Introduction to HDF5

Quincey Koziol
Director of Core Software & HPC
The HDF Group

Why HDF5?

- Have you ever asked yourself:
  - How will I deal with one-file-per-processor in the petascale era?
  - Do I need to be an “MPI and Lustre pro” to do my research?
  - Where is my checkpoint file?
- HDF5 hides all complexity so you can concentrate on Science
  - Optimized I/O to single shared file

Goal

- Introduce you to HDF5
  - HDF5 data model
  - HDF5 programming model
  - Parallel access to HDF5
  - HDF5 performance tuning hints

WHAT IS HDF5?

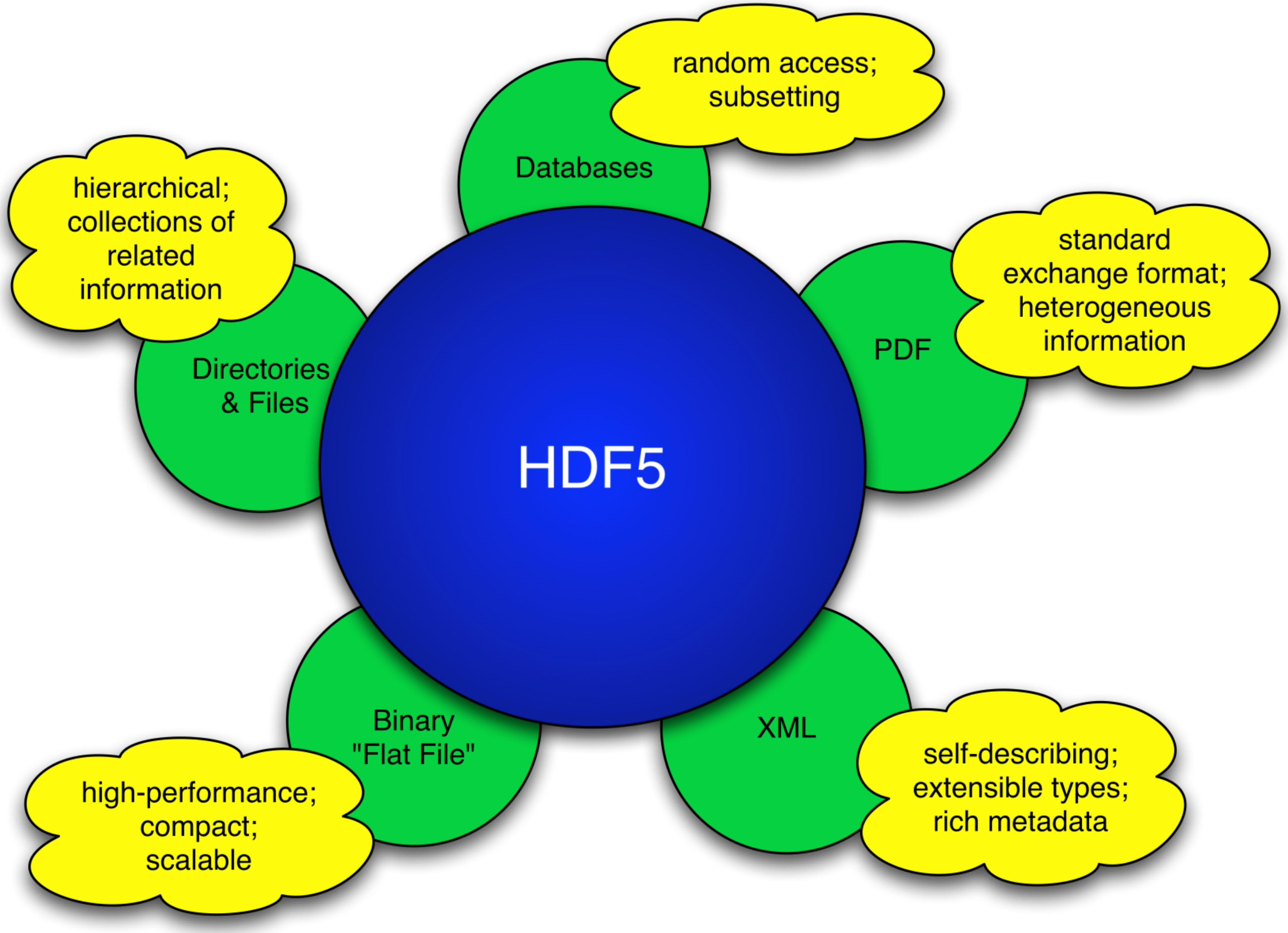
What is HDF5?

- HDF5 == Hierarchical Data Format, v5
- 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

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 is like …


HDF5
Databases
random access; subsetting
Directories & Files
hierarchical; collections of related information
PDF
standard exchange format; heterogeneous information
Binary "Flat File"
high-performance; compact; scalable
XML
self-describing; extensible types; rich metadata

What is HDF5?

- **A versatile data model** that can represent very complex data objects and a wide variety of metadata.
- **A completely portable file format** with no limit on the number or size of data objects stored.
- **An open source software library** that runs on a wide range of computational platforms, from cell phones to massively parallel systems, and implements a high-level API with C, C++, Fortran, and Java interfaces.
- **A rich set of integrated performance features** that allow for access time and storage space optimizations.
- **Tools and applications** for managing, manipulating, viewing, and analyzing the data in the collection.

Why use HDF5?

- Challenging data:
  - Application data that pushes the limits of what can be addressed by traditional database systems, XML documents, or in-house data formats.
- Software solutions:
  - For very large datasets, very fast access requirements, or very complex datasets.
  - To easily share data across a wide variety of computational platforms using applications written in different programming languages.
  - That take advantage of the many open-source and commercial tools that understand HDF5.
  - Enabling long-term preservation of data.

Why HDF5?

- Have you ever asked yourself:
  - How will I deal with changes in storage technology?
  - Do I need to be an “I/O Pro” to do my research?
  - How do I read data in my old files?
- HDF5 hides all this complexity so you can concentrate on science
  - Optimized I/O to single shared file

Who uses HDF5?

- Examples of HDF5 user communities
  - Astrophysics
  - Astronomers
  - NASA Earth Science Enterprise
  - Dept. of Energy Labs
  - Supercomputing centers in US, Europe and Asia
  - Financial Institutions
  - NOAA
  - Manufacturing industries
  - Many others
- For a more detailed list, visit
  - http://www.hdfgroup.org/HDF5/users5.html

Brief History of HDF

1987 At NCSA (University of Illinois), a task force formed to create an architecture-independent format and library:

AEHOO (All Encompassing Hierarchical Object Oriented format)

Became HDF

Early 1990's NASA adopted HDF for Earth Observing System project

1996 DOE's ASC (Advanced Simulation and Computing) Project began collaborating with the HDF group (NCSA) to create "Big HDF" (Increase in computing power of DOE systems at LLNL, LANL and Sandia National labs, required bigger, more complex data files).

"Big HDF" became HDF5.

HDF5 was released with support from DOE Labs, NASA, NCSA

2006 The HDF Group spun off from University of Illinois as non-profit corporation

The HDF Group

- Established in 1988
  - 18 years at University of Illinois' National Center for Supercomputing Applications
  - 8years as independent non-profit company, "The HDF Group"
- The HDF Group owns HDF4 and HDF5
  - HDF4 & HDF5 formats, libraries, and tools are open source and freely available with BSD-style license
- Currently employ ~35 FTEs
  - *Looking for more developers now!*

The HDF Group Mission

To ensure long-term accessibility of HDF data through sustainable development and support of HDF technologies.

