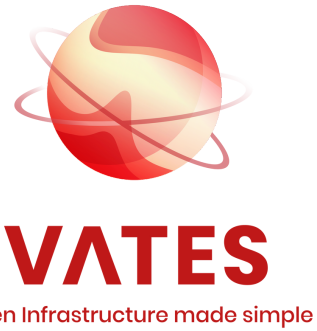
Using SPDK with the Xen hypervisor

Damien Thenot - damien.thenot@vates.fr



Myself

PhD Student at Télécom SudParis

Interested in virtualization, OS dev

Working at Vates, doing virtualization software in Grenoble

XCP-ng

- XCP-ng, a distribution of Xen with a powerful administration stack, XAPI
- Looking to continually improve the platform: new technologies, better integration

`https://xcp-ng.org`

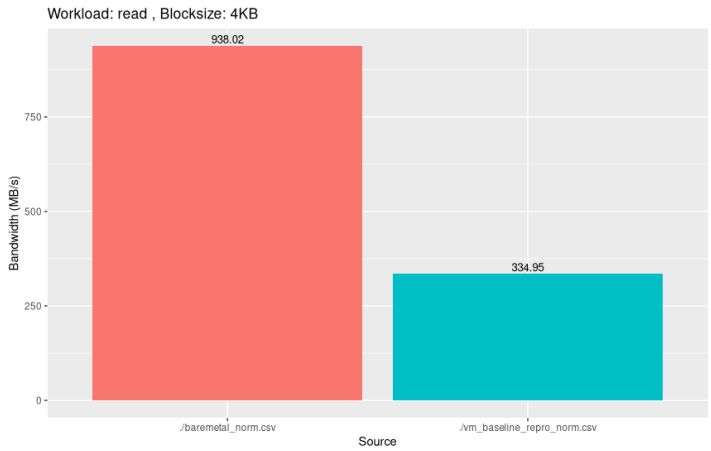
Our Goals

Problem!

Performance of the current storage stack isn't scaling with the performance of new storage

Our approach should:

- Improve virtualization storage performance
- Preserve the security given by the memory sharing mechanism in Xen
- Minimally impact users virtual machines content



Xen

A type-1 hypervisor for x86 (originally)

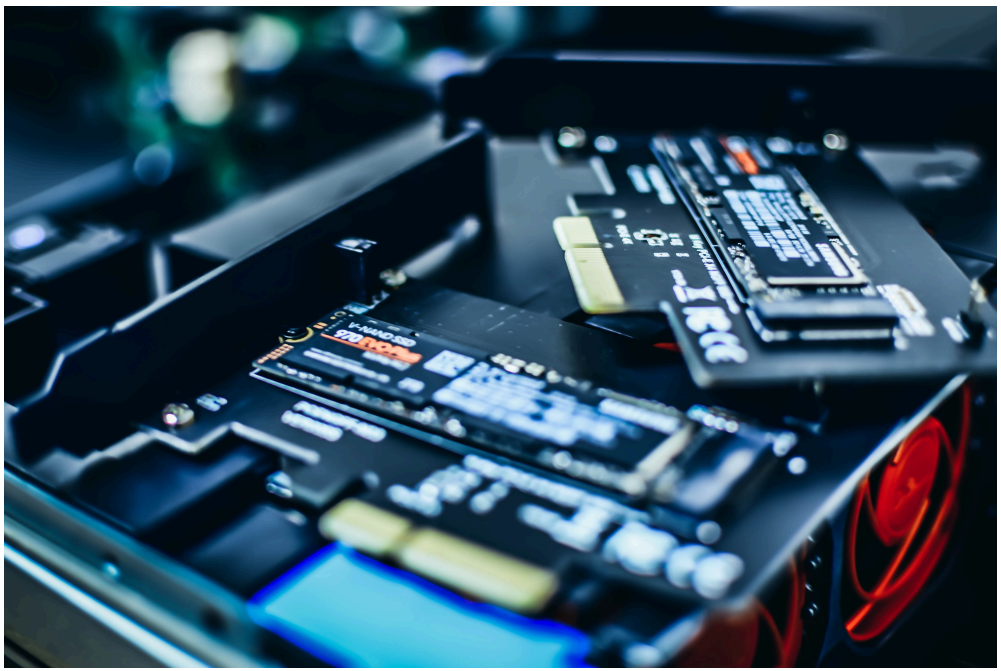
Dom0

A virtual machine, called Dom0, to control HW devices
What devices is not handled by Xen directly is given to this VM, i.e. network, storage

