DAOS & PMEM Usage

Johann Lombardi, Intel AXG

Notices and Disclaimers

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration.

No product or component can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. For more complete information about performance and benchmark results, visit http://www.intel.com/benchmarks .

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. 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 complete information visit http://www.intel.com/benchmarks .

Intel Advanced Vector Extensions (Intel AVX) provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more at http://www.intel.com/go/turbo.

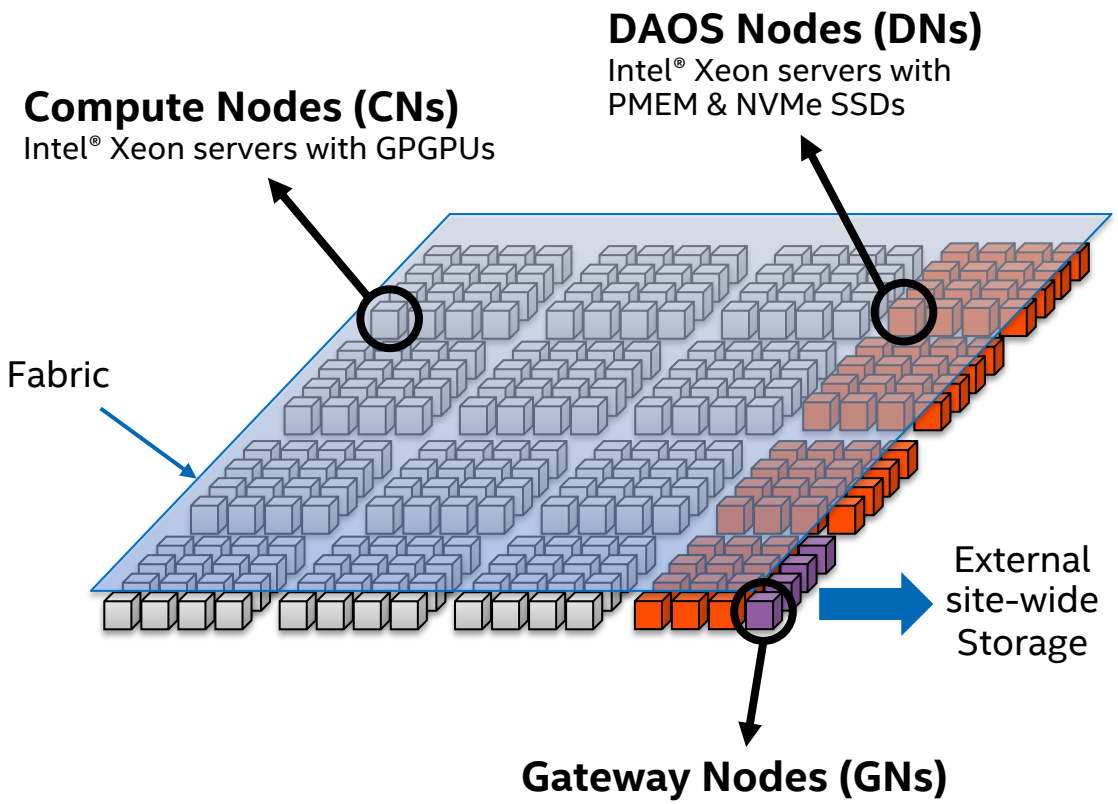
Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

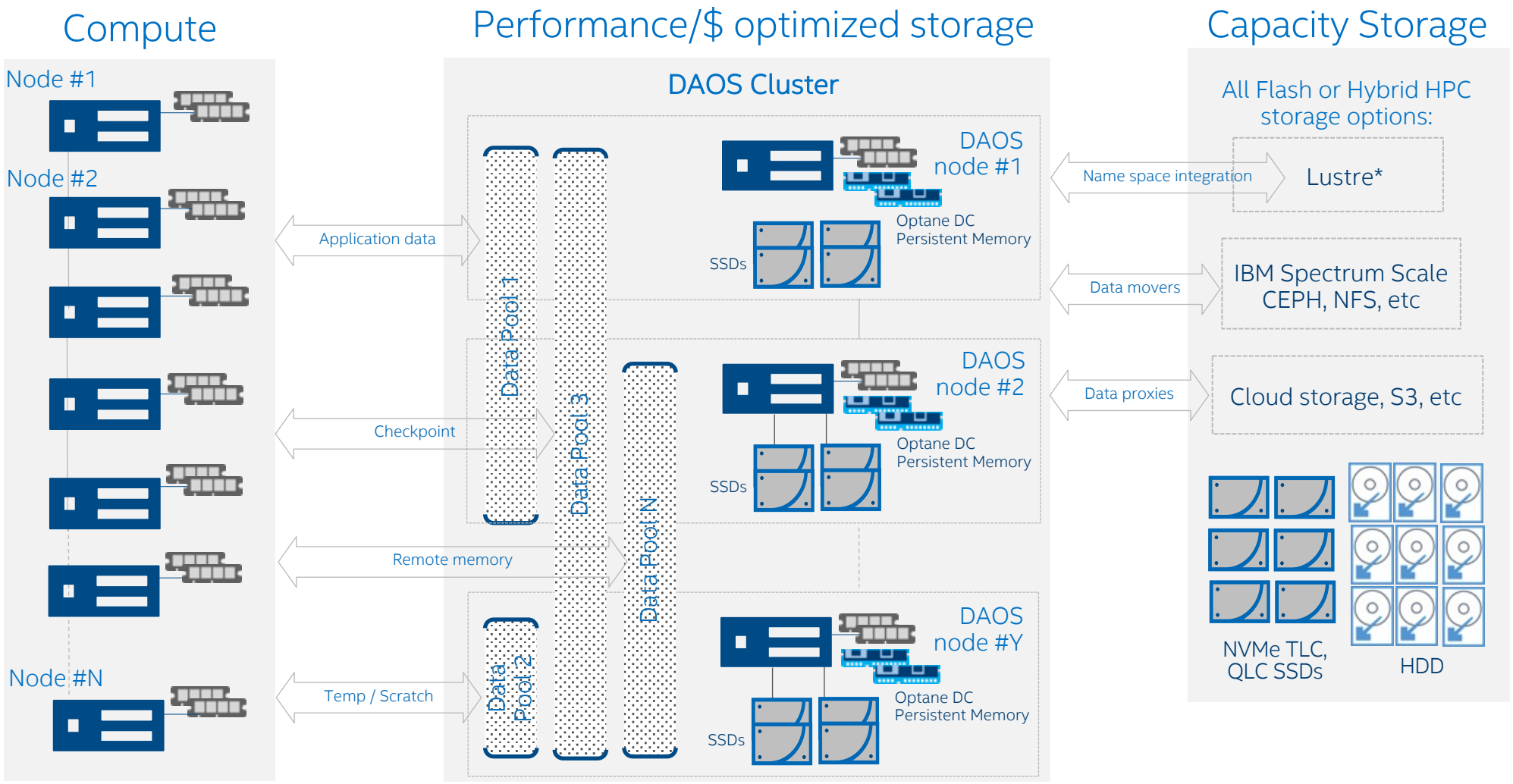
What is DAOS?

- A new, innovative distributed parallel file system based on Intel® Optane™ Persistent Memory and Intel® TLC and QLC NVMe SSDs
- Overcomes industry bottlenecks in block-based IO and POSIX
- Delivers exceptionally high bandwidth and IOPS on commodity servers, meeting the demands of HPC, AI, and high-performance data analytics
- Can be utilized either as a standalone file system, or as a performance tier integrated with existing storage systems

