Red Hat

# Bitmaps and NBD: Building Blocks of Change Block Tracking

Eric Blake (eblake @ redhat.com)
15:30 Friday October 30th, 2020

Incremental backups depend on change block tracking: if you know which portions of a disk image have changed, you can optimize your backups to visit just those portions. This talk explores how qemu uses **qcow2 dirty bitmaps** to track changed blocks, and how the **Network Block Device (NBD) protocol** can then expose that for use in backup scenarios.

# Change Block Tracking

Red Hat

Two common approaches to tracking changed blocks:

| Approach | Trade-offs |
| --- | --- |
| Generation tag | • Monotonically increasing tag tracked for each cluster<br>• Set of changes since a point in time is all clusters with generation id larger than the value at that point<br>• Multiple points in time tracked by same amount of metadata |
| Dirty bitmap | • Bitmap that gets updated with each changed cluster<br>• Less metadata storage required for all changes since a single point in time<br>• Multiple bitmaps required to track changes since multiple points in time |

Both approaches also have to balance granularity: more precision on what changed requires more metadata but can produce smaller incremental backups.

# Qcow2 images in qemu

Red Hat

- qcow2 is qemu's preferred image format
- qemu first used in-memory bitmaps internally to implement 'block-stream' in qemu v1.1 (2012)
- qemu v2.6 (2016) added extension header 0x23852875 to qcow2v3 for describing persistent bitmaps that live within the image
- an autoclear feature bit ensures that if any other program modifies the image without updating bitmaps, then the bitmaps are rendered inconsistent

See also this 2015 KVM Forum presentation: Backing Chain management in QEMU and libvirt, Eric Blake

# Network Block Device (NBD) support in qemu

Red Hat

- v0.10 (2008): qemu can connect as client over NBD, and qemu-nbd introduced as standalone server tool
- v1.3 (2012): qemu can serve images over NBD concurrently with live guest (nbd-server-add QMP command), facilitating live storage migration
- v2.6 (2016): qemu was the first NBD client and server implementation to support TLS
- v2.12 (2018): qemu was first NBD client and server implementation to support BLOCK_STATUS extension
- v3.0 (2018): qemu added "qemu:dirty-bitmap:*name*" to expose persistent dirty bitmaps

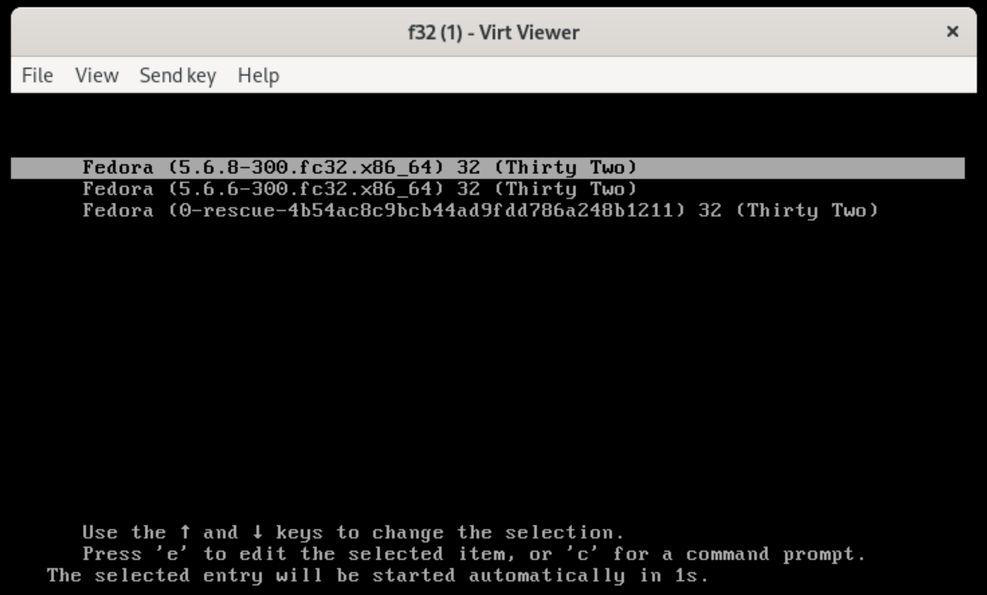
See also this 2019 KVM Forum presentation: Making the Most of NBD, Eric Blake and Rich Jones

