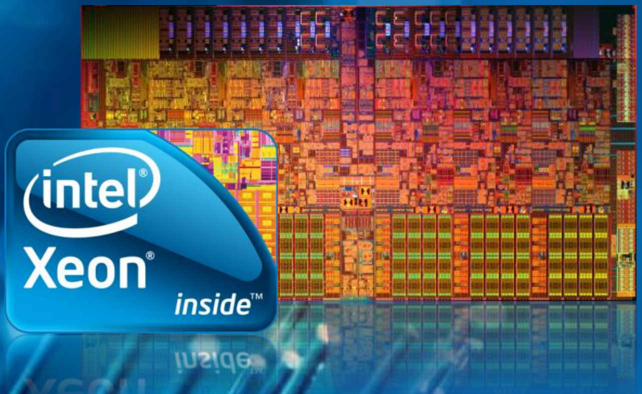


# Persistent Memory: What's Done, Coming Soon, Expected Long-term

Andy Rudoff
Principal Engineer
NVM Software
Intel Corporation

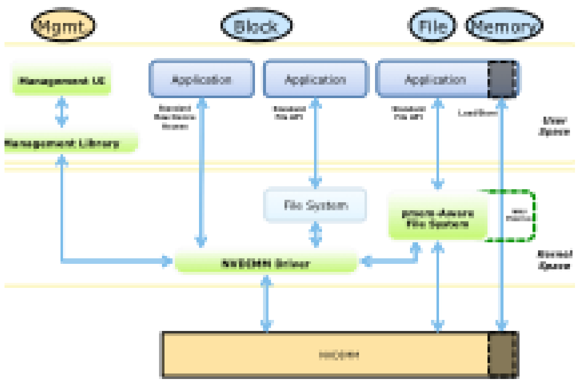
August 2015

## The Plumbers “three-slides” Rule

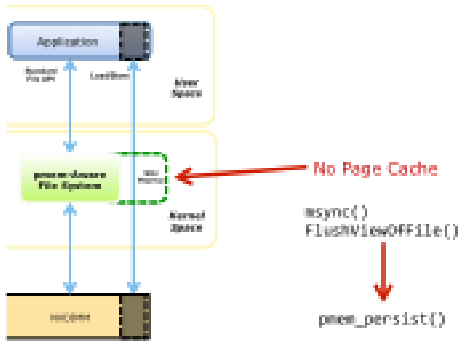
- What has been done – current state
- What is about to happen – announced products
- What is expected to happen – next decade

## Open NVM Programming Model

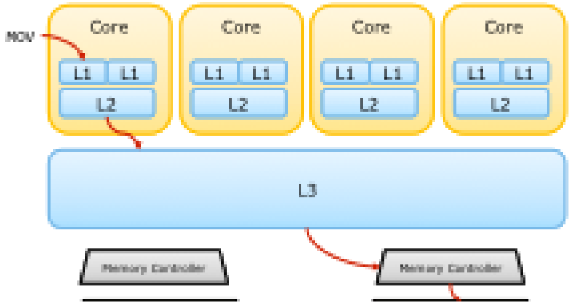
SNIA
50+ Member Companies
HITACHI
DELL
ORACLE
intel
IBM
SAMSUNG
NetApp
Microsoft
FUJITSU
EMC
hp
HUAWEI



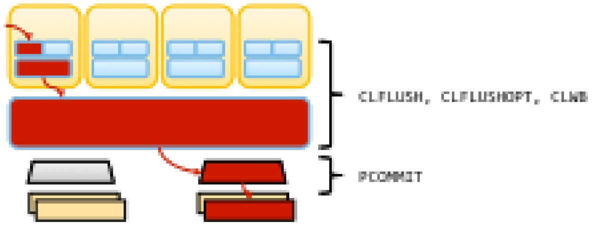
## Where Writes are Cached



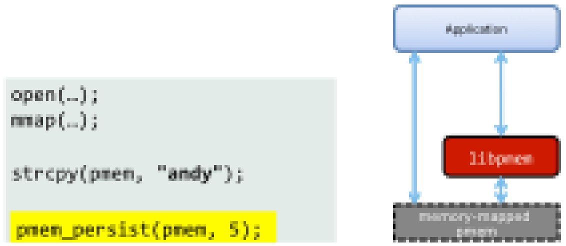
## The Data Path



## Two Levels of Flushing Writes

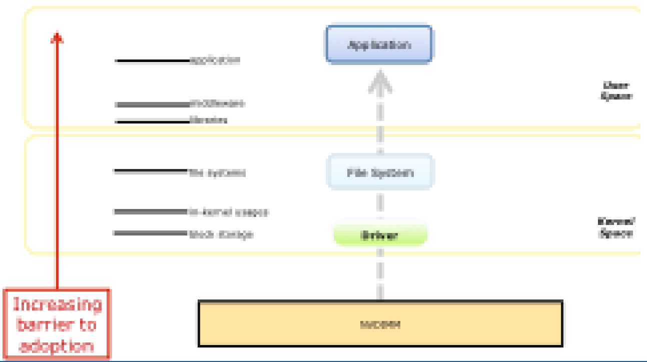


## libpmem Load/Store Persistence

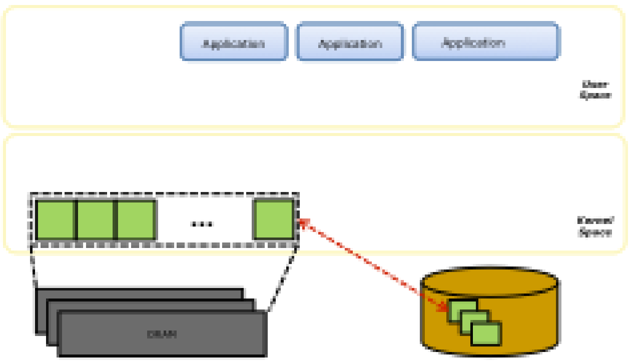


```
open(…);
mmap(…);

strcpy(pmem, "andy");

pmem_persist(pmem, 5);
```

