3.6. index 模块

在 glusterfs 中，index 也是一个很重要的模块，对于一个文件要进行修改或者更新的时候，因为 glusterfs 的无中心架构，因此并没有一个元数据中心来记录当前哪个文件被更改，而一旦出现异常需要进行处理，对异常文件的寻找和修复，那么就非常麻烦了。因此在很早以前，glusterfs 关于文件异常修复，需要扫描全部文件，那样的做法是不可行的，而为了解决这个问题，就引入了 heal 的概念，也就是自愈，而想要实现该功能，index 就是一个非常重要的内容了。

3.6.1. dirty 目录

为了了解这方面的内容，先简单做个试验感受一下。

```
root@gfs01:~# gluster volume info test-index

Volume Name: test-index
Type: Replicate
Volume ID: 58675259-84eb-4e85-8a46-9d9a9bfb48f6
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x 3 = 3
Transport-type: tcp
Bricks:
Brick1: 10.0.12.2:/glusterfs/test-index
Brick2: 10.0.12.9:/glusterfs/test-index
Brick3: 10.0.12.12:/glusterfs/test-index
Options Reconfigured:
cluster.granular-entry-heal: on
storage.fips-mode-rchecksum: on
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off

```

```
22.
23. root@gfs02:~# tree   /glusterfs/test-index/.glusterfs/indi
    ces/
24. /glusterfs/test-index/.glusterfs/indices/
25. ├── dirty
26. ├── entry-changes
27. └── xattrop
```

在创建 volume 之后，这里会有一个隐藏目录.glusterfs,在前面讲 gfid 的时候就提到了，那么该目录下还有一个特殊的重要目录 indices，这下面有三个目录，暂时都是没有内容的，那么这里到底有什么作用呢？下面继续测试一下。

```
1.  root@gfs01:~# df -h /mnt/test-index/
2.  Filesystem             Size  Used Avail Use% Mounted on
3.  10.0.12.2:test-index    40G  8.1G   30G  22% /mnt/test-ind
    ex
4.
5.
6.  root@gfs01:~# dd if=/dev/zero  of=/mnt/test-index/1.txt b
    s=1k count=100000
7.  100000+0 records in
8.  100000+0 records out
9.  102400000 bytes (102 MB, 98 MiB) copied, 14.1931 s, 7.2 M
    B/s
10.
11. root@gfs02:~# tree   /glusterfs/test-index/.glusterfs/indi
    ces/
12. /glusterfs/test-index/.glusterfs/indices/
13. ├── dirty
14. │   ├── a9f2710b-ea25-471b-bb3b-bd799e76fa1f
15. │   └── dirty-495a4ce0-8117-4bac-992e-5c9bb69260b9
16. ├── entry-changes
17. └── xattrop
18.
19. 3 directories, 2 files
```

这里挂载 volume 之后,然后使用 dd 命令创建文件 ,这里接着在 brick 目录下查看该隐藏文件，可以看到 dirty 目录下这里多出来了两个文件，其中一个文件是没有带上前缀 dirty 的，那么这个文件名这一串其实是什么呢？这个其实就

是 brick 目录下实际文件的 gfid 编号，而且这两个文件的 inode 是一样的。

```
root@gfs02:~# ls -i /glusterfs/test-index/.glusterfs/indices/dirty/a9f2710b-ea25-471b-bb3b-bd799e76fa1f
2101879 /glusterfs/test-index/.glusterfs/indices/dirty/a9f2710b-ea25-471b-bb3b-bd799e76fa1f
root@gfs02:~# ls -i /glusterfs/test-index/1.txt
2101878 /glusterfs/test-index/1.txt

root@gfs02:~# getfattr -d -m . -e hex /glusterfs/test-index/1.txt
getfattr: Removing leading '/' from absolute path names
# file: glusterfs/test-index/1.txt
trusted.afr.dirty=0x000000010000000000000000
trusted.gfid=0xa9f2710bea25471bbb3bbd799e76fa1f
trusted.gfid2path.46104af11fea85fe=0x30303030303030302d3030303030302d303030302d303030302d303030303030303030303030312f312e747874
trusted.glusterfs.mdata=0x010000000000000000000000060e0224400000000302721440000000060e0224400000000302721440000000060e01d4a000000000952863e
```

