Remote Persistent Memory

Tomasz Gromadzki

intel

Software Architect
PMDK, RPMem

Notice and Disclaimers

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Intel technologies may require enabled hardware, software or service activation.

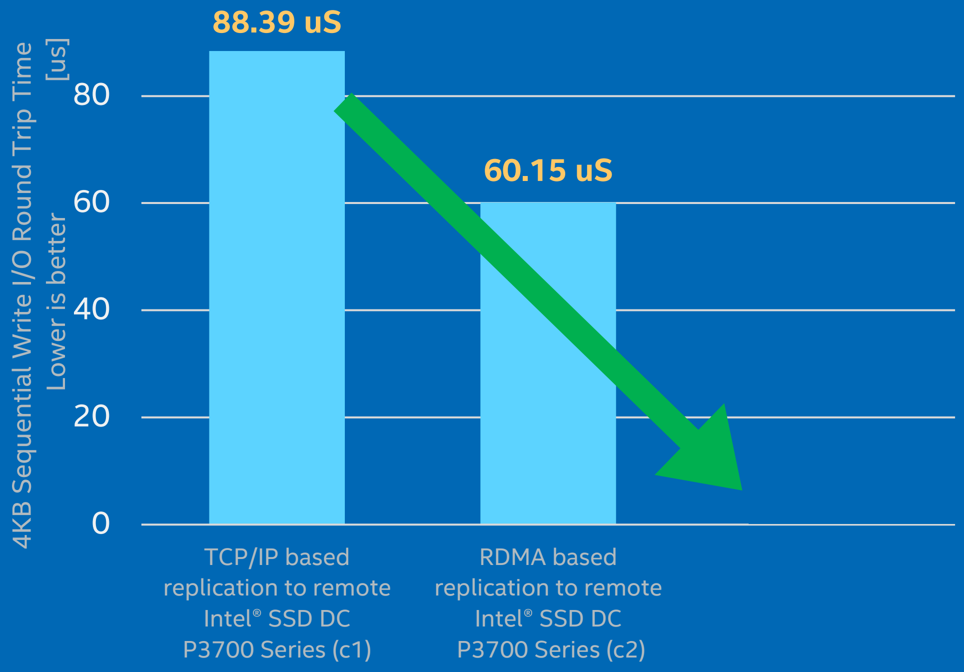
Your costs and results may vary.

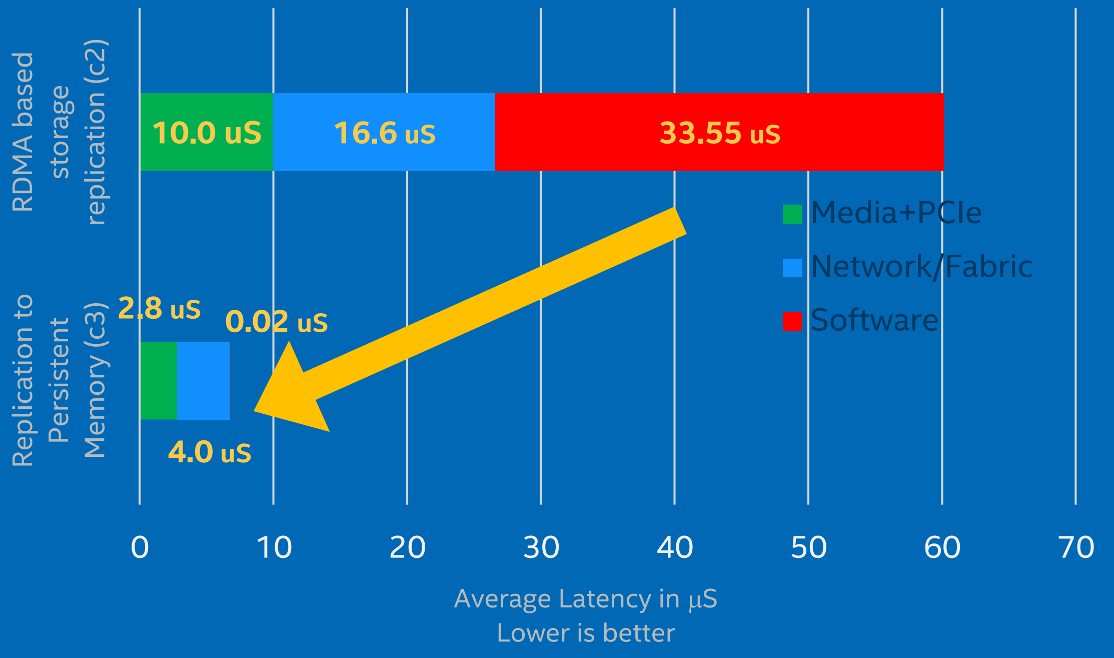
Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

Agenda

Remote Persistent Memory (RPMem)

RDMA with PMem - motivation





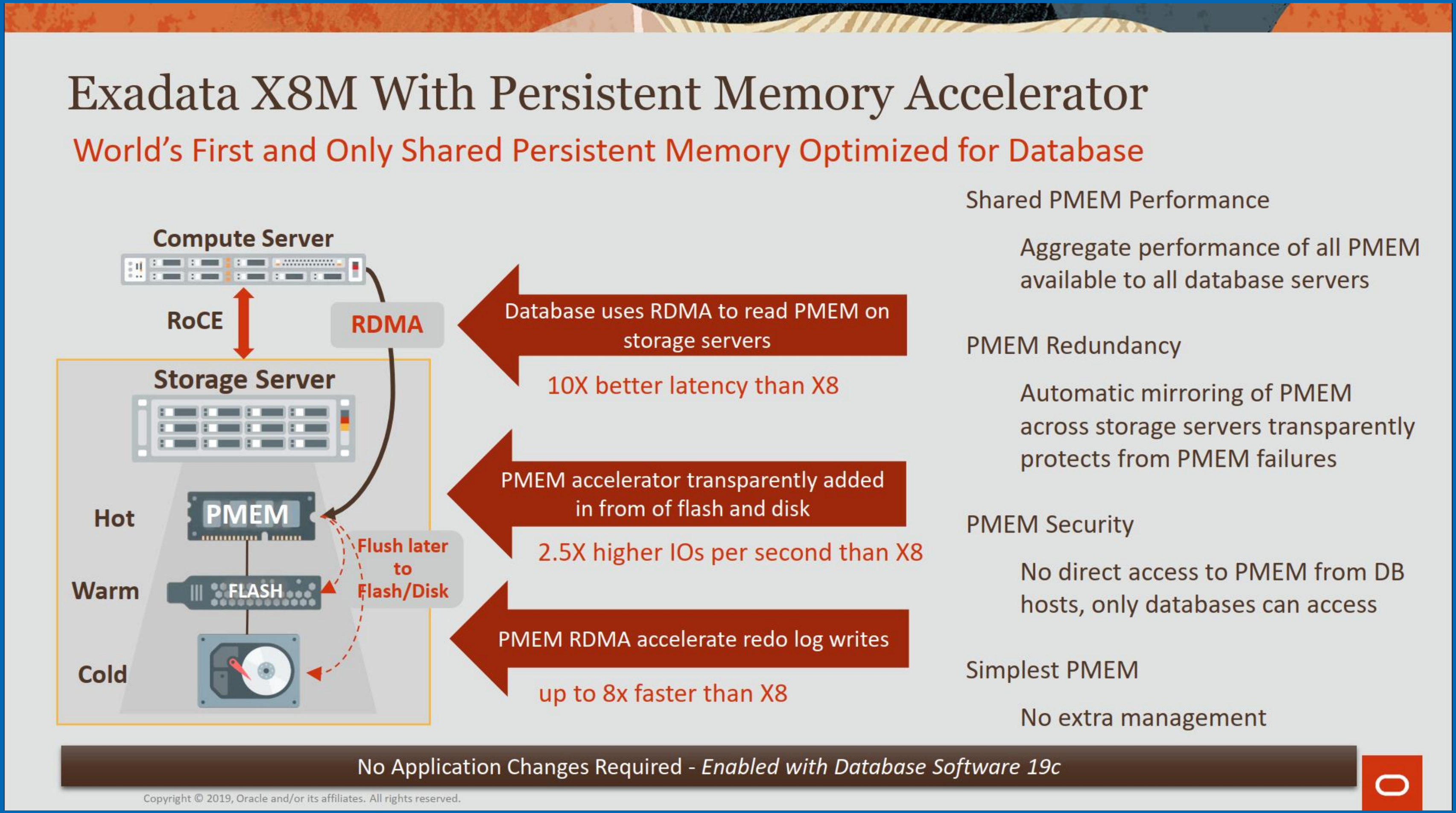
- RDMA and PMem complement each other very well
- Up to ~8x lower latency
- Replicated data can be processed immediately

