环境:Centos7.8

需要安装的包:

```
[root@ks0 ~]# rpm -qa --last | grep "1[2,3] Oct 2020" | awk '{print $1}'
coreutils-8.22-24.el7.x86_64
coreutils-debuginfo-8.22-24.el7.x86_64
libdwarf-debuginfo-20130207-4.el7.x86_64
libdwarf-tools-20130207-4.el7.x86_64
libdwarf-static-20130207-4.el7.x86_64
libdwarf-devel-20130207-4.el7.x86_64
libxml2-debuginfo-2.9.1-6.el7.4.x86_64
xz-debuginfo-5.2.2-1.el7.x86_64
userspace-rcu-debuginfo-0.7.16-1.el7.x86_64
libaio-debuginfo-0.3.109-13.el7.x86_64
attr-debuginfo-2.4.46-13.el7.x86_64
acl-debuginfo-2.2.51-15.el7.x86_64
util-linux-debuginfo-2.23.2-63.el7.x86_64
openssl-debuginfo-1.0.2k-19.el7.x86_64
zlib-debuginfo-1.2.7-18.el7.x86_64
krb5-debuginfo-1.15.1-46.el7.x86_64
sssd-debuginfo-1.12.2-58.el7.x86_64
nss-softokn-debuginfo-3.44.0-8.el7_7.x86_64
pam-debuginfo-1.1.8-23.el7.x86_64
libsepol-debuginfo-2.5-10.el7.x86_64
keyutils-debuginfo-1.5.8-3.el7.x86_64
e2fsprogs-debuginfo-1.42.9-17.el7.x86_64
libselinux-debuginfo-2.5-15.el7.x86_64
pcre-debuginfo-8.32-17.el7.x86_64
libverto-debuginfo-0.2.5-4.el7.x86_64
glibc-debuginfo-2.17-307.el7.1.x86_64
glibc-debuginfo-common-2.17-307.el7.1.x86_64
gpg-pubkey-b6792c39-53c4fbdd
gcc-debuginfo-4.8.5-39.el7.x86_64
gcc-base-debuginfo-4.8.5-39.el7.x86_64
kernel-debuginfo-3.10.0-229.el7.x86_64

使用debuginfo 安装
debuginfo-install -y glib libdwarf gcc libaio libverto libselinux e2fsprogs zlib libxml2
userspace-rcu attr coreutils util-linux acl openssl zlib krb5 sssd nss pam libsepol
```

安装systemtap

```
 yum install systemtap systemtap-runtime
 stap-prep
 kernel-debuginfo
```

```
kernel-debuginfo-common
kernel-devel

查询包地址: http://rpm.pbone.net
```

探测遇到错误:

- 探测函数名写错

```
$semantic error: while resolving probe point: identifier 'process' at mem.stap:12:7
        source: probe process("/usr/lib64/xxx.so").function("xxx_func").return {
                      ^

semantic error: no match (similar functions: xxx_func)

跟踪的进程, 或者so中没有"xxx_func"的函数, 可以使用 readelf -a /usr/lib64/xxx.so  | grep
xxx_func
```

- 参数错误

```
ERROR: probe overhead exceeded threshold

解决 -DSTP_NO_OVERLOAD
```

```
ERROR: Skipped too many probes, check MAXSKIPPED or try again with stap -t for more
details.

解决: -DMAXSKIPPED=102400
```

```
ERROR: Array overflow, check MAXMAPENTRIES near identifier....

解决: -DMAXMAPENTRIES=1024000
```

