DAOS Middleware Update

DAOS User Group – SC 2020

Mohamad Chaarawi

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 .

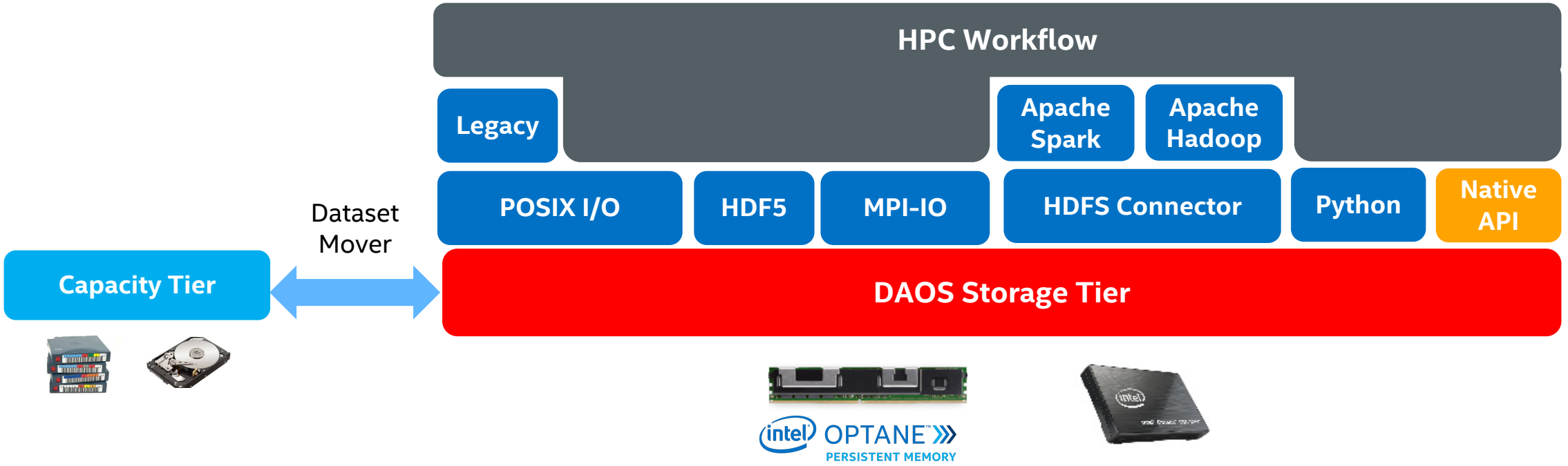
Intel Advanced Vector Extensions (Intel AVX) provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 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.

