Deploying pNFS solution over Distributed Storage

Jiffin Tony Thottan
Niels de Vos

Agenda

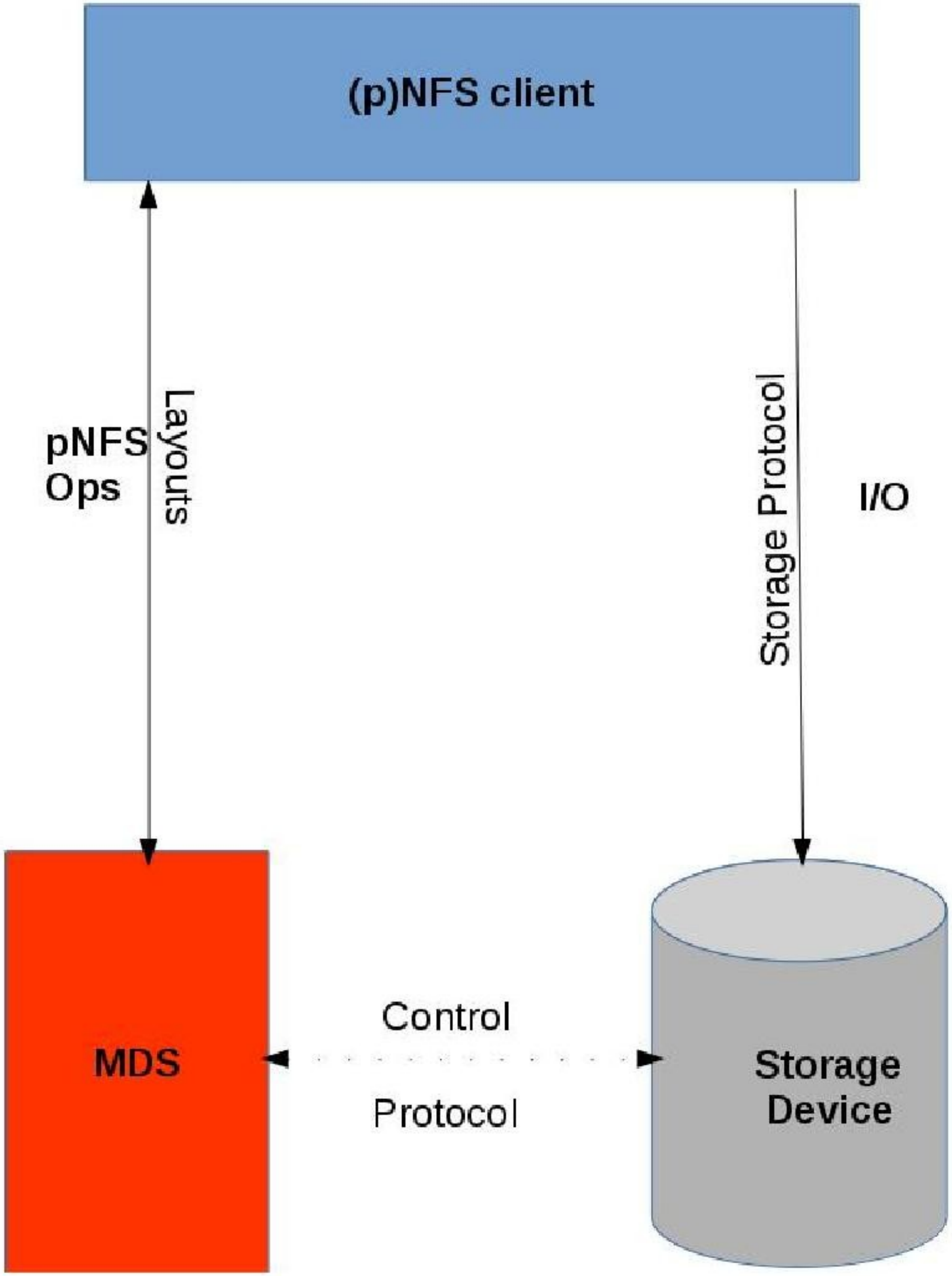
- pNFS protocol
- NFS-Ganesha
- GlusterFS
- Integration
- Challenges
- Configuring pNFS cluster
- Demo

pNFS Protocol

Overview

- pNFS is introduced as part of nfsv4.1 (RFC5661) in 2006
- clients access storage devices directly and in parallel
- Data and Metadata handled in two different paths

