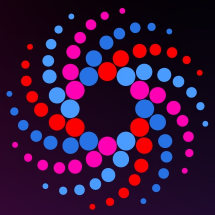


SC22
Dallas, TX | hpc accelerates.

Tutorial on IO Middleware with DAOS

Mohamad Chaarawi, AXG, Intel

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.

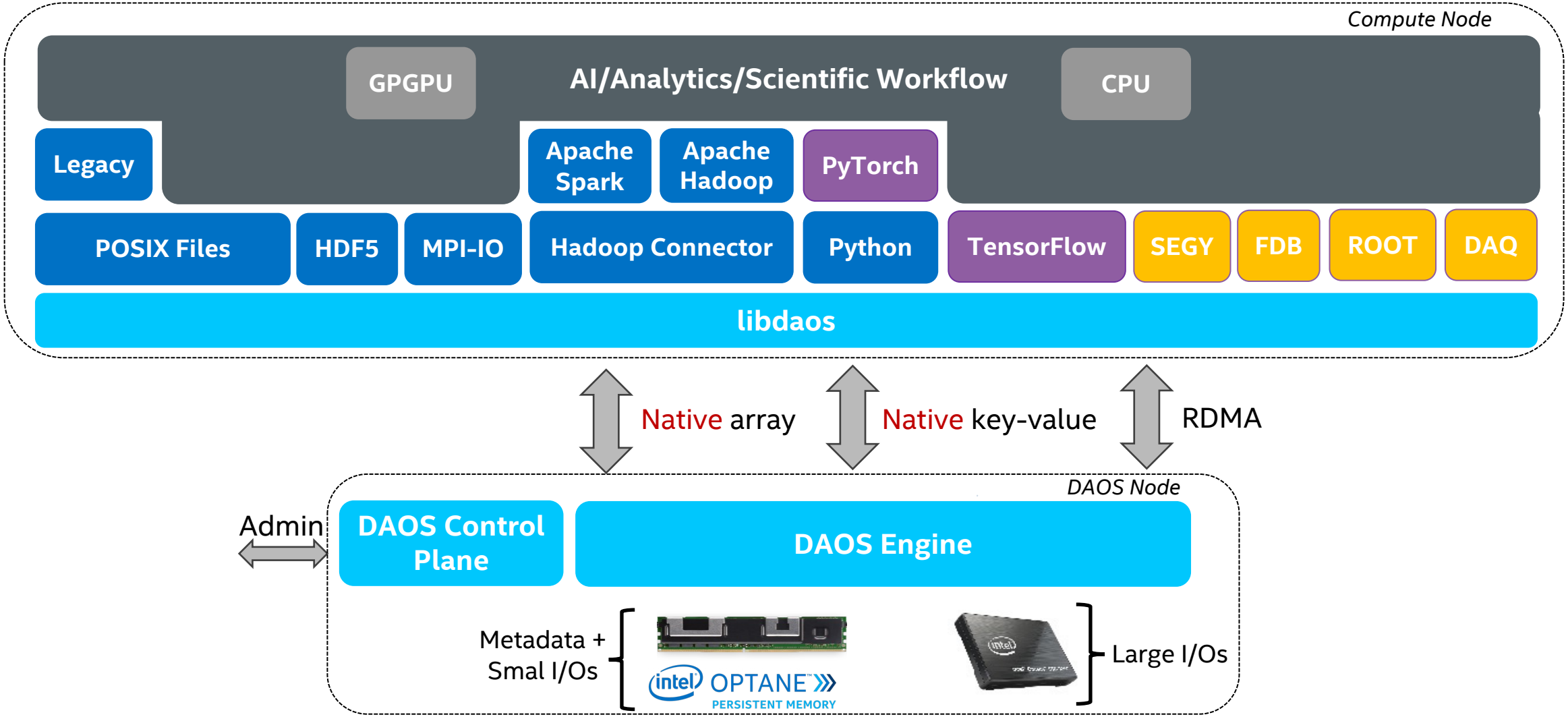
Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

