redhat.

A block layer overview

Red Hat
Kevin Wolf

8 November 2012

Section 1

Overview

Parts of the QEMU block subsystem

- Virtual devices
  - IDE, virtio-blk, ...
- Backend
  - Block drivers
    - raw, qcow2, ...
    - file, nbd, iscsi, gluster...
  - I/O throttling, copy on read, ...
- Block jobs
  - Streaming, mirroring, commit, ...
- External tools
  - qemu-img, qemu-nbd, ...

Configuring a block device

On the command line:

```
-hda test.img
```

…is a shortcut for…

```
-drive file=test.img,if=ide,cache=writeback,aio=threads
```

…is a shortcut for…

```
-drive file=test.img,id=ide0-hd0,if=none,cache=writeback,aio=threads
-device ide-drive,bus=ide.0,drive=ide0-hd0
```

Section 2

Virtual devices

Available devices (I)

- Emulation of real hardware: Best compatibility
  - IDE:
    - Supported by basically every OS
    - Slow, only one request at a time
  - AHCI:
    - Supported by recent OSes
    - No live migration yet (to come with 1.3 or 1.4)
  - SCSI:
    - Emulated controllers used to be unreliable
    - QEMU 1.2 has new *megasas* device
    - Can't boot from SCSI disks
  - Floppy, CD-ROM, USB Storage, ...

Available devices (II)

- Paravirtual devices: Best performance
  - virtio-blk:
    - Required drivers meanwhile commonly available
    - Relatively small feature set
  - virtio-scsi:
    - Still new, drivers only in very recent Linux
    - Uses SCSI command set
- SCSI passthrough (scsi-generic/scsi-block)
  - Supports features of the real hardware
  - Still uses one of the emulated SCSI controllers
  - Needs a real block device, not just an image file

Configuring advanced properties

- Show all supported options for a device:

```
$ x86_64-softmmu/qemu-system-x86_64 -device ide-drive,help
ide-drive.drive=drive
ide-drive.logical_block_size=blocksize
ide-drive.physical_block_size=blocksize
ide-drive.min_io_size=uint16
ide-drive.opt_io_size=uint32
ide-drive.bootindex=int32
ide-drive.discard_granularity=uint32
ide-drive.ver=string
ide-drive.wwn=hex64
ide-drive.serial=string
ide-drive.model=string
ide-drive.unit=uint32
```

- Setting options on the command line:

```
$ x86_64-softmmu/qemu-system-x86_64 \
  -drive file=test.img,if=none,id=mydisk \
  -device ide-drive,bus=ide.0,drive=mydisk,physical_block_size=4096
```

Section 3

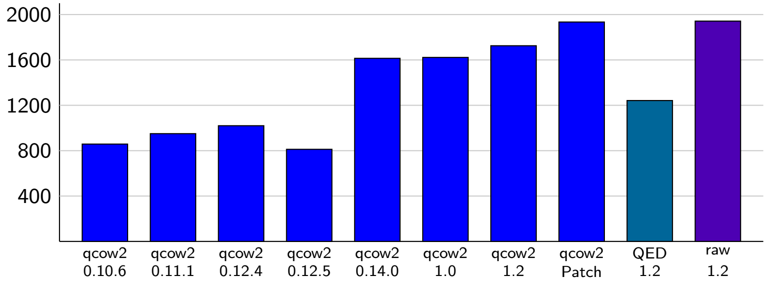
Backends

Image format

- raw
  - Highest possible performance
  - Almost no features (like snapshots etc.)
- qcow2
  - Lots of features
    - Sparse images
    - Snapshots (internal and external)
    - Encryption
    - Compression
  - Somewhat slower (esp. initial writes)
- VMDK, VHD, VDI...
  - Provided for compatibility
  - Best to convert to raw or qcow2 for running VMs

Image format performance

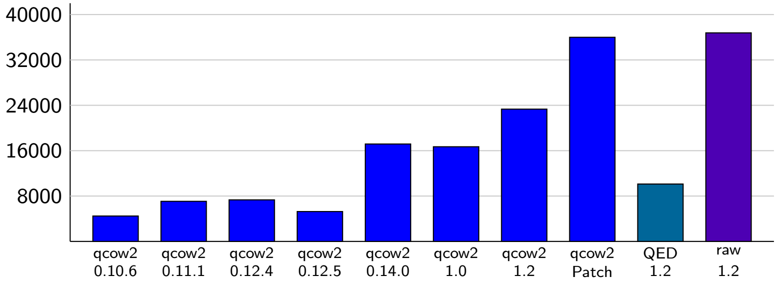
- Pain point of image formats is initial writes (cluster allocation)
- RFC patches for qcow2 (Delayed COW) help to close the gap
  - For the single-threaded case anyway



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

Image format performance

- Pain point of image formats is initial writes (cluster allocation)
- RFC patches for qcow2 (Delayed COW) help to close the gap
  - For the single-threaded case anyway



Write throughput in kB/s during sequential cluster allocation; 256k blocks; cache=none (iozone)

Backing storage

- File
  - Local file system
  - NFS
  - `-drive file=disk.img`
- Block device
  - Whole disk or partition
  - Logical volume
  - External implementation of iscsi, NBD, ...
  - `-drive file=/dev/sda3`
- NBD
  - `-drive file=nbd:localhost:10809`
- glusterfs
  - `-drive file=gluster+tcp://1.2.3.4/testvol/a.img`
- ...

