Persistent Memory and the NVM Programming Model

Andy Rudoff
Data Center Software
Intel Corporation
andy.rudoff@intel.com

Questions

- Why now?
  - Basic programming model is decades old!
- What changes?
  - Incremental changes vs major disruptions
- What does this mean to FreeBSD?
  - Proposed changes and open questions

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Questions

- Why now?
  - Basic programming model is decades old!
- What changes?
  - Incremental changes vs major disruptions
- What does this mean to FreeBSD?
  - Proposed changes and open questions

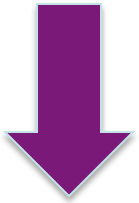
2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Two Movements Afoot

- Why now?
  - Basic programming model is decades old!

Block Mode Innovations

- Atomics
- Access hints
- NVM-oriented operations



Emerging NVM Technologies

- Performance
- Performance
- Perf… okay, Cost

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

SNIA* NVM Programming TWG

- Members
  - EMC, Fujitsu, Fusion-io, HP, HGST, Inphi, Intel, Intuitive Cognition Consulting, LSI, Microsoft, NetApp, PMC-Sierra, Qlogic, Red Hat, Samsung, Seagate, Sony, Symantec, Viking, Virident, VMware
  - Calypso Systems, Cisco, Contour Asset Management, Dell, FalconStor, Hitachi, Huawei, IBM, IDT, Marvell, Micron, NEC, OCZ, Oracle, SanDisk, Tata Consultancy Services, Toshiba

*** Storage Networking Industry Association: www.snia.org**

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

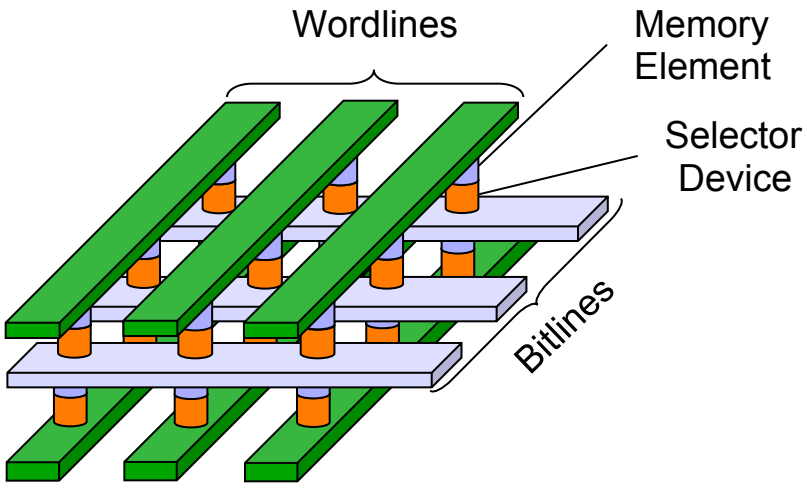
Questions

- Why now?
  - Basic programming model is decades old!
- What changes?
  - Incremental changes vs major disruptions
- What does this mean to FreeBSD?
  - Proposed changes and open questions

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Next Generation Scalable NVM

Scalable Resistive Memory Element



Cross Point Array in Backend Layers ~$4l^2$ Cell

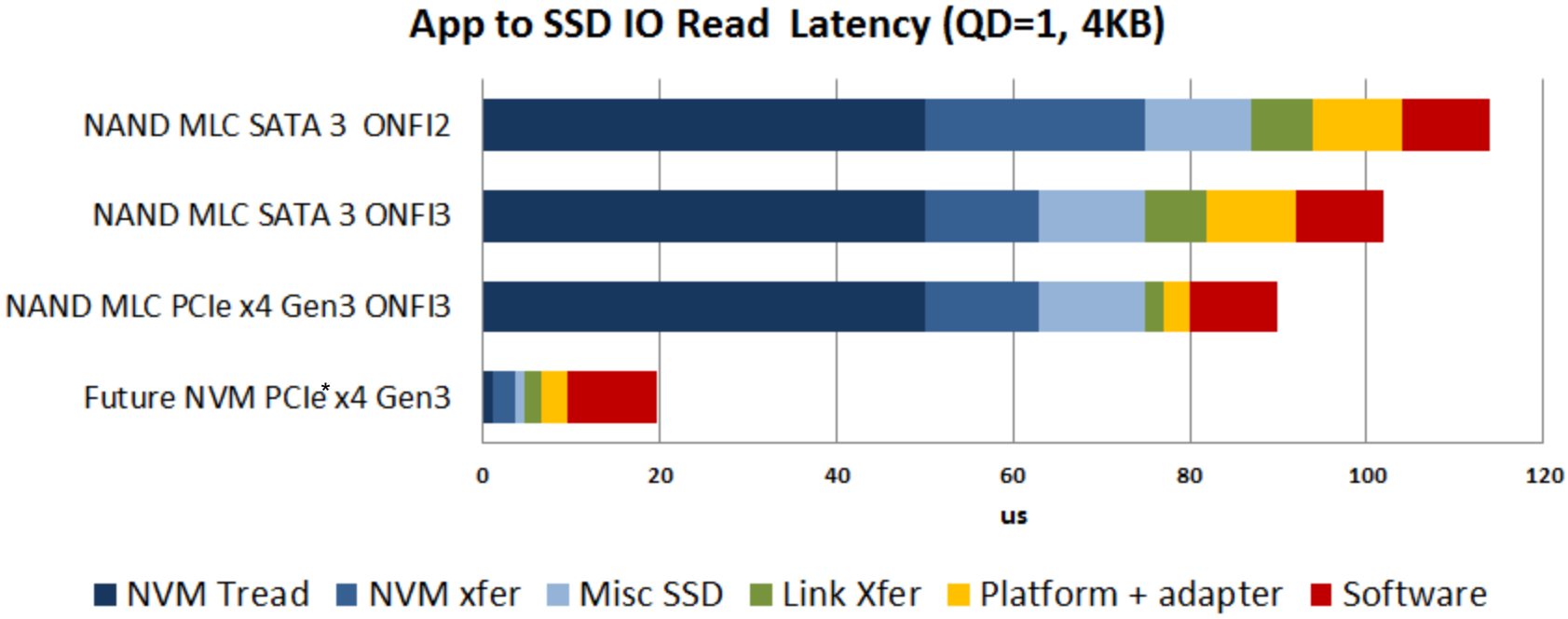
Resistive RAM NVM Options

Family	Defining Switching Characteristics
Phase Change Memory	Energy (heat) converts material between crystalline (conductive) and amorphous (resistive) phases
Magnetic Tunnel Junction (MTJ)	Switching of magnetic resistive layer by spin-polarized electrons
Electrochemical Cells (ECM)	Formation / dissolution of "nano-bridge" by electrochemistry
Binary Oxide Filament Cells	Reversible filament formation by Oxidation-Reduction
Interfacial Switching	Oxygen vacancy drift diffusion induced barrier modulation

Many candidate next generation NVM technologies. Offer ~ 1000x speed-up over NAND, closer to DRAM

