Experiences Designing a Persistent Memory SDK

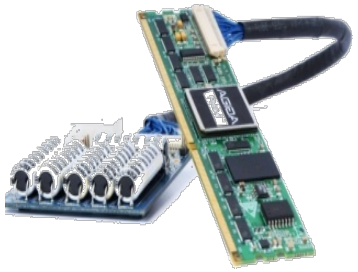
Paul von Behren – Software Architect
Intel Corporation

Agenda

- Overview of Persistent Memory Programming
- Walkthrough of Key Programming Challenges
- Introduction to Persistent Memory APIs
- Example Using Persistent Memory APIs
- Summary

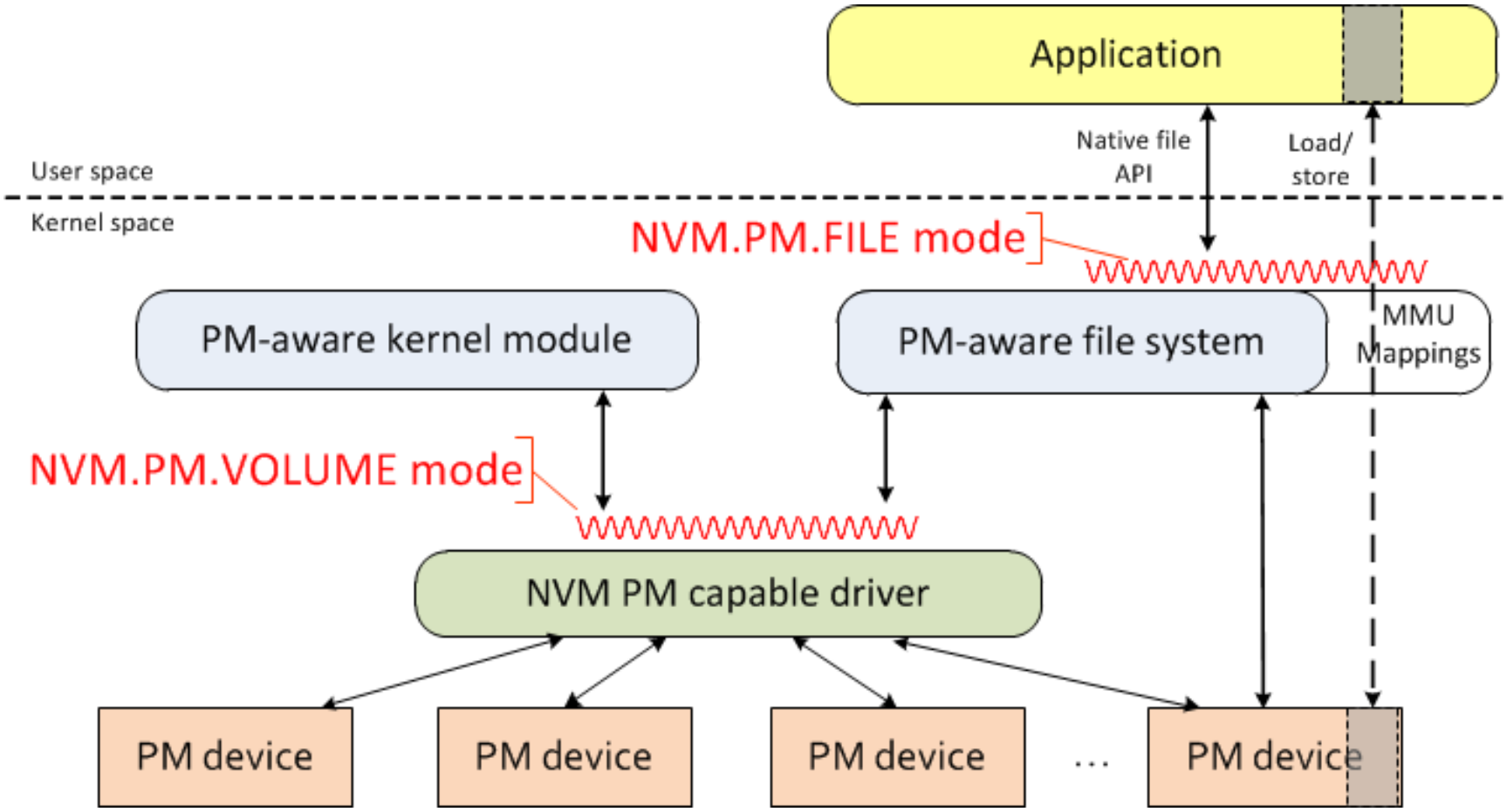
What is Persistent Memory?