- Extremely low software overhead
- No CPU involved in data transfer, pure HW data path

See backup for workloads and configurations. Results may vary.

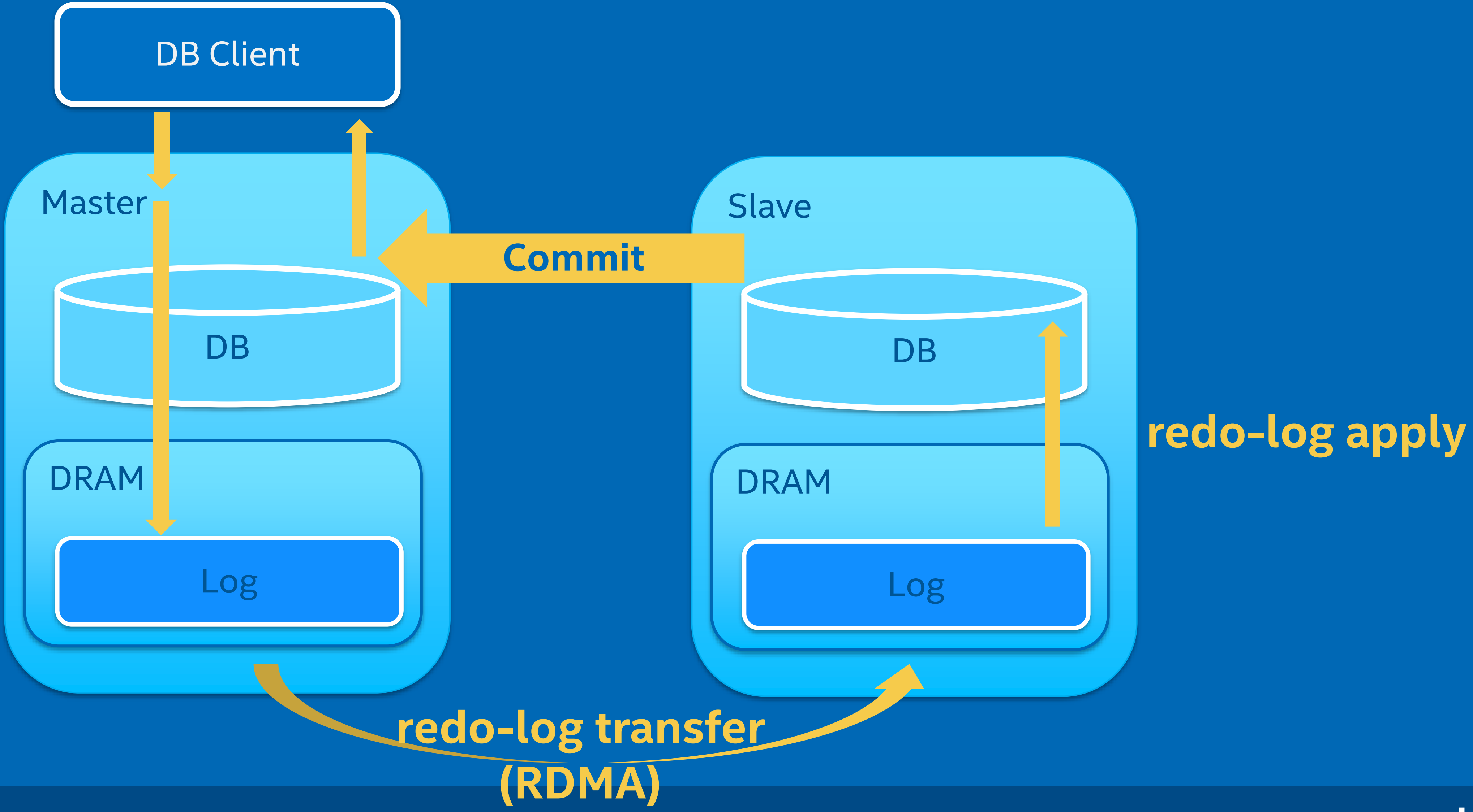
Oracle Exadata X8M

RPMem fits well into distributed database systems

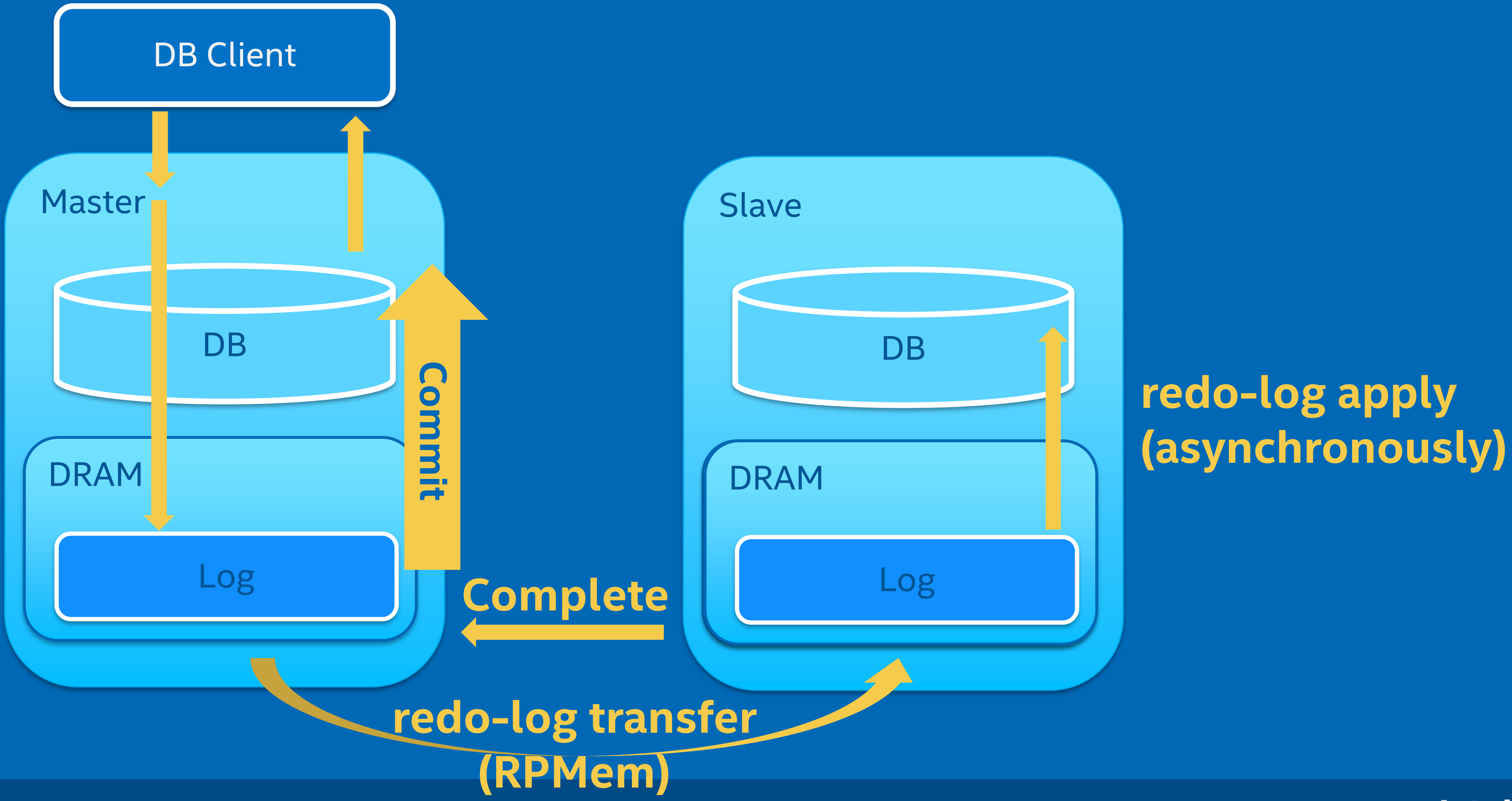


https://www.oracle.com/a/ocom/docs/dc/em/exadatastrategyroadmap-final2a.pdf

DBMS with synchronous replica



DBMS with synchronous PMem replica



RPMem Solution Fundamentals

RPMem over Traditional RDMA and Existing Intel Server Platform

