Persistent Memory Over Fabrics

Paul Grun, Cray Inc
Stephen Bates, Eideticom
Rob Davis, Mellanox Technologies

Agenda

- Persistent Memory as viewed by a consumer, and some guidance to the fabric community
- Implications for building a Remote Persistent Memory service
- Approaches to prototyping

Scope for Today's Discussion

- Usage
  - Persistent Memory as a target of memory operations
  - Persistent Memory as a target of I/O operations (out of scope*)
- Locality
  - A PM device accessed over a network
  - A local PM device attached to an I/O bus or a memory channel (out of scope*)

*out of scope for this session

The Consumer's View

Some guidance to the fabric community

Paul Grun, Cray Inc

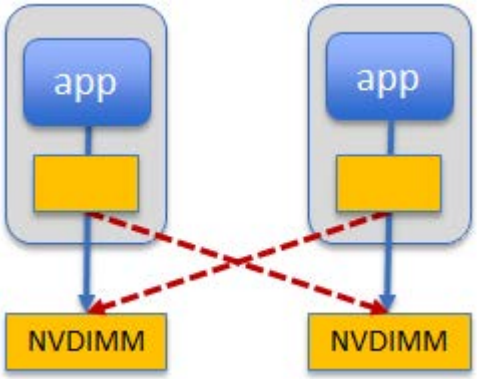
What the Consumer Would Like

The means to take full advantage of Remote Persistent Memory

- Treat it like memory as much as possible, while still
- Taking advantage of its persistence characteristics

Who are these consumers, and how will they use this new technology?

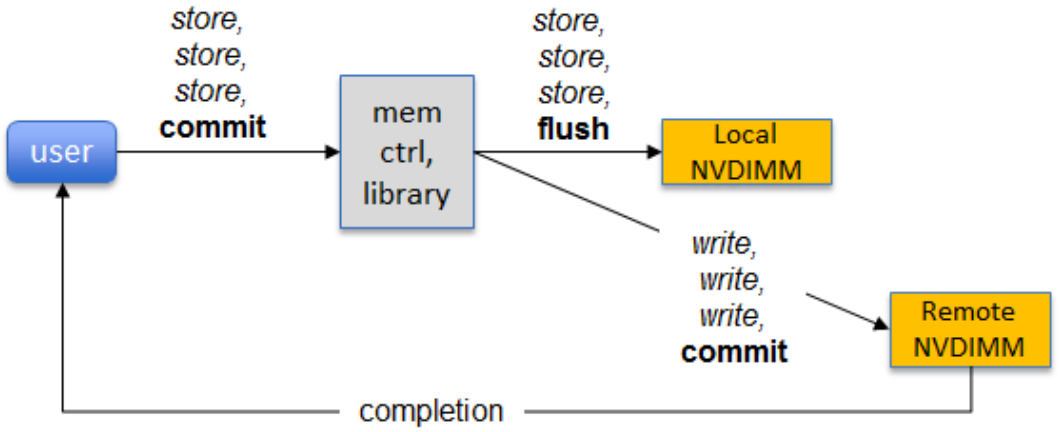
Use Case: High Availability, Replication



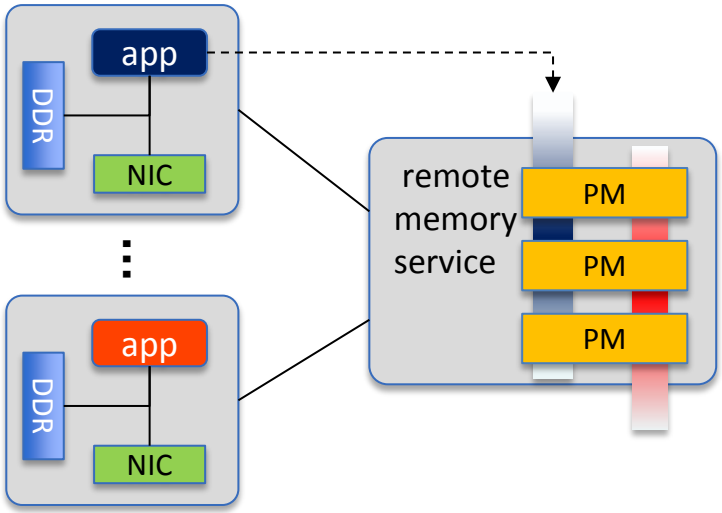
What it looks like

Usage: replicate data that is stored in local PM across a fabric and store it in remote PM

How it works



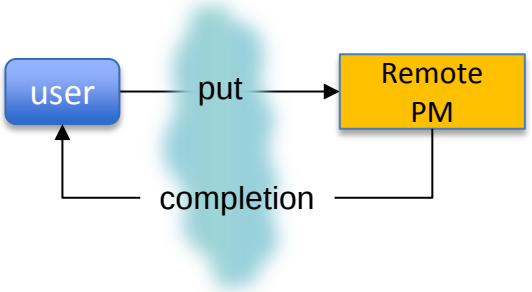
Use Case: Remote Persistent Memory



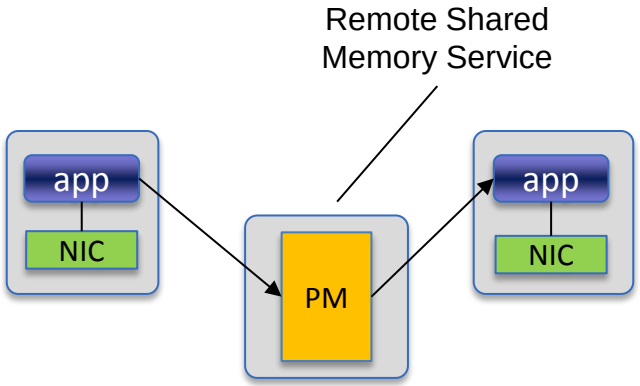
What it looks like

Usage: Expand on-node memory capacity, while taking advantage of persistence (or not). Disaggregate memory from compute.

How it works



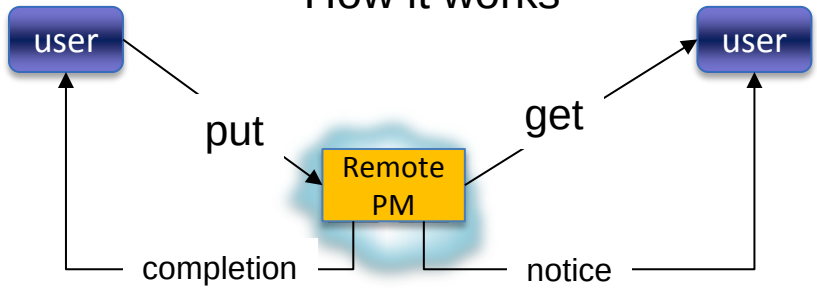
Use Case: Shared Persistent Memory



What it looks like

Usage: Information is shared among the elements of a distributed application. Persistence can be used to guard against node failure.

How it works



Objective for Fabric Developers

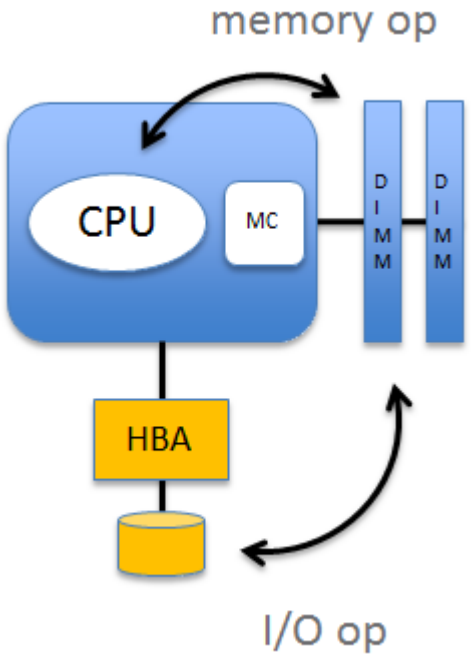
Present *Remote Persistent Memory* to the consumer as much like local memory as practicable

Yes, but what does that mean?