## Preparing files for a guest

```
$ virt-builder fedora-32 -o Base1.qcow2 --format=qcow2 --hostname=f32 --ssh-inject=root --root-password=password:12345 --selinux-relabel
[   4.1] Downloading: http://builder.libguestfs.org/fedora-32.xz
[   5.1] Planning how to build this image
[   5.1] Uncompressing
[  20.9] Converting raw to qcow2
[  22.4] Opening the new disk
[  47.6] Setting a random seed
[  47.6] Setting the hostname: f32
[  47.7] SSH key inject: root
[  48.9] Setting passwords
[  50.1] SELinux relabelling
[  61.0] Finishing off
                   Output file: Base1.qcow2
                   Output size: 6.0G
                 Output format: qcow2
            Total usable space: 5.4G
                    Free space: 4.0G (74%)

$ qemu-img create -f qcow2 Base2.qcow2 100M
Formatting 'Base2.qcow2', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=104857600 lazy_refcounts=off refcount_bits=16

$
```

## Setting up for libvirt

```
$ virt-install --import --name=f32 --ram=2048 --os-variant=fedora32 --disk=path=Base1.qcow2,format=qcow2 --disk=path=Base2.qcow2,format=qcow2

Starting install...
Running graphical console command: virt-viewer --connect qemu:///system --wait f32
```



## Setting up for libvirt

```
$ virt-install --import --name=f32 --ram=2048 --os-variant=fedora32 --disk=path=Base1.qcow2,format=qcow2 --disk=path=Base2.qcow2,format=qcow2

Starting install...
Running graphical console command: virt-viewer --connect qemu:///system --wait f32

(virt-viewer:764891): GLib-GObject-WARNING **: 08:51:05.024: value "64" of type 'gint' is invalid or out of range for property 'desktop-width' of type 'gint'

(virt-viewer:764891): GLib-GObject-WARNING **: 08:51:05.024: value "64" of type 'gint' is invalid or out of range for property 'desktop-height' of type 'gint'
Domain creation completed.
You can restart your domain by running:
  virsh --connect qemu:///system start f32

$ virsh start f32
Domain f32 started


$ ip=root@$(virsh domifaddr f32 | sed -n "s,.*ipv4 *\(.*\)/.*,\1,p")

$ ssh $ip "bash -c 'mke2fs /dev/vdb; mkdir -p /mnt/img; mount /dev/vdb /mnt/img; touch /mnt/img/a; sync'"
The authenticity of host '192.168.122.94 (192.168.122.94)' can't be established.
ECDSA key fingerprint is SHA256:FicniEGhfdmL7oTJQPn0Vk29Ba4sT2gh7smE1H6U+UA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.122.94' (ECDSA) to the list of known hosts.
mke2fs 1.45.5 (07-Jan-2020)
Discarding device blocks: done
Creating filesystem with 102400 1k blocks and 25688 inodes
Filesystem UUID: cf5a6566-42b4-4987-8f4c-9e79010e4bdd
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done


$ virsh shutdown f32
Domain f32 is being shutdown


$
```

## Using 'qemu-img bitmap'

```
$ qemu-img info -f qcow2 Base2.qcow2
image: Base2.qcow2
file format: qcow2
virtual size: 100 MiB (104857600 bytes)
disk size: 1.76 MiB
cluster_size: 65536
Format specific information:
    compat: 1.1
    compression type: zlib
    lazy refcounts: false
    refcount bits: 16
    corrupt: false
    extended l2: false

$ qemu-img bitmap -f qcow2 --add Base2.qcow2 bmap0

$ qemu-img info -f qcow2 Base2.qcow2
image: Base2.qcow2
file format: qcow2
virtual size: 100 MiB (104857600 bytes)
disk size: 1.77 MiB
cluster_size: 65536
Format specific information:
    compat: 1.1
    compression type: zlib
    lazy refcounts: false
    bitmaps:
        [0]:
            flags:
                [0]: auto
            name: bmap0
            granularity: 65536
    refcount bits: 16
    corrupt: false
    extended l2: false

$
```

```
    lazy refcounts: false
    refcount bits: 16
    corrupt: false
    extended l2: false

$ qemu-img bitmap -f qcow2 --add Base2.qcow2 bmap0

$ qemu-img info -f qcow2 Base2.qcow2
image: Base2.qcow2
file format: qcow2
virtual size: 100 MiB (104857600 bytes)
disk size: 1.77 MiB
cluster_size: 65536
Format specific information:
    compat: 1.1
    compression type: zlib
    lazy refcounts: false
    bitmaps:
        [0]:
            flags:
                [0]: auto
            name: bmap0
            granularity: 65536
    refcount bits: 16
    corrupt: false
    extended l2: false

$ echo hello | guestfish -a Base2.qcow2 run : mount /dev/sda / : ls / : upload - /b
a
lost+found

$ qemu-nbd -f qcow2 -B bmap0 Base2.qcow2 &
[1] 765465

$ nbdinfo --map=qemu:dirty-bitmap:bmap0 nbd://localhost
        0      65536    1  dirty
    65536     196608    0  clean
   262144      65536    1  dirty
   327680     131072    0  clean
   458752     131072    1  dirty
   589824  104267776    0  clean


$
```

