SDC 19

September 23-26, 2019
Santa Clara, CA

Intel® Optane™ DC Persistent Memory Performance Review

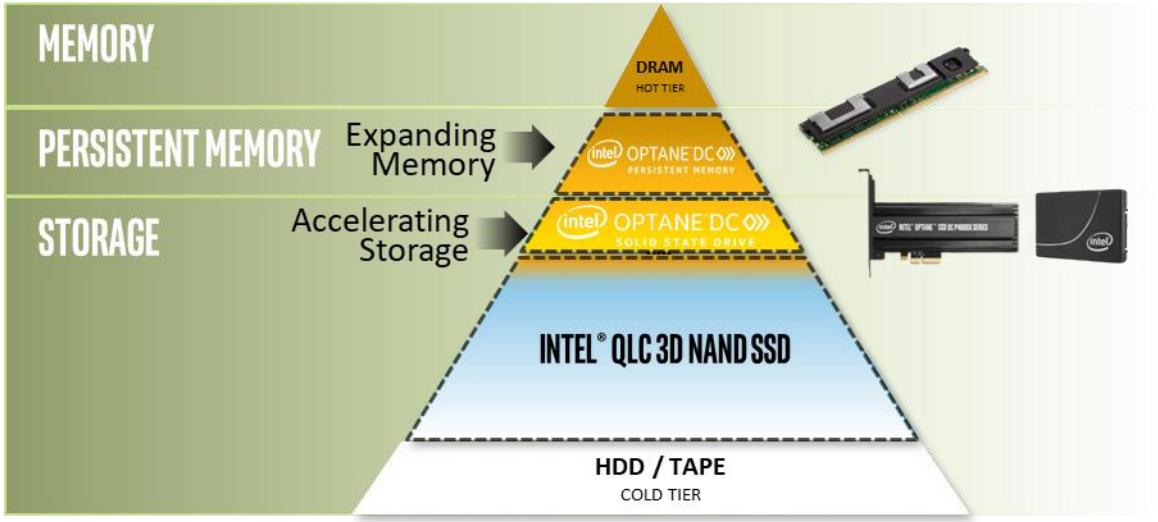
Low-Latency Replication with Remote Persistent Memory

Michael Strassmaier
Intel Corporation

Agenda

- Intel® Optane™ DC Persistent Memory
  - Product and Performance Overview
- The Challenges and Importance of Replication Performance
- Traditional Data Replication
- Replication with RDMA and Persistent Memory (using PMDK and the `librpmem` library)
- PMDK API Support
- Performance Considerations

