



dm-qcow2: device-mapper-based QCOW2 storage for containers



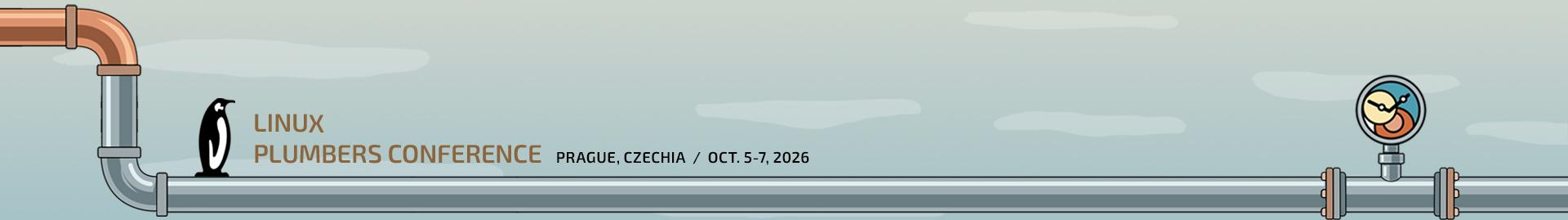
LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Andrei Zhadchenko (Virtuozzo)

What do we want from storage?

- thin provisioning
- shared data
- snapshots
- backups
- migration
- performance



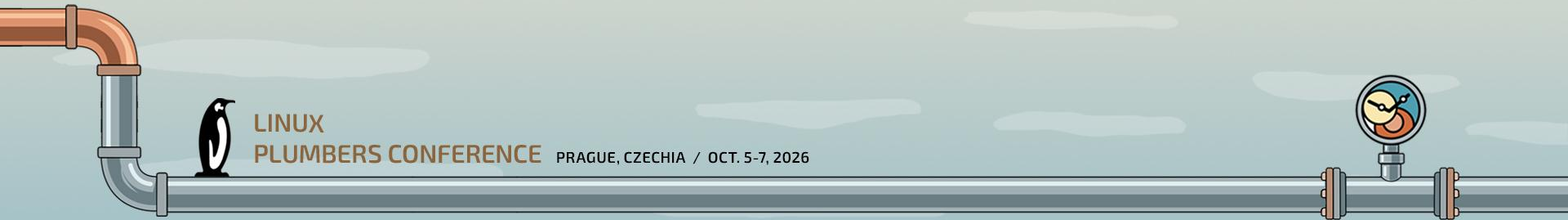
Existing options

shared host fs as files (dirs, overlays): no per-container quotas, no live snapshots, poor backup support

btrfs/zfs: host dependency: host fs in now a storage

dm-thin: dedicated pools, metadata separated, hard to copy/ship/import

qemu-nbd: daemon per disk, extra switches, etc.



qcow2 in kernel

- qcow2 is feature-rich and is designed for VM storage
- mature ecosystem
- no daemon
 - don't need to manage daemon lifetime
 - daemon cgroup complications
 - ublk performance is exceptional but still has losses (QD1 2 vs 6 vs 8 μ s)
- reclaim can go through driver, OOM exists



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Why device-mapper?

- device managing
- queue (if based on dm-rq)
- I/O freeze
- reloading
- stackable

rhel7 ploop: ~20k LoC

vs

dm-qcow2 ~7k LoC

backup + migration helpers ~1k



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



request vs bio-based

fully fledged request queue:

- merging
- reordering
- scheduler
- io.cost io.latency wbt

a bit more cpu intensive
dm-rq made for 'clone rq and
resubmit' to another device

more new dm-core features



LINUX
PLUMBERS CONFERENCE

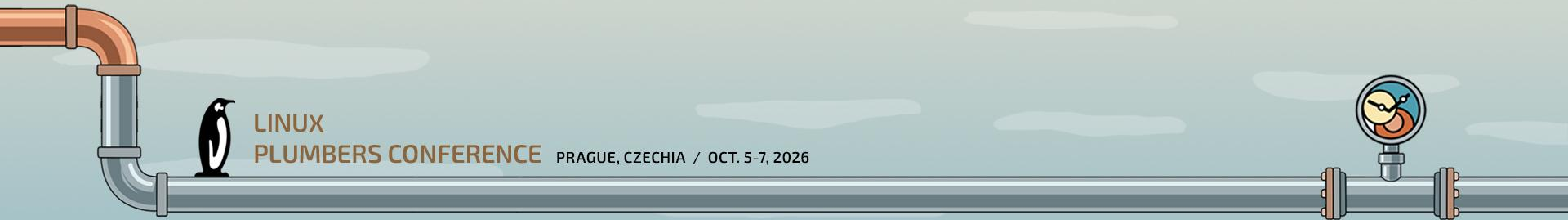
PRAGUE, CZECHIA / OCT. 5-7, 2026



Management operations

Easy to do with device-mapper:

