qcow2

Red Hat
Kevin Wolf
15 August 2011

Section 1

qcow2 format basics

redhat.

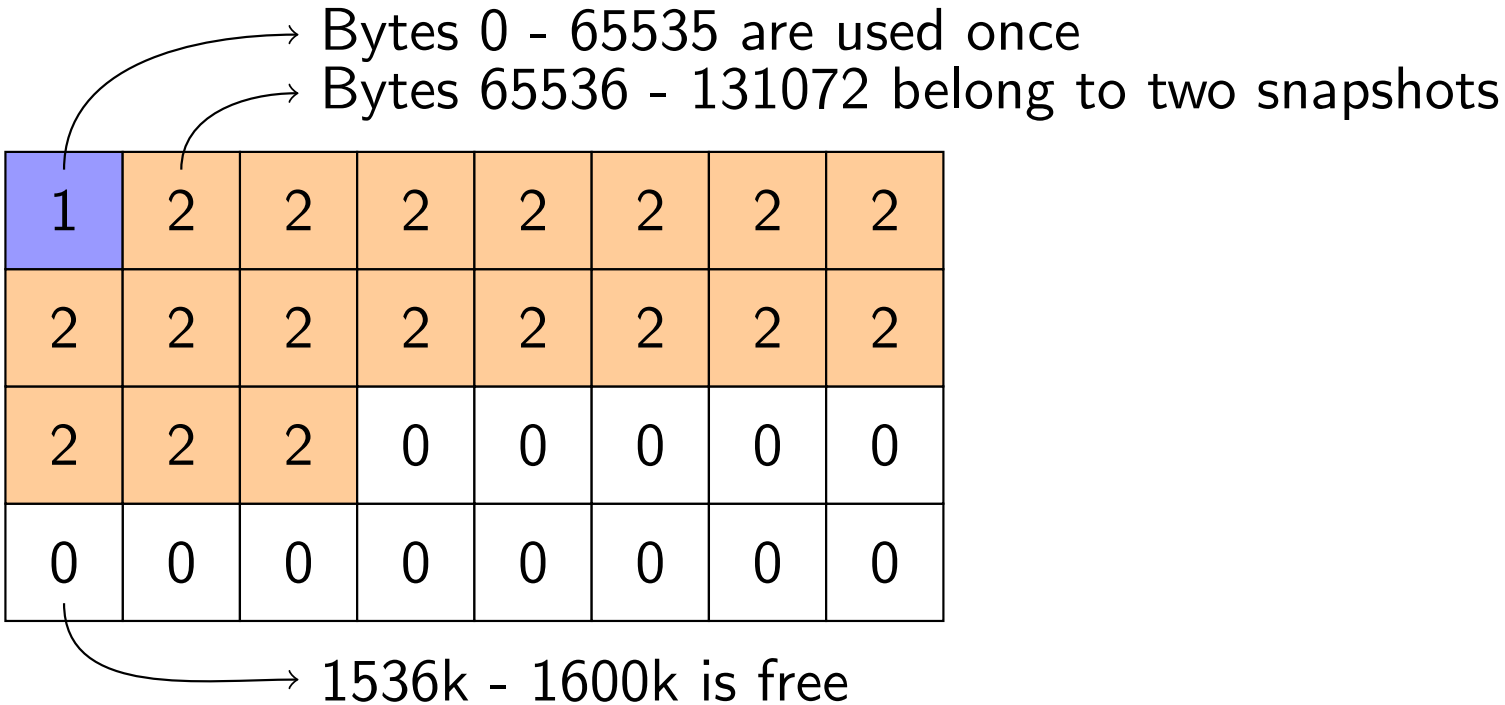
Overview of qcow2 features

- Sparse images
- Snapshots
  - Internal or external
  - Internal snapshots can contain VM state
- Encryption
- Compression
- Can be used on block devices

redhat.