可以看到这里其实就是一个硬链接了，然后当写入完成的时候，该文件也会被删除的，就只剩下以 dirty 为前缀开头的文件了。

所以从这个简单的小实验中可以看到，dirty 这个文件就是用来记录正在改变的文件的。

3.6.2. xattrop 目录

如果这里在创建过中出现异常呢？例如 brick 进程被杀死了，这里会有怎样的表现呢？下面继续来实验一下。另外这里为了测试的准确性，再次重新创建了该 volume，因此会和上面的实验部分内容有点不一样，但是不影响测试情况。

```
root@gfs02:~# ps -ef |grep 2617591
```

```
2.  root     2617591       1  0 16:59 ?        00:00:00 /usr/
   sbin/glusterfsd -s 10.0.12.2 --volfile-id test-index.10.0.
   12.2.glusterfs-test-index -p /var/run/gluster/vols/test-in
   dex/10.0.12.2-glusterfs-test-index.pid -S /var/run/gluster
   /21ffe8c1aae2d06c.socket --brick-name /glusterfs/test-inde
   x -l /var/log/glusterfs/bricks/glusterfs-test-index.log --
   xlator-option *-posix.glusterd-uuid=69da12d3-7d23-408b-add
   3-e20e60ae725f --process-name brick --brick-port 49164 --x
   lator-option test-index-server.listen-port=49164
3.
4.
5.  root@gfs02:~# tree  /glusterfs/test-index/.glusterfs/indi
   ces/
6.  /glusterfs/test-index/.glusterfs/indices/
7.  ├── dirty
8.  │   ├── dirty-1bf6eb50-1087-42a2-a430-a04b4c9efda8
9.  │   └── e9759499-e824-4a99-8265-a63894c386a2
10. ├── entry-changes
11. └── xattrop
12.
13. 3 directories, 2 files
14. root@gfs02:~# kill -9 2617591
15. root@gfs02:~# tree  /glusterfs/test-index/.glusterfs/indi
   ces/
16. /glusterfs/test-index/.glusterfs/indices/
17. ├── dirty
18. │   ├── dirty-1bf6eb50-1087-42a2-a430-a04b4c9efda8
19. │   └── e9759499-e824-4a99-8265-a63894c386a2
20. ├── entry-changes
21. └── xattrop
22.
23. 3 directories, 2 files
```

这里首先找到其中一个 brick 的 pid ,然后在创建文件的过程中杀掉该进程，接着再查看 indices 的目录结构，等到任务完成的时候，这里再次查看其它健康的 brick 的该目录情况。

```
1.  root@gfs03:~# tree  /glusterfs/test-index/.glusterfs/indi
   ces/
2.  /glusterfs/test-index/.glusterfs/indices/
```

```
├── dirty
│   └── dirty-c9493c55-3b96-4b13-a93f-dc05409ee34e
├── entry-changes
└── xattrop
    ├── e9759499-e824-4a99-8265-a63894c386a2
    └── xattrop-c9493c55-3b96-4b13-a93f-dc05409ee34e

3 directories, 3 files

root@gfs01:~# tree  /glusterfs/test-index/.glusterfs/indices/
/glusterfs/test-index/.glusterfs/indices/
├── dirty
│   └── dirty-027d5224-8211-483e-8bf1-8b088222f6f8
├── entry-changes
└── xattrop
    ├── e9759499-e824-4a99-8265-a63894c386a2
    └── xattrop-027d5224-8211-483e-8bf1-8b088222f6f8

3 directories, 3 files
```

从这里可以看到，其它的两个节点，在 dirty 目录下面是没有了以 gfid 开头的文件，但是在 xattrop 中是存在该文件的，也就是说，这里当这里有 brick 没有完成请求的时候，需要对文件进行异常修复处理时，那么这里就会在 xattrop 目录下保留该文件了，这样方便记录知道哪些文件需要进行修复自愈。

```
root@gfs01:~# gluster volume heal test-index info summary

Brick 10.0.12.2:/glusterfs/test-index
Status: Transport endpoint is not connected
Total Number of entries: -
Number of entries in heal pending: -
Number of entries in split-brain: -
Number of entries possibly healing: -

Brick 10.0.12.9:/glusterfs/test-index
Status: Connected
Total Number of entries: 1
Number of entries in heal pending: 1
Number of entries in split-brain: 0
```

```
Number of entries possibly healing: 0

Brick 10.0.12.12:/glusterfs/test-index
Status: Connected
Total Number of entries: 1
Number of entries in heal pending: 1
Number of entries in split-brain: 0
Number of entries possibly healing: 0

root@gfs01:~# gluster volume heal test-index info
Brick 10.0.12.2:/glusterfs/test-index
Status: Transport endpoint is not connected
Number of entries: -

Brick 10.0.12.9:/glusterfs/test-index
/1.txt
Status: Connected
Number of entries: 1

Brick 10.0.12.12:/glusterfs/test-index
/1.txt
Status: Connected
Number of entries: 1
```