Closing The Gap in the Memory/Storage Hierarchy

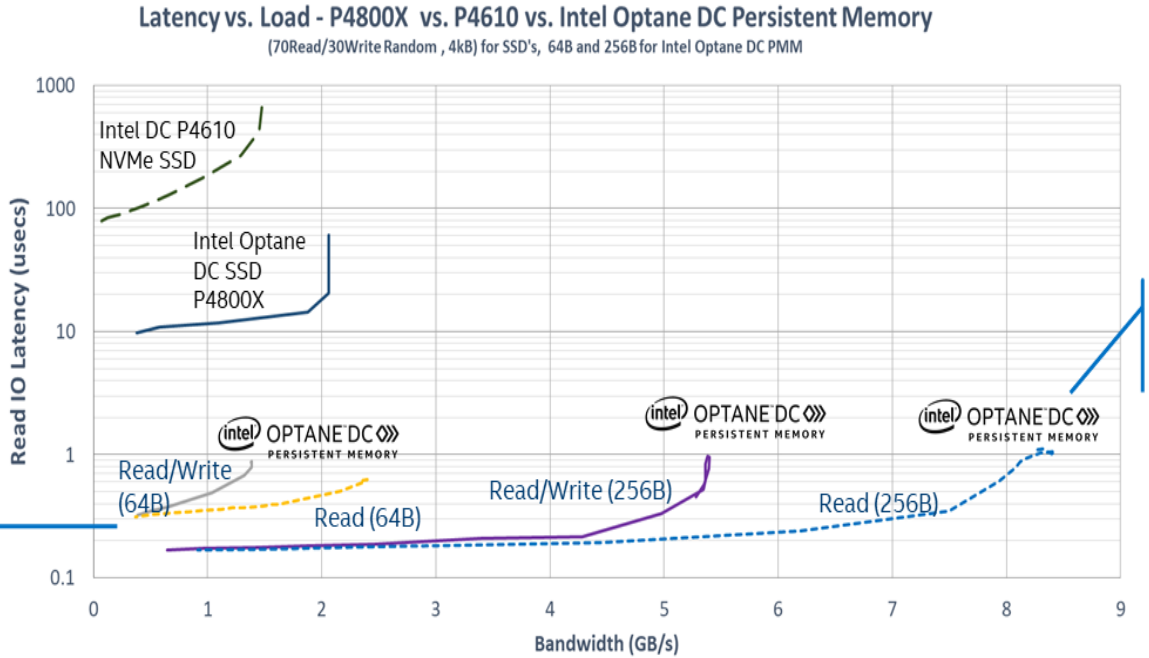


Glossary

Optane:	Intel's memory media technology
Optane SSD:	Solid-State Drive built with Optane media
Persistent Memory	Byte addressable load/store memory
PMEM	Persistent Memory
DC PMM	Data Center Persistent Memory Module

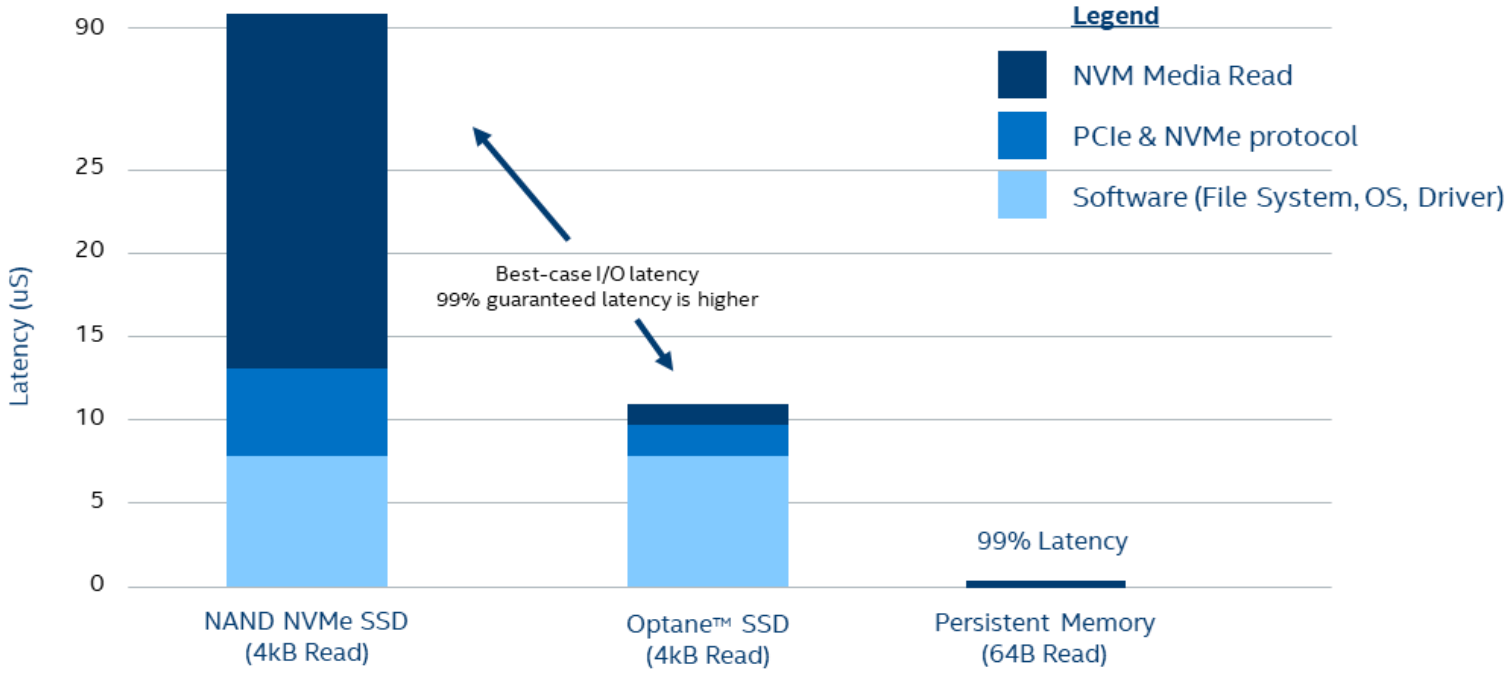
Intel® Optane™ DC Persistent Memory latency

- Orders of magnitude lower latency than SSD
- 2X read/write bandwidth vs disk, with one module, more with multiple modules



1. 256B granularity (64B accesses). Note 4K granularity gives about same performance as 256B