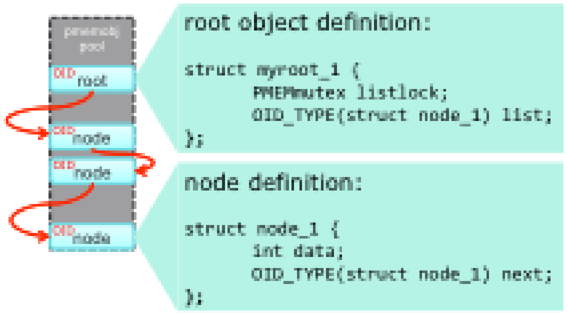
## Assume pmem Exists…
(and the cost is viable)



## Paging from the OS Page Cache



## Linked List Example

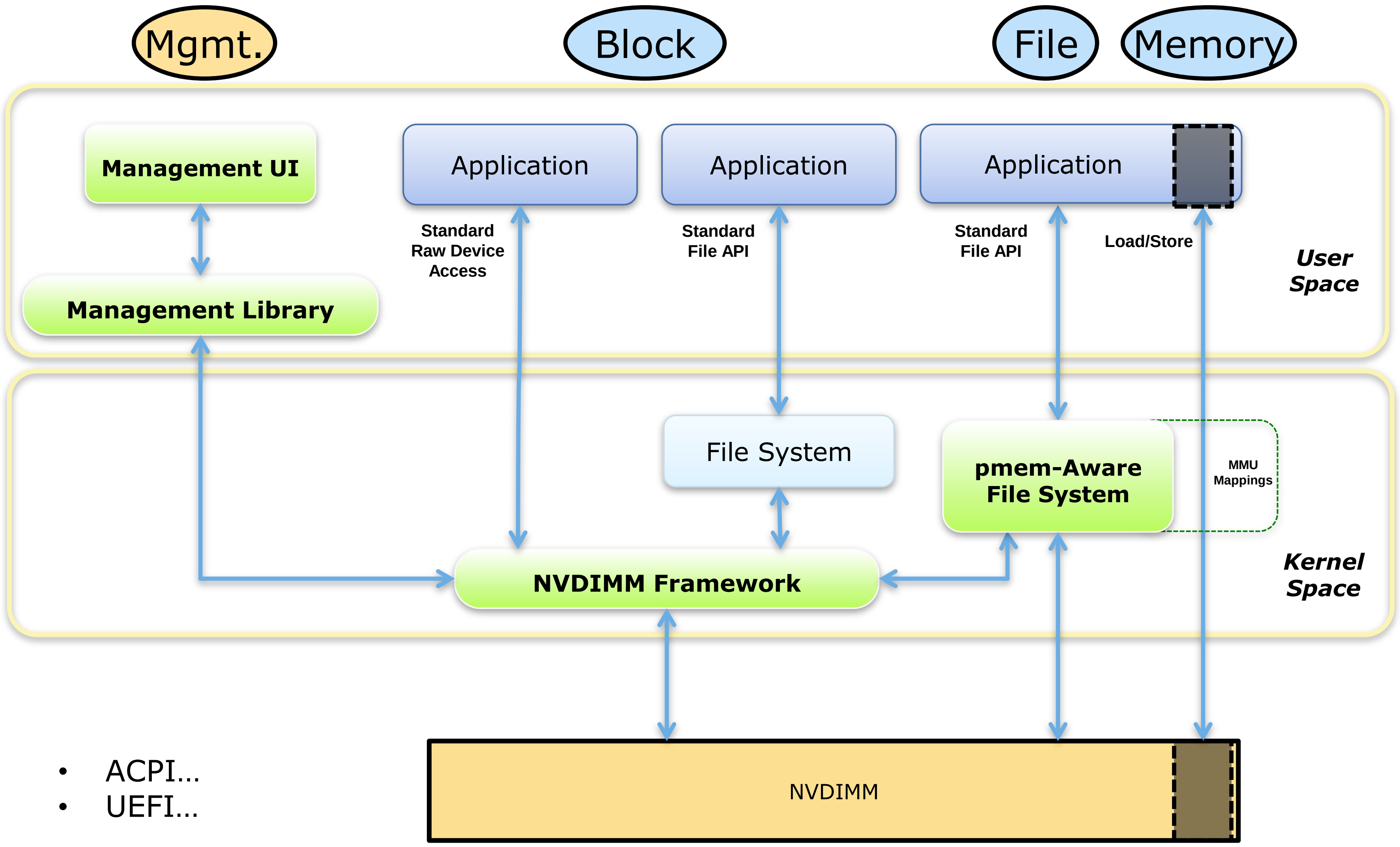


root object definition:

```
struct myroot_1 {
        PMEMmutex listlock;
        OID_TYPE(struct node_1) list;
};
```

node definition:

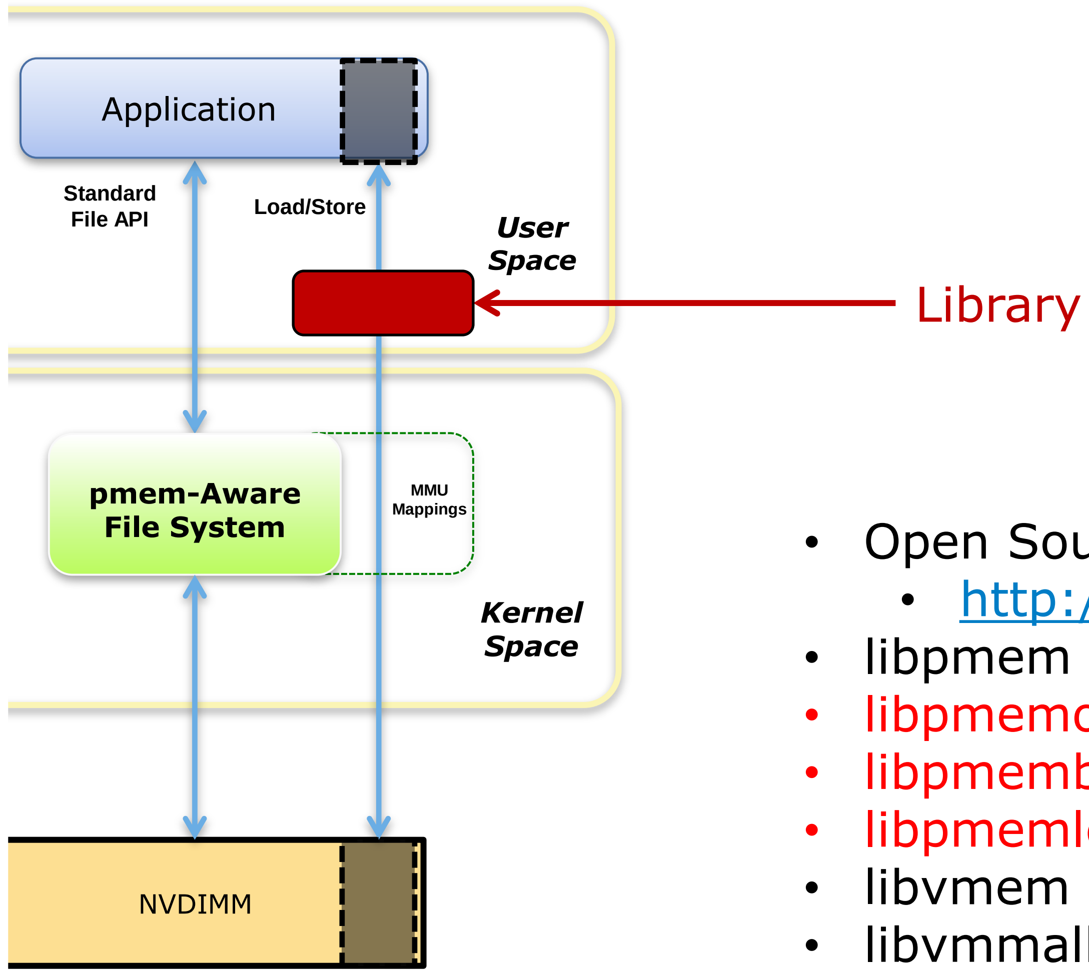
```
struct node_1 {
        int data;
        OID_TYPE(struct node_1) next;
};
```


