UDisk 迁移方案及部署

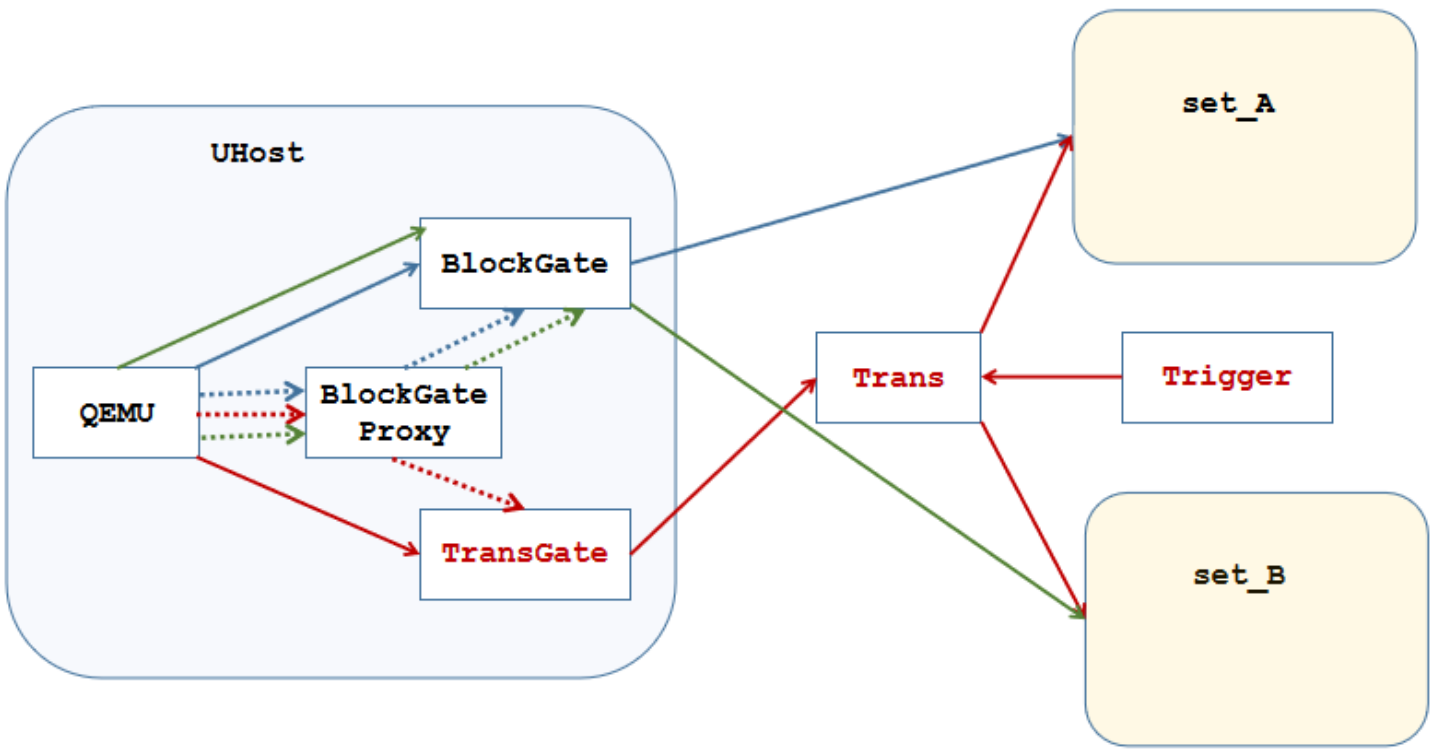
Last edited by **alan.ding** 3 weeks ago

第一部分 原理篇

目的

UDisk V4跨set迁移并屏蔽底层存储的影响, (HDD/SSD, RAW/TCP)之间可以相互迁移.

架构图



图中蓝色线表示迁移前数据流；图中红色线表示迁移中数据流；图中绿色线表示迁移后数据流。
BlockGateProxy会根据是否有迁移任务决策挂载到BlockGate或者TransGate,具体流程在 ''' 迁移流程''' 模块详述.