- Load/store accessible
  - Would reasonably stall CPU for a load from PMEM
  - No paging (at least not by the OS)
- Not NAND
  - At least not NAND directly
  - Some DRAM-backed-by-NAND variants available today
- In the future, built on emerging NVM technologies
- Other memory ideas work:
  - Cache coherency
  - DMA



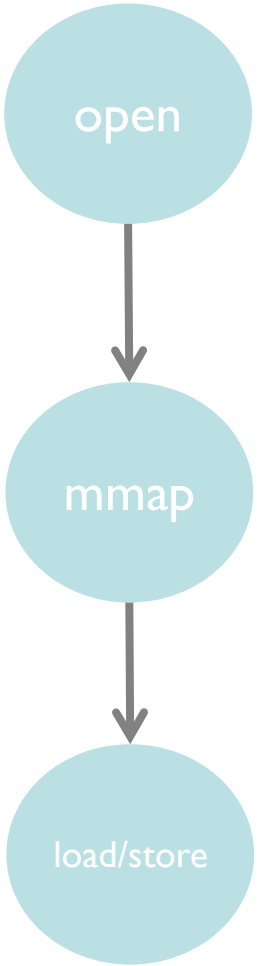
Think "Battery-backed DIMMs"

Software Architecture for Persistent Memory
(from SNIA *NVM Programming Model*)



Why Memory-Mapped Files?

- Naming model is familiar
  - No new namespace – commonality among vendors
- Well-known permission model
  - File system permission checks must pass before use
  - Extended file permissions like ACLs can work
- Ubiquitous file management commands
  - Create/delete/rename works as expected
  - File-based backup can work



Agenda

- Overview of Persistent Memory Programming
- Walkthrough of Key Programming Challenges
- Introduction to Persistent Memory APIs
- Example Using Persistent Memory APIs
- Summary

Programming with Memory-Mapped Files

- Memory-mapped files: 30-year-old interface!
  - Mature, well worn interface
  - Some central mechanisms use it, like shared libraries
- Once Persistent Memory is mapped, programmers want:
  - An allocator
    - Like malloc() in C, new in C++, etc.
  - Transactions
    - How to keep data structures consistent across power failure
- Making changes persistent
  - msync() works as expected
  - **New instructions available for persistent memory**

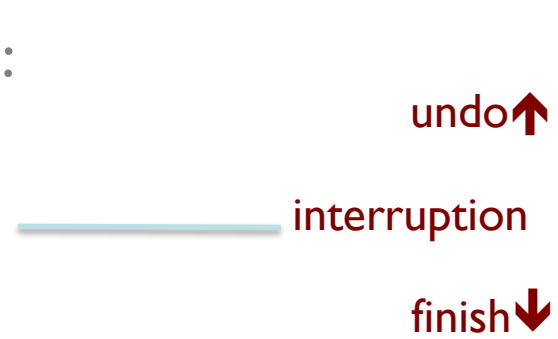
Memory-mapped files provides the RAW access to applications

Powerfail Safe Data Structures

- Start with a linked-list data structure:

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

- Traditional memory allocation steps:
  - Reserved the memory
  - Fill it in (prepare for use)
  - Link it in

