

# **HDF5 Advanced Topics Selections Object's Properties Storage Methods and Filters**

HDF and HDF-EOS Workshop IX

November 30, 2005

# Topics

Goal: Introduce HDF5 selections and object's properties

Hyperslab and Point Selection

HDF5 Dataset properties

*I/O and Storage Properties (filters)*

HDF5 File properties

*I/O and Storage Properties (drivers)*

# Working with Selections

# What is a Selection?

A portion of a dataset's dataspace:

- **Hyperslab:** It can be a logically contiguous collection of points in a dataspace, or it can be a regular pattern of points or blocks in a dataspace.
- **Individual Points:** Selected points in the dataspace
- **Results of Set Operations on hyperslabs or points (union, difference, ...)**

# Hyperslab Selection

Dataset

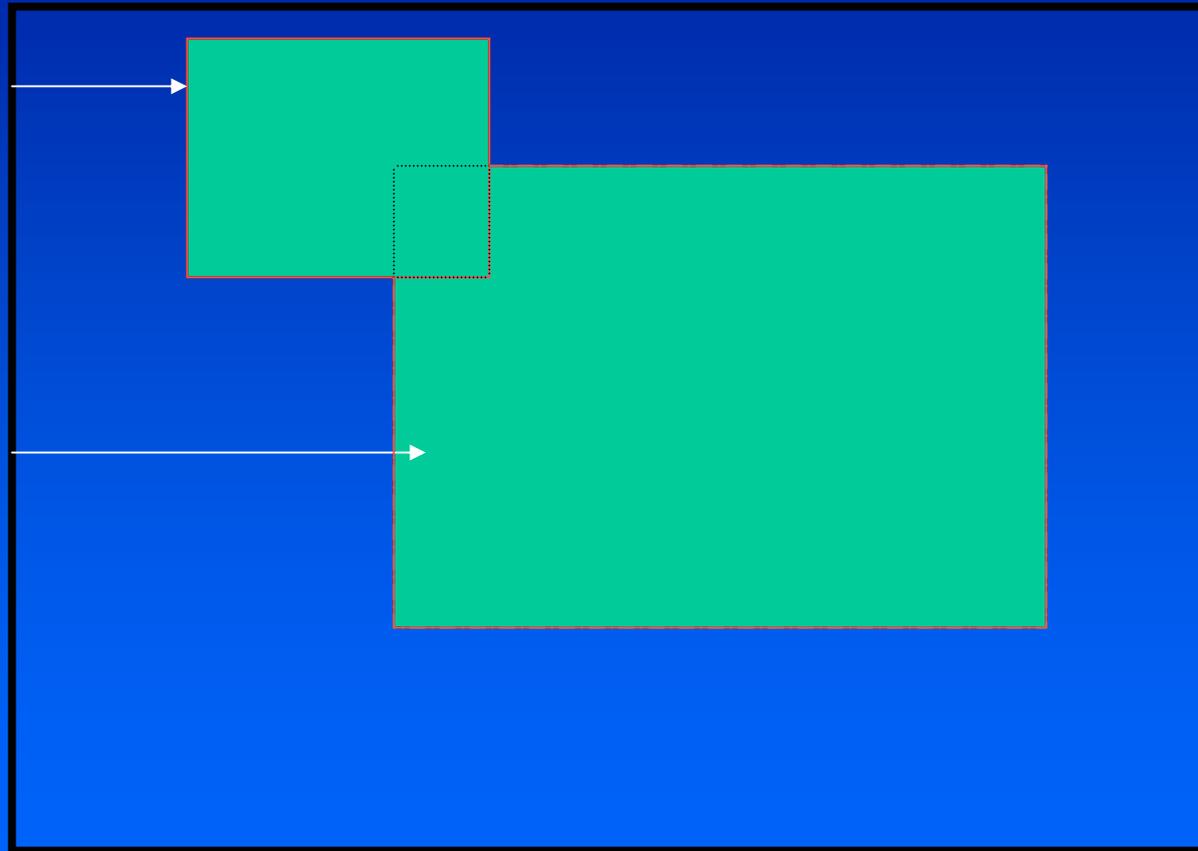
Hyperslab

+

Hyperslab

=

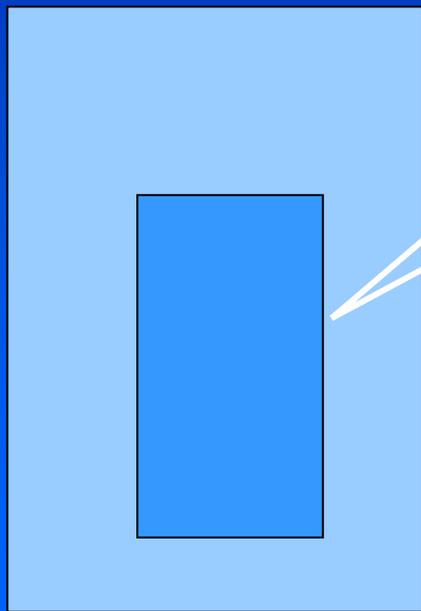
Union of  
Hyperslabs



# Reading Dataset into Memory from File

**File**

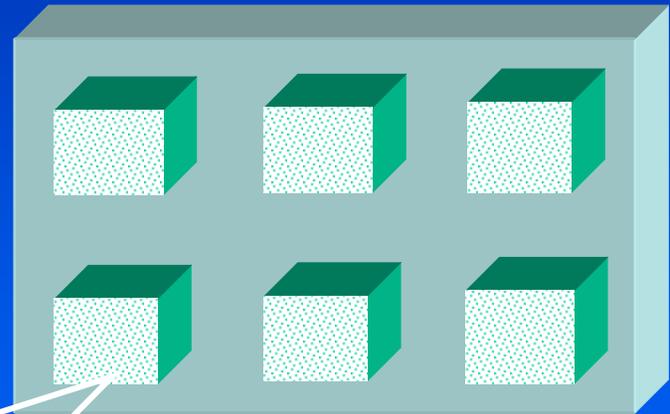
*2D array of 16-bit ints*



*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.**

# Steps for Making Selections

- Open the file
- Open the dataset
- Create a file dataspace for the dataset
- Create a memory dataspace for the dataset
- Make the selection(s)
- Read from or write to the dataset
- Close the dataset, file dataspace, memory dataspace, and file