模块设计

新模块

- TransGate: 负责将qemu请求透传给trans模块。
- Trans： 模拟两个set的block_gate功能，负责读写IO和迁移IO。
- Trigger： 负责迁移的控制,包括迁移任务创建,触发需要迁移的IO给Trans等。
  代码路径: git@git.ucloudadmin.com:block/migrate_udiskv4.git

表设计

复用udisk mongodb。
库中增加t_migrate_udisk_task表，表结构如下：

```
extern_id: string, 盘外部id
size: int, 盘大小，单位GB
trans_ip: string，迁移trans进程ip
trans_port: int, 迁移trans进程port
status：int, 盘迁移状态，8位整数。-1:任务不存在 0:初始默认状态 1:搬迁中 2:成功搬迁
finish_offset：int64, 已经成功搬迁的位置, 单位Byte，即相对盘起始位置的偏移
start_time: int， 迁移开始时间，unixtimestamp
end_time： int， 迁移结束时间，unixtimestamp
```

迁移流程

> 以迁移盘bs-test为例，假设该盘在set1000，迁移到新set3000。

1、创建bs-test迁移任务:

- 1.1 trigger收到迁移请求，向access发送请求在set3000创建新盘bs-test_new
  此时，access表中t_lc_extent_info中映射为：

```
bs-test:  1000
bs-test-new:  3000
```

- 1.2 在t_migrate_udisk_task表里插入迁移任务，初始状态为0: 开始搬迁. 表明bs-test处于迁移中，以供BlockGateProxy决策选择TransGate.

2、 获取bs-test的挂载状态。

- bs-test处于挂载中
  a、 发送ReloginRequest给BlockGate,断开BlockGate和Qemu的连接。
  b、 Qemu触发重连，通过BlockGateProxy连接到TransGate。
  c、 TransGate将login请求转发给Trans。
  d、 Trans模拟BlockGate的login,分别挂载到set1000和set3000。
  e、 Trans login 结束后，通知Trigger连接自己。（确保同一Trans的同一线程处理TransGate和Trigger的请求）
- bs-test没有挂载
  a、 通知Trigger模拟Qemu向Trans发送login请求。
  b、 Trans模拟BlockGate的login,分别挂载到set1000和set3000。

> 注意：迁移过程中不允许扩容，在access中实现。

3、bs-test开始搬迁

- 3.1 trigger触发搬迁。
  a. 从盘头到盘尾按照1M TPC分片依次向Trans发送搬迁请求。该请求对于trans来说算一次IO，该IO会从set1000读，向set3000写。
  b. 迁移完一个分片，开始发送该分片的CHECK请求，Trans会读取set1000/set3000数据对比MD5。
- 3.2 用户IO经过TransGate发往Trans. 读请求Trans会读取set1000；写请求Trans会双写，既写set1000/set3000。
- 3.3 Trans会接收两类IO请求，一类是Trigger发送的Migrate/Check请求；一类是TransGate转发的用户读写请求。trans会无差别的对待该两类请求。Trans维护两个队列，pending list和inflight list。当IO请求到来时，如果和inflight list IO有重叠或者inflight list已到最大值，则放入pending list，等待下次处理。
- 3.4 Trigger每完成一次搬迁请求，会更新该盘数据库中目前搬迁位置，防止Trigger重启丢失进度。
- 3.5 Trigger搬迁完时，会修改t_migrate_udisk_task中bs-test状态置为迁移结束，以供BlockGateProxy决策选择BlockGate。

4、 bs-test搬迁结束

- 4.1 Trigger迁移bs-test结束，会通知access修改旧盘和新盘信息，包括:
  a. 通知set3000 metaserver依据bs-test更新bs-test-new信息。将bs-test-new extern_id 改为bs-test。
  c. 通知set1000 metaserver将bs-test extern_id 改为bs-test_old。
  b. 修改t_lc_extent_info表, bs-test变为bs-test_old，bs-test_new变为bs-test。

```
bs-test_old: 1000
bs-test: 3000
```

- 4.2 Trigger通知Trans断开和TransGate连接，从而断开Qemu和TransGate连接。
- 4.3 Qemu通过BlockGateProxy连接到BlockGate，从而迁移结束。

第二部分 部署篇

TransGate

https://git.ucloudadmin.com/mikasa/cc/tree/arvin.zhang/deploy/set/usre_udisk_playbook/roles/transgate-install
部署在宿主机上，每台宿主机一个.

Trans

https://git.ucloudadmin.com/mikasa/cc/tree/arvin.zhang/deploy/set/usre_udisk_playbook/roles/trans-install
每个机房部署一批,一般部署在set1000的所有chunk上.

