Working with Collections of HDF5 Files

Problem

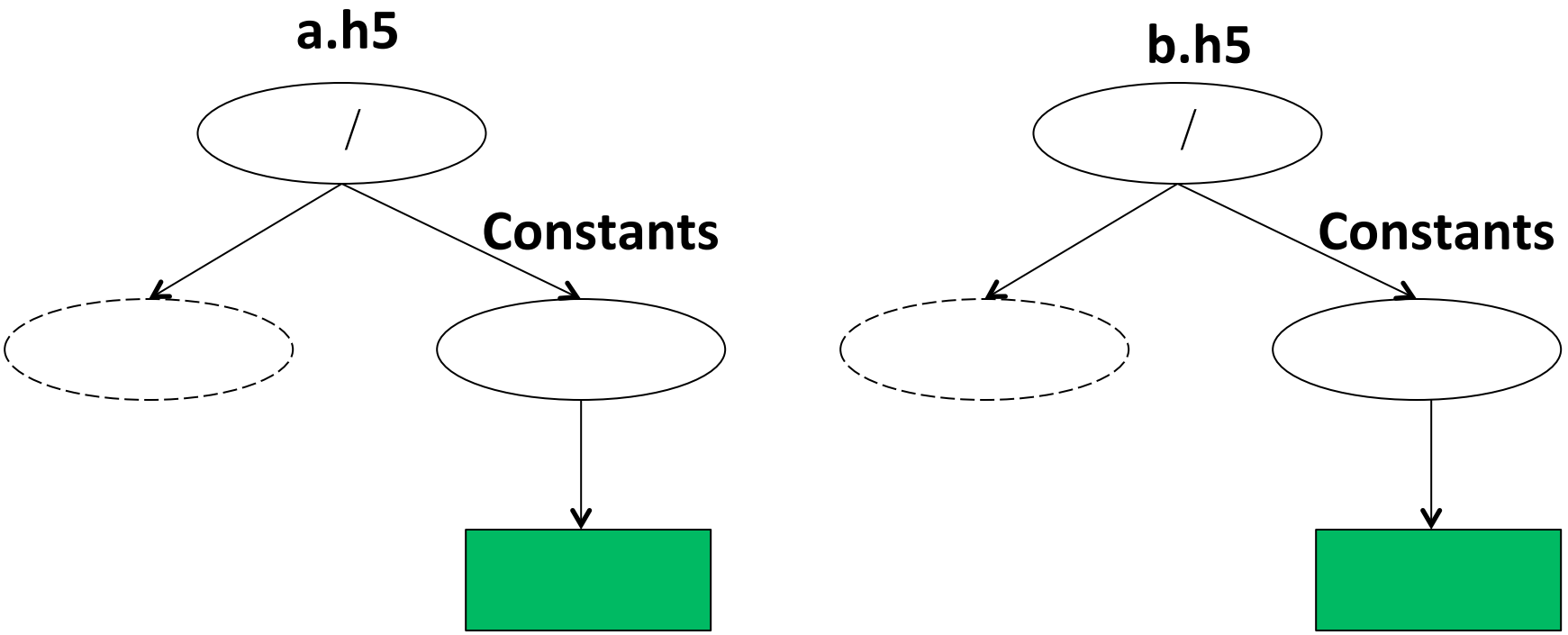
- How to access data stored across HDF5 files?
  - Combine all data in one file
  - Go through a collection of the files and access each of them
  - Use HDF5 features to facilitate access and create data view you or your users need
    - File Mounting (HDF5 1.0.0)
    - External links (HDF5 1.8.0)
    - Virtual Datasets (upcoming HDF5 1.10.0)

HDF5
File Mounting

Problem: Data Duplication

- How to share data from the same HDF5 file?
  - Use case: A file c.h5 contains data which is constant for all problems. The output of a particular physics application is dumped into a.h5 and b.h5 and the physics expects various constants from c.h5 in the “constants” group of the two data files to run simulation.

