

Introduction to HDF5 Data Model, Programming Model and Library APIs

**HDF and HDF-EOS Workshop IX
November 30, 2005**

Goals

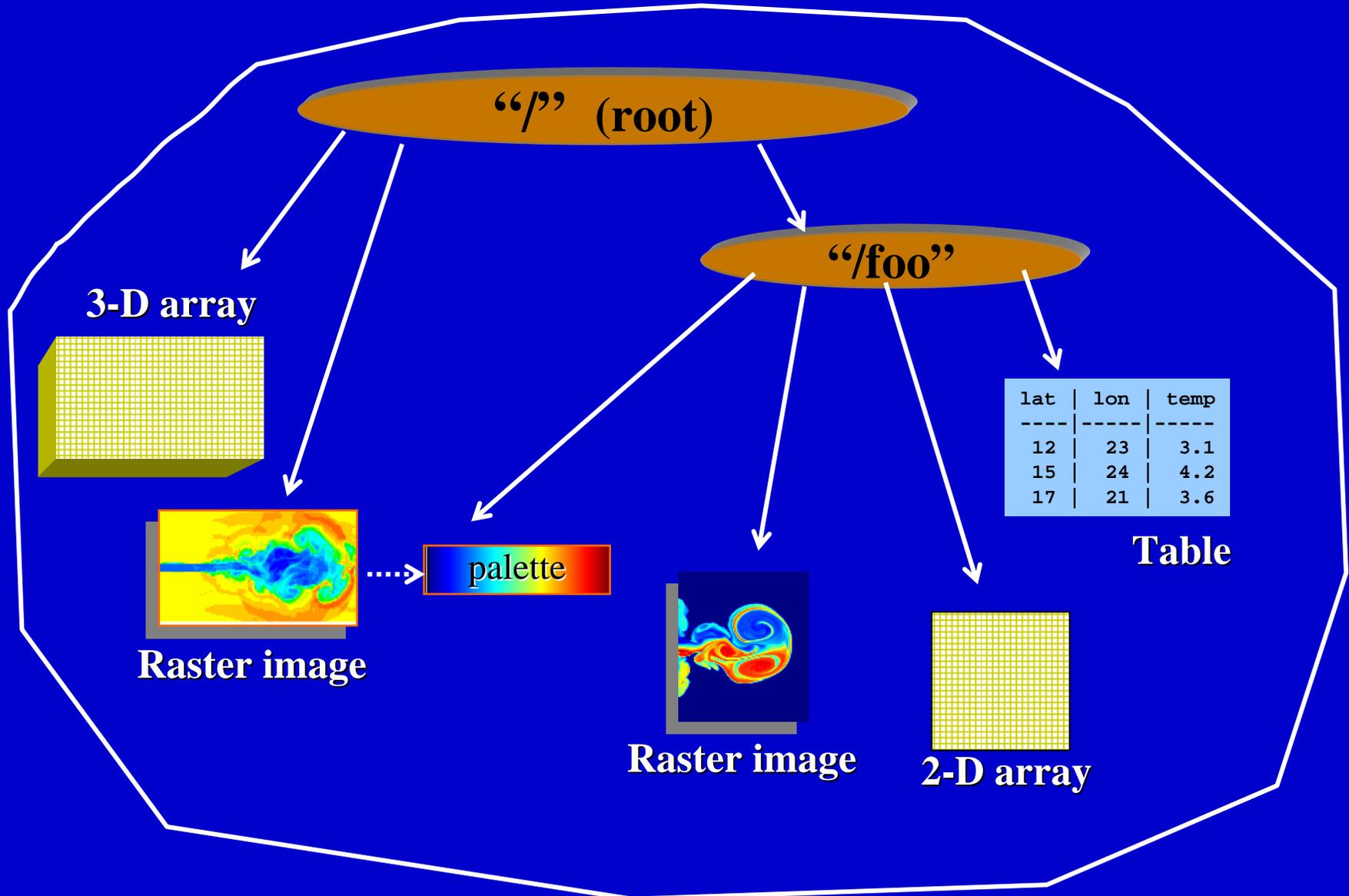
- **Introduce HDF5**
- **Provide a basic knowledge of how data can be organized in HDF5 & how it is used by applications.**
- **To provide some examples of how to read and write HDF5 files**

What is HDF5?

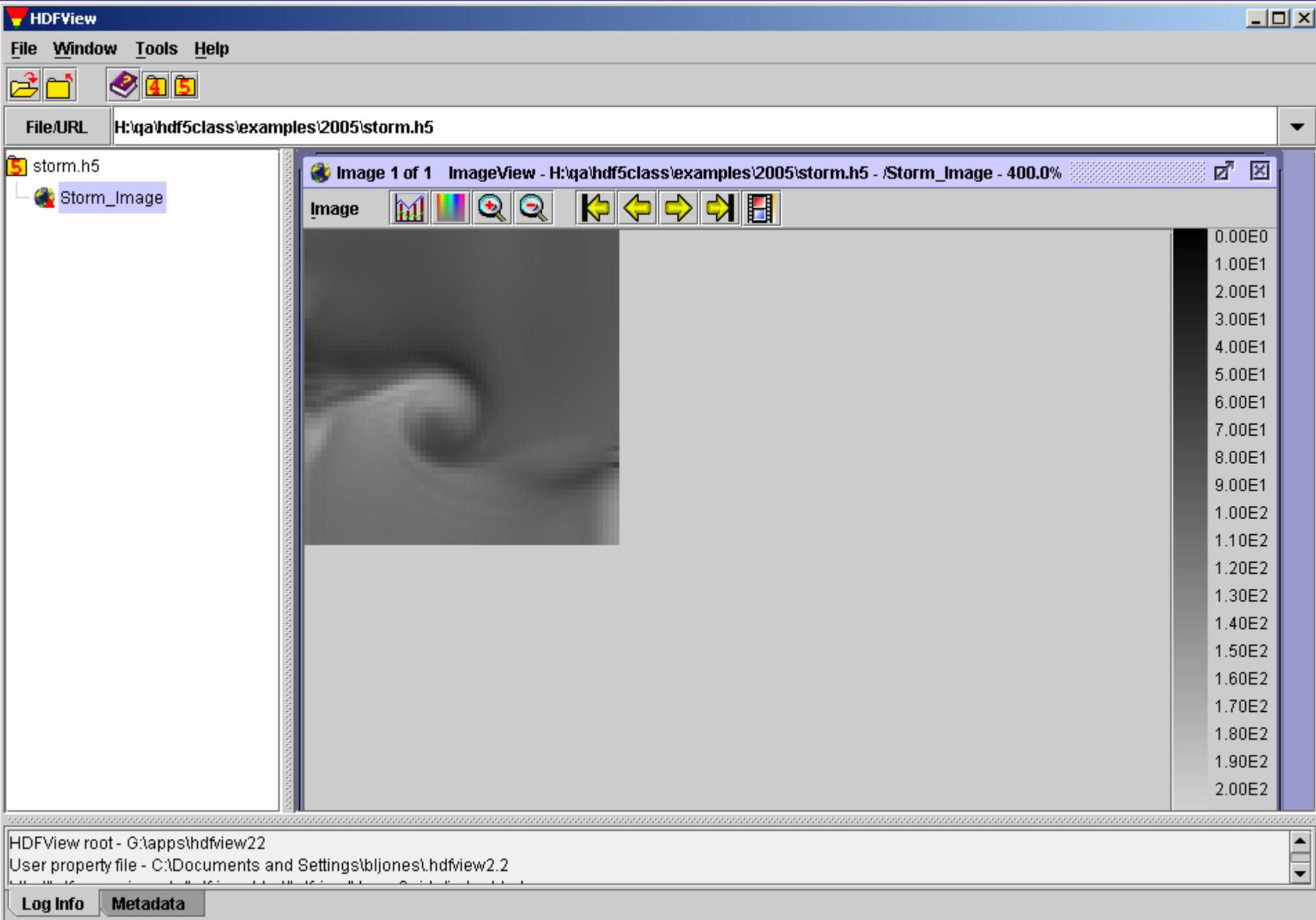
What is HDF5?

