dm-log-writes

This target takes 2 devices, one to pass all IO to normally, and one to log all of the write operations to. This is intended for file system developers wishing to verify the integrity of metadata or data as the file system is written to. There is a log_write_entry written for every WRITE request and the target is able to take arbitrary data from userspace to insert into the log. The data that is in the WRITE requests is copied into the log to make the replay happen exactly as it happened originally.

Log Ordering

We log things in order of completion once we are sure the write is no longer in cache. This means that normal WRITE requests are not actually logged until the next REQ_PREFLUSH request. This is to make it easier for userspace to replay the log in a way that correlates to what is on disk and not what is in cache, to make it easier to detect improper waiting/flushing.

This works by attaching all WRITE requests to a list once the write completes. Once we see a REQ_PREFLUSH request we splice this list onto the request and once the FLUSH request completes we log all of the WRITEs and then the FLUSH. Only completed WRITEs, at the time the REQ_PREFLUSH is issued, are added in order to simulate the worst case scenario with regard to power failures. Consider the following example (W means write, C means complete):

> W1,W2,W3,C3,C2,Wflush,C1,Cflush

The log would show the following:

> W3,W2,flush,W1....

Again this is to simulate what is actually on disk, this allows us to detect cases where a power failure at a particular point in time would create an inconsistent file system.

Any REQ_FUA requests bypass this flushing mechanism and are logged as soon as they complete as those requests will obviously bypass the device cache.

Any REQ_OP_DISCARD requests are treated like WRITE requests. Otherwise we would have all the DISCARD requests, and then the WRITE requests and then the FLUSH request. Consider the following example:

> WRITE block 1, DISCARD block 1, FLUSH

If we logged DISCARD when it completed, the replay would look like this:

> DISCARD 1, WRITE 1, FLUSH

which isn't quite what happened and wouldn't be caught during the log replay.

Target interface

i. Constructor

   log-writes <dev_path> <log_dev_path>

dev_path	Device that all of the IO will go to normally.
log_dev_path	Device where the log entries are written to.

ii. Status

   <#logged entries> <highest allocated sector>

#logged entries	Number of logged entries
highest allocated sector	Highest allocated sector

iii. Messages

mark <description>

You can use a dmsetup message to set an arbitrary mark in a log. For example say you want to fsck a file system after every write, but first you need to replay up to the mkfs to make sure we're fsck'ing something reasonable, you would do something like this:

```
mkfs.btrfs -f /dev/mapper/log
dmsetup message log 0 mark mkfs
<run test>
```

This would allow you to replay the log up to the mkfs mark and then replay from that point on doing the fsck check in the interval that you want.

Every log has a mark at the end labeled "dm-log-writes-end".

Userspace component

There is a userspace tool that will replay the log for you in various ways. It can be found here: https://github.com/josefbacik/log-writes

Example usage

Say you want to test fsync on your file system. You would do something like this:

```
TABLE="0 $(blockdev --getsz /dev/sdb) log-writes /dev/sdb /dev/sdc"
dmsetup create log --table "$TABLE"
mkfs.btrfs -f /dev/mapper/log
dmsetup message log 0 mark mkfs

mount /dev/mapper/log /mnt/btrfs-test
<some test that does fsync at the end>
dmsetup message log 0 mark fsync
md5sum /mnt/btrfs-test/foo
umount /mnt/btrfs-test

dmsetup remove log
replay-log --log /dev/sdc --replay /dev/sdb --end-mark fsync
mount /dev/sdb /mnt/btrfs-test
md5sum /mnt/btrfs-test/foo
<verify md5sum's are correct>

Another option is to do a complicated file system operation and verify the file
system is consistent during the entire operation.  You could do this with:

TABLE="0 $(blockdev --getsz /dev/sdb) log-writes /dev/sdb /dev/sdc"
dmsetup create log --table "$TABLE"
mkfs.btrfs -f /dev/mapper/log
dmsetup message log 0 mark mkfs

mount /dev/mapper/log /mnt/btrfs-test
<fsstress to dirty the fs>
btrfs filesystem balance /mnt/btrfs-test
umount /mnt/btrfs-test
dmsetup remove log

replay-log --log /dev/sdc --replay /dev/sdb --end-mark mkfs
btrfsck /dev/sdb
replay-log --log /dev/sdc --replay /dev/sdb --start-mark mkfs \
      --fsck "btrfsck /dev/sdb" --check fua
```

And that will replay the log until it sees a FUA request, run the fsck command and if the fsck passes it will replay to the next FUA, until it is completed or the fsck command exists abnormally.