man stap可以看到RESOURCE LIMITS这一栏下面有很多资源配置的参数。

```
MAXNESTING
Maximum number of nested function calls. Default determined by script analysis, with a
bonus 10 slots added for recursive scripts.

MAXSTRINGLEN
Maximum length of strings, default 128.

MAXTRYLOCK
Maximum number of iterations to wait for locks on global variables before declaring
possible deadlock and skipping the probe, default 1000.

MAXACTION
Maximum number of statements to execute during any single probe hit (with interrupts
disabled), default 1000.

MAXACTION_INTERRUPTIBLE
Maximum number of statements to execute during any single probe hit which is executed
with interrupts enabled (such as begin/end probes),
default (MAXACTION * 10).

MAXMAPENTRIES
Maximum number of rows in any single global array, default 2048.

MAXERRORS
Maximum number of soft errors before an exit is triggered, default 0, which means that
the first error will exit the script.

MAXSKIPPED
Maximum number of skipped probes before an exit is triggered, default 100. Running
systemtap with -t (timing) mode gives more details about
skipped probes. With the default -DINTERRUPTIBLE=1 setting, probes skipped due to
reentrancy are not accumulated against this limit.

MINSTACKSPACE
Minimum number of free kernel stack bytes required in order to run a probe handler,
default 1024. This number should be large enough for
the probe handler′ s own needs, plus a safety margin.

MAXUPROBES
Maximum number of concurrently armed user-space probes (uprobes), default somewhat larger
than the number of user-space probe points named
in the script. This pool needs to be potentialy large because individual uprobe objects
(about 64 bytes each) are allocated for each
process for each matching script-level probe.
STP_MAXMEMORY
Maximum amount of memory (in kilobytes) that the systemtap module should use, default
unlimited. The memory size includes the size of the
module itself, plus any additional allocations. This only tracks direct allocations by
the systemtap runtime. This does not track indirect
allocations (as done by kprobes/uprobes/etc. internals).

TASK_FINDER_VMA_ENTRY_ITEMS
Maximum number of VMA pages that will be tracked at runtime. This might get exhausted for
system wide probes inspecting shared library vari-
ables and/or user backtraces. Defaults to 1536.

STP_PROCFS_BUFSIZE
Size of procfs probe read buffers (in bytes). Defaults to MAXSTRINGLEN. This value can be
overridden on a per-procfs file basis using the
procfs read probe .maxsize(MAXSIZE) parameter.
```

使用技巧

被跟踪进程需要的条件

```
1、编译需要-g（带上debug信息）

2、同时被跟踪的进程或者so中需要有debug segment
[root@ks1 ~]# readelf -a /usr/local/lib/glusterfs/7.7/xlator/cluster/ec.so  | grep debug
  [26] .debug_aranges     PROGBITS         0000000000000000  00087295
  [27] .debug_info        PROGBITS         0000000000000000  000876d5
  [28] .debug_abbrev      PROGBITS         0000000000000000  0016376b
  [29] .debug_line        PROGBITS         0000000000000000  0016acdc
  [30] .debug_str         PROGBITS         0000000000000000  00185a76
  [31] .debug_loc         PROGBITS         0000000000000000  0019561d
  [32] .debug_ranges      PROGBITS         0000000000000000  0024fd2f
```

gluster的进程说明

```
glusterfs：client端挂载server端的卷以及在client端的相关操作
glusterd：Gluster elastic volume management daemon。glusterd与卷管理有关系，而且代码集中在xlators/mgmt/glusterd/src下面。
glusterfsd：start a glusterfs server，server端的进程启动。
gluster：Gluster Concole Manager，to run the program and display gluster prompt
```

