# Block Layer Status Report

Red Hat
Kevin Wolf <kwolf@redhat.com>
Stefan Hajnoczi <stefanha@redhat.com>

KVM Forum 2013

# Block layer introduction

- Kevin and Stefan maintain the QEMU block layer
- Block layer enables virtual disk, CD-ROM, floppy drives
- 11 image formats (qcow2, vmdk, vhdx, etc)
  - Native formats: raw, qcow2
- 9 protocols (file, Gluster, iSCSI, etc)
- This presentation covers current work

redhat.

# Part I
# Image formats

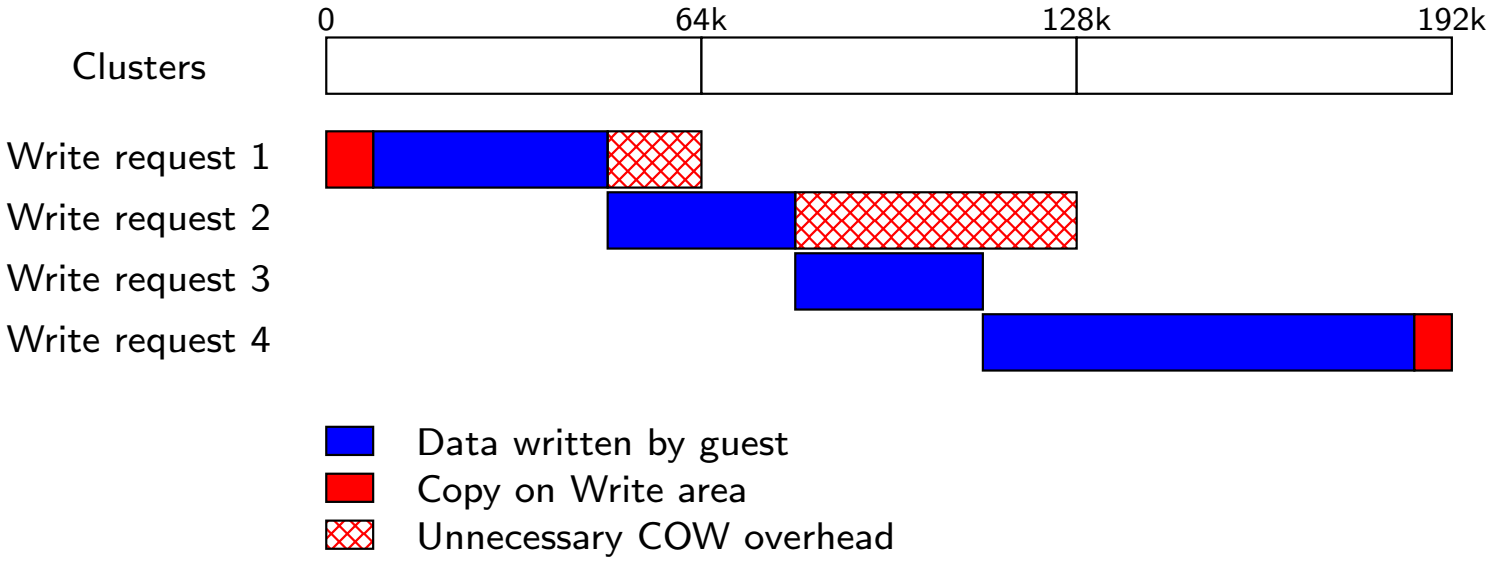
# qcow2: Data deduplication

- Detect clusters with identical data
  $\Rightarrow$ store them only once
- Challenge:
  - Can't keep hashes for all clusters in memory
  - Disk is slow
- Benoît Canet will talk more about this

# qcow2: Corruption prevention

- Additional safety measure to protect metadata...
  - ...in already corrupted image files
  - ...against qemu bugs
- Offsets of most metadata structures are in memory
- Attempt to overwrite metadata that shouldn't be?
  - Fail the request without overwriting metadata
  - Mark the image corrupted
  - Make it read-only until after `qemu-img check -r all`

