Internals of incremental backup handling in qemu

- Glossary
- Relationships of bitmaps, checkpoints and VM disks
- Integration with external snapshots
  - External snapshot terminology
  - Handling of bitmaps during snapshots
  - Manipulating bitmaps in shell
  - Checking bitmap health
  - Creating external snapshots manually
  - Committing external snapshots manually

Libvirt's implementation of incremental backups in the `qemu` driver uses qemu's `block-dirty-bitmaps` under the hood to track the guest visible disk state changes corresponding to the points in time described by a libvirt checkpoint.

There are some semantica implications with how libvirt creates and manages the bitmaps which de-facto become API as they are written into the disk images, and this document will try to summarize them.

Glossary

See the knowledge base article on domain state capture for a deeper explanation of some of the concepts.

Checkpoint

> A libvirt object which represents a named point in time of the life of the vm where libvirt tracks writes the VM has done, thereby allowing a backup of only the blocks which changed. Note that state of the VM memory is _not_ captured.

A checkpoint can be created either explicitly via the corresponding API (although this isn't very useful on its own), or simultaneously with an incremental or full backup of the VM using the `virDomainBackupBegin` API which allows a next backup to only copy the differences.

Backup

A copy of either all blocks of selected disks (full backup) or blocks changed since a checkpoint (incremental backup) at the time the backup job was started. (Blocks modified while the backup job is running are not part of the backup!)

Snapshot

Similarly to a checkpoint it's a point in time in the lifecycle of the VM but the state of the VM including memory is captured at that point allowing returning to the state later.

Blockjob

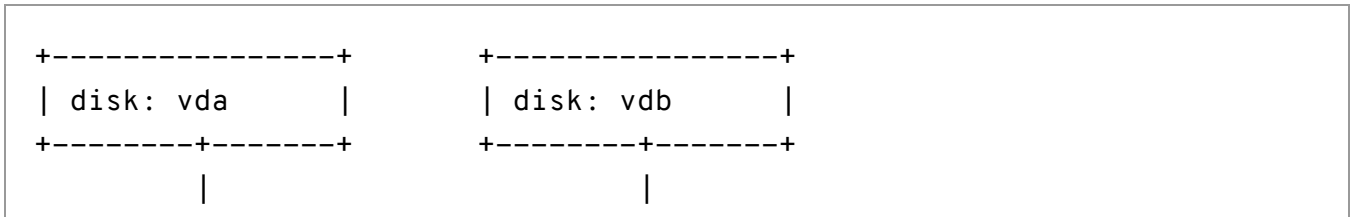
A long running job which modifies the shape and/or location of the disk backing chain (images storing the disk contents). Libvirt supports `block pull` where data is moved up the chain towards the active layer, `block commit` where data is moved down the chain towards the base/oldest image. These blockjobs always remove images from the backing chain. Lastly `block copy` where image is moved to a different location (and possibly collapsed moving all of the data into the new location into the one image).

block-dirty-bitmap (bitmap)

A data structure in qemu tracking which blocks were written by the guest OS since the bitmap was created.

Relationships of bitmaps, checkpoints and VM disks

When a checkpoint is created libvirt creates a block-dirty-bitmap for every configured VM disk named the same way as the chcheckpoint. The bitmap is actively recording which blocks were changed by the guest OS from that point on. Other bitmaps are not impacted by any way as they are self-contained:



```
+--------v-------+       +--------v-------+
| vda-1.qcow2    |       | vdb-1.qcow2    |
|                |       |                |
| bitmaps: chk-a |       | bitmaps: chk-a |
|          chk-b |       |          chk-b |
|                |       |                |
+----------------+       +----------------+
```

Bitmaps are created at the same time to track changes to all disks in sync and are active and persisted in the QCOW2 image. Other formats currently don't support this feature.

Modification of bitmaps outside of libvirt is not recommended, but when adhering to the same semantics which the document will describe it should be safe to do so, even if we obviously can't guarantee that.

