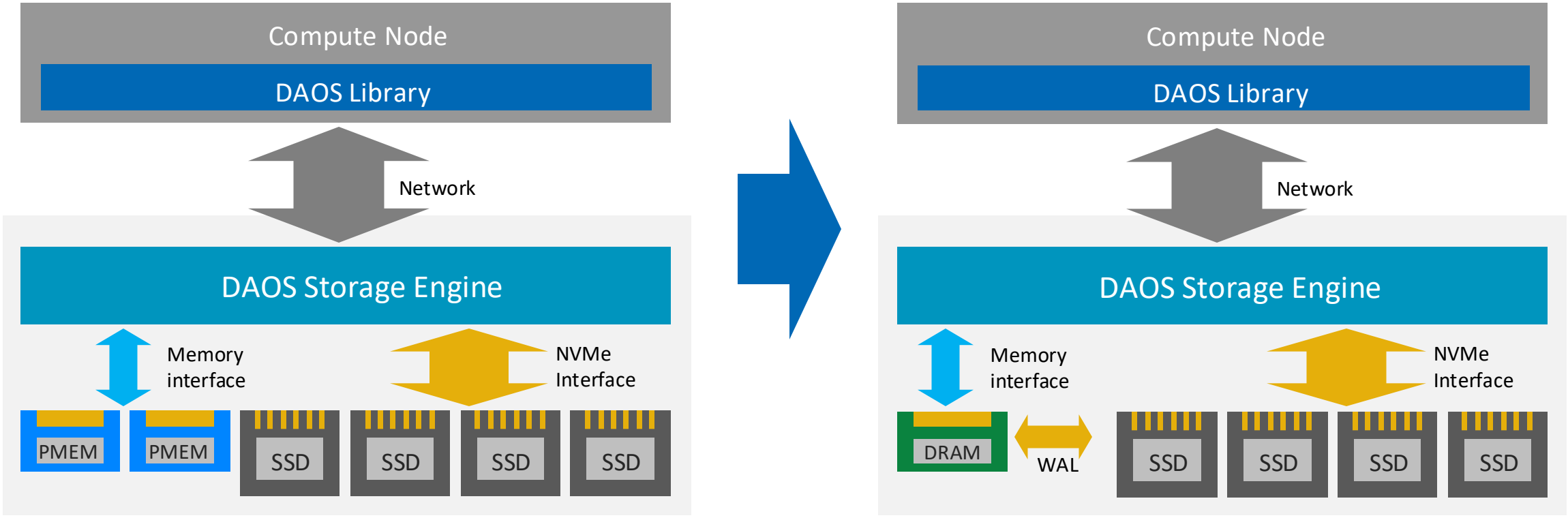
DAOS: Metadata on SSD
Status update

Liang Zhen, Principal Engineer, Intel

DUG24

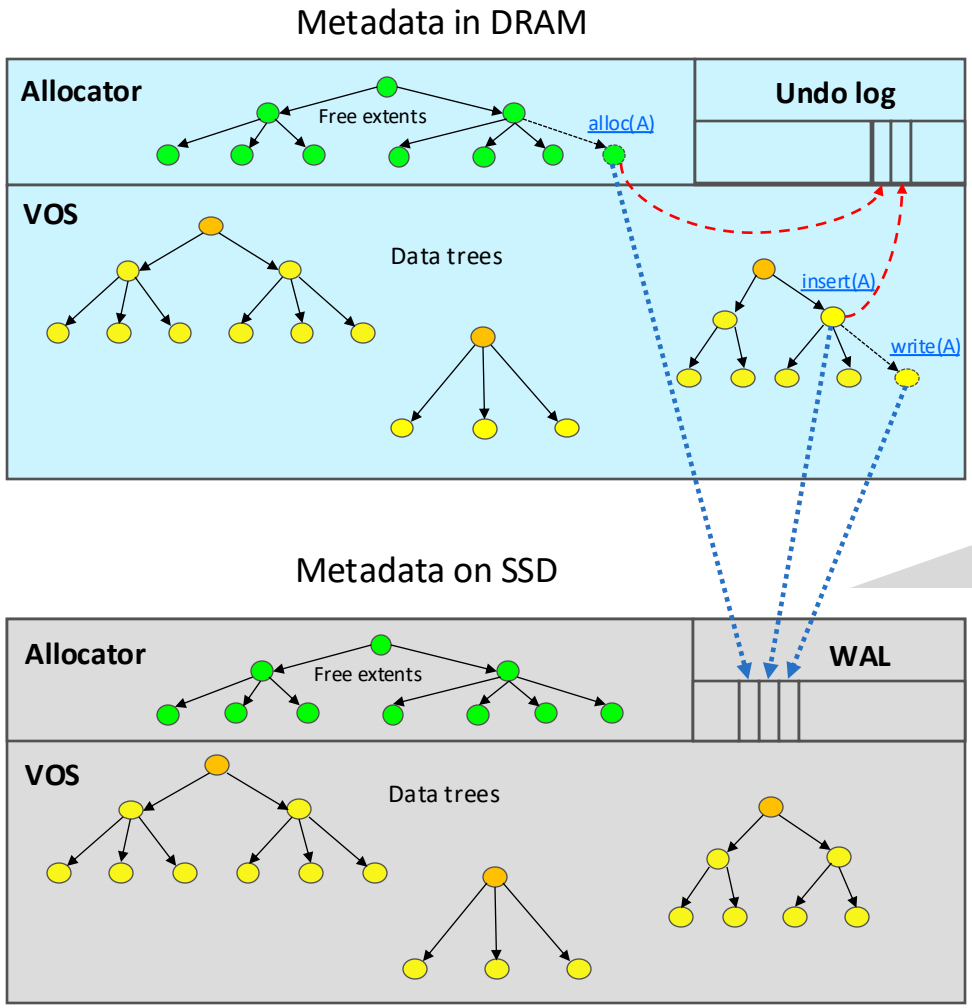