## More with qemu-nbd

```
$ qemu-img create -f qcow2 -b Base2.qcow2 -F qcow2 Overlay.qcow2
Formatting 'Overlay.qcow2', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=104857600 backing_file=Base2.qcow2 backing_fmt=qcow2 lazy_refcounts=off re
fcount_bits=16

$ qemu-img info --backing-chain -f qcow2 Overlay.qcow2
image: Overlay.qcow2
file format: qcow2
virtual size: 100 MiB (104857600 bytes)
disk size: 196 KiB
cluster_size: 65536
backing file: Base2.qcow2
backing file format: qcow2
Format specific information:
    compat: 1.1
    compression type: zlib
    lazy refcounts: false
    refcount bits: 16
    corrupt: false
    extended l2: false

image: Base2.qcow2
file format: qcow2
virtual size: 100 MiB (104857600 bytes)
disk size: 1.84 MiB
cluster_size: 65536
Format specific information:
    compat: 1.1
    compression type: zlib
    lazy refcounts: false
    bitmaps:
        [0]:
            flags:
                [0]: auto
            name: bmap0
            granularity: 65536
    refcount bits: 16
    corrupt: false
    extended l2: false

$
```

```
image: Base2.qcow2
file format: qcow2
virtual size: 100 MiB (104857600 bytes)
disk size: 1.84 MiB
cluster_size: 65536
Format specific information:
    compat: 1.1
    compression type: zlib
    lazy refcounts: false
    bitmaps:
        [0]:
            flags:
                [0]: auto
            name: bmap0
            granularity: 65536
    refcount bits: 16
    corrupt: false
    extended l2: false

$ echo world | guestfish -a Overlay.qcow2 run : mount /dev/sda / : upload - /c

$ qemu-nbd -f qcow2 -A Overlay.qcow2 &
[1] 765630

$ nbdinfo --map=qemu:allocation-depth nbd://localhost | head
         0      65536    1  local
     65536     196608    2  backing depth 2
    262144      65536    1  local
    327680     131072    2  backing depth 2
    458752     131072    1  local
    589824    7798784    0  unallocated
   8388608      65536    2  backing depth 2
   8454144     196608    0  unallocated
   8650752     262144    2  backing depth 2
   8912896    7864320    0  unallocated

$ qemu-img commit -f qcow2 Overlay.qcow2
[1]+  Done                    qemu-nbd -f qcow2 -A Overlay.qcow2
Image committed.

$ qemu-img bitmap --remove Base2.qcow2 bmap0

$
```

# Initial qemu push-mode incremental backup

- drive-backup introduced in 2015
- create a new bitmap with a full backup
- future incremental backups use the prior state of the bitmap to create external file, as well as reset the bitmap

Initial state

Image.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | - | A | A | - | A | - | - |

See also this 2015 KVM Forum presentation: Incremental backups: Good things come in in small packages!, John Snow

# Initial qemu push-mode incremental backup

- drive-backup introduced in 2015
- create a new bitmap with a full backup
- future incremental backups use the prior state of the bitmap to create external file, as well as reset the bitmap

Full backup

Image.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |
| - | - | - | - | - | - | - | - |

FullBackup.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |

See also this 2015 KVM Forum presentation: Incremental backups: Good things come in in small packages!, John Snow

# Initial qemu push-mode incremental backup

- drive-backup introduced in 2015
- create a new bitmap with a full backup
- future incremental backups use the prior state of the bitmap to create external file, as well as reset the bitmap

Time elapses

Image.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **-** | **A** | **B** | **B** | **B** | **-** | **-** |
| - | - | - | X | X | X | - | - |

FullBackup.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |

See also this 2015 KVM Forum presentation: Incremental backups: Good things come in in small packages!, John Snow

# Initial qemu push-mode incremental backup

- drive-backup introduced in 2015
- create a new bitmap with a full backup
- future incremental backups use the prior state of the bitmap to create external file, as well as reset the bitmap

First incremental backup

Image.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **-** | **A** | **B** | **B** | **B** | **-** | **-** |
| - | - | - | - | - | - | - | - |

FullBackup.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |

IncBackup1.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **-** | **-** | **-** | **B** | **B** | **B** | **-** | **-** |

See also this 2015 KVM Forum presentation: Incremental backups: Good things come in in small packages!, John Snow

