Introduction to HDF5

Quincey Koziol
The HDF Group
Unidata netCDF Workshop
October 28-29, 2010
What is HDF5?

• Open **file format**
  • Designed for high volume or complex data

• Open source **software**
  • Works with data in the format

• A **data model**
  • Structures for data organization and specification
HDF = Hierarchical Data Format

- HDF4 is the first HDF
  - Originally called HDF; last major release was version 4

- HDF5 benefits from lessons learned with HDF4
  - Changes to file format, software, and data model
  - HDF5 and HDF4 are different

- No plans for an HDF6!
HDF5 is like ...

- Databases
- Random access; subsetting
- Hierarchical; collections of related information
- Directories & Files
- Standard exchange format; heterogeneous information
- HDF5
- Binary "Flat File"
- High-performance; compact; scalable
- XML
- Self-describing; extensible types; rich metadata
HDF5 is designed ...

- for high volume and/or complex data
- for every size and type of system (portable)
- for flexible, efficient storage and I/O
- to enable applications to evolve in their use of HDF5 and to accommodate new models
- to support long-term data preservation
HDF5 Technology Platform

- **HDF5 data model**
  - The “building blocks” for data organization and specification

- **HDF5 software**
  - Library, language interfaces, tools

- **HDF5 file format**
  - Bit-level organization of HDF5 file

Let’s look at....
HDF5 Data Model

Dataset
Group
Attribute

HDF5 Objects
File
Link
Datatype
Dataspace

a.k.a. HDF5 Abstract Data Model
a.k.a. HDF5 Logical Data Model
An HDF5 file is a container that holds data objects.
• HDF5 datasets **organize and contain** “raw data values”.
  • HDF5 datatypes describe individual data elements.
  • HDF5 dataspaces describe the logical layout of the data elements.
HDF5 Dataspaces

• Describe the logical layout of the elements in an HDF5 dataset
  • NULL
    • no elements
  • Scalar
    • single element
  • Simple array (most common)
    • multiple elements organized in a rectangular array
      • rank = number of dimensions
      • dimension sizes = number of elements in each dimension
      • maximum number of elements in each dimension
        • may be fixed or unlimited
HDF5 Dataspaces

Two roles:

Dataspace contains spatial information (logical layout) about a dataset stored in a file

- Rank and dimensions
- Permanent part of dataset definition

Partial I/O: Dataspace describes application’s data buffer and data elements participating in I/O
HDF5 Dataset & Dataspace

• HDF5 datasets organize and contain “raw data values”.

• HDF5 dataspaces describe the logical layout of the data elements.
HDF5 Datatypes

- Describe individual data elements in an HDF5 dataset
- Wide range of datatypes supported
  - Signed/unsigned Integer
  - Float
  - User-defined (e.g., 13-bit integer)
  - Fixed and variable-length strings
  - Variable length sequences
  - Arrays
  - Compound (similar to C structs)
  - Enumerated
  - Many more ...
HDF5 Dataset

- Rank: 2
- Dimensions: 5 x 3
- Datatype: 32-bit Integer
- Dataspase: Rank = 2, Dimensions = 5 x 3

October 28, 2010
HDF5 Dataset with Compound Datatype

Dataspace:
- Rank = 2
- Dimensions = 5 x 3

Compound Datatype:
- int16
- char
- int32
- 2x3x2 array of float32
HDF5 Dataset & Datatype

- HDF5 datasets organize and contain “raw data values”.
  - HDF5 datatypes describe individual data elements.
• HDF5 datasets organize and contain “raw data values”.
  • HDF5 datatypes describe individual data elements.
  • HDF5 dataspaces describe the logical layout of the data elements.
HDF5 Data Model: Are we there yet?

HDF5 Objects

- Group and Link
- Attribute
- Dataspace
- Datatype
- Dataset
- File

October 28, 2010

HDF/HDF-EOS Workshop XIV

www.hdfgroup.org
HDF5 Attributes

• Typically contain user metadata

• Have a name and a value

• Are associated with HDF5 objects.

• Value is described by a datatype and a dataspace
  • analogous to a dataset
HDF5 groups and links organize data objects.