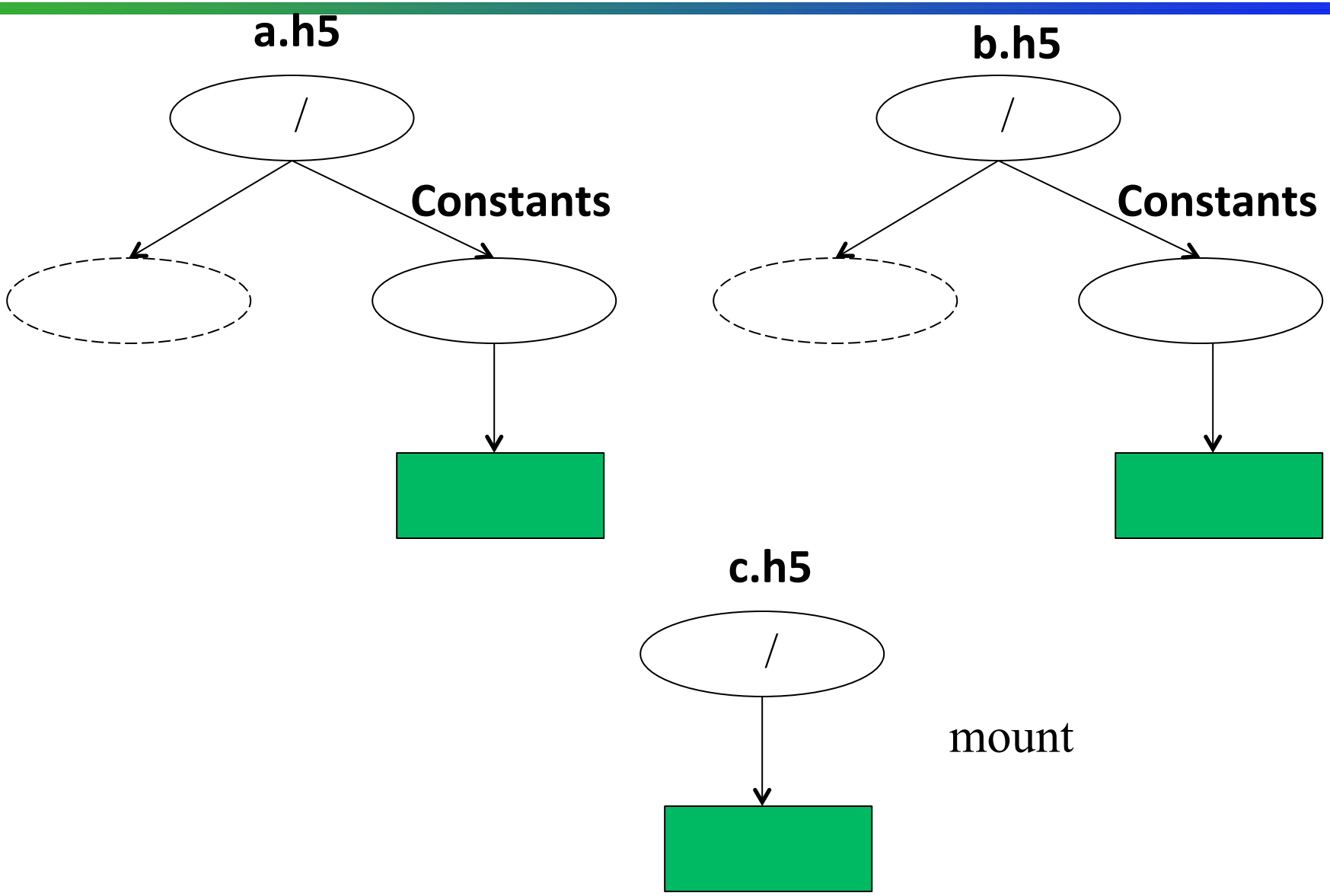
Use case



Problem: Data Duplication

- Solution – File Mounting
  - Instead of duplicating the contents of c.h5 into every output file before running simulation, we simply make the content of c.h5 available under “CONSTANTS” group in each file.

Use case



HDF5 File Mounting

- Similar to UNIX file system mount operation
- The group structure and data in one HDF5 file becomes accessible to an application as a part of another HDF5 file at run time /
- See https://www.hdfgroup.org/HDF5/doc/H5.user/MountingFiles.html for more information

- Open the files.
- Choose the **mount point** in the first file (the parent file).
- Use `H5Fmount` to mount the second file (the child file) in the first file.
- Work with the objects in the second file as if they were members of the mount point group in the first file.
- Unmount the second file using `H5Funmount` when the work is done.

Programming Example

```
/*
 * Open files a.h5 and c.h5
 */
fid1 = H5Fopen(“a.h5”, H5F_ACC_RDONLY, H5P_DEFAULT);
fid2 = H5Fopen(“c.h5”, H5F_ACC_RDONLY, H5P_DEFAULT);
/*
 * Mount C.h5 file under CONSTANTS in the first file.
 */
H5Fmount(fid1, "/CONSTANTS", fid2, H5P_DEFAULT);
/*
 * Access dataset D in the first file under /CONSTANTS/D name.
 */
did = H5Dopen2(fid1, "/CONSTANTS/D", H5P_DEFAULT);


…..
 H5Funmount(fid1, "/CONSTANTS”);
```

