Towards Multi-Thread HDF5

John Mainzer john.mainzer@lifeboat.llc
Elena Pourmal elena.pourmal@lifeboat.llc

Outline

- HDF5 at Present
- Multi-thread HDF5
- Towards Multi-thread VOL Support
- Bypass VOL
- Progress to Date – Phase I
- Progress to Date – Phase II

HDF5 at Present

- Fundamentally a single thread library.
- Thread safety supported via a global mutex – only one thread active in the library at a time.
- This constraint is imposed on VOL connectors, even if they can support multi-thread operation.

Multi-Thread HDF5

True multi-thread support has been requested for a long time.

- Retrofitting multi-thread support to an existing large, and largely un-documented code base is a daunting task
- Thus no progress beyond the global mutex – allowing thread safety but not multi-thread execution.

The VOL layer changes the picture.

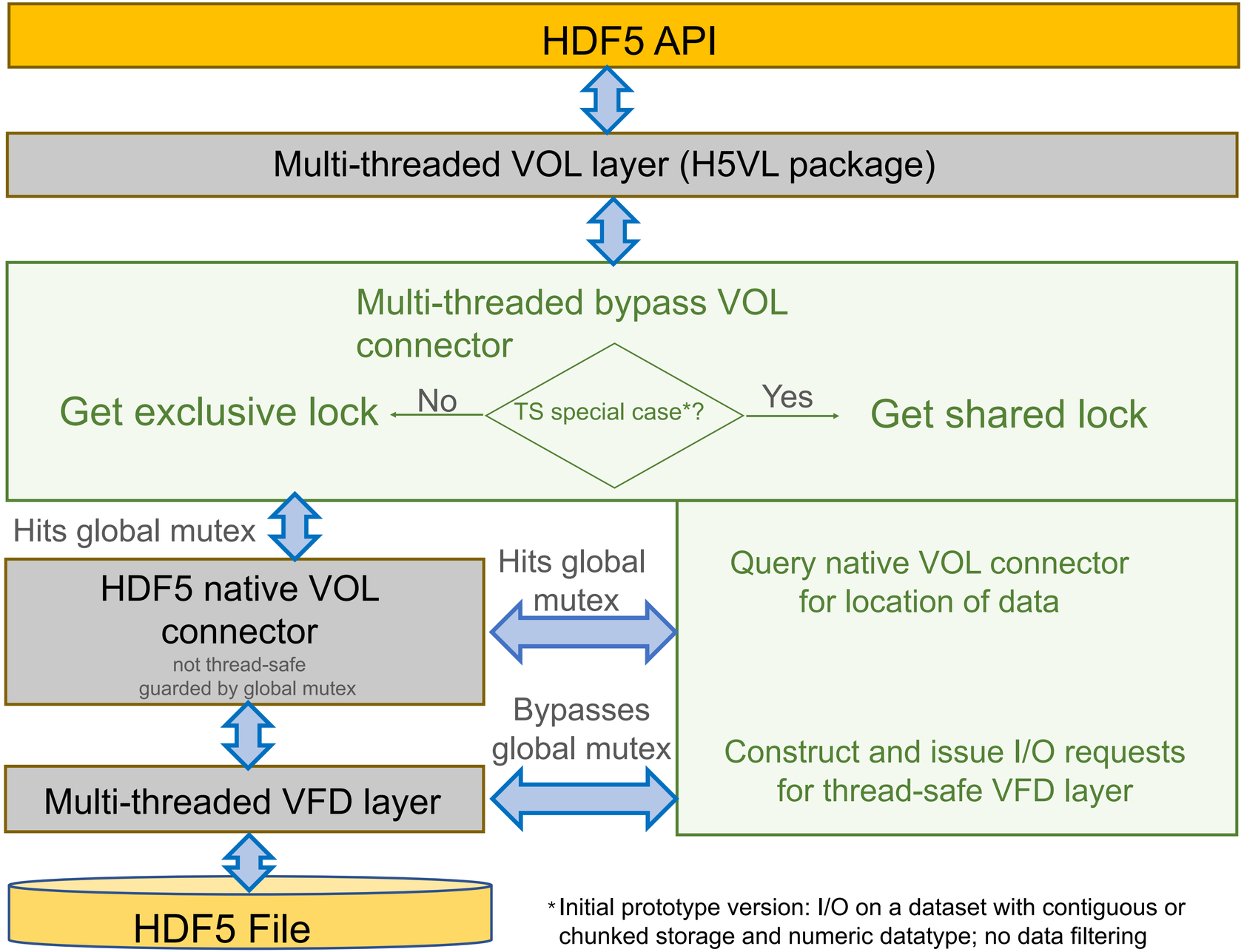
- Pushing the global mutex down a bit, would allow multiple threads of execution into VOL connectors that support it.
- Only need to retrofit multi-thread support onto a small number of HDF5 packages to do this – H5E (error reporting), H5I (index), H5P (property lists), H5CX (context), and H5VL (VOL).
  - Multi-thread versions of the H5S (selections) and H5FD (file driver) packages are desirable, but not necessary for the initial prototype.

