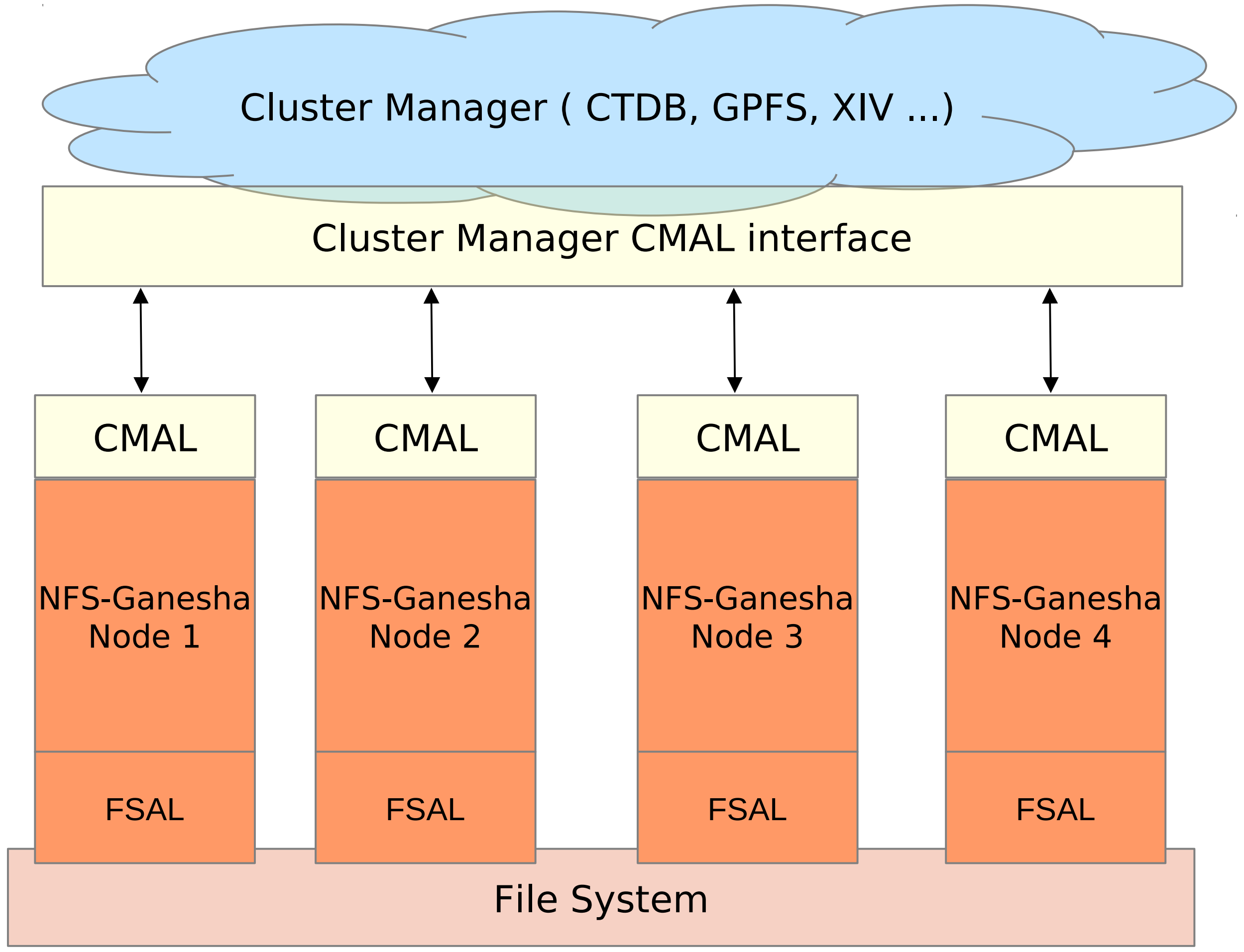
What is CMAL ?

