Persistent Memory over Fabrics

Rob Davis, Mellanox Technologies
Chet Douglas, Intel
Paul Grun, Cray, Inc
Tom Talpey, Microsoft

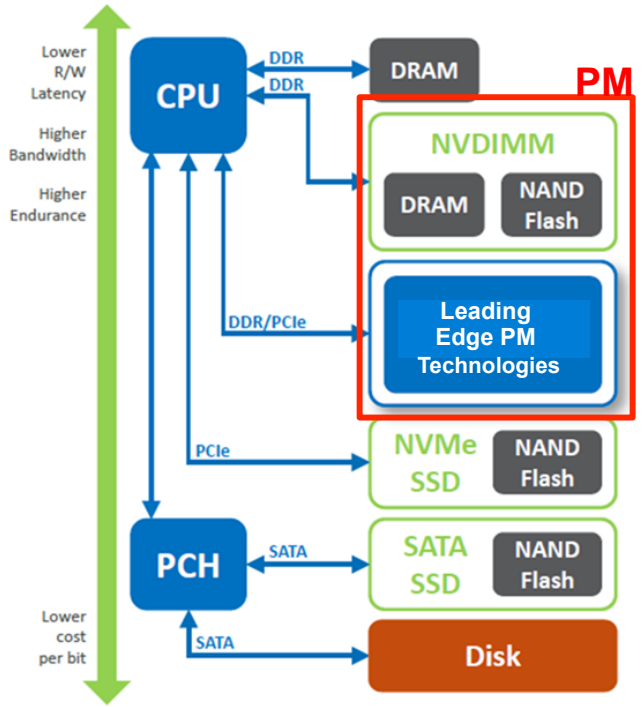
Agenda

- The Promise of Persistent Memory over Fabrics
  - Driving new application models
- How to Get There
  - An Emerging Ecosystem
  - Encouraging Innovation
  - Describing an API
- API Implementations
  - Flush/commit is the way to go
  - But how to optimize these operations?

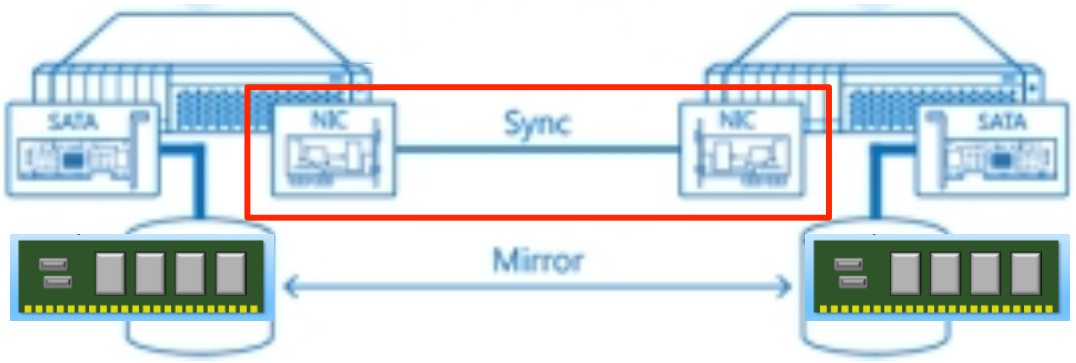
Flash Memory Summit

Persistent Memory (PM)

- Persistent Memory (PM) Benefits
  - Considerably faster than NAND Flash
  - Performance accessible via PCIe or DDR interfaces
  - Lower cost/bit than DRAM
  - Significantly denser than DRAM



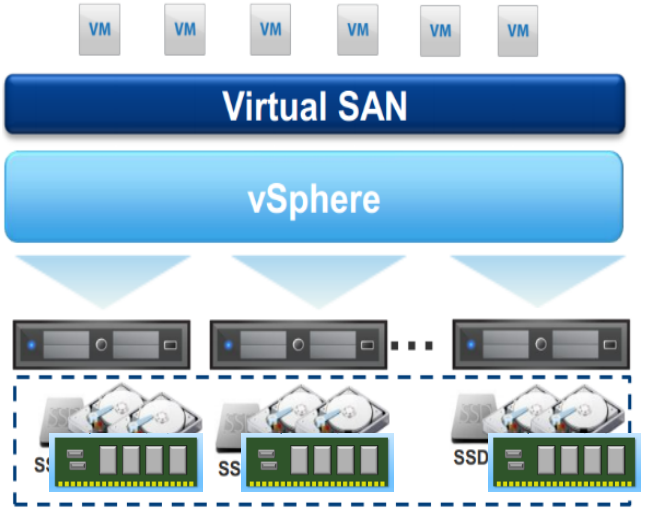
PM Needs High Availability (HA)