- **File format for storing scientific data**
 - To store and organize all kinds of data
 - To share data , to port files from one platform to another
 - To overcome a limit on number and size of the objects in the file
- **Software for accessing scientific data**
 - Flexible I/O library (parallel, remote, etc.)
 - Efficient storage
 - Available on almost all platforms
 - C, F90, C++ , Java APIs
 - Tools (HDFView, utilities)

Example HDF5 file



Viewing an HDF5 File with HDFView



Example HDF5 Application

```
#include<stdio.h>
#include "H5IM.h"

#define WIDTH  57                /* dataset dimensions */
#define HEIGHT 57
#define RANK  2

int main (void)
{
    hid_t    file;                /* file handle */
    herr_t   status;
    unsigned char data[WIDTH][HEIGHT]; /* data to write */
    int      i, j, num, val;
    FILE *fp;

    fp = fopen ("storm110.txt", "r"); /* Open ASCII file */

    for (i=0; i<WIDTH; i++)          /* Read Values into 'data' buffer */
        for (j=0; j<HEIGHT; j++) {
            num = fscanf (fp, "%d ", &val);
            data[i][j] = val;
        }

    file = H5Fcreate ("storm.h5", H5F_ACC_TRUNC, H5P_DEFAULT, H5P_DEFAULT); /* Create File */

    status = H5IMmake_image_8bit (file, "Storm_Image", WIDTH, HEIGHT, /* Create Image */
                                  (const unsigned char *)data);

    status = H5Fclose (file);        /* Close File */
}
```

HDF5 Data Model

HDF5 file

- **Primary Objects**
 - Groups
 - Datasets
- **Additional means to organize data**
 - Attributes
 - Sharable objects
 - Storage and access properties

HDF5 Dataset

- **Data array**
 - ordered collection of identically typed data items distinguished by their indices
- **Metadata**
 - Dataspace – rank, dimensions, other spatial info about dataset
 - Datatype
 - Attribute list – user-defined metadata
 - Special storage options – how array is organized

Dataset Components

Metadata

Dataspace

Rank Dimensions

3

Dim_1 = 4

Dim_2 = 5

Dim_3 = 7

Datatype

IEEE 32-bit float

Storage info

Chunked

Compressed

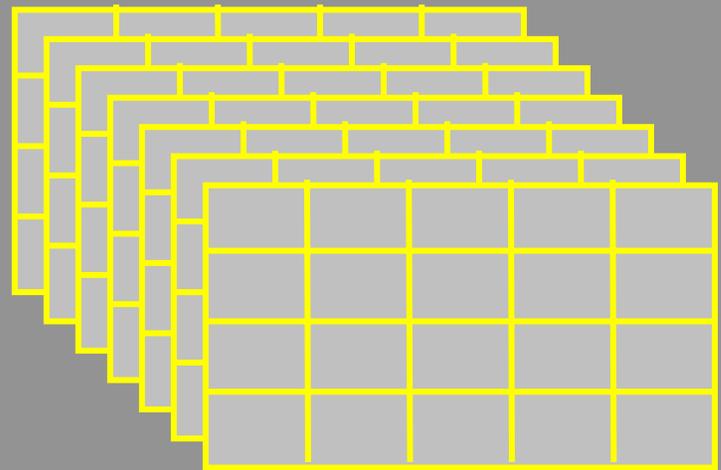
Attributes

Time = 32.4

Pressure = 987

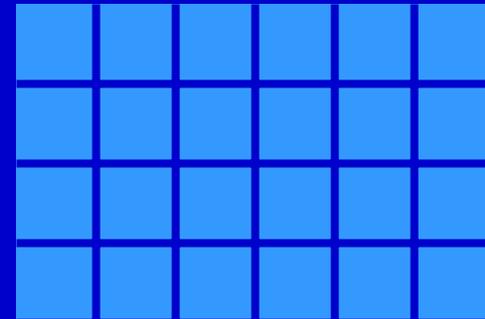
Temp = 56

Data



Dataspaces

- **Dataspace – spatial info about a dataset**
 - Rank and dimensions
 - Permanent part of dataset definition
 - Subset of points, for partial I/O
 - Needed only during I/O operations
- **Apply to datasets in memory or in the file**



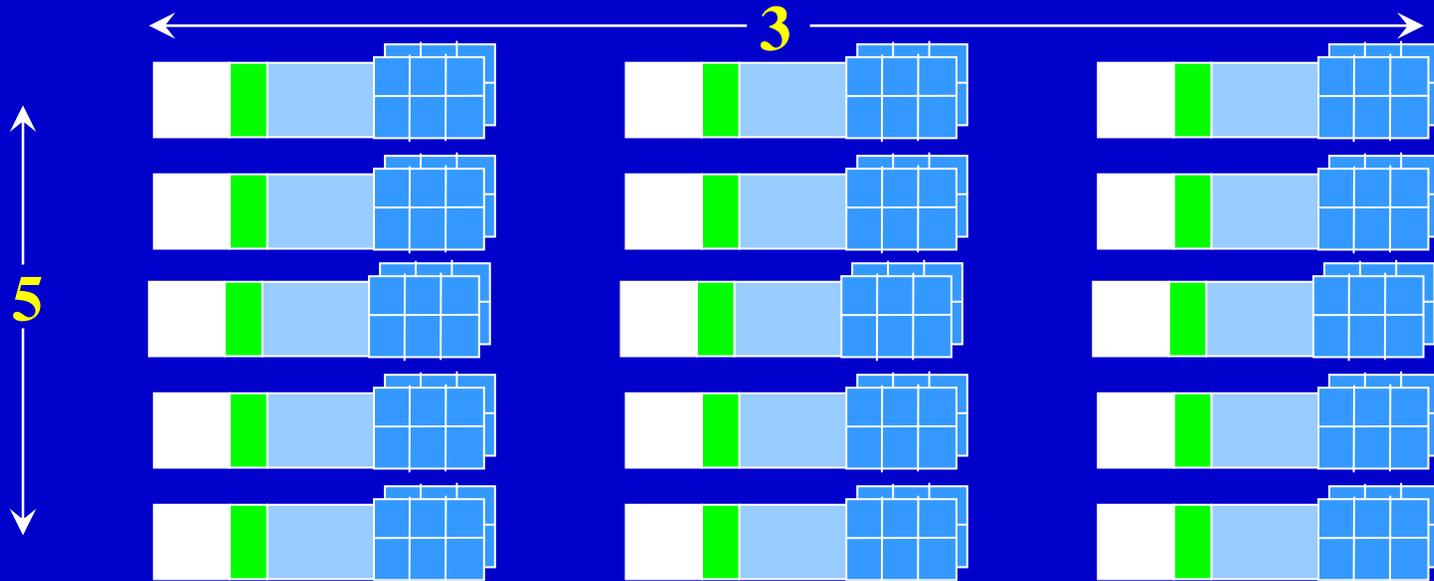
Rank = 2

Dimensions = 4x6

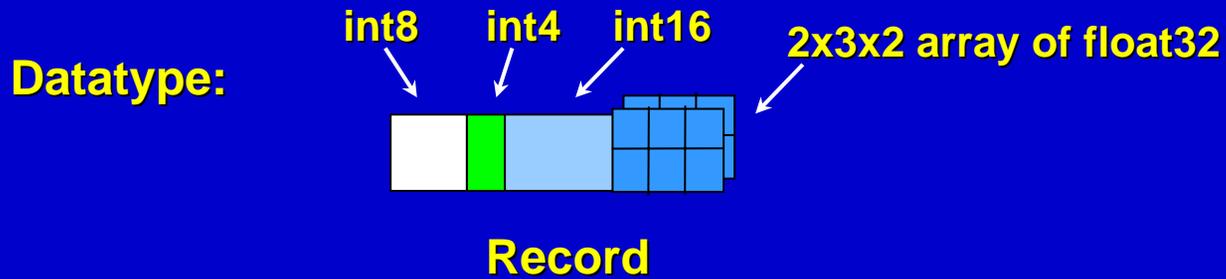
Datatypes (array elements)

- **Datatype – how to interpret a data element**
 - Permanent part of the dataset definition
- **HDF5 atomic types**
 - normal integer & float
 - user-definable (e.g. 13-bit integer)
 - variable length types (e.g. strings)
 - pointers - references to objects/dataset regions
 - enumeration - names mapped to integers
 - array
- **HDF5 compound types**
 - Comparable to C structs
 - Members can be atomic or compound types

