PMDK - State of the Project

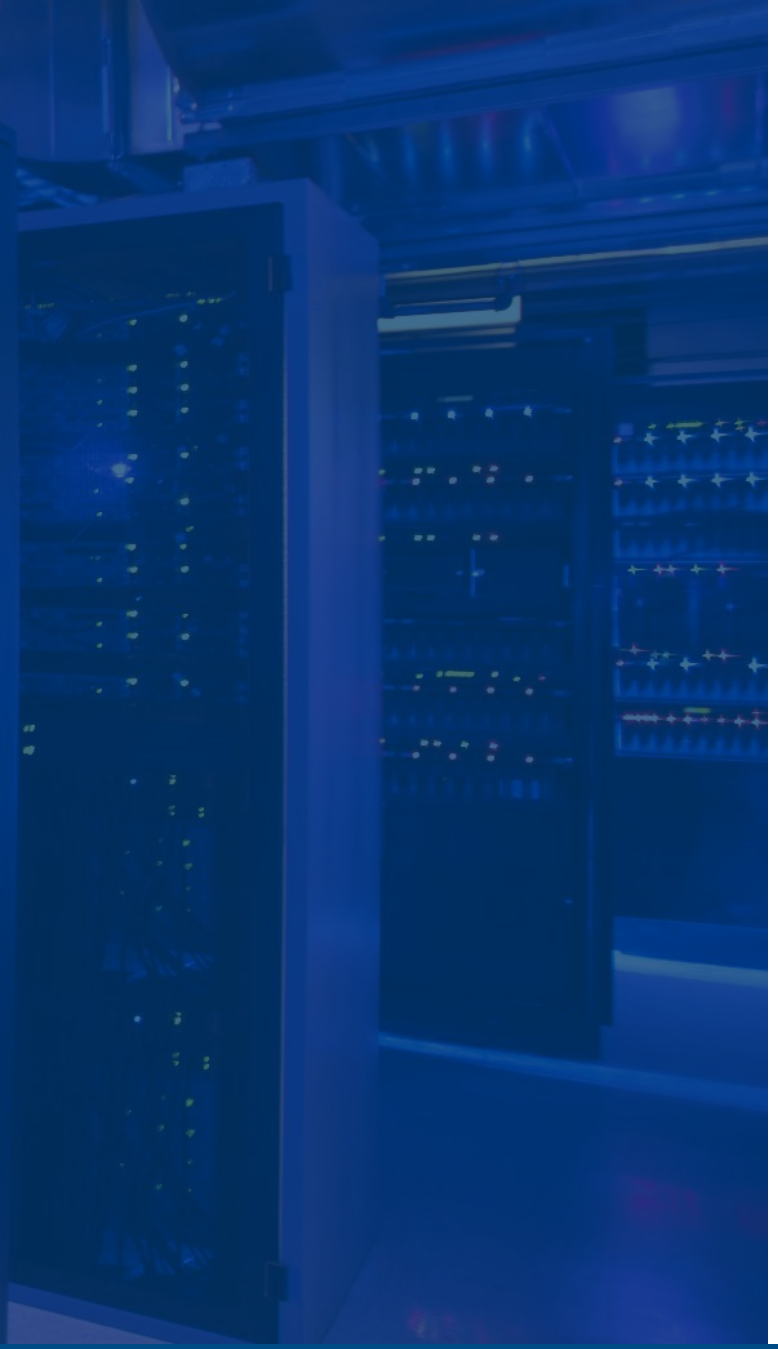
Persistent Memory Development Kit

Andy Rudoff

Senior Principal Engineer
Intel

Piotr Balcer

Software Engineer
Intel



Agenda

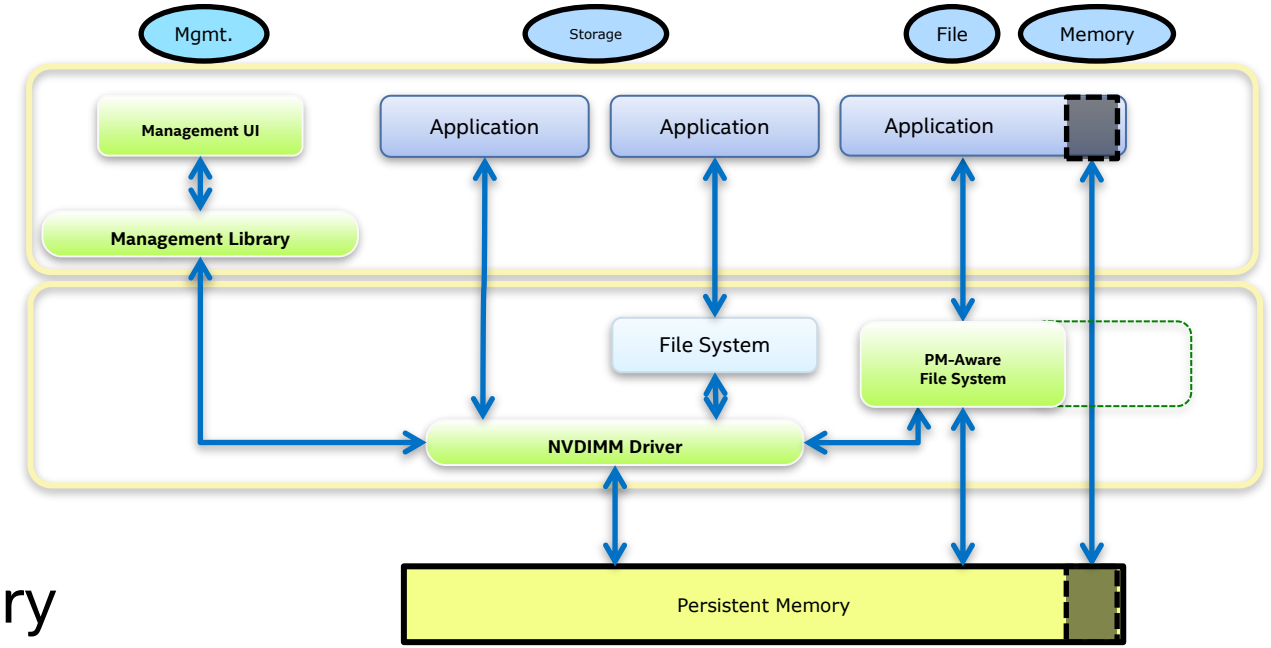
1. **Brief Historical Overview**
   How we came to be
2. **Directions and goals of PMDK**
   What are we doing and why
3. **Current state of the project**
   What have we done so far
4. **A look into the future**
   What are we doing next

01

Brief historical overview

The Persistent Memory Programming Model

- Exposing pmem to applications
  - How to name it, re-attach to it
  - How to enforce permissions
  - How to back it up, manage it
  - The short answer:
    - Access pmem as files
    - Standard file APIs work, memory mapping gives direct access
    - Vendor-neutral
- Published by SNIA NVM Programming TWG



Ancient History

- June 2012
  - Formed the NVM Programming TWG
  - Immediate participation from key OSVs, ISVs, IHVs
- January 2013
  - Held the first PM Summit (actually called “NVM Summit”)
- July 2013
  - Created first GitHub thought experiments (“linux-examples”)
- January 2014
  - TWG published rev 1.0 of the NVM Programming Model

Building on the SNIA PMem Programming Model

Open a pmem file on a pmem-aware file system

Map it into your address space

Okay, you've got a pointer to 3TB of memory, have fun!

- The model is necessary, but not sufficient for an easy to program resource

Gathering requirements yielded fairly obvious top priorities:

- Need a way to track pmem allocations (like malloc/free, but pmem-aware)
- Need a way to make transactional updates
- Need a library of pmem-aware containers: lists, queues, etc.
- Need to make pmem programming not so error-prone

02

Direction and goals of PMDK

Solving real problems using persistent memory

PMem is multidimensional. It's both memory and storage.

