QEMU Backup

Maxim Nestratov, Virtuozzo

Vladimir Sementsov-Ogievskiy, Virtuozzo

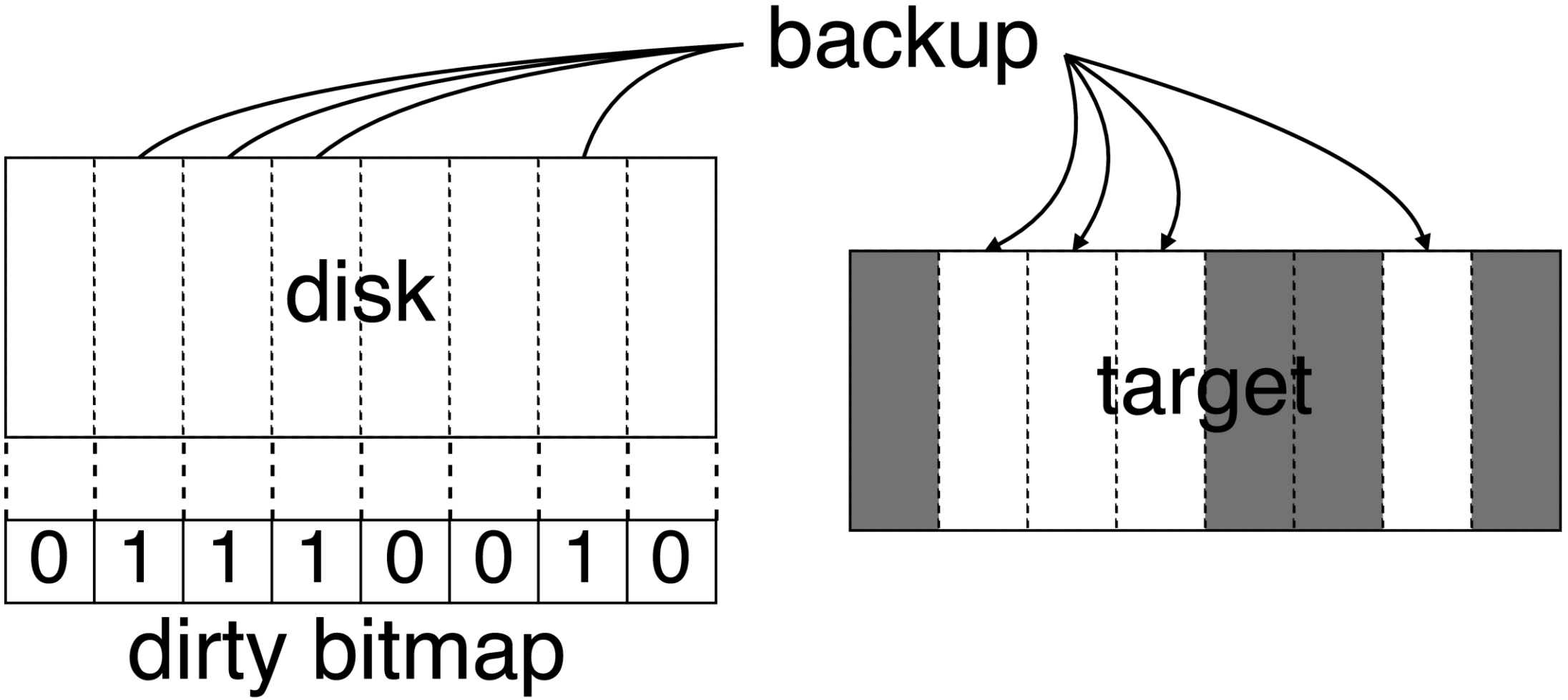
QEMU Backup

Vladimir Sementsov-Ogievskiy, Virtuozzo

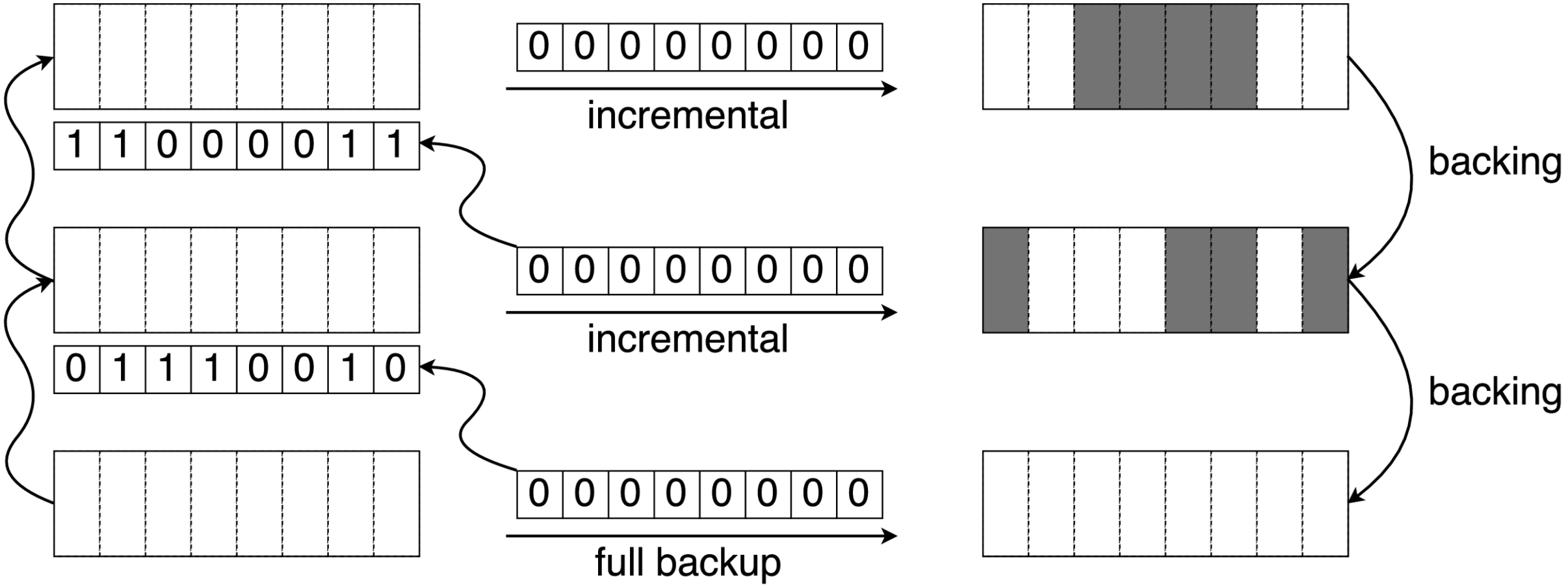
Full featured backup

- Online backup
  - Fast
  - Not very invasive for the guest
- Incremental
  - Dirty bitmaps persistence and migration
- External backup API

Incremental



Incremental



Incremental: persistent

- Qcow2 bitmaps merged into 2.9
  - Several named bitmaps
  - Size equals image size
  - Sparse format
- Other formats are under discussion

Incremental: migration

Variants:

- Fist approach: meta bitmaps
- Current approach: postcopy
- Through storage (works for qcow2)

Incremental: snapshots

Internal snapshots

- Dirty bitmaps correspond to active state
- Switch to snapshot = the whole disk is dirty

External snapshots

- Everything is possible

Performance: current work

Backup = simple copy + COW (write notifiers)

- Current approach:
  - Sequential copying + sequential notifiers
- New arc:
  - Queues of requests
  - Several copying coroutines
  - Notifiers just increase priority of request
  - Earlier notifier release