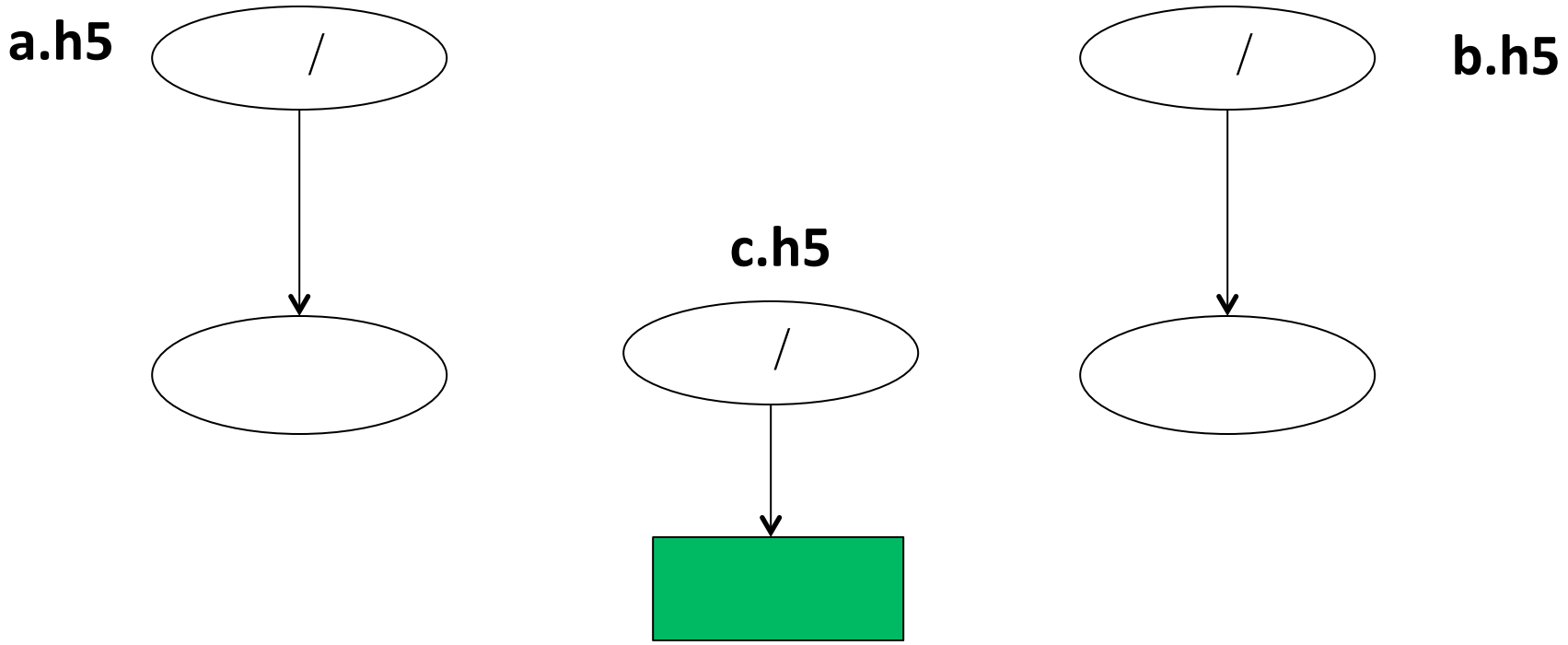
HDF5
External Links

Problem: Data Duplication

- How to point to data stored in another HDF5 file?
  - Use case: The output of a particular physics application is dumped into a.h5 and b.h5 and constants from c.h5 were used for simulation.
  - We want to **preserve** information that c.h5 was used when we created a.h5 and b.h5

Use case

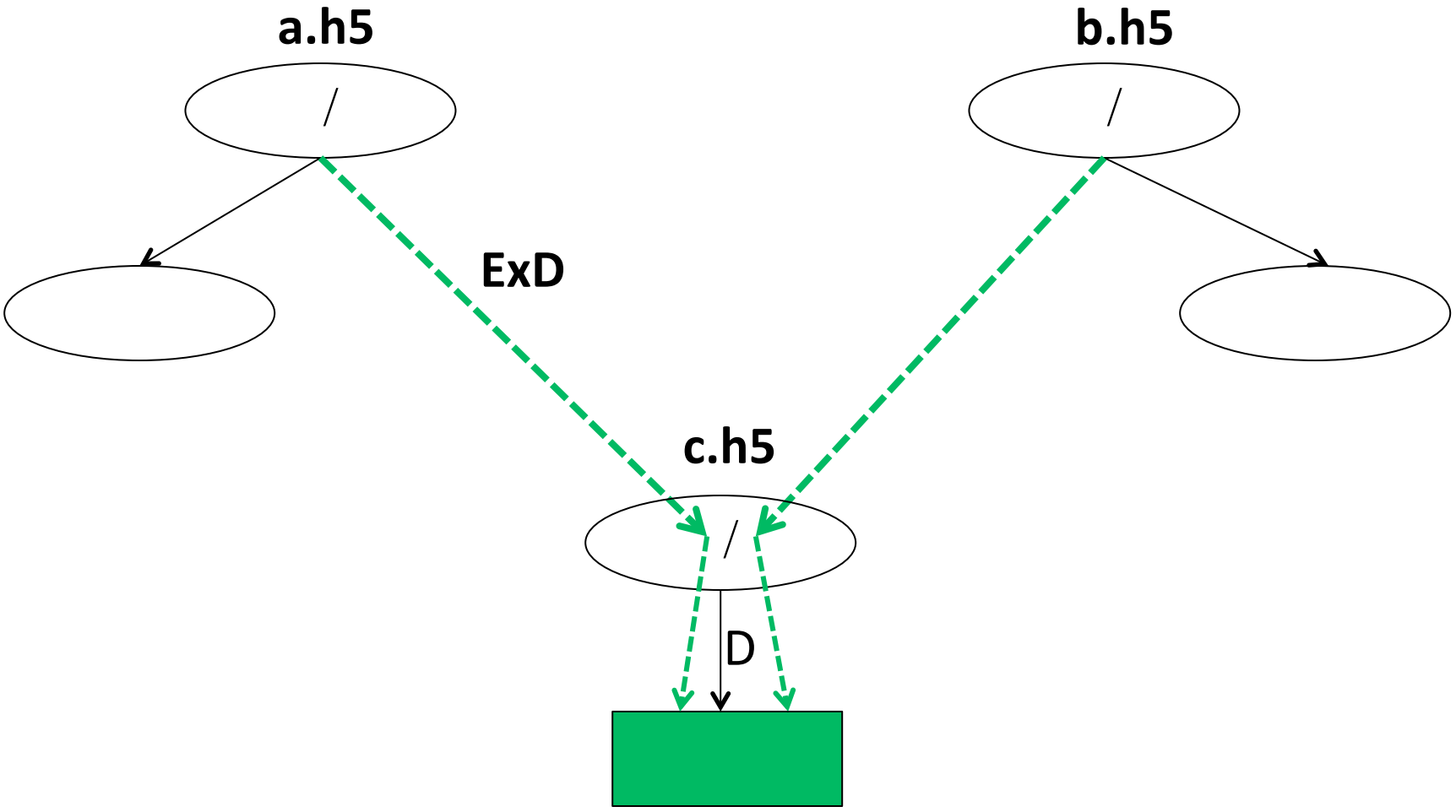
Application uses data in c.h5 to generate data in a.h5 and b.h5