HDF5 dataset: array of records



Dimensionality: 5 x 3

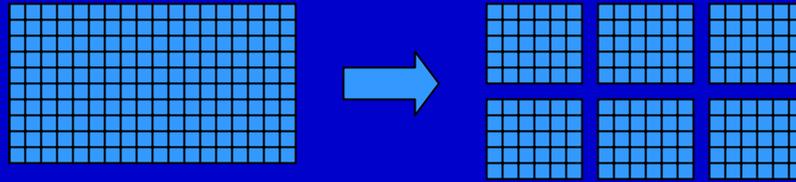


Attributes

- **Attribute** – data of the form “name = value”, attached to an object
- **Operations are scaled-down versions of the dataset operations**
 - Not extendible
 - No compression
 - No partial I/O
- **Optional for the dataset definition**
- **Can be overwritten, deleted, added during the “life” of a dataset**

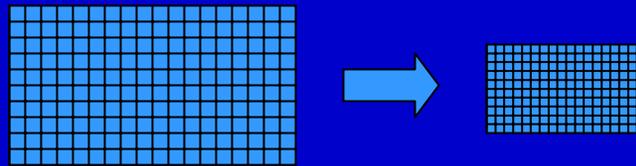
Special Storage Options

chunked



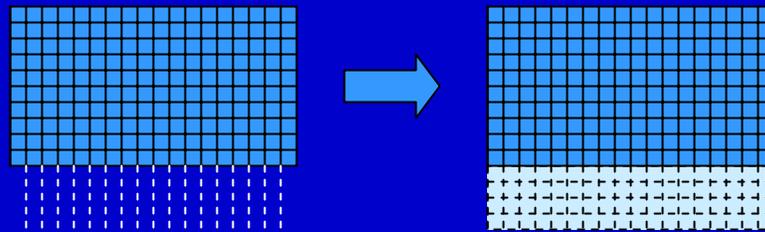
**Better subsetting
access time;
extendible**

compressed



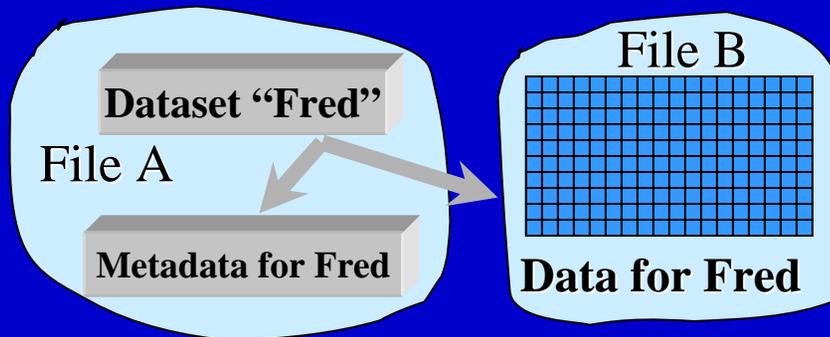
**Improves storage
efficiency,
transmission speed**

extendible



**Arrays can be
extended in any
direction**

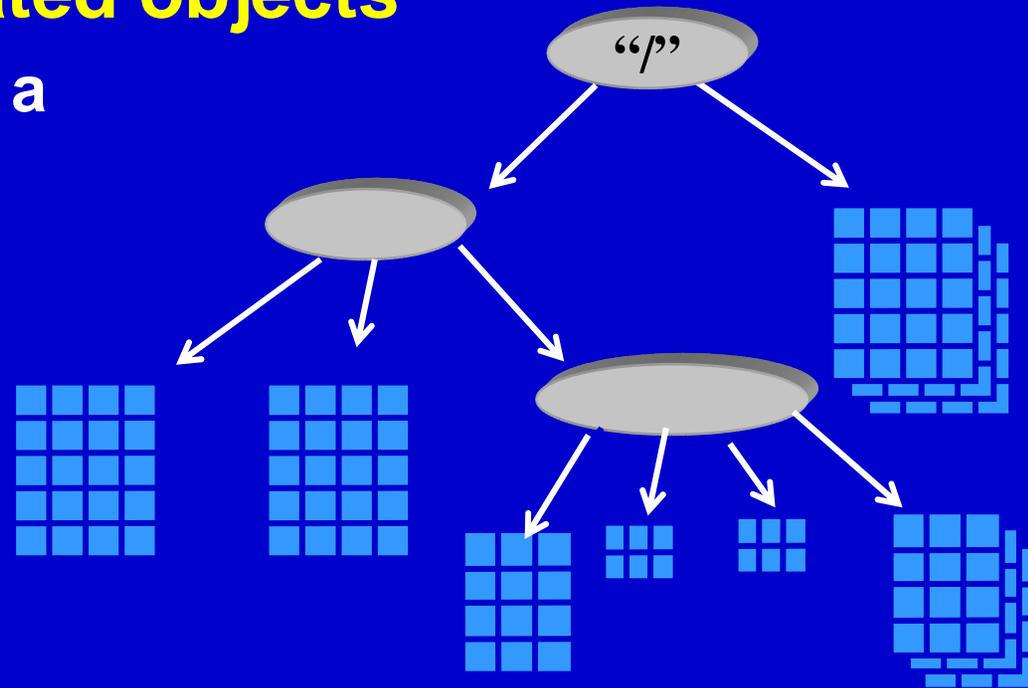
**External
file**



**Metadata in one file,
raw data in another.**

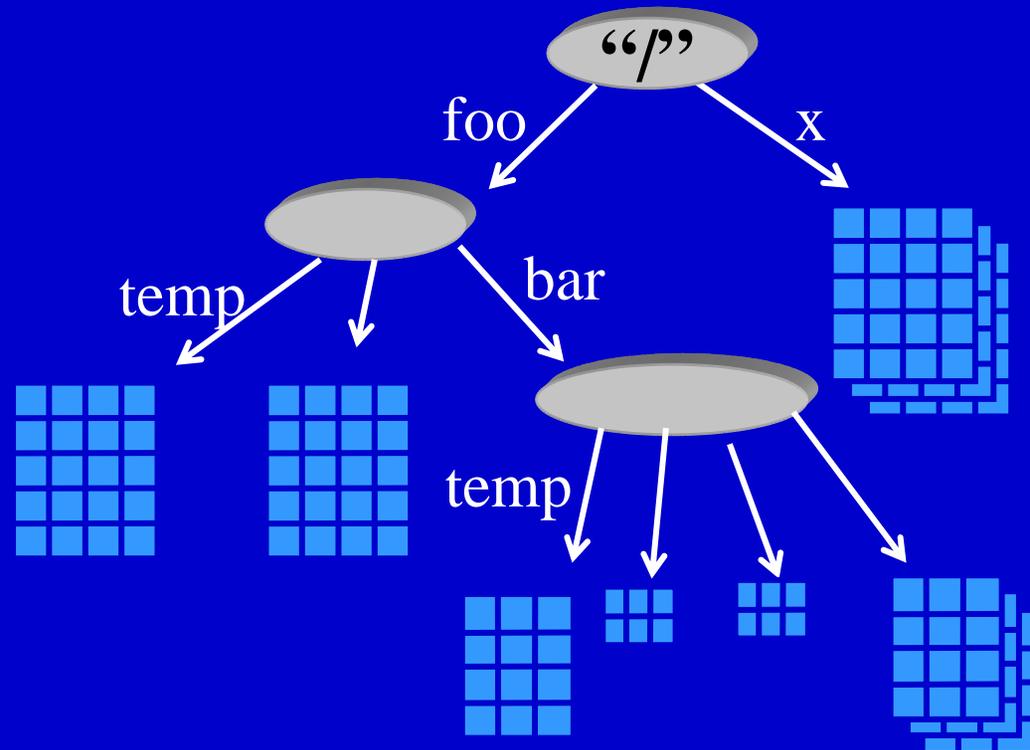
Groups

- **Group** – a mechanism for describing collections of related objects
- Every file starts with a root group
- Can have attributes
- Similar to UNIX directories, but cycles are allowed