# Initial qemu push-mode incremental backup

- drive-backup introduced in 2015
- create a new bitmap with a full backup
- future incremental backups use the prior state of the bitmap to create external file, as well as reset the bitmap

More time elapses

Image.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **-** | **C** | **C** | **B** | **B** | **-** | **-** |
| - | - | X | X | - | - | - | - |

FullBackup.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |

IncBackup1.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **-** | **-** | **-** | **B** | **B** | **B** | **-** | **-** |

See also this 2015 KVM Forum presentation: Incremental backups: Good things come in in small packages!, John Snow

# Initial qemu push-mode incremental backup

- drive-backup introduced in 2015
- create a new bitmap with a full backup
- future incremental backups use the prior state of the bitmap to create external file, as well as reset the bitmap

Second incremental backup

Image.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **-** | **C** | **C** | **B** | **B** | **-** | **-** |
| - | - | - | - | - | - | - | - |

FullBackup.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |

IncBackup1.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **-** | **-** | **-** | **B** | **B** | **B** | **-** | **-** |

IncBackup2.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **-** | **-** | **C** | **C** | **-** | **-** | **-** | **-** |

See also this 2015 KVM Forum presentation: Incremental backups: Good things come in in small packages!, John Snow

# Initial qemu pull-mode differential backup

- expose the bitmap over NBD for third-party access
- create a new bitmap on each backup
- transfer which bitmap is enabled; all others are disabled
- merge sequence of bitmaps to perform differential backup

Initial state

Image.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | - | A | A | - | A | - | - |

See also this 2018 KVM Forum presentation: Facilitating Incremental Backup, Eric Blake

# Initial qemu pull-mode differential backup

- expose the bitmap over NBD for third-party access
- create a new bitmap on each backup
- transfer which bitmap is enabled; all others are disabled
- merge sequence of bitmaps to perform differential backup

Full backup

Image.qcow2

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **A** | - | **A** | **A** | - | **A** | - | - |
| *b0* | - | - | - | - | - | - | - | - |

FullBackup.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | - | **A** | **A** | - | **A** | - | - |

See also this 2018 KVM Forum presentation: Facilitating Incremental Backup, Eric Blake

# Initial qemu pull-mode differential backup

- expose the bitmap over NBD for third-party access
- create a new bitmap on each backup
- transfer which bitmap is enabled; all others are disabled
- merge sequence of bitmaps to perform differential backup

Time elapses

Image.qcow2

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **A** | **-** | **A** | **B** | **B** | **B** | **-** | **-** |
| *b0* | - | - | - | X | X | X | - | - |

FullBackup.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |

See also this 2018 KVM Forum presentation: Facilitating Incremental Backup, Eric Blake

# Initial qemu pull-mode differential backup

- expose the bitmap over NBD for third-party access
- create a new bitmap on each backup
- transfer which bitmap is enabled; all others are disabled
- merge sequence of bitmaps to perform differential backup

First incremental backup

Image.qcow2

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **A** | **-** | **A** | **B** | **B** | **B** | **-** | **-** |
| *b0* | - | - | - | X | X | X | - | - |
| *b1* | - | - | - | - | - | - | - | - |

FullBackup.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |

IncBackup1.qcow2

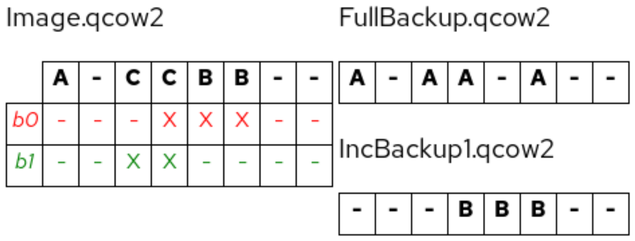
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **-** | **-** | **-** | **B** | **B** | **B** | **-** | **-** |

See also this 2018 KVM Forum presentation: Facilitating Incremental Backup, Eric Blake

# Initial qemu pull-mode differential backup

- expose the bitmap over NBD for third-party access
- create a new bitmap on each backup
- transfer which bitmap is enabled; all others are disabled
- merge sequence of bitmaps to perform differential backup

More time elapses

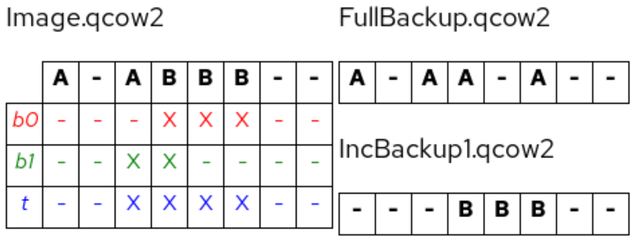



See also this 2018 KVM Forum presentation: Facilitating Incremental Backup, Eric Blake

