SDC 20

BY Developers FOR Developers

Storage Developer Conference
September 22-23, 2020

Persistent Memory Programming Without All That Cache Flushing

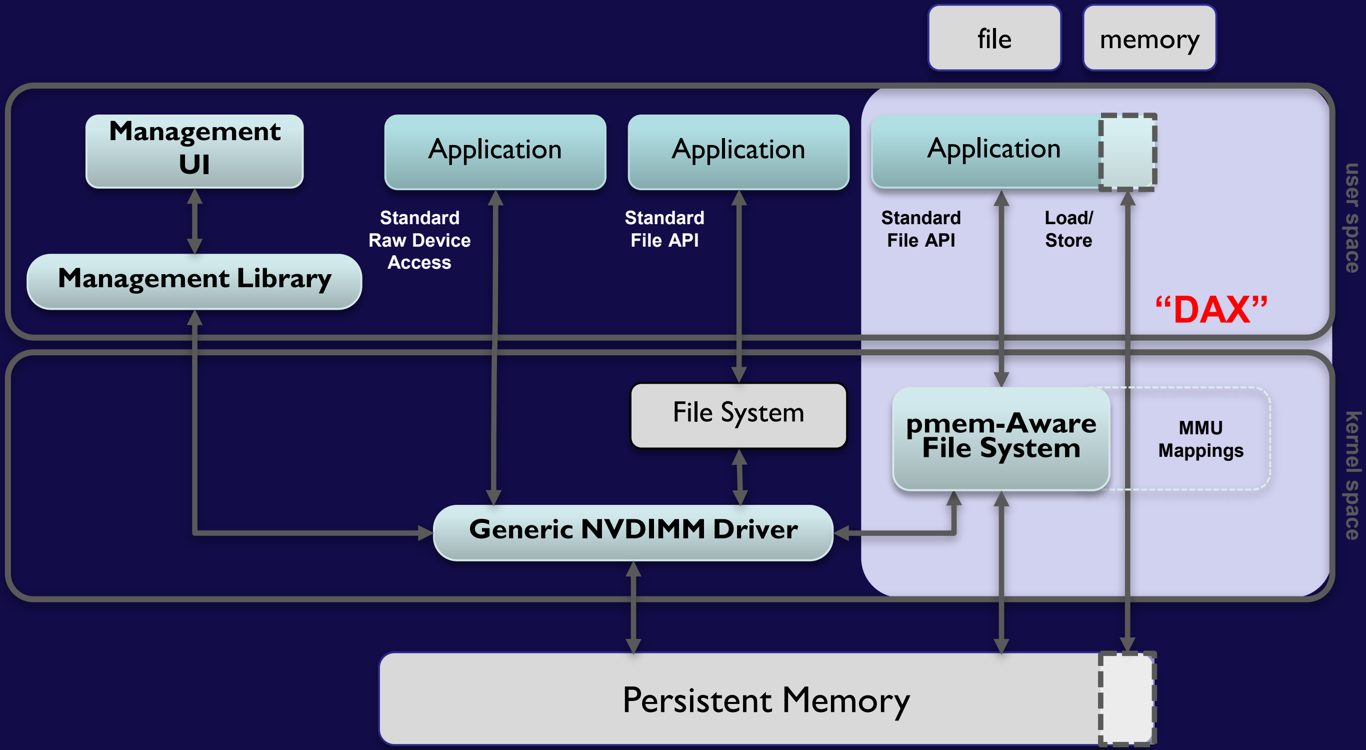
Andy Rudoff
Intel