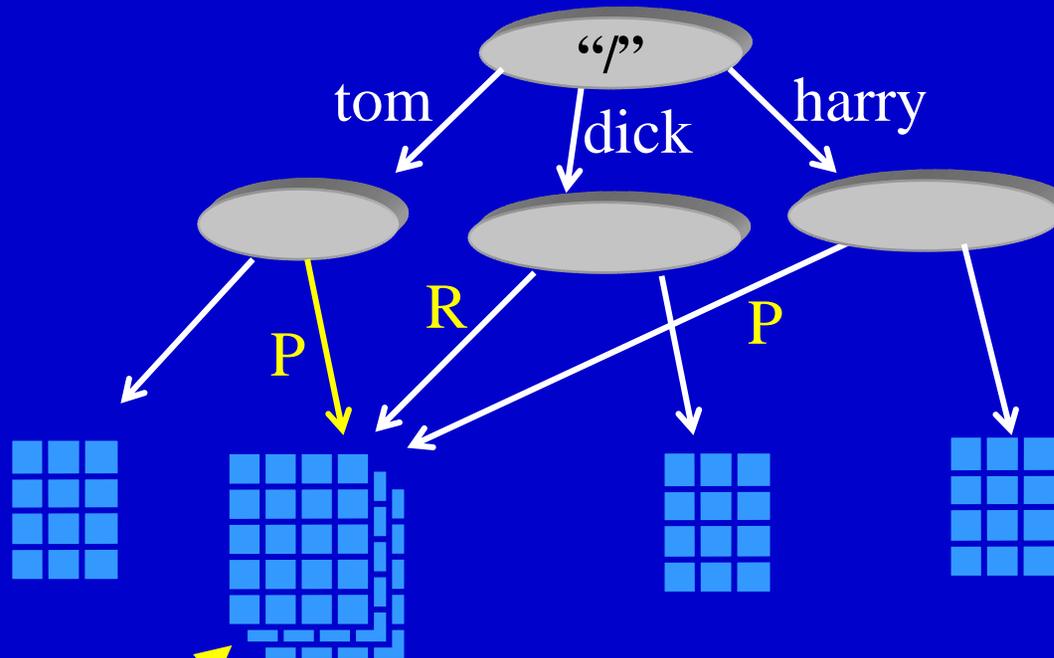


HDF5 objects are identified and located by their pathnames

/ (root)
/x
/foo
/foo/temp
/foo/bar/temp



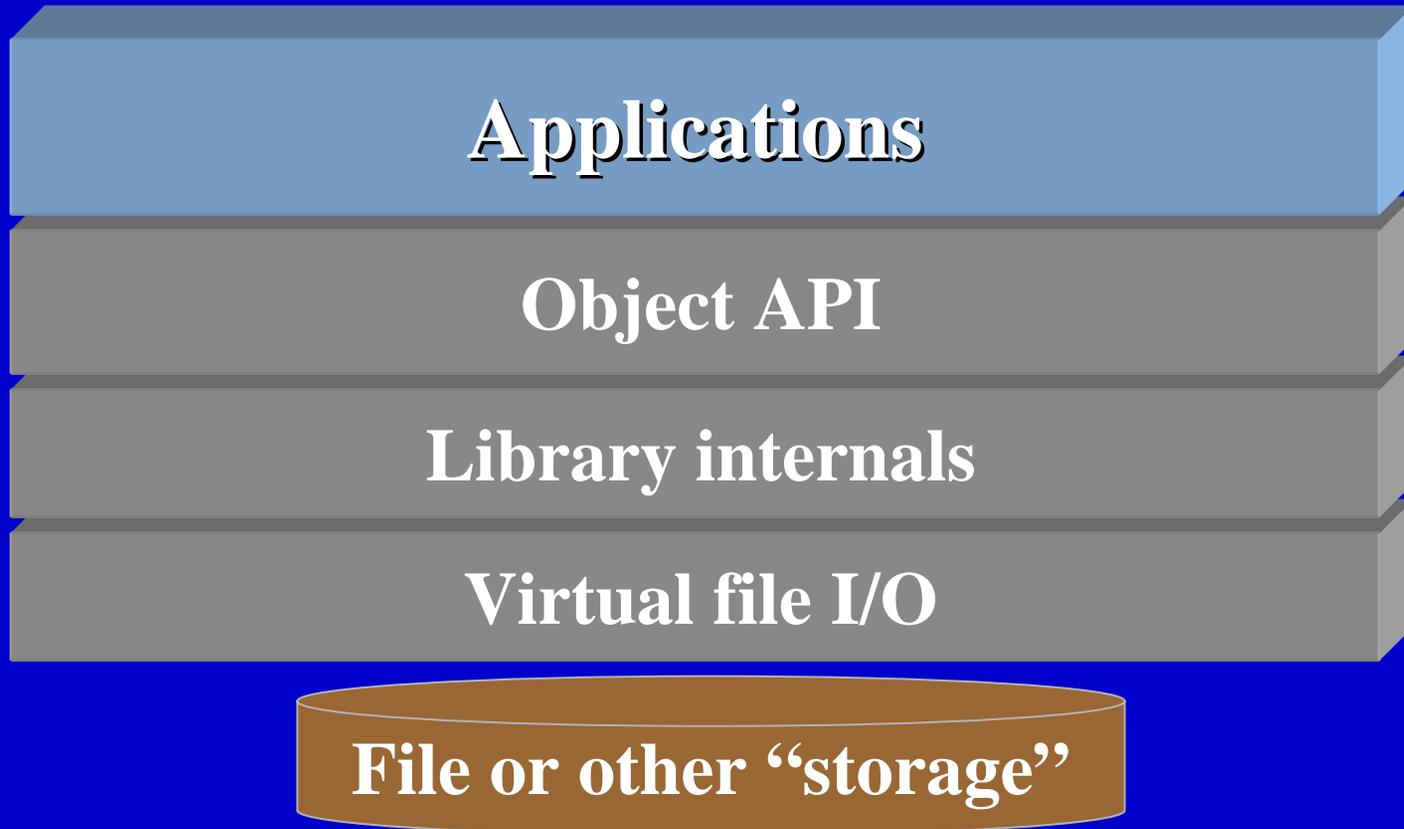
Groups & members of groups can be shared



/tom/P
/dick/R
/harry/P

HDF5 I/O Library

Structure of HDF5 Library



Structure of HDF5 Library

Object API (C, Fortran 90, Java, C++)

- Specify objects and transformation and storage properties
- Invoke data movement operations and data transformations



Library internals (C)

- Performs data transformations and other prep for I/O
- Configurable transformations (compression, etc.)