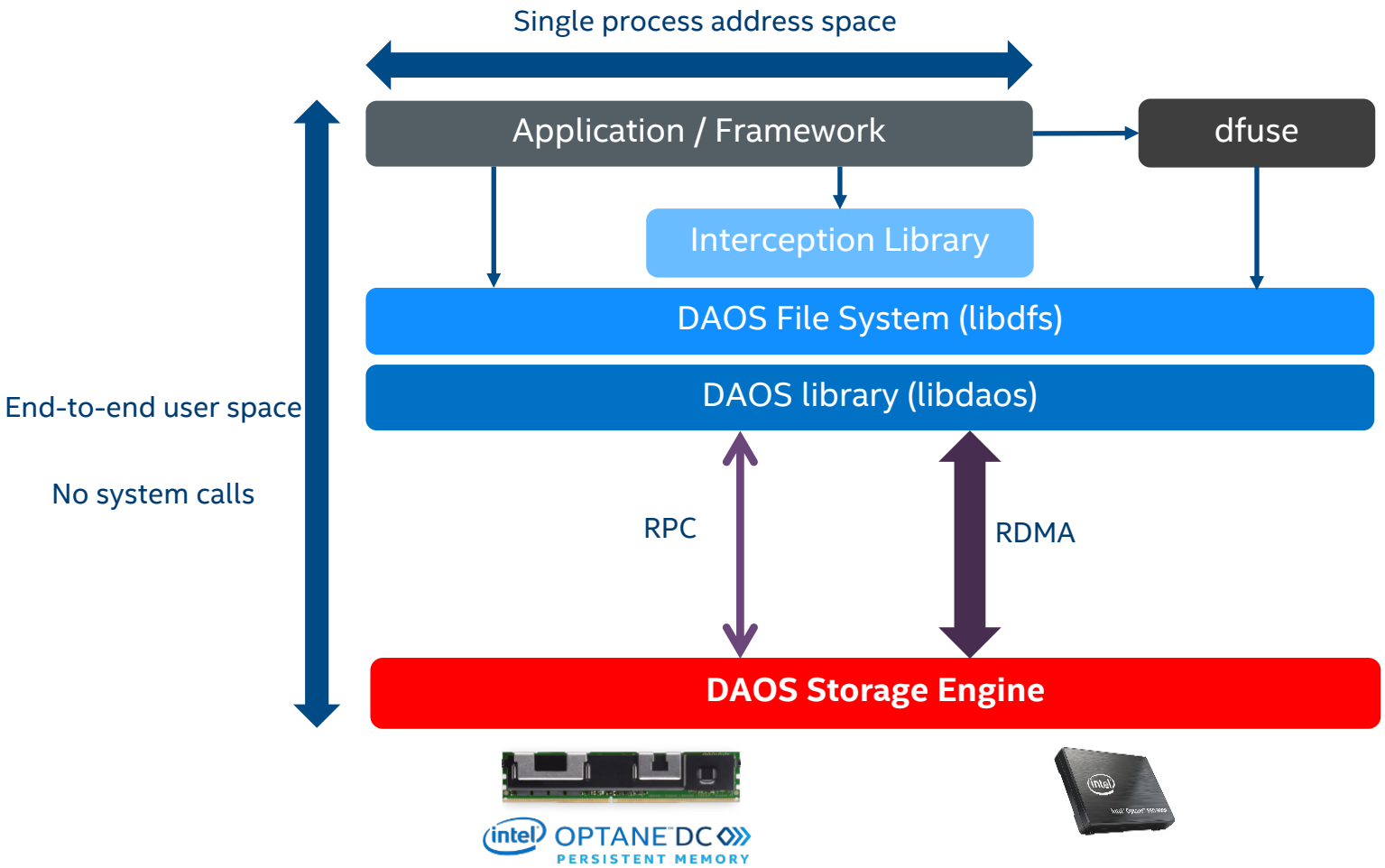
Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

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.

Application Interface



POSIX Software Stack



- User space DFS library with an API like POSIX.
  - Requires application changes (new API)
- DFUSE plugin to support POSIX API
  - No application changes
  - Limited performance
  - Can enable caching over single node for better metadata performance, mmap support, read ahead optimization.
- DFUSE + IL
  - No application changes, runtime LD_PRELOAD
  - Good raw data I/O performance

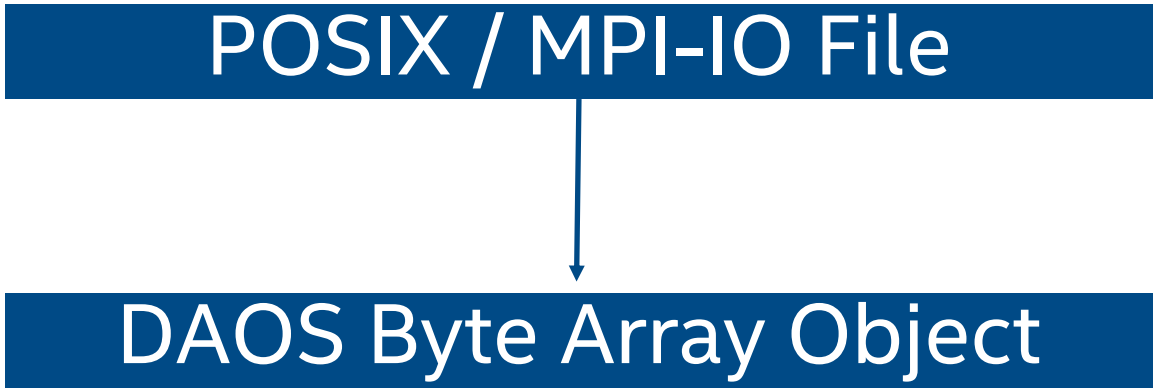
DFS Flavors for Consistency

- Provide two modes that offer different levels of consistency for users:
  1. Relaxed mode for well-behaved applications prioritizing performance over concurrency control.
  2. A more Balanced mode for applications that require stricter consistency at the cost of performance (default mode).
- Benchmark both modes using mdtest and use the more consistent mode when performance becomes close.