- As memory, it's more affordable and bigger than DRAM.
  - Enabling previously impossible (or impossibly expensive) use-cases on multi-terabyte heterogenous memory systems.
  - As storage, it's an order of magnitude faster compared to other solutions.
- Enabling ultra-low latency retrievals and transactions, potentially also reducing overall memory cost by bypassing the cache.
  - As both, it's unique.
  - Enabling new designs that require new unique solutions.

Persistent Memory as Memory

- Persistent Memory is bigger, but slower than DRAM.
- PMem is only one of the different kinds of memory that can be present in a heterogeneous memory system.
  - Applications typically assume that all memory is the same.
    - Hardware or the OS can be made to emulate this status quo (Memory Mode, Memory Tiering).
  - … but, even today, that's simply not the case.
    - NUMA, High-Bandwidth Memory, PMEM and more.

PMDK helps applications with intelligent and scalable memory placement.

Persistent Memory as Storage

- Persistent Memory is smaller, but faster than traditional storage.
  - This is not unprecedented. SSDs were a similar disruption.
  - Techniques developed then, make sense now.
    - Storage caching & tiering, separating data from write-ahead logs, …
- Thanks to DAX, Persistent Memory can also reduce the reliance on page cache in applications that use memory-mapped I/O.
  - This reduces cost and guarantees stable latency unhindered by page faults.

PMDK helps developers to modify existing storage solutions.

Persistent Memory as both Memory and Storage

- Database storage engine design is essentially a study on how to mask the large difference between storage and memory.
  - We don't have to do that any more… sort of :)
- Persistent Memory is a new tier that bridges the gap between Memory and Storage.
  - Enables new techniques that reduce access latency and write amplification.
    - Fault tolerant algorithms still need to log data but can now do so using a single load/store instructions at cacheline granularity.

PMDK helps developers use novel techniques that merge memory and storage.

General Directions and Goals

“Make easy things easy and hard things possible”

- Larry Wall, about Perl programming language.

- PMDKs goal was, is, and always will be making Persistent Memory programming easy.
- But also enable solving complex and possibly challenging problems commonly encountered by users.
  - This is done through a multi-layered stack of solutions, with each building block adding new functionality on top of the previous one.
  - Applications can choose their desired level of abstraction.

03

Current state of the project

Persistent Memory Development Kit

A diverse stack of solutions

volatile use cases

libvmemcache

Space-efficient scalable memory-oriented embeddable caching solutions.

libmemkind

Memory allocator with specialized per-kind heap allocation capabilities (PMem, DRAM);
Now with APIs for heterogeneous systems.

remote use cases

librpma

Easy to use library for remote memory access over RDMA; Exposes PMem specific primitives.

persistent use cases

pmemkv-java | pmemkv-python | pmemkv-js | pmemkv-…

libpmemkv

Persistent Memory key-value store; Easy to get started with.

libpmemobj-cpp

C++ bindings for libpmemobj and PMem-STL; The easiest and most idiomatic way to write persistent applications.

