Async VOL: Transparent Asynchronous I/O using Background Threads

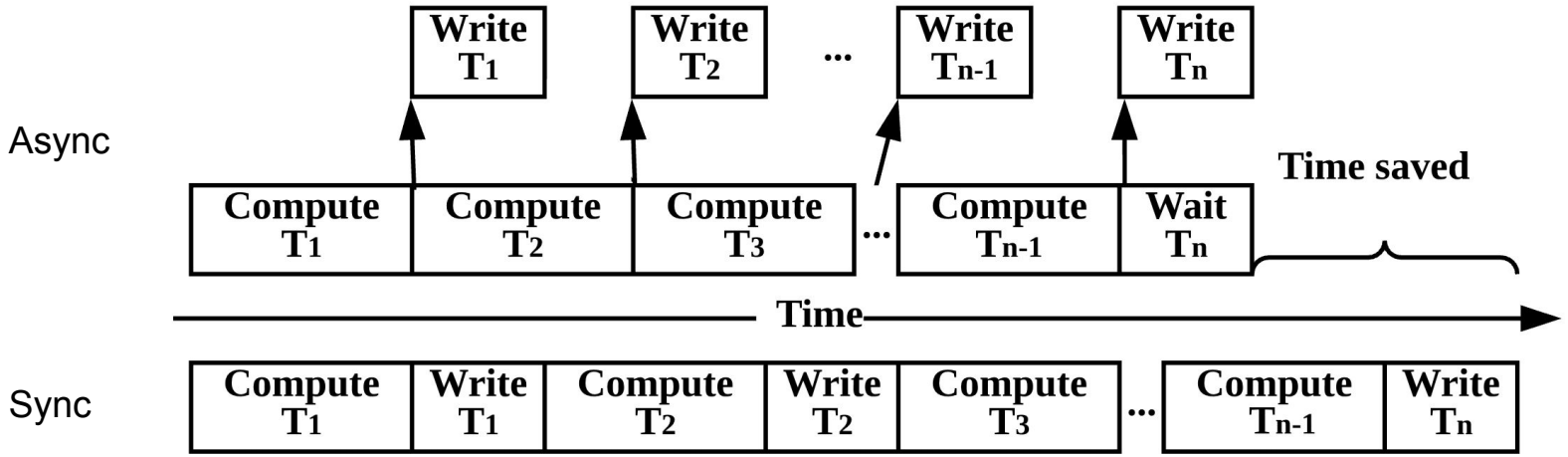
Houjun Tang[1], Quincey Koziol[2], Suren Byna[1],
John Mainzer[3], Huihuo Zheng[4], John Ravi[5]

[1] Berkeley Lab, [2] Amazon, [3] The HDF Group,
[4] Argonne National Lab, [5] NC State University

