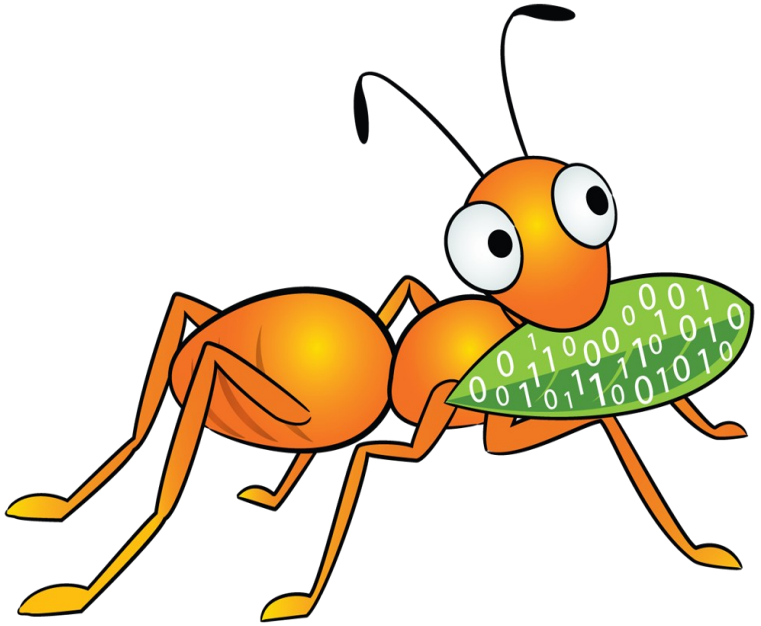
Open Source, Scale-out clustered NAS using nfs-ganesha and GlusterFS



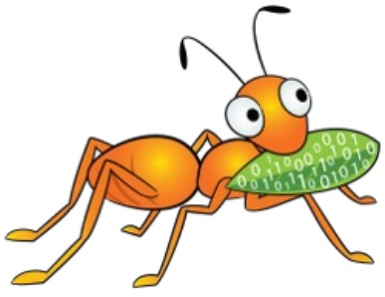
Anand Subramanian

Senior Principal Engineer, Red Hat

anands@redhat.com

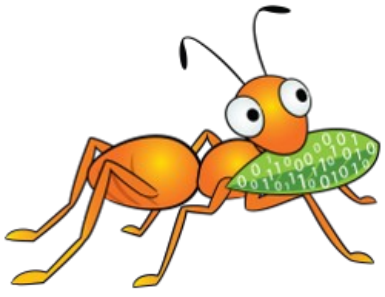
Agenda

- **Introduction**
  - **GlusterFS**
  - **NFSv4**
  - **nfs-ganesha**
- **Nfs-ganesha Architecture**
- **Integration of nfs-ganesha with GlusterFS**
  - **Low-cost scalable clustered NAS solution**
- **Performance**
- **Future Directions**

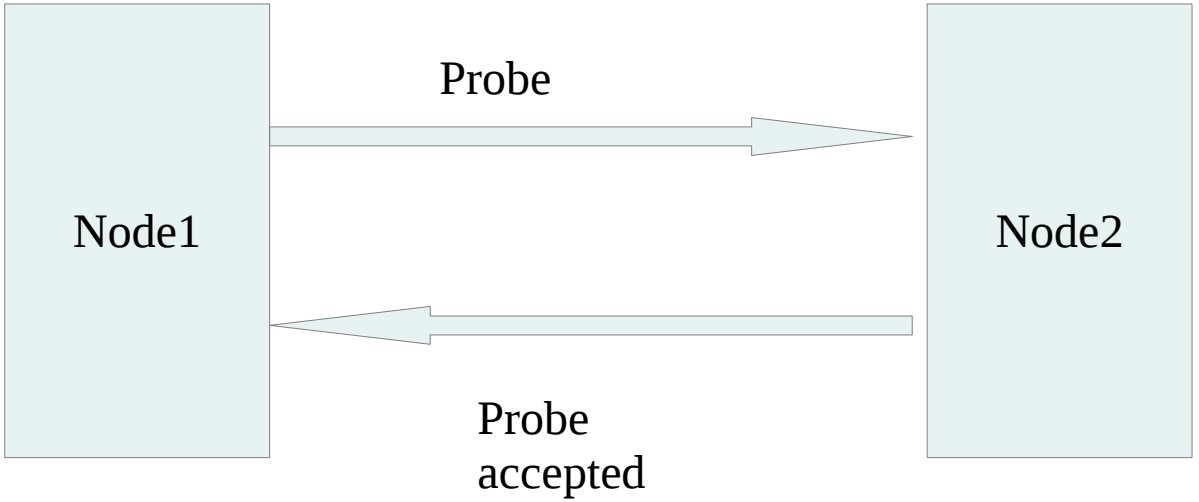


What is GlusterFS?

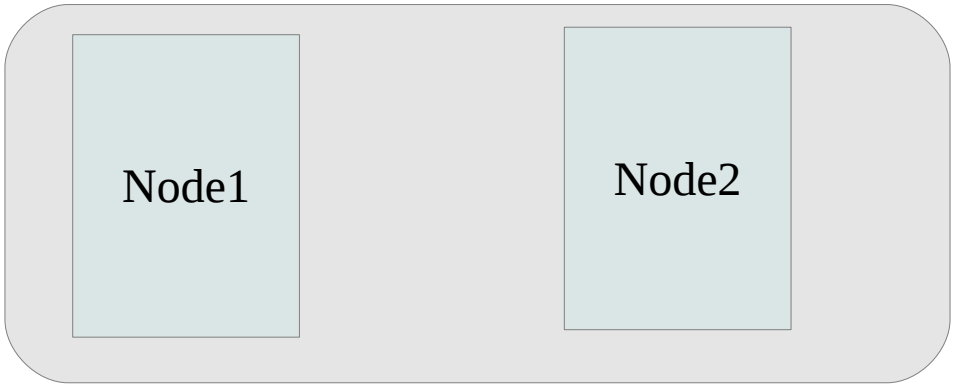
- A general purpose scale-out distributed file system.
- Aggregates storage exports over network interconnect to provide a single unified namespace.
- Filesystem is stackable and completely in userspace.
- Layered on disk file systems that support extended attributes.

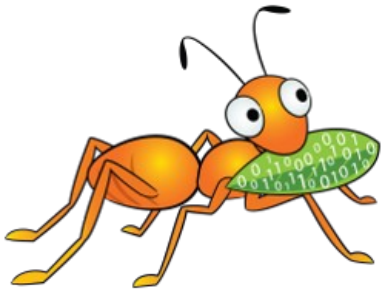


GlusterFS concepts – Trusted Storage Pool



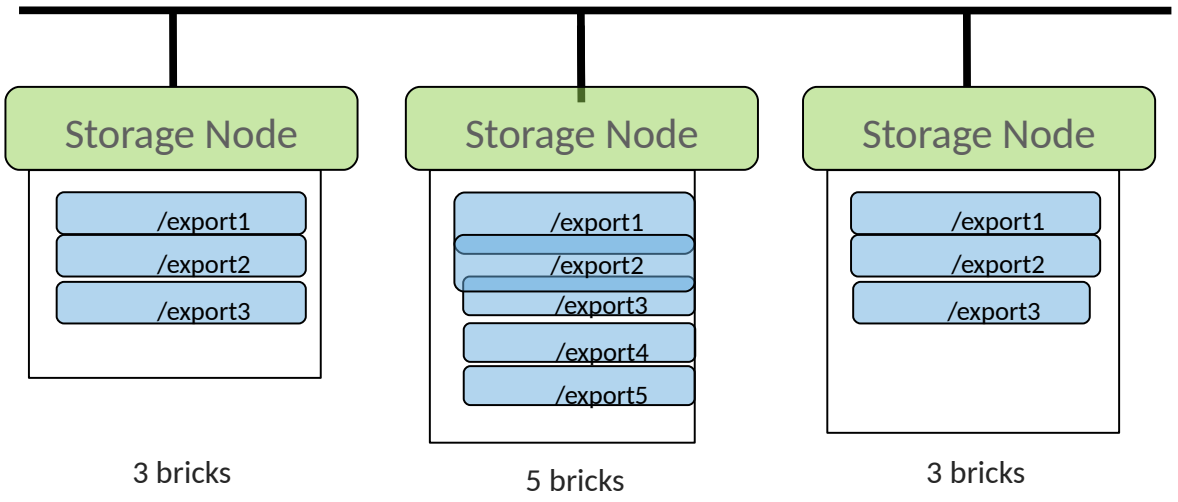
Node 1 and Node 2 are peers in a trusted storage pool