Persistent Memory Programming Model

There are a number of ways SW can accomplish this on an Intel platform.

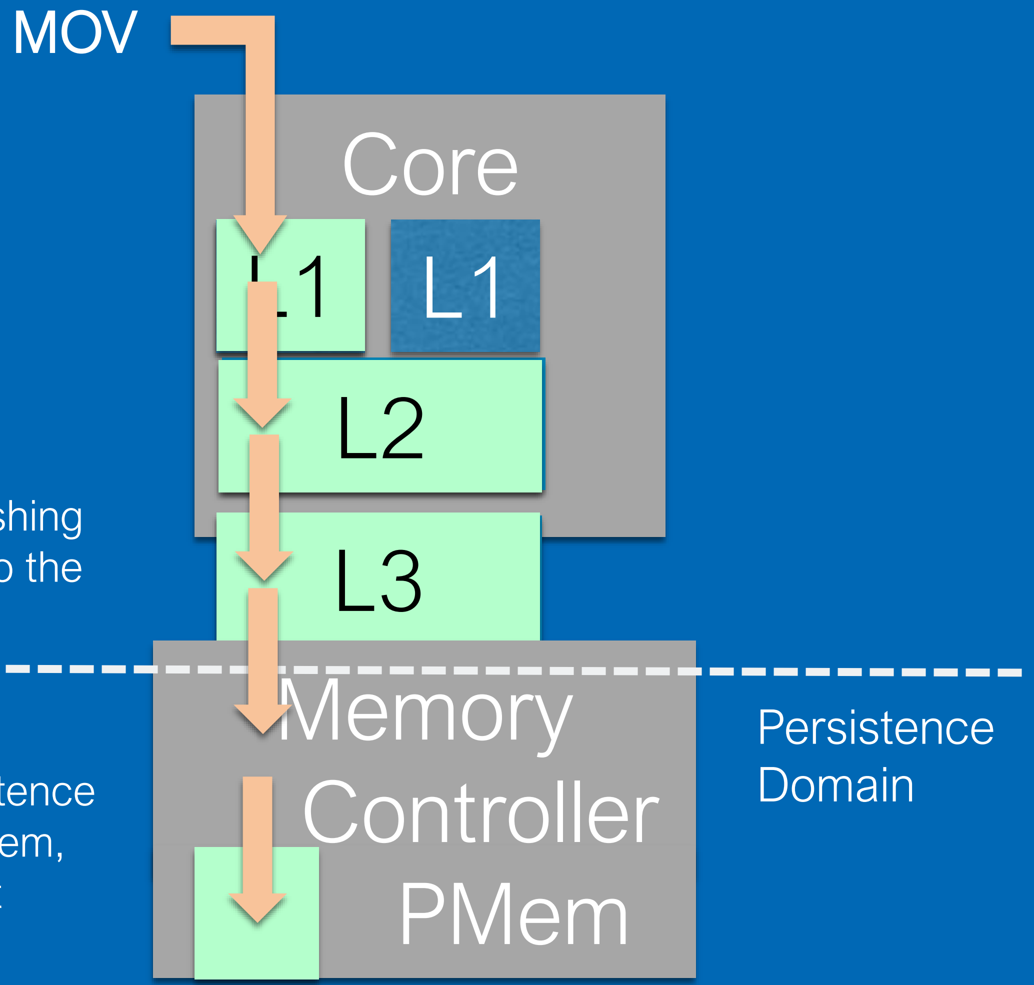
Follow the **MOV** with one of the following:

CLWB + fence
CLFLUSHOPT + fence
CLFLUSH
WBINVD (kernel only)

to force data into the Persistence Domain

SW is responsible for flushing data from CPU caches to the Persistence Domain

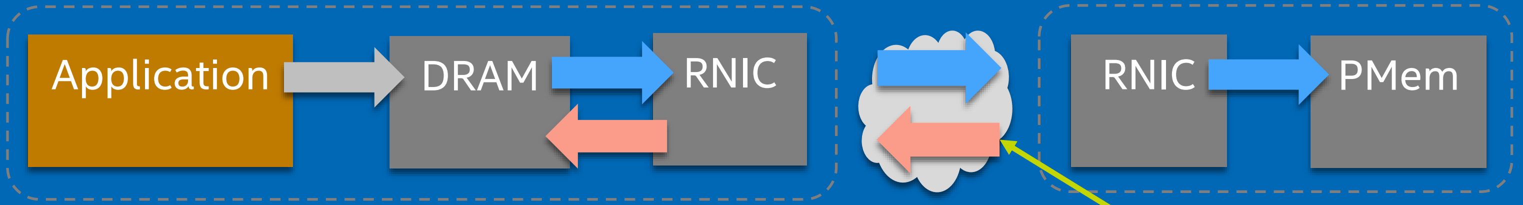
Platform guarantees data flushed to the Persistence Domain is written to PMem, should power be lost



