HDF5 vs. Other Binary File Formats

Introduction to the HDF5's most powerful features

HDF5 vs. Others in a Nutshell

- Portable self-described files
- No limitation on the file size
- Fast and flexible I/O including parallel
- Internal compression

HDF5 vs. Others

- Data Model
  - General data model that enables complex data relationships and dependencies
  - Unlimited variety of datatypes
- File
  - Portable self-described binary file with random access
  - Flexible storage mechanism
    - Internal compression with unlimited number of compression methods
    - Ability to access sub-sets of compressed data
    - Ability to share data between files

HDF5 vs. Others

- Library
  - Flexible, efficient I/O
    - Partial I/O
    - Parallel I/O
    - Customized I/O
  - Data transformation during I/O
    - Type conversions
    - Custom transformations
  - Portable
    - Available for numerous platforms and compilers

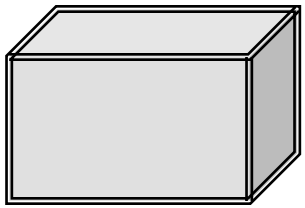
Outline

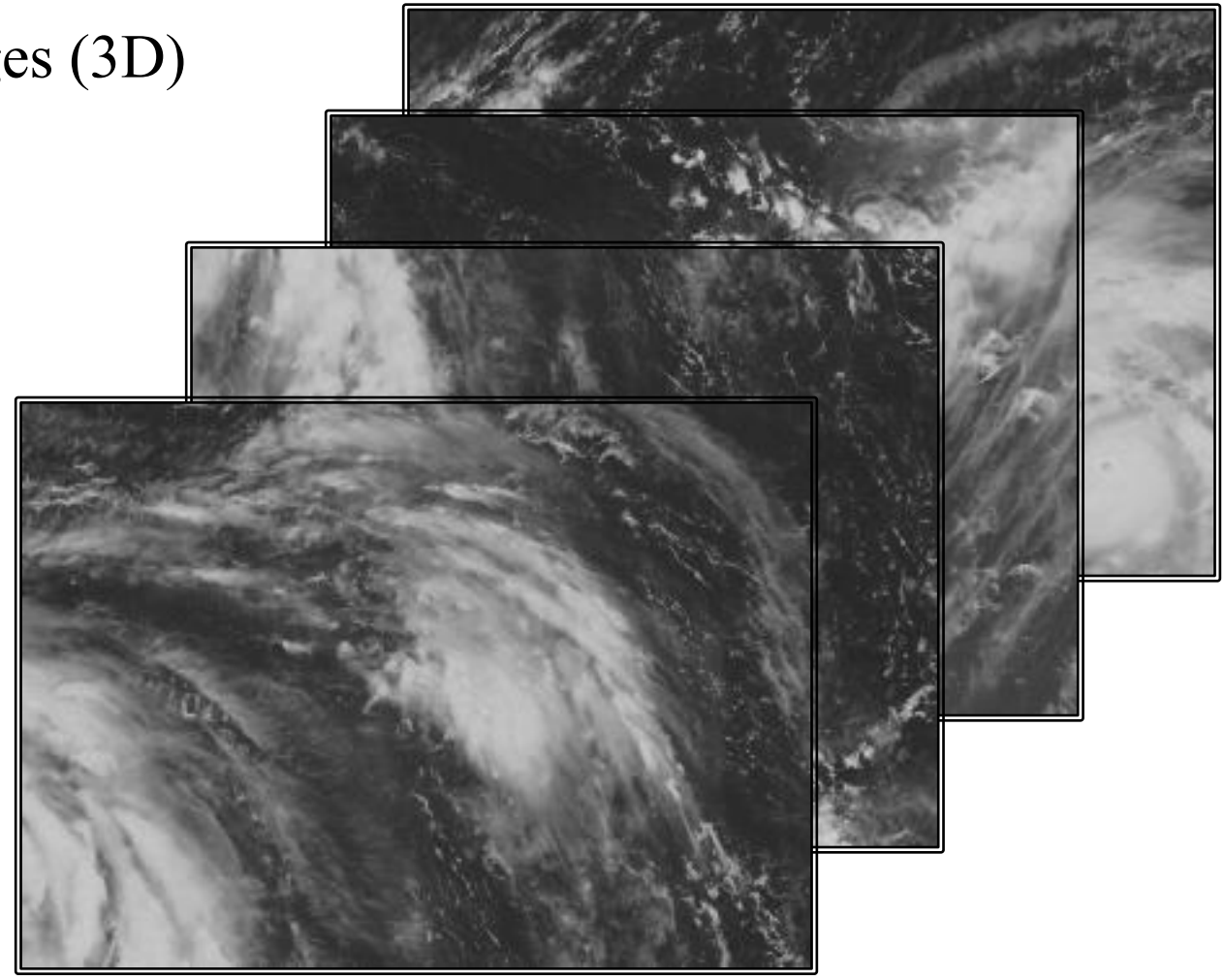
- Partial I/O in HDF5
- HDF5 datatypes
- Storage mechanisms
  - External
  - Chunking and compression

PARTIAL I/O OR WORKING WITH SUBSETS

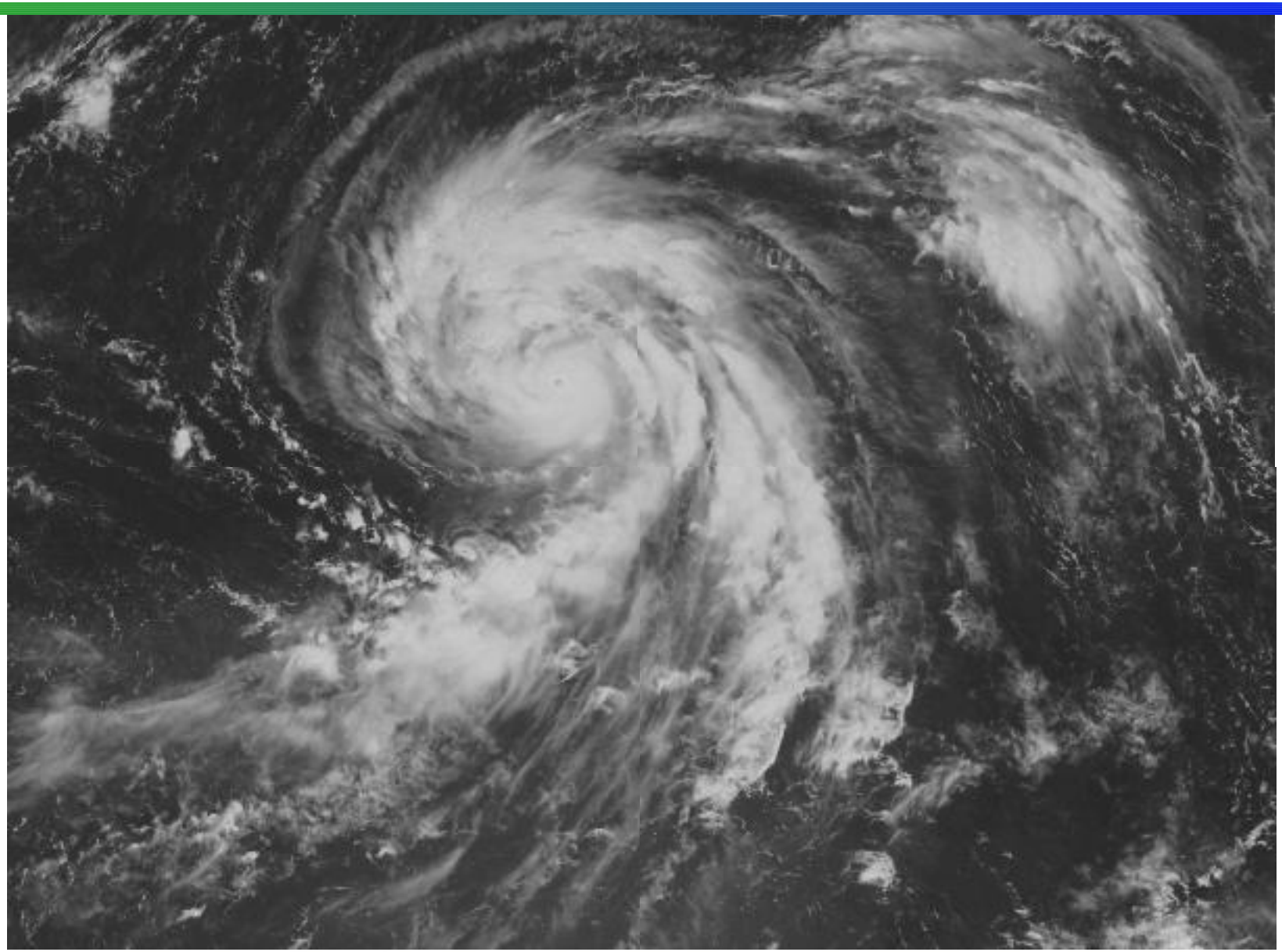
Collect data one way ….

Array of images (3D)