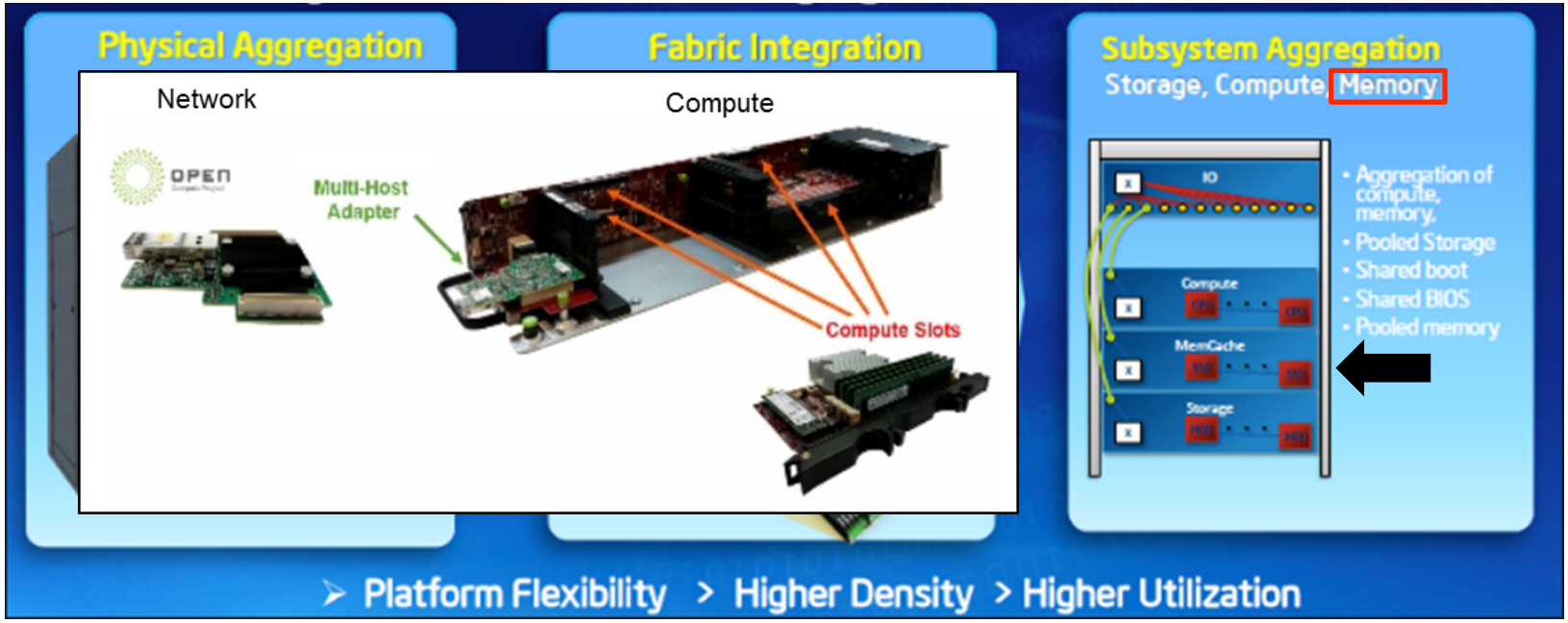
	PM	NAND
Read Latency	~100ns	~20,000ns
Write Latency	~500ns	~50,000ns

Applications Enabled by PM

- Compute Disaggregation
- Scale out file/object systems
- In-Memory Database
- Machine learning
- Hyperconverged Infrastructure



Compute Disaggregation



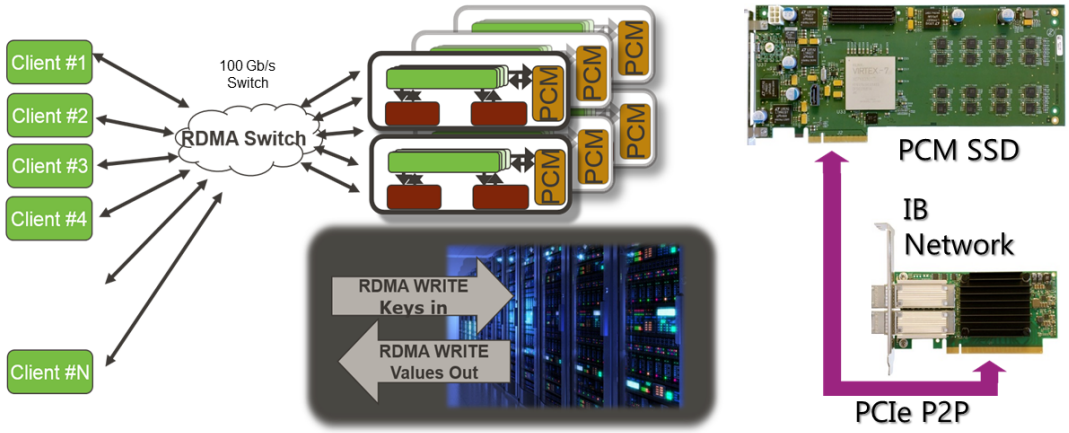
Core Requirements for Networked PM

- PM is really fast
  - Needs ultra low-latency networking
- PM has very high bandwidth
  - Needs ultra efficient, transport offload, high bandwidth networks
  - Remote accesses must not add significant latency
- PM networking requires:
  - Predictability, Fairness, Very Low Packet Loss