DAOS Architecture Evolution



With Persistent Memory

Without Persistent Memory

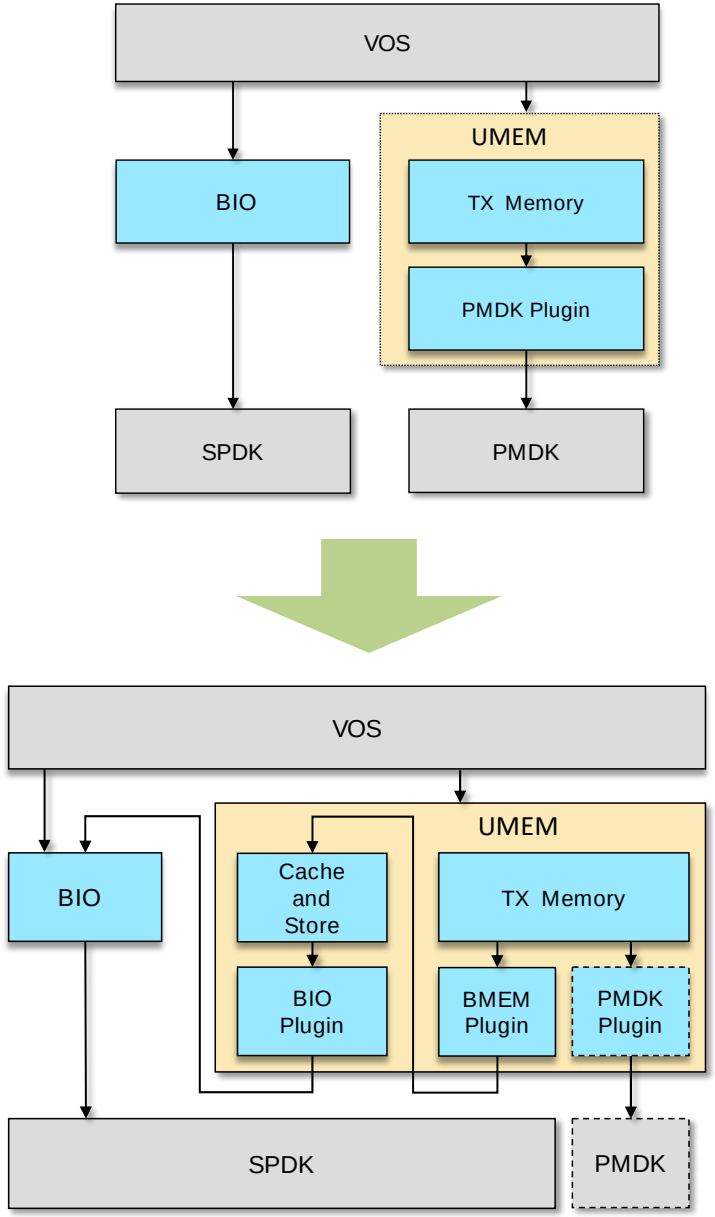
DAOS: Metadata on SSD