DAOS Ecosystem

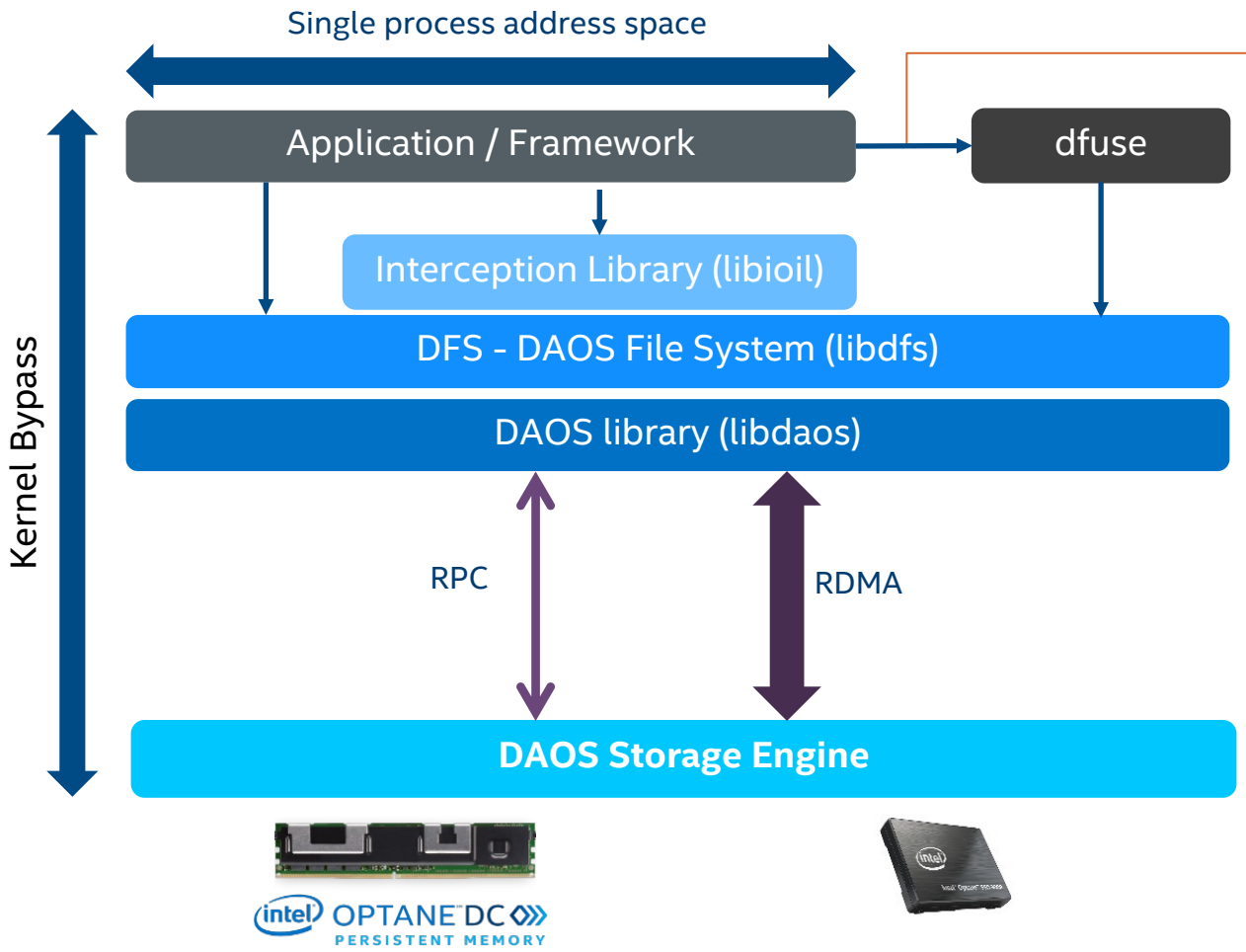
Generic I/O middleware supported today

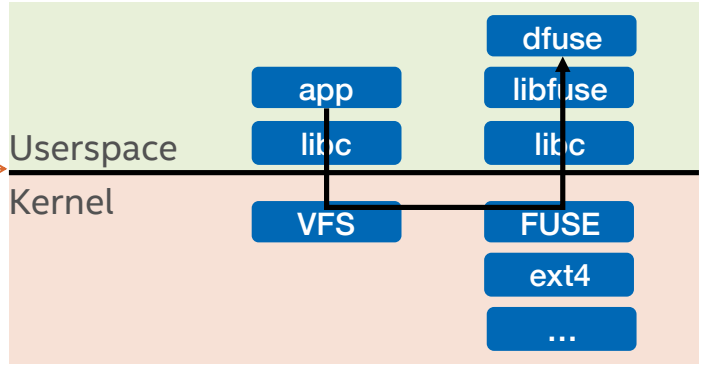
Domain-specific data models under development in co-design with partners