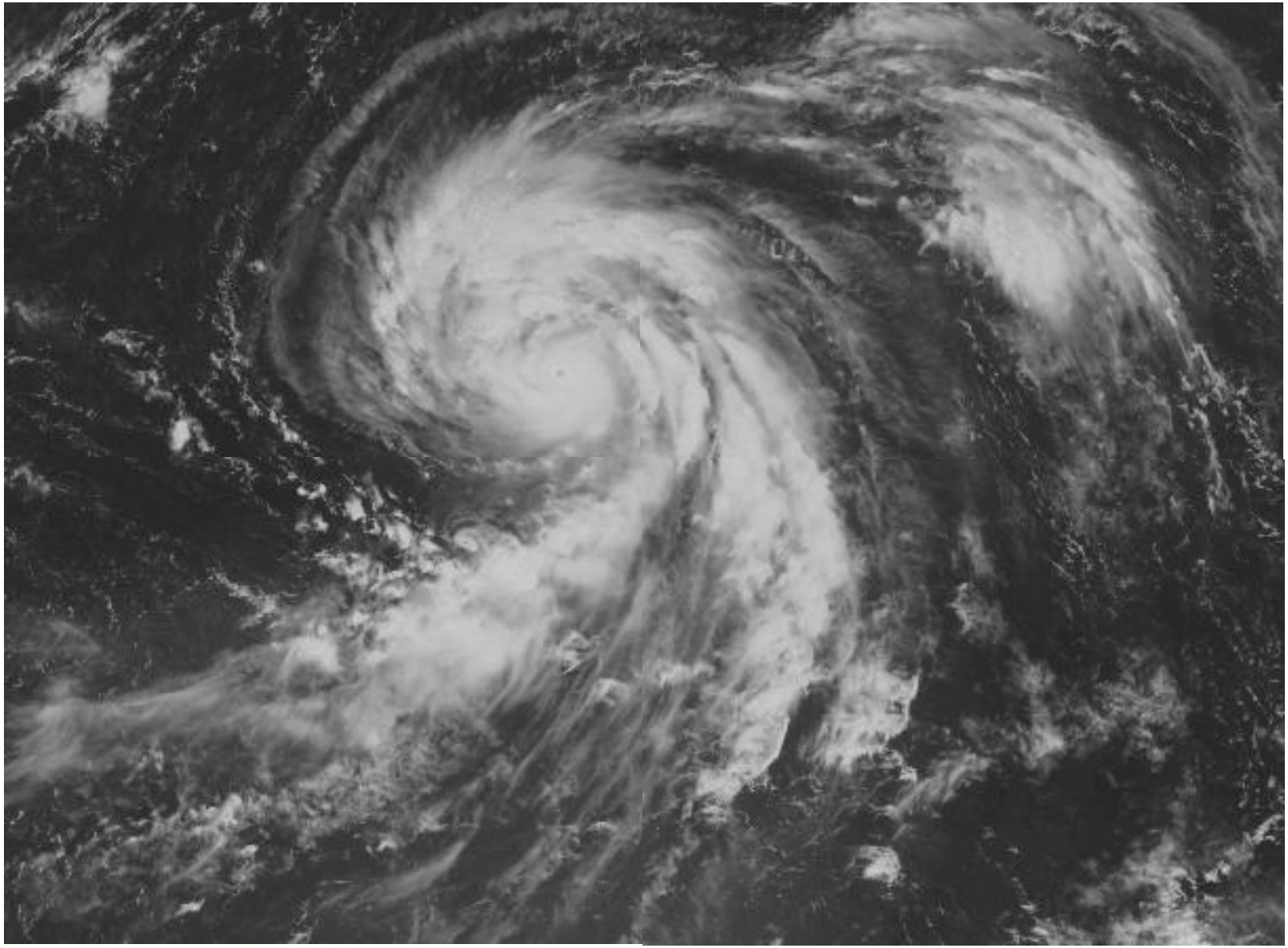




Display data another way …



Data is too big to read….



HDF5 Library Features

- HDF5 Library provides capabilities to
  - Describe subsets of data and perform write/ read operations on subsets
    - Hyperslab selections and partial I/O
  - Store descriptions of the data subsets in a file
    - Region references
  - Use efficient storage mechanism to achieve good performance while writing/reading subsets of data

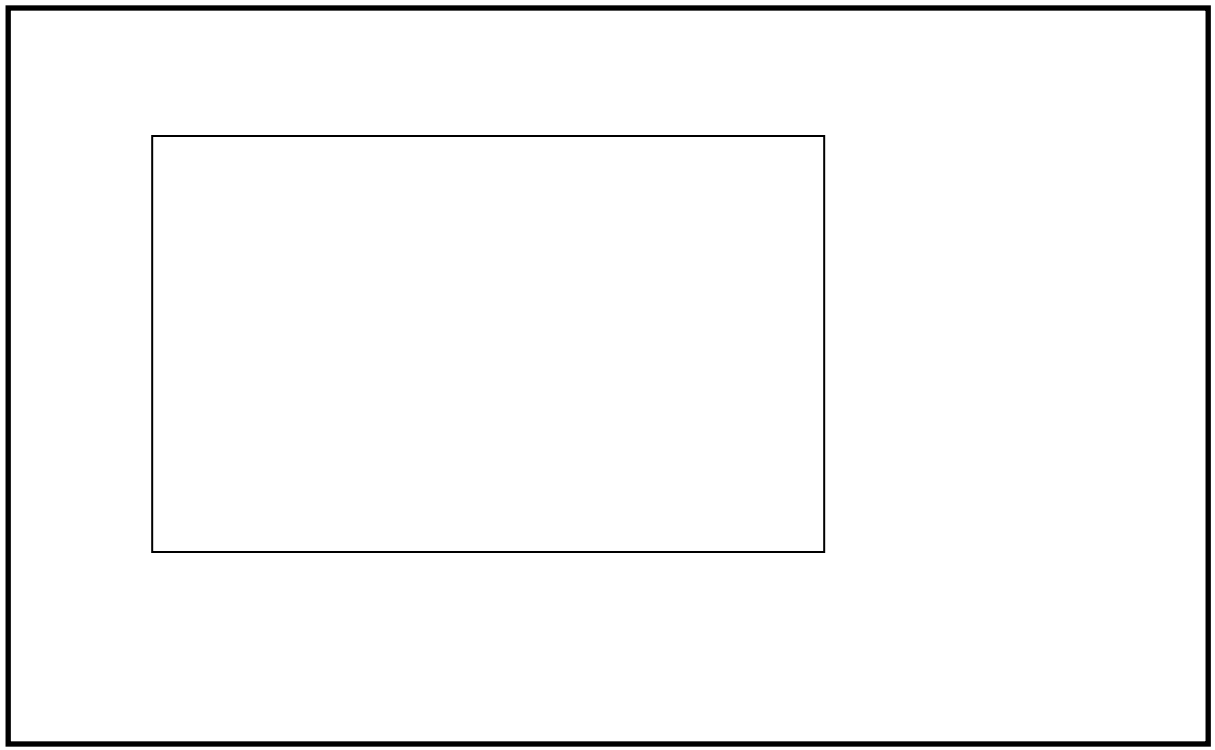
How to Describe a Subset in HDF5?

- 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 a “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

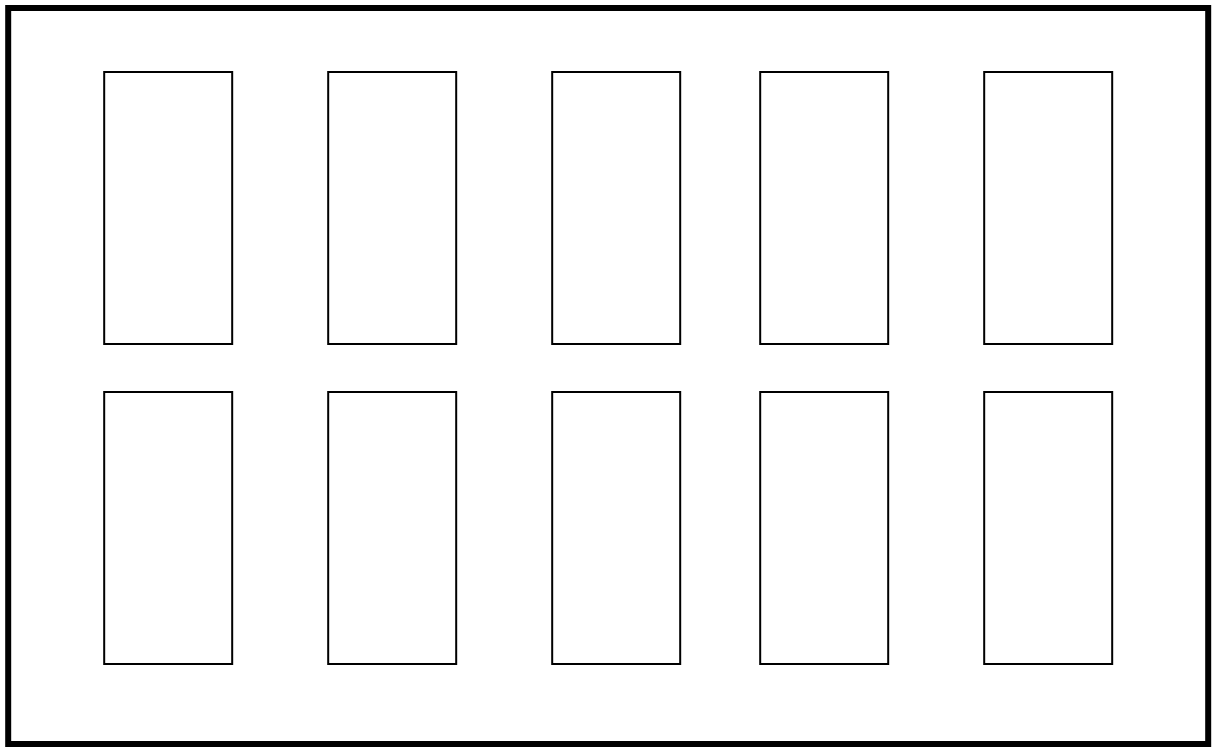
- Two types of selections
  - Hyperslab selection
    - Simple hyperslab (sub-array)
    - Regular hyperslab (patterns of sub-arrays)
    - Result of set operations on hyperslabs (union, difference, …)
  - Point selection
- Hyperslab selection is especially important for doing parallel I/O in HDF5 (See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

