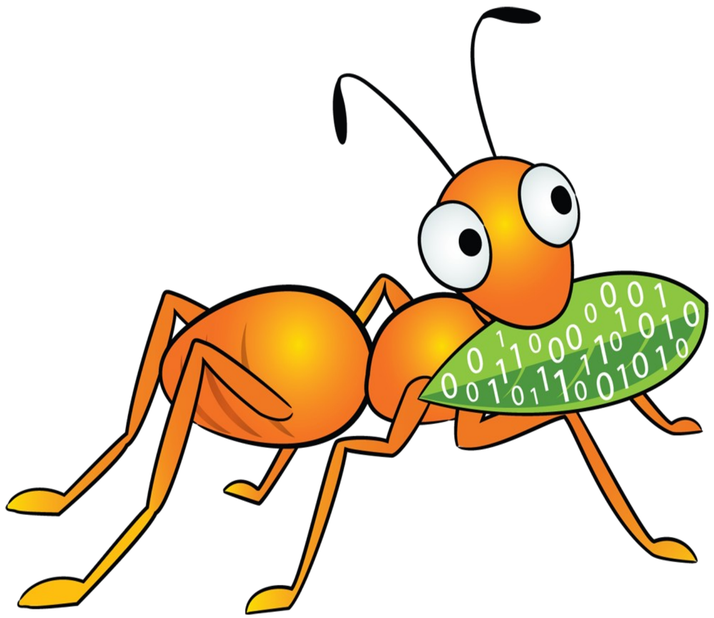
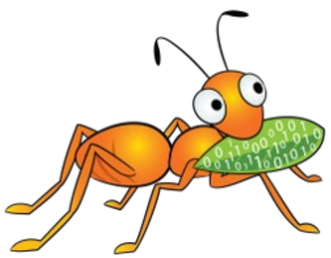
GlusterFS: Advancements in Automatic File Replication (AFR)



Ravishankar N.
Software Engineer, Red Hat
ravishankar@redhat.com
Oct 6th, LCE_EU- 2015

Agenda

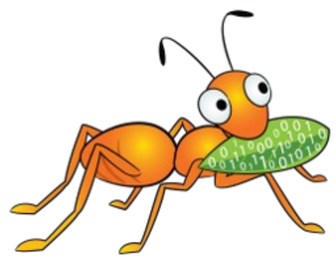
- What is GlusterFS- The 5 minute intro
- The Automatic File Replication (AFR) translator
- Recent improvements to AFR
  * glfsheal- A gfapi based application
  * commands for split-brain resolution
  * Arbiter volumes
- Upcoming enhancements to AFR
  * granular entry and data self-heals
  * throttling of self-heal fops
  * Multi-threaded self-heal



What is GlusterFS

gluster lingo knowledge check

- gluster server
- bricks
- Peers, trusted storage pool
- volume
- gluster client
- access protocols
- volume options
- translators
- graphs
- gfid
- glusterfsd, glustershd, nfs, glusterd, snapd, bitd



What is GlusterFS

A picture is worth a thousand words.. (words that you might already know.)

