SWMR Phase II Proposal

The HDF Group

The document describes the work that needs be done to bring the HDF5 SWMR prototype developed during the Single Writer/Multiple Readers (SWMR) Phase I Project into the HDF5 software.

1 Background

During March – June 2013 The HDF Group worked on prototyping a new feature in the HDF5 library to allow one process to append data to the existing datasets while multiple processes read data from the modified datasets (SWMR Phase I). The goal of this work was to identify deficiencies of the prototype implementation and to estimate the effort required releasing the SWMR feature as a part of the maintainable HDF5 software.

In Phase I The HDF Group documented the design of the HDF5 SWMR feature as described in [1], developed a work breakdown structure (WBS) for the SWMR implementation and integration, and provided the estimates for each work package in WBS. The HDF Group also delivered a version of the HDF5 library with a SWMR prototype implementation for Use Cases 1.7 – 1.9 described in [1].

The Phase I deliverables were:[1]

1. SWMR documents [2] -
    a. SWMR design document
    b. SWMR semantics document
    c. SWMR test design document
2. C programs and a shell script to run the programs to test a file system for POSIX compliance [3].
3. SWMR prototype to demonstrate Use Cases 1.7 – 1.9 [3] as described in [1].
4. Estimate for Phase II (this document)

Findings from Phase I and recommendations are summarized in the next section.

[1] This list of deliverables as described in the modification to the DLS contract issued on June 14, 2013.

2 SWMR Phase I findings and recommendations

Current prototype implementation of SWMR is based on an assumption that HDF5 file for the SWMR access resides on a file system compliant with the POSIX I/O semantics. Two major features of POSIX I/O semantics access are critical for the implementation:

1. POSIX I/O writes must be performed in a sequentially consistent manner.
2. Writes to the file must appear as atomic operations to any readers that access the file during write.

These semantics apply to any processes that access the file from any location.

Investigation of a SWMR regression test failures showed that assumption 2 above was not valid on several file systems including Lustre and Linux ext3/4. Checksum on a piece of HDF5 metadata was incorrect showing the "torn write".

To mitigate the issue the HDF5 developers considered two enhancements to the HDF5 file format and library:

1. Retry a read operation when a "torn" metadata record is found.
2. Add sentinel bytes to the HDF5 metadata records to assure consistent view of the record.

The HDF Group added a "retry read" operation to the HDF5 prototype code that exposed the "torn write" problem with the SWMR regression test on Linux systems to show the feasibility of the approach 1.

The HDF5 developers discussed the "sentinel bytes" solution and propose NOT to implement it for the following reasons:

- All metadata records except B1-tree structures have 32-bit checksum associated with them. The risk of getting a "torn write" metadata record with the correct checksum is 1 in 4 billion and considered low. Thus, introduction of the sentinel bytes, while providing a second line of defense against the "torn write", is not necessary.
- B1-tree structures are used for chunk indexing in all HDF5 versions including 1.8. Having checksum on the B1-tree structures is required for the SWMR implementation to assure consistency of the metadata record, but it introduces a file format change as new data structures such as fixed arrays, extensible arrays, and B2-trees do.

  B1-tree chunk indexing is not recommended due to its poor performance for data append operation; thus the new data structures should be used instead.

  Changing the file format and the library to use a poorly performed feature (B1-trees for chunk indexing) will be a waste of resources and increases the cost of implementation and future maintenance while not providing any benefits to the applications.

The following approach is proposed based on Phase I findings:

1. SWMR implementation will use new data structures such as fixed arrays, extensible arrays, and B2-trees for chunk indexing.

   The new structures make the SWMR created files incompatible with the HDF5 1.8 based applications. The h5repack utility can be used to mitigate the issue after SWMR applications finish file modifications.

2. The current SWMR prototype will be enhanced to address the “torn write” issues on non-POSIX compliant file systems by introducing “retry read” on metadata records.

3 SWMR Phase II work

The table describes the work packages for integrating the SWMR feature into HDF5.

Task	Work Package and its description	Deliverable
1.	***HDF5 Library modifications to support SWMR*** We will modify metadata (MD) cache clients to perform "retry read" operation on metadata records with wrong checksum. We will also provide knobs for application to define the number of retries and provide retry statistics.	1. HDF5 source code with the new feature 2. Updated File Format specification 3. Design document on MD cache implementation for SWMR
2.	***Extended "black box" testing for SWMR*** We will create a test suite to exercise raw data modification scenarios including a proposed DLS NeXus approach of writing a master file with external links to the HDF5 files.	HDF5 source code with the "black box" test suite that passes on Linux, Solaris, Mac OS X, and Windows
3.	***Extended "white box" testing for SWMR*** We will implement low-level testing for the data structures used by SWMR.	HDF5 source code with the "white box" test suite that passes on Linux, Solaris, Mac OS X, and Windows
4.	***SWMR APIs extensions*** We will implement proposed extensions to the HDF5 library as described in Appendices A, B and D of the RFC THG 2013- 02-06.v8 document [1]	HDF5 source code with the implemented features
5.	***SWMR release candidate***	Fully integrated with the HDF5 1.9 branch and documented SWMR feature ready for official HDF5 1.10.0 release
	Total:	

We propose to perform the work in 5 stages as listed in the table. This approach will allow us to mitigate the risks and to adjust the remaining work to address the issues that may be discovered during each stage. It will also provide the customers with a working piece of software after the completion of each stage.

References

1. RFC THG 2013- 02-06.v8 document "SWMR Requirements and Use Cases", ftp://ftp.hdfgroup.uiuc.edu/pub/outgoing/SWMR/doc/SWMR%20Use%20Cases-2013-03-13.pdf
2. SWMR design documentation, ftp://ftp.hdfgroup.uiuc.edu/pub/outgoing/SWMR/doc/
3. HDF5 SWMR source code including POSIX test, http://svn.hdfgroup.uiuc.edu/hdf5/branches/revise_chunks/

Revision History

June 14, 2013:	Version 1 circulated for comment within The HDF Group.
December 11, 2013	Version 2 was modified to simplify table in Section 3 and posted on FTP for reference