Every HDF5 file has a root group

- **Parameters**: 10;100;1000
- **SimOut**
  - lat | lon | temp
  - 12 | 23 | 3.1
  - 15 | 24 | 4.2
  - 17 | 21 | 3.6

- **Viz**
  - Timestep 36,000
  - Experiment Notes:
    - Serial Number: 99378920
    - Date: 3/13/09
    - Configuration: Standard 3

- **SimOut**
  - Parameters 10;100;1000

October 28, 2010
NetCDF Workshop
HDF5 Technology Platform

- **HDF5 data model**
  - The “building blocks” for data organization and specification

- **HDF5 software**
  - Library, language interfaces, tools
HDF5 Home Page

HDF5 home page: [http://hdfgroup.org/HDF5/](http://hdfgroup.org/HDF5/)

- Latest release: HDF5 1.8.5 (1.8.6 coming in November!)

HDF5 source code:

- Written in C, and includes optional C++, Fortran 90 APIs, and High Level APIs
- Contains command-line utilities (h5dump, h5repack, h5diff, ..) and compile scripts

HDF5 pre-built binaries:

- When possible, include C, C++, F90, and High Level libraries. Check ./lib/libhdf5.settings file.
- Built with and require the SZIP and ZLIB external libraries
HDF5 API and Applications

Applications
Domain Data Objects

EOS Application
EOS library

HDF5 Library

Storage
HDF5 Software Layers & Storage

- **Language Interfaces**: C, Fortran, C++
- **HDF5 Data Model Objects**: Groups, Datasets, Attributes, ...
- **Tunable Properties**: Chunk Size, I/O Driver, ...

**Internals**:
- Memory Mgmt
- Datatype Conversion
- Filters
- Chunked Storage
- Version Compatibility
- and so on...

**Virtual File Layer**
- Posix I/O
- Split Files
- MPI I/O
- Custom

**HDF5 File Format**
- File
- Split Files
- File on Parallel Filesystem

**Tools**
- ... High Level APIs
- h5dump tool
- h5repack tool
- HDFview tool
- Java Interface

**HDF5 Library**
- H5dump tool
- HDFview tool
- h5repack tool

**I/O Drivers**
- Virtual File Layer
- I/O Drivers

**Storage**
- HDF5 File Format
- File
- Split Files
- File on Parallel Filesystem

October 28, 2010

NetCDF Workshop
Useful Tools For New Users

h5dump:
Tool to “dump” or display contents of HDF5 files

h5cc, h5c++, h5fc:
Scripts to compile applications

HDFView:
Java browser to view HDF4 and HDF5 files
http://www.hdfgroup.org/hdf-java-html/hdfview/
Introduction to HDF5 Programming Model and APIs
General Programming Paradigm

- Object is opened or created
- Object is accessed, possibly many times
- Object is closed

- Properties of object are optionally defined
  - Creation properties
  - Access properties
Order of Operations

- An order is imposed on operations by argument dependencies

For Example:

A file must be opened before a dataset
because the dataset open call requires a file handle as an argument.

- Objects can be closed in any order.
The General HDF5 API

- Currently C, Fortran 90, Java, and C++ bindings.
- C routines begin with prefix `H5?`
  
  ? is a character corresponding to the type of object the function acts on

Example Functions:

- **H5D**: Dataset interface  
  
  e.g., `H5Dread`

- **H5F**: File interface  
  
  e.g., `H5Fopen`

- **H5S**: dataSpace interface  
  
  e.g., `H5Sclose`
HDF5 Defined Types

For portability, the HDF5 library has its own defined types:

- **hid_t**: object identifiers (native integer)
- **hsize_t**: size used for dimensions (unsigned long or unsigned long long)
- **herr_t**: function return value
- **hvl_t**: variable length datatype

Note: This is not an exhaustive list!

