Virtually Persistent Data

Andrew Warfield
XenSource

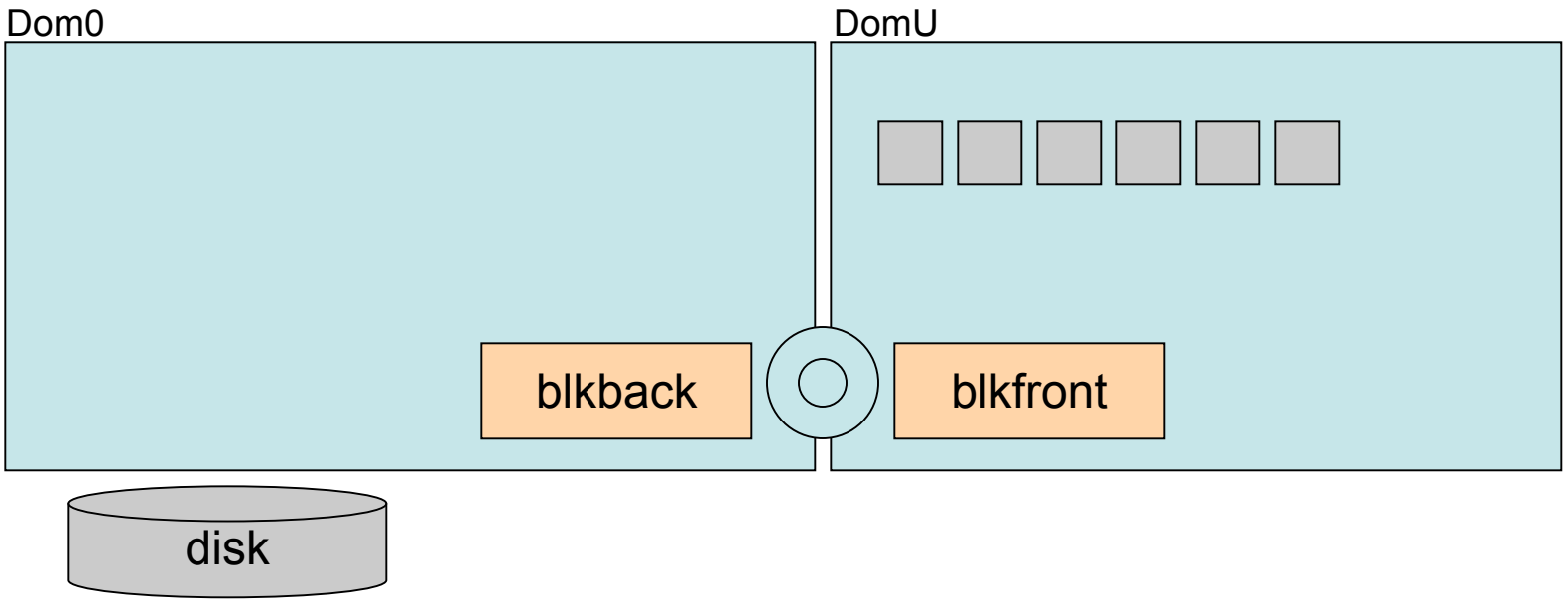
Quick Overview

- Blktap driver overview/update
  - Performance vs. Safety
  - Architecture
  - Tapdisks
- Block consistency and live migration
  - Problem, current solution, future solution

Why BlkTap?

- It turns out that performance isn't the only requirement for VM storage.
- Other requirements:
  - Correctness
  - Managability (e.g. file-based disks)
  - Crazier things (CoW/encryption/network)
- Doing these things at the Linux block layer is tricky.