1. suspend
 2. management operation
 3. reload/resume
- snapshotting
 - resizing
 - adding backup/migration layers



Snapshots

- qcow2 supports internal + external snapshots
- dm-qcow2: focus on external
- different devices can share base image(s)

Snapshot plan:

1. suspend device (disk.qcow2)
2. create external snapshot with qemu-img
3. reload the device with bigger chain (top.qcow2 -> disk.qcow2)



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Backups

snapshot-based

- + very simple
- post-backup management pain
- interleaves with snapshots
- incremental?

proper write tracking + bitmaps

- + incremental backups
- + recoverability
- extra code
- potentially complicated extra utilities



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



write-tracking

Passive: bitmaps
stored within qcow2 file

historically for OpenVz:
submit_bio_noacct() hook + control via
IOCTLs. Load/unload on device
creation/destruction

dm-era:
persistent, needs device

Active: dm-push-backup

small dm target that holds writes
until acknowledged

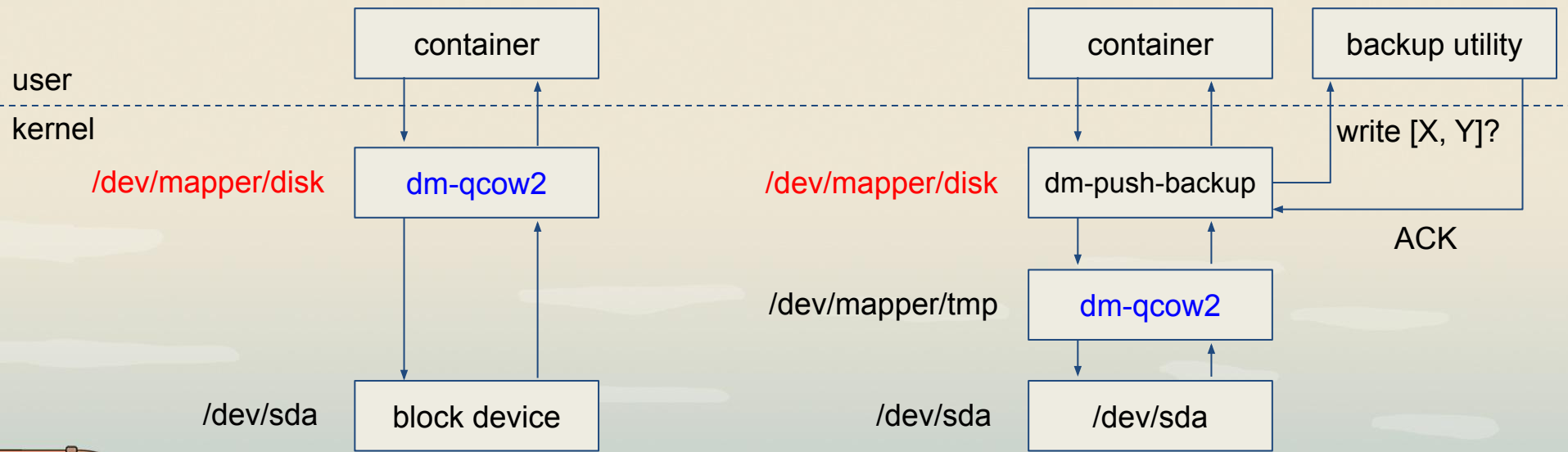


LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Stackability/reload (dm-push-backup)



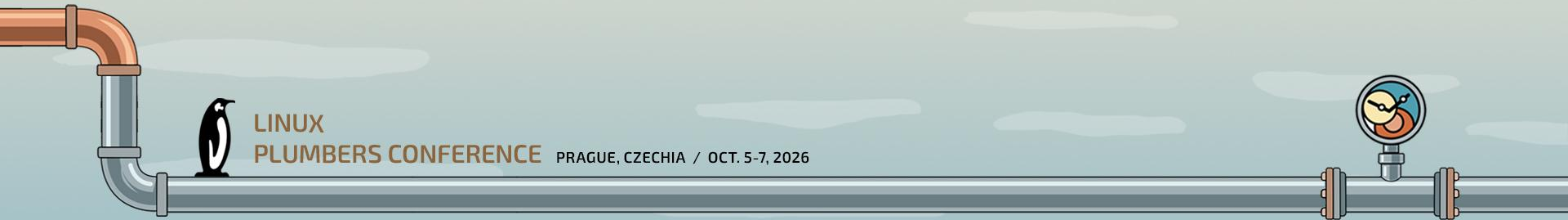
holes?

No holes in disks currently.

