SDC
STORAGE DEVELOPER CONFERENCE
SNIA ■ SANTA CLARA, 2016

Application Access to Persistent Memory – The State of the Nation(s)!

Stephen Bates, Paul Grun,
Tom Talpey, Doug Voigt

Microsemi, Cray, Microsoft, HPE

The Suspects

Stephen Bates
Microsemi

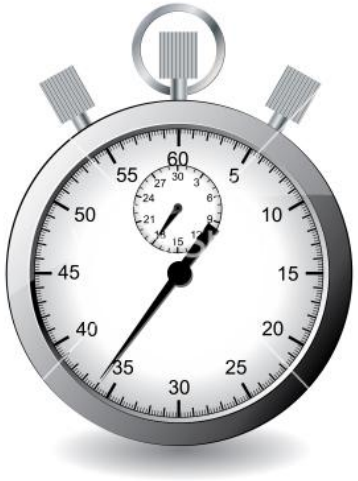
Paul Grun
Cray

Tom Talpey
Microsoft

Doug Voigt
HPE

We've Come a Long Way, Baby!

Persistent Memory (PM)

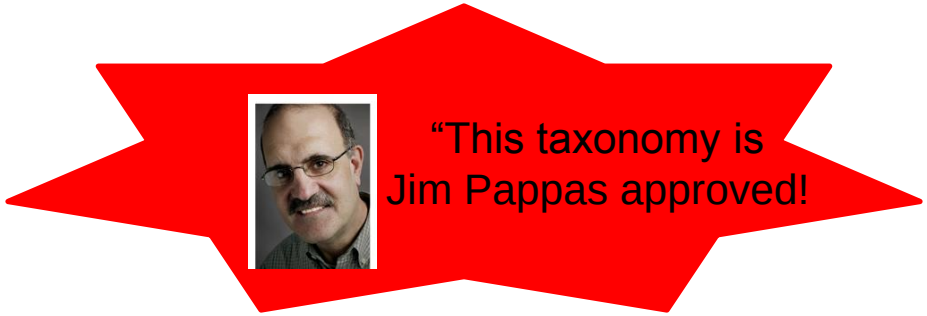


Low Latency

Memory Semantics

Storage Features

Taxonomy



- **NVM** – Non-Volatile Memory. All types, including those that are not byte-addressable
- **PM** – Persistent Memory. Sometimes PMEM is used but we use PM in this talk
- **NVMe** – NVM Express. A block protocol to run over PCIe, RDMA or Fibre Channel. A SATA/SAS replacement.
- **NVMP** – NVM Programming Model. Application-visible NVM behavior
- **NVMf** – NVMe over Fabrics. NVMe extended over fabrics

Low Latency

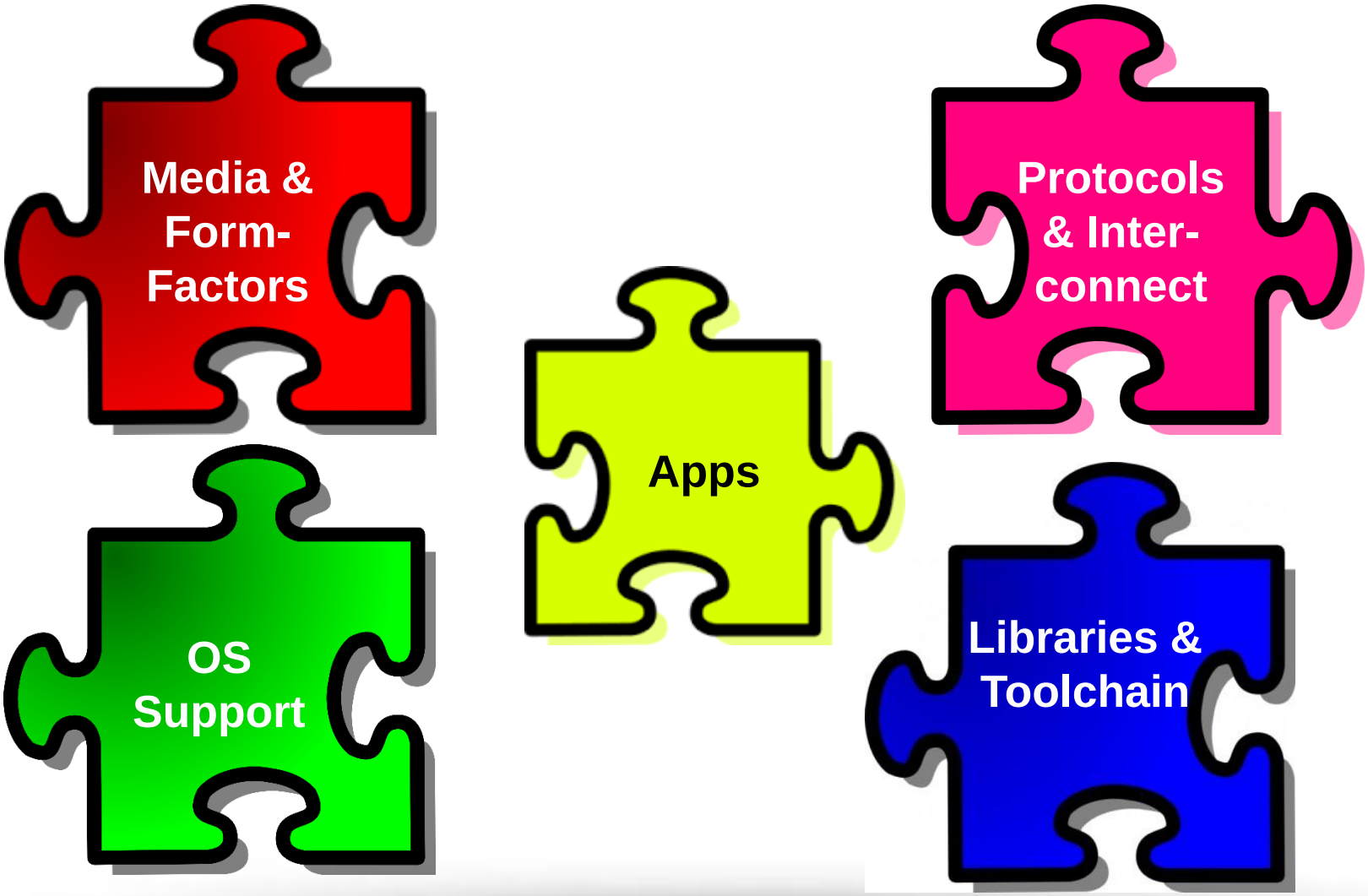
	Latency	Relative
L1 Cache Read	0.5ns	1
L2 Cache Read	7ns	14
DRAM Read	100ns	200
The PM Opportunity		
NVMe DRAM SSD Read	10us	20,000
NVMe NAND SSD Read	150us	300,000
SAS HDD	500us	1,000,000