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

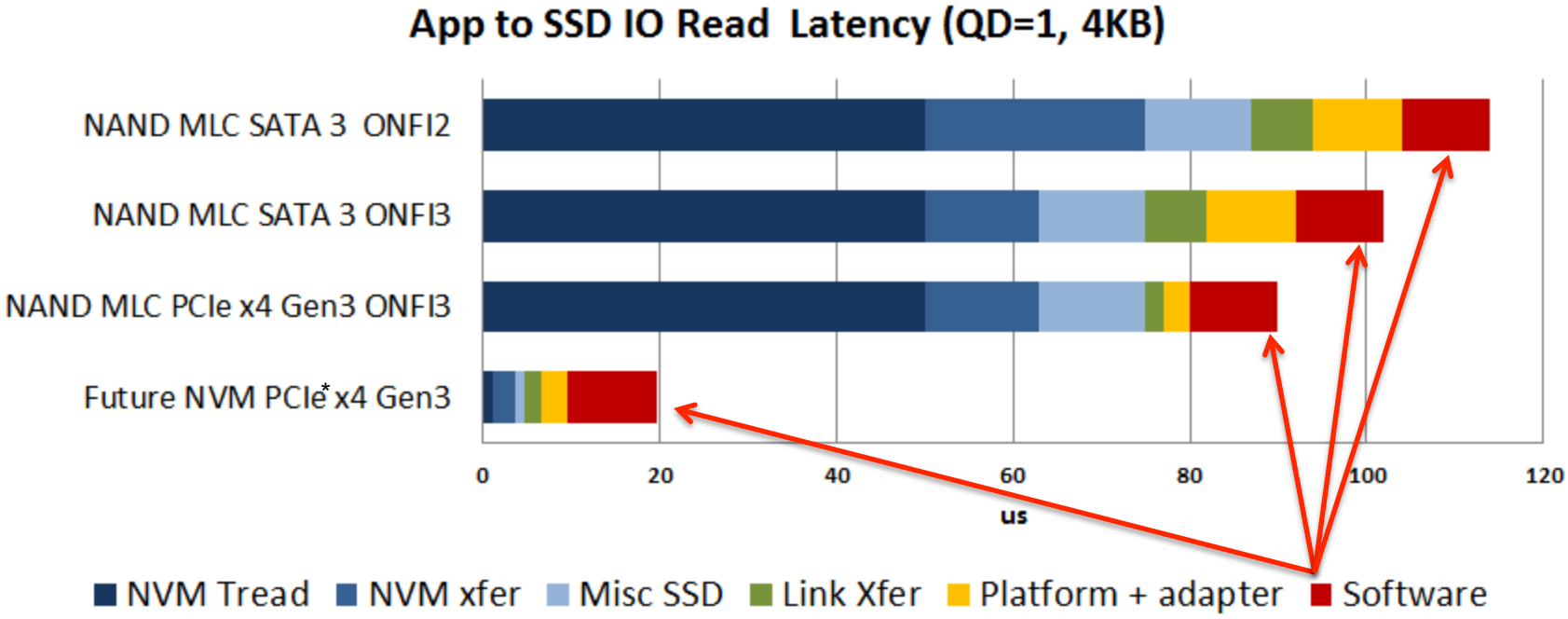
Exploiting Next Generation NVM



- With Next Generation NVM, the NVM is no longer the bottleneck
  - Need optimized platform storage interconnect
  - Need optimized software storage access methods

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

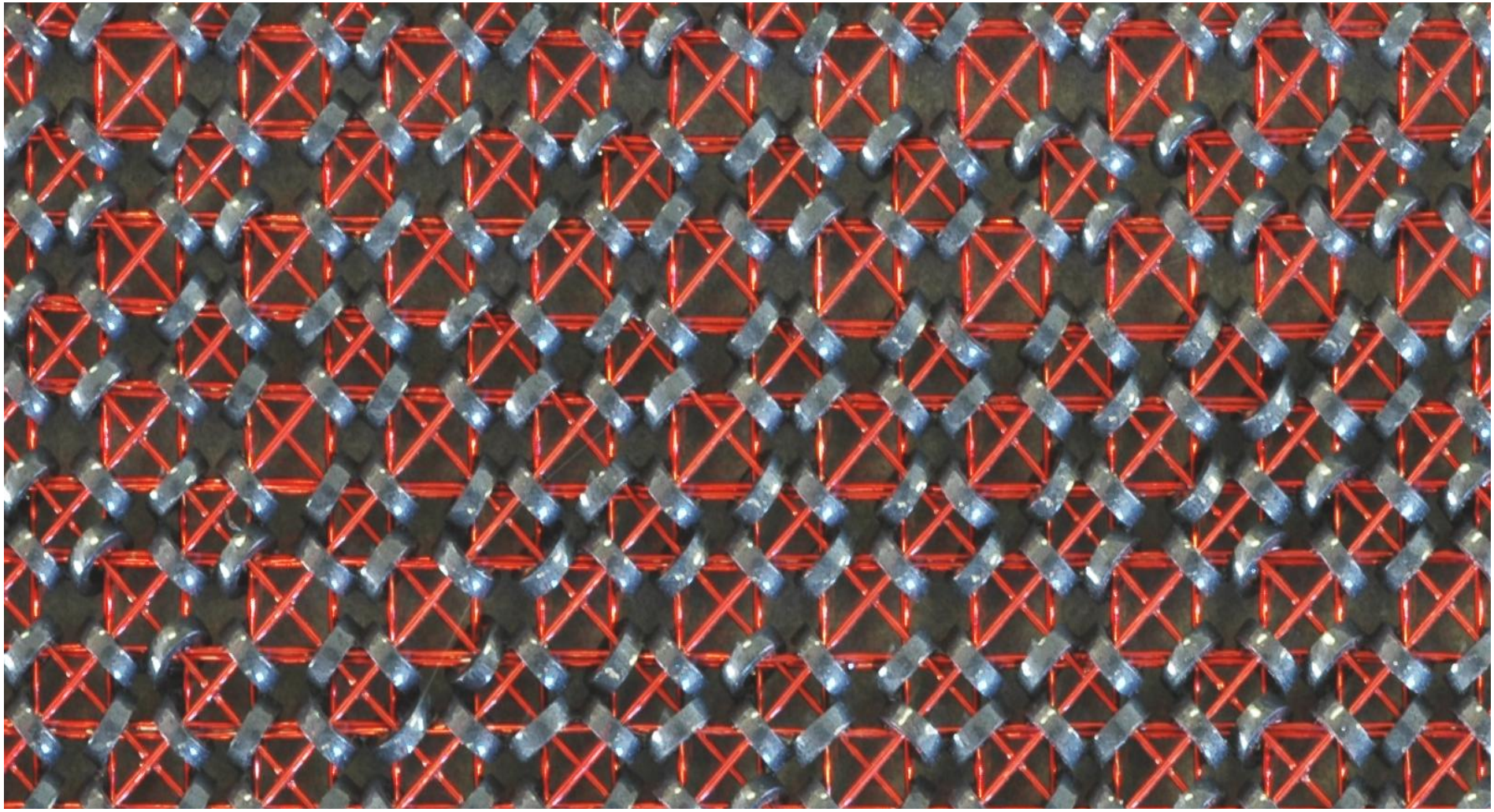
Exploiting Next Generation NVM



- With Next Generation NVM, the NVM is no longer the bottleneck
  - Need optimized platform storage interconnect
  - Need optimized software storage access methods

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Persistent Memory



2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Persistent Memory Definition

- Byte-addressable
  - As far as the programmer is concerned
- Load/Store access
  - Not demand-paged
- Memory-like performance
  - **Would reasonably stall a CPU load waiting for PM**
- Probably DMA-able
  - Including RDMA

- For modeling, think: Battery-backed DRAM

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

PMem in This Presentation…

- Is **not** tablet-like memory for entire system
  - "Transparent Persistent Memory" is another topic
- Is **not** NAND Flash
  - At least not directly
    - perhaps with aggressive caching
- Is **not** block-oriented

