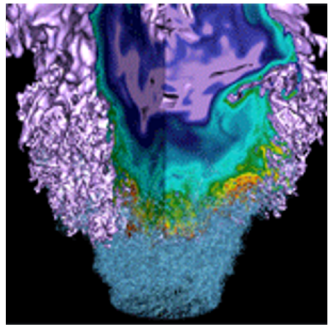
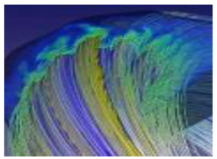
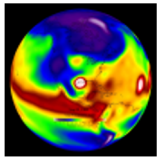
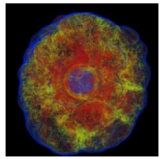
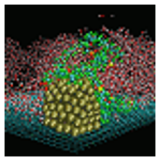
Virtual Object Layer (VOL) Connectors





Quincey Koziol

HDF5 User Group
October 15, 2020
koziol@lbl.gov

U.S. DEPARTMENT OF ENERGY | Office of Science

EXASCALE COMPUTING PROJECT

BERKELEY LAB

Goals

- Virtual Object Layer (VOL) Introduction
- High-level Overview of VOL Architecture
- How to create a new VOL connector

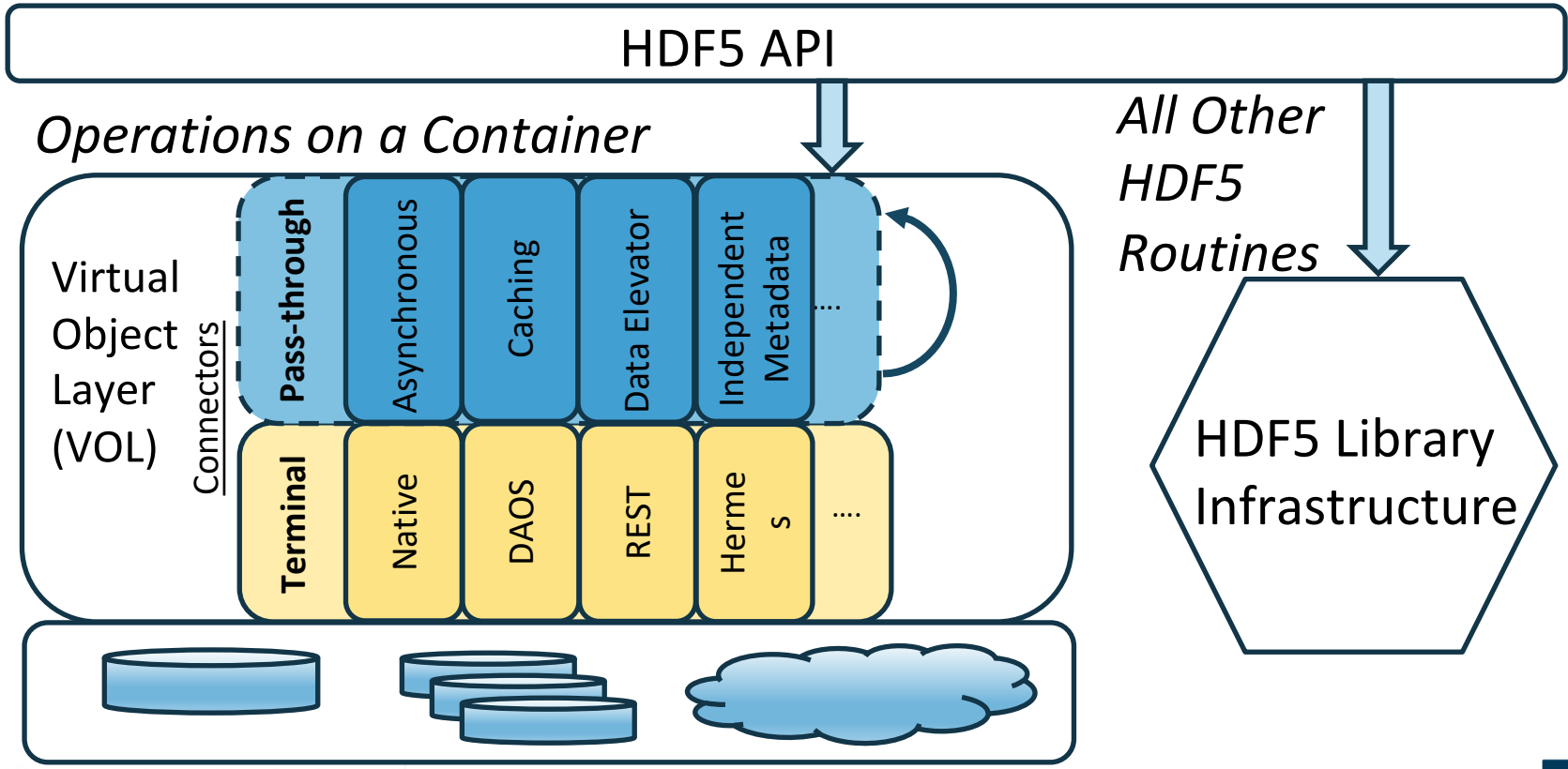
Virtual Object Layer (VOL)