`herr_t H5Sselect_hyperslab (hid_t space_id, H5S_seloper_t op, const hsize_t *offset, const hsize_t *stride, const hsize_t *count, const hsize_t *block)`

<i>space_id</i>	<b>IN:</b>	Identifier of dataspace
<i>op</i>	<b>IN:</b>	Selection operator to use H5S_SELECT_SET: replace existing selection w/parameters from this call
<i>offset</i>	<b>IN:</b>	Array with starting coordinates of hyperslab
<i>stride</i>	<b>IN:</b>	Array specifying which positions along a dimension to select
<i>count</i>	<b>IN:</b>	Array specifying how many blocks to select from the dataspace, in each dimension
<i>block</i>	<b>IN:</b>	Array specifying size of element block (NULL indicates a block size of a single element in a dimension)

`herr_t H5Sselect_hyperslab` (*hid\_t space\_id, H5S\_seloper\_t op, const hsize\_t \*offset, **const hsize\_t \*stride**, const hsize\_t \*count, const hsize\_t \*block*)

- space\_id*    **IN:**    Identifier of dataspace
- op*         **IN:**    Selection operator to use  
                 H5S\_SELECT\_SET: replace existing selection  
   w/parameters from this call
- offset*     **IN:**    Array with starting coordinates of hyperslab
- stride*     **IN:**    **Array specifying which positions along a  
dimension to select**
- count*      **IN:**    Array specifying how many blocks to select from  
the dataspace, in each dimension
- block*       **IN:**    Array specifying size of element block (NULL  
indicates a block size of a single element in  
a dimension)



`herr_t H5Sselect_hyperslab` (*hid\_t space\_id, H5S\_seloper\_t op, const hsize\_t \*offset, const hsize\_t \*stride, const hsize\_t \*count, **const hsize\_t \*block***)

*space\_id*    **IN:**    Identifier of dataspace

*op*        **IN:**    Selection operator to use  
                 H5S\_SELECT\_SET: replace existing selection  
   w/parameters from this call

*offset*    **IN:**    Array with starting coordinates of hyperslab

*stride*    **IN:**    Array specifying which positions along a  
   dimension to select

*count*     **IN:**    Array specifying how many blocks to select from  
   the dataspace, in each dimension

*block*     **IN:**    **Array specifying size of element block (NULL  
   indicates a block size of a single element in  
   a dimension)**