For C, include hdf5.h in your HDF5 application.
The HDF5 API

• For flexibility, the API is extensive
  ✓ 300+ functions

• This can be daunting... but there is hope
  ✓ A few functions can do a lot
  ✓ Start simple
  ✓ Build up knowledge as more features are needed
Basic Functions

- \texttt{H5Fcreate} (\texttt{H5Fopen})\hfill create (open) File
  - \texttt{H5Screate} \_\_\_ simple/\texttt{H5Screate}\hfill create dataSpace
    - \texttt{H5Dcreate} (\texttt{H5Dopen})\hfill create (open) Dataset
      - \texttt{H5Dread}, \texttt{H5Dwrite}\hfill access Dataset
      - \texttt{H5Dclose}\hfill close Dataset
  - \texttt{H5Sclose}\hfill close dataSpace
- \texttt{H5Fclose}\hfill close File
<table>
<thead>
<tr>
<th><strong>Data Spaces:</strong></th>
<th>H5Sselect_hyperslab (Partial I/O)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>H5Sselect_elements (Partial I/O)</td>
</tr>
<tr>
<td></td>
<td>H5Dget_space</td>
</tr>
<tr>
<td><strong>Groups:</strong></td>
<td>H5Gcreate, H5Gopen, H5Gclose</td>
</tr>
<tr>
<td><strong>Attributes:</strong></td>
<td>H5Acreate, H5Aopen_name, H5Aclose,</td>
</tr>
<tr>
<td></td>
<td>H5Aread, H5Awrite</td>
</tr>
<tr>
<td><strong>Property lists:</strong></td>
<td>H5Pcreate, H5Pclose</td>
</tr>
<tr>
<td></td>
<td>H5Pset_chunk, H5Pset_deflate</td>
</tr>
</tbody>
</table>
High Level APIs

- Included along with the HDF5 library
- Simplify steps for creating, writing, and reading objects.
- Do not entirely ‘wrap’ HDF5 library
Example HDF5 Code
Steps to Create a File

1. Decide on properties the file should have and create them if necessary:
   • Creation properties
   • Access properties
   • We will use Default properties.

2. Create the file

3. Close the file and the property lists, as needed
Code: Create a File

hid_t file_id;
herr_t status;

file_id = H5Fcreate("file.h5", H5F_ACC_TRUNC, 
                    H5P_DEFAULT, H5P_DEFAULT);

status = H5Fclose (file_id);

"/" (root)

Note: Return codes not checked for errors in code samples.
Steps to Create a Dataset

1. Define dataset characteristics
   a) Datatype – integer
   b) Dataspace - 4x6
   c) Properties if needed, or use H5P_DEFAULT

2. Decide where to put it
   2. Group or root group

3. Create dataset in file

4. Close everything
HDF5 Pre-defined Datatype Identifiers

HDF5 defines* set of Datatype Identifiers per HDF5 session.

For example:

<table>
<thead>
<tr>
<th>C Type</th>
<th>HDF5 File Type</th>
<th>HDF5 Memory Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>H5T_STD_I32BE</td>
<td>H5T_NATIVE_INT</td>
</tr>
<tr>
<td></td>
<td>H5T_STD_I32LE</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>H5T_IEEE_F32BE</td>
<td>H5T_NATIVE_FLOAT</td>
</tr>
<tr>
<td></td>
<td>H5T_IEEE_F32LE</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>H5T_IEEE_F64BE</td>
<td>H5T_NATIVE_DOUBLE</td>
</tr>
<tr>
<td></td>
<td>H5T_IEEE_F64LE</td>
<td></td>
</tr>
</tbody>
</table>

* Value of datatype is NOT fixed
Pre-defined File Datatype Identifiers

Examples:

- **H5T_IEEE_F64LE**: Eight-byte, little-endian, IEEE floating-point
- **H5T_STD_I32LE**: Four-byte, little-endian, signed two's complement integer

**NOTE**: What you see in the file. Name is the same everywhere and explicitly defines a datatype.

*STD= “An architecture with a semi-standard type like 2’s complement integer, unsigned integer...”*
Pre-defined Native Datatypes

Examples of predefined native types in C:

- H5T_NATIVE_INT  (int)
- H5T_NATIVE_FLOAT (float )
- H5T_NATIVE_UINT  (unsigned int)
- H5T_NATIVE_LONG  (long )
- H5T_NATIVE_CHAR  (char )

NOTE: Memory types.
Different for each machine.
Used for reading/writing.
Code: Create a Dataset

```c
1  hid_t     dataspace_id;
2  hsize_t    dims[2];

.  .  .
5  dims[0] = 4;
6  dims[1] = 6;
7  dataspace_id = H5Screate_simple (2, dims, NULL);
```

Define a dataspace

- rank
- current dims
Code: Create a Dataset

1 hid_t file_id, dataset_id, dataspace_id;

.
.
.