The Essential Background

With my SNIA Hat On…

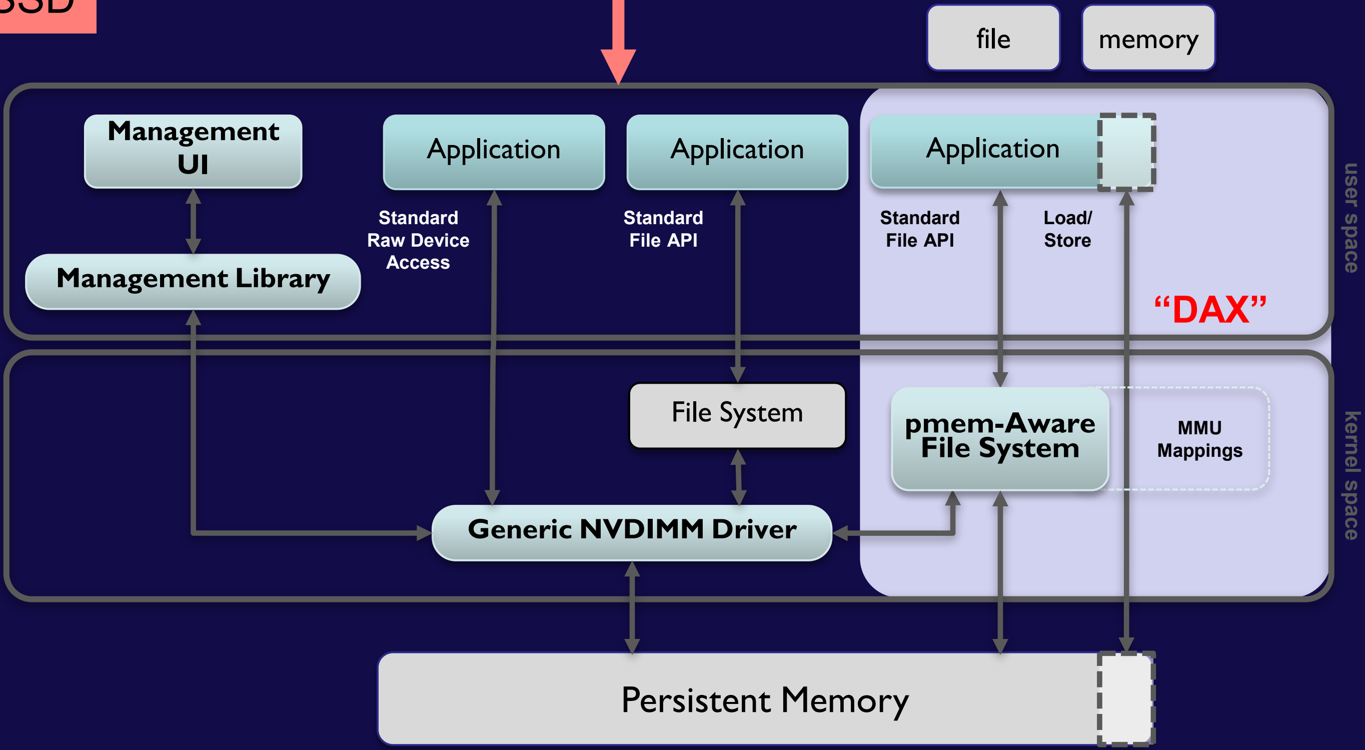
- What is pmem?
  - Byte addressable
  - Reasonable to wait for a load
- How is pmem exposed to applications?
  - NVM Programming Model