# Initial qemu pull-mode differential backup

- expose the bitmap over NBD for third-party access
- create a new bitmap on each backup
- transfer which bitmap is enabled; all others are disabled
- merge sequence of bitmaps to perform differential backup

Differential backup







See also this 2018 KVM Forum presentation: Facilitating Incremental Backup, Eric Blake

# Mixing backups with external snapshots

- managing disabled bitmaps in libvirt proved to be too complex
- each checkpoint creates a new bitmap, but all bitmaps are left active
- bitmaps are copied or merged alongside block job tasks

Initial state

Base.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | - | **A** | **A** | - | **A** | - | - |

# Mixing backups with external snapshots

- managing disabled bitmaps in libvirt proved to be too complex
- each checkpoint creates a new bitmap, but all bitmaps are left active
- bitmaps are copied or merged alongside block job tasks

Full backup

Base.qcow2

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |
| *b0* | - | - | - | - | - | - | - | - |

FullBackup.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |

# Mixing backups with external snapshots

- managing disabled bitmaps in libvirt proved to be too complex
- each checkpoint creates a new bitmap, but all bitmaps are left active
- bitmaps are copied or merged alongside block job tasks

Time elapses

Base.qcow2

| | **A** | **-** | **A** | **B** | **B** | **B** | **-** | **-** |
|---|---|---|---|---|---|---|---|---|
| *b0* | - | - | - | X | X | X | - | - |

FullBackup.qcow2

| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |
|---|---|---|---|---|---|---|---|

# Mixing backups with external snapshots

- managing disabled bitmaps in libvirt proved to be too complex
- each checkpoint creates a new bitmap, but all bitmaps are left active
- bitmaps are copied or merged alongside block job tasks

## Create external snapshot

Base.qcow2

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **A** | **-** | **A** | **B** | **B** | **B** | **-** | **-** |
| *b0* | - | - | - | X | X | X | - | - |

FullBackup.qcow2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |

Overlay.qcow2

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| *b0* | - | - | - | - | - | - | - | - |

# Mixing backups with external snapshots

- managing disabled bitmaps in libvirt proved to be too complex
- each checkpoint creates a new bitmap, but all bitmaps are left active
- bitmaps are copied or merged alongside block job tasks

More time elapses

Base.qcow2

| | **A** | **-** | **A** | **B** | **B** | **B** | **-** | **-** |
|---|---|---|---|---|---|---|---|---|
| *b0* | - | - | - | X | X | X | - | - |

FullBackup.qcow2

| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |
|---|---|---|---|---|---|---|---|

Overlay.qcow2

| | **-** | **-** | **C** | **C** | **-** | **-** | **-** | **-** |
|---|---|---|---|---|---|---|---|---|
| *b0* | - | - | X | X | - | - | - | - |

# Mixing backups with external snapshots

- managing disabled bitmaps in libvirt proved to be too complex
- each checkpoint creates a new bitmap, but all bitmaps are left active
- bitmaps are copied or merged alongside block job tasks

First incremental backup

Base.qcow2

| | **A** | **-** | **A** | **B** | **B** | **B** | **-** | **-** |
|---|---|---|---|---|---|---|---|---|
| *b0* | - | - | - | X | X | X | - | - |

FullBackup.qcow2

| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |
|---|---|---|---|---|---|---|---|

IncBackup1.qcow2

| **-** | **-** | **C** | **C** | **B** | **B** | **-** | **-** |
|---|---|---|---|---|---|---|---|

Overlay.qcow2

| | **-** | **-** | **C** | **C** | **-** | **-** | **-** | **-** |
|---|---|---|---|---|---|---|---|---|
| *b0* | - | - | X | X | - | - | - | - |
| *b1* | - | - | - | - | - | - | - | - |
| *t* | - | - | X | X | X | X | - | - |

# Mixing backups with external snapshots

- managing disabled bitmaps in libvirt proved to be too complex
- each checkpoint creates a new bitmap, but all bitmaps are left active
- bitmaps are copied or merged alongside block job tasks

Yet more time goes by

Base.qcow2

| | **A** | **-** | **A** | **B** | **B** | **B** | **-** | **-** |
|---|---|---|---|---|---|---|---|---|
| *b0* | - | - | - | X | X | X | - | - |

FullBackup.qcow2

| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |
|---|---|---|---|---|---|---|---|

IncBackup1.qcow2

| **-** | **-** | **C** | **C** | **B** | **B** | **-** | **-** |
|---|---|---|---|---|---|---|---|

Overlay.qcow2

| | **-** | **-** | **C** | **C** | **-** | **-** | **D** | **-** |
|---|---|---|---|---|---|---|---|---|
| *b0* | - | - | X | X | - | - | X | - |
| *b1* | - | - | - | - | - | - | X | - |