Simple Hyperslab



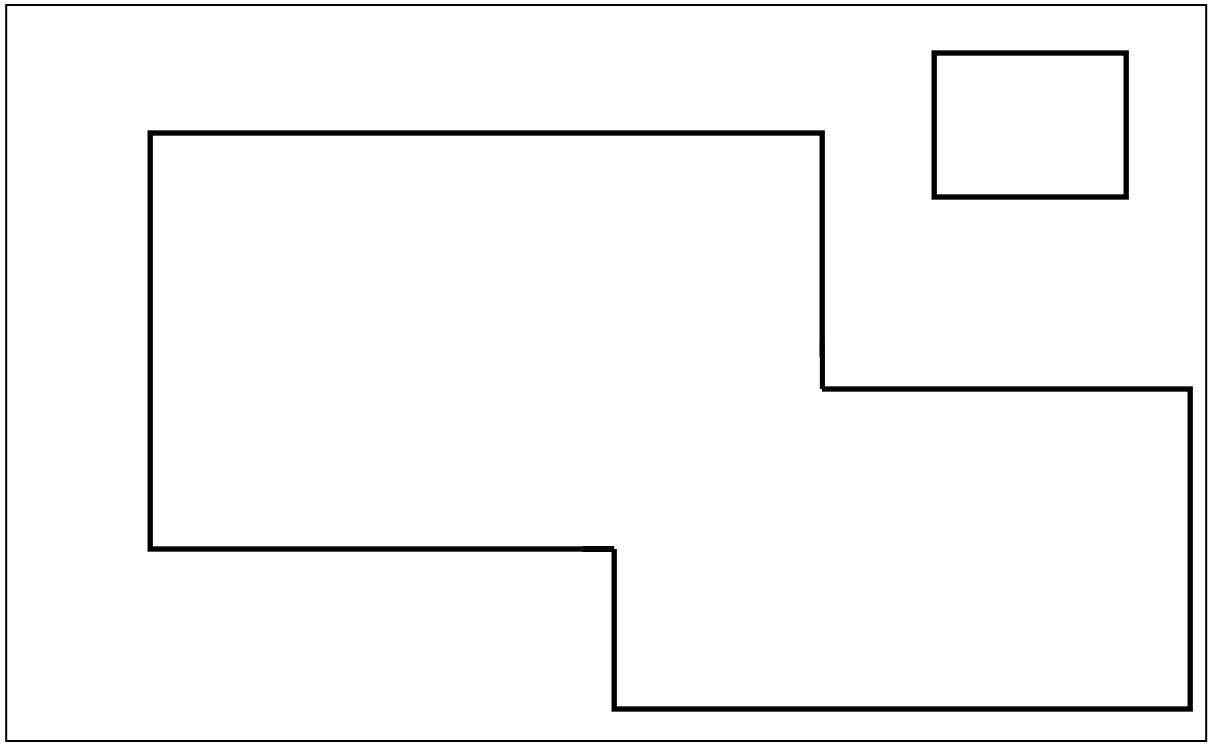
Contiguous subset or sub-array

Regular Hyperslab



Collection of regularly spaced equal size blocks

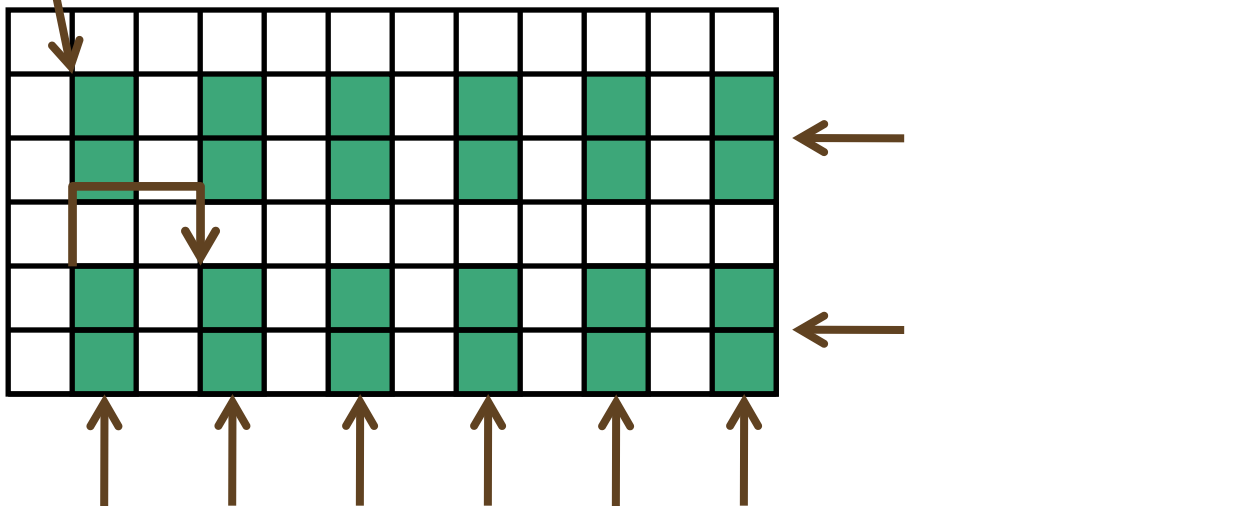
Hyperslab Selection



Result of union operation on three simple hyperslabs

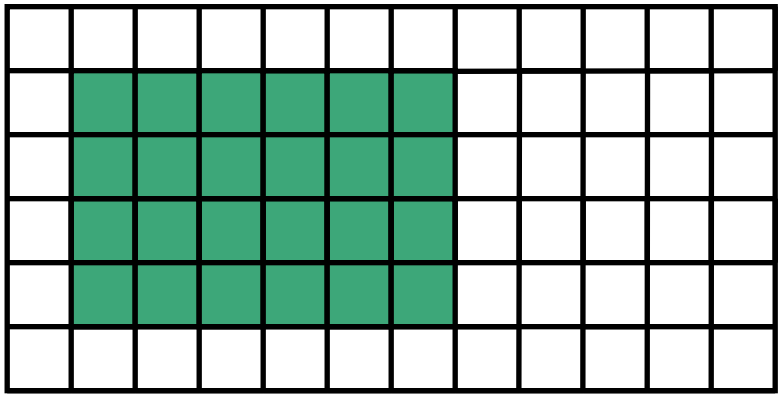
Hyperslab Description

- Offset - 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)
- *Everything is "measured" in number of elements*



Simple Hyperslab Description

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

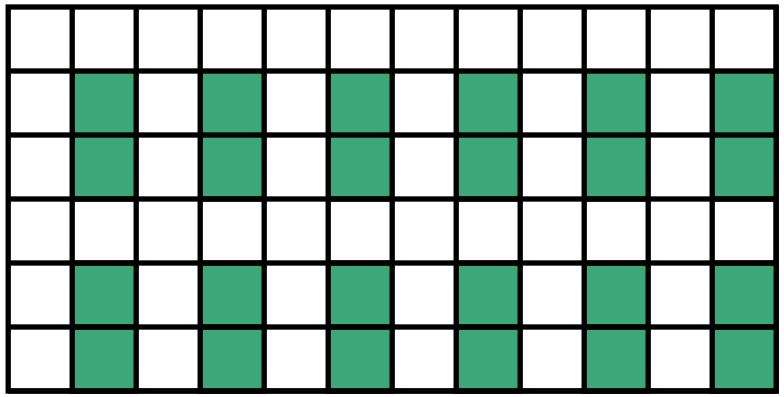


No performance penalty for one way or another

Hyperslab Selection Troubleshooting

- Selected elements have to be within the current space extent
- offset + (count-1) *stride + block =< dim
- Checking in the horizontal direction we have

  1 + (6-1) * 2 + 1 = 12



dim = (6, 12)
offset = (1, 1)
stride = (3, 2)
count = (2, 6)
block = (2, 1)

Example : Reading Two Rows

Data in a file
4x6 matrix

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

Buffer in memory
1-dim array of length 14

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Example: Reading Two Rows

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

```
start  = {1,0}
count  = {2,6}
block  = {1,1}
stride = {1,1}
```

```
filespace = H5Dget_space (dataset);
H5Sselect_hyperslab (filespace, H5S_SELECT_SET,
                     start, NULL, count, NULL)
```

Example: Reading Two Rows

```
start[1] = {1}
count[1] = {12}
dim[1]   = {14}
```

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

```
memspace = H5Screate_simple(1, dim, NULL);
H5Sselect_hyperslab (memspace, H5S_SELECT_SET,
                     start, NULL, count, NULL)
```

Example: Reading Two Rows

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

```
H5Dread (…, …, memspace, filespace, …, …);
```

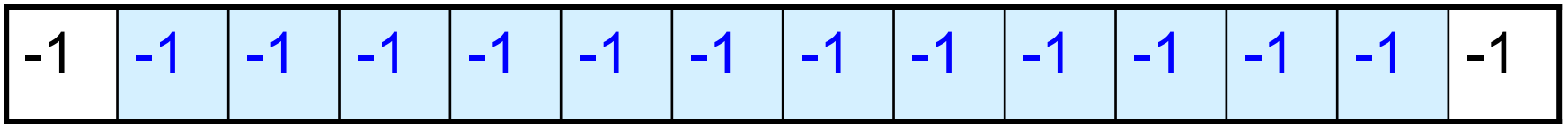
-1	7	8	9	10	11	12	13	14	15	16	17	18	-1

Things to Remember

- *Number* of elements selected in a file and in a memory buffer *must be the same*
  - `H5Sget_select_npoints` returns number of selected elements in a hyperslab selection
- HDF5 partial I/O is tuned to move data between selections that have the same dimensionality
- *Allocate a buffer of an appropriate size* when reading data; use `H5Tget_native_type` and `H5Tget_size` to get the correct size of the data element in memory.

Example: Reading Two Rows

```
start[1] = {1}
count[1] = {12}
dim[1]   = {14}
```




```
memspace = H5Screate_simple(1, dim, NULL);
H5Sselect_hyperslab (memspace, H5S_SELECT_SET,
                     start, NULL, count, NULL)
```

Example:h5ex_d_hyper.c

```
HDF5 "h5ex_d_hyper.h5" {
GROUP "/" {
   DATASET "DS1" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SIMPLE { ( 6, 8 ) / ( 6, 8 ) }
      DATA {
      (0,0): 0, 1, 0, 0, 1, 0, 0, 1,
      (1,0): 1, 1, 0, 1, 1, 0, 1, 1,
      (2,0): 0, 0, 0, 0, 0, 0, 0, 0,
      (3,0): 0, 1, 0, 0, 1, 0, 0, 1,
      (4,0): 1, 1, 0, 1, 1, 0, 1, 1,
      (5,0): 0, 0, 0, 0, 0, 0, 0, 0
      }
   }
}
}
```

HDF5 DATATYPES

An HDF5 Datatype is…

- A description of dataset element type
- Grouped into “classes”:
  - Atomic – integers, floating-point values
  - Enumerated
  - Compound – like C structs
  - Array
  - Opaque
  - References
    - Object – similar to soft link
    - Region – similar to soft link to dataset + selection
  - Variable-length
    - Strings – fixed and variable-length
    - Sequences – similar to Standard C++ vector class

HDF5 Datatypes

- HDF5 has a rich set of pre-defined datatypes and supports the creation of an unlimited variety of complex user-defined datatypes.
- Self-describing:
  - Datatype definitions are stored in the HDF5 file *with* the data.
  - Datatype definitions include information such as *byte order (endianness), size, and floating point representation* to fully describe how the data is stored and to insure portability across platforms.

HDF5 Datatypes (cont)

- Self-describing:
  - Datatype definitions are stored in the HDF5 file *with* the data.
  - Datatype definitions include information such as *byte order (endianness), size, and floating point representation* to fully describe how the data is stored and to insure portability across platforms.