Cache options

	Use host page cache	Guest disk WCE
writeback	yes	enabled
none	no	enabled
writethrough	yes	disabled
directsync	no	disabled

- Default mode is **writeback** since 1.2
- Write cache enabled (**WCE**) is safe for correct guest OS
  - WCE **improves write performance** a lot
  - Some older OSes are broken and ignore write caches
    - Risk of data corruption on host crash
    - Turn off WCE (only) for those (automatic on virtio-blk)
- `cache=unsafe` e.g. for installation

Cache options

	Use host page cache	Guest disk WCE
writeback	yes	enabled
none	no	enabled
writethrough	yes	disabled
directsync	no	disabled

- Usually you don't want to use the host page cache
  - The guest has already a page cache
  - Data would be duplicated – waste of memory
- But it can make sense in some cases
  - Many guests sharing the host cache
  - Short-lived guests
- **Must** bypass host page cache for safe live migration

AIO mode

- `-drive aio=threads` (Userspace thread pool)
  - Default mode
  - Tends to perform better on file systems
  - On all POSIX platforms
- `-drive aio=native` (Linux AIO)
  - Tends to perform better on block devices
  - Only on Linux
  - Requires O_DIRECT (`cache=none/directsync`)

Image format options

- During image creation:

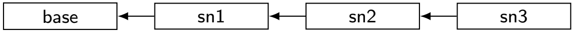
```
$ ./qemu-img create -f qcow2 -o help /tmp/test.qcow2
Supported options:
size                 Virtual disk size
compat               Compatibility level (0.10 or 1.1)
backing_file         File name of a base image
backing_fmt          Image format of the base image
encryption           Encrypt the image
cluster_size         qcow2 cluster size
preallocation        Preallocation mode (allowed values: off, metadata)
lazy_refcounts       Postpone refcount updates
$ qemu-img create -f qcow2 -o compat=1.1,lazy_refcounts=on \
  /tmp/test.qcow2 4G
```

- -blockdev will enable driver-specific command line options

Section 4
Block jobs

Snapshots

- External snapshots (backing files):

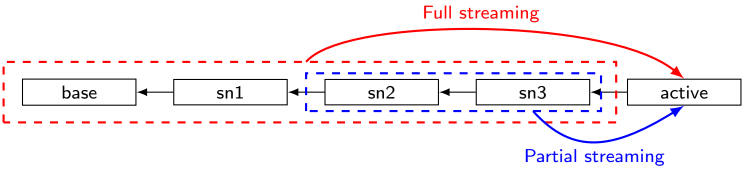


  - COW layer over backing files (of any image format) saves delta
  - Cheap to create
  - Deleting a snapshot means copying all data
- Internal snapshots (`savevm`/`loadvm`, qcow2 only):
  - Snapshot saved in the same image file
  - Creation and deletion both with some cost
    - Modify metadata, but no copy of data required
  - Can contain VM state
  - No live snapshots (VM stops while saving snapshot)
  - Receives less testing $\Rightarrow$ Stability?

Block jobs

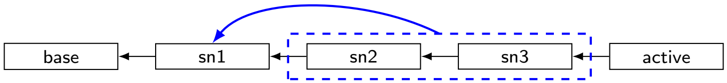
- Introduced in QEMU 1.1 and extended in each release since
- Long-running background jobs on block devices
  - Live storage migration
  - Deleting external snapshots
- Started and controlled using monitor commands
  - Starting: Type specific command (e.g. `block-stream`)
  - Completion: Automatically or with `block-job-complete`
  - `block-job-cancel`
  - `block-job-pause/resume`
  - `block-set-speed`
  - `query-block-jobs`

Image streaming



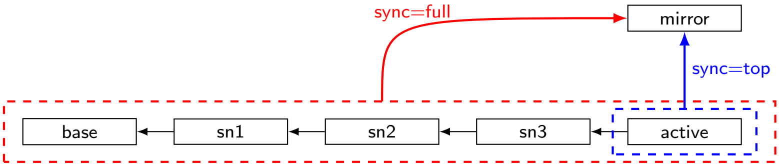
- Pull data from backing files into active layer
- Backing files become redundant and can be removed
- Use cases:
  - Copy an image from a slow source in the background while running the VM
  - Delete topmost external snapshots
- Since QEMU 1.1

Live commit



- Apply delta to backing file
  - Delete external snapshots
- Will be in QEMU 1.3
  - Committing active layer not supported yet

Image mirroring



- Live storage migration
  - Copies data into new image
  - Guest writes are mirrored into the copy
- Either full chain or only active layer
- Will be in QEMU 1.3

Builtin NBD server

- Not a block job, strictly speaking
- Allows storage migration without shared storage
  - Destination QEMU starts NBD server
  - Source QEMU mirrors its image using an NBD connection
- QEMU 1.3 or 1.4

The end.

Thanks for listening.