The Programming Model

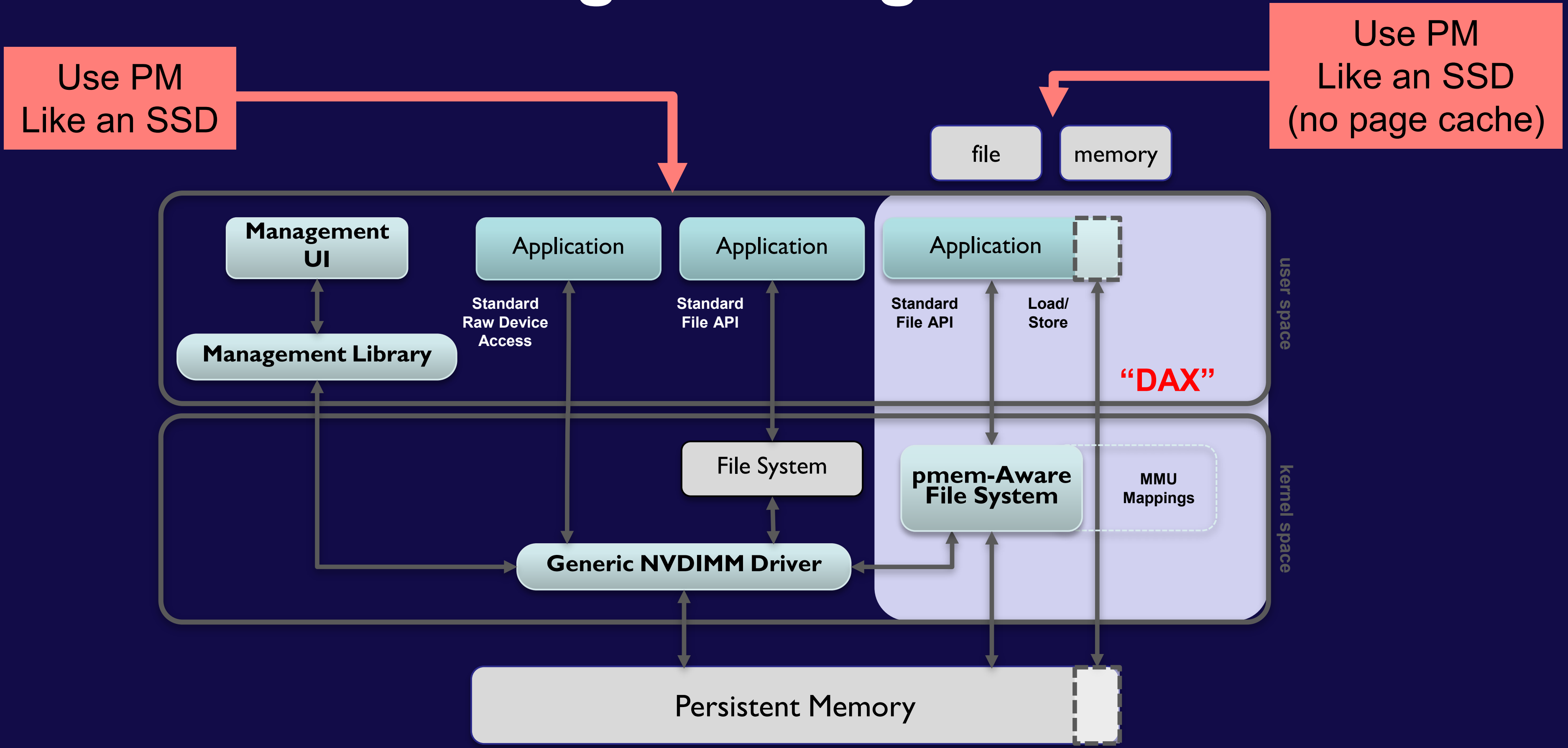


The Programming Model

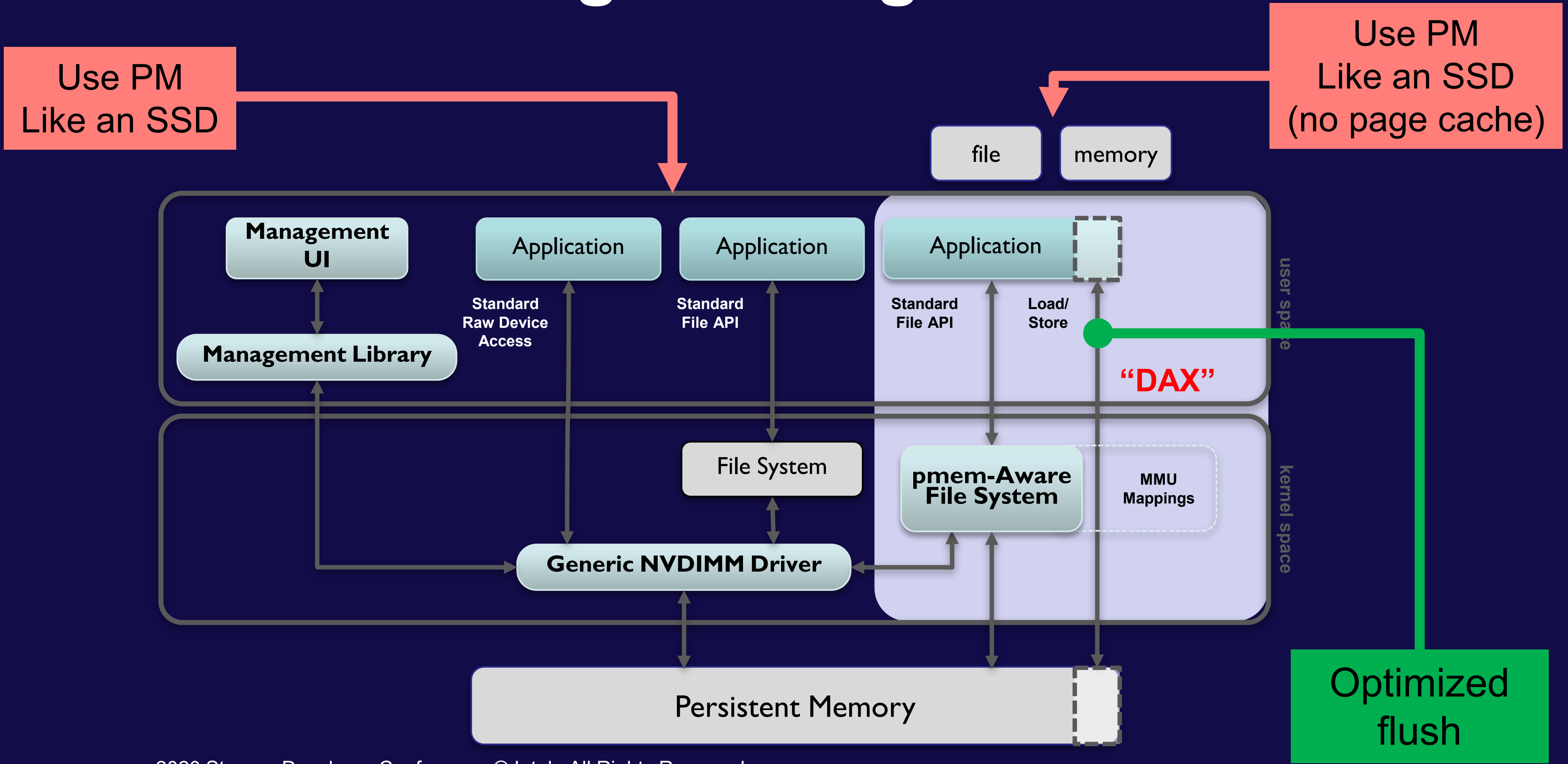
Use PM Like an SSD



The Programming Model



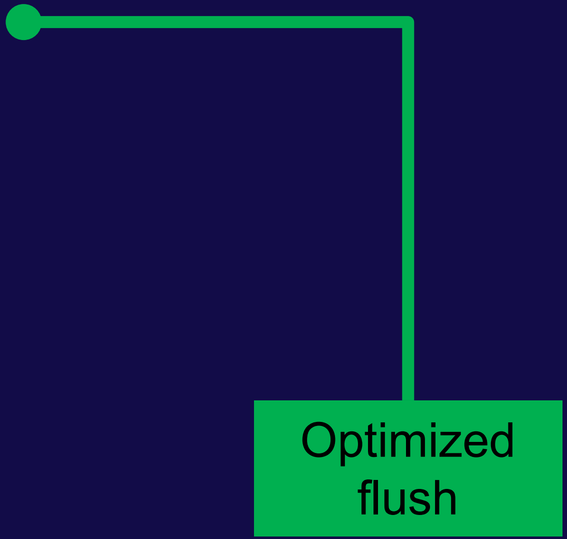
The Programming Model



2020 Storage Developer Conference. © Intel. All Rights Reserved.

Flushing…

- Flushing is painful
  - Error prone
- Flushing is not new
  - POSIX requires it
- Can we get rid of flushing?