Problem: Data Duplication

- How to point to data stored in another HDF5 file?
  - Instead of copying the contents of c.h5 into every physics output file we simply point from files a.h5 and b.h5 to a dataset in c.h5

Use case



External link “ExD” points to file c.h5 and dataset /D in it.

Programming Example

```
  /* Create a file a.h5 */
file_id = H5Fcreate("a.h5", H5F_ACC_TRUNC, H5P_DEFAULT,
H5P_DEFAULT);
    …….
    /* Create an external link in the source file pointing to the
target dataset D in a file c.h5. */
H5Lcreate_external("c.h5", "/D", file_id, "/ExD", H5P_DEFAULT,
H5P_DEFAULT);

  /* Now one can access the /D dataset using ExD link */
dataset_id = H5Dopen(source_file_id, "ExD", ……);
H5Dread (dataset_id,……);

/* We will get data stored in /D */
```

External Links for JPSS products packaging

- Example of how external links can be used in JPSS products
- Currently the name of the geolocation file is stored in an attribute on the root group.
- User has to know how to interpret the attribute
- One can create and external link to geolocation fields in the geolocation file instead:

```
H5Lcreate_external("SATMS_xx.h5", "/All_Data/ATMS-SDR-GEO_All", file_id, "/All_Data/ATMS-SDR-GEO_All/", H5P_DEFAULT, H5P_DEFAULT);
```

GATMO_*.h5

- Transparent to HDF5 applications

HDF5
Virtual Dataset

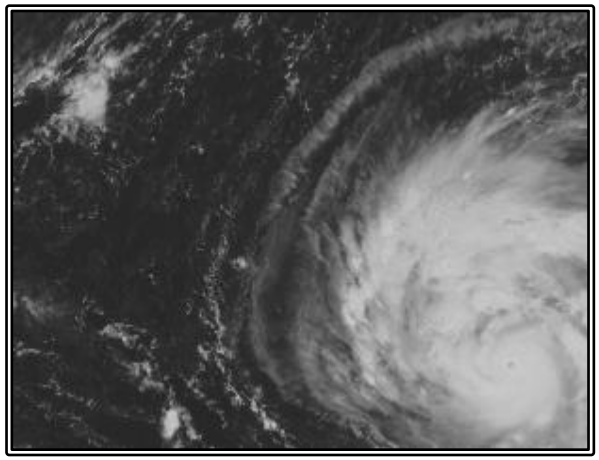
CHALLENGE

Challenge

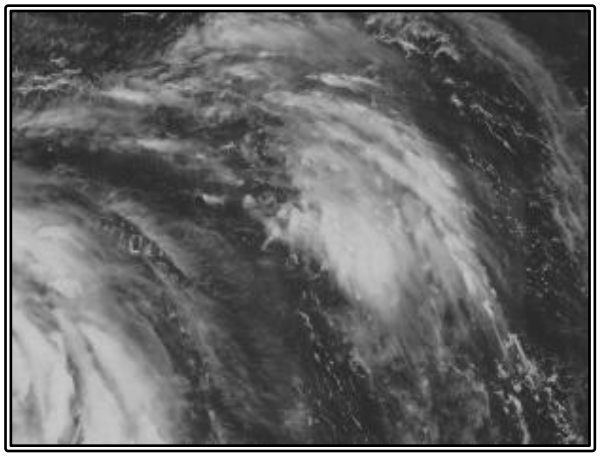
- How to view data stored across the HDF5 files as an HDF5 dataset on which normal operations can be performed?
  - High-level approach
    - Special library that applications like MATLAB and H5Py will need to use
    - Example : THREDDS Data Server based on OPeNDAP http://www.unidata.ucar.edu/software/thredds/current/tds/TDS.html
  - Native HDF5 implementation
    - Transparent to applications

TWO SIMPLE USE CASES

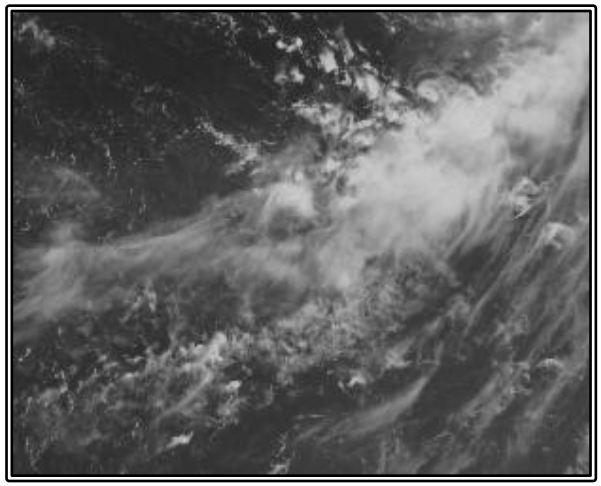
Collect data one way ….