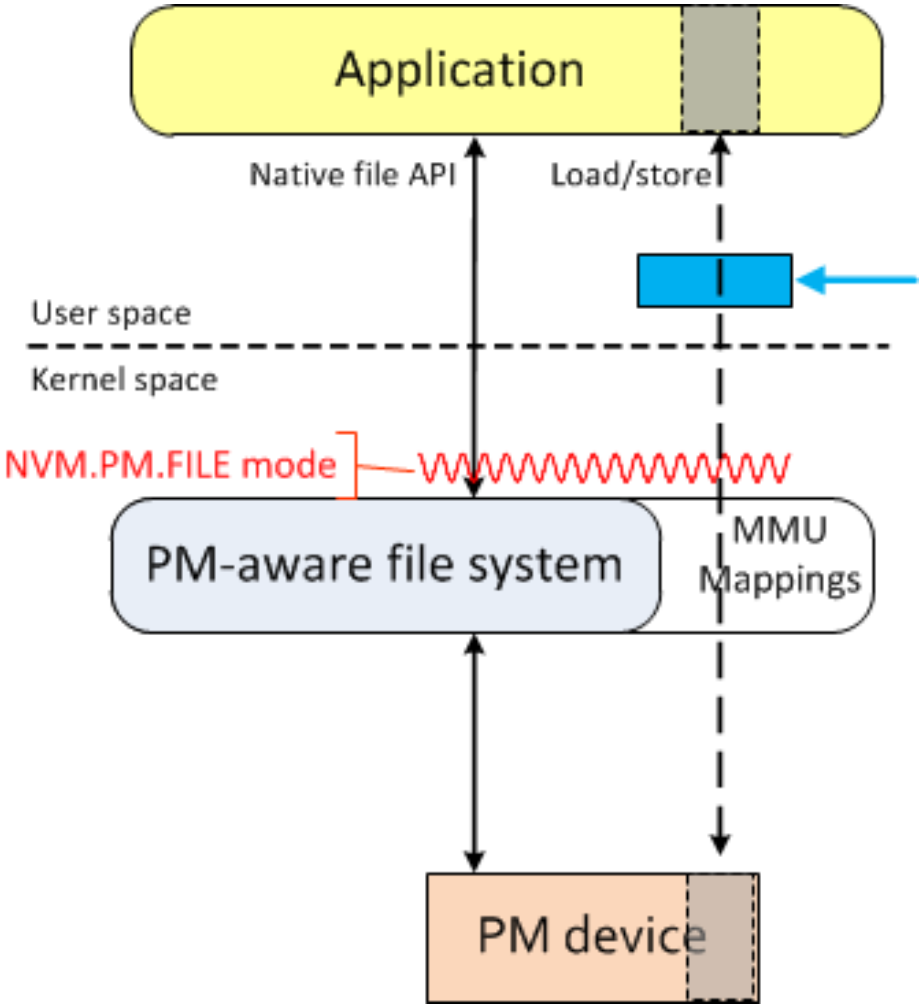


Agenda

- Overview of Persistent Memory Programming
- Walkthrough of Key Programming Challenges
- Introduction to Persistent Memory APIs
- Example Using Persistent Memory APIs
- Summary

Where Do Persistent Memory APIs Live?

- The *NVM Library* builds on the raw mmap API
- The library is a convenience, not a requirement
- The library is open source, evolving with community involvement
- See the library on github at http://pmem.io/nvml



Overview of NVM Library APIs

- PMEM: The basics
  - Flush to persistence
- PMEMTRN: Persistent Memory Transactional
  - Malloc broken into steps to make transactional
  - Interruption-safe transactions for PM
  - NVML routines all start with pmemtrn_ prefix
- PMEMBLK: Persistent Memory carved into blocks
  - Pool is divided up into a specific chunk size
  - Single block writes to the pool are atomic
  - NVML routines all start with pmemblk_ prefix
- VMEM: Volatile Memory Allocator
  - Use PM as volatile memory via malloc/free-like calls
  - Leverage capacity
  - Don't bother flushing for durability
  - Pool "resets" on application restart
  - NVML routines all start with vmem_ prefix
- PMEMLOG: Log file (append-mostly)
  - Common use case, write mostly
  - Append operation very cheap
  - Read through (for log shipping) also optimized
  - NVML routines all start with pmemlog_ prefix

The Basic PMEM Support

```
#include <libpmem.h>

cc ... -lpmem (or -lpmem_debug)

Flush-to-persistence support:

int pmem_is_pmem(void *addr, size_t len);
void pmem_persist(void *addr, size_t len, int flags);
void pmem_flush(void *addr, size_t len, int flags);
void pmem_fence(void);
void pmem_drain(void);
```