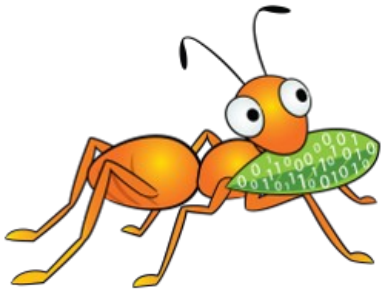




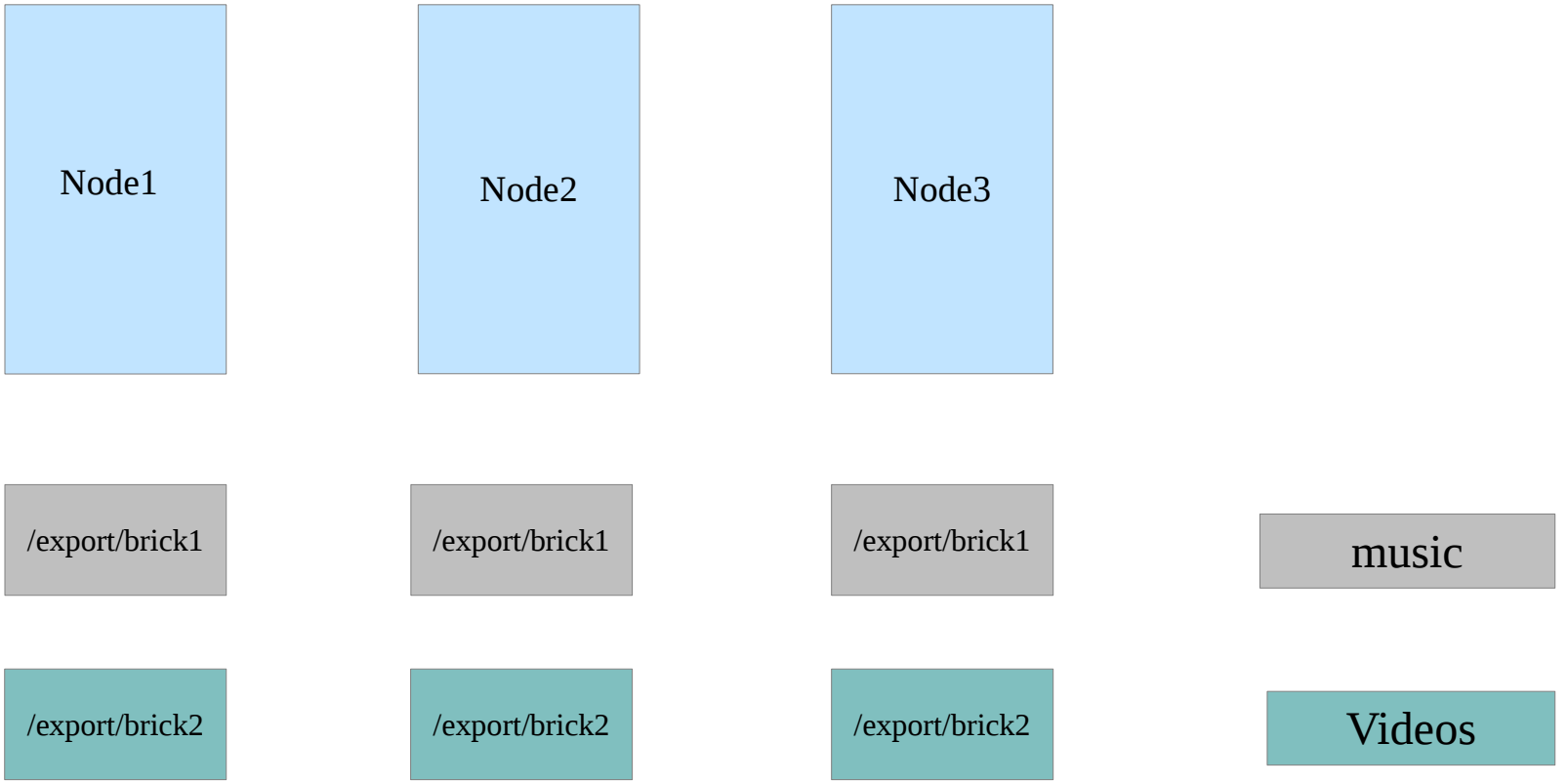
GlusterFS concepts - Bricks

- A brick is the combination of a node and an export directory – for e.g. hostname:/dir
- Each brick inherits limits of the underlying filesystem
- Ideally, each brick in a cluster should be of the same size





GlusterFS concepts - Volumes

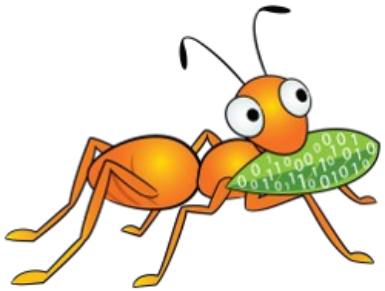


Volume Types

- Type of a volume is specified at the time of volume creation

- Volume type determines how and where data is placed

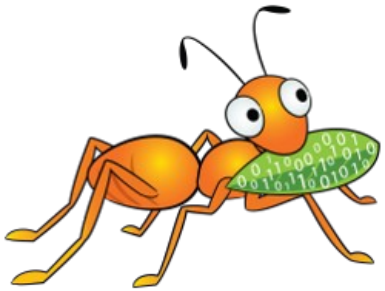
- Following volume types are supported in glusterfs:
    a) Distribute
    b) Stripe
    c) Replication
    d) Distributed Replicate
    e) Striped Replicate
    f) Distributed Striped Replicate



Access Mechanisms

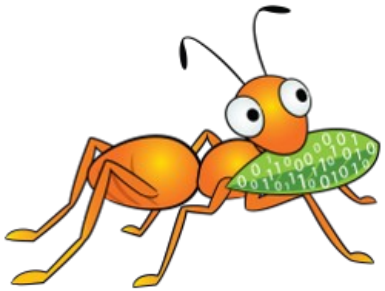
Gluster volumes can be accessed via the following mechanisms:

- FUSE based Native protocol
- NFS – V3, V4, V4.1/pNFS, V4.2
- SMB
- libgfapi
- ReST/HTTP
- HDFS

