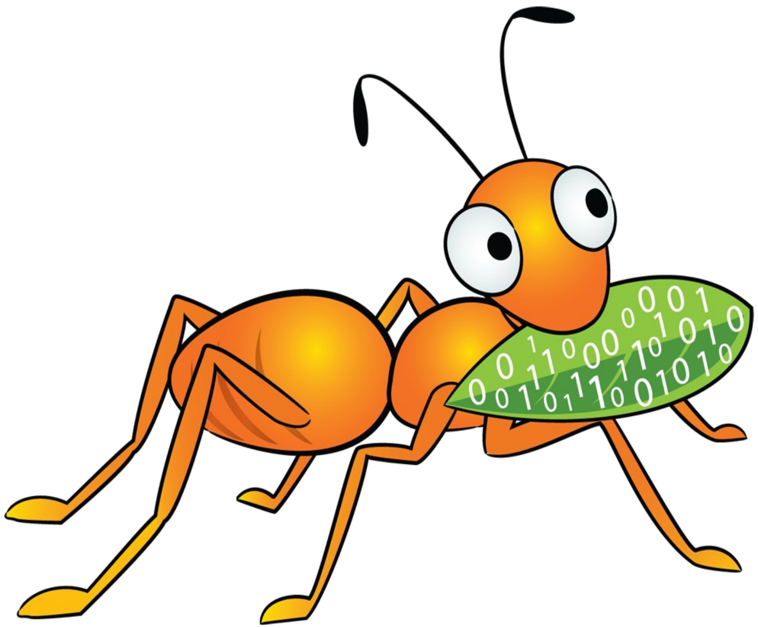
GlusterFS – Architecture & Roadmap



Vijay Bellur
GlusterFS co-maintainer

http://twitter.com/vbellur

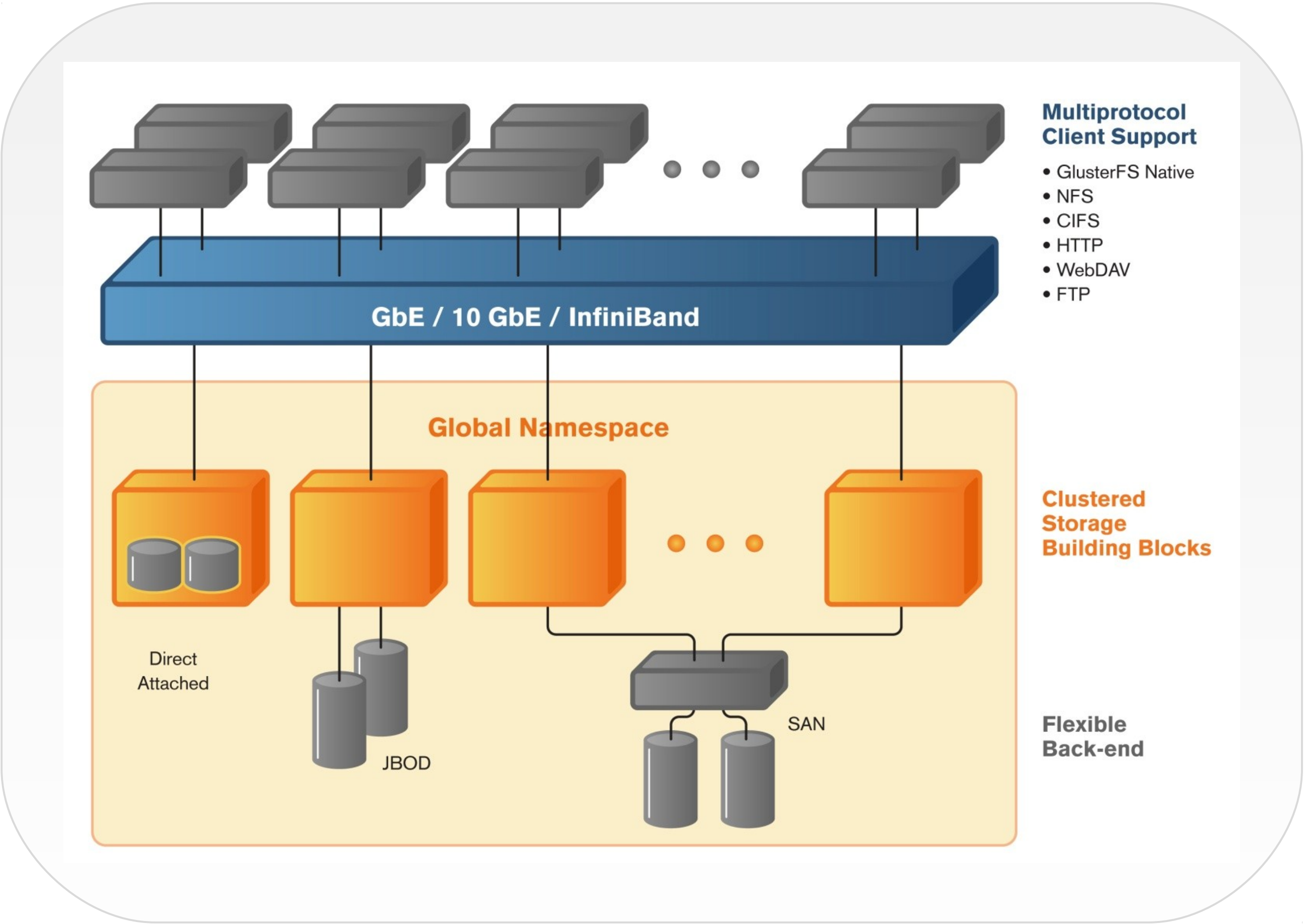
Agenda

- **What is GlusterFS?**
- **Architecture**
- **Integration**
- **Use Cases**
- **Future Directions**
- **Challenges**
- **Q&A**

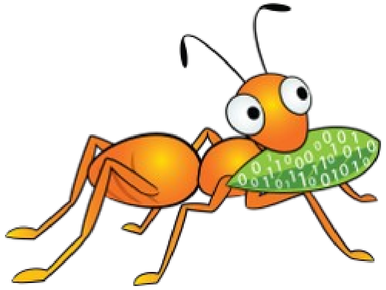
What is GlusterFS?

- 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.

Typical GlusterFS Deployment



- Global namespace
- Scale-out storage building blocks
- Supports thousands of clients
- Access using GlusterFS native, NFS, SMB and HTTP protocols
- Linear performance scaling



GlusterFS Architecture – Foundations

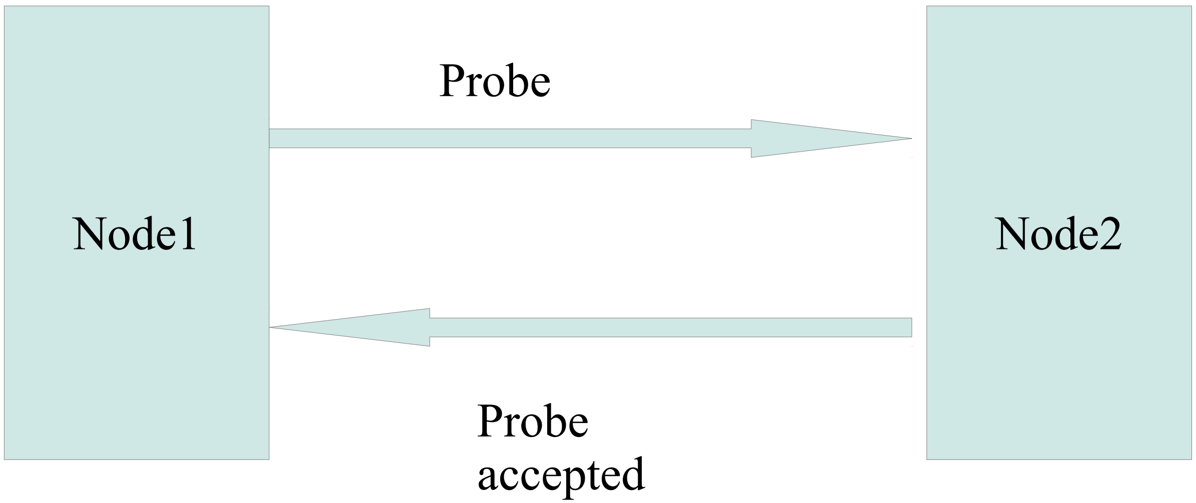
- Software only, runs on commodity hardware
- No external metadata servers
- Scale-out with Elasticity
- Extensible and modular
- Deployment agnostic
- Unified access
- Largely POSIX compliant

Concepts & Algorithms

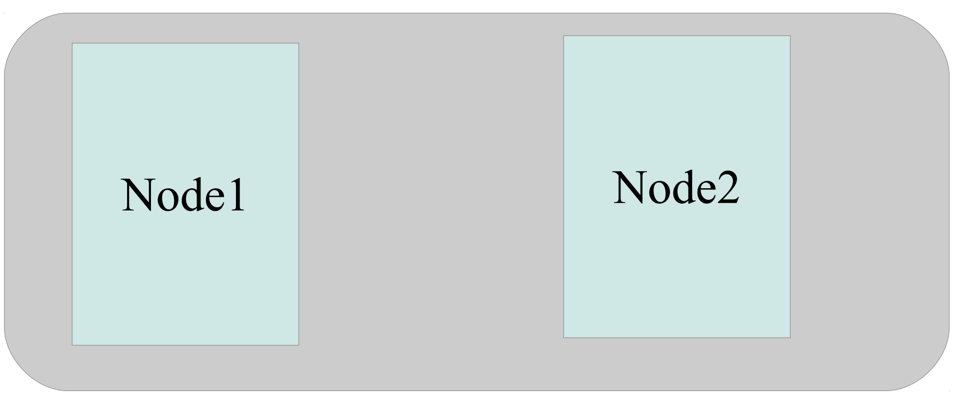
GlusterFS concepts – Trusted Storage Pool

- Trusted Storage Pool (cluster) is a collection of storage servers.
- Trusted Storage Pool is formed by invitation – “probe” a new member from the cluster and not vice versa.
- Logical partition for all data and management operations.
- Membership information used for determining quorum.
- Members can be dynamically added and removed from the pool.

