Accelerating Operations on Persistent Memory Device via Hardware-based Memory Offloading Technique

Ziye Yang, Intel

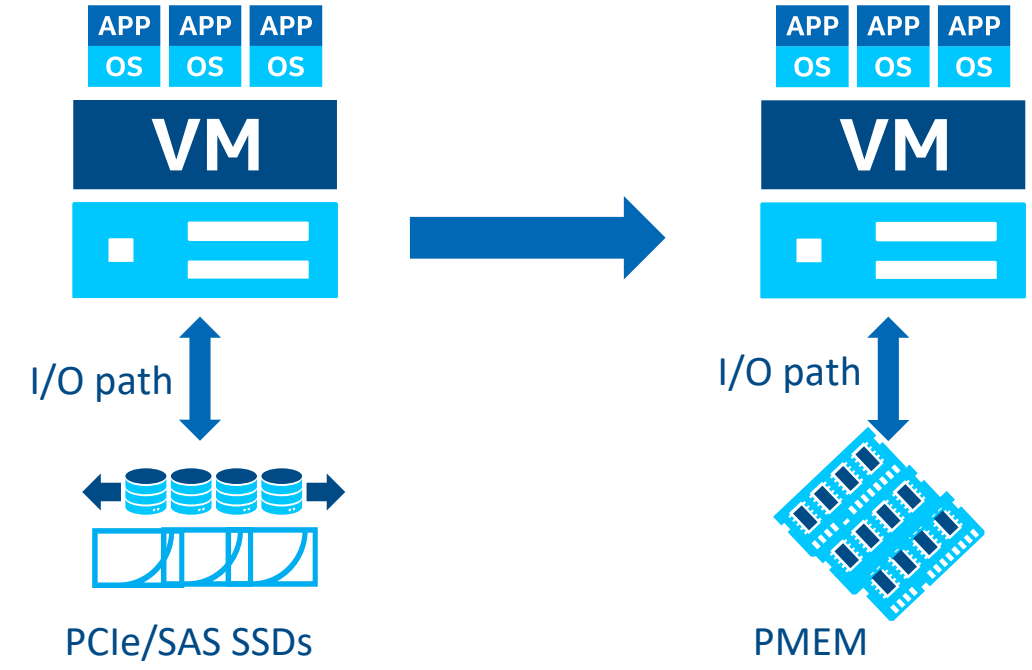
Agenda

- Background & Problem
- Our proposed solution
- Use Case1: Memory offloading technique enabling in SPDK
- Use Case2: Memory offloading technique enabling in Ceph
- Conclusion

Background & Problem

Background & Problem

- With more and more deployment of non-volatile memory devices (e.g., **Intel® Optane™ Persistent Memory**) deployed in datacenter, CPU resource on the host becomes a ***bottleneck*** while operating on PMEM devices with AD (App direct) mode under high workloads instead of using traditional HDDs/SSDs.



Host CPU:
I am **fine** to serve **20** VMs, my brothers have ***DMA ability*** and can help me complete I/Os!

Host CPU:
I can only serve **15** VMs and **very busy**, ***no DMA capability*** is available from PMEM.

Background & Problem (Cont.)

- The root cause:
  - Data operations “between (persistent) Memory” is different from “data operations between Memory and device”. For the latter case, there is (R)DMA capability on device, and CPU will not be the bottleneck.
- So how to offload the work from CPU with offloading techniques while operating on PMEM is a valid requirement.
  - For example, cloud providers would like to utilize the important CPU resources in customer visible workloads (e.g., those host CPUs should be virtualized to launch more customers’ Virtual Machines) instead of using more CPU cycles on PMEM.