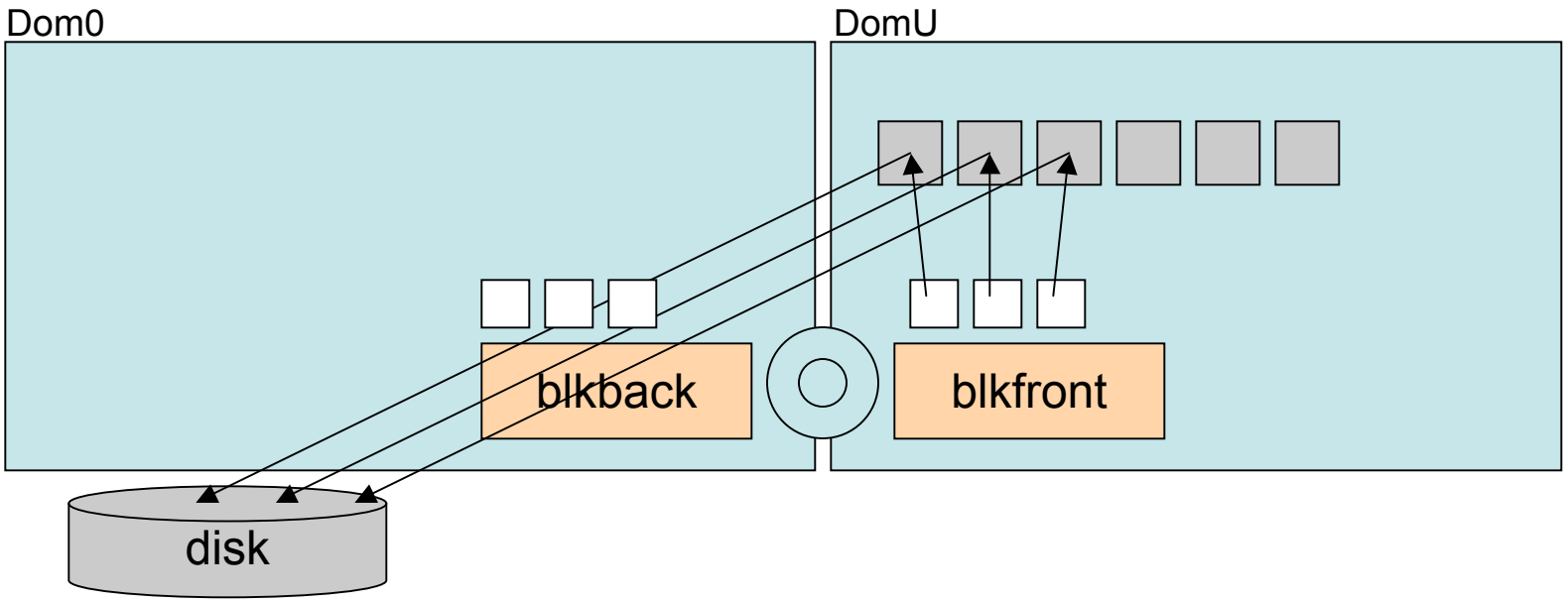
Example: blkback + loopback



Example 1: Normal block device operation.

- *Go through safety concerns, mention fix in loopback2, also mention qos/block shed issues.*