Mgmt.
Block
File
Memory
Management UI
Management Library
Application
Application
Application
Standard Raw Device Access
Standard File API
Standard File API
Load/Store
User Space
File System
pmem-Aware File System
MMU Mappings
NVDIMM Framework
Kernel Space
NVDIMM

- ACPI…
- UEFI…

# NVM Library: pmem.io

## 64-bit Linux Initially

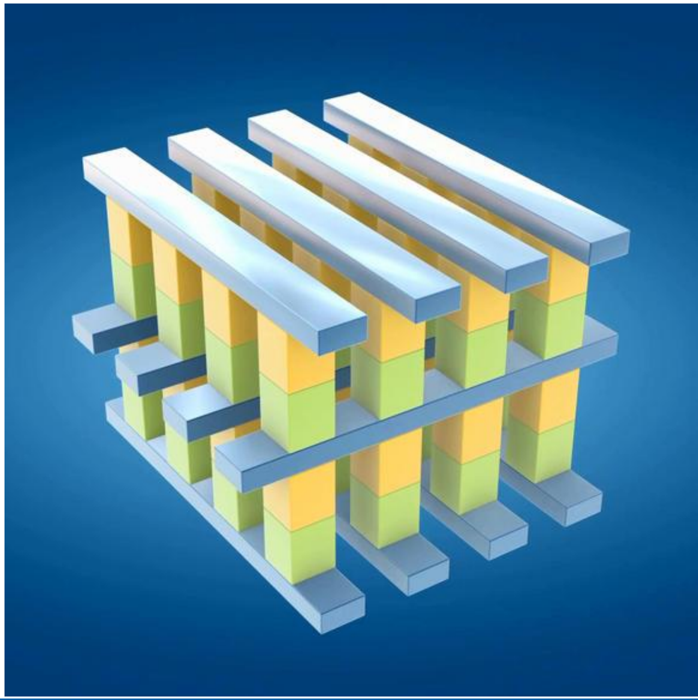


- Open Source
  - http://pmem.io
- libpmem
- libpmemobj
- libpmemblk
- libpmemlog
- libvmem
- libvmmalloc

Transactional

# Persistent Memory

- Byte addressable persistence
  - Fast enough to load directly
  - Usually on memory bus
- NVDIMMs available today
- 3D XPoint™ Memory
  - Persistent
  - (up to) 1000X faster than NAND
  - (up to) 1000X endurance
  - **6TB per 2-socket system**
  - **Cheaper than DRAM**
  - SSDs first (demonstrated this week)
  - Intel DIMMs for next gen platform