Our proposed solution

Our proposed solution

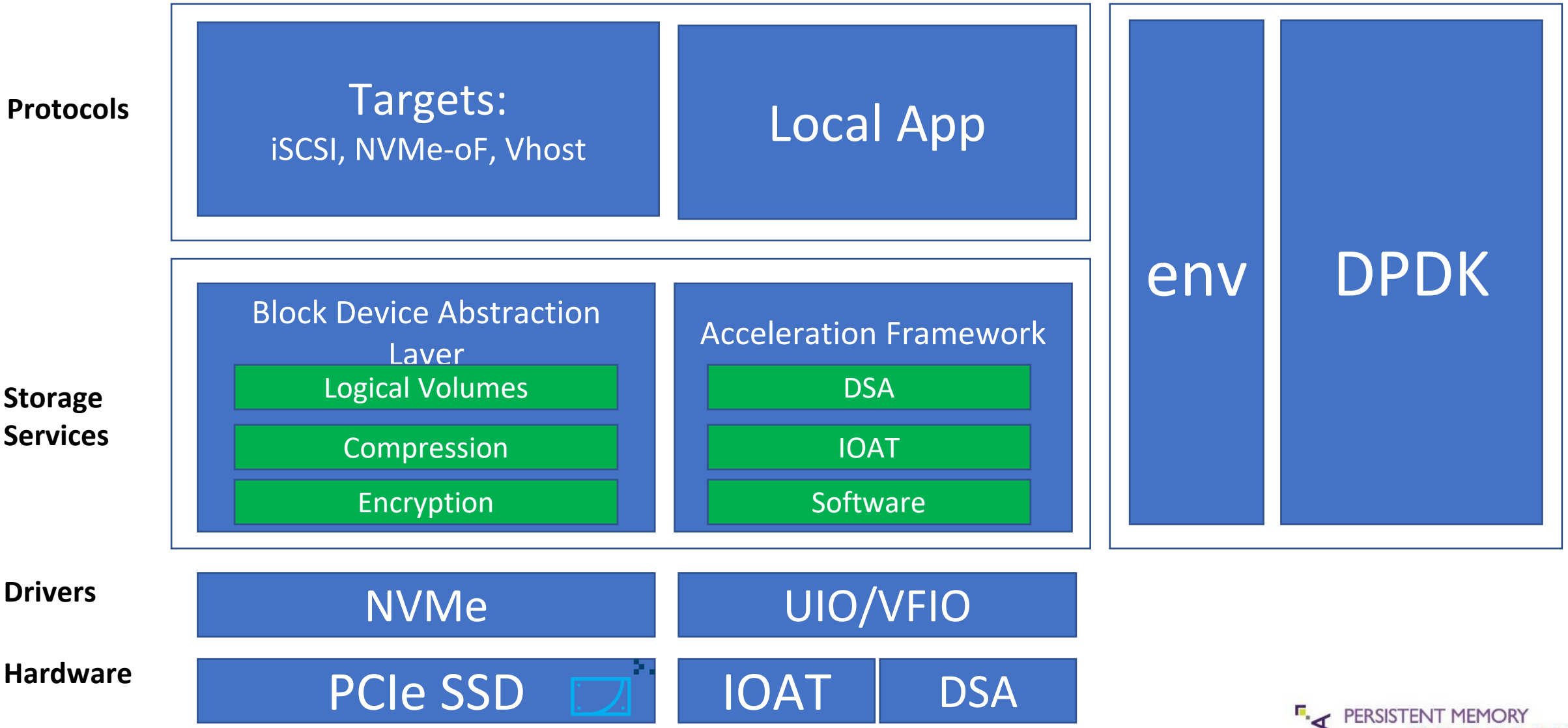
- Solution: Leverage some memory offloading devices (e.g., Intel's IOAT or Data Streaming Accelerator (DSA), or other vendors' techniques)
- Those memory offloading devices should support offloading the following operations on PMEM, i.e.,
  - Memory movement between DRAM and PMEM
- If we select those devices, there should be user-friendly software (in kernel/user space) to support the offloading requirements.

Our proposed solution (cont.)