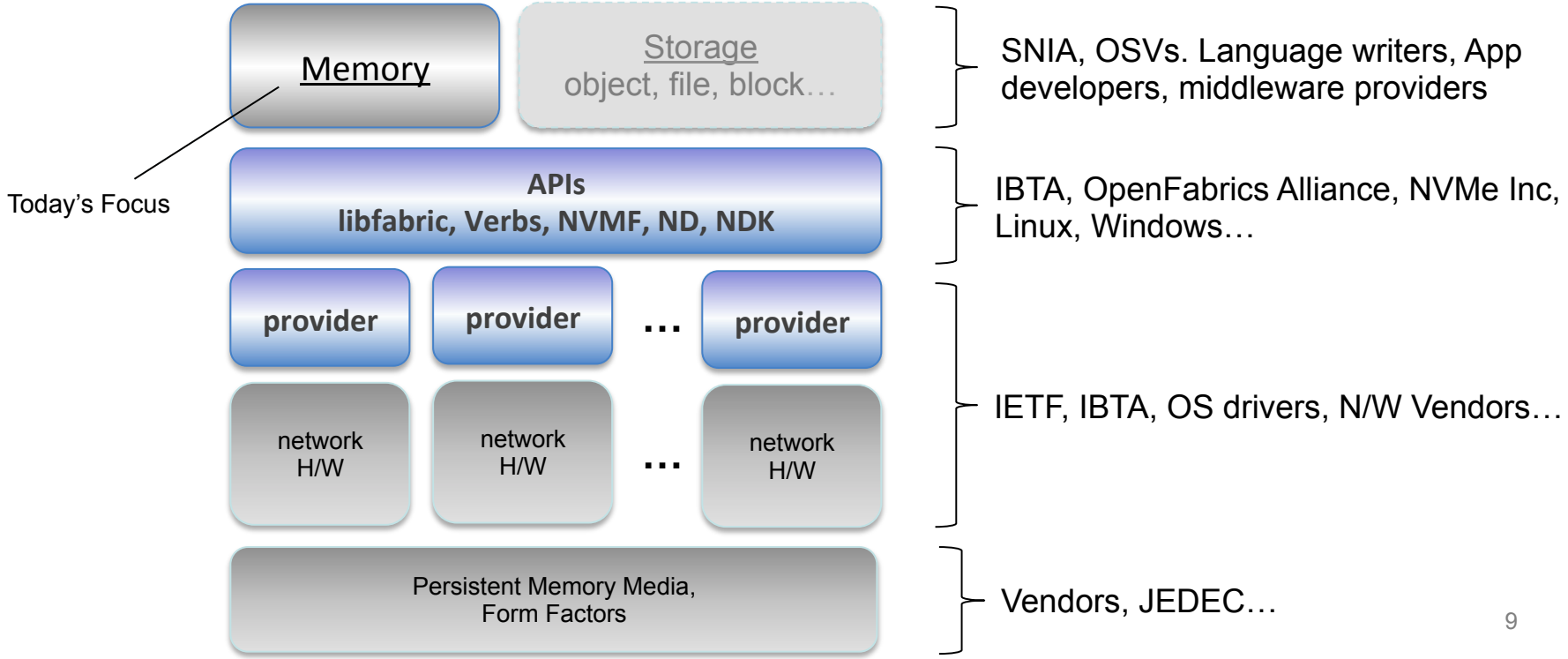
	PM	NAND
Read Latency	~100ns	~20,000ns
Write Latency	~500ns	~50,000ns

We Can Get There

- HGST live demo at FMS 2015
- PM based Key-Value store over 100Gb/s fabric
- Key-Value Store fetch from non-volatile memory in 5usec