NVMe SSDs are (relatively) high latency!

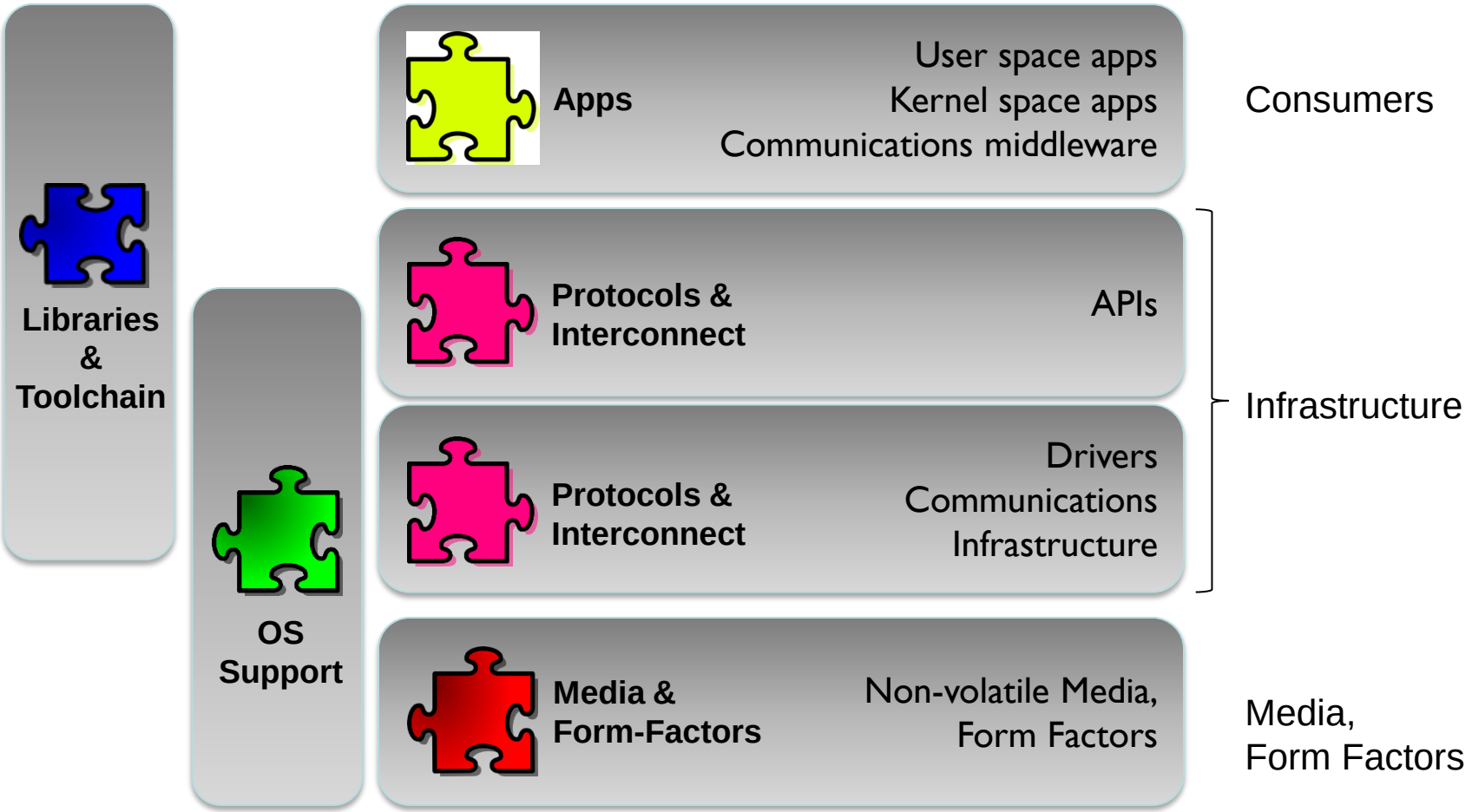
PM provides persistence at memory-like speeds and semantics

Where Are We?

What is Needed?