8 dataset_id = H5Dcreate (file_id, "A", H5T_STD_I32BE, dataspace_id, H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
Code: Create a Dataset

```c
hid_t file_id, dataset_id, dataspace_id;
hsize_t dims[2];
herr_t status;

file_id = H5Fcreate("file.h5", H5F_ACC_TRUNC, H5P_DEFAULT, H5P_DEFAULT);
dims[0] = 4;
dims[1] = 6;
dataspace_id = H5Screate_simple(2, dims, NULL);

dataset_id = H5Dcreate(file_id,"A",H5T_STD_I32BE, dataspace_id, H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
status = H5Dclose(dataset_id);
status = H5Sclose(dataspace_id);
status = H5Fclose(file_id);
```

Terminate access to dataspace, dataset, file
Example Code - H5Dwrite

```c
status = H5Dwrite (dataset_id, H5T_NATIVE_INT, H5S_ALL, H5S_ALL, H5P_DEFAULT, wdata);
```

- Dataset ID from H5Dcreate/H5Dopen
- Memory Datatype
status = H5Dwrite (dataset_id, H5T_NATIVE_INT, H5S_ALL, H5S_ALL, H5P_DEFAULT, wdata);

To Modify Dataspace:
H5Sselect_hyperslab
H5Sselect_elements
Example Code – H5Dwrite

```c
status = H5Dwrite (dataset_id, H5T_NATIVE_INT, H5S_ALL, H5S_ALL, H5P_DEFAULT, wdata);
```

Data Transfer Property List
(MPI I/O, Transformations,...)
Example Code – H5Dread

```c
status = H5Dread (dataset_id, H5T_NATIVE_INT, H5S_ALL, H5S_ALL, H5P_DEFAULT, rdata);
```
#include "hdf5_hl.h"

file_id = H5Fcreate("file.h5", H5F_ACC_TRUNC, H5P_DEFAULT, H5P_DEFAULT);

status = H5LTmake_dataset (file_id, "A", 2, dims, H5T_STD_I32BE, data);

status = H5Fclose (file_id);
High Level APIs

- HDF5 Lite
- HDF5 Image
- HDF5 Table
- HDF5 Dimension Scales
- HDF5 Packet Table
Steps to Create a Group

1. Decide where to put it – “root group”

2. Define properties or use H5P_DEFAULT

5. Create group in file.

4. Close the group.
Example: Create a Group

Create a Group

```
A

"/" (root)

4x6 array of integers

file.h5

B
```
hid_t file_id, group_id;
...

/* Open "file.h5" */
file_id = H5Fopen ("file.h5", H5F_ACC_RDWR,
               H5P_DEFAULT);

/* Create group "/B" in file. */
group_id = H5Gcreate (file_id,"B", H5P_DEFAULT,
                  H5P_DEFAULT, H5P_DEFAULT);

/* Close group and file. */
status = H5Gclose (group_id);
status = H5Fclose (file_id);
HDF5 Tutorial:
http://www.hdfgroup.org/HDF5/Tutor/

HDF5 Example Code:
http://www.hdfgroup.org/ftp/HDF5/examples/examples-by-api/
HDF5 Technology Platform

- **HDF5 data model**
  - The “building blocks” for data organization and specification

- **HDF5 software**
  - Library, language interfaces, tools

- **HDF5 file format**
  - Bit-level organization of HDF5 file

Let’s look at ....
HDF5 File Format

• Defined by the *HDF5 File Format Specification*.  

• Specifies the bit-level organization of an HDF5 file on storage media.

• HDF5 library adheres to the File Format, so for the most part basic users do not need to know the guts of this information.
HDF5 Technology Platform

- **HDF5 data model**
  - The “building blocks” for data organization and specification

- **HDF5 software**
  - Library, language interfaces, tools

- **HDF5 file format**
  - Bit-level organization of HDF5 file
Thank You!
Questions/comments?