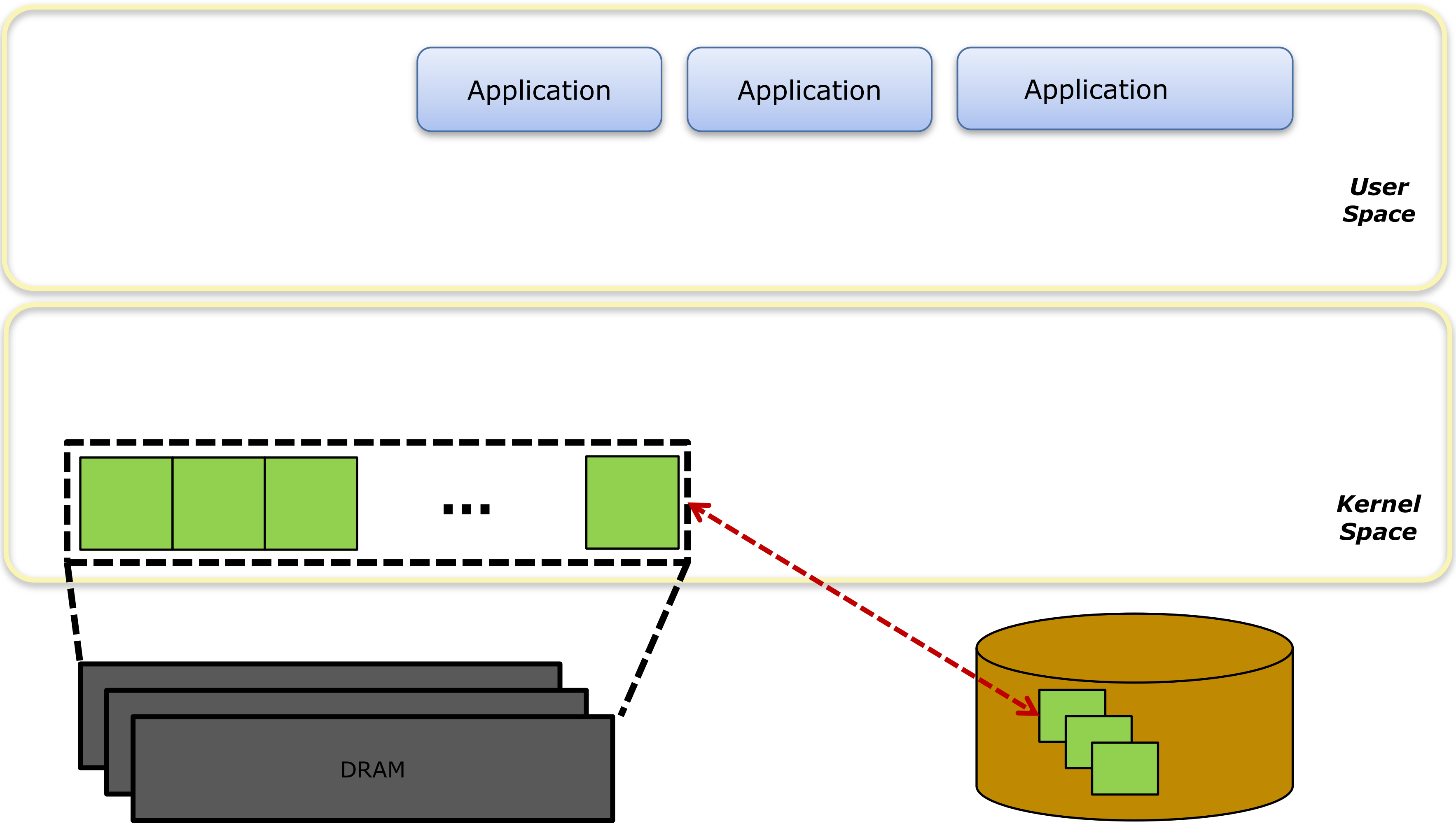


# The Future

- Some more basics
  - RAS, Replication, RDMA
- Many emerging memory types
  - Each with different performance characteristics
  - Each with different cost & capacity
  - Sometimes with different RAS characteristics
  - NUMA locality still applies
  - And sometimes it is non-volatile
- Application Transparent
  - The OS manages the tiers of memory
  - The server space overcomes their fear of paging
  - Used by OS components, run-times, libraries…
- Non-Application Transparent
  - Expose it all, administratively and via APIs
  - More help for transactions and replication