Datatype Conversion

- Datatypes that are compatible, but not identical are converted automatically when I/O is performed
- Compatible datatypes:
  - All atomic datatypes are compatible
  - Identically structured array, variable-length and compound datatypes whose base type or fields are compatible
  - Enumerated datatype values on a “by name” basis
- Make datatypes identical for best performance

Datatypes Used in JPSS files

- In h5dump output search for “DATATYPE” keyword; those correspond to (…)

```
DATATYPE  H5T_STRING      C string
DATATYPE  H5T_STD_U8BE    unsigned char
DATATYPE  H5T_STD_U16BE   short (big-endian)
DATATYPE  H5T_STD_I32BE   int (big-endian)
DATATYPE  H5T_IEEE_F32BE  float (big-endian)
DATATYPE  H5T_STD_U64BE   unsigned long long
DATATYPE  H5T_REFERENCE { H5T_STD_REF_OBJECT }
DATATYPE  H5T_REFERENCE { H5T_STD_REF_DSETREG }
```

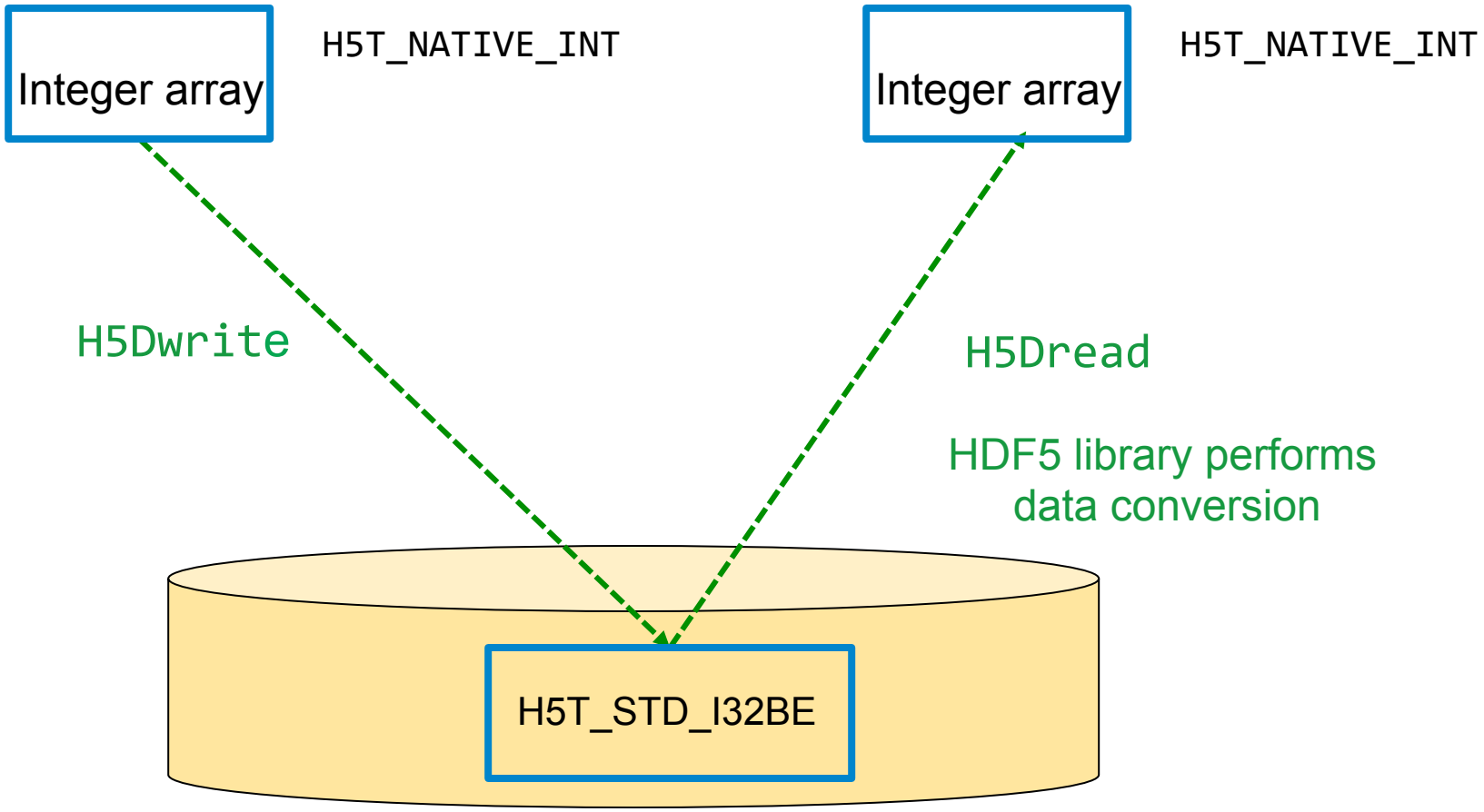
Native types from each language are mapped to HDF5 pre-defined datatypes

https://www.hdfgroup.org/HDF5/doc/RM/PredefDTypes.html

Datatype Conversion Example

Array of integers on AIX platform
Native integer is big-endian, 4 bytes

Array of integers on x86_64 platform
Native integer is little-endian, 4 bytes



Datatype Conversion

Datatype of data in the file

```
dataset = H5Dcreate(file, DATASETNAME, H5T_STD_I32LE,
                    space, H5P_DEFAULT, H5P_DEFAULT);
```

Datatype of data in memory buffer

```
H5Dwrite(dataset, H5T_NATIVE_INT, H5S_ALL, H5S_ALL,
                  H5P_DEFAULT, buf);
```

By specifying 4-bytes little-endian in the file, one will avoid data conversion when reading on little-endian machines

STORING STRINGS

Storing Strings in HDF5

- Array of characters (Array datatype or extra dimension in dataset)
  - Quick access to each character
  - Extra work to access and interpret each string
- Fixed length

```
string_id = H5Tcopy(H5T_C_S1);
H5Tset_size(string_id, size);
```

  - Wasted space in shorter strings
  - Can be compressed if used in datasets
- Variable length

```
string_id = H5Tcopy(H5T_C_S1);
H5Tset_size(string_id, H5T_VARIABLE);
```

  - Overhead as for all VL datatypes
  - Compression will not be applied to actual data in datasets

Storing Strings in HDF5

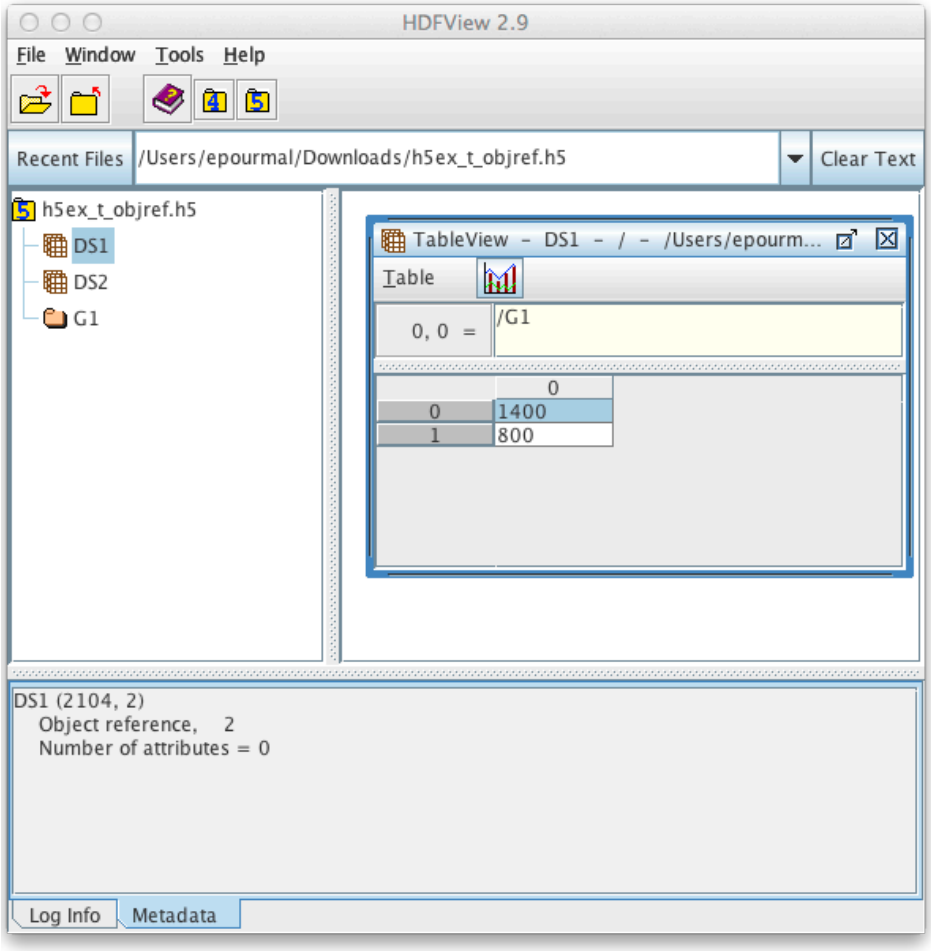
- Store the data in two datasets:
  - 1D char dataset stores concatenated strings.
  - 2D dataset stores (start, end) indices of each string in the first dataset.
    - Essentially a "hand-constructed" version of how we store variable length data.
    - Since the data are stored in datasets, they can be chunked, compressed, etc.
    - Slower than other access methods.
- When using attribute, use VL string H5_SCALAR dataspace instead H5S_SIMPLE to save some programming effort