# Mixing backups with external snapshots

- managing disabled bitmaps in libvirt proved to be too complex
- each checkpoint creates a new bitmap, but all bitmaps are left active
- bitmaps are copied or merged alongside block job tasks

## Block commit

Base.qcow2

| | **A** | **-** | **C** | **C** | **B** | **B** | **D** | **-** |
|---|---|---|---|---|---|---|---|---|
| *b0* | - | - | X | X | X | X | X | - |
| *b1* | - | - | - | - | - | - | X | - |

FullBackup.qcow2

| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |
|---|---|---|---|---|---|---|---|

IncBackup1.qcow2

| **-** | **-** | **C** | **C** | **B** | **B** | **-** | **-** |
|---|---|---|---|---|---|---|---|

# Mixing backups with external snapshots

- managing disabled bitmaps in libvirt proved to be too complex
- each checkpoint creates a new bitmap, but all bitmaps are left active
- bitmaps are copied or merged alongside block job tasks

## Second incremental backup

Base.qcow2

| | **A** | **-** | **C** | **C** | **B** | **B** | **D** | **-** |
|---|---|---|---|---|---|---|---|---|
| *b0* | - | - | X | X | X | X | X | - |
| *b1* | - | - | - | - | - | - | X | - |
| *t* | - | - | - | - | - | - | X | - |

FullBackup.qcow2

| **A** | **-** | **A** | **A** | **-** | **A** | **-** | **-** |
|---|---|---|---|---|---|---|---|

IncBackup1.qcow2

| **-** | **-** | **C** | **C** | **B** | **B** | **-** | **-** |
|---|---|---|---|---|---|---|---|

IncBackup2.qcow2

| **-** | **-** | **-** | **-** | **-** | **-** | **D** | **-** |
|---|---|---|---|---|---|---|---|

## Demonstration with libvirt

```
$ virsh start f32
Domain f32 started


$ virsh backup-begin f32
error: Operation not supported: incremental backup is not supported yet


$ virsh shutdown f32
Domain f32 is being shutdown


$ virsh dumpxml f32 | sed 's|<domain type=.kvm.>|<domain type="kvm" xmlns:qemu="http://libvirt.org/schemas/domain/qemu/1.0"> <qemu:capabilities><qemu:add capability="incremental-backup"/></qemu:capabilities>|' > f32.xml

$ virsh define f32.xml
Domain f32 defined from f32.xml


$
```

# Full push backup with bitmap creation

```
$ virsh start f32
Domain f32 started


$ cat backup.xml
<domainbackup mode='push'>
  <disks>
    <disk name='vda' type='file'>
      <target file='/home/eblake/kvmforum/2020/talk/demo.d/Full.qcow2'/>
      <driver type='qcow2'/>
    </disk>
  </disks>
</domainbackup>

$ cat checkpoint.xml
<domaincheckpoint>
  <name>check1</name>
</domaincheckpoint>

$ qemu-img create -f qcow2 Full.qcow2 6G
Formatting 'Full.qcow2', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=6442450944 lazy_refcounts=off refcount_bits=16

$ virsh backup-begin --reuse-external f32 backup.xml checkpoint.xml
Backup started


$ virsh backup-dumpxml f32
<domainbackup mode='push'>
  <disks>
    <disk name='vda' backup='yes' type='file' backupmode='full'>
      <driver type='qcow2'/>
      <target file='/home/eblake/kvmforum/2020/talk/demo.d/Full.qcow2'/>
    </disk>
    <disk name='vdb' backup='no'/>
  </disks>
</domainbackup>


$
```