# Hyperslab Example (1-D)

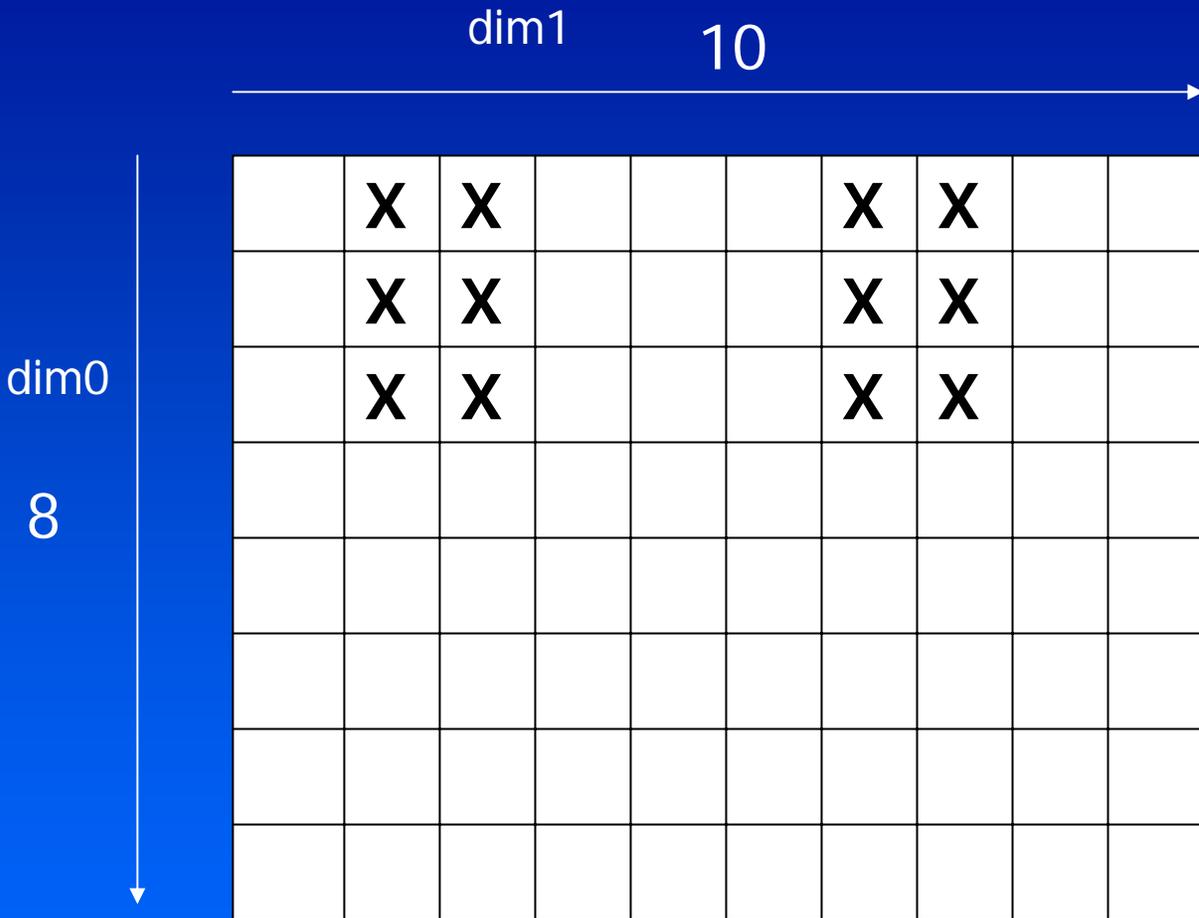


offset (0) = 1

block (0) = 1  
(or NULL)

stride (0) = 2

# Hyperslab Example



To select X's

Dataset size = {8, 10}

Offset = {0, 1}

Block size = {3, 2}

Count = {1, 2}

Stride = {4, 5}

0-based

What happens if you change Stride = {2, 5} ? (won't work)

What happens if you change Count = {2, 2} ?

# Hyperslab Example

10

8

	X	X				X	X		
	X	X				X	X		
	X	X				X	X		
	X	X				X	X		
	X	X				X	X		
	X	X				X	X		

To select X's

Dataset size= {8, 10}

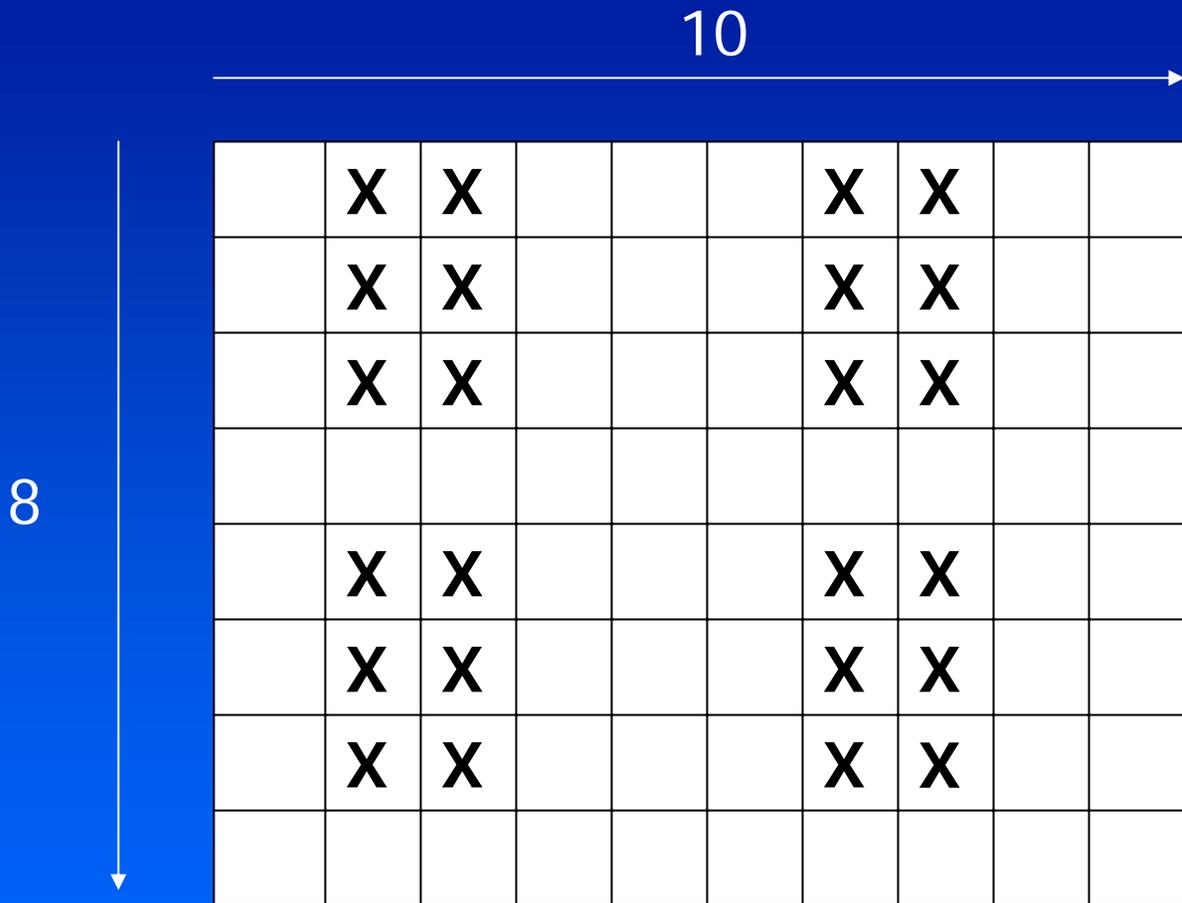
Offset= {0, 1}

Block size= {3, 2}

Count= {2, 2}

Stride= {4, 5}

# Hyperslab Example



## To select X's

Dataset size= {8, 10}

Offset= {0, 1}

Block size= {3, 2}

Count= {2, 2}

Stride= {4, 5}

What happens if you changed Block size= {1, 1} ?