- Proposed **C**luster **M**anagement **A**bstraction **L**ayer (CMAL)
- Provides Abstraction for Cluster Manager Support
- Cluster Manager agnostic
- Modeled after FSAL (File System Abstraction Layer)
- Manages inter-node communication among cluster nodes
- Generic enough to implement many clustering interactions for cDRC, Recovery, DLM etc features
- Can provide cluster communication between MDS and DS in pNFS configuration

CMAL


Network Forechannel
RPC Dispatcher
Dup Req
RPC Sec GSS
Network Backchannel
NFS V3, NFS v4.x / pNFS
RQUOTA, 9P
Logging
LTTng
Admin
DBUS
Cache Inode
SAL
FSAL
FSAL UP
Backend FSAL (PROXY, GPFS, LUSTRE, VFS, ...)
CMAL
CMAL UP
Backend CMAL (CTDB, GPFS, GLUSTER, CEPH...)
AVL / Hash Table

CMAL


Cluster Manager (CTDB, GPFS, XIV ...)
Cluster Manager CMAL interface
CMAL
CMAL
CMAL
CMAL
NFS-Ganesha Node 1
NFS-Ganesha Node 2
NFS-Ganesha Node 3
NFS-Ganesha Node 4
FSAL
FSAL
FSAL
FSAL
File System

Use Cases for CMAL

- Clustered Duplicate Reply Cache
- High Availability – Recovery and Failover
- Distributed Lock Manager
- Distributed Consistency

Duplicate Reply Cache (DRC)

- ❒ Idempotent Operations: can be performed more than once with equivalent results
- ❒ Non-idempotent Operations: Cannot be reprocessed; may fail if tried a second time.
- ❒ What is Duplicate Reply Cache ?

  The NFS Reply Cache or Duplicate Reply Cache (DRC) provides a way for the server to identify duplicate requests and avoid repeated processing of non-idempotent operations.

  Mostly, DRC is maintained in the node's RAM

  For example:

  A file can be removed only once.

Clustered Duplicate Reply Cache (cDRC)

- In a clustered environment there are issues because duplicate reply cache is not cluster aware.
- In the case of failure, if the NFS request is replayed on another server, than that server has no knowledge of the reply cache and may result in incorrect behavior

Should we live with this ?

At least try alleviate (not completely eliminate) the problem

CMAL and cDRC

- ❐ Use the CMAL and Cluster Manager support.
- ❐ Store the DRC in the cluster
- ❐ CMAL and CMAL UP interfaces provide the generic APIs that can be used by the cluster manager to store / retrieve the DRC entries.
- ❐ CMAL is cluster manager agnostic; so any cluster manager can be plugged in the backend to provide the APIs

CMAL and cDRC - Design

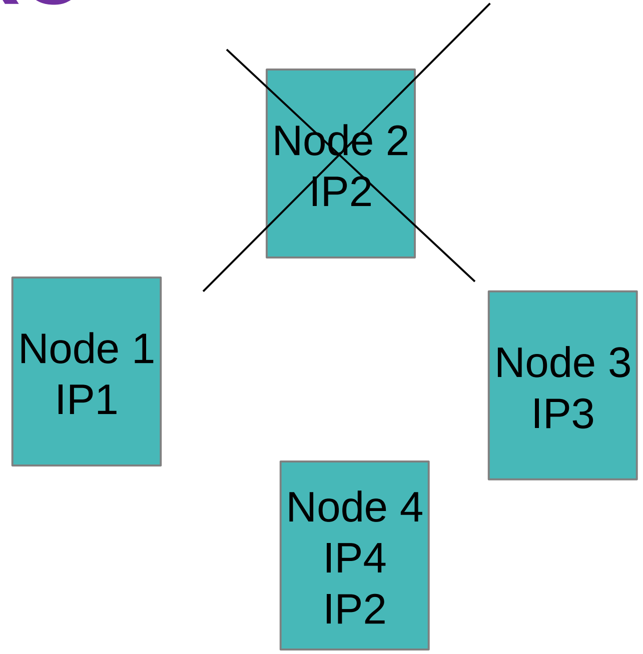
- DRC entries in NFS-Ganesha are stored on Server IP-Granularity. So if a node has multiple IP Addresses, it will have a Duplicate Reply Cache for each IP Address
- Cluster Manager decides the redundancy level for DRC (i.e tolerate single-node failure or double fault etc).
- For each node in the cluster, the cluster manager identifies backup nodes.
- On every NFS request, the reply is first stored in the local DRC. It is also sent to the backup node using the CMAL API.
- Backup node stores the DRC entry, either in the Cluster Manager's memory or in Ganesha memory or persistently on disk.