MPI-IO Driver for DAOS

The DAOS MPI-IO driver is implemented within the I/O library in MPICH (ROMIO).

- Added as an ADIO driver
- Portable to Open-MPI, Intel MPI, etc.
- https://github.com/pmodels/mpich



Special DAOS Object:
- 1 Level Key
- 1 Byte records
- Configurable Chunk Size

MPI Files use the same DFS mapping to the DAOS Object Model

- MPI Files can be accessed through the DFS API
- MPI Files can be accessed through regular POSIX with a dfuse mount over the container.

Application works seamlessly by just specifying the use of the driver by appending "**`daos:`**" to the path.

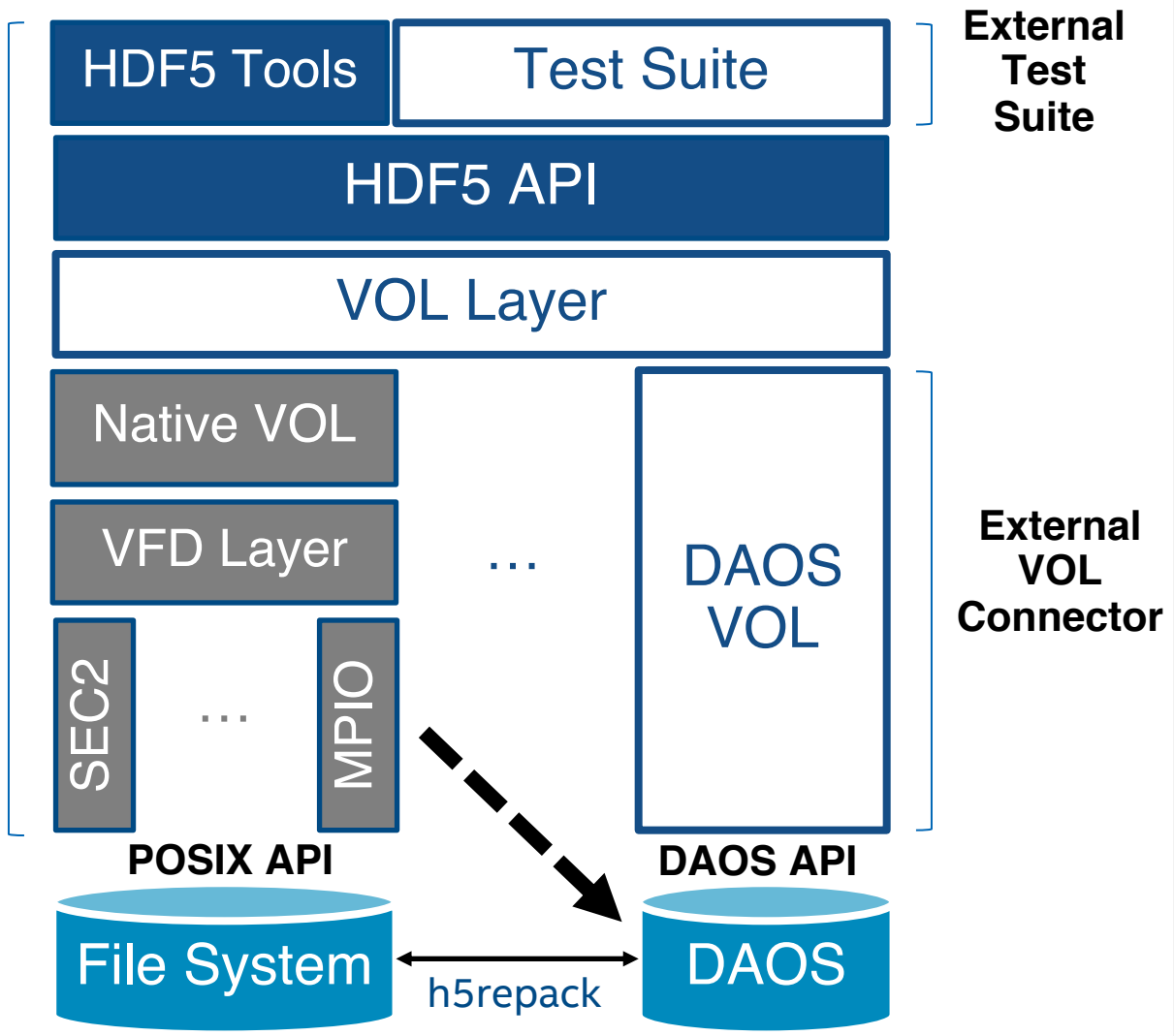
MPI-IO ROMIO driver (https://github.com/pmodels/mpich/tree/master/src/mpi/romio/adio/ad_daos)