Memory vs I/O

Memory operations
Data is moved between a CPU register and a memory location. Memory location is identified by a real or virtual memory address. Fast and synchronous, while avoiding CPU stalls.

I/O
An extent (block) of data is transferred between memory and a storage device. On the storage device, the block is identified by an abstract, protocol-specific identifier (e.g. an LBA). Uses asynchronous I/O techniques.



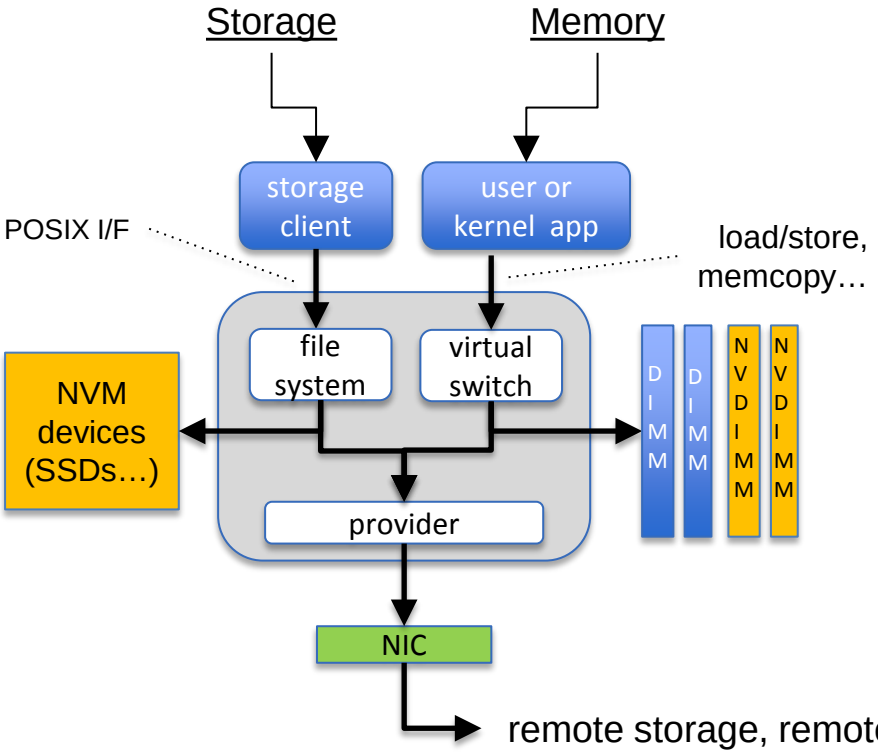
So What Do We Have to Do?

- Streamline the API to look more like memory operations
  → Use memory references instead of storage identifiers
  → Focus on puts and gets instead of block read/block write

- Manage asynchronicity – a necessary evil
  → Explicit control over when persistence occurs
  → Create synchronization points using fabric acknowledgements

- Make it FAST
  → Emphasize wire efficiency, eliminate round trips
  → No software in the target

Streamlined APIs



Think about accessing PM using memory paradigms, as opposed to storage paradigms…

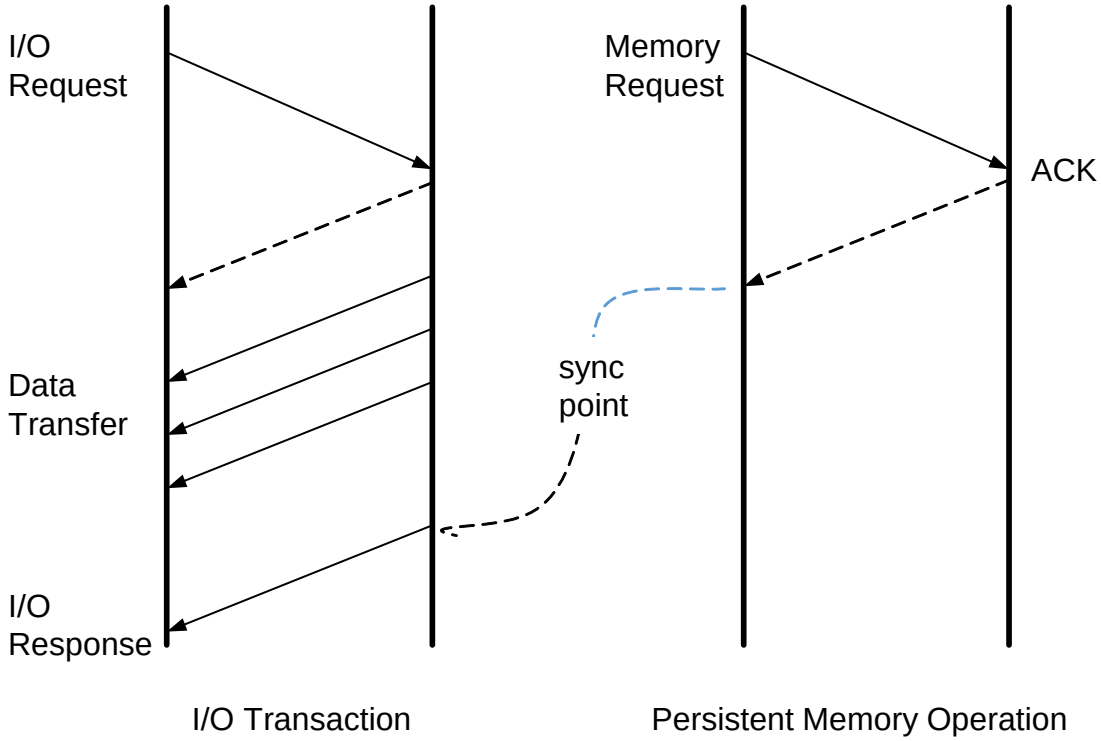
…even if the data at the far end is stored as a memory mapped file.