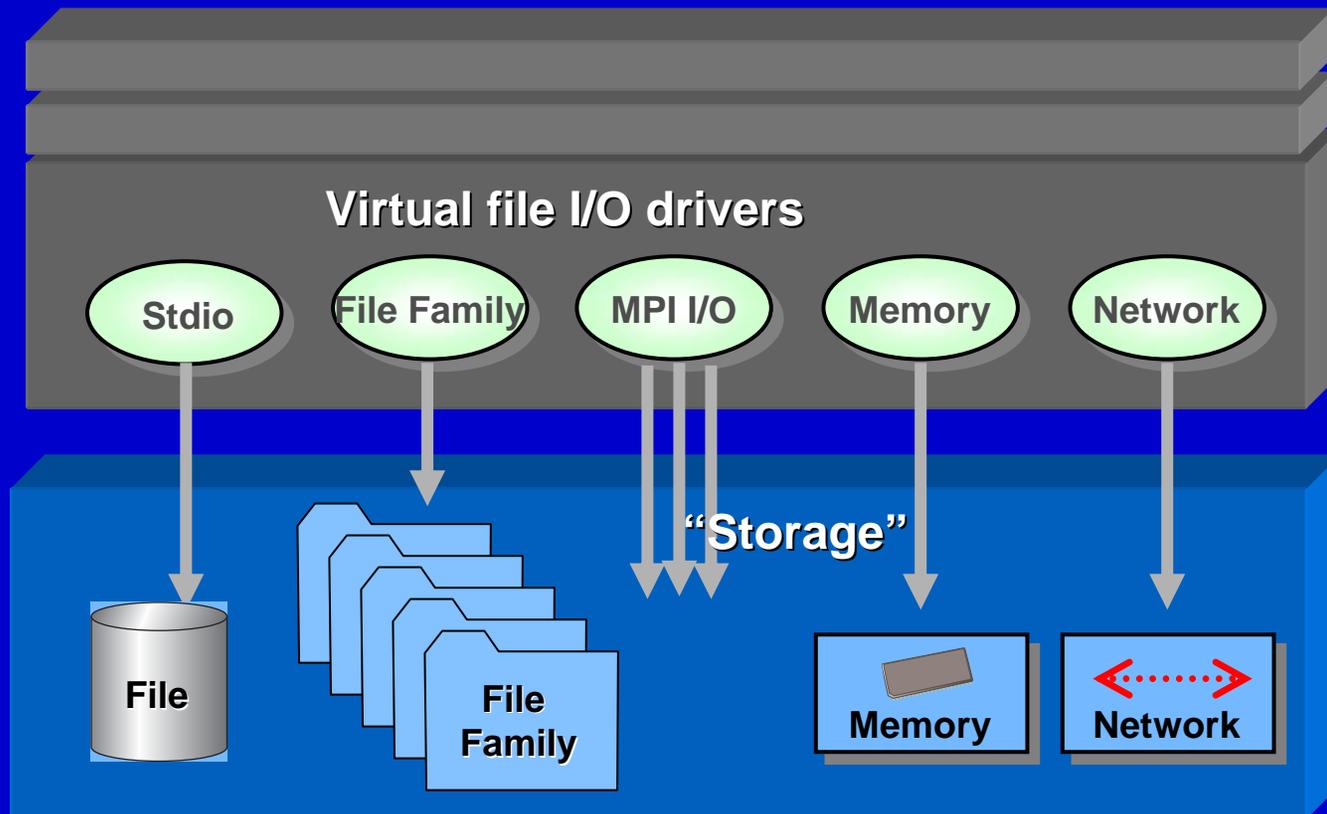


Virtual file I/O (C only)

- Perform byte-stream I/O operations (open/close, read/write, seek)
- User-implementable I/O (stdio, network, memory, etc.)

Virtual File I/O Layer

- A public API for writing I/O drivers
- Allows HDF5 to interface to disk, the network, memory, or a user-defined device



Intro to HDF5 API

Programming model for sequential access

Goals

- Describe the HDF5 programming model
- Give a feel for what it's like to use the general HDF5 API
- Review some of the key concepts of HDF5

General API Topics

- **General info about HDF5 programming**
- **Creating an HDF5 file**
- **Creating a dataset**
- **Writing and reading a dataset**

The General HDF5 API

- **Currently has C, Fortran 90, Java and C++ bindings.**
- **C routines begin with prefix H5*, where * is a single letter indicating the object on which the operation is to be performed.**
- **Full functionality**

Example APIs:

H5D : Dataset interface *e.g.*.. **H5Dread**

H5F : File interface *e.g.*.. **H5Fopen**

H5S : dataSpace interface *e.g.*.. **H5Sclose**

The General Paradigm

- *Properties (called creation and access property lists) of objects are defined (optional)*
- **Objects are opened or created**
- **Objects then accessed**
- **Objects finally closed**

Order of Operations

- **The library imposes an order on the operations by argument dependencies**
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, and reusing a closed object will result in an error**

HDF5 C Programming Issues

For portability, HDF5 library has its own defined types:

hid_t:	object identifiers (native <i>integer</i>)
hsize_t:	size used for dimensions (<i>unsigned long</i> or <i>unsigned long long</i>)
hssize_t:	for specifying coordinates and sometimes for dimensions (<i>signed long</i> or <i>signed long long</i>)
herr_t:	function return value
hvl_t:	variable length datatype

For **C**, include `#include hdf5.h` at the top of your HDF5 application.

h5dump

Command-line Utility for Viewing HDF5 Files

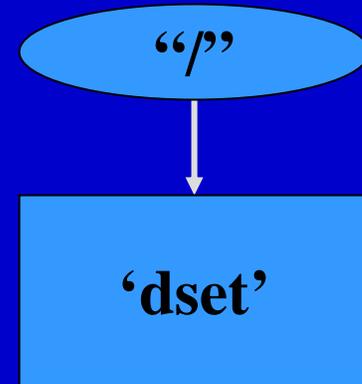
```
h5dump [--header] [-a ] [-d <names>] [-g <names>]
      [-l <names>] [-t <names>] <file>
```

- header** Display header only; no data is displayed.
- a <names>** Display the specified attribute(s).
- d <names>** Display the specified dataset(s).
- g <names>** Display the specified group(s) and all the members.
- l <names>** Displays the value(s) of the specified soft link(s).
- t <names>** Display the specified named datatype(s).

<names> is one or more appropriate object names.

Example of h5dump Output

```
HDF5 "dset.h5" {
GROUP "/" {
  DATASET "dset" {
    DATATYPE { H5T_STD_I32BE }
    DATASPACE { SIMPLE ( 4, 6 ) / ( 4, 6 ) }
    DATA {
      1, 2, 3, 4, 5, 6,
      7, 8, 9, 10, 11, 12,
      13, 14, 15, 16, 17, 18,
      19, 20, 21, 22, 23, 24
    }
  }
}
}
```



Creating an HDF5 File

Steps to Create a File

- 1. Specify File Creation and Access Property Lists, if necessary**
- 2. Create a file**
- 3. Close the file and the property lists, if necessary**

Property Lists

- **A property list is a collection of values that can be passed to HDF5 functions at lower layers of the library**
- **File Creation Property List**
 - Controls file metadata
 - Size of the user-block, sizes of file data structures, etc.
 - Specifying H5P_DEFAULT uses the default values
- **Access Property List**
 - Controls different methods of performing I/O on files
 - Unbuffered I/O, parallel I/O, etc.
 - Specifying H5P_DEFAULT uses the default values.

**hid_t H5Fcreate (const char *name, unsigned flags,
hid_t create_id, hid_t access_id)**

name	IN:	Name of the file to access
flags	IN:	File access flags
create_id	IN:	File creation property list identifier
access_id	IN:	File access property list identifier

herr_t H5Fclose (hid_t file_id)

file_id **IN:** Identifier of the file to terminate access to

Example 1

Create a new file using default properties



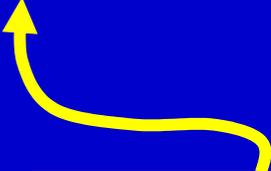
```
1 hid_t file_id;  
2 herr_t status;  
3 file_id = H5Fcreate ("file.h5", H5F_ACC_TRUNC,  
4                    H5P_DEFAULT, H5P_DEFAULT);  
5  
6 status = H5Fclose (file_id);
```

Example 1

```
1 hid_t      file_id;
2 herr_t     status;

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

4 status = H5Fclose (file_id);
```



Terminate access to
the File

h5_crtfile.c

```
1  #include <hdf5.h>
2  #define FILE "file.h5"
3
4  main() {
5
6      hid_t      file_id;    /* file identifier */
7      herr_t     status;
8
9      /* Create a new file using default properties. */
10     file_id = H5Fcreate (FILE, H5F_ACC_TRUNC,
11                          H5P_DEFAULT, H5P_DEFAULT);
12
13     /* Terminate access to the file. */
14     status = H5Fclose (file_id);
15 }
```

Example 1: h5dump Output

```
HDF5 "file.h5" {  
GROUP "/" {  
}  
}
```



'/'

Create a Dataset

Dataset Components

Metadata

Dataspace

Rank Dimensions

3

Dim_1 = 4

Dim_2 = 5

Dim_3 = 7

Datatype

IEEE 32-bit float

Attributes

Time = 32.4

Pressure = 987

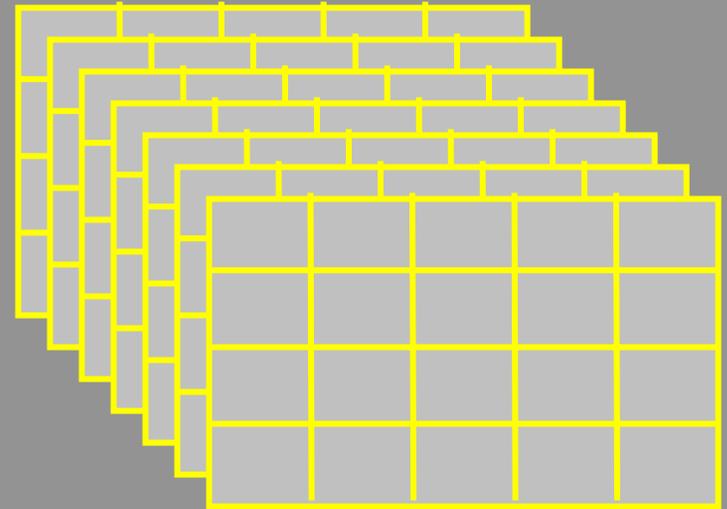
Temp = 56

Storage info

Chunked

Compressed

Data



Steps to Create a Dataset

1. Obtain location ID where dataset is to be created
2. Define dataset characteristics (datatype, dataspace, dataset creation property list, if necessary)
3. Create the dataset
4. Close the datatype, dataspace, and property list, if necessary
5. Close the dataset