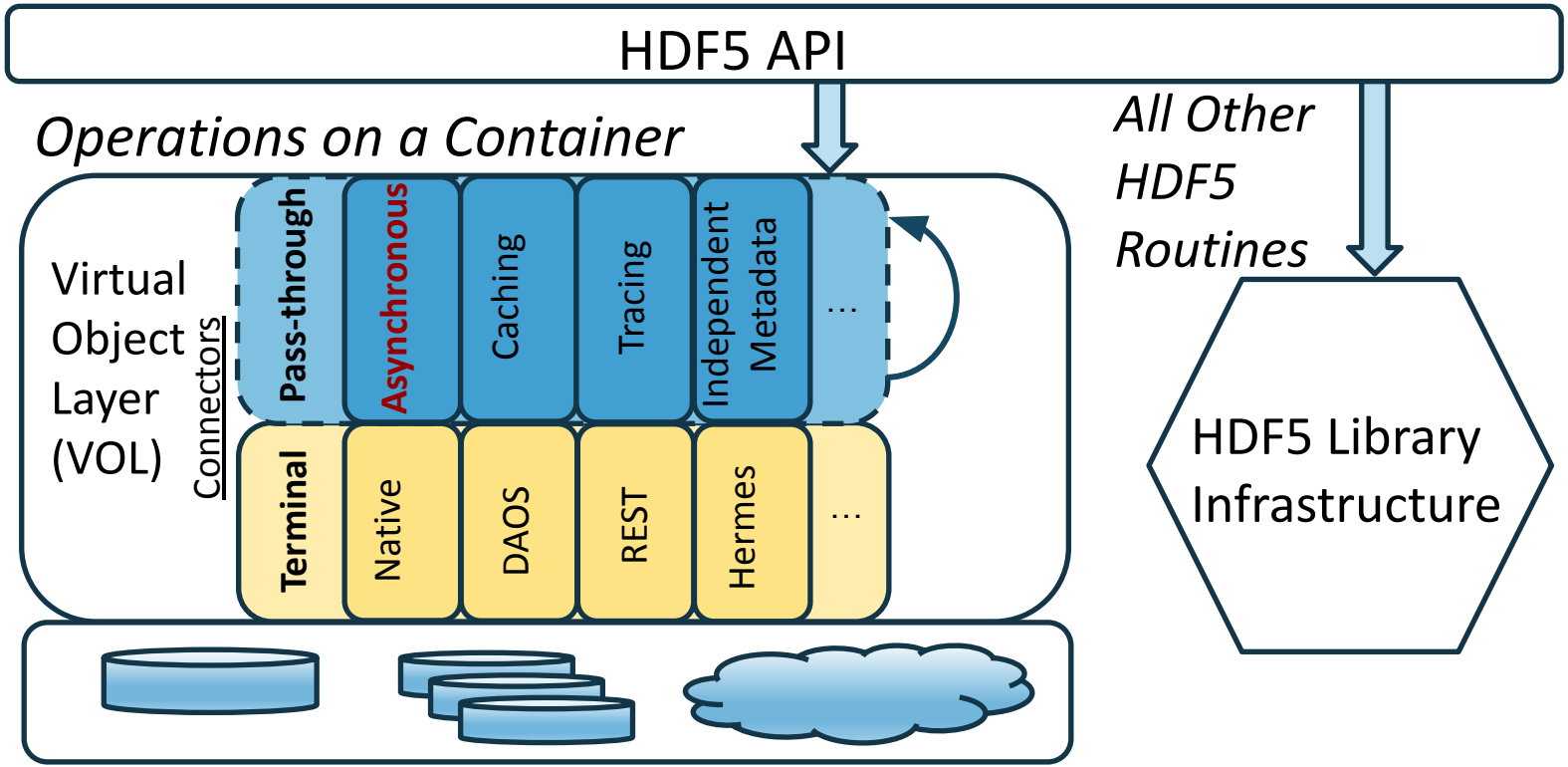
BERKELEY LAB The HDF Group Argonne NATIONAL LABORATORY

ECP EXASCALE COMPUTING PROJECT

Why Async?


Async
Write T1
Write T2
...
Write Tn-1
Write Tn
Compute T1
Compute T2
Compute T3
...
Compute Tn-1
Wait Tn
Time saved
Time
Sync
Compute T1
Write T1
Compute T2
Write T2
Compute T3
...
Compute Tn-1
Write Tn

Virtual Object Layer (VOL)



Spack build coming soon

How to use Async VOL

- **Preparation**
  - HDF5: `git clone https://github.com/HDFGroup/hdf5.git`
  - Async VOL + Argobots: `git clone --recursive https://github.com/hpc-io/vol-async.git`
- **Installation**
  - Compile HDF5 develop branch, with thread-safety support
  - Compile Argobots
  - Compile Async VOL connector
- **Set environment variables**
  - `export LD_LIBRARY_PATH=$VOL_DIR/src:$H5_DIR/lib:$ABT_DIR/lib:$LD_LIBRARY_PATH`
  - `export HDF5_PLUGIN_PATH="$VOL_DIR/src"`
  - `export HDF5_VOL_CONNECTOR="async under_vol=0;under_info={}"`

Detailed instructions: https://hdf5-vol-async.readthedocs.io

Implicit and *Explicit* Asynchronous I/O Execution

- **Implicit**
  - For unmodified HDF5 applications
  - Can be transparently invoked by setting environment variables
  - Dataset writes and reads always block unless stacking with ***Cache VOL***

- **Explicit**
  - For applications that want more control of async operations
    - Uses an “event set” to manage async operations
  - Can extract more performance, e.g. enable async read and write

Explicit Control with EventSet API

- Track and inspect multiple I/O operations with an ***EventSet ID***
- Async version of HDF5 APIs
  - H5Fcreate_async(fname, …, ***es_id***)
  - H5Dwrite_async(dset, …, ***es_id***)
  - …
- Event set control
  - H5EScreate()
  - H5ESwait()
  - H5ESclose()
- Error checking
  - H5ESget_err_status()
  - H5ESget_err_info()