Application Latencies:
NAND vs. Optane SSD vs. Optane Persistent Memory



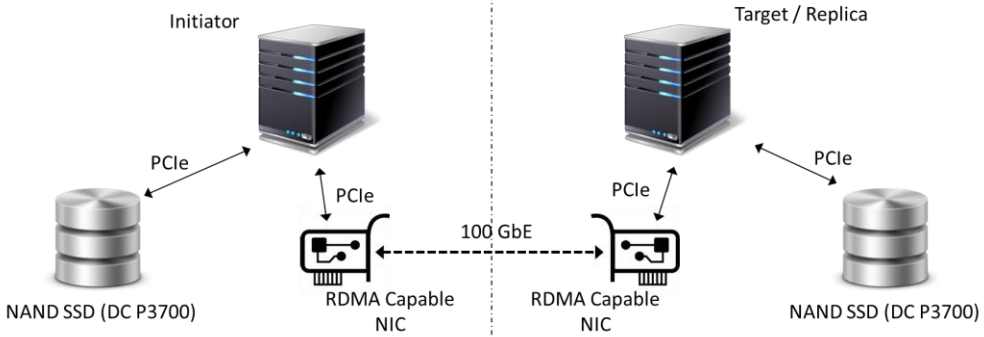
Performance results are based on testing as of Feb 24, 2019 and may not reflect all publicly available security updates. No product or component can be absolutely secure. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations 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 information go to www.intel.com/benchmarks. 1-3

The Challenges and Importance of Replication Performance

- Latency
  - Can the data replication tool keep current with transaction log generation, or not?
  - Has the time period while data is out of sequence been minimized (near real-time replication is most optimal)?
- Resource consumption
  - Includes CPU, memory, storage, and network resources.

Traditional Data Replication

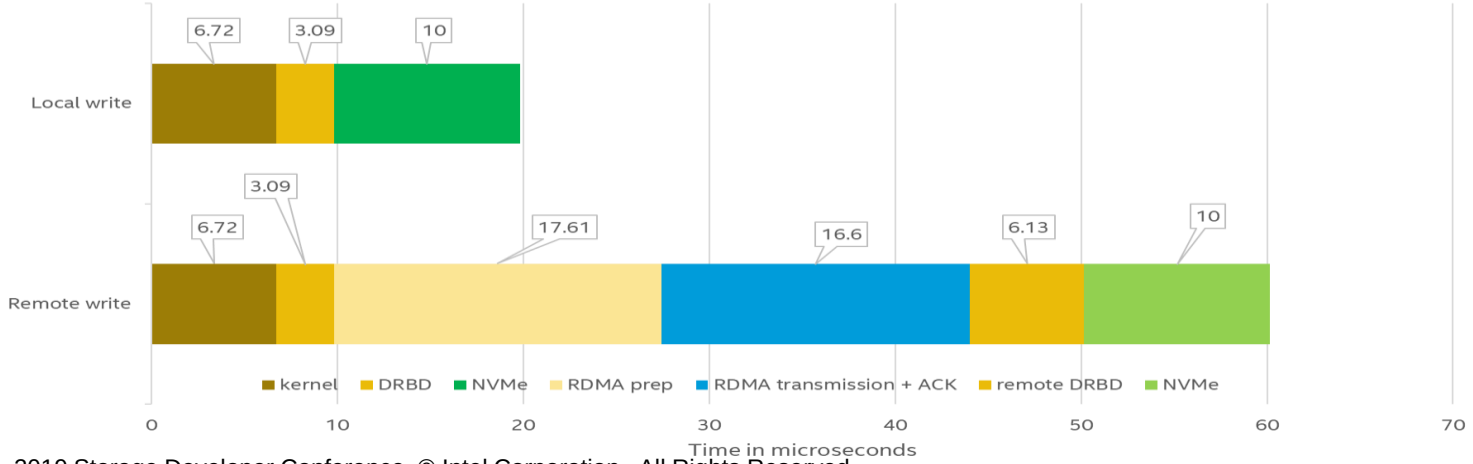
- Traditional I/O bound remote data replication
  - Normally implemented as a kernel driver, management applications and shell scripts.
- Data replication API
  - Layers logical block devices over existing local block devices (usually SSD/HDD) on participating cluster nodes.
  - Writes to primary node are transferred to the lower-level block device and simultaneously propagated to the secondary node(s).