Managing Asynchronicity

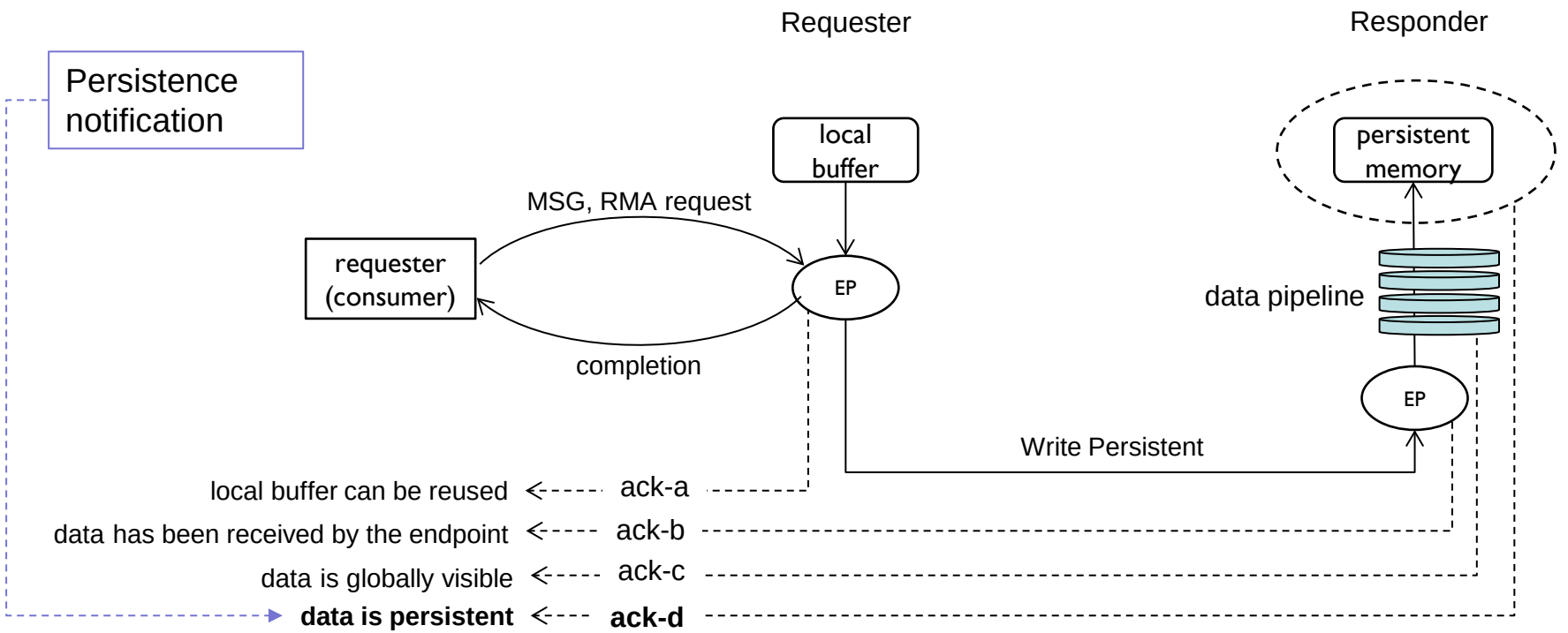
Storage protocols have a synch point built in to the protocol

But operations to remote persistent memory do not

→ Mechanisms to control persistence, and to achieve synchronization, have to be added to the API and the fabric protocol

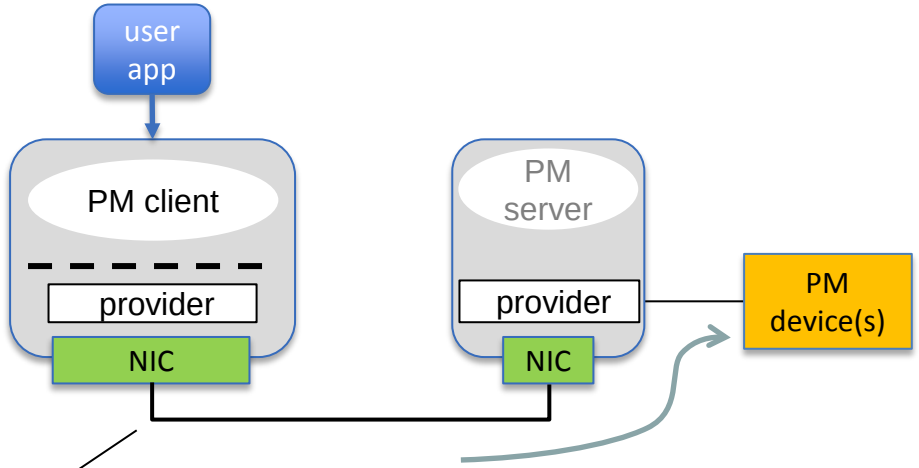


Managing Asynchronicity



What We'll Need

Enhanced APIs
- Expose PM device characteristics via the API
- Memory registration
- Put/Get style semantics
- Control persistence
- Completion semantics



- faster wires (always)
- small message performance
- efficient wire utilization
- synchronization (completions) in hardware

- more sophisticated access mechanisms
- more capable NICs
- improved wire protocols

What We'll Need

- Better APIs
  → Match the lingua franca of the application
  → Incorporate semantics to control Persistence

- Faster wires
  → Small messages at high transaction rates look more like memory ops

- Clever target devices
  → Eliminate layers at the target end

Objective – Make references to remote memory as fast as possible …
neglecting the speed of light and other practical considerations

Hardware and Software for PMoF Today!

Stephen Bates, Eideticom

Persistent Memory (PM)

Low Latency

Memory Semantics

Storage Features

Persistent Memory (PM) OVER FABRICS OF

2018

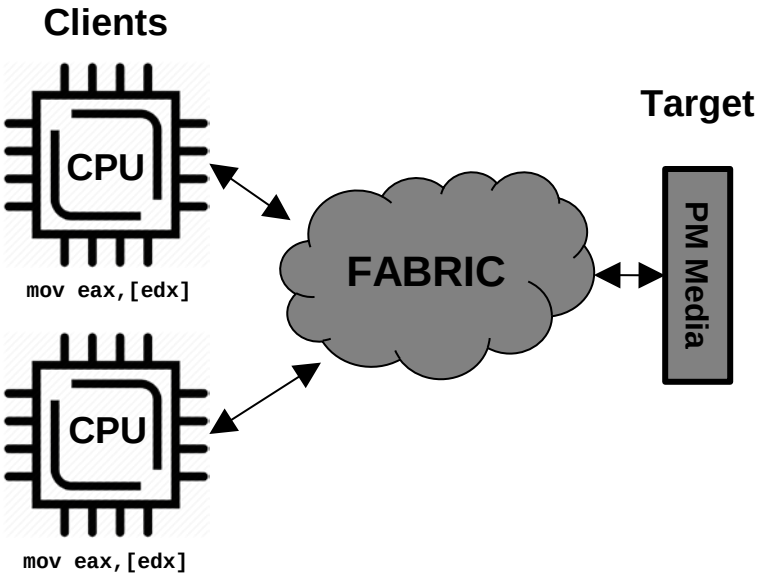
Low Latency

Memory Semantics

Storage Features

2018

The Holy Grail of PMoF



Loads and stores on a client CPU affect Persistent Memory across the fabric!

The knights that say “c”!

We are a loooong way from here!

Coming Soon to a Cinema Near You!

GEN Z

A New Fabric

featuring
Optional coherency
NVMe support
Scale

Coming in 2020

CCIX

The ARMpire Strikes Back

featuring
Off the CPU bus
Accelerator support
Cache coherency
Scale?

Coming Soon??