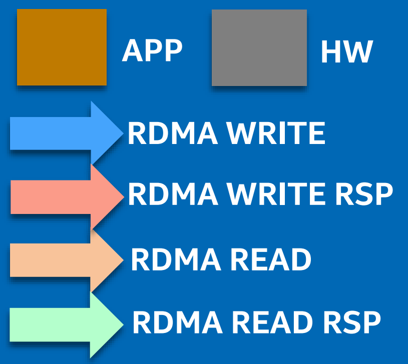
This same model applies when accessing PMem over a network

Implementing the PMem programming model over an RDMA network

Initiator System | RDMA Network | Target System

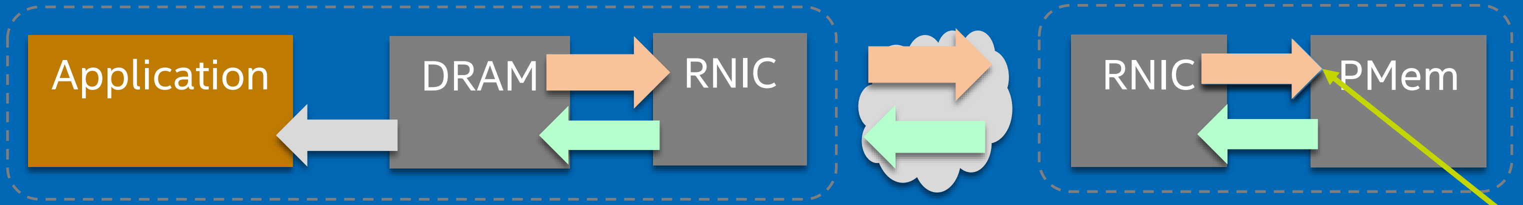


Writing data to remote persistent memory



As the programming model states, the Application will need to execute a flush to make sure the data has been flushed to the persistence domain.

The RDMA Write response only tells the Application that the Target RNIC has sent the write data on to the final storage location in persistent memory.



Flushing previously written data to remote persistent memory

The RDMA Read forces any write data to the persistence domain before the RDMA Read can proceed. This is because the ordering rules of both RDMA and PCIe.

Appliance Persistency Method

Existing Software Solutions for RPMem

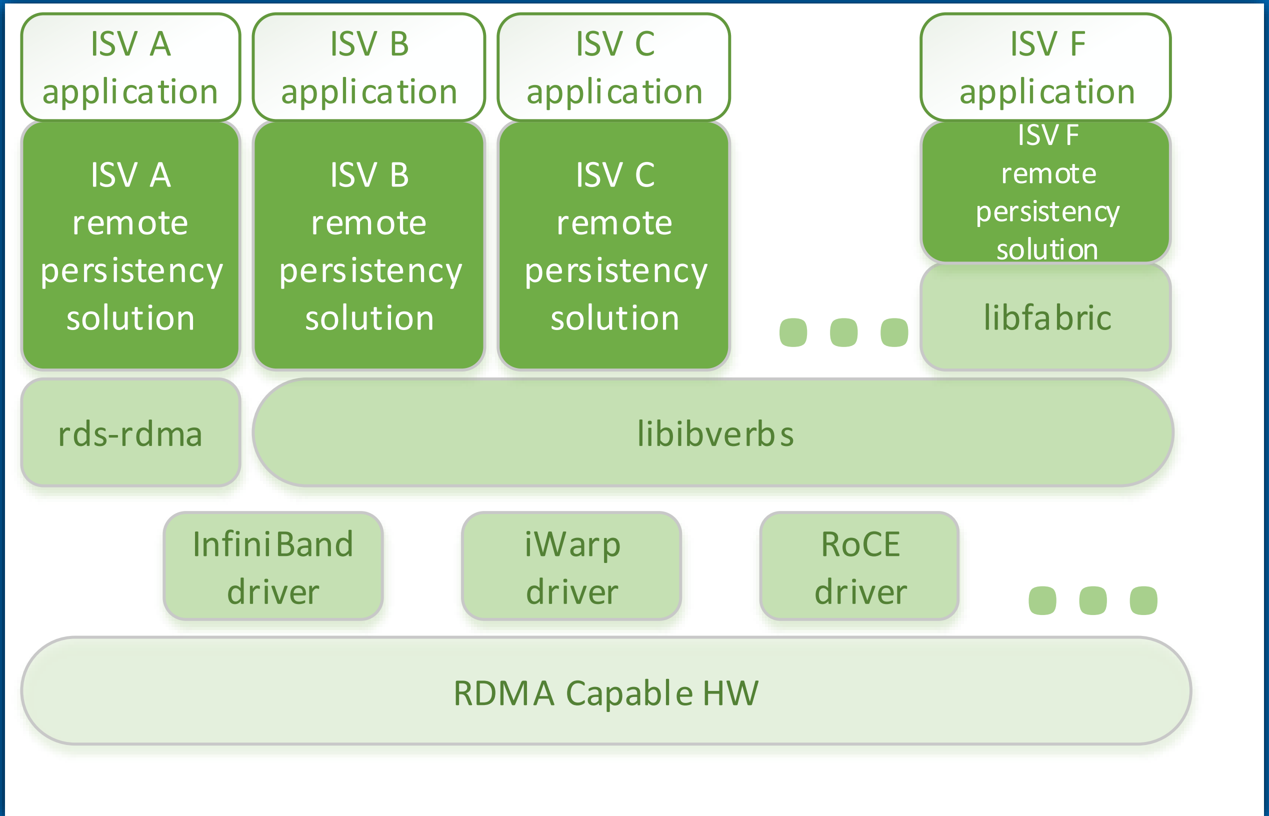
Linux environment to run RPMem

- **RDMA accesses Intel® Optane™ PMem in the same way it accesses DRAM**

- Remote PMem (RPMem) is about well-known technologies (like PCIe, RDMA) used in a new way