## External snapshot

```
$ cat snapshot.xml
<domainsnapshot>
  <name>snap1</name>
  <disks>
    <disk name='vda'>
      <source file='/home/eblake/kvmforum/2020/talk/demo.d/Overlay.qcow2'/>
    </disk>
    <disk name='vdb' snapshot='no'/>
  </disks>
</domainsnapshot>

$ virsh snapshot-create --disk-only f32 snapshot.xml
Domain snapshot snap1 created from 'snapshot.xml'

$ virsh domblklist f32
 Target   Source
------------------------------------------------------------
 vda      /home/eblake/kvmforum/2020/talk/demo.d/Overlay.qcow2
 vdb      /home/eblake/kvmforum/2020/talk/demo.d/Base2.qcow2


$ qemu-img info --backing-chain Overlay.qcow2
qemu-img: Could not open 'Overlay.qcow2': Could not open 'Overlay.qcow2': Permission denied

$
```

```
$ qemu-img info --backing-chain Overlay.qcow2
qemu-img: Could not open 'Overlay.qcow2': Could not open 'Overlay.qcow2': Permission denied

$ sudo -u qemu qemu-img info --backing-chain -U Overlay.qcow2
[sudo] password for eblake:
image: Overlay.qcow2
file format: qcow2
virtual size: 6 GiB (6442450944 bytes)
disk size: 6.89 MiB
cluster_size: 65536
backing file: /home/eblake/kvmforum/2020/talk/demo.d/Base1.qcow2
backing file format: qcow2
Format specific information:
    compat: 1.1
    compression type: zlib
    lazy refcounts: false
    refcount bits: 16
    corrupt: false

image: /home/eblake/kvmforum/2020/talk/demo.d/Base1.qcow2
file format: qcow2
virtual size: 6 GiB (6442450944 bytes)
disk size: 1.36 GiB
cluster_size: 65536
Format specific information:
    compat: 1.1
    compression type: zlib
    lazy refcounts: false
    bitmaps:
        [0]:
            flags:
                [0]: auto
            name: check1
            granularity: 65536
    refcount bits: 16
    corrupt: false

$ virsh shutdown f32
Domain f32 is being shutdown

$
```

```
$ qemu-img info --backing-chain Overlay.qcow2
image: Overlay.qcow2
file format: qcow2
virtual size: 6 GiB (6442450944 bytes)
disk size: 7.95 MiB
cluster_size: 65536
backing file: /home/eblake/kvmforum/2020/talk/demo.d/Base1.qcow2
backing file format: qcow2
Format specific information:
    compat: 1.1
    compression type: zlib
    lazy refcounts: false
    bitmaps:
        [0]:
            flags:
                [0]: auto
            name: check1
            granularity: 65536
    refcount bits: 16
    corrupt: false
    extended l2: false

image: /home/eblake/kvmforum/2020/talk/demo.d/Base1.qcow2
file format: qcow2
virtual size: 6 GiB (6442450944 bytes)
disk size: 1.36 GiB
cluster_size: 65536
Format specific information:
    compat: 1.1
    compression type: zlib
    lazy refcounts: false
    bitmaps:
        [0]:
            flags:
                [0]: auto
            name: check1
            granularity: 65536
    refcount bits: 16
    corrupt: false
    extended l2: false

$
```

```
file format: qcow2
virtual size: 6 GiB (6442450944 bytes)
disk size: 7.95 MiB
cluster_size: 65536
backing file: /home/eblake/kvmforum/2020/talk/demo.d/Base1.qcow2
backing file format: qcow2
Format specific information:
    compat: 1.1
    compression type: zlib
    lazy refcounts: false
    bitmaps:
        [0]:
            flags:
                [0]: auto
            name: check1
            granularity: 65536
    refcount bits: 16
    corrupt: false
    extended l2: false

image: /home/eblake/kvmforum/2020/talk/demo.d/Base1.qcow2
file format: qcow2
virtual size: 6 GiB (6442450944 bytes)
disk size: 1.36 GiB
cluster_size: 65536
Format specific information:
    compat: 1.1
    compression type: zlib
    lazy refcounts: false
    bitmaps:
        [0]:
            flags:
                [0]: auto
            name: check1
            granularity: 65536
    refcount bits: 16
    corrupt: false
    extended l2: false

$ virsh start f32
Domain f32 started


$
```