这里也可以从这里知道一些具体的信息情况,heal info 中显示了当前有一个 brick 是无法连接上的，这里是需要留意的。那么这里要怎么解决这个问题呢？可以再次重启 brick 的进程即可，这里使用命令 start force 即可，然后启动之后，可以观察一下异常 brick 所在节点的 heal 日志，也就是 glustershd.log 日志，下面简单查看一下日志的情况。

```
root@gfs02:~# cat  /var/log/glusterfs/glustershd.log
...
//这里是 brick 进程被杀死时的日志
[2021-07-03 09:28:49.541314 +0000] D [MSGID: 0] [client.c:2231:client_rpc_notify] 16-test-index-client-0: disconnected from test-index-client-0. Client process will keep trying to connect to glusterd until brick's port is available
```

```
5.  [2021-07-03 09:28:51.550508 +0000] I [MSGID: 100041] [glu
    sterfsd-mgmt.c:1034:glusterfs_handle_svc_attach]  0-gluster
    fs: received attach request for volfile [{volfile-id=shd/t
    est-index}]
6.  [2021-07-03 09:28:51.550550 +0000] I [MSGID: 100040] [glu
    sterfsd-mgmt.c:108:mgmt_process_volfile] 0-glusterfs: No c
    hange in volfile, countinuing []
7.  ...
8.
9.
10. //下面开始是强制启动 brick 进程，重新连接的日志
11. [2021-07-03 09:28:52.544202 +0000] I [MSGID: 114046] [cli
    ent-handshake.c:855:client_setvolume_cbk]  16-test-index-cl
    ient-0: Connected, attached to remote volume [{conn-name=t
    est-index-client-0}, {remote_subvol=/glusterfs/test-index}]

12. [2021-07-03 09:28:52.544216 +0000] D [MSGID: 0] [client-h
    andshake.c:675:client_post_handshake]  16-test-index-client
    -0: No fds to open - notifying all parents child up
13. [2021-07-03 09:28:52.544230 +0000] D [MSGID: 0] [afr-comm
    on.c:5977:afr_get_halo_latency]  16-test-index-replicate-0:
     Using halo latency 99999
14. [2021-07-03 09:28:52.544494 +0000] D [rpc-clnt-ping.c:93:
    rpc_clnt_remove_ping_timer_locked] (--> /lib/x86_64-linux-
    gnu/libglusterfs.so.0(_gf_log_callingfn+0x199)[0x7f45431b3
    209] (--> /lib/x86_64-linux-gnu/libgfrpc.so.0(+0x13bd2)[0x
    7f454315ebd2] (--> /lib/x86_64-linux-gnu/libgfrpc.so.0(+0x
    14401)[0x7f454315f401] (--> /lib/x86_64-linux-gnu/libgfrpc.
    so.0(rpc_clnt_submit+0x30f)[0x7f454315b5df] (--> /usr/lib/
    x86_64-linux-gnu/glusterfs/9.2/xlator/protocol/client.so(+
    0x16396)[0x7f453d9c7396] ))))) 16-: 10.0.12.12:49163: ping
     timer event already removed
15. ...
16. //这里开始进行 heal 的信息获取
17. [2021-07-03 09:28:52.544964 +0000] D [MSGID: 0] [client-r
    pc-fops_v2.c:916:client4_0_getxattr_cbk]  16-test-index-cli
    ent-2: resetting op_ret to 0 from 59
18. [2021-07-03 09:28:52.544957 +0000] D [rpc-clnt-ping.c:194:
    rpc_clnt_ping_cbk] 16-test-index-client-2: Ping latency is
     0ms
```

```
19. [2021-07-03 09:28:52.545098 +0000] D [MSGID: 0] [syncop-u
    tils.c:538:syncop_is_subvol_local]  16-test-index-client-2:
     subvol test-index-client-2 is not local
20. [2021-07-03 09:28:52.545193 +0000] D [MSGID: 0] [client-r
    pc-fops_v2.c:916:client4_0_getxattr_cbk]  16-test-index-cli
    ent-1: resetting op_ret to 0 from 59
21. [2021-07-03 09:28:52.545262 +0000] D [MSGID: 0] [syncop-u
    tils.c:538:syncop_is_subvol_local]  16-test-index-client-1:
     subvol test-index-client-1 is not local
22. [2021-07-03 09:28:52.545610 +0000] D [rpc-clnt-ping.c:194:
    rpc_clnt_ping_cbk] 16-test-index-client-1: Ping latency is
     0ms
23. [2021-07-03 09:28:52.545668 +0000] D [MSGID: 0] [client-r
    pc-fops_v2.c:916:client4_0_getxattr_cbk]  16-test-index-cli
    ent-0: resetting op_ret to 0 from 59
24. [2021-07-03 09:28:52.545708 +0000] D [MSGID: 0] [syncop-u
    tils.c:538:syncop_is_subvol_local]  16-test-index-client-0:
     subvol test-index-client-0 is local
25. [2021-07-03 09:28:52.545739 +0000] D [MSGID: 0] [afr-self
    -heald.c:1053:afr_shd_index_healer]  16-test-index-replicat
    e-0: starting index sweep on subvol test-index-client-0
26. ..
27. //这里开始进行自愈异常文件
28. [2021-07-03 09:28:52.547566 +0000] D [MSGID: 0] [afr-self
    -heald.c:415:afr_shd_index_heal] 16-test-index-replicate-0:
     got entry: e9759499-e824-4a99-8265-a63894c386a2 from test
    -index-client-0
29. [2021-07-03 09:28:52.547648 +0000] D [MSGID: 0] [stack.h:
    509:copy_frame] 16-stack: groups is null (ngrps: 0) [Inval
    id argument]
30. [2021-07-03 09:28:52.547978 +0000] D [MSGID: 0] [client-r
    pc-fops_v2.c:916:client4_0_getxattr_cbk]  16-test-index-cli
    ent-0: resetting op_ret to 0 from 43
31. [2021-07-03 09:28:52.548508 +0000] D [MSGID: 0] [afr-self
    -heal-common.c:2396:afr_selfheal_unlocked_inspect]  16-test
    -index-replicate-0: SIZE mismatch 77826048 vs 102400000 on
     test-index-client-1 for gfid:e9759499-e824-4a99-8265-a638
    94c386a2
32. [2021-07-03 09:28:52.548582 +0000] D [MSGID: 0] [afr-self
    -heal-common.c:2396:afr_selfheal_unlocked_inspect]  16-test
    -index-replicate-0: SIZE mismatch 77826048 vs 102400000 on
     test-index-client-2 for gfid:e9759499-e824-4a99-8265-a638
    94c386a2
```

```
33. //留意这里显示的,是 data-heal 出现问题了.
34. [2021-07-03 09:28:52.548887 +0000] D [MSGID: 0] [afr-self
    -heal-common.c:2560:afr_selfheal_do]  16-test-index-replica
    te-0: heals needed for e9759499-e824-4a99-8265-a63894c386a
    2: [entry-heal=0, metadata-heal=0, data-heal=1]
35.
36. ....
37. //这里看到文件自愈成功了
38. [2021-07-03 09:28:53.275487 +0000] D [MSGID: 0] [client-r
    pc-fops_v2.c:1536:client4_0_xattrop_cbk]  16-test-index-cli
    ent-2: resetting op_ret to 0 from 0
39. [2021-07-03 09:28:53.275860 +0000] I [MSGID: 108026] [afr
    -self-heal-common.c:1742:afr_log_selfheal]  16-test-index-r
    eplicate-0: Completed data selfheal on e9759499-e824-4a99-
    8265-a63894c386a2. sources=[1] 2  sinks=0
```