## BACKUP

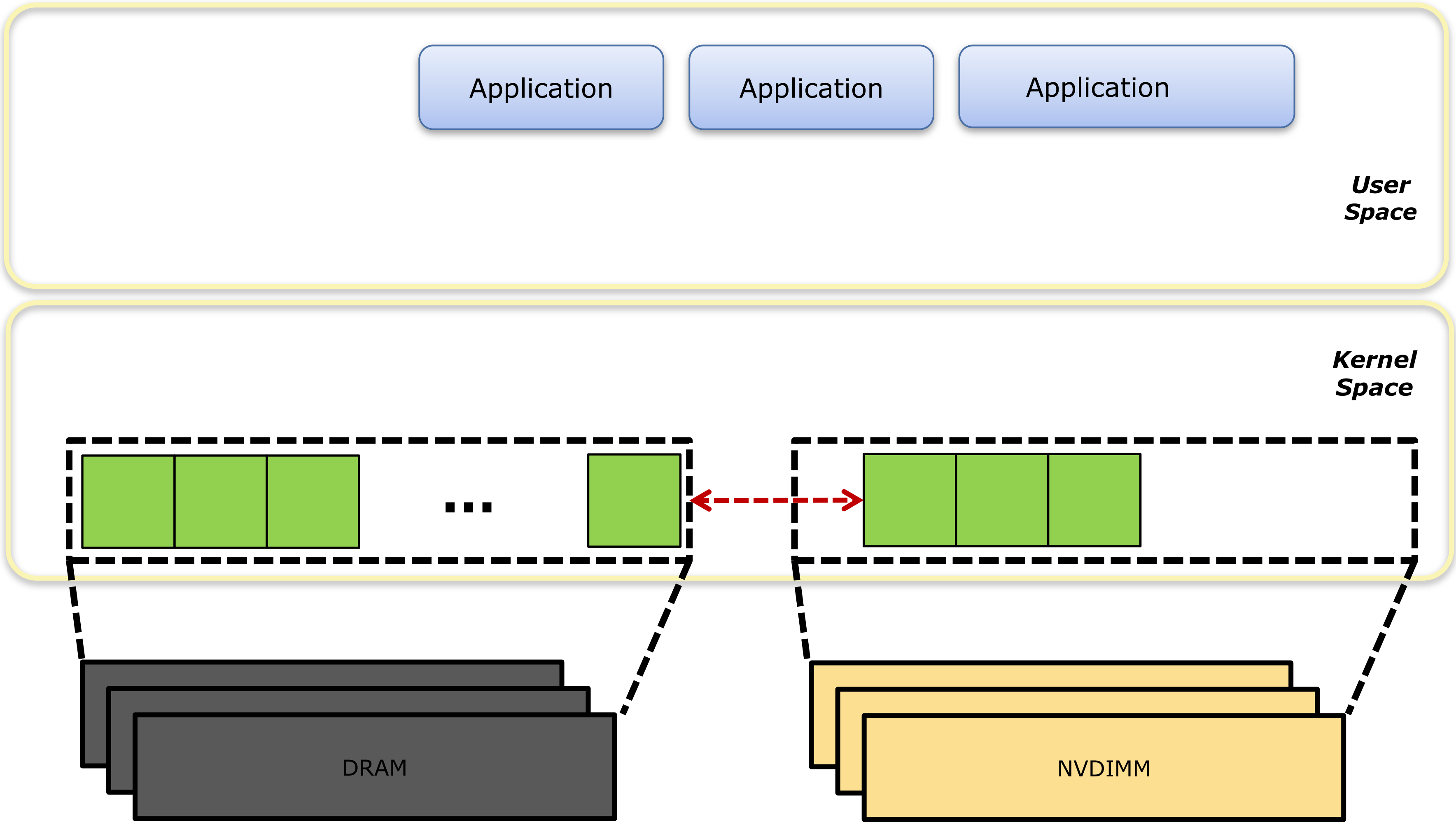
# Paging from the OS Page Cache



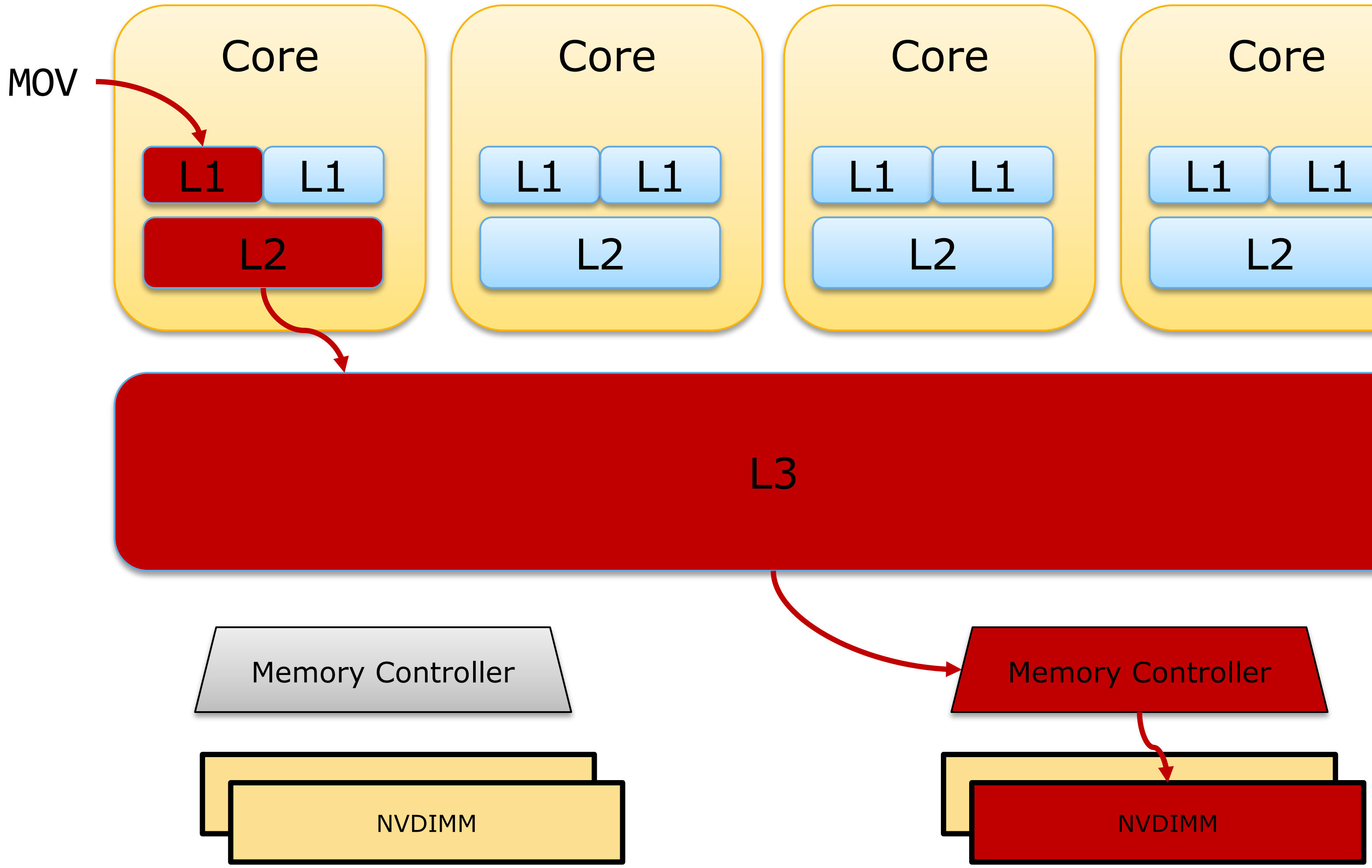
# Attributes of Paging
## (and why everyone avoids it)

- Major page faults
  - Block I/O (page I/O) on demand
  - Context switch – there and back again
  - Latency of block stack
- Available memory looks much larger
  - But penalty of fault is significant
- Page in must pick a victim
  - Based on simplistic R/M metric
  - Can surprise an application
- Many enterprise apps opt-out
  - Managing page cache themselves
  - Using intimate date knowledge for paging decisions
- Interesting example: Java GC