## Incremental pull mode backup

```
$ cat backup2.xml
<domainbackup mode="pull">
  <incremental>check1</incremental>
  <server name="localhost" port="10809"/>
</domainbackup>

$ cat checkpoint2.xml
<domaincheckpoint>
  <name>check2</name>
</domaincheckpoint>

$ ip=root@$(virsh domifaddr f32 | sed -n "s,.*ipv4 *\(.*\)/.*,\1,p")

$ ssh $ip "bash -c 'touch /before; sync'"

$ virsh backup-begin f32 backup2.xml checkpoint2.xml
Backup started


$ ssh $ip "bash -c 'touch /after; sync'"

$ virsh backup-dumpxml f32
<domainbackup mode='pull'>
  <incremental>check1</incremental>
  <server transport='tcp' name='localhost' port='10809'/>
  <disks>
    <disk name='vda' backup='yes' type='file' backupmode='incremental' incremental='check1' exportname='vda' exportbitmap='backup-vda'>
      <driver type='qcow2'/>
      <scratch file='/home/eblake/kvmforum/2020/talk/demo.d/Overlay.qcow2.check2'/>
    </disk>
    <disk name='vdb' backup='yes' type='file' backupmode='incremental' incremental='check1' exportname='vdb' exportbitmap='backup-vdb'>
      <driver type='qcow2'/>
      <scratch file='/home/eblake/kvmforum/2020/talk/demo.d/Base2.qcow2.check2'/>
    </disk>
  </disks>
</domainbackup>


$
```

```
</domainbackup>


$ nbdinfo --list nbd://localhost
protocol: newstyle-fixed without TLS
export="vda":
        export-size: 6442450944
        contexts:
                qemu:dirty-bitmap:backup-vda
                base:allocation
        is_rotational: false
        is_read_only: true
        can_cache: true
        can_df: true
        can_fast_zero: false
        can_flush: true
        can_fua: true
        can_multi_conn: true
        can_trim: false
        can_zero: false
        block_size_minimum: 1
        block_size_preferred: 4096
        block_size_maximum: 33554432
export="vdb":
        export-size: 104857600
        contexts:
                qemu:dirty-bitmap:backup-vdb
                base:allocation
        is_rotational: false
        is_read_only: true
        can_cache: true
        can_df: true
        can_fast_zero: false
        can_flush: true
        can_fua: true
        can_multi_conn: true
        can_trim: false
        can_zero: false
        block_size_minimum: 1
        block_size_preferred: 4096
        block_size_maximum: 33554432


$
```

## Testing the backup

```
$ nbdinfo --map=qemu:dirty-bitmap:backup-vda nbd://localhost/vda | head
         0     2097152    0  clean
   2097152       65536    1  dirty
   2162688   538640384    0  clean
 540803072     2162688    1  dirty
 542965760   264437760    0  clean
 807403520       65536    1  dirty
 807469056   913244160    0  clean
1720713216      131072    1  dirty
1720844288    39321600    0  clean
1760165888       65536    1  dirty

$ cat copy.sh
# $1 is source, $2 is destination
qemu-img create -f qcow2 -b "$1" -F raw "$2"
nbdinfo --map=qemu:dirty-bitmap:backup-${1##*/} "$1" |
while read offset len x type; do
  [ "$type" = dirty ] || continue
  qemu-io -C -c "r $offset $len" -f qcow2 "$2"
done
qemu-img rebase -u -f qcow2 -b Full.qcow2 -F qcow2 "$2"

$ ./copy.sh nbd://localhost/vda IncBack.qcow2
```

```
read 65536/65536 bytes at offset 5451874304
64 KiB, 1 ops; 00.00 sec (76.529 MiB/sec and 1224.4713 ops/sec)
read 65536/65536 bytes at offset 5452005376
64 KiB, 1 ops; 00.00 sec (79.127 MiB/sec and 1266.0391 ops/sec)
read 65536/65536 bytes at offset 5453119488
64 KiB, 1 ops; 00.00 sec (80.271 MiB/sec and 1284.3417 ops/sec)

$ virsh domjobabort f32


$ virsh shutdown f32
Domain f32 is being shutdown


$ guestfish -a IncBack.qcow2 run : mount /dev/sda4 / : ls /
before
bin
boot
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var

$ ls -l *.qcow2
-rw-r--r--. 1 qemu   qemu   1459552288 Oct 29 08:58 Base1.qcow2
-rw-r--r--. 1 eblake eblake    2228256 Oct 29 09:01 Base2.qcow2
-rw-r--r--. 1 eblake eblake 1452998656 Oct 29 08:58 Full.qcow2
-rw-r--r--. 1 eblake eblake   67633152 Oct 29 09:01 IncBack.qcow2
-rw-------. 1 eblake eblake   42991680 Oct 29 09:01 Overlay.qcow2

$
```

# In summary

Red Hat

- qcow2 images with dirty bitmaps can do change block tracking.
- NBD protocol exposing those bitmaps opens possibilities.
- Libvirt aids incremental backup across backing chains.
- Expect more use cases as more software plays with incremental backups.

## *Get it*

- qemu ≥ 5.1 https://qemu.org plus patches:
  - block-dirty-bitmap-populate https://lists.gnu.org/archive/html/qemu-devel/2020-09/msg00978.html
  - more contexts in qemu-nbd https://lists.gnu.org/archive/html/qemu-devel/2020-10/msg02708.html
- libnbd ≥ 1.5.4 https://github.com/libguestfs/libnbd plus patches:
  - honor qemu-nbd -A https://www.redhat.com/archives/libguestfs/2020-October/msg00077.html
- libvirt ≥ 6.7.0 https://libvirt.org
- This talk: https://repo.or.cz/eblake-techtalks.git
- Presentation software: Tech Talk PSE 1.2

## *Any Questions?*