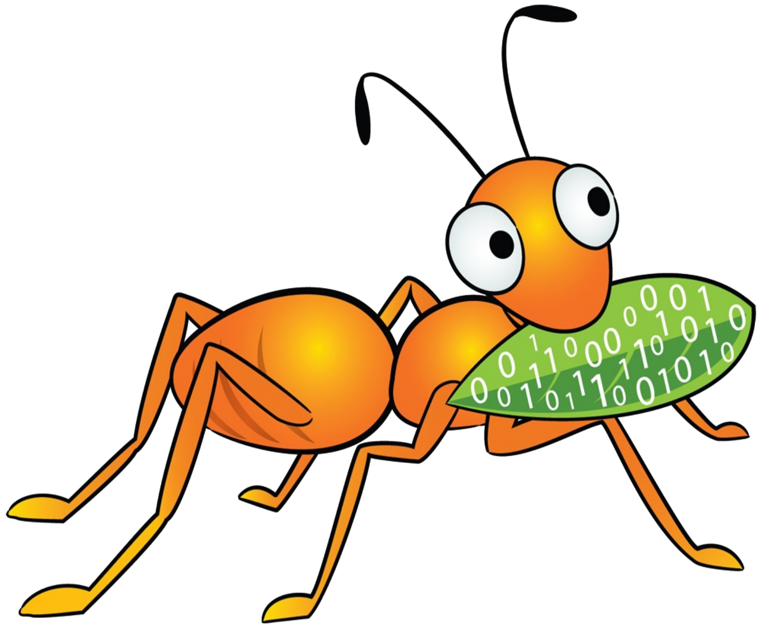
Replication Techniques in Gluster

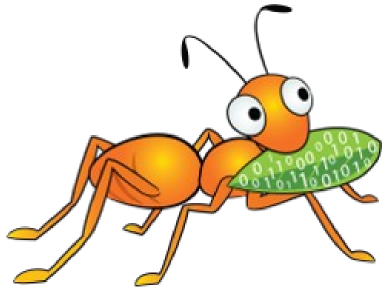


Niels de Vos
GlusterFS co-maintainer

ndevos@redhat.com
ndevos on IRC
@nixpanic on Twitter

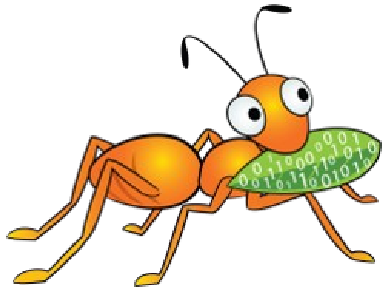
Agenda

- Basic Gluster introduction
- Synchronous vs asynchronous
  - Geo-replication
- Client-side vs Server-side
  - Automatic File Replication
  - Disperse / Erasure Coding
  - Journal Based Replication



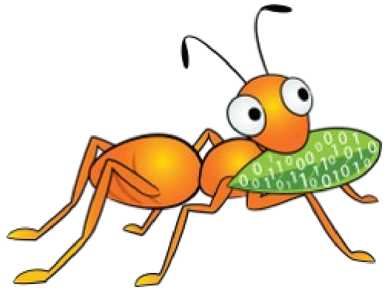
What is GlusterFS?

- Scalable, general-purpose storage platform
  - POSIX-y Distributed File System
  - Object storage (swift)
  - Distributed block storage (qemu)
  - Flexible storage (libgfapi)
- No Metadata Server
- Heterogeneous Commodity Hardware
- Flexible and Agile Scaling
  - Capacity – Petabytes and beyond
  - Performance – Thousands of Clients



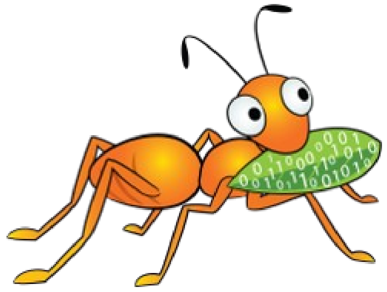
Terminology

- Brick
  - Fundamentally, a filesystem mountpoint
  - A unit of storage used as a ***capacity*** building block
- Translator
  - Logic between the file bits and the Global Namespace
  - Layered to provide GlusterFS ***functionality***