Goals of The HDF Group

- Maintain and evolve HDF for sponsors and communities that depend on it
- Provide support to the HDF communities through consulting, training, tuning, development, research
- Sustain the company for the long term to assure data access over time

The HDF Group Services

- Helpdesk and Mailing Lists
  - Available to all users as a first level of support: **help@hdfgroup.org**
  - User Community Mailing List: hdf-forum@lists.hdfgroup.org
- Priority Support
  - Rapid issue resolution and advice
- Consulting
  - Needs assessment, troubleshooting, design reviews, etc.
- Training
  - Tutorials and hands-on practical experience
- Enterprise Support
  - Coordinating HDF activities across departments
- Special Projects
  - Adapting customer applications to HDF
  - New HDF features and tools
  - Research and Development

HDF5 DATA MODEL

HDF5 Technology Platform

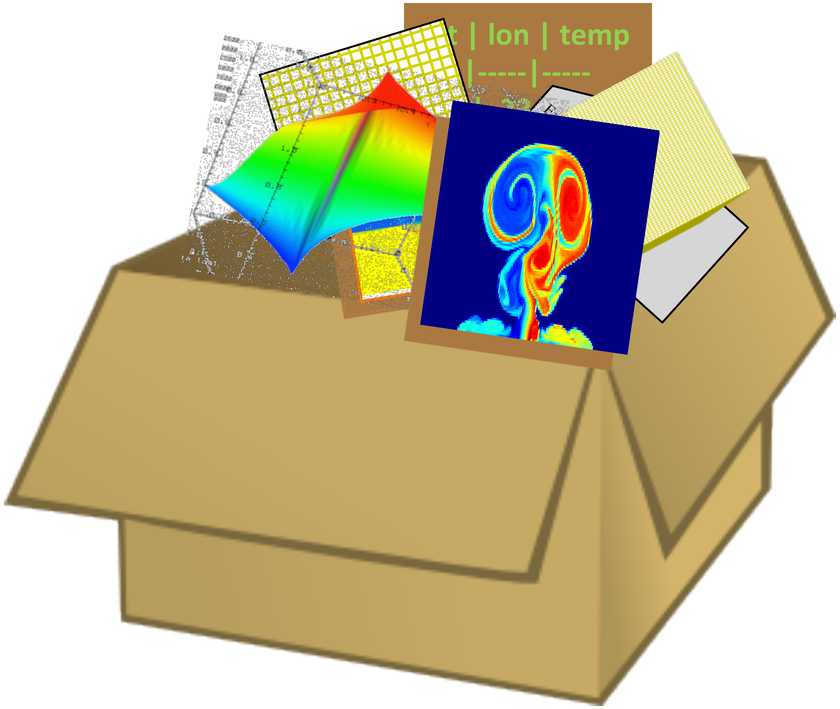
- HDF5 Abstract Data Model
  - Defines the “building blocks” for data organization and specification
  - Files, Groups, Links, Datasets, Attributes, Datatypes, Dataspaces

- HDF5 Software
  - Tools
  - Language Interfaces
  - HDF5 Library

- HDF5 Binary File Format
  - Bit-level organization of HDF5 file
  - Defined by HDF5 File Format Specification

HDF5 File

An HDF5 file is a **container** that holds data objects.



HDF5 Data Model

Dataset

Link

Group

Datatype

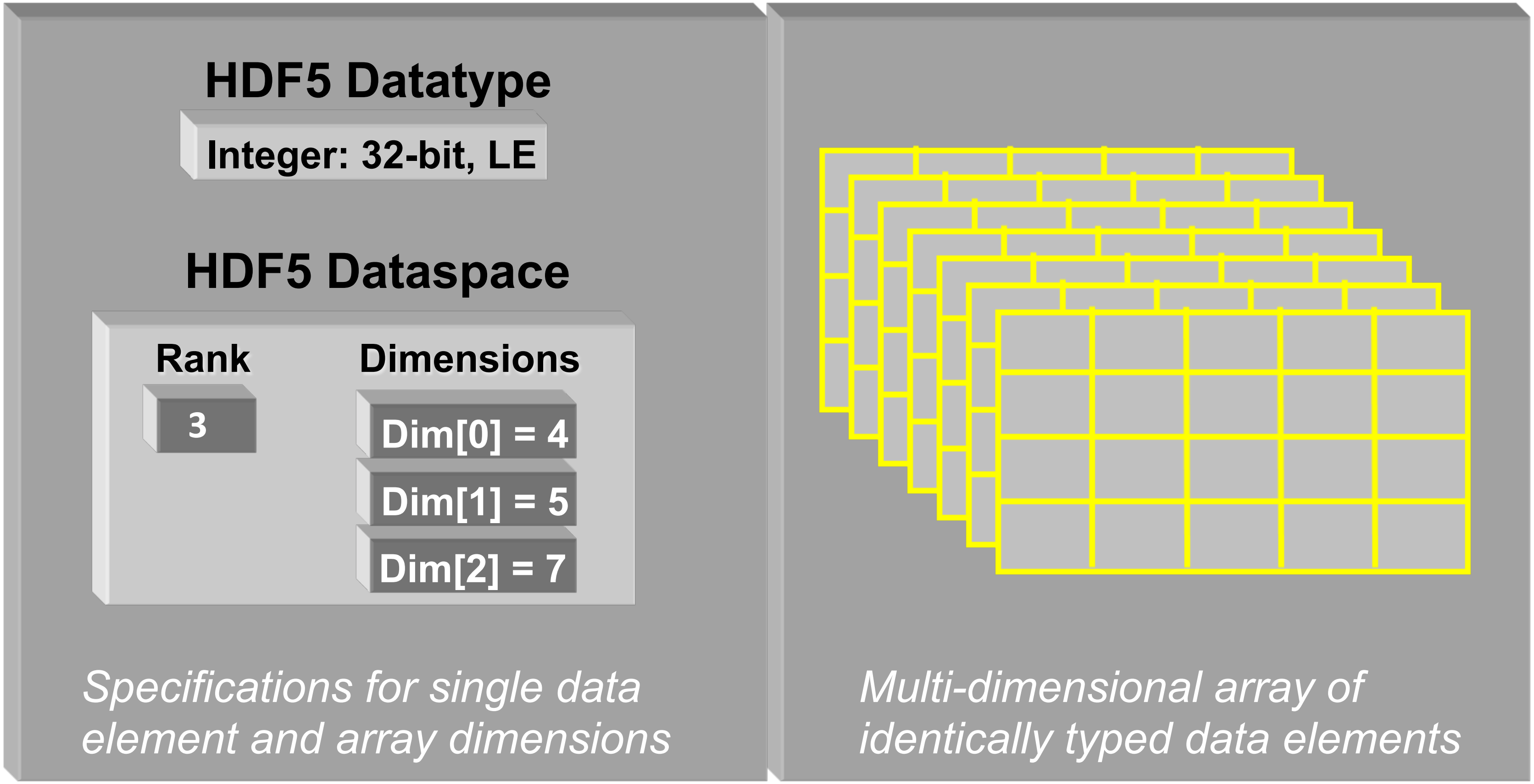
HDF5 Objects

Attribute

Dataspace

File

HDF5 Dataset



HDF5 Datatype

Integer: 32-bit, LE

HDF5 Dataspace

Rank	Dimensions

3	Dim[0] = 4
	Dim[1] = 5
	Dim[2] = 7

Specifications for single data element and array dimensions

Multi-dimensional array of identically typed data elements

- HDF5 datasets **organize and contain** data elements.
  - HDF5 datatype describes individual data elements.
  - HDF5 dataspace describes the logical layout of the data elements.

