Improving the performance of the qcow2 format

KVM Forum 2017

Alberto Garcia <berto@igalia.com>

Introduction to the qcow2 format

igalia

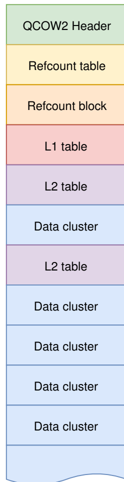
The qcow2 file format

- qcow2: native file format for storing disk images in QEMU.
- Multiple features:
  - Grows on demand.
  - Supports backing files.
  - Internal snapshots.
  - Compression.
  - Encryption.
- Can achieve good performance (comparable to raw files), but it depends on the scenario.
- Making it faster may require:
  - A correct configuration.
  - Changes in the QEMU driver.
  - Changes in the format itself.

igalia

Structure of a qcow2 file

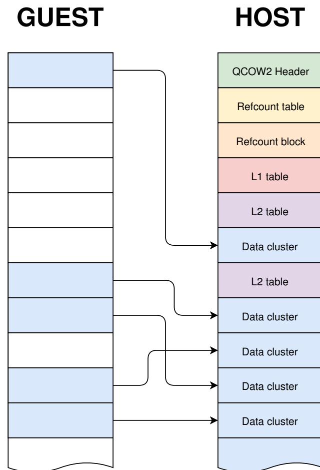
A qcow2 file is divided into clusters of equal size
(min: 512 bytes - default: 64 KB - max: 2 MB)



igalia

Structure of a qcow2 file

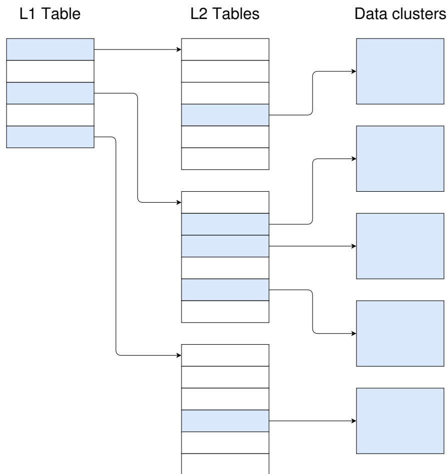
The virtual disk as seen by the VM is divided into guest clusters of the same size



igalia

L1 and L2 tables

The L1 and L2 tables map guest addresses as seen by the VM into host addresses in the qcow2 file



igalia

Backing files

- If QEMU tries to read data from a cluster that hasn't been allocated, it goes to the backing file in order to get the data.
- Backing files don't need to have the same format or cluster size as the active image.
- They can be chained: a backing file can have its own backing file.

igalia

The problems of L1 and L2 tables

Cluster mapping: L1 and L2 tables

- The L1 and L2 tables map guest clusters to host clusters.
- There's only one L1 table per image (per snapshot, actually), but it's small so it can be kept in RAM.
- Several L2 tables, allocated on demand as the image grows.
- Each time we need to access a data cluster (read or write) we need to go to its L2 table.
- This is one additional I/O operation per request: severe impact in performance.
- Solution: keep the L2 tables in RAM too.

The qcow2 L2 cache

- QEMU keeps a cache of L2 tables to speed up disk access.
- The maximum amount of L2 metadata depends on the disk size and the cluster size.
- Problem: large images need large amounts of metadata, so we cannot keep everything in memory.

Cluster size (=L2 table size)	Max. L2 size per TB
64 KB	128 MB (2048 tables)
128 KB	64 MB (512 tables)
256 KB	32 MB (128 tables)
512 KB	16 MB (32 tables)
1 MB	8 MB (8 tables)
2 MB	4 MB (2 tables)

igalia

Using the qcow2 L2 cache

- The cache keeps full L2 tables in memory.
- Default cache size: 1MB.
- It can be changed with the *l2-cache-size* option:

```
-drive file=img.qcow2,l2-cache-size=8M
```

- With the default cluster size (64 KB) it's enough for a 8 GB disk image.
- Setting the right cache size has a *dramatic* effect on performance.
- Example: random 4K read requests on a fully populated 20GB image (SSD backend).

L2 cache size	Average IOPS
1 MB	5100
1.5 MB	7300
2 MB	12700
2.5 Mb	63600

How much cache do we need?