New technology → NVMe

"New" technology (more than 10 yo)

- ⇒ available on most hardware from consumer to business
- From decent with SATA to best of flash storage performance with NVMe



NVMe :

- Standardized protocol
- High performance parallel design

⇒ adapted to highly parallel flash storage

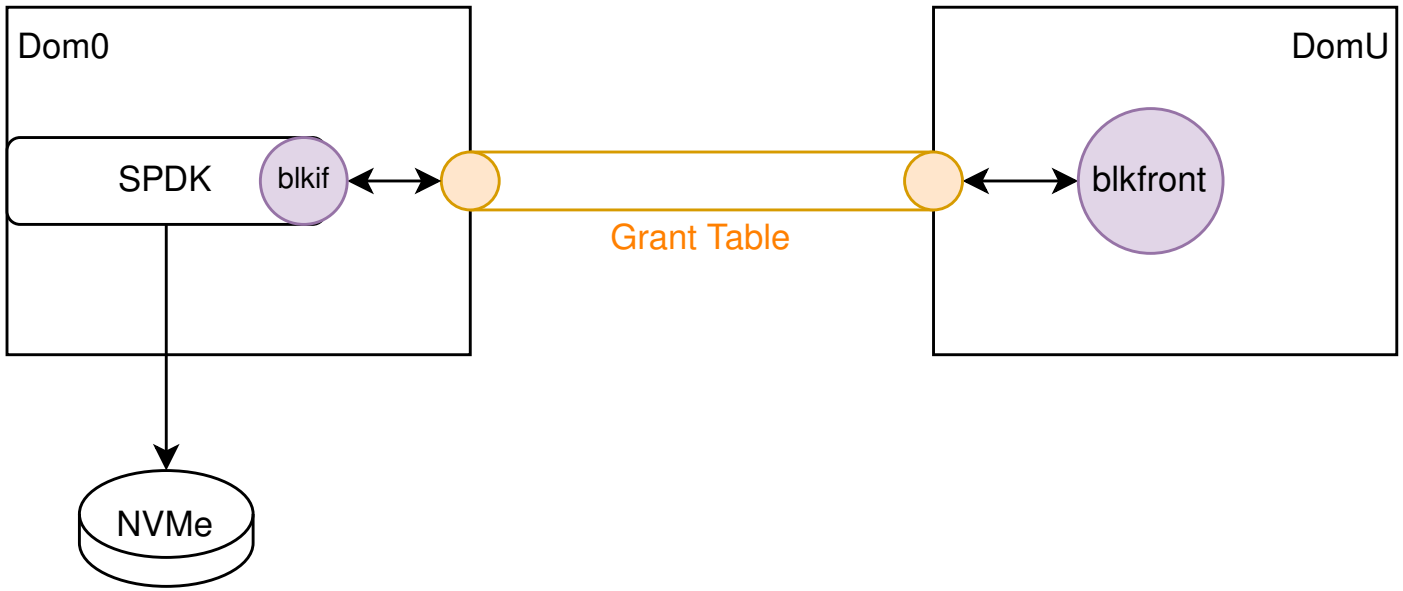
blkif

blkif is a protocol used to transmit block requests from a virtual machine to a hardware domain.

- A block driver in the guest kernel *xen-blkfront* registering as a normal block device */dev/xvda*
- A backend *tapdisk* to multiplex access to storage in the HW domain

Grant Table

A Xen interface to share specific memory with other virtual machines

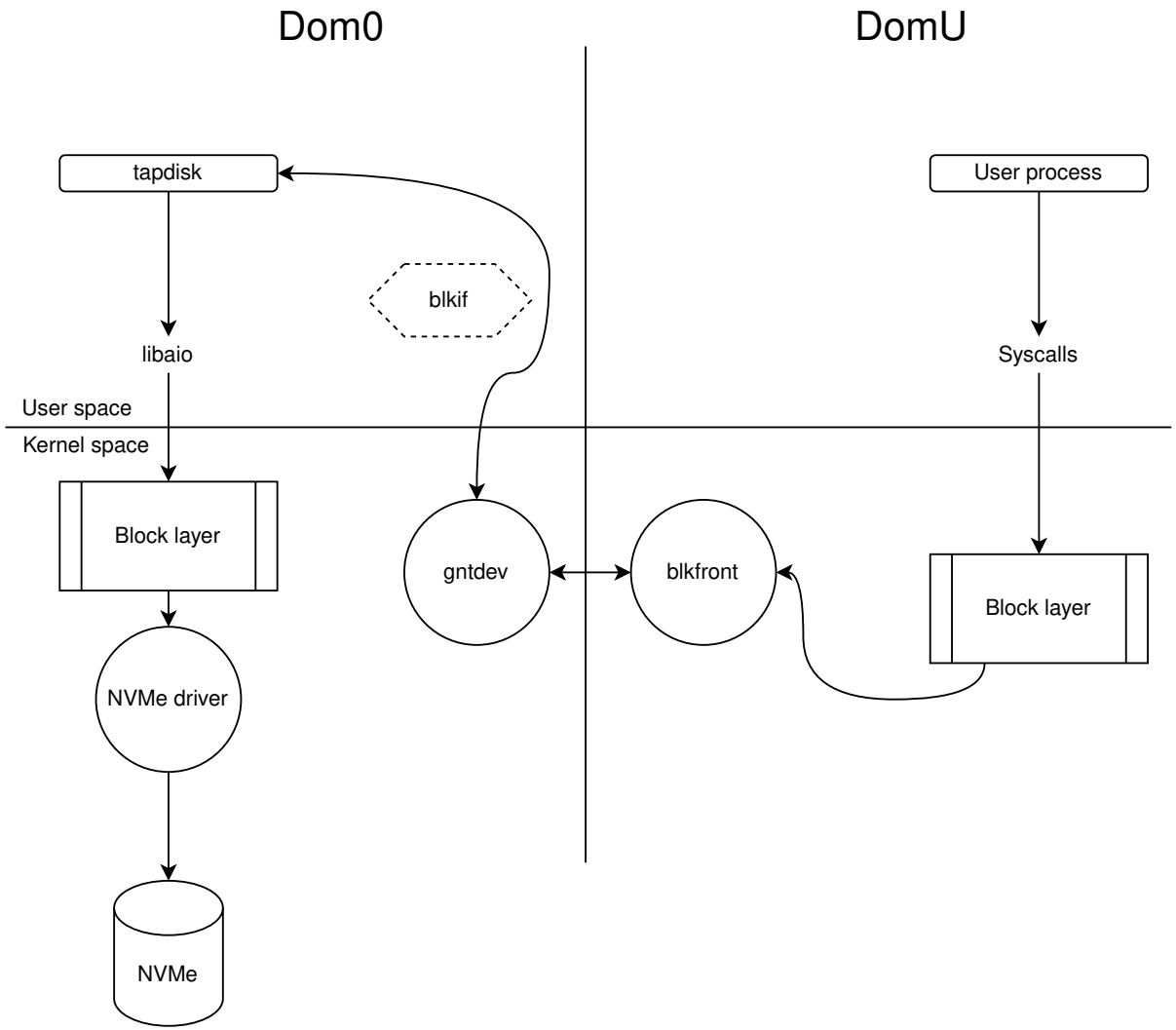


SPDK - Storage Performance Development Kit

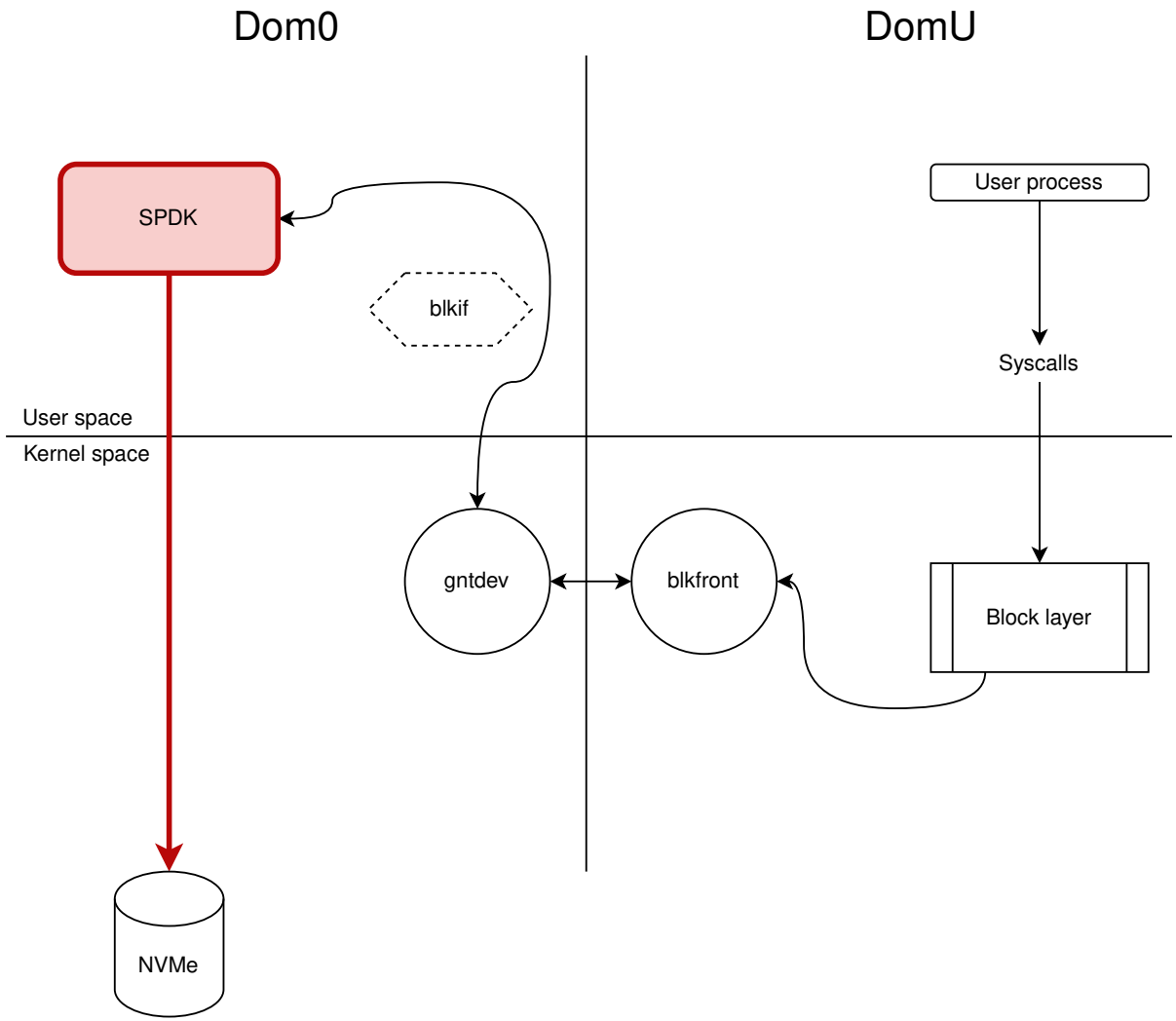
`https://spdk.io`

- Userspace driver for NVMe storage
- Created by Intel
- Similar to DPDK (Data Plane Development Kit)

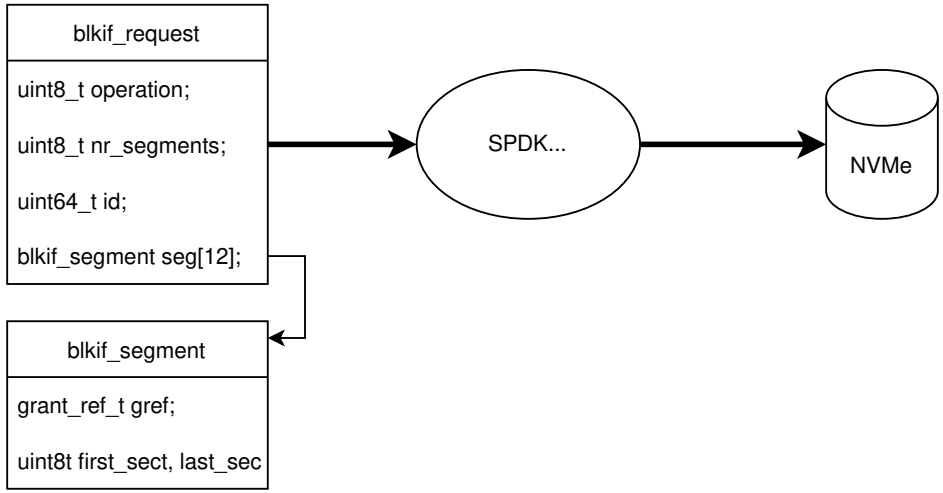
tapdisk - current backend in XCP-ng



SPDK


Dom0
DomU
SPDK
blkif
User process
Syscalls
User space
Kernel space
gntdev
blkfront
Block layer
NVMe

blkif request to SPDK interface



blkif operation $\Rightarrow$ SPDK API

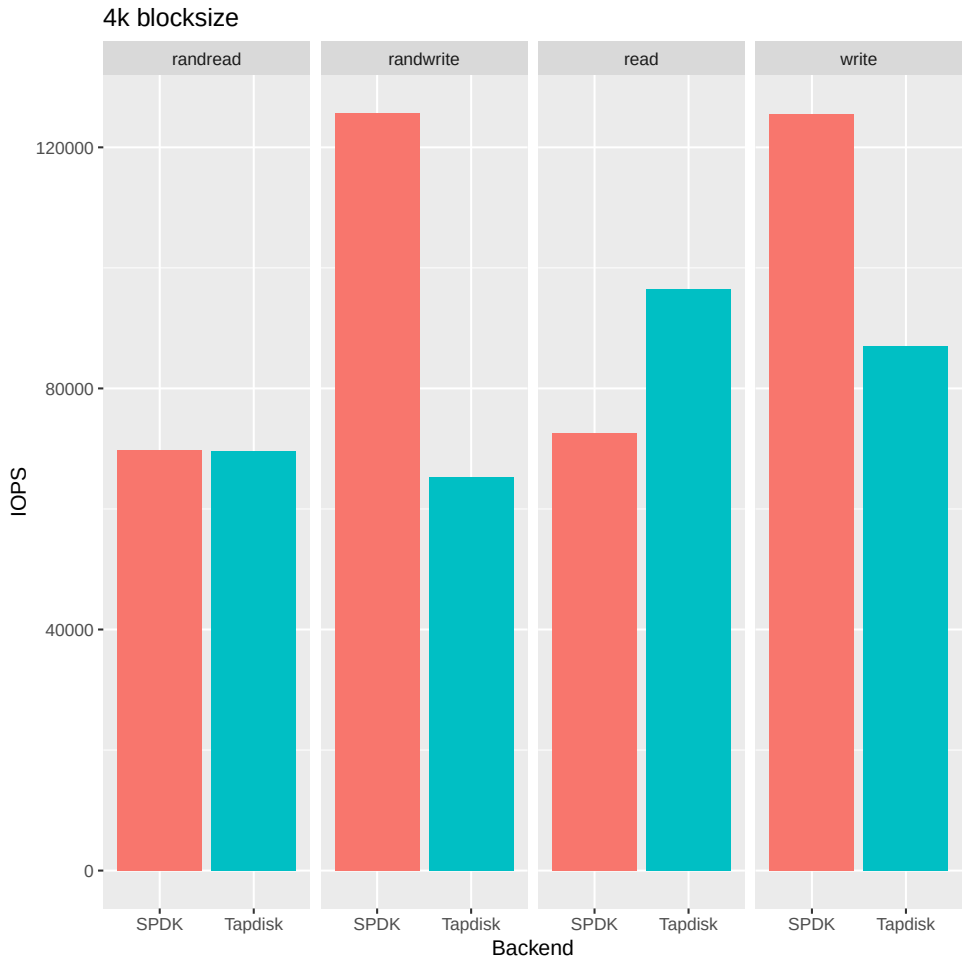
BLKIF_OP_READ $\Rightarrow$ *spdk_bdev_read*

BLKIF_OP_WRITE $\Rightarrow$ *spdk_bdev_write*

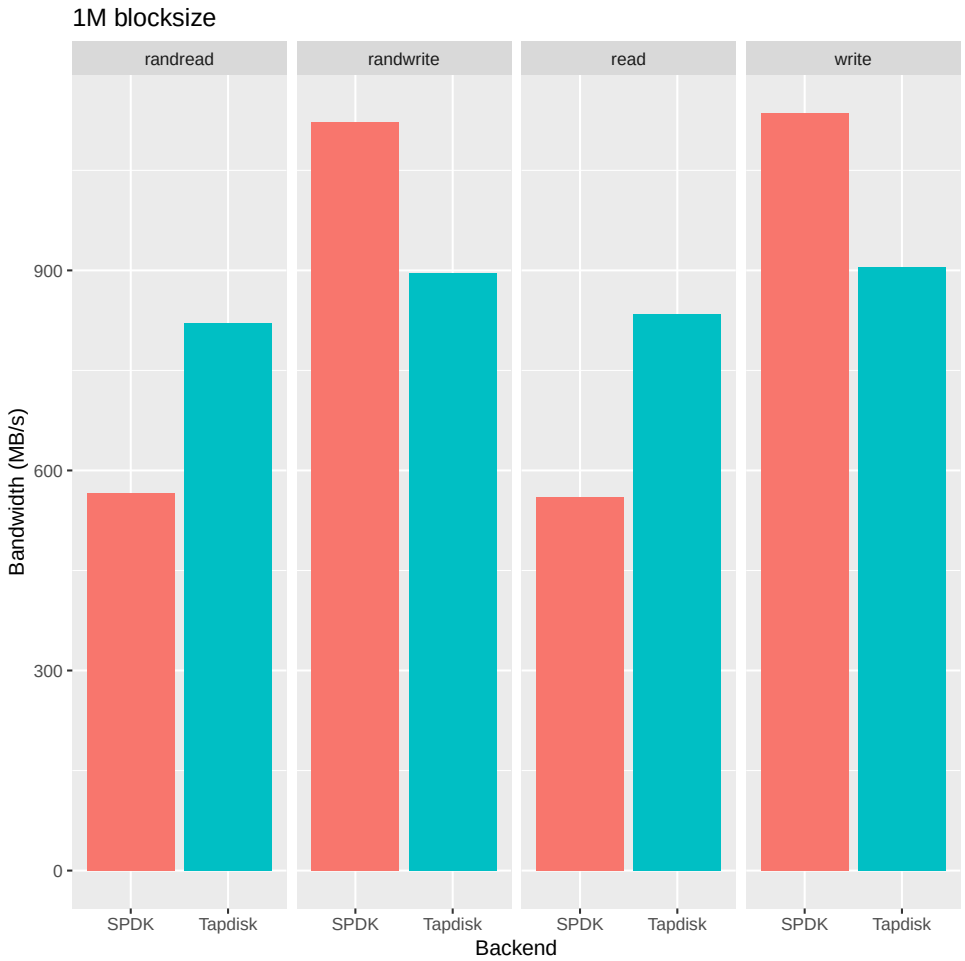
Handling of a request

- Get a request from the blkif ring
- Allocate memory with the SPDK allocator (we need DMA-able memory)
- For WRITE requests:
  - copy data from grant references
  - call spdk_bdev_write to write on the NVMe
- For READ requests:
  - call spdk_bdev_read to obtain data from the NVMe
  - on the callback of the read, copy data to the grant references
- On callback of both, we need to confirm request status on the blkif ring
- Disallocate memory

Current results



Current results



Recap

- Reducing the cost of storage virtualization stack, avoiding Linux storage stack duplication by bypassing Dom0 storage stack
- Usage of open and widely used userspace driver
- A relevant technology $\Rightarrow$ NVMe is a widely deployed technology
- Keeping security provided by the Grant Table memory sharing mechanism of Xen

Future works

- Finish the READ optimization
- No copy of data ⇒ Zero-copy / DMA from guest memory
- Ideas to improve the Grant Table interface