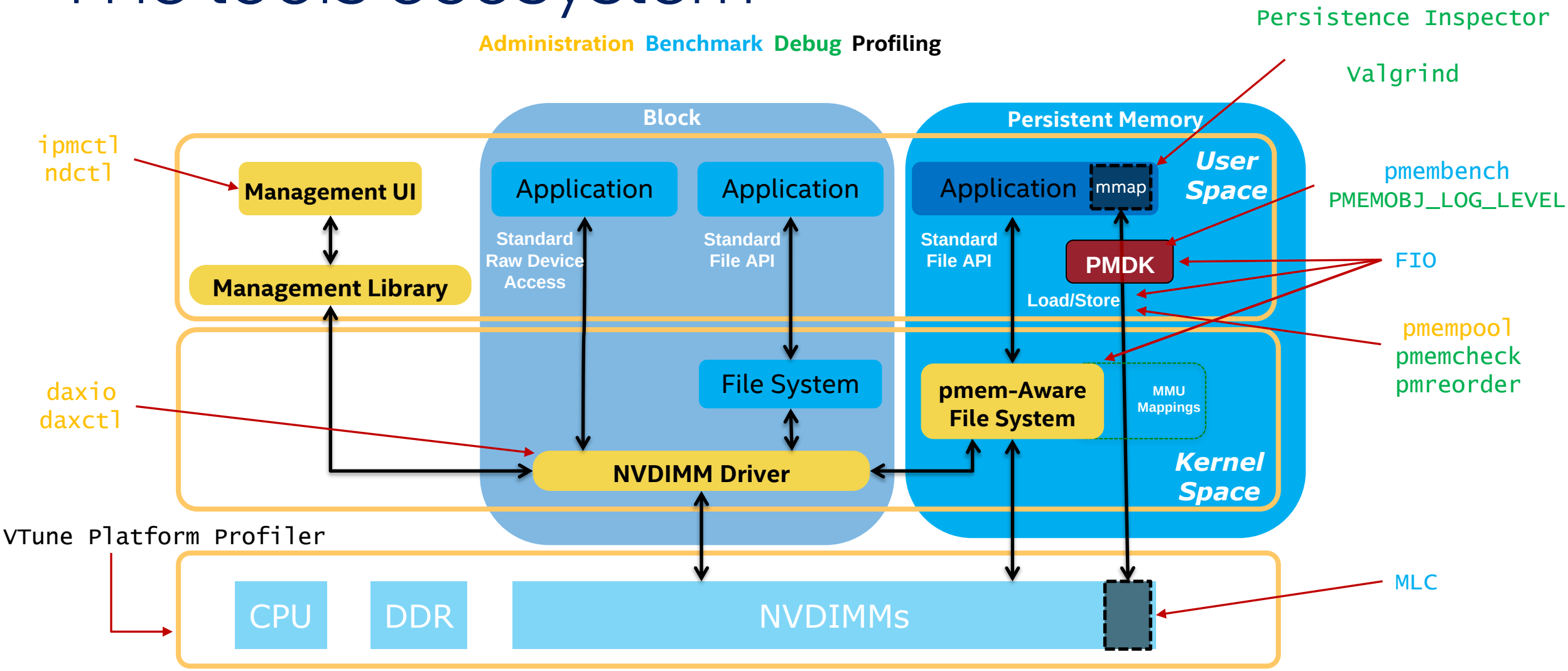
libpmemobj

General purpose transactional object store; Provides memory allocation and transactions needed for complex logic.

libpmem2

Essentials; low-level API that provides an abstraction for all the necessary primitives an application needs to use PMem.

The tools ecosystem



04

A look into the future

Heterogeneity & Ease of Use

- Existing PMDK solutions primarily focus on exposing Persistent Memory. ~~Obviously…~~
- But PMem doesn't exist in a vacuum. It's usually a component in a larger diverse system and has to co-exist with dedicated memory & storage devices like DRAM or NVMe SSDs.
- Leaving it up to the application developer to integrate all of that allows for greater flexibility but makes it hard to create comprehensive solutions.

- We've now started to work on solutions that give developers choice with regards to the level of integration they want, "making easy things easy and difficult possible."

Pmemkv - Hybrid Engine

- Existing pmemkv engines primarily use PMem for both metadata and data storage.
- This has the benefit of ensuring strong consistency but comes at a cost of higher latency for most operations.

- We are now working on hybrid pmemkv engine that will take advantage of DRAM to store some of its structures.
- This will give users the choice between higher performance and lower main memory consumption.

Memkind - Memory Allocation Tiering

- New versions of memkind now enable developers to allocate memory based on its properties (latency, capacity, bandwidth) rather than some explicit name (DRAM, PMEM, HBM).
- But those allocation calls still need to be managed explicitly, introducing some complexity.