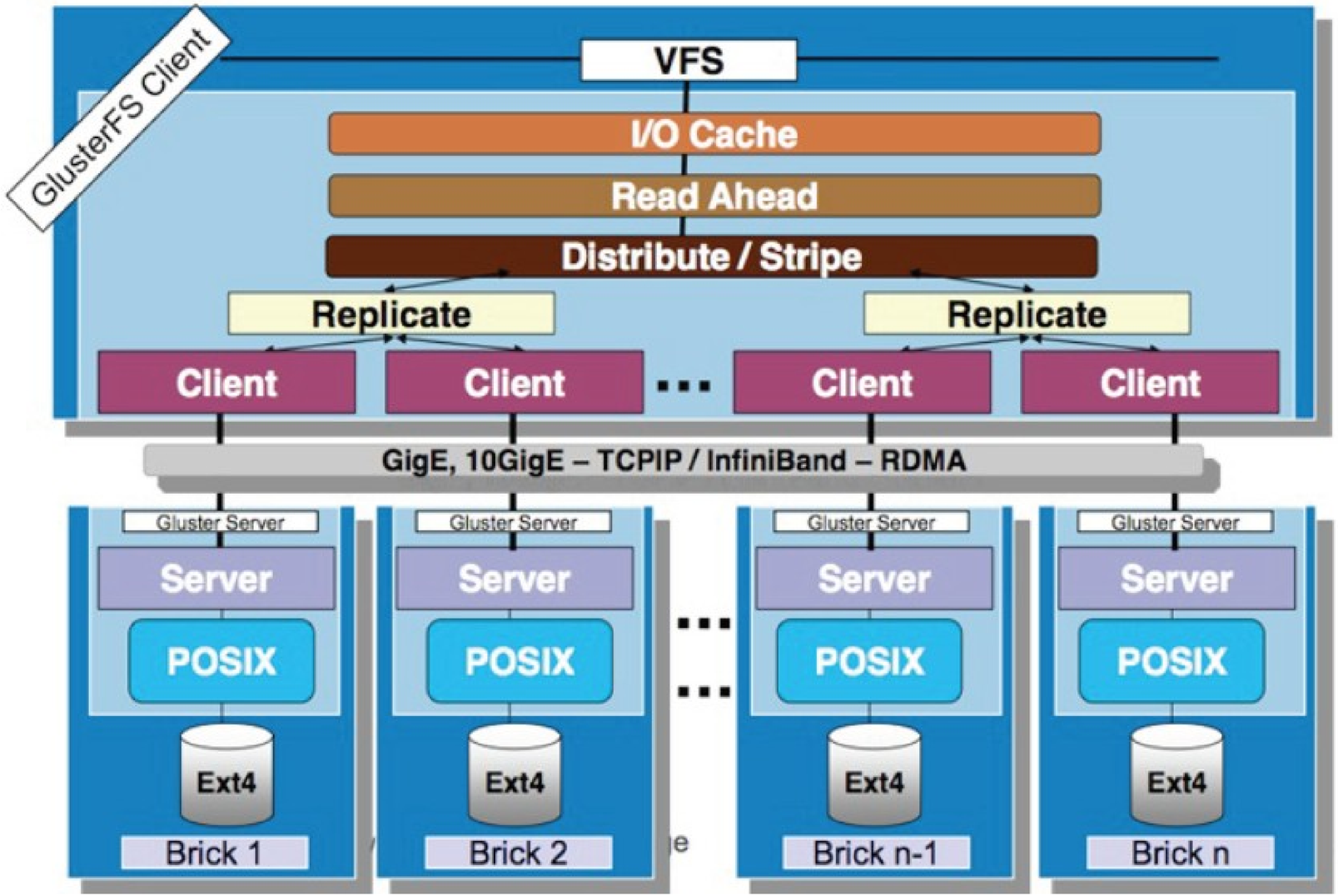
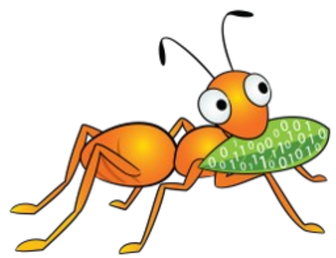
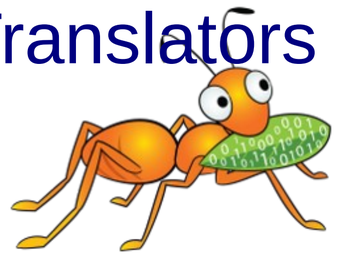


Image courtesy http://www.slideshare.net/openstackindia/glusterfs-and-openstack



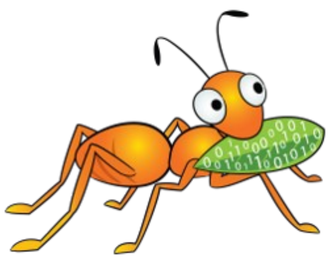
Translators 101

- Each gluster process is made of 'translators' (xlators) stacked on top of each other in a particular fashion to form a 'graph'.
- An xlator can be present on the client side or server side or both.
- Every File Op issued by the application (create, write, read etc.) passes through each of the xlators before hitting the disk.
- The xlator can do appropriate things to the FOP or just pass it down to the next xlator.
- A detailed introduction can be found at http://www.gluster.org/community/documentation/index.php/Translators



The AFR translator

- A client-side xlator that performs synchronous replication.
- Replicates writes to all bricks of the replica → Uses a transaction model.
- Serves reads from one of the bricks of the replica. Each file has a different 'read-subvolume' brick .
- Provides high availability when one of the bricks go down.
- Heals files that were created/deleted/modified when the brick comes back up.