Traditional I/O Bound Data Replication

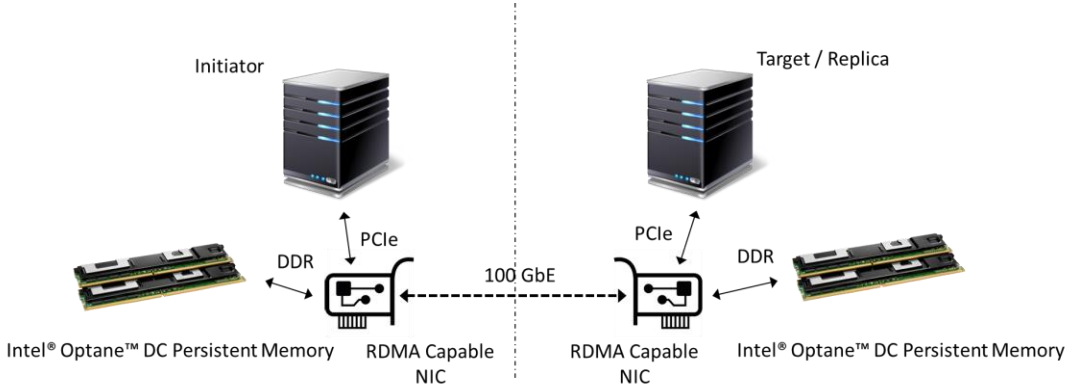
- Intel replication test case
  - Memory/Storage: Intel® SSD DC P3700 Series 400GB
  - File size: 10 GB file on 40 GB ext4 file system
  - Block size: 4096 / Data size: 4096
- Traditional data replication (DRBD) require multiple data hops
  - Total latency for 4KB replication = 60.15 microseconds

Replication average latency by component - 4KB sequential writes



Replication with RDMA and Persistent Memory (using the PMDK `librpmem` library)

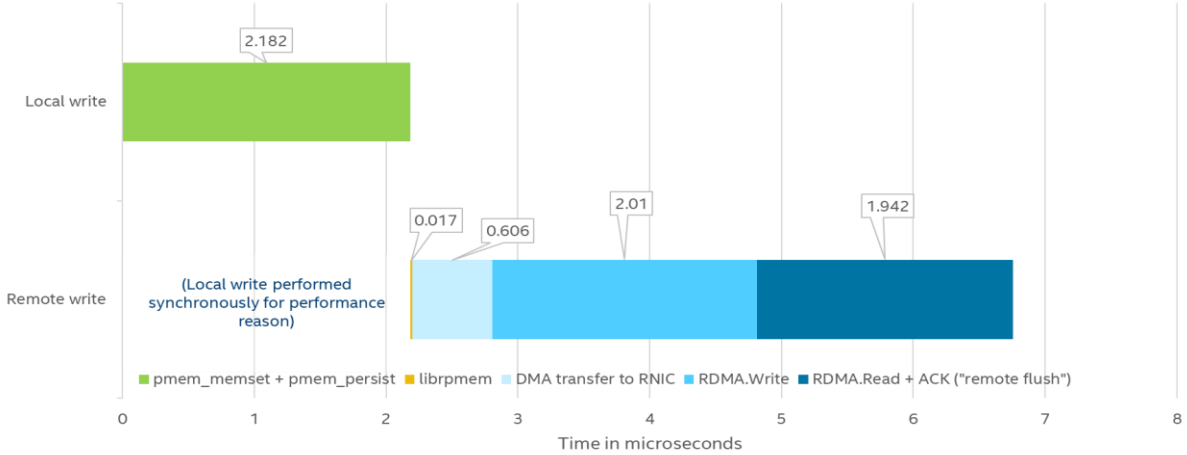
- RDMA is low latency high-speed network interface that controls movement of data between the initiator node and sink buffer on the target node (one-sided operation).
  - Direct memory access (DMA) allows data movement on a platform to be offloaded to a hardware DMA engine that moves that data on behalf of the CPU.
- Once persistent memory is accessible via remote network connection, significantly lower latency can is achieved as compared with writing to remote SSD or block storage device.
- Replication via direct memory access using persistent memory over a high-speed network connection is a superior solution for IaaS deployments.