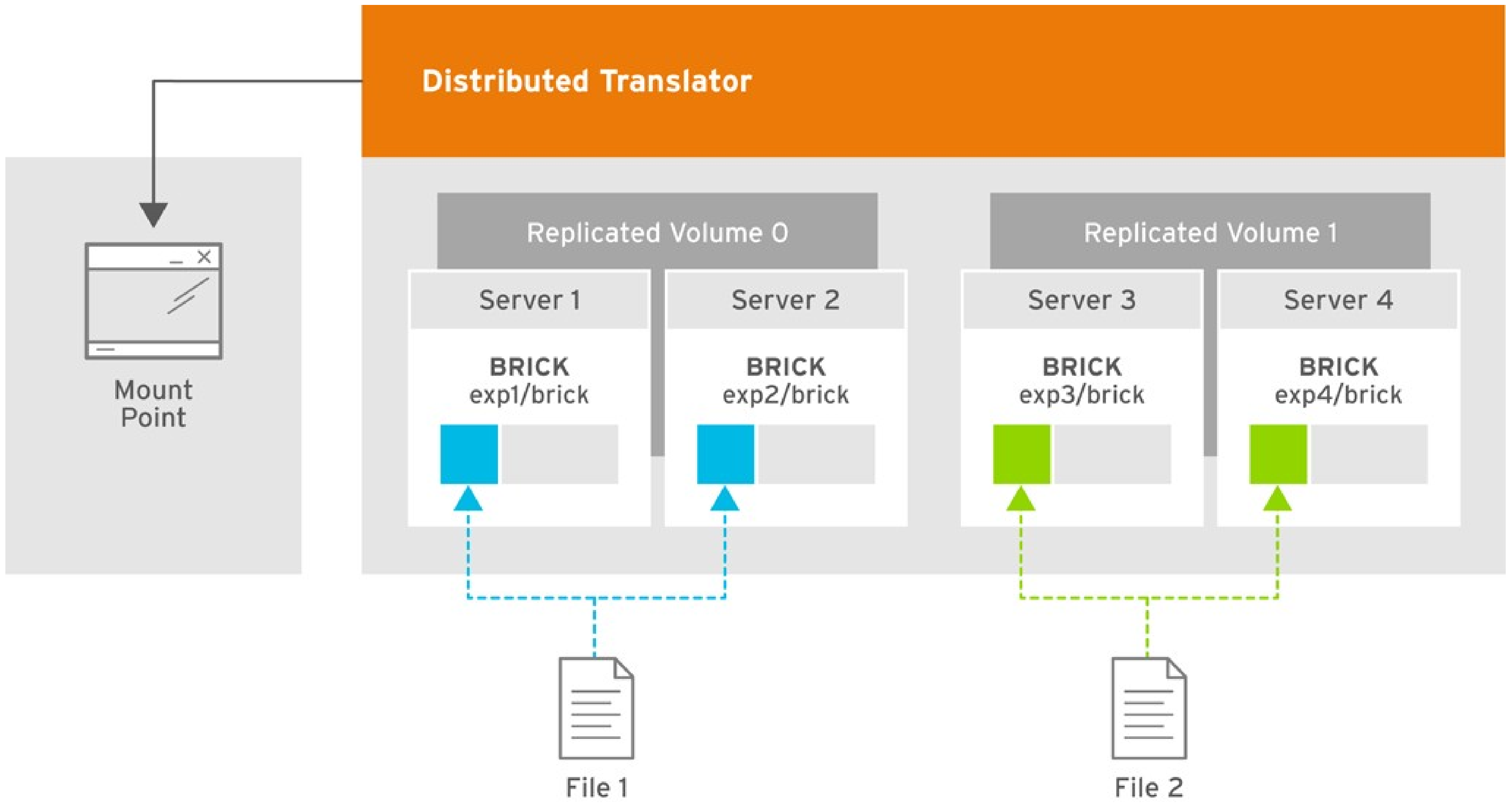
Terminology

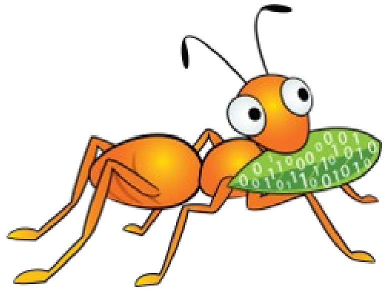
- Volume
  - Bricks combined and passed through translators
  - Ultimately, what's presented to the end user
- Peer / Node
  - Server hosting the brick filesystems
  - Runs the Gluster daemons and participates in volumes
- Trusted Storage Pool
  - A group of peers, like a “Gluster cluster”



Scalability of replication

- Distribution across replicated subvolumes



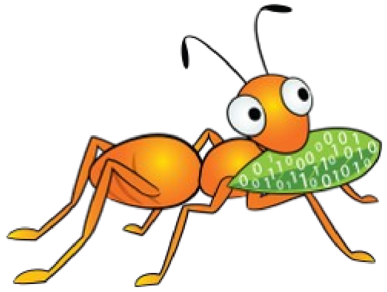


Synchronous vs Asynchronous

Synchronous	**Asynchronous**
All copies are always up to date	There is a delay in replication of data
Requires low latency	Latency not critical
Client are blocked while replication	Replication is done in the background
Mixing of volume types not possible	Can have different types on master/slave
AFR, disperse and JBR	Geo-replication

Geo-replication

- Asynchronous replication
- Master (read-write) to one or more slaves (read-only)
- Used for disaster recovery, content delivery networks, ...
- Workers from the master volume push data to the slave
- Change detection on bricks, sync through the volume
  - Filesystem crawl, or based on changelog
- Monitoring of worker processes
- Automatic fail-over to other slave in case of problems



Geo-replication

- Checkpoint feature
  - verification of contents at a given time
- Can run as non-root user
- Recently added tool `schedule_georep` can be used to replicate only during certain times