Trigger

https://git.ucloudadmin.com/mikasa/cc/tree/arvin.zhang/deploy/set/usre_udisk_playbook/roles/trigger-install
每个机房部署一个,一般部署在第一个access所在的虚机上.

第三部分 迁移工具篇

迁移工具在limax中: /root/limax/auto_migrate/new_migrate/udisk_migrate_tool_v2.py
使用方法:

```
Usage: python udisk_migrate_tool_v2.py [-h] [-f] <-c file> <-d disk_id>
例如:
python udisk_migrate_tool_v2.py -c conf/migrate_HB05.cfg -d bsi-o5qdd2dm
```

其中-d表示对应的udisk. -c指定配置文件,内容为:

```
[common]
region = hb05          ## 迁移盘所在region
target_set = 3013      ##  迁移目标set

[network]

[log]
log_path = /var/log/udisk/migrate/  # 迁移日志路径,产生文件为<extern_id>.log
log_level = debug
```

迁移任务主要流程:

```
1. CheckUbsState. 通过access获取迁移盘相关信息,status非0状态的盘直接返回error.
2. MigrateTaskCreate. 向trigger发送请求,创建迁移任务.
3. UDiskRelogin. 向BlockGate发送relogin请求，BlockGate重新login时，因为存在迁移任务，会挂载到Trans上
4. MigrateTaskExecute. 执行迁移任务.
5. 监控迁移进度. 查找t_migrate_udisk_task中迁移盘的finish_offset并输出进度. 迁移状态为MIGRATE_COMPL
   trigger迁移完成后，会发送LogoutRequest, Trans断开与TransGate的链接，触发Qemu重新挂载.
6. CheckResult. 检查迁移盘是否已经在target_set.
7. 迁移完成. 输出<extern_id> migrate success.
```

迁移常见故障及处理

(故障处理流程中，以udisk=bs-migrate,从set1000迁移到set3000为例.access对应的mongodb库为access_db)

1. relogin失败(会有电话告警)

原因: 一般是由于udisk挂载的虚机对应的宿主机没有升级BlockGate,或者是云盘本身状态异常(多次卸载挂载等导致挂载状态异常)
解决方法:
这种情况下,还没有开始迁移,但是迁移任务已经创建好了,需要做如下回退操作(以bs-migrate为例):
a. 删除迁移任务:
在access_db中,db.t_migrate_udisk_task.find({extern_id:"bs-migrate"}),查找到迁移任务，确认finish_offset=0,表示还没有开始迁移. 直接删除.
b. 删除中间状态云盘bs-migrate_new
b.0 调用pb接口删除bs-migrate_new，并触发回收
b.1 回收结束后，在access_db t_lc_extern_info中删除bs-migrate_new
b.2 回收结束后，在set3000_db t_lc_info中删除bs-migrate_new
c. 重启set3000 metaserver: 因为metaserver会缓存所有lc信息,需要重启后重新load db.
d. 升级宿主机上BlockGate,之后重新迁移即可.
a-b步骤自动化脚本: limax/auto_migrate/new_migrate/tools/migrate_relogin_rollback.py 该脚本需要修改，否则chunk会有脏pc

2. 迁移结束,更新db失败(有电话告警，以及BlockGate的挂载失败告警)

原因:一般是由于MongoDB更新失败,需要手动更新DB
a. 确认迁移任务结束.
在access_db中,db.t_migrate_udisk_task.find({extern_id:"bs-migrate"}),查找到迁移任务，确认status=MIGRATE_COMPLETE,迁移已经结束.
b. 在set1000 db中,把t_lc_info中bs-migrate更新成bs-migrate_old
c. 在set3000 db中,把t_lc_info中bs-migrate_new更新成bs-migrate
d. 在access_db, 把t_lc_extern_info里, bs-migrate修改成bs-migrate_old, bs-migrate_new修改成bs-migrate.
自动化脚本: limax/auto_migrate/new_migrate/tools/migrate_finish_dbissue_successor.py

3. migrate queue full or exist

原因: 一般是由于超过trigger队列深度
解决方法:
这种情况下,还没有开始迁移,但是迁移任务已经创建好了，且挂载到了trans上,需要做如下回退操作(以bs-migrate为例):
a. 删除迁移任务:
在access_db中,db.t_migrate_udisk_task.find({extern_id:"bs-migrate"}),查找到迁移任务，确认finish_offset=0,表示