On Being a Good Citizen

- The library never:
  - Exits
  - Forks or Joins threads
  - Uses signals
  - Calls select()
- Caller can supply:
  - Custom malloc(), etc.
- Debug version of the library:
  - Traces all calls, errors, lots of details
  - Assertion checking

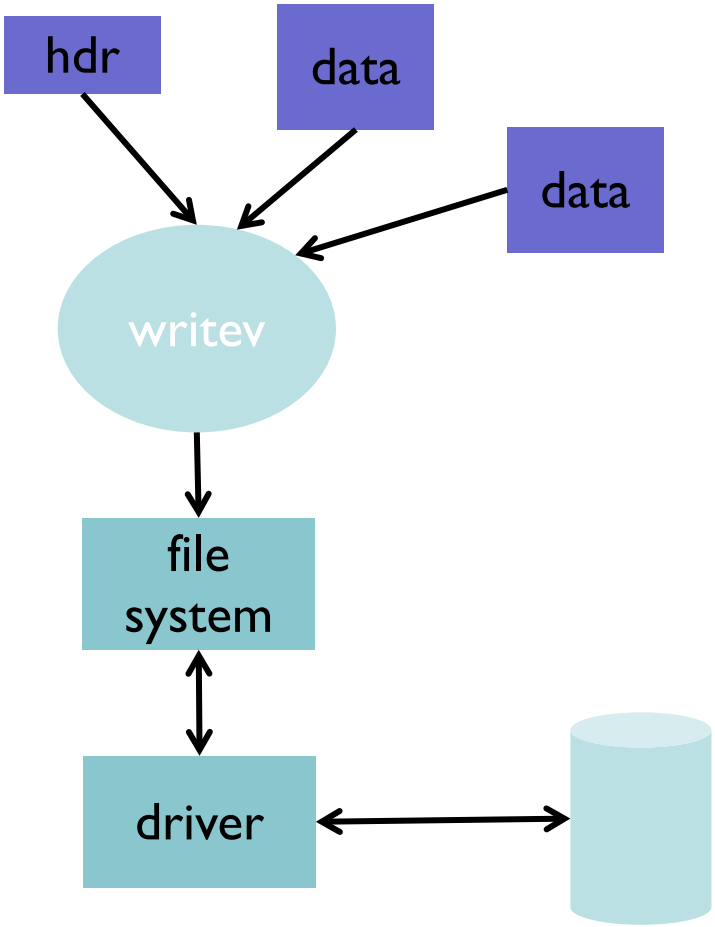
Agenda

- Overview of Persistent Memory Programming
- Walkthrough of Key Programming Challenges
- Introduction to Persistent Memory APIs
- Example Using Persistent Memory APIs
- Summary

Log File Example: Appending a Record

- The writev() system call is often used:
  - `writev(fd, iov, iovcnt)`
  - Handy for grabbing header, data from separate locations in memory
- Not atomic
  - Well, POSIX says “atomic with respect to other reads and writes”
  - Certainly not powerfail atomic
- Fairly long code path
  - Includes file system
  - Potentially multiple trips through the block stack for metadata updates

Appending with writev()



