redhat.

Using NVDIMM under KVM

Applications of persistent memory in virtualization

Stefan Hajnoczi <stefanha@redhat.com>
FOSDEM 2017

About me

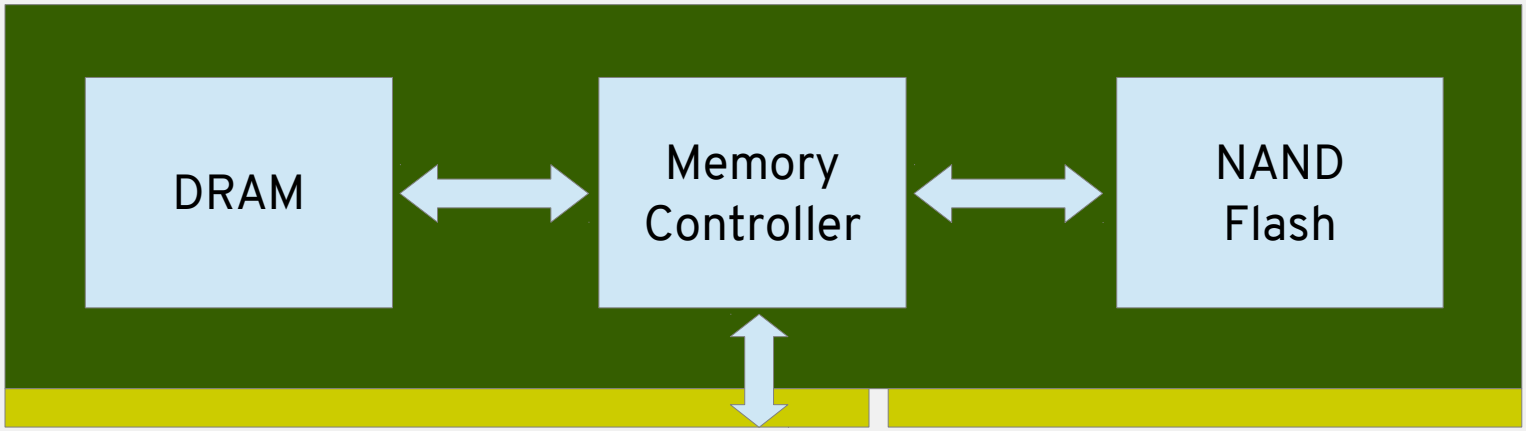
QEMU contributor since 2010

Focus on storage, tracing, performance

Work in Red Hat's virtualization team

Reviewer of NVDIMM emulation patches in QEMU

NVDIMM-N hardware



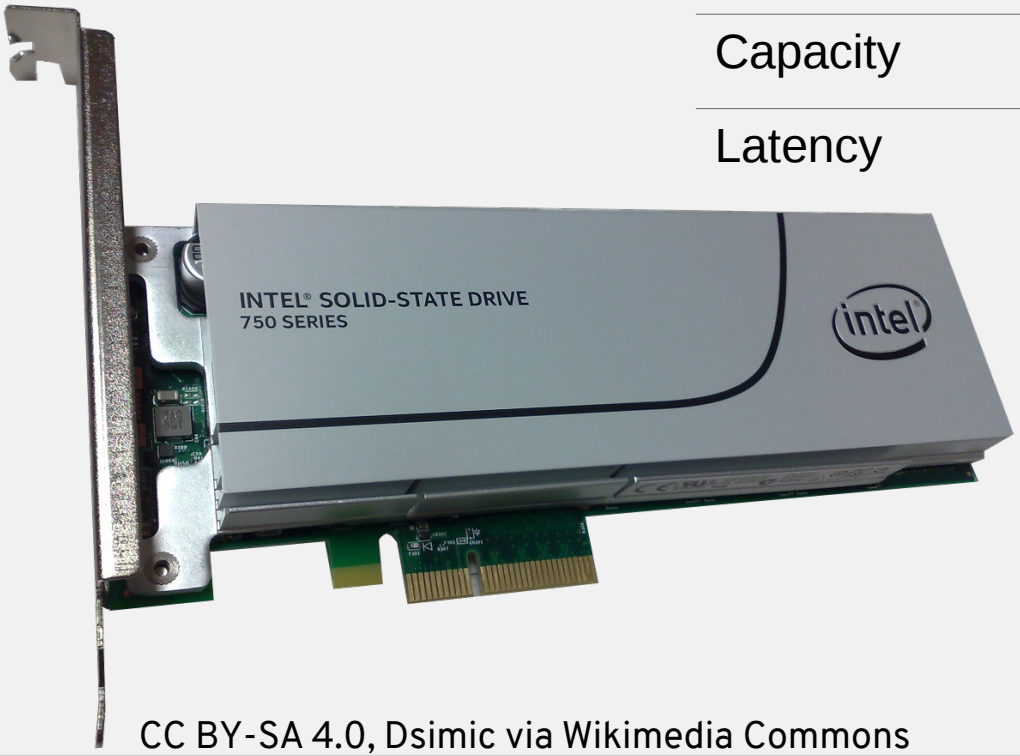
It's DDR4 RAM with one key feature:

Saves data to flash in event of power failure

Details in JEDEC JESD245 & JESD248 standards

Not to be confused with NVMe

	NVDIMM	NVMe
Form factor	DIMM	PCIe
Device type	Memory	Block
Capacity	10's of GB	1's of TB
Latency	10's of ns	10's of us



CC BY-SA 4.0, Dsimic via Wikimedia Commons

Both are non-volatile but otherwise totally different device types

Use cases for NVDIMM

Really fast writes particularly interesting for:

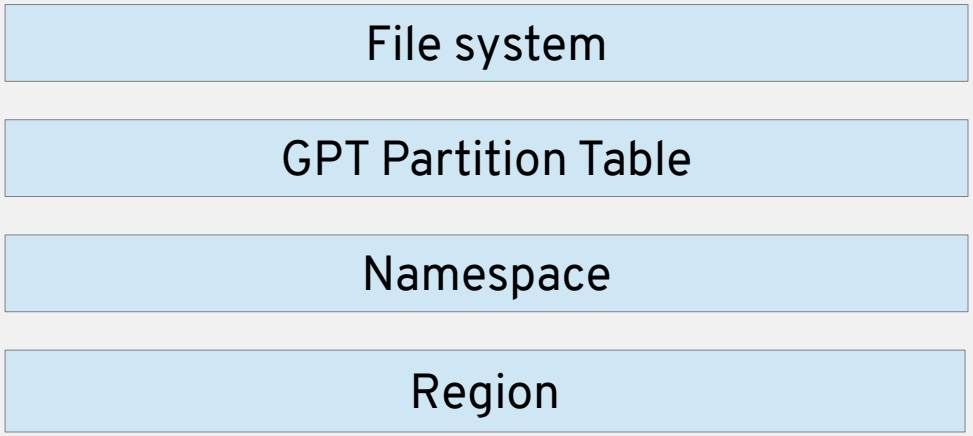
In-memory databases – get persistence for free*!

Databases – transaction logs

File & storage systems – frequently updated metadata

* need to follow programming model (explained later)

Managing data on NVDIMMs



Multiple NVDIMMs can be interleaved in a *region*
Regions are carved up into *namespaces*
Standard GPT/file system/etc stack inside namespaces