- The amount of L2 metadata for a certain disk image is

$$\frac{disk_size \times 8}{cluster_size}$$

- Problem: this formula is too complicated. Why would the user need to know about it?
- QEMU should probably have a good default... but what's a good default?
- Alternative: instead of saying how much memory we want, we can say how much disk we want to cover.
  - This has already been discussed, see RedHat bug #1377735.

igalia

How much cache do we need?: cluster sizes

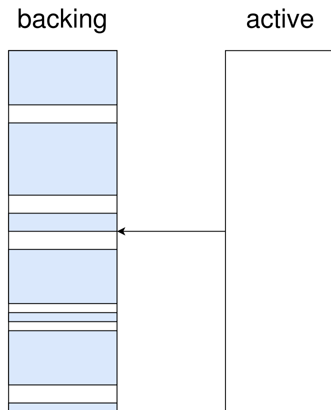
- Increasing the cluster size is an easy way to reduce the metadata size.

$$l2_size = \frac{disk_size \times 8}{cluster_size}$$

- Pros:
  - Same performance with a smaller cache.
  - Reduces fragmentation.
- Cons:
  - Slower allocations.
  - Wastes more disk space.

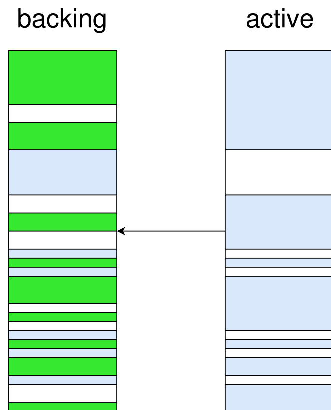
How much cache do we need?: backing files

- Problem: each qcow2 image has its own cache. Backing images also need theirs!
- Things get worse: cached tables in backing files might end up being unnecessary.



How much cache do we need?: backing files

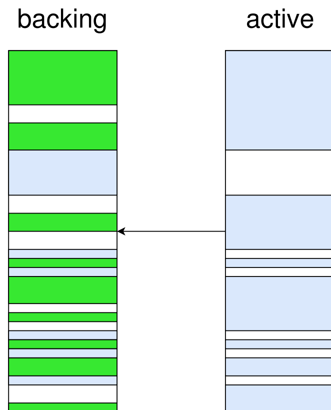
- Problem: each qcow2 image has its own cache. Backing images also need theirs!
- Things get worse: cached tables in backing files might end up being unnecessary.



How much cache do we need?: backing files

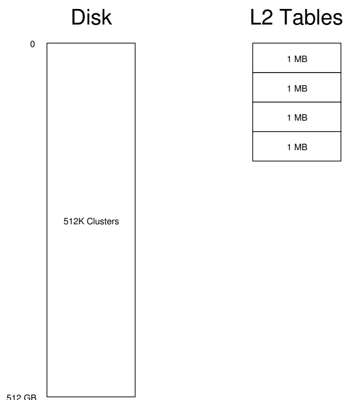
- Solution: we can clean unused cache entries using the `cache-clean-interval` setting:

```
-drive file=hd.qcow2,cache-clean-interval=60
```



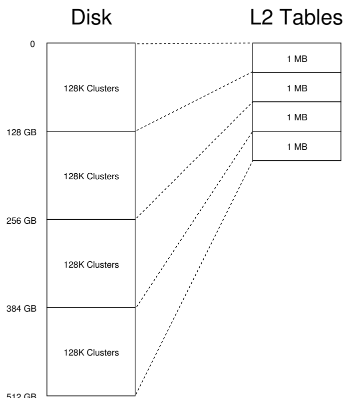
Large cluster sizes means large L2 tables

- An L2 table is always one cluster in size, and each cache entry can only store one full L2 table. This means:
  - More I/O if we only need few entries in an L2 table.
  - Inflexible and inefficient use of the cache memory.



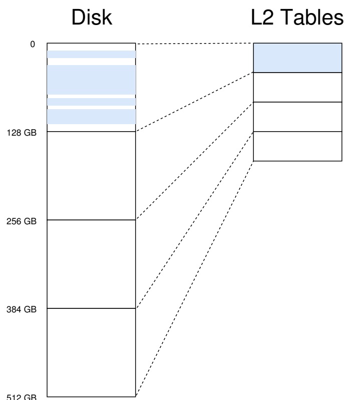
Large cluster sizes means large L2 tables

- An L2 table is always one cluster in size, and each cache entry can only store one full L2 table. This means:
  - More I/O if we only need few entries in an L2 table.
  - Inflexible and inefficient use of the cache memory.



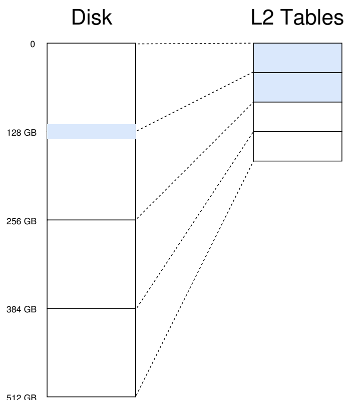
Large cluster sizes means large L2 tables

- An L2 table is always one cluster in size, and each cache entry can only store one full L2 table. This means:
  - More I/O if we only need few entries in an L2 table.
  - Inflexible and inefficient use of the cache memory.