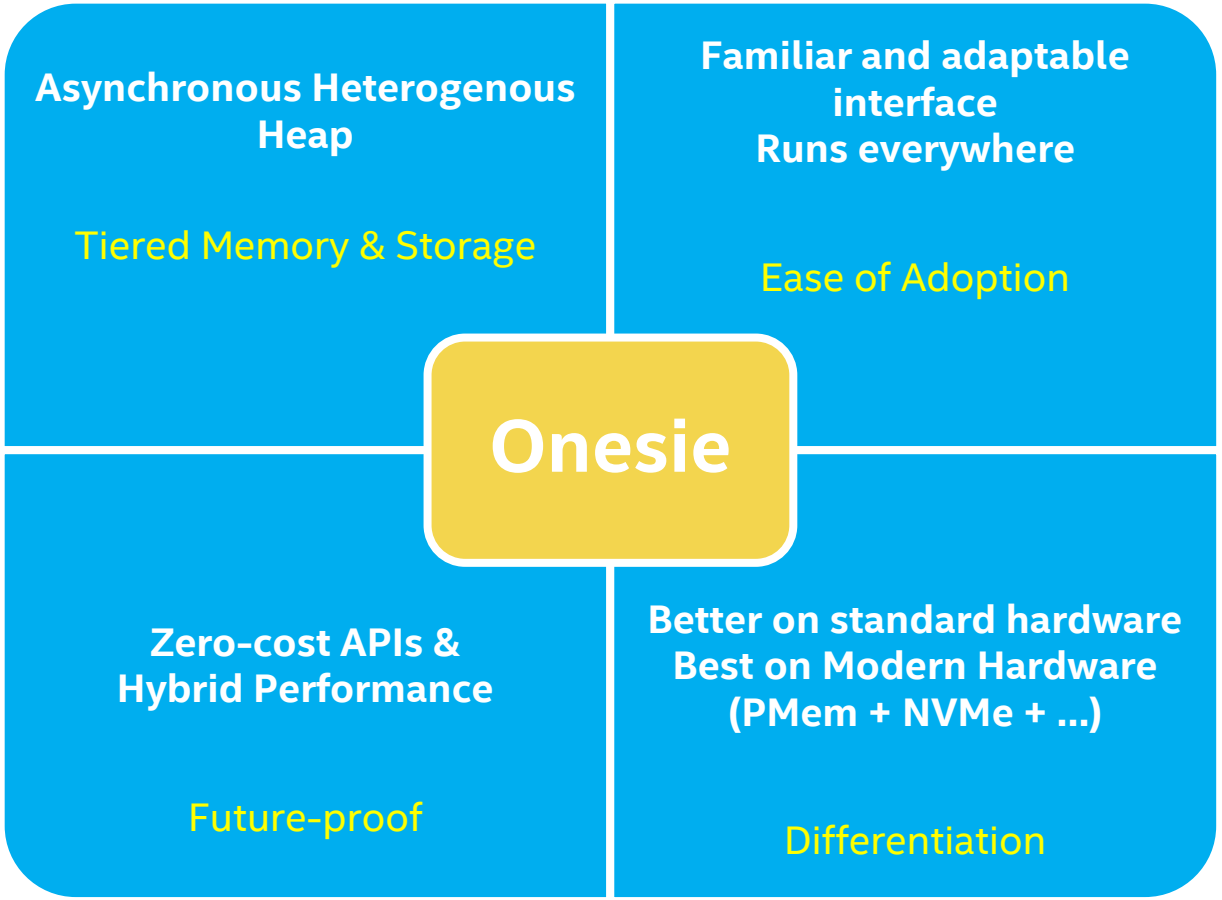
- We are now working on a library for automatic memory tiering at the memory allocation level. This software will be loadable like any other memory allocator, transparently managing memory at user-space level.
  - Initially, the heuristics to choose between different types of memory will be simple and based on allocation size.
  - But our intention is to research the possibility of leveraging malloc call stack as well as runtime profiling for better data placement decisions.