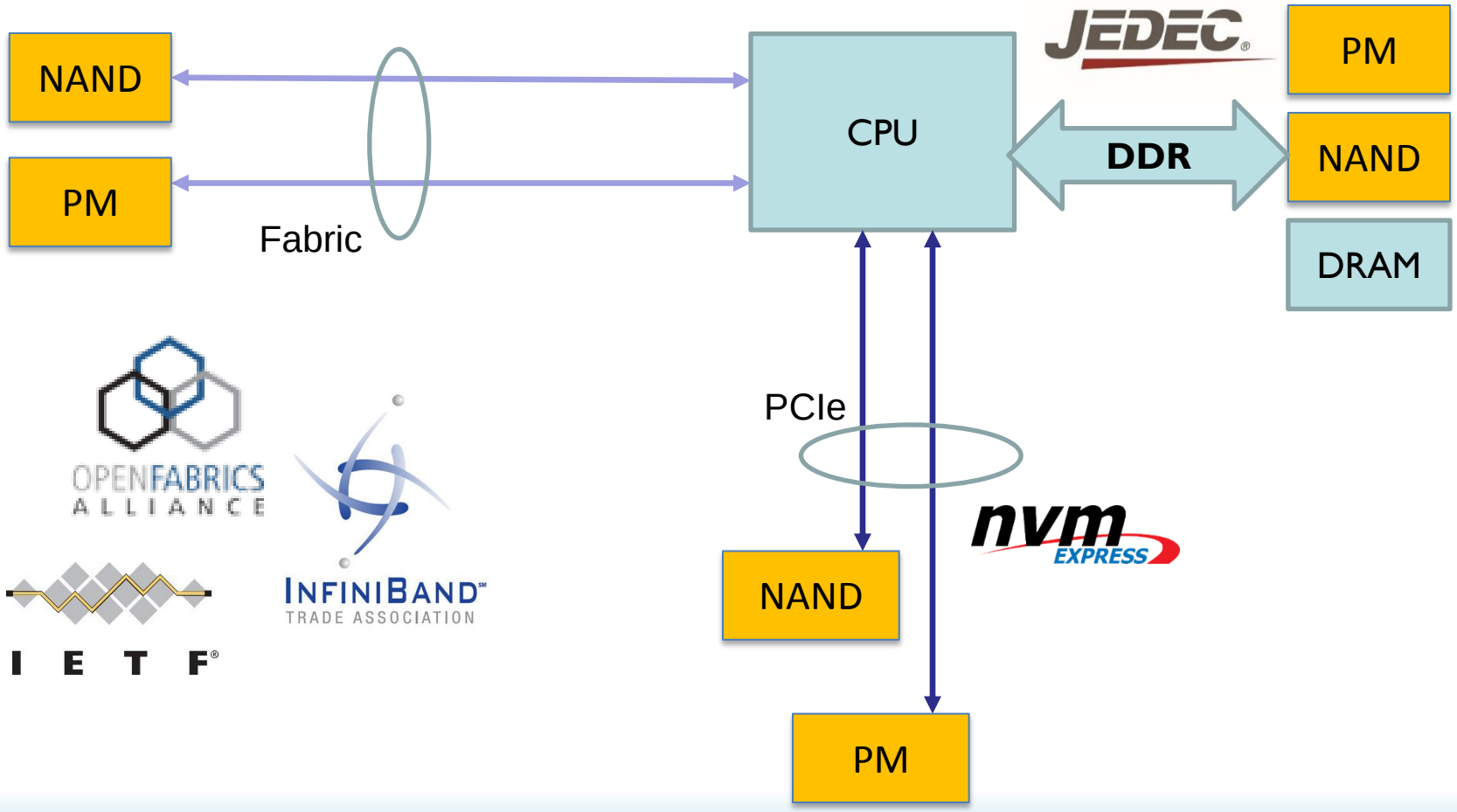


Lots of Moving Parts

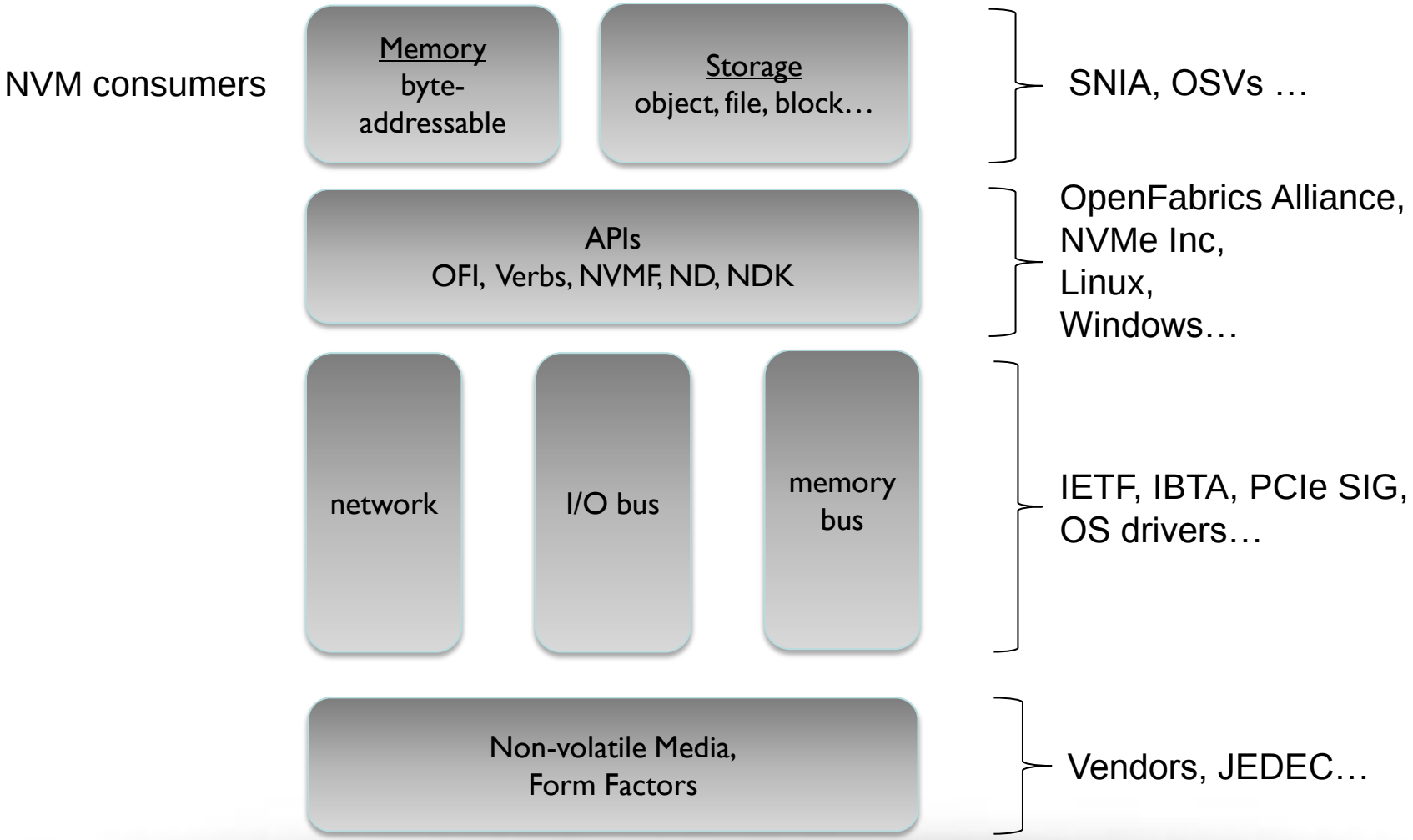


