Device-mapper snapshot support

Device-mapper allows you, without massive data copying:

- To create snapshots of any block device i.e. mountable, saved states of the block device which are also writable without interfering with the original content;
- To create device "forks", i.e. multiple different versions of the same data stream.
- To merge a snapshot of a block device back into the snapshot's origin device.

In the first two cases, dm copies only the chunks of data that get changed and uses a separate copy-on-write (COW) block device for storage.

For snapshot merge the contents of the COW storage are merged back into the origin device.

There are three dm targets available: snapshot, snapshot-origin, and snapshot-merge.

- snapshot-origin <origin>

which will normally have one or more snapshots based on it. Reads will be mapped directly to the backing device. For each write, the original data will be saved in the <COW device> of each snapshot to keep its visible content unchanged, at least until the <COW device> fills up.

- snapshot <origin> <COW device> <persistent?> <chunksize> [<# feature args> [<arg>]*]

A snapshot of the <origin> block device is created. Changed chunks of <chunksize> sectors will be stored on the <COW device>. Writes will only go to the <COW device>. Reads will come from the <COW device> or from <origin> for unchanged data. <COW device> will often be smaller than the origin and if it fills up the snapshot will become useless and be disabled, returning errors. So it is important to monitor the amount of free space and expand the <COW device> before it fills up.

<persistent?> is P (Persistent) or N (Not persistent - will not survive after reboot). O (Overflow) can be added as a persistent store option to allow userspace to advertise its support for seeing "Overflow" in the snapshot status. So supported store types are "P", "PO" and "N".

The difference between persistent and transient is with transient snapshots less metadata must be saved on disk - they can be kept in memory by the kernel.

When loading or unloading the snapshot target, the corresponding snapshot-origin or snapshot-merge target must be suspended. A failure to suspend the origin target could result in data corruption.

Optional features:

> discard_zeroes_cow - a discard issued to the snapshot device that maps to entire chunks to will zero the corresponding exception(s) in the snapshot's exception store.
>
> discard_passdown_origin - a discard to the snapshot device is passed down to the snapshot-origin's underlying device. This doesn't cause copy-out to the snapshot exception store because the snapshot-origin target is bypassed.
>
> The discard_passdown_origin feature depends on the discard_zeroes_cow feature being enabled.

- snapshot-merge <origin> <COW device> <persistent> <chunksize> [<# feature args> [<arg>]*]

takes the same table arguments as the snapshot target except it only works with persistent snapshots. This target assumes the role of the "snapshot-origin" target and must not be loaded if the "snapshot-origin" is still present for <origin>.

Creates a merging snapshot that takes control of the changed chunks stored in the <COW device> of an existing snapshot, through a handover procedure, and merges these chunks back into the <origin>. Once merging has started (in the background) the <origin> may be opened and the merge will continue while I/O is flowing to it. Changes to the <origin> are deferred until the merging snapshot's corresponding chunk(s) have been merged. Once merging has started the snapshot device, associated with the "snapshot" target, will return -EIO when accessed.

How snapshot is used by LVM2

When you create the first LVM2 snapshot of a volume, four dm devices are used:

1. a device containing the original mapping table of the source volume;
2. a device used as the <COW device>;
3. a "snapshot" device, combining #1 and #2, which is the visible snapshot volume;
4. the "original" volume (which uses the device number used by the original source volume), whose table is replaced by a "snapshot-origin" mapping from device #1.

A fixed naming scheme is used, so with the following commands:

```
lvcreate -L 1G -n base volumeGroup
lvcreate -L 100M --snapshot -n snap volumeGroup/base
```

we'll have this situation (with volumes in above order):

```
# dmsetup table|grep volumeGroup

volumeGroup-base-real: 0 2097152 linear 8:19 384
volumeGroup-snap-cow: 0 204800 linear 8:19 2097536
volumeGroup-snap: 0 2097152 snapshot 254:11 254:12 P 16
volumeGroup-base: 0 2097152 snapshot-origin 254:11

# ls -lL /dev/mapper/volumeGroup-*
brw-------  1 root root 254, 11 29 ago 18:15 /dev/mapper/volumeGroup-base-real
brw-------  1 root root 254, 12 29 ago 18:15 /dev/mapper/volumeGroup-snap-cow
brw-------  1 root root 254, 13 29 ago 18:15 /dev/mapper/volumeGroup-snap
brw-------  1 root root 254, 10 29 ago 18:14 /dev/mapper/volumeGroup-base
```

How snapshot-merge is used by LVM2

A merging snapshot assumes the role of the "snapshot-origin" while merging. As such the "snapshot-origin" is replaced with "snapshot-merge". The "-real" device is not changed and the "-cow" device is renamed to <origin name>-cow to aid LVM2's cleanup of the merging snapshot after it completes. The "snapshot" that hands over its COW device to the "snapshot-merge" is deactivated (unless using lvchange --refresh); but if it is left active it will simply return I/O errors.

A snapshot will merge into its origin with the following command:

```
lvconvert --merge volumeGroup/snap
```

we'll now have this situation:

```
# dmsetup table|grep volumeGroup

volumeGroup-base-real: 0 2097152 linear 8:19 384
volumeGroup-base-cow: 0 204800 linear 8:19 2097536
volumeGroup-base: 0 2097152 snapshot-merge 254:11 254:12 P 16

# ls -lL /dev/mapper/volumeGroup-*
brw-------  1 root root 254, 11 29 ago 18:15 /dev/mapper/volumeGroup-base-real
brw-------  1 root root 254, 12 29 ago 18:16 /dev/mapper/volumeGroup-base-cow
brw-------  1 root root 254, 10 29 ago 18:16 /dev/mapper/volumeGroup-base
```

How to determine when a merging is complete

The snapshot-merge and snapshot status lines end with:

<sectors_allocated>/<total_sectors> <metadata_sectors>

Both <sectors_allocated> and <total_sectors> include both data and metadata. During merging, the number of sectors allocated gets smaller and smaller. Merging has finished when the number of sectors holding data is zero, in other words <sectors_allocated> == <metadata_sectors>.

Here is a practical example (using a hybrid of lvm and dmsetup commands):

```
# lvs
  LV      VG          Attr   LSize Origin  Snap%  Move Log Copy%  Convert
  base    volumeGroup owi-a- 4.00g
  snap    volumeGroup swi-a- 1.00g base  18.97

# dmsetup status volumeGroup-snap
0 8388608 snapshot 397896/2097152 1560
                                  ^^^^ metadata sectors

# lvconvert --merge -b volumeGroup/snap
  Merging of volume snap started.

# lvs volumeGroup/snap
  LV      VG          Attr   LSize Origin  Snap%  Move Log Copy%  Convert
  base    volumeGroup Owi-a- 4.00g          17.23

# dmsetup status volumeGroup-base
0 8388608 snapshot-merge 281688/2097152 1104

# dmsetup status volumeGroup-base
0 8388608 snapshot-merge 180480/2097152 712

# dmsetup status volumeGroup-base
0 8388608 snapshot-merge 16/2097152 16
```

Merging has finished.

```
# lvs
  LV      VG          Attr   LSize Origin  Snap%  Move Log Copy%  Convert
  base    volumeGroup owi-a- 4.00g
```