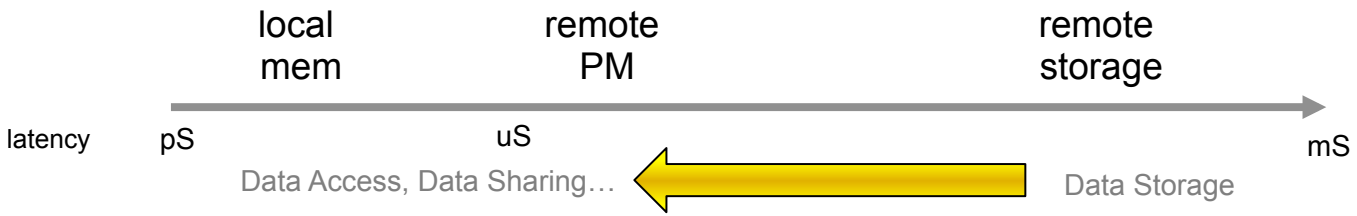


How to get there – the PM Ecosystem



No one argues with this…

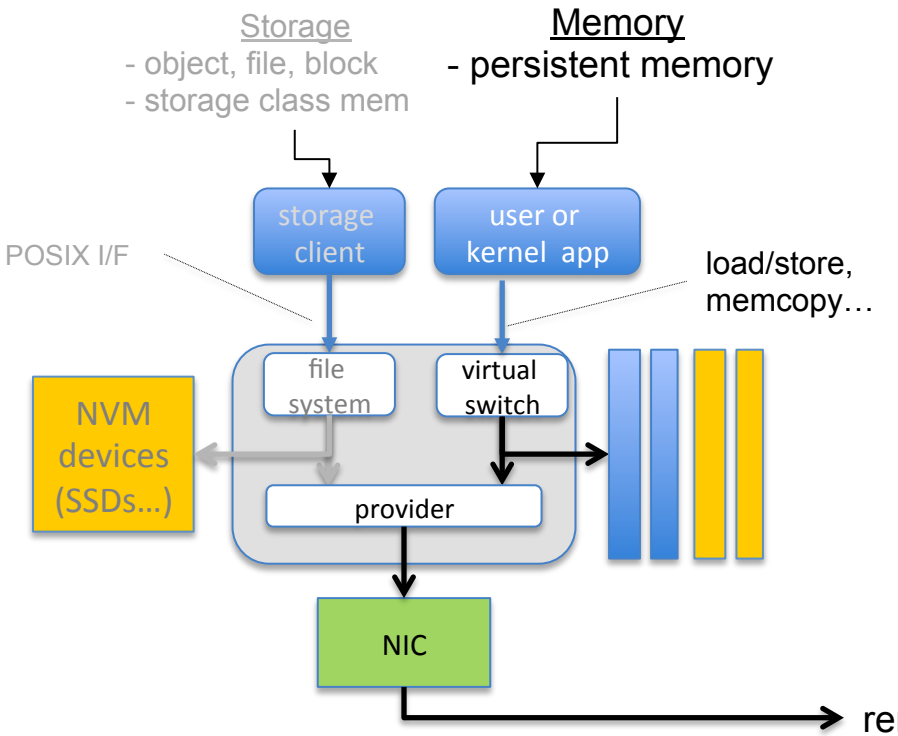
Persistent memory changes the way that applications store, access, communicate and share information



- What happens when apps can access shared information at near-memory speeds?
- How will apps evolve to take advantage of pools of global, shared, PM?
- What happens when apps evolve in the ways that they access, share and consume data?

What should network architects do to prepare?