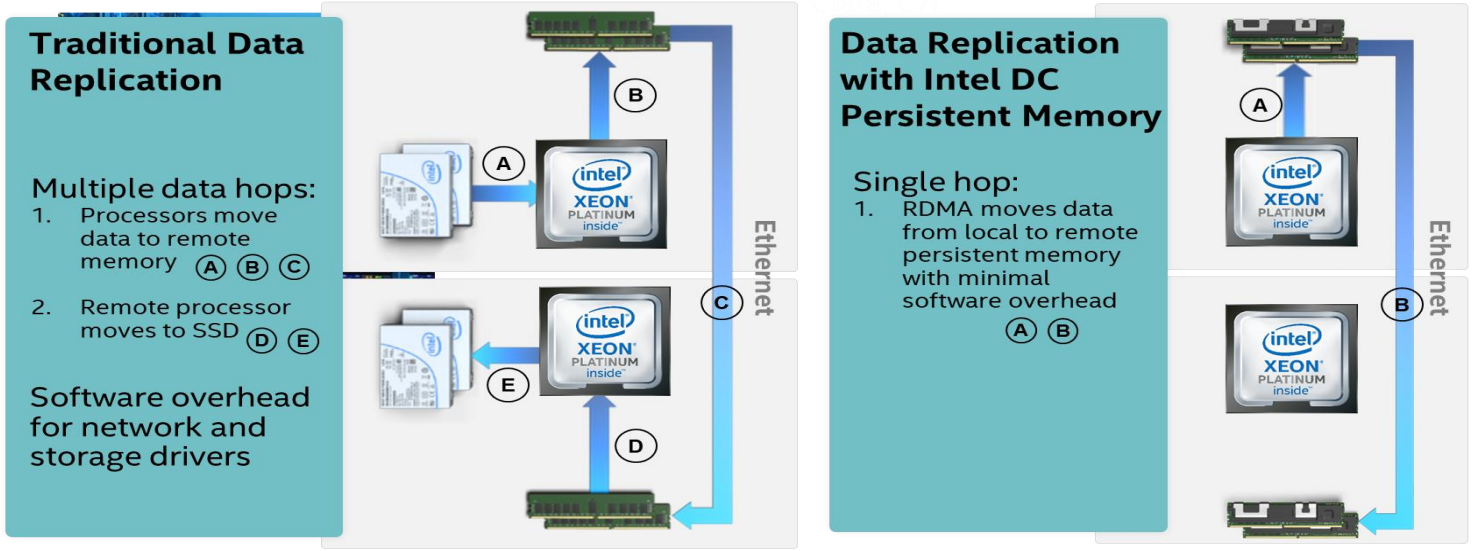
Replication using RPMEM

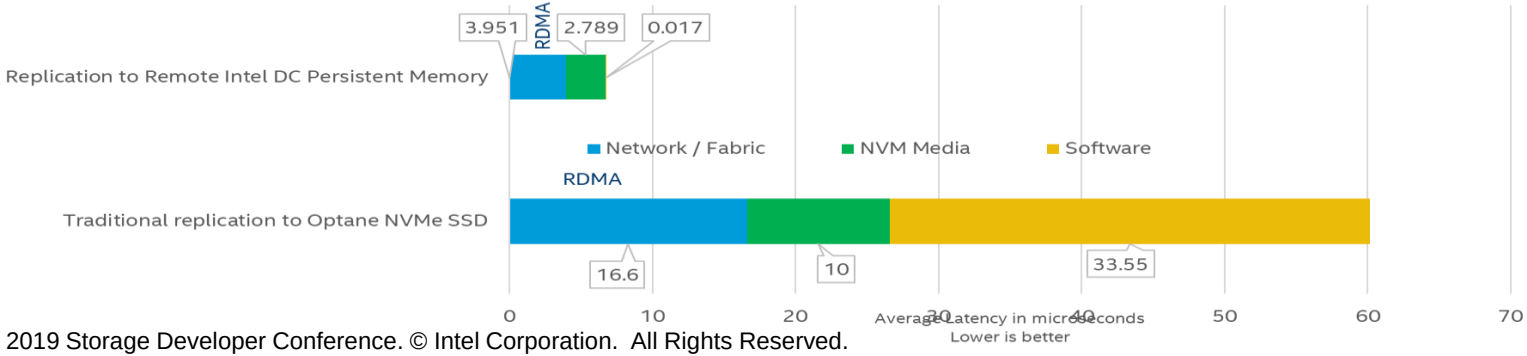
- RPMEM Intel test configuration
  - Memory: 2x 256GB Intel® Optane™ DC Persistent Memory
  - File size: 10 GB Device DAX
  - Software: PMDK 1.6
- RDMA replication using RPMEM requires a single hop
  - Total latency for 4KB replication = 6.76 microseconds

RPMEM avg. latency by component (4KB sequential write)