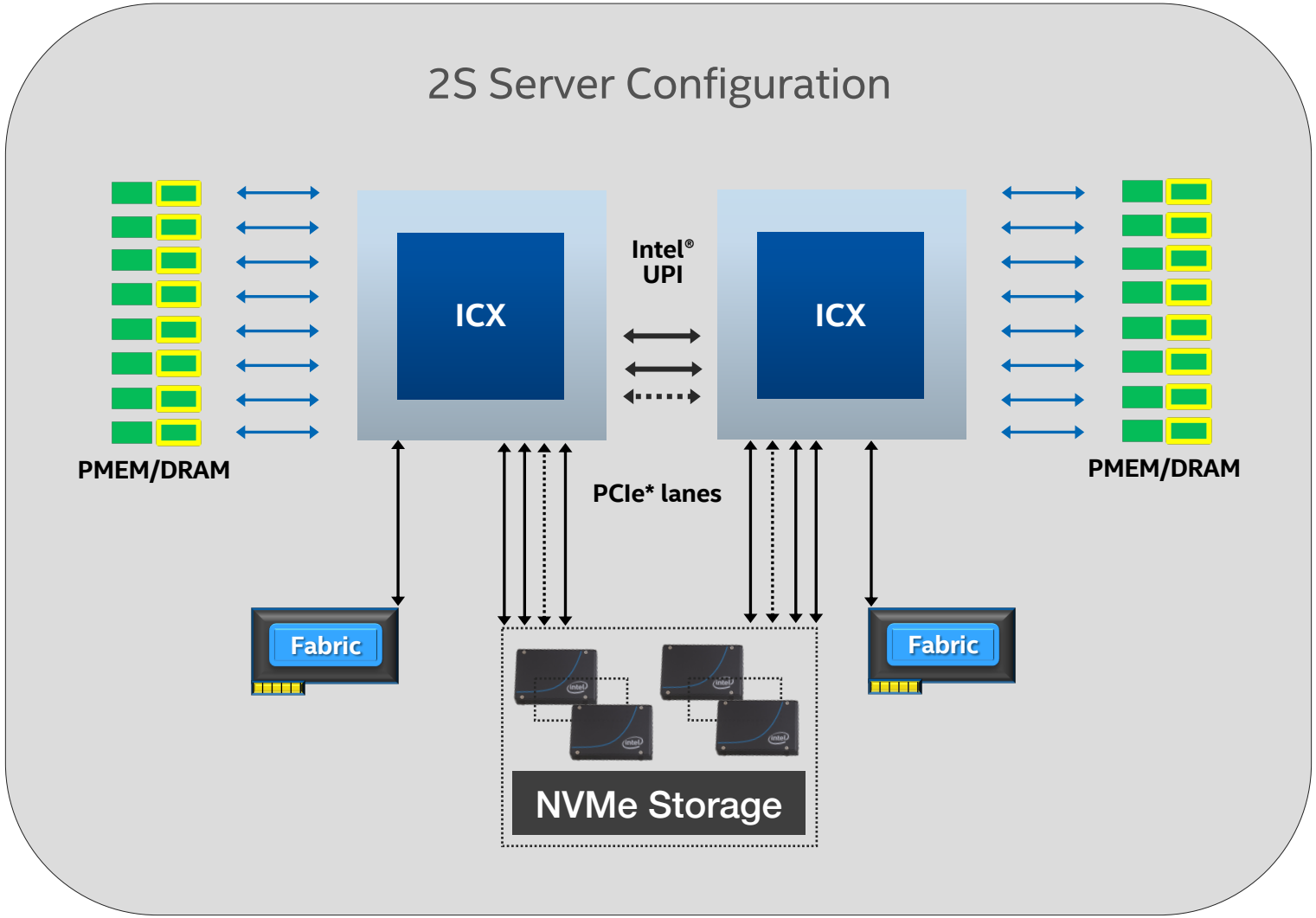


Generational leap forward in storage performance

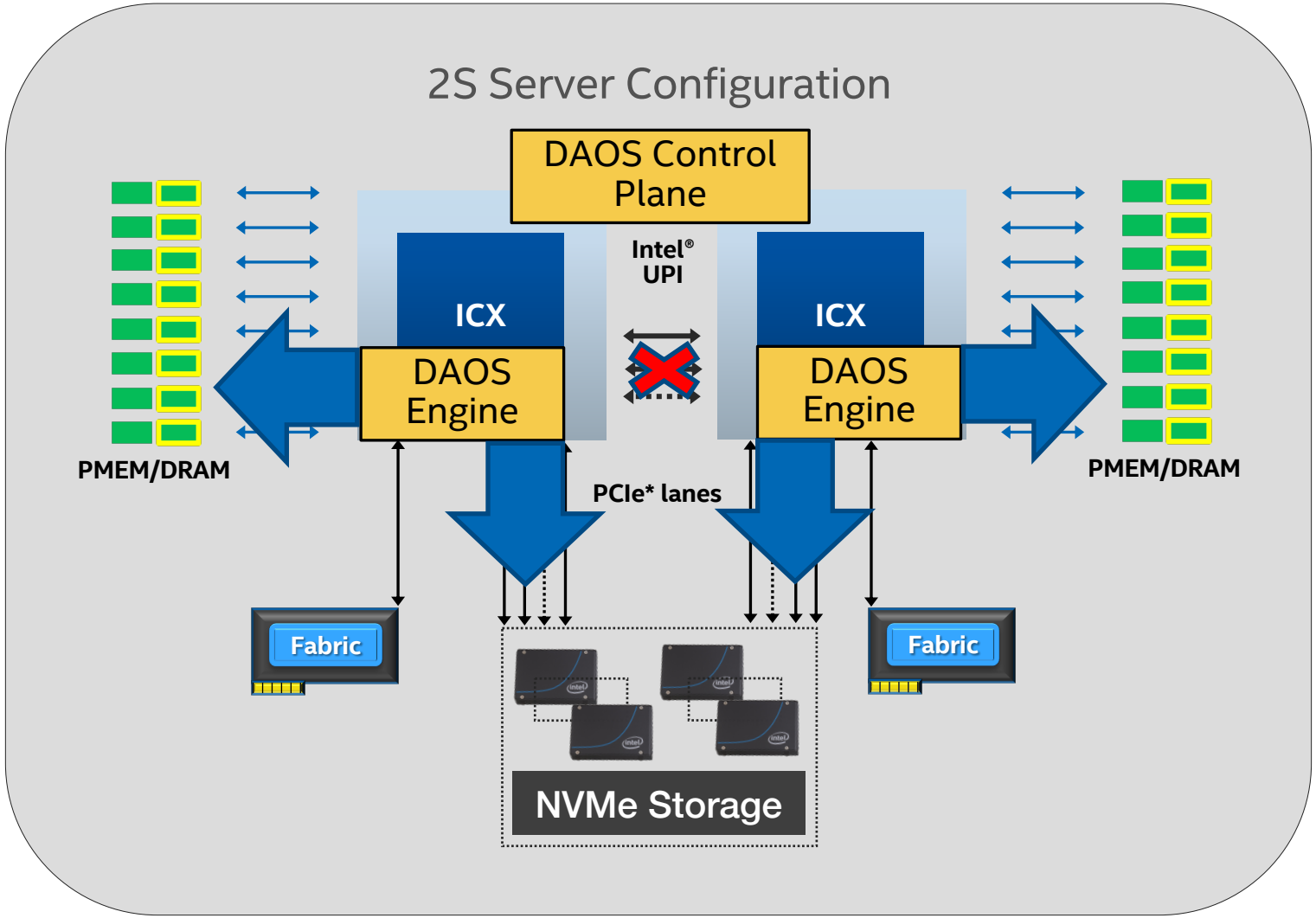
DAOS in the Overall Cluster Architecture