RPMem Software Stack

Linux environment

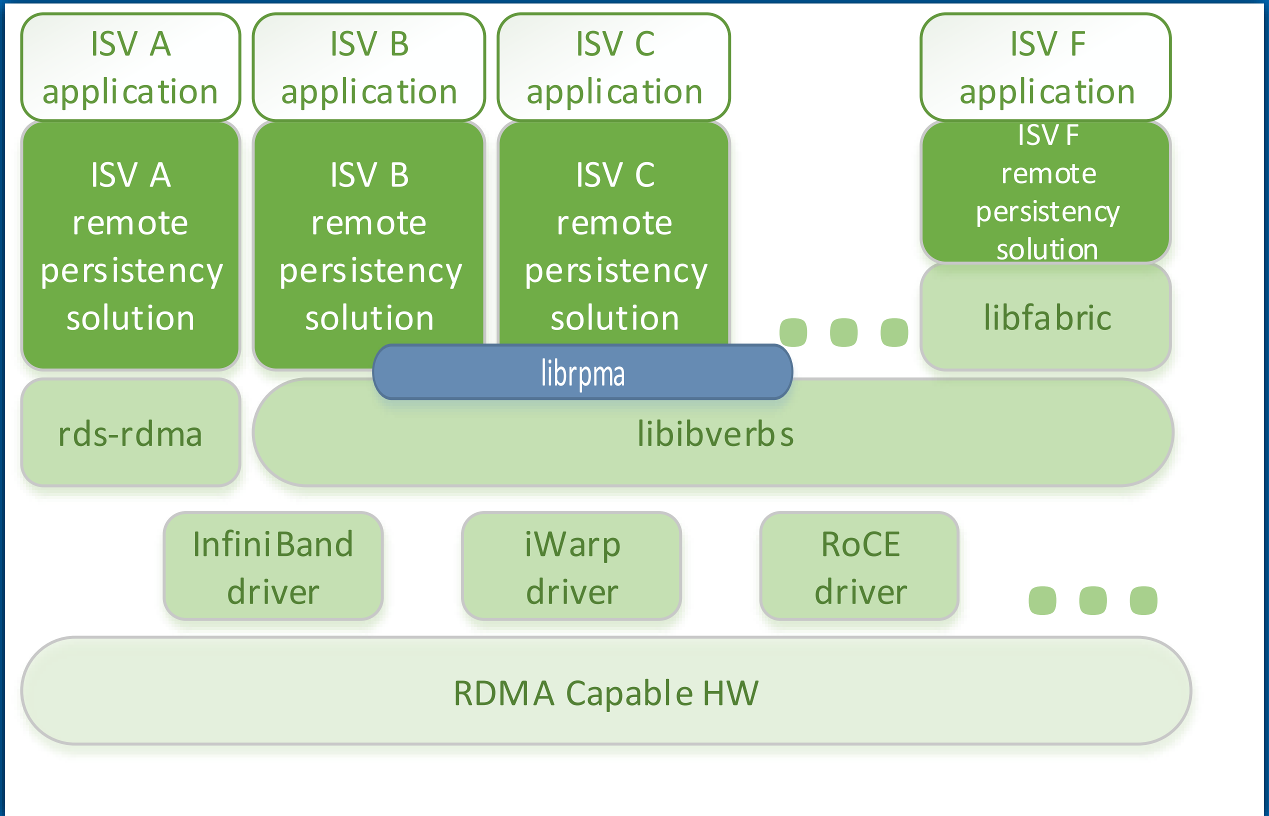


Since RPMem is based on existing RDMA networking interface, remote durability solution can be built on the top of:

- libibverbs library
- rds-rdma kernel module
- libfabric library

The new librpma focuses on RPMem usability

Linux environment

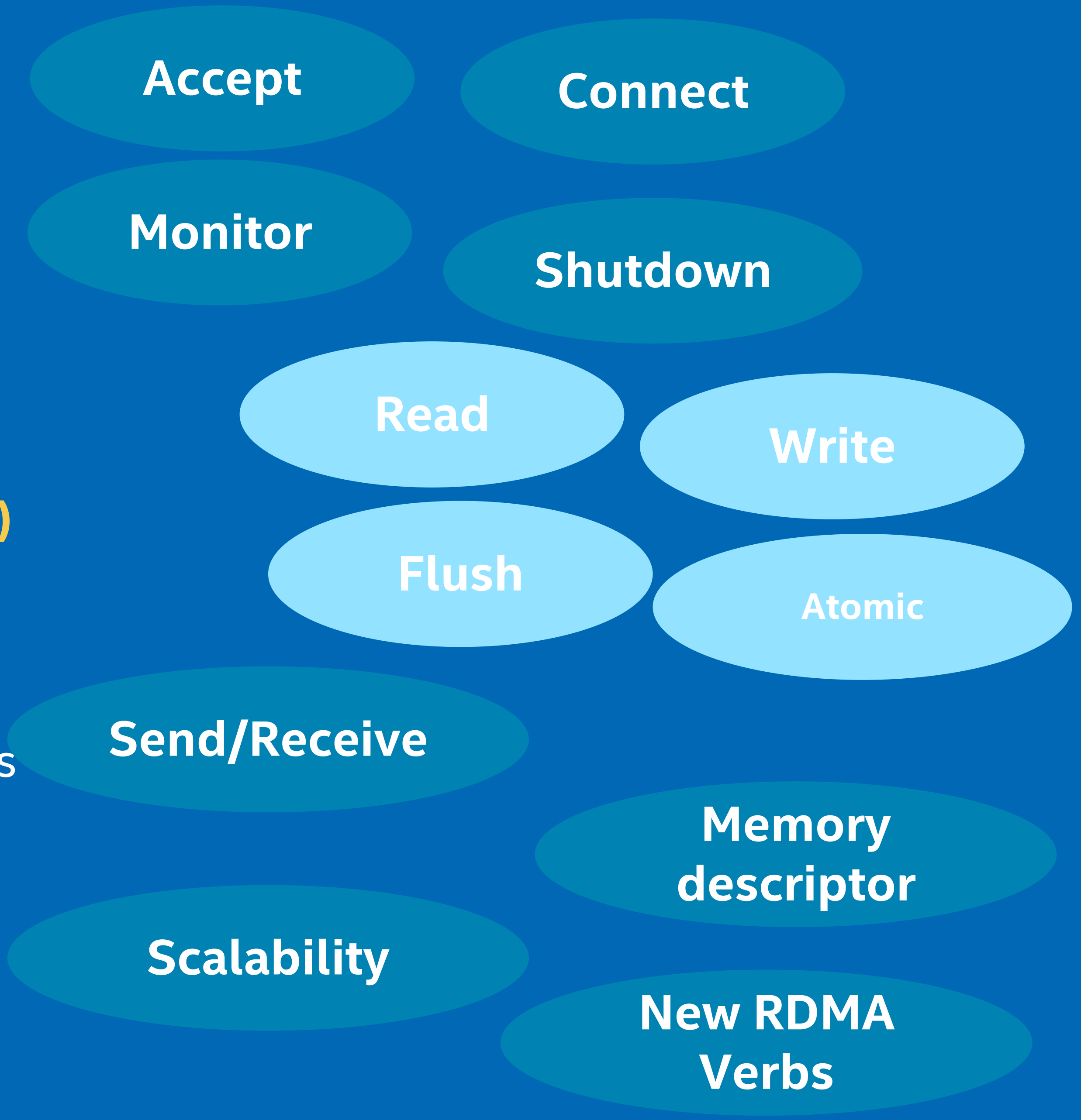


- memcpy-like API
- hides RDMA complexity
- an application can freely manage PMem all the time
- minimum dependencies

up to 50% RPMem source code reduction in an application that moves from libibverbs to librpma

librpma API

- Connection management
  - to ensure operations consistency
  - to hide RDMA complexity
- **Remote Peristent Memory Access (RPMA)**
  - Read, Write, Flush, Atomic write
- Messaging
  - also with PMEM-backed message buffers
- Memory management
  - r_key exchange support
- Ready to incorporate
  RDMA Memory Placement Extension



Basic example – memory registration

Initiator node

```
rpma_mr_reg(peer,
        ptr, size,
        RPMA_MR_USAGE_WRITE_SRC,
        &src_mr);
```

Target node

```
rpma_mr_reg(peer,
        ptr, size,
        RPMA_MR_USAGE_WRITE_DST |
        RPMA_MR_USAGE_FLUSH_TYPE_PERSISTENT,
        &dst_mr);
```

