Persistent Memory Programming:
The Current State of the Ecosystem

MSST 2017

Andy Rudoff

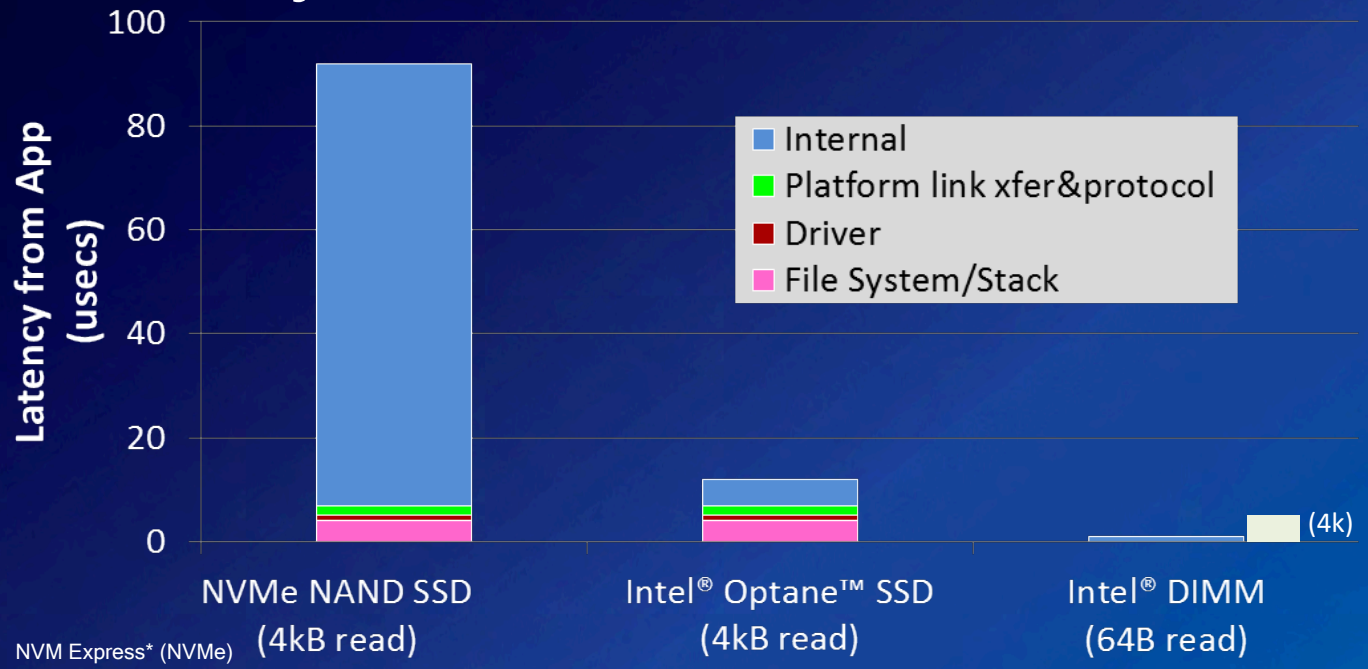
Non-Volatile Memory Software Architect

Intel Corporation

DCG
Data Center Group

Persistent Memory

Optimized System Interconnect



Reach full potential of 3D XPoint™ Technology by connecting it as Memory

Sources: "Storage as Fast as the rest of the system" 2016 IEEE 8th International Memory Workshop and measurement, Intel® Optane™ SSD measurements and Intel P3700 measurements, and technology projections

Java PersistentSortedMap

```
PersistentSortedMap employees = new PersistentSortedMap();

…
employees.put(id, data);
```

No flush calls.
Transactional.
Java library handles it all.

See “pilot” project at: `https://github.com/pmem/pcj`

How Did We Get from a DIMM to a Java API?

Persistent Memory

High-Level Language Support

```
PersistentSortedMap

employees.put(id, data);
```

How Did We Get from a DIMM to a Java API?