Where does PM sit?

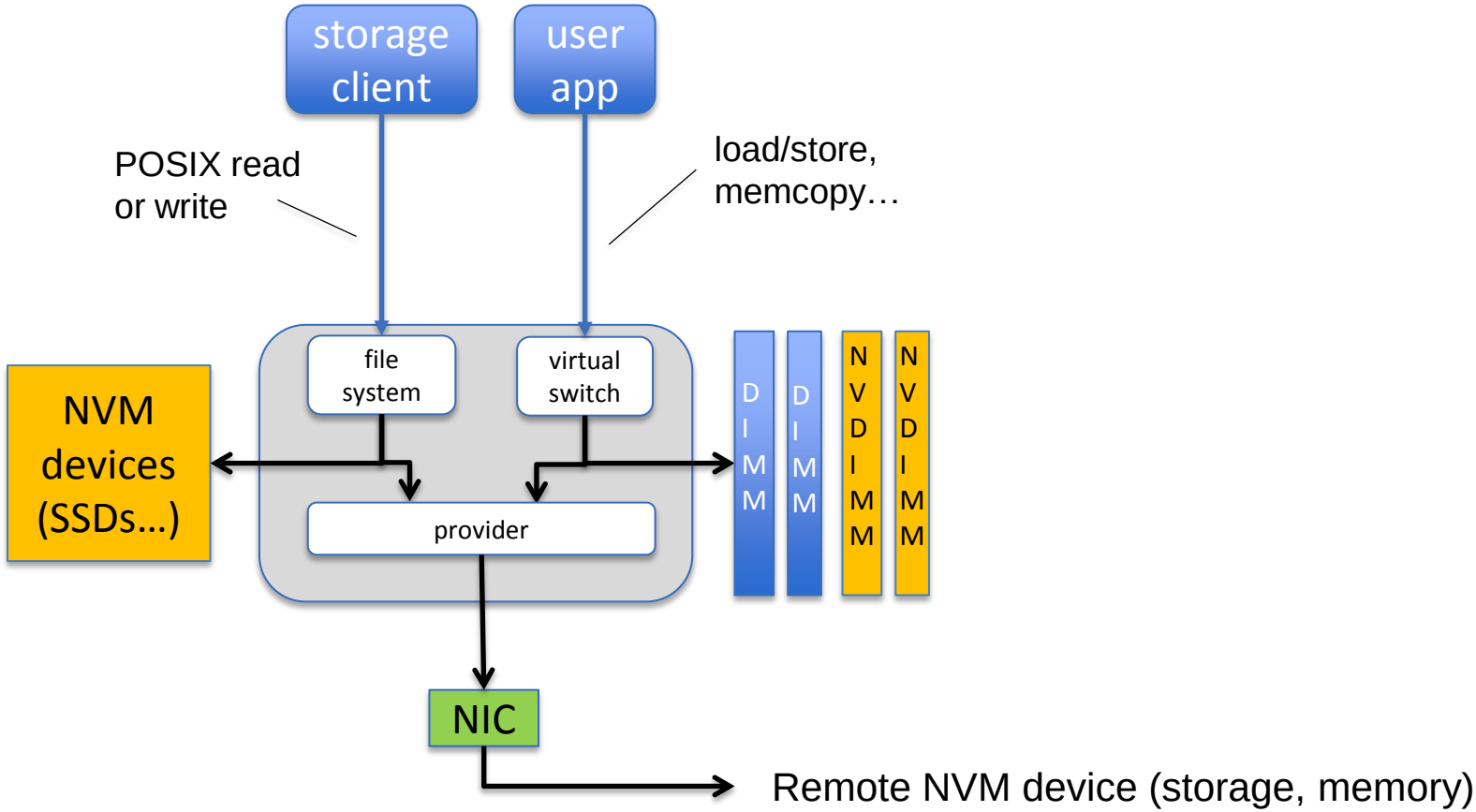
(Answer – anywhere it wants to)