HDF5 Dataspace

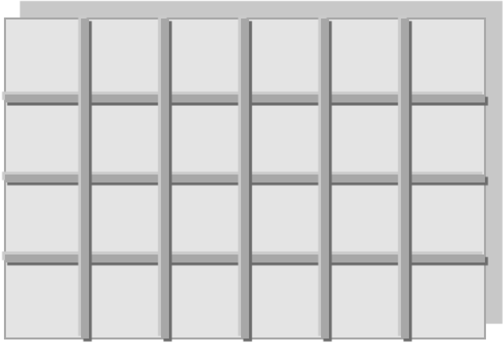
- Describes the logical layout of the elements in an HDF5 dataset
  - NULL
    - no elements
  - Scalar
    - single element
  - Simple array (*most common*)
    - multiple elements organized in a rectangular array
      - rank = number of dimensions
      - dimension sizes = number of elements in each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

HDF5 Dataspace

Two roles:

Dataspace contains spatial information

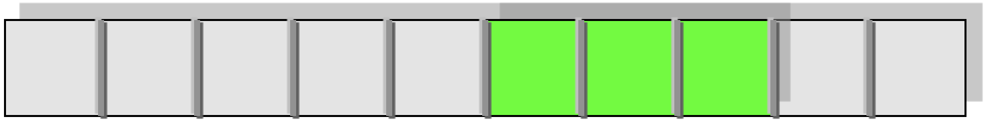
- Rank and dimensions
- Permanent part of dataset definition



Rank = 2

Dimensions = 4x6

Partial I/0: Dataspace describes application’s data buffer and data elements participating in I/O



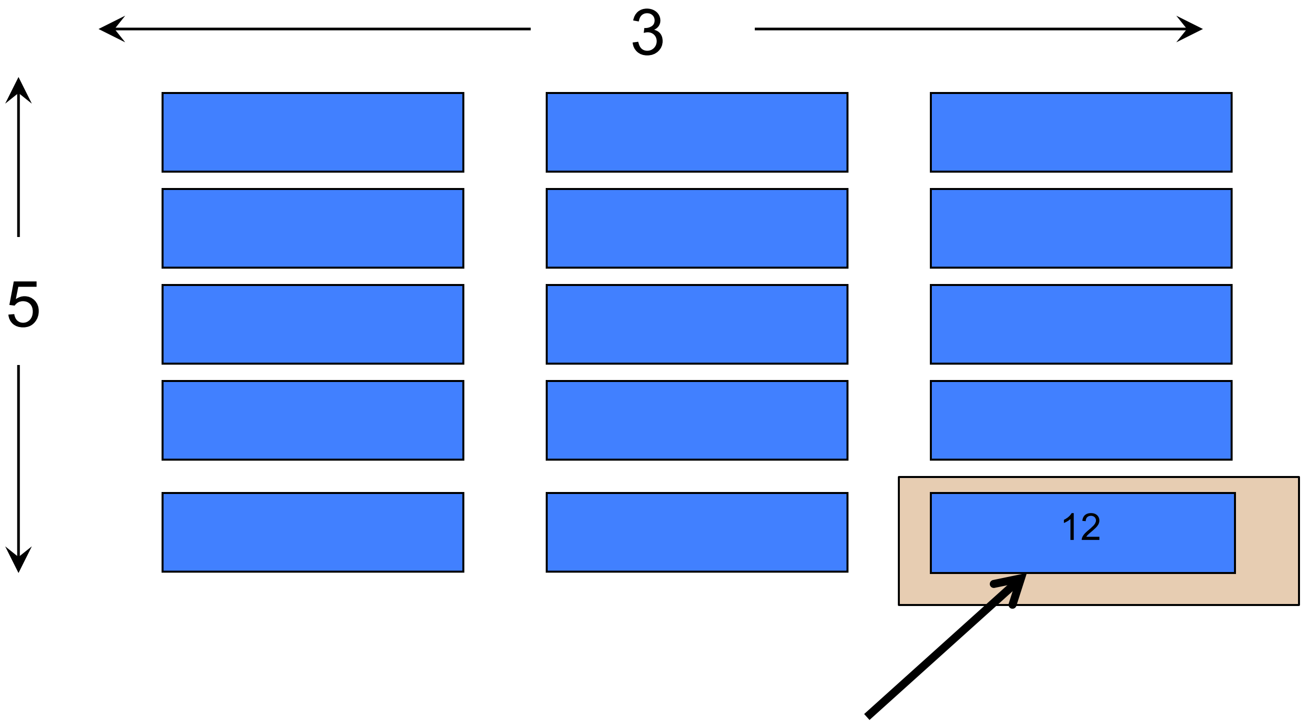
Rank = 1

Dimension = 10

HDF5 Datatypes

- Describe individual data elements in an HDF5 dataset
- Wide range of datatypes supported
  - Integer
  - Float
  - Enum
  - Array
  - User-defined (e.g., 13-bit integer)
  - Variable-length types (e.g., strings, vectors)
  - Compound (similar to C structs)
  - More …