  - Maybe sometimes…

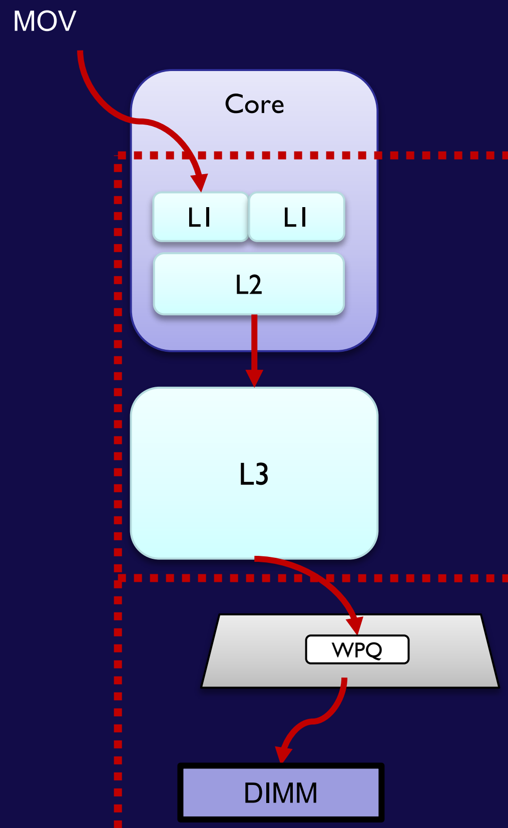


Optimized flush

With my Intel Hat On…

- What is Intel® Optane™ PMem?
  - Byte addressable persistence
  - Performance in ns
- How does pmem work on Intel platforms?
  - Plugs into the memory bus
  - Cache coherent

PMem on Intel Hardware



CPU CACHES

CLWB + fence
-or-
CLFLUSHOPT + fence
-or-
CLFLUSH
-or-
NT stores + fence
-or-
WBINVD (kernel only)