Persistent Memory

High-Level Language Support

```
PersistentSortedMap

employees.put(id, data);
```

Programming Models

“Programming Model”
At least four meanings…

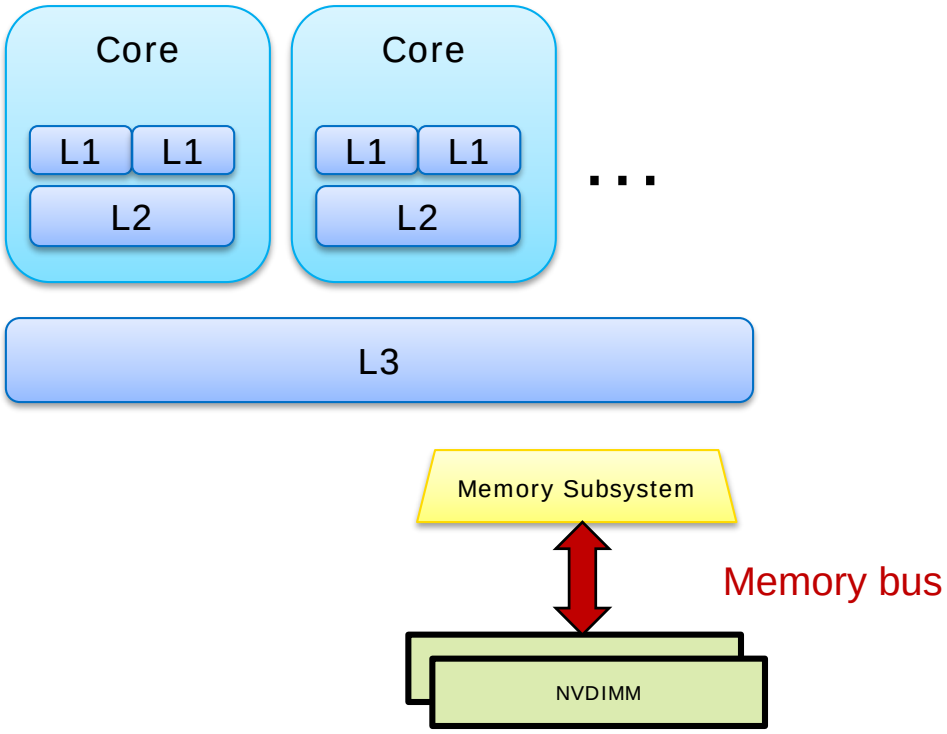
1. Interface between HW and SW

2. Instruction Set Architecture (ISA)

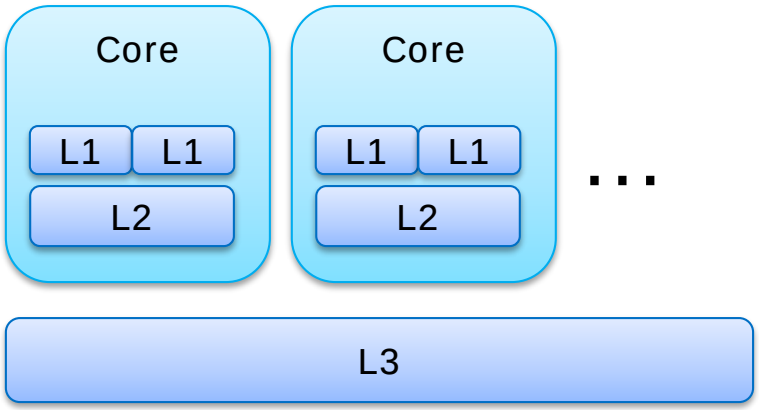
3. Exposed to Applications (by the OS)