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Persistent Memory Attributes

- PM does not
  - Surprise the program with unexpected latencies
    - No major page faults
  - Kick other things out of memory
  - Use the page cache unexpectedly
- PM stores aren't durable until data is flushed
  - Is this a new, inconvenient attribute of PM?
  - Or is this something that's been around for decades?
- PM may not always stay at the same address
  - Physically
  - Virtually

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Types of Persistent Memory

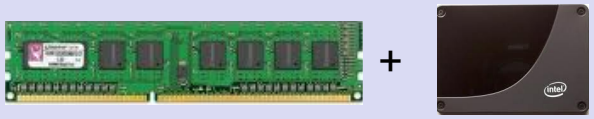
- Battery-backed DRAM
  - Practical or not, a good model for software design
- DRAM saved on power failure
  - DRAM speeds at run-time
  - Expense = DRAM + additional logic
- NVM with significant caching
  - Reliance on cache means assumptions on workloads
- Next generation NVM
  - Bit and pieces of information available
  - Still quite a bit unknown/emerging here

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Existing Memory use cases

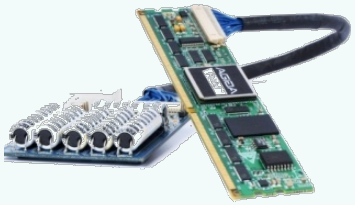
Volatile — **Typical** Use Case

Volatile System memory
- DRAM DIMMs
- High Throughput, High Cost



SSD Caching
- DB in HDD/SAN cached by SSD/DRAM

Persistent — **Emerging** Use Case



NVDIMM
- Raid Cache
- Persistent RamDisk
- Checkpointing

Copy to Flash
- Sections of DRAM used for Persistent Write Back Cache (WBC)
- Transfer WBC to Flash (power loss)

NVM used as Memory
- Extended App Virtual Memory
- Auto Commit Mem (persistent)

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

The Value of Persistent Memory

- Data sets addressable with no DRAM footprint
  - At least, up to application if data copied to DRAM
- Typically DMA (and RDMA) to PM works as expected
  - RDMA directly to persistence – no buffer copy required!
- The “Warm Cache” effect
  - No time spend loading up memory
- Byte addressable
- Direct user-mode access
  - No kernel code in data path

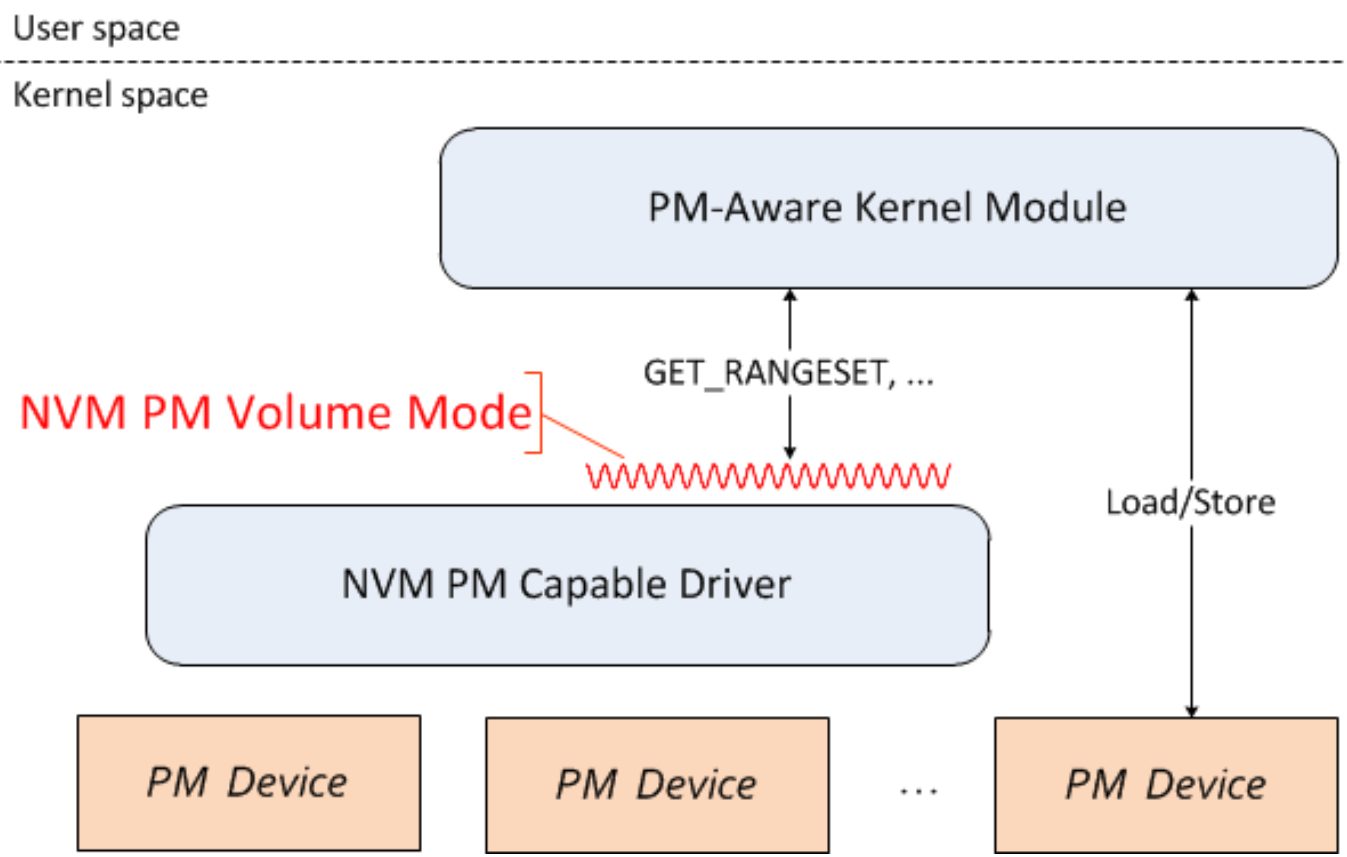
2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Two Persistent Memory Programming Models (two *so far*…)

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

NVM.PM.VOLUME

- NVM Volumes
  - PM Capable
  - A list of physical ranges of NVM
- Operations
  - GET_RANGESET
  - …