Enablement in progress



POSIX I/O Support





- User space DFS library with an API like POSIX.
  - Requires application changes (new API)
  - Kernel Bypass, no client cache
- DFUSE plugin to support POSIX API
  - No application changes
  - Fuse Kernel Supports data (wb and ra) & metadata caching (stat, open, etc.)
- DFUSE + IL
  - No application changes, runtime LD_PRELOAD
  - Kernel Bypass for raw data IO only.

How to use DFS?

- You should have access to a pool (identified by a string label).
- Create a POSIX container with the daos tool:
  - `daos cont create mypool --label=mycont --type=POSIX`
  - Alternatively, you can programmatically create a container to use directly in your application (if you are using DFS and changing your app).
- Open the DFS mount:
  - `dfs_connect (mypool, mycont, O_RDWR, .. &dfs);`
  - `dfs_disconnect (dfs);`

DFS API

POSIX	DFS
mkdir(), rmdir()	dfs_mkdir(), dfs_rmdir()
open(), close(), access()	dfs_open(), dfs_release(),dfs_lookup()
pwritev(), preadv()	dfs_read/write()
{set,get,list,remove}xattr()	dfs_{set,get,list,remove}xattr
stat(), fstat()	dfs_stat(),ostat()
readdir() …	dfs_readdir() …

- Mostly 1-1 mapping from POSIX API to DFS API.
- Instead of File & Directory descriptors, use DFS objects.
- All calls need the DFS mount which is usually done once initialization.

DFS Example + Demo

```
…
fd = open(file_name, O_CREAT|O_RDWR, 0600);
/** set up iov */
…
pwritev(fd, iov, 1, offset);
preadv(fd, iov, 1, offset);
close(fd);
```

```
…
dfs_open(dfs, NULL, file_name, 0600, O_CREAT|O_RDWR, 0, 0, NULL, &file);
/** setup sgl (DAOS iov) */
…
dfs_write(dfs, file, &sgl, offset, NULL);
dfs_read(dfs, file, &sgl, offset, &bytes_read, NULL);
dfs_release(file);
```

Another example to consider:

- IOR & mdtest DFS driver (https://github.com/hpc/ior)