OpenCAPI

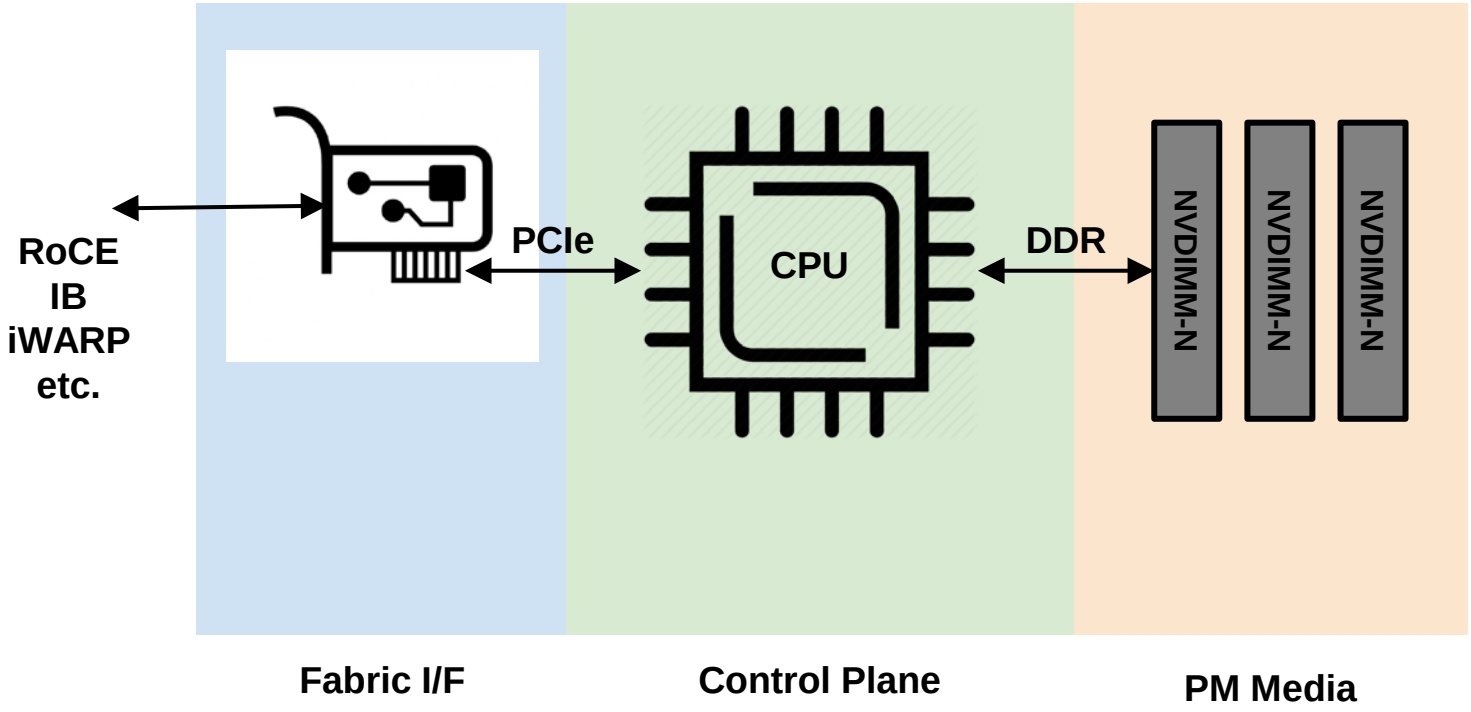
The Return of the Big Blue

featuring
Off the CPU bus
Accelerator support
Cache coherency

Now Showing in Select Cinemas

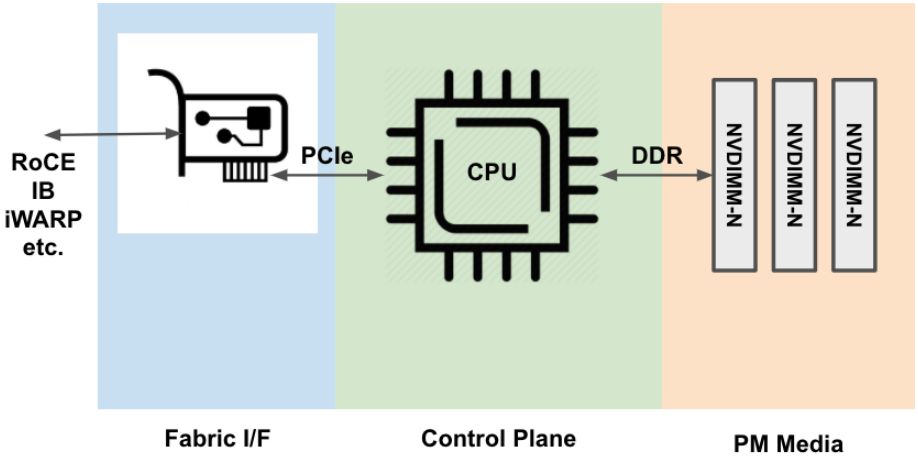
BUT WHAT CAN I SEE TODAY???

An PMoF Target Today: Hardware (v1-ddr)



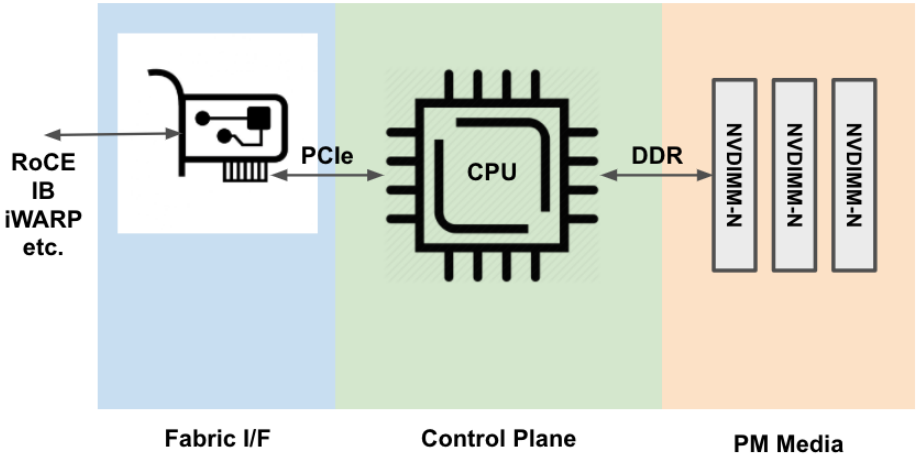
An PMoF Target Today: Hardware (v1-ddr)

Houston, we have some problems….



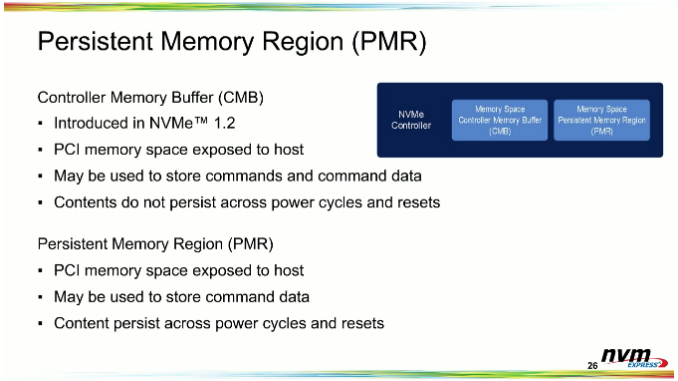
An PMoF Target Today: Hardware (v1-ddr)

- Fabric I/F
  - Require CPU utilization on the client side.
  - Not true load/store on client.
  - Challenging to scale.
  - Non-coherent (client and target)
- Control Plane
  - Uh, why is the CPU in the way?
  - Very CPU/ISA dependent
    - DDIO
    - cache effects
- PM Media
  - Expensive
  - Not hot-swappable
  - Capacity/Scale issues
  - MoBo support (ADR) required



NVMe Persistent Memory Regions

- A standards based PCIe PM interface!
- Takes a Controller Memory Buffer and adds persistence.
- Essentially a persistent PCIe BAR.
- Includes methods for write barriers and flushing.
- Not ARCH specific.
- Can be tied to RDMA so remote flush/barrier is possible.
- NVDIMM-PCIe ;-)....