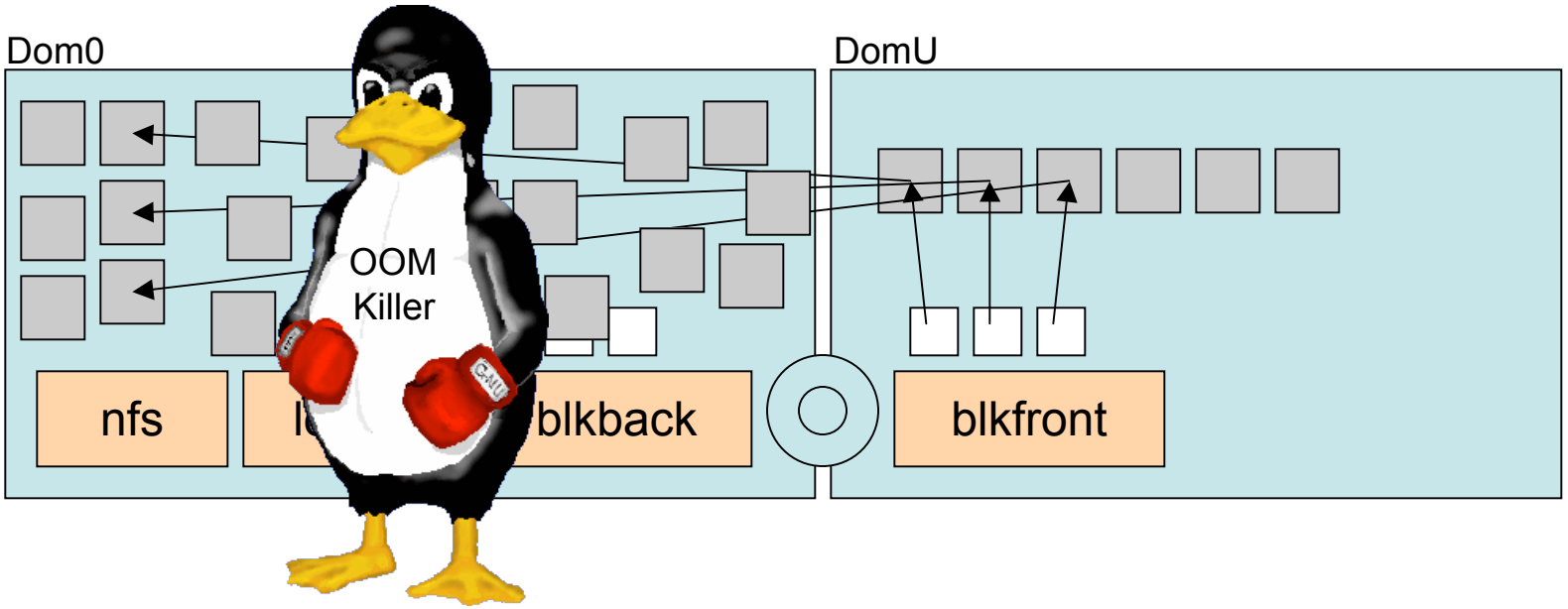
Example: blkback + loopback



Example 1: Normal block device operation.

- *Blkback preserves the semantics of the physical disk.*
- *DomU is notified of completion once data is really written.*

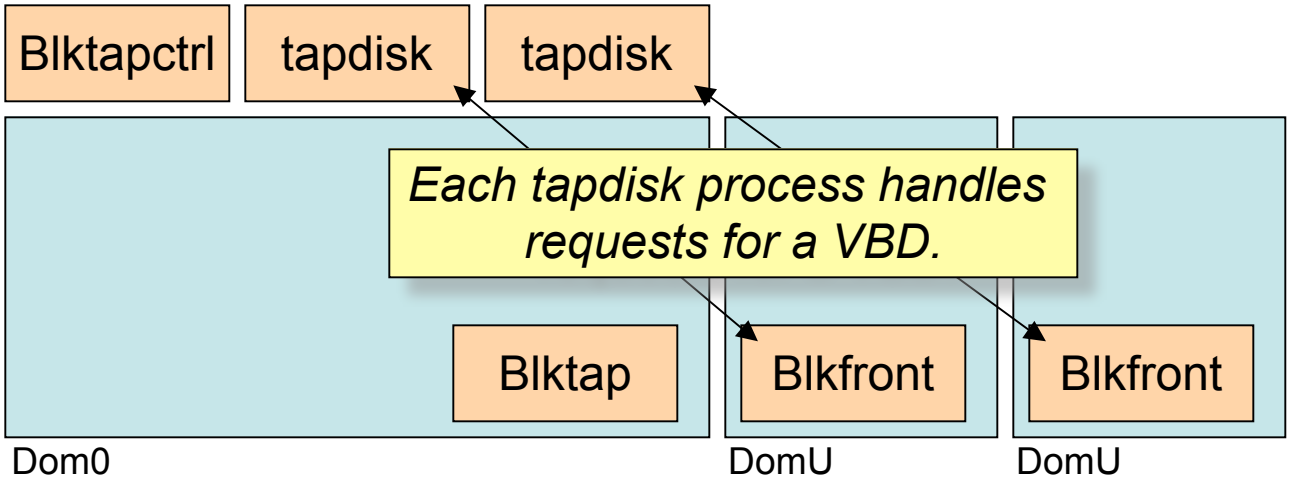
Example: blkback + loopback



Example 2: Loopback and NFS

- *New loopback driver should fix this.*
- *Associating requests with a process has qos/sheduling benefits.*