# Hyperslab Example

10

8

	X					X			
	X					X			

To select X's

Dataset size= {8, 10}

Offset= {0, 1}

Block size= {1, 1}

Count= {2, 2}

Stride= {4, 5}

# Example: Selection from Dataset - C $Y = 6$

		offset = {1,2}							
			block[1]=1						count[0] = 3
X = 5	block[0]=1		X	X	X	X	X	X	
			X	X	X	X	X	X	
			X	X	X	X	X	X	

- 1 offset [0] = 1;
- 2 offset [1] = 2;
- 3 count [0] = 3;
- 4 count [1] = 4;
- 5 status = H5Sselect\_hyperslab (dataspace,  
H5S\_SELECT\_SET,offset,NULL, count, NULL);

count[1] = 4

# Set Up Memory Dataspace

```
dimsm[0] = 3;
```

```
dimsm[1] = 4;
```

```
memspace = H5Screate_simple (2, dimsm, NULL);
```

# Read/Write Using Selection

```
status = H5Dread (... , ..., memspace, dataspace, ..., ...);
```

number of elements selected in memory  
space must be the same as the number of elements  
selected in dataspace

# Individual Points Selection

herr\_t H5Sselect\_elements (*hid\_t space\_id*, *H5S\_seloper\_t op*,  
*size\_t num\_elem*, *const hsize\_t \*\*coord*)

*space\_id*    **IN:**    Identifier of the dataspace

*op*        **IN:**    Selection operator to use  
            H5S\_SELECT\_SET: replace existing selection  
            with parameters from this call

*num\_elem* **IN:**    Number of elements to be selected

*coord*    **IN:**    A 2-D array specifying the coordinates of the  
            elements being selected

`herr_t H5Sselect_elements (hid_t space_id, H5S_seloper_t op,  
size_t num_elem, const hsize_t **coord)`

*space\_id*    **IN:**    Identifier of the dataspace

*op*            **IN:**    Selection operator to use  
                 **H5S\_SELECT\_SET:** replace existing selection  
                 with parameters from this call

*num\_elem*   **IN:**    Number of elements to be selected

*coord*        **IN:**    A 2-D array specifying the coordinates of the  
                 elements being selected

`herr_t H5Sselect_elements (hid_t space_ID, H5S_seloper_t op,  
size_t num_elem, const hsize_t **coord)`

*space\_id*    **IN:**    Identifier of the dataspace

*op*        **IN:**    Selection operator to use  
H5S\_SELECT\_SET: replace existing selection  
with parameters from this call

*num\_elem* **IN:**    Number of elements to be selected

*coord*     **IN:**    A 2-D array specifying the coordinates of the  
elements being selected

`herr_t H5Sselect_elements (hid_t spaceEH5S_seloper_t op, size_t num_elem, const hsize_t **coord)`

*space\_id*    **IN:**    Identifier of the dataspace

*op*        **IN:**    Selection operator to use  
H5S\_SELECT\_SET: replace existing selection  
with parameters from this call

*num\_elem* **IN:**    Number of elements to be selected

*coord*     **IN:**    A 2-D array specifying the coordinates of the  
elements being selected

# Example

val

53	59
----	----



0	53 (0,1)	0	59 (0,3)
0	0	0	0
0	0	0	0

Writes 53 and 59 to coordinates (0,1) and (0,3) in first dataset.

## Example: C Code

```
1 hsize_t coord[2][2];
```

### Get the dataspace identifier from the file

```
2 sid = H5Dget_space (dataset1);
```

### Set the selected point positions

```
3 coord[0][0] = 0; coord[0][1] = 3;
```

```
4 coord[1][0] = 0; coord[1][1] = 1;
```

### Select the elements in the file space

```
5 ret = H5Sselect_elements (sid, H5S_SELECT_SET, 2,  
                           (const hssize_t **)coord);
```

# Memory Dataspace

```
hsize_t marray[] = {2};
```

```
...
```

```
mid1 = H5Screate_simple (1, marray, NULL); .
```

# Read/Write Using Selection

```
status = H5Dread (... , ..., memspace, dataspace, ..., ...);
```

The number of elements selected in the memory space must be the same number as is selected in the dataspace.

# HDF5 Properties

# Properties Definition

- Mechanism to control different features of the HDF5 objects
  - There are default values for these features
  - HDF5 H5P (Property List) interface allows users to modify the default features
    - At object creation time (creation properties)
    - At object access time (access or transfer properties)

# Properties Definitions

- A property list is a list of name-value pairs
- A property list is passed as an optional parameters to the HDF5 APIs
- Property lists are used/ignored by all the layers of the library, as needed

# Type of Properties

- Predefined and User defined property lists
- Predefined:
  - File creation
  - File access
  - Dataset creation
  - Dataset access

# Properties (Example)

## HDF5 File

- `H5Fcreate(..., creation_prop_id, ...)`
- Creation properties (how file is created?)
  - **Library's defaults**
    - no user's block
    - predefined sizes of offsets and addresses of the objects in the file (64-bit for DEC Alpha, 32-bit on Windows)
  - **User's settings**
    - User's block
    - 32-bit sizes on 64-bit platform
    - Control over B-trees for chunking storage (split factor)

# User's Block

- User block stores user-defined information (e.g. ASCII text to describe a file) at the beginning of the file
- h5jam – utility to add user block to HDF5 file

# Properties (Example)

## HDF5 File

- `H5Fcreate(..., access_prop_id)`
- Access properties or drivers (How is file accessed? What is the physical layout on the disk?)
  - **Library defaults**
    - STDIO Library (UNIX `fwrite`, `fread`)
  - **User's defined**
    - MPI I/O for parallel access
    - Family of files (100 Gb HDF5 represented by 50 2Gb UNIX files)
    - Size of the chunk cache