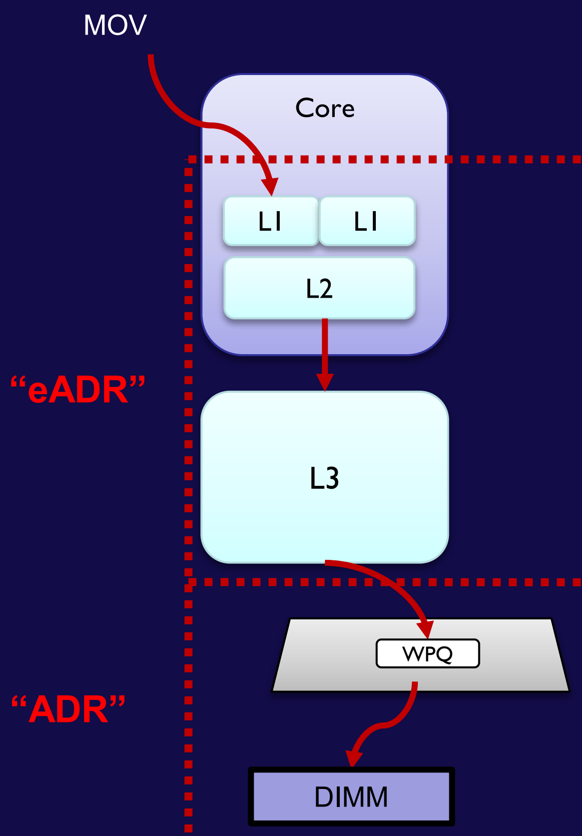
Custom
Power fail protected domain
indicated by ACPI property:
CPU Cache Hierarchy

WPQ

ADR
-or-
WPQ Flush (kernel only)

Minimum Required
Power fail protected domain:
Memory subsystem

PMem on Intel Hardware



CPU CACHES

CLWB + fence
-or-
CLFLUSHOPT + fence
-or-
CLFLUSH
-or-
NT stores + fence
-or-
WBINVD (kernel only)

Custom
Power fail protected domain
indicated by ACPI property:
CPU Cache Hierarchy

WPQ

ADR
-or-
WPQ Flush (kernel only)

Minimum Required
Power fail protected domain:
Memory subsystem

The next level down…

(platform)

- ACPI
  - NFIT reports all pmem installed
  - NFIT says if CPU caches are auto-flushed
- OS abstracts this info away
  - Applications don’t parse ACPI/NFIT
  - Applications consume the abstractions

Fully Leveraging PMem

Lots of Ways to Use PMem

No App Flushing	App-Manages Flushing
Transparent Volatile Memory	Persistent data structures in memory-mapped pmem, accessed directly via loads and stores
Volatile use of pmem	
Storage (App may flush page cache, but not stores to pmem directly)	
?	

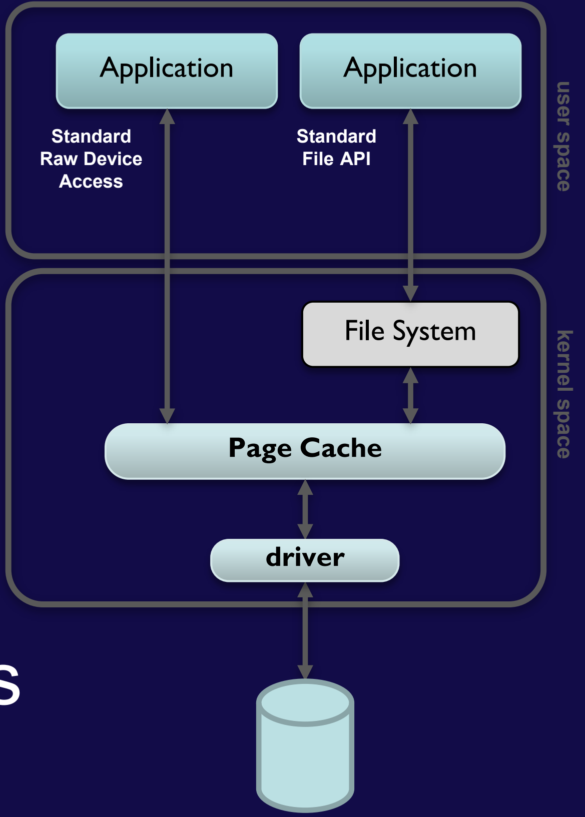
The Benefit of Fine-Grained Persistence

- Saved bandwidth
  - Modify a byte on storage:
    - Read 4k, change byte, write 4k
  - Modify a byte on pmem:
    - Store byte (HW: read 64B, change byte, write 64B)
- Transactions
  - Like storage, but fine-grained updates

What is Flush-on-Fail?