Performance: ideas and plans

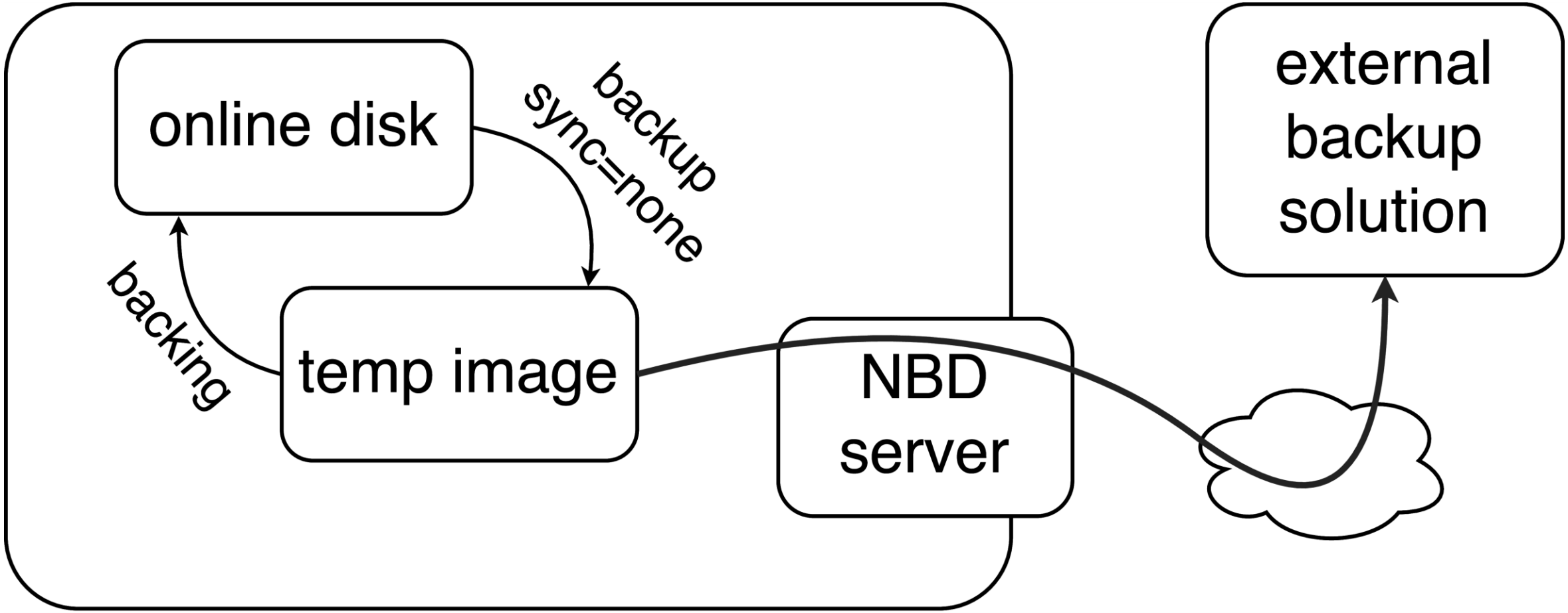
How to handle COW?

- Current: guest wait for backup
- Reverse delta: read COW area to local delta

External backup API

- Image fleecing scheme
- Incremental backup
  - NBD block-status extension
  - Additional API for dirty bitmap management

External backup API: image fleecing



External backup API: NBD block-status

- Current NBD: payload only for READ
- Extension: structured replies
- Extension: block-status
  - Negotiation phase: select metadata contexts
  - Transmission phase
    - New command NBD_CMD_BLOCK_STATUS
    - Reply chunk contains extent descriptors

QEMU Backup summary

Merged:

- Qcow2 bitmaps

Done in Virtuozzo:

- New backup architecture (async IO)
- Bitmaps migration
- NBD block-status extension

Near future:

- External backup API

Libvirt Backup API

Maxim Nestratov, Virtuozzo

Libvirt Backup API first proposal

- "Push" backups (managed)
- "Pull" backups (external)