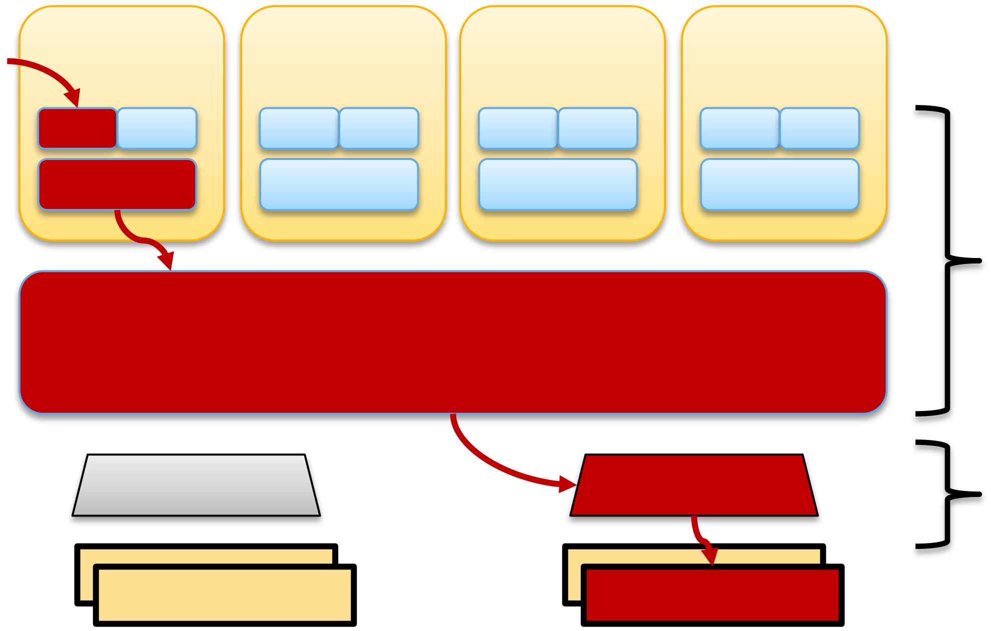
# Paging to pmem



# Hiding Places



# Two Levels of Flushing Writes



CLFLUSH, CLFLUSHOPT, CLWB

PCOMMIT

## Crossing the 8-byte Store

```
open(…);
mmap(…);

strcpy(pmem, "andy rudoff");

pmem_persist(pmem, 12);
```

*crash*

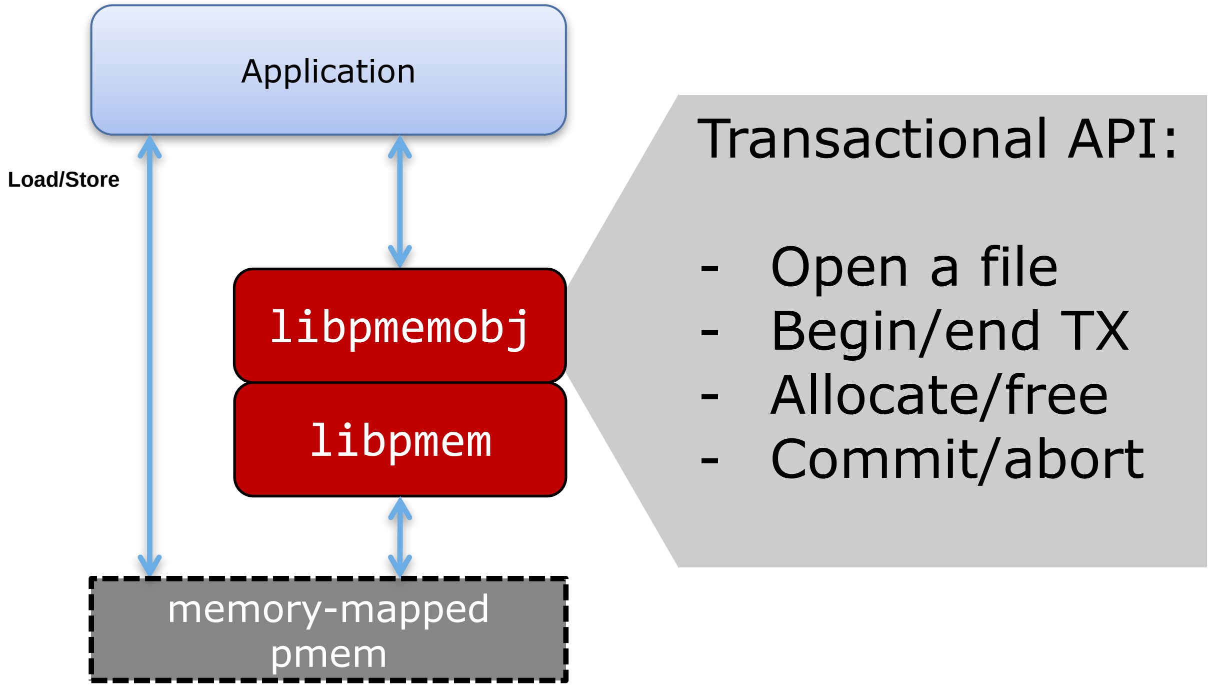
Which Result?

```
1. "\0\0\0\0\0\0\0\0\0…"
2. "andy\0\0\0\0\0\0…"
3. "andy rud\0\0\0\0\0…"
4. "\0\0\0\0\0\0\0\0off\0\0..."
5. "andy rudoff\0"
```

# Visibility vs Powerfail Atomic

| Feature | Atomicity |
|---|---|
| Atomic Store | 8 byte powerfail atomicity<br>Much larger visibility atomicity |
| TSX | Programmer must comprehend XABORT, cache flush can abort |
| LOCK CMPXCHG | *non-blocking* algorithms depend on CAS, but CAS doesn’t include flush to persistence |

# Transactional Object Store

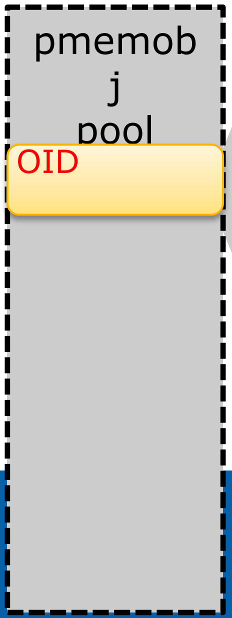


Transactional API:

- Open a file
- Begin/end TX
- Allocate/free
- Commit/abort

# Simple pmemobj Transaction

```
TX_BEGIN_LOCK(pop, TX_LOCK_MUTEX, &op->mylock) {

        TX_STRCPY(op->name, "andy rudoff");

} TX_END
```



Object definition:

```
struct myobj {
    PMEMmutex mylock;
    char name[NAMELEN];
};
```

# Two Types of Atomicity

```
TX_BEGIN_LOCK(pop, TX_LOCK_MUTEX, &op->mylock) {

        TX_STRCPY(op->name, "andy rudoff");

} TX_END
```