CMAL and cDRC - Design

- On a failover, the new node that takes over will read all the DRC entries from the backup node (before serving any requests for the new IP).

- CMAL UP interface is used for communication from the Cluster Manager to NFS Ganesha.

cDRC



Primary DRC	Backup DRC
IP1 (Node 1)	IP2 (Node 2)
IP2 (Node 2)	IP3 (Node 3)
IP3 (Node 3)	IP4 (Node 4)
IP4 (Node 4)	IP1 (Node 1)

1. Node 2 fails
2. Node 4 takes over IP2
3. Node 4 reads the DRC entries for IP2 from IP3 (Node 3)

Primary DRC	Backup DRC
IP1 (Node 1)	IP2 (Node 4)
IP2 (Node 4)	IP3 (Node 3)
IP3 (Node 3)	IP4 (Node 4)
IP4 (Node 4)	IP1 (Node 1)

Recovery

- Recovery Requirement

  Both client and server know when the other fails and keeps the state in a sane state

- Failures can be attributed to
  - Client failure
  - Network partition
  - Server failure

Server Failure

- When a node in a cluster fails, we want to seamlessly failover to another node.
  - Move the IP Addresses associated with the Failed node to another node referred to as the Failover Node.
  - Recover any state from the failed node onto the Failover node.
- Difference kinds of failures which will trigger recovery
  - Ganesha daemon failure
  - Node Failure
  - IP move/ failure
  - Network hiccups

CMAL and Recovery

- Store the state (locks, open, delegations) in the cluster. This interface will go through the CMAL layer
    - cmal_store_state(ip_address, state);

- On every lock request, store the state. Cluster manager decides where/how in the cluster the state is stored
    - Can store it on shared filesystem
    - Can store on a primary / backup nodes in the cluster
    - Store it in Cluster Manager's memory

CMAL and Recovery

- Failure will cause two kinds of events
  - RELEASE_IP
  - TAKE_IP
- CMAL Layer provides APIs for handling the IP movement.
    - cmal_release_ip(ip_address);
    - cmal_take_ip(ip_address, state);
- On receiving a RELEASE_IP event, release all the state associated with the failed node
- On TAKE_IP, the takeover node will recover the state from the failed node

CMAL and Grace Period

- Whenever recovery is started, all the nodes need to enter into Grace Period.
- Grace period needs to be co-ordinated across all nodes in the cluster
  - Avoid granting conflicting locks
  - Allow clients to reclaim their existing locks
- CMAL provides APIs to notify Ganesha to start / end Grace period.
  - cmal_start_grace()
  - cmal_end_grace()
  - cmal_is_in_grace()

Final Thoughts

- Most enterprise users of NFS-Ganesha are in a clustered environment; so NFS-Ganesha needs to interact with the Cluster Manager.
- CMAL or Cluster Management Abstraction Layer is the interface between NFS-Ganesha and the Cluster Manager.
- CMAL will make NFS-Ganesha seamlessly work with any backend Cluster Manager.
- NFS-Ganesha is a better fit in a Clustered environment !!!

NFS-Ganesha Links

- NFS-Ganesha is available under the terms of the LGPv3 license
- NFS-Ganesha project homepage on github

  https://github.com/nfs-ganesha/nfs-ganesha/wiki
- Github source

  https://github.com/nfs-ganesha/nfs-ganesha
- Download page

  http://sourceforge.net/projects/nfs-ganesha/files
- Mailing lists

  nfs-ganesha-devel@lists.sourceforge.net
  nfs-ganesha-support@lists.sourceforge.net
  nfs-ganesha-announce@lists.sourceforge.net

Legal Statements

- This work represents the view of the author and does not necessarily represent the view of IBM.
- IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries.
- UNIX is a registered trademark of The Open Group in the United States and other countries .
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Other company, product, and service names may be trademarks or service marks of others
- CONTENTS are "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you. This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Author/IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.