通过这里对日志的追踪分析，可以看到当重新连接之后，会获取其他正常节点的 heal 信息，其实就是 xattrop 目录的信息，确定这里要修复的异常文件。接着这里就开始进行修复了 ,同时这里可以留意到日志中还会判断该异常文件是出现了 data-heal 异常，也就是说这里是数据出现问题了，而不是文件的元信息出现问题了，这一点在 AFR 中也是非常重要的。那么这里自愈结束之后，也会把 xattrop 目录下的文件删除的。

3.6.3. entry-changes 目录

这里隐藏目录 indices 下面还有一个特殊的目录 entry-changes,那么这个目录的作用是什么呢？其实这个目录是用来记录要修复的文件目录的结构情况的，也就是哪个文件属于哪个目录下面，这里做一个小实验来感受一下。

```
1.  root@gfs01:~# gluster volume status  test-index
2.  Status of volume: test-index
```

```
Gluster process                                 TCP Port  RDM
A Port  Online  Pid
------------------------------------------------------------
----------------------
Brick 10.0.12.2:/glusterfs/test-index           49164     0
          Y       2635161
Brick 10.0.12.9:/glusterfs/test-index           49165     0
          Y       2776280
Brick 10.0.12.12:/glusterfs/test-index          49163     0
          Y       2665705
Self-heal Daemon on localhost                   N/A       N/A
          Y       128017
Self-heal Daemon on gfs02                       N/A       N/A
          Y       109807
Self-heal Daemon on 10.0.12.3                   N/A       N/A
          Y       15298
Self-heal Daemon on 10.0.12.7                   N/A       N/A
          Y       15526
Self-heal Daemon on 10.0.12.9                   N/A       N/A
          Y       2515251

Task Status of Volume test-index
------------------------------------------------------------
----------------------
There are no active volume tasks

root@gfs01:~# hostname -i
10.0.12.12

root@gfs01:~# kill -9 2665705

root@gfs01:~# cd /mnt/test-index/

root@gfs01:/mnt/test-index# touch 1.txt

root@gfs01:/mnt/test-index# mkdir -p a

root@gfs01:/mnt/test-index# mkdir -p b

root@gfs01:/mnt/test-index# touch a/a-01.txt

root@gfs01:/mnt/test-index# touch a/a-2.txt

root@gfs01:/mnt/test-index# touch b/b-2.txt
```

```
36.
37. root@gfs01:/mnt/test-index# touch b/b-1.txt
```