2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

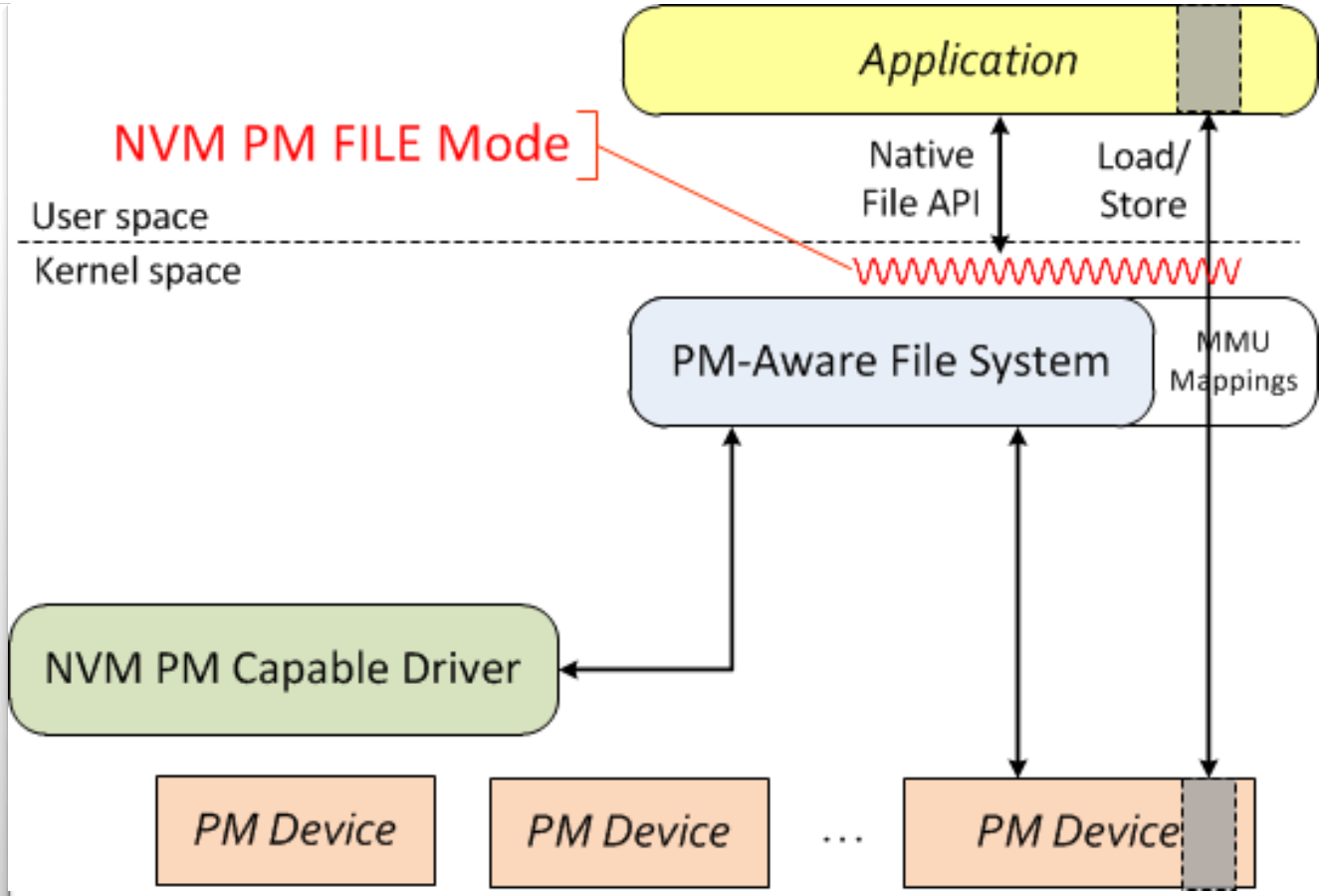
Who Uses NVM.PM.VOLUME?

- Kernel modules
- File systems
  - Maybe to expose PM
  - Maybe just to use it internally
- Memory management
  - Example: Multi-tiered page cache
- Other storage stack components
  - RAID
  - Caches
  - Clustered I/O
- Future NVM Programming models we haven't thought of yet

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

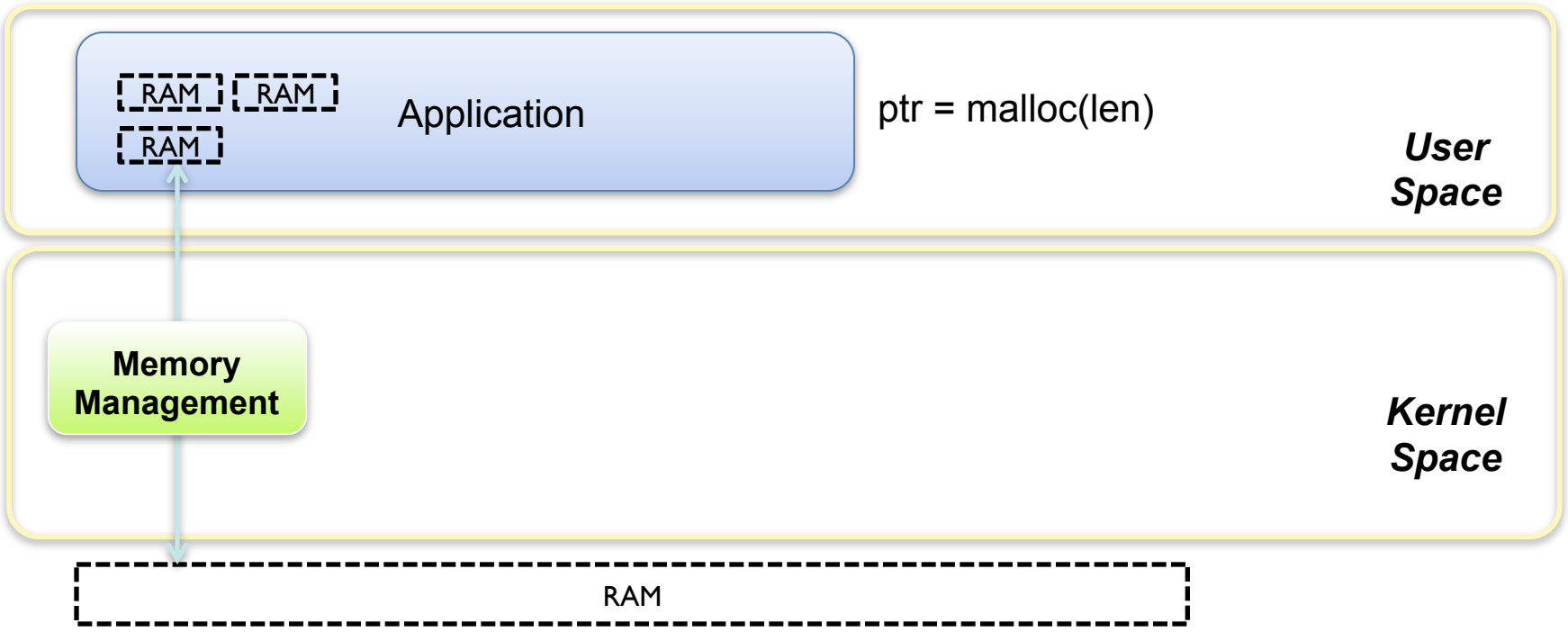
NVM.PM.FILE

- NVM Files
  - PM Capable
  - Native file APIs and management
- Operations
  - Native open/close read/ write
  - NVM.PM.FILE.MAP
  - …