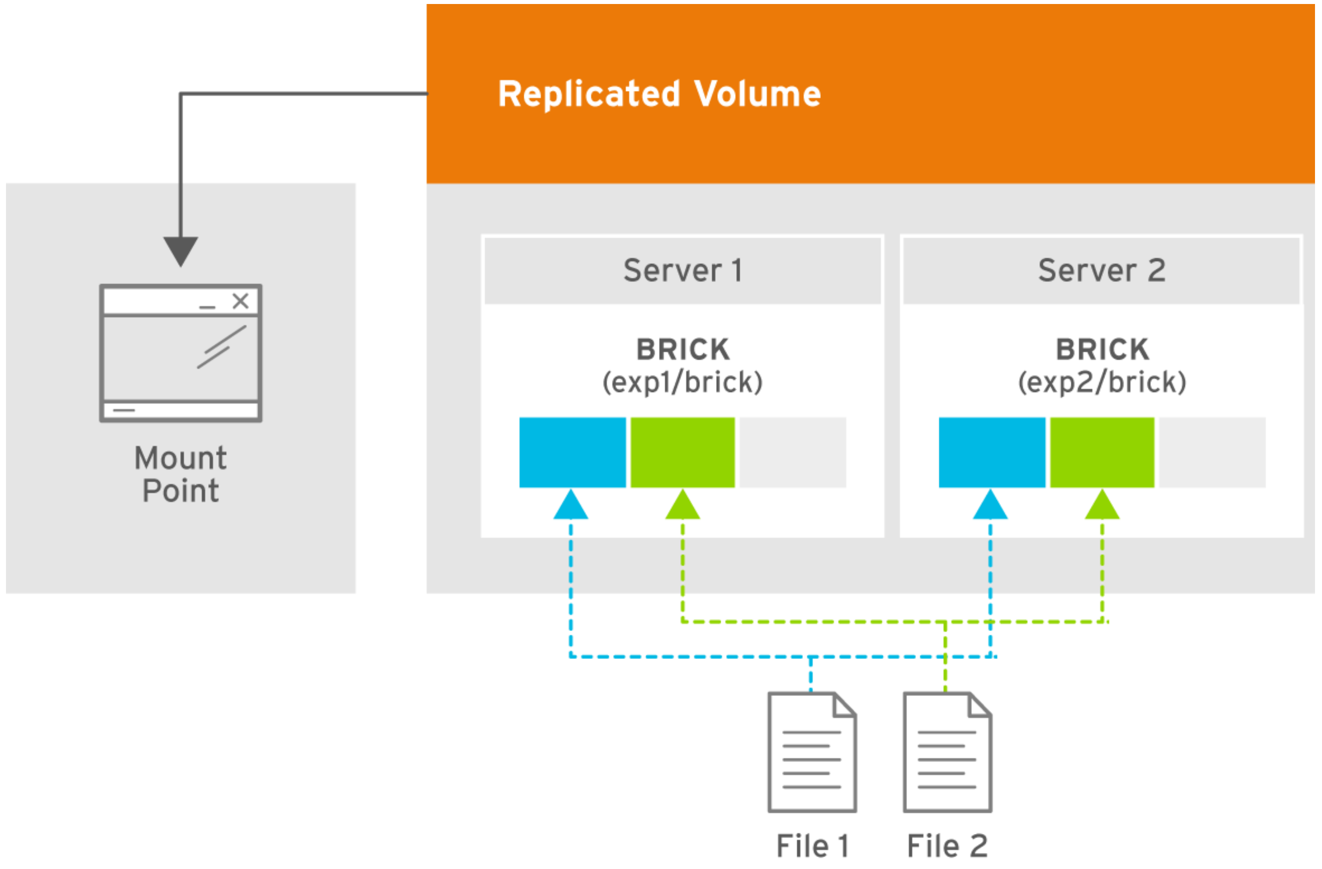
Client-side vs Server-side

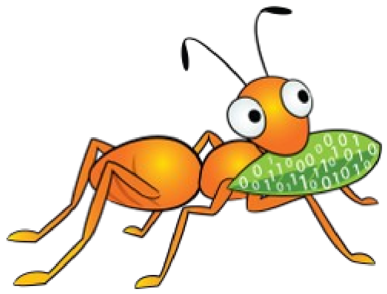
- Replication done by the client
  - FUSE mounts, libgfapi (QEMU)
- Gluster clients can be servers too
  - Gluster/NFS, NFS-Ganesha, Samba, Apache httpd

- AFR and disperse are working on the client-side
- JBR is doing server-side replication