这里首先再次重新创建一个 volume，在删除旧的之前，要先把 brick 目录删掉，否则隐藏文件保留的话，是无法在原来的位置上重新创建的并且挂载。接着这里找到其中一个节点的 brick 的 pid，然后 kill 掉进入到挂载目录进行创建文件，这时候当前被 kill 掉节点下的隐藏目录是没有任何文件的，因为进程已经被 kill 掉了。

```
1.  root@gfs02:~# tree  /glusterfs/test-index/.glusterfs/indi
   ces/
2.  /glusterfs/test-index/.glusterfs/indices/
3.  ├── dirty
4.  │   └── dirty-db686567-3b6f-4391-b7e7-3d0c1b20fe7b
5.  ├── entry-changes
6.  │   ├── 00000000-0000-0000-0000-000000000001
7.  │   │   ├── 1.txt
8.  │   │   ├── a
9.  │   │   └── b
10. │   ├── 6eb86739-a0f7-4d20-91c8-af2030a6a712
11. │   │   ├── b-1.txt
12. │   │   └── b-2.txt
13. │   ├── ba93fe3a-c92d-44d6-a81e-a224fe27d1c1
14. │   │   ├── a-01.txt
15. │   │   └── a-2.txt
16. │    └── entry-changes-db686567-3b6f-4391-b7e7-3d0c1b20fe7
   b
17. └── xattrop
18.     ├── 00000000-0000-0000-0000-000000000001
19.     ├── 15644cf1-83c1-4807-9b70-d43198d7abc9
20.     ├── 2022bfa3-96cd-43c2-b9b8-e28425a13cca
21.     ├── 6eb86739-a0f7-4d20-91c8-af2030a6a712
22.     ├── 8740d0ae-63c7-4d6c-a8f1-5a07490e8865
23.     ├── 91cec102-72b7-424d-a936-0bf1e2683cc3
24.     ├── ba93fe3a-c92d-44d6-a81e-a224fe27d1c1
25.     ├── ecfe29ff-34bc-47c3-a03f-303443517c5b
26.     └── xattrop-db686567-3b6f-4391-b7e7-3d0c1b20fe7b
27.
28. 6 directories, 18 files
29.
```

```
30.
31. root@gfs03:~# tree   /glusterfs/test-index/.glusterfs/indi
    ces/
32. /glusterfs/test-index/.glusterfs/indices/
33. ├── dirty
34. │   └── dirty-9f366381-bb65-4a25-8bf5-dff349508ffc
35. ├── entry-changes
36. │   ├── 00000000-0000-0000-0000-000000000001
37. │   │   ├── 1.txt
38. │   │   ├── a
39. │   │   └── b
40. │   ├── 6eb86739-a0f7-4d20-91c8-af2030a6a712
41. │   │   ├── b-1.txt
42. │   │   └── b-2.txt
43. │   ├── ba93fe3a-c92d-44d6-a81e-a224fe27d1c1
44. │   │   ├── a-01.txt
45. │   │   └── a-2.txt
46. │   └── entry-changes-9f366381-bb65-4a25-8bf5-dff349508ff
    c
47. └── xattrop
48.     ├── 00000000-0000-0000-0000-000000000001
49.     ├── 15644cf1-83c1-4807-9b70-d43198d7abc9
50.     ├── 2022bfa3-96cd-43c2-b9b8-e28425a13cca
51.     ├── 6eb86739-a0f7-4d20-91c8-af2030a6a712
52.     ├── 8740d0ae-63c7-4d6c-a8f1-5a07490e8865
53.     ├── 91cec102-72b7-424d-a936-0bf1e2683cc3
54.     ├── ba93fe3a-c92d-44d6-a81e-a224fe27d1c1
55.     ├── ecfe29ff-34bc-47c3-a03f-303443517c5b
56.     └── xattrop-9f366381-bb65-4a25-8bf5-dff349508ffc
57.
58. 6 directories, 18 files
```

这里通过其他两个正常的目录可以看到这里 entry-changes 目录下其实记录的就是文件和目录的结构情况，也就是要进行自愈的文件结构，这里 1.txt 是在 brick 的根目录下创建的，因此这里的上层目录 gfid 就是最后为 1，其他全为 0 的。

接着这里还是再次强制启动 brick 的进程，也就是使用命令 start force，然后这里的目录就会恢复了。

```
root@gfs02:~# tree /glusterfs/test-index/.glusterfs/indices/
/glusterfs/test-index/.glusterfs/indices/
├── dirty
│   └── dirty-db686567-3b6f-4391-b7e7-3d0c1b20fe7b
├── entry-changes
│   └── entry-changes-db686567-3b6f-4391-b7e7-3d0c1b20fe7b
└── xattrop
    └── xattrop-db686567-3b6f-4391-b7e7-3d0c1b20fe7b

3 directories, 3 files
```

注意如果这里 entry-changes 的目录还是没有恢复的话，那么考虑是否 heal 进程有异常情况出现了，这里可以根据日志来排查一下，或者直接对 glusterd 进程进行重启，这里 heal 进程其实是由 glusterd 创建的。

通过这三个目录的实验，那么对于 AFR 来说，有了这里的内容，才好对文件进行更多的处理。

3.6.4. heal 连接数

这里有一个地方是非常值得注意的，那就是 heal 进程与所有的 bricks 都会相连的，使用 status clients 可以看到。哪怕这个 volume 的 brick 不是当前节点的，那么也会连接到 heal 进程上面，也就是说，每一个集群节点默认都有一个 glustershd 进程，然后这个进程会和所有的 bricks 进行连接，下面来感受一下。

```
root@gfs01:~# gluster volume info test-shd-num

Volume Name: test-shd-num
Type: Replicate
Volume ID: 7b40457b-846a-4ace-90a6-3143fdc11782
```

```
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x 3 = 3
Transport-type: tcp
Bricks:
Brick1: 10.0.12.2:/glusterfs/test-shd-num
Brick2: 10.0.12.9:/glusterfs/test-shd-num
Brick3: 10.0.12.12:/glusterfs/test-shd-num
Options Reconfigured:
cluster.granular-entry-heal: on
storage.fips-mode-rchecksum: on
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off


root@gfs01:~# gluster volume status  test-shd-num clients
Client connections for volume test-shd-num
----------------------------------------------
Brick : 10.0.12.2:/glusterfs/test-shd-num
Clients connected : 5
Hostname         BytesRead    BytesWritten      OpVersion
--------         ---------    ------------      ---------
10.0.12.9:49075     1480          1460         90000
10.0.12.2:48912     4972          5356         90000
10.0.12.12:49058    1868          2264         90000
10.0.12.3:49071     1252          968         90000
10.0.12.7:49062     1252          968         90000
----------------------------------------------
Brick : 10.0.12.9:/glusterfs/test-shd-num
Clients connected : 5
Hostname         BytesRead    BytesWritten      OpVersion
--------         ---------    ------------      ---------
10.0.12.9:49088     4732          5012         90000
10.0.12.2:48937     1720          1804         90000
10.0.12.12:49054    1716          1804         90000
10.0.12.3:49070     1252          968         90000
10.0.12.7:49061     1252          968         90000
----------------------------------------------
Brick : 10.0.12.12:/glusterfs/test-shd-num
Clients connected : 5
Hostname         BytesRead    BytesWritten      OpVersion
--------         ---------    ------------      ---------
```

```
10.0.12.12:49146      8372          9368          90000
10.0.12.3:49075       1252          968           90000
10.0.12.7:49074       1252          968           90000
10.0.12.9:49079       1484          1460          90000
10.0.12.2:48936       1720          1804          90000
-----------------------------------------------
```