2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

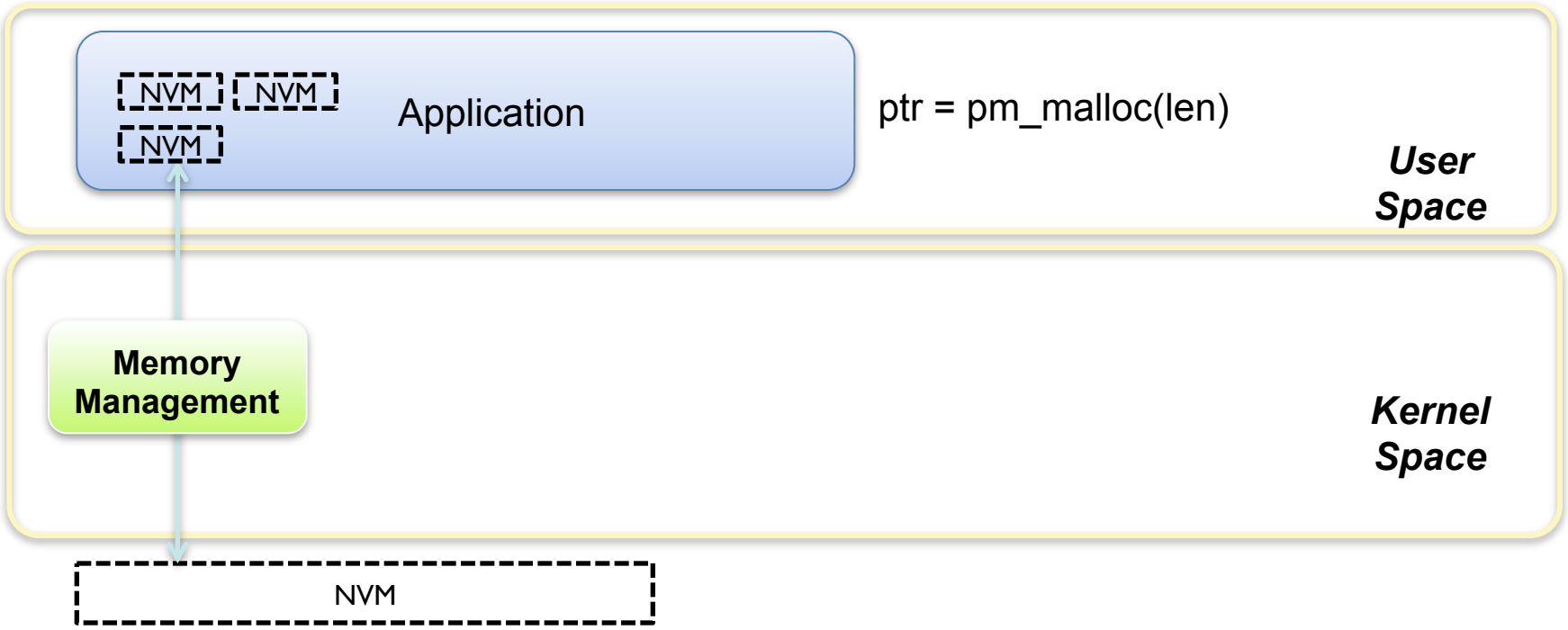
Application Memory Allocation



- Well-worn interface, around for decades
- Memory is gone when application exits
  - Or machine goes down

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Application NVM Allocation



- Simple, familiar interface, *but then what?*
  - Persistent, so apps want to “attach” to regions
  - Need to manage permissions for regions
  - Need to resize, remove, …, **backup** the data

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Who Uses NVM.PM.FILE?

- Applications
  - Persistent data sets, requiring addressability without impacting DRAM footprint
  - Persistent caches
- Usages that must reconnect with blobs of persistence
  - Naming
  - Permissions
- Potentially kernel modules requiring some of the above features

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

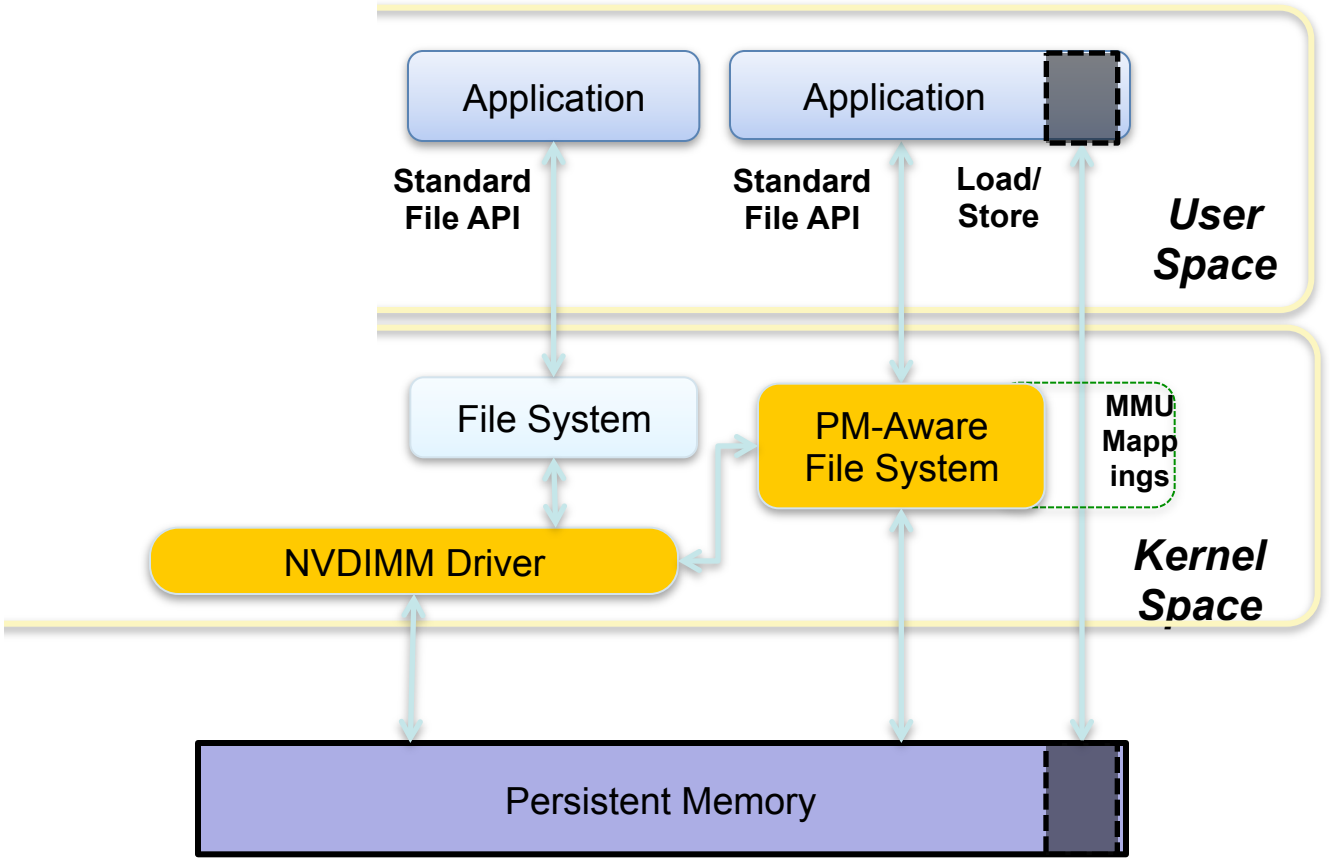
Questions

- Why now?
  - Basic programming model is decades old!
- What changes?
  - Incremental changes vs major disruptions
- What does this mean to FreeBSD?
  - Proposed changes and open questions