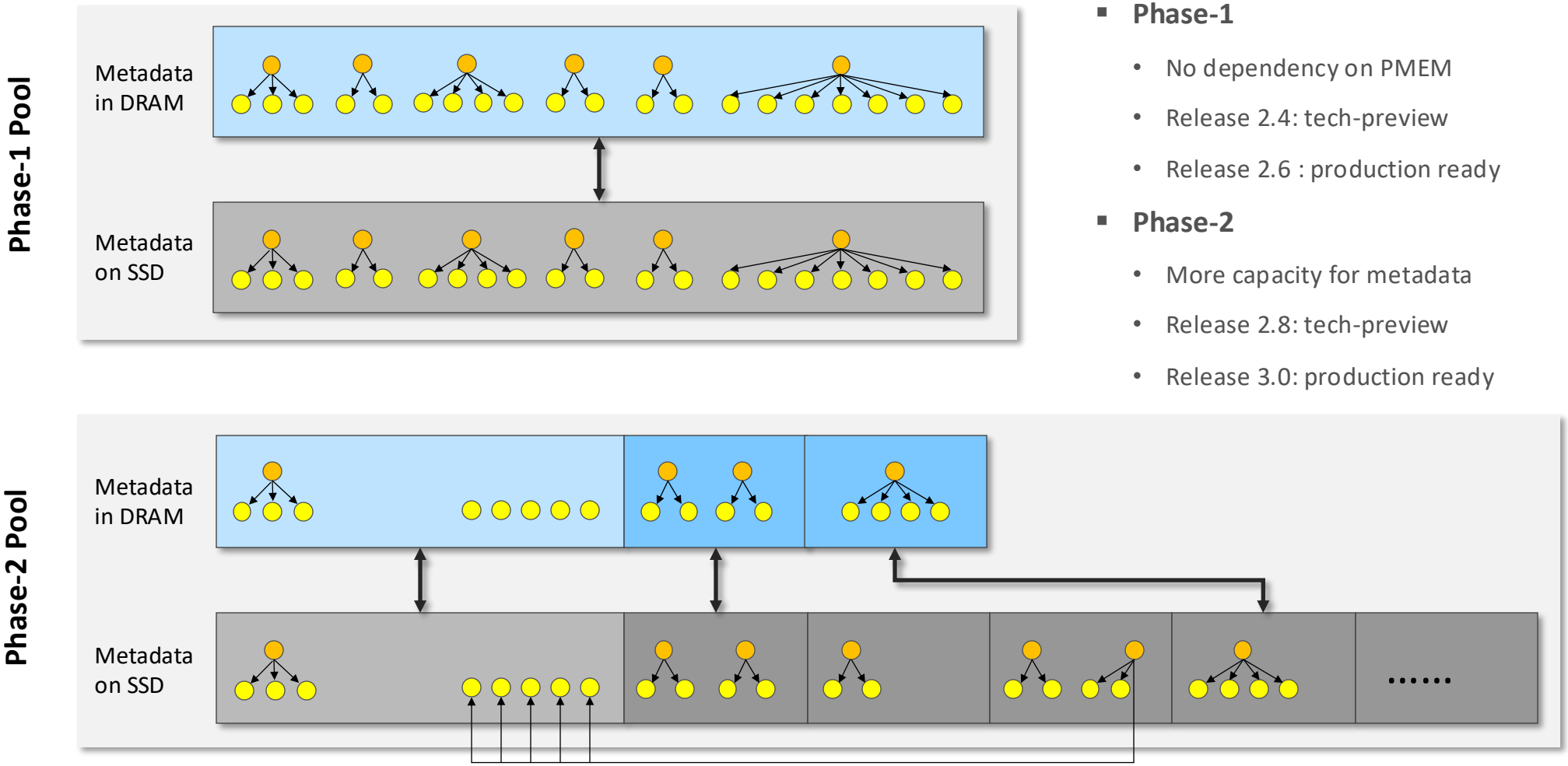
- Log all memory changes in a contiguous buffer
- Submit the buffer to SSD as one write (atomicity)