Step 1

Step 1. Obtain the *location identifier* where the *dataset* is to be created

Location Identifier: the file or group identifier in which to create a dataset

Step 2

Step 2. Define the dataset characteristics

- datatype (e.g. integer)
- dataspace (2 dimensions: 100x200)
- dataset creation properties (e.g. chunked and compressed)

Standard Predefined Datatypes

Examples:

H5T_IEEE_F64LE Eight-byte, little-endian, IEEE floating-point

H5T_IEEE_F32BE Four-byte, big-endian, IEEE floating point

H5T_STD_I32LE Four-byte, little-endian, signed two's complement integer

H5T_STD_U16BE Two-byte, big-endian, unsigned integer

NOTE:

- These datatypes (DT) are the same on all platforms
- These are DT handles generated at run-time

Standard Predefined Datatypes

Examples:

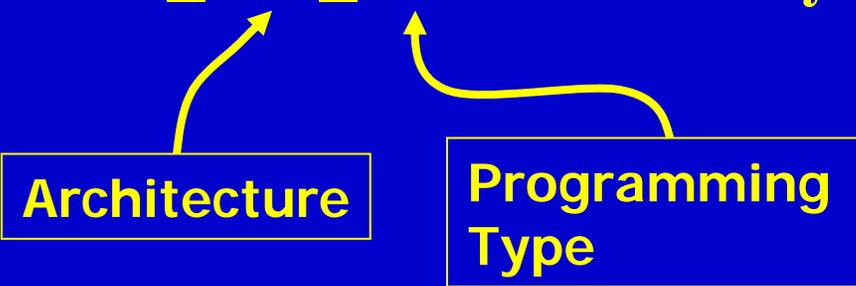
H5T_IEEE_F64LE Eight-byte, little-endian, IEEE floating-point

H5T_IEEE_F32BE Four-byte, big-endian, IEEE floating point

H5T_STD_I32LE Four-byte, little-endian, signed two's complement integer

H5T_STD_U16BE **Two-byte, big-endian, unsigned integer**

Architecture



```
graph TD; A[Architecture] --> H5T[H5T_STD_U16BE]; B[Programming Type] --> H5T;
```

Programming
Type

Native Predefined Datatypes

Examples of predefined native types in C:

<code>H5T_NATIVE_INT</code>	(int)
<code>H5T_NATIVE_FLOAT</code>	(float)
<code>H5T_NATIVE_UINT</code>	(unsigned int)
<code>H5T_NATIVE_LONG</code>	(long)
<code>H5T_NATIVE_CHAR</code>	(char)

NOTE:

- These datatypes are **NOT** the same on all platforms
- These are DT handles generated at run-time

Dataspaces

- **Dataspace: size and shape of dataset and subset**
 - Dataset
 - Rank: number of dimension
 - Dimensions: sizes of all dimensions
 - Permanent – part of dataset definition
 - Subset
 - Size, shape and position of selected elements
 - Needed primarily during I/O operations
 - Not permanent
 - (Subsetting not covered in this tutorial)
- **Applies to arrays in memory or in the file**

Creating a Simple Dataspace

```
hid_t H5Screate_simple (int rank,  
                        const hsize_t * dims,  
                        const hsize_t * maxdims)
```

rank **IN:** Number of dimensions of dataspace

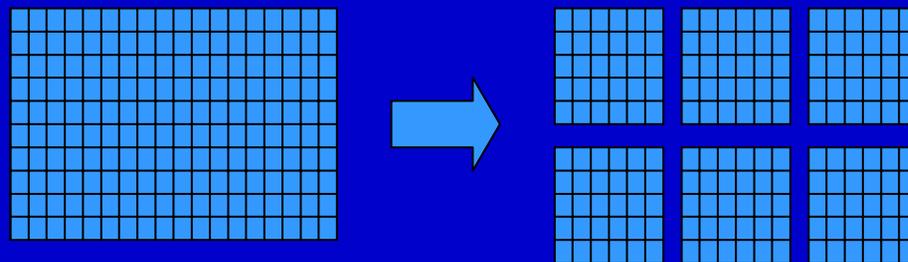
dims **IN:** An array of the size of each dimension

maxdims **IN:** An array of the maximum size of each dimension
A value of **H5S_UNLIMITED** specifies the
unlimited dimension.
A value of **NULL** specifies that *dims* and *maxdims*
are the same.

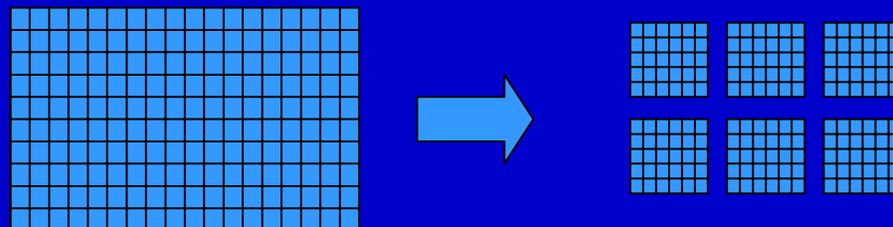
Dataset Creation Property List

The *dataset creation property list* contains information on how to organize data in storage.

Chunked



Chunked &
compressed



Property List Example

- **Creating a dataset with "deflate" compression**

```
create_plist_id = H5Pcreate(H5P_DATASET_CREATE);  
H5Pset_chunk(create_plist_id, ndims, chunk_dims);  
H5Pset_deflate(create_plist_id, 9);
```

Remaining Steps to Create a Dataset

3. **Create the dataset**
4. **Close the datatype, dataspace, and property list, if necessary**
5. **Close the dataset**

**hid_t H5Dcreate (hid_t loc_id, const char *name,
hid_t type_id, hid_t space_id,
hid_t create_plist_id)**

loc_id **IN:** Identifier of file or group to create the dataset within

name **IN:** The name of (the link to) the dataset to create

type_id **IN:** Identifier of datatype to use when creating the dataset

space_id **IN:** Identifier of dataspace to use when creating the dataset

create_plist_id **IN:** Identifier of the dataset creation property list (or **H5P_DEFAULT**)

Example 2 – Create an empty 4x6 dataset

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

Create a new file

```
4 file_id = H5Fcreate ("dset.h5", H5F_ACC_TRUNC,  
                      H5P_DEFAULT, H5P_DEFAULT);  
  
5 dims[0] = 4;  
6 dims[1] = 6;  
7 dataspace_id = H5Screate_simple (2, dims, NULL);  
  
8 dataset_id = H5Dcreate(file_id,"dset",H5T_STD_I32BE,  
                        dataspace_id, H5P_DEFAULT);  
  
9 status = H5Dclose (dataset_id);  
10 status = H5Sclose (dataspace_id);  
11 status = H5Fclose (file_id);
```

Example 2 – Create an empty 4x6 dataset

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

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

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

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

9 status = H5Dclose (dataset_id);
10 status = H5Sclose (dataspace_id);
11 status = H5Fclose (file_id);
```

Create a dataspace

rank

current dims

Set maxdims to current dims

Example 2 – Create an empty 4x6 dataset

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

```
4 file_id = H5Fcreate ("dset.h5", H5F_ACC_TRUNC,  
                      H5P_DEFAULT, H5P_DEFAULT);
```

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

Pathname

Datatype

Create a dataset

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

Dataspace

Property list
(default)

```
9 status = H5Dclose (dataset_id);  
10 status = H5Sclose (dataspace_id);  
11 status = H5Fclose (file_id);
```

Example 2 – Create an empty 4x6 dataset

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

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

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

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

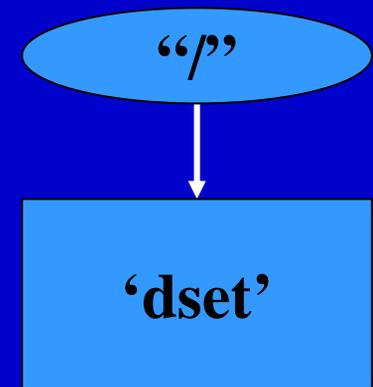
Terminate access to dataset, dataspace, & file