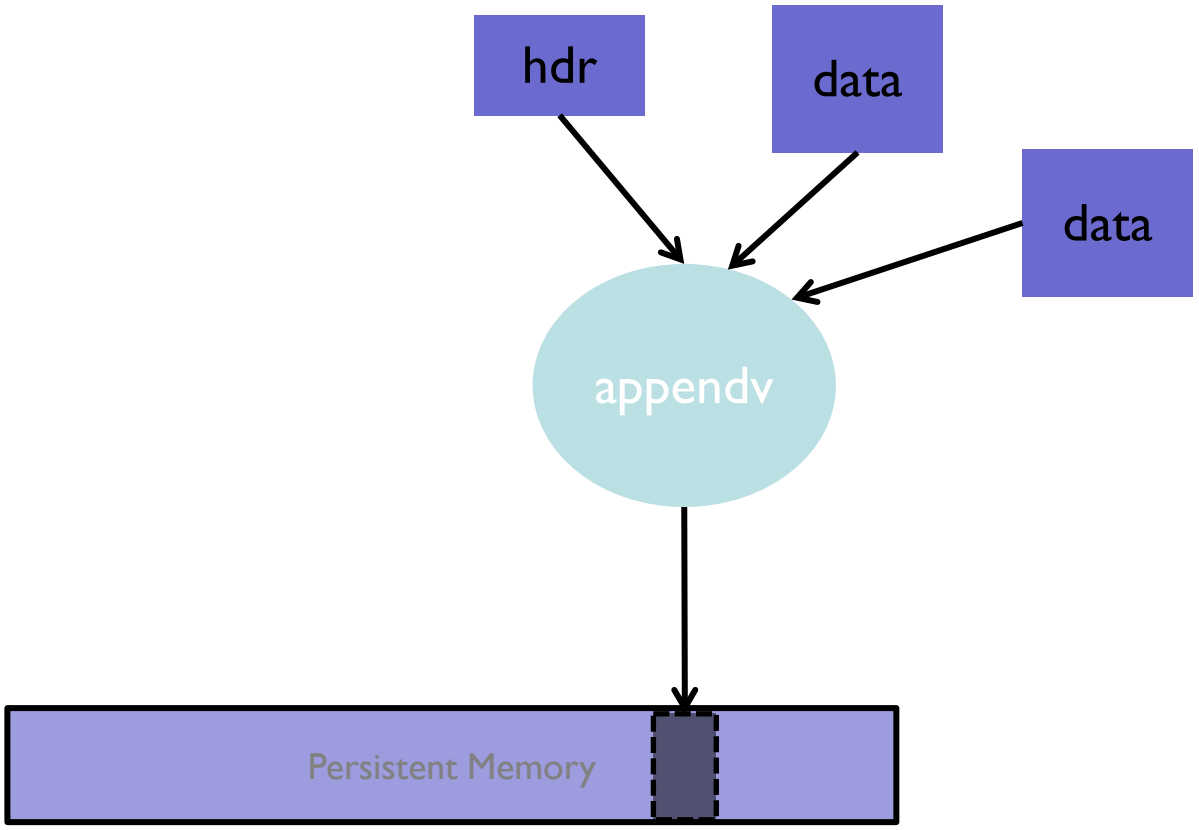
The PMEMlog API

Support for Persistent Memory Logs:

```
PMEMlog *pmemlog_map(int fd);
void pmemlog_unmap(PMEMlog *plp);
size_t pmemlog_nbyte(PMEMlog *plp);
int pmemlog_append(PMEMlog *plp, const void *buf, size_t count);
int pmemlog_appendv(PMEMlog *plp, const struct iovec *iov, int iovcnt);
off_t pmemlog_tell(PMEMlog *plp);
void pmemlog_rewind(PMEMlog *plp);
void pmemlog_walk(PMEMlog *plp, size_t chunksize ,
          int (*process_chunk)(const void *buf, size_t len, void *arg),
          void *arg);
```

Algorithm Converted for Persistent Memory

- `pmemlog_appendv(plp, iov, iovcnt)`



Agenda

- Overview of Persistent Memory Programming
- Walkthrough of Key Programming Challenges
- Introduction to Persistent Memory APIs
- Example Using Persistent Memory APIs
- Summary

Summary

- Persistent Memory is Coming!
- Basic application access is by memory-mapping files
- Hard problems such as transactions are being solved by the NVM Library
- As a community, we all benefit by working towards a full-featured, performance-tuned NVM Library

The NVM Library Enables Persistent Memory Aware Applications

Call to Action

- Try out early versions of the NVM Library
  - Start here: http://pmem.io/nvml

- Contribute to the library!
  - File bugs
  - Request enhancements
  - Help write tutorials and sample programs
  - Help enhance the library

Additional Sources of Information

- Get the latest (work in progress) NVM Library:
  - pmem.io/nvml
  - (redirects to github)
- Participate in community discussion:
  - groups.google.com/group/pmem
- Find us on IRC:
  - #pmem on oftc

Q&A