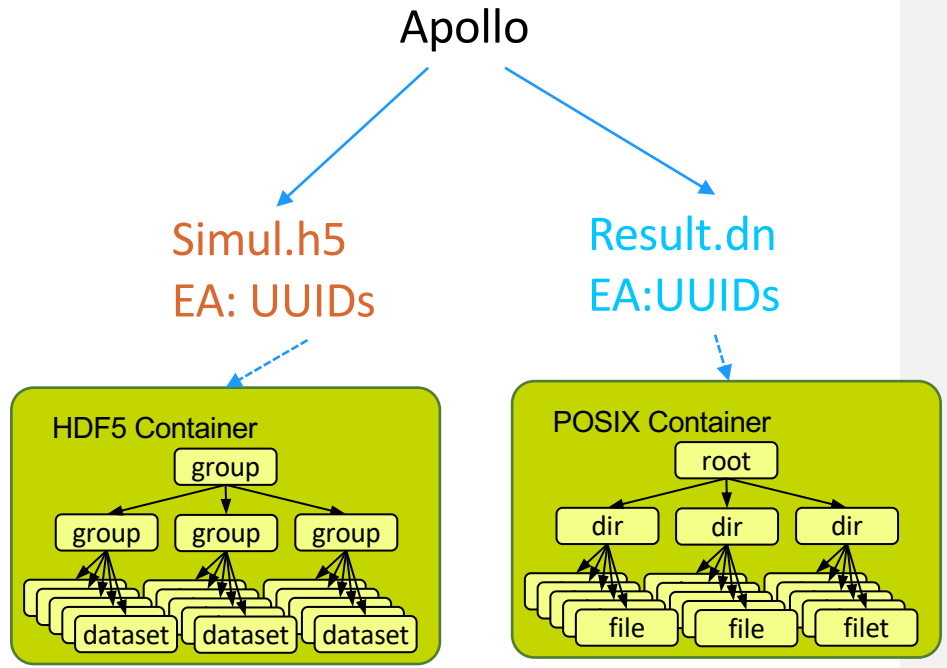
DFUSE

- To mount an existing POSIX container with dfuse, run the following command:
  - `dfuse --pool mypool --container mycont -m /mnt/dfuse`
  - No one can access your container / mountpoint unless access is provided on the pool and container (through ACLs).
  - Multi-user concurrent support is planned for DAOS 2.4
- Now you have a parallel file system under /mnt/dfuse on all nodes where that is mounted
  - Access files / directories as any namespace in the container, and applications can run without any modifications (the easy path).
- Interception Library:
  - This library works in conjunction with dfuse and allow to interception of POSIX I/O calls and issue the I/O operations directly from the application context through libdaos without any application changes.
  - This provides kernel-bypass for I/O. To use this set the LD_PRELOAD to point to the shared library in the DOAS install dir
    - LD_PRELOAD=/path/to/daos/install/lib/libioil.so

To Intercept or Not Intercept

- The dfuse interception library bypasses the fuse kernel and sends IO to DFS directly
  - No caching (no writeback / readahead)
  - Works well for bulk IO.
  - Does not work well for very tiny IO (order of bytes) where dfuse without interception takes advantage of the kernel write back and read ahead caching to reduce IO over the wire.

Unified Namespace



DAOS pool & container identified by labels/uuids

Concept: store those uuids in POSIX XATTR

- Take path instead of pool/container labels
- Libraries/tools can extract them from POSIX XATTR
- libduns

Path created is a special file or directory pointing at the container

- Directories for POSIX containers, files otherwise
- Special file/directory is effectively empty in the hosting filesystem (just one XATTR)

UNS Demo

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
  - Available in Intel MPI
  - https://github.com/pmodels/mpich
- 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.
- How to use this driver?
  - MPICH: append “daos:” to the file name/path or set env variable:
    - `ROMIO_FSTYPE_FORCE="daos:"`
  - Intel MPI: set env variables:
    - I_MPI_EXTRA_FILESYSTEM=1
    - I_MPI_EXTRA_FILESYSTEM_FORCE=daos