Converting Existing Code

```
// MPI Init
MPI_Init(...);


// Synchronous file create
fid = H5Fcreate(...);
// Synchronous group create
gid = H5Gcreate(fid, ...);
// Synchronous dataset create
did = H5Dcreate(gid, ..);
// Synchronous dataset write
status = H5Dwrite(did, ..);
// Synchronous dataset read
status = H5Dread(did, ..);
...
// Synchronous file close
H5Fclose(fid);
// Continue to computation

...




...
// Finalize
```

```
// Use MPI_THREAD_MULTIPLE
MPI_Init_thread(..., MPI_THREAD_MULTIPLE, &provided);
// Create an event set to track async operations
es_id = H5EScreate();
// Asynchronous file create
fid = H5Fcreate_async(.., es_id);
// Asynchronous group create
gid = H5Gcreate_async(fid, .., es_id);
// Asynchronous dataset create
did = H5Dcreate_async(gid, .., es_id);
// Asynchronous dataset write
status = H5Dwrite_async(did, .., es_id);
// Asynchronous dataset read
status = H5Dread_async(did, .., es_id);
...
// Asynchronous file close
status = H5Fclose_async(fid, .., es_id);
// Continue to computation, overlapping with async operations
...
// Finished computation, Wait for all previous operations in the
    event set to complete
H5ESwait(es_id, H5ES_WAIT_FOREVER, &n_running, &op_failed);
// Close the event set
H5ESclose(es_id);
...
// Finalize
```

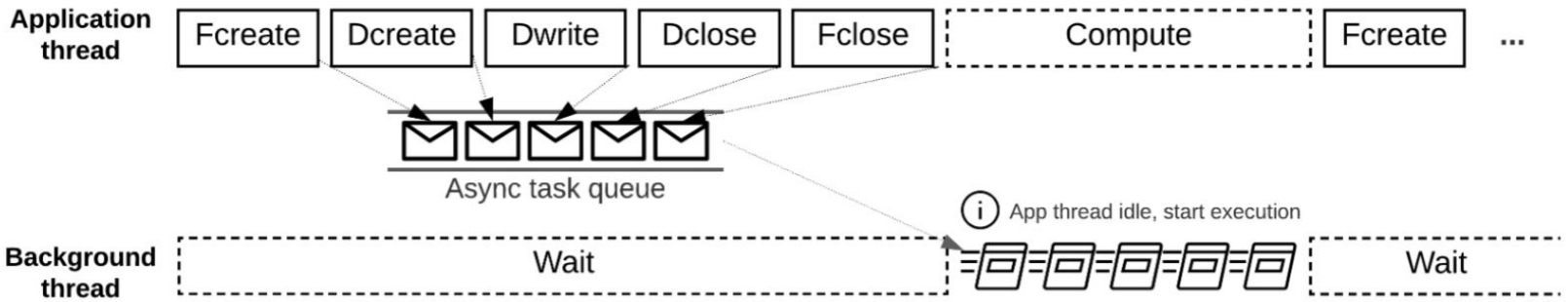
Error Handling

- If an async operation fails, all of its dependent children will not execute
  - If an operation in an event set fails, no further operations can be added to the event set
- An additional error message indicating the parent's failure is appended to the error stack:

```
Async VOL-DIAG: Error detected in Async VOL (0.1) thread 0:
  #000: h5_vol_external_async_native.c line 5766 in async_dataset_create_fn(): Parent task failed
       major: Virtual Object Layer
       minor: Unable to create file
HDF5-DIAG: Error detected in HDF5 (1.13.0) thread 0:
  #001: ../../src/H5VLcallback.c line 3977 in H5VLgroup_create(): unable to create group
       major: Virtual Object Layer
       minor: Unable to create file
  #002: ../../src/H5VLcallback.c line 3904 in H5VL__group_create(): group create failed
       major: Virtual Object Layer
       minor: Unable to create file
  #003: ../../src/H5VLnative_group.c line 72 in H5VL__native_group_create(): unable to create group
       major: Symbol table
       minor: Unable to initialize object
...
```