Drill a hole to seek holes

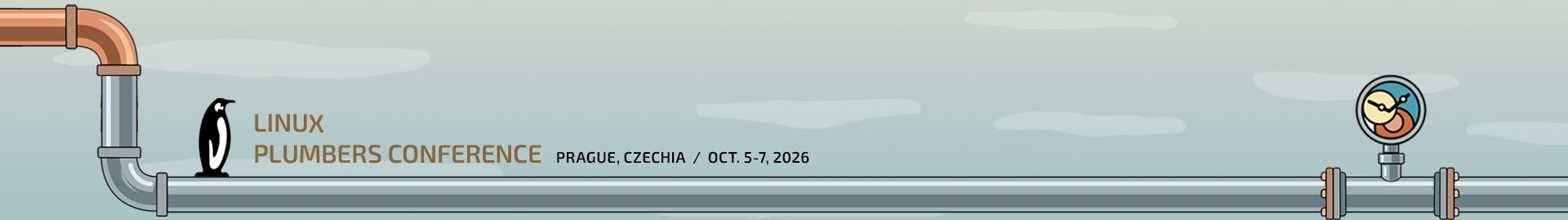
- add to prototype in `block_device_operations`
- call from `blkdev_llseek()`
- add prototypes to device-mapper

Maybe also useful for dm-thin/loop/ublk?



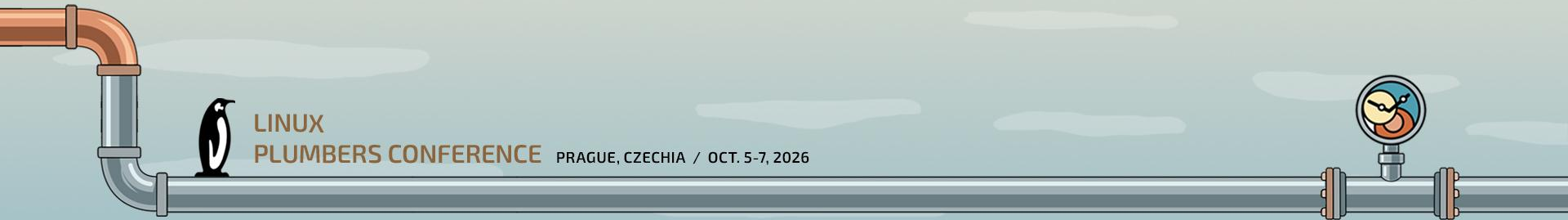
Other problems

- Unprivileged containers are not currently possible: device-mapper requires CAP_SYS_ADMIN
- ext4/xfs are not reentrant: journal handles are taken from current->journal_info



Bonus: VM/Container storage unification

- can be used as QEMU disk
- single pipeline for backups
- single pipeline for snapshots
- allows in-kernel IO request handling (vhost-blk)



Bonus: statistics are nice to have

dm-stat exists

- + per-device
- + fast
- only combined histogram

eBPF

- + flexibility
- stacked devices



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Performance (no batching)

brd ramdisk + qcow2 file on ext4

```
fio --filename /dev/mapper/dm-qcow-test --direct=1 --rw=randwrite --bs=4k  
    --ioengine=io_uring --iodepth=64 --numjobs=X
```

IOPS / median latency

	raw file on ext4	dm-qcow2 (request) device from ext4 file	dm-qcow2 (bio) device from ext4 file
1 thread	398K / 187µs	380K / 218µs	444K / 168µs
2 threads	778K / 193µs	787K / 224µs	866K / 197µs
4 threads	1485K / 214µs	1006K / 313µs	1118K / 284µs



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Performance (heavy batching)

brd ramdisk + qcow2 file on ext4

```
fio --filename /dev/mapper/dm-qcow-test --direct=1 --rw=randwrite --bs=4k  
    --ioengine=io_uring --iodepth=64 --iodepth_batch_submit=32 --numjobs=X
```

IOPS / median latency

	raw file on ext4	dm-qcow2 (request) device from ext4 file	dm-qcow2 (bio) device from ext4 file
1 thread	643K / 49 μ s	1195K / 27 μ s	584K / 77 μ s
2 threads	1224K / 52 μ s	2266K / 28 μ s	1071K / 84 μ s
4 threads	2147K / 59 μ s	2952K / 47 μ s	1707K / 99 μ s



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



How to touch?

<https://github.com/azhadchenko/dm-qcow2-out-of-tree> out of tree module (master branch is request-based and bio is bio based)

<https://github.com/azhadchenko/incus/tree/dm-qcow2> incus PoC with snapshots (deletion is offline-only due to PoC) and resize

incus + dm-qcow2 supports:

- online resize
- online snapshot creation



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Summary of uncertainties

- request-based vs bio-based
- device holes support
- bitmaps (in-dm-qcow2 vs per-block-device vs separate dm target)
- dm-stats vs BPF

highly unlikely for now:

- unprivileged dm-qcow2
- filesystem re-entrancy on the same stack