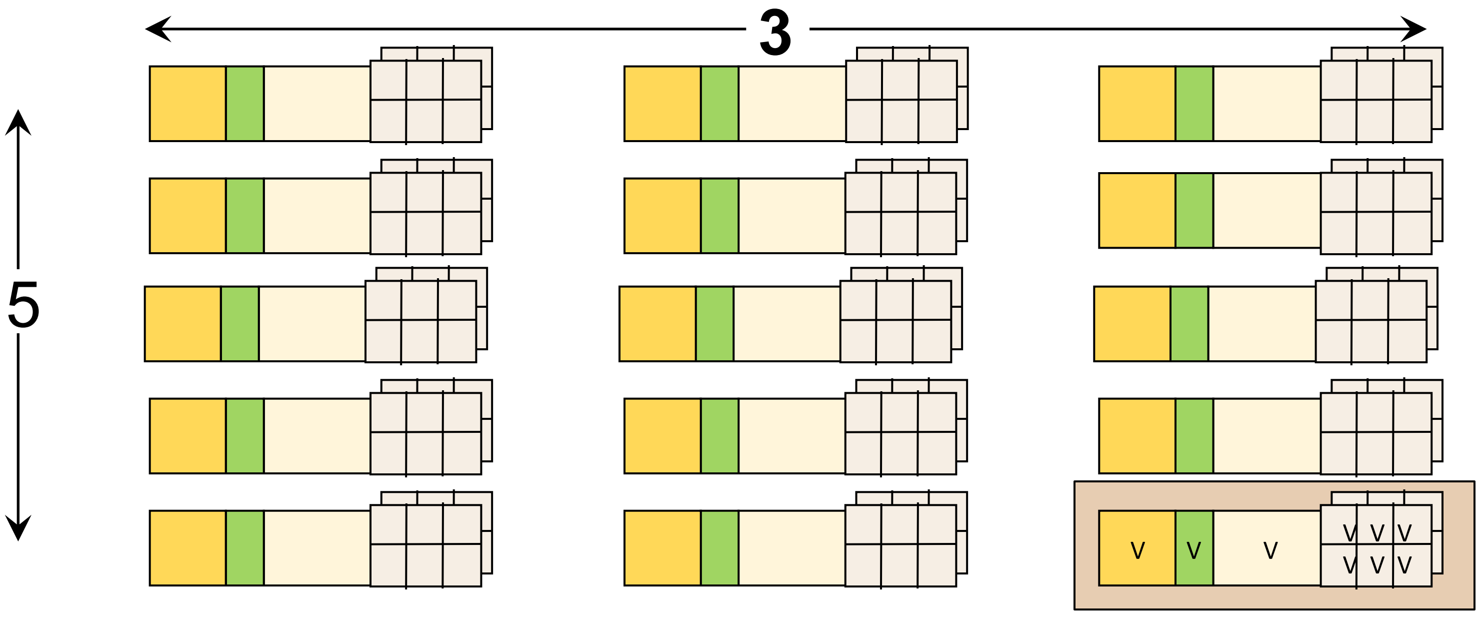
HDF5 Dataset

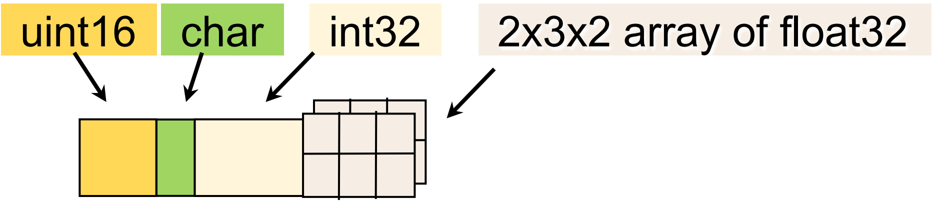


Datatype: 32-bit Integer

Dataspace: Rank = 2
Dimensions = 5 x 3

HDF5 Dataset with Compound Datatype





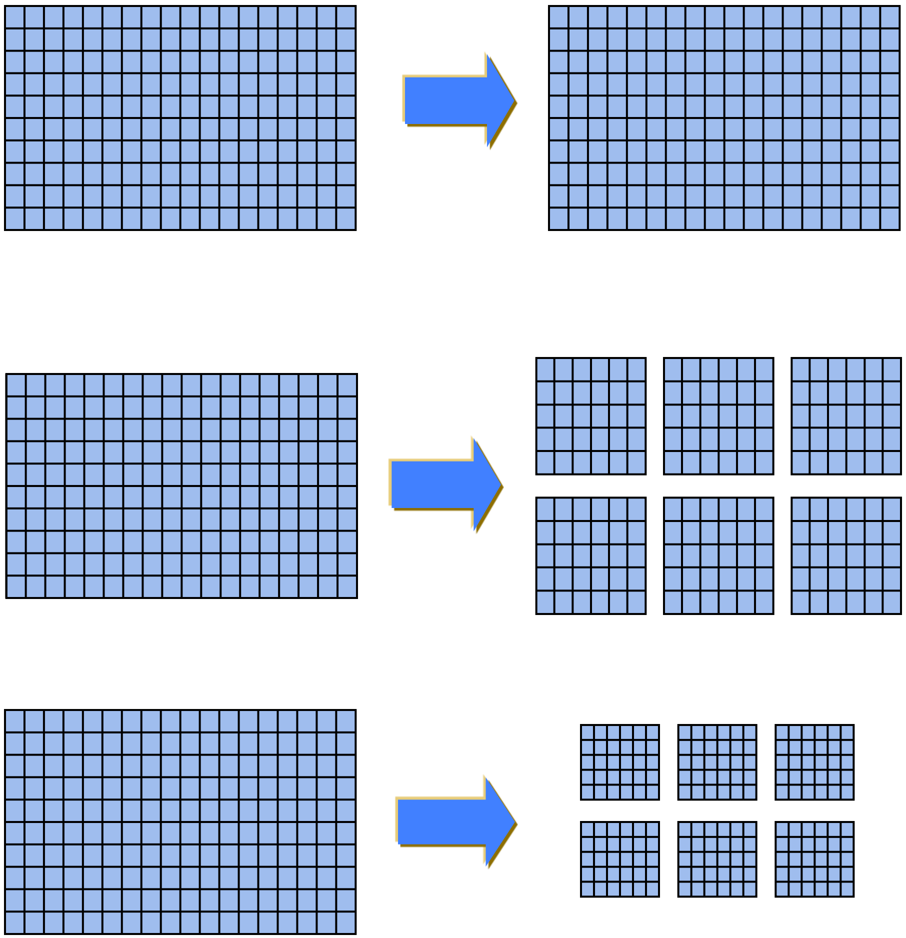
Compound Datatype:

Dataspace: Rank = 2
Dimensions = 5 x 3

How are data elements stored?

Buffer in memory

Data in the file



Contiguous (default)

Data elements stored physically adjacent to each other

Chunked

Better access time for subsets; extendible

Chunked & Compressed

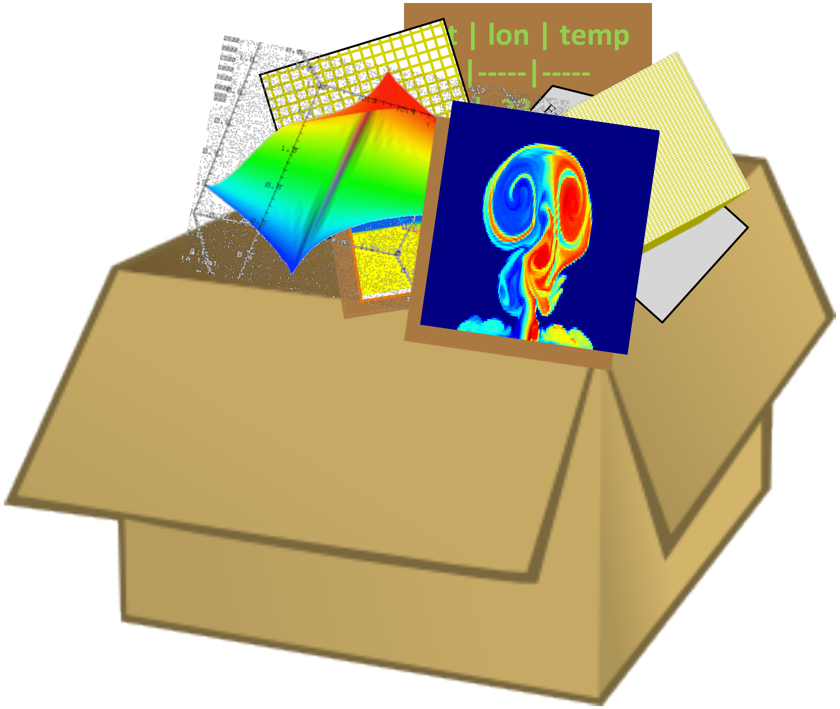
Improves storage efficiency, transmission speed

HDF5 Attributes

- Typically contain user metadata
- Have a <u>name</u> and a <u>value</u>
- Attributes “decorate” HDF5 objects
- Value is described by a datatype and a dataspace
- Analogous to a dataset, but do not support partial I/O operations; nor can they be compressed or extended

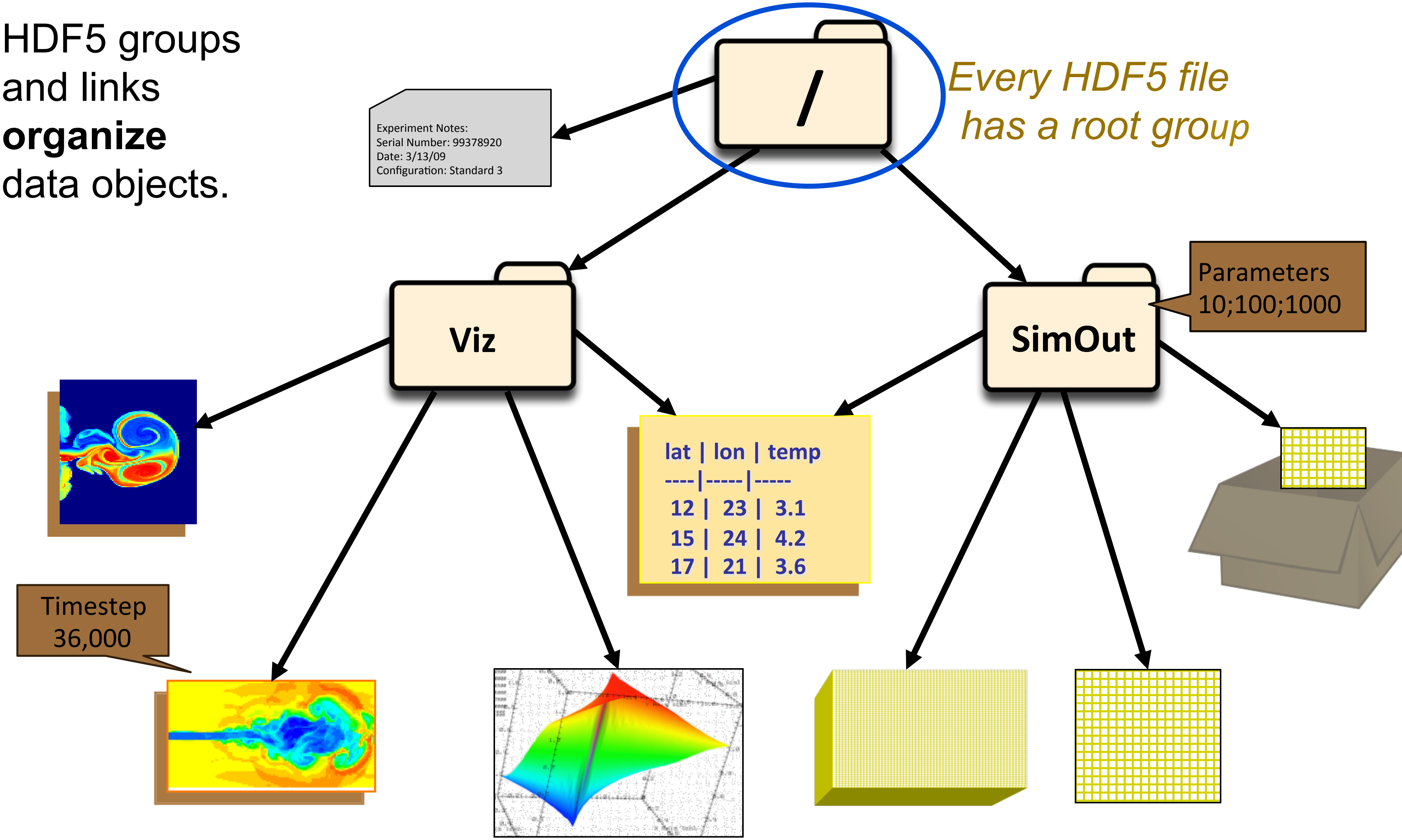
HDF5 File

An HDF5 file is a **smart container** that holds data objects.