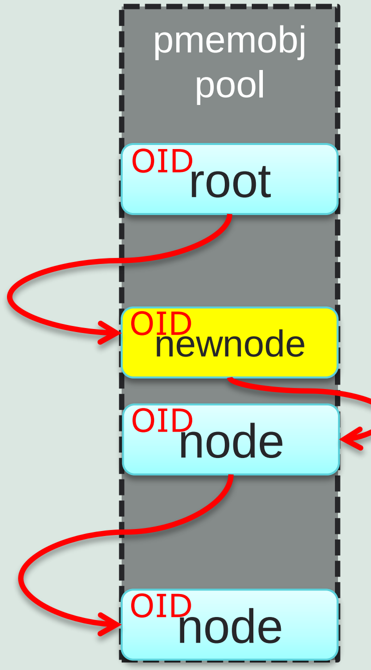
Powerfail
Atomicity

Multi-Thread
Atomicity

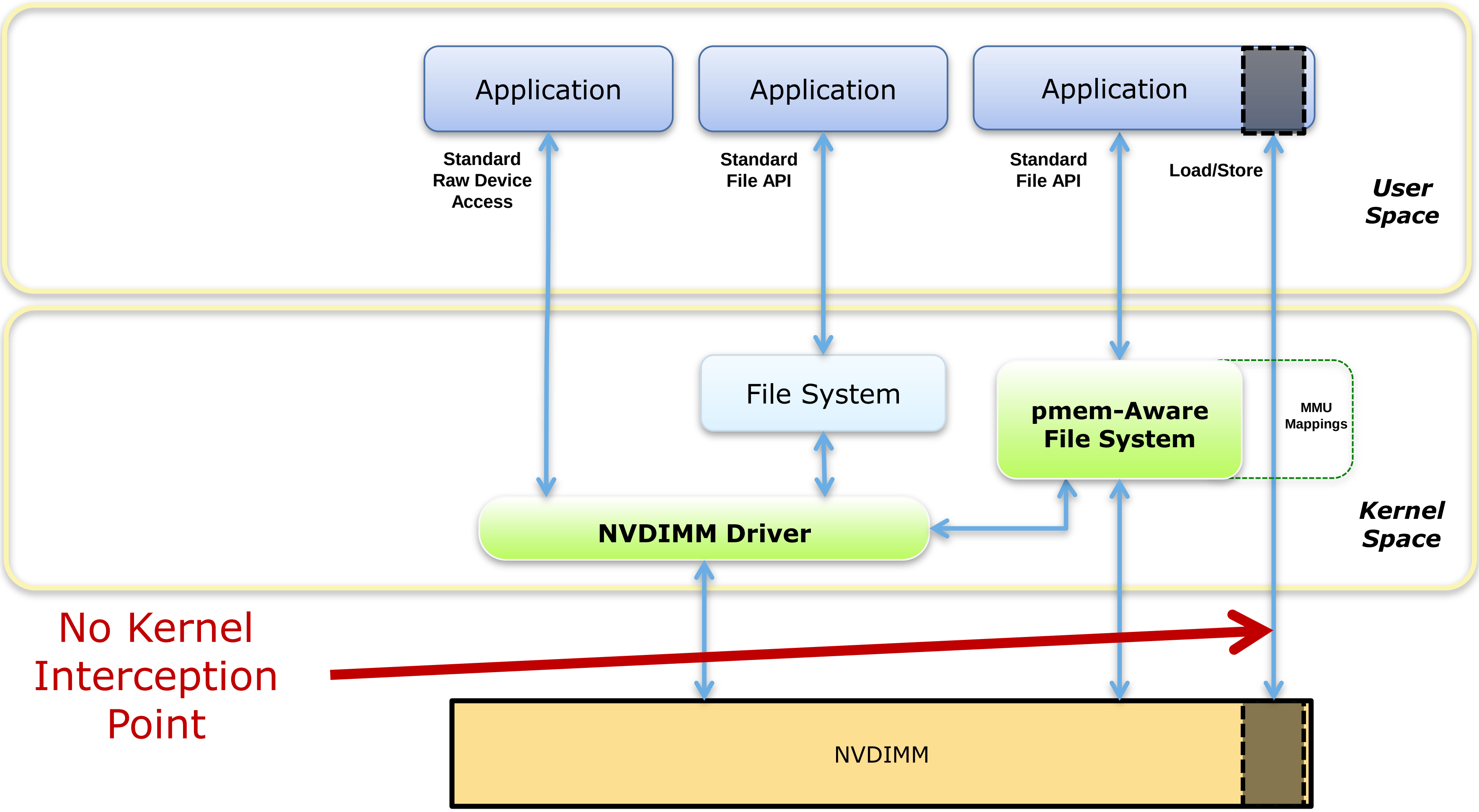
# In libpmemobj Macro Magic

## (the *assembly language* of pmem programming)

```
TX_BEGIN_LOCK(Pop, TX_LOCK_MUTEX, &D_RW(rootoid)->listlock) {
        OID_TYPE(struct node) newnodeoid =
                                TX_ZALLOC(struct node, 0);

        D_RW(newnodeoid)->data = data;
        D_RW(newnodeoid)->nextoid = D_RO(rootoid)->headoid;
        TX_ADD(rootoid);
        D_RW(rootoid)->headoid = newnodeoid;

} TX_ONABORT {
        perror("transaction failed");
        /* ... */
} TX_END
```