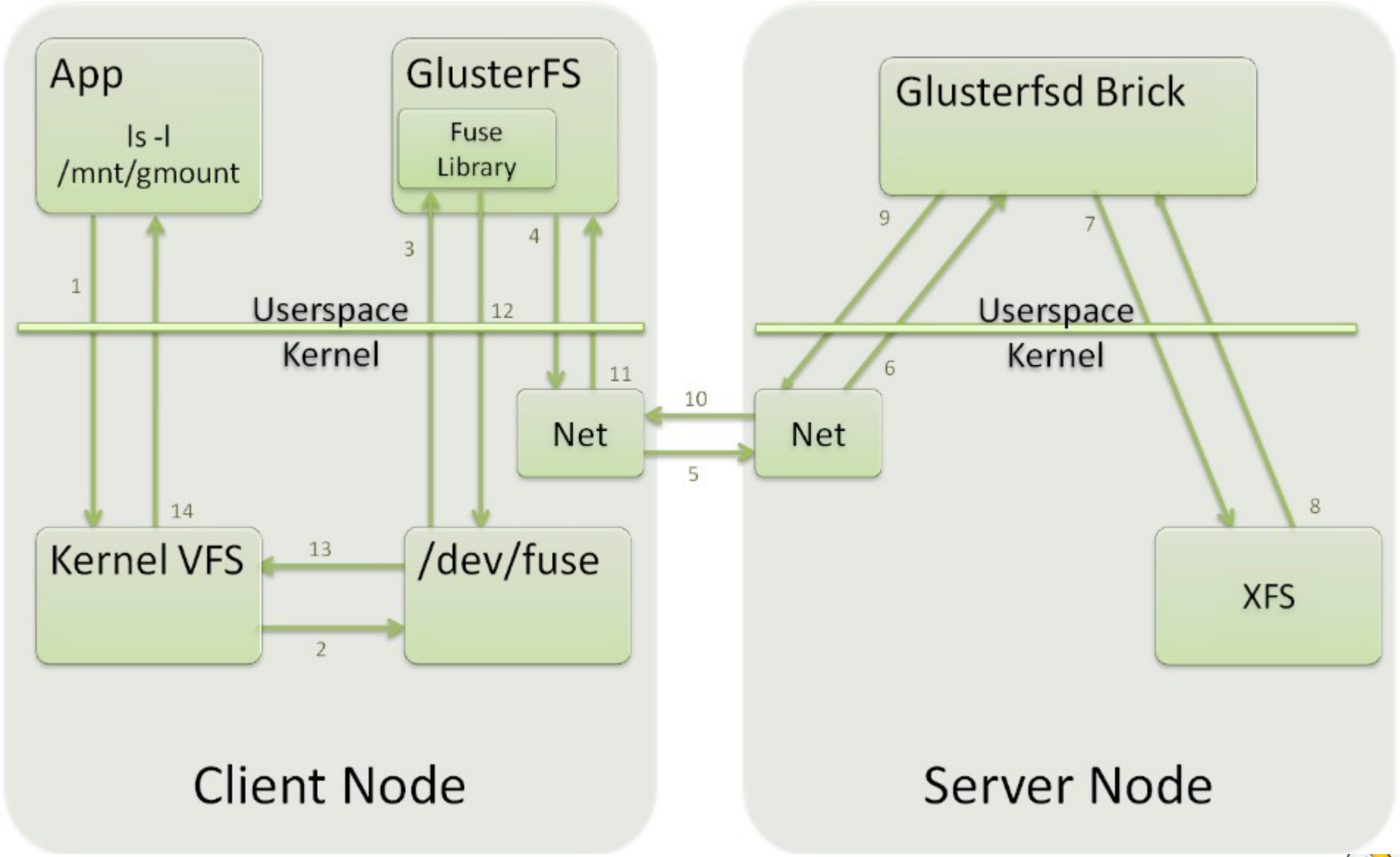


libgfapi

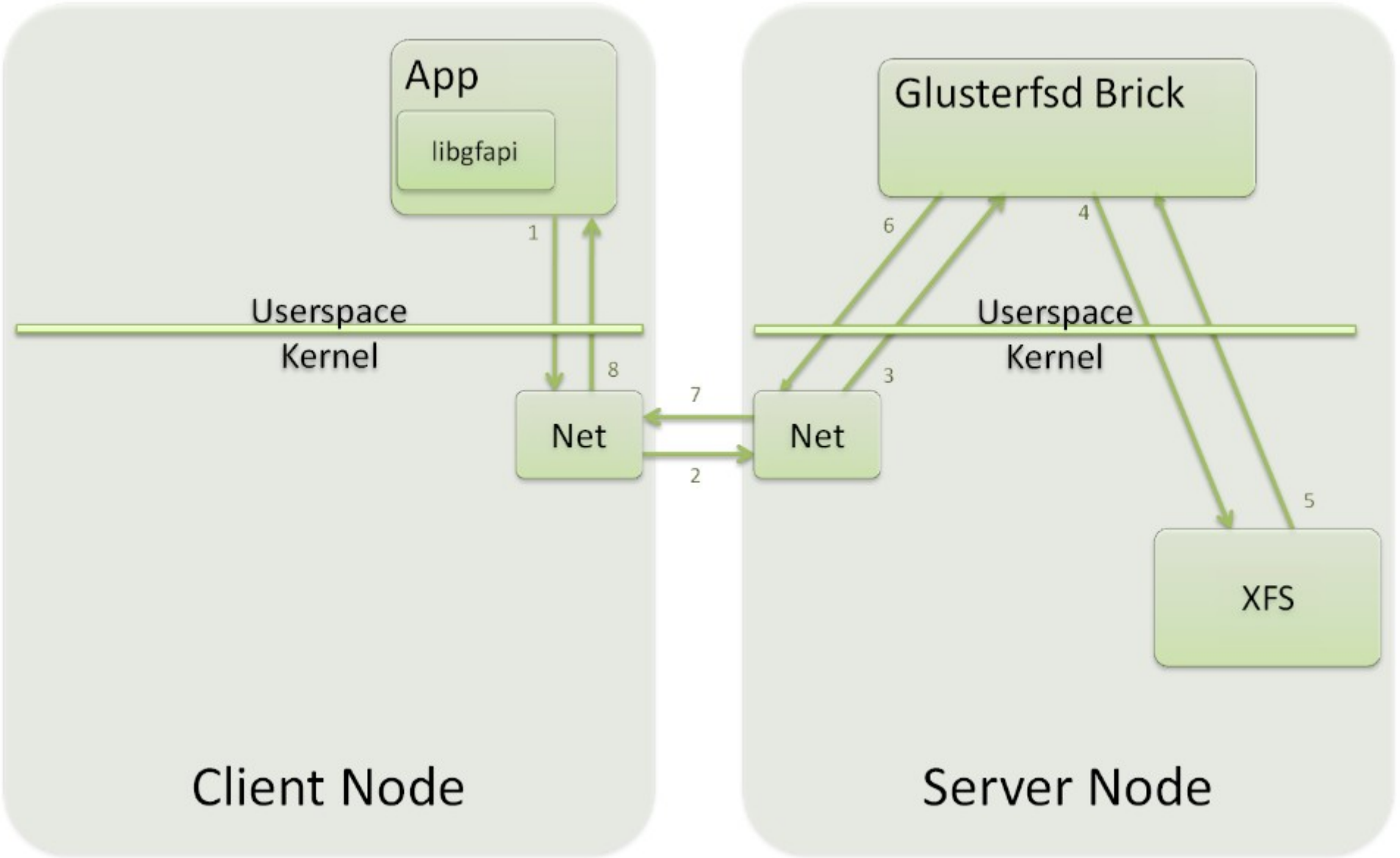
- GlusterFS Exposes APIs for accessing the filesystem/Gluster volumes
- Reduces context switches
- qemu, samba integrated with libgfapi
- Both sync and async interfaces available
- Emerging bindings for various languages – python, Go etc.



libgfapi v/s FUSE – FUSE access

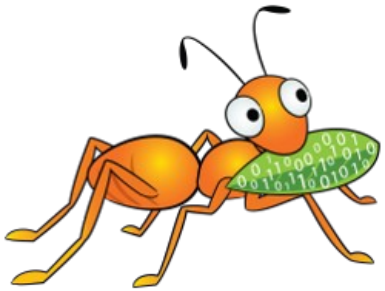


libgfapi v/s FUSE – libgfapi access



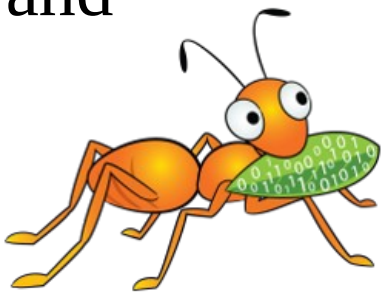
Ecosystem Integration

- Currently GlusterFS filesystem is integrated with various ecosystems:
  - OpenStack
  - Samba
  - Ganesha
  - oVirt
  - qemu
  - Hadoop
  - pcp
  - Proxmox
  - uWSGI etc.