Rationalizing the Problem Space

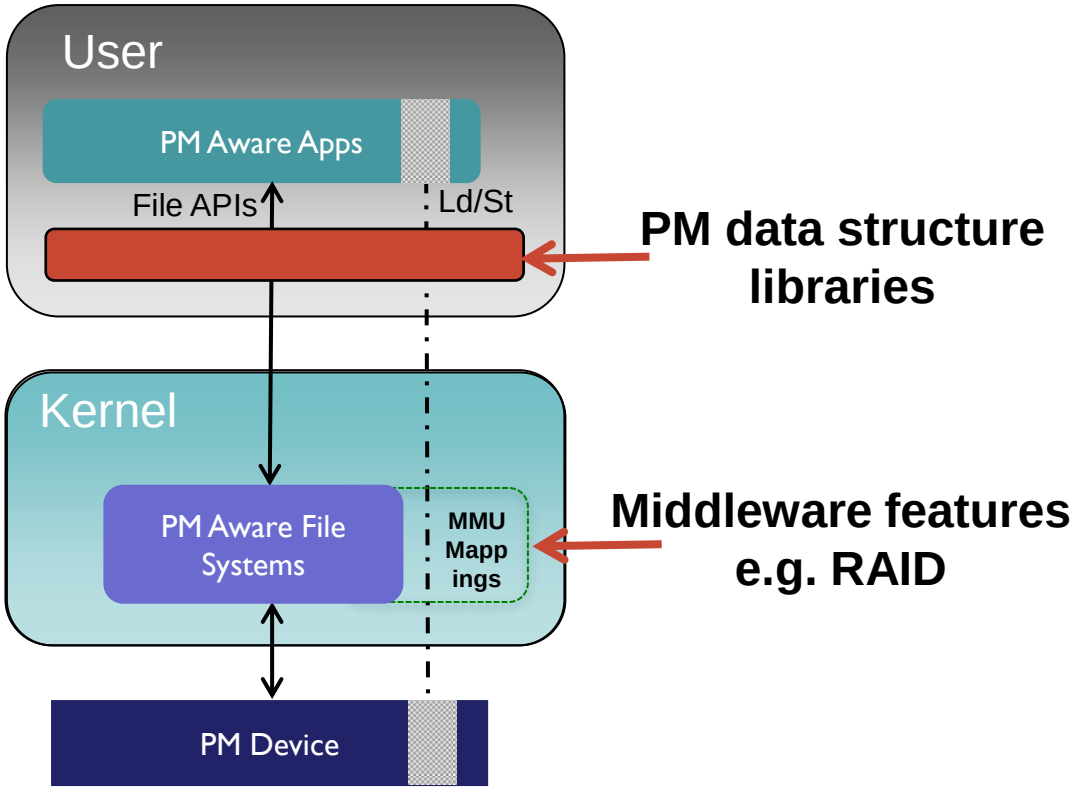


Start with Consumers of NVM Services

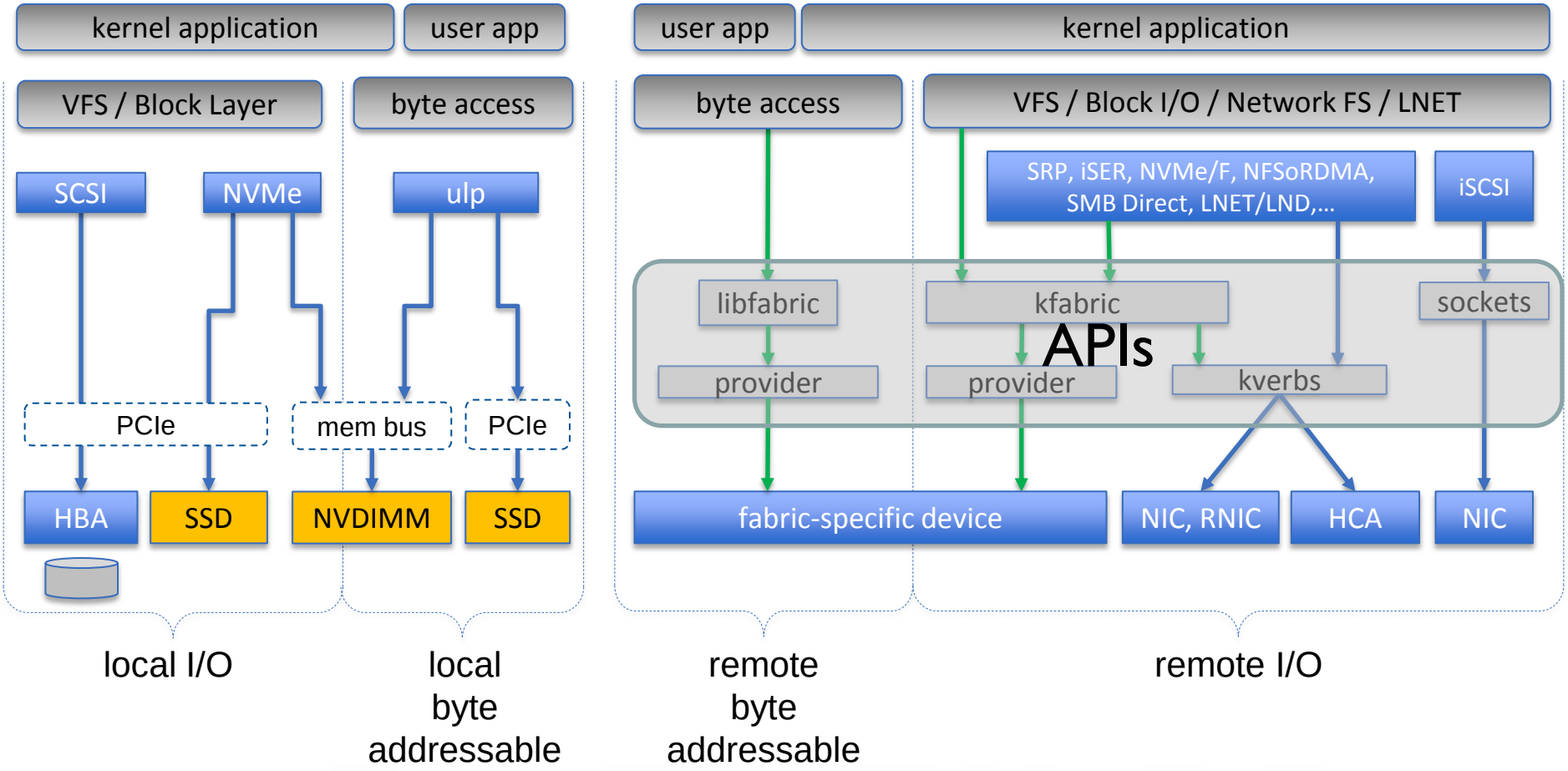


Application View

- SNIA NVMP
  - Describe application visible behaviors
  - APIs align with OSs
- PM File System Actions
  - Map – expose PM directly to applications
  - Optimized Flush – make data persistent