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

The Kernel Components

New Components



2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

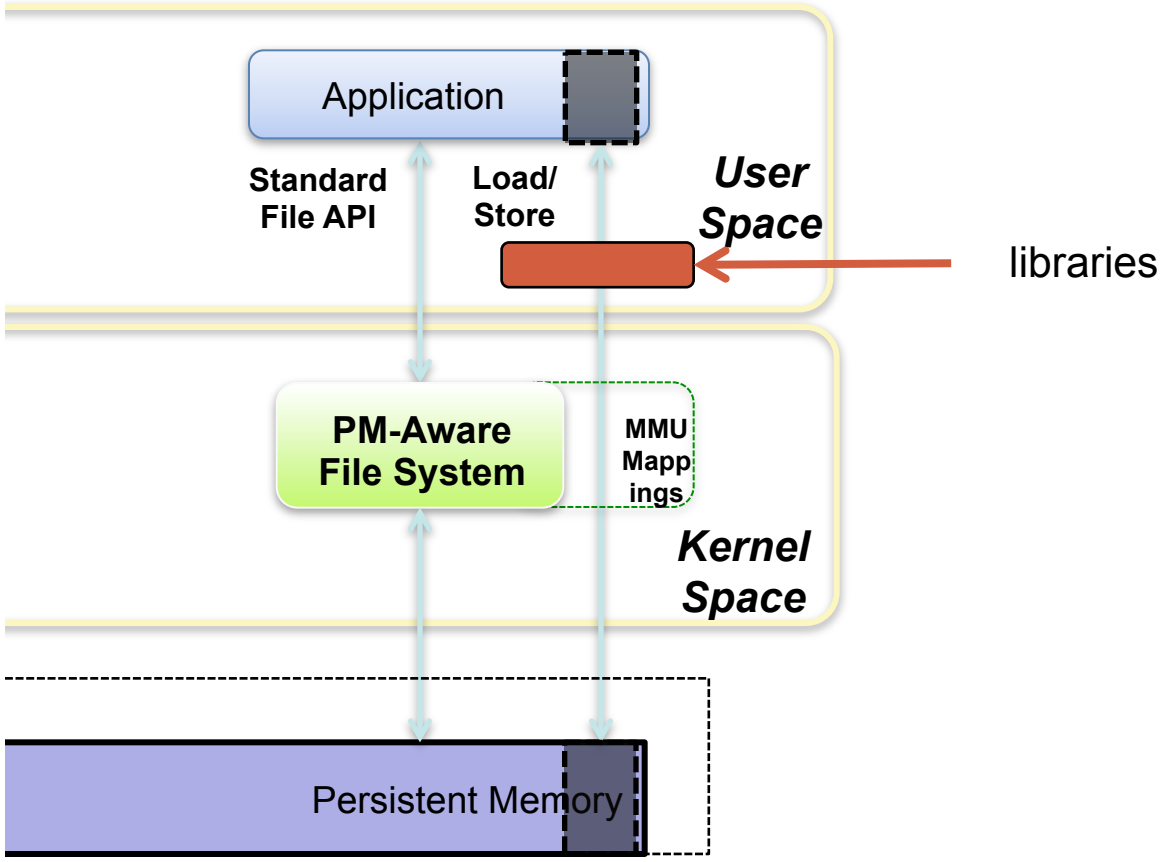
Memory-Mapped Data Structures

- Obtain memory-mapped access with mmap()
- Facility has been around for decades
  - In pretty much all modern operating systems
- Typically demand-paged
  - UNIX also uses it for hardware access (e.g., graphics)
- Usually two types:
  - Shared
  - Copy-on-write (“Private”)

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Building on the Basic PM Model

- NVM.PM.FILE programming model "surfaces" PM to application
- Still somewhat raw at that point
- **Build on it with additional libraries**
- Eventually turn to language extensions



2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Creating Resilient Data Structures

- Once PMem is mapped into application, a malloc-like interface is required
- Traditional:
  - ptr = pmemalloc(pmem_pool, len)
- New and interesting issues:
  - What if program crashes before anything links to ptr?
    - Memory leak – **Persistent** memory leak
  - What if program is half-way through linking it in?
    - Transactions required? How complex?
  - How to test issues that didn't happen w/volatile memory?
    - Fault-injection testing
- Not just for malloc – true for all PM data structures

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

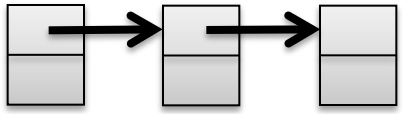
Full Example

- https://github.com/pmem/linux-examples
- libpmem
  - Helper functions for PM API
- libpmemalloc
  - Example of a PM-safe malloc library
- **Fault injection framework**
  - Proof-by-exhaustion of algorithm resilience
  - At least for simple examples…

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

What Might `malloc` Look Like?

- Start with a linked-list data structure:



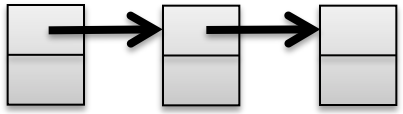
```
struct node {
        struct node *next_;
        int value;
};
```

- Traditional “malloc” is now done in steps:
  - Reserved the memory
  - Fill it in (prepare for use)
  - Link it in

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

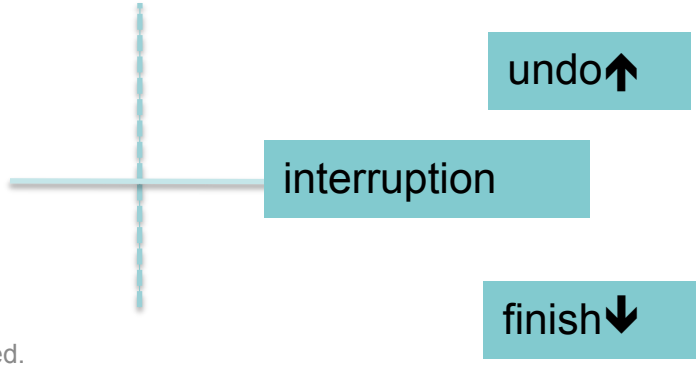
malloc Example...

- Start with a linked-list data structure:



```
struct node {
        struct node *next_;
        int value;
};
```

- Traditional “malloc” is now done in steps:
  - Reserved the memory
  - Fill it in (prepare for use)
  - Link it in



2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Summary

- NVM Programming Models are Evolving
  - Incremental block device features
  - Disruptive: Persistent Memory!
- The Industry is Aligning Around New Models
  - SNIA NVM Programming TWG
  - Emerging Research, Products
- Apps Can Leverage Persistent Memory
  - Using File System/DB Techniques
  - Using Libraries & Language Extensions

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Open Questions

- XIP Support in FreeBSD
  - Current thinking on approach
  - mmap/msync control point in file system?
  - Alternatively, core kernel PM support?
- ACPI support in FreeBSD for NVDIMMs?
  - Load driver based on ACPI device

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

For More…

- SNIA NVM Programming TWG
  - http://snia.org/forums/sssi/nvmp
- Linux Pmem Examples:
  - https://github.com/pmem/linux-examples
- Linux PMFS:
  - https://github.com/linux-pmfs
- andy.rudoff@intel.com

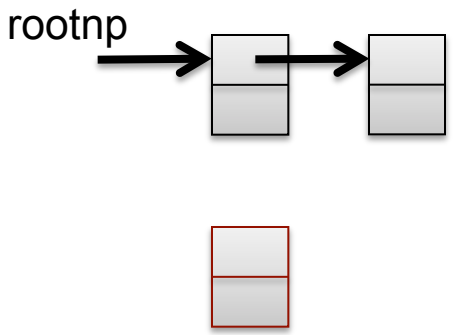
2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Some Additional Details

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Breaking `malloc` Into Steps

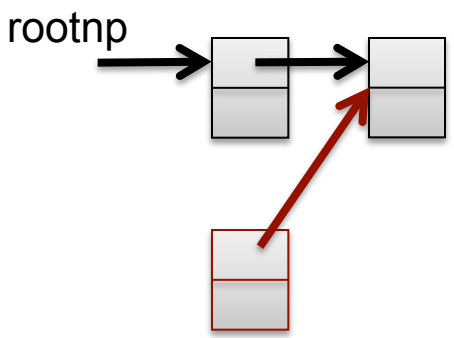
```
np = pmemalloc_reserve(sizeof(*np));

/* link it in at the beginning of the list */
np->next = rootnp;
np->value = value;
pmemalloc_onactive(np, &rootnp, np);


pmemalloc_activate(np);
```