USING DATATYPE TO REFERENCE DATA

Reference Datatypes

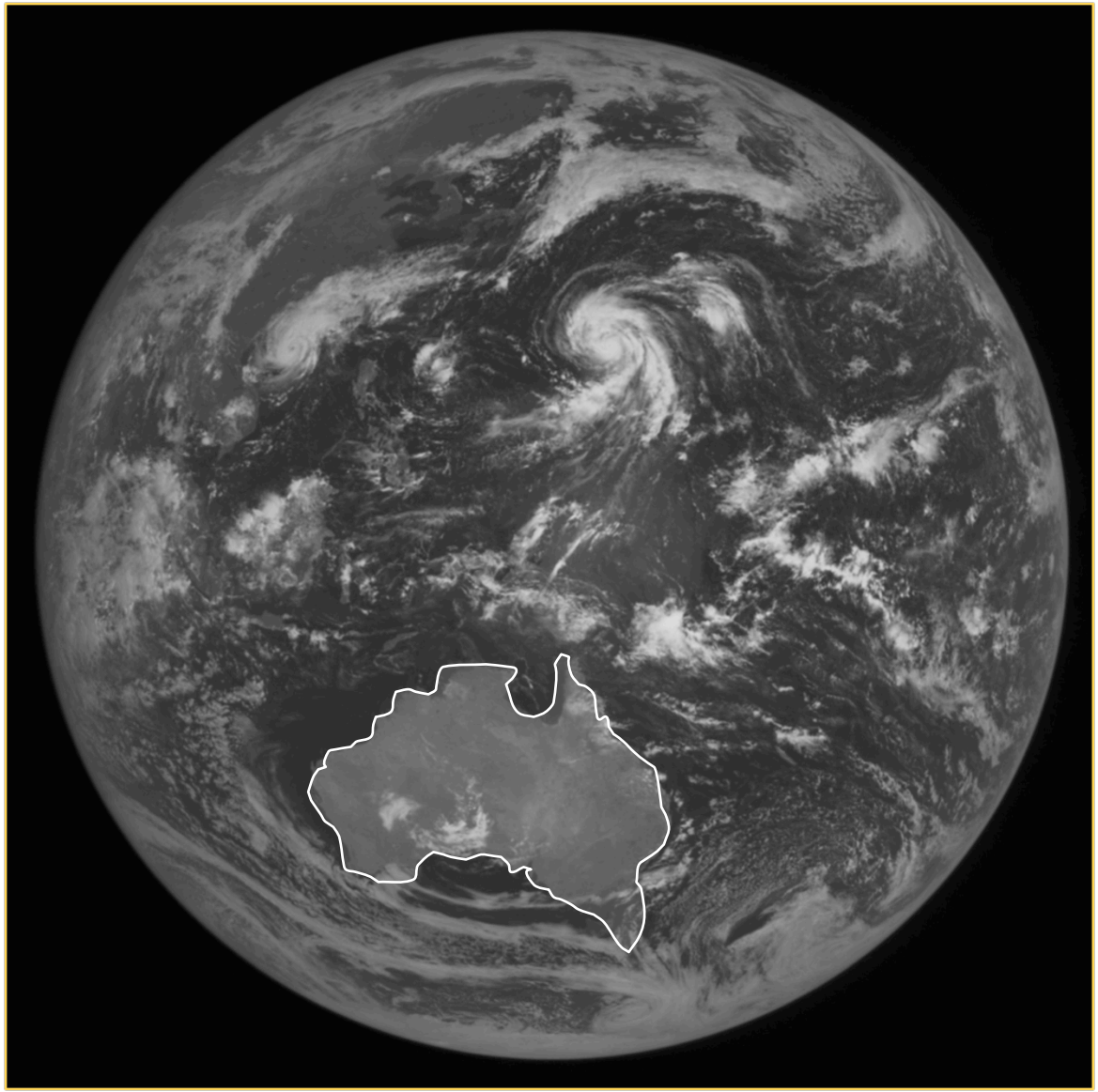
- Object Reference
  - Pointer to an object in a file
  - Predefined datatype H5T_STD_REF_OBJ
- Dataset Region Reference
  - Pointer to a dataset + dataspace selection
  - Predefined datatype H5T_STD_REF_DSETREG

Reference to Dataset

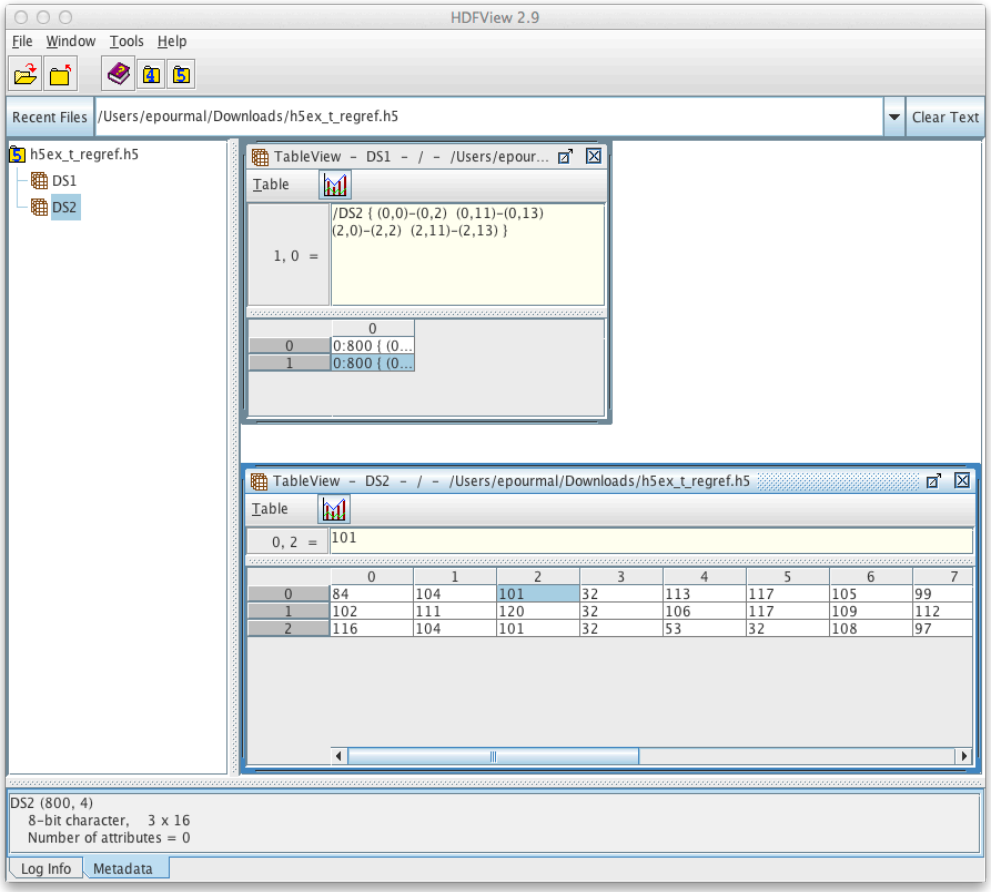



```
h5dump h5ex_t_objref.h5
HDF5 "h5ex_t_objref.h5" {
GROUP "/" {
   DATASET "DS1" {
      DATATYPE  H5T_REFERENCE { H5T_STD_REF_OBJECT }
      DATASPACE  SIMPLE { ( 2 ) / ( 2 ) }
      DATA {
      (0): GROUP 1400 /G1 , DATASET 800 /DS2
      }
   }
   DATASET "DS2" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  NULL
      DATA {
      }
   }
   GROUP "G1" {
   }
}
}
```

Saving Selected Region in a File



Reference to Regions




```
HDF5 "h5ex_t_regref.h5" {
GROUP "/" {
  DATASET "DS1" {
    DATATYPE  H5T_REFERENCE { H5T_STD_REF_DSETREG }
    DATASPACE  SIMPLE { ( 2 ) / ( 2 ) }
    DATA {
      DATASET /DS2 {(0,1), (2,11), (1,0), (2,4)},
      DATASET /DS2 {(0,0)-(0,2), (0,11)-(0,13), (2,0)-(2,2), (2,11)-(2,13)}
    }
  }
  DATASET "DS2" {
    DATATYPE  H5T_STD_I8LE
    DATASPACE  SIMPLE { ( 3, 16 ) / ( 3, 16 ) }
    DATA {
    (0,0): 84, 104, 101, 32, 113, 117, 105, 99, 107, 32, 98, 114, 111, 119,
    (0,14): 110, 0,
    (1,0): 102, 111, 120, 32, 106, 117, 109, 112, 115, 32, 111, 118, 101,
    (1,13): 114, 32, 0,
    (2,0): 116, 104, 101, 32, 53, 32, 108, 97, 122, 121, 32, 100, 111, 103,
    (2,14): 115, 0
    }
  }
}
}
```

Example: h5ex_t_objref.c

```
HDF5 "h5ex_t_objref.h5" {
GROUP "/" {
   DATASET "DS1" {
      DATATYPE  H5T_REFERENCE
{ H5T_STD_REF_OBJECT }
      DATASPACE  SIMPLE { ( 2 ) / ( 2 ) }
      DATA {
      (0): GROUP 1400 /G1 , DATASET 800 /DS2
      }
   }
   DATASET "DS2" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  NULL
      DATA {
      }
   }
   GROUP "G1" {
   }
}
}
```

Example: h5ex_t_objref.c

```
HDF5 "h5ex_t_regref.h5" {
GROUP "/" {
   DATASET "DS1" {
      DATATYPE  H5T_REFERENCE { H5T_STD_REF_DSETREG }
      DATASPACE  SIMPLE { ( 2 ) / ( 2 ) }
      DATA {
         DATASET /DS2 {(0,1), (2,11), (1,0), (2,4)},
         DATASET /DS2 {(0,0)-(0,2), (0,11)-(0,13), (2,0)-(2,2), (2,11)-(2,13)}
      }
   }
   DATASET "DS2" {
      DATATYPE  H5T_STD_I8LE
      DATASPACE  SIMPLE { ( 3, 16 ) / ( 3, 16 ) }
      DATA {
      (0,0): 84, 104, 101, 32, 113, 117, 105, 99, 107, 32, 98, 114, 111, 119, 110, 0,
      (1,0): 102, 111, 120, 32, 106, 117, 109, 112, 115, 32, 111, 118, 101, 114, 32, 0,
      (2,0): 116, 104, 101, 32, 53, 32, 108, 97, 122, 121, 32, 100, 111, 103, 115, 0
      }
   }
}
}
```

Object References in JPSS file

```
h5dump -d /Data_Products/VIIRS-M16-SDR/VIIRS-M16-SDR_Aggr /mnt/hdf/JPSS/CLASS/122293334/
SVM16_npp_d20120424_t0302382_e0304023_b02536_c20120502162515786022_noaa_ops.h5
HDF5 "/mnt/hdf/JPSS/CLASS/122293334/
SVM16_npp_d20120424_t0302382_e0304023_b02536_c20120502162515786022_noaa_ops.h5" {
DATASET "/Data_Products/VIIRS-M16-SDR/VIIRS-M16-SDR_Aggr" {
   DATATYPE  H5T_REFERENCE { H5T_STD_REF_OBJECT }
   DATASPACE  SIMPLE { ( 16 ) / ( H5S_UNLIMITED ) }
   DATA {
   (0): DATASET 7152 /All_Data/VIIRS-M16-SDR_All/Radiance ,
   (1): DATASET 4925568 /All_Data/VIIRS-M16-SDR_All/BrightnessTemperature ,
   (2): DATASET 9843656 /All_Data/VIIRS-M16-SDR_All/ModeScan ,
   (3): DATASET 9846024 /All_Data/VIIRS-M16-SDR_All/ModeGran ,
   (4): DATASET 9848568 /All_Data/VIIRS-M16-SDR_All/PadByte1 ,
   (5): DATASET 9850936 /All_Data/VIIRS-M16-SDR_All/NumberOfScans ,
   (6): DATASET 9853304 /All_Data/VIIRS-M16-SDR_All/NumberOfMissingPkts ,
   (7): DATASET 9855672 /All_Data/VIIRS-M16-SDR_All/NumberOfBadChecksums ,
   (8): DATASET 9858040 /All_Data/VIIRS-M16-SDR_All/NumberOfDiscardedPkts ,
   (9): DATASET 9861088 /All_Data/VIIRS-M16-SDR_All/QF1_VIIRSMBANDSDR ,
   (10): DATASET 12321576 /All_Data/VIIRS-M16-SDR_All/QF2_SCAN_SDR ,
   (11): DATASET 12323944 /All_Data/VIIRS-M16-SDR_All/QF3_SCAN_RDR ,
   (12): DATASET 12326312 /All_Data/VIIRS-M16-SDR_All/QF4_SCAN_SDR ,
   (13): DATASET 12328680 /All_Data/VIIRS-M16-SDR_All/QF5_GRAN_BADDETECTOR ,
   (14): DATASET 12331376 /All_Data/VIIRS-M16-SDR_All/RadianceFactors ,
   (15): DATASET 12333744 /All_Data/VIIRS-M16-SDR_All/BrightnessTemperatureFactors
   }
```