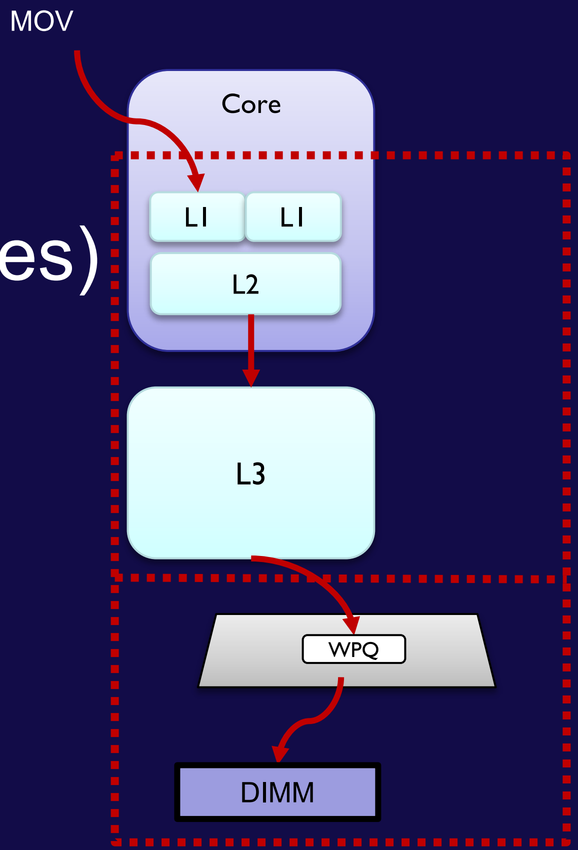
Flush-on-Fail is Not New

- Storage write caches
  - Best effort flush on fail
- SSDs
  - Write buffer
- NVDIMMs
  - Copy to flash on power loss



Why Aren't CPU Caches Always Persistent?

- Stored Energy Requirement
  - Power cores (execute flushes)
  - Power memory
- Platform support
  - More than a capacitor
- Cost versus Benefit
  - Cost of battery vs perf gain



In a World…

Where CPU Caches Are Persistent

Visibility vs Persistence

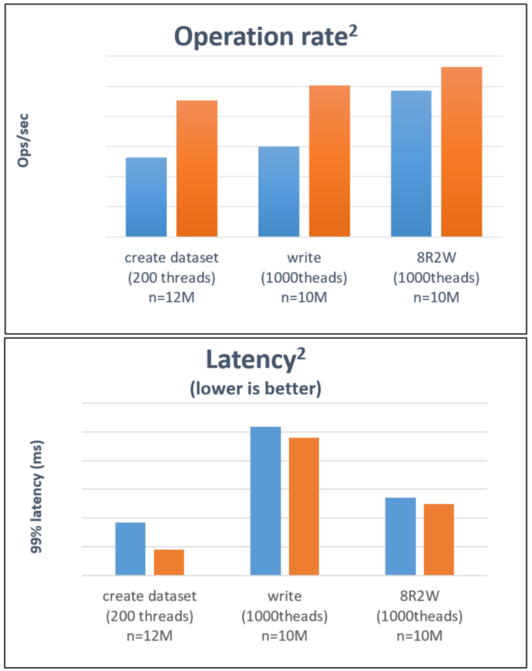
Visibility != Persistence	Visibility == Persistence
MOV X, 10 MOV Y, 20 … MOV eax, X *visible* … CLWB X CLWB Y … SFENCE *persistent*	MOV X, 10 MOV Y, 20 … MOV eax, X *visible (persistent?)* … SFENCE *persistent*

When is this actually needed?

Performance Benefit

- Modified Cassandra[1] for PMem
- Ran with and without eADR
    - PMDK supports this
    - Actual eADR not required

Projections with eADR on Cassandra



[1] The public version of Cassandra was not used here – instead a modified version for PMem was used to collect these projected results.

[2] Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Performance projections are based on testing as of Feb 11, 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product or component can be absolutely secure. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit http://www.intel.com/benchmarks .

Non-Blocking Algorithms

("lock free")

- Compare and swap
  - LOCK CMPXCHG

```
ATOMIC CAS(ptr, old, new) {
        val = *ptr
        if (val == old)
                *ptr = new;
        return val;
}
```

- There's no atomic compare/exchange/flush
  - Pmem version of algorithm is different
  - Example: flush-on-read
    - Performance overhead, especially w/invalidate

Restricted Transactional Memory (RTM)

- Instead of taking lock…
  - XBEGIN/XEND
  - Optimistic locking
  - On transaction abort, XABORT
    - Fall back to traditional locking code
- Cache flush always causes XABORT