4. The Programmer Experience

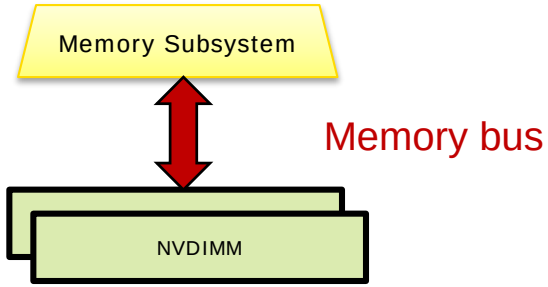
Programming Model: SW Interface to HW



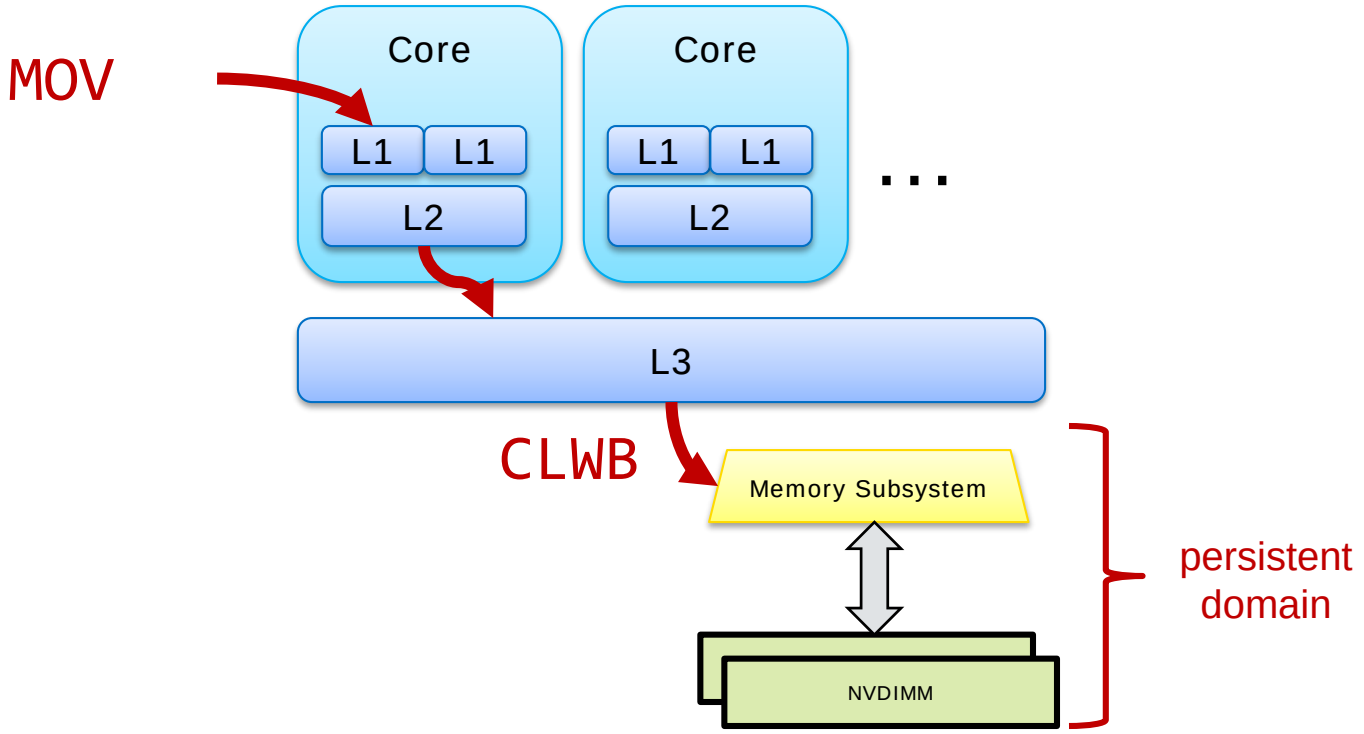
Programming Model: SW Interface to HW



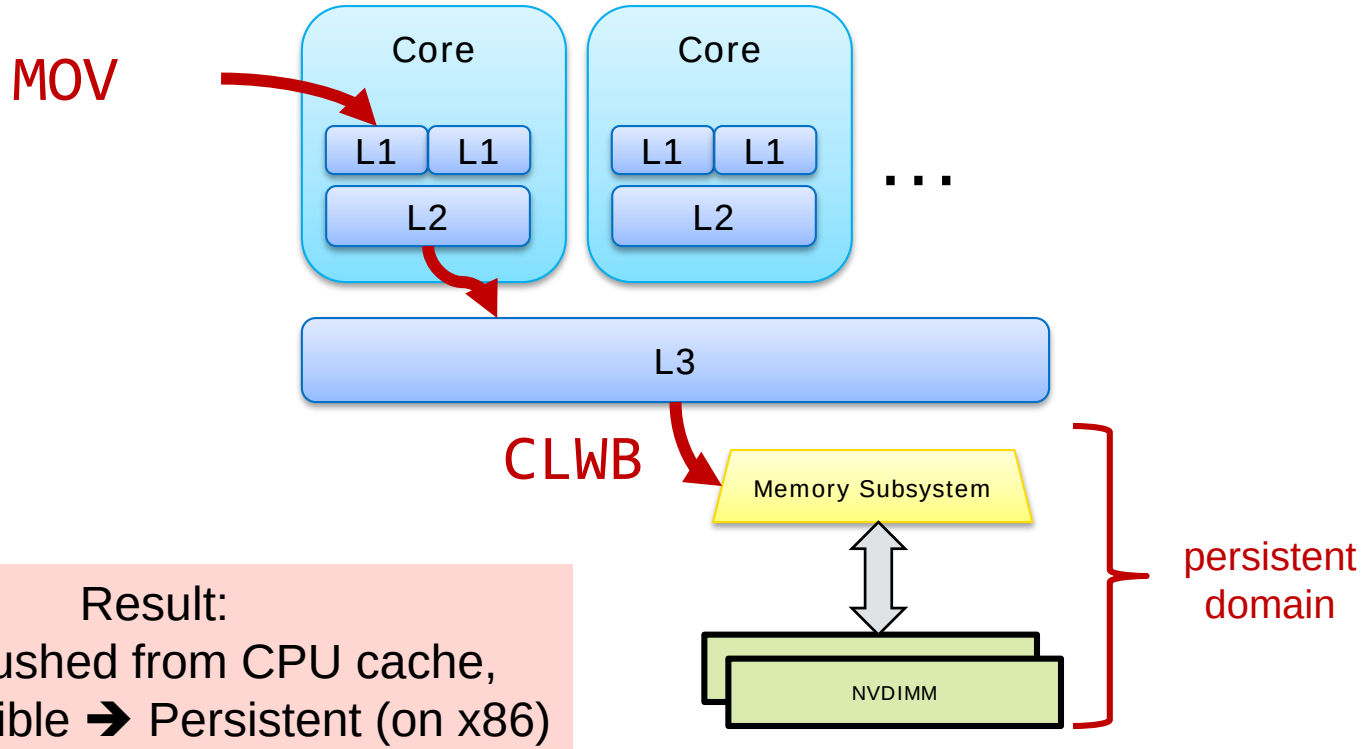
Result:
Persistent Memory hardware accessed like memory (cache coherent).
Exposed by ACPI 6.0+ on x86.