```
9 status = H5Dclose (dataset_id);
10 status = H5Sclose (dataspace_id);
11 status = H5Fclose (file_id);
```

Example2: h5dump Output

An empty 4x6 dataset

```
HDF5 "dset.h5" {
GROUP "/" {
  DATASET "dset" {
    DATATYPE { H5T_STD_I32BE }
    DATASPACE { SIMPLE ( 4, 6 ) / ( 4, 6 ) }
    DATA {
      0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0
    }
  }
}
}
```



Writing and Reading Datasets

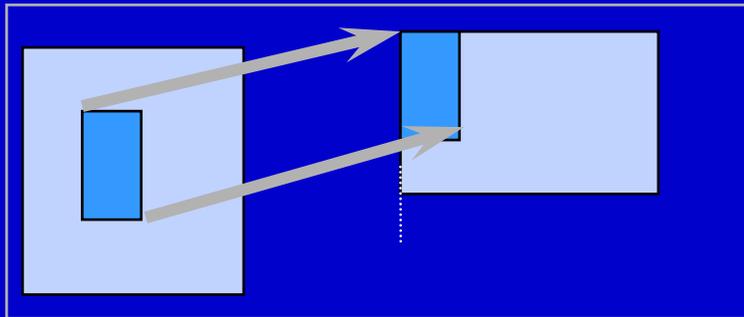
Dataset I/O

- **Dataset I/O involves**
 - reading or writing
 - all or part of a dataset
 - Compressed/uncompressed
- **During I/O operations data is translated between the source & destination (file-memory, memory-file)**
 - Datatype conversion
 - data types (e.g. 16-bit integer => 32-bit integer) of the same class
 - Dataspace conversion
 - dataspace (e.g. 10x20 2d array => 200 1d array)

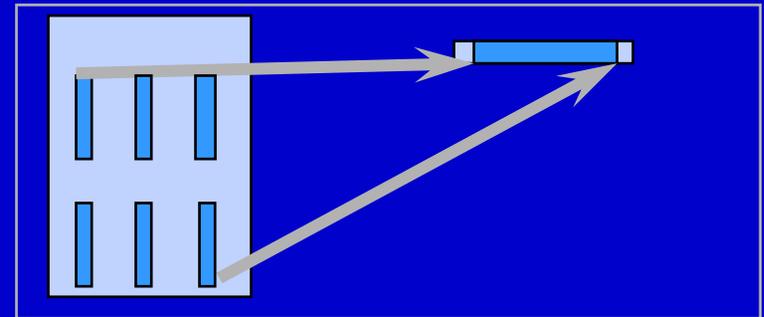
Partial I/O

- **Selected elements (called selections) from source are mapped (read/written) to the selected elements in destination**
- **Selection**
 - Selections in memory can differ from selection in file
 - Number of selected elements is always the same in source and destination
- **Selection can be**
 - Hyperslabs (contiguous blocks, regularly spaced blocks)
 - Points
 - Results of set operations (union, difference, etc.) on hyperslabs or points

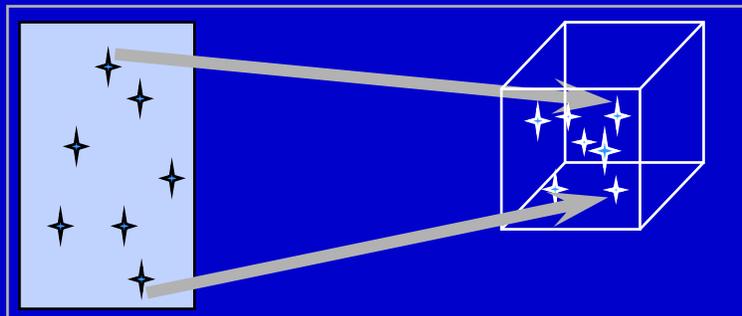
Sample Mappings between File Dataspaces and Memory Dataspaces



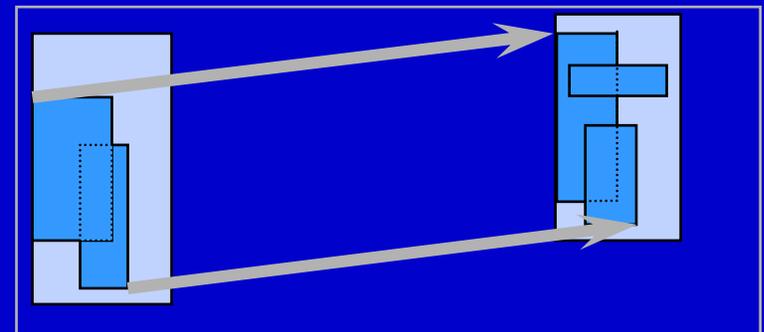
(a) Hyperslab from a 2D array to the corner of a smaller 2D array



(b) Regular series of blocks from a 2D array to a contiguous sequence at a certain offset in a 1D array



(c) A sequence of points from a 2D array to a sequence of points in a 3D array.

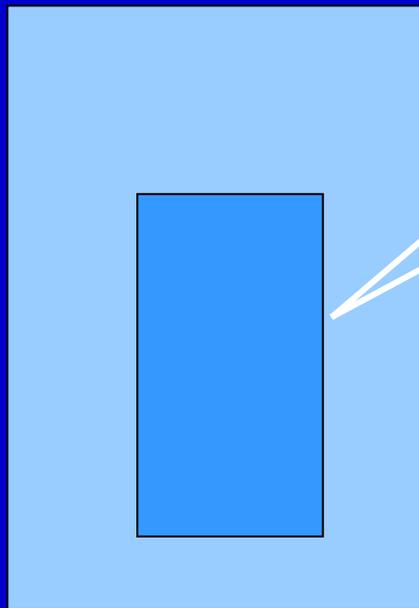


(d) Union of hyperslabs in file to union of hyperslabs in memory.

Reading Dataset into Memory from File

File

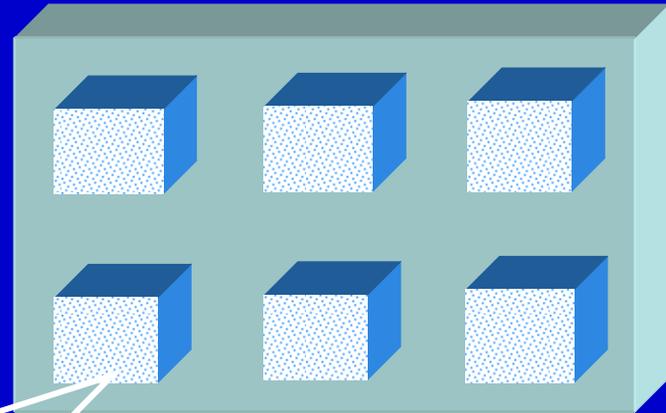
2D array of 16-bit ints



2-d array

Memory

3D array of 32-bit ints



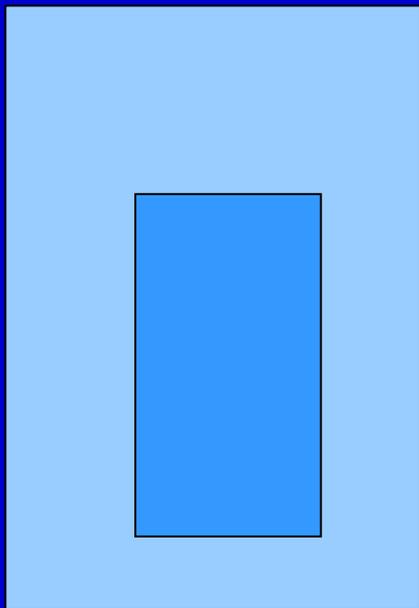
Regularly spaced series of cubes

The only restriction is that the number of selected elements on the left be the same as on the right.

Reading Dataset into Memory from File

File

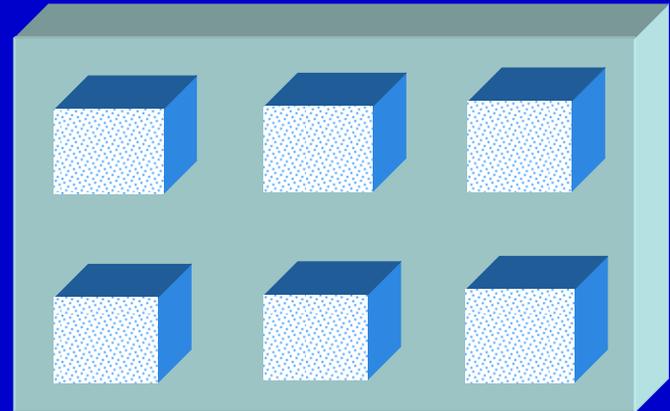
2D array of 16-bit ints



Read

Memory

3D array of 32-bit ints



Steps for Dataset Writing/Reading

1. **If necessary, open the file to obtain the file ID**
2. **Open the dataset to obtain the dataset ID**
3. **Specify**
 - Memory datatype
 - *! Library “knows” file datatype – do not need to specify!*
 - Memory dataspace
 - File dataspace
 - Transfer properties (optional)
4. **Perform the desired operation on the dataset**
5. **Close dataspace, datatype and property lists**

Data Transfer Property List

The data transfer property list is used to control various aspects of the I/O, such as caching hints or collective I/O information.

hid_t H5Dopen (hid_t loc_id, const char *name)

loc_id	IN:	Identifier of the file or group in which to open a dataset
name	IN:	The name of the dataset to access

NOTE: File datatype and dataspace are known when a dataset is opened

**herr_t H5Dwrite (hid_t dataset_id, hid_t mem_type_id,
hid_t mem_space_id, hid_t file_space_id,
hid_t xfer_plist_id, const void * buf)**

dataset_id	IN: Identifier of the dataset to write to
mem_type_id	IN: Identifier of memory datatype of the dataset
mem_space_id	IN: Identifier of the memory dataspace (or H5S_ALL)
file_space_id	IN: Identifier of the file dataspace (or H5S_ALL)
xfer_plist_id	IN: Identifier of the data transfer properties to use (or H5P_DEFAULT)
buf	IN: Buffer with data to be written to the file

Example 3 – Writing to an existing dataset