Basic pNFS architecture



pNFS terminologies

- MDS – Meta Data Server
  - NFSv4.1 server that supports the pNFS protocol. It provides access to the name space. Also handles I/O in case of failures
- Storage Devices
  - where actual data resides
- Storage Protocol
  - Used between client and Storage devices. It can be NFS, iSCSI, OSD etc
- Control Protocol
  - It maintains cluster coherence and it resides out side of scope standard NFS protocol.

pNFS Layouts

Provides ability to access data for the clients, four types :

- File Layout (*mentioned in RFC5661*)
- Block Layout (*mentioned in RFC5663*)
- Object Layout (*mentioned in RFC5664*)
- Flexfile Layout (*https://tools.ietf.org/html/draft-ietf-nfsv4-flex-files-07*)

pNFS Operations

Following operations are performed from client to MDS :

- GETDEVICEINFO*(device id)*
  - gets information about storage devices
- LAYOUTGET*(file handle, offset, length)*
  - fetch file information about data in the form of layout
- LAYOUTRETURN*(file handle, offset, length, stateid)*
  - releases the layout
- LAYOUTCOMMIT*(file handle, clientid, range,stateid)*
  - commits write using layout to the MDS

pNFS call back operation

Following are notifications send from MDS to client :

- CB_LAYOUTRECALL
  - recalls layout granted to a client
- CB_RECALLABLE_OBJ_AVAIL
  - previously denied layout is available
- CB_NOTIFY_DEVICEID
  - informs client device id is invalid
- CB_RECALL_ANY
  - recalls delegations/layouts whose state can hold by the server