Region References in JPSS file

```
HDF5 "/mnt/hdf/JPSS/CLASS/GMODO-SVM01/GMODO-
SVM01_npp_d20130506_t1827179_e1832583_b07894_c20150408194443913666_XXXX_XXX.h5" {
DATASET "/Data_Products/VIIRS-MOD-GEO/VIIRS-MOD-GEO_Gran_2" {
   DATATYPE  H5T_REFERENCE { H5T_STD_REF_DSETREG }
   DATASPACE  SIMPLE { ( 22 ) / ( H5S_UNLIMITED ) }
   DATA {
      DATASET /All_Data/VIIRS-MOD-GEO_All/StartTime {(96)-(143)},
      DATASET /All_Data/VIIRS-MOD-GEO_All/MidTime {(96)-(143)},
      DATASET /All_Data/VIIRS-MOD-GEO_All/Latitude {(1536,0)-(2303,3199)},
      DATASET /All_Data/VIIRS-MOD-GEO_All/Longitude {(1536,0)-(2303,3199)},
      DATASET /All_Data/VIIRS-MOD-GEO_All/SolarZenithAngle {(1536,0)-(2303,3199)},
      DATASET /All_Data/VIIRS-MOD-GEO_All/SolarAzimuthAngle {(1536,0)-(2303,3199)},
      DATASET /All_Data/VIIRS-MOD-GEO_All/SatelliteZenithAngle {(1536,0)-(2303,3199)},
      DATASET /All_Data/VIIRS-MOD-GEO_All/SatelliteAzimuthAngle {(1536,0)-(2303,3199)},
      DATASET /All_Data/VIIRS-MOD-GEO_All/Height {(1536,0)-(2303,3199)},
      DATASET /All_Data/VIIRS-MOD-GEO_All/SatelliteRange {(1536,0)-(2303,3199)},
      DATASET /All_Data/VIIRS-MOD-GEO_All/SCPosition {(96,0)-(143,2)},
      DATASET /All_Data/VIIRS-MOD-GEO_All/SCVelocity {(96,0)-(143,2)},
      DATASET /All_Data/VIIRS-MOD-GEO_All/SCAttitude {(96,0)-(143,2)},
      DATASET /All_Data/VIIRS-MOD-GEO_All/SCSolarZenithAngle {(96)-(143)},
      DATASET /All_Data/VIIRS-MOD-GEO_All/SCSolarAzimuthAngle {(96)-(143)},
      …
}
```

STORAGE

HDF5 Dataset

Metadata

Dataspace

Rank

3

Dimensions

Dim_1 = 4

Dim_2 = 5

Dim_3 = 7

Datatype

IEEE 32-bit float

Storage info

Chunked

Compressed

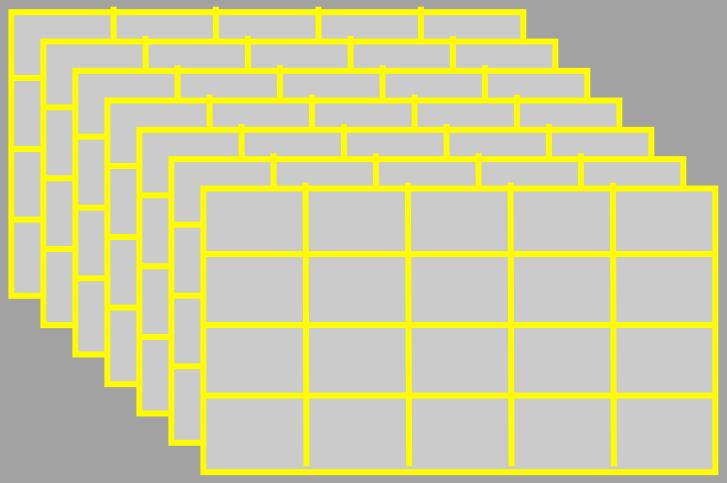
Attributes

Time = 32.4

Pressure = 987

Temp = 56

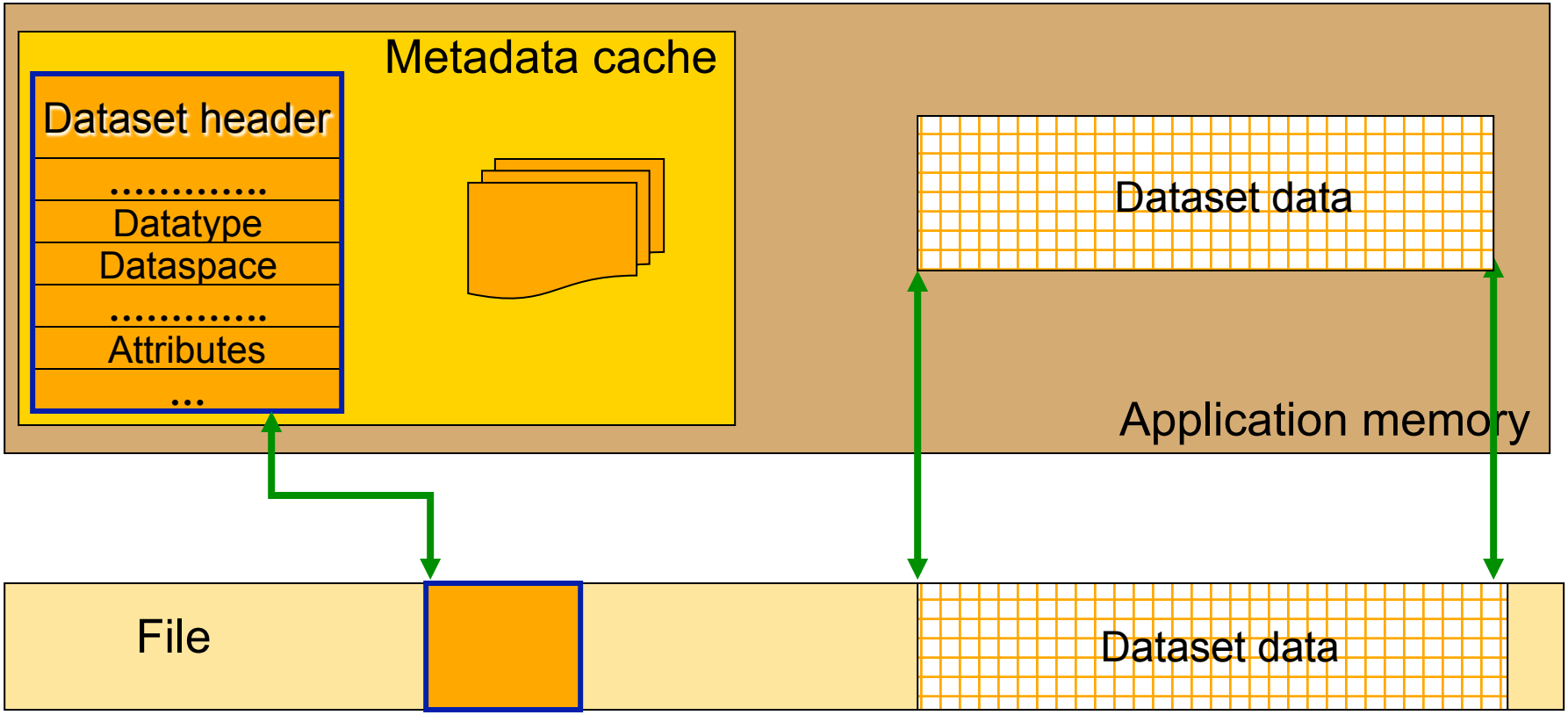
Dataset data



CONTIGUOUS STORAGE

Contiguous storage layout

- Data stored in one contiguous block in HDF5 file



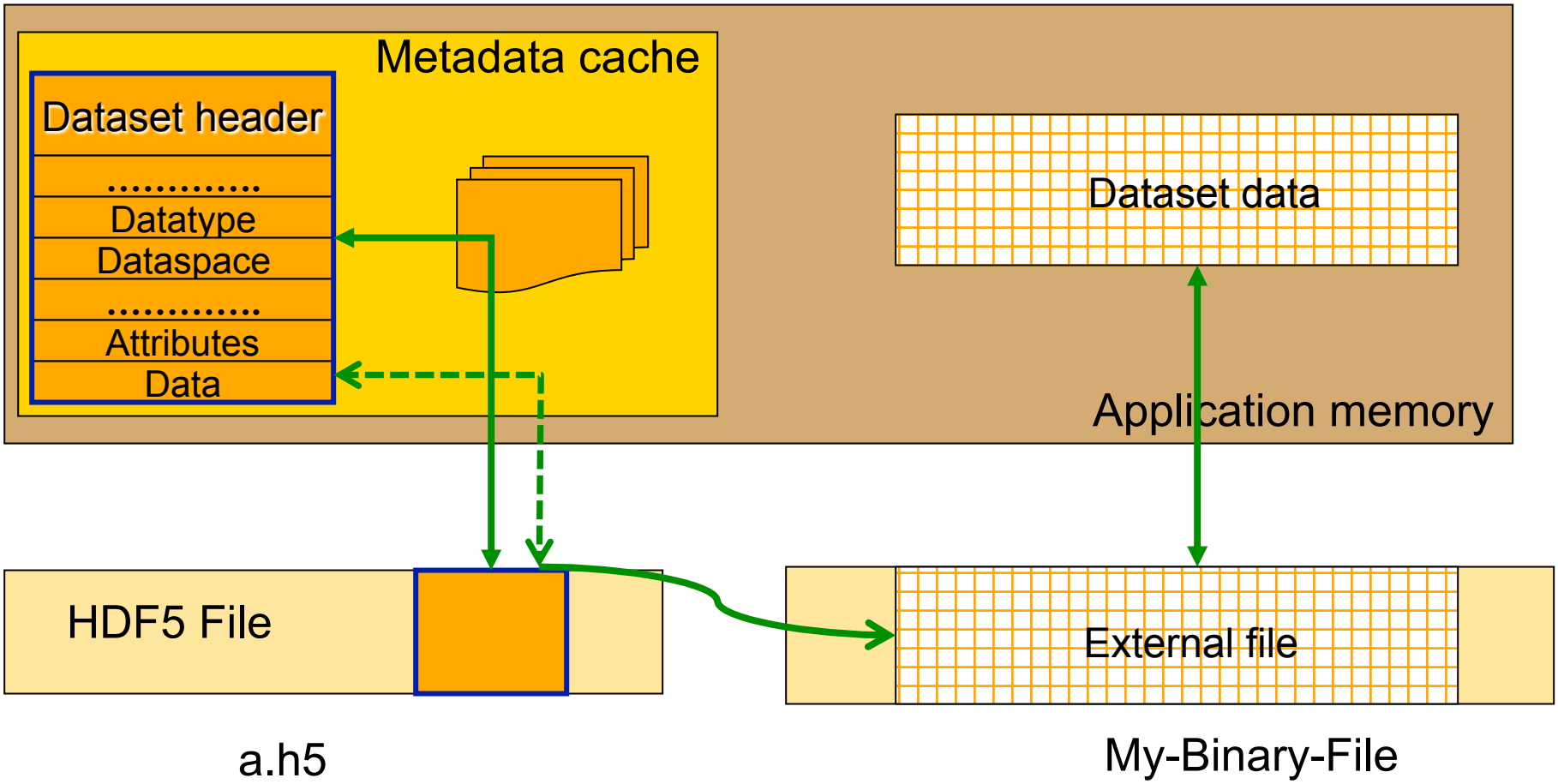
Contiguous Storage

- Pros:
  - Default storage mechanism for HDF5 dataset
  - Allows sub-setting
  - Efficient access to the whole dataset or two contiguous (in the file) subset
  - Can be easily located in the HDF5 file
- Cons:
  - No compression
    - Will be enabled in HDF5 1.10.0 release
  - Data cannot be added