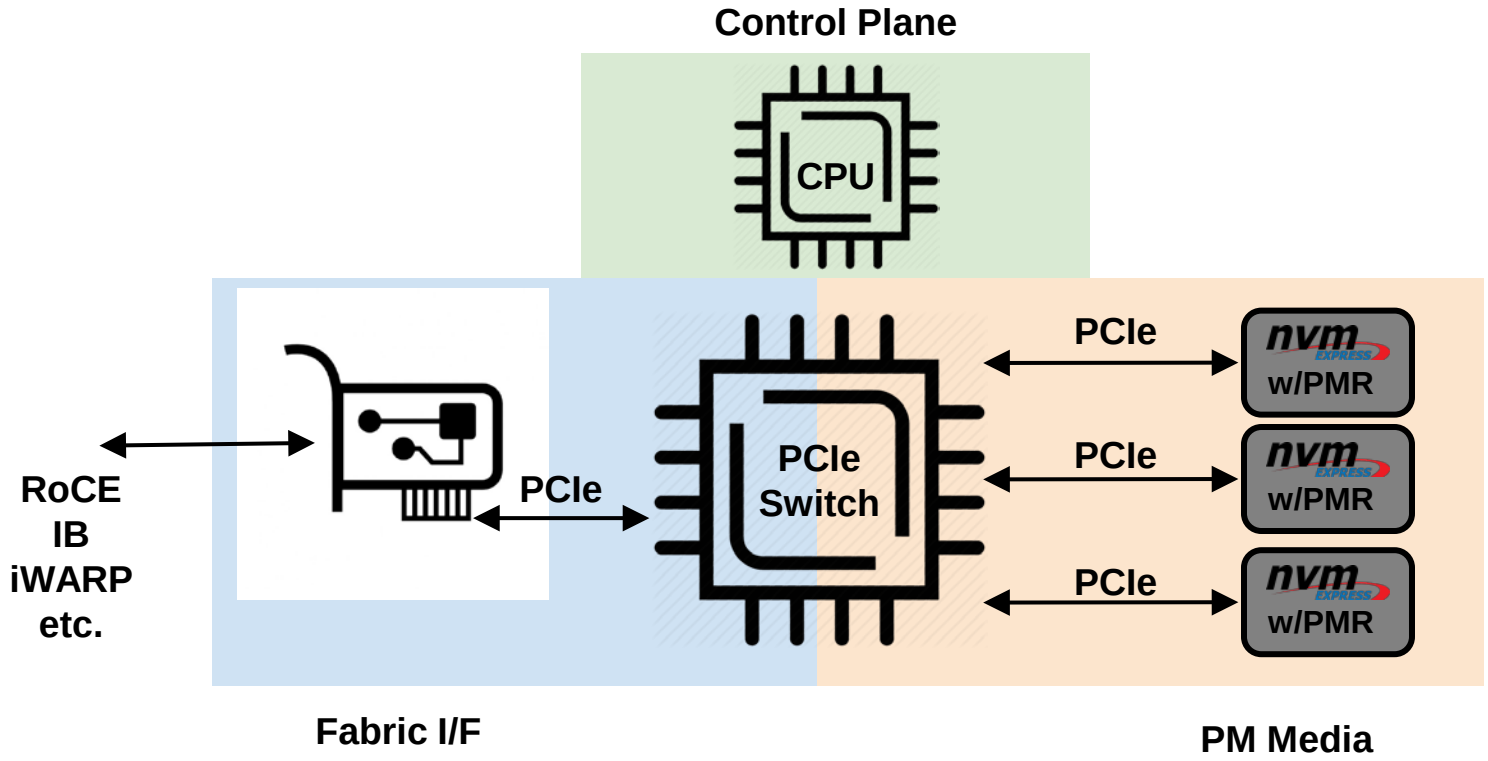
Microsemi

EVERSPIN TECHNOLOGIES The MRAM Company

TOSHIBA

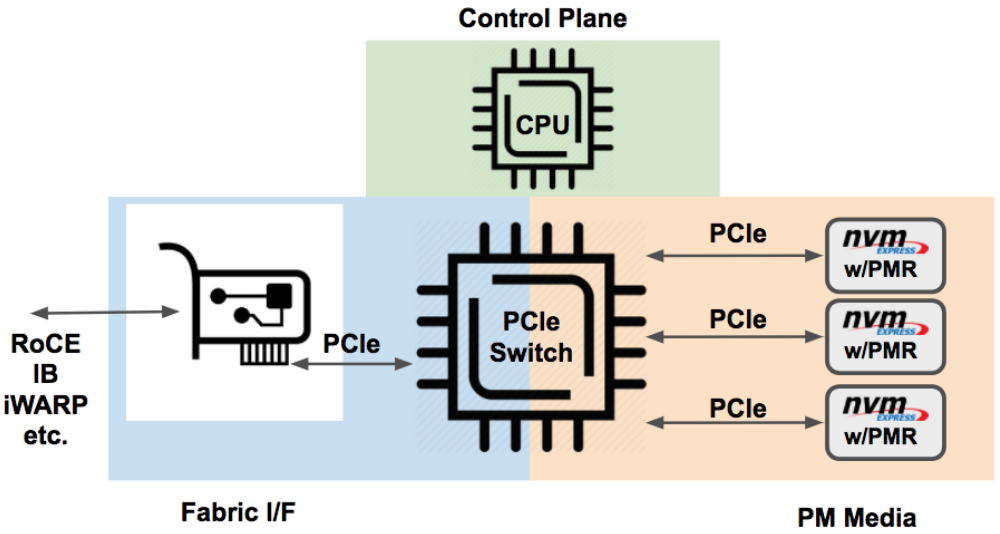
http://news.toshiba.com/press-release/electronic-devices-and-components/toshiba-memory-corporation-introduces-worlds-first-e

An PMoF Target Today: Hardware (v2-pcie)



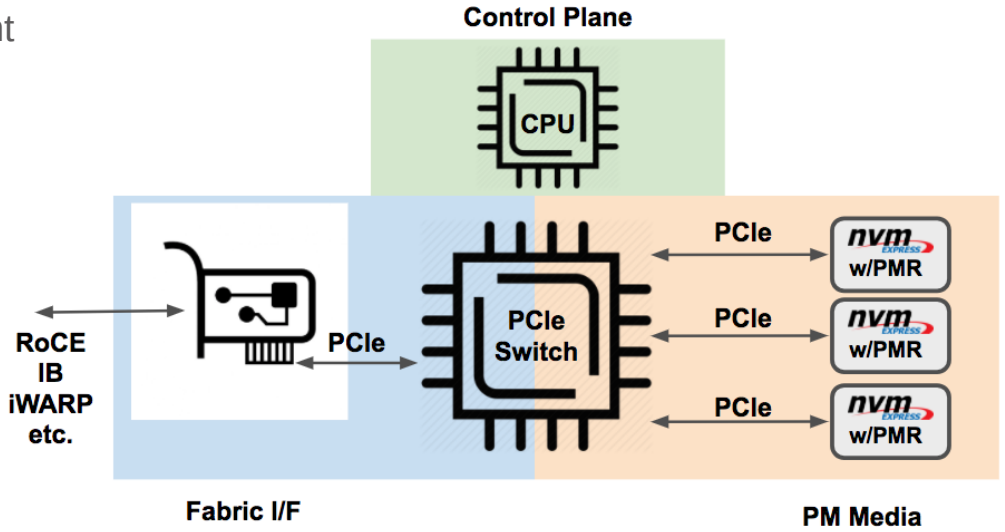
Building a PMoF Target Today: Hardware (v2-pcie)

Houston, we have fewer problems….