Automatic File Replication

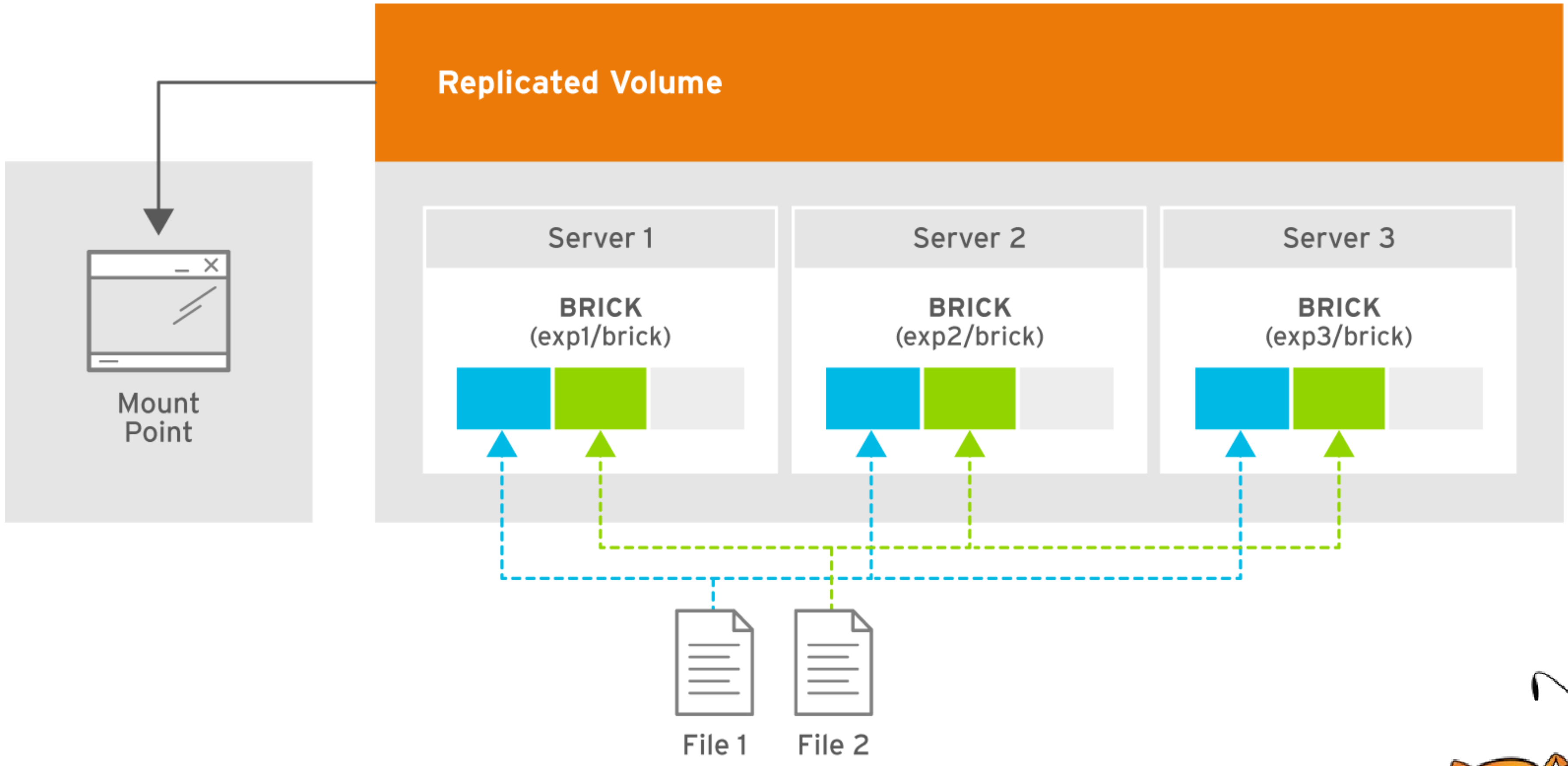
- Copies files to multiple bricks
- *Similar* to file-level RAID 1





Automatic File Replication

- Copies files to multiple bricks
- Arbiter volume to reduce the storage need



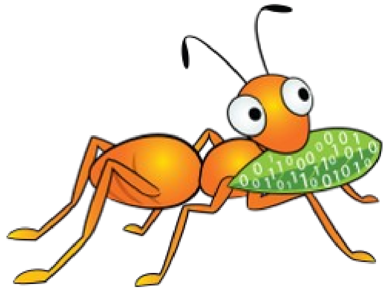
AFR Transactions

All modification FOPs (create, write, delete etc.) happen inside a 5-stage transaction:

1. Lock
2. Pre-op – set a dirty xattr on the file
3. Write
4. Post-op – clear the dirty xattr and set pending xattrs for failed writes.
5. Unlock