Reference counts

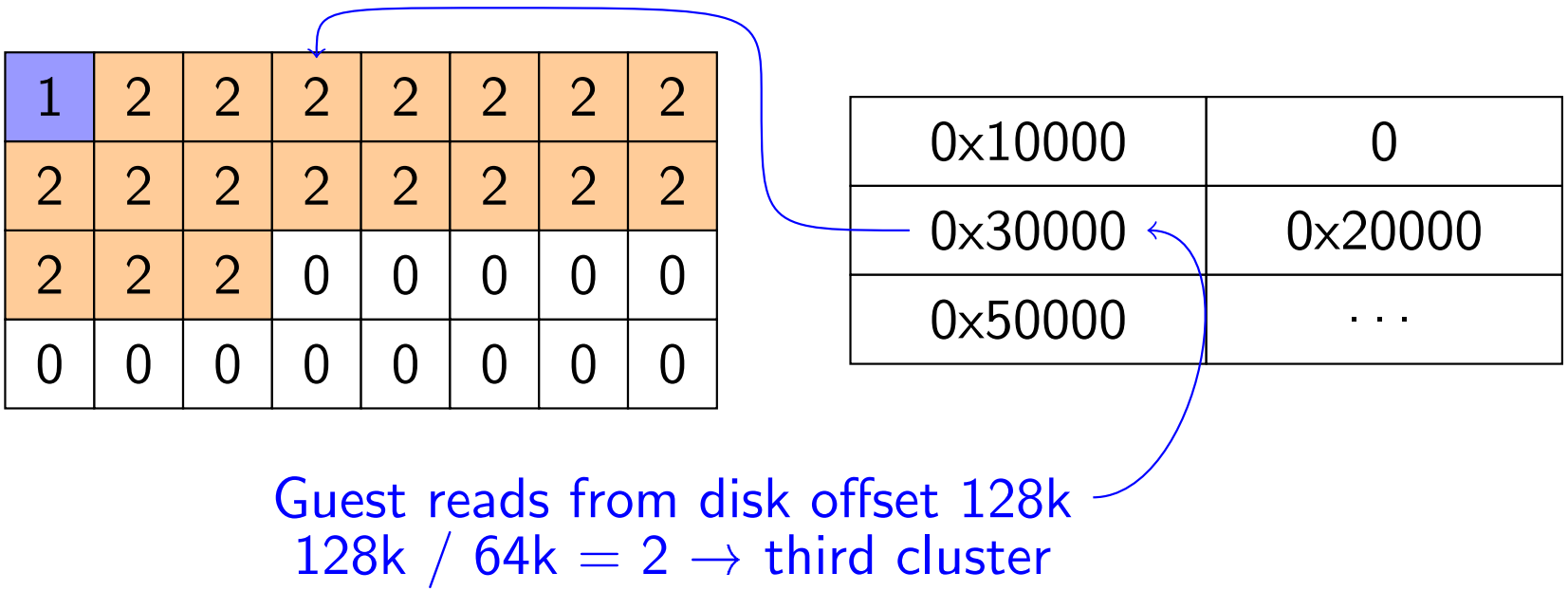
- Image file is divided into clusters (64k default)
- Two-level refcount table tracks used clusters
- Clusters can be shared by multiple internal snapshots



redhat.