HDF5 Groups and Links

HDF5 groups and links **organize** data objects.



Every HDF5 file has a root group

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

Parameters
10;100;1000

Timestep
36,000

```
lat | lon | temp
----|-----|-----
 12 |  23 |  3.1
 15 |  24 |  4.2
 17 |  21 |  3.6
```

HDF5 SOFTWARE

HDF5 Technology Platform

- HDF5 Abstract Data Model
  - Defines the “building blocks” for data organization and specification
  - Files, Groups, Links, Datasets, Attributes, Datatypes, Dataspaces
- HDF5 Software
  - Tools
  - Language Interfaces
  - HDF5 Library
- HDF5 Binary File Format
  - Bit-level organization of HDF5 file
  - Defined by HDF5 File Format Specification

HDF5 Home Page

HDF5 home page: http://hdfgroup.org/HDF5/

- Latest release: HDF5 1.8.13 (1.8.14 coming in November 2014)

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

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 HDF5 files

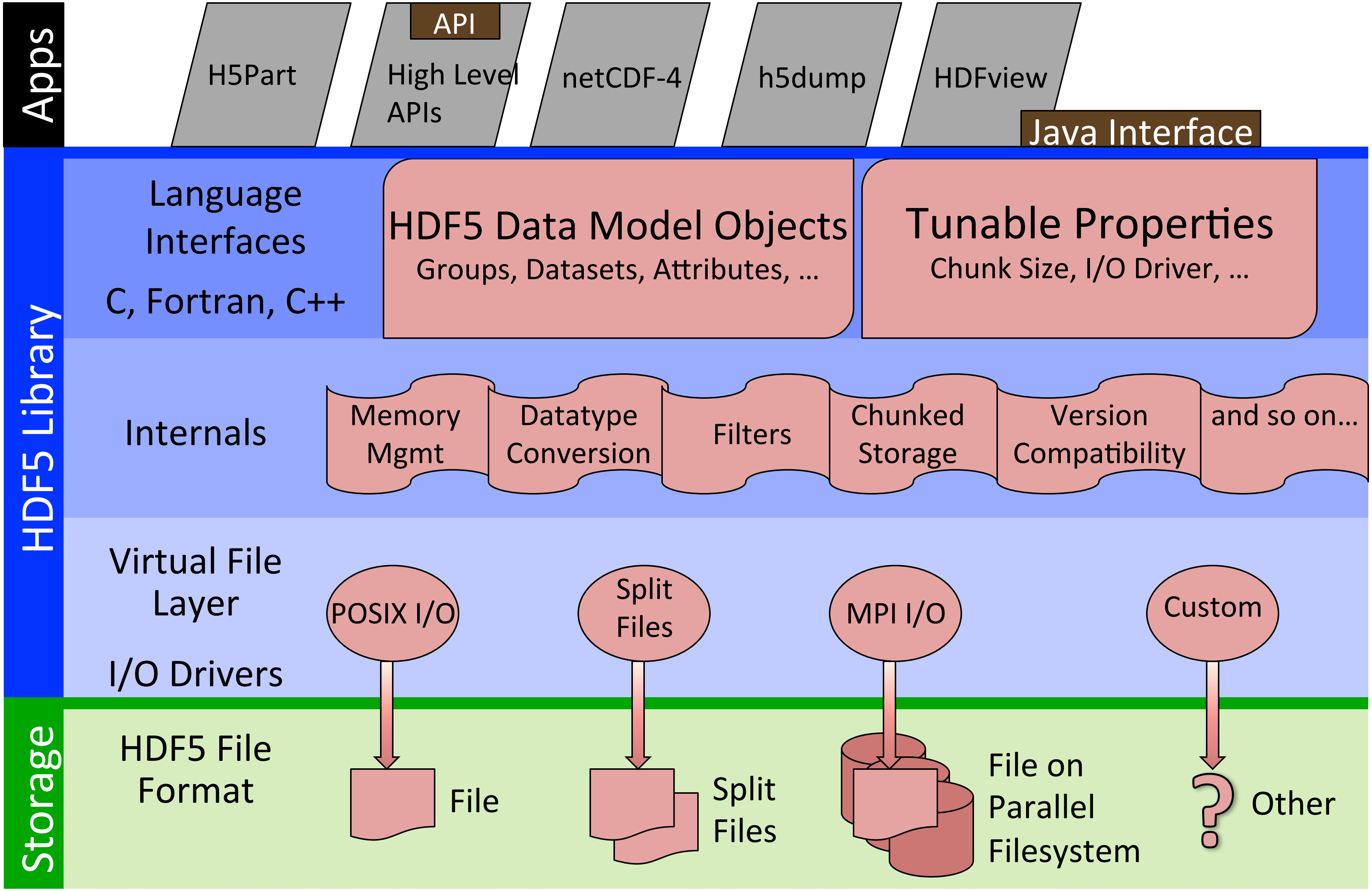
http://www.hdfgroup.org/hdf-java-html/hdfview/

HDF5 Examples (C, Fortran, Java, Python, Matlab)

http://www.hdfgroup.org/ftp/HDF5/examples/

HDF5 PROGRAMMING MODEL AND API

HDF5 Software Layers & Storage



The General HDF5 API

- C**,** Fortran, Java, C++, and .NET bindings
- IDL, MATLAB, Python (H5Py, PyTables)
- C routines begin with prefix H5*?*

  ? is a character corresponding to the type of object the function acts on

Example Functions:

H5D :	**D**ataset interface	*e.g.,* **H5Dread**
H5F :	**F**ile interface	*e.g.,* **H5Fopen**
H5S :	data**S**pace interface	*e.g.,* **H5Sclose**

The HDF5 API

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

Victorinox
Swiss Army
Cybertool 34

- 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

General Programming Paradigm

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

- Properties of object are <u>optionally</u> defined
  - ✓Creation properties (e.g., use chunking storage)
  - ✓Access properties

Basic Functions

H5**F**create (H5**F**open)	*create (open) File*
H5**S**create_simple/H5**S**create	*create dataSpace*
H5**D**create (H5**D**open)	*create (open) Dataset*
H5**D**read, H5**D**write	*access Dataset*
H5**D**close	*close Dataset*
H5**S**close	*close dataSpace*
H5**F**close	*close File*

Other Common Functions

Data**S**paces:	H5Sselect_hyperslab (Partial I/O) H5Sselect_elements (Partial I/O) H5Dget_space
Data**T**ypes:	H5Tcreate, H5Tcommit, H5Tclose H5Tequal, H5Tget_native_type
Groups:	H5Gcreate, H5Gopen, H5Gclose
Attributes:	H5Acreate, H5Aopen_name, H5Aclose, H5Aread, H5Awrite
Property lists:	H5Pcreate, H5Pclose H5Pset_chunk, H5Pset_deflate