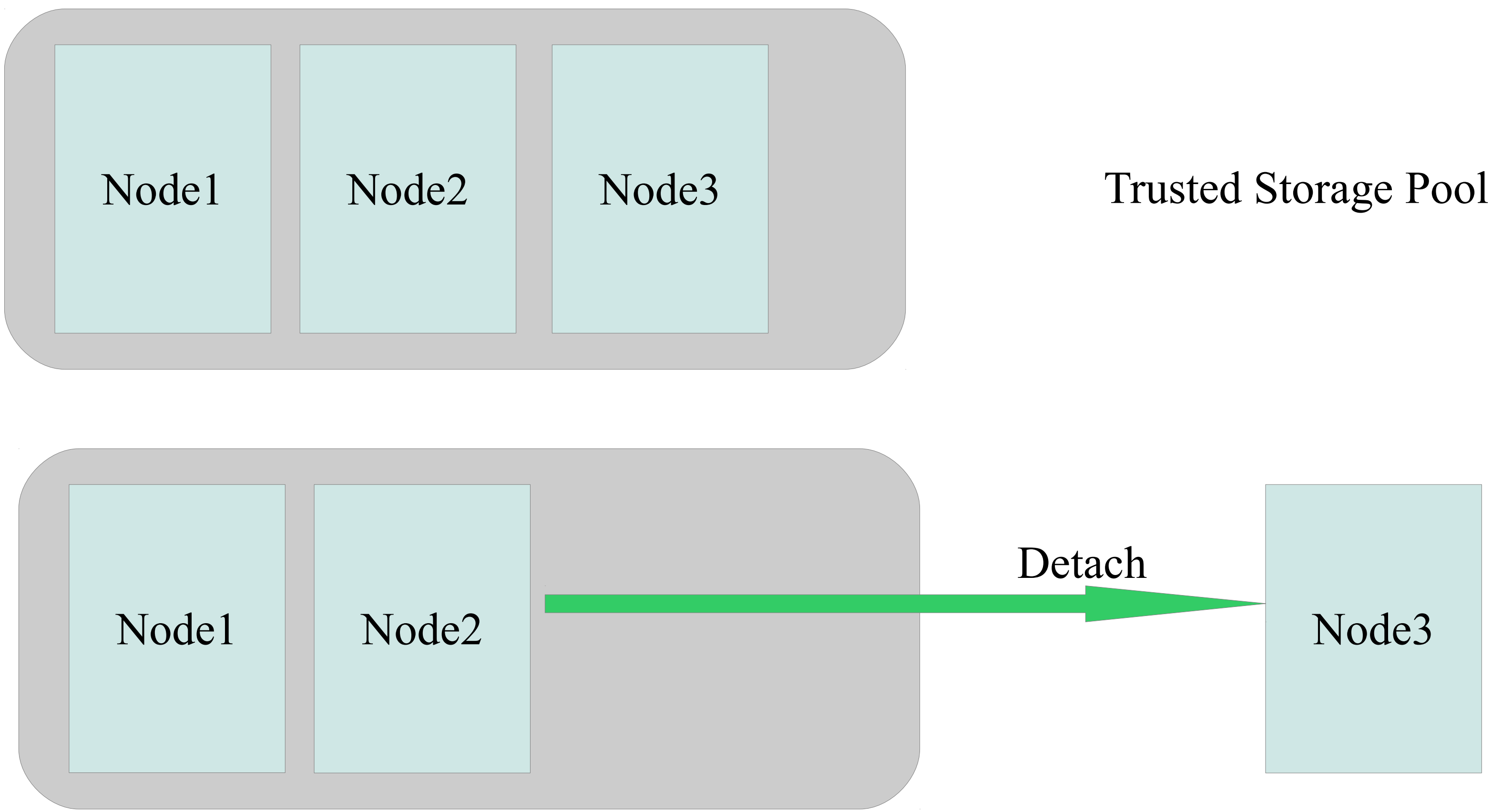
GlusterFS concepts – Trusted Storage Pool



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

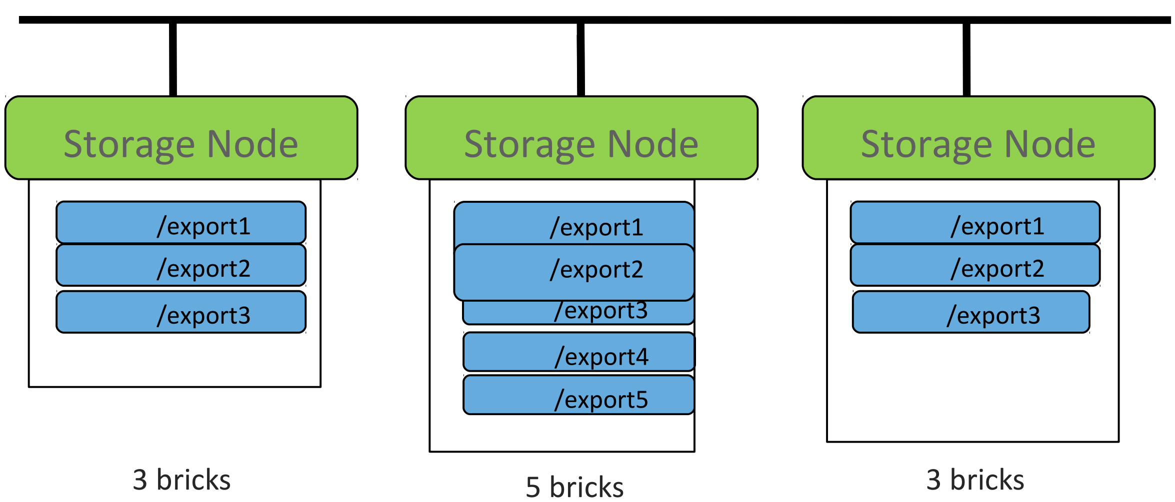


GlusterFS concepts – Trusted Storage Pool



GlusterFS concepts - Bricks

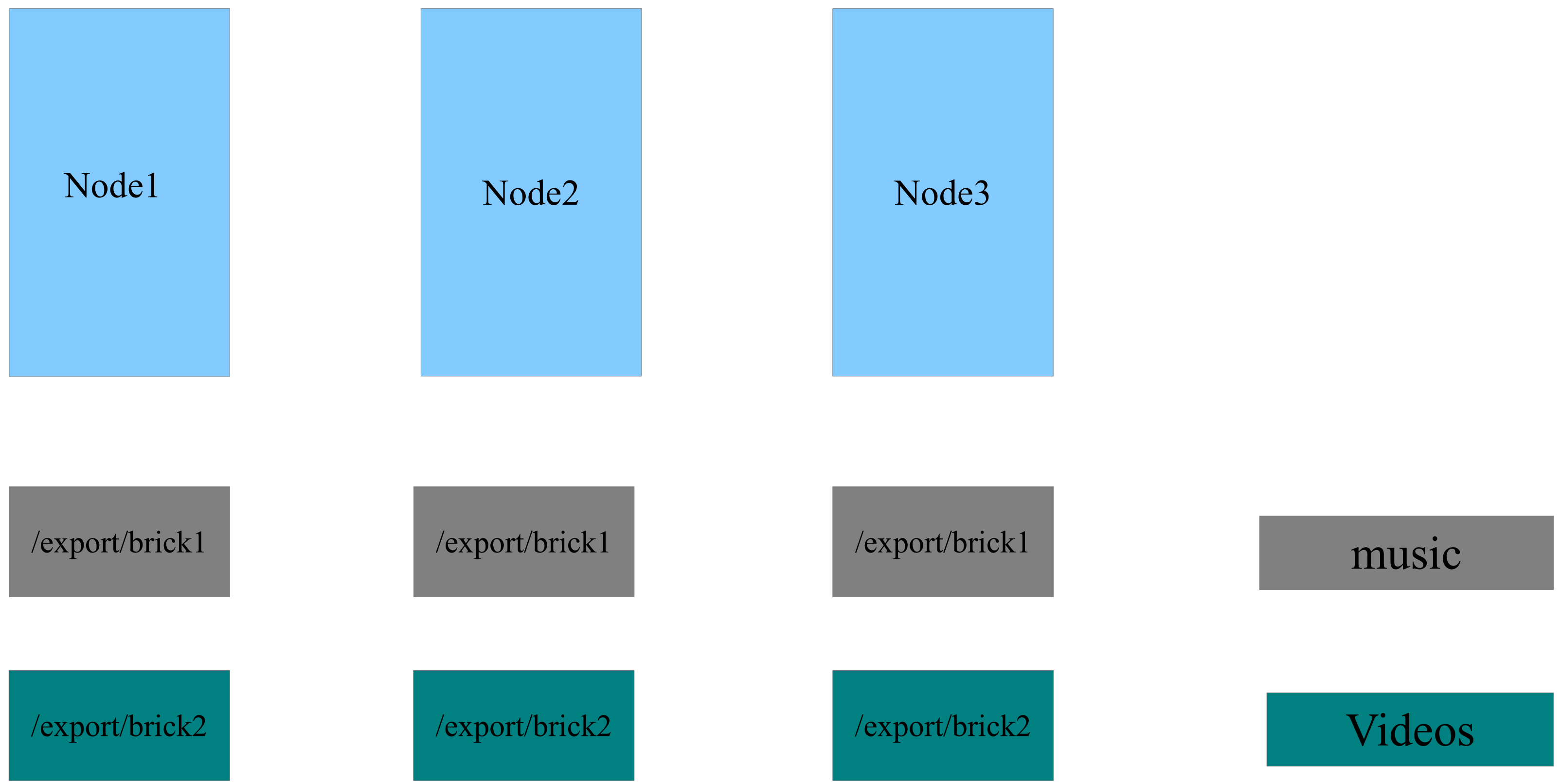
- A brick is the combination of a node and an export directory – for e.g. hostname:/dir
- Each brick inherits limits of the underlying filesystem
- No limit on the number bricks per node
- Ideally, each brick in a cluster should be of the same size



GlusterFS concepts - Volumes

- A volume is a logical collection of bricks.
- Volume is identified by an administrator provided name.
- Volume is a mountable entity and the volume name is provided at the time of mounting.
  - ***mount -t glusterfs server1:/<volname> /my/mnt/point***
- Bricks from the same node can be part of different volumes

GlusterFS concepts - Volumes

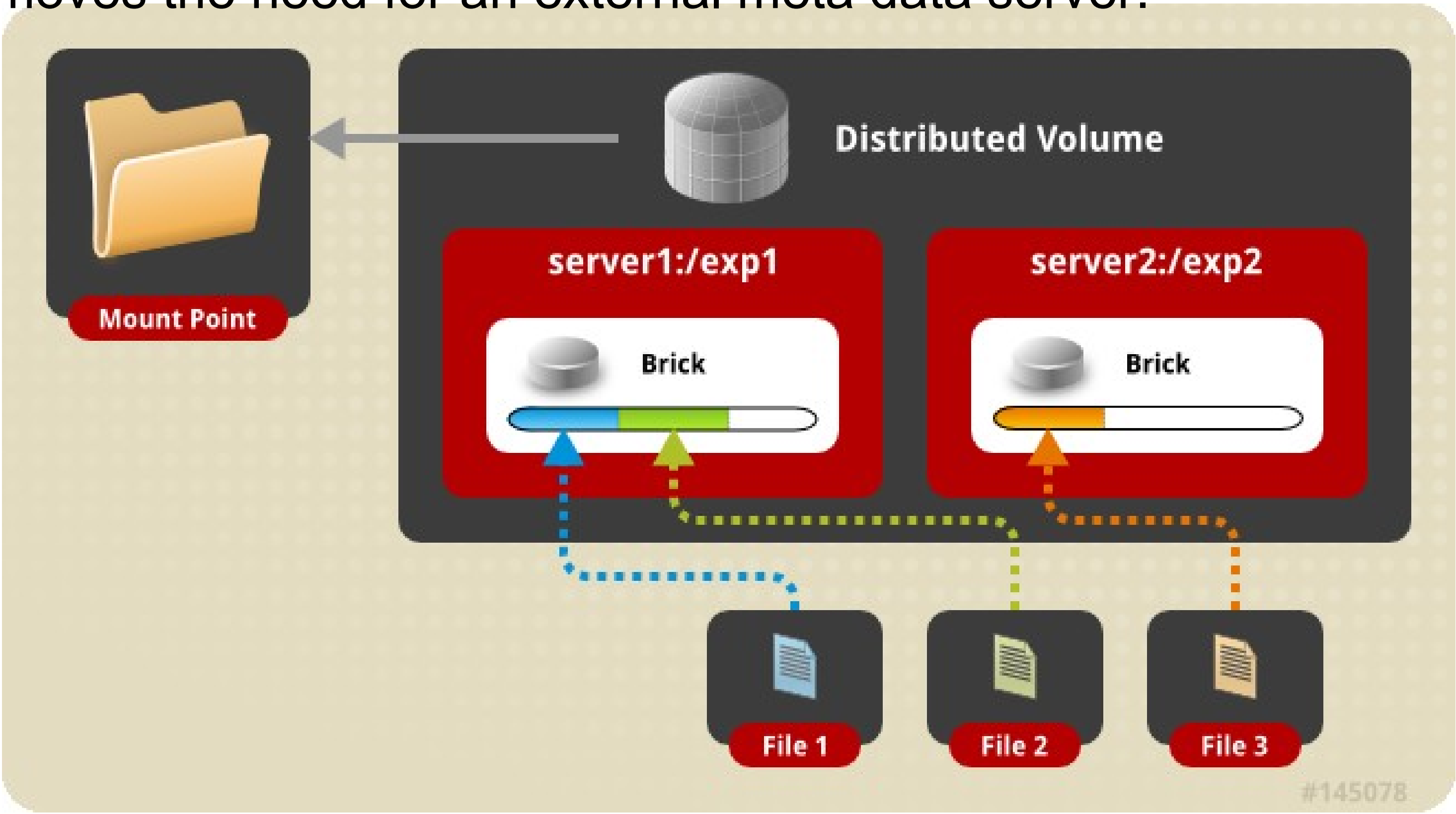


Volume Types

- 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

Distributed Volume

- Distributes files across various bricks of the volume.
- Directories are present on all bricks of the volume.
- Single brick failure will result in loss of data availability.
- Removes the need for an external meta data server.