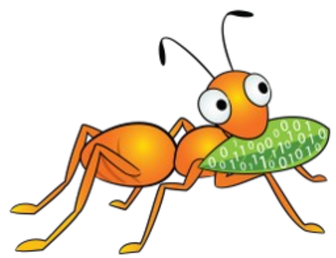


AFR xlator- The write transaction model

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

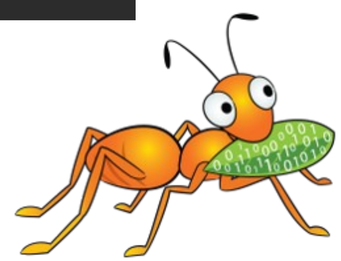
* all of AFR's xattrs begin with '*trusted.afr.*'



Let's consider a 1x2 replicated volume:

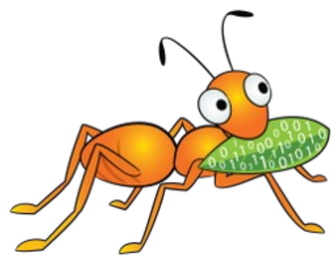
- state of AFR xattrs on the bricks after a Pre-op:

```
[root@vm1 ~]# getfattr -d -m . -e hex /bricks/brick1/file
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick1/file
security.selinux=0x73797374656d5f753a6f626a6563745f723a756e6c6162656c65645f743a733000
trusted.afr.dirty=0x000000010000000000000000
trusted.bit-rot.version=0x020000000000000560cc9b8000d803a
trusted.gfid=0x1a7dca918f294c98bb78ca96794db884

[root@vm1 ~]#
[root@vm1 ~]# getfattr -d -m . -e hex /bricks/brick2/file
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick2/file
security.selinux=0x73797374656d5f753a6f626a6563745f723a756e6c6162656c65645f743a733000
trusted.afr.dirty=0x000000010000000000000000
trusted.bit-rot.version=0x020000000000000560cc9b90002a33c
trusted.gfid=0x1a7dca918f294c98bb78ca96794db884
```



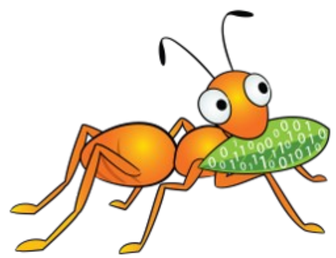
- state of AFR xattrs after the Post-op when write succeeds on <u>both</u> bricks:

```
[root@vm1 ~]# getfattr -d -m . -e hex /bricks/brick1/file
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick1/file
security.selinux=0x73797374656d5f753a6f626a6563745f723a756e6c6162656c65645f743a733000
trusted.afr.dirty=0x000000000000000000000000
trusted.bit-rot.version=0x020000000000000560cc9b8000d803a
trusted.gfid=0x1a7dca918f294c98bb78ca96794db884

[root@vm1 ~]#
[root@vm1 ~]# getfattr -d -m . -e hex /bricks/brick2/file
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick2/file
security.selinux=0x73797374656d5f753a6f626a6563745f723a756e6c6162656c65645f743a733000
trusted.afr.dirty=0x000000000000000000000000
trusted.bit-rot.version=0x020000000000000560cc9b90002a33c
trusted.gfid=0x1a7dca918f294c98bb78ca96794db884
```



- state of AFR xattrs after the Post-op when write succeeds on only <u>one</u> of the bricks, say brick1

```
[root@vm1 glusterd]# getfattr -d -m . -e hex  /bricks/brick1/file
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick1/file
security.selinux=0x73797374656d5f753a6f626a6563745f723a756e6c6162656c65645f743a733000
trusted.afr.dirty=0x000000000000000000000000
trusted.afr.testvol-client-1=0x000000010000000000000000
trusted.bit-rot.version=0x020000000000000560cc9b8000d803a
trusted.gfid=0x1a7dca918f294c98bb78ca96794db884

[root@vm1 glusterd]#
[root@vm1 glusterd]# getfattr -d -m . -e hex  /bricks/brick2/file
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick2/file
security.selinux=0x73797374656d5f753a6f626a6563745f723a756e6c6162656c65645f743a733000
trusted.afr.dirty=0x000000010000000000000000
trusted.bit-rot.version=0x020000000000000560cc9b90002a33c
trusted.gfid=0x1a7dca918f294c98bb78ca96794db884
```



self-healing

- Healing happens when a brick that went down comes back online.
- Healing is done via 3 methods:

  a) A dedicated self-heal daemon (which has the AFR xlator in its stack) which periodically scans */brick/.glusterfs/indices/xattrop* for the list of files that need heal.

  b)From the mount when the file is accessed.