An PMoF Target Today: Hardware (v2-pcie)

- Fabric I/F
  - Require CPU utilization on the client side.
  - Not true load/store on client.
  - Challenging to scale.
  - Non-coherent (client and target)
- Control Plane
  - ~~Uh, why is the CPU in the way?~~
  - ~~Very CPU/ISA dependent~~
    - ~~DDIO~~
    - ~~cache effects~~
- PM MediaExpensive
  - ~~Not hot-swappable~~
  - ~~Capacity/Scale issues~~
  - ~~MoBo support (ADR) required~~



- Also
  - Decouples target-side CPU DDR from performance
  - Decouples target-side CPU PCIe from performance
  - Fabric I/F can be upgraded in time (Star Wars!)

Building a PMoF Target Today: Software

An PMoF Target Today: Software

Notable mentions include crail.io, memGluster and Octopus….

Protocol	Low-Latency	Memory Semantic	Storage Features	Over a Fabric	Comment
NVMe-oF	Yes	No	Yes	Yes	Block today, could be updated for PMRs
Ext4 w/DAX	Yes	Yes	Yes	No	DAX needs to be applied to remote FS
nPFS[1]	Yes	Yes	Yes	Yes	Does not actually exist yet ;-)
ZuFS	Yes	Yes	Yes	No	Cool but does not support fabrics
PMEM[2]	Yes	No	Yes	No	Turns PM into a block device.
librpmem[3]	Yes	Yes	Yes	Yes	Library to build upon!

[1] https://tools.ietf.org/id/draft-hellwig-nfsv4-rdma-layout-00.html
[2] https://nvdimm.wiki.kernel.org/
[3] http://pmem.io/pmdk/librpmem/


It’s the software stupid!

RDMA VERBs Extensions for Persistency and Consistency

Rob Davis, Mellanox

- Remote Direct Memory Access (RDMA) Background

RDMA – What?

- Remote
  - Data transfers between nodes in a network
- Direct
  - No Operating System Kernel involvement in transfers
  - Everything about a transfer offloaded onto Interface Card
- Memory
  - Transfers between user space application virtual memory
  - No extra copying or buffering
- Access
  - Send, receive, read, write, atomic operations
  - Byte or Block Access

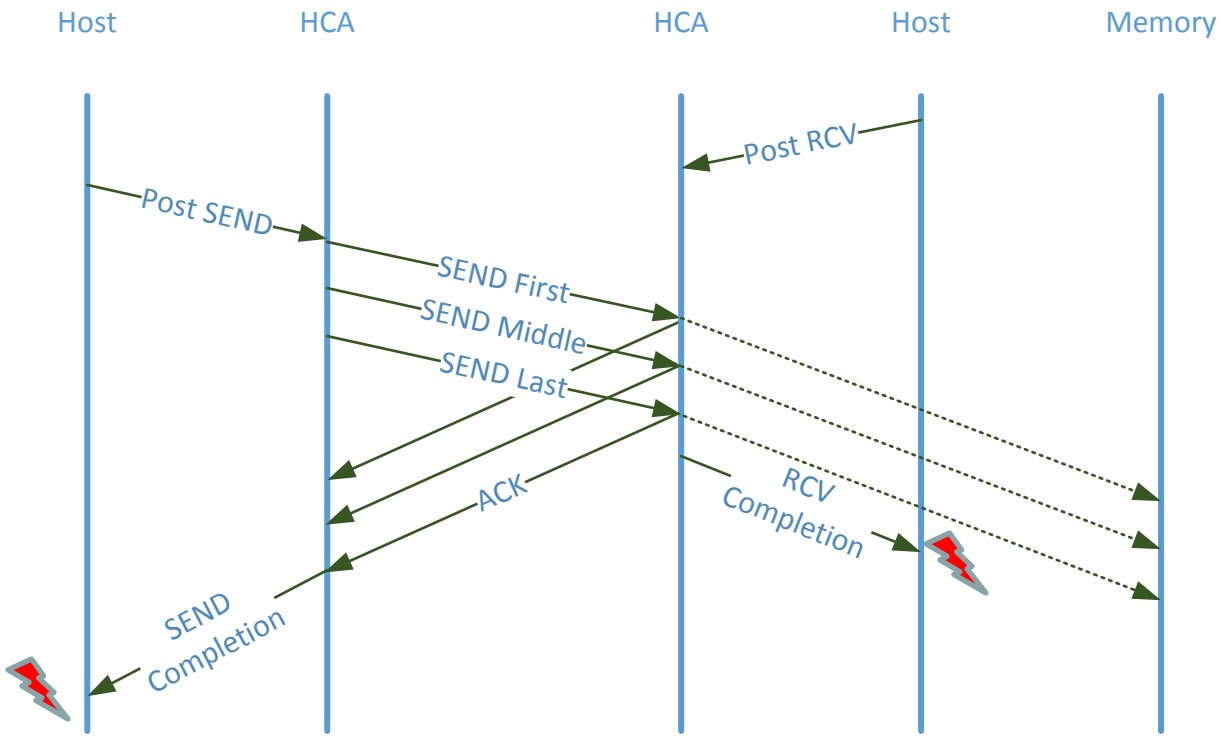
RDMA - Why?

- Latency (<usec)
- Zero-copy
- Hardware based one sided memory to remote memory operations
- OS and network stack bypasses
- Reliable credit base data and control delivery by hardware
- Network resiliency
- Scale out with standard converged network (Ethernet/InfiniBand)

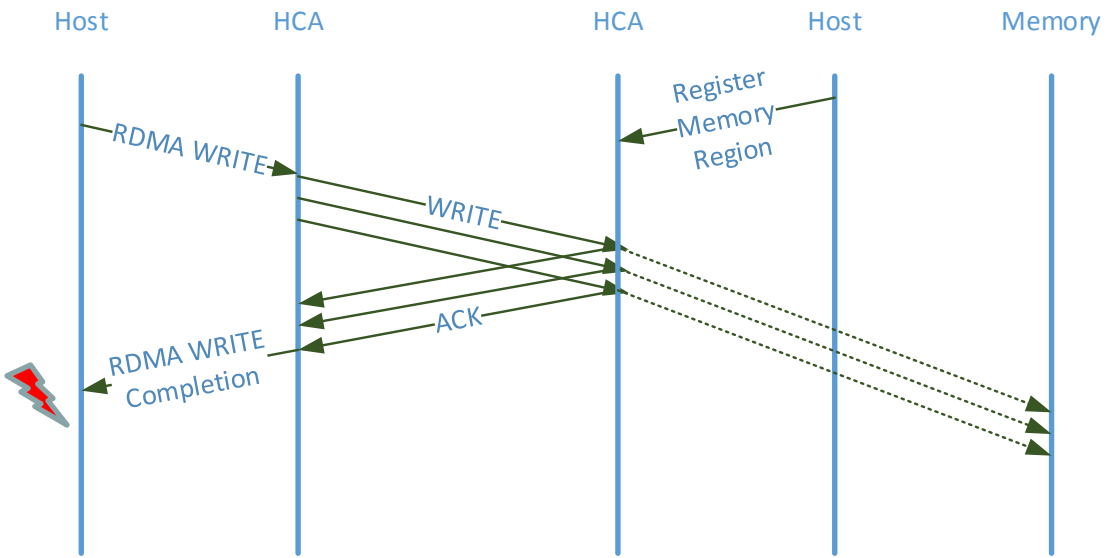
RDMA – How?