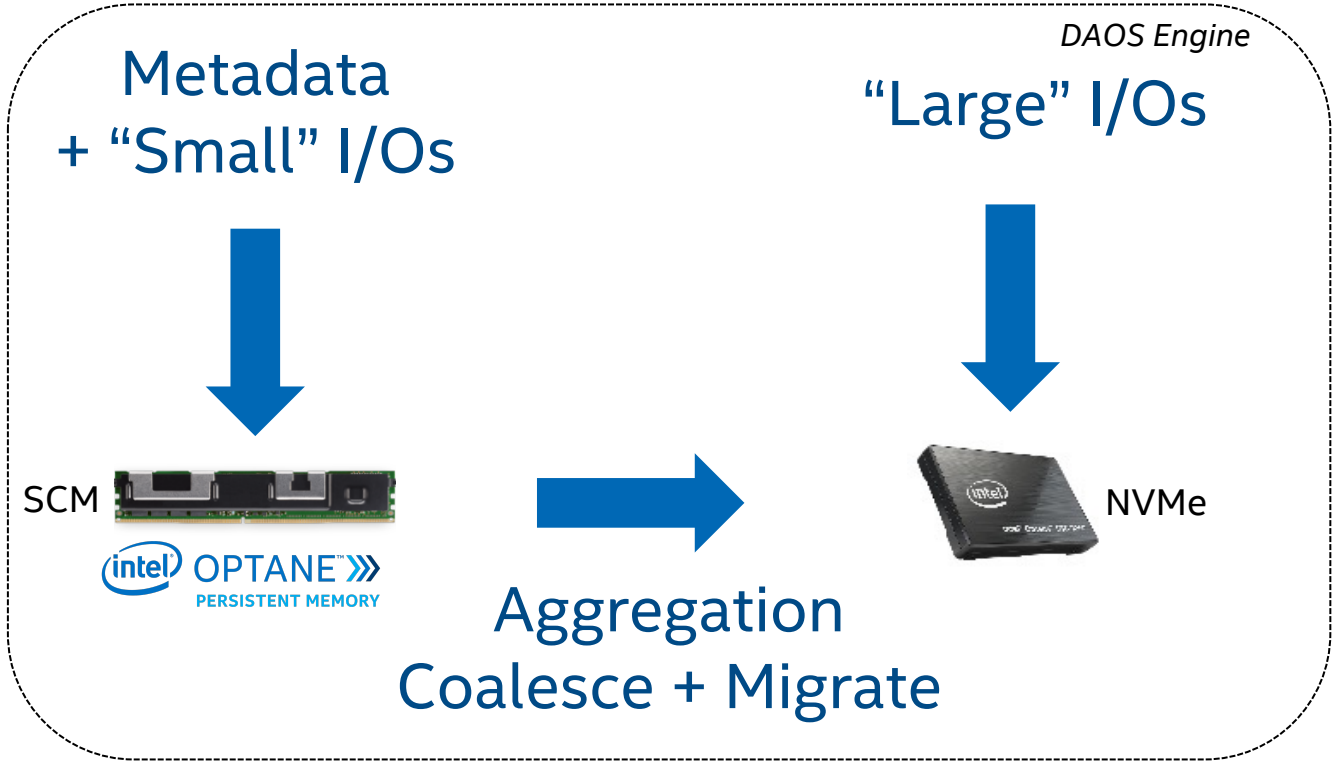
DAOS Node Design



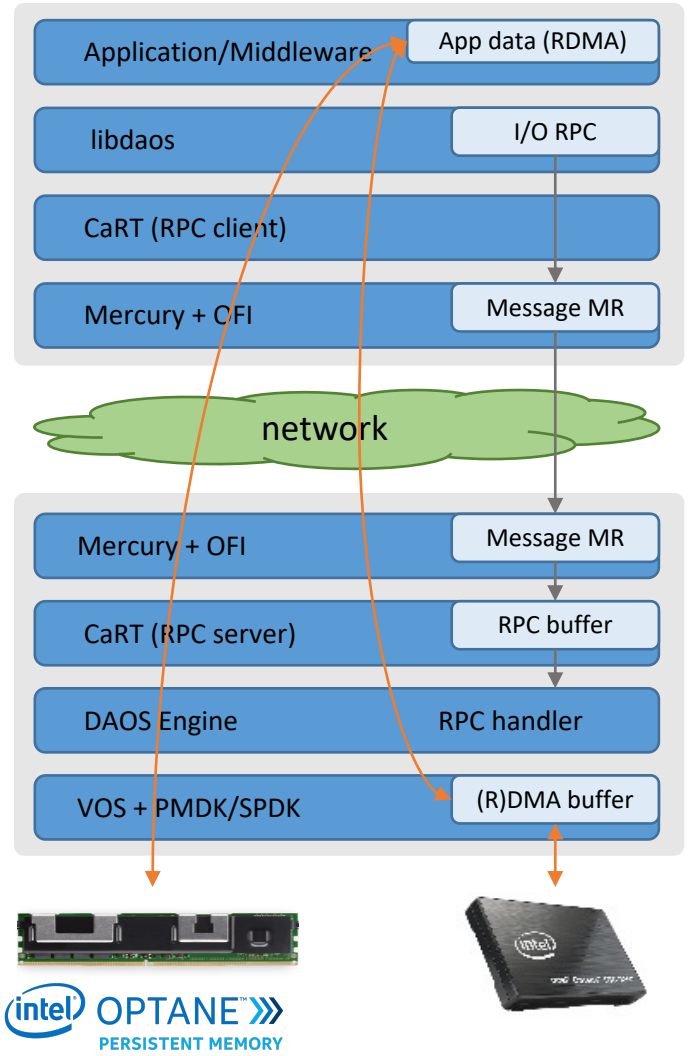
DAOS Node Design



Engine: Media Management



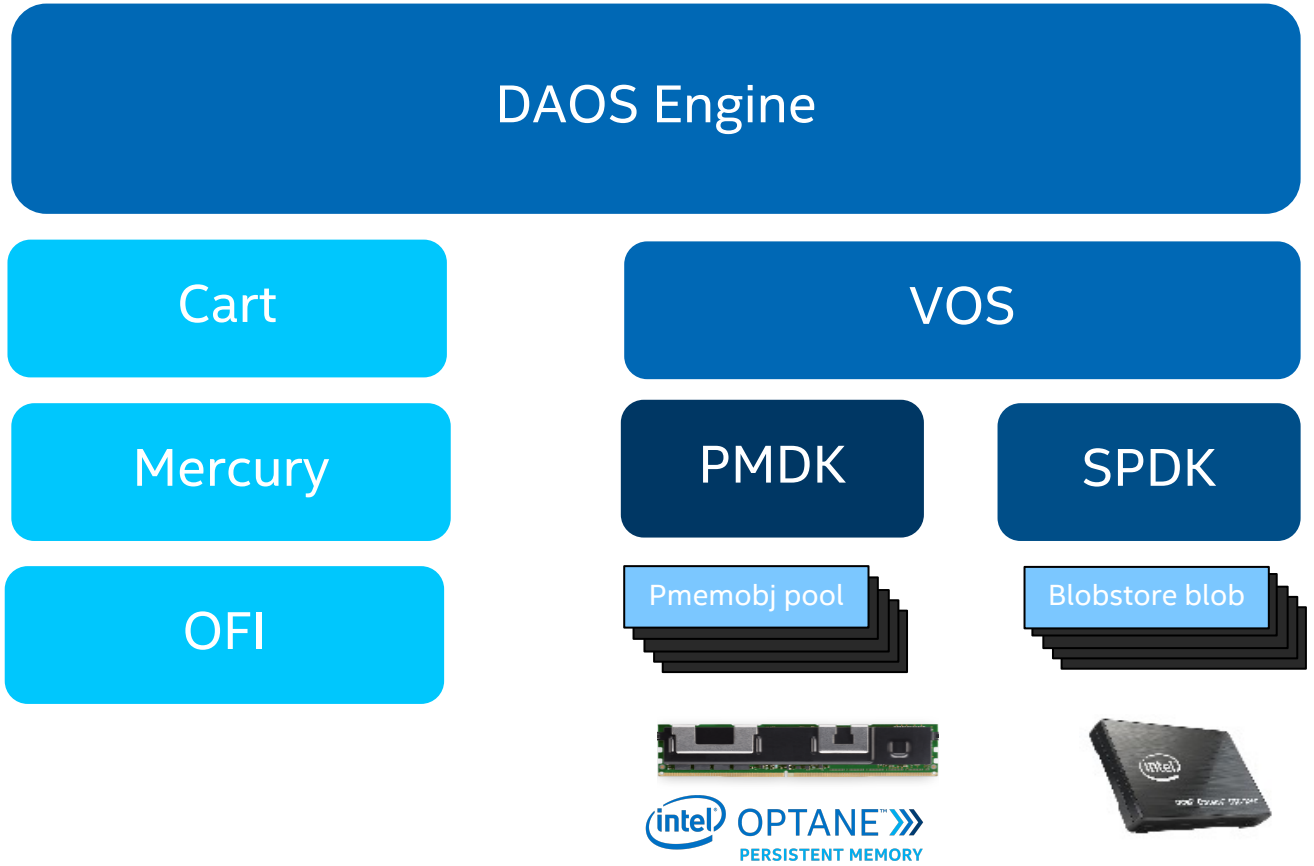
DAOS I/O Flow



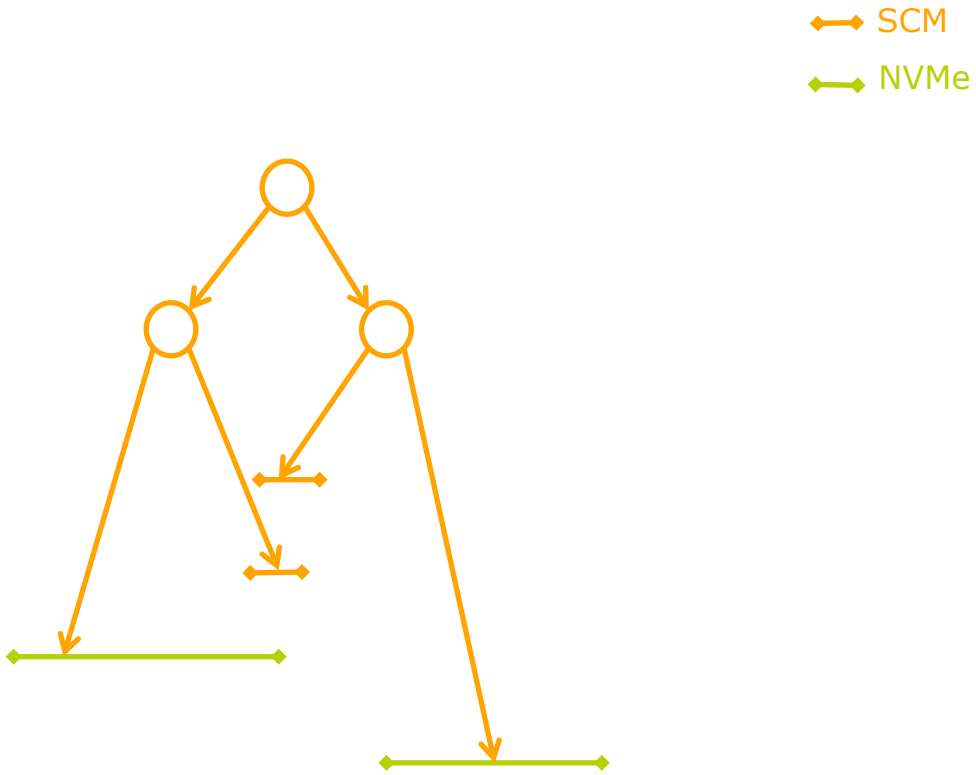
- No memory copy for RDMA capable network
- Client registers MR for each bulk I/O (MR register can be expensive on some providers)
- Server currently registers MR for each I/O, but it can be optimized by pre-registering all DMA buffers and persistent memory as RDMA buffers