nfsv4.1 as Storage Protocol

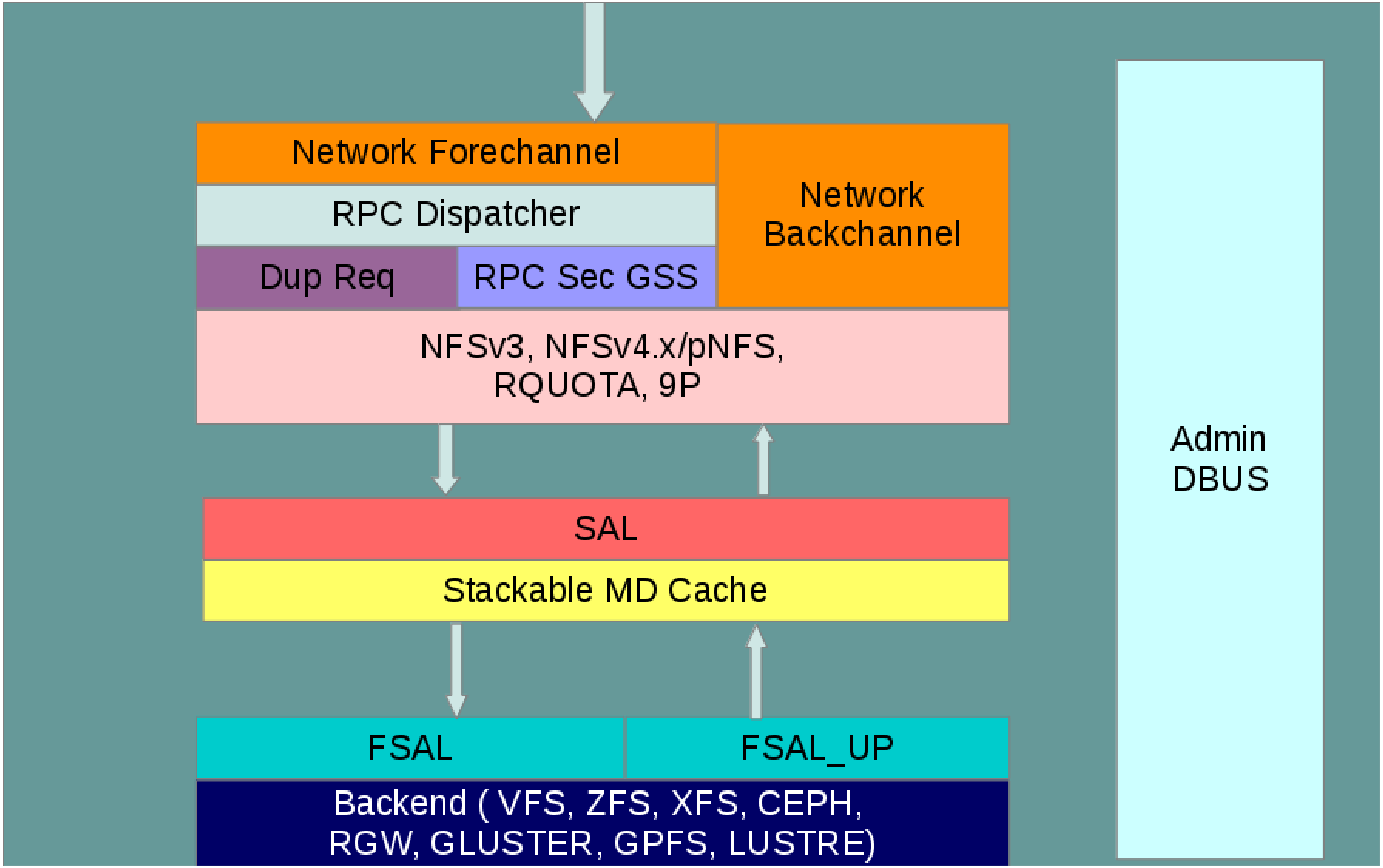
If storage devices is nfsv4.1 server(Data Server) , following additional ops should be defined

- ds_write
- ds_read
- ds_commit

NFS-Ganesha

- A user-space, protocol-complaint NFS server
- Supports NFS v3, 4.0, 4.1, pNFS and 9P from the Plan9 operating system.
- Provides a File System Abstraction Layer(FSAL) to plug in to any own storage mechanism
- Can provide simultaneous access to multiple file systems.
- Small but active and growing community ; CEA, Red Hat, IBM are active participants