Onesie - scalable data solution

Heterogeneous Heap with Log-Structured Asynchronous Transactions

- Embedded transactional storage engine backend
  - Written in Rust, with bindings for C and C++
- Designed for modern memory and storage technologies...
  (PMem, NVMe, CXL, HBM, DSA, RDMA, ZNS)
- ... but usable on ordinary hardware with minimal dependencies.

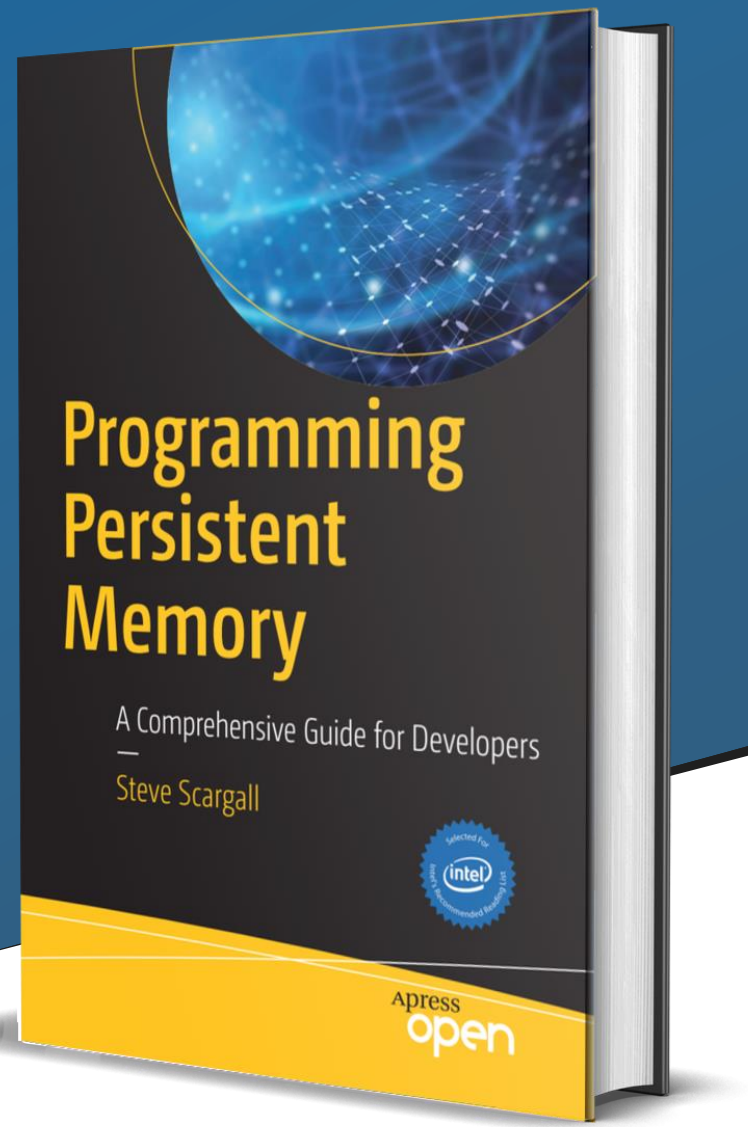
```
rt.block_on(onesie.run(|tx| async move {
    let root_handle = tx.root::<MyIndex>()?;
    let root = tx.read(root_handle).await?;
    root.idx.insert(&tx, “Hello”,
                    “PMem”).await?;
    Ok(())
}))?;
```



Master Persistent Memory Programming

Are you ready to begin?

https://pmem.io/book



Call to Action

- "Solving real problems using persistent memory"
  - Do you have a real problem that Persistent Memory can help solve?
    - Great! Get involved and tell us about it.
- Do you think this is an interesting research opportunity?
  - So do we! Get involved and share your ideas with the community.
- Want to just play around with examples?
  - You can get started right now. No need for real hardware.

https://pmem.io/


intel®
SPDK, PMDK, Intel® Performance Analyzers
Virtual Forum