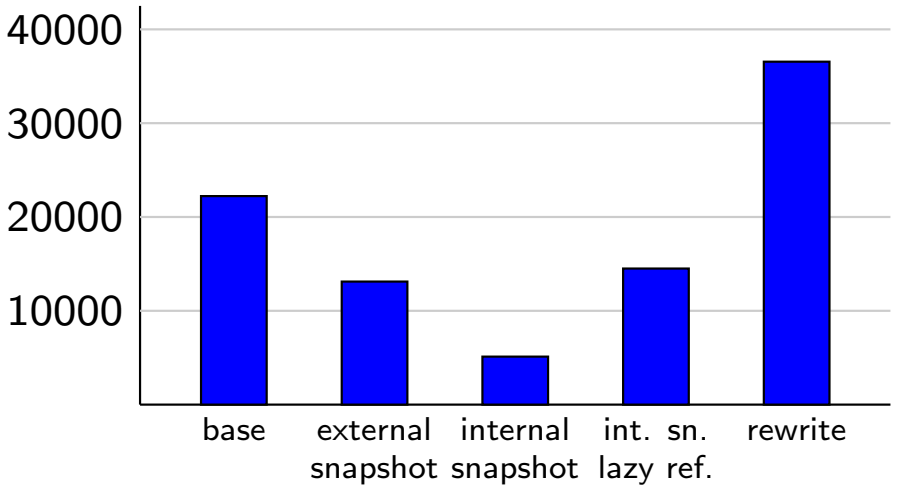
# qcow2: Performance (Copy on Write)




- For simple images, COW is the only relevant overhead
  - Delayed COW can fix it
  - Near-raw performance even for allocations

# qcow2: Performance (Internal COW)

- Internal snapshots, compression
- Internal COW is extremely expensive
  - Need two disk flushes per request for ordering metadata updates
- Lazy refcounts can mitigate it




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

# qcow2: Journalling

We considered introducing a journal for

- Delayed COW
  - Would allow to delay across flushes
    - cache=writethrough flushes after each request!
  - ...but good enough without a journal
- Improve internal COW performance
  - ...but lazy refcounts can mitigate it
- No more cluster leaks on crashes and errors

Conclusion: Perhaps later

# Non-native image formats

- VHDX read/write support:
  - Journalling support
  - Creating VHDX images
- VMDK support for newer versions

redhat.

Part II

# Block device configuration

# Driver-specific options

- Traditionally: Options encoded in “filename” string
  - Only for protocols, not for formats
  - No way to configure backing files
  - Characters with special meaning (colon in filename?)
- Examples:
  - `nbd:localhost:1234`
  - `fat:floppy:rw:/tmp/vvfat_dir`
  - `blkdebug:/tmp/blkdebug.cfg:/tmp/test.qcow2`
- New: Separate, driver-specific options
  - `-drive file.driver=nbd,file.host=localhost`
  - `-drive file=test.qcow2,lazy-refcounts=on`

# drive_add

HMP `drive_add` isn't suitable for QMP:

- Parsing strings instead of structured JSON data
- Device configuration mixed with backend configuration
- Convenience magic gets in the way
  - e.g. automatic deletion of backend after unplug

# blockdev-add

Introduce a separate `blockdev-add` QMP command:

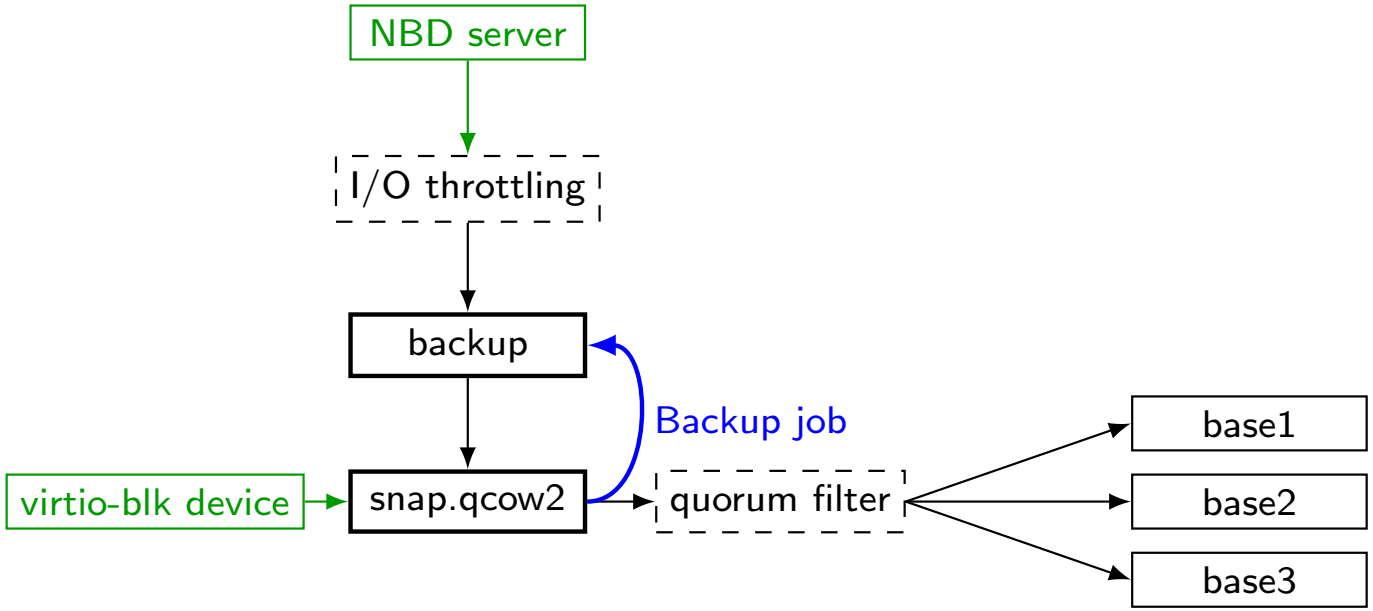
- In qemu.git master now
- Configures only backend aspects
- Command line: Exact mapping of JSON structure
- Doesn't provide copy-on-read and I/O throttling
  - Should become block filters
- Network protocol support still to be done