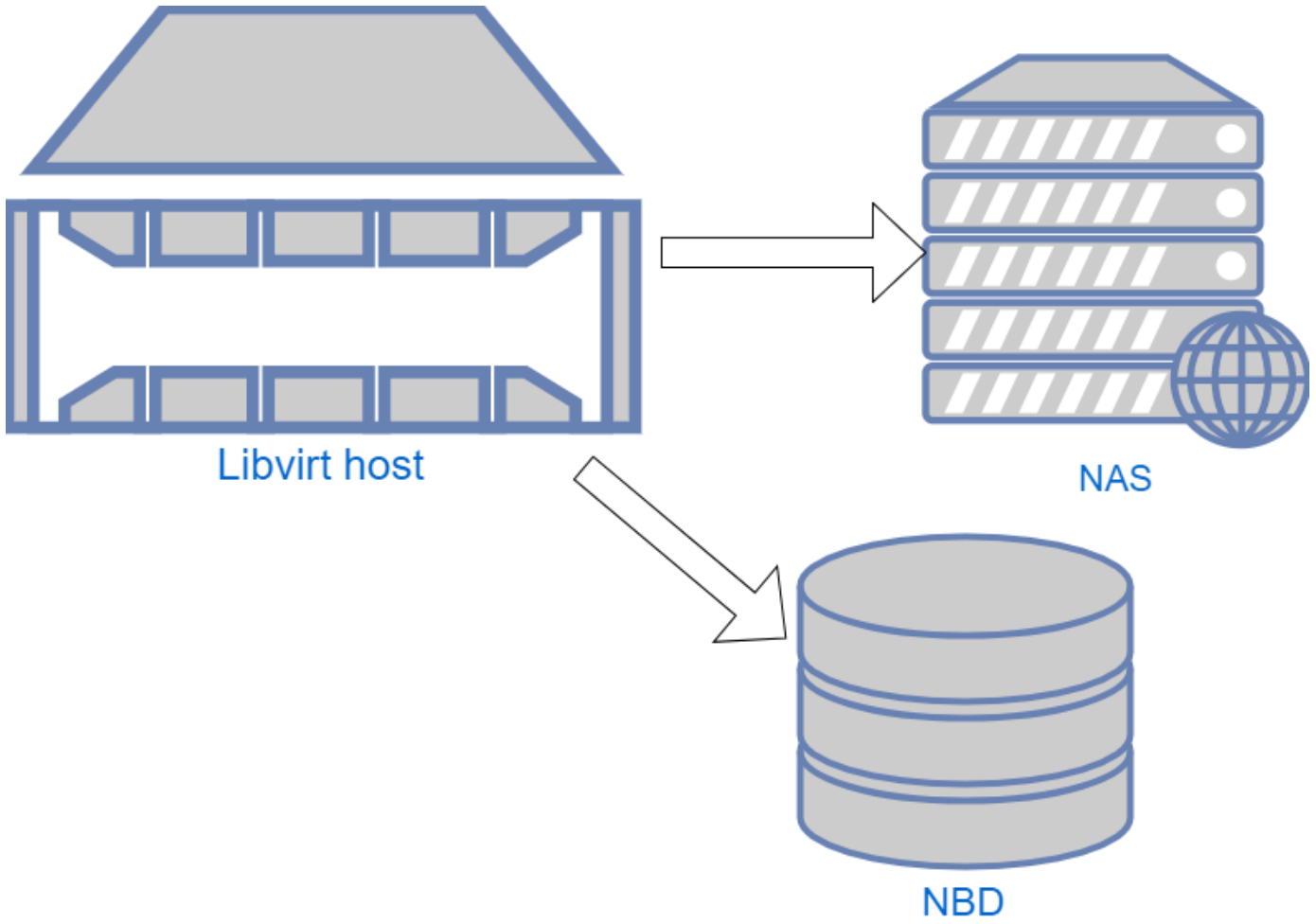
Why not snapshots?

- Different storage
- Incremental backups
- Multiple chains
- Agentless

"Push" or "Managed" Backups

- A new set of functions similar to snapshots
- Create, List, Delete, Edit
- Managed by libvirt
- Local to host
- Can be saved to any supported block device
- NAT friendly
- Based on “drive-backup” QMP command

Managed backup scheme



Managed backup concerns

- Hard to use in clusters
- Guest performance influence due to network throughput
- A lot of code to implement
- Access to backup storage

"Pull" or "External" Backups

- Mostly two functions: Start/Stop
- Exposes block device via NBD protocol
- Uses NBD protocol extension for incremental backups
- Based on blockdev-add/blockdev-del and blockdev-backup QMP commands