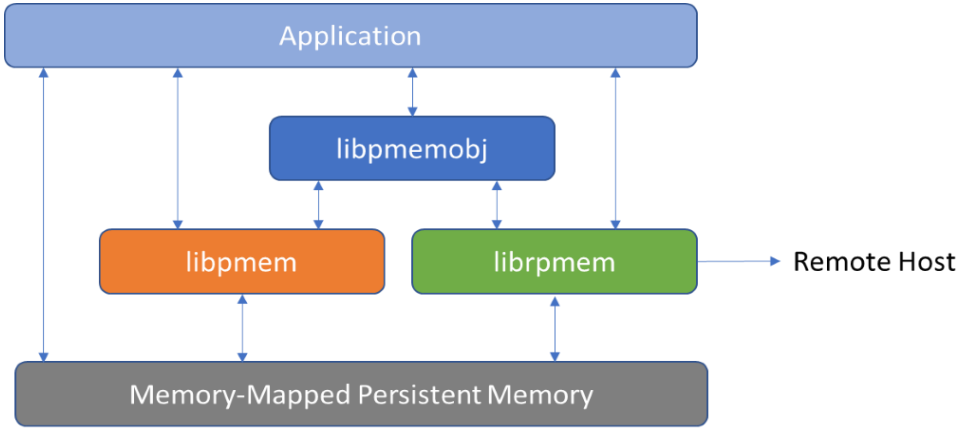
Data Replication with Persistent Memory



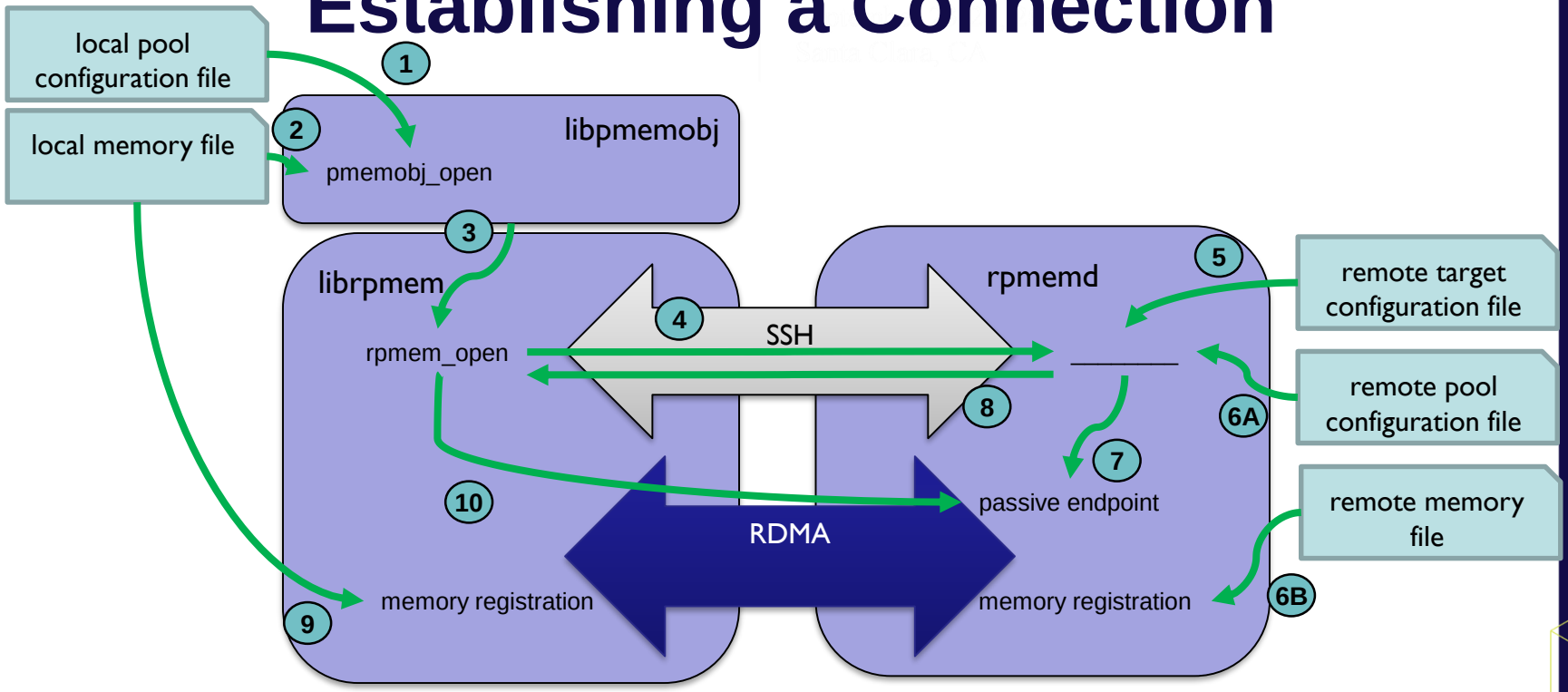


PMDK API Support

- PMDK v1.6 implements both the *general purpose remote replication* method and the *appliance remote replication* method in the `librpmem` library.
  - `librpmem` library implements the synchronous replication of local writes to persistent memory on one or more remote systems.
  - `librpmem` is a low-level library, that allows other libraries to use its replication features. Applications using `libpmemobj` can replicate local writes to the initiator's persistent memory to remotely connected target persistent memory ranges.

