Live Block Operations:
Snapshots, Merging, and Mirroring

Jeff Cody
Red Hat
KVM Forum 2012, Barcelona
What this covers

- Background of live block operations
- Live snapshots
- Live snapshot merge:
  - Block Stream
    - QEMU v1.1
  - Block Commit
    - QEMU v1.3
- Drive Mirroring
  - QEMU v1.3
Live Block Operations
Live Block Operations

- Manipulate block storage devices and data, while guest is running
- Can be synchronous, or asynchronous
  - Synchronous operations occur in QAPI handler
  - Asynchronous operations use block jobs
Live Snapshots
Live Snapshots

- Always synchronous
- Refers to 'external' snapshots only
  - New snapshot must be an image format that supports backing files
- Transactional QMP Command, and atomic across multiple devices
  - Since 1.1
Live Snapshots

Two methods:
- Multiple devices
- Single device
Live Snapshots

Two methods:
  ● Multiple devices
  ● Single device

Example:

```json
{ "execute": "transaction", "arguments":
  { 'actions': [
    { 'type': 'blockdev-snapshot-sync', 'data':
      { 'device': 'virtio0', 'snapshot-file': '/tmp/driveA-snp-1.img' } },
    { 'type': 'blockdev-snapshot-sync', 'data':
      { 'device': 'virtio1', 'snapshot-file': '/tmp/driveB-snp-1.img' } }
  ] } }
```
Live Snapshots

Two methods:
- Multiple devices
- Single device

Single Drive Example:

```json
{  "execute": "transaction",  "arguments":
    {  'actions': [
        { 'type': 'blockdev-snapshot-sync', 'data':
            { 'device': 'virtio0', 'snapshot-file': '/tmp/driveA-snp-1.img' } ] }]
}
```

```json
{  "execute": "blockdev-snapshot-sync",  "arguments":
    {  "device": "virtio0",  "snapshot-file": "/tmp/driveA-snp-1.img"} }
```
Live Snapshots

QEMU Device Active Layer

QEMU Device

{ "execute": "blockdev-snapshot-sync", "arguments": {
  "device": "virtio0", "snapshot-file": "snap-1.img"
} }
# Live Snapshots

<table>
<thead>
<tr>
<th>BlockDriverState</th>
<th>BlockDriverState</th>
</tr>
</thead>
<tbody>
<tr>
<td>snap-1.img</td>
<td>RootBase.img</td>
</tr>
<tr>
<td>total_sectors</td>
<td>total_sectors</td>
</tr>
<tr>
<td>read_only</td>
<td>read_only</td>
</tr>
<tr>
<td>open_flags</td>
<td>open_flags</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>*job</td>
<td>*job</td>
</tr>
</tbody>
</table>

**Active BDS**
Live Snapshots

BlockDriverState

RootBase.img
- total_sectors
- read_only
- open_flags
  - *job

Active BDS

BlockDriverState

snap-1.img
- total_sectors
- read_only
- open_flags
  - *job
Live Snapshots

BlockDriverState

RootBase.img
- total_sectors
- read_only
- open_flags
  - *job

BlockDriverState

snap-1.img
- total_sectors
- read_only
- open_flags
  - *job

Active BDS
Live Snapshot - multiple devices

- Safe and atomic
- Each image created before live QEMU image chain is modified
- If any image file creation fails, operation is abandoned without touching the image chain
- On failure, will leave 'mouse droppings'
Live Merge
Live Merge

There is no “Live Merge” command!

Instead, we have two commands:
• block-stream
• block-commit
Block Stream and Block Commit

• Asynchronous; run as a block job while guest is live.

• Issues BLOCK_JOB_COMPLETED event on completion, with type 'stream' and 'commit'.
Block Commit and Stream

QAPI Commands:

```json
{ 'command': 'block-commit',
  'data': { 'device': 'str',
             '*base': 'str',
             'top': 'str',
             '*speed': 'int' } }
```

```json
{ 'command': 'block-stream',
  'data': { 'device': 'str',
             '*base': 'str',
             '*speed': 'int',
             '*on-error': 'BlockdevOnError' } }
```
## Block Stream and Block Commit Differences

<table>
<thead>
<tr>
<th>block-stream</th>
<th>block-commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Merges towards active layer</td>
<td>• Merges towards base</td>
</tr>
<tr>
<td>• Intermediate images remain valid</td>
<td>• Intermediate images become invalid</td>
</tr>
<tr>
<td>• Merges to the active layer only</td>
<td>• Can commit between any intermediate images below the active layer.</td>
</tr>
<tr>
<td>• Since v1.1</td>
<td>• Since v1.3</td>
</tr>
</tbody>
</table>
Sample image chain:

RootBase → Snap-1 → Snap-2 → Snap-3

Active BDS
Block Stream

Block-stream, from Snap-1:

- RootBase
- Snap-1
- Snap-2
- Snap-3

Active BDS: Snap-3

Copy sectors: RootBase to Snap-3
Block Commit

Sample image chain:

RootBase → Snap-1 → Snap-2 → Snap-3

Active BDS
Block-commit, from Snap-2 as top:

- RootBase
- Snap-1
- Snap-2
- Snap-3

Active BDS

Copy sectors:
- RootBase
- Snap-3

Active BDS
Block Commit
Block Commit

- Snap-A-1
- Snap-A-2
- Snap-B-1
- Snap-B-2
- Active
- Snap-1
- RootBase
- Snap-1 + Snap-B-1 + Snap-B-2
- Active

Copy sectors

Snap-A-1 (invalid)
Snap-A-2 (invalid)
Snap-B-1 (invalid)

All Children of Snap-1 are now invalid
Block Commit - Why use it?

- Potentially faster
  - Snapshots likely smaller than backing image
- Can collapse into RAW backing file
Block Commit - What happens

- Block job created
- Sectors copied in block job to 'base', above 'base' through 'top'.
- Backing file updated in the overlay of 'top'
- Intermediate images dropped from chain
Block Stream - What happens

- Block job created
- Sectors copied in block job to active layer, between 'base' and active layer.
- Backing file updated in active layer
- Unused images closed
Block Commit - What's Next

- Commit of active layer
  - Guest still writing to image
- Commit intermediate images in order
Block Stream - What's Next

- Stream to intermediate images, not just active layer.
Drive Mirror
Drive Mirror

- Mirrors the writes of a block device to new target
- 3 Sync Modes
  - Top
    - Copies the topmost image data to target, plus all new writes
  - Full
    - Copies all image data in the chain to target, plus all new writes
  - None
    - Copies only new writes to target

```
{ 'command': 'drive-mirror',
  'data': { 'device': 'str', 'target': 'str', '*format': 'str',
   'sync': 'MirrorSyncMode', '*mode': 'NewImageMode',
   '*speed': 'int', '*on-source-error': 'BlockdevOnError',
   '*on-target-error': 'BlockdevOnError' } }
```
Drive Mirror

- Uses dirty bitmap to know which sectors to copy over
- If copying 'static' data (Top and Full sync modes), dirty bitmap is first initialized.
Any Questions?

"Judge a man by his questions rather than his answers"

- Voltaire*
QEMU Live Block Operations: Snapshots, Merging, and Mirroring | Jeff Cody

More information:

http://wiki.qemu.org/Features/Snapshots

IRC: jtc on OFTC (#qemu)
Email: jcody@redhat.com
Any Questions?

“Judge a man by his questions rather than his answers”

- Voltaire*

* OK, that is actually an incorrect quote – the real quote is:
  It is easier to judge the mind of a man by his questions rather than his answers
  - Pierre-Marc-Gaston