NFS-Ganesha architecture



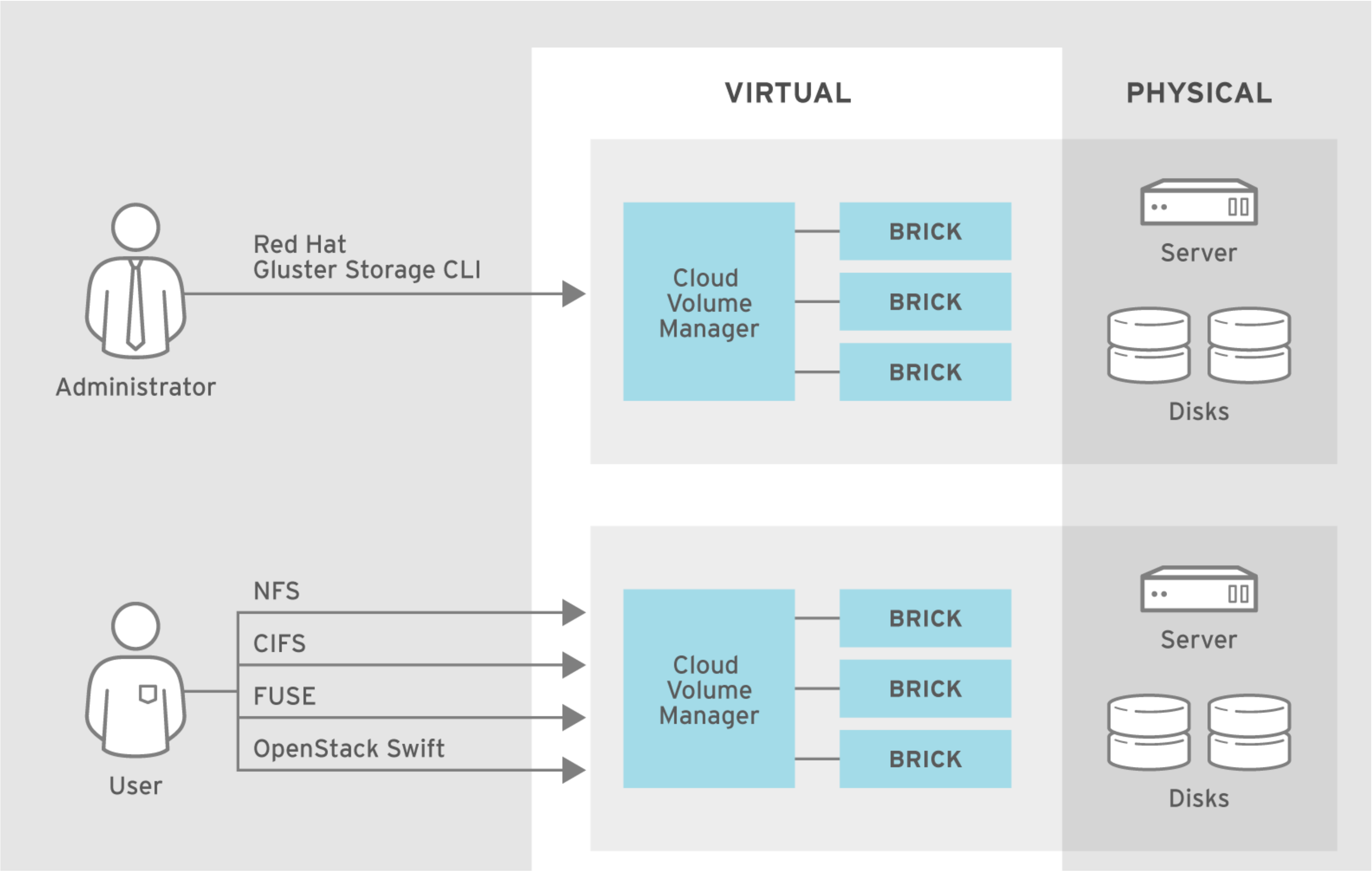
Benefits of NFS-Ganesha

- Can manage huge meta-data caches
- Dynamically export/unexport entries using D-Bus mechanism.
- Easy access to the services operating in the user-space (like Kerberos, NIS, LDAP)
- Provides better security and authentication mechanism for enterprise use
- Portable to any Unix-like file-systems

GlusterFS

- An open source, scale-out distributed file system
- Software Only and operates in user-space
- Aggregates Storage into a single unified namespace
- No metadata server architecture
- Provides a modular, stackable design
- Runs on commodity hardware

GlusterFS architecture



GlusterFS Design

- Data is stored on disk using native formats (e.g. ext4, XFS)
- Has following components
  - Servers known as storage bricks (glusterfsd daemon), export local filesystem as volume
  - Clients (glusterfs process), creates composite virtual volumes from multiple remote servers using stackable translators
  - Management service (glusterd daemon) manages volumes and cluster membership
  - Gluster cli tool

Integration =
GlusterFS + NFS-Ganesha + pNFS

- Introduced in glusterfs 3.7, NFS-Ganesha 2.3
- Supports File Layout
- Entire file will present in a single node
- gfid passed with layout for the communications
- All symmetric architecture – ganesha process can act both as MDS and DS