Programming Model: Instruction Set Architecture

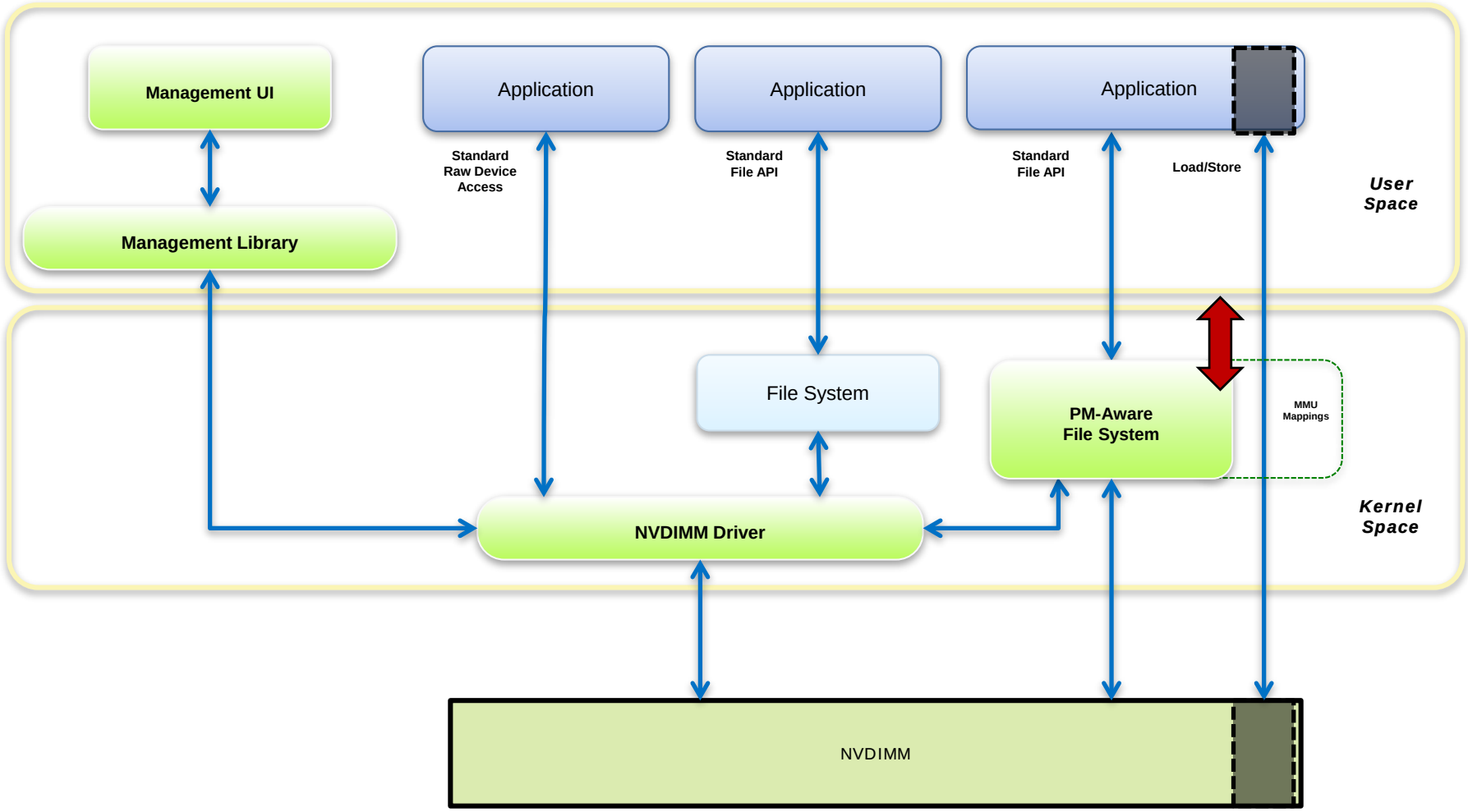


Programming Model: Instruction Set Architecture