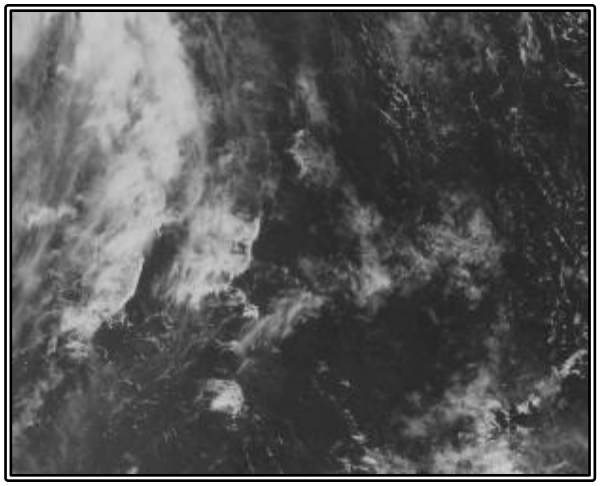
File: a.h5
Dataset /A



File: b.h5
Dataset /B



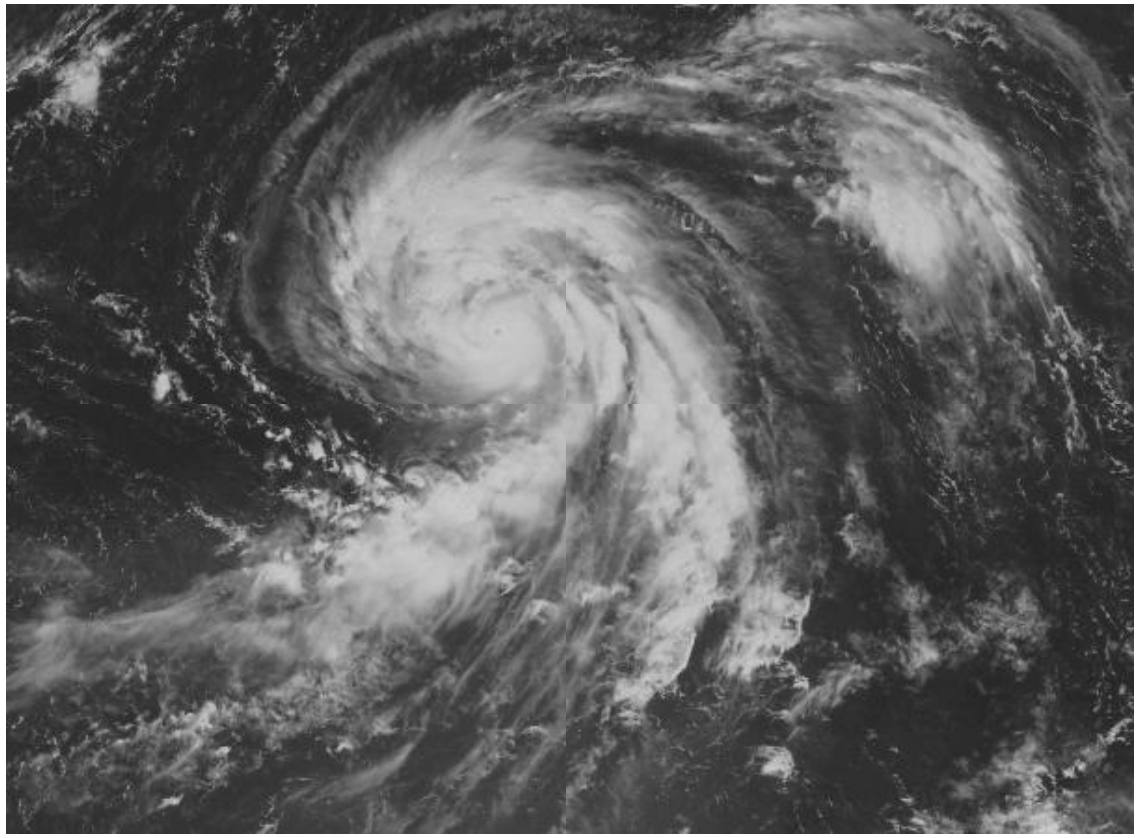
File: c.h5
Dataset /C



File: d.h5
Dataset /D

Present it in a different way…

Whole image



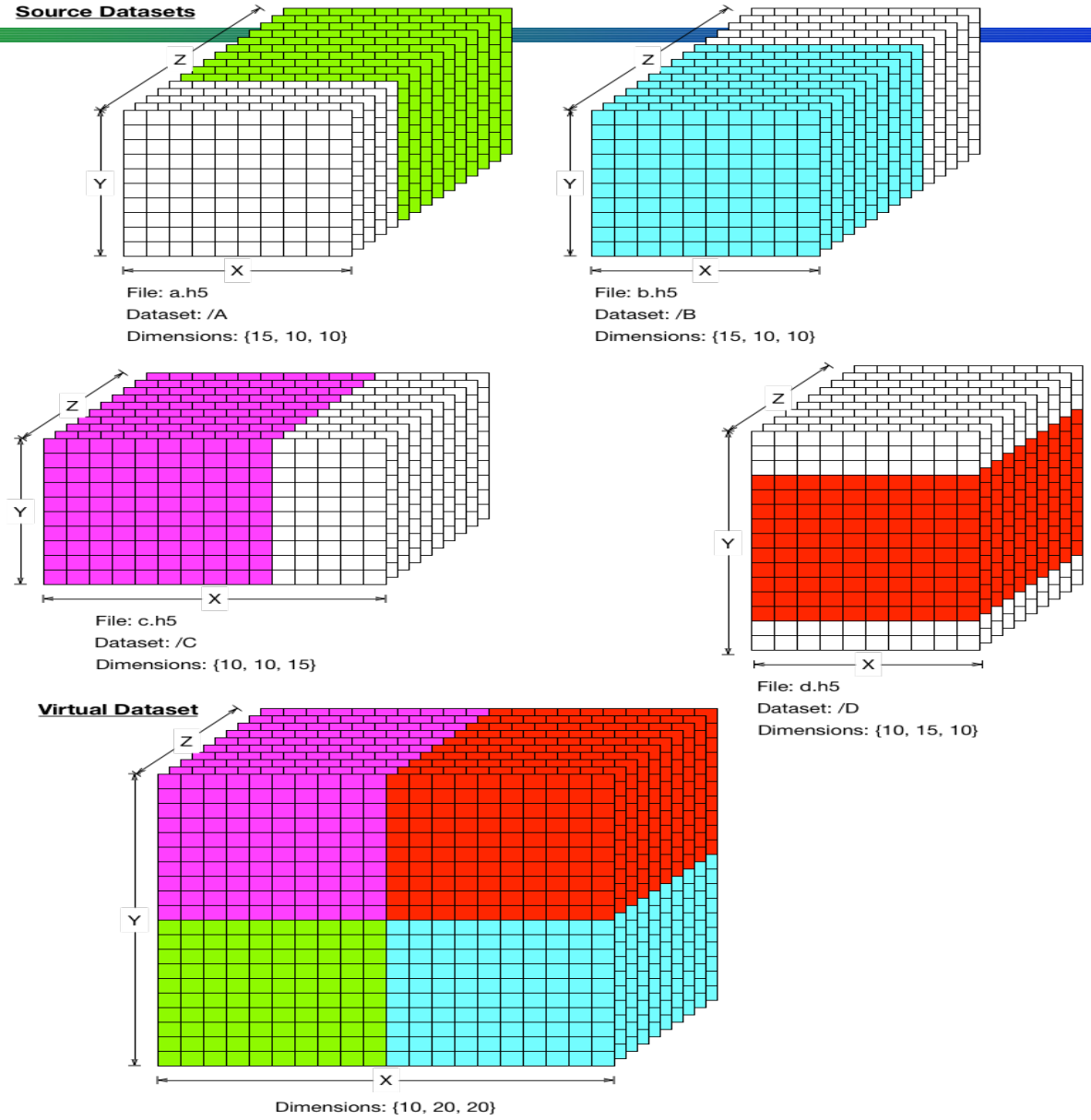
File: F.h5
Dataset /D

Present it in a different way…

Subset of data

File: F.h5
Dataset /F

VDS Example


Source Datasets
Z
Y
X
File: a.h5
Dataset: /A
Dimensions: {15, 10, 10}
File: b.h5
Dataset: /B
Dimensions: {15, 10, 10}
File: c.h5
Dataset: /C
Dimensions: {10, 10, 15}
File: d.h5
Dataset: /D
Dimensions: {10, 15, 10}
Virtual Dataset
Dimensions: {10, 20, 20}

SYNCHROTRON COMMUNITY USE CASES

Common Characteristics

- New detectors have high rates and parallel architecture
- Multiple processes are writing compressed parts of the images into HDF5 files in parallel
- No synchronization between writing processes
- Detectors generate 3-10 GB data per second