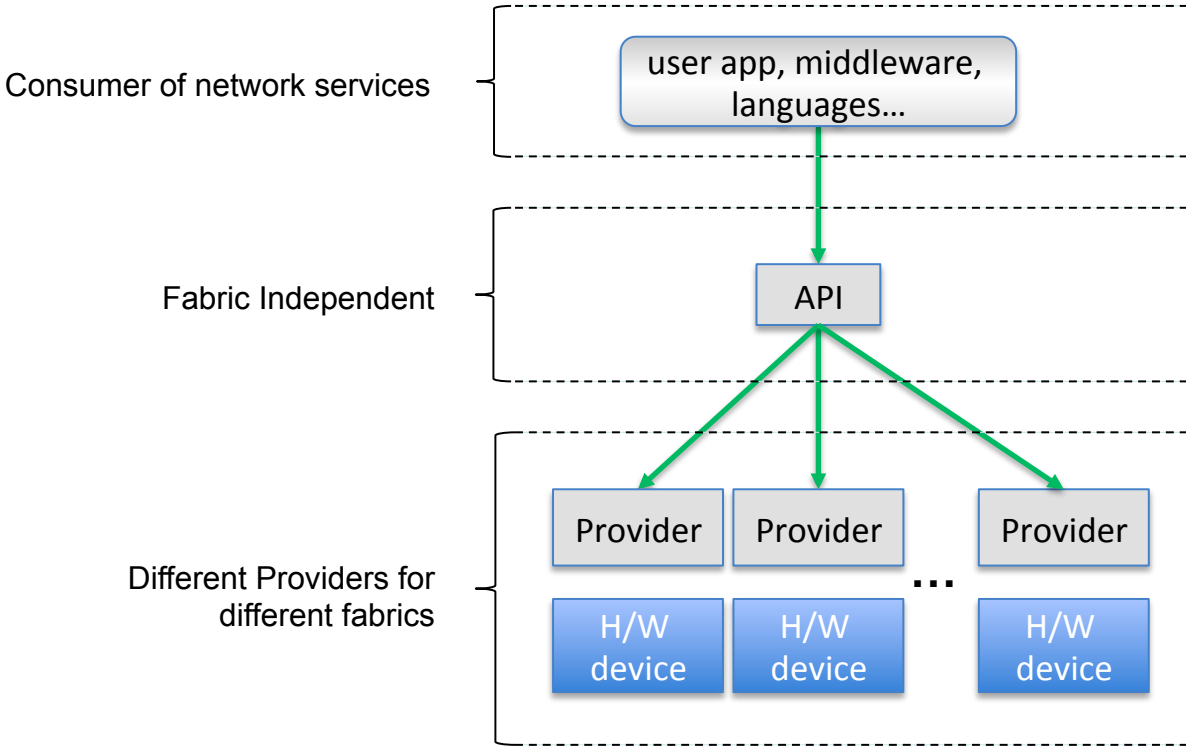
Focus on memory operations



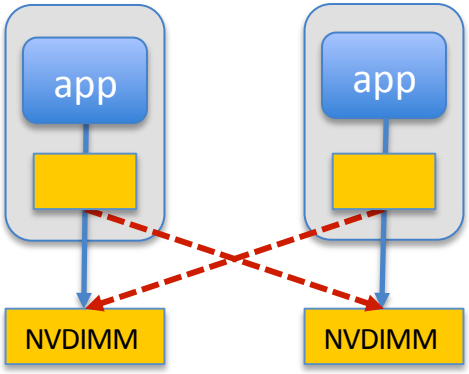
Think about accessing PM using memory paradigms, as opposed to storage paradigms…