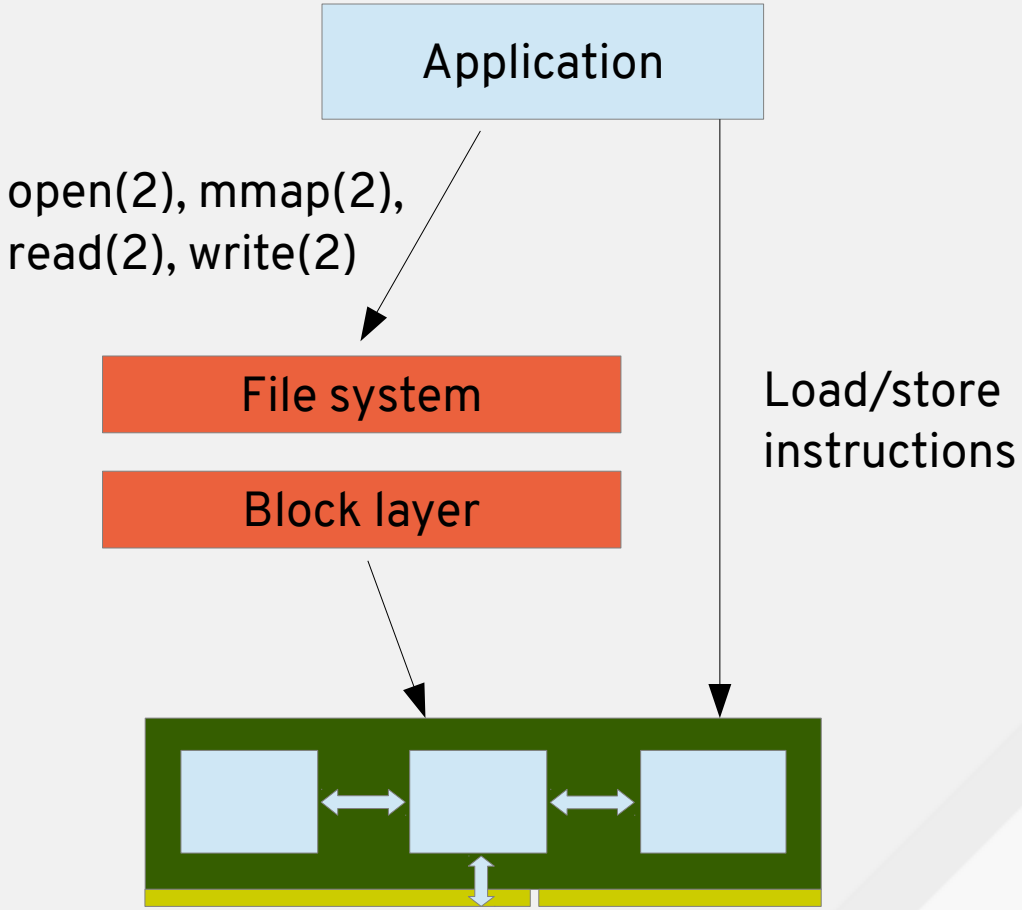
Data is identified by filename or device path

Bypassing the I/O stack

I/O bypasses kernel when accessing mmap of pmem via *DAX* device

Linux kernel has DAX support

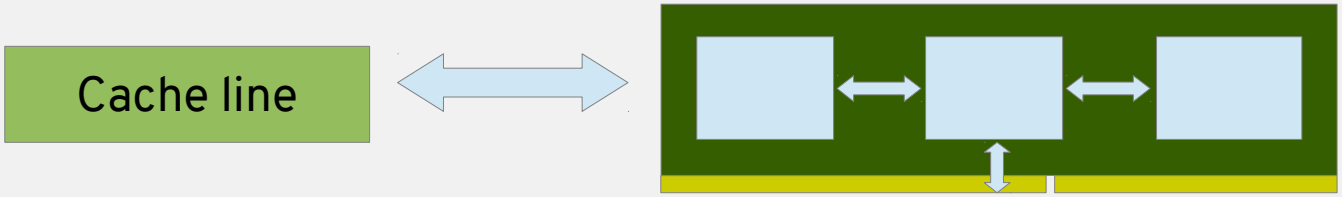
DAX means page cache is bypassed



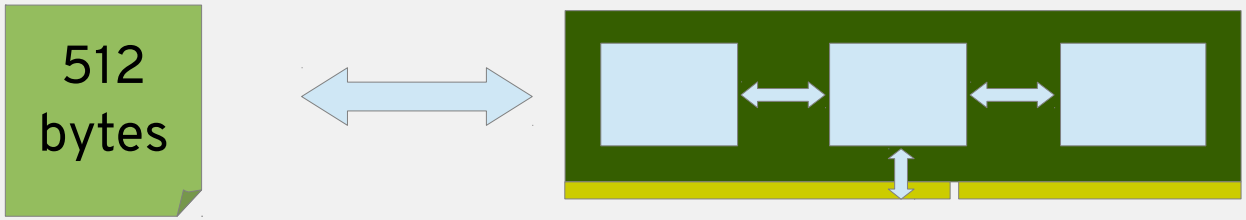
Programming model

Modes of operation:

1) Persistent memory – byte-addressable



2) Block window – block I/O



Described in pmem.io specifications

Persistent memory mode

Load – use regular load instructions

Store – flush cache line after store or
use non-temporal store

Error handling – Machine Check Exception on read but hard to handle in applications

Robustness – Map only data you need to protect against stray writes or use Memory Protection Keys

Block window mode

Block device semantics:

- Sector-based I/O
- Immediate error notification
- Data not exposed to stray memory writes