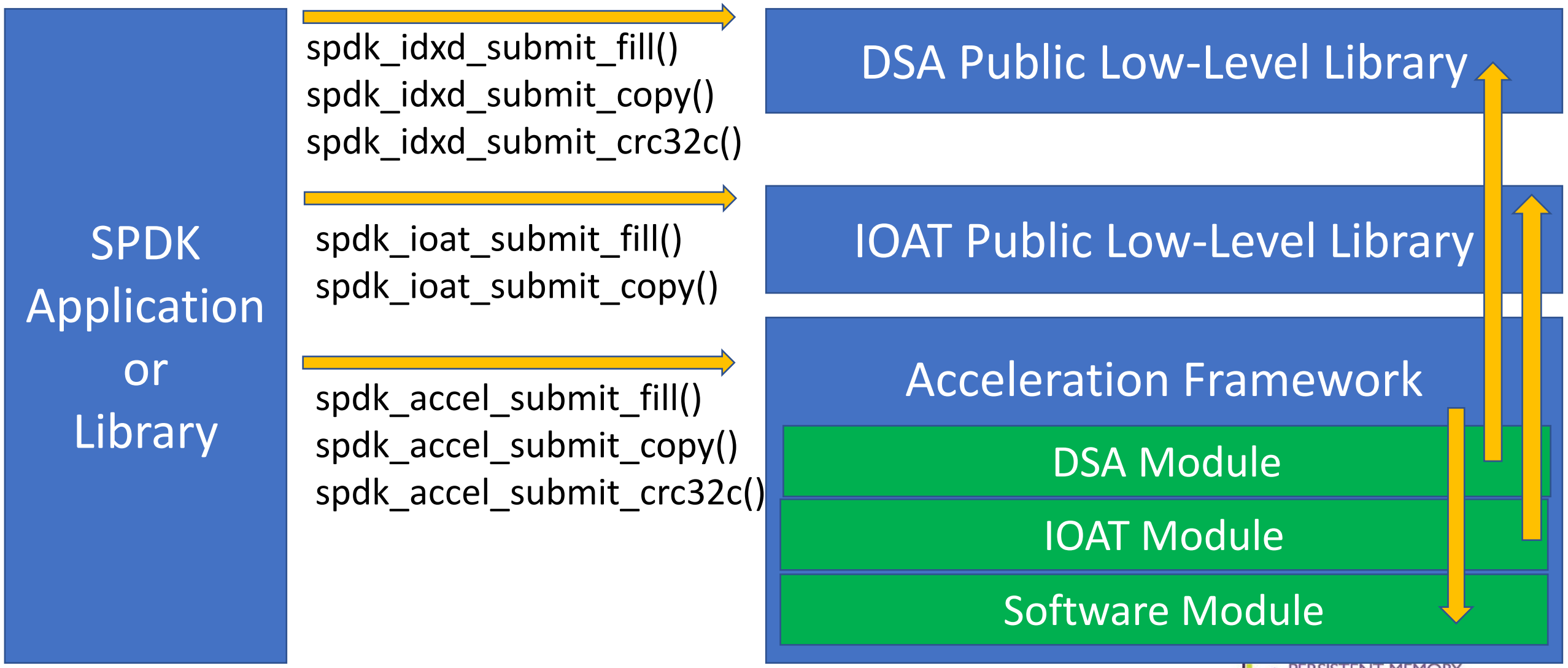
- How to efficiently leverage those memory offloading devices?
- Sync VS Async?
  - Sync: Integration overhead is low; The advantage will not be clear in CPU utilization and performance improvement for the workloads.
  - ASYNC: Integration overhead is high; There will be CPU resource saving and performance improvement for the workloads.
- To demonstrate that our idea is practical, two examples are used in
  - Integrating IOAT/DSA in Storage performance Development Kit (SPDK) and accelerate applications via SPDK framework.
  - Integrating DML library to drive DSA and accelerate PMEM in Ceph