从上面可以看出，这里 volume 的 brick 只是 2,9,12 三个节点，但是创建之后这里还有其他节点的连接，然后查看一下就可以知道是和 glustershd 的进程连接，那么这里根据官方的说法，这是一个待优化的项目，另外这里当一个节点有多个 glustershd 进程的时候，并不会增加连接数。

3.6.5. AFR

那么有了前面提到的三个目录，就可以准确知道哪些文件正在修改，并且哪些文件需要修复了，而修复和自愈,在 glusterfs 中有一个专门的算法叫做 AFR，不过在讲解 AFR 之前这里需要先了解一下对于一个写操作流程到底是怎样的。而在了解写操作，也就是涉及写入的 fop，我们需要区分一下 glusterfs 中常用的 fop，到底哪些 fop 涉及修改文件，哪些涉及读取的。

fop 类型	fop	备注
写入	afr_create afr_mknodafr_mkdir afr_link afr_symlink afr_rename afr_unlink afr_rmdir afr_do_writev afr_truncate	这里的 fop 都是会修改文件的。

	afr_ftruncate afr_setattr ， afr_fsetattr afr_setxattr afr_fsetxattr afr_removexattr afr_fremovexattr afr_fallocate afr_discard afr_zerofill afr_xattrop afr_fxattrop afr_fsync	
读取	afr_readdir afr_access afr_stat afr_fstat afr_readlink afr_getxattr afr_fgetxattr afr_readv afr_seek	这里的 fop 只是涉及到读取，不会涉及修改文件。

那么在知道了 fop 的一些分类之后 ,对于写入操作的 fop 流程是怎样的呢？这里其实是分为五个步骤的:

1) 锁定阶段: 在所有 brick 上锁定正在修改的文件，以便其他客户端的 AFR 在尝试同时修改同一文件时被阻止。
2) 预操作阶段: 在所有参与的 brick 上将 xattr 扩展属性（trusted.afr.dirty）增加 1，作为即将发生的 FOP 的指示（在下一阶段）,也就是记录正在操作。
3) fop 操作阶段: 这里 volume 的 bricks 执行 fop 操作。
4) 完成阶段: fop 执行完成之后,在执行成功的 brick 上第二步中的扩展属性增

加的数值减少 1 ，正常情况下，这时候扩展属性应该全为 0,证明文件执行成功了。

5) 解锁阶段：释放在阶段 1 中获取的锁。任何竞争客户端现在都可以继续进行自己的写入事务。

那么通过这几个步骤可以知道，一旦文件写入异常，也就是在扩展属性上面是可以看到的，下面做一个小的测试来感受一下。

```
root@gfs01:/mnt/test-afr# gluster volume info test-afr

Volume Name: test-afr
Type: Replicate
Volume ID: 6637cea5-2ead-4c3f-a319-9ac7e32d3fd1
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x 3 = 3
Transport-type: tcp
Bricks:
Brick1: 10.0.12.2:/glusterfs/test-afr
Brick2: 10.0.12.9:/glusterfs/test-afr
Brick3: 10.0.12.12:/glusterfs/test-afr
Options Reconfigured:
cluster.granular-entry-heal: on
storage.fips-mode-rchecksum: on
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

这里先创建一个 volume 然后挂载，接着创建一个文件 1.txt，接着 kill 掉其中一个 brick 进程，再对 1.txt 文件使用进行修改，这里使用命令 date > 1.txt 导入时间，这时候对比一下正常和非正常的 brick 下的 1.txt 文件属性

```
//下面是正常 brick
root@gfs02:~# getfattr -d -m . -e hex /glusterfs/test-afr/1.txt
getfattr: Removing leading '/' from absolute path names
# file: glusterfs/test-afr/1.txt
```