Integration with external snapshots

External snapshot terminology

External snapshots on a disk level consist of layered chains of disk images. An image in the chain can have a `backing image` placed below. Any chunk in the current image which was not written explicitly is transparent and if read the data from the backing image is passed through. An image placed on top of the current image is called `overlay`.

The bottommost backing image at the end of the chain is also usually described as `base image`.

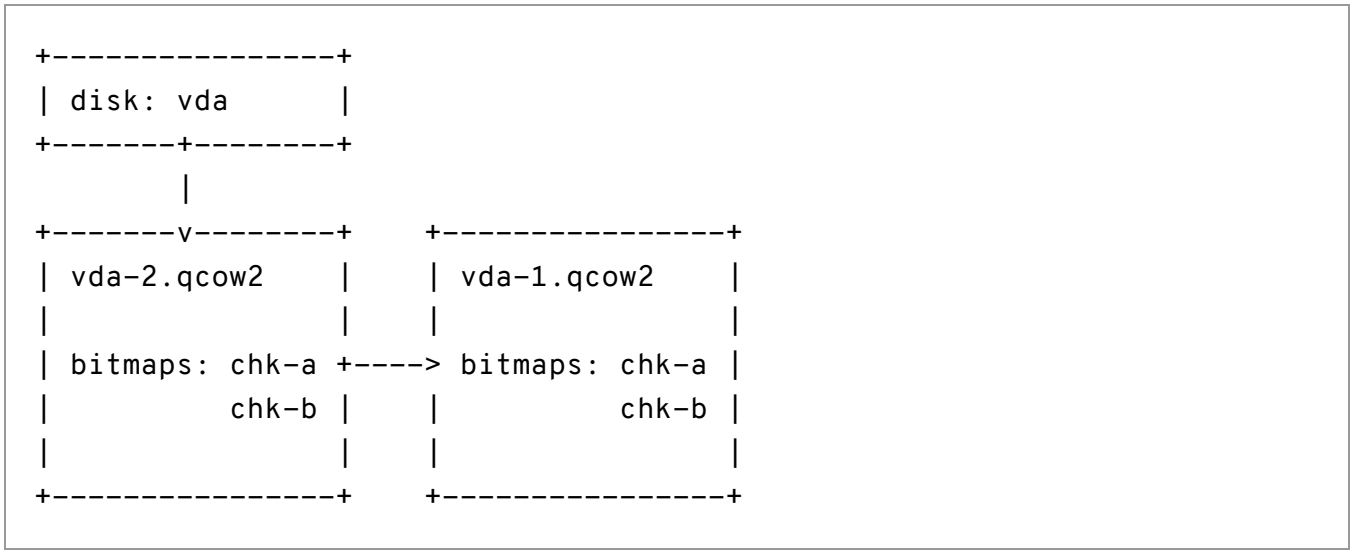
The topmost overlay is the image which is being written to by the VM and is also described as the `active` layer or image.

Handling of bitmaps during snapshots

Creating an external snapshot involves adding a overlay on top of the previously active image. Libvirt requires that all `block-dirty-bitmaps` which correspond to the checkpoint must be created in the new overlay before any write from the guest reaches the overlay to continue tracking which blocks are dirtied.

Since there are no new bitmaps created by `qemu` or `qemu-img` by default when creating an overlay, we need to re-create the appropriate bitmaps (see below) in the new overlay based on the previously active bitmaps in the active image. The new bitmaps are created with the same granularity.

After taking a snapshot of the `vda` disk from the example above placed into `vda-2.qcow2` the following topology will be created:



Manipulating bitmaps in shell

NOTE: Any of the examples expect that the full image chain isn't used by any running VM at the time.