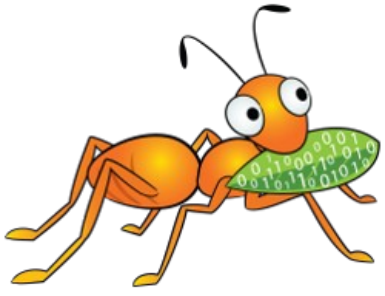
GlusterFS and NFS

- Gluster had its own NFS v3 server for access to the filesystem/storage volumes
- NFSv4 as a protocol has developed
  - Been ironed out for almost a decade
  - Slow at first but increased interest and adoption across the industry
  - Firewall friendly, UNIX and Windows friendliness, better security
  - Paves the way for the interesting stuff in 4.1/4.2 and beyond



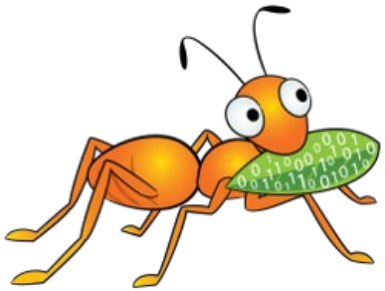
nfs-ganesha

- **User-space NFS server; developed by Daniel Philippe in FR**
- **Supports V2, V3, V4, v4.1/pNFS, v4.2**
- **Can manage huge meta-data and data caches**
- **Provision to exploit FS specific features**
- **Can serve multiple types of File Systems at the same time**
- **Can act as a Proxy server**
- **Cluster Manager agnostic**
- **Small but active and growing community**
- **Active participants - IBM, Panasas, Redhat, LinuxBox, CES etc.**

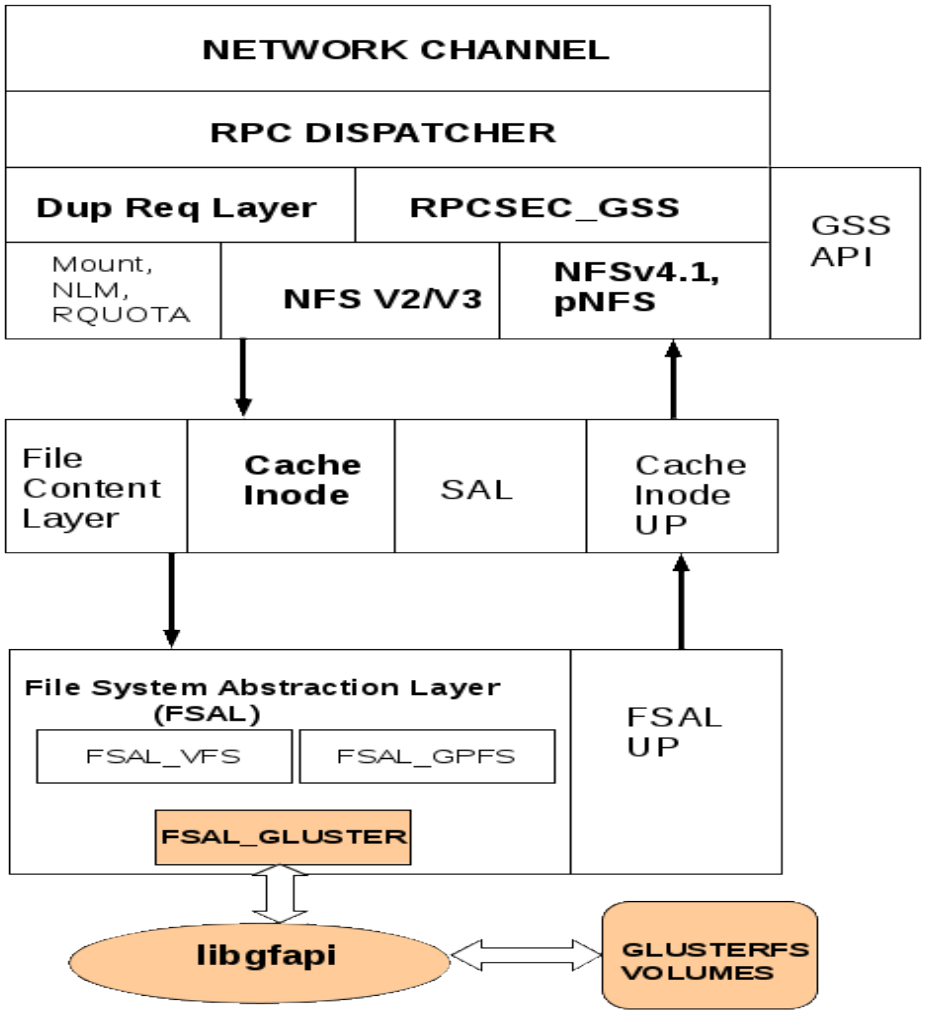


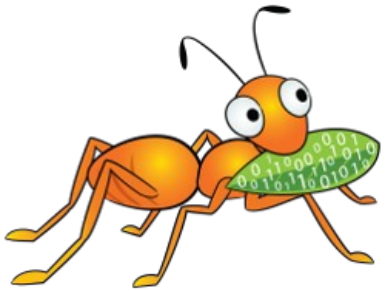
So, why user space?

- **What is so great about user-space?**
  - **User Space is more flexible than kernel**
  - **Easy restarts, failover, failback implementation**
  - **Clustering becomes natural and easy**
  - **Targeted and aggressive caching capabilities**
  - **Flexible and Plug-able FSAL : FS Specific features can be exploited**
  - **Multi-Protocol easier to implement**
  - **Easier to achieve multi-tenancy**
  - **Easy to monitor and control resource consumption and even extend to other features like enforcing QoS**
  - **Manageability and debug-ability are major plus**
  - **Finally – GlusterFS is a user-space FS**



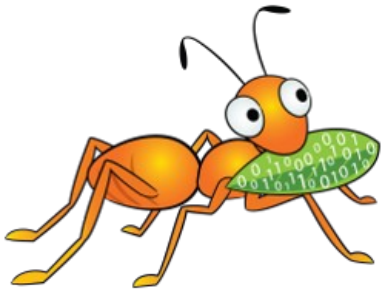
nfs-ganesha : Architecture





Other Enhancements

- **Dynamic Exports**
- **V4 R-O and W-O Delegations**
- **FSAL enhancements**
- **ACL support.**
- **Open upgrade support.**
- **Stackable FSALs.**
- **PseudoFS as first class FSAL.**
- **LTTng Integration.**
- **RDMA support through libmooshika.**

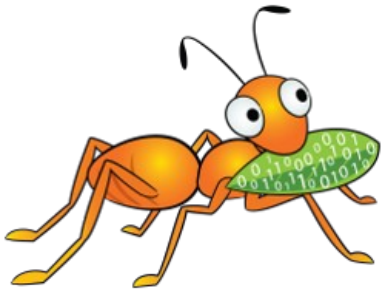


CMAL (Cluster Manager Abstraction Layer)

- Proposed Cluster Manager Abstraction Layer (CMAL) (under design and prototyping in the community):
  - makes it cluster agnostic. Works with any backend CM with a pluggable library
  - Fail-over and IP move handling
  - Enables handling of DLM across cluster or any cluster state
  - Cluster wide Grace period can be easily managed
  - Enables creation of a cluster of ganesha heads or can plugin into any existing cluster in the backend

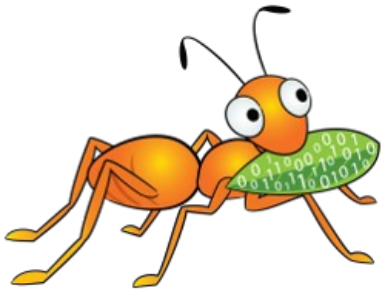
CMAL (Cont'd...)