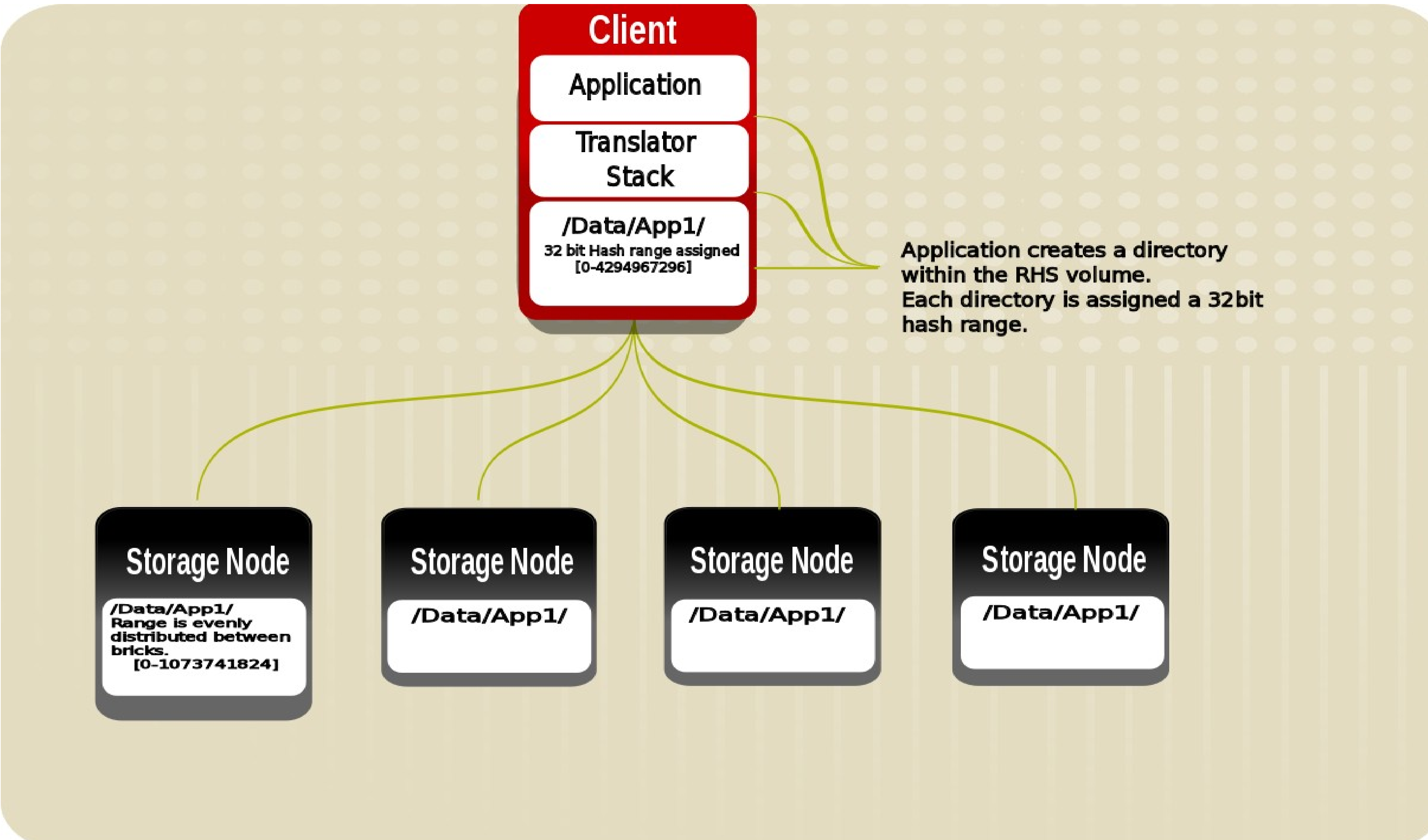
How does a distributed volume work?

- Uses Davies-Meyer hash algorithm.
- A 32-bit hash space is divided into N ranges for N bricks
- At the time of directory creation, a range is assigned to each directory.
- During a file creation or retrieval, hash is computed on the file name.

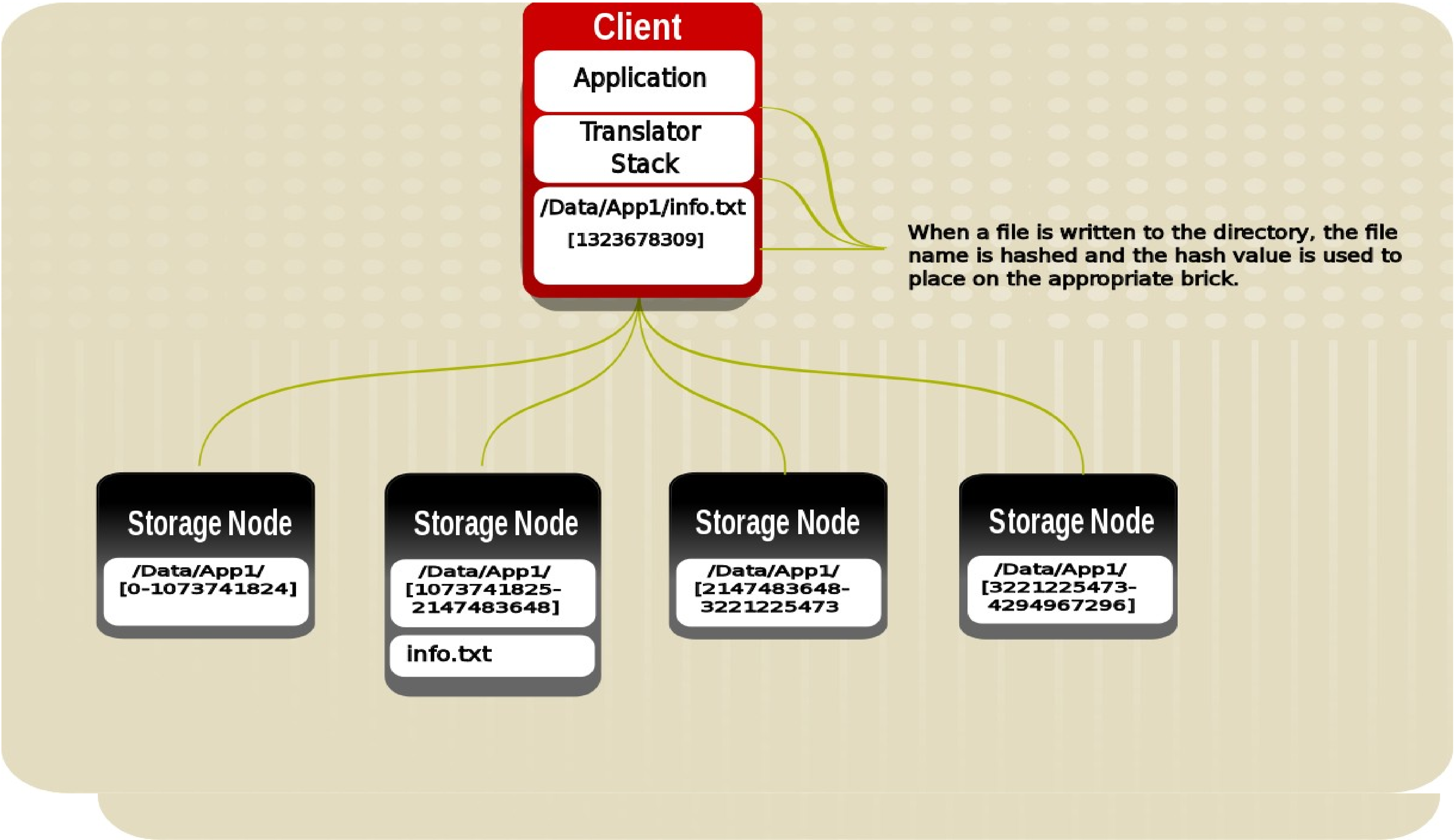
This hash value is used to locate or place the file.

- Different directories in the same brick end up with different hash ranges.

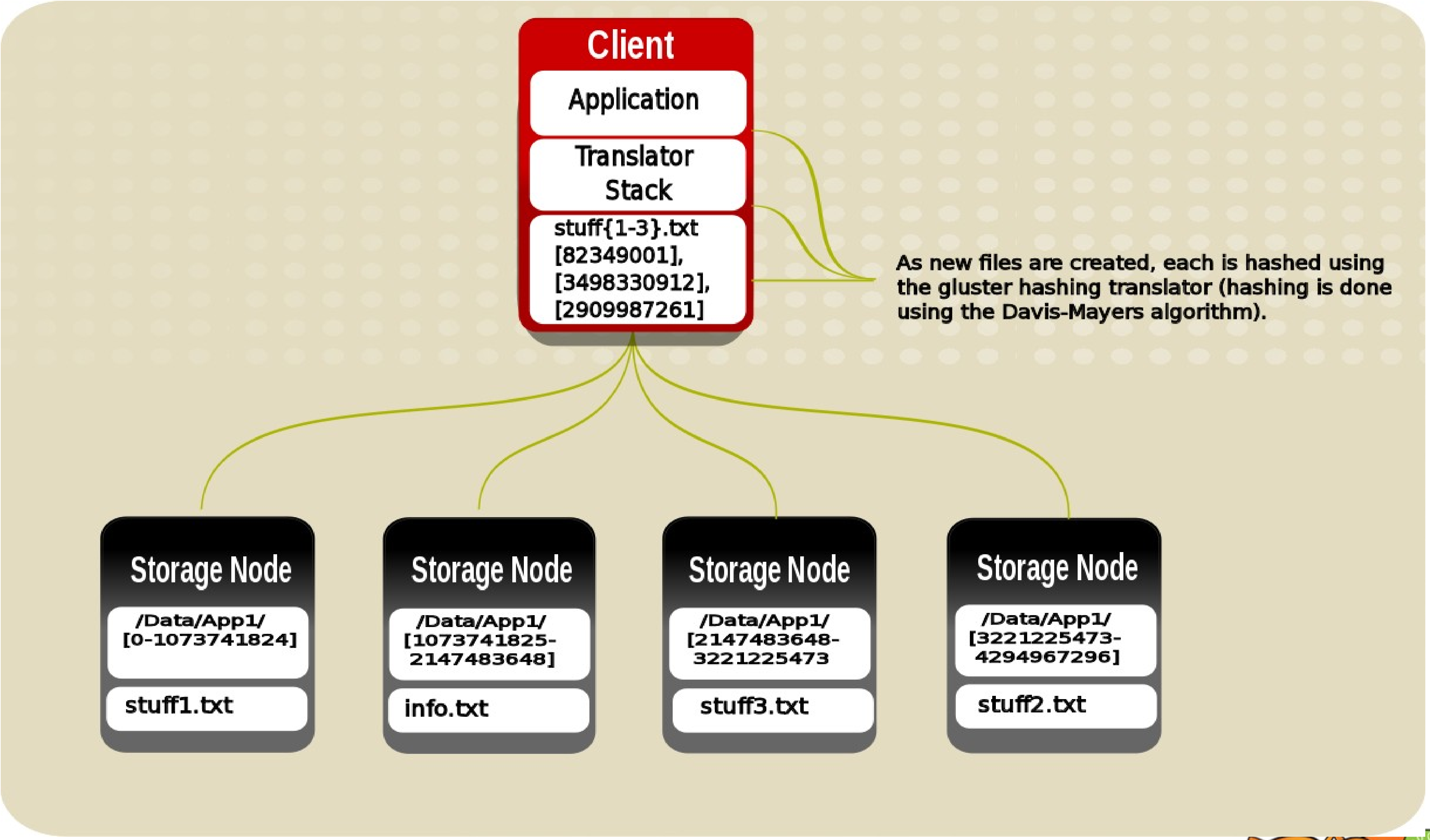
How does a distributed volume work?