C EXAMPLES

How to compile HDF5 applications

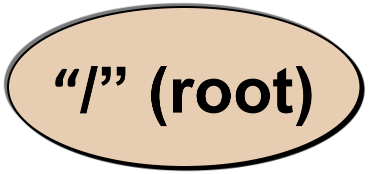
- **`h5cc`** – HDF5 C compiler command
- **`h5fc`** – HDF5 F90 compiler command
- **`h5c++`** - HDF5 C++ compiler command
- To compile:
  - **`% h5cc h5prog.c`**
  - **`% h5fc h5prog.f90`**
  - **`% h5c++ h5prog.cpp`**

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)

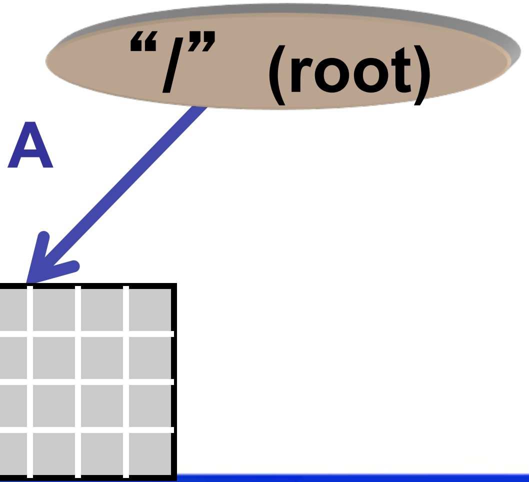
Note: Return codes not checked for errors in code samples.

Code: Create a Dataset

```
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);
```



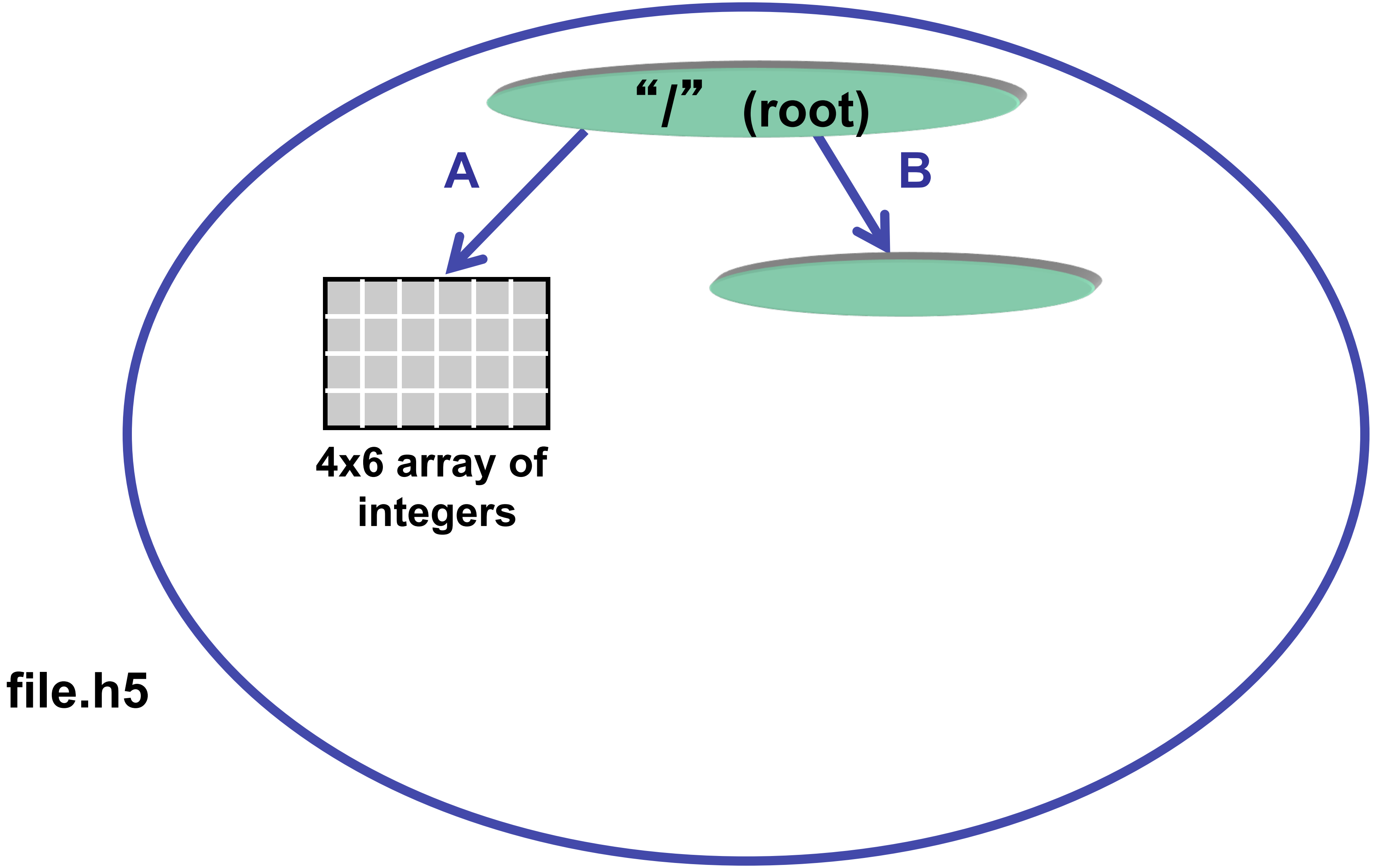
Code: Create a Group

```
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);
```

Example: Create Dataset & Group



Output of h5dump

```
$ h5dump file.h5

HDF5 "file.h5" {
GROUP "/" {
   DATASET "A" {
      DATATYPE  H5T_STD_I32BE
      DATASPACE  SIMPLE { ( 4, 6 ) / ( 4, 6 ) }
      DATA {
      (0,0): 0, 0, 0, 0, 0, 0,
      (1,0): 0, 0, 0, 0, 0, 0,
      (2,0): 0, 0, 0, 0, 0, 0,
      (3,0): 0, 0, 0, 0, 0, 0
      }
   }
   GROUP "B" {
   }
}
}
```

Example Code - H5Dwrite

```
 int  wdata[4][6];

 /* Initialize the dataset. */
 for (i = 0; i < 4; i++)
    for (j = 0; j < 6; j++)
         wdata[i][j] = i * 6 + j + 1;
…. .
status = H5Dwrite (dataset_id, H5T_NATIVE_INT,
           H5S_ALL,H5S_ALL, H5P_DEFAULT, wdata);
```

Output of h5dump after writing

```
$ h5dump file.h5
HDF5 "file.h5" {
GROUP "/" {
   DATASET "A" {
      DATATYPE  H5T_STD_I32BE
      DATASPACE  SIMPLE { ( 4, 6 ) / ( 4, 6 ) }
      DATA {
      (0,0): 1, 2, 3, 4, 5, 6,
      (1,0): 7, 8, 9, 10, 11, 12,
      (2,0): 13, 14, 15, 16, 17, 18,
      (3,0): 19, 20, 21, 22, 23, 24
      }
   }
   GROUP "B" {
   }
}
}
```

PARTIAL I/O IN HDF5

How to write a row?

```
$ h5dump file.h5

HDF5 "file.h5" {
GROUP "/" {
   DATASET "A" {
      DATATYPE  H5T_STD_I32BE
      DATASPACE  SIMPLE { ( 4, 6 ) / ( 4, 6 ) }
      DATA {
      (0,0): 0, 0, 0, 0, 0, 0,
      (1,0): 1, 2, 3, 4, 5, 6,
      (2,0): 0, 0, 0, 0, 0, 0,
      (3,0): 0, 0, 0, 0, 0, 0
      }
   }
   GROUP "B" {
   }
}
}
```