Use Case1: Memory offloading technique enabling in SPDK

SPDK high level architecture



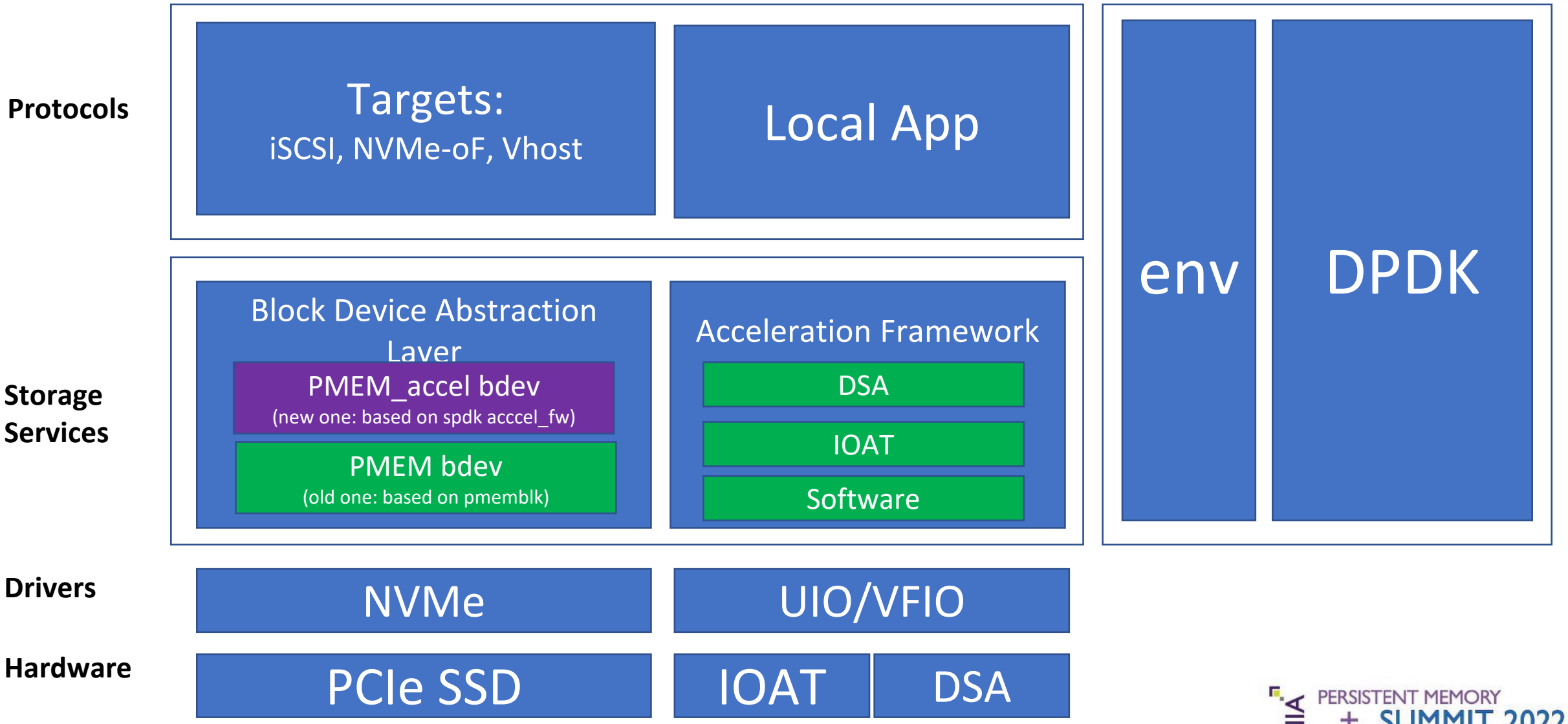
SPDK acceleration framework (accel_fw)



How to offload operations on PMEM?