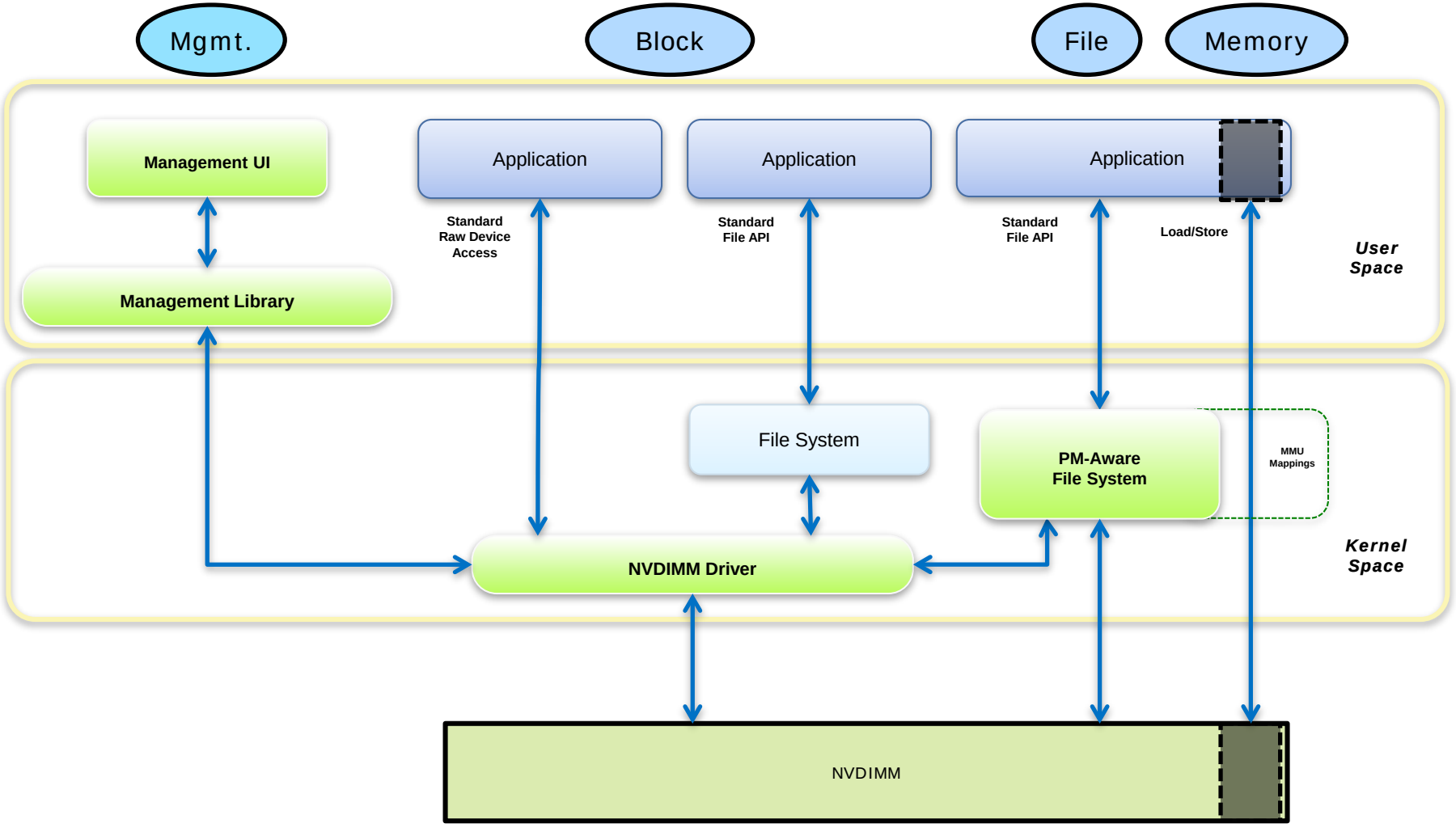
Result:
Stores flushed from CPU cache, globally-visible ➔ Persistent (on x86)