2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Breaking `malloc` Into Steps

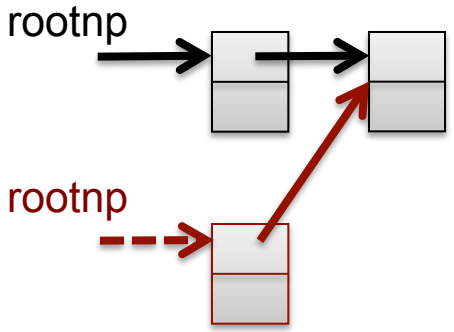
```
np = pmemalloc_reserve(sizeof(*np));

/* link it in at the beginning of the list */
np->next = rootnp;
np->value = value;
pmemalloc_onactive(np, &rootnp, np);


pmemalloc_activate(np);
```



2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Breaking `malloc` Into Steps

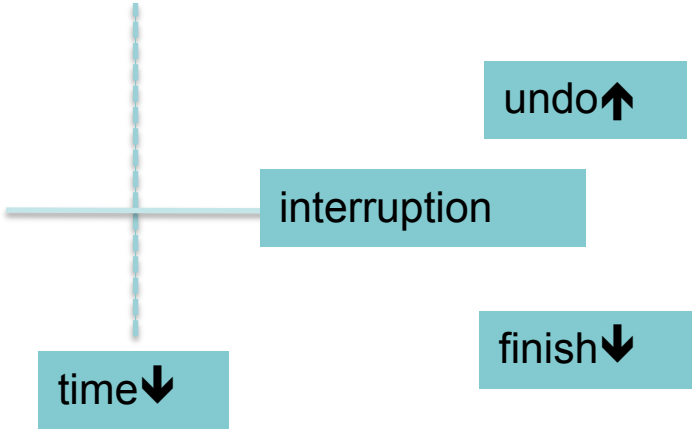
```
np = pmemalloc_reserve(sizeof(*np));

/* link it in at the beginning of the list */
np->next = rootnp;
np->value = value;
pmemalloc_onactive(np, &rootnp, np);


pmemalloc_activate(np);
```



2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Breaking `malloc` Into Steps

```
np = pmemalloc_reserve(sizeof(*np));

/* link it in at the beginning of the list */
np->next = rootnp;
np->value = value;
pmemalloc_onactive(np, &rootnp, np);

pmemalloc_activate(np);
```



2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

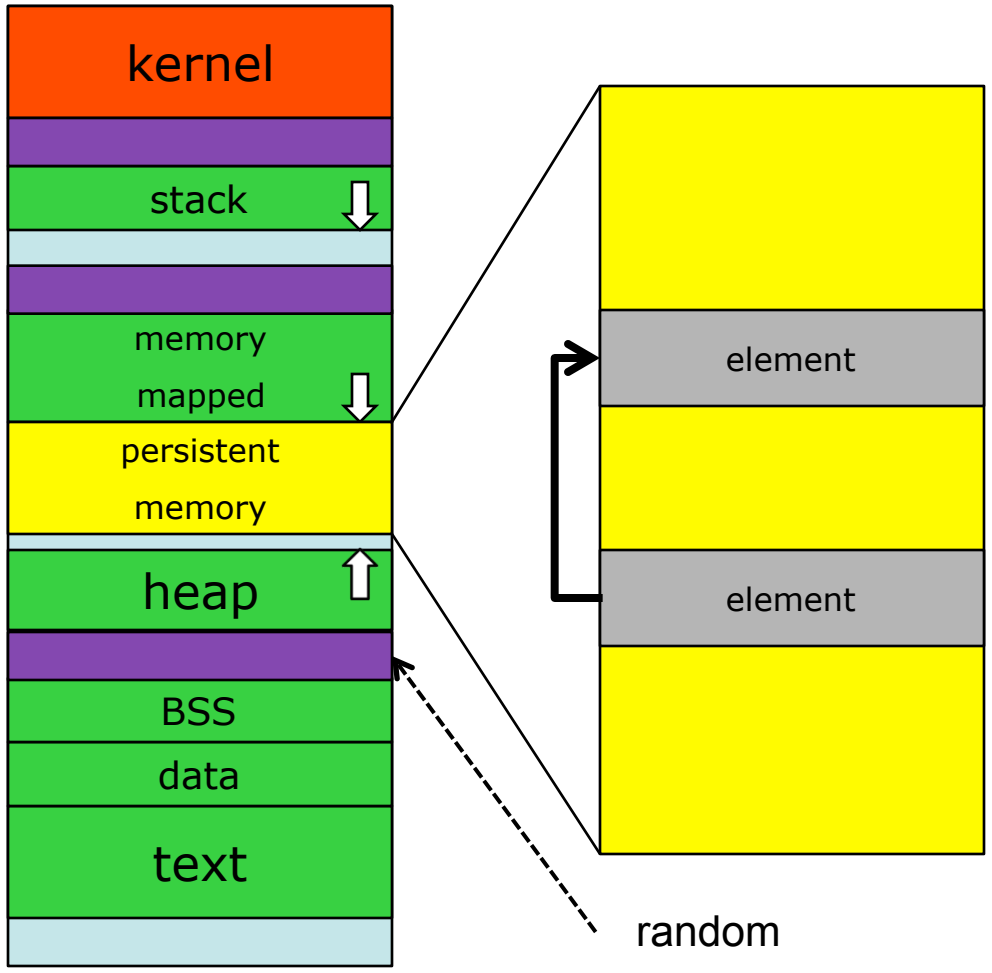
With the Details…

```
if ((np_ = pmemalloc_reserve(pmp, sizeof(*np_))) == NULL)
        FATALSYS("pmemalloc_reserve");

/* link it in at the beginning of the list */
PMEM(pmp, np_)->next_ = sp->rootnp_;
PMEM(pmp, np_)->value = value;
pmemalloc_onactive(pmp, np_, (void **)&sp->rootnp_, np_);


pmemalloc_activate(pmp, np_);
```