Large cluster sizes means large L2 tables

- An L2 table is always one cluster in size, and each cache entry can only store one full L2 table. This means:
  - More I/O if we only need few entries in an L2 table.
  - Inflexible and inefficient use of the cache memory.



igalia

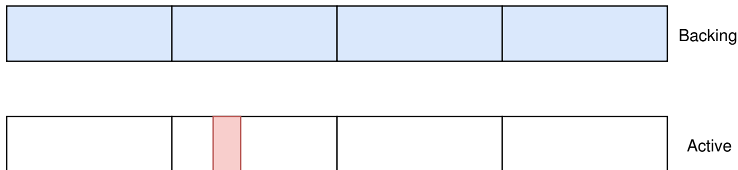
Solution: reduce the cache granularity

- Instead of reading complete L2 tables, make the cache read smaller portions: *L2 slices*.
- Less disk I/O.
- The size of the slice can be adjusted to match that of the host filesystem.
- The qcow2 on-disk format does not need to change.
- The qcow2 driver in QEMU needs relatively few changes.
- Patches available in the mailing list!
- Example: random 4K reads (SSD backend).

Disk size	**Cluster size**	**L2 cache**	**QEMU master**	**4K slices**
16 GB	64 KB	1 MB [8 GB]	5000 IOPS	12700 IOPS
2 TB	2 MB	4 MB [1 TB]	576 IOPS	11000 IOPS

The problems of cluster allocation

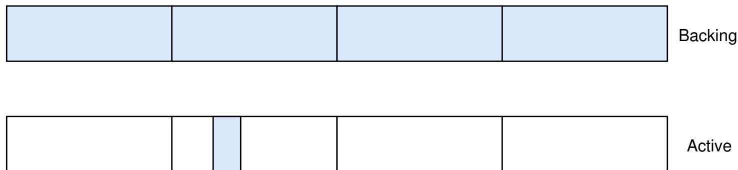
Cluster allocation and copy-on-write



- Allocating a cluster means filling it completely with data.
- If the guest write request is small, the rest must be filled with old data (e.g from a backing image).
- QEMU used up to five operations for this: 2 reads, 3 writes.
- It can be done optimally with just two: 1 read, 1 write.
- New algorithm already available in QEMU 2.10.
- Average increase of IOPS: 60 % (HDD), 15 % (SSD).

igalia

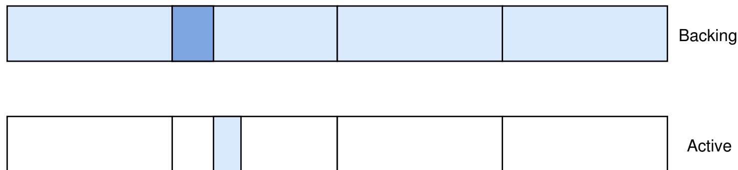
Cluster allocation and copy-on-write



- Allocating a cluster means filling it completely with data.
- If the guest write request is small, the rest must be filled with old data (e.g from a backing image).
- QEMU used up to five operations for this: 2 reads, 3 writes.
- It can be done optimally with just two: 1 read, 1 write.
- New algorithm already available in QEMU 2.10.
- Average increase of IOPS: 60 % (HDD), 15 % (SSD).

igalia

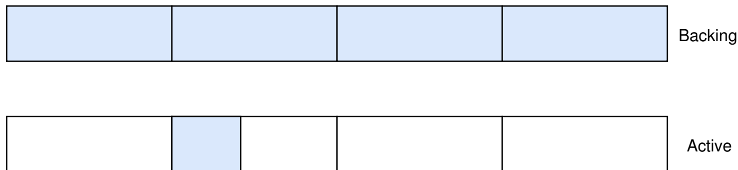
Cluster allocation and copy-on-write



- Allocating a cluster means filling it completely with data.
- If the guest write request is small, the rest must be filled with old data (e.g from a backing image).
- QEMU used up to five operations for this: 2 reads, 3 writes.
- It can be done optimally with just two: 1 read, 1 write.
- New algorithm already available in QEMU 2.10.
- Average increase of IOPS: 60 % (HDD), 15 % (SSD).

igalia

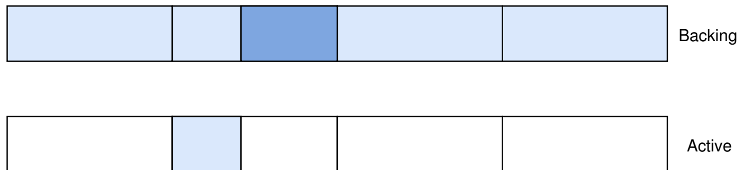
Cluster allocation and copy-on-write



- Allocating a cluster means filling it completely with data.
- If the guest write request is small, the rest must be filled with old data (e.g from a backing image).
- QEMU used up to five operations for this: 2 reads, 3 writes.
- It can be done optimally with just two: 1 read, 1 write.
- New algorithm already available in QEMU 2.10.
- Average increase of IOPS: 60 % (HDD), 15 % (SSD).