# Properties (Example)

## HDF5 Dataset

- `H5Dcreate(..., creation_prop_id)`
- Creation properties (how dataset is created)
  - **Library's defaults**
    - Storage: Contiguous
    - Compression: None
    - Space is allocated when data is first written
    - No fill value is written
  - **User's settings**
    - Storage: Compact, or chunked, or external
    - Compression
    - Fill value
    - Control over space allocation in the file for raw data
      - at creation time
      - at write time

# Properties (Example)

## HDF5 Dataset

- `H5Dwrite<read>( ..., access_prop_id )`
- Access (transfer) properties
  - **Library defaults**
    - 1MB conversion buffer
    - Error detection on read (if was set during write)
    - MPI independent I/O for parallel access
  - **User defined**
    - MPI collective I/O for parallel access
    - Size of the datatype conversion buffer
    - Control over partial I/O to improve performance

# Properties

## Programming model

- Use predefined property type
  - `H5P_FILE_CREATE`
  - `H5P_FILE_ACCESS`
  - `H5P_DATASET_CREATE`
  - `H5P_DATASET_ACCESS`
- Create new property instance
  - `H5Pcreate`
  - `H5Pcopy`
  - `H5Fget_access_plist; H5Fget_create_plist`
  - `H5Dget_create_plist`
- Modify property (see H5P APIs)
- Use property to modify object feature
- Close property when done
  - `H5Pclose`

# Properties

## Programming model

- General model of usage: get plist, set values, pass to library

```
hid_t plist = H5Pcreate(copy);  
H5Pset_foo( plist, vals);  
H5Xdo_something( xid, ..., plist);  
H5Pclose(plist);
```

# HDF5 Dataset Creation Properties and Predefined Filters

# Dataset Creation Properties

- Storage Layout
  - Contiguous (default)
  - Compact
  - Chunked
  - External
- Filters applied to raw data
  - Compression
  - Checksum
- Fill value
- Space allocation for raw data in the file

# Dataset Creation Properties

## Storage Layouts

Storage layout is important for I/O performance and size of the HDF5 files

# Storage Layout: Contiguous (default)

- Used when data will be written/read at once
- Sub-sampling can be faster than chunked
- `H5Dcreate(...,H5P_DEFAULT)`

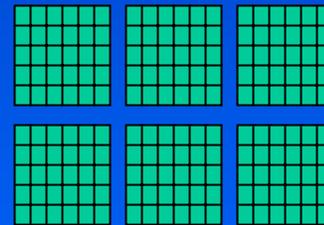
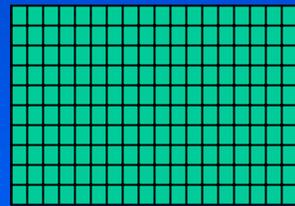
# Storage Layout: Compact

- Used for small datasets (order of O(bytes)) for better I/O
- Raw data is written/read at the time when dataset is open
- File is less fragmented

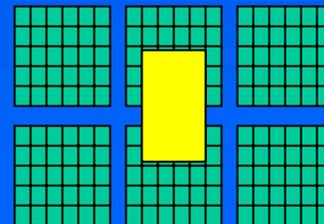
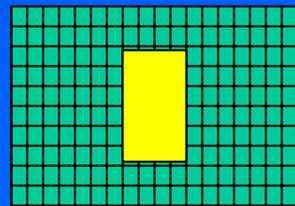
# Storage Layout: Chunked

- Chunked layout is needed for
  - Extendible datasets
  - Compression and other filters
  - To improve partial I/O for big datasets

chunked



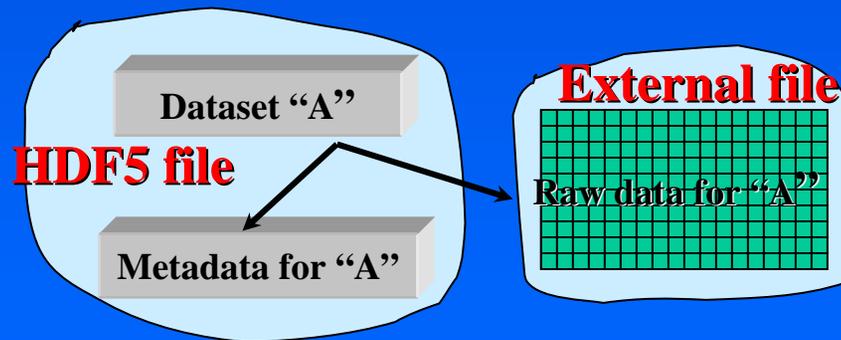
Better subsetting  
access time;  
extendible



Only two chunks will be  
written/read

# Storage Layout: External

- Dataset's raw data is stored in an *external* file
- Easy to include *existing* data into HDF5 file
- Easy to *export* raw data if application needs it
- **Disadvantage:** user has to keep track of additional files to preserve integrity of the HDF5 file



Raw data can be stored in external file

# Setting Storage Layout

```
hid_t plist = H5Pcreate (H5P_DATASET_CREATE);
```

```
Compact:    H5Pset_layout (plist, H5D_COMPACT)
```

```
Chunked:    H5Pset_chunk (plist, rank, ch_dims);
```

```
External:    H5Pset_external (plist, "raw_data.ext", offset, size);
```

```
dset_id = H5Dcreate (... , ... , ..., plist);
```

```
H5Pclose (plist);
```

# HDF5 Dataset Creation Filters

Filters are a mechanism to manipulate data while transferring it between memory and disk.