Excalibur Detector Hardware Architecture

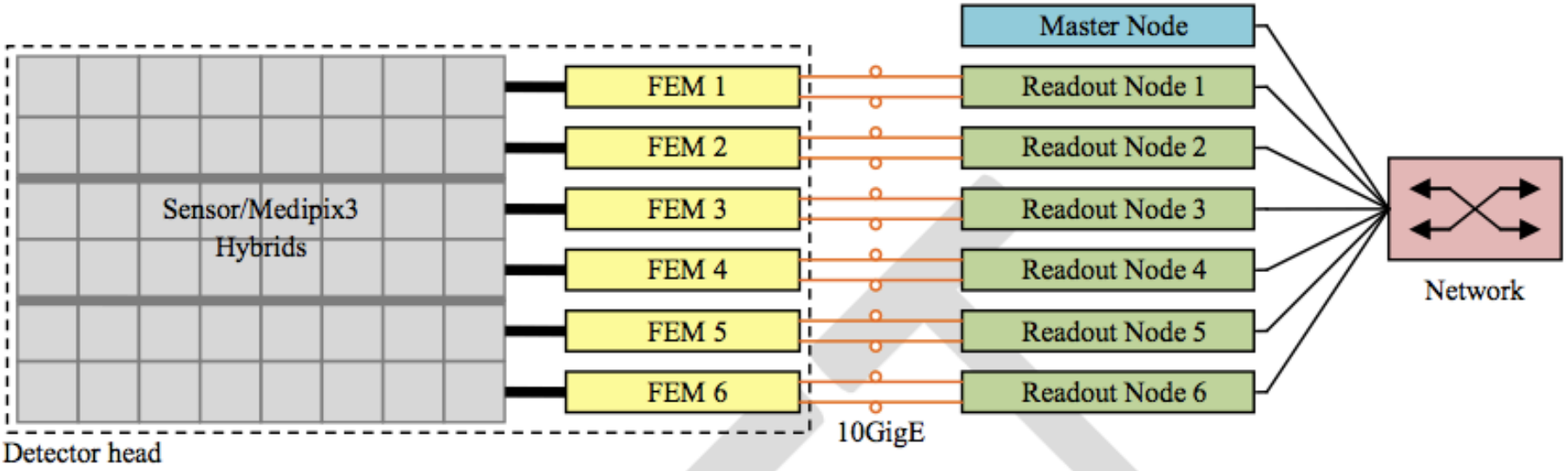


Figure 1: Excalibur hardware architecture.

Courtesy DLS