Data flow

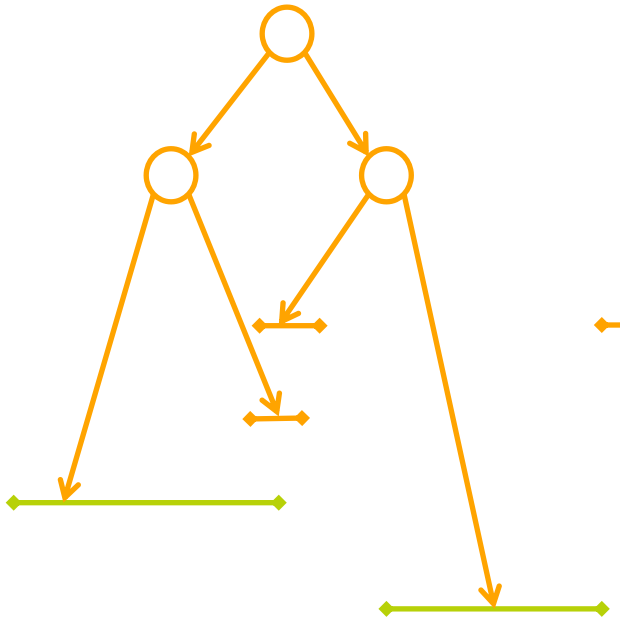
Engine Software Stack



I/O Operation Flow



I/O Operation Flow



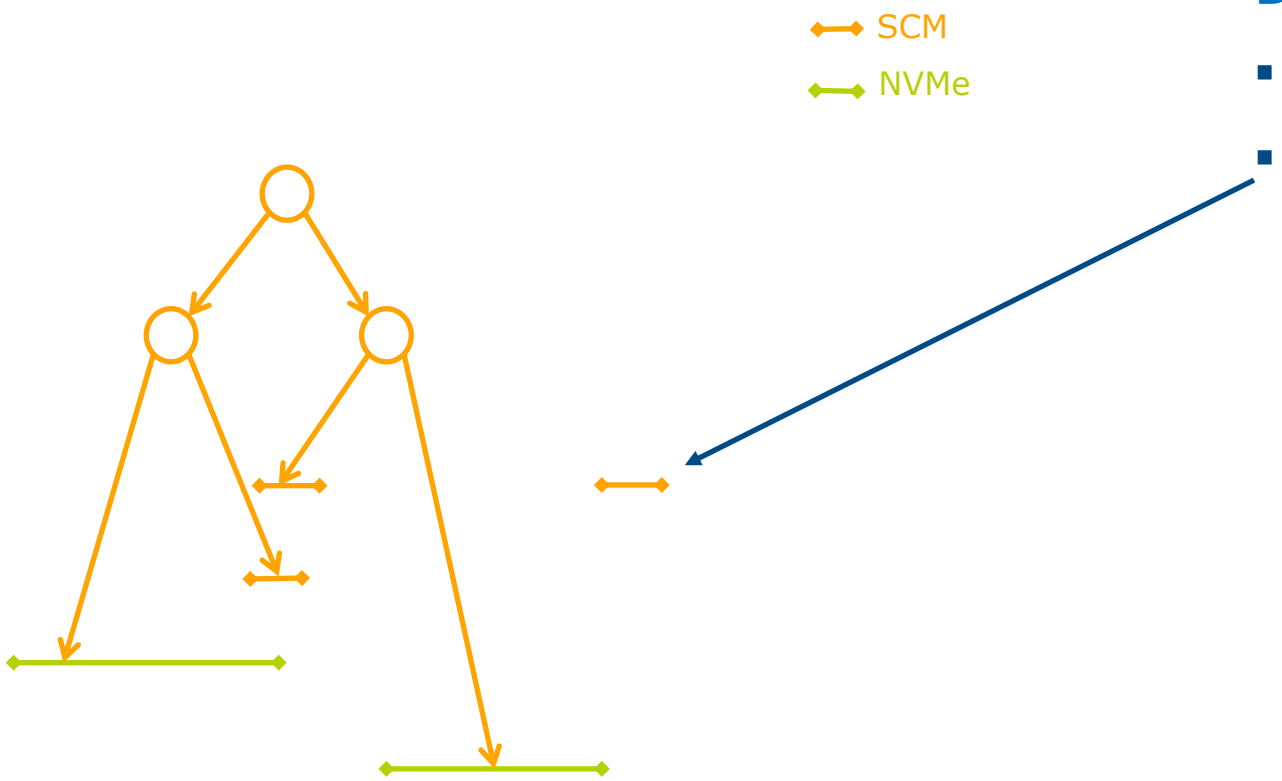
SCM

NVMe

DAOS XStream

- pmemobj_reserve() new buffer

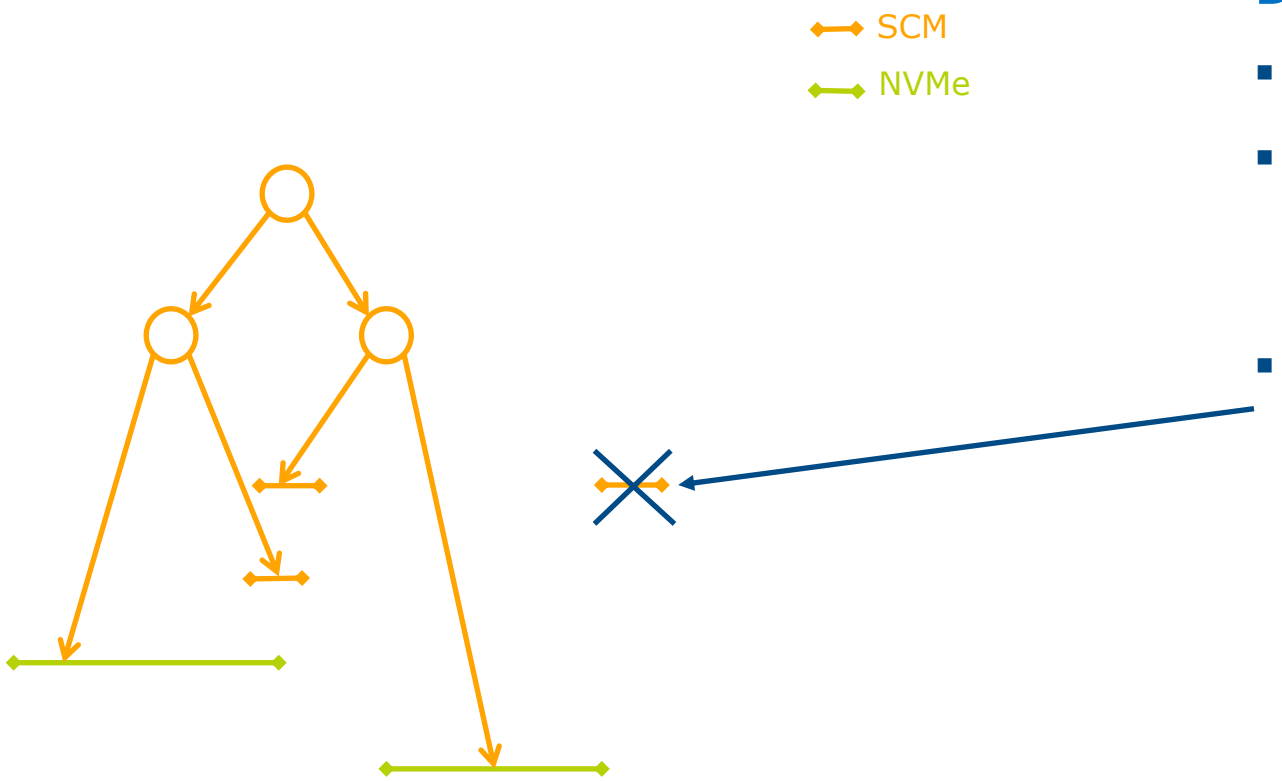
I/O Operation Flow



DAOS XStream

- pmemobj_reserve() new buffer
- Start RDMA transfer to newly allocated buffer
  - Switch to other ULT until completion is reported