(conti..) Integration

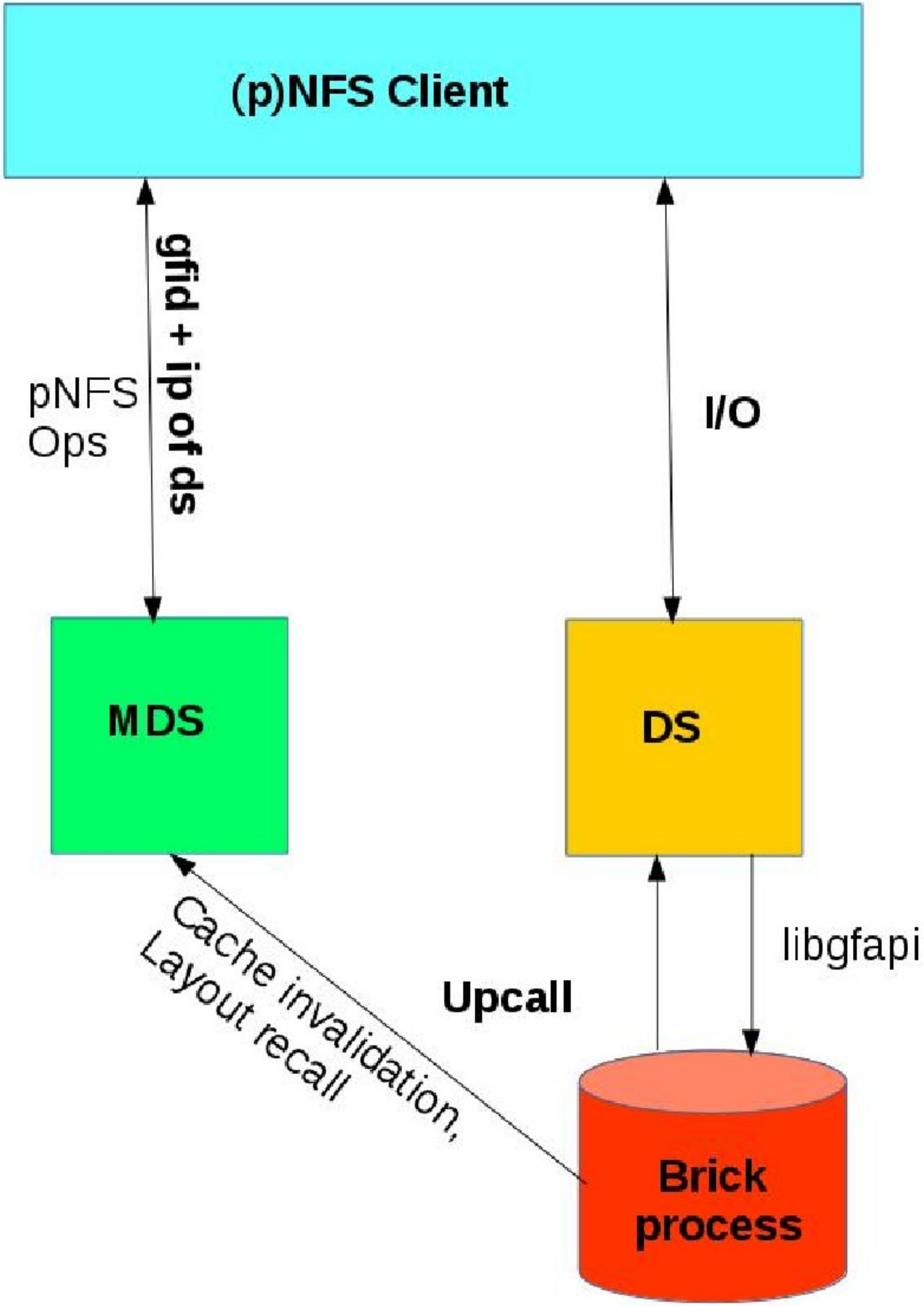
- Commit through DS
- Only single MDS per volume
- Ganesha talks to glusterfs server using libgfapi
- Upcall used to sync between MDS and DS

Libgfapi

- A user-space library with APIs for accessing Gluster volumes.
- Reduces context switches.
- Many applications integrated with libgfapi (qemu, samba, NFS Ganesha).
- Both sync and async interfaces available.
- C and python bindings.
- Available via 'glusterfs-api*' packages.