Chunks of a dataset can be arranged in a *pipeline* so that output of one filter becomes input of the next filter.

# Dataset Creation Properties

## Compression and other Pipeline Filters

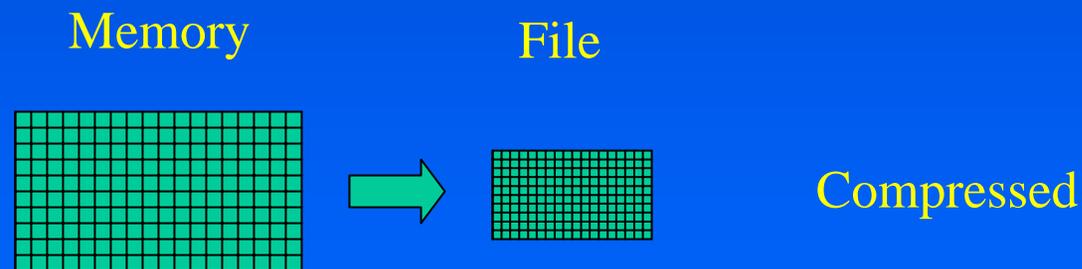
- HDF5 predefined filters (H5P interface)
  - Compression (gzip, szip)
  - Shuffling and checksum filters
- User defined filters (H5Z and H5P interfaces)
  - Example: Bzip2 compression  
<http://hdf.ncsa.uiuc.edu/HDF5/papers/papers/bzip2/>

# Compression and other Pipeline Filters (continued)

- Currently used only with *chunked* datasets
- Filters can be combined together
  - Shuffle + checksum filter + GZIP
  - Checksum filter + user define encryption filter
- Filters are called in the order they are defined on writing and in the reverse order on reading
- The order is important!
- User is responsible for “filter pipeline sanity”
  - GZIP + SZIP + shuffle doesn't make sense
  - Shuffle + SZIP does

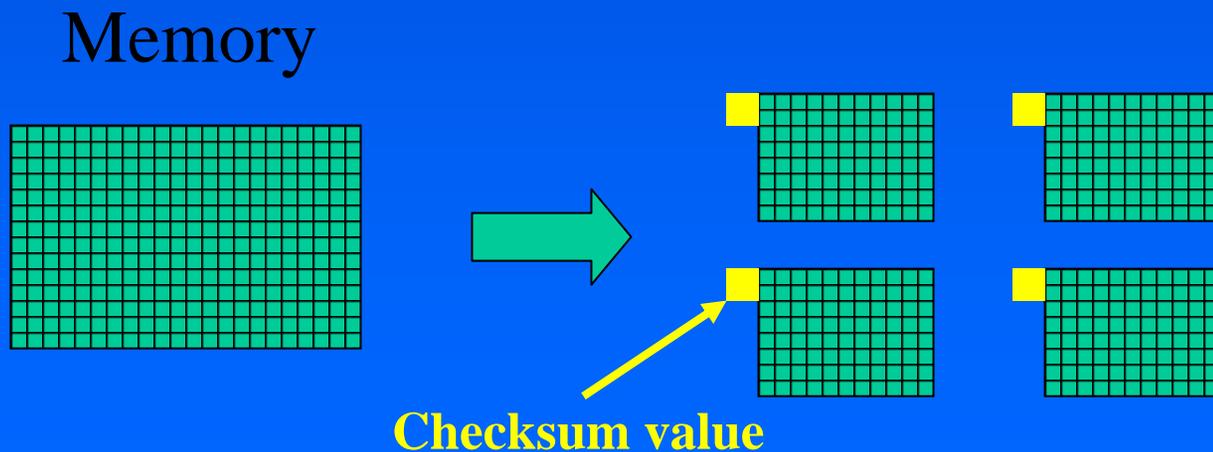
# Creating compressed Dataset

- Compression
  - Improves transmission speed
  - Improves storage efficiency
  - Requires chunking
  - May increase CPU time needed for compression



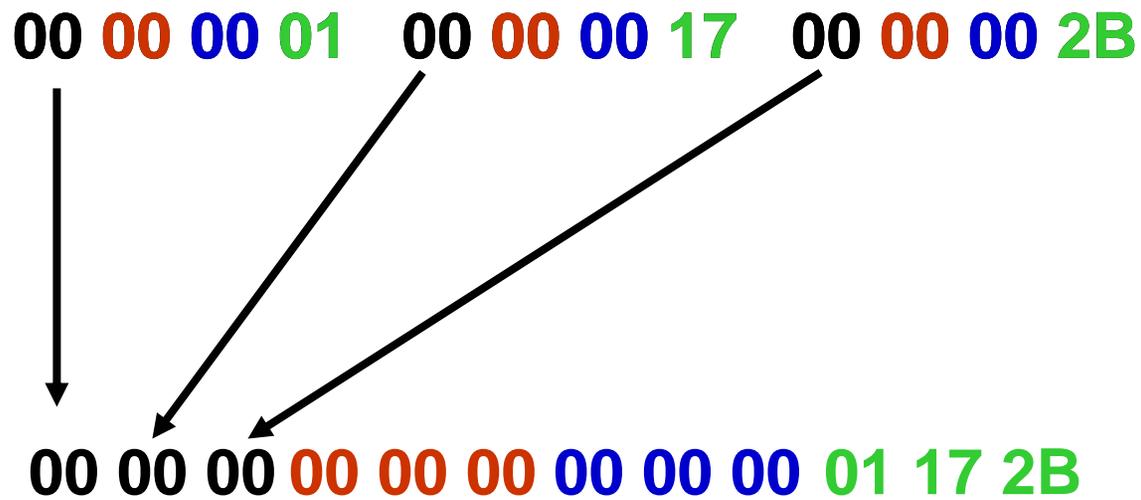
# Checksum Filter

- HDF5 includes the Fletcher32 checksum algorithm for error detection.
- It is automatically included in HDF5
- To use this filter you must add it to the filter pipeline with `H5Pset_filter`.



