dm-era

Introduction

dm-era is a target that behaves similar to the linear target. In addition it keeps track of which blocks were written within a user defined period of time called an 'era'. Each era target instance maintains the current era as a monotonically increasing 32-bit counter.

Use cases include tracking changed blocks for backup software, and partially invalidating the contents of a cache to restore cache coherency after rolling back a vendor snapshot.

Constructor

era <metadata dev> <origin dev> <block size>

metadata dev	fast device holding the persistent metadata
origin dev	device holding data blocks that may change
block size	block size of origin data device, granularity that is tracked by the target

Messages

None of the dm messages take any arguments.

checkpoint

Possibly move to a new era. You shouldn't assume the era has incremented. After sending this message, you should check the current era via the status line.

take_metadata_snap

Create a clone of the metadata, to allow a userland process to read it.

drop_metadata_snap

Drop the metadata snapshot.

Status

<metadata block size> <#used metadata blocks>/<#total metadata blocks> <current era> <held metadata root | '-'>

metadata block size	Fixed block size for each metadata block in sectors
#used metadata blocks	Number of metadata blocks used
#total metadata blocks	Total number of metadata blocks
current era	The current era
held metadata root	The location, in blocks, of the metadata root that has been 'held' for userspace read access. '-' indicates there is no held root

Detailed use case

The scenario of invalidating a cache when rolling back a vendor snapshot was the primary use case when developing this target:

Taking a vendor snapshot

- Send a checkpoint message to the era target
- Make a note of the current era in its status line
- Take vendor snapshot (the era and snapshot should be forever associated now).

Rolling back to an vendor snapshot

- Cache enters passthrough mode (see: dm-cache's docs in cache.txt)
- Rollback vendor storage
- Take metadata snapshot
- Ascertain which blocks have been written since the snapshot was taken by checking each block's era
- Invalidate those blocks in the caching software
- Cache returns to writeback/writethrough mode

Memory usage

The target uses a bitset to record writes in the current era. It also has a spare bitset ready for switching over to a new era. Other than that it uses a few 4k blocks for updating metadata:

```
(4 * nr_blocks) bytes + buffers
```

Resilience

Metadata is updated on disk before a write to a previously unwritten block is performed. As such dm-era should not be effected by a hard crash such as power failure.

Userland tools

Userland tools are found in the increasingly poorly named thin-provisioning-tools project:

https://github.com/jthornber/thin-provisioning-tools