HDF5 VOL Architecture

New Component

Enhanced Component

Native Component

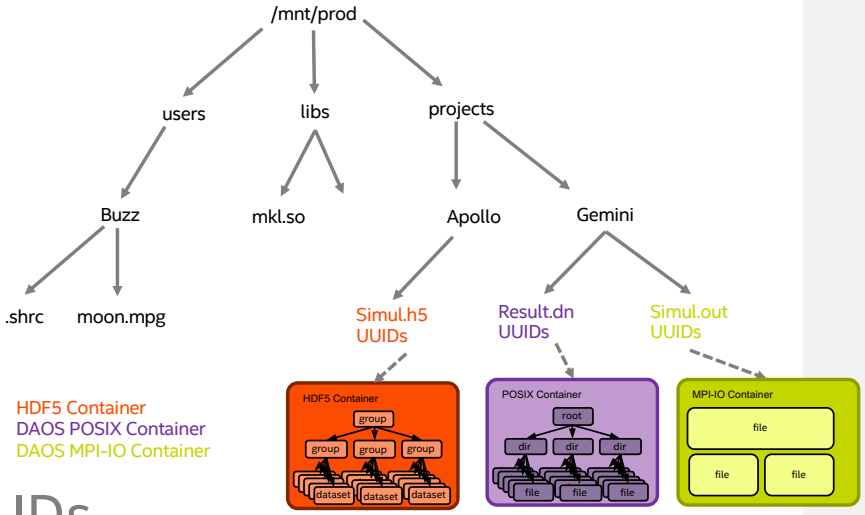


- Three main components:
  - HDF5 Library
  - DAOS VOL Connector
  - (External) HDF5 Test Suite

HDF5 DAOS VOL Connector – Current Status

- No longer requires separate version of HDF5:
  - Compatible with main develop branch of HDF5
  - Compatible with 1.12.x release series of HDF5 with VOL support
- Currently supported features:
  - New H5M MAP API to expose K/V interface to HDF5 users
  - Variable length datatypes are now also supported
  - Tools support (h5dump, h5ls, h5repack, etc.)
- Coming soon:
  - Independent metadata writes (= independent object creation)
  - Asynchronous metadata and raw data operations
  - Real application evaluation + HDF5 middleware (NetCDF4, PIO)
- Available from: https://github.com/HDFGroup/vol-daos
  - See user's guide for more detailed list of supported features

Unified Namespace



DAOS Object Store:

- Addresses pools, containers with uuids; objects with 128-bit IDs.
- Applications/Users are used to access files / directories in a traditional namespace

Unified Namespace allows users to create links between a file/dir in a system namespace to DAOS pools & containers:

- daos container create --path=/mnt/project1/userA/NS1 --pool=uuid --type=POSIX/HDF5/etc.
- Path created above becomes a special file or directory (depending on container type) with an extended attribute with the pool and container information.
- Accessing that path from DAOS aware middleware will make the link on the fly with the DAOS UNS library.

MariaDB Engine

- MySQL server enables backend engines to be written as plugins. Some example engines:
  - InnoDB: default engine
    - Can write a backend to InnoDB using DFS API – simplest approach for support
  - MyISAM
- DAOS engine:
  - new engine for native MySQL mapping to DAOS
  - leverage in-storage computing:
    - Engine condition pushdown optimization to avoid iterating over all DB rows to construct the result of the query
    - New DAOS API for filtering to be added
  - Still in development


intel