# Shuffling filter

- Predefined HDF5 filter
- Not a compression; change of byte order in a stream of data



## Effect of data shuffling (H5Pset\_shuffle + H5Pset\_deflate)

- Write 4-byte integer dataset 256x256x1024 (256MB)
- Using chunks of 256x16x1024 (16MB)
- Values: random integers between 0 and 255

	File size	Total time	Write Time
No Shuffle	102.9MB	671.049	629.45
Shuffle	67.34MB	83.353	78.268

Compression combined with shuffling provides

- Better compression ratio
- Better I/O performance

# Enabling Filters

```
hid_t plist = H5Pcreate (H5P_DATASET_CREATE);  
H5Pset_chunk (plist, ndims, chkdims);
```

```
GZIP Compression: H5Pset_deflate (plist, level);
```

```
SZIP Compression: H5Pset_szip (plist, options-mask, numpixels);
```

```
Checksum Filter: H5Pset_filter (plist, H5Z_FILTER_FLETCHER32,  
                                0, 0, NULL);
```

```
Shuffle Filter w/GZIP: H5Pset_shuffle(plist);  
                        H5Pset_deflate(plist, level);
```

```
dset_id = H5Dcreate (... , ... , ..., plist);  
H5Pclose (plist);
```

# User-defined Filters

# Standard Interface for User-defined Filters

- `H5Zregister` : Register filter so that HDF5 knows about it
- `H5Zunregister`: Unregister a filter
- `H5Pset_filter`: Adds a filter to the filter pipeline
- `H5Pget_filter`: Returns information about a filter in the pipeline
- `H5Zfilter_avail`: Check if filter is available

# HDF5 Dataset Access (Transfer) Properties

# Dataset Access/Transfer Properties

- Improve performance
- H5Pset\_buffer
  - Sets the size of the datatype conversion buffer during I/O (default is 1MB)
- Other functions

# File Creation Properties

**hid\_t H5Fcreate (const char \*name, unsigned flags,  
hid\_t create\_id, hid\_t access\_id)**

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

# File Creation Properties

- H5Pset\_userblock
  - User block stores user-defined information (e.g ASCII text to describe a file) at the beginning of the file
  - Sets the size of the user block
  - 512 bytes, 1024 bytes, ... ( $2^N$  for  $N > 7$ ).
- H5Pset\_sizes
  - Sets the byte size of the offsets and lengths used to address objects in the file
- Others

# File Access Properties

# File Access Properties (Performance)

- H5Pset\_cache (this function is changing in 5-1.8)
  - Sets raw data chunk parameters
  - Improper size will degrade performance
- H5Pset\_meta\_block\_size
  - Reduces the number of small objects in the file
  - Block of metadata is written in a single I/O operation (default 2K)
  - VFL driver has to set  
H5FD\_AGGREGATE\_METADATA
- H5Pset\_sieve\_buffer
  - Improves partial I/O

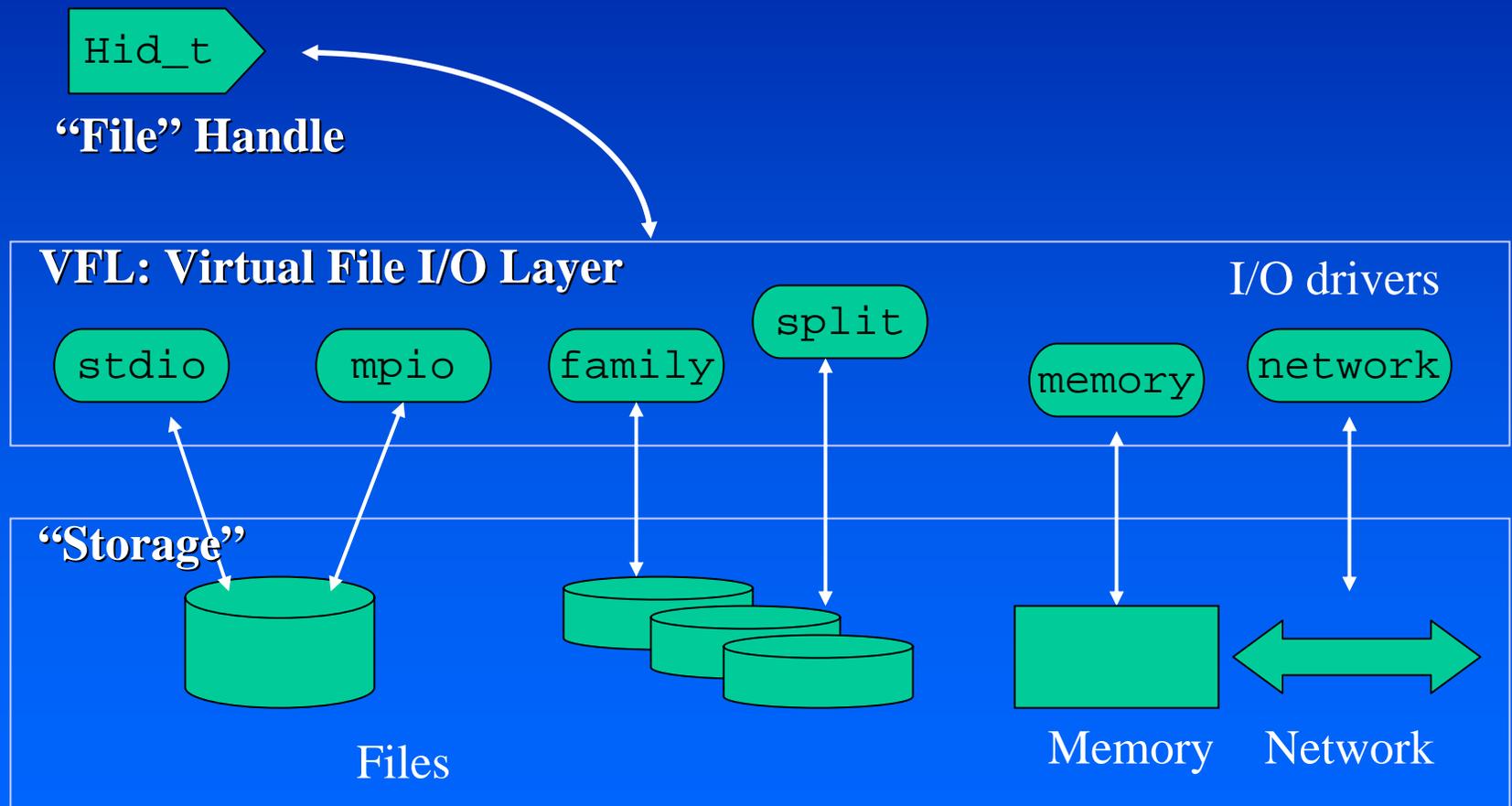
# File Access Properties (Physical storage and Usage of Low-level I/O Libraries)