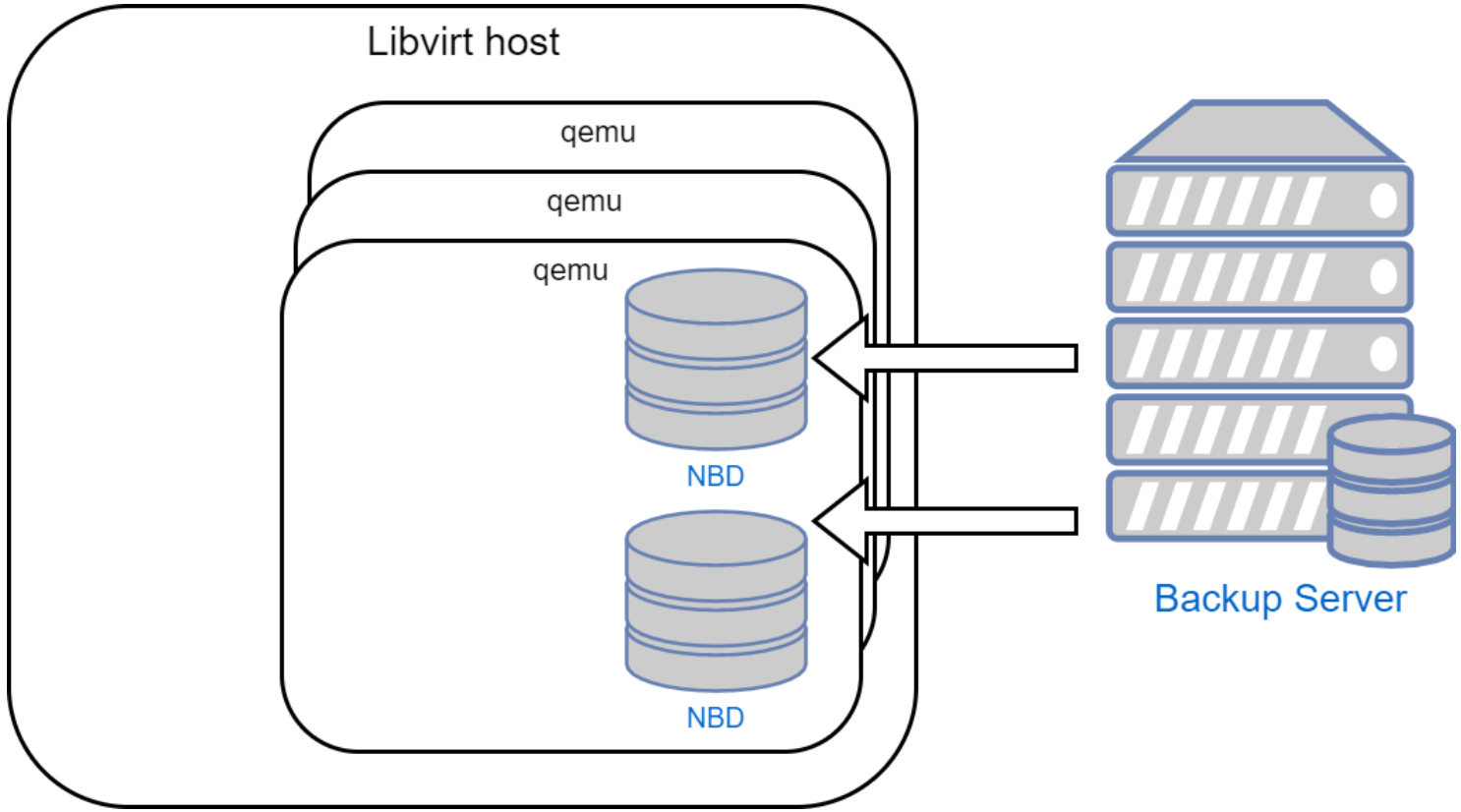
External backups advantages

- Tolerant to guest performance
- Controlled externally
- Cluster friendly

External backups concerns

- NAT unfriendly
- NBD dependent

External backup scheme



Current intention

- Let’s go with “external” scheme first
- It’s easier to implement
- Has more advantages in comparison with “push” backup scheme

Libvirt API proposal

```
virDomainBackupPtr virDomainBackupStart(virDomainPtr dom,
                                        const char *xmlDesc,
                                        unsigned int flags);
int virDomainBackupStop(virDomainPtr dom,
                        virDomainBackupPtr backup,
                        unsigned int flags);

int virDomainBackupList(virDomainPtr domain,
                        virDomainBackupPtr **backups,
                        unsigned int flags);

char* virDomainBackupGetXMLDesc(virDomainBackupPtr backup,
                                unsigned int flags);
```

virDomainBackupStart

xml parameter looks like

```
<domainbackup>
    <blockserver port="7000">
        <listen type='address' address='1.2.3.4'/>
    </blockserver>
    <disk name='sda'>
        <fleece file="/fleece_a"/>
    </disk>
</domainbackup>
```

virDomainBackupStart continued

Successful start will add the following section

```
<blockserver port="7000">
    <listen type='address' address='1.2.3.4'/>
</blockserver>
```

flags - VIR_DOMAIN_BACKUP_START_QUIESCE

virDomainBackupStart continued

- Call “blockdev-add” for each disk in domain
- Gather all created devices into a transaction
- Start the transaction via “blockdev-backup”

virDomainBackupStop

- Successful stop will remove reference from NBD server
- Will stop server when reference count reaches zero

virDomainBackupStop continued

- Stop NBD server
- Cancel backup block job for each disk with “block-job-cancel” QMP command
- Delete backup devices created with “blockdev-add” via “blockdev-del”
- Delete all fleece files created

Questions

- Do we need to have a separate call to list backups? Or just report all information in domain xml.
- Do we need to start blockservers manually?
- Incremental backup interface.
- Restore: should we introduce a new set of functions or just reuse existing facilities.
- Should backup API be generic or QEMU specific.

Thank you. Questions?

Maxim mnestratov@virtuozzo.com

Vladimir vsementsov@virtuozzo.com