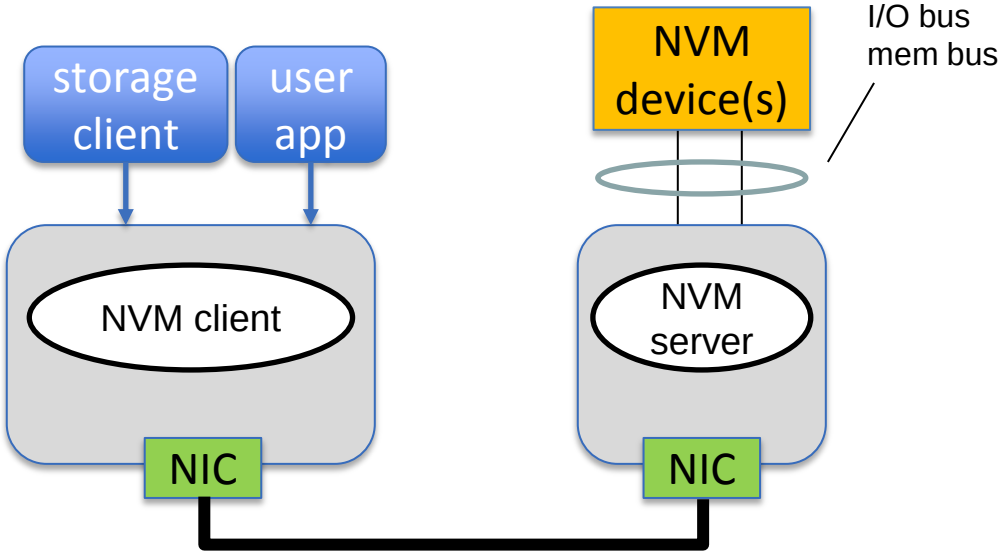
Possible Stack for NVM Access



14

Optimizing Fabrics for NVM

- Add persistence semantics to RDMA protocols
  - Register persistent memory regions
  - Completion semantics to ensure persistence, consistency
  - Client control of persistence
- Solve the “write-hole” problem
- Lots of Initiatives underway!

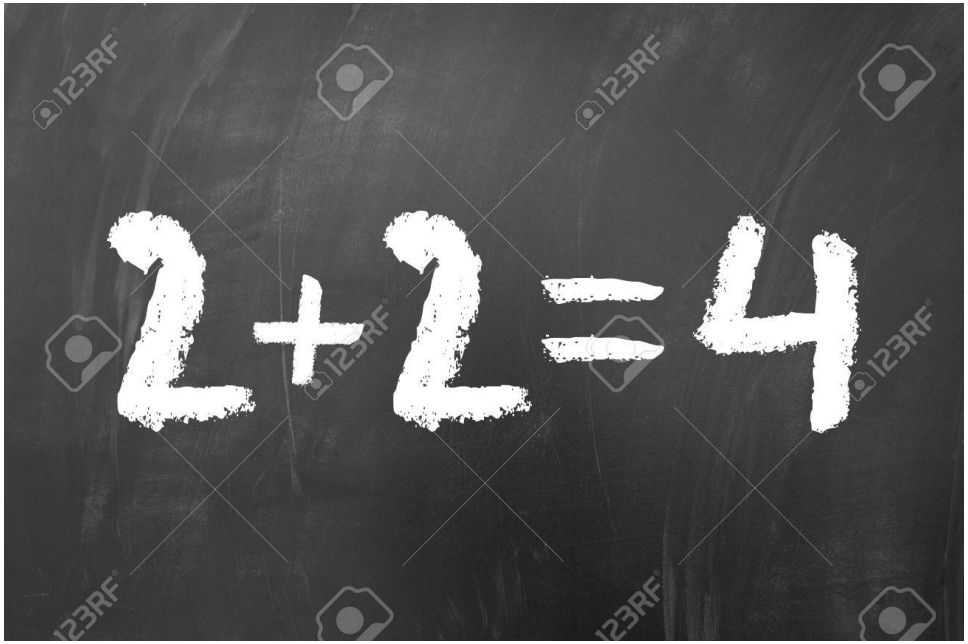


Can we make this work for NVDIMMs?
NVMe SSDs with CMBs?