VFL layer file drivers:

- Define physical storage of the HDF5 file
  - Memory driver (HDF5 file in the application's memory)
  - Stream driver (HDF5 file written to a socket)
  - Split(multi) files driver
  - Family driver
- Define low level I/O library
  - MPI I/O driver for parallel access
  - `STDIO` vs. `SEC2`

# Files needn't be files - Virtual File Layer

VFL: A public API for writing I/O drivers

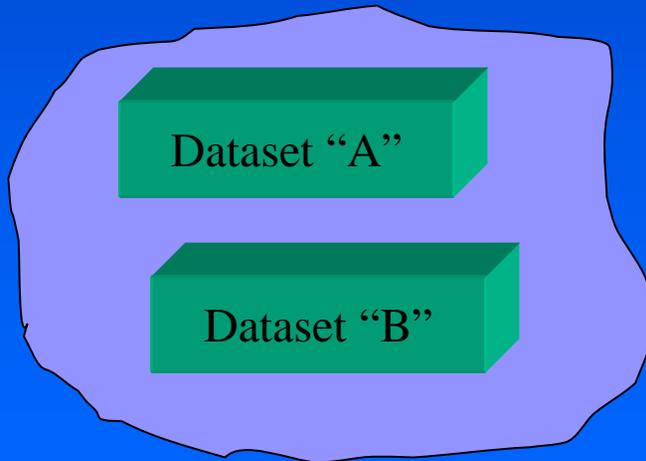


# Split Files

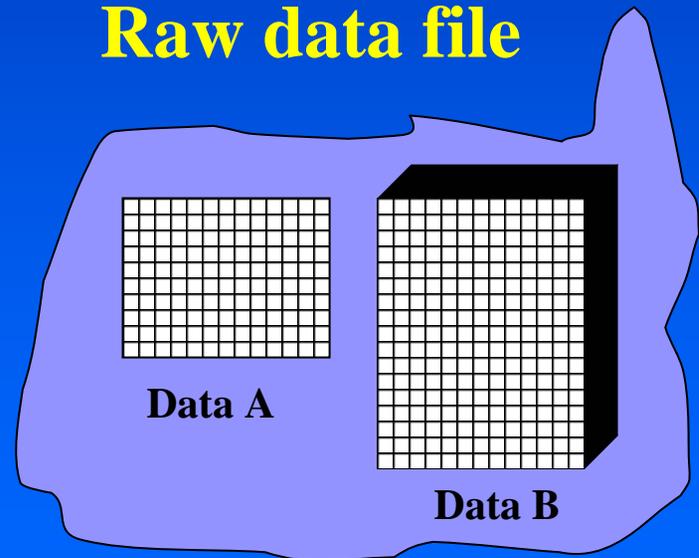
- Allows you to split metadata and data into separate files
- May reside on different file systems for better I/O
- *Disadvantage*: User has to keep track of the files

## HDF5 file

### Metadata file



### Raw data file



# File Families

- Allows you to access files larger than 2GB on file systems that don't support large files
- Any HDF5 file can be split into a family of files and vice versa
- A family member size must be a power of two

# Modifying File Access Properties

```
hid_t plist = H5Pcreate (H5P_FILE_ACCESS);
```

```
Split Files:    H5Pset_fapl_split (plist, ".met", H5P_DEFAULT,  
                                     ".dat", H5P_DEFAULT);  
File Family:    H5Pset_fapl_family (plist, family_size,  
                                     H5P_DEFAULT);
```

```
file_id = H5Fcreate (... , ... , ..., plist);  
H5Pclose (plist);
```

# HDF Information

- HDF Information Center
  - <http://hdf.ncsa.uiuc.edu/>
- HDF Help email address
  - [hdfhelp@ncsa.uiuc.edu](mailto:hdfhelp@ncsa.uiuc.edu)
- HDF users mailing list
  - [hdfnews@ncsa.uiuc.edu](mailto:hdfnews@ncsa.uiuc.edu)

# 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>).