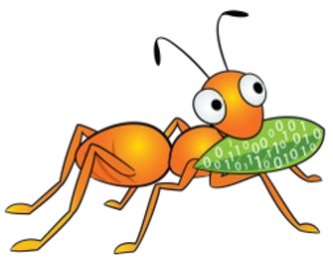
  c) Using the CLI: `gluster volume heal <VOLNAME>

- The direction of heal (i.e. the 'source' brick and the 'sink' brick) is determined by examining the trusted.afr* xattrs.

In the previous slide, the xattr of 'file' on brick1 (trusted.afr.testvol-client-0) blames brick-2 (trusted.afr.testvol-client-1):

i.e. `trusted.afr.testvol-client-1=0x00000001000000000000000`

Which means the self-heal of file's contents happens from brick-1 to brick-2



Split-brains

- Split-brain is a state where each brick blames the other one for the file in question
- How do we end up in split-brain?
  - Brick-1 goes down, writes happen on the file
  - Brick-2 goes down, brick-1 comes up, writes happen to the file.
  - Now we have afr xattrs blaming each other- i.e. split-brain. Self healing cannot happen- no definite source and sink.

```
[root@vm1 ~]# getfattr -d -m . -e hex  /bricks/brick1/file
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick1/file
security.selinux=0x73797374656d5f753a6f626a6563745f723a756e6c6162656c65645f743a733000
trusted.afr.dirty=0x000000000000000000000000
trusted.afr.testvol-client-1=0x000000010000000000000000
trusted.bit-rot.version=0x02000000000000000560cc9b8000d803a
trusted.gfid=0x5eecb6782b8f433e8a5efffc9dd7d08c

[root@vm1 ~]# getfattr -d -m . -e hex  /bricks/brick2/file
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick2/file
security.selinux=0x73797374656d5f753a6f626a6563745f723a756e6c6162656c65645f743a733000
trusted.afr.dirty=0x000000000000000000000000
trusted.afr.testvol-client-0=0x000000010000000000000000
trusted.bit-rot.version=0x03000000000000000560cfa52000277e3
trusted.gfid=0x5eecb6782b8f433e8a5efffc9dd7d08c
```

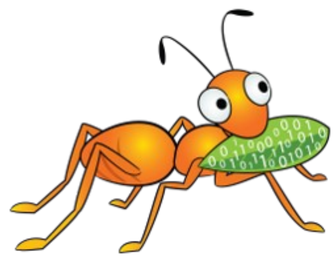
  - A brick doesn't always have to be down. Even network disconnects can lead to this situation. In short, the client cannot 'talk' with the brick, whatever be the reason.
  - When a file that is in split-brain is accessed by the client, it gets EIO.

Recent improvements to AFR

Improvements to heal info

- Better reporting of files that need heal and those in split-brain.
- Implemented using *glfsheal*- A program written using libgfapi to give information about pending heals.
  - Invoked when you run `gluster volume heal <VOLNAME> info`. No change from a user PoV.
  - Replaces the reporting traditionally done by the self-heal daemon- Better, faster, stronger!

```
[root@vm1 ~]# gluster volume heal testvol info
Brick 127.0.0.2:/bricks/brick1
/file - Is in split-brain

/hello.txt
Number of entries: 2

Brick 127.0.0.2:/bricks/brick2
/file - Is in split-brain

Number of entries: 1

[root@vm1 ~]#
```



Split brain resolution

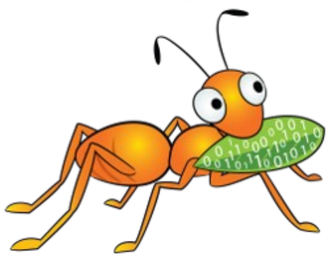
So you ended up in a split-brain. How do you get out of it?