Programming Model: Exposing to Applications



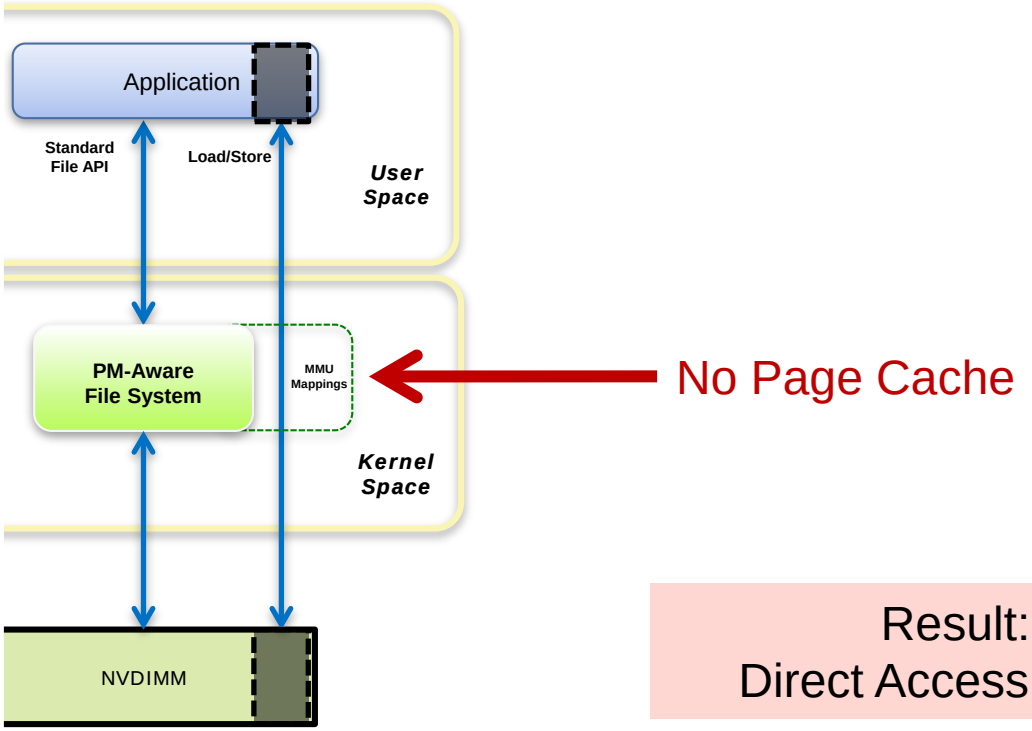
SNIA NVM Programming Model

(Persistent Memory Portion)