Upcall Infrastructure

- A generic and extensible framework.
  - used to maintain states in the glusterfsd process for each of the files accessed
  - sends notifications to the respective glusterfs clients in case of any change in that state.
- Cache-Invalidation
  - Invalidate cache used by glusterfs client process
  - #gluster vol set <volname> \
    features.cache-invalidation on/off



(p)NFS Client

pNFS Ops

gfid + ip of ds

I/O

MDS

DS

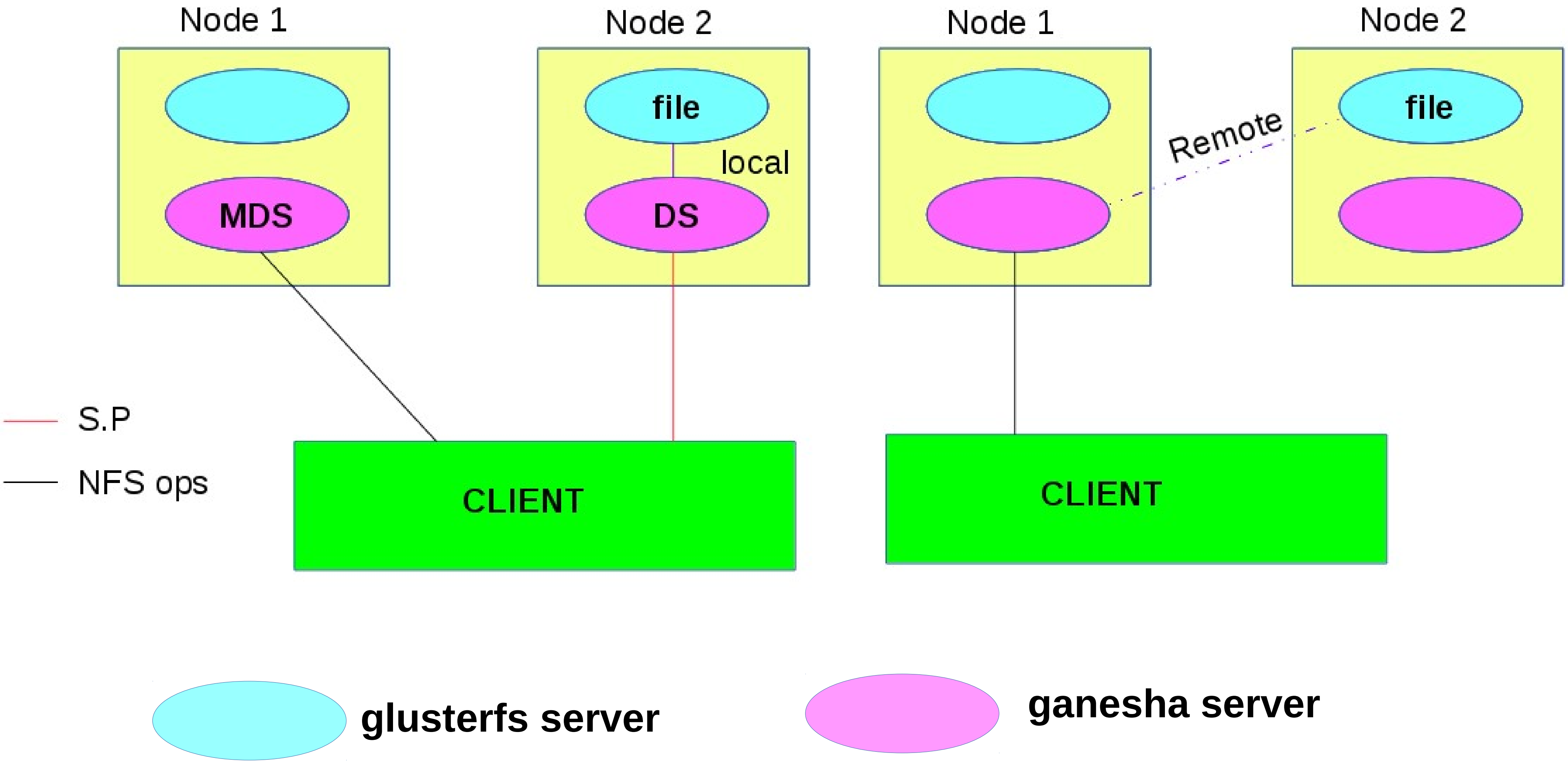
Cache invalidation,
Layout recall

Upcall

libgfapi

Brick
process

pNFS v/s NFSv4



Advantages

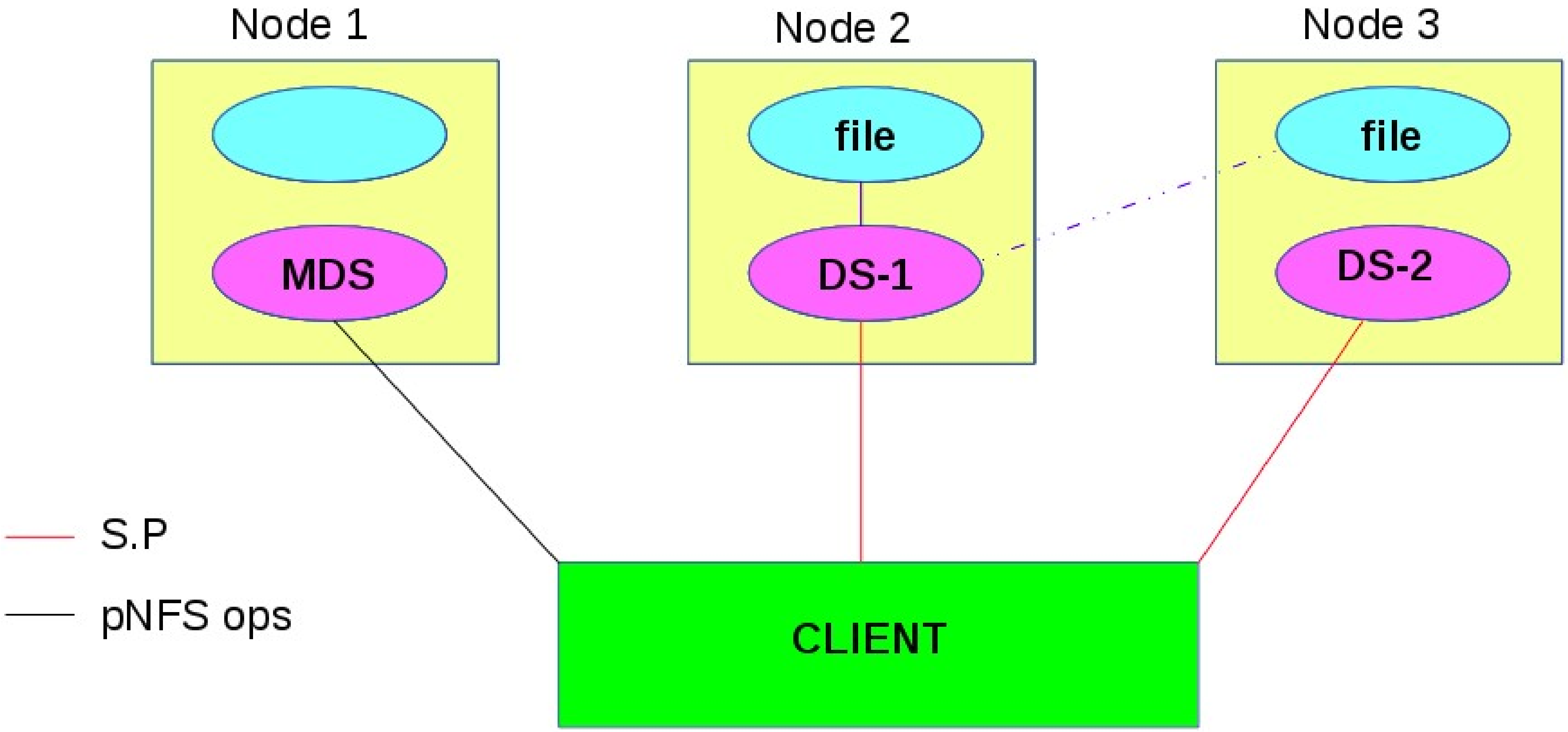
- Better bandwidth utilization
- Avoids additional network hops
- Requires no additional node to serve as MDS
- On different volume configurations