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

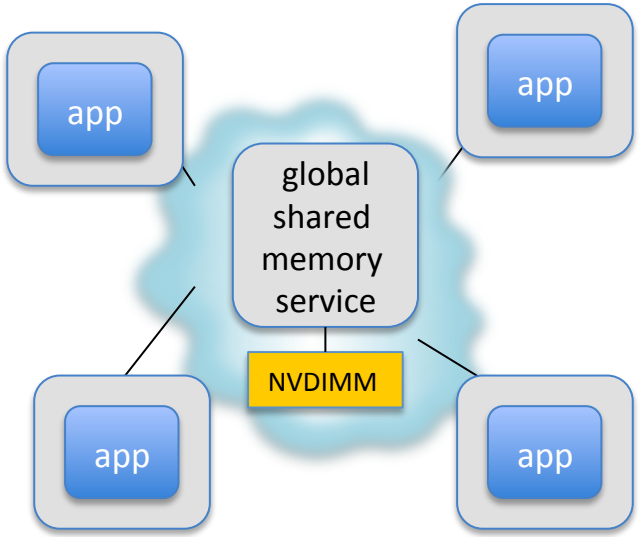
Simultaneous Equations to Solve



Representative Use Cases



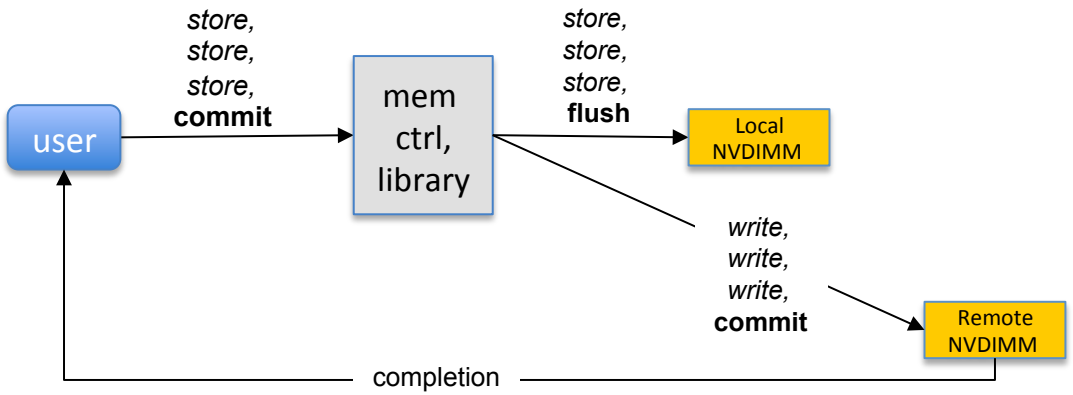
Replication
High Availability use case



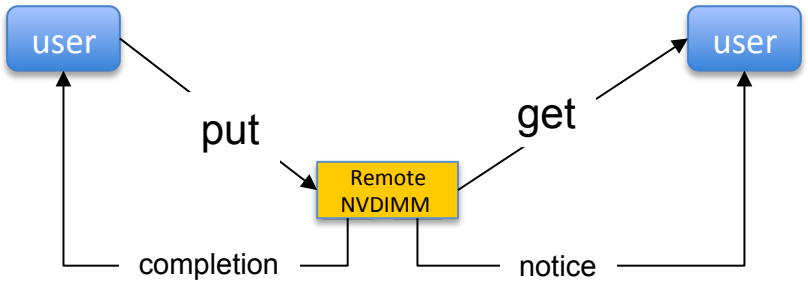
Shared PM
Remote Access, Storage use cases

How they work

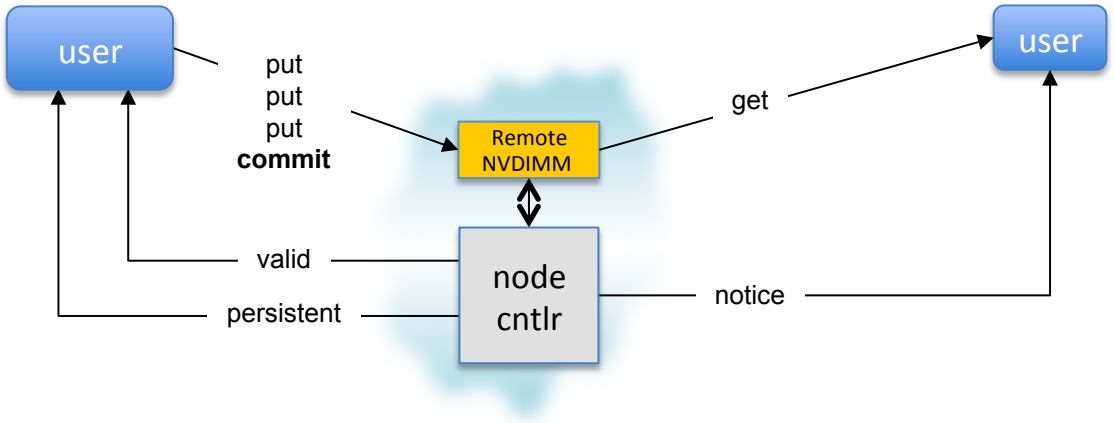
High Availability Use Case



Remote Shared Memory Use Case

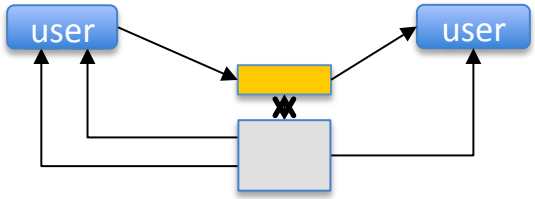


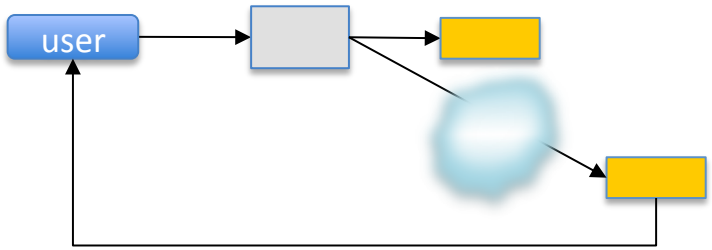
Shared PM - A Closer Look



API Requirement: Distinguish between data which *is globally valid*, and data which *has been made persistent*