When Visibility == Persistence

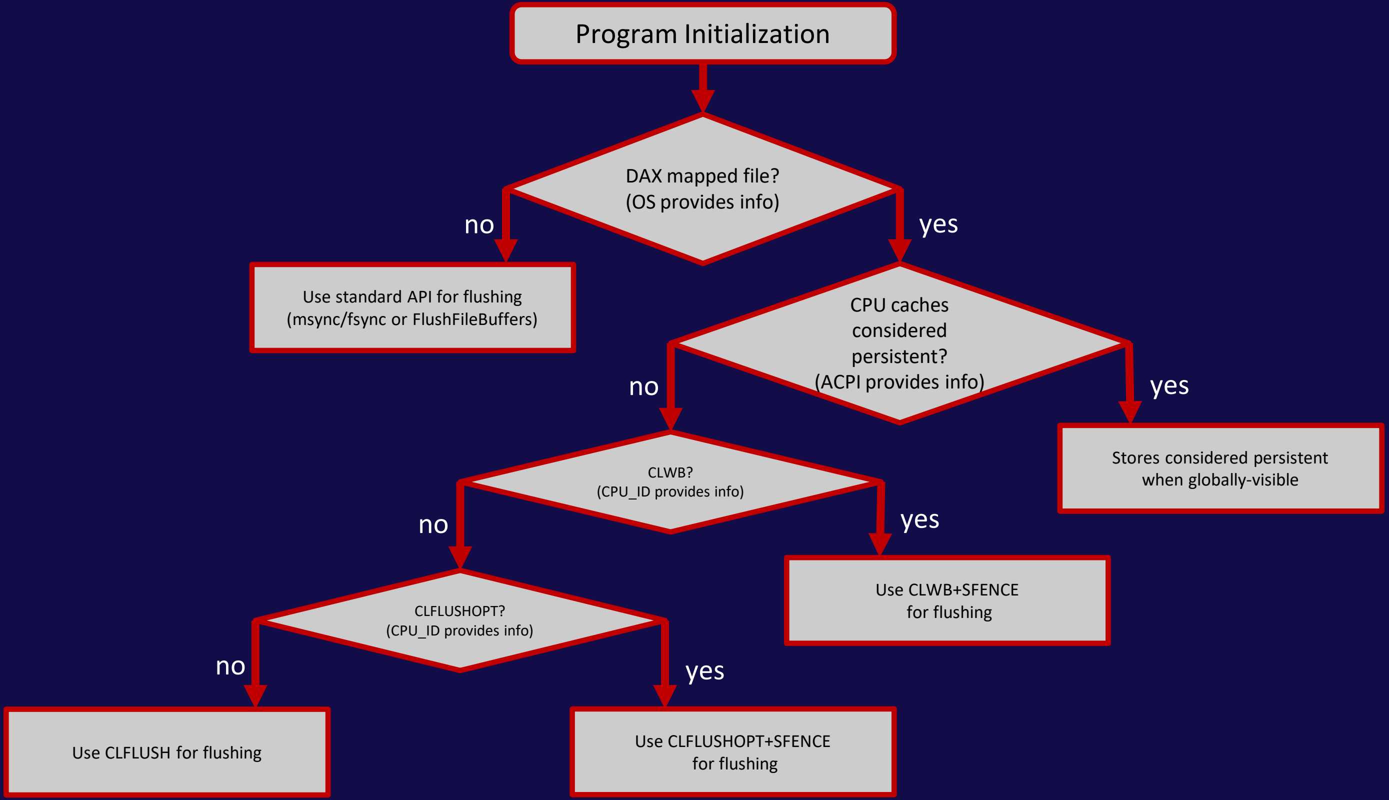
- `LOCK CMPXCHG` works as expected
  - Doesn’t solve other volatile assumptions in code
- `XBEGIN/XEND` can work
  - But XABORT still falls back to traditional locks
  - Locks in pmem require special handling

The Inconvenient Truth

- Most code is riddled with Volatile memory assumptions
  - Examples: memory allocator, garbage collector
- Persisting memory doesn’t persist thread state
  - Instruction pointer is an important part of lock state!
- Platforms that require flushing will exist for a long time
  - App could check for persistent caches and bail out
    - Reducing the usefulness of pmem for that app