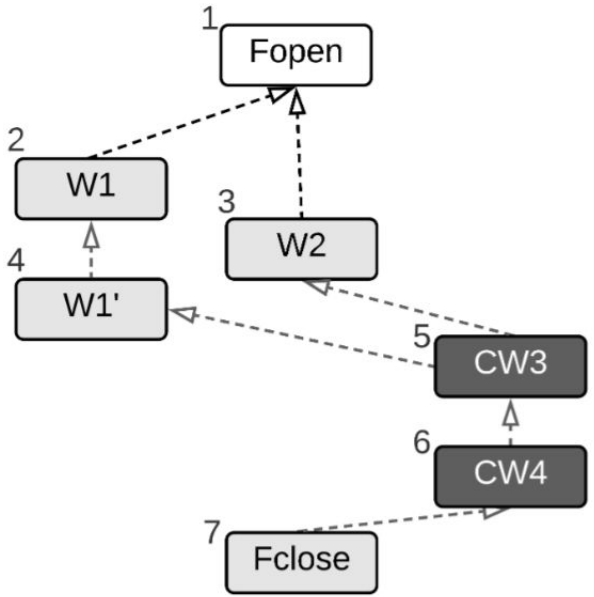
Async VOL with Application Status Detection

- Asynchronous task queue
- Background thread execution



Transparent Dependency Management

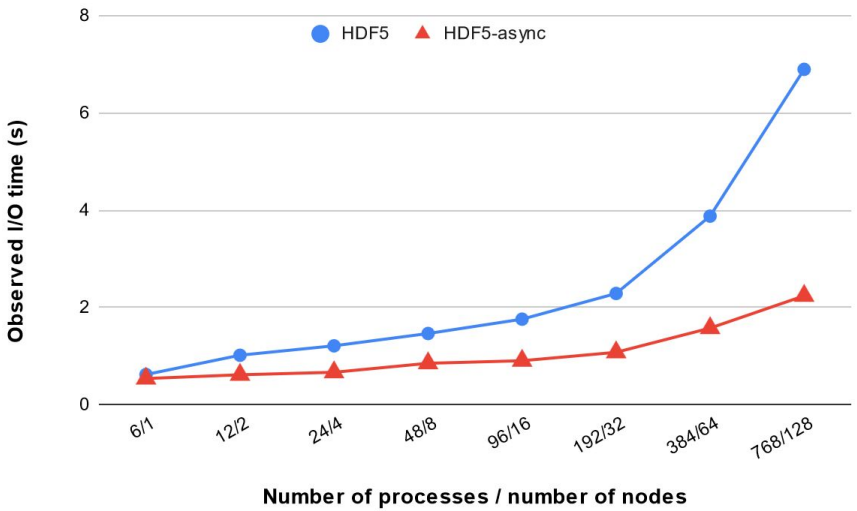
- All I/O operations can only be executed after a successful file create/open.
- A file close operation can only be executed after all previous operations in the file have been completed.
- All read or write operations must be executed after a prior write operation to the same object.
- All write operations must be executed after a prior read operation to the same object.
- All collective operations must be executed in the same order with regard to other collective operations.
- Only one collective operation may be in execution at any time (among all the threads on a process).



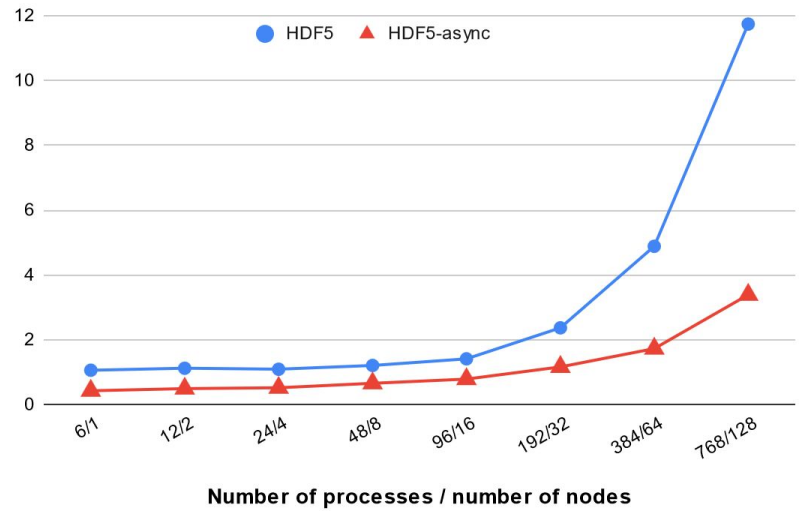
Best Practices

- To achieve best performance
  - Application should have sufficient non-I/O time for asynchronous operations to overlap with
- Avoid application status check
  - When application has an I/O phase that writes data to a file, can inform async vol to start execution at file close time: `export HDF5_ASYNC_EXE_FCLOSE=1`
- Automatic user buffer management
  - When application has extra memory to spare, async VOL can malloc and memcpy the user's buffer when adding `-DENABLE_WRITE_MEMCPY=1` at compile time
    - Env variable `HDF5_ASYNC_MAX_MEM_MB` allows control of memory usage limit
    - Synchronous write when the limit is reached
  - More advanced capabilities available when stacking with ***Cache VOL***
    - Memory and node-local SSD locations for temporary data storage
    - Also support read operations.