- **VOL Framework is an abstraction layer within HDF5 Library**
  - Redirects I/O operations into VOL "connector", immediately after an API routine is invoked
  - Non-I/O operations handled with library "infrastructure"
- **VOL Connectors**
  - Implement storage for HDF5 objects, and "methods" on those objects
    - Dataset create, write / read selection, query metadata, close, …
  - Can be transparently invoked from a dynamically loaded library, without modifying application source code
    - Or even rebuilding the app binary!
  - Can be stacked, allowing many types of connectors
    - "Pass-through" and "Terminal" connector types

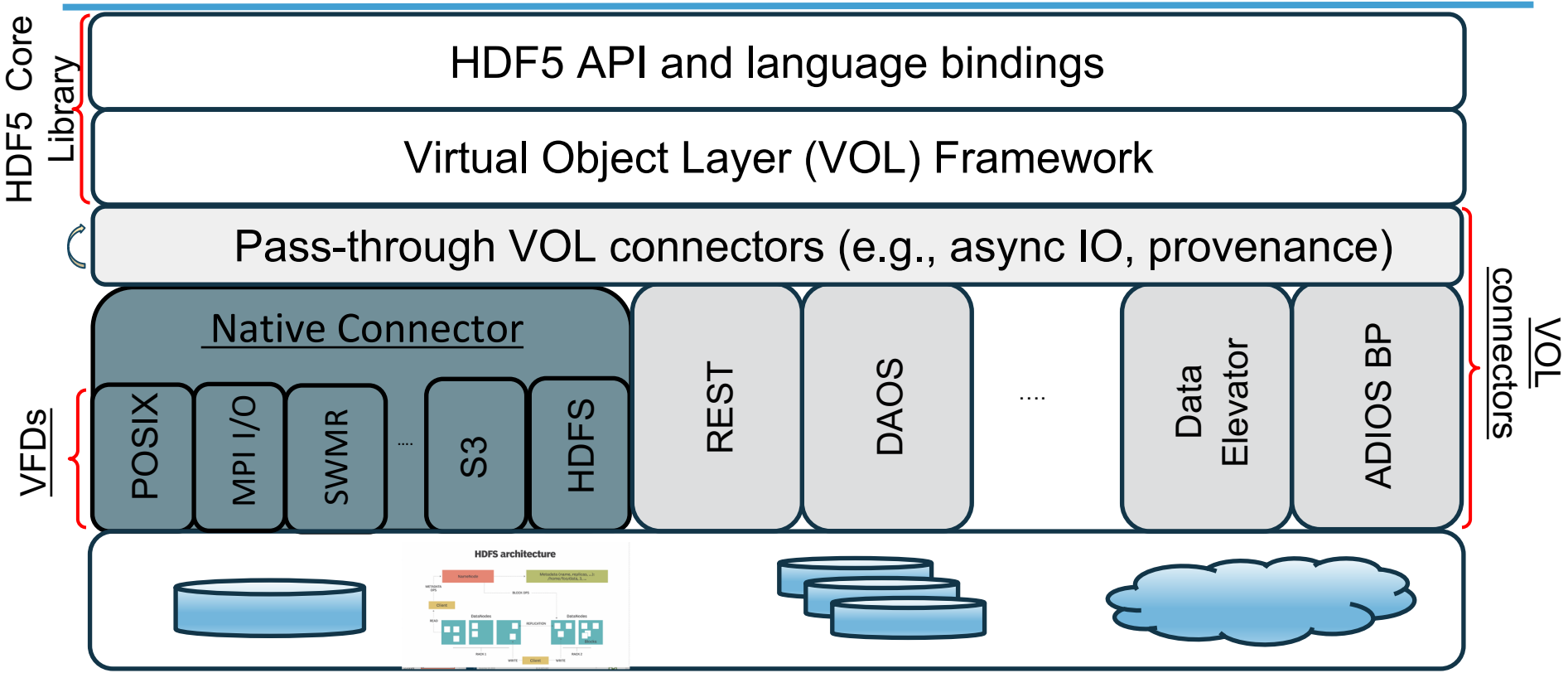
Virtual Object Layer (VOL) Connectors

- **Pass-through - can be stacked, must eventually have terminal connector**
  - Examples:
    - Provenance tracking
    - Asynchronous I/O
    - Caching
- **Terminal - non-stackable, final connector**
  - Examples:
    - Remote access (e.g. cloud, streaming, etc)
    - Non-HDF5 file access (e.g. ADIOS BP, netCDF “classic”, etc)
    - Object stores (e.g. DAOS, HSDS, etc)