还没有开始迁移. 直接删除.
b. 重启trans_gate，使其挂到block_gate。
c. 删除中间状态云盘bs-migrate_new
c.0 调用pb接口删除bs-migrate_new，并触发回收
c.1 回收结束后，在access_db t_lc_extern_info中删除bs-migrate_new
c.2 回收结束后，在set3000_db t_lc_info中删除bs-migrate_new
d. 重启set3000 metaserver: 因为metaserver会缓存所有lc信息,需要重启后重新load db.
a-b步骤自动化脚本: limax/auto_migrate/new_migrate/tools/migrate_relogin_rollback.py 该脚本需要修改，否则chunk会有脏pc

同一个盘多次迁移

即bs-migrate从set1000迁移到set3000, 之后再从set3000迁移到set3001, 需要:
a. 在access_db中,db.t_migrate_udisk_task.find({extern_id:"bs-migrate"}),查找到迁移任务,删除迁移任务.
b. 在access_db中, 把bs-migrate_old删除.
c. 方舟删除bs-migrate_new.

源盘删除机制

目前为了防止迁移异常导致客户数据问题,迁移完成后没有立刻删除源盘,一般是迁移完1-2周，客户使用没有异常，调用pb命令删除原盘.

查找源盘

limax/udisk/tool/migrate_find_old.py可以查找某个机房内的所有源盘(_old结尾的盘).

删除源盘

直接发送删除命令给access即可。注意调用的删除盘是_old结尾的源盘.

迁移限制

1. 每个源端set只允许3个并发迁移(Trigger中实现,已经完成).
2. 支持多个set并行迁移(已经支持,验证完成).
3. trigger迁移并发度控制: migrate_iodepth,可以通过配置文件配置,默认为1.

03.31问题追踪

1. 迁移流程阐述
   owner: mamo.yang
   进展: 完成
   已经在文档上补充，详细流程在"迁移流程"模块

2. limax记录部署trans/trigger
   owner: arvin.zhang
   进展: 进行中
   根据机房部署逐步完成部署记录.

3. 迁移日志放在/data目录,日志后缀修改为migrate.
   owner: mamo.yang
   进展: 完成
   已经在迁移脚本和配置中更新.

4. t_migarate_task加索引
   owner: arvin.zhang
   进展: 完成
   在对应mongodb库执行db.t_migrate_udisk_task.createIndex({extern_id:1})即可.

5. 迁移close链接说明
   owner: mamo.yang
   进展: 完成
   已经在迁移脚本流程中说明.

6. 删除中间状态信息，脚本化
   owner: mamo.yang
   进展: 完成
   limax/auto_migrate/new_migrate/tools/migrate_relogin_rollback.py

7. 监控迁移进度，没有变化则告警
   owner: mamo.yang
   进展: 完成
   udisk_migrate_tool_v2.py中连续2min无进展,调用FinishMigrateTask请求结束任务，并告警.

8. access/metaserver/trigger/db超时时间10s
   owner: arvin.zhang
   进展: 进行中
   扫描修改配置 db_timeout = 10即可.

9. 源盘删除使用运营系统
   owner: han.jia
   进展: 进行中

10. 之前丢数据问题梳理
    owner: mamo.yang
    进展: 完成
    临时修改代码，不执行relogin迁移，导致挂载的盘按照没有挂载的方式迁移，迁移完成不会触发TransGate断开/qemu连接重新挂载.
    在此期间盘一直挂载在老架构，读写正常. 如果触发了重新连接，从开始迁移到重连，这段时间写数据丢失.
    目前不会禁用relogin功能,不会触发该问题.

11. 迁移回滚/终止
    owner: mamo.yang
    进展: 完成.
    trigger提供FinishMigrateUDiskTask接口,结束迁移任务并更新db中extern_id迁移状态为无效
    工具参见： /root/limax/auto_migrate/new_migrate/finish_migrate_task.py

12. 测试trans跑多个盘
    owner: mamo.yang
    进展: 完成
    并发迁移5个实例, 一共部署172.27.34.173，172.27.34.174两个trans.
    image

13. 跨机房挂载
    owner: mamo.yang
    进展: 完成
    access已经支持跨机房挂载功能, 迁移指定云盘所在机房的access即可.

14. 连续2min迁移无进展，直接触发任务终止
    owner: mamo.yang
    进展: 完成
    udisk_migrate_tool_v2.py中连续2min无进展,调用FinishMigrateTask请求结束任务，并告警.