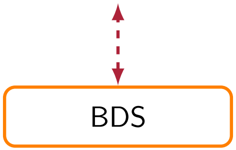
More Block Device Configuration

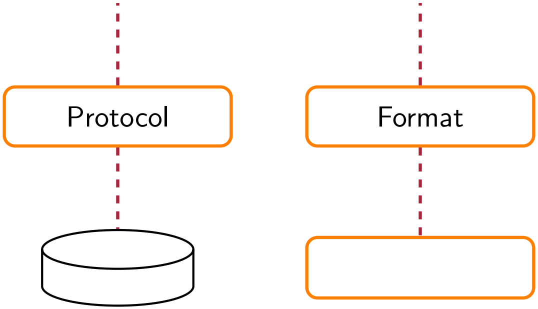
Max Reitz <mreitz@redhat.com>
Kevin Wolf <kwolf@redhat.com>
KVM Forum 2014

Part I
What we have

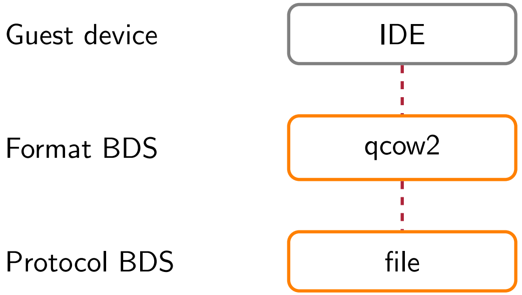
BlockDriverState



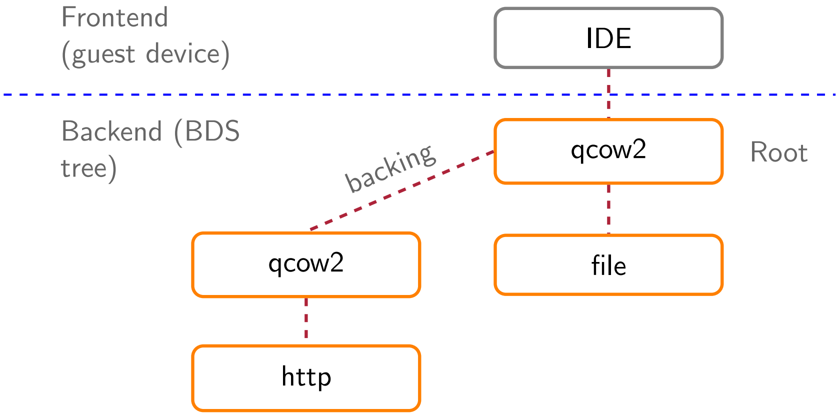
BDS types



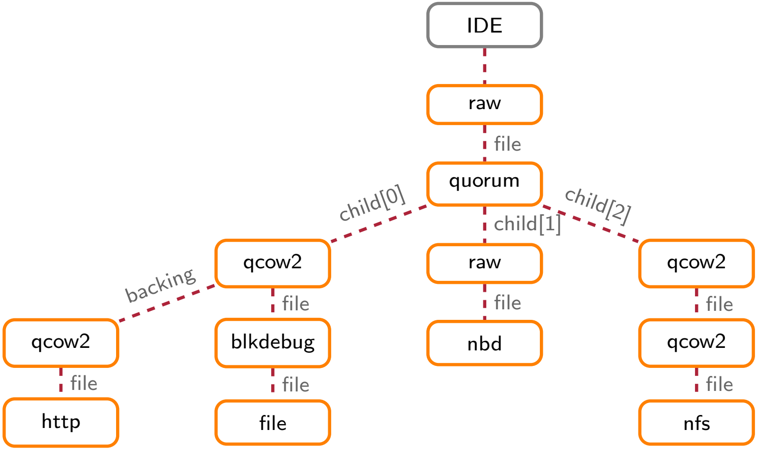
Simple BDS tree



Terminology



What is possible today



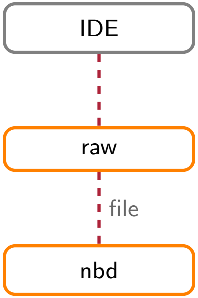
How to configure this?