Speedup with VPIC-IO and BDCATS-IO on Summit

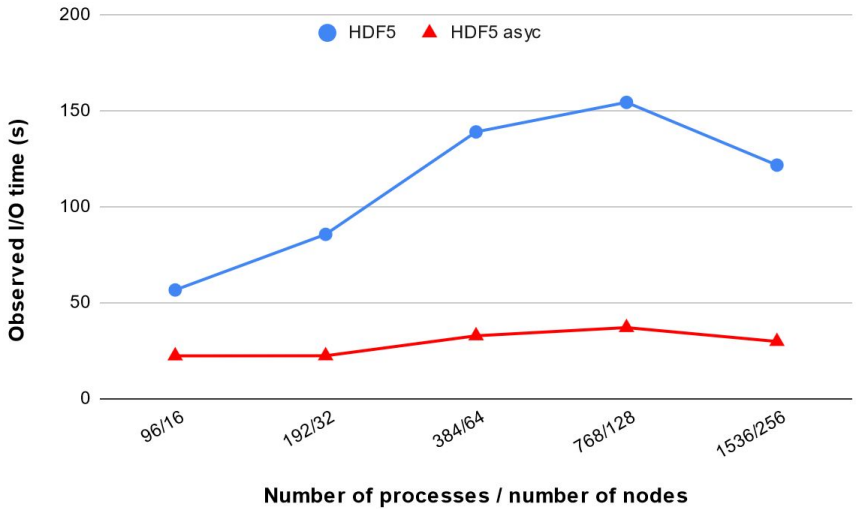


VPIC-IO, 8x32MB write per process, 5 steps total

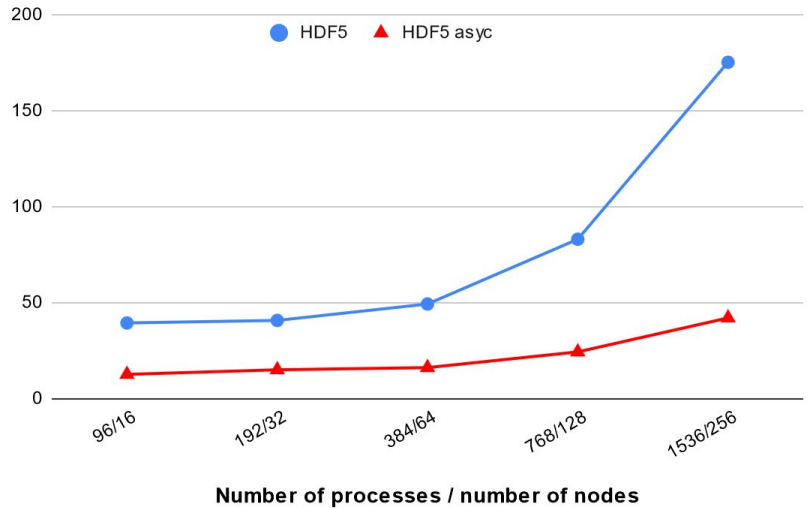


BDCATS-IO, 8x32MB read per process , 5 steps total

Speedup with AMReX Applications on Summit



NyX workload, single refinement level, writes 385GB x 5 steps



Castro workload, single refinement level, writes 385GB x 5 steps

Future Work

- More real application integration
- Merge compatible operations
  - If two async dataset write operations are putting data into same dataset, can merge into only one call
  - Turn multiple ‘normal’ group create operations into a single ‘multi’ group create operation
- Dynamically setting of tuning parameters
  - HDF5 alignment, collective metadata, deferred flush, etc.
  - MPI-IO hints, collective buffer size/count, etc.
  - File system stripping, Lustre stripe size/count, et.c
- Reduce interference with application’s MPI communications
  - Currently may introduce 2 - 5% overhead

https://github.com/hpc-io/vol-async

https://hdf5-vol-async.readthedocs.io

Thanks!

Questions?