High-Level Overview



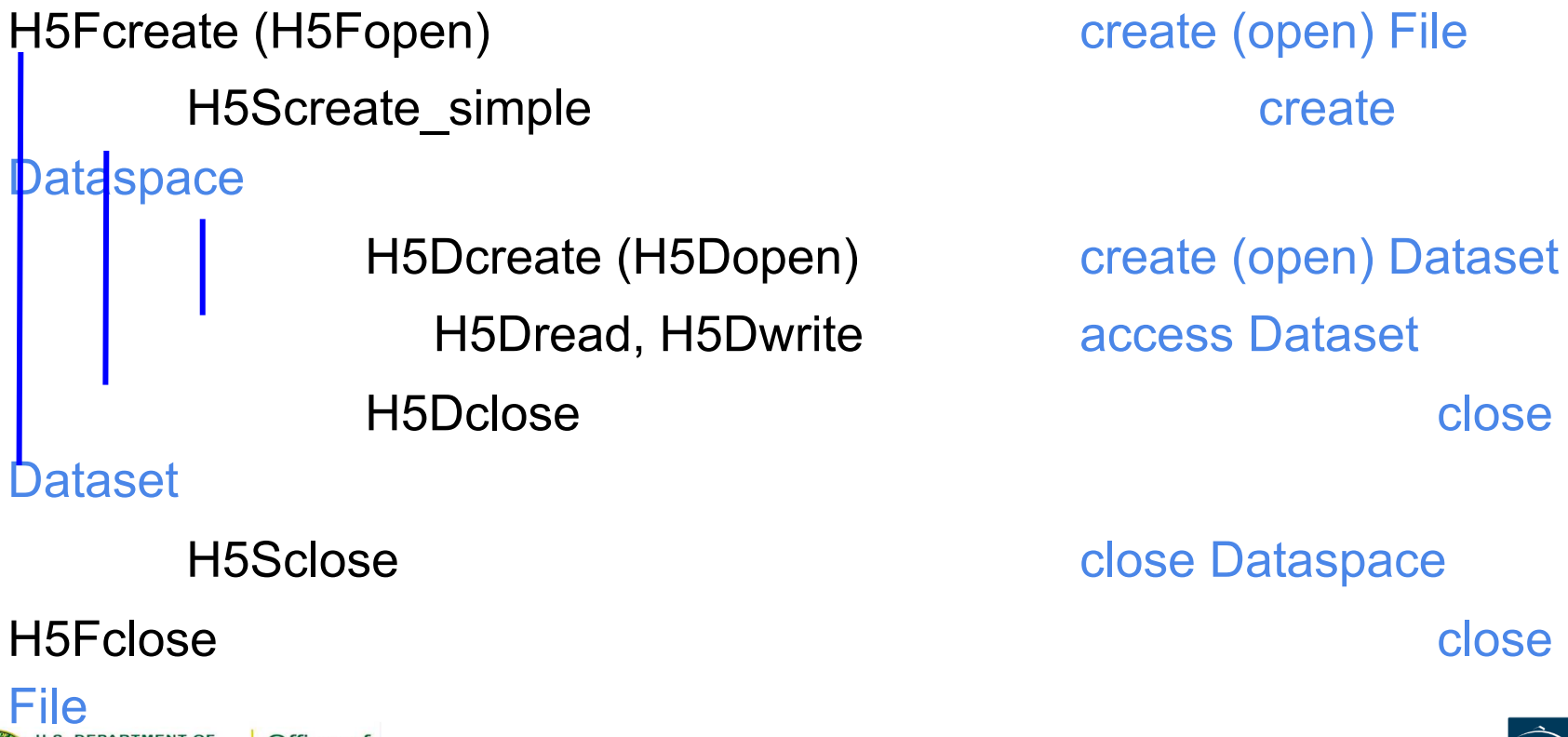
HDF5 Library Architecture



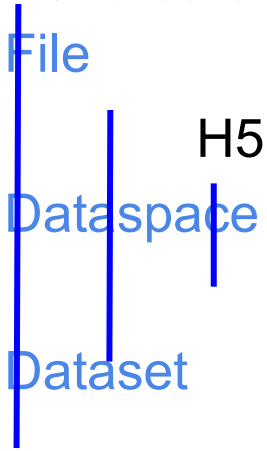
Virtual Object Layer (VOL) Implementation

- **VOL Framework is an abstraction layer within HDF5 Library**
  - Redirects I/O operations into VOL “connector”, immediately after an API routine is invoked
  - Non-I/O operations handled with library “infrastructure”

Standard HDF5 “Skeleton”



Standard HDF5 “Skeleton”



H5Fcreate (H5Fopen) — VOL: create (open)

File

H5Screate_simple — Infra: create

Dataspace

H5Dcreate (H5Dopen) — VOL: create (open)

Dataset

H5Dread, H5Dwrite — VOL: access Dataset

H5Dclose — VOL: close Dataset

H5Sclose — Infra: close Dataspace

H5Fclose — VOL

Virtual Object Layer (VOL)

- **VOL Framework is an abstraction layer within HDF5 Library**
  - Redirects I/O operations into VOL “connector”, immediately after an API routine is invoked

Typical HDF5 API Routine that invokes VOL

```
herr_t H5Dread(hid_t dset_id, hid_t mem_type_id, hid_t mem_space_id, hid_t file_space_id, hid_t dxpl_id, void *buf /*out*/)
{
    H5VL_object_t *vol_obj   = NULL;
    herr_t        ret_value = SUCCEED; /* Return value */

    FUNC_ENTER_API(FAIL)
    H5TRACE6("e", "iiiiix", dset_id, mem_type_id, mem_space_id, file_space_id, dxpl_id, buf);

    <<Check arguments>>

    /* Get dataset object */
    if (NULL == (vol_obj = (H5VL_object_t *)H5I_object_verify(dset_id, H5I_DATASET)))
        HGOTO_ERROR(H5E_ARGS, H5E_BADTYPE, FAIL, "dset_id is not a dataset ID")

    /* Read the data */
    if ((ret_value = H5VL_dataset_read(vol_obj, mem_type_id, mem_space_id, file_space_id, dxpl_id, buf, H5_REQUEST_NULL)) < 0)
        HGOTO_ERROR(H5E_DATASET, H5E_READERROR, FAIL, "can't read data")

done:
    FUNC_LEAVE_API(ret_value)
} /* end H5Dread() */
```