Legacy syntax:

```
-hda nbd:localhost:10809
```

Separate backend and frontend;
Option syntax for backend:

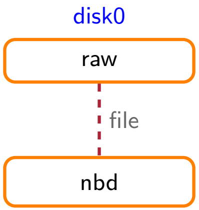
```
-drive if=none,\
id=disk0,driver=raw,\
file.driver=nbd,\
file.host=localhost

-device ide-hd,drive=disk0
```



blockdev-add

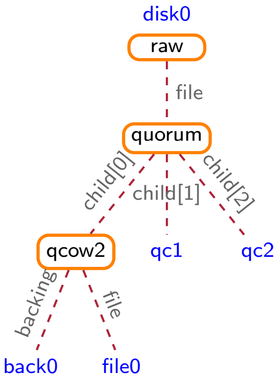
```
-drive if=none,id=disk0,driver=raw,\
file.driver=nbd,file.host=localhost
```

```
{ "execute": "blockdev-add",
  "arguments": {
    "options": {
      "id":     "disk0",
      "driver": "raw",
      "file": {
        "driver": "nbd",
        "host":   "localhost"
      } } } }
```



What is possible today (again)

```
"options": {
  "id":     "disk0",
  "driver": "raw",
  "file": {
    "driver": "quorum",
    "vote-threshold": 2,
    "children": [ {
      "driver":  "qcow2",
      "backing": "back0",
      "file":    "file0",
    }, "qc1", "qc2" ] } }
```



`node-name` **vs.** `id`

- BDS directly connected to guest devices (legacy!)
- BDS `id` used to connect both
- `blockdev-add`: `id` $\Longleftrightarrow$ top-level BDS

  $\rightarrow$ `id` $\approx$ guest-accessible BDS

- Naming and accessing non-top-level BDS?
  $\rightarrow$ `node-name`
- Common namespace for `id` and `node-name`

Options in filenames, revisited

```
json:{"driver":     "qcow2",
      "cache-size": 0,
      "file": {
        "driver":   "file",
        "filename": "file.qcow2"
      } }
```

Why?

- Some options *cannot* be given in filenames (e.g. qcow2 metadata cache size)
- Sometimes you can only give filenames (e.g. backing file field in COW files)

Part II

Some unsolved problems

Section 1

Make everything use blockdev-add

Opening block devices (blockdev-add)

If you use `blockdev-add`...

- All required options are specified
- Optional options get defaults
  - We’ll get to the problems there later
- Building the graph is straightforward

Opening block devices (-drive)

If you use `-drive`, we'd like to translate that into a clean `blockdev-add`, but...

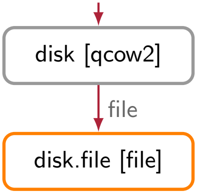
- Not even the block driver (image format) is required
  - More involved magic to fill in defaults
- New nodes may be automatically created
- No specification of this magic exists
- We need to stay compatible with old versions

Inherited options (I)

Determining cache mode for a single node:

1. Explicitly specified
   - e.g. `-drive cache.direct=on`
2. Default is `cache=writeback`

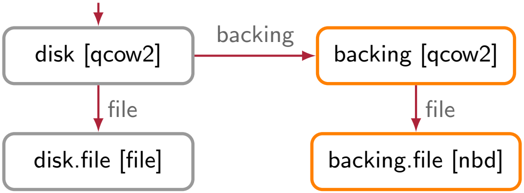
Inherited options (II)



Determining cache mode for the protocol level:

1. Explicitly specified
2. Inherit from parent node (`disk`)
3. Default is `cache=writeback`

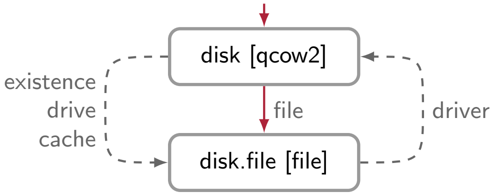
Inherited options (III)



Determining cache mode for the backing file:

1. Explicitly specified
2. Backing file path: `json:{cache:...}`
3. Inherit from parent node (`disk`)
4. Default is `cache=writeback`

It goes both ways



- Format probing: Must open protocol layer first
- Options for protocol depend on format layer
- Protocol only added by default if format driver requires a protocol

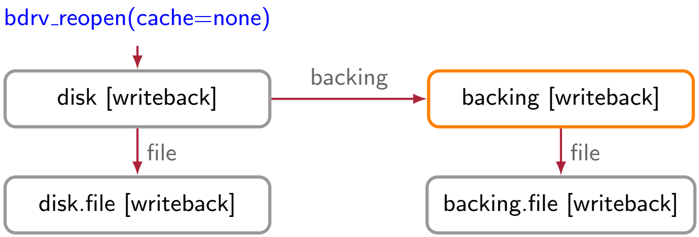
It goes both ways

- Format probing: Must open protocol layer first
- Options for protocol depend on format layer
- Protocol only added by default if format driver requires a protocol

