HDF5 introduction

G. Giuliani

ICTP - Earth System Physics Section

Introductory School on Parallel
Programming and Parallel Architecture
for High-Performance Computing
Monday 10 October 2016

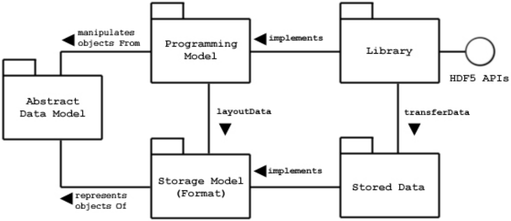
Why Scientific Data Format?

- Portability
- Named datasets
- Hierarchical data organization
- Metadata attributes
- Physical storage space reduction
- Standard for visualization
- High percormance I/O

HDF5 is a format to store large numerical arrays of homogeneous type organizing them hyerarchically and tagging them with arbitrary metadata.

HDF5 data model

Data Model: The Hierarchical Data Format (HDF) implements a model for managing and storing data which decouples through the API the USER view from the on-file storage model and storage mechanism.

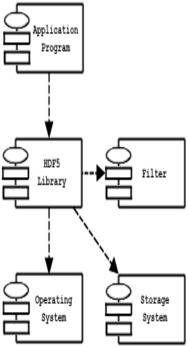


HDF5 data model

The HDF5 Library implements the Programming Model and Abstract Data Model

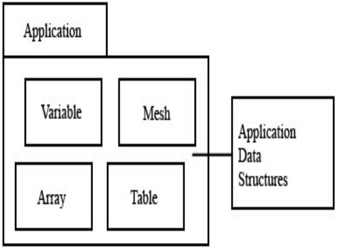
- calls the Operating System or other Storage Management software (e.g., the MPI/IO Library) to store and retrieve persistent data.
- links to other software, such as filters for compression.
- is linked to an application program

The application program implements problem specific algorithms and data structures, and calls the HDF5 Library to store and retrieve data.



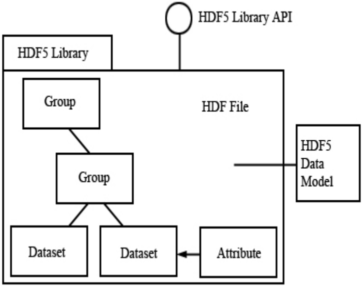
Application view

The application view is how the data are organized in the user application, which is relevant to the user running the program.



HDF5 API abstract model

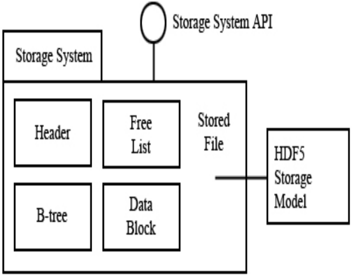
Through the HDF5 API the programmer translates the application view into the HDF5 abstract data view.



HDF5 storage model

The objects of the HDF5 Abstract Data Model are mapped to the objects of the HDF5 Storage Model, and stored in a storage medium. The stored objects include:

- header blocks
- free lists
- data blocks
- B-trees
- link to other stored objects



Abstract model

The Abstract Data Model (ADM) defines concepts for defining and describing complex data stored in files.

- File - a contiguous string of bytes in a computer storage
- Group - a collection of objects (including groups)
- Dataset - a multidimensional array of Data Elements, with Attributes and other metadata.
- Datatype - a description of a specific class of data element, including its storage layout as a pattern of bits.
- Dataspace - a description of the dimensions of a multidimensional array.
- Attribute - a named data value associated with a group, dataset, or named datatype
- Property List - a collection of parameters controlling options in the library.

How to map?

If you have SIMPLE data then you are mostly concerned about:

- The Dataset is any array like object you want to store on disk
- The Group is a logical container which contains datasets or other group
- The Attribute is any user defined bit of metadata to attach to file, groups, datasets

Examples

CAVEAT : will chose Python here!!! Extensive documentation for the native C/C++/FORTRAN/Java APIs can be found on the HDF5 site !

Write and read from file

```
import numpy as np
import h5py
f = h5py.File("test.h5")
mydata = f.create_dataset("varname",shape=(1024,),
                dtype=’int32’,compression=’gzip’)
mydata[:] = np.arange(1024)
f.close( )
f = h5py.File("test.h5")
f["varname"][-1]
1023
```

Groups and attibutes

```
f = h5py.File("test.h5")
mydata = np.arange(1024)
f["/newbase/mydata1"] = mydata
f["/newbase/mydata1"].attrs["long_name"] = "this is
mydata = mydata + 1024
f["/newbase/mydata2"] = mydata
f["/newbase/mydata1"].attrs["long_name"] = "this is
f.close( )
f = h5py.File("test.h5")
f.keys( ) # python 3 is list(f.keys())
['newbase']
f["newbase"].keys( )
['mydata1', 'mydata2']
```

Going parallel

HDF5 must be built with at least the following options:

```
./configure --enable-parallel --enable-shared
```

Often, a parallel version of HDF5 will be available through your package manager. You can check to see what build options were used by using the program h5cc:

```
h5cc -showconfig | grep Parallel
```

Using parallel

```
mpirun -np 4 script.py
```

```
from mpi4py import MPI
import numpy as np
import h5py
comm = MPI.COMM_WORLD
rank , nproc = comm.rank , comm.size
mine = 1000//nproc
start , end = rank*mine , start+mine
f = h5py.File("test.h5", "w",
              driver="mpio", comm=comm)
dset = f.create_dataset("data",(1000,),dtype=’f4’)
dset[start:end] = rank+1
f.close
```