- Transport built on simple primitives deployed for 15 years in the industry
  - **Queue Pair (QP)** – RDMA communication end point
  - **Connect** for establishing connection mutually
  - RDMA **Registration** of memory region (REG_MR) for enabling virtual network access to memory
  - **SEND** and **RCV** for reliable two-sided messaging
  - RDMA **READ** and RDMA **WRITE** for reliable one-sided memory to memory transmission

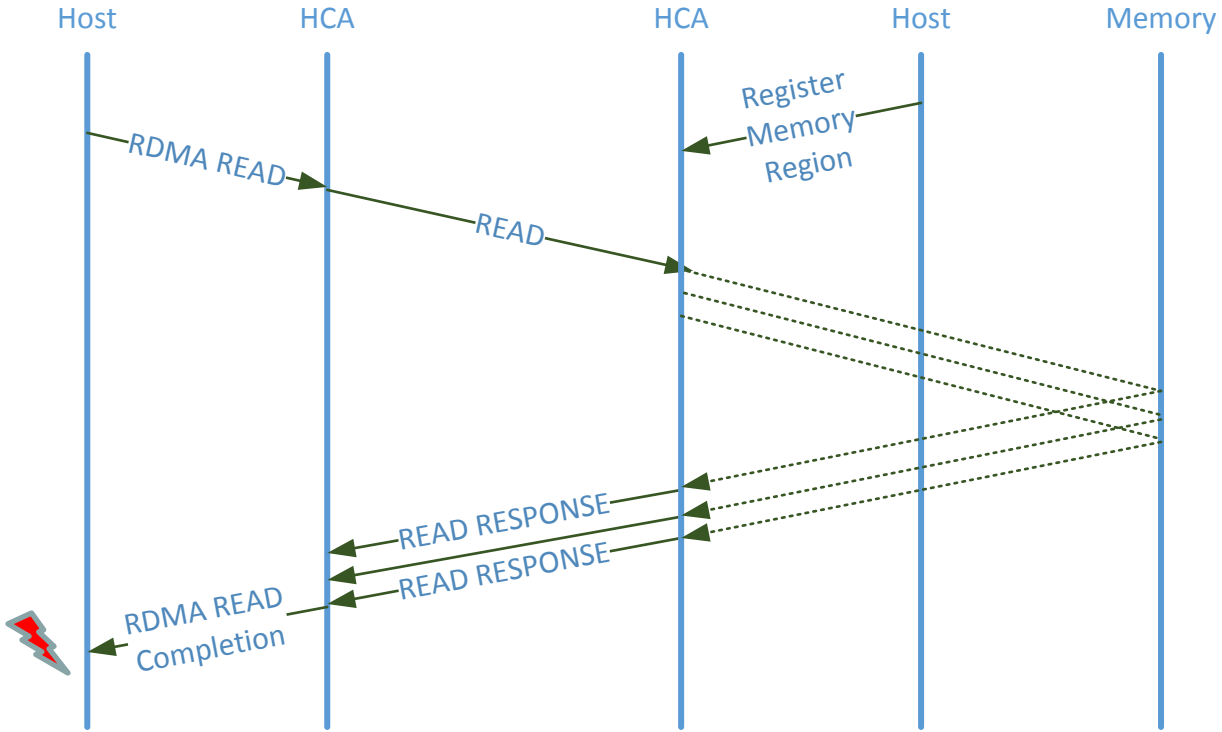
SEND/RCV



RDMA WRITE



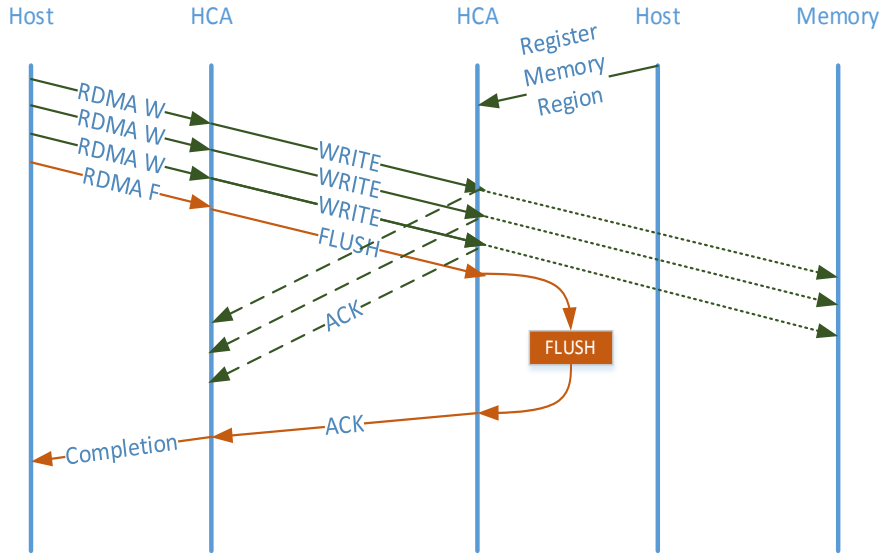
RDMA READ



- RDMA Memory Reliability Extensions for Persistency and Consistency

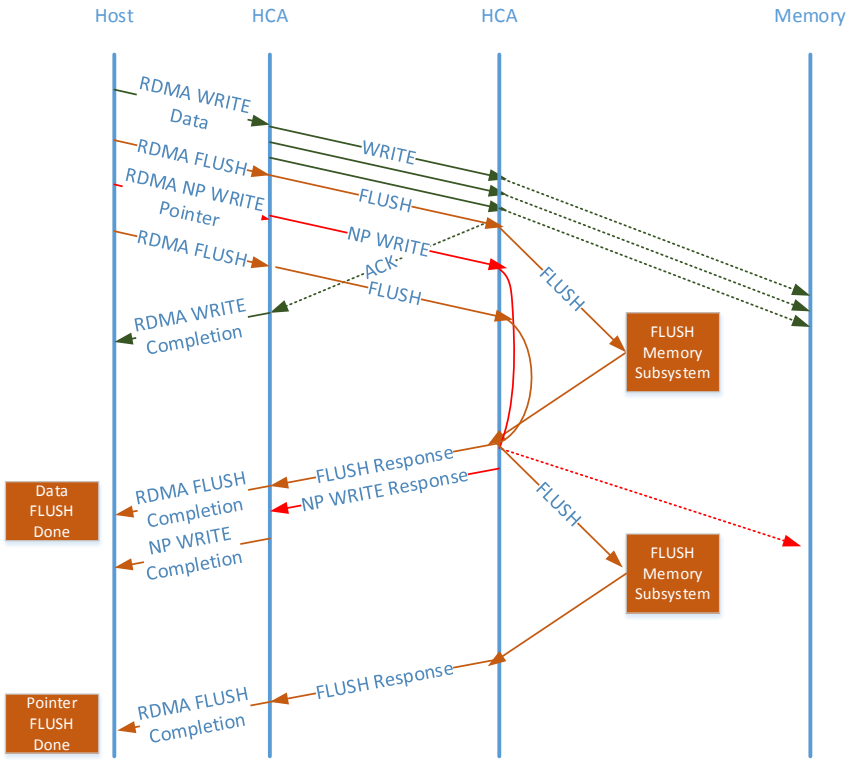
RDMA Extensions for Memory Persistency and Consistency

- RDMA FLUSH Extension
  - New RDMA operation
  - Higher Performance and Standard
  - Straightforward Evolutionary fit to RDMA Transport
- Target guarantees
  - Consistency
  - Persistency