# blockdev-add on the JSON level

```
{ "execute": "blockdev-add",
  "arguments": {
      "options": {
        "driver":  "qcow2",
        "id":      "my_disk",
        "discard": "unmap",
        "cache":   { "direct": true,
                     "writeback": true },
        "file":    { "driver": "file",
                     "filename": "/tmp/test.qcow2" },
        "backing": { "driver": "qcow2",
                     "file": { "driver": "file",
                               "filename": "/dev/fdset/4" } } } } }
```

# Giving users full control

- Flexibility to create complex structures
  - Block filters
  - NBD server
  - User access to any node in the graph



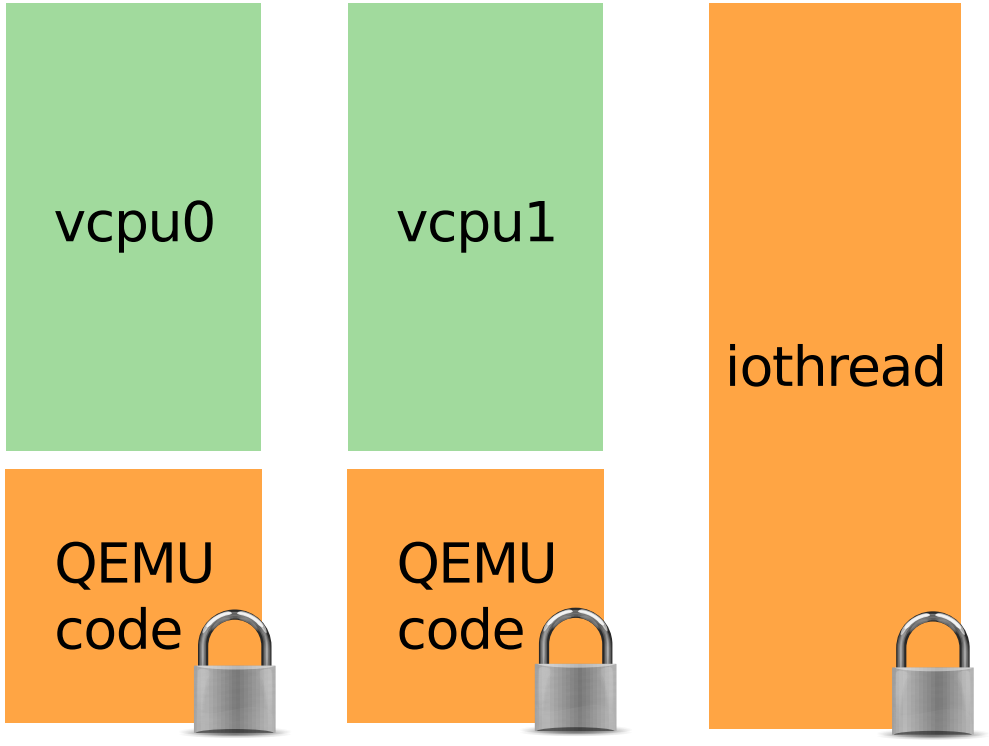

- Markus Armbruster and Kevin Wolf will talk more about this

redhat.