See Confluence - DLS – Virtual Dataset Phase 0 for the document

Excalibur Chip Layout and Gap detail

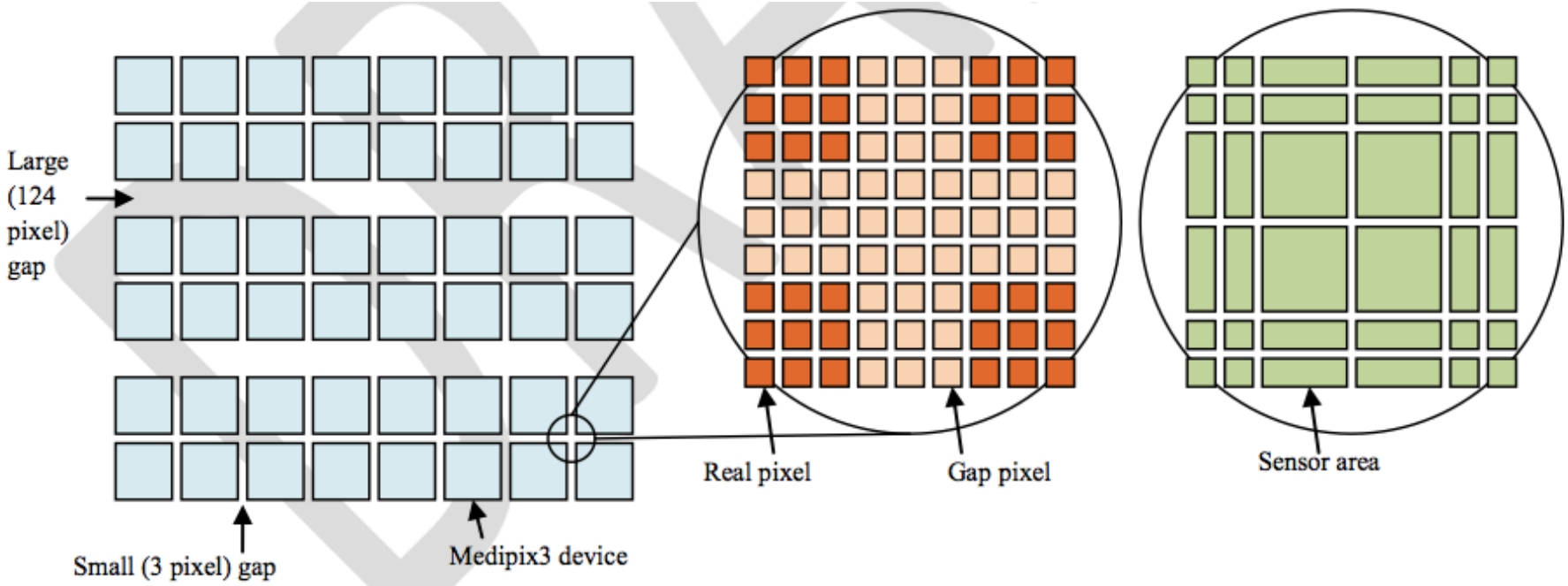
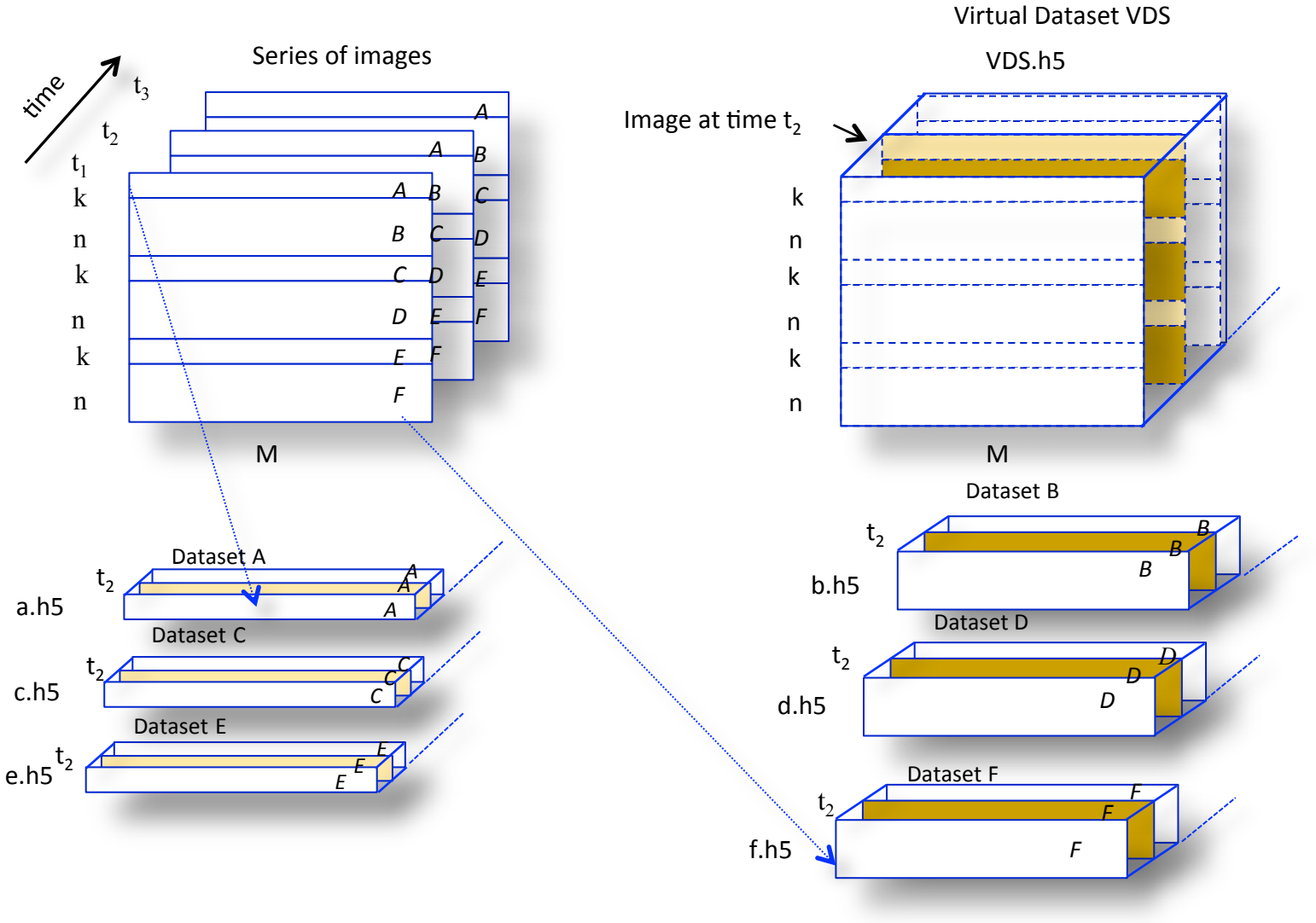