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

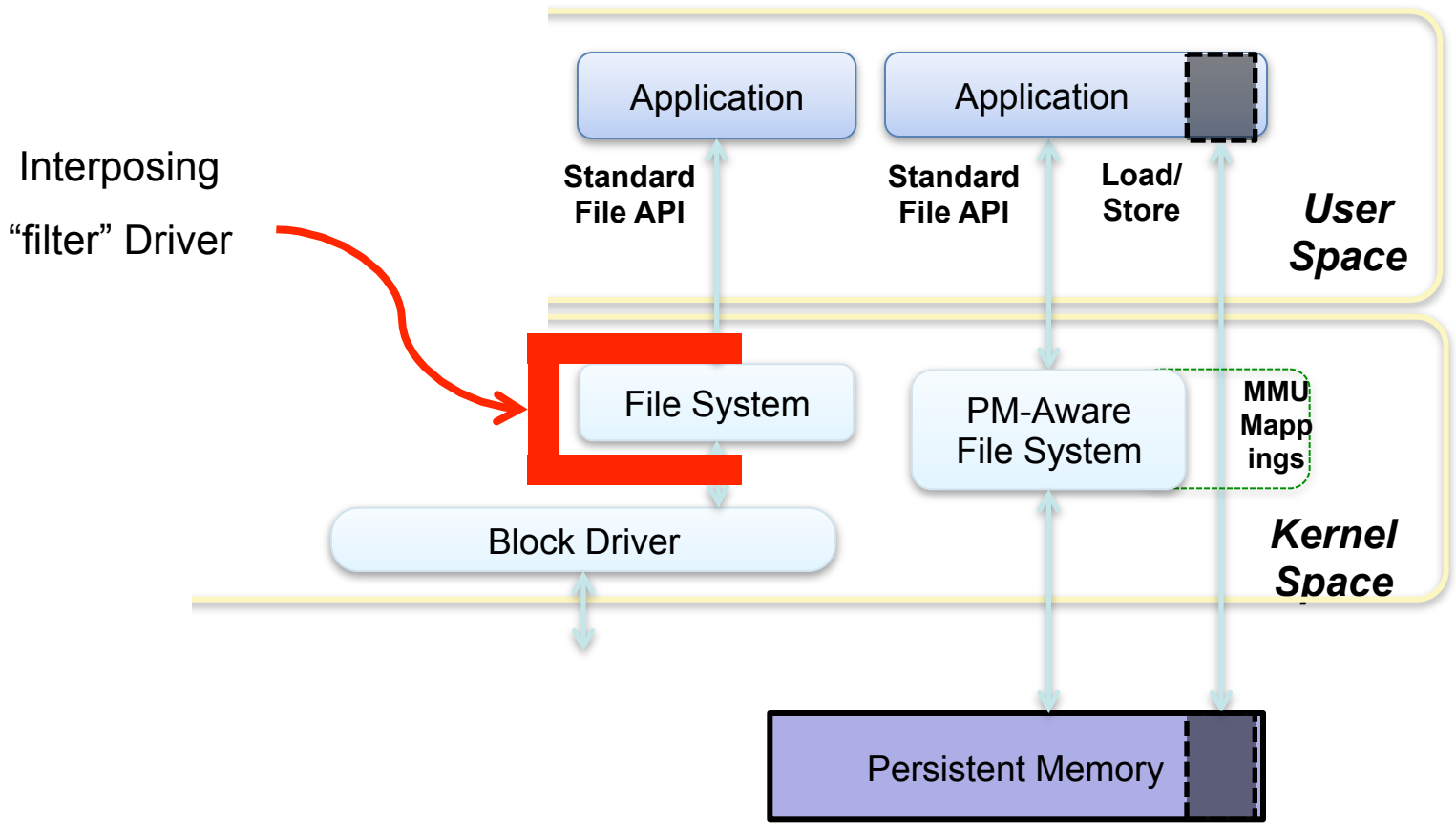
Position Dependence

- Can pointers work across sessions?
- Will a PM file be mapped at the same address every time?



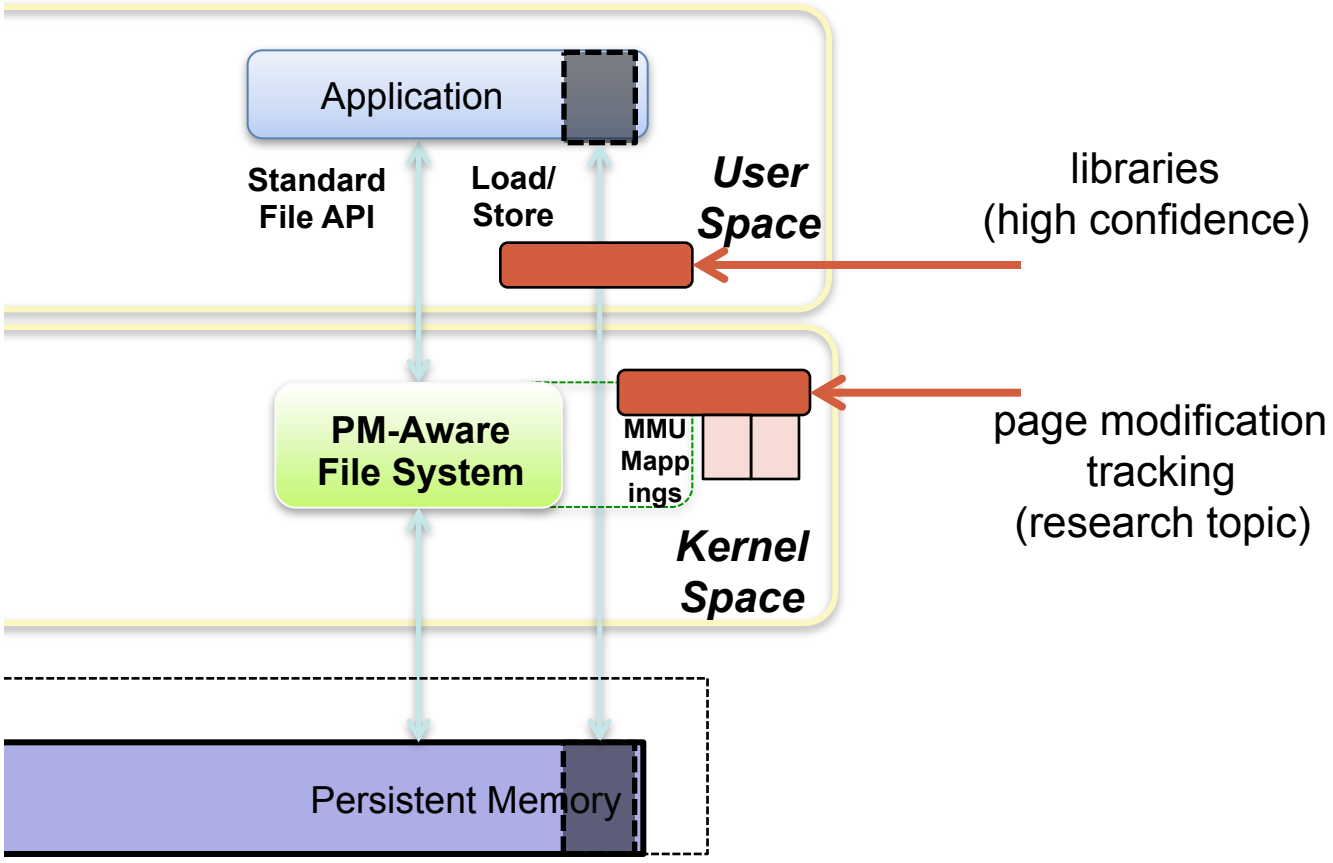
2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

The “C-Clamp”



2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

PM Interposition Points



2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

SNIA NVM Programming TWG

- Charter: Develop specifications for new software "programming models" as NVM becomes a standard feature of platforms
  - Scope:
    - In-kernel NVM programming models
    - Kernel-to-application programming models
  - Programming models specify the exact technical behavior, up to (but not including) the OS specific API semantics
- APIs
  - Each OSV codes the programming models to specific to OS

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.

Programming Model vs. API

- OSVs own their kernel APIs
  - Cannot define these in a committee and push on OSVs
  - Cannot define one API for multiple OS platforms
    - Serious differences on how things work in the kernel
    - Goes against independent innovation
  - Next best thing is to agree on overall model
    - With OSV collaboration
    - Then engage OSV to define and implement API
- Similar situation in user-space
  - A common API doesn't always make sense
    - Violates the "when in Rome" design principle
    - Example: the UNIX* versus the Windows* event models
  - Ultimately: want OSV to ship and maintain the API

2013 FreeBSD Vendor Summit © Intel Corporation. All Rights Reserved.