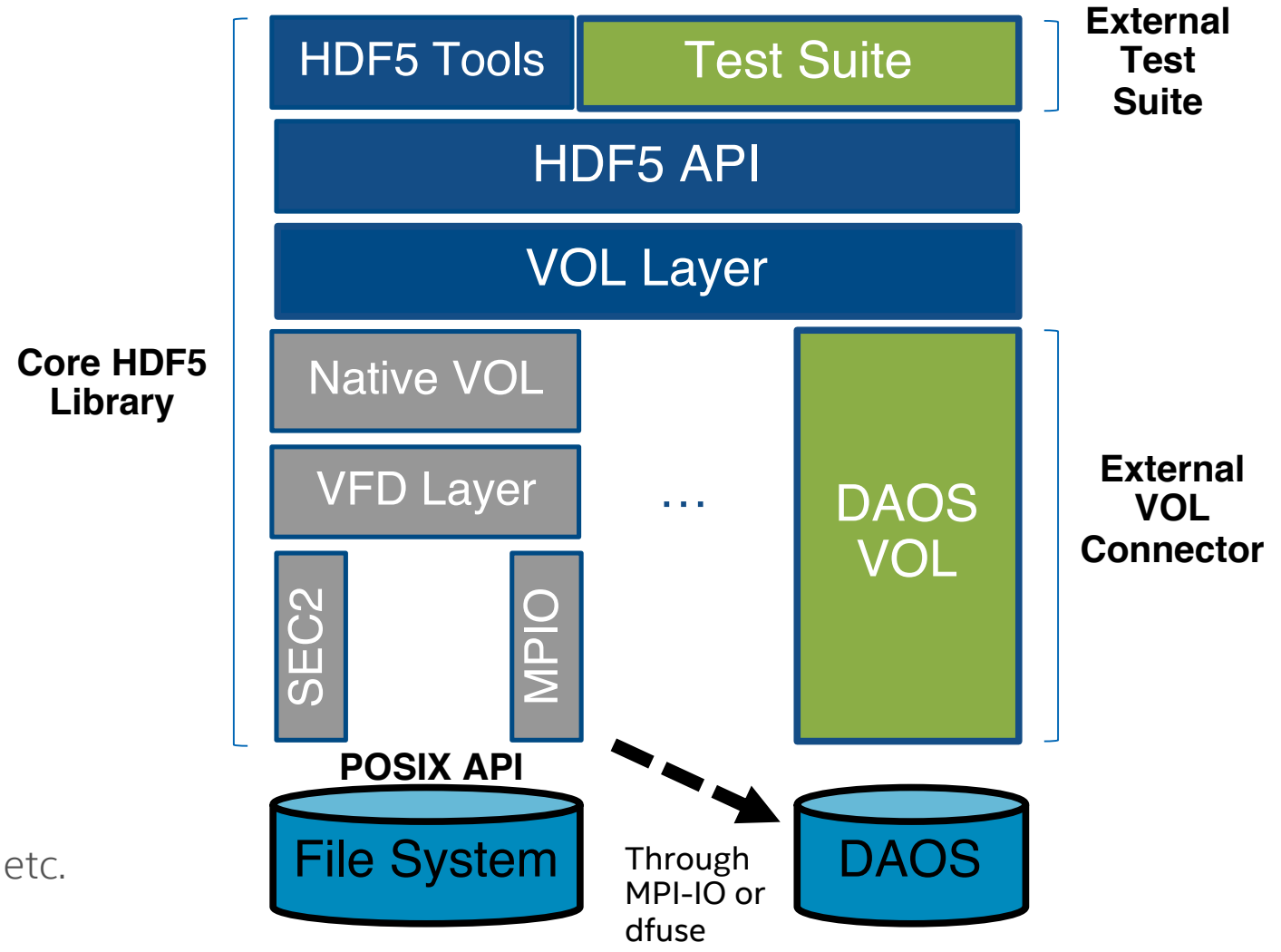
HDF5 VOL Architecture

New Component

Enhanced Component

Native Component

- Three main components:
  - HDF5 Library
  - DAOS VOL Connector
  - (External) HDF5 Test Suite
- Tools support:
  - h5dump, h5ls, h5diff, h5repack, h5copy, etc.