How does a distributed volume work?



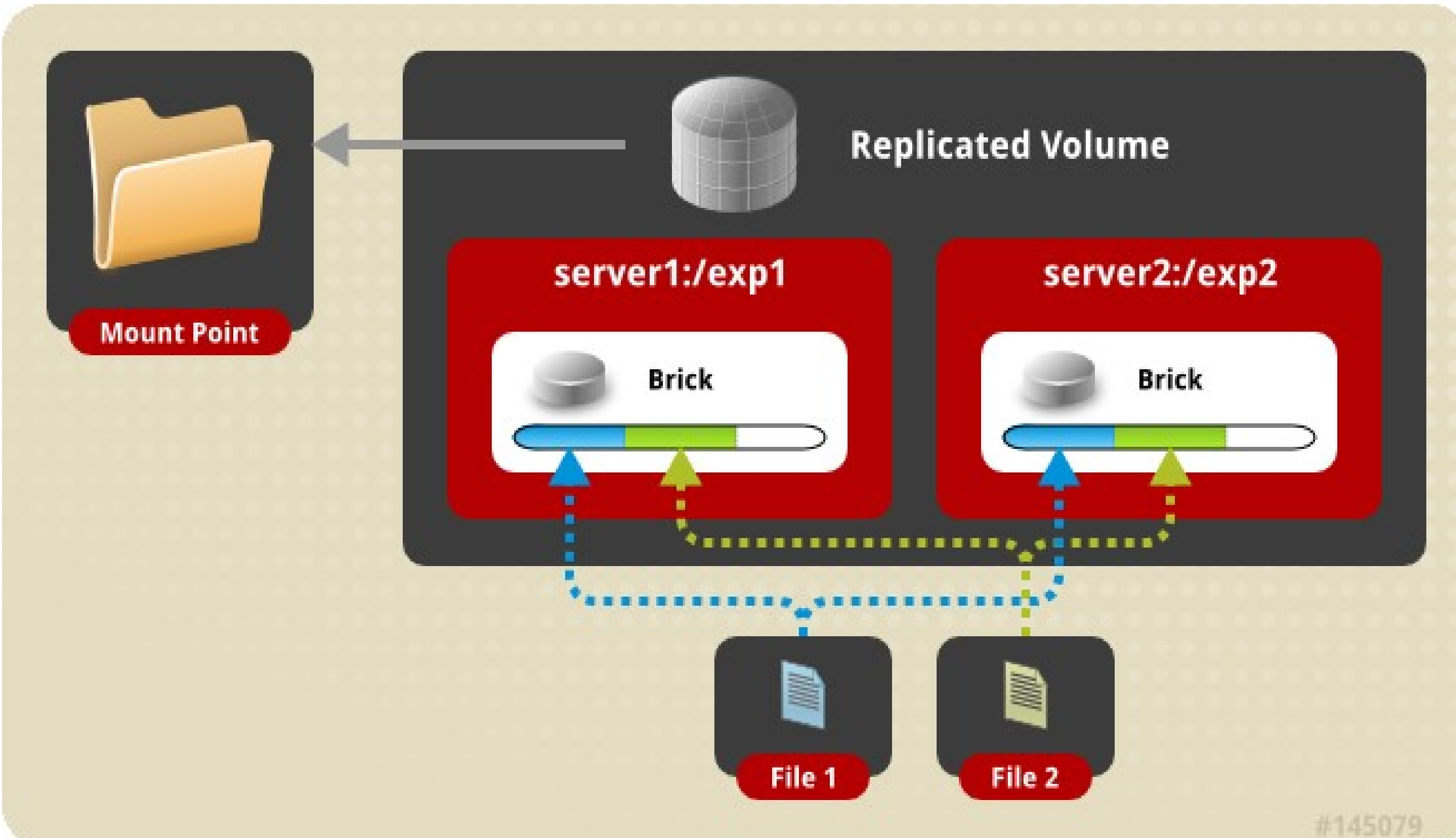
How does a distributed volume work?



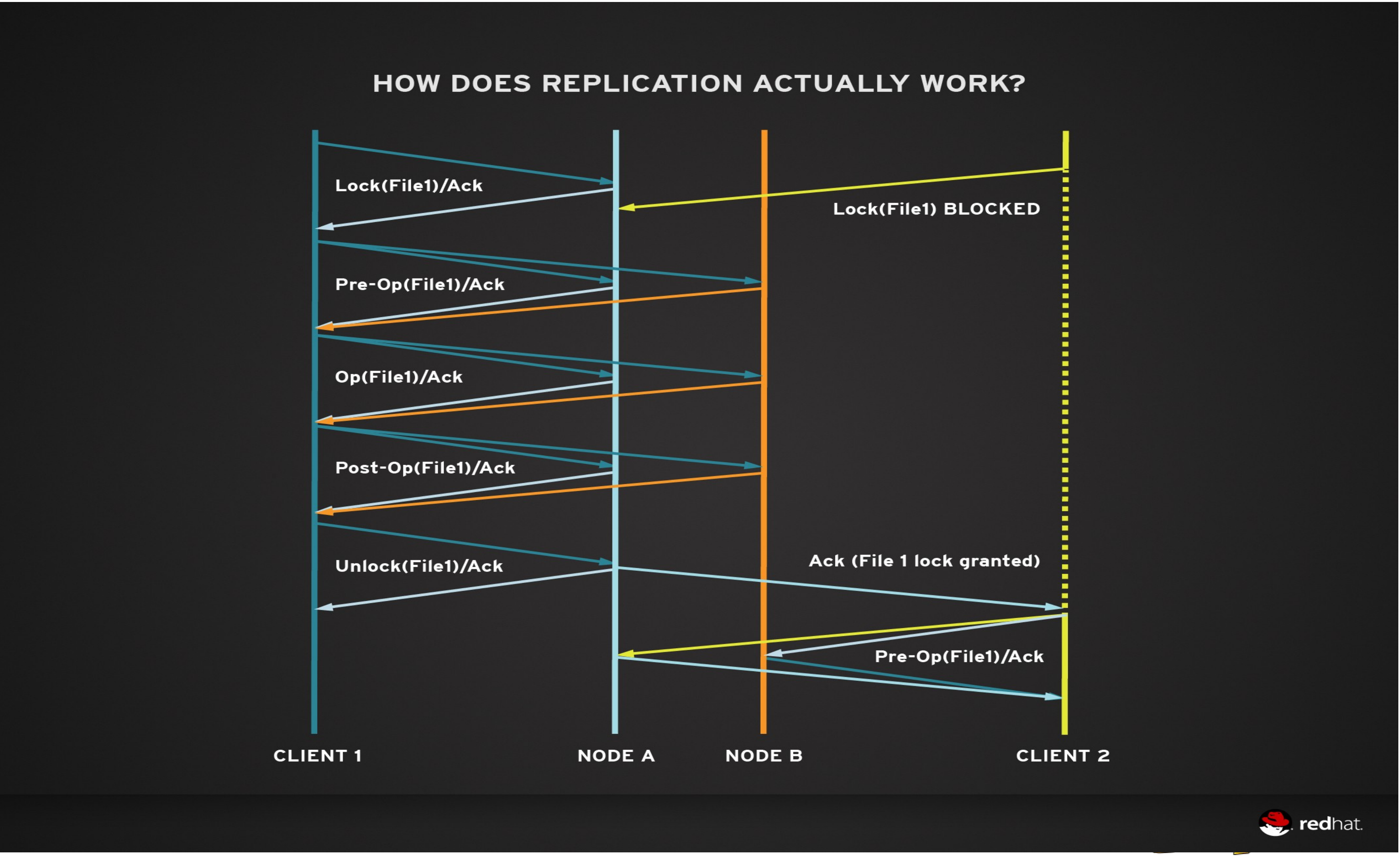
Replicated Volume

- Synchronous replication of all directory and file updates.
- Provides high availability of data when node failures occur.
- Transaction driven for ensuring consistency.
- Changelogs maintained for re-conciliation.
- Any number of replicas can be configured.

How does a replicated volume work?



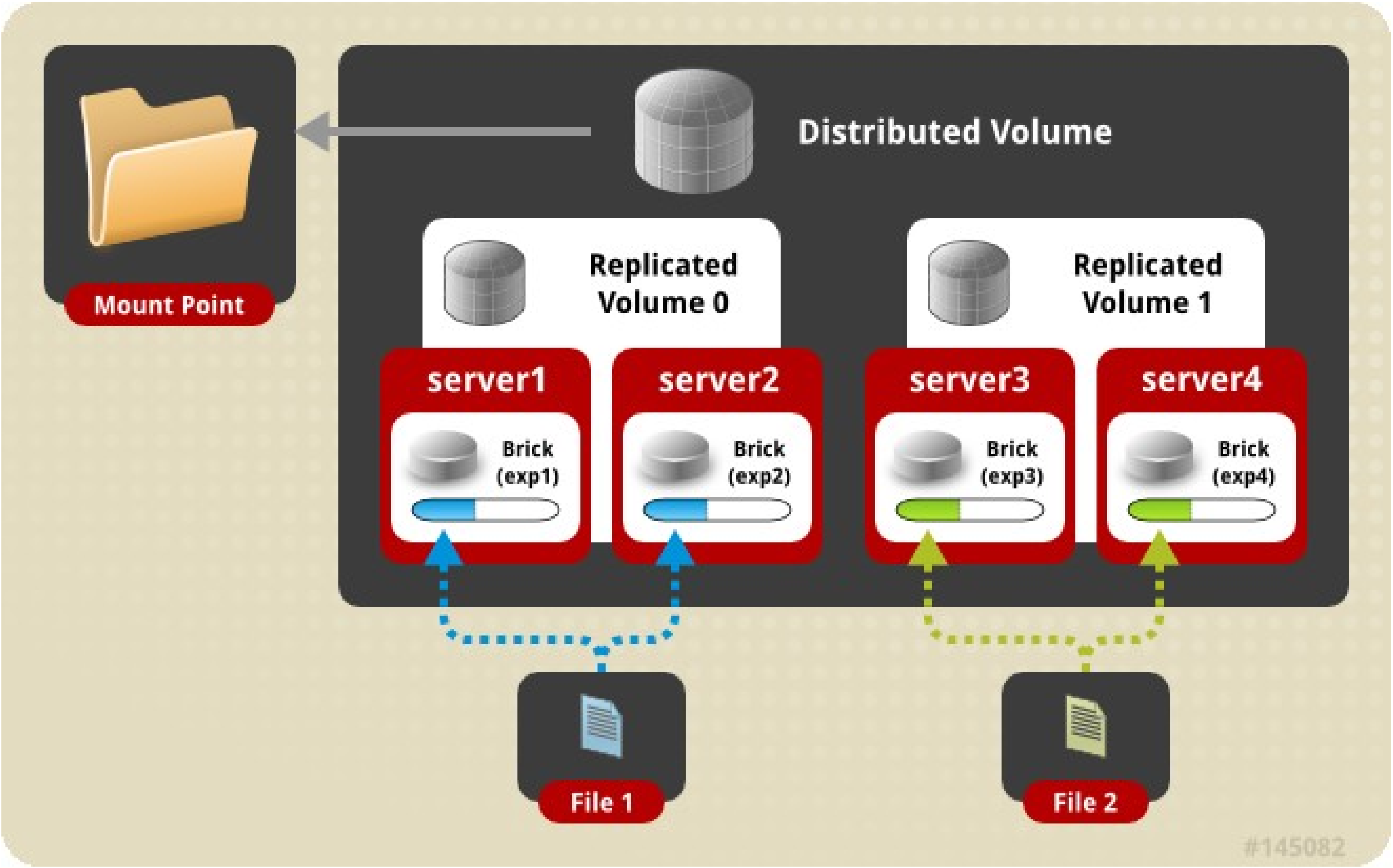
How does a replicated volume work?