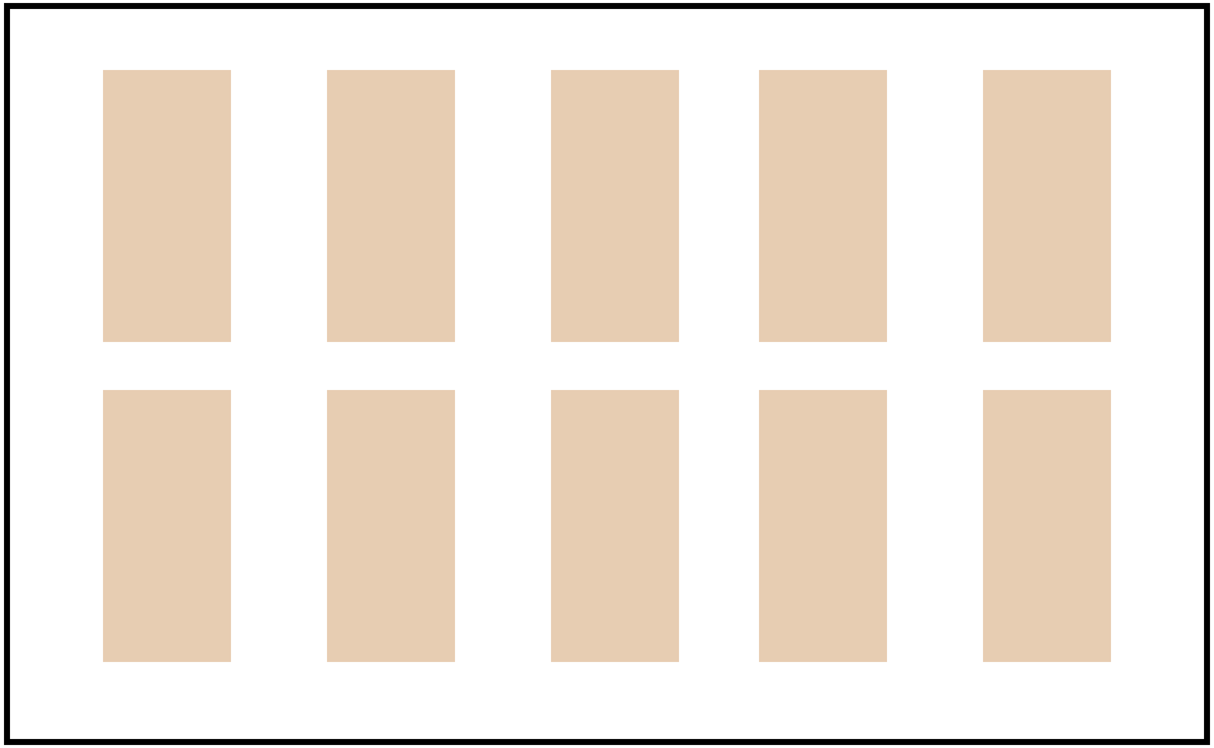
How to Describe a Subset in HDF5?

- Before writing and reading a subset of data one has to describe it to the HDF5 Library.
- HDF5 APIs and documentation refer to a subset as a “selection” or “hyperslab selection”.
- If specified, HDF5 Library will perform I/O on a selection *only* and not on all elements of a dataset.

Types of Selections in HDF5

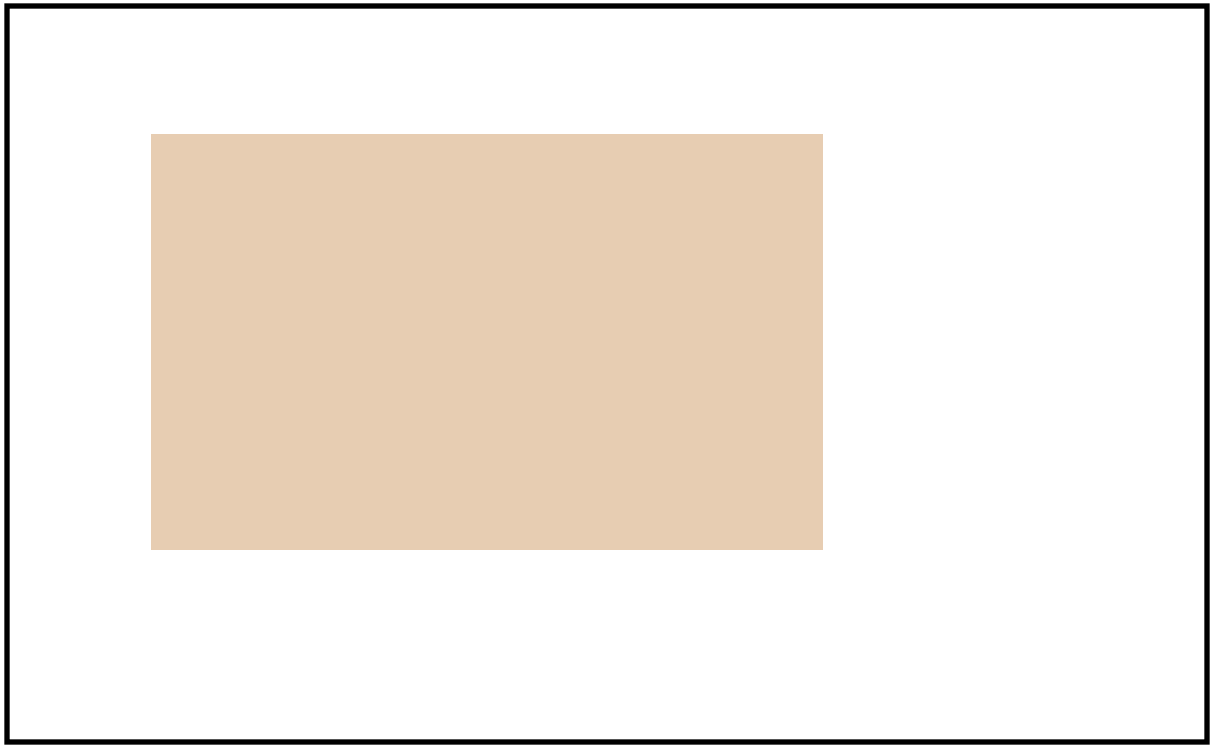
- Two types of selections
  - Hyperslab selection
    - Regular hyperslab
    - Simple hyperslab
    - Result of set operations on hyperslabs (union, difference, …)
  - Point selection
- Hyperslab selection is especially important for doing parallel I/O in HDF5 (See Parallel HDF5 Tutorial)

Regular Hyperslab



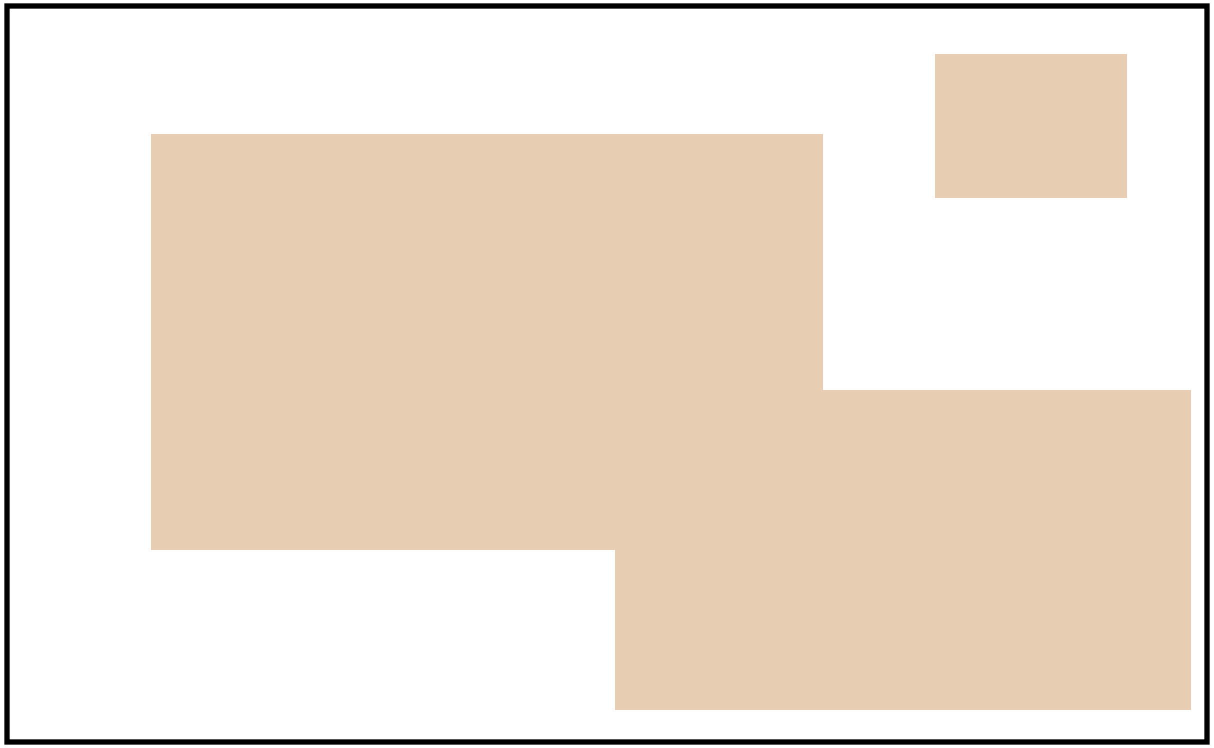
Collection of regularly spaced blocks of equal size