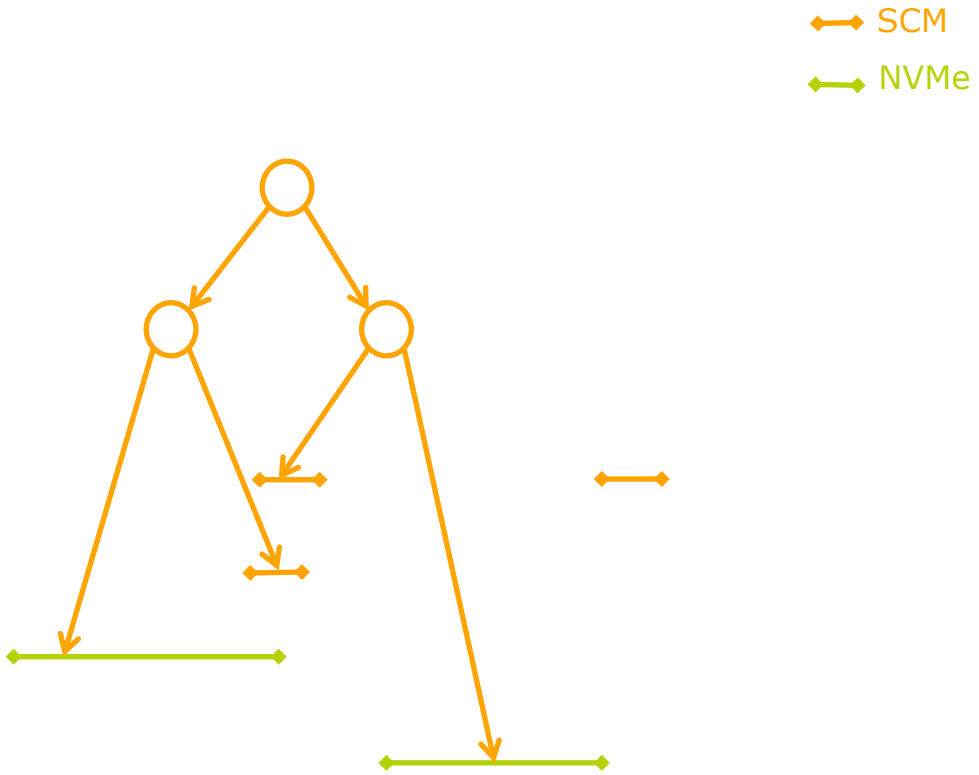
I/O Operation Flow



DAOS XStream

- pmemobj_reserve() new buffer
- Start RDMA transfer to newly allocated buffer
  - Switch to other ULT until completion is reported
- If RDMA transfer failed, free buffer with pmemobj_cancel()

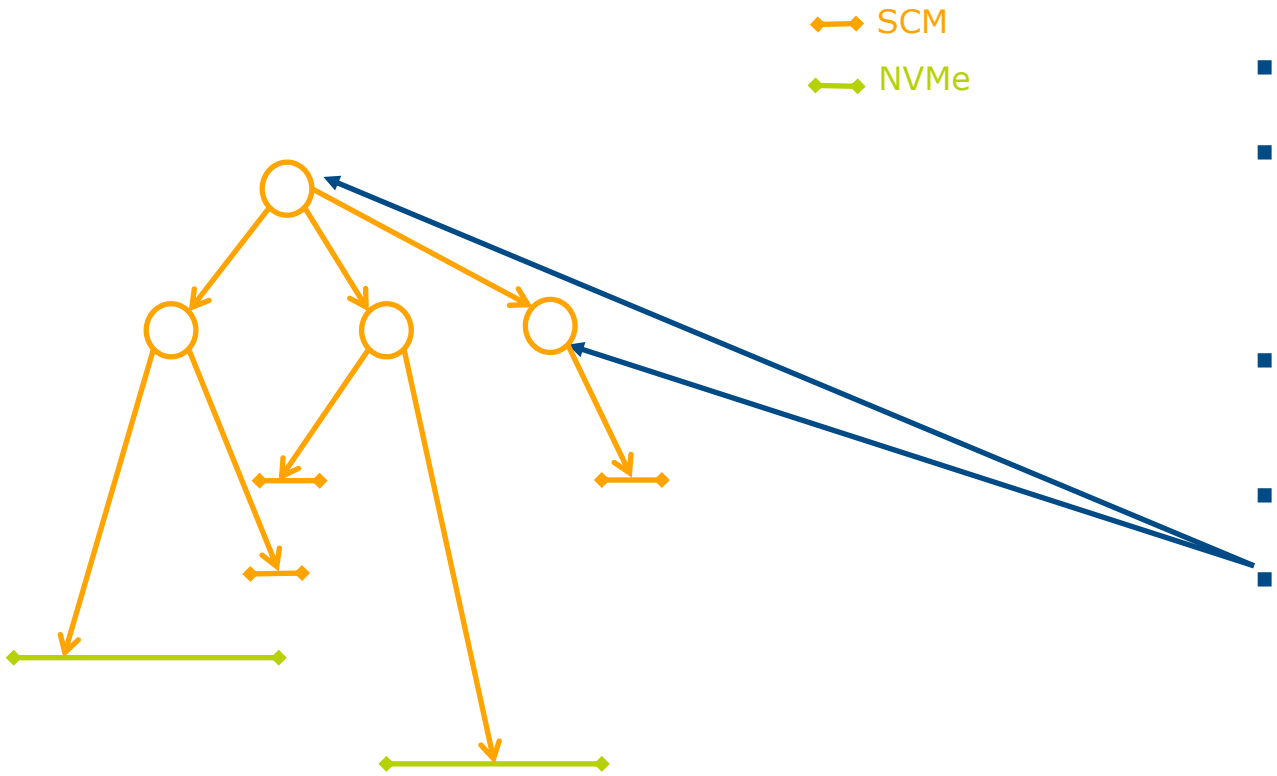
I/O Operation Flow



DAOS XStream

- pmemobj_reserve() new buffer
- Start RDMA transfer to newly allocated buffer
  - Switch to other ULT until completion is reported
- If RDMA transfer failed, free buffer with pmemobj_cancel()
- Otherwise, start pmemobj transaction