Metadata on SSD Phase-1

- In Release 2.6
- No semantic change to API
- Low latency I/O
  - Two SSD writes per update
    - One for WAL, another for data, they are parallelized for small I/O
    - With PMEM: one SSD write, one PMEM transaction, they are serialized
  - Fetch metadata from DRAM, no change to data path
- Minimum impact on performance
- Differences from PMEM mode
  - Extra step during recovery
  - Number of objects per engine
    - limited by DRAM capacity



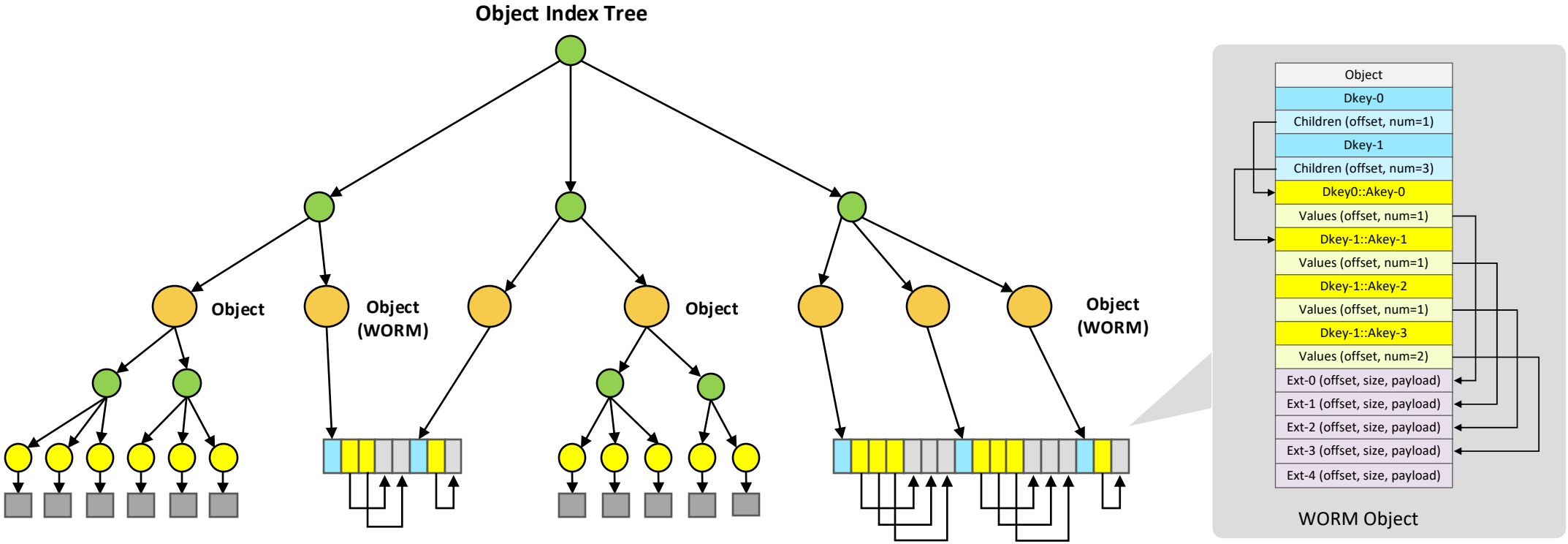
MD-on-SSD Phase-2: Evictable Metadata Bucket



- **Phase-1**
  - No dependency on PMEM
  - Release 2.4: tech-preview
  - Release 2.6 : production ready
- **Phase-2**
  - More capacity for metadata
  - Release 2.8: tech-preview
  - Release 3.0: production ready

WORM Object

- Regular object (key-array)
  - **Hierarchical** format: B+tree and Rectangle tree
- Write-Once-Read Many (WORM) object
  - **Flattened** trees stored in contiguous space