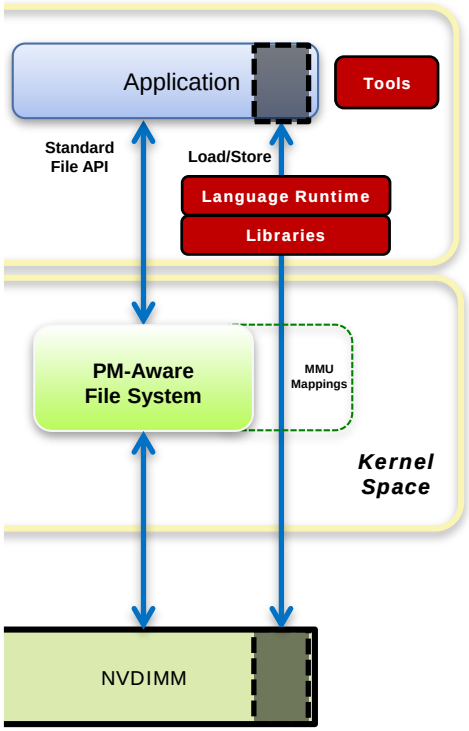
Memory-Mapped Files

The Heart of the PM Programming Model



Result:
Direct Access (DAX)

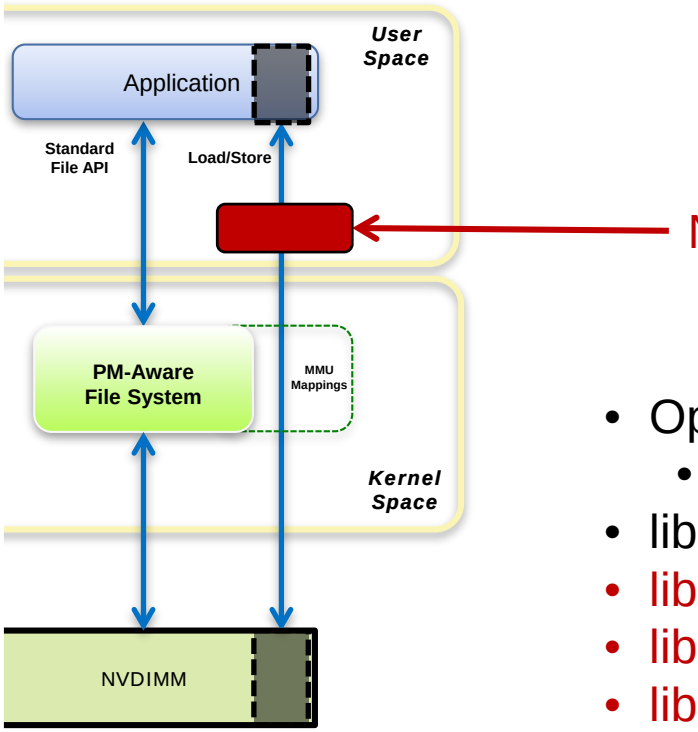
Programming Model:
The Programmer Experience