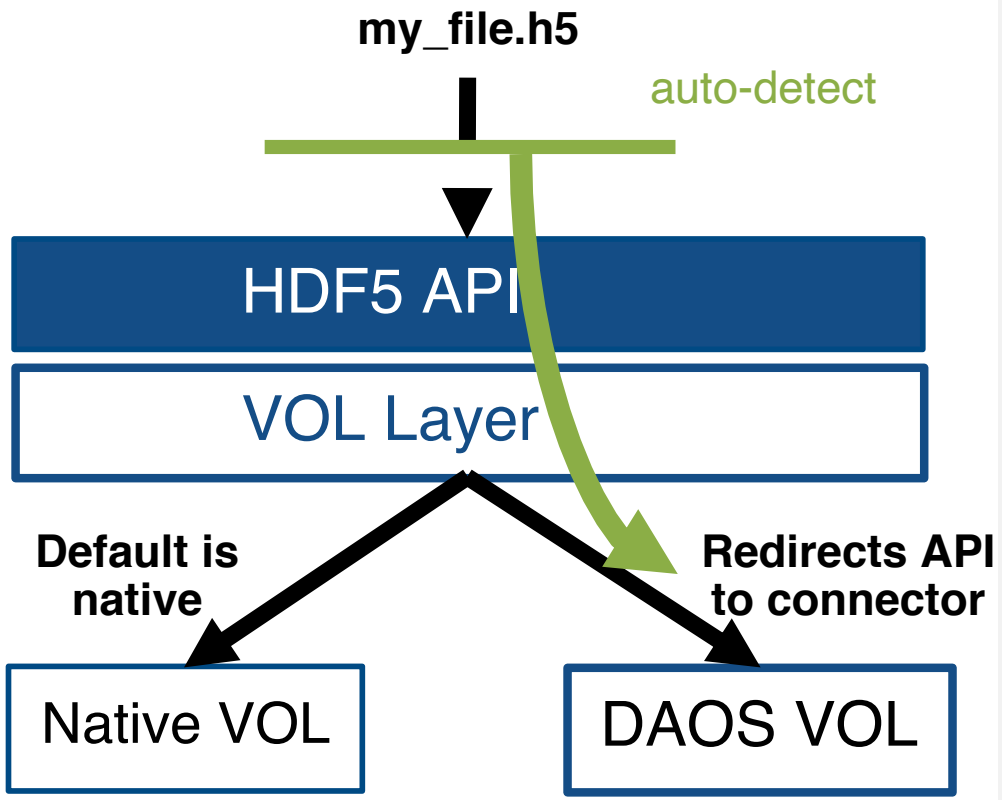


VOL Connector Selection

- Creation and use of HDF5 files in DAOS
  - Minimal or no code changes for application developer
  - Two ways to tell which connector to use
    - HDF5 file access property list:
      - `H5Pset_fapl_daos()`
    - Environment variable:

```
HDF5_VOL_CONNECTOR=daos
HDF5_PLUGIN_PATH=/path/to/connector/folder
DAOS_POOL = <pool label>
```

  - Unified Namespace
    - Does not require setting the pool env variable. The pool and container uuids are embedded in extended attribute in a stub file in the namespace.



HDF5 Example + Demo

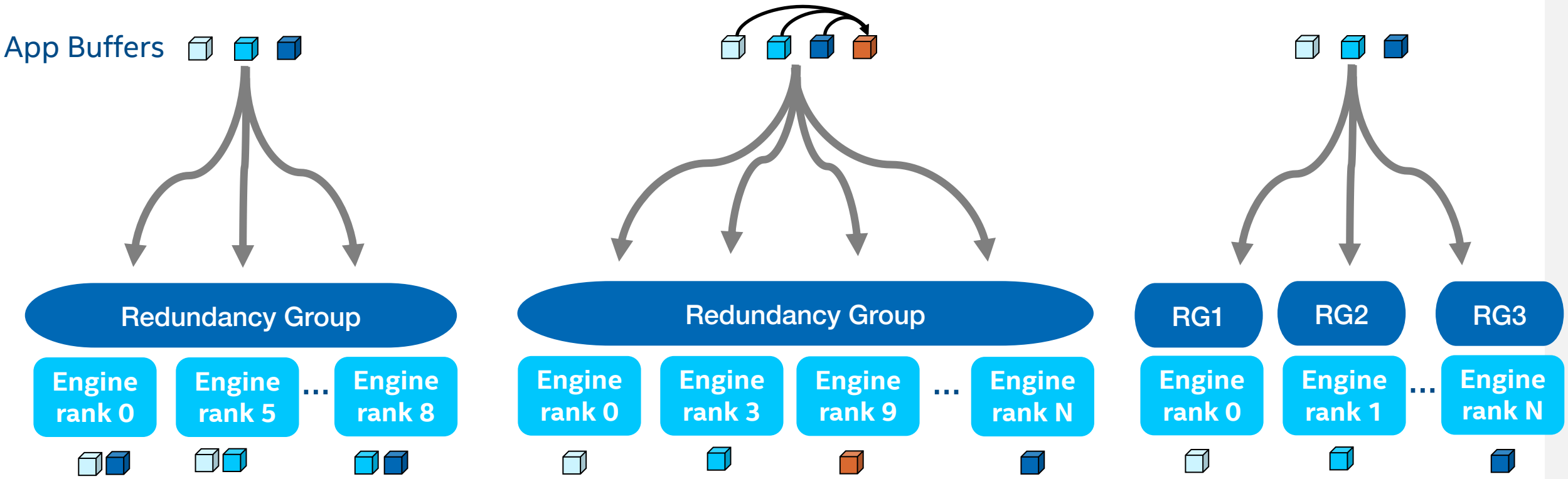
Current Repositories

- All repositories are in GitHub
- HDF5 base repository:
  - https://github.com/HDFGroup/hdf5
  - Use 1.13.1 release
- DAOS VOL Connector repository:
  - https://github.com/HDFGroup/vol-daos
  - Use latest release
- External HDF5 test suite repository:
  - https://github.com/HDFGroup/vol-tests
- Build / Run instructions:
  - https://docs.daos.io/v2.2/user/hdf5/?h=hdf5#hdf5-daos-vol-connector