So we can't easily translate everything into plain `blockdev-add` options in a wrapper. ☹

Section 2
Reopening

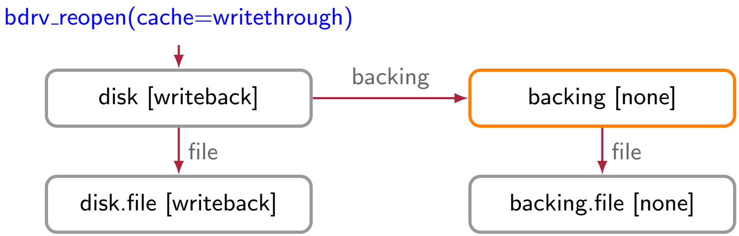
Reopen: The traditional case



Cache mode of backing file wasn't explicitly configurable:

- All nodes inherited from the root
- Reopen affects all nodes

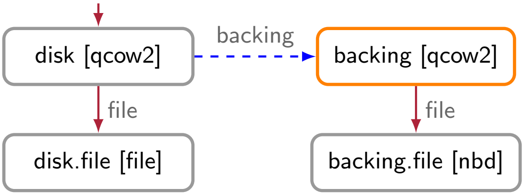
Reopen: blockdev-add world (I)



What happens if the backing file has...

- ...inherited the cache mode
- ...an explicitly set cache mode
- ...got its cache mode from a json: filename

No inheritance with references

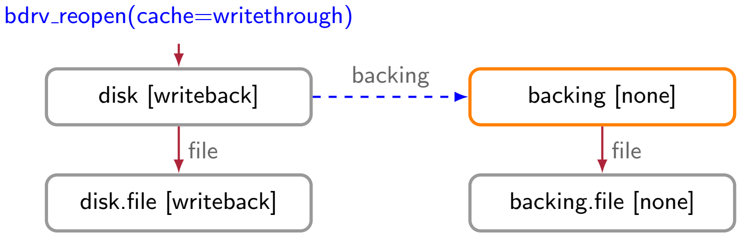


Cache mode for a separately created backing file:

1. Explicitly specified
2. Default is `cache=writeback`

Don't change options when a node is referenced

Reopen: blockdev-add world (II)



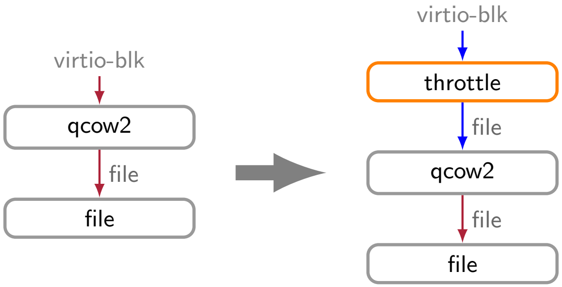
What happens if the backing file was created separately and has...

- ...an explicitly set cache mode
- ...the default cache mode

Section 3

Dynamic Reconfiguration

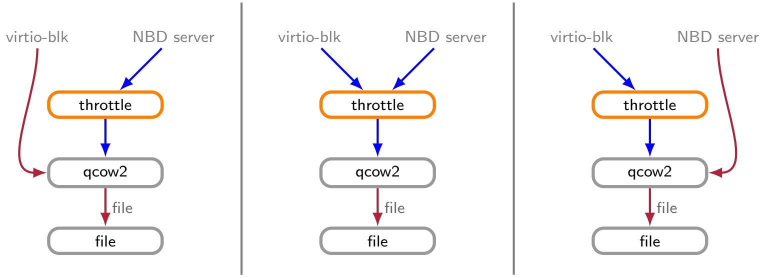
Adding and removing nodes



Let's add a new node at runtime:

- Take a live snapshot
- Add an I/O throttling filter
- ...

...but where?



Need to specify where to insert the node

- Set of arrows to replace
- Works only for nodes with one child
- A general solution looks complex – necessary?

Addressing nodes and edges

Nodes can be addressed by their `node-name`

Edges can be addresed by node + role

- role: file, backing, child[3], ...

Complication: Automatically created nodes

- e.g. for block jobs or throttling QMP commands
- Makes non-explict changes to the graph

Block jobs and reconfiguration

Long-running background jobs may be affected by changes to the graph

- Disable conflicting reconfiguration commands
- Don't restrict functionality too much
- Jeff Cody will talk about this

Questions?