Mapping

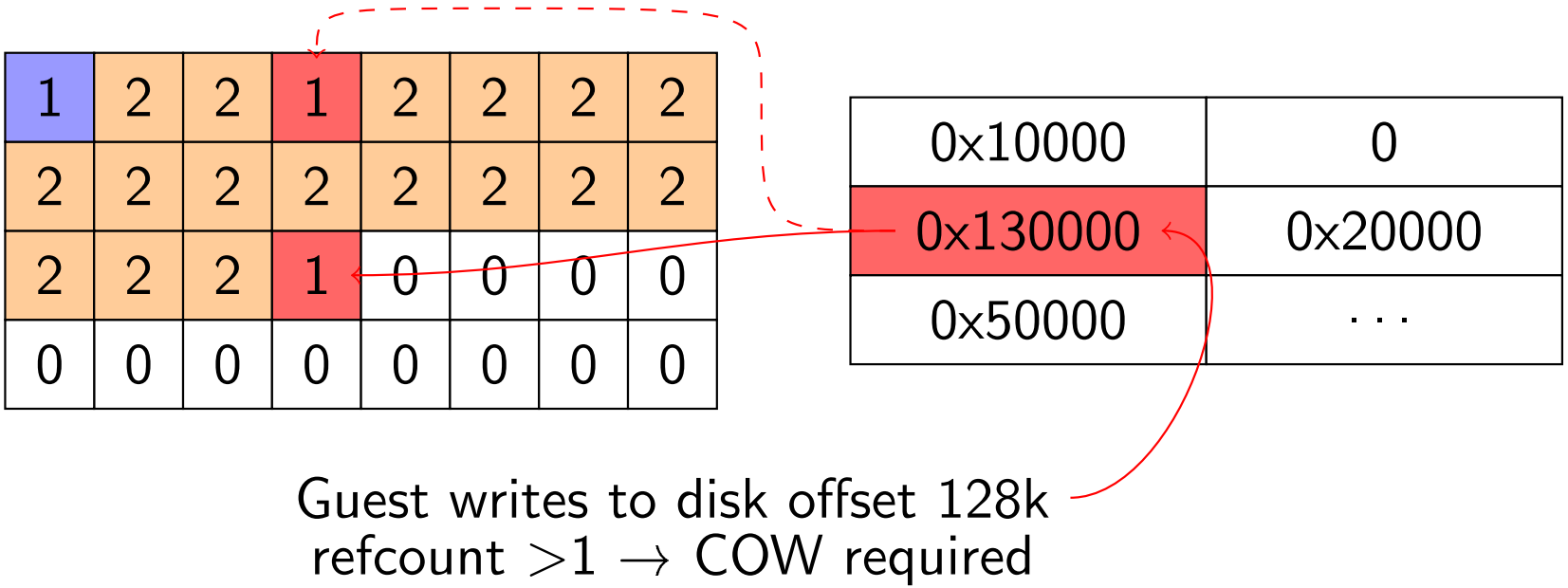
- Two-level lookup table
- L1/L2 table map virtual disk offsets to image file offsets



redhat.

Allocating a cluster

- Requires updates to L2 table and refcounts
- Crashes in the middle must not corrupt the image: Order of updates is important



redhat.

Section 2

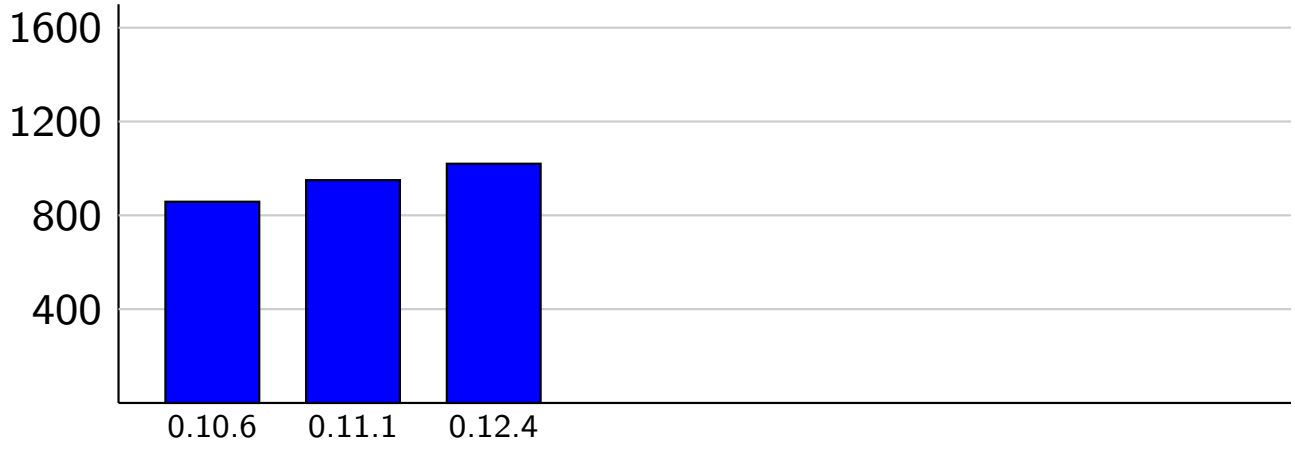
One year ago

redhat.