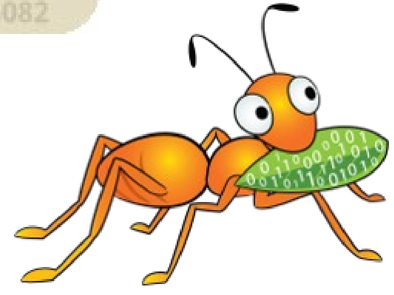


Distributed Replicated Volume

- Distribute files across replicated bricks
  - Number of bricks must be a multiple of the replica count
  - Ordering of bricks in volume definition matters
- Scaling and high availability
- Reads get load balanced.
- Most preferred model of deployment currently.

Distributed Replicated Volume





Striped Volume

- Files are striped into chunks and placed in various bricks.
- Recommended only when very large files greater than the size of the disks are present.
- Chunks are files with holes – this helps in maintaining offset consistency.
- A brick failure can result in data loss. Redundancy with replication is highly recommended (striped replicated volumes).

Elastic Volume Management

Application transparent operations that can be performed in the storage layer.

- Addition of Bricks to a volume
- Remove brick from a volume
- Rebalance data spread within a volume
- Replace a brick in a volume
- Performance / Functionality tuning

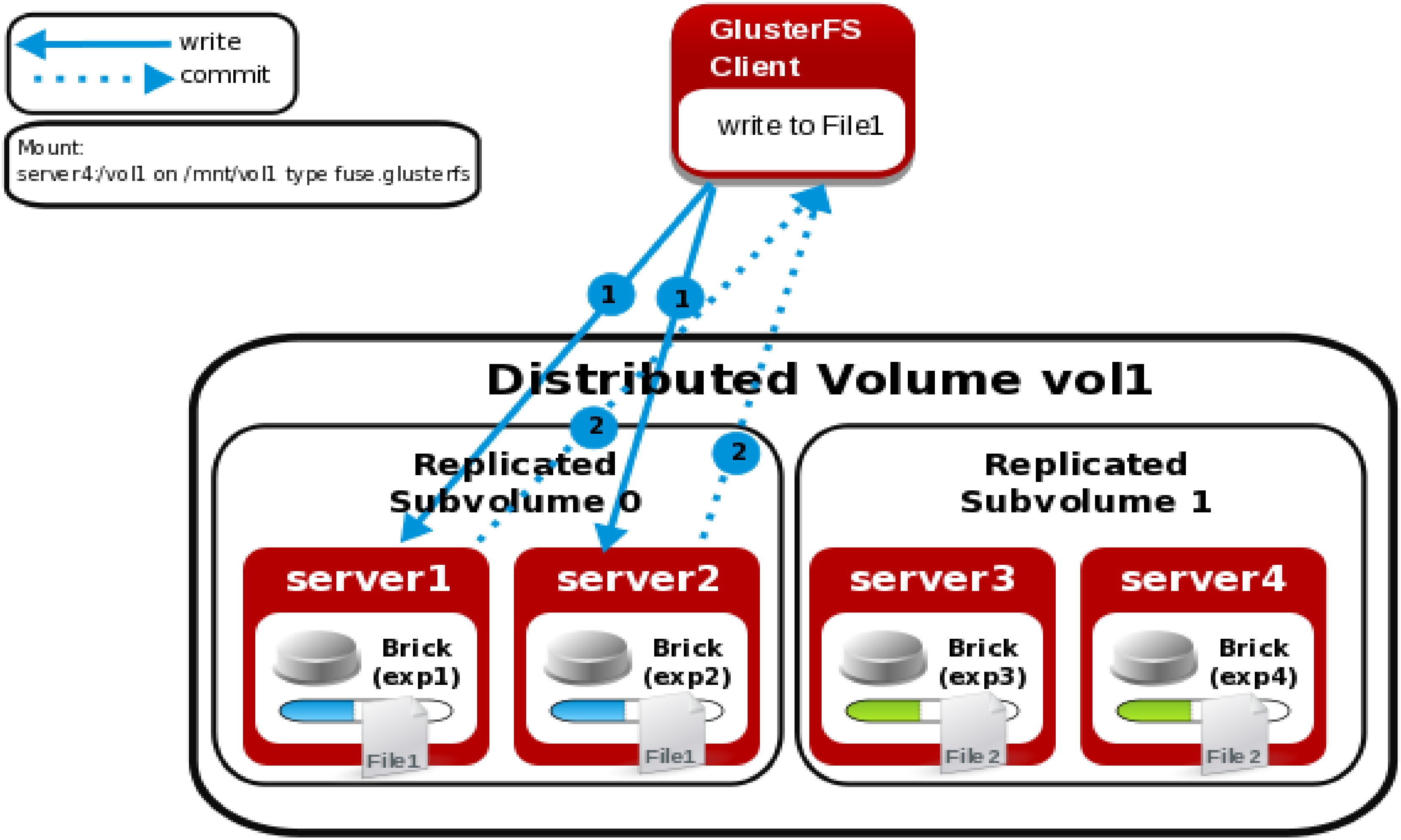
Access Mechanisms

Access Mechanisms

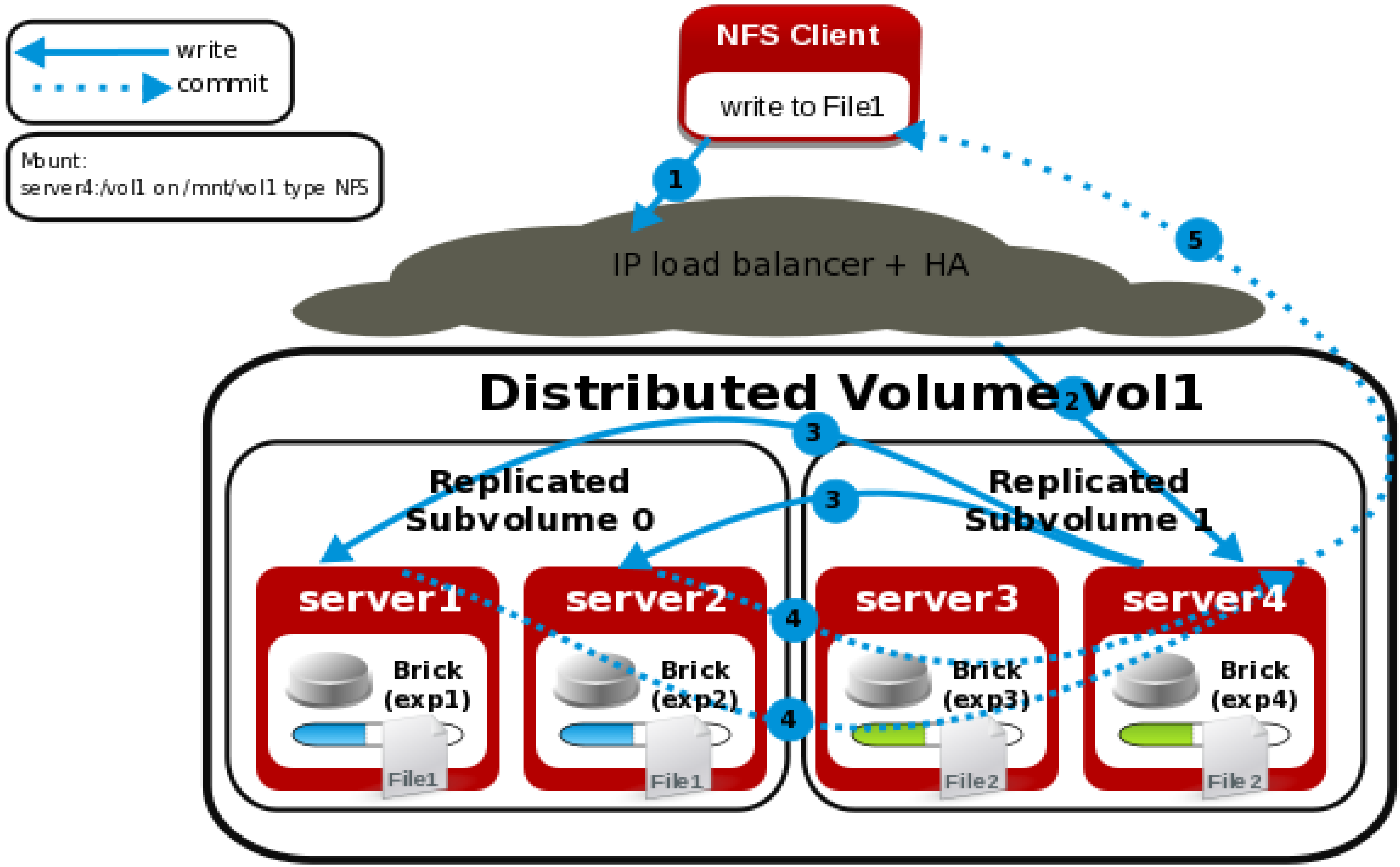
Gluster volumes can be accessed via the following mechanisms:

- FUSE based Native protocol
- NFSv3
- SMB
- libgfapi
- ReST/HTTP
- HDFS

FUSE based native access



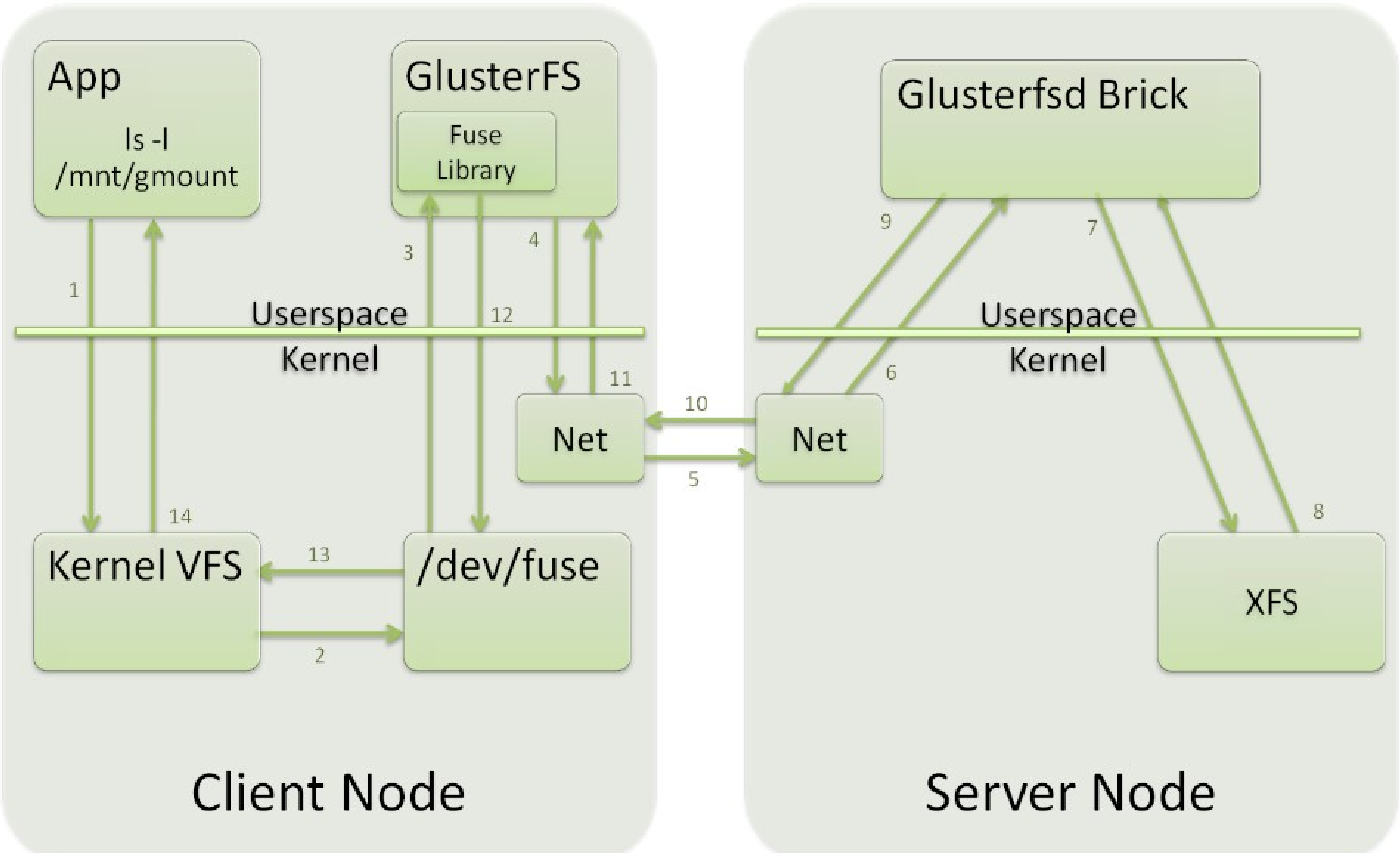
NFS access


write
commit
Mount:
server4:/vol1 on /mnt/vol1 type NFS
NFS Client
write to File1
1
5
IP load balancer + HA
Distributed Volume vol1
2
3
3
Replicated Subvolume 0
Replicated Subvolume 1
server1
server2
server3
server4
4
4
Brick (exp1)
Brick (exp2)
Brick (exp3)
Brick (exp4)
File1
File1
File2
File 2