```
trusted.afr.dirty=0x000000000000000000000000
trusted.afr.test-afr-client-1=0x000000020000000000000000
trusted.gfid=0x21ea642138cc475e850a8bc7e1c5cc5c
trusted.gfid2path.46104af11fea85fe=0x30303030303030302d303030302d303030302d303030302d3030303030303030303030312f312e747874
trusted.glusterfs.mdata=0x010000000000000000000000060e54082000000001b967e1d0000000060e54082000000001b967e1d0000000060e5405b000000003b3d42a9

root@gfs01:/mnt/test-afr# getfattr -d -m . -e hex /glusterfs/test-afr/1.txt
getfattr: Removing leading '/' from absolute path names
# file: glusterfs/test-afr/1.txt
trusted.afr.dirty=0x000000000000000000000000
trusted.afr.test-afr-client-1=0x000000020000000000000000
trusted.gfid=0x21ea642138cc475e850a8bc7e1c5cc5c
trusted.gfid2path.46104af11fea85fe=0x30303030303030302d303030302d303030302d303030302d3030303030303030303030312f312e747874
trusted.glusterfs.mdata=0x010000000000000000000000060e54082000000001b967e1d0000000060e54082000000001b967e1d0000000060e5405b000000003b3d42a9

//这里是异常的 brick
root@gfs03:~# getfattr -d -m . -e hex /glusterfs/test-afr/1.txt
getfattr: Removing leading '/' from absolute path names
# file: glusterfs/test-afr/1.txt
trusted.gfid=0x21ea642138cc475e850a8bc7e1c5cc5c
trusted.gfid2path.46104af11fea85fe=0x30303030303030302d303030302d303030302d303030302d3030303030303030303030312f312e747874
trusted.glusterfs.mdata=0x010000000000000000000000060e5405b000000003b3d42a90000000060e5405b000000003b3d42a90000000060e5405b000000003b3d42a9
```

从这里可以看到，首先异常的 brick 这里没有显示扩展属性 trusted.afr.dirty，接着正常的 brick 的文件扩展属性，这里的扩展属性 trusted.afr.test-afr-client-1，这里表示该 volume 名为 test-afr 的第二个 brick 出现异常了，接着可以查看一下 heal info 信息来核实一下。

```
root@gfs02:~# gluster volume heal test-afr info
Brick 10.0.12.2:/glusterfs/test-afr
/1.txt
Status: Connected
Number of entries: 1

Brick 10.0.12.9:/glusterfs/test-afr
Status: Transport endpoint is not connected
Number of entries: -

Brick 10.0.12.12:/glusterfs/test-afr
/1.txt
Status: Connected
Number of entries: 1
```

这里就指出了其中一个 brick 是没有连接上的 ,那么这里如果强制启动的话，再使用 heal 命令则可以修复该问题了。

那么下面再来看另外一种测试。

```
root@gfs01:/mnt/test-afr# dd if=/dev/zero of=2.txt bs=1k count=1000000

root@gfs03:~# gluster volume status test-afr
Status of volume: test-afr
Gluster process                         TCP Port  RDMA Port  Online  Pid
------------------------------------------------------------------------------
Brick 10.0.12.2:/glusterfs/test-afr     49167     0          Y       3616131
Brick 10.0.12.9:/glusterfs/test-afr     49169     0          Y       3764494
Brick 10.0.12.12:/glusterfs/test-afr    49166     0          Y       3676444
Self-heal Daemon on localhost           N/A       N/A        Y       2515251
Self-heal Daemon on 10.0.12.3           N/A       N/A        Y       15298
Self-heal Daemon on 10.0.12.7           N/A       N/A        Y       15526
```

```
Self-heal Daemon on gfs02               N/A       N/A        Y       2
636830
Self-heal Daemon on gfs01               N/A       N/A        Y       1
28017

Task Status of Volume test-afr
------------------------------------------------------------------------------
There are no active volume tasks

root@gfs03:~# kill -9 3764494
root@gfs03:~# ls /glusterfs/test-afr/
2.txt
```

这里首先恢复启动 volume,然后使用 dd 命令在挂载目录下创建一个比较大的文件，而在创建过程中，还是把其中一个 brick 进程使用 kill 命令销毁，接着终止 dd 命令，查看一下正常和异常 brick 的文件扩展属性。

```
//下面是正常 brick
root@gfs01:/mnt/test-afr# getfattr -d -m . -e hex /glusterfs/tes
t-afr/2.txt
getfattr: Removing leading '/' from absolute path names
# file: glusterfs/test-afr/2.txt
trusted.afr.dirty=0x000000000000000000000000
trusted.afr.test-afr-client-1=0x000000020000000000000000
trusted.gfid=0x11a0da6539874e87980399ab64552333
trusted.gfid2path.8954a07f86db8340=0x3030303030303030
2d303030302d303030302d303030302d30303030303030303
03030312f322e747874
trusted.glusterfs.mdata=0x010000000000000000000000006
0e5452300000001006ff9a0000000060e5452300000000100
6ff9a0000000060e544e40000000034ad3322

root@gfs01:/mnt/test-afr# du -sh /glusterfs/test-afr/2.txt
346M    /glusterfs/test-afr/2.txt

root@gfs02:~# getfattr -d -m . -e hex /glusterfs/test-afr/2.txt
getfattr: Removing leading '/' from absolute path names
# file: glusterfs/test-afr/2.txt
trusted.afr.dirty=0x000000000000000000000000
trusted.afr.test-afr-client-1=0x000000020000000000000000
```