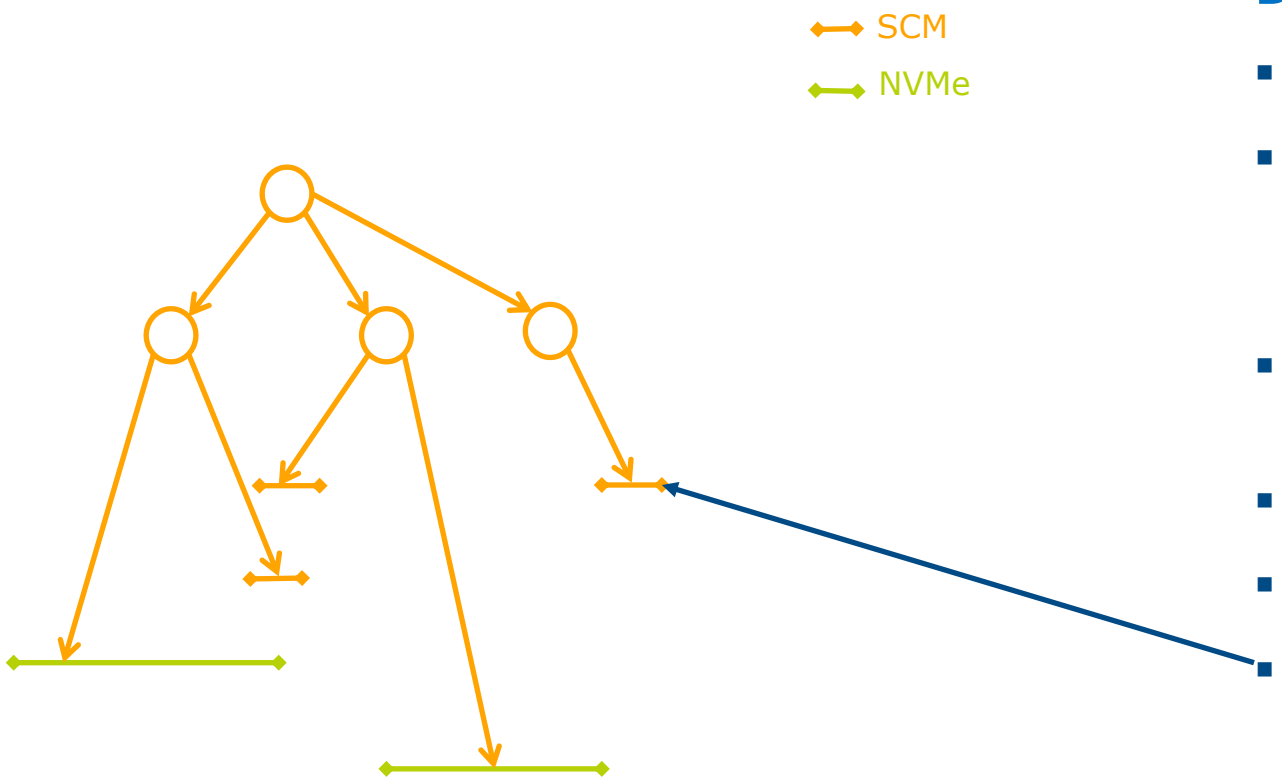
I/O Operation Flow



DAOS XStream

- pmemobj_reserve() new buffer
- Start RDMA transfer to newly allocated buffer
  - Switch to other ULT until completion is reported
- If RDMA transfer failed, free buffer with pmemobj_cancel()
- Otherwise, start pmemobj transaction
- Modify index to insert new extent

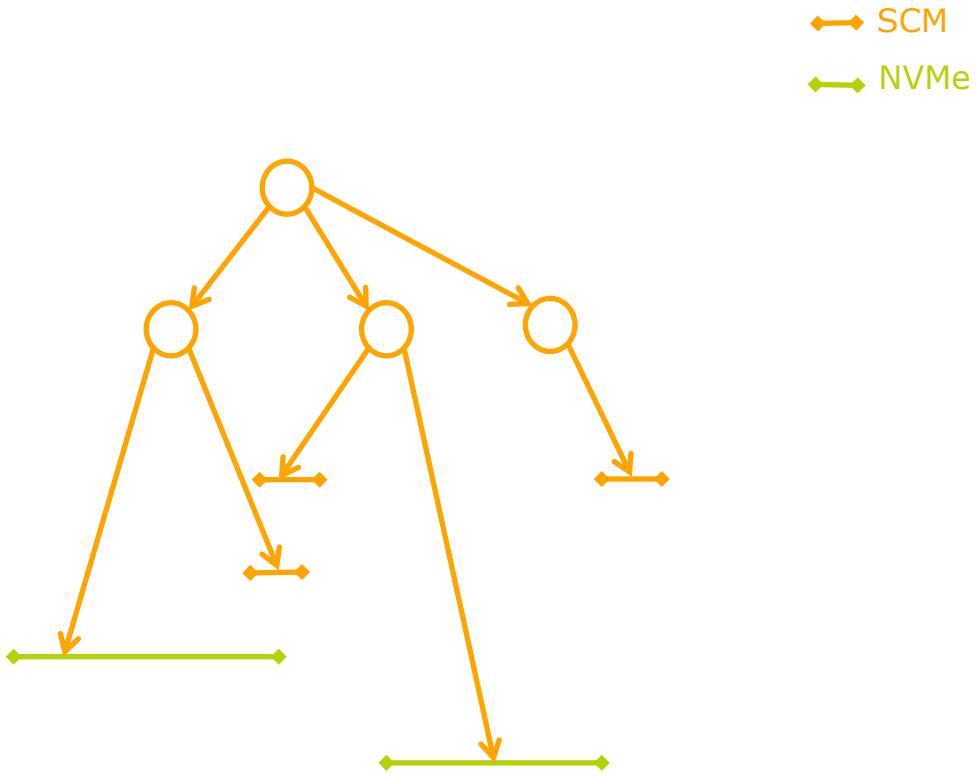
I/O Operation Flow



DAOS Xstream

- pmemobj_reserve() new buffer
- Start RDMA transfer to newly allocated buffer
  - Switch to other ULT until completion is reported
- If RDMA transfer failed, free buffer with pmemobj_cancel()
- Otherwise, start pmemobj transaction
- Modify index to insert new extent
- pmemobj_tx_publish()

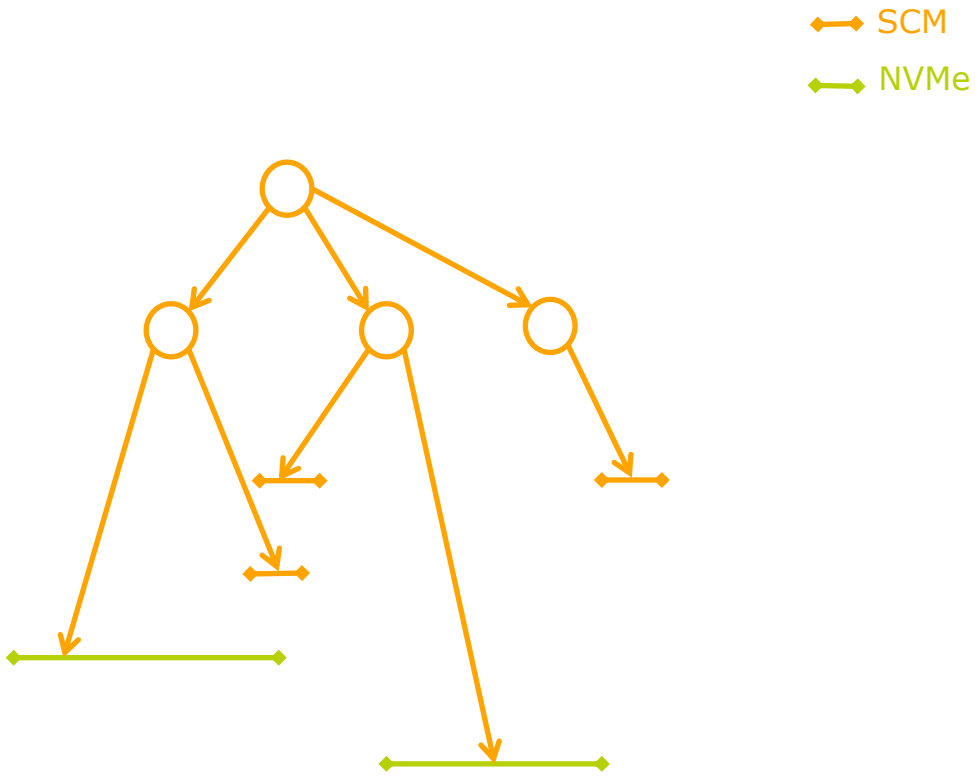
I/O Operation Flow



DAOS Xstream

- pmemobj_reserve() new buffer
- Start RDMA transfer to newly allocated buffer
  - Switch to other ULT until completion is reported
- If RDMA transfer failed, free buffer with pmemobj_cancel()
- Otherwise, start pmemobj transaction
- Modify index to insert new extent
- pmemobj_tx_publish()
- Commit pmemobj transaction