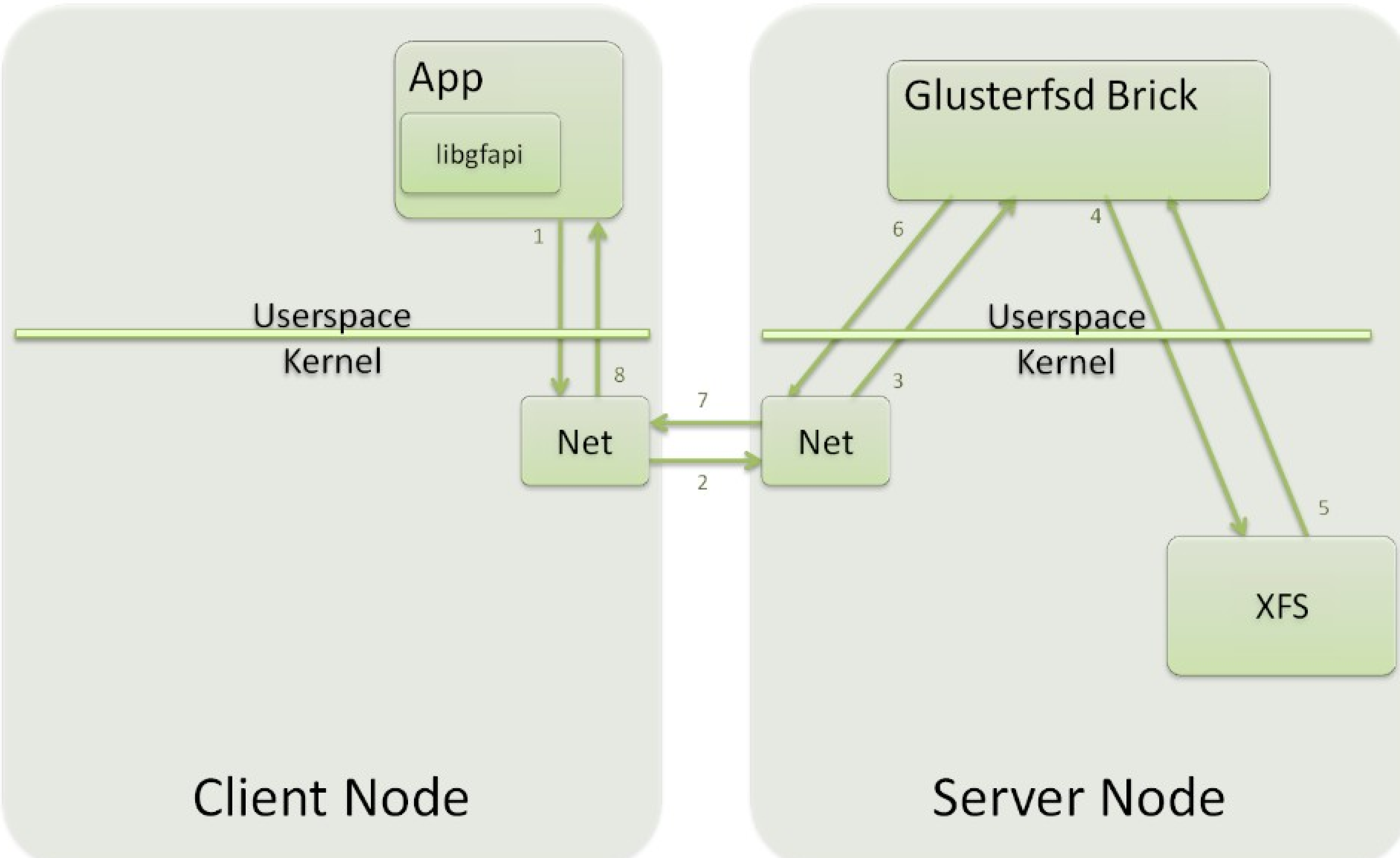
libgfapi

- Exposes APIs for accessing Gluster volumes.
- Reduces context switches.
- qemu, samba, NFS Ganesha integrated with libgfapi.
- Both sync and async interfaces available.
- Emerging bindings for various languages.