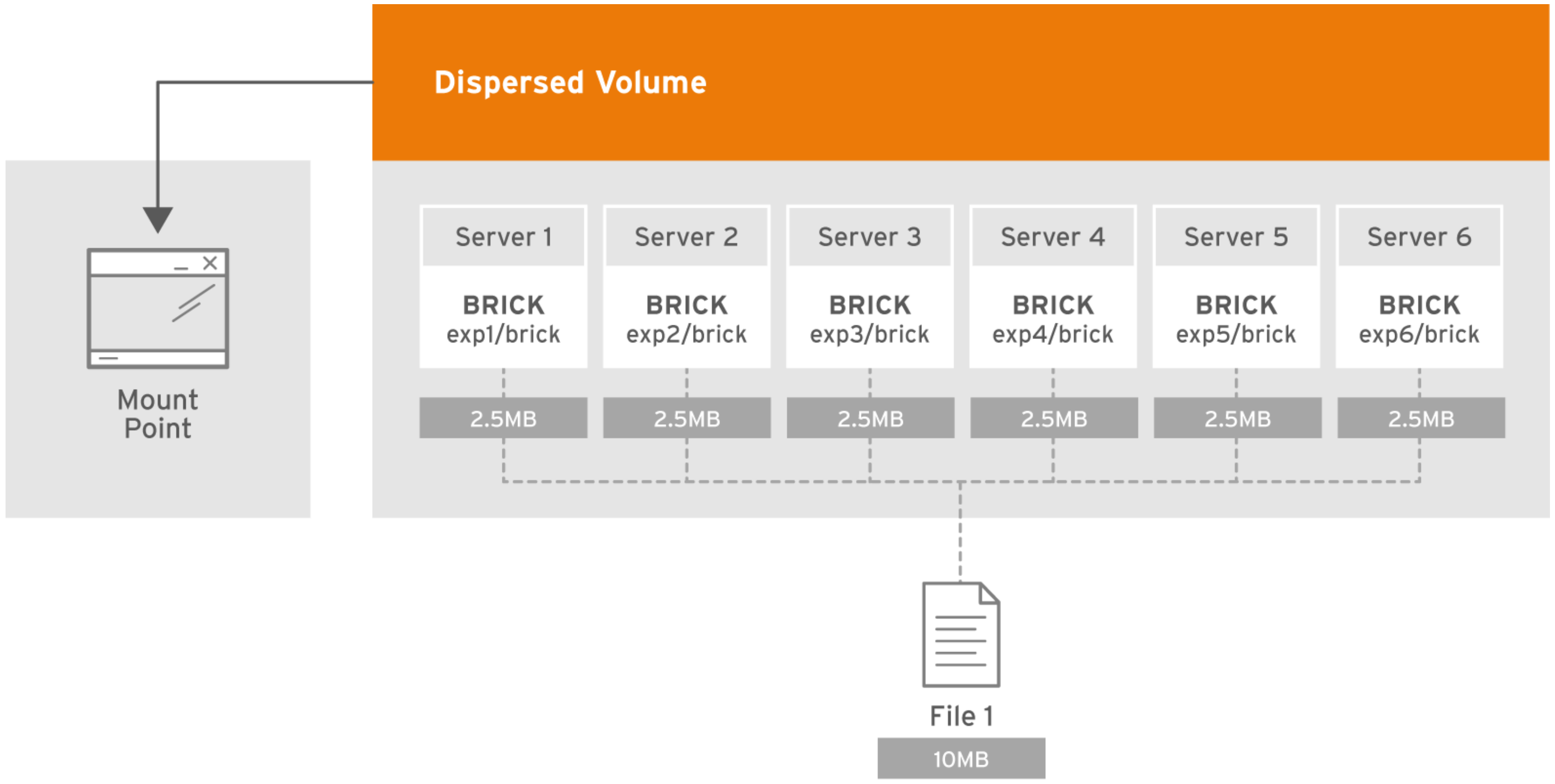
AFR Improvements

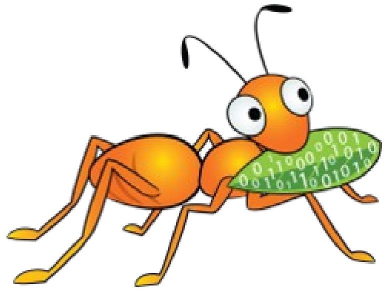
- Compound Operations
  - Combine multiple operations, reducing roundtrips
  - i.e. CREATE + WRITE for object storage PUSH
- Policy based split-brain resolution
  - Heal from known good brick
  - Heal by selected attributes (mtime, size, …)
- Throttling translator
- Granular entry self-heal
  - Improves performance for directory healing