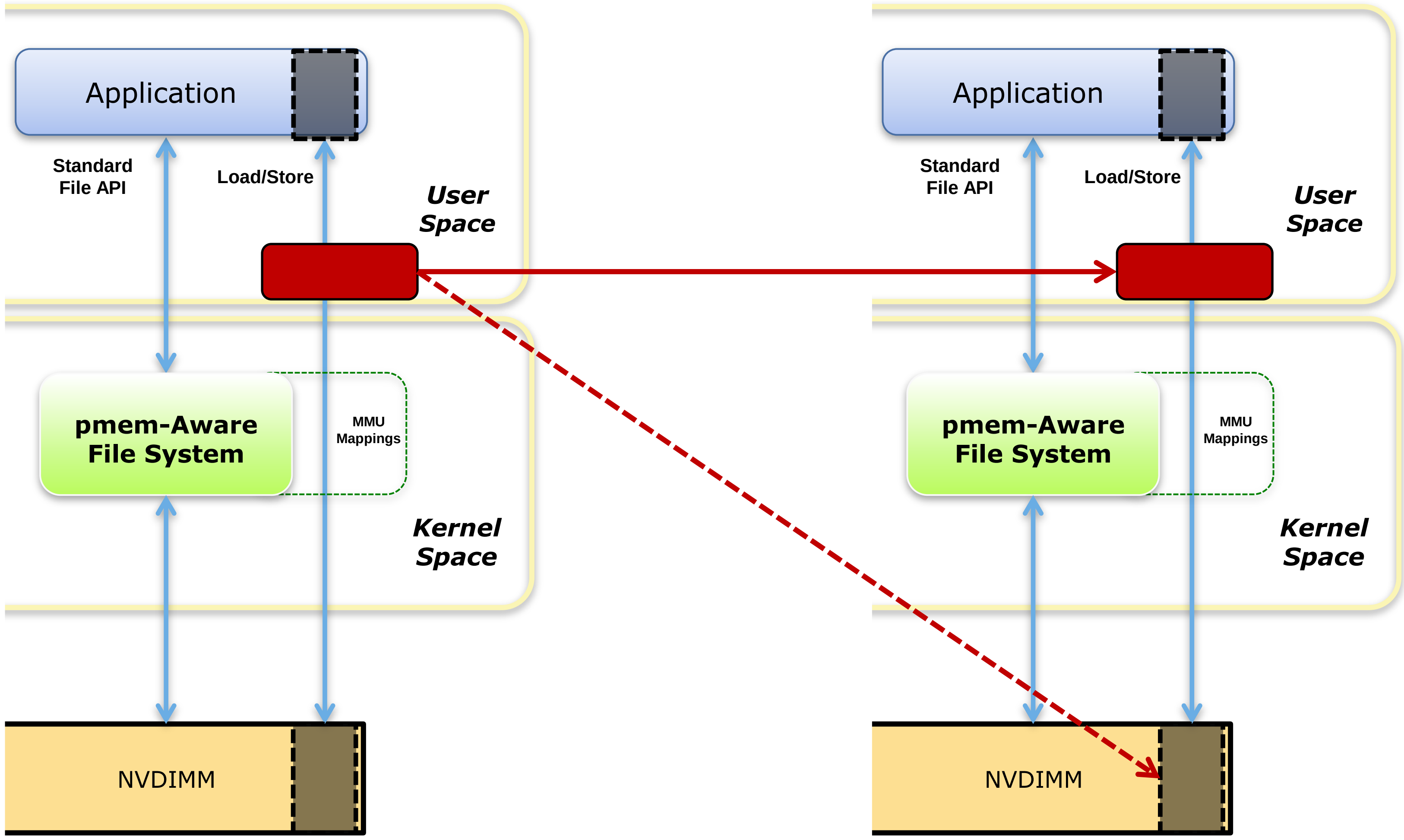


# Replication Challenge of pmem

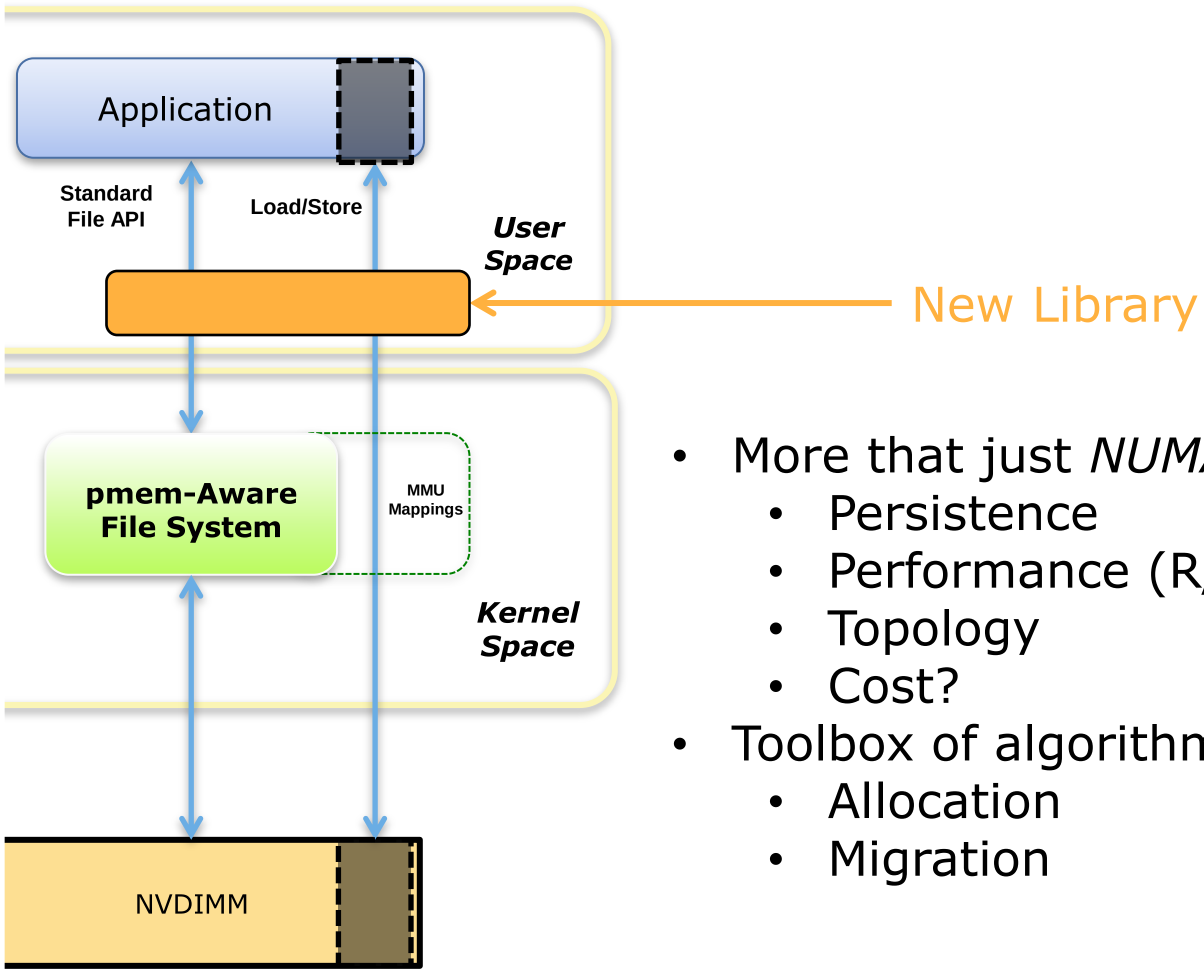


No Kernel Interception Point

# RDMA to pmem


Application
Standard File API
Load/Store
User Space
pmem-Aware File System
MMU Mappings
Kernel Space
NVDIMM
Application
Standard File API
Load/Store
User Space
pmem-Aware File System
MMU Mappings
Kernel Space
NVDIMM

# Evolving libnuma (and libmemkind)



- More that just *NUMA distance*
  - Persistence
  - Performance (R/W)
  - Topology
  - Cost?
- Toolbox of algorithms
  - Allocation
  - Migration

## For More Information…

- SNIA NVM Programming Model
  - http://www.snia.org/forums/sssi/nvmp
- Intel Architecture Instruction Set Extensions Programming Reference
  - https://software.intel.com/en-us/intel-isa-extensions
- Open Source NVM Library work
  - http://pmem.io
- Linux kernel support & instructions
  - https://github.com/01org/prd

# Even More Information…

- ACPI 6.0 NFIT definition (used by BIOS to expose NVDIMMs to OS)
  - http://www.uefi.org/sites/default/files/resources/ACPI_6.0.pdf
- Open specs providing NVDIMM implementation examples, layout, BIOS calls:
  - http://pmem.io/documents/
- Google group for pmem programming discussion:
  - http://groups.google.com/group/pmem