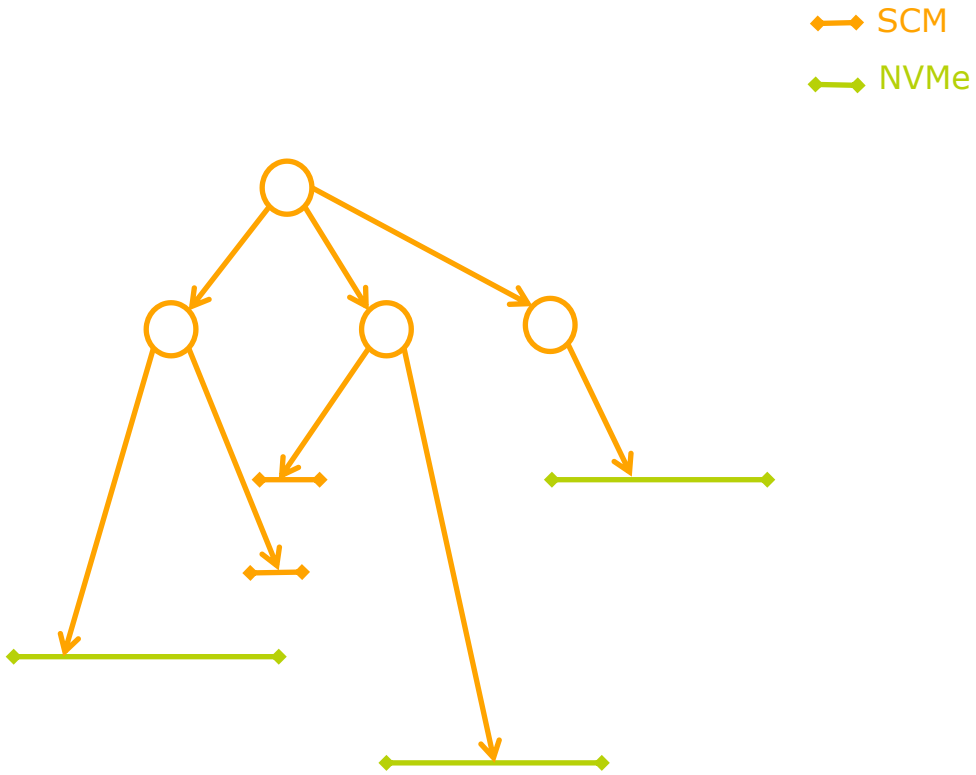
I/O Operation Flow



DAOS Xstream

- pmemobj_reserve() new buffer
- Start RDMA transfer to newly allocated buffer
  - Switch to other ULT until completion is reported
- If RDMA transfer failed, free buffer with pmemobj_cancel()
- Otherwise, start pmemobj transaction
- Modify index to insert new extent
- pmemobj_tx_publish()
- Commit pmemobj transaction
- Reply to the client

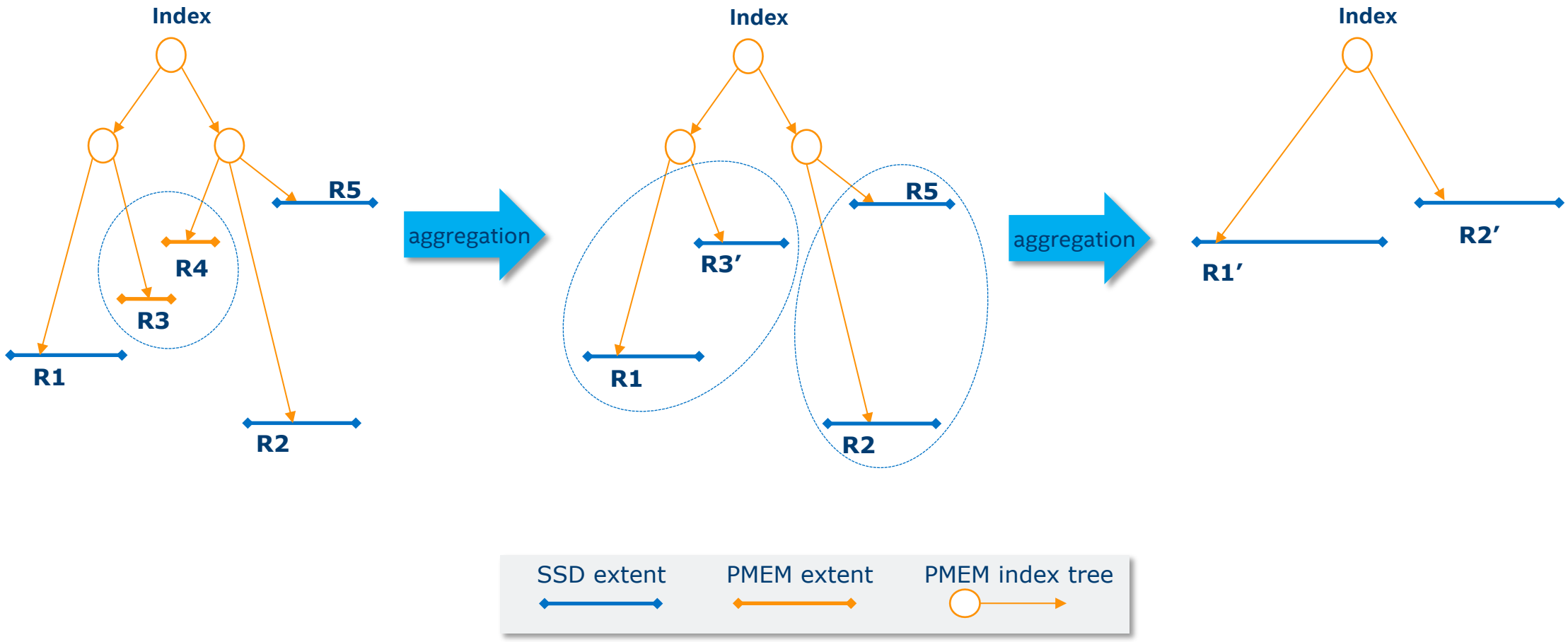
I/O Operation Flow



Extent in NVMe

- Track NVMe allocated/free space in data structures maintained in pmemobj pool
- Same processing flow

Data Aggregation

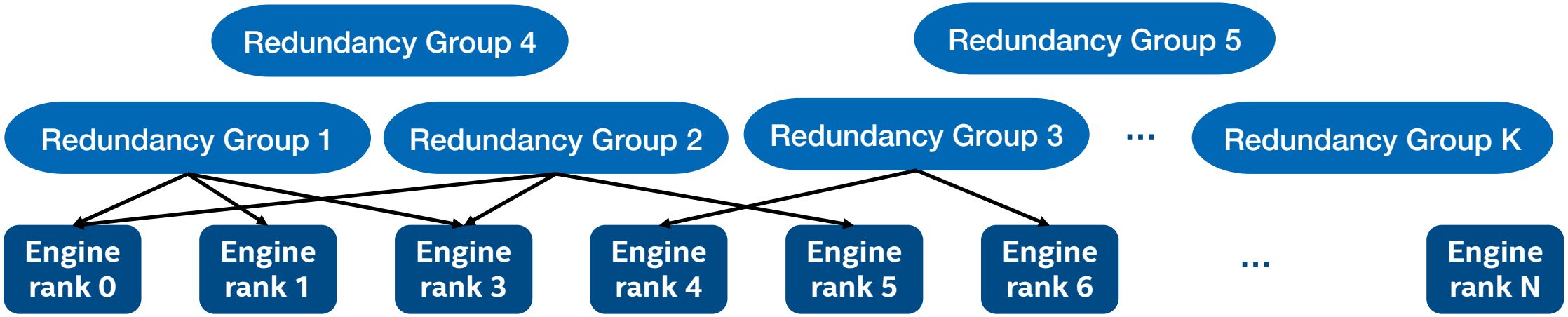


ACID Semantic of DAOS Operations

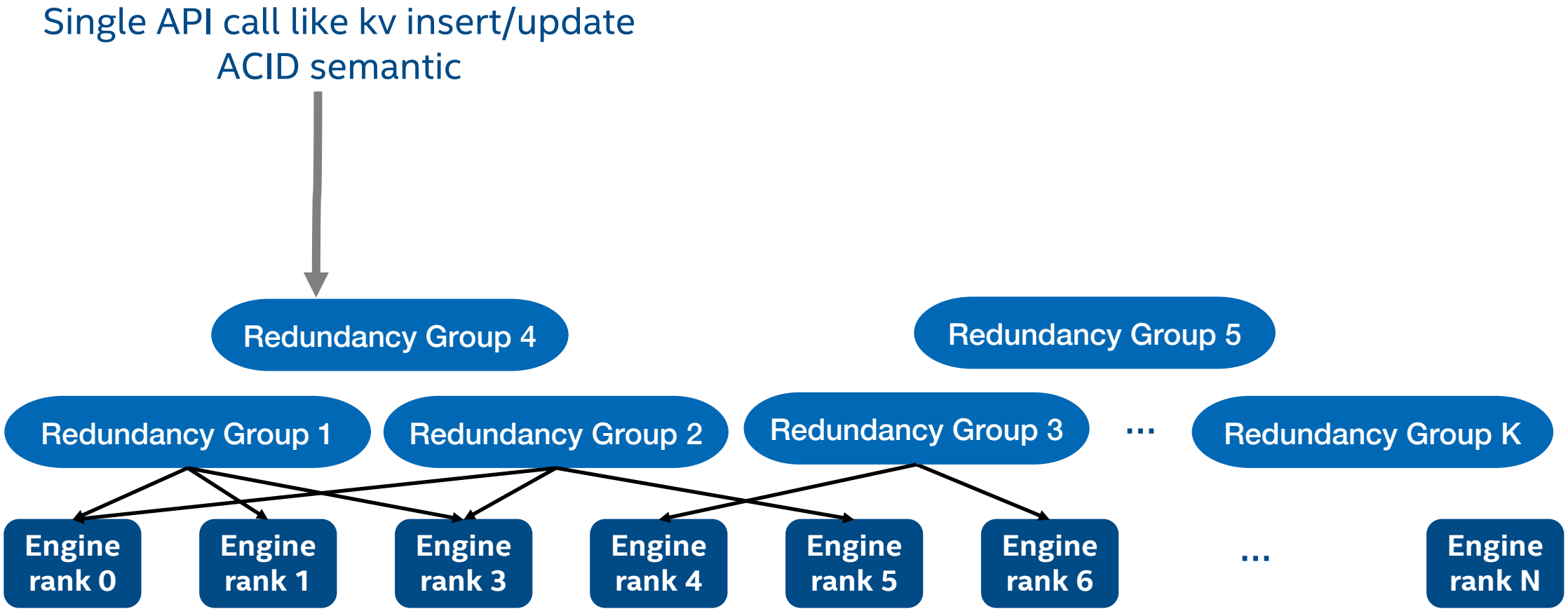
Single API call like kv insert/update
ACID semantic