- Before glusterfs-3.7- Manually examining the trusted.afr* xattrs and resetting the appropriate ones, then running the heal command. See this link.
- Since 3.7, we have two ways to resolve data and metadata split-brains.

  a) Policy based resolution: server side, done with gluster CLI. typically by the admin. Works by invoking *glfsheal*.

  b) Mount point based resolution: client side, done with virtual xattrs, typically by the user.

  But there's a gotcha! These commands do not work for gfid split-brains. They still need manual examination.



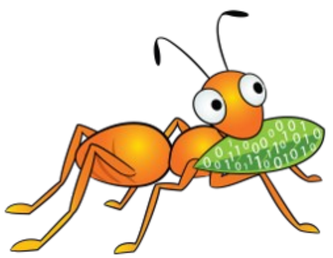
a) Policy based:

- `gluster volume heal <VOLNAME> split-brain bigger-file <FILE>`
- `gluster volume heal <VOLNAME> split-brain source-brick <HOSTNAME:BRICKNAME> <FILE>`
- `gluster volume heal <VOLNAME> split-brain <HOSTNAME:BRICKNAME>`

b) Mount based:

- `getfattr -n replica.split-brain-status <FILE>`
- `setfattr -n replica.split-brain-choice -v "choiceX" <FILE>`
- `setfattr -n replica.split-brain-heal-finalize -v <heal-choice> <FILE>`

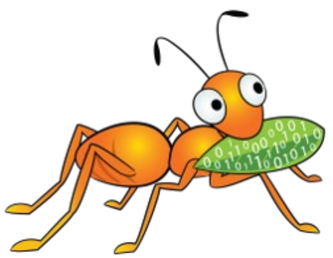
• Click here for a detailed example of how to use these commands.



But I don't want split-brains.

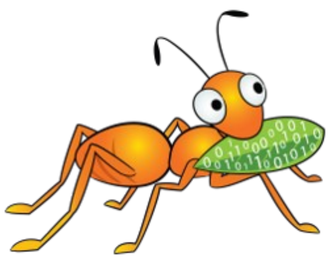
- Not possible with replica-2, without losing high availability. Both bricks need to be up for quorum.
- Use replica-3 with client-quorum enabled. ==> Works. The best solution if 3x storage space is not a concern.
- But is there a sweet-spot between replica 2 and replica-3?

  Yes! Presenting the Arbiter configuration (a.k.a. **Arbiter volume**) for replica-3.



What's the Arbiter volume all about ?

- A replica-3 volume where the 3rd brick only stores file metadata and no data.
- Consumes less space compared to a full blown replica-3
- Takes full file locks for all writes, as opposed to range locks. (so theoretically, it would be slow for multi-writer scenarios).
- Does not allow a write FOP if it can result in a split-brain- unwinds with ENOTCONN
- Client-quorum is enabled by default for arbiter volumes too (i.e. 2 bricks need to be up for writes to go through).



What's the Arbiter volume all about ?

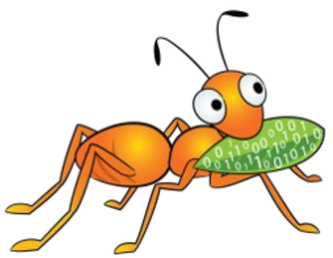
- Syntax for arbiter volume creation:

```
gluster volume create <VOLNAME> replica 3 arbiter 1 host1:brick1
host2:brick2 host3:brick3
```

- How to check if a volume is normal replica-3 or an arbiter?

`$mount_point/.meta/graphs/active/$V0-replicate-0/options/arbiter-count` exists and its value is 1

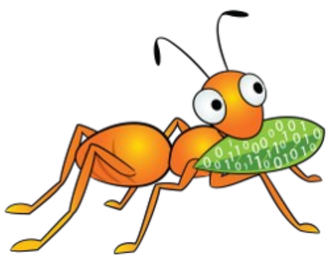
- How do self-heals work for arbiter volumes?
  - Arbiter brick cannot be used for data self-heal. Entry and metadata self-heals work.