Basic example – RPMem write

Initiator node

Target node

```
rpma_write(conn,
    dst_mr, dst_offset,
    src_mr, src_offset,
    KILOBYTE, RPMA_F_COMPLETION_ON_ERROR, NULL);
```

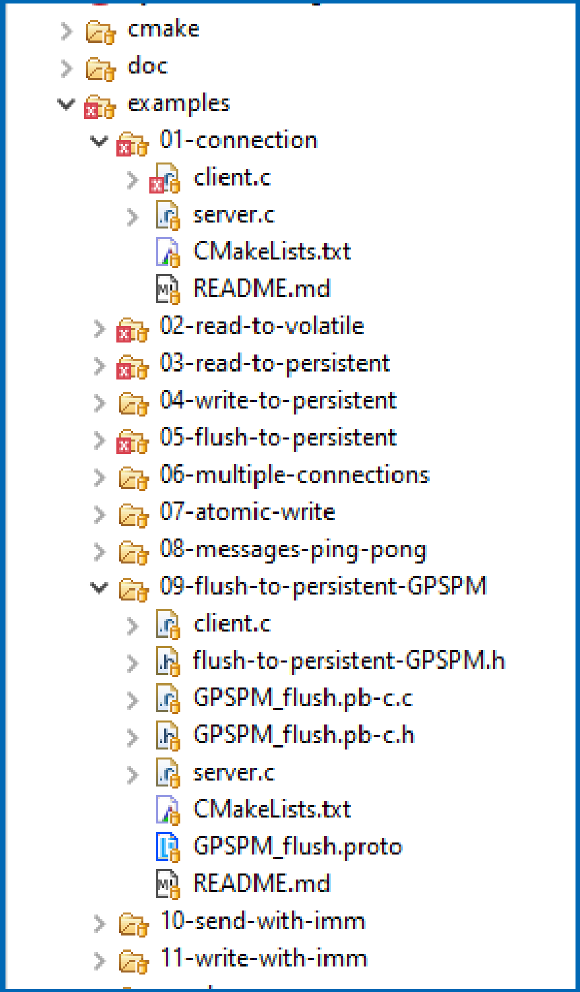
NOP

```
rpma_flush(conn,
    dst_mr, dst_offset,
    KILOBYTE, RPMA_FLUSH_TYPE_PERSISTENT,
    RPMA_F_COMPLETION_ALWAYS, FLUSH_ID);
```

NOP

https://github.com/pmem/rpma/tree/master/examples/05-flush-to-persistent

Included librpma examples

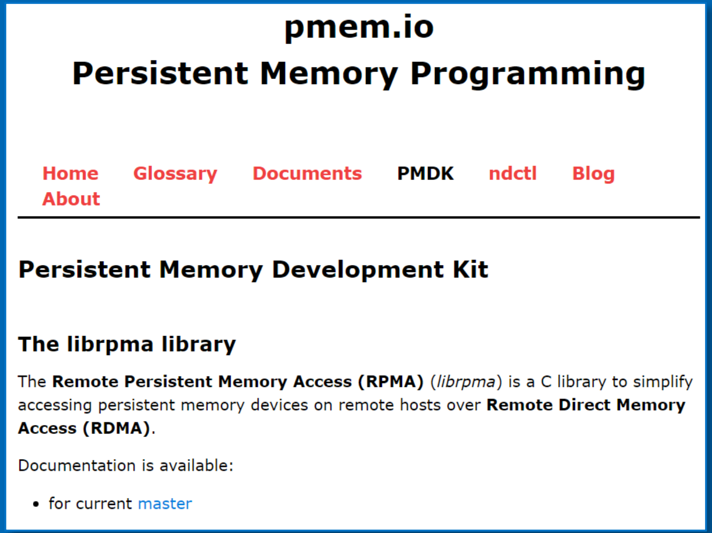


- Connection establishment and management
- Read/write from/to DRAM/PMem
- Multiple connections handling (scalability)
- Atomic write
- Messaging
- Send/Write with immediate data
- Flush to persistent (GPSPM)
- …

More documentation is available

Visit

- **pmem.io/rpma** for official documentation
- git**hub.com/pmem/rpma** to
  - build library
  - run examples
  - **setup benchmarking environment**



Neither an Intel® Optane™ PMem nor RDMA capable NIC is required to run examples. See examples documentation for details.

RPMem benchmarking toolset

For easy performance analysis

github.com/axboe/fio (github.com/pmem/fio)

- read/write, bandwidth/latency, DRAM vs RPMEM (devdax/fsdax)
- numjobs, blocksize, iodepth, readwrite

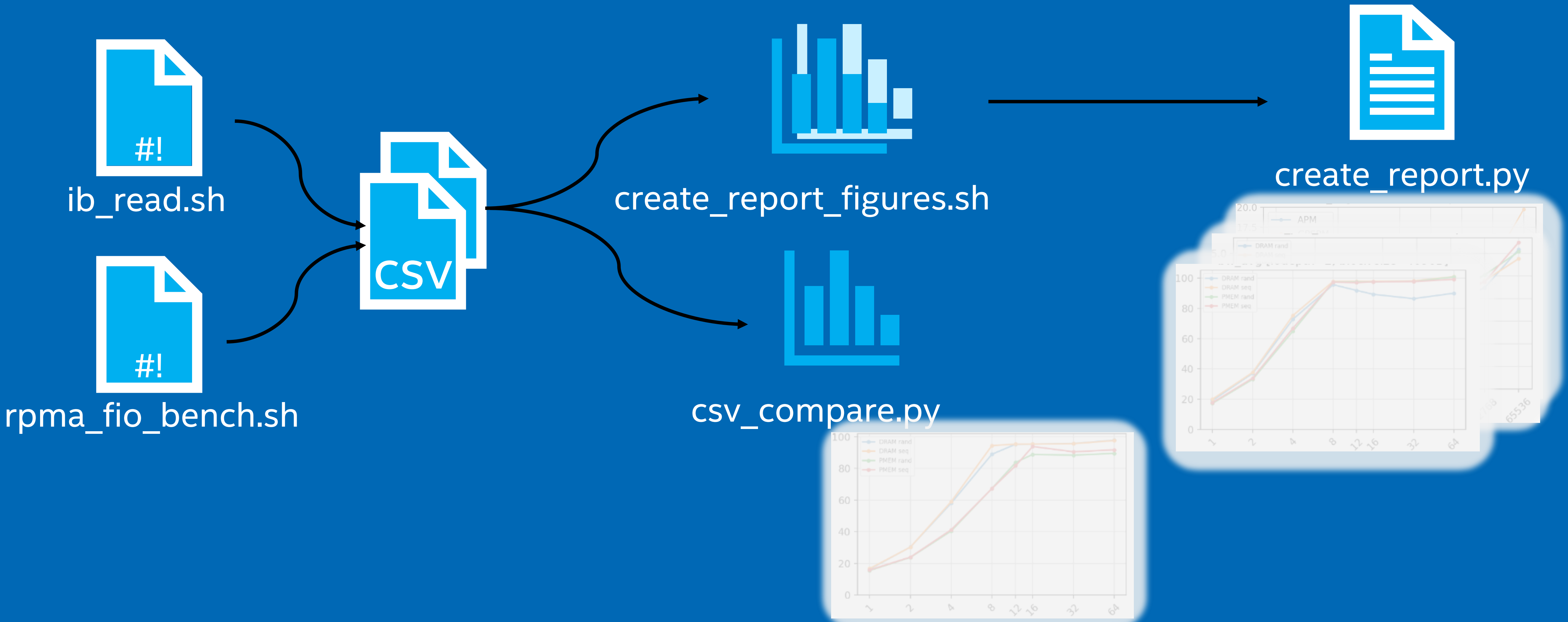
github.com/pmem/rpma/tree/master/tools/perf