跟踪client

```
挂载client
mount.glusterfs ks0:test_volume /mnt/test_volume log_level=trace


确定client进程：
[root@ks1 systemtap]# lsof /var/log/glusterfs/mnt-test_volume.log
COMMAND    PID USER   FD   TYPE DEVICE SIZE/OFF     NODE NAME
glusterfs 6078 root    5w   REG    8,3 16031457 71621323 /var/log/glusterfs/mnt-test_volume.log

[root@ks1 systemtap]# ps axu | grep 6078
root      6078  2.4  1.0 640464  8064 ?        SLsl 11:33   0:01 /usr/local/sbin/glusterfs --process-name fuse --volfile-server=ks0 --volfile-id=test_volume /mnt/test_volume

由于/usr/local/sbin/glusterfs中并没有全部需要跟踪的符号，被跟踪的符号在动态库中
[root@ks1 systemtap]#  cat /proc/6078/maps  | grep gluster | awk '{print $6}' | sort | uniq -c
      2 /usr/local/libexec/glusterfs/ec-code-dynamic.m6MMli
      4 /usr/local/lib/glusterfs/7.7/rpc-transport/socket.so
      4 /usr/local/lib/glusterfs/7.7/xlator/cluster/dht.so
      4 /usr/local/lib/glusterfs/7.7/xlator/cluster/ec.so
      4 /usr/local/lib/glusterfs/7.7/xlator/debug/io-stats.so
      4 /usr/local/lib/glusterfs/7.7/xlator/features/utime.so
      4 /usr/local/lib/glusterfs/7.7/xlator/meta.so
      4 /usr/local/lib/glusterfs/7.7/xlator/mount/fuse.so
      4 /usr/local/lib/glusterfs/7.7/xlator/performance/io-cache.so
      4 /usr/local/lib/glusterfs/7.7/xlator/performance/io-threads.so
      4 /usr/local/lib/glusterfs/7.7/xlator/performance/md-cache.so
      4 /usr/local/lib/glusterfs/7.7/xlator/performance/open-behind.so
      4 /usr/local/lib/glusterfs/7.7/xlator/performance/quick-read.so
      4 /usr/local/lib/glusterfs/7.7/xlator/performance/read-ahead.so
      4 /usr/local/lib/glusterfs/7.7/xlator/performance/readdir-ahead.so
      4 /usr/local/lib/glusterfs/7.7/xlator/performance/write-behind.so
      4 /usr/local/lib/glusterfs/7.7/xlator/protocol/client.so
      4 /usr/local/lib/libglusterfs.so.0.0.1
      3 /usr/local/sbin/glusterfsd

所以需要跟踪的是动态库，可以更具需要跟踪所需的so
[root@ks1 systemtap]# cat glusrter_trace.stp
probe process("/usr/local/lib/glusterfs/7.7/xlator/cluster/ec.so").function("*").call,
      process("/usr/local/lib/glusterfs/7.7/xlator/meta.so").function("*").call,
      process("/usr/local/lib/glusterfs/7.7/xlator/mount/fuse.so").function("*").call,

process("/usr/local/lib/glusterfs/7.7/xlator/protocol/client.so").function("*").call,
      process("/usr/local/lib/libglusterfs.so.0.0.1").function("*").call,
      process("/usr/local/sbin/glusterfsd").function("*").call
{
   printf("%s -> %s, pid:%d, tid:%d\n", thread_indent(4), ppfunc(), pid(), tid());
}


probe process("/usr/local/lib/glusterfs/7.7/xlator/cluster/ec.so").function("*").return,
      process("/usr/local/lib/glusterfs/7.7/xlator/meta.so").function("*").return,
      process("/usr/local/lib/glusterfs/7.7/xlator/mount/fuse.so").function("*").return,

process("/usr/local/lib/glusterfs/7.7/xlator/protocol/client.so").function("*").return,
      process("/usr/local/lib/libglusterfs.so.0.0.1").function("*").return,
      process("/usr/local/sbin/glusterfsd").function("*").call
{
   thread_indent(-4);
}


开始跟踪：
stap -d /usr/local/sbin/glusterfs -x `lsof /var/log/glusterfs/mnt-test_volume.log  | grep
-v COMMAND | awk '{print $2}'` -v glusrter_trace.stp  > /tmp/systemtap_trace.txt 2>&1
```

---

gluster文章

- gluster EC说明
- gluster Developer-guide
- Glusterfs hacker guide 1
- Glusterfs hacker guide 2
- Glusterfs hacker guide 3
- GlusterFS的数据分布(DHT)和文件副本(AFR)机制
- gluster浅析

- 刘爱贵 gluster分析
- GlusterFS技术概要分析
- 深入理解GlusterFS之数据均衡
- 换个视角深入理解GlusterFS，GlusterFS缺点分析
- 深入理解GlusterFS之POSIX接口
- Gluster Disperse Volume Troubleshooting – HEAL
- Glusterfs DHT(hash分布)源代码分析
- GlusterFS数据存储脑裂修复方案最全解析
- GlusteFS:自我修复（Selfheal）源码分析