- Provides an abstraction for cluster manager support
- Manages intra-node communication among cluster nodes
- Generic enough to implement many clustering interactions
- Modeled after FSAL (File System Abstract Layer)
- CMAL layer would have function pointers based on the Cluster Manger (a pluggable shared library for different CM interaction for use with different filesystems/FSALs)
- Works for both clustered and non-clustered filesystems (e.g. some that don't have any built-in clustering support)



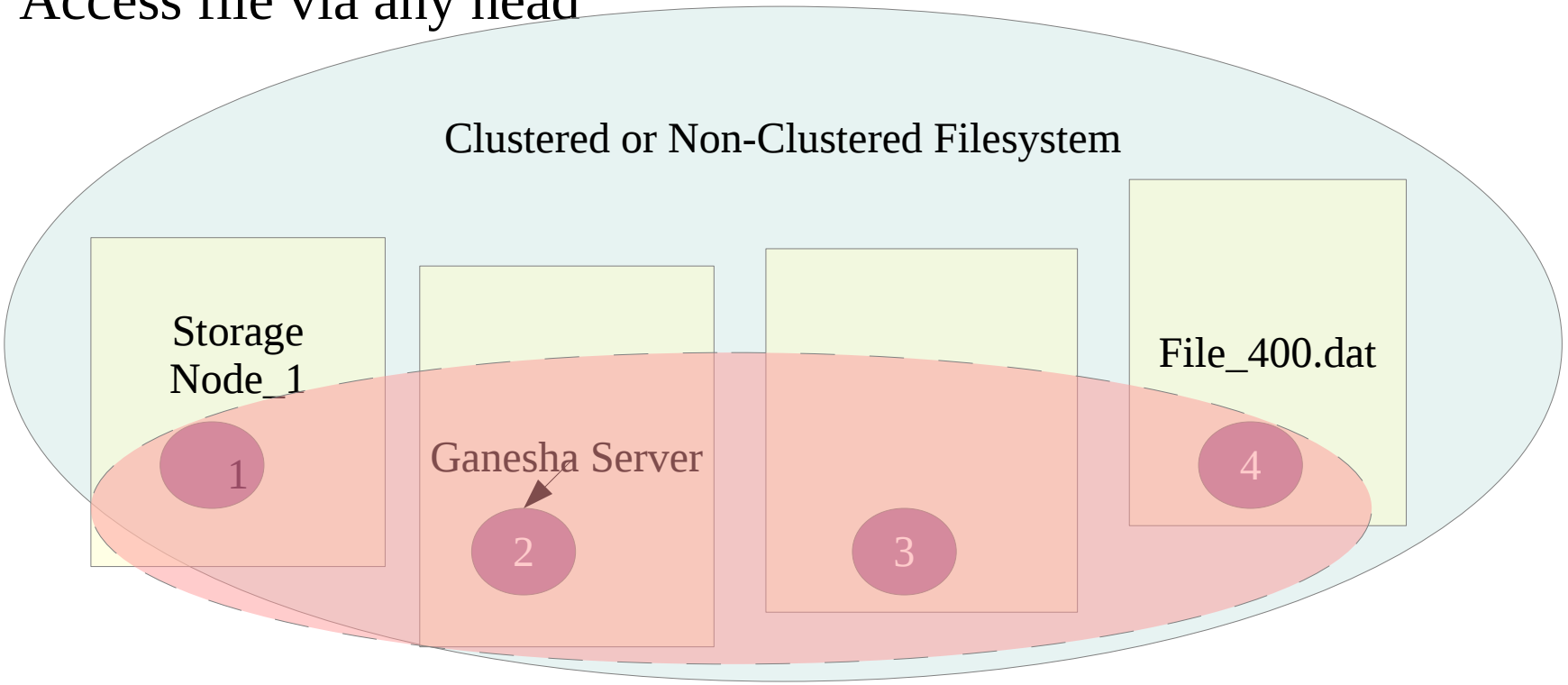
CMAL (Cont'd...)

- NLM makes it very complex but NFS-Ganesha architecture is up for the challenge. :)
- On the first lock/last unlock, Ganesha calls Cluster Manager provided interface to register(monitor)/ unregister(unmonitor) the client-ip,server-ip pair
- When the ip is moved (manual/failover), CM sends sm_notify to clients of the affected service-ip.
- CM generates events, release-ip and take-ip for corresponding server nodes, so that state shall be released from the source node, and acquired by the destination node.
- Depending on the lock granularity, corresponding locks/file systems or entire cluster could be put into NFS grace period

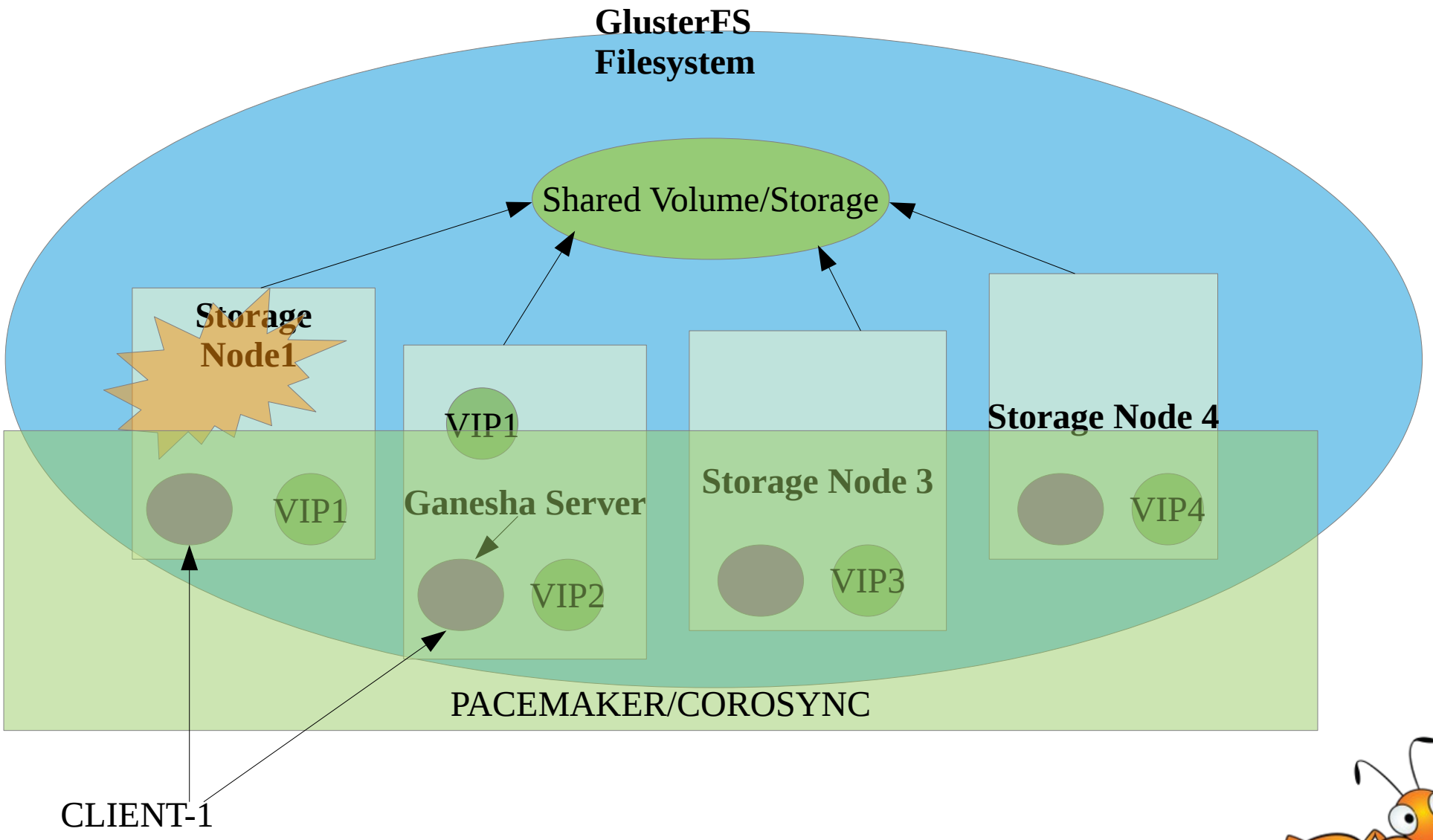


Ganesha in Cluster-mode

- Multi-head NFS with integrated HA (active-active)
- Access file via any head

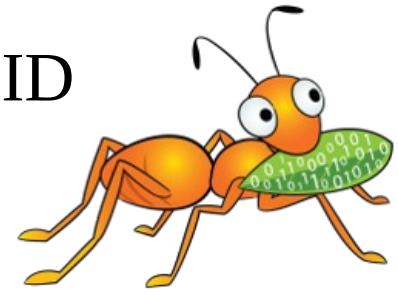


Clustered NAS



V4 Multi-head

- Pacemaker/Corosync the underlying clustering infra managing the ganesha heads & Virtual IP associated with each ganesha head
- Persistent shared storage (another gluster vol) shared by the ganesha heads to store cluster-state (lock state /var/lib/nfs)
- ON FO client gets redirected to a pre-designated head (specified in a config file)
- Entire cluster of ganesha heads put in NFS CLUSTER_GRACE
- V3 and V4 lock recovery procceds via SM_NOTIFY calls or via V4 lock reclaim process during grace period
  - We rely on STALE CLIENTID and STALE STATEID