But:

- No DAX, traditional read(2)/write(2) only
- Hard to virtualize efficiently, not yet implemented in QEMU

ndctl utility and NVM Library

ndctl utility manages NVDIMMs, regions, and namespaces

https://github.com/pmem/ndctl

NVM Library APIs offer:

- Low-level access to pmem
- Higher-level data structures and memory allocators

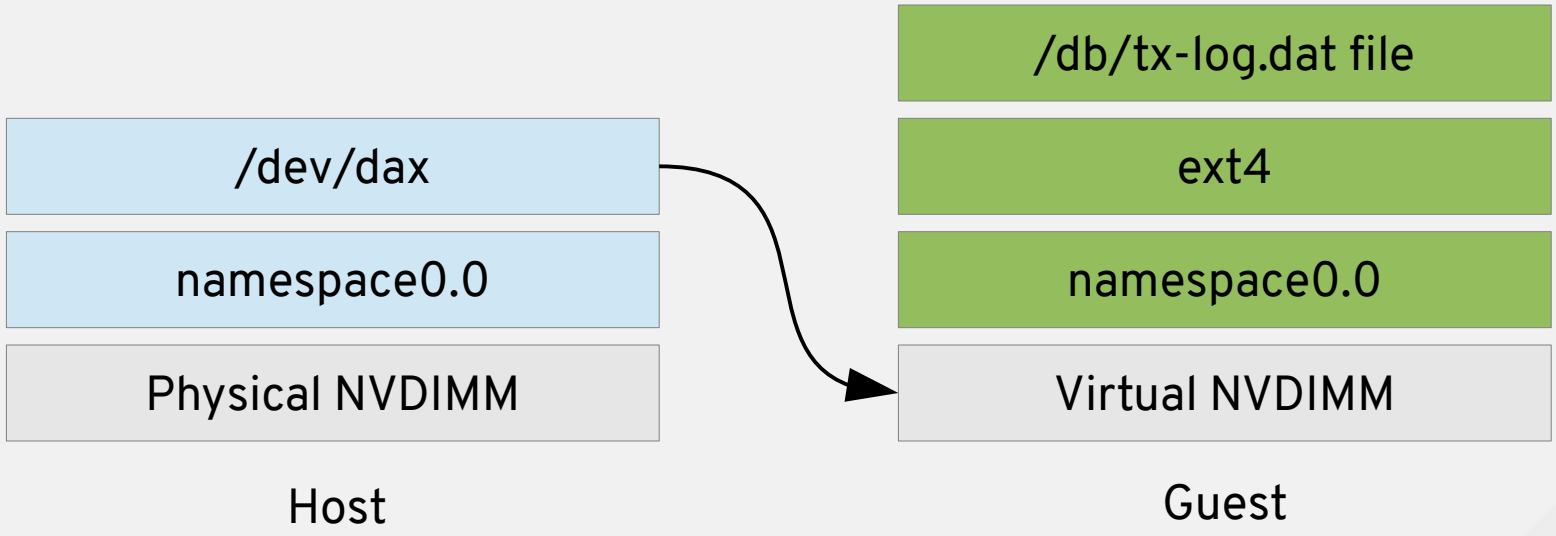
http://pmem.io/nvml/

NVDIMM pass-through in QEMU

Pass-through of entire namespace (files too in the future)

Label area is emulated, guest cannot alter host label area

Guest directly accesses host pmem – no vmexits!