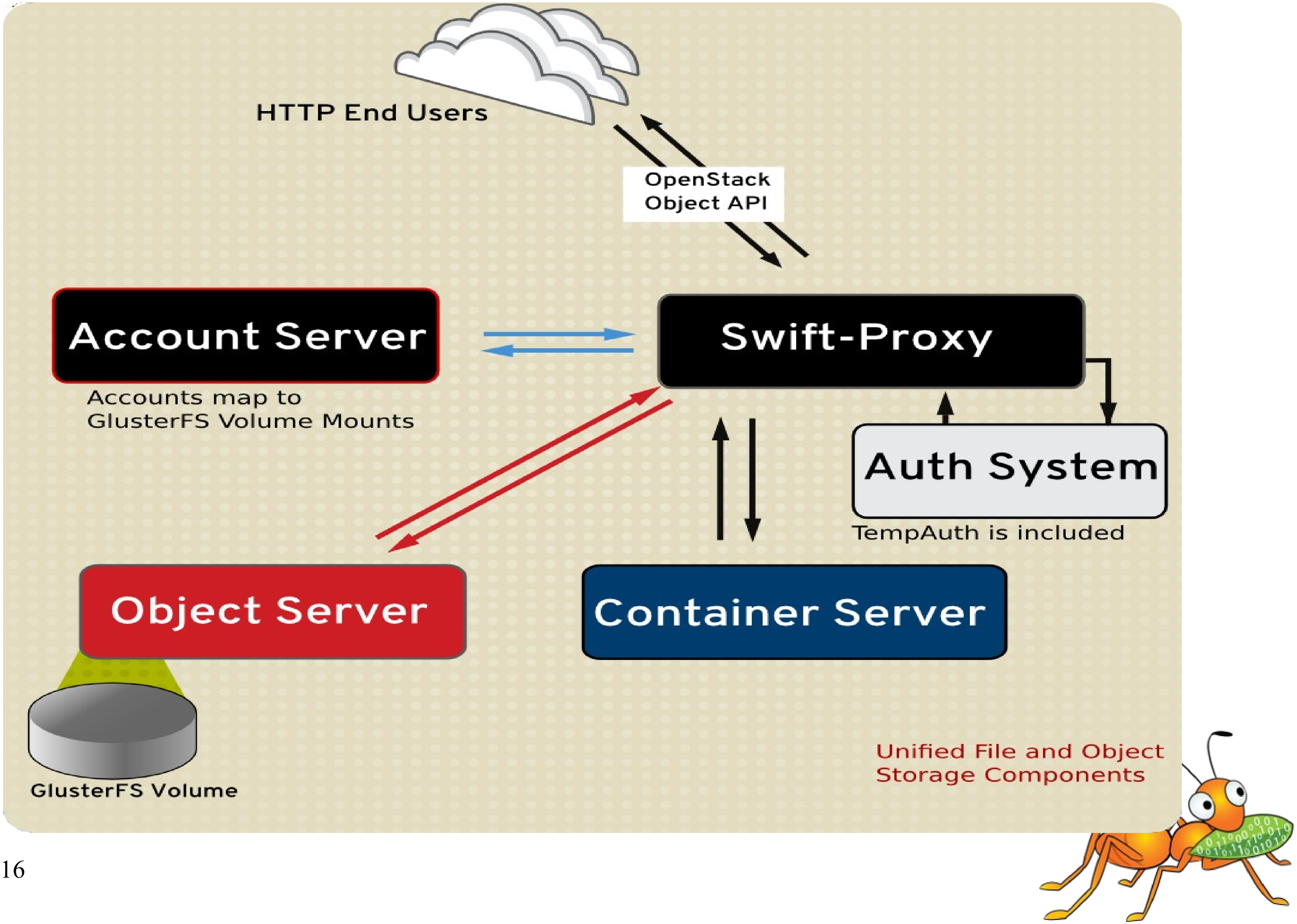
libgfapi v/s FUSE – FUSE access



libgfapi v/s FUSE – libgfapi access



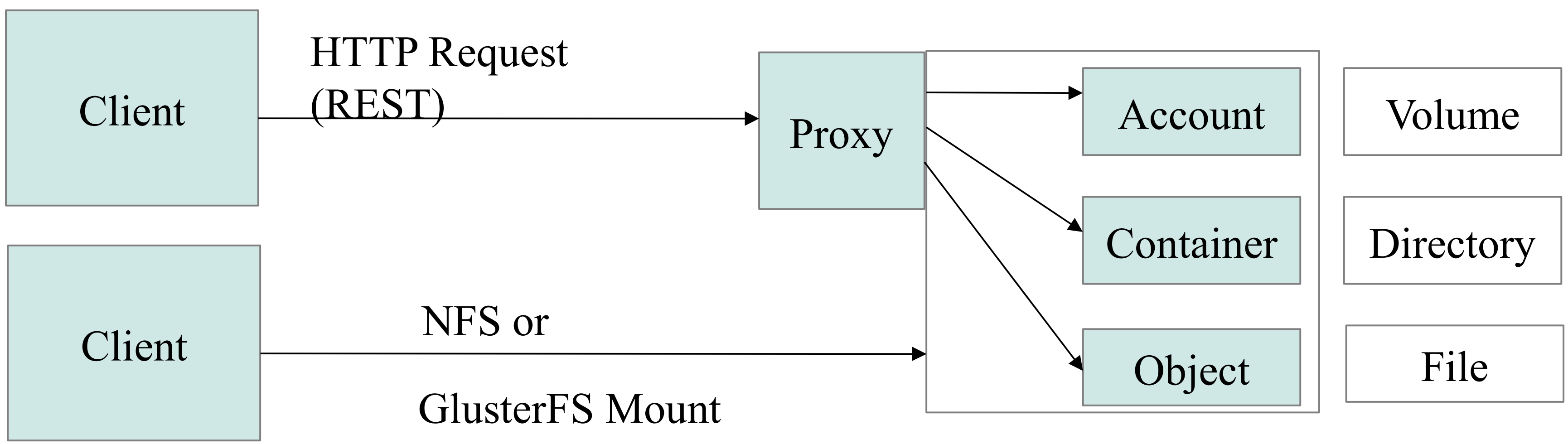
ReST based access



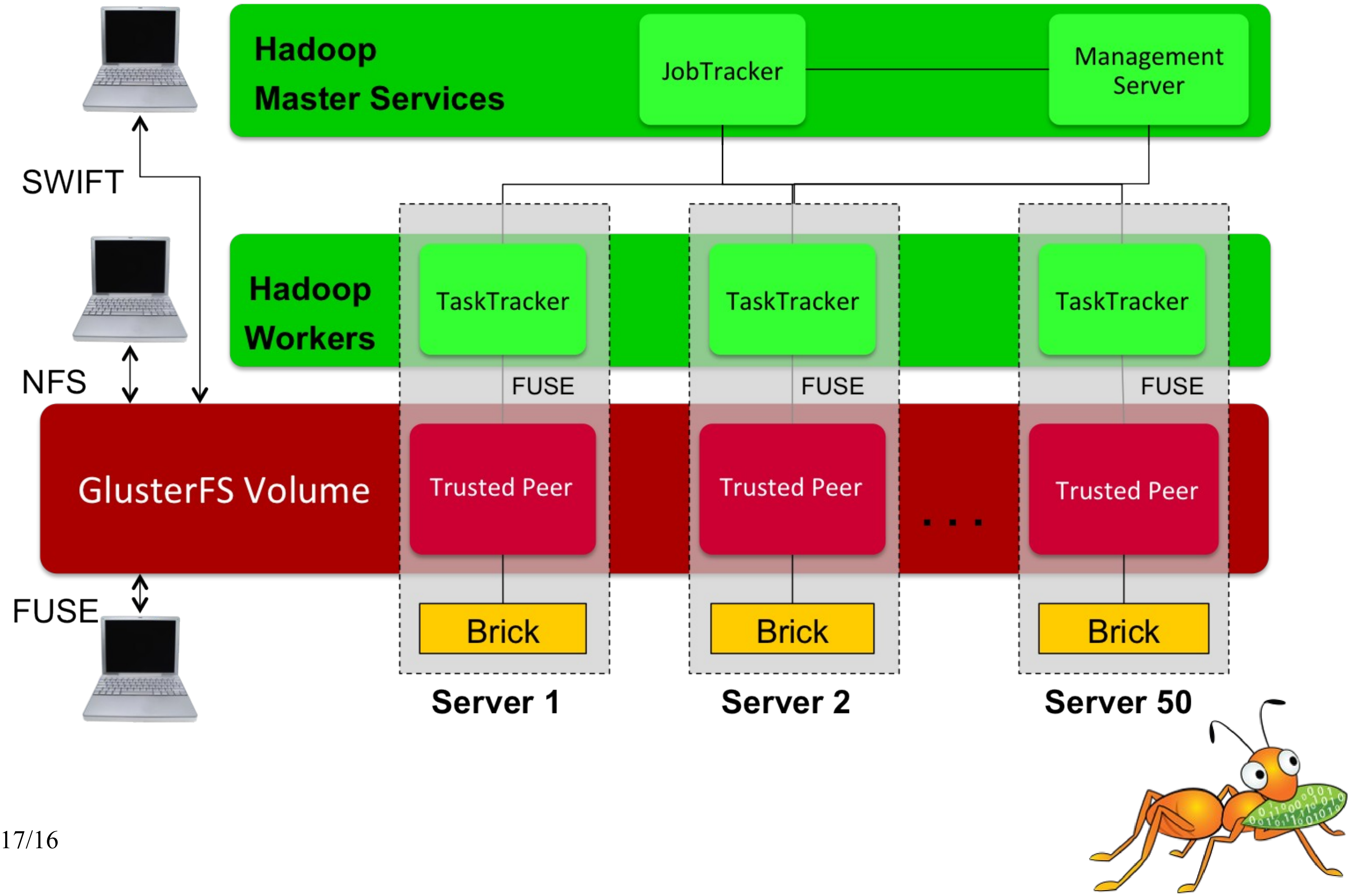
ReST - G4S

Unified File and object view.

Entity mapping between file and object building blocks



Hadoop access

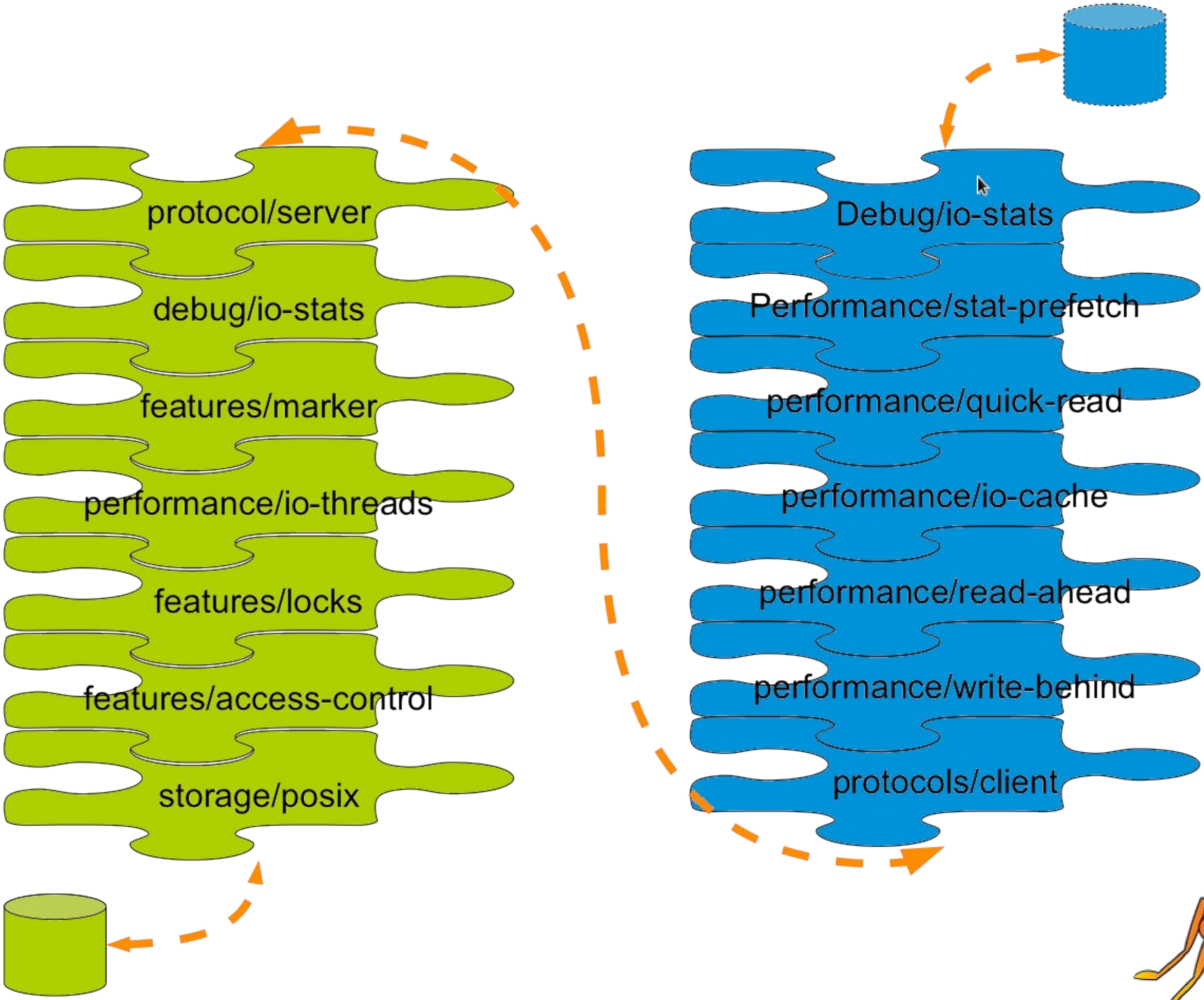


Implementation

Translators in GlusterFS

- Building blocks for a GlusterFS process.
- Based on Translators in GNU HURD.
- Each translator is a functional unit.
- Translators can be stacked together for achieving desired functionality.
- Translators are deployment agnostic – can be loaded in either the client or server stacks.