	Performance Improvement		**Load balancing**
	Read	Write	
Distribute	Yes	Yes	No
Replicate	Yes	No	Yes
Disperse	No	No	Yes

Challenges

- Layout information
  - gfid + location + offset + iomode
- Perform I/O without open on DS
  - Similar to anonymous fd writes/reads
- Maintains cache coherency b/w MDS and DS
  - Using cache invalidation feature of upcall infra

- Load balancing between DS servers
  - If there are multiple DSes are available , MDS need to chose one among which guarantees local read or writes



- Store layout information as leases (in development)
  - Lease infrastructure provided by glusterfs server stores information about a layout. So when a conflict requests comes it can recall layout.

Special Mention : Rick Macklem

- pNFS for Gluster NFS server
- GlusterFS has it own inbuild nfsv3 server either know as gNFS or Gluster NFS will act as DS
- Flex File Layout support
- MDS – FreeBSD NFS server exporting GlusterFS volume

Configuring pNFS

- Create and start a glusterfs volume
  - gluster v create <volname> <options> <brick info>
  - gluster v start <volname>
- Turn on cache-invalidation
  - gluster v set <volname> cache-invalidation on
- Adding configuration option for MDS in ganesha conf
  - GLUSTER { PNFS_MDS = true; }
- Start nfs-ganesha process on all storage nodes
  - systemctl start nfs-ganesha
- Mount the volume in the client
  - mount -t nfs -o vers=4.1 \
    <ip of MDS>:/<volname> <mount point>

Future Directions with GlusterFS Features

- Lease Infrastructure
    - Multiple MDS support
- Storhaug
    - HA cluster for MDS
    - configure option for pNFS
- Adding support for sharded volume
- Integrating with DHT2

Demo

- Creating cluster of nodes
- Creating a GlusterFS distribute volume (3x1)
- Exporting volumes via Ganesha server
- Mounting the volume via pNFS protocol
- Performing write operation on client.

References

- Links (Home Page):
  - https://github.com/nfs-ganesha/nfs-ganesha/wiki
  - http://www.gluster.org
- References:
  - http://gluster.readthedocs.org
  - http://blog.gluster.org
  - https://tools.ietf.org/html/rfc5661
  - http://events.linuxfoundation.org/sites/events/files/slides/pnfs.pdf
  - https://people.freebsd.org/~rmacklem/pnfs-setup.txt

Contact

- Mailing lists:
  - nfs-ganesha-devel@lists.sourceforge.net
  - gluster-users@gluster.org
  - gluster-devel@gluster.org

- IRC:
  - #ganesha on freenode
  - #gluster and #gluster-dev on freenode

Q & A

Thank You