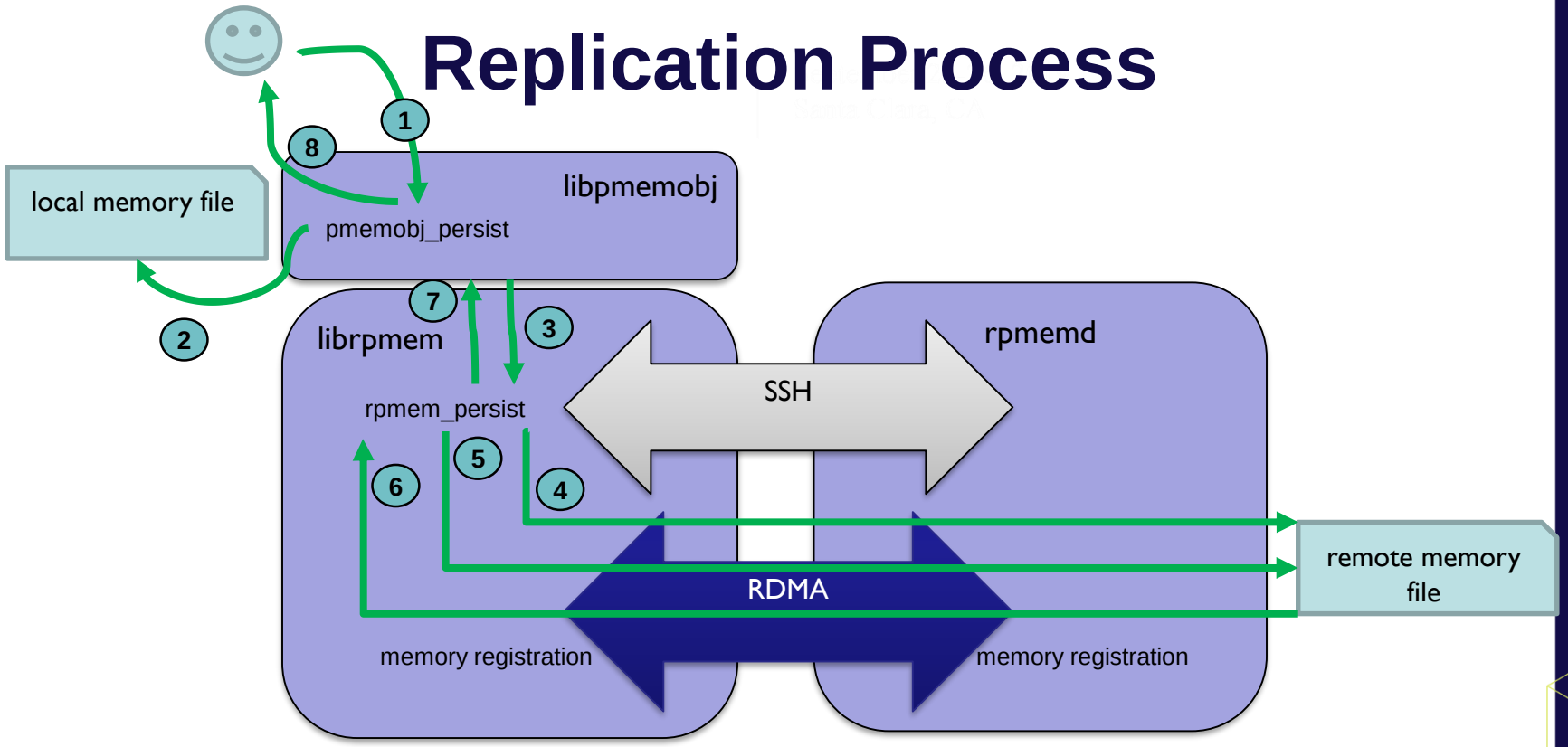


Establishing a Connection



1. user provides a local pool configuration file to pmemobj_open API call
2. libpmemobj maps local memory file
3. libpmemobj calls rpmem_open API call with connection details obtained in step 1
4. librpmem requests access to the memory described in the remote pool configuration file
5. rpmemd obtains target capabilities reading a remote target configuration file
6. rpmemd registers the remote memory file described in step 4 for RDMA
7. rpmemd starts listening for incomming RDMA connections
8. rpmemd sends back details required to establish RDMA connection
9. librpmem registers the local memory mapped in step 2. for RDMA
10. librpmem establish RDMA connection

Replication Process



1. Let's assume a user performs pmemobj_persist or any other libpmemobj API call designed to change the persistent state
2. libpmemobj performs changes on local memory file
3. libpmemobj calls rpmem_persist on all of its remote replicas
4. librpmem posts RDMA.Write to the remote memory file
5. Immediately posts RDMA.Read to flush RDMA.Write into persistence (assuming APM mode)
6. RDMA.Read completion guarantees change to the remote memory file is persistent
7. librpmem returns from rpmem_persist
8. Libpmemobj returns from pmemobj_persist

Performance Considerations

- Block sizes
  - 512KB+ block sizes can achieve good performance.
  - As the size of replication the writes gets smaller, the network overhead becomes a larger portion of the total latency and performance can suffer.
  - Typical native block storage size is 4K, avoiding some of the inefficiencies seen with small transfers.
  - If the persistent memory replaces a traditional SSD and data is written remotely to the SSD, the latency improvements with persistent memory can be 10x or more.

Summary

- Significantly lower replication latency can be achieved with Optane DC™ Persistent Memory as compared to remote SSD or legacy block storage device.
  - RDMA with RPMEM bypasses the software stack reducing CPU utilization and network storage overhead.
- RDMA with RPMEM writes remote data directly to the final persistent memory location as a single hop.