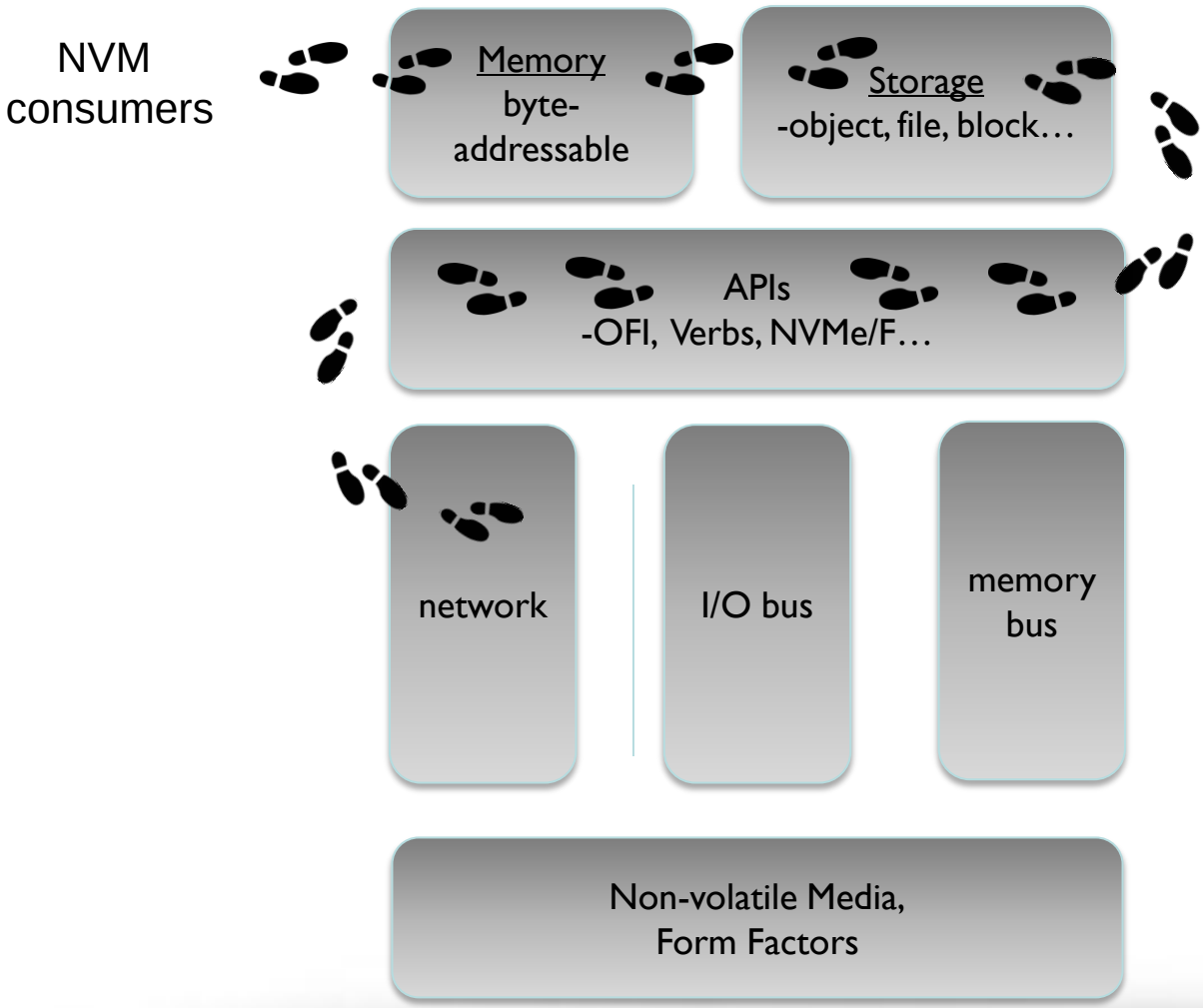
Simple Math


```
        NVMe
+       RDMA
    -------
    AWESOME
```

```
          PM
+       RDMA
    --------
    AWESOME^2
```

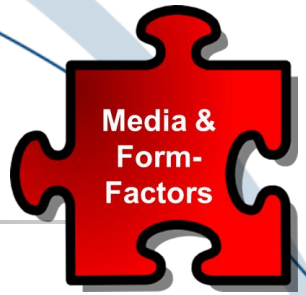


…continuing down the stack



- NVMe over Fabrics – Present an NVMe block device to a client over RDMA or Fibre Channel
- NVMe Controller Memory Buffers – Standardize (persistent) PCIe memory on NVMe devices. NVDIMM-N on PCIe bus?
- LightNVM – A low-level SSD interface that is more aligned to the underlying media (NAND)

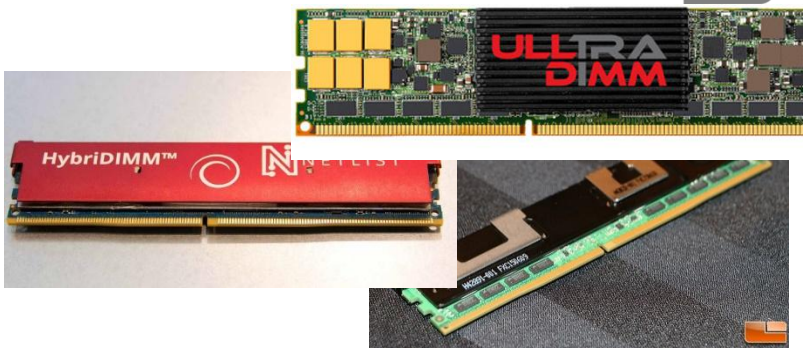
Media



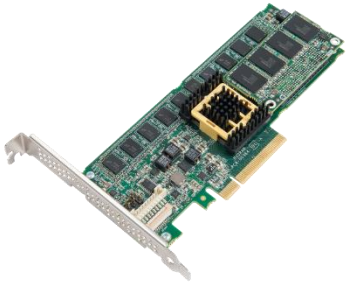
	Category	Vendors	Comments	
NVM	**DRAM Drop-In**	Everspin Micron Toshiba/SK Hynix	DRAM like latency Super-Cap Replacement Not for bulk storage **Memory Interface**	PM
NVM	**Storage Class Memory**	Micron-Intel SanDisk Toshiba Crossbar Nantero	Faster than NAND, Cheaper/Slower than DRAM Byte Addressable **Block and Memory Interfaces**	PM
NVM	**NAND**	Micron Toshiba SanDisk SK Hynix Samsung	Lowest cost Slow (for NVM) Not byte addressable cheap and plentiful **Block Interface**	NOT PM

PM Form Factors

NVDIMM-N



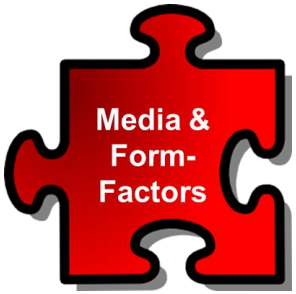
NVDIMM-P



Not-NAND NVMe

NAND NVMe

PM Form Factors



Form-Factor	Media	Latency	Memory Semantics	Storage Features
NVDIMM-N	DRAM/ MRAM			
NVDIMM-P	NAND/ PM			
Non-NAND NVMe	DRAM/ PM			
NAND NVMe	NAND			

Form factors impact Features
(No DMA engines on a DIMM!)