Development of performance

Many performance improvements until 0.12:

- Allocate multiple clusters at once
- Increase cluster size from 4k to 64k
- Avoid unnecessary reads
- Pretty much every obvious optimisation



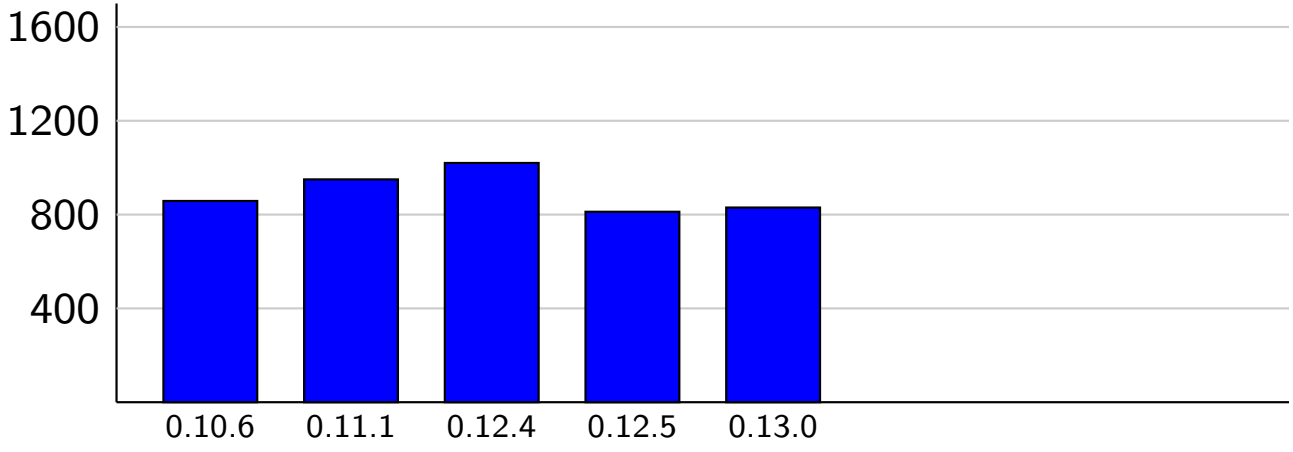
Write throughput in kB/s during cluster allocation; 8k blocks; cache=none

redhat.

Development of performance

However...

- cache=writethrough as default
  - Makes the default perform badly
- Need to ensure ordering for metadata updates
  - Obvious fix: Add fsync after each update
  - Makes everything else perform badly



Write throughput in kB/s during cluster allocation; 8k blocks; cache=none

redhat.

Synchronous metadata updates

- No parallelism
- VCPU can be blocked
  - Can be measured with jitterd
    http://git.codemonkey.ws/cgit/jitterd.git/
  - Worst effects with cache=writethrough

```
[root@localhost jitterd]# ./jitterd -f -m 2 -p 100 -r 10
[2011:08:12 13:29:50] jitterd -m 2 -p 100 -r 10
[2011:08:12 13:29:50] only reporting chatter greater than 200 millisecond(s).
[2011:08:12 13:30:20] chatter over 10 second(s) is 3855 ms with a peak of 390 ms
[2011:08:12 13:30:30] chatter over 10 second(s) is 1438 ms with a peak of 617 ms
[2011:08:12 13:30:40] chatter over 10 second(s) is 560 ms with a peak of 29 ms
...
```

Section 3

Let's invent new formats

redhat.

QED

- Original idea: qcow2 without a refcount table
  - No fsyncs needed for ordering any more
  - ...except that we rely on the file size now
  - File size and mapping updates must be ordered
- Implemented idea: qcow2 without a refcount table, but with a dirty flag
  - Set dirty flag while image is opened
  - After crash a check is required to make the image consistent
- Fully asynchronous implementation
- Only some of qcow2's features

redhat.

FVD

- Separate allocation size from COW size
  - Large clusters (number of allocations, fragmentation)
  - Still keep reasonably small COW size
- Journal for metadata

redhat.