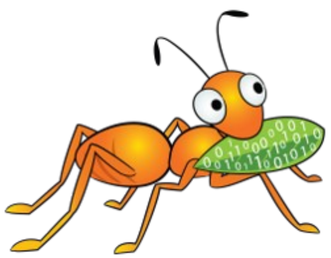
Upcoming enhancements

- Granular entry self-heals
  - Current algorithm uses afr xattrs to indicate 'a directory needs healing' but does not give the list of files that need heal. Uses expunge-impunge method.
  - The proposed change is to store the names of files that need heal in .glusterfs/indices/entry-changes/<parent-dir-gfid>/ and heal only them.
- Granular data self-heals
  - Likewise for data heals. As of today, we copy the entire file contents while healing.
  - The proposed change is to store a bit-map in the xattr to indicate the 'range' that needs heal.

See http://review.gluster.org/#/c/12257/



- Performance and throttling improvements:
  - current implementation: one thread per brick for index heals, acting on one file at a time.
  - Multi threading can speed things up. Patch by Richard Wareing of facebook under review.
  - Not without problems (high cpu/network usage). Need to introduce throttling.
    - Exploring Token Bucket Filters- already used by bit rot daemon.
  - Compounding of FOPS.



Epilogue

- AFR dev team: Pranith Kumar, Anuradha Talur, Krutika Dhanajay and myself.

  Find us on IRC at freenode , #gluster-users or #gluster-devel: pranithk, atalur, kdhanajay, itisravi

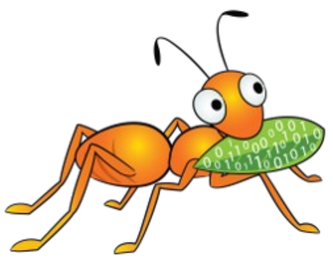
- Show me the code! `` `git log xlators/cluster/afr` ``
- Documentation related to AFR (some are a bit dated).

https://github.com/gluster/glusterfs/blob/master/doc/developer-guide/afr/self-heal-daemon.md

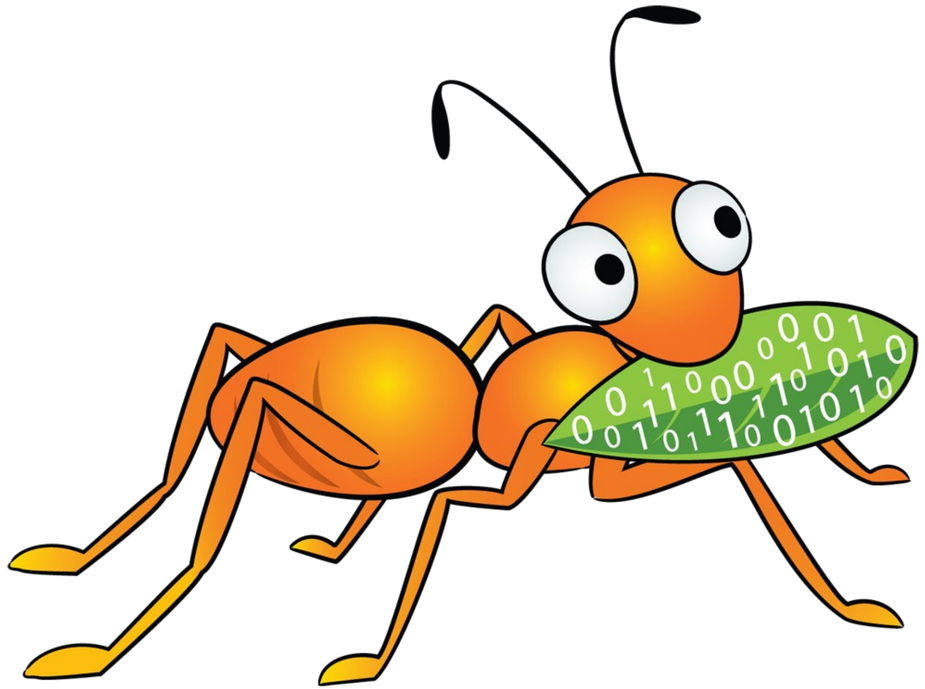
https://github.com/gluster/glusterfs/blob/master/doc/developer-guide/afr/afr-locks-evolution.md

https://github.com/gluster/glusterfs-specs/blob/master/done/Features/afr-v1.md

https://github.com/gluster/glusterfs-specs/blob/master/done/Features/afr-statistics.md



Questions/ comments ?



Thank you and stay tuned to the mailing-list!