Typical HDF5 API Routine that invokes VOL

```
herr_t H5Dread(hid_t dset_id, hid_t mem_type_id, hid_t mem_space_id, hid_t file_space_id, hid_t dxpl_id, void *buf /*out*/)
{
    H5VL_object_t *vol_obj   = NULL;
    herr_t        ret_value = SUCCEED; /* Return value */

    FUNC_ENTER_API(FAIL)
    H5TRACE6("e", "iiiiix", dset_id, mem_type_id, mem_space_id, file_space_id, dxpl_id, buf);

    <<Check arguments>>

    /* Get dataset object */
    if (NULL == (vol_obj = (H5VL_object_t *)H5I_object_verify(dset_id, H5I_DATASET)))
        HGOTO_ERROR(H5E_ARGS, H5E_BADTYPE, FAIL, "dset_id is not a dataset ID")

    /* Read the data */
    if ((ret_value = H5VL_dataset_read(vol_obj, mem_type_id, mem_space_id, file_space_id, dxpl_id, buf, H5_REQUEST_NULL)) < 0)
        HGOTO_ERROR(H5E_DATASET, H5E_READERROR, FAIL, "can't read data")

done:
    FUNC_LEAVE_API(ret_value)
} /* end H5Dread() */
```

Typical HDF5 API Routine that invokes VOL

```
herr_t H5Dread(hid_t dset_id, hid_t mem_type_id, hid_t mem_space_id, hid_t file_space_id, hid_t dxpl_id, void *buf /*out*/)
{
    H5VL_object_t *vol_obj   = NULL;
    herr_t        ret_value = SUCCEED; /* Return value */

    FUNC_ENTER_API(FAIL)
    H5TRACE6("e", "iiiiix", dset_id, mem_type_id, mem_space_id, file_space_id, dxpl_id, buf);

    <<Check arguments>>

    /* Get dataset object */
    if (NULL == (vol_obj = (H5VL_object_t *)H5I_object_verify(dset_id, H5I_DATASET)))
        HGOTO_ERROR(H5E_ARGS, H5E_BADTYPE, FAIL, "dset_id is not a dataset ID")

    /* Read the data */
    if ((ret_value = H5VL_dataset_read(vol_obj, mem_type_id, mem_space_id, file_space_id, dxpl_id, buf, H5_REQUEST_NULL)) < 0)
        HGOTO_ERROR(H5E_DATASET, H5E_READERROR, FAIL, "can't read data")

done:
    FUNC_LEAVE_API(ret_value)
} /* end H5Dread() */
```

Typical HDF5 API Routine that invokes VOL

```
herr_t H5Dread(hid_t dset_id, hid_t mem_type_id, hid_t mem_space_id, hid_t file_space_id, hid_t dxpl_id, void *buf /*out*/)
{
    H5VL_object_t *vol_obj   = NULL;
    herr_t        ret_value = SUCCEED; /* Return value */

    FUNC_ENTER_API(FAIL)
    H5TRACE6("e", "iiiiix", dset_id, mem_type_id, mem_space_id, file_space_id, dxpl_id, buf);

    <<Check arguments>>

    /* Get dataset object */
    if (NULL == (vol_obj = (H5VL_object_t *)H5I_object_verify(dset_id, H5I_DATASET)))
        HGOTO_ERROR(H5E_ARGS, H5E_BADTYPE, FAIL, "dset_id is not a dataset ID")

    /* Read the data */
    if ((ret_value = << async⇒native dataset read >>n_type_id, mem_space_id, file_space_id, dxpl_id, buf, H5_REQUEST_NULL)) < 0)
        HGOTO_ERROR(H5E_DATASET, H5E_READERROR, FAIL, "can't read data")

done:
    FUNC_LEAVE_API(ret_value)
} /* end H5Dread() */
```