# Part III
# Dataplane

# I/O emulation scalability bottleneck

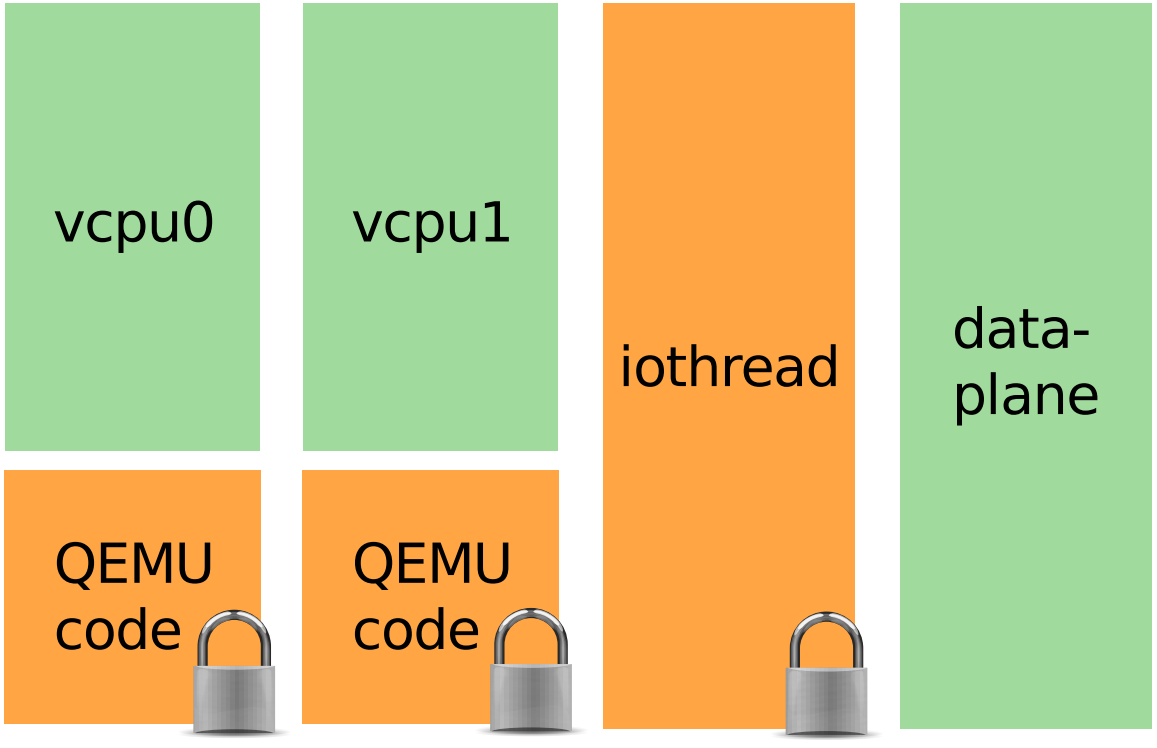
- I/O emulation is bottlenecked on the Big QEMU Lock
- SMP host & guest results in lock contention