`qemu-img info` command reports information about dirty bitmaps in an image:

```
$ qemu-img info -f qcow2 vda-1.qcow2
 image: vda-1.qcow2
 file format: qcow2
 virtual size: 100 MiB (104857600 bytes)
 disk size: 220 KiB
 cluster_size: 65536
 Format specific information:
     compat: 1.1
     compression type: zlib
     lazy refcounts: false
     bitmaps:
         [0]:
             flags:
                 [0]: in-use
                 [1]: auto
             name: chk-a
             granularity: 65536
         [1]:
             flags:
                 [0]: auto
```

```
            name: chk-b
            granularity: 65536
    refcount bits: 16
    corrupt: false
```

The `flags` have following meanings:

`auto` - **recording**

> The bitmap is automatically activated when the image is opened for writing and thus it's actively recording writes.

`in-use` - **inconsistent**

> The bitmap was not properly saved when the qemu process was shut down last time thus didn't consistently record all the changed sectors.

It's recommended to use `--output=json` parameter to work with a machine readable output rather than trying to process the human readable output by scripts. For processing JSON in shell the `jq` tool can be used.

The `qemu-img bitmap` command allows modification of block-dirty-bitmaps of an offline image. It supports the following operations relevant to this document (see man page for full list of operations):

`--add NAME`

Creates a new bitmap named `NAME`. Optionally `-g` can be used to specify granularity.

`--remove NAME`

Deletes bitmap `NAME`.

`--merge SRCBITMAP -b SRCFILE -F SRCFILEFMT DSTBITMAP`

Merges bitmap `SRCBITMAP` from `SRCFILE` into `DSTBITMAP`.

Checking bitmap health

QEMU optimizes disk writes by only updating the bitmaps in certain cases. This also can cause problems in cases when e.g. QEMU crashes.

For a chain of corresponding bitmaps in a backing chain images to be considered valid and eligible for use for an incremental backup with `virDomainBackupBegin` the bitmaps intended to

be used must conform to the following rules:

1. active/topmost image must contain the bitmap
2. if a bitmap with the same name is contained in one of the backing images it must be a contiguous subchain starting from the topmost image which contains the bitmaps (no gaps)
3. all of the above bitmaps must be marked as **recording**
4. all of the above bitmaps must not be **inconsistent**

(See also the `qemuBlockBitmapChainIsValid` helper method in `src/qemu/qemu_block.c`)

Creating external snapshots manually

To create the same topology outside of libvirt (e.g when doing snapshots offline) the following pseudo-algorithm ensures that the new image after snapshot will work with backups. `OVERLAY` corresponds to the new overlay image, `ACTIVE` corresponds to the topmost image of the active chain prior to the snapshot.

```
create image OVERLAY on top of ACTIVE

for each BITMAP in ACTIVE:
    let GRANULARITY = granularity of BITMAP in ACTIVE

    if BITMAP isn't RECORDING or is INCONSISTENT:
        continue

    create RECORDING bitmap named BITMAP in OVERLAY with GRANULARITY
```

Committing external snapshots manually

`block commit` refers to an operation where data from a subchain of the backing chain is merged down into the backing image of the subchain removing all images in the subchain.

`COMMIT_TOP` refers to the top of the subchain to merge into `COMMIT_BASE` (which stays in the new chain).

It's strongly advised to use `virDomainBlockCommit` API in libvirt directly if possible. Inactive VMs can be started with `VIR_DOMAIN_START_PAUSED` flag (`virsh start --paused`) to prevent OS from running.

Otherwise the following pseudo-algorithm can be used:

Note: A `valid` bitmap chain is a set of images containing bitmaps which conform to the rules about valid bitmaps mentioned above.

```
commit data from COMMIT_TOP to COMMIT_BASE

let BITMAPS = valid bitmap chains in COMMIT_TOP

for each BITMAP in BITMAPS
    let GRANULARITY = granularity of BITMAP in ACTIVE

    if BITMAP is not present in COMMIT_BASE:
        create RECORDING bitmap named BITMAP in COMMIT_BASE with GRANULARITY

    for each IMAGE between COMMIT_TOP(inclusive) and COMMIT_BASE(exclusive):
        if BITMAP is not present in IMAGE:
            break

        merge BITMAP in IMAGE into BITMAP in COMMIT_BASE
```