Customizable Translator Stack

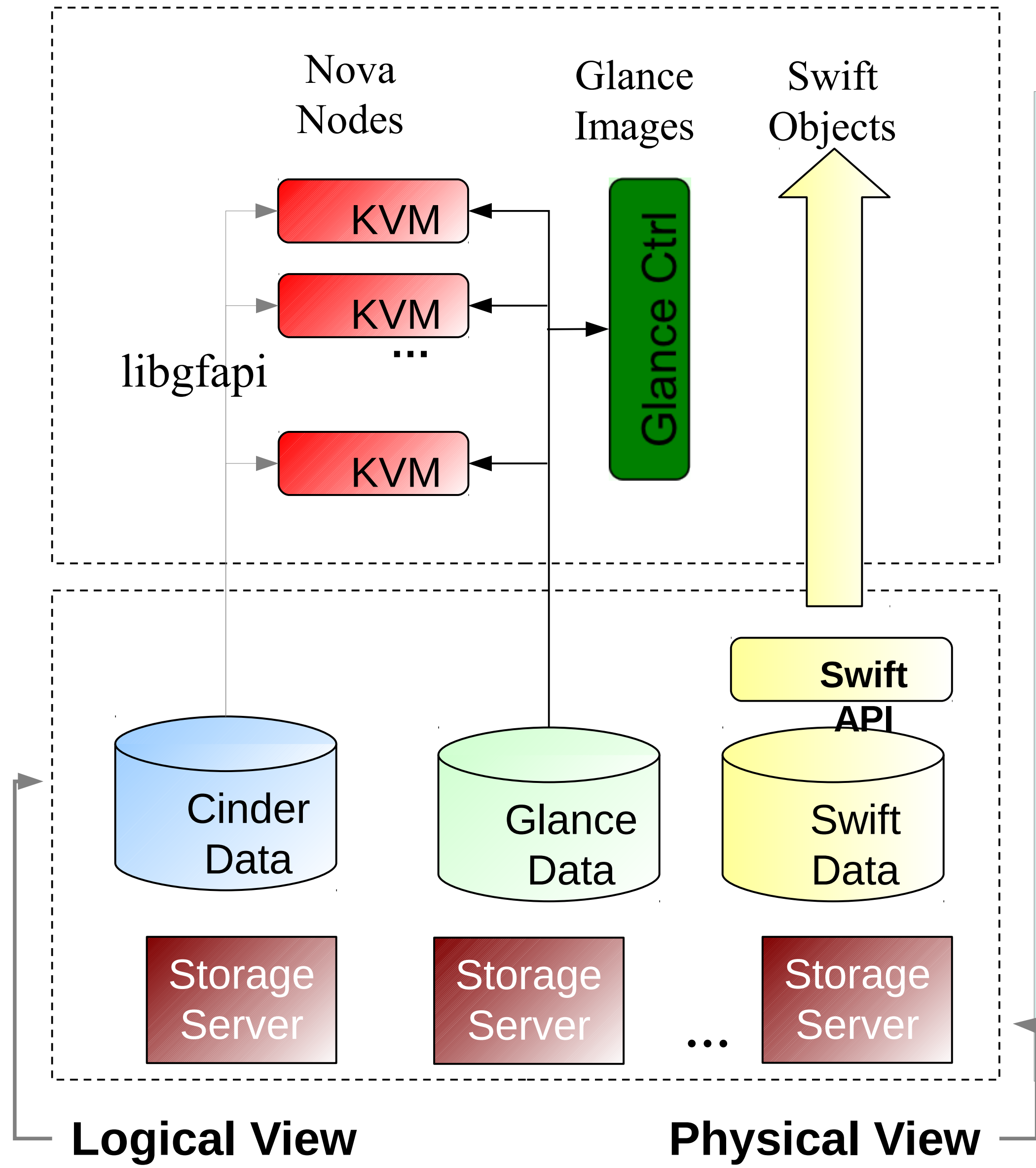


Ecosystem Integration

Ecosystem Integration

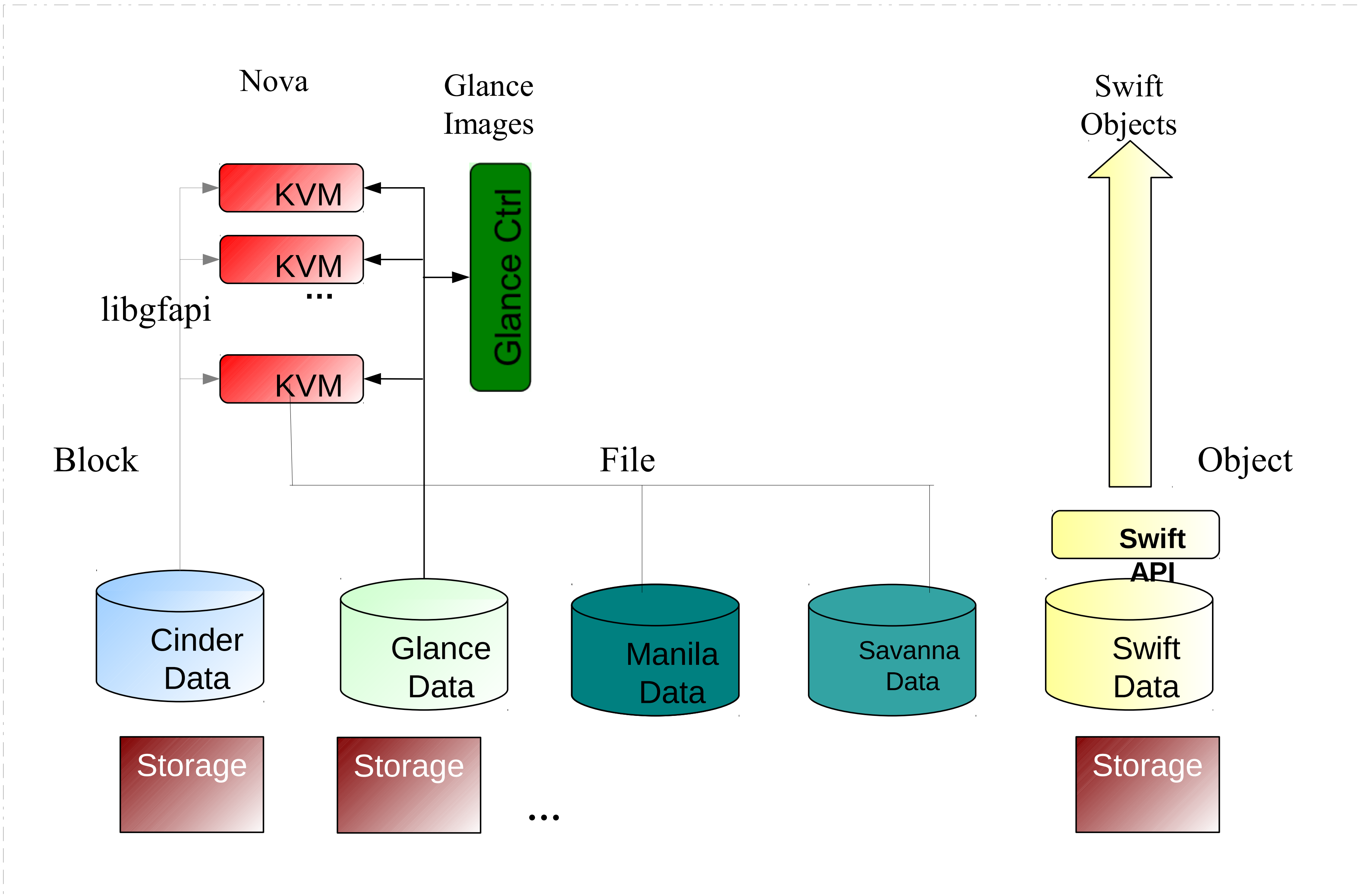
- Currently integrated with various ecosystems:
  - OpenStack
  - Samba
  - Ganesha
  - oVirt
  - qemu
  - Hadoop
  - pcp
  - Proxmox
  - uWSGI

OpenStack Havana and GlusterFS – Current Integration



- Separate Compute and Storage Pools
- GlusterFS directly provides Swift object service
  - Integration with Keystone
  - GeoReplication for multi-site support
- Swift data also available via other protocols
- Supports non-OpenStack use in addition to OpenStack use

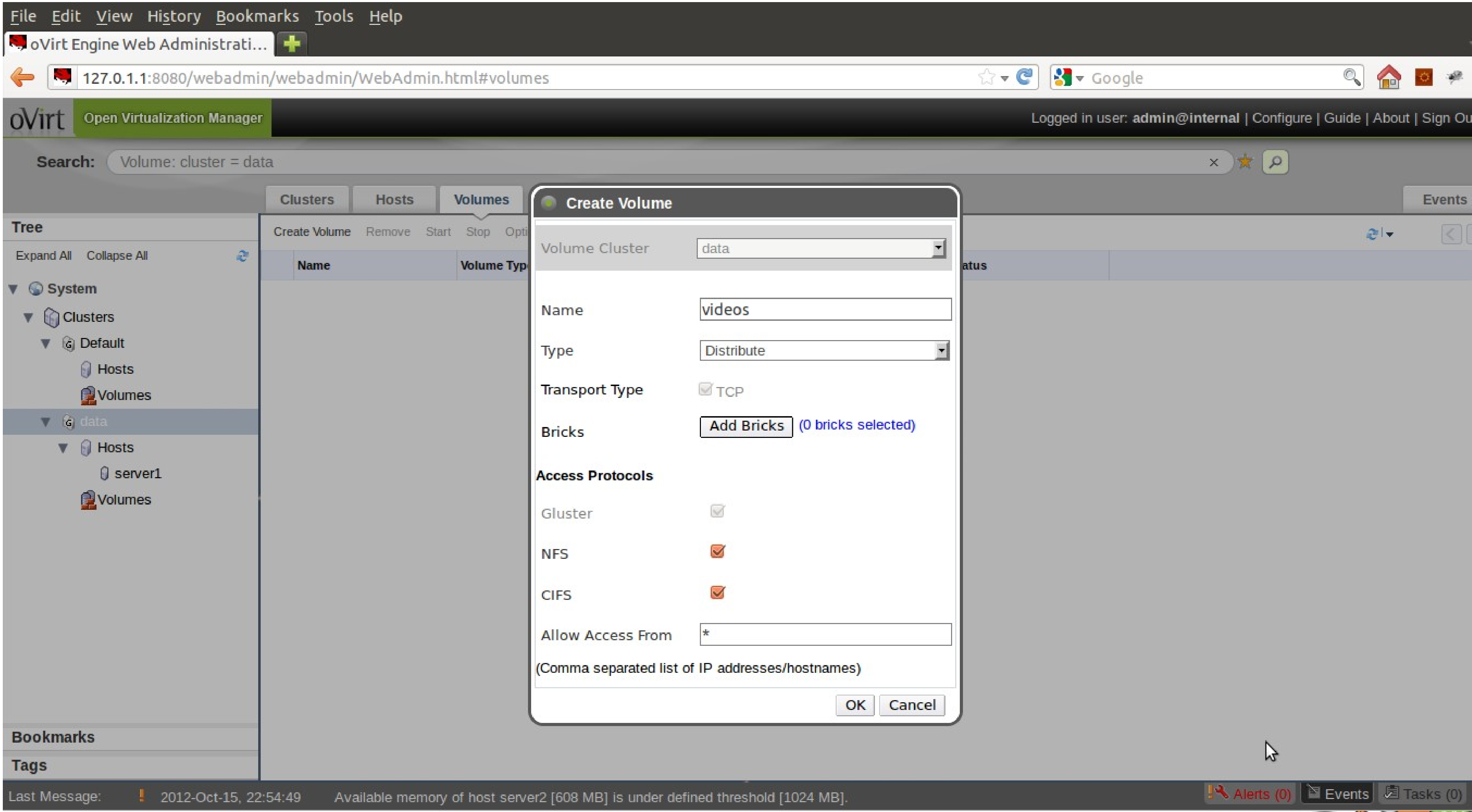
OpenStack and GlusterFS – Future Integration



GlusterFS & oVirt

- Trusted Storage Pool and Gluster Volume management - oVirt 3.1
- FUSE based posixFS support for VM image storage - oVirt 3.1
- libgfapi based Gluster native storage domain - oVirt 3.3
- Manage converged virtualization and storage clusters in oVirt
- ReST APIs & SDK for GlusterFS management.

GlusterFS & oVirt



Use Cases - current

- Unstructured data storage
- Archival
- Disaster Recovery
- Virtual Machine Image Store
- Cloud Storage for Service Providers
- Content Cloud
- Big Data
- Semi-structured & Structured data

Future Directions

New Features in GlusterFS 3.5

- Distributed geo-replication
- File snapshots
- Compression translator
- Multi-brick Block Device volumes
- Readdir ahead translator
- Quota Scalability

Beta Features in GlusterFS 3.5

- Disperse translator for Erasure Coding
- Encryption at rest
- Support for bricks on Btrfs
- libgfapi support for NFS Ganesha (NFS v4)

Geo-replication in 3.5

- Before 3.5
  - Merkle tree based optimal volume crawling
  - Single driver on the master
  - SPOF
- In 3.5
  - Based on changelog
  - One driver per replica set on the master
  - No SPOF

Quota in 3.5

- Before 3.5
  - Client side enforcement
  - Configuration in volume files would block scalability
  - GFID accesses could cause incorrect accounting
  - Only hard quota supported
- In 3.5
  - Server side enforcement
  - Better configuration management for scalability.
  - GFID to path conversion enables correct accounting.
  - Both hard and soft quotas supported

Prominent Features beyond GlusterFS 3.5

- Volume snapshots
- New Style Replication
- pNFS access with NFS Ganesha
- Data tiering / HSM
- Multi master geo-replication
- Support Btrfs features
- Caching improvements
- libgfchangelog
- and more...

Challenges

- Scalability – 1024 nodes, 72 brontobytes?
- Hard links
- Rename
- Monolithic tools
- Monitoring
- Reduce Capex and Opex

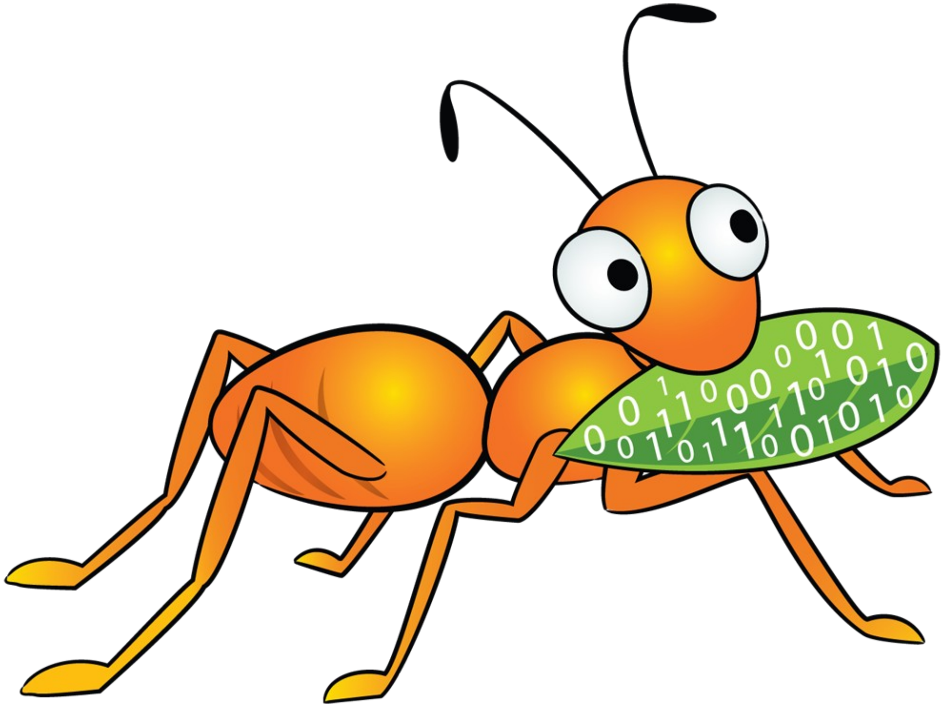
Resources

Mailing lists:
gluster-users@gluster.org
gluster-devel@nongnu.org

IRC:
#gluster and #gluster-dev on freenode

Links:
http://www.gluster.org
http://hekafs.org
http://forge.gluster.org
http://www.gluster.org/community/documentation/index.php/Arch

Thank you!



Vijay Bellur
vbellur at redhat dot com