Result:
Safer, less error-prone,
idiomatic in common languages

NVM Libraries: pmem.io

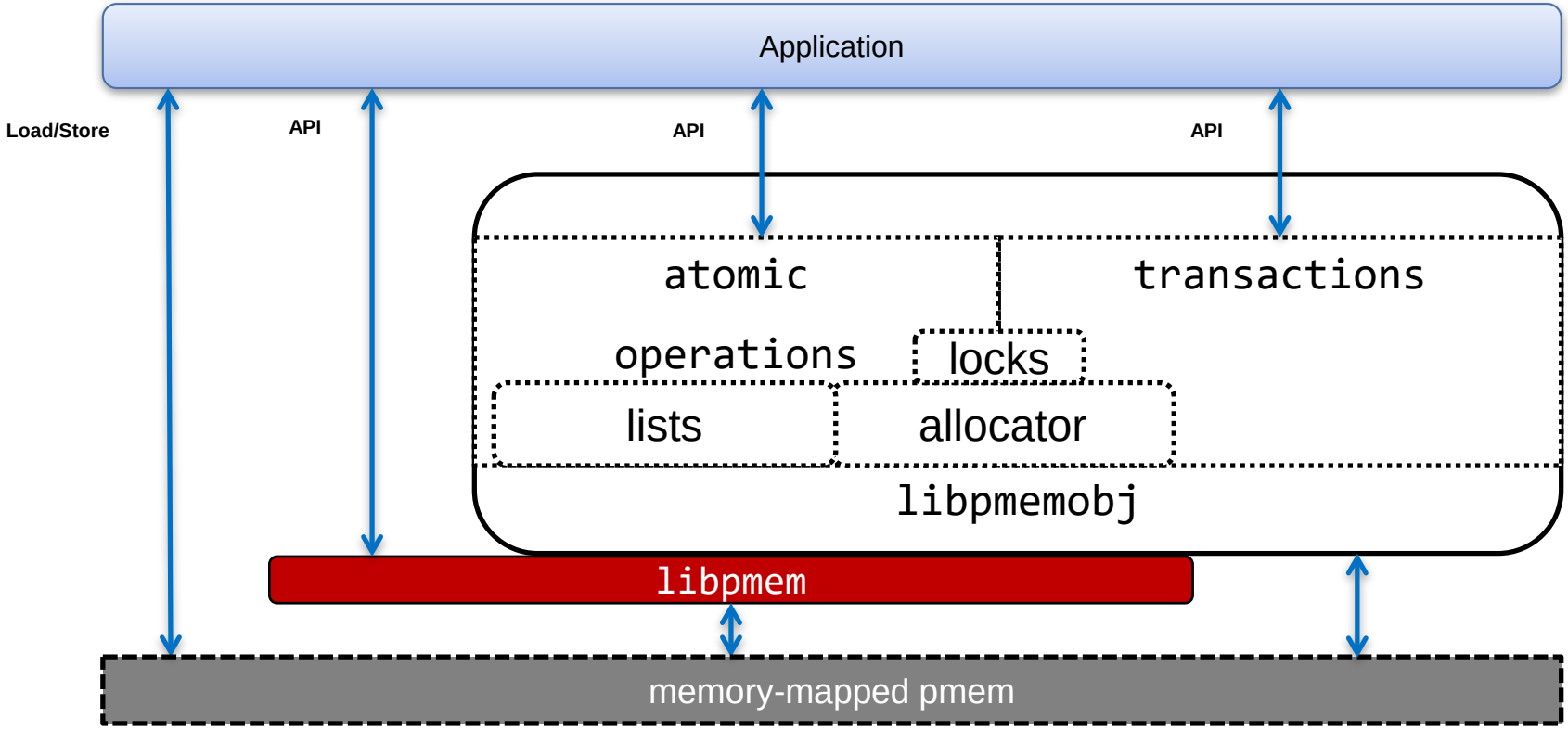
C/C++ on Linux and Windows



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

Transactional

libpmemobj


Application
Load/Store
API
API
API
atomic
transactions
operations
locks
lists
allocator
libpmemobj
libpmem
memory-mapped pmem

State of Ecosystem Today

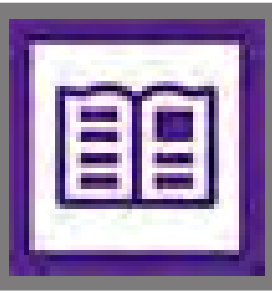
OS Detection of NVDIMMs	**ACPI 6.0+**
OS Exposes pmem to apps	***DAX* provides SNIA Programming Model** Fully supported: • Linux (ext4, XFS) • Windows (NTFS)
OS Supports Optimized Flush	Specified, but evolving (ask when safe) • Linux: **unsafe** except Device DAX • (and new file systems like **NOVA**) • Windows: **safe**
Remote Flush	Proposals under discussion (works today with extra round trip)
Deep Flush	Upcoming Specification
Transactions, Allocators	Built on above via libraries and languages: • http://pmem.io **Much more language support to do**
Virtualization	All VMMs planning to support PM in guest (KVM changes upstream, Xen coming, others too…)

NVM Programming Model Resources

www.snia.org/PM

SNIA Standards Portfolio

- NVM Programming Model v1.2a – *Draft for public review*
- NVM Programming Model v1.1- *SNIA Technical Position*
- NVM Programming Model v1.0 - *SNIA Technical Position*



SNIA Technical White Papers

- NVM PM Remote Access for High Availability
- Persistent Memory Atomics and Transactions

SNIA Videos and Presentations

- The SNIA NVM Programming Model – Latest Developments and Challenges
- Persistent Memory Summit 2017

NVM Libraries
http://pmem.io

C/C++

- C++ bindings: http://pmem.io/nvml/cpp_obj
- libpmemobj page: http://pmem.io/nvml/libpmemobj
- Upstream in some distros already, Windows preview available

NVML Source Tree

- https://github.com/pmem/nvml

Persistent Collections for Java (experimental)

- https://github.com/pmem/pcj

Enhanced valgrind for Persistent Memory

- https://github.com/pmem/valgrind