Towards Multi-Thread VOL Support

Lifeboat is pursuing this strategy -- objectives are to:

- Retrofit multi-thread support on the required HDF5 packages
- Push the HDF5 global mutex down to allow multiple threads into VOL connectors that support it.
- Develop the Bypass VOL to allow limited multi-thread I/O on HDF5 files.

All modifications to the HDF5 library and related documentation to be contributed to the HDF5 open source project.



* Initial prototype version: I/O on a dataset with contiguous or chunked storage and numeric datatype; no data filtering

Bypass VOL Concept

- Query HDF5 library for the location of raw data
- Execute raw data I/O in parallel in multiple threads

Basic concept has been implemented outside the HDF5 library with good results

Can't implement fully until support for multi-thread VOLS is available, but a prototype single thread version has been implemented.

Plan to develop this version as far as practical, and then convert to multi-thread as an initial test case for multi-thread VOL support

Phase I

In phase I, we reviewed the target modules to identify issues and start designing solutions:

- Found many more interactions between modules than we had hoped.
- Found data structures that would require complex lock ordering schemes in a multi-thread environment – these will have to be redesigned and re-implemented.
- Callback functions in some public APIs present potential deadlock issues. Iterators are particularly problematic.
- Thus, it became obvious that we should use atomics and lock free data structures to the extent possible to minimize the potential for deadlocks both within and between modules.
- Reworking a few public APIs (mostly iterators) for multi-thread is also under consideration. Obviously, this should be avoided where practical and useful.
- Resulting RFCs are on github – they will be updated as phase 2 proceeds.

Phase II -- Objectives:

Implement at least minimal multi-thread VOL support:

- Develop working prototype within a year, and release it for comment
- With input from the user community, develop production code, and
- Contribute it to the HDF5 open source project.

In parallel, implement the Bypass VOL as

- an initial test case for multi-thread VOL support,
- a demonstrator, and
- a product.

Phase II – Plan of attack and status (1):

- Start by retrofitting multi-thread support on H5I (index). To do this, must
  - Implement a lock free hash table – prototype done and available on github
  - Integrate lock free hash table into H5I – done, passes serial tests
  - Retrofit multi-thread support on to the remainder of H5I and write a test suite using the public API. Must do this in a serial build for now as the mutex hasn't been pushed down yet.
- Retrofit multi-thread support on H5E (error reporting) and write a test suite using the public API. As above, must do this in a serial build for now.
- Complete the lock free hash table. In particular, use H5E for error reporting so that it can be properly integrated into the HDF5 library.
- Implement a multi-thread configure option for the HDF5 library
  - Implement infrastructure to support pushing the mutex further down into the HDF5 library, and
  - Apply it to H5E and H5I. When done, calls to external H5E and H5I APIs should not hit the mutex unless they result in calls deeper into the library.

Phase II – Plan of attack and status (2):

- Retrofit multi-thread support on H5P (property lists) To do this, must
  - Re-design and re-implement basic data structures for both performance and thread safety. Some subtle semantic changes possible as a result – discussion on forum as this gets closer.
  - Integrate new data structures into H5P and verify that current serial tests pass.
  - Push global mutex down below H5P as per H5E and H5I.
  - Write a multi-thread test suite using the public API.
- Retrofit multi-thread support on H5CX, push the mutex below it, and write associated multi-thread test suite.
- Ditto with H5VL. Some VOL API changes may be necessary to allow VOL connectors to indicate whether they need the global mutex.
- Finish up Bypass VOL and use it for integration testing.

Acknowledgement

This work is supported by the U.S. Department of Energy, Office of Science under Award number DE-SC0023583 and "Toward multi-threaded concurrency in HDF5”.

References

1. https://github.com/LifeboatLLC/MT-HDF5/tree/main/design_docs
2. https://github.com/LifeboatLLC/MT-HDF5/tree/main/LFHT

Thank you!

Questions?

Lifeboat

www.lifeboat.llc
info@lifeboat.llc

U.S. DEPARTMENT OF ENERGY
AWARDEE™