Engine rank 0 | Engine rank 1 | Engine rank 3 | Engine rank 4 | Engine rank 5 | Engine rank 6 | ... | Engine rank N

Placement & Declustering

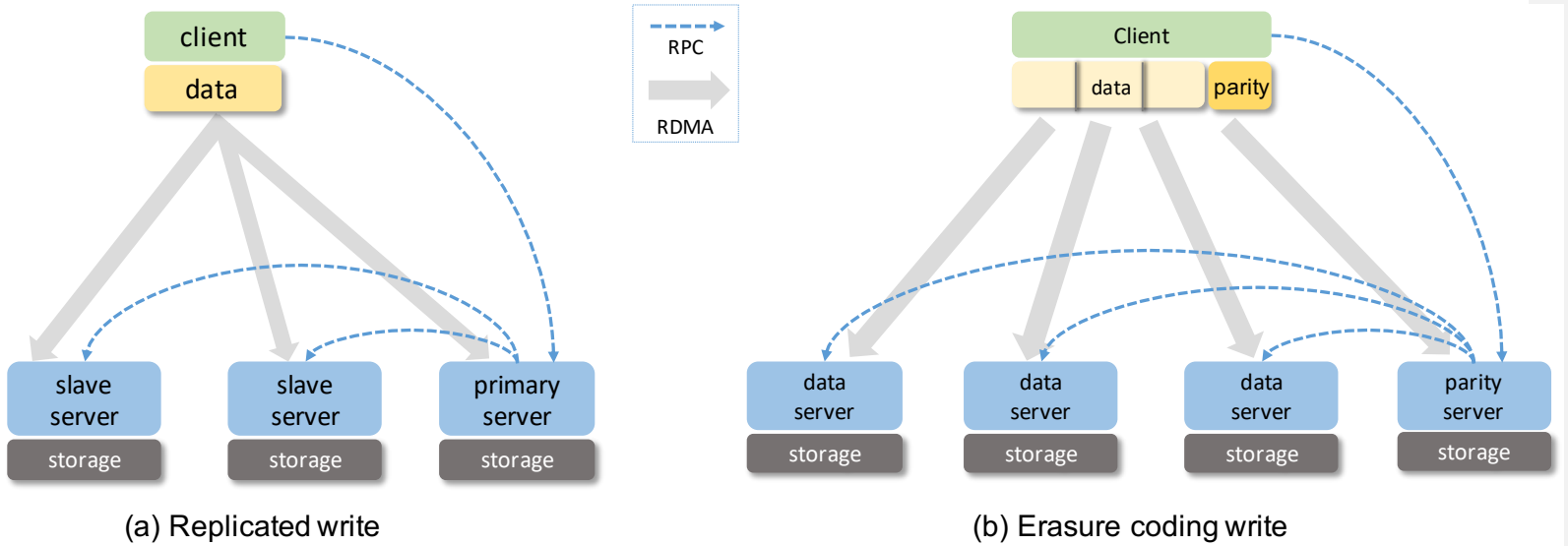


ACID Semantic of DAOS Operations (EC/RP)



Replication & Erasure Code Protocol

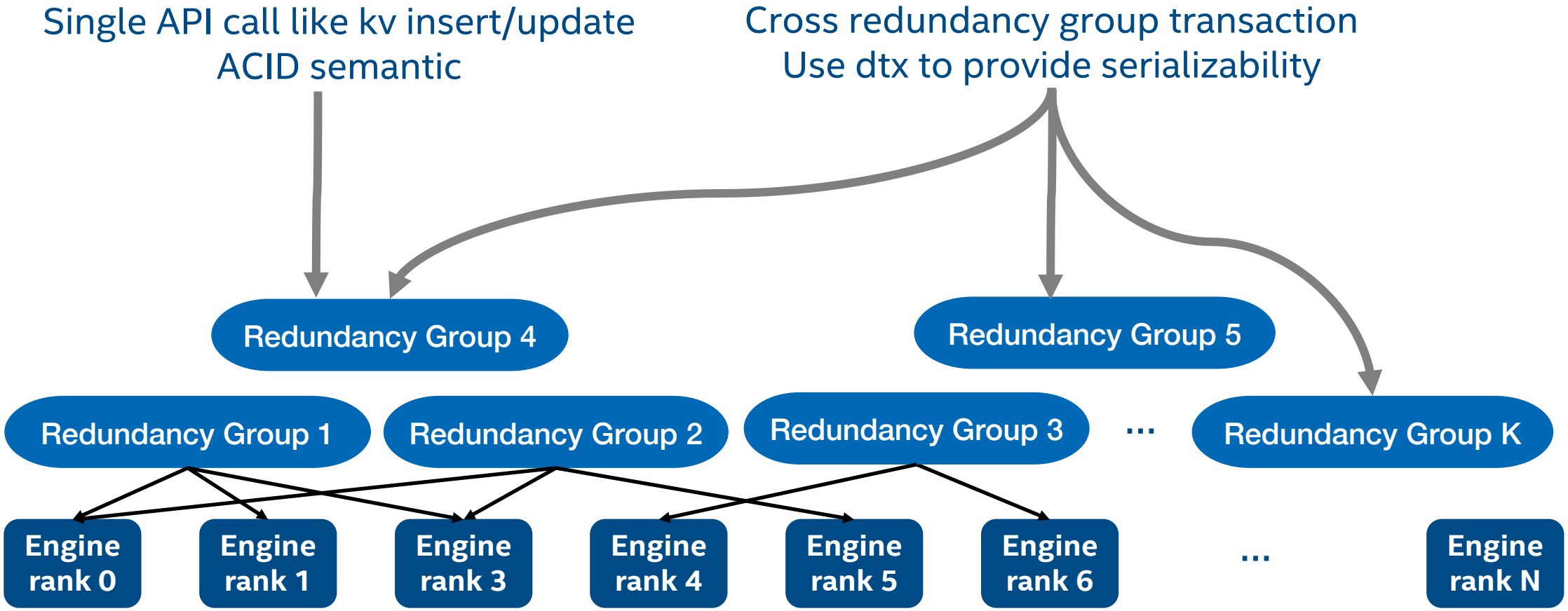
- Atomicity of distributed I/O
  - Replication, Erasure coding
- Two-phase commit based protocol
  - Prepare + commit
  - VOS is a write log
    - Indexed writes
    - Transaction status



(a) Replicated write

(b) Erasure coding write

Concurrency Control



Distributed Transaction

- DAOS distributed transactions (dtx):
  - No locking
  - Based on MVCC
  - Serializable
  - Similarity with Spanner & CockroachDB

```
        daos_tx_open(coh, &th, ...);
restart:
        daos_obj_fetch(..., th, ...);
        daos_obj_update(..., th, ...);
        daos_obj_fetch(..., th, ...);
        daos_obj_update(..., th, ...);
        daos_obj_dkey_punch(..., th, ...);
        rc = daos_tx_commit(th, ...);
        if (rc == -DER_RESTART) {
                daos_tx_restart(th, ...);
                goto restart;
        }
        daos_tx_close(th, ...);
```

Summary

- DAOS Overview
- Details of
  - how DAOS uses PMDK internally to guarantee internal consistency when processing a client request
  - erasure and replication protocol
- Introduction of distributed transactions crossing multiple redundancy groups


intel