PM Scenarios

- PM region as a block device (a la persistent ram disk).
- Filesystems support: direct access to the memory (e.g DAX), PM aware FS (e.g. m1fs).
- You can put your files, databases etc. on top.
- **Remember we are crawling right now!**
- Soon: Shared persistent memory

Libraries and Toolchains

EASY

```
int main(int argc, char *argv)
{
  printf("Hello, PMEM World!\n");
  return 0;
}
```

Vs

HARD

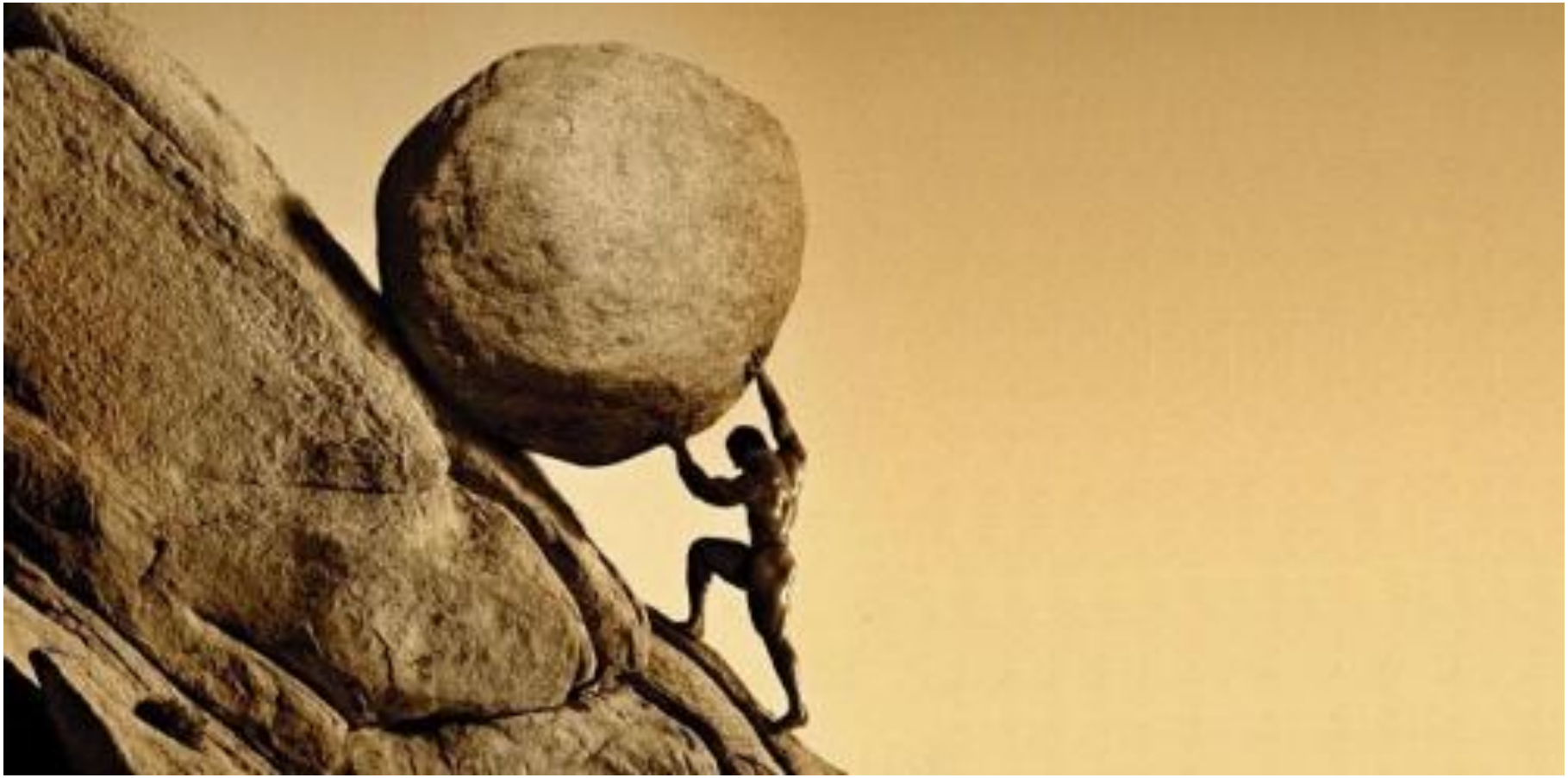
```
section .text
global _start
_start:
        mov edx,len
        mov ecx,msg
        mov ebx,1
        mov eax,4
        int 0x80

        mov eax,1
        int 0x80

section .data
msg db 'Hello, PMEM World!',0xa
len equ $ - msg
```

Make it easy for applications to utilize PM, regardless of OS and ARCH!

Call to Arms



Call to Arms

- **Libraries and Toolchains**: NVML for non-x86, integration into glibc/gcc etc.
- **Media & Form Factors**: Production PM, appropriate PM form factors.
- **Protocols and Interconnect**: Enhancements to NVMe and RDMA, PM over Fabrics, standardization of memory channels.
- **OS Support**: ISA updates, DAX devices, Other OS, new OSes?

Conclusions

- We are almost walking! Help out if you can
- If you want sub 10us access to persistent data then PM may be for you
- The CPU vendors have a lot of say in the interconnect but some open options exist too
- Toolchains, libraries and OSes are adapting
- New applications will complete the jigsaw and lead to revenue