```
1 hid_t      file_id, dataset_id;  
2 herr_t     status;  
3 int        i, j, dset_data[4][6];
```

Initialize buffer

```
4 for (i = 0; i < 4; i++)  
5     for (j = 0; j < 6; j++)  
6         dset_data[i][j] = i * 6 + j + 1;
```

```
7 file_id = H5Fopen ("dset.h5", H5F_ACC_RDWR, H5P_DEFAULT);  
8 dataset_id = H5Dopen (file_id, "dset");
```

```
9 status = H5Dwrite (dataset_id, H5T_NATIVE_INT,  
                    H5S_ALL, H5S_ALL, H5P_DEFAULT, dset_data);
```

Example 3 – Writing to an existing dataset

```
1 hid_t      file_id, dataset_id;  
2 herr_t     status;  
3 int        i, j, dset_data[4][6];
```

```
4 for (i = 0; i < 4; i++)  
5     for (j = 0; j < 6; j++)  
6         dset_data[i][j] = i * 6 + j + 1;
```

Open existing file and dataset

```
7 file_id = H5Fopen ("dset.h5", H5F_ACC_RDWR, H5P_DEFAULT);  
8 dataset_id = H5Dopen (file_id, "dset");
```

```
9 status = H5Dwrite (dataset_id, H5T_NATIVE_INT,  
                   H5S_ALL, H5S_ALL, H5P_DEFAULT, dset_data);
```

Example 3 – Writing to an existing dataset

```
1 hid_t      file_id, dataset_id;
2 herr_t     status;
3 int        i, j, dset_data[4][6];

4 for (i = 0; i < 4; i++)
5     for (j = 0; j < 6; j++)
6         dset_data[i][j] = i * 6 + j + 1;

7 file_id = H5Fopen ("dset.h5", H5F_ACC_RDWR, H5P_DEFAULT);
8 dataset_id = H5Dopen (file_id, "dset");
```

Write to dataset

```
9 status = H5Dwrite (dataset_id, H5T_NATIVE_INT,
                    H5S_ALL, H5S_ALL, H5P_DEFAULT, dset_data);
```

Example 3: h5dump Output

```
HDF5 "dset.h5" {
  GROUP "/" {
    DATASET "dset" {
      DATATYPE { H5T_STD_I32BE }
      DATASPACE { SIMPLE ( 4, 6 ) / ( 4, 6 ) }
      DATA {
        1, 2, 3, 4, 5, 6,
        7, 8, 9, 10, 11, 12,
        13, 14, 15, 16, 17, 18,
        19, 20, 21, 22, 23, 24
      }
    }
  }
}
```

HDF5 High Level APIs

High Level APIs

- **Make HDF5 easier to use**
- **Encourage standard ways to store objects**
- **Included with HDF5 library**

However:

- **Currently available in C**
- **Still need HDF5 calls (H5Fopen/H5Fclose...)**

High Level APIs

- **HDF5 Lite (H5LT):** Functions that simplify the steps need to create/read datasets and attributes
- **HDF5 Image (H5IM):** Functions for creating images in HDF5.
- **HDF5 Table (H5TB):** Functions for creating tables (collections of records) in HDF5.
- **Others ...** (dimension scales, packet)

HDF5 LT

Programming model

```
file_id = H5Fcreate( "test.h5", ..... );
```

```
/* Call some High Level function */
```

```
H5Lsome_function( file_id, ...extra parameters );
```

```
status = H5Fclose( file_id );
```

Creating a dataset with HDF5 Lite

Specific type functions for each numeric type:

C language type	Function	HDF5 type
char	H5LTmake_dataset_char	H5T_NATIVE_CHAR
short	H5LTmake_dataset_short	H5T_NATIVE_SHORT
int	H5LTmake_dataset_int	H5T_NATIVE_INT
long	H5LTmake_dataset_long	H5T_NATIVE_LONG
float	H5LTmake_dataset_float	H5T_NATIVE_FLOAT
double	H5LTmake_dataset_double	H5T_NATIVE_DOUBLE

Reading a dataset with HDF5 Lite

Specific type functions also for reading:

C language type	Function	HDF5 type
char	H5LTread_dataset_char	H5T_NATIVE_CHAR
short	H5LTread_dataset_short	H5T_NATIVE_SHORT
int	H5LTread_dataset_int	H5T_NATIVE_INT
long	H5LTread_dataset_long	H5T_NATIVE_LONG
float	H5LTread_dataset_float	H5T_NATIVE_FLOAT
double	H5LTread_dataset_double	H5T_NATIVE_DOUBLE

HDF5 Image (H5IM)

- **Purpose: to defines a standard storage for HDF5 datasets that are intended to be interpreted as images**
- **2 types of images (8bit indexed to a palette, and 24bit with 3 color planes (RGB))**
- **Also palette functions**

HDF5 Image

- *Image functions*
- H5IMmake_image_8bit
- H5IMmake_image_24bit
- H5IMget_image_info
- H5IMread_image
- H5IMis_image

HDF5 Image

Palette functions

- H5IMmake_palette
- H5IMlink_palette
- H5IMunlink_palette
- H5IMget_npalettes
- H5IMget_palette_info
- H5IMget_palette
- H5IMis_palette

HDF5 Image (H5IM)

Programming example

```
file_id = H5Fcreate ( "ex_image1.h5", H5F_ACC_TRUNC, H5P_DEFAULT,  
                    H5P_DEFAULT );  
  
/* Write image */  
status = H5IMmake_image_8bit ( file_id, "Image1", WIDTH, HEIGHT, data );  
  
/* Make a palette */  
status = H5IMmake_palette ( file_id, "Palette", pal_dims, pal );  
  
/* Attach a palette to the image dataset */  
status = H5IMlink_palette ( file_id, "Image1", "Palette" );  
  
/* Close the file. */  
status = H5Fclose ( file_id );
```

For more information...



- **HDF website**

- <http://hdf.ncsa.uiuc.edu/>



- **HDF5 Information Center**

- <http://hdf.ncsa.uiuc.edu/HDF5/>



- **HDF Helpdesk**

- hdfhelp@ncsa.uiuc.edu or

- help@hdfgroup.org

Thank you

This presentation is based upon work supported in part by a Cooperative Agreement with the National Aeronautics and Space Administration (NASA) under NASA grant NNG05GC60A. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NASA. Other support provided by NCSA and other sponsors and agencies (<http://hdf.ncsa.uiuc.edu/acknowledge.html>).