- We design and implement a new bdev (named as pmem_accel) based on **persistent memory** and spdk **accel_fw** to demonstrate that hardware-based memory offloading engine (e.g., IOAT/DSA) can help improve the performance.
  - For this new bdev, we will not consider some complicated I/O usage scenarios (e.g., I/O atomicity for large size, sequence order, which should be addressed and guaranteed by upper layer logic).
- Our purpose is to leverage IOAT/DSA in spdk accel_fw to reduce the CPU participation overhead.
- Ideally, such offloading should be provided by Persistent Memory Development Kit (PMDK), but currently PMDK library has some limitations, i.e.,
  - *It only provides the synchronized API interface. (This will be an ongoing work inside PMDK)*
  - *It does not have the ability to plugin memory offloading engines inside PMDK.*

PMEM_accel bdev
(Based on PMEM device and spdk accel_fw)



Pmem_accel bdev summary

- Current situation:
- We choose to use DAX mode directly. We want to continue discussing with customers whether DAX mode can fully satisfy customers' requirements. We provide the ongoing patch in public to notify our progress.
- In this usage, file system can not be viewed directly on the char device. The pmem devices can be partitioned into different character devices.
  - A SPDK process will directly use one or several /dev/dax* devices directly, and those operations on bdev based on pmem can be offloaded via DSA in SPDK accel_fw.

Use Case2: Memory offloading technique enabling in Ceph

Phase1: Offloading PMEM operations in Bluestore

- Proposed approach:
  - Leverage **DML** or **idxd-config** lib to invoke DSA capabilities to offload the operations between memory and PMEM.
- Step1: Use the sync API in DML to offload CPU operations.
  - Expectation: Functionality can work (e.g., data operation persistency and correctness).. **(Status: Nearly done) Two patches are merged**
  - https://github.com/ceph/ceph/pull/45609 (Support devdax usage on PMEM)
  - https://github.com/ceph/ceph/pull/44230/ (Leverage DML to operate on PMEM)
- Step2: Use the async API in DML to offload CPU operations.
  - Expectation: Have performance improvement and mitigate the CPU pressure. Due to current Ceph architecture, this work is not easy. (Different with applications which use SPDK's async programming framework)

Phase2: Offloading PMEM operations in Seastore

- Seastore uses Seastar framework, and we expect the good async behavior in Seastore.
- Then we may do the following works:
  - Use DSA to offload some memory operations, e.g., dualcast, copy, crc32c.
  - Use DSA to continue optimizing operations on PMEM in Seastore with async behavior.

Conclusion

Conclusion

- We realize the overhead/bottleneck of CPU to conduct data operations between (persistent) memory while performing storage workloads.
  - Data operations “between (persistent) Memory” is different from “data operations between Memory and device”. For the latter case, there is (R)DMA capability on device, and CPU is not very busy.
  - We need memory operation offloading engines (e.g., IOAT/DSA) to offload the work by CPU.
- We introduce two usage cases to offload PMEM operations via memory offloading engines, i.e.,
  - Leverage spdk accel_fw to offload operations on PMEM based bdev.
  - Leverage DML to drive DSA to offload PMEM in Ceph.
- Next step: We will continue improving the performance on PMEM via memory offloading engine based on DSA.
  - For performance report, we will publish it later after SPR is launched.

Q & A

Please take a moment to rate this session.

Your feedback is important to us.

Backup

Two possible usage on PMEM in AD mode.

- PMEM in App Direct (AD) mode can be used for persistency purpose. And the PMEM usage in AD mode can be divided into
- FSDAX mode
  - Format the pmem device with “fsdax” choice. Users get block devices, e.g., /dev/pmem0, and users can create file system (especially kernel supported file system) upon that.
- DAX Mode
  - Format the pmem device with “dax” choice . Then we get char devices, e.g., /dev/dax0.0, then you directly operate on such device.