- Amdahl's Law: only parallel parts can scale

# Getting around the bottleneck today

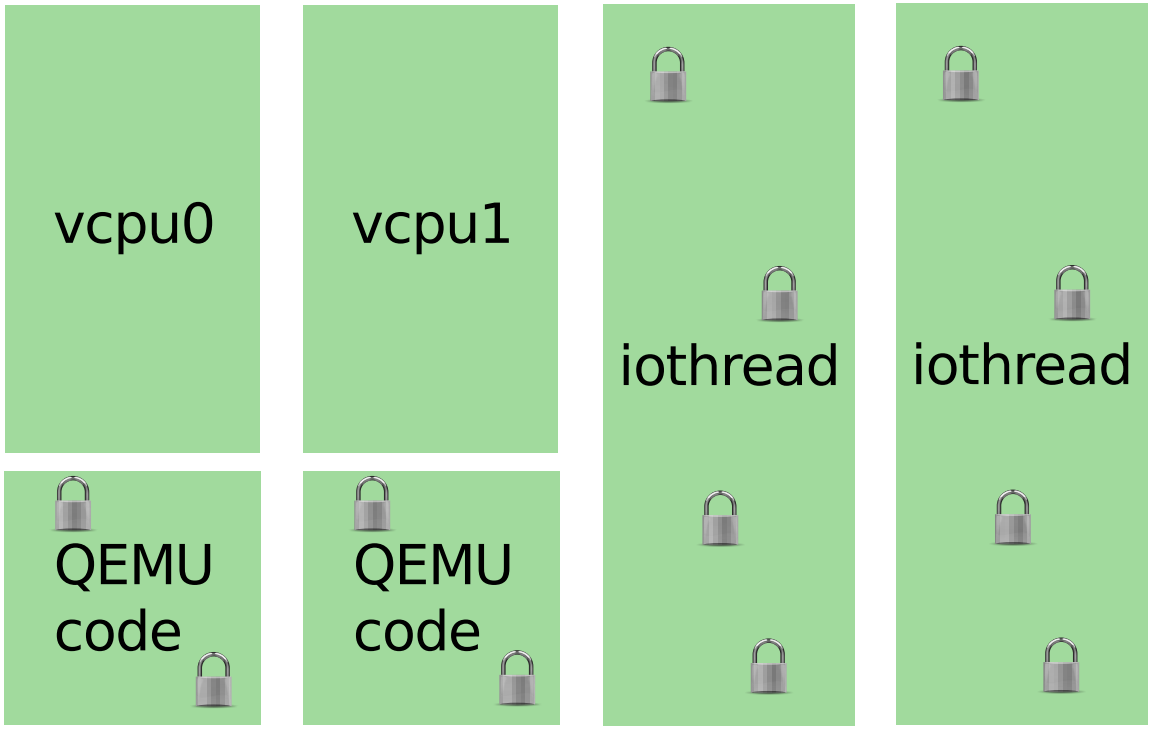
- Dedicated thread for raw image file I/O




- Duplicates QEMU code to achieve thread-safety
- Does not support image formats, I/O throttling, block jobs, NBD exports, monitor commands, hot unplug

# Solving the bottleneck properly

- User-configurable number of iothreads
- Devices can be bound to iothreads




- Introduces fine-grained locking into QEMU
- Block layer features work in a multi-iothread world

# Current work

- Per-AioContext timers by Alex Bligh
- Virtio thread-safe memory API conversion by Paolo Bonzini
- Thread-safe BH APIs by Ping Fan Liu
- AioContext acquire/release by Stefan Hajnoczi
- In other words, infrastructure is being put in place

# Future work

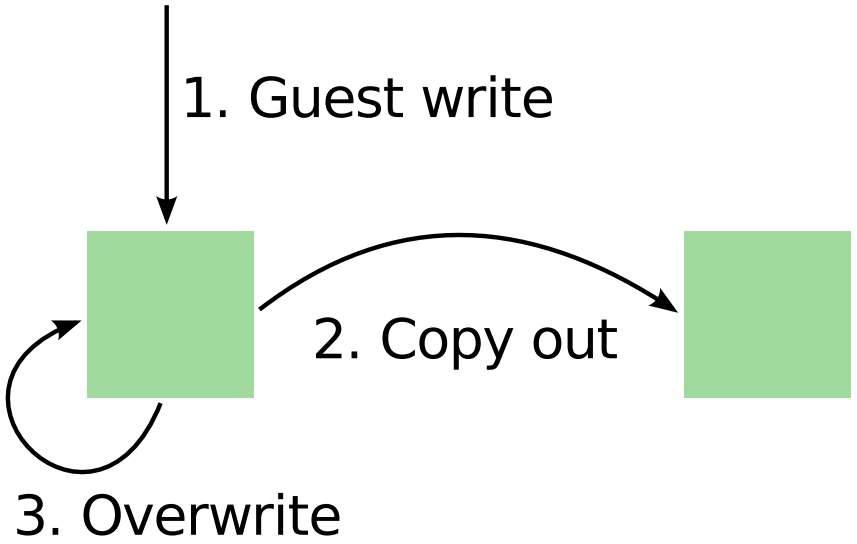
- Management APIs for defining iothreads (see Mike Roth's QContext presentation)
- Performance investigation to find best configurations
- Converting devices beyond virtio-blk

redhat.