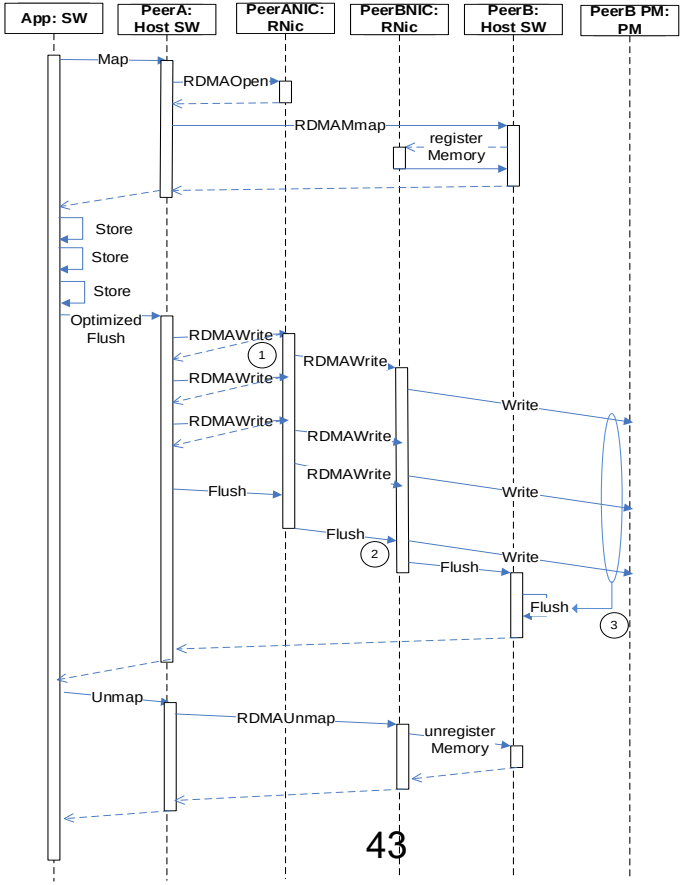
RDMA Non Posted WRITE

- Goal: Eliminate 2-Phase-Commit Requester-Fence-Roundtrip
  - …while maintaining RDMA FLUSH (NP) Semantics without blocking posted operations
- New Transport Operation: Non Posted WRITE
  - Leverages Native Non Posted Operations Semantics
    - Natural fit with existing transport protocol
    - Ordering
    - Constrained to responder resources limitation of number of outstanding operations
    - Error Handling (e.g. Repeated)
- Two phase commit example
  - Use Non-Posted WRITE after FLUSH for pointer update
  - Avoids need for Requester Side Fencing (extra roundtrip)



Matches SNIA NVM PM Model RDMA to PMEM for High Availability

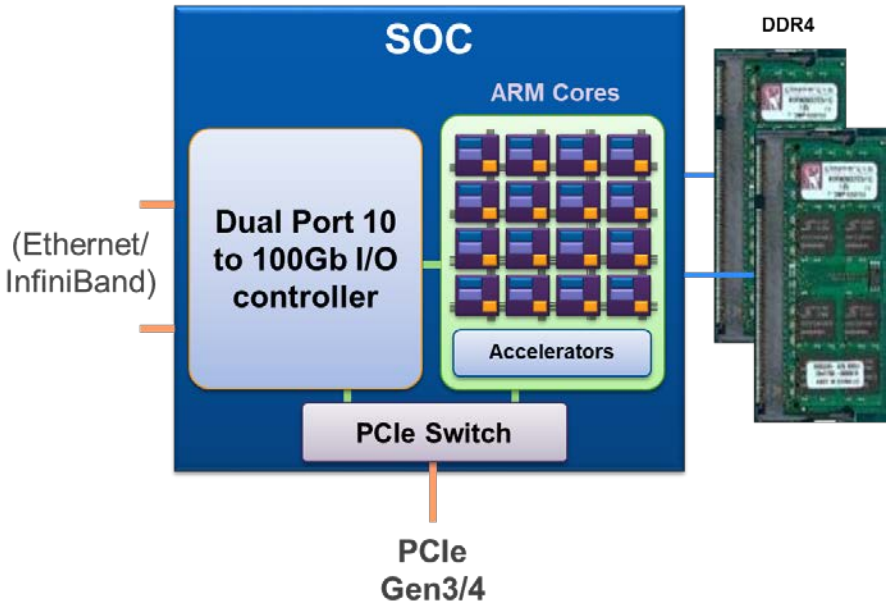
- MAP
  - Memory address for the file
  - Memory address + Registration of the replication
- SYNC
  - Write all the “dirty” pages to remote replication
  - FLUSH the writes to persistency
- UNMAP
  - Invalidate the registered pages for replication



- Platforms for trying RDMA for Persistent Memory over Fabrics (PMoF)

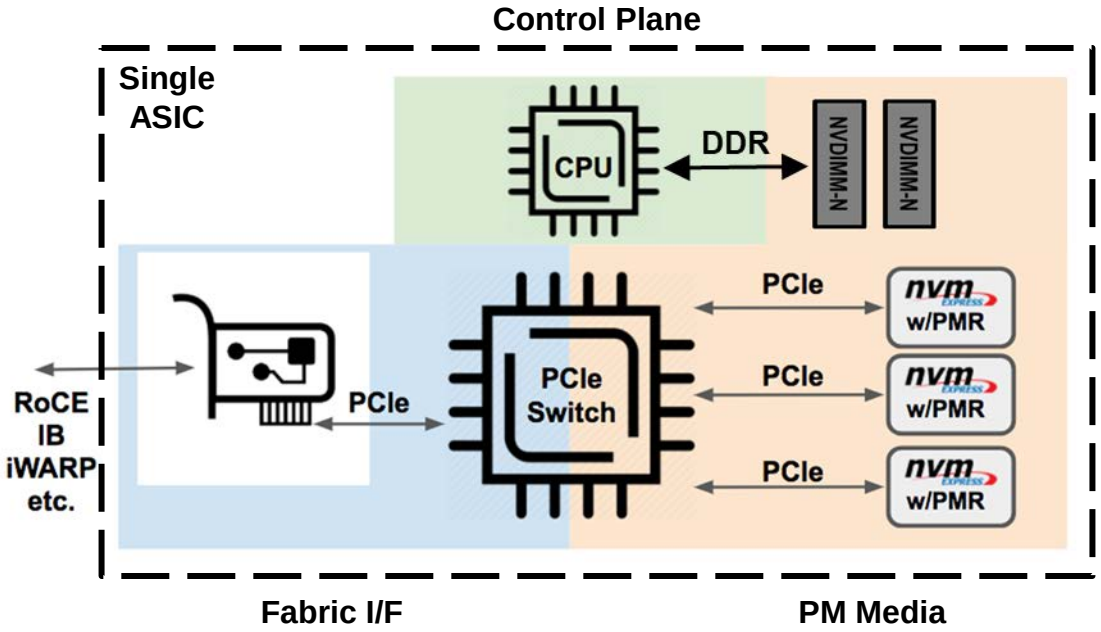
Available Today

- SOC reference platform(s)
  - Pluggable interfaces for PM devices
- High enough performance to test Persistent Memory over Fabrics (PMoF)
  - Over 8MIOPs with 512B MTU
  - Less than 3µsec latency
  - Less than 1% CPU utilization



Fit the requirements for PMoF testing

- Open source Programmable Control Plane
- Multiple standard 100Gb low latency IO interfaces
- Multiple standard persistent memory interfaces



Thanks!

grun@cray.com
stephen@eideticom.com
robd@mellanox.com