Disperse / Erasure Coding

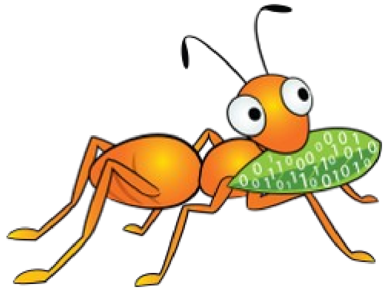
- *Similar* to RAID 5/6 over the network
- Encoded fragments of files





Disperse Improvements

- Systematic algorithm
  - Only encode parity fragments
  - Prevents decoding on READ for most common cases
  - Finer grained recovery possible, restore from fewer available fragments
- Hardware acceleration (ASM code, larger CPU words)
- Caching of inverse matrices
- New algorithm to calculate the inverse matrices
- Reducing `memcpy()` calls



Journal Based Replication

- Planned for Gluster 4.0 (end of the year?)
- Server to server
  - Faster I/O path for most deployments/workloads
- The leader instructs the followers
- Client connects to a brick, finds the leader
- Flexible Consistency
  - Issue Count: no. followers to complete before leader
  - Completion Count: no. bricks to complete

Journal Based Replication

- Journal with Full Data Logging
- All followers keep their own journal
- The journal can be on a separate (fast) device
- Entries can be kept in memory until `fsync()`
- One journal per term
  - Leadership change starts a new term
- Transactions do not require a post-op like AFR

Journal Based Replication

- Separate daemon for membership changes
- Information about terms retrieved from `etcd`
  - `etcd` is also used with GlusterD 2.0
- Information within terms retrieved from the bricks

Resources

Mailing lists:

- gluster-users@gluster.org
- gluster-devel@gluster.org

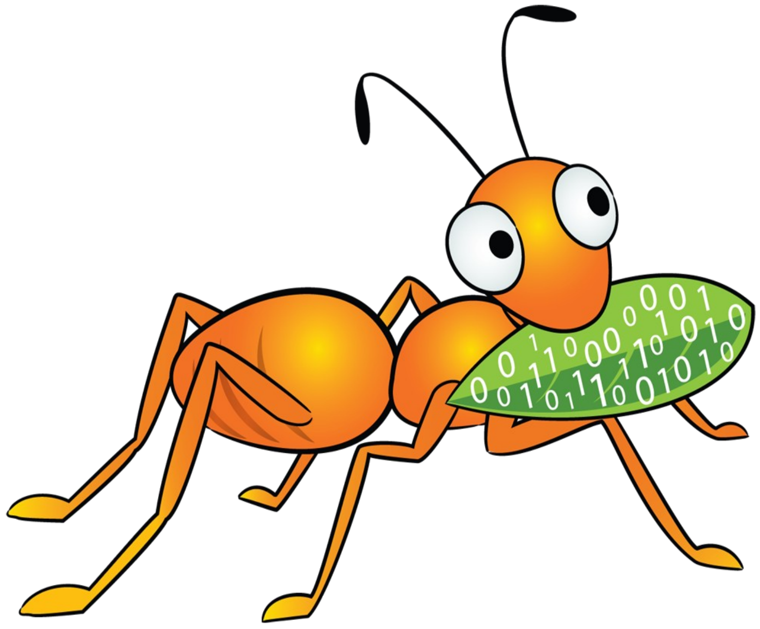
IRC:

- #gluster and #gluster-dev on Freenode

Links:

- http://gluster.org/
- http://gluster.readthedocs.org/
- https://github.com/gluster/

Thank you!



Niels de Vos
ndevos@redhat.com
ndevos on IRC
@nixpanic on Twitter