RFC: Switching to a 64-bit *hid_t* Space in HDF5

Mohamad Chaarawi

Quincey Koziol

The HDF5 library creates an object identifier (also called a handle), of type *hid_t*, for all objects that the user requests access to. This identifier is used in all further accesses to the object. When the object is closed, the identifier becomes invalid.

The type *hid_t* is currently defined in the HDF5 library as a 32-bit integer (*int*). We found that many large scale applications run out of object identifiers. This RFC proposes to change the internal type definition of *hid_t* to a 64-bit integer (*int64_t*) to address the issue. The change will make the identifier allocation algorithm much simpler and faster since the huge space for the identifiers granted by the 64-bit integer type makes it unnecessary to reuse identifiers previously consumed. Applications that work with a huge number of objects will see performance improvement.

Introduction

An object in the HDF5 library is accessed through a unique identifier that the library returns when creating or opening the object. The identifier is of type *hid_t* which is currently defined as a 32-bit integer inside the HDF5 library. Objects that are accessed using identifiers include HDF5 files, groups, datasets, datatypes, property lists, and attributes. If an object is opened twice, two different identifiers are issued, although they access the same object.

The 32-bit integer limits the number of unique identifiers that the library can create to access objects to what a 32-bit integer can hold ($2^{32} - 1$). Recently, this has been a problem as some applications create more objects than the limit. Employing an algorithm to recycle and reuse identifiers for the objects that have been closed proved insufficient as it still limits the total number of open objects. Furthermore, it imposes a performance penalty. The intuitive solution is to expand the identifiers space to 64-bits, which will be more than enough for applications in the foreseeable future.

Motivation

Many applications that use HDF5 for I/O today create a very large number of HDF5 objects. As we are moving to the exascale era, this number will only increase. We have already seen a few users complaining about the limit of the number of objects that can be opened concurrently. In the prototype implementation of the DOE FastForward storage I/O project, many objects were needed to access one cell/element of an HDF5 dataset. The limit for the number of open objects in the HDF5 library was encountered for prototype/testing workloads. In multithreaded applications, this limit is multiplied by the number of threads calling into the library.

Clearly there is a need to allow more objects to be created and furthermore allow more concurrent objects to be open. As mentioned earlier, an algorithm to recycle identifiers that have been closed and rendered invalid was used; however that had some limitations, as each new identifier issued after the identifier range maximum has been hit requires a linear search through the identifier range to look for an available value (which is an $O(n^2)$ algorithm). Furthermore it doesn't solve the issue of having a large number of objects accessible concurrently.

Approach

Proposed Solution

We propose to change the definition of the hid_t type inside the library from a 32-bit signed integer to a 64-bit signed integer. This will significantly expand the identifiers space for the users of the HDF5 library, which should be more than enough for applications in the foreseeable future.

The change in the library is simple and has been done in a branch off the trunk:

https://svn.hdfgroup.uiuc.edu/hdf5/features/64bit_hid_t

Effect on third party software

We anticipate that some third party software and applications that were programmed under the assumption that the *hid_t* type is a 32-bit integer would need some rework. Note that the assumption is wrong, since a variable of the *hid_t* type should always be treated as handle that is transient and not persistent and whose type is private to the HDF5 library.

Recommendation

We recommend implementing the change in a branch, testing the branch thoroughly, and merging it into the trunk when all daily tests pass. The feature will be released with the 1.10 series.

As soon as trunk with the new feature passes all daily tests, we will create a snapshot and will contact our major stakeholders to test it with their software. We will be working with our major customers to resolve any problems that will arise from this change.

Revision History

July 22, 2014: Version 1 circulated for comment within The HDF Group.