Introduction to HDF5
Outline

• What is HDF5?
• Introduction to HDF5 Data Model
• Introduction to HDF5 programming model and APIs
• Example HDF5 code
WHAT IS HDF5?
DATA MODEL
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 ...

- Hierarchical; collections of related information
- High-performance; compact; scalable
- Binary "Flat File"
- Databases
- Random access; subsetting
- PDF
- Standard exchange format; heterogeneous information
- 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
File
Link
Datatype
Dataspace

HDF5 Objects

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 datasets organize and contain “raw data values”.

- HDF5 dataspace describe the logical layout of the data elements.
HDF5 Datatypes

• Describe individual data elements in an HDF5 dataset
• Wide range of datatypes supported
  • Integer
  • Float
  • Unsigned
  • User-defined (e.g., 13-bit integer)
  • Variable length types (e.g., strings)
  • Compound (similar to C structs)
  • Many more ...
HDF5 Dataset

Datatype: 32-bit Integer

Dataspace: Rank = 2
Dimensions = 5 x 3
HDF5 Dataset with Compound Datatype

Dataspace:  

- Rank = 2
- Dimensions = 5 x 3
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 ✓
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

Experiment Notes:
Serial Number: 99378920
Date: 3/13/09
Configuration: Standard 3

<table>
<thead>
<tr>
<th>lat</th>
<th>lon</th>
<th>temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>23</td>
<td>3.1</td>
</tr>
<tr>
<td>15</td>
<td>24</td>
<td>4.2</td>
</tr>
<tr>
<td>17</td>
<td>21</td>
<td>3.6</td>
</tr>
</tbody>
</table>

Timestep 36,000
HDF5 Technology Platform

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

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

Let’s look at ....
HDF5 home page:  http://hdfgroup.org/HDF5/

- Latest release: HDF5 1.8.11 (Released May, 2013)

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

HDF5 Library

Storage

Domain Data Objects

Applications

EOS Application

EOS library

MATLAB
HDF-EOS5

• Data Model
  • Grid
  • Swath
  • Point

• Library
  • Implements HDF-EOS Data model in HDF5
  • Takes advantage of HDF5 chunking, compression, data organization
HDF-EOS5 File Organization
netCDF-4

• Data Model
  • Variable (HDF5 dataset)
  • Dimension Scale (HDF5 Dim. Scales)
  • Attributes (HDF5 attribute)
  • Group (HDF5 group)

• Library
  • Implements the model in many formats (netCDF 3.*, HDF4, CDM, including HDF5)
  • Takes advantage of HDF5 chunking, compression, data organization, parallel access
netCDF-4 Architecture

- **netCDF-3 applications**
- **netCDF-4 applications**
- **HDF5 applications**

**netCDF-4 Library**

**netCDF-3 Interface**

**HDF5 Library**

- netCDF files
- netCDF-4 HDF5 files
- HDF5 files
NetCDF Java

NetCDF-Java version 2.2 architecture

Scientific Datatypes
- Grid
- Station
- Image

NetcdfDataset

THREDDS
- Catalog.xml

OpenDAP

ADDE

NetcdfFile
- HDF5
- NetCDF-3
- NetCDF-4

I/O service provider
- GRIB
- GINI

NIDS
- Nexrad
- DMSP
INTRODUCTION TO HDF5 PROGRAMMING MODEL AND APIs
Useful Tools For New Users

h5cc, h5c++, h5fc:
  Scripts to compile applications
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
Show reference manual on the web...
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

H5Fcreate (H5Fopen)  create (open) File

H5Screate_simple/H5Screate  create dataSpace

H5Dcreate (H5Dopen)  create (open) Dataset

H5Dread, H5Dwrite  access Dataset

H5Dclose  close Dataset

H5Sclose  close dataSpace

H5Fclose  close File
Other Common Functions

**DataSpaces:**
- H5Sselect_hyperslab (Partial I/O)
- H5Sselect_elements (Partial I/O)
- H5Dget_space

**Groups:**
- H5Gcreate, H5Gopen, H5Gclose

**Attributes:**
- H5Acreate, H5Aopen_name, H5Aclose, H5Aread, H5Awrite

**Property lists:**
- H5Pcreate, H5Pclose
- H5Pset_chunk, H5Pset_deflate
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);

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
hid_t dataspace_id;
hsize_t dims[2];

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);
```

**Define a dataspace**

- **rank**: current dims
- **current dims**: max sizes = current sizes
- **NULL means**: current sizes

9/21/15
1 hid_t file_id, dataset_id, dataspace_id;

.  
.  
.

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);

8 dataset_id = H5Dcreate(file_id,"foo",H5T_STD_I32BE,
                           dataspace_id, H5P_DEFAULT,H5P_DEFAULT,
                           H5P_DEFAULT);

Datatype Properties (Link Creation, Dataset Creation and Access)

Where to put it
Size & shape
Name
Datatype
Properties (Link Creation, Dataset Creation and Access)
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
- Buffer containing your data
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,...)
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, "foo", 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

3. Create group in file.

4. Close the group.
Example: Create a Group

```
Create a Group
```

```
A
```

```
B
```

```
4x6 array of integers
```

```
"/" (root)
```

```
file.h5
```

```
9/21/15
```
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

Recall...
This work was supported by SGT under Prime Contract No. NNG12CR31C, funded by NASA.

Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of SGT or NASA.
Thank You!

Questions?