# Part IV
# Image fleecing

# Point-in-time snapshots

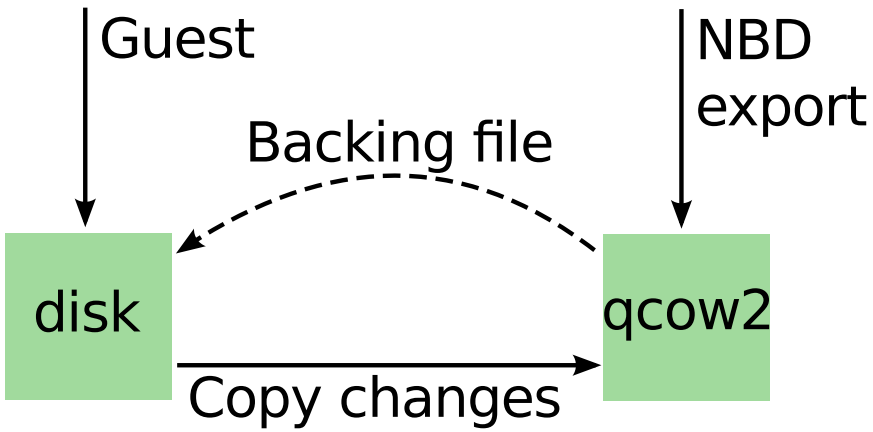
- drive-backup command copies out contents of a drive
- Data is copied out before guest modifications




- No cleanup required unlike deleting external snapshot
- Use case: backing up disk while guest is running
- Available in QEMU 1.6, by Dietmar Maurer and Stefan Hajnoczi

# Image fleecing

- Point-in-time snapshot as read-only NBD export
- Use qcow2 backing file feature instead of copying entire disk
- Throw away qcow2 file when NBD export is destroyed



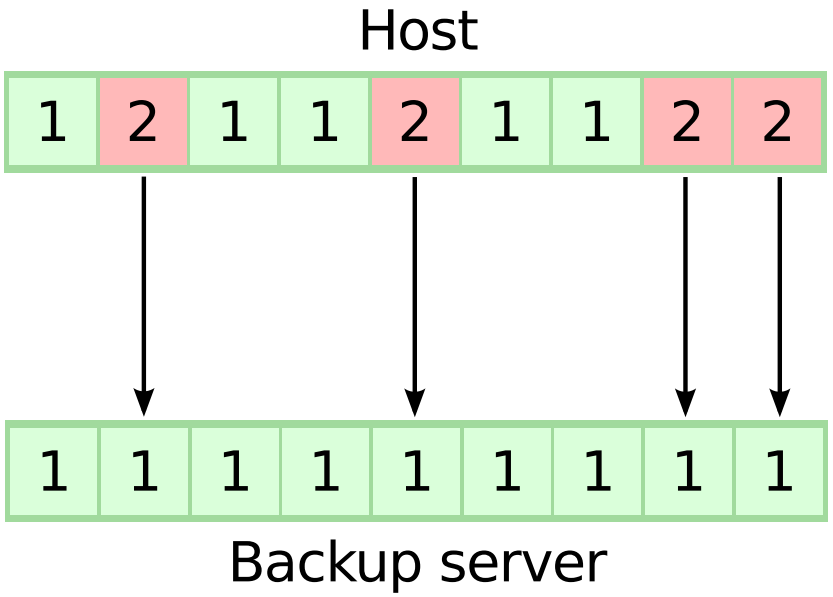

- Use case: backup applications, virus scanners, etc
- Patches being worked on by Fam Zheng and Ian Main

# Incremental backup

- Not yet implemented, looking for requirements & developers
- Only copy blocks that changed since last snapshot
- Maintain a persistent dirty bitmap
- Support for storage array, file system, and volume manager offload
- Implicit API: write out dirty blocks over NBD
- Explicit API: fetch dirty block bitmap
- Use case: efficient periodic backups

# Image syncing

- Extension of dirty bitmap idea, not implemented
- Dirty bitmap only supports one user at a time
- Per-block revision counter




- Multiple users can synchronize the image or copy dirty blocks
- Use case: opportunistic replication, multi-user incremental backup

redhat.

# Part V
# qemu-img map

# qemu-img map

- New bdrv_get_block_status() API by Paolo Bonzini
- Allocation, LBA mapping, and zero status information exposed
- New qemu-img map command for external programs
- Allocation information also used for sparse block migration, work by Peter Lieven


# The end.

Thanks for listening.