- **`rpma_fio_bench.sh`** – to collect performance data
  - Fio job files templates
- **`csv_compare.py`** for results comparison (research, manual analysis)
- **`create_report.sh`** for comprehensive performance report
  - report template could be adjusted

pmem.io/rpma/

- Performance – Tuning - for best configuration practices
- Direct Write to PMem - for step-by-step how to achieve RPMEM-readiness

RPMem benchmarking process

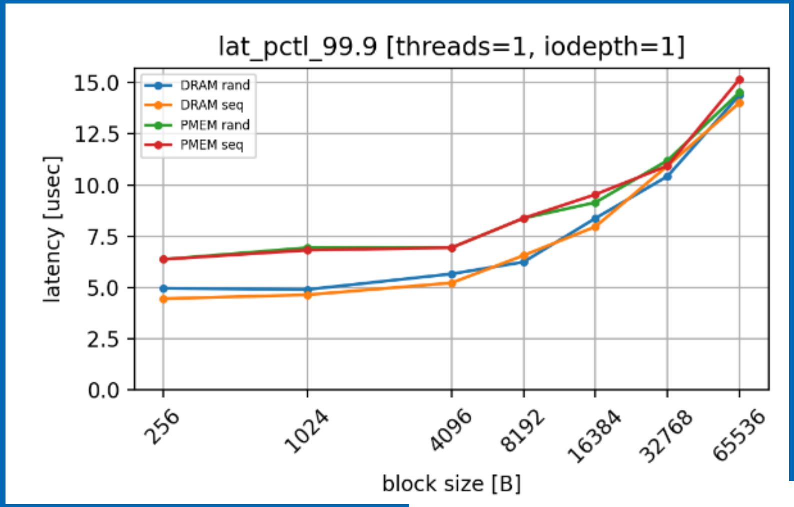


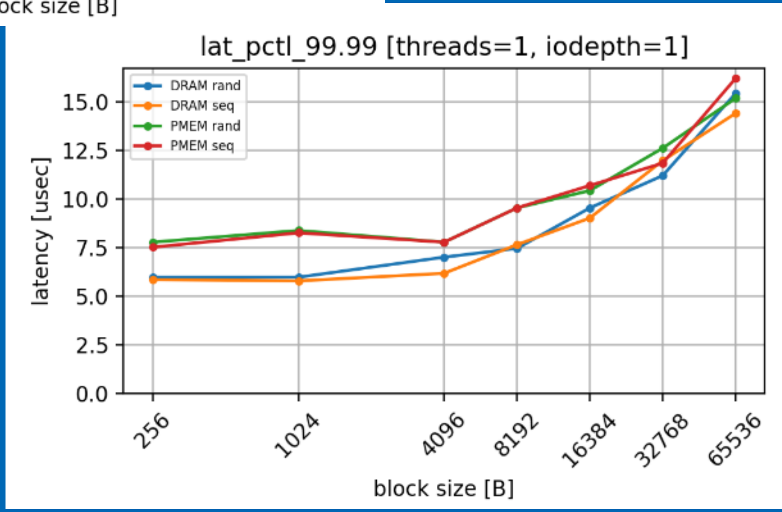
https://github.com/pmem/rpma/tree/master/tools/perf

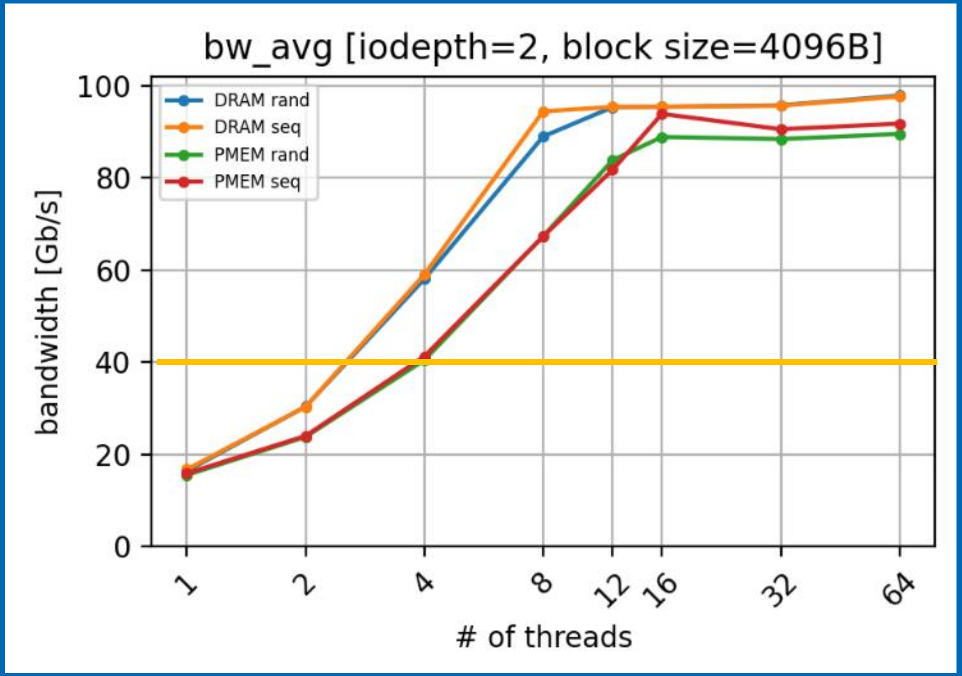
RPMem performance

How fast RPMem can be?