The Beginnings of a List of API Reqmts





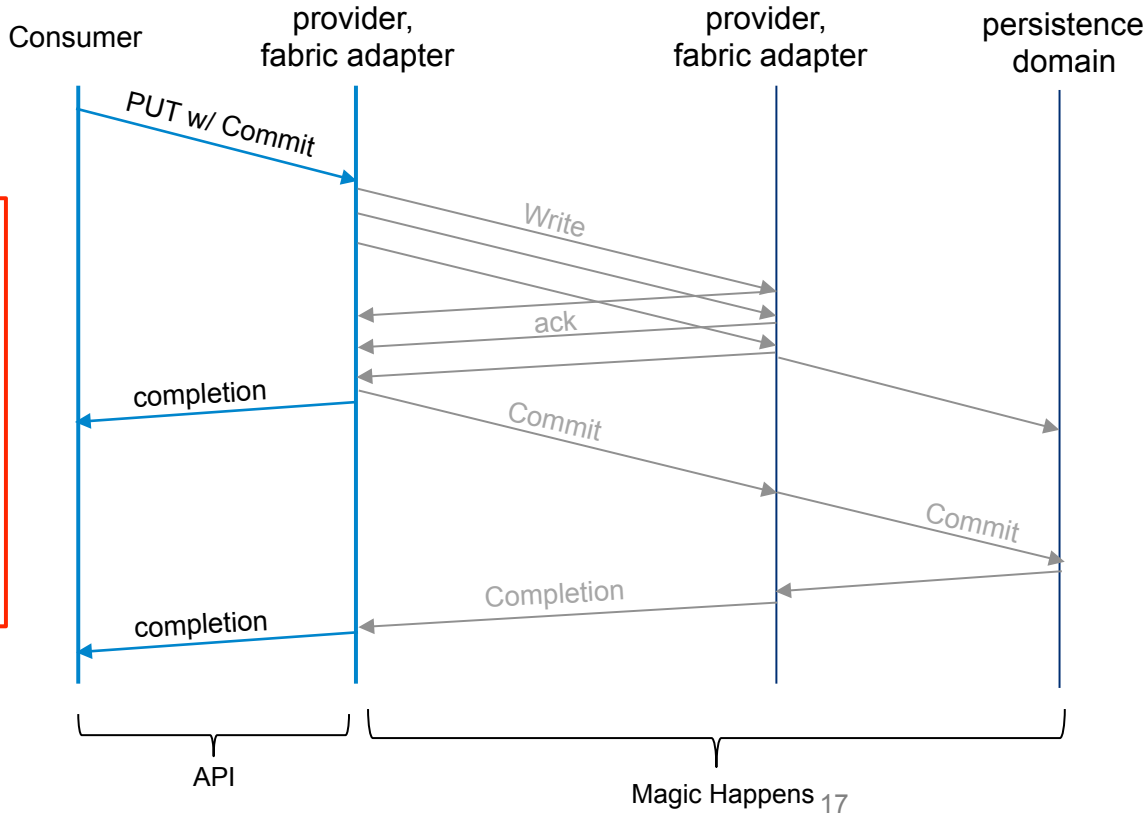
Applications Requirements – a starting point

- Ability to force flushes of local caches to remote persistent memory
- Perform non-cached writes to remote to persistent memory
- Identify the persistence characteristics of a remote memory device
- Distinguish between data which is globally valid and data which is remotely persistent

Basic API Features (example)

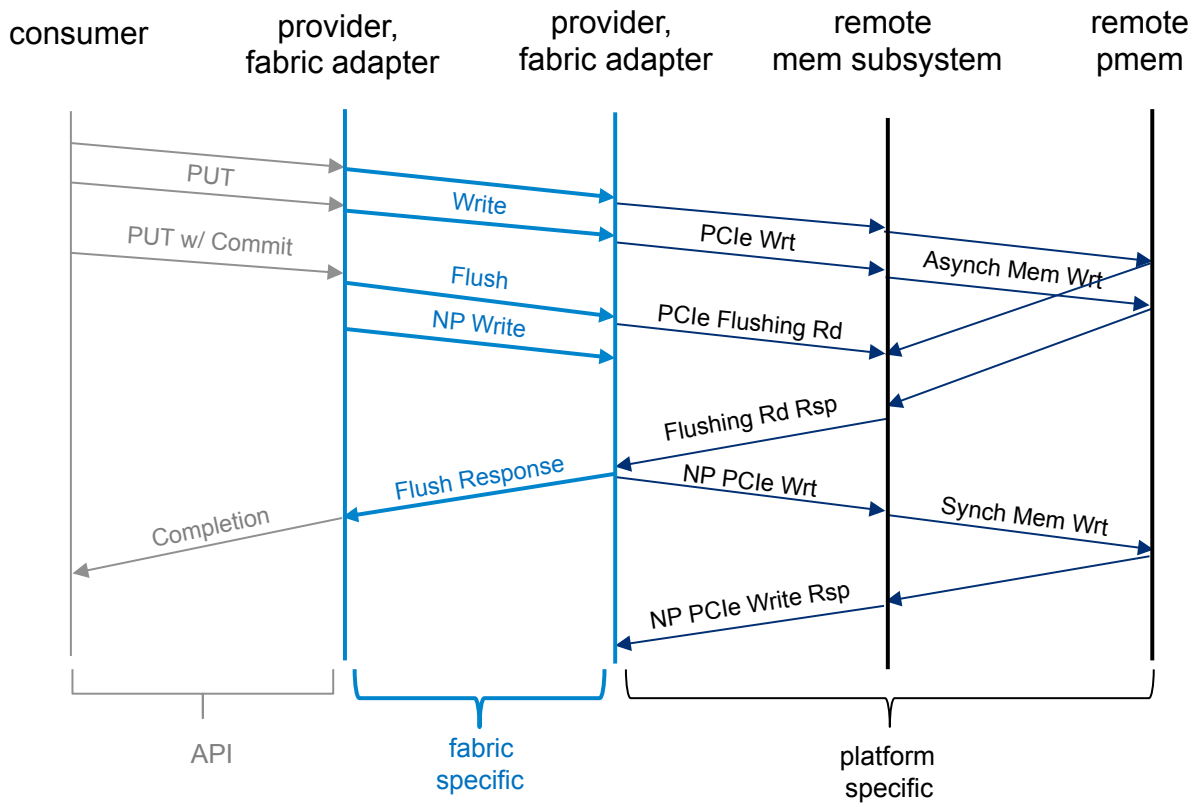
Plenty of work to be done to define the semantics of the API