  - Traditional replication over block storage requires RDMA move into DRAM on a remote server followed by a second local operation to move the remote write data into the final storage location.
- If the persistent memory is being utilized as an SSD replacement, as in this performance test case, the typical native block storage size is 4K, avoiding some of the inefficiencies seen with small transfers.
- As demonstrated in our test performance data, replication with RPMEM using persistent memory can deliver 10x or even greater performance as compared to traditional replication solutions.

Resources

- PMDK Resources:
  - Home: https://pmem.io
  - PMDK: https://pmem.io/pmdk/
  - PMDK Source Code: https://github.com/pmem/PMDK
  - Google Group: https://groups.google.com/forum/#!forum/pmem
  - Intel Developer Zone: https://software.intel.com/persistent-memory
- NDCTL: https://pmem.io/ndctl/
- IPMCTL: https://github.com/intel/ipmctl
- MemKind: https://memkind.github.io/memkind/
- LLPL: https://github.com/pmem/llpl
- PCJ: https://github.com/pmem/pcj
- SNIA NVM Programming Model: https://www.snia.org/tech_activities/standards/curr_standards/npm
- Getting Started Guides: https://docs.pmem.io

Backup

DRBD Configuration

Storage Type	Intel® SSD DC P3700 Series 400GB (SSDPEDMD400G4)
Software	DRBD 9.0.18_3.10.0_957-1 + RDMA transport 2.0.13_3.10.0_957-20190611.el7 + fio-3.14
File size	10 GB file on 40 GB ext4 file system
OS	CentOS Linux release 7.6.1810 + kernel 3.10.0-957.el7.x86_64

RPMEM Configuration

Memory	2x 256GB Intel® Optane™ DC Persistent Memory / socket (interleaved)
Software	PMDK 1.6
File size	10 GB Device DAX
OS	Fedora 29 + kernel 4.20.13-200.fc29

19

Test Configuration

	fio	pmembench (rpmem_persist)
workload	rw=write	mem-mode=seq-wrap
data size	bs=4096	data-size=4096
	direct=1	persist-relaxed=true
		no-memset=false

Where:

- rw=write – sequential writes
- direct=1 - use of non-buffered I/O (O_DIRECT). File I/O is done directly to/from user-space buffers.

- mem-mode=seq-wrap – sequential writes
- persist-relaxed=true - use of RDMA.Write + RDMA.Read to assure persistency of the data on the remote node
- no-memset=false – use of memset (storing the data locally) is a part of the process

20