Typical HDF5 API Routine that invokes VOL

```
herr_t H5Dread(hid_t dset_id, hid_t mem_type_id, hid_t mem_space_id, hid_t file_space_id, hid_t dxpl_id, void *buf /*out*/)
{
    H5VL_object_t *vol_obj   = NULL;
    herr_t        ret_value = SUCCEED; /* Return value */

    FUNC_ENTER_API(FAIL)
    H5TRACE6("e", "iiiiix", dset_id, mem_type_id, mem_space_id, file_space_id, dxpl_id, buf);

    <<Check arguments>>

    /* Get dataset object */
    if (NULL == (vol_obj = (H5VL_object_t *)H5I_object_verify(dset_id, H5I_DATASET)))
        HGOTO_ERROR(H5E_ARGS, H5E_BADTYPE, FAIL, "dset_id is not a dataset ID")

    /* Read the data */
    if ((ret_value = << DAOS dataset read >>bj, mem_type_id, mem_space_id, file_space_id, dxpl_id, buf, H5_REQUEST_NULL)) < 0)
        HGOTO_ERROR(H5E_DATASET, H5E_READERROR, FAIL, "can't read data")

done:
    FUNC_LEAVE_API(ret_value)
} /* end H5Dread() */
```

Typical HDF5 API Routine that invokes VOL

```
herr_t H5Dread(hid_t dset_id, hid_t mem_type_id, hid_t mem_space_id, hid_t file_space_id, hid_t dxpl_id, void *buf /*out*/)
{
    H5VL_object_t *vol_obj   = NULL;
    herr_t        ret_value = SUCCEED; /* Return value */

    FUNC_ENTER_API(FAIL)
    H5TRACE6("e", "iiiiix", dset_id, mem_type_id, mem_space_id, file_space_id, dxpl_id, buf);

    <<Check arguments>>

    /* Get dataset object */
    if (NULL == (vol_obj = (H5VL_object_t *)H5I_object_verify(dset_id, H5I_DATASET)))
        HGOTO_ERROR(H5E_ARGS, H5E_BADTYPE, FAIL, "dset_id is not a dataset ID")

    /* Read the data */
    if ((ret_value = << ADIOS BP dataset read >>em_type_id, mem_space_id, file_space_id, dxpl_id, buf, H5_REQUEST_NULL)) < 0)
        HGOTO_ERROR(H5E_DATASET, H5E_READERROR, FAIL, "can't read data")

done:
    FUNC_LEAVE_API(ret_value)
} /* end H5Dread() */
```

Virtual Object Layer (VOL)

- **VOL Framework is an abstraction layer within HDF5 Library**
  - Redirects I/O operations into VOL “connector”, immediately after an API routine is invoked
- **VOL Connector implementation**
  - Define callbacks for HDF5 data model operations
  - “Terminates” call by performing action directly, or “passes operation through” by invoking VOL API connector interface

VCD100: VOL Connector Development 100

- **Subscribe to the hdf5vol mailing list:**
  - Email hdf5vol-subscribe@hdfgroup.org with “subscribe” as subject
- **Clone the “external pass-through” example VOL connector**
  - An “external” VOL connector that has all VOL callbacks implemented as transparent “no-ops”, just invoking the underlying VOL connector
    - *External VOL connectors can be loaded with environment variables*
  - https://bitbucket.hdfgroup.org/projects/HDF5VOL/repos/external_pass_through/browse
- **Build the external pass-through connector with logging enabled:**
  - Follow instructions in README in the git repo
- **Modify to your purposes**

Acknowledgements and Credits

- **Part of the development and productization of the feature has been funded by the DOE Exascale Computing Project**

- **Designed and implemented by teams at ANL, LBNL and the HDF Group**

Argonne NATIONAL LABORATORY

BERKELEY LAB
Lawrence Berkeley National Laboratory

The HDF Group

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Questions, Comments, Feedback?

Thank You!