Simple Hyperslab



Contiguous subset or sub-array

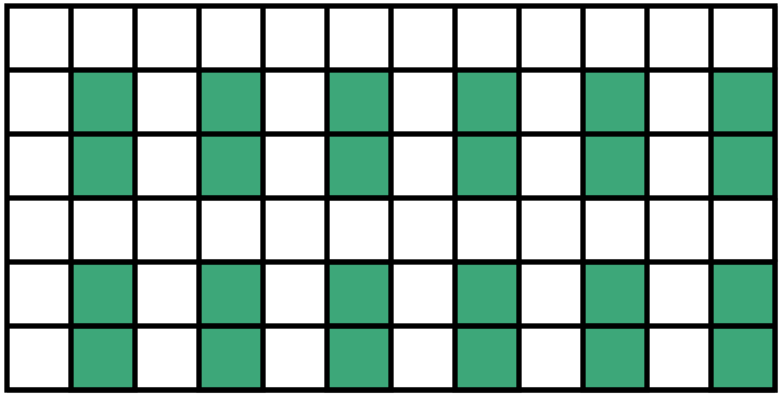
Hyperslab Selection



Result of union operation on three simple hyperslabs

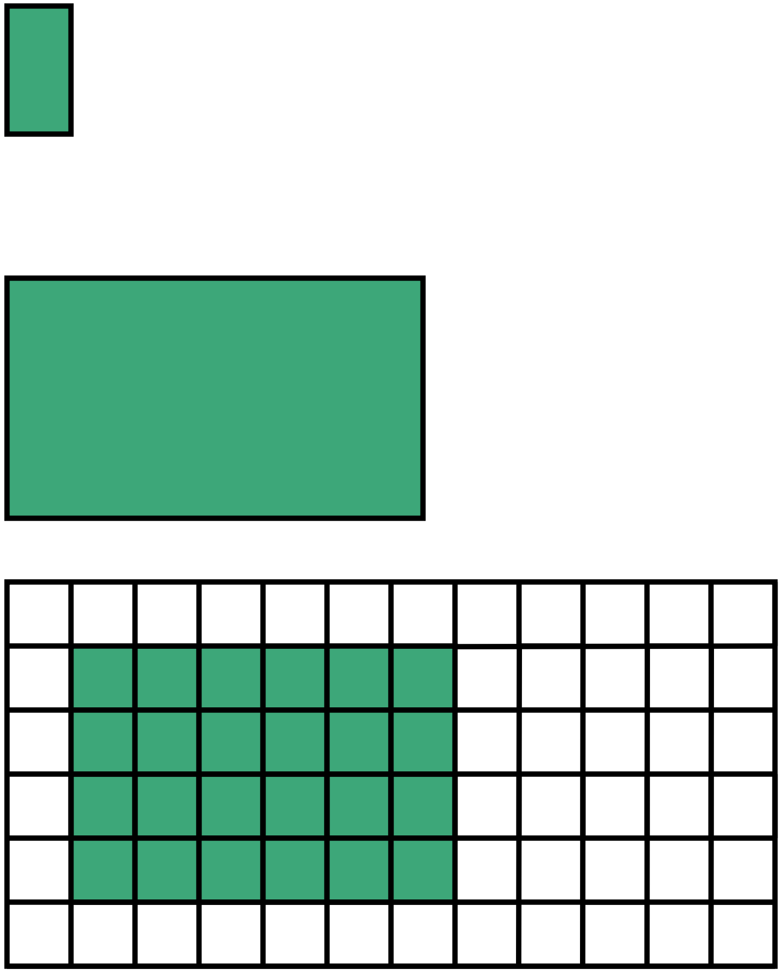
HDF5 Hyperslab Description

- *Everything is "measured" in number of elements*
- Start - starting location of a hyperslab (1,1)
- Stride - number of elements that separate each block (3,2)
- Count - number of blocks (2,6)
- Block - block size (2,1)



Simple Hyperslab Description

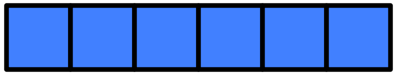
- Two ways to describe a simple hyperslab
- As *several* blocks
  - **Stride – (1,1)**
  - **Count – (2,6)**
  - Block – (2,1)
- As *one* block
  - Stride – (1,1)
  - Count – (1,1)
  - **Block – (4,6)**



No performance penalty for one way or another

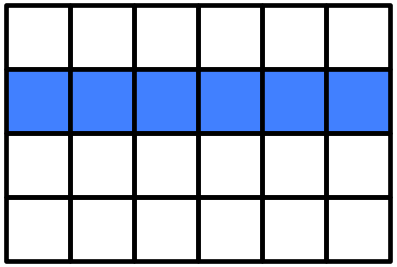
Writing a row

- Memory space selection is 1-dim array of size 6



- File space selection

start = {1,0}, stride = {1,1}, count = {1,6}, block = {1,1}



Number of elements selected in memory should be the same as selected in the file

Writing a row

```
hid_t       mspace_id, fspace_id;
hsize_t     dims[1] = {6};
hsize_t     start[2], count[2];
…..
/* Create memory dataspace */
mspace_id = H5Screate_simple(RANK, dims, NULL);

/* Get file space identifier from the dataset */
fspace_id = H5Dget_space(dataset_id);

/* Select hyperslab in the dataset to write too */
start[0] = 1;
start[1] = 0;
count[0] = 1;
count[1] = 6;
status = H5Sselect_hyperslab(fspace_id, H5S_SELECT_SET,
         start, NULL, count, NULL);
H5Dwrite(dataset_id, H5T_NATIVE_INT, mspace_id, fspace_id,
         H5P_DEFAULT, wdata);
```

HDF5 FILE FORMAT

HDF5 Technology Platform

- HDF5 Abstract Data Model
  - Defines the “building blocks” for data organization and specification
  - Files, Groups, Links, Datasets, Attributes, Datatypes, Dataspaces

- HDF5 Software
  - Tools
  - Language Interfaces
  - HDF5 Library

- HDF5 Binary File Format
  - Bit-level organization of HDF5 file
  - Defined by HDF5 File Format Specification

HDF5 File Format

- Defined by the *HDF5 File Format Specification.*

http://www.hdfgroup.org/HDF5/doc/H5.format.html

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

- HDF5 library adheres to the File Format, users do not need to know the guts of this information.

HDF5 Roadmap

- Concurrency
  - Single-Writer/Multiple-Reader (SWMR)
  - Internal threading
- Virtual Object Layer (VOL)
- Data Analysis
  - Query / View / Index APIs
- Native HDF5 client/server

- Performance
  - Scalable chunk indices
  - Metadata aggregation and Page buffering
  - Asynchronous I/O
  - Variable-length records
- Fault tolerance
- Parallel I/O
- I/O Autotuning

Thank You!

Questions?