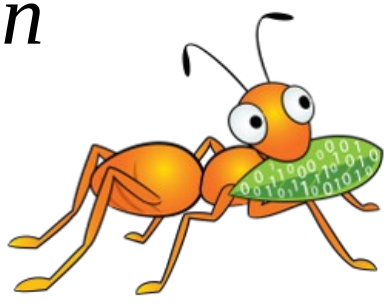
V4 Multi-head (Cont'd...)

- Upcall infrastructure added to GlusterFS for
  - Cache Inode invalidation in nfs-ganesha
  - LEASE LOCK requests conflicts (DELEGATION recall)
  - Support for share reservations etc.

- Is this true NFSv4 active-active?
  - “True” active-active is when clients may not even get a stale stateid and are completely oblivious to changes on server side
  - Downside – too much cluster-aware state needed

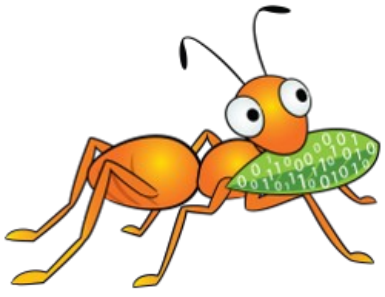
Scaling out

- Add one or more nodes (each with one or more bricks) to the GlusterFS trusted storage pool
  - Normal GlusterFS scale-out operation during run-time
  - Seamless addition of nodes/bricks
- Add one or more nfs-ganesha V4 heads for added availability, scale-out, load-balancing etc.
  - Current limit is 16 (corosync limitation)
  - Scale to 64+ NFS ganesha heads using Pacemaker-Remote
- *Very low cost scale out clustered NAS solution*



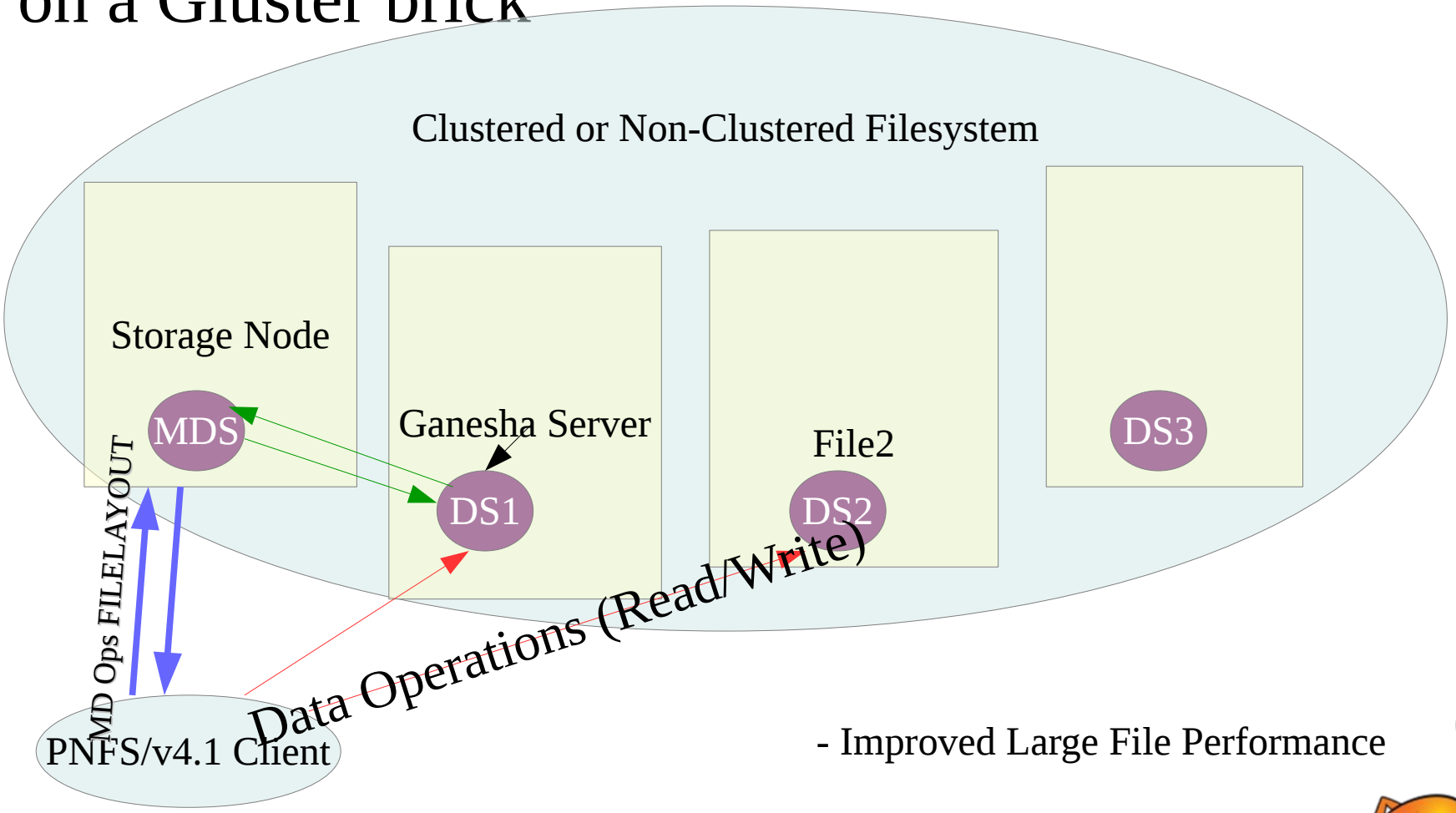
pNFS

- nfs-ganesha supports the pNFS/V4.1 protocol
- FSAL architecture enables easy plugin into core protocol support for any filesystem
- Support for pNFS protocol ops added to FSAL_GLUSTER
- nfs-ganesha and GlusterFS integration means pNFS/V4.1 suppport for GlusterFS Volumes