Figure 2. Excalibur Medipix3 chip layout and gap details.

Courtesy DLS

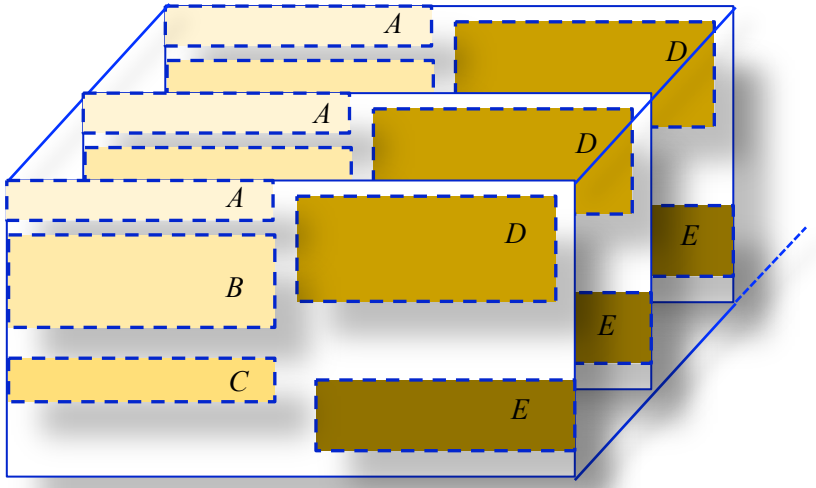
See Confluence - DLS – Virtual Dataset Phase 0 for the document

Unlimited Use Case



Use Case with Gaps

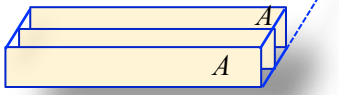
Virtual Dataset VDS with “gaps”



VDS.h5

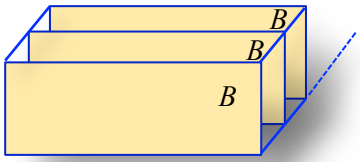
Dataset A

a.h5



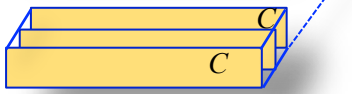
Dataset B

b.h5



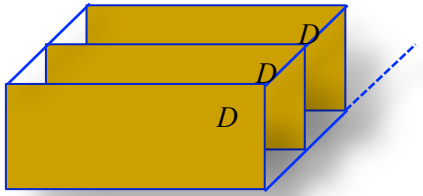
Dataset C

c.h5



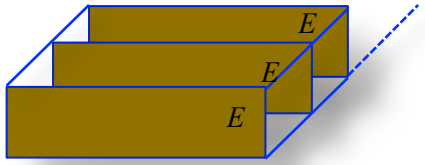
Dataset D

d.h5