EXTERNAL DATASET

External Dataset

Data stored in one contiguous block in external binary file



External Storage

- Pros:
  - Mechanism to reference data stored in a non-HDF5 binary file
  - Can be easily “imported” to HDF5 with h5repack
  - Allows sub-setting
  - Efficient access to the whole dataset or to contiguous (in the file) subset
- Cons:
  - Two or more files
  - No compression
  - Data cannot be added

External Example and Demo

```
HDF5 "h5ex_d_extern-new.h5" {
GROUP "/" {
  DATASET "DSExternal" {
    DATATYPE  H5T_STD_I32BE
    DATASPACE  SIMPLE { ( 28 ) / ( 28 ) }
    STORAGE_LAYOUT {
      CONTIGUOUS
      EXTERNAL {
        FILENAME h5ex_d_extern.data SIZE 112 OFFSET 0
      }
    }
    FILTERS {
      NONE
    }
    FILLVALUE {
      FILL_TIME H5D_FILL_TIME_IFSET
      VALUE  0
    }
    ALLOCATION_TIME {
      H5D_ALLOC_TIME_LATE
    }
    DATA {
    (0): 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3,
    (22): 3, 3, 3, 3, 3, 3
    }
....
```

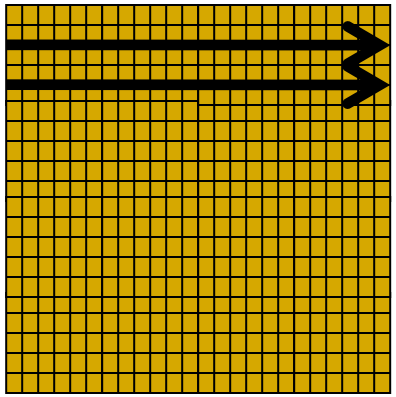
```
[   0   0   0   0   0   0   0]
[   1   1   1   1   1   1   1]
[   2   2   2   2   2   2   2]
[   3   3   3   3   3   3   3]
```

CHUNKING IN HDF5

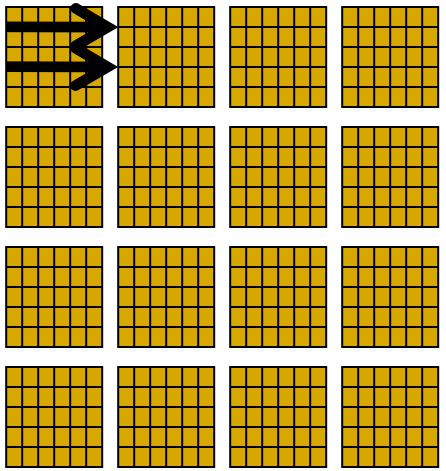
What is HDF5 Chunking?

- Data is stored in chunks of predefined size
- Two-dimensional instance may be referred to as data tiling
- HDF5 library usually writes/reads the whole chunk

Contiguous

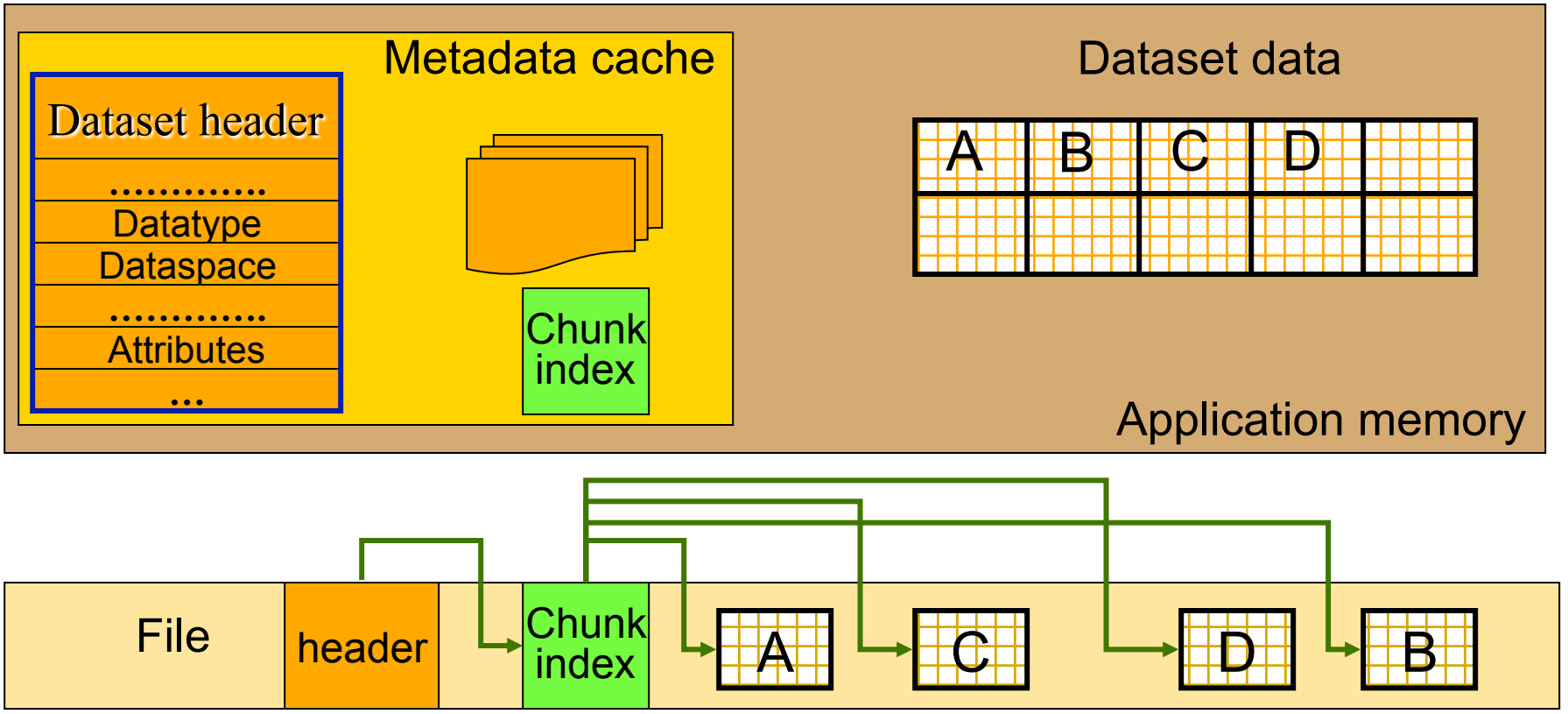


Chunked



What is HDF5 Chunking?

- Dataset data is divided into equally sized blocks (chunks).
- Each chunk is stored separately as a contiguous block in HDF5 file.

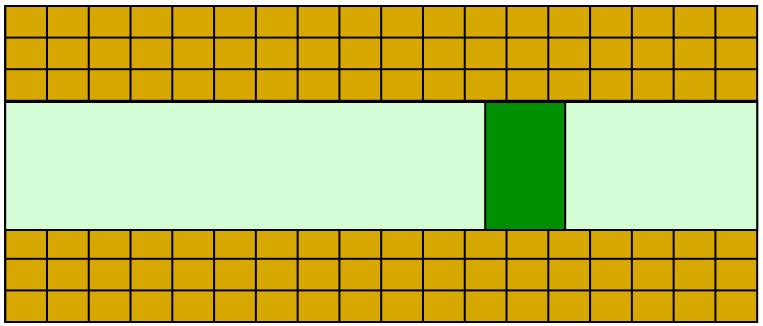


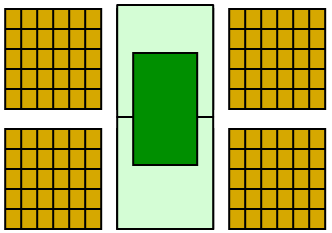
Why HDF5 chunking?

- Chunking is required for several HDF5 features
  - Applying compression and other filters like checksum
  - Expanding/shrinking dataset dimensions and adding/”deleting” data

Why HDF5 Chunking?