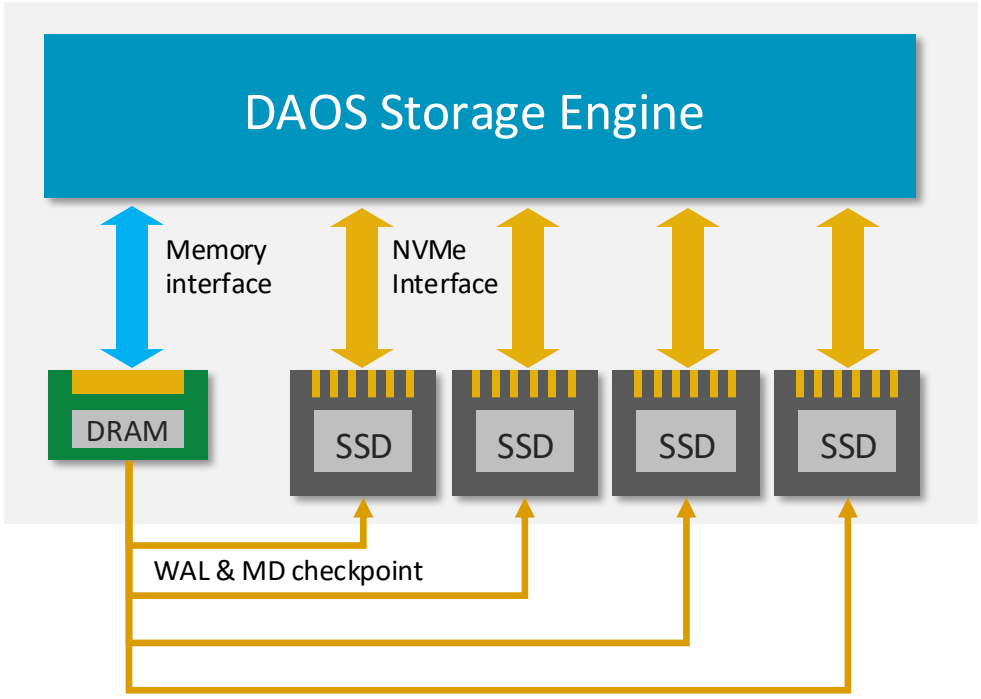


WORM Object: Flattening

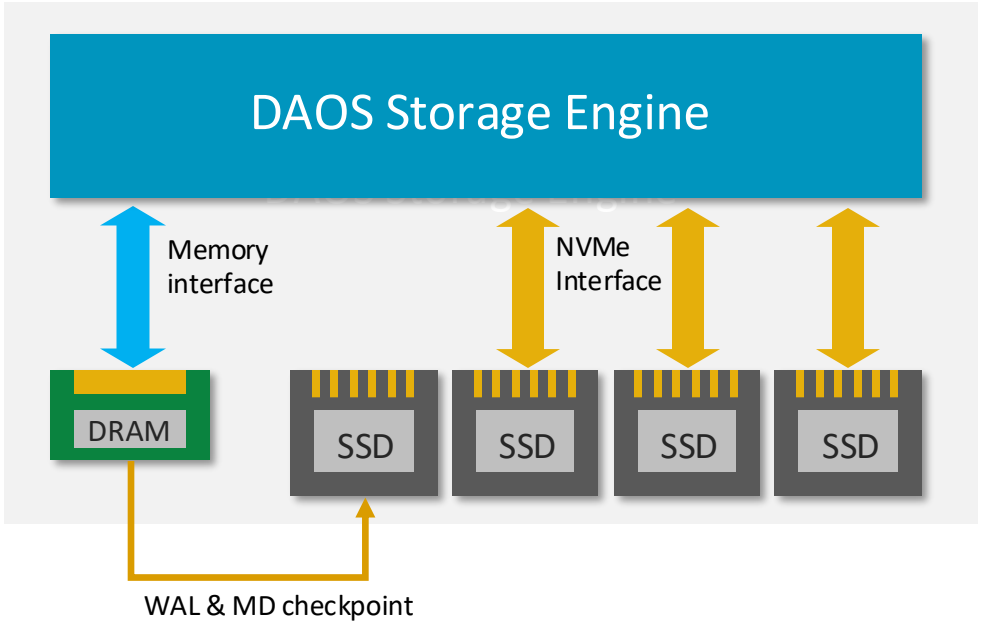
- **Stored in hierarchical format initially**
  - VOS is essentially an indexed write log: MVCC and distributed transaction
  - Application can submit writes in multiple RPCs (append, or random write)
- **WORM object**
  - Mostly for small object: linear search is inefficient for large object
  - Reduce I/O latency and space overhead of small objects
  - Reduce space fragmentation
- **Flattening service**
  - Turn hierarchical object into flattened format
  - Triggered by API or scan results
- **API extension: writing flattened object from client?**

MD-on-SSD Configuration

Mixed mode



Dedicated SSD for MD/WAL



mdtest easy (4:16 and 16: 64)


intel