Lots more to figure out how to implement the semantics over a given wire



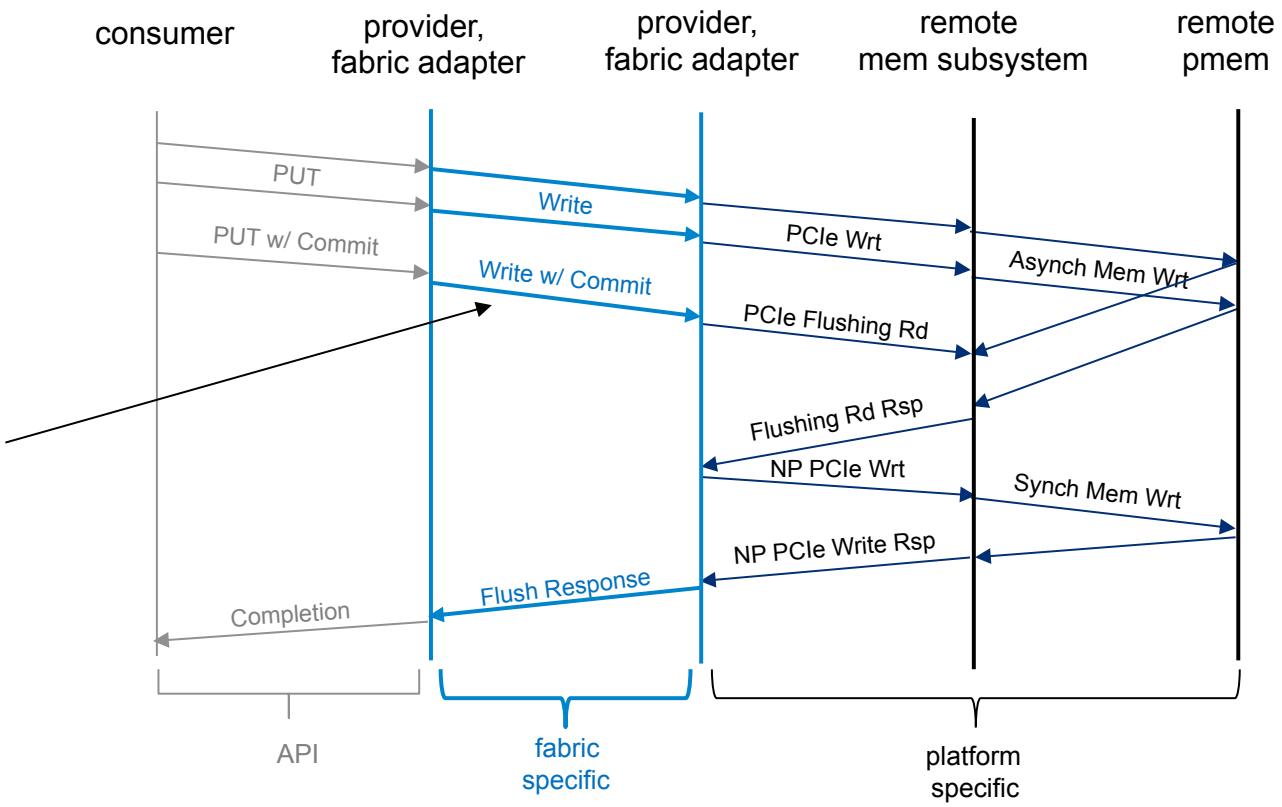
Fabric Implementations

Utilize FLUSH to force previous writes on the connection to be made durable before the flush is completed



Alternate Fabric Implementation

An alternative approach: combine the FLUSH and following WRITE into a WRITE w/ COMMIT



Fabric Implementations

Challenge to the industry is to extend the existing fabric protocols to provide:

- Efficient, low overhead PMEM semantics
- Simple provider fabric adapter implementation
- Flexible, extensible across common fabric types

Q & A, Discussion

- Rob Davis, Mellanox Technologies
- Chet Douglas, Intel
- Paul Grun, Cray, Inc
- Tom Talpey, Microsoft

Thanks!