Why new formats are bad

- New formats concentrate on few main features
  - "Uninteresting" features are left out
  - Some people switch, others can't
  - Fragmentation of user base instead of standard format
  - Need features from both formats A and B? Bad luck!
- Completely new codebase
  - Need to maintain (develop, test, ...) more code
  - Less confidence in correctness
- Image conversion required to take advantage of new features
- External tools need to be taught about the format
- Better improve the existing formats!

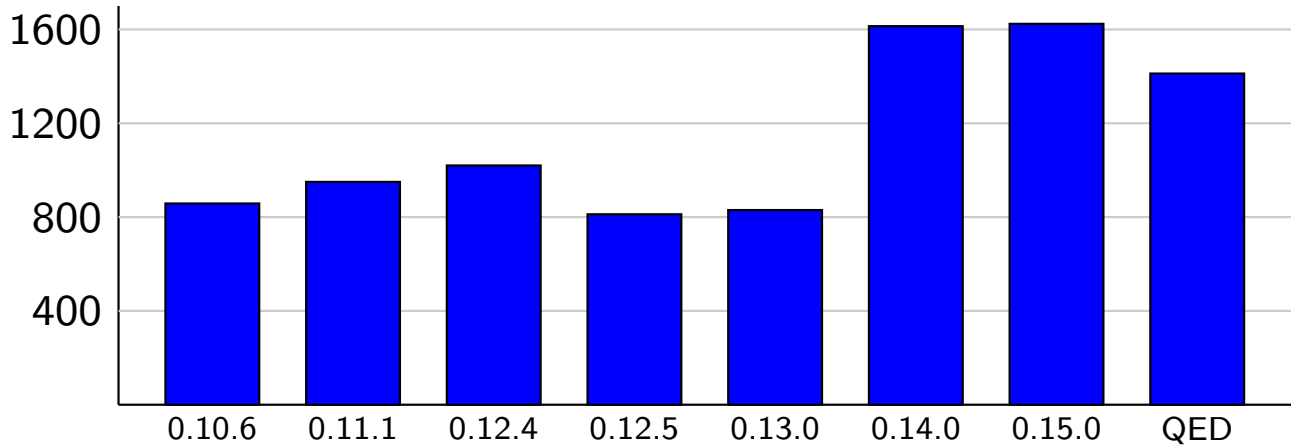
Section 4

Today (qemu 0.15.0)

redhat.

Qcow2Cache

- Introduce a writeback metadata cache
- Improves cluster allocation performance with writeback cache modes (none/writeback/unsafe)
  - Some benchmarks improved by factor 10 or more
- cache=writethrough behaves as before



Write throughput in kB/s during cluster allocation; 8k blocks; cache=none

redhat.

Other recent improvements

- Zero-copy read/write
  - Yes, 0.13 qcow2 used a bounce buffer for everything
- Support for bdrv_discard
  - Not (yet?) passed through to file system
  - Discarded space is reused for cluster allocations

Section 5

qemu 1.x

redhat

Coroutines

- Think of cooperatively scheduled threads
- Allow to make synchronous code asynchronous with minimal effort
  - Code can only be interrupted in known places
  - You can still call functions that are not thread-safe

```
block/qcow2-cluster.c |   26 +++---
block/qcow2.c         |  240 ++++++++++++++++++-------------------------------------
block/qcow2.h         |    5 +-
3 files changed, 102 insertions(+), 169 deletions(-)
```

- VCPU is no longer blocked during metadata access
- Throughput pretty much the same as before
- Future: More parallelism by finer grained locking

redhat.

Random improvements with current format

- Qcow2Cache optimisations:
  - Writethrough mode is too strict:
    We don't need to flush caches after each metadata write
  - Write out only dirty parts of cached parts
- Optimise cluster allocation with no backing file/snapshot (no reason to do COW there)
- Image resizing
  - Shrinking is currently missing, easy to implement
  - What to do with internal snapshots?

redhat.

qcow2 version 3

- RFC patches for specification on mailing list
- Add an optional dirty bit ("QED mode")
- Add zero clusters
  - Keep sparseness with image streaming
  - Discard with backing file
- Configurable refcount width
- Subclusters
  - Separation of allocation and COW size
  - 32 subclusters per cluster allow 64k/2M configuration
  - Less metadata, less fragmentation
  - Allows preallocation even with backing files

The end.

Thanks for listening.