```
trusted.gfid=0x11a0da6539874e87980399ab64552333
trusted.gfid2path.8954a07f86db8340=0x3030303030303030
2d303030302d303030302d303030302d303030303030303030
03030312f322e747874
trusted.glusterfs.mdata=0x010000000000000000000000006
0e5452300000001006ff9a0000000060e5452300000000100
6ff9a0000000060e544e40000000034ad3322

root@gfs02:~# du -sh /glusterfs/test-afr/2.txt
346M    /glusterfs/test-afr/2.txt


//下面是异常 brick 的
root@gfs03:~# du -sh /glusterfs/test-afr/2.txt
67M /glusterfs/test-afr/2.txt

root@gfs03:~# getfattr -d -m . -e hex /glusterfs/test-afr/2.txt
getfattr: Removing leading '/' from absolute path names
# file: glusterfs/test-afr/2.txt
trusted.afr.dirty=0x000000010000000000000000
trusted.gfid=0x11a0da6539874e87980399ab64552333
trusted.gfid2path.8954a07f86db8340=0x3030303030303030
2d303030302d303030302d303030302d303030303030303030
03030312f322e747874
trusted.glusterfs.mdata=0x010000000000000000000000006
0e544f000000000170e52b00000000060e544f000000000170
e52b00000000060e544e40000000034ad3322
```

从这里上面的文件扩展属性可以留意到 trusted.afr.dirty 中，异常的 brick 中间有一个数字为 1 的，而正常的 brick 都是 0 的，这个也就是和上面提到的五个步骤呼应的，这里因为异常的 brick 的操作还没完成但是被异常销毁了，因此并没有执行到第四步，所以还会保留该数值的。那么接着我们再来看看 indices 目录的情况是如何的。

```
//下面是正常 brick
root@gfs03:~# tree  /glusterfs/test-afr/.glusterfs/indices/
/glusterfs/test-afr/.glusterfs/indices/
```

```
├── dirty
│   ├── 11a0da65-3987-4e87-9803-99ab64552333
│   └── dirty-b14af3d5-3645-4530-bf53-837807620b2f
├── entry-changes
└── xattrop

3 directories, 2 files


root@gfs02:~# tree  /glusterfs/test-afr/.glusterfs/indices/
/glusterfs/test-afr/.glusterfs/indices/
├── dirty
│   └── dirty-87eb2ed4-e5da-48f1-8367-9b00177ff9df
├── entry-changes
└── xattrop
    ├── 11a0da65-3987-4e87-9803-99ab64552333
    └── xattrop-87eb2ed4-e5da-48f1-8367-9b00177ff9df

3 directories, 3 files


//下面是异常 brick 的
root@gfs03:~# tree  /glusterfs/test-afr/.glusterfs/indices/
/glusterfs/test-afr/.glusterfs/indices/
├── dirty
│   ├── 11a0da65-3987-4e87-9803-99ab64552333
│   └── dirty-b14af3d5-3645-4530-bf53-837807620b2f
├── entry-changes
└── xattrop

3 directories, 2 files
```

下面来关注了解一下 trusted.afr.dirty 这个文件扩展属性的数字规律。这里的数字是前 12 位和文件异常有关的，每 4 位作为一个分类，其中前 4 位是和数据有关的，接着 4 位是和元数据有关的，最后 4 位和 entry 目录有关，也就是说，可能在操作文件的过程中异常也是分为三种异常的，有些异常是修改了文件的属性，但是并没有修改数据，而有些是修改了数据的等等，因此这里修复。

那么这里什么时候发生自愈修复呢？一方面每个节点默认是有一个glustershd 进程的，这个进程在一段时间内检查是否需要修复的，当然也可以触发命令进行修复，执行如下命令。

```
1. root@gfs02:~# gluster volume heal test-afr
2. Launching heal operation to perform index self heal on volume
   test-afr has been unsuccessful:
3. Glusterd Syncop Mgmt brick op 'Heal' failed. Please check glus
   tershd log file for details.
```

那么提到了自愈的问题，脑裂也是可能出现的，所谓的脑裂，如一个 3 副本的复制卷，这里其中超过半数的文件属性不一致或者出现异常的时候，就会有脑裂出现了，而这时候，通过正常的 heal 可能无法修复问题，那么这时候使用 heal info 命令，可以看到有些文件会提示 Is in split-brain，也就是存在脑裂风险。那么这里修复的方式有多种，其中可以以最新的时间作为修复，也就是使用如下命令。

```
1. sudo gluster volume heal  {VOLUME-NAME} split-brain latest-
   mtime  {FILE-NAME}
```

这里最后的 FILE-NAME 就是 heal info 命令中提示存在脑裂的文件名称，然后执行可以修复，当然这里还有可以根据 bigger-file 最大的文件，source-brick 自行选择恰当的 brick 来修复等。

章节语:

这一章的内容很多，从 gfid 到数据内存模型再到 AFR 这些，涉及到 glusterfs 很多核心的概念和模块，而不同的模块作用不同，在阅读理解的时候，如果对于部分模块的细节不是很理解，可以考虑先放下，先宏观理解一下 glusterfs 的整体架构和其中部分核心内容，接着再慢慢去理解不同的细节。