Architecture



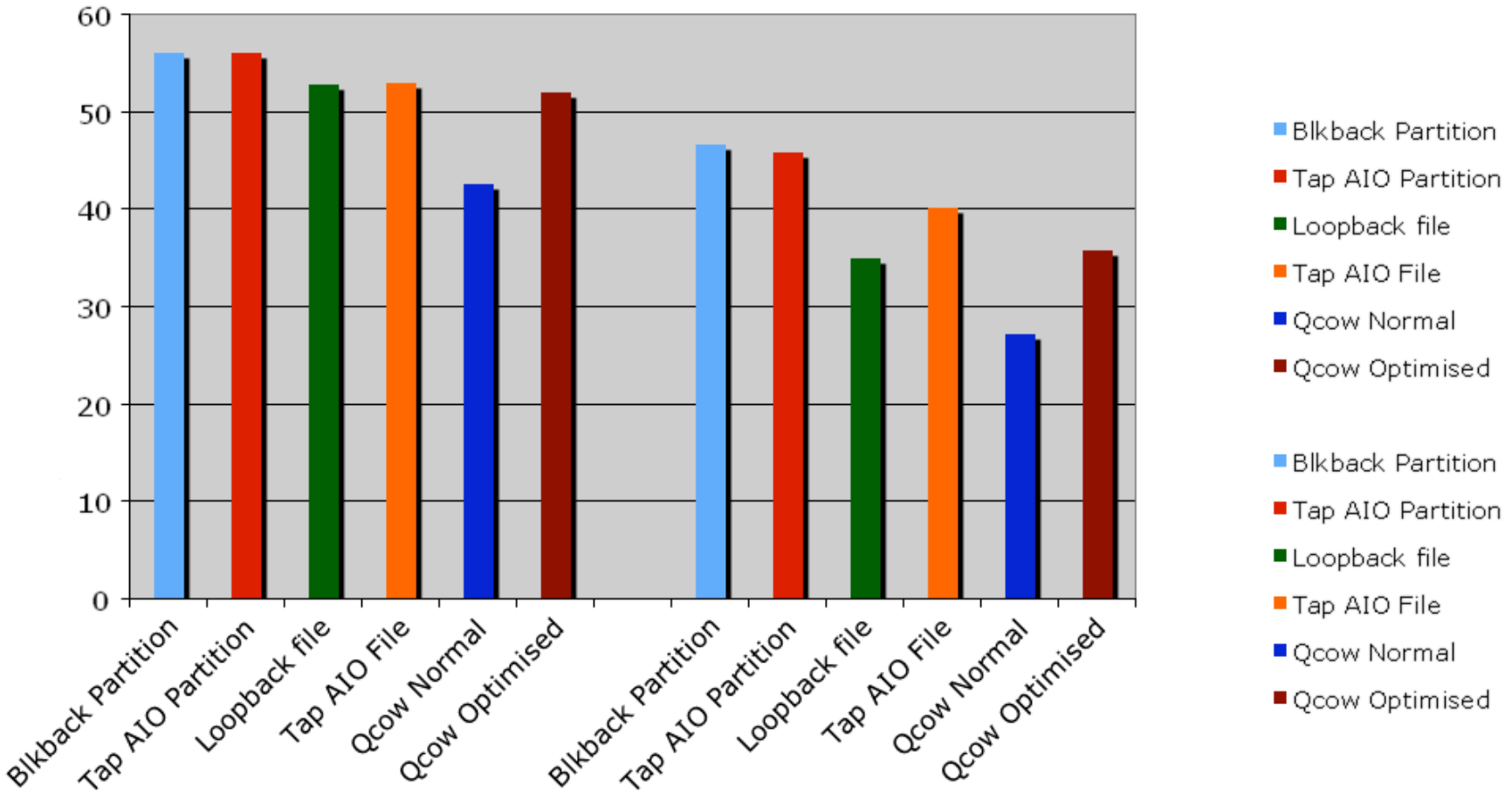
- *Blktap handles IDC*
- *Tapdisk implements a virtual block device*
  - *Synchronous, AIO, QCoW, VMDK, Shared mem*
- *Zero-copy throughout*
- *Tapdisks are isolated processes*

Current tapdisks

- Individual tapdisk drivers may be implemented as plugins.
- Generally a few hundred lines of code.
- Very similar to qemu's block plugins, but with an asynchronous interface.
- Currently have asynchronous raw (device or image file), QCoW, VMDK, and shared-memory disks.

```
struct tap_disk
  tapdisk_aio = {
          "tapdisk_aio",
          sizeof(struct
           tdaio_state),
  tdaio_open,
  tdaio_queue_read,
  tdaio_queue_write,
  tdaio_submit,
  tdaio_get_fd,
  tdaio_close,
  tdaio_do_callbacks,
};
```

Blktap Performance



CoW/Image formats

- *Many image formats now exist for block devices*
  - *QCoW, VMDK, VHD*
- *Work a lot like page tables: mapping tree for block address resolution*
  - *Typically use a bitmap at the bottom level*
- *Ensuring consistency adds overhead*
  - *Metadata is dependent on data*
  - *Must be written before acknowledging request to DomU*
- *We are currently working with a modified version of the QCoW format*
  - *4K blocks, all-at-once extent allocation, bitmaps.*

Migration Consistency

- Drivers use request shadow rings to handle migration
- Unacked requests are reissued on arrival
- Slight risk of write-after-write hazard.
- Now fixed, more optimal plan for later.

What's next?

- Copy on Write
- Live snapshots
- Maybe some network-based/distributed storage.

end

Performance(2)