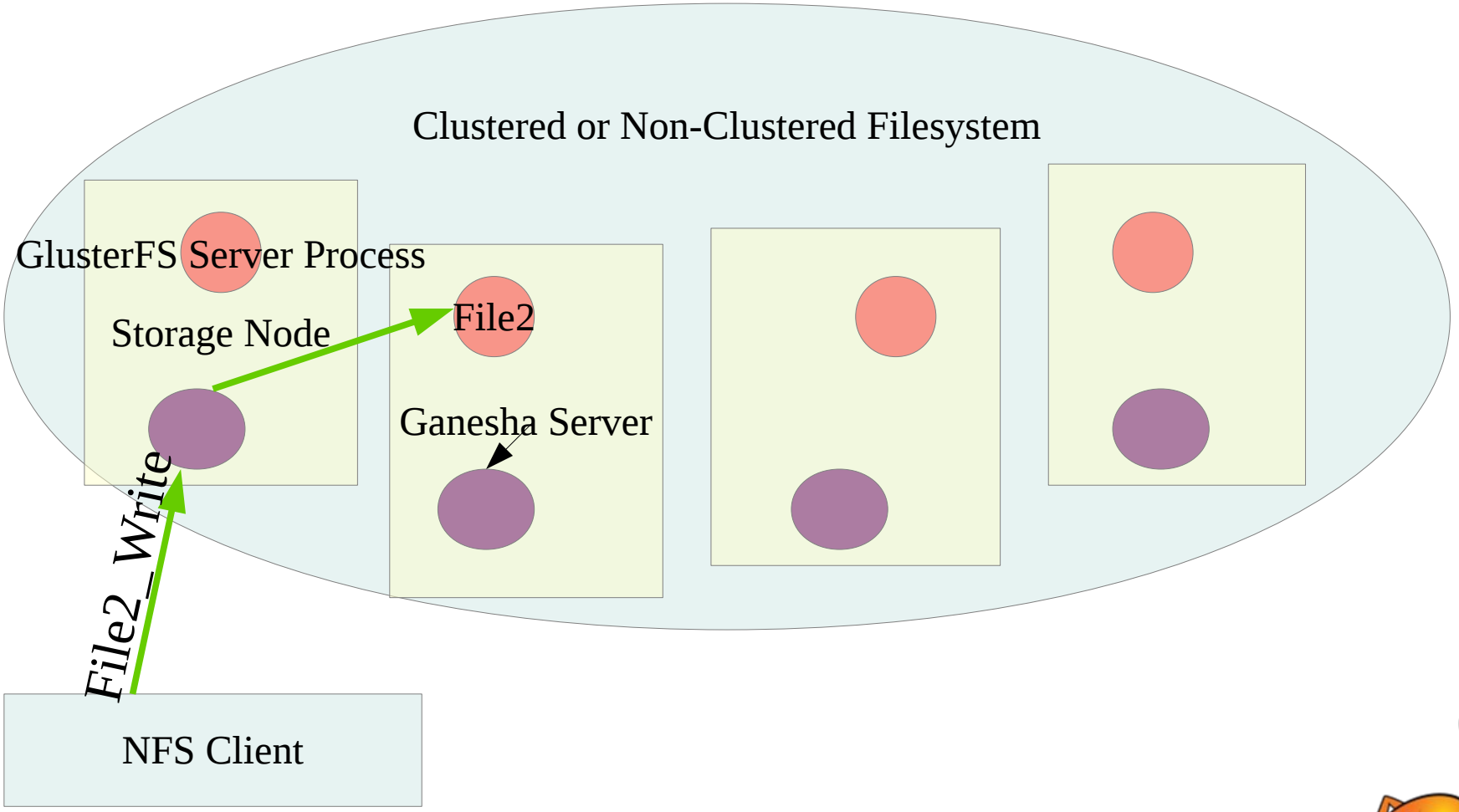
GlusterFS pNFS access using nfs-ganesha

- Supports only PNFS FILE LAYOUT – a file resides on a Gluster brick



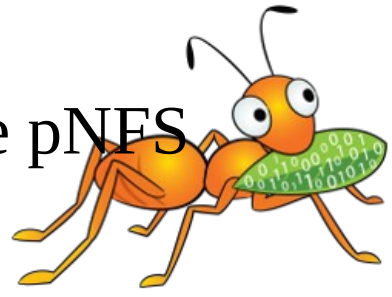
- Improved Large File Performance

Ganesha in NFSv4 Cluster-mode

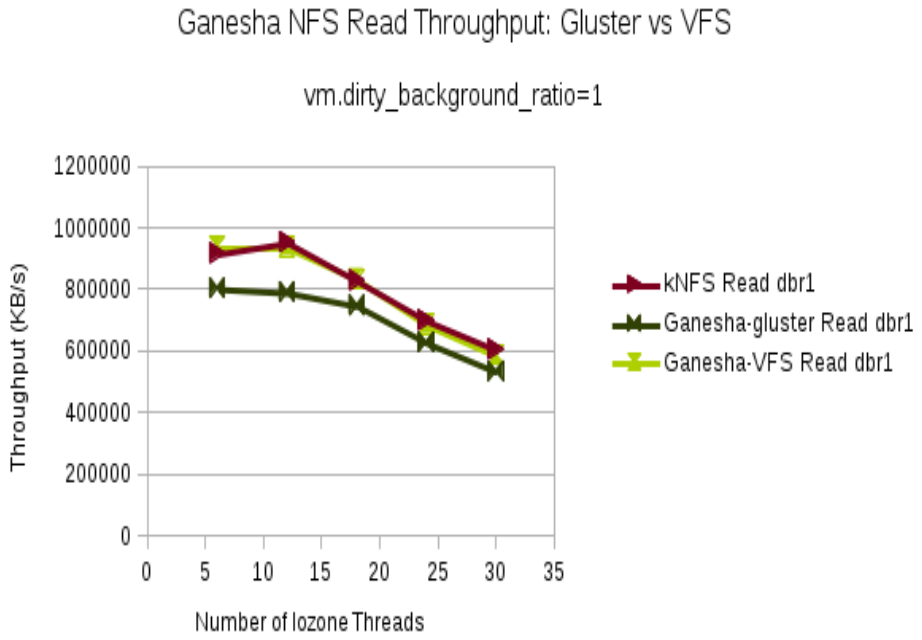


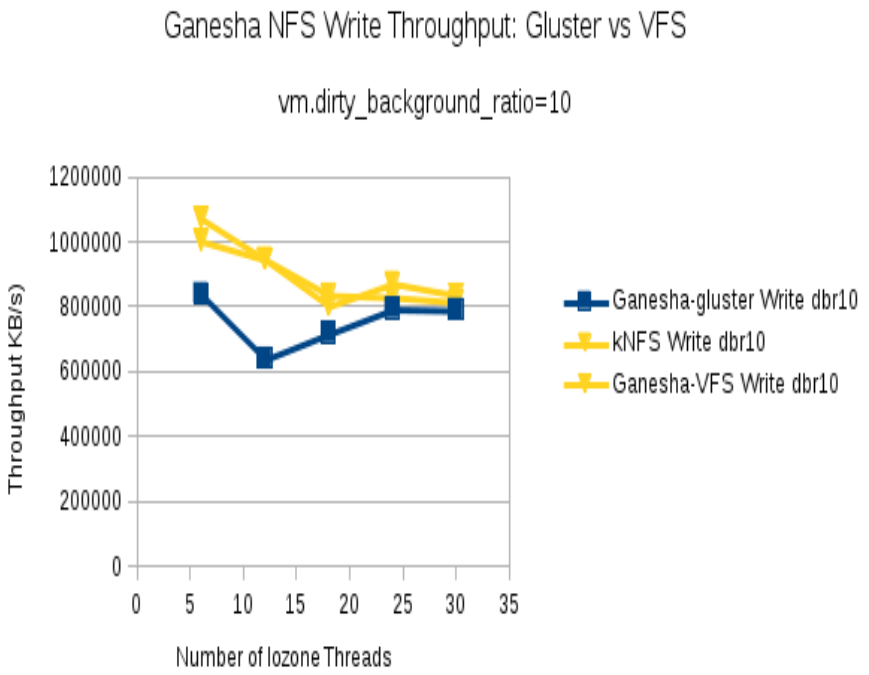
pNFS Benefits

- Better bandwidth utilization & load balancing across the storage pool
- Improved large file reads/writes as expected
- More interesting configurations
  - All symmetric : every node can act as MDS as well as DS
  - Any combination of MDS-es and DS-es possible
  - Works but needs UPCALL support from GlusterFS wrt LAYOUT RECALL etc.
- Scale out by adding as many ganesha nodes as storage pool nodes depending on access requirements!
- Low cost : Probably the least expensive all open source pNFS server for FILE_LAYOUT? :)