Write to Remote Persistent Memory

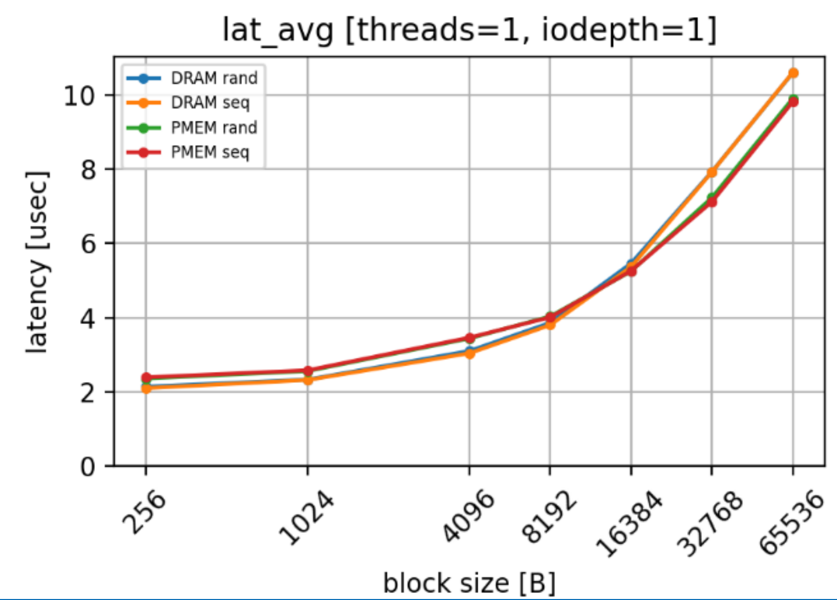


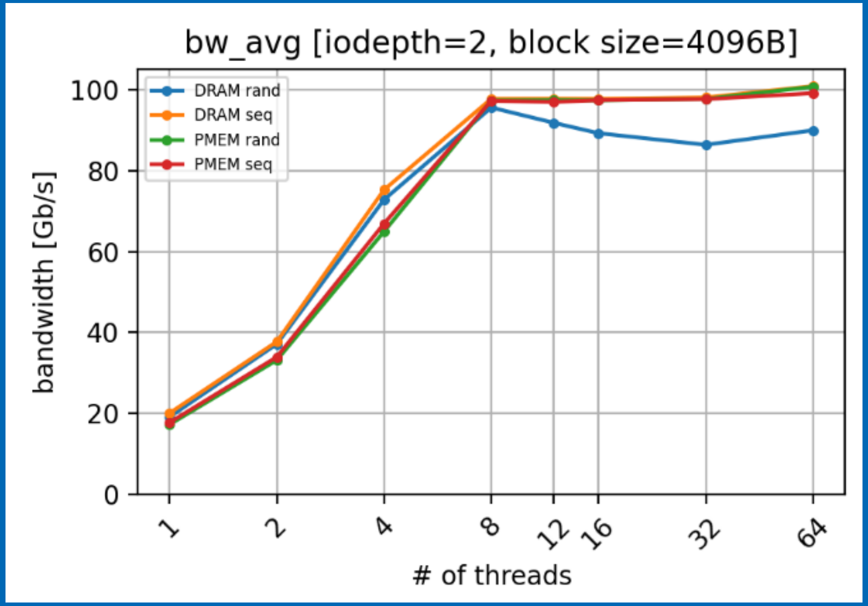




See backup for workloads and configurations. Results may vary. (Config 4.)

RDMA-based remote memory read





See backup for workloads and configurations. Results may vary. (Config 4.)

Backup

System configuration

Backup - System Configuration

Config 1 (c1) - 2 nodes with Intel® Xeon® 2nd Gen Scalable Processor (24c) (HT off, Intel® Speed Step enabled, Intel® Turbo Boost Technology enabled) with DRAM: (per socket) 6 slots / 32GB / 2666 MT/s, (384GB DRAM total per system) running CentOS Linux release 7.6.1810, kernel 3.10.0-1062.7.1.x86_64, 100GbE Mellanox CX-5, FIO version 3.14, DRBD version 9.11.0-1.el7, Intel® SSD DC P3700 Series 400GB. Production released BKC, invulnerable to all known to date „speculative execution" CVEs, test by Intel on 1/23/2020.

Config 2 (c2) - 2 nodes with Intel® Xeon® 2nd Gen Scalable Processor (24c) (HT off, Intel® Speed Step enabled, Intel® Turbo Boost Technology enabled) with DRAM: (per socket) 6 slots / 32GB / 2666 MT/s, (384GB DRAM total per system) running CentOS Linux release 7.6.1810, kernel 3.10.0-1062.7.1.x86_64, 100GbE Mellanox CX-5, rdma-core 22.1-3.el7, FIO version 3.14, DRBD version 9.11.0-1.el7, RDMA transport 2.0.13, Intel® SSD DC P3700 Series 400GB. Production released BKC, invulnerable to all known to date „speculative execution" CVEs, test by Intel on 1/23/2020.

Config 3 (c3) - 2 nodes with Intel® Xeon® 2nd Gen Scalable Processor (24c) (HT off, Intel® Speed Step enabled, Intel® Turbo Boost Technology enabled) with DRAM: (per socket) 6 slots / 32GB / 2666 MT/s, PMem: (per socket) 6 slots/256GB Intel® Optane™ PMem 100 series modules (384GB DRAM, 3072GB PMem total per system) running CentOS Linux release 7.6.1810, kernel 3.10.0-1062.7.1.x86_64, 100GbE Mellanox CX-5, rdma-core 22.1-3.el7 , PMDK 1.7, libfabric v1.7.0, Production released BKC, invulnerable to all known to date „speculative execution" CVEs, RNIC Work Queue Size == 384 elements, test by Intel on 1/23/2020.

Backup - System Configuration

Config 4 (c4) - 2 nodes with Intel® Xeon® 2nd Gen Scalable Processor (24c) (HT off, Intel® Speed Step enabled, Intel® Turbo Boost Technology enabled) with DRAM: (per socket) 6 slots / 32GB / 2666 MT/s, PMem: (per socket) **6 slots / 256GB Intel® Optane™ PMem 100 series modules** (384GB DRAM, 3072GB PMem total per system) running CentOS Linux release: 8.2, kernel: 4.18.0-193.28.1.el8_2.x86_64, 100GbE Mellanox CX-5, rdma-core: 51mlnx1 1.51258 libibverbs1.10.30.0, PMDK 1.6.1, FIO version fio-3.23-419-g79ae6,
Production released BKC, invulnerable to all known to date „speculative execution" CVEs, test by Intel on 2/23/2021.

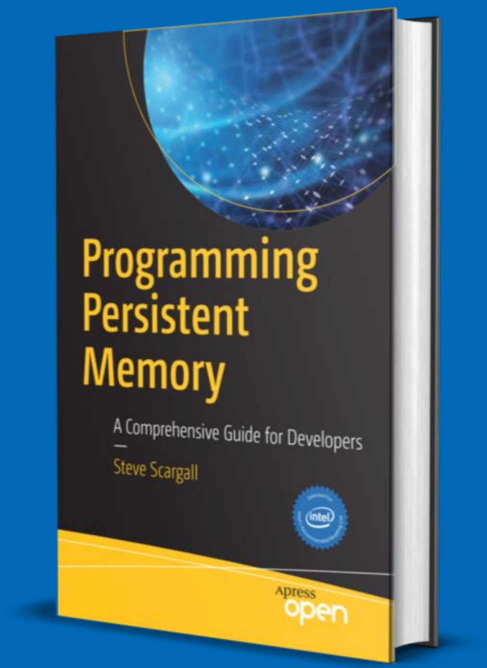
Resources

RPMem whitepapers and tutorials

- software.intel.com/en-us/articles/persistent-memory-replication-over-traditional-rdma-part-1-understanding-remote-persistent
- software.intel.com/content/www/us/en/develop/articles/white-paper-remote-pmem-over-traditional-rdma.html
- pmem.io/rpma

RPMem chapter in Programming Persistent Memory book

- pmem.io/book



Current RPMem development in PMDK

- github.com/pmem/rpma
  - github.com/pmem/rpma/tree/master/examples
    github.com/pmem/rpma/tree/master/tools/perf
- github.com/pmem/fio

Q & A

Thank You!

pmem.io/rpma

github.com/pmem/rpma

intel®

SPDK, PMDK, Intel® Performance Analyzers | **Virtual Forum**