Questions about thread-safety and concurrent access

Thread-Safety Concurrent Access Troubleshooting Issues

Thread-Safety

Is HDF5 multi-threaded?

No, HDF5 is not multi-threaded.

Is HDF5 thread-safe?

The HDF5 library can be built in thread-safe mode. The thread-safe version of the HDF5 library effectively serializes the HDF5 library calls. It is thread-safe but not thread-efficient.

The thread-safe version of the HDF5 library uses POSIX threads (Pthreads) on Unix and OS X, and Win32 threads on Windows. To build a thread-safe version of the library, specify the *--enable-threadsafe* option when configuring. *--with-pthread=DIR* can be used if your pthreads library is found in a non-standard location, though this is not necessary on most systems.

```
./configure --enable-threadsafe --with-pthread=DIR
```

In CMake, simply enable the HDF5_ENABLE_THREADSAFE option when configuring/building.

Please note that the threadsafe feature is only maintained and supported for C because the thread-safety lock is currently only handled at the C library level. This means that the Java, Fortran, C++, and high-level libraries are not officially supported when built with thread-safety. You can work around this by either using the C API exclusively and/or by creating and maintaining your own locks to serialize HDF5 library access. Also note that, although the library can be built with thread-safety and the higher-level language wrappers by using the *--enable-unsupported* configure option (ALLOW_UNSUPPORTED in CMake), this is definitely not recommended.

For further information on thread-safe HDF5, see the Thread-safe HDF5 page.

The HDF Group has a design plan for a more efficient implementation of thread-safety, but currently does not have the resources to implement the plan. If you are interested in supporting this effort, please contact the HDF Helpdesk.

Can you run Parallel HDF5 and the thread-safe feature together ?

By default, you cannot build Parallel HDF5 with the thread-safe feature. You will receive a configure error if you try this combination. A check was added to the configure to disallow this, as it is not tested and may fail.

However, the *--enable-unsupported* configure option enables you to get around this check at your own risk. (You can actually view the configure file to see where the check is made.) If the build completes properly and the tests pass, then the installation should be okay.

Concurrent Access

Does HDF5 support concurrent access to one or more HDF5 file(s) from multiple threads in a single process?

Concurrent access to one or more HDF5 file(s) from multiple threads in the same process is supported with a thread-safe build of HDF5.

Concurrent access to one or more HDF5 file(s) from multiple threads in the same process will NOT work with a non-thread-safe build of the HDF5 library. The pre-built binaries that are available for download are NOT thread-safe.

Users are often surprised to learn that (1) concurrent access to different datasets in a single HDF5 file and (2) concurrent access to different HDF5 files both require a thread-safe version of the HDF5 library. Although each thread in these examples is accessing different data, the HDF5 library modifies global data structures that are independent of a particular HDF5 dataset or HDF5 file. HDF5 relies on a

semaphore around the library API calls in the thread-safe version of the library to protect the data structure from corruption by simultaneous manipulation from different threads. Examples of HDF5 library global data structures that must be protected are the freespace manager and open file lists.

Does HDF5 support concurrent access to a single dataset from multiple processes?

If all processes are reading, then, yes, HDF5 (serial) does support this. If there are any processes that are writing, then you must use the "Single Write Multiple Read" (SWMR) feature available in HDF5-1.10. This feature is not available in earlier releases.

Multiple processes should not be confused with multiple threads. Below is a summary regarding multiple "things" accessing a file:

- Multiple processes DO NOT require the thread-safe library (SWMR is required if there is a writer, but not thread-safety).
- Multiple threads DO require the thread-safe library.

The reason for this is that separate processes do not share memory and cannot affect each other. Threads DO share memory and CAN interfere with each other.

Can you read an HDF5 file while it is being written to?

It is possible for multiple processes to read an HDF5 file when it is being written to, and still read correct data. (The following steps should be followed, EVEN IF the dataset that is being written to is different than the datasets that are read.)

Here's what needs to be done:

- Call H5Fflush() from the writing process.
- The writing process _must_ wait until either a copy of the file is made for the reading process, or the reading process is done accessing the file (so that more data isn't written to the file, giving the reader an inconsistent view of the file's state).
- The reading process _must_ open the file (it cannot have the file open before the writing process flushes its information, or it runs the risk of having its data cached in memory being incorrect with respect to the state of the file) and read whatever information it wants.
- The reading process must close the file.
- The writing process may now proceed to write more data to the file.

There must also be some mechanism for the writing process to signal the reading process that the file is ready for reading and some way for the reading process to signal the writing process that the file may be written to again.

Troubleshooting Issues

Keep in mind that thread-safety has nothing to do with particular file operations, but is about library operations. Regardless of read-only or not, all library operations are not thread-safe unless the thread-safe library is used.

Some things can still go wrong even if building HDF5 with thread-safety turned on:

- A thread-safe library was not actually built. Check the libhdf5.settings file to see if thread-safety was included.
- The application is actually linking to a non-thread-safe version of the library installed elsewhere on the system.
- The application is using the High Level, Fortran, Java, or C++ wrappers, none of which have been verified to be thread-safe.
- There is a non-HDF5 bug in the application.

Verify the following:

- A serial program works when linked to your library
- H5_IS_LIBRARY_THREADSAFE returns TRUE.
  If that works and you are not using a wrapper, then the library should work.

If the issue cannot be resolved, please send us a minimum working example in C which shows the failure. Multi-threaded programming can be difficult and there may be an unseen issue in the code.