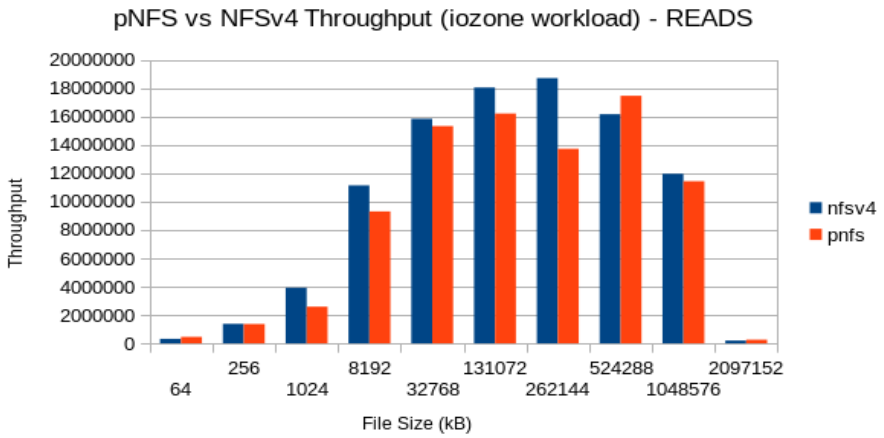


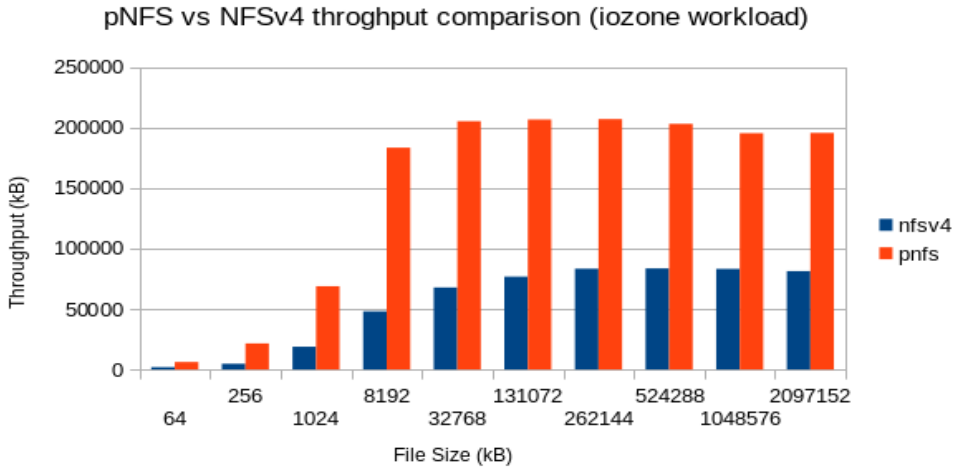
Performance (Ganesha+Gluster in V4 versus kernel-NFS)

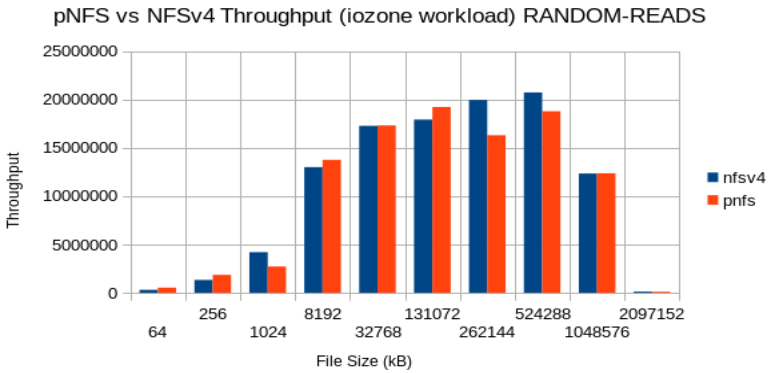


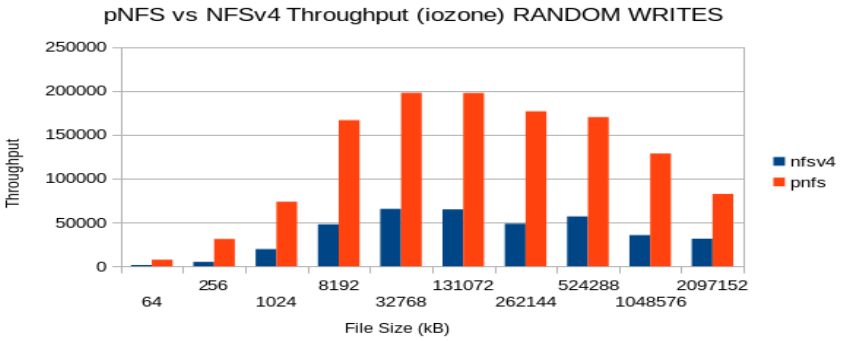


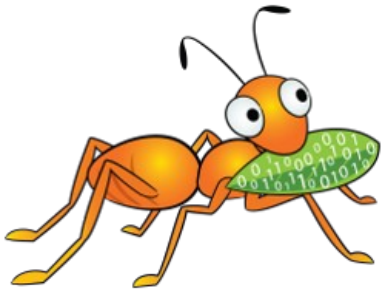
Ganesha + Gluster (V4 vs pNFS)











Next Steps

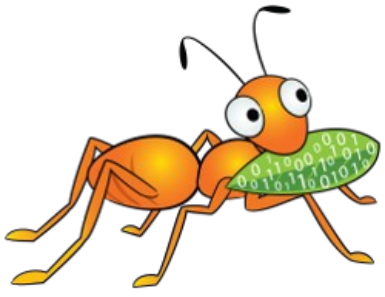
- NFSv4 paves the way forward for interesting stuff
- Adding NFSV4.2 feature support for GlusterFS. Use case : better support for VM environments:
  - ALLOCATE/DEALLOCATE – allows VMs to do space reclamation, hole-punching
  - Server side copy support (client does not need to be involved in copying of a large file)
  - ADB – init a very large file with a pattern without transferring the entire file across the network
  - FedFS
  - Labeled-NFS
  - Flex File Layouts in pNFS

Resources

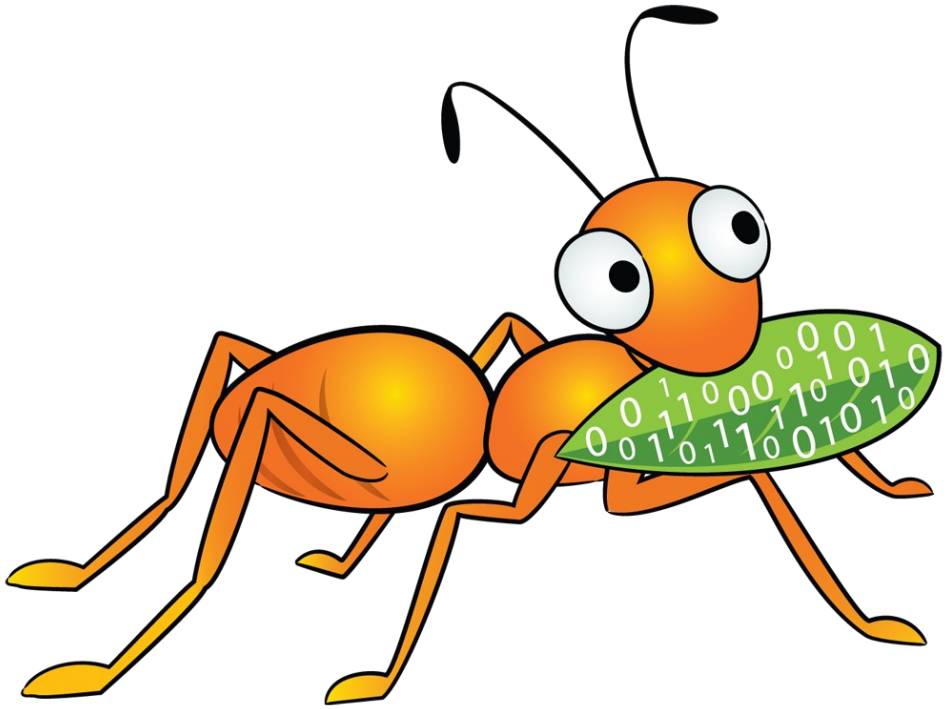
Mailing lists:
nfs-ganesha-devel@lists.sourceforge.net
gluster-users@gluster.org
gluster-devel@nongnu.org

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

Links (Home Page):
http://sourceforge.net/projects/nfs-ganesha
http://www.gluster.org



Thank you!



Anand Subramanian

Twitter : @ananduw

anands@redhat.com