igalia

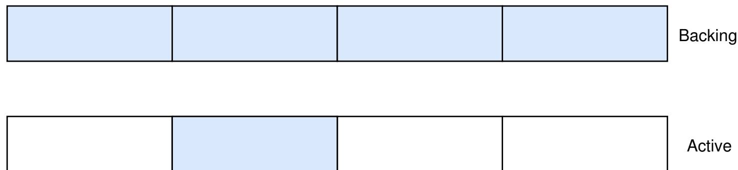
Cluster allocation and copy-on-write



- Allocating a cluster means filling it completely with data.
- If the guest write request is small, the rest must be filled with old data (e.g from a backing image).
- QEMU used up to five operations for this: 2 reads, 3 writes.
- It can be done optimally with just two: 1 read, 1 write.
- New algorithm already available in QEMU 2.10.
- Average increase of IOPS: 60 % (HDD), 15 % (SSD).

igalia

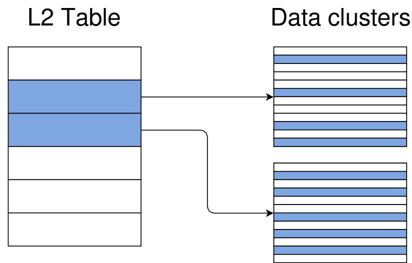
Cluster allocation and copy-on-write



- Allocating a cluster means filling it completely with data.
- If the guest write request is small, the rest must be filled with old data (e.g from a backing image).
- QEMU used up to five operations for this: 2 reads, 3 writes.
- It can be done optimally with just two: 1 read, 1 write.
- New algorithm already available in QEMU 2.10.
- Average increase of IOPS: 60 % (HDD), 15 % (SSD).

igalia

Subcluster allocation



- Divide each data cluster into subclusters and allocate each one individually.
- Reduces allocation overhead while keeping some benefits of large clusters.

Subcluster allocation: benefits, problems and status

- Last proposed in April 2017, prototype shows 2 to 4 times more IOPS during allocations.
- If subcluster size equals request size, no copy-on-write needed: 10 times faster.
- Other benefits: it would allow preallocation of images with backing files.
- Problems:
  - Incompatible changes to the on-disk format.
  - Increases the complexity of the qcow2 driver.
  - Increases data fragmentation in the image.

Space preallocation

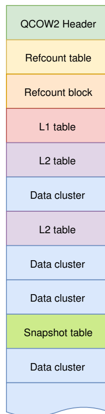
- When writing to a newly-allocated cluster we must fill it with old data when necessary (copy-on-write).
- If there was no old data, the request is padded with zeroes.
- Instead of writing those zeroes, we can use fallocate() to preallocate and empty the cluster first.
- Requires support from the OS and the filesystems (ext4, xfs, ...).
- Patches in the mailing list (by Anton Nefedov).

igalia

Other considerations

qcow2 overlap checks

- Sanity checks before writing to a qcow2 image.
- They verify that a given offset doesn't overlap with existing metadata sections.
- Available since QEMU 1.7.
- Problem: some of these checks are relatively expensive.



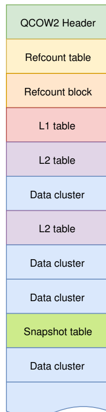
igalia

qcow2 overlap checks

Constant time	Cached data	Needs disk access
main-header active-l1 refcount-table snapshot-table	active-l2 refcount-block inactive-l1	inactive-l2

- *inactive-l2* is disabled by default (it needs to read all snapshots' L1 tables).
- *refcount-block* is particularly expensive even with small images. Optimized in QEMU v2.9.
- Checks can be configured with

```
overlap-check.<check-name>=[on|off]
overlap-check=[constant|cached|all|none]
```



igalia

Status summary

Status summary

- qcow2 L2 cache:
  - Size and cleanup timer are configurable.
  - Probably needs better defaults or configuration options.
- L2 slices:
  - Patches in the mailing list.
- COW with two I/O operations instead of five:
  - Available in QEMU 2.10.
- COW with preallocation instead of writing zeroes:
  - Patches in the mailing list.
- Subcluster allocation:
  - RFC status. Requires changes to the on-disk format.
- Metadata overlap checks:
  - Slowest check optimized in QEMU 2.9.
  - Other checks can be disabled manually if needed.

Acknowledgments

Questions?

igalia

Thank you!

igalia