A new device mapper snapshot implementation

FUJITA Tomonori

fujita.tomonori@lab.ntt.co.jp

NTT Cyber Space Laboratories

Snapshot, for what?

- File server
- Virtual Machine disk image management
- Remote Replication
- Whatever you want to add here

How can we implement snapshot?

- File systems (btrfs, finally)
  - This could do better than DM since file systems know all the details
- DM
  - It works for all the file systems

<u>File</u> systems can do better
but DM still can do something useful

What does DM support now?

- Two snapshot implementations in DM
  - Transient
    - The information for snapshot is only in memory
    - Everything is lost when you shut down your machine
  - Persistent
    - The information for snapshot is in disk

What's wrong with DM's persistent snapshot?

- The performance drops in inverse proportion to the number of snapshots
- The metadata for snapshot must be in memory

It's not designed to keep snapshots for long time

I’m not exaggerating

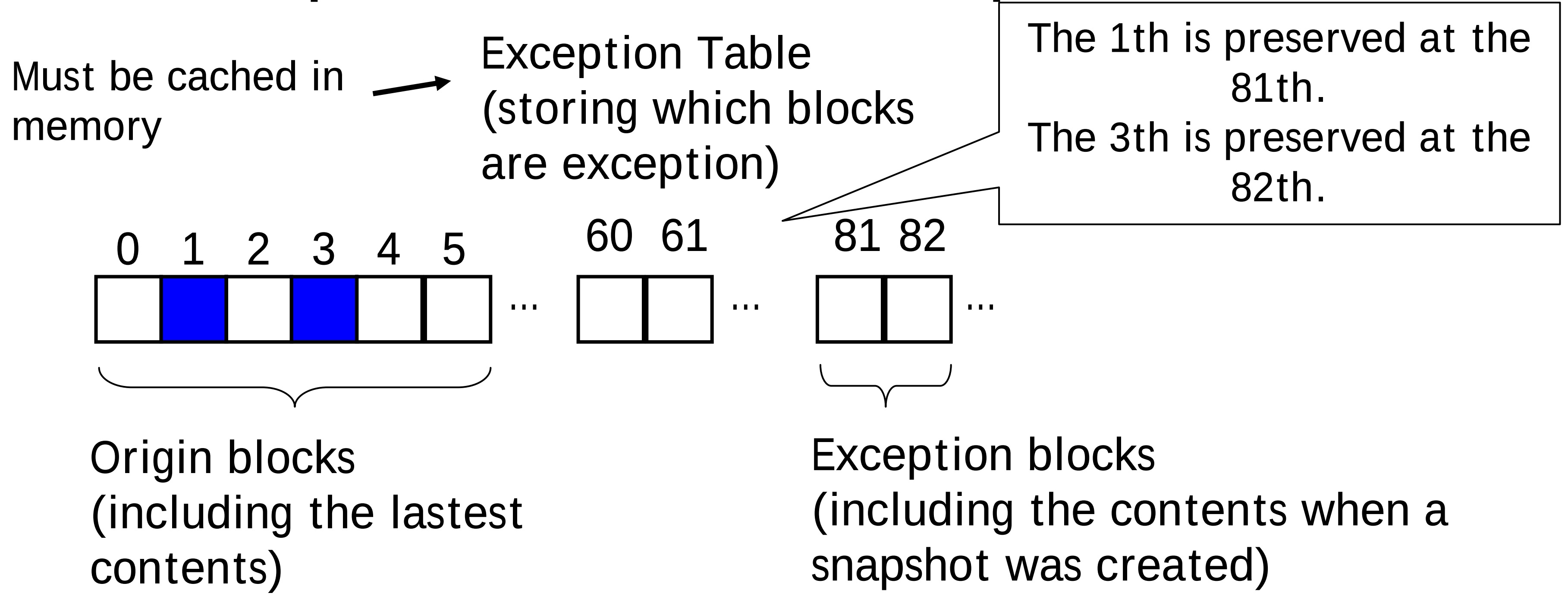
A co-worker recently did some tests on DM snapshots using bonnie, and here is a rough summary of what he got as write throughput:

No Snapshots - 373 MB/s
One Snapshots - 55 MB/s
Two Snapshots - 16 MB/s
Four Snapshots - 14 MB/s
Eight Snapshots - 6.5 MB/s

dm-devel mailing list:
http://marc.info/?l=dm-devel&m=116322663722428&w=2

How persistent snapshot works

- Each snapshot has the own set of exception table and exception blocks


Must be cached in memory
Exception Table (storing which blocks are exception)
The 1th is preserved at the 81th.
The 3th is preserved at the 82th.
0 1 2 3 4 5
60 61
81 82
…
Origin blocks (including the lastest contents)
Exception blocks (including the contents when a snapshot was created)

How a write works with persistent snapshot

1. Sees the exception table of the first snapshot if a block to write is preserved
2. Sees the exception table of the second snapshot if a block to write is preserved
3. Does the same for all the snapshots
4. Writes the original block to the exception block of all the snapshots
5. Updates the exception table of all the snapshots

More snapshots, more writes (seeks), and poor performance !

My new snapshot implementation

All the snapshots share the exception table and exception blocks

- The number of snapshots is not related with the number of writes
- The exception table is small and can be cached efficiently in memory

The new snapshot code is not written from scratch

- The code is based on Zumastor (http://zumastor.org/), remote replication software
  - The primary server periodically sends the delta between two snapshots to the secondary
  - The secondary server always has the consistent data
- Zumastor snapshot code works in user space
  - I rewrote the code for kernel space

The limitations

- The limitations of the btree format used for exception table
  - The maximum number of snapshots is 64
  - Supports writable snapshot but it doesn’t support the snapshot of a snapshot

Status

- 2.6.29-rc1 has the flexible framework to support multiple snapshot implementations (thanks to RedHat DM developers)
  - The code for 2.6.28 was posted to dm-devel
    - git://git.kernel.org/pub/scm/linux/kernel/git/tomo/linux-2.6-misc.git dm-snap
  - I’m updating the code for 2.6.29-rc1
- TODO
  - Needs to finish the journaling code for unexpected crashes
  - Needs to work on performance improvement
  - Needs more testings

Future work

- Remote replication support (or reinventing Zumastor)
  - Adding the interface enable user space to get the delta between snapshots
  - Userspace daemon periodically sends the delta