- eADR means better performance, but not simpler code

Application Responsibilities

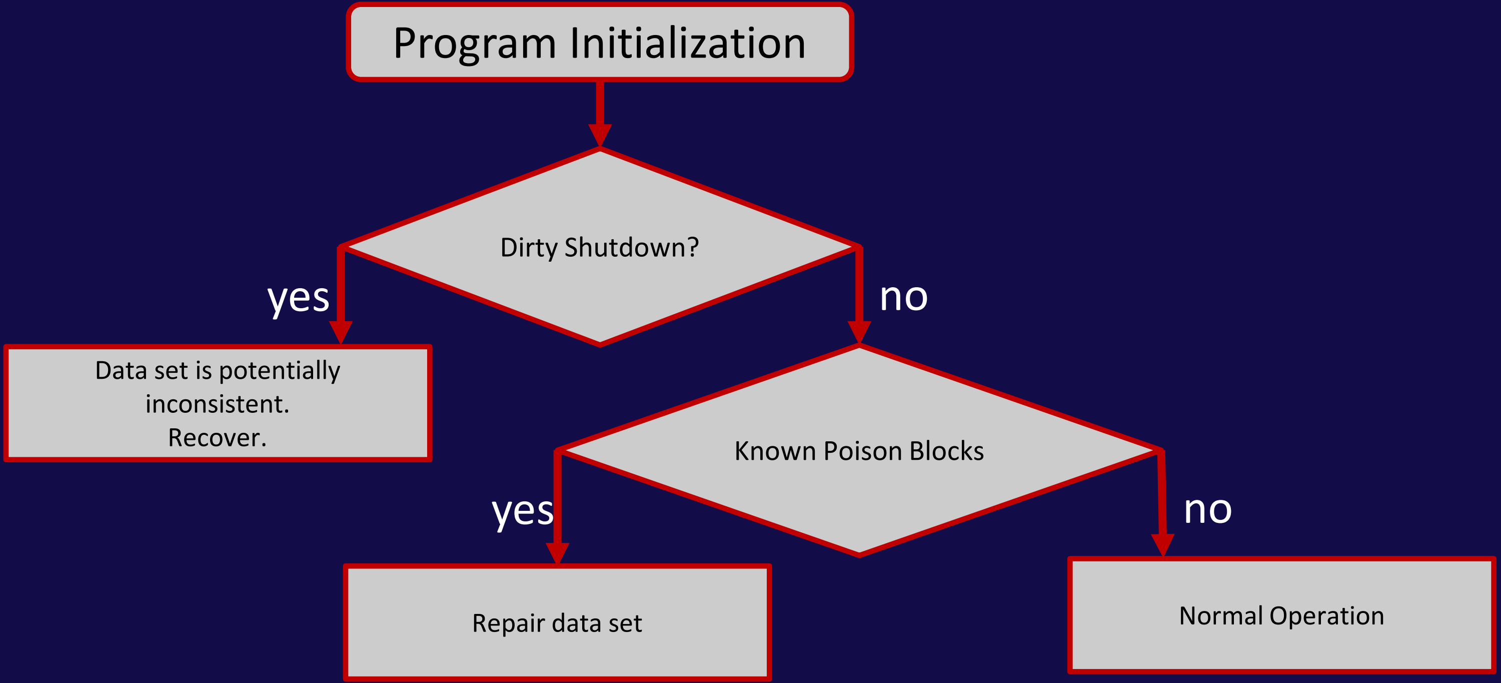


When Flush-on-Fail Fails

The Dirty Shutdown Count

- Programming model includes this idea
  - ADR failure => Dirty Shutdown
- eADR does not introduce new mechanism
  - ADR failure, eADR failure, same to SW

Application Responsibilities



Gaining Trust in the Ecosystem

- The Good News
  - Dirty shutdowns are rare
  - Think of them as device failures
    - How often do you replace a failed DIMM?
- The Bad News
  - Think of them as device failures
    - Restore data from backup/redundant copy

- The success of eADR is tied to gaining trust in it

Making PMem Programming Easier

Easier Programming

- PMDK
  - Already comprehends persistent CPU caches
  - Removes flushes when possible
- The Book
  - (see http://pmem.io)

- Program for persistent CPU caches now

The No-Powerfail Environment

Instead of a Battery
Can We Use a UPS?

- Instead of surprise power loss
  - UPS tells system to shutdown
  - All shutdowns are normal, as far as pmem
- Issue: The BIOS reports persistent CPU caches
  - It knows the platform has eADR
  - It doesn’t know if the system has a UPS
  - It doesn’t know how loaded a UPS is

The Modern “UPS”

- Datacenter Wide
  - Unless you’re in a doctor’s office, servers don’t have a UPS anymore, they have datacenter power
- All shutdowns are orderly shutdowns
  - Except when they aren’t

Summary

Summary

- “Persistent Memory Machines are Coming!”
  - Available for quite a while now
- “ISVs are Adapting to pmem!”
  - Large number have, libraries like PMDK help
- “Persistent CPU Caches are coming!”
  - Follow the programming model to benefit

Please take a moment
to rate this session.

Your feedback matters to us.