- If used appropriately chunking improves partial I/O for big datasets





Only two chunks are involved in I/O

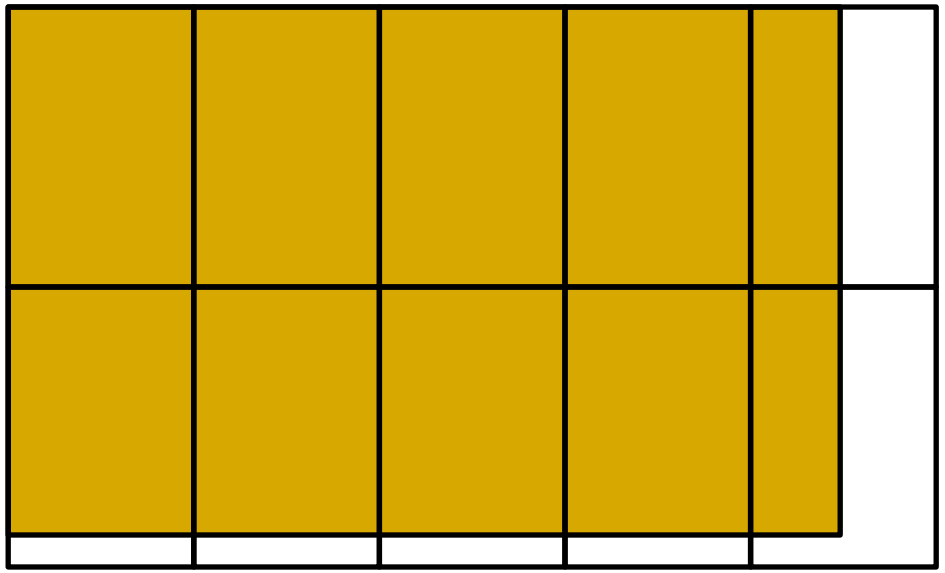
Creating Chunked Dataset

1. Create a dataset creation property list.
2. Set property list to use chunked storage layout.
3. Create dataset with the above property list.

```
dcpl_id   = H5Pcreate(H5P_DATASET_CREATE);
rank = 2;
ch_dims[0] = 100;
ch_dims[1] = 200;
H5Pset_chunk(dcpl_id, rank, ch_dims);
dset_id = H5Dcreate (…, dcpl_id);
H5Pclose(dcpl_id);
```

Creating Chunked Dataset

- Things to remember:
  - Chunk always has the same rank as a dataset
  - Chunk's dimensions do not need to be factors of dataset's dimensions
  - *Caution: May cause **more** I/O than desired (see white portions of the chunks below)*



Chunking Limitations

- Limitations
  - Chunk dimensions cannot be bigger than dataset dimensions
  - *Number of elements a chunk is limited to 4GB*
    - H5Pset_chunk fails otherwise
  - *Total size chunk is limited to 4GB*
    - Total size = (number of elements) * (size of the datatype)
    - H5Dwrite fails later on

Writing or Reading Chunked Dataset

1. Chunking is transparent to the application.
2. Use the same set of operations as for contiguous dataset, for example,

```
H5Dopen(…);
H5Sselect_hyperslab (…);
H5Dread(…);
```

3. Selections do not need to coincide precisely with the chunk boundaries.

Creating Compressed Dataset

1. Create a dataset creation property list
2. Set property list to use chunked storage layout
3. Set property list to use filters
4. Create dataset with the above property list

```
dcpl_id = H5Pcreate(H5P_DATASET_CREATE);
rank = 2;
ch_dims[0] = 100;
ch_dims[1] = 100;
H5Pset_chunk(dcpl_id, rank, ch_dims);
H5Pset_deflate(dcpl_id, 9);
dset_id = H5Dcreate (…, dcpl_id);
H5Pclose(dcpl_id);
```

Example: h5_d_unlimgzip.c

H5_d_unlimgzip.h5

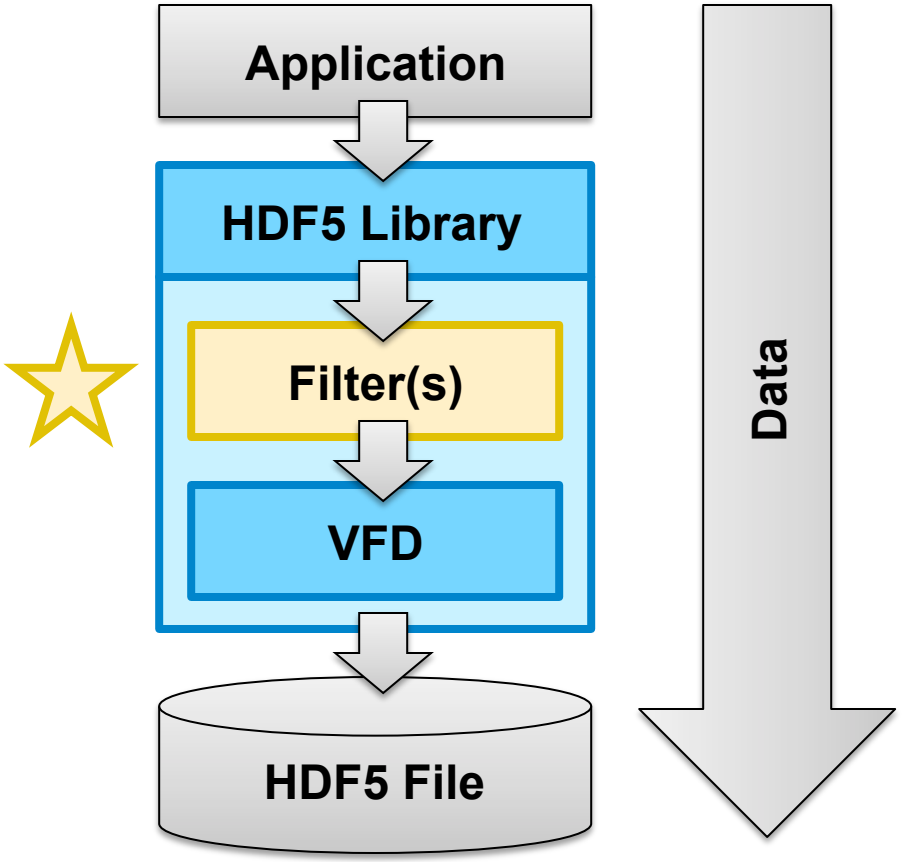
```
[  0  -1  -2  -3  -4  -5  -6   7   8   9 ]
[  0   0   0   0   0   0   0   7   8   9 ]
[  0   1   2   3   4   5   6   7   8   9 ]
[  0   2   4   6   8  10  12   7   8   9 ]
[  0   1   2   3   4   5   6   7   8   9 ]
[  0   1   2   3   4   5   6   7   8   9 ]
```

First write
Second write

HDF5 FILTERS AND COMPRESSION

What is an HDF5 filter?

- Data transformation performed by the HDF5 library during I/O operations



What is an HDF5 filter?

HDF5 filters (or *built-in* filters)

- Supported by The HDF Group (internal)
- Come with the HDF5 library source code

User-defined filters

- Filters written by HDF5 users and/or available with some applications (h5py, PyTables)
- May be or may not be registered with The HDF Group

HDF5 filters

- Filters are arranged in a pipeline so the output of one filter becomes the input of the next filter
- The filter pipeline can be only applied to
  - *Chunked* datasets
    - HDF5 library passes each chunk through the filter pipeline on the way to or from disk
  - Groups
    - Link names are stored in a local heap, which may be compressed with a filter pipeline
- The filter pipeline is permanent for dataset or a group

Applying filters to a dataset

```
dcpl_id   = H5Pcreate(H5P_DATASET_CREATE);
cdims[0] = 100;
cdims[1] = 100;
H5Pset_chunk(dcpl_id, 2, cdims);
H5Pset_shuffle(dcpl);
H5Pset_deflate(dcpl_id, 9);
dset_id = H5Dcreate (…, dcpl_id);
H5Pclose(dcpl_id);
```

Applying filters to a group

```
gcpl_id   = H5Pcreate(H5P_GROUP_CREATE);
H5Pset_deflate(dcpl_id, 9);
group_id = H5Gcreate (…, gcpl_id, …);
H5Pclose(gcpl_id);
```

Internal HDF5 Filters

- Internal filters are implemented by The HDF Group and come with the library
  - FLETCHER32
  - SHUFFLE
  - SCALEOFFSET
  - NBIT

- HDF5 internal filters can be configured out using --disable-filters="filter1, filter2, .."

External HDF5 Filters

- External HDF5 filters rely on the third-party libraries installed on the system
  - GZIP
    - By default HDF5 configure uses ZLIB installed on the system
    - Configure will proceed if ZLIB is not found on the system
  - SZIP (added by NASA request)
    - Optional; have to be configured in using --with-szlib=/path….
    - Configure will proceed if SZIP is not found
    - Comes with a license http://www.hdfgroup.org/doc_resource/SZIP/Commercial_szip.html
    - Decoder is free; for encoder see the license terms

Checking available HDF5 Filters

- Use API (`H5Zfilter_avail`)
- Check libhdf5.settings file

Features:

Parallel HDF5: no

.....

I/O filters (external): deflate(zlib),szip(encoder)

.....

Internal filters are always present now.

Third-party HDF5 filters

- Compression methods supported by HDF5 user community

  http://www.hdfgroup.org/services/contributions

  - LZO, BZIP2, BLOSC (PyTables)
  - LZF (h5py)
  - MAFISC
    - The Website has a patch for external module loader
- Registration process
  - Helps with filter's provenance

Example: h5dump output on BZIP2 data

```
HDF5 "h5ex_d_bzip2.h5" {
GROUP "/" {
   DATASET "DS-bzip2" {
      ...
      }
      FILTERS {
         UNKNOWN_FILTER {
            FILTER_ID 307
            COMMENT bzip2
            PARAMS { 9 }
         }
      }
   .....
      }
      DATA {h5dump error: unable to print data

      }
```

Problem with using custom filters

- "Off the shelf" HDF5 tools do not work with the third-party filters
  - h5dump, MATLAB and IDL, etc.
- Solution
  - Use dynamically loaded filters

https://www.hdfgroup.org/HDF5/doc/Advanced/DynamicallyLoadedFilters/HDF5DynamicallyLoadedFilters.pdf

DYNAMICALLY LOADED FILTERS IN HDF5

Dynamically loaded filters

- The HDF5 third-party filters are available as shared libraries or DLLs on the user's system.
- There are predefined default locations where the HDF5 library searches the shared libraries or DLLs with the HDF5 filter functions.

  /usr/local/hdf5/lib/plugin
- The default location may be overwritten by an environment variable.

  HDF5_PLUGIN_PATH
- Once a filter plugin library is loaded, it stays loaded until the HDF5 library is closed.

Programming Model

- When create set the filter using the filter ID

```
dcpl = H5Pcreate (H5P_DATASET_CREATE);
status = H5Pset_filter (dcpl, (H5Z_filter_t)307,
         H5Z_FLAG_MANDATORY, (size_t)6, cd_values);
dset = H5Dcreate (file, DATASET, H5T_STD_I32LE, space,
        H5P_DEFAULT, dcpl,….);
status = H5Dwrite (dset, H5T_NATIVE_INT, H5S_ALL, H5S_ALL,
          H5P_DEFAULT, wdata[0]);
```

- Transparent on read

Plugin Example

http://svn.hdfgroup.uiuc.edu/hdf5_plugins/trunk/

- BZIP2
  - Bzip2 filter implemented in PyTables
  - Built with configure or CMake
  - Used as acceptance test for the feature
  - More plugins can be added in the future

Demo

- h5dump before and after
- h5repack

  `h5repack -f UD={ID:k; N:m; CD_VAL:[n1,…,nm]}…`
- BZIP2 example

```
h5repack -f UD={ID:307; N:1; CD_VAL:[9]} file1.h5 file2.h5
```

Acknowledgement

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?