Field Guide

- POSIX API:
- IO intensive: use dfuse + IL
- Using IO streaming (fread, fwrite, etc.)
  - Try dfuse with --enable-caching
- Metadata intensive (a lot of files, directories, stat, etc.):
  - dfuse will work, but limited performance
  - Look into changing IO code to use DFS
- MPI-IO:
  - Use DAOS MPIIO DAOS driver (if using recent MPICH, Intel MPI)
  - Otherwise use dfuse + IL
- HDF5:
- Is the daos-vol installed / available?
  - Use it ☺
- Is your app using parallel HDF5?
  - Use MPIIO backend if MPI library has it.
- Use dfuse + IL

DAOS Data Model: Distribution & Fault Tolerance



Replication

Erasure-code

Sharding

DAOS Object Class

- Each DAOS object is identified by an object ID which encode the object class.
- The object class controls the redundancy and sharding of the object and allows the client to do algorithmic placement when accessing the object.
  - Redundancy:
    - None: S
    - N-way Replication: RP_n
    - Erasure Code: EC_dPp (d = data, p = parity)
  - Sharding:
    - Single target/group:
      - S1, RP_nG1, EC_dPpG1
    - Widely / max sharding:
      - SX, RP_nGX, EC_dPpGX

Container Level Redundancy Setting

- On Container create, one can set the redundancy factor property (`--properties=rf:n`):
  - 0 (default): no redundancy
    - One can create objects with higher redundancy after
  - 1: tolerate 1 failure domain
    - At least RP2 or ECxP1
  - Up to 4
- DAOS automatically selects an object class for files and directories using the redundancy factor.
  - Widely striped, EC oclass for files (SX, EC_xP1GX, EC_xP2GX, etc.)
  - Single stripe, RP oclass for directories (S1, RP_2GX, RP_3GX, etc.)

Setting the Object Class

- Users can select the object class if the default is not appropriate.
  - Default works well if files are large, directories are small (not many entries)
- DFS API allows users to pass an object class when creating an object
  - Passing 0 would use the default
- DAOS tool allows also to create a **new** file with a specific oclass.
  - `daos fs set-attr --path=/mnt/dfuse/testfile --oclass=RP_2G1`
- DAOS tool allows changing the object class on a directory to use that oclass for all entries created in that directory (does not change the oclass of the directory itself)
  - `mkdir /mnt/dfuse/dir1`
  - `daos fs set-attr --path=/mnt/dfuse/dir1 --oclass=RP_2G1`

Striping Considerations

- Widely striped objects
  - Best IO BW (read/write) for single shared file, Highest IOPS for entry creation in single shared directory
  - Slow file stat (file size), ls (directory listing)
    - Must query all targets
    - Caching with dfuse helps in this case – limited to single node or read-only
- Small stripe objects
  - Limited size and bandwidth to number of targets
  - Good for FPP or DPP
  - Fast stat, ls (not querying all targets on storage system)

Redundancy Considerations

- Erasure Code performance characterization for IO:
  - Best for large IO access patterns
  - Full stripe write: 33% performance hit
  - Partial stripe write: 66% performance hit
  - Read should be the same.
- Replication:
  - Best for metadata objects (directories)
  - Small files (<= 16k)
  - Write IOPS: n x slower
  - Read IOPS: equal or better – more shards to serve concurrent requests

Challenges / Future Work

- Selecting correct object class still not straightforward:
  - Auto object class selection works fine but is not ideal in some situations.
- Extend (POSIX) container create options to allow users to provide hints as to what their expected dataset looks like:
  - small/large directories
  - small/large files
  - Based on those hints, the auto object class selection can be tuned to select better object class for performance considerations in such scenarios
- Extend DAOS FS tool to accept hints too for setting object class on parent directories or new files.
- *DAOS 2.4 features*


intel