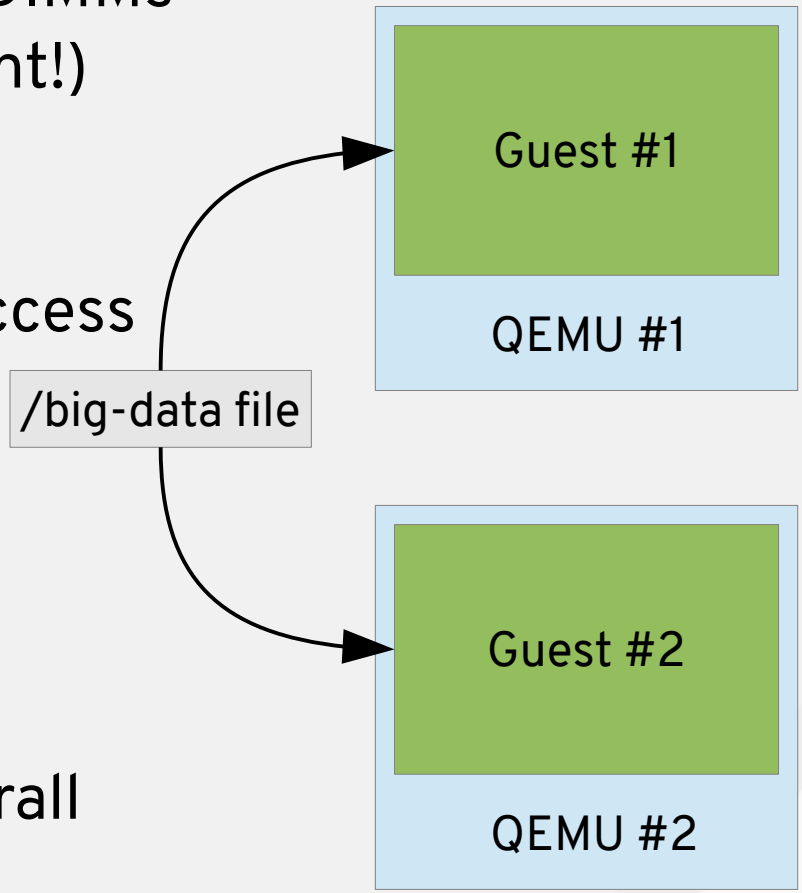
Fake NVDIMM in QEMU

Non-DAX host files as guest NVDIMMs
(Careful: stores are not persistent!)

Example:
Two guests sharing read-only access to a host file

Bypasses guest page cache if DAX is enabled inside guest

Avoids copy-in and reduces overall memory footprint



Future QEMU use cases

QEMU maintains frequently updated metadata:

- Allocation maps and refcounts in disk image files
- Dirty bitmap for incremental disk backup

NVDIMM could be used to speed up these features

Requires extensions to disk image formats to split frequently used metadata into separate DAX file

Thank you

Application developers → NVM Library: http://pmem.io/nvml/

High-level overview → SNIA NVM Programming Model (NPM) 1.1

https://goo.gl/d4YHPI

Low-level details → NVDIMM specifications: http://pmem.io/documents/

QEMU command-line syntax → docs/nvdimm.txt

Status February 2017:

Linux 4.1+	QEMU 2.6+	libvirt

My blog → http://blog.vmsplice.net/ IRC → stefanha on Freenode & OFTC

Special thanks to...

Haozhong Zhang

Ross Zwisler

Guangrong Xiao

Dan Williams

Jeff Moyer

...for feedback and discussion

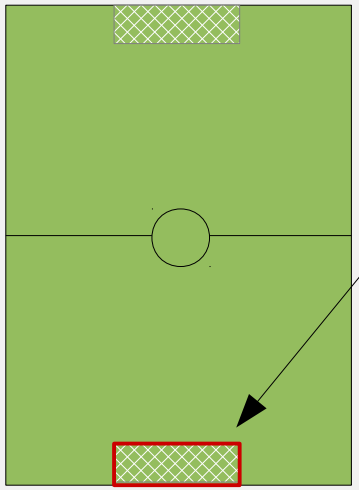
Backup slides

Persistence domains

A regular store instruction is not enough to make data persistent!

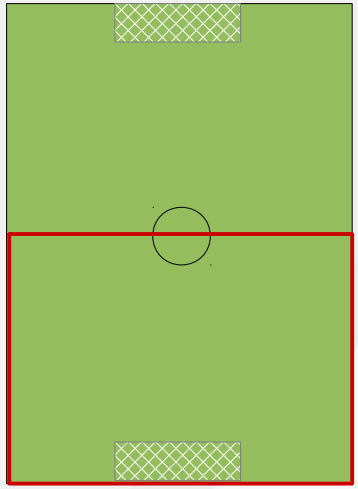
Data must reach hardware-dependent "*persistence domain*"

On Intel that means CLFLUSHOPT + SFENCE on platforms with ADR feature



Score if ball lands in goal

Score if ball lands anywhere on opposing side!



Block Translation Table

Provides **atomic sector I/O**

Prevents torn write problem if power failure occurs during a sector write operation

Optional layer on top of pmem or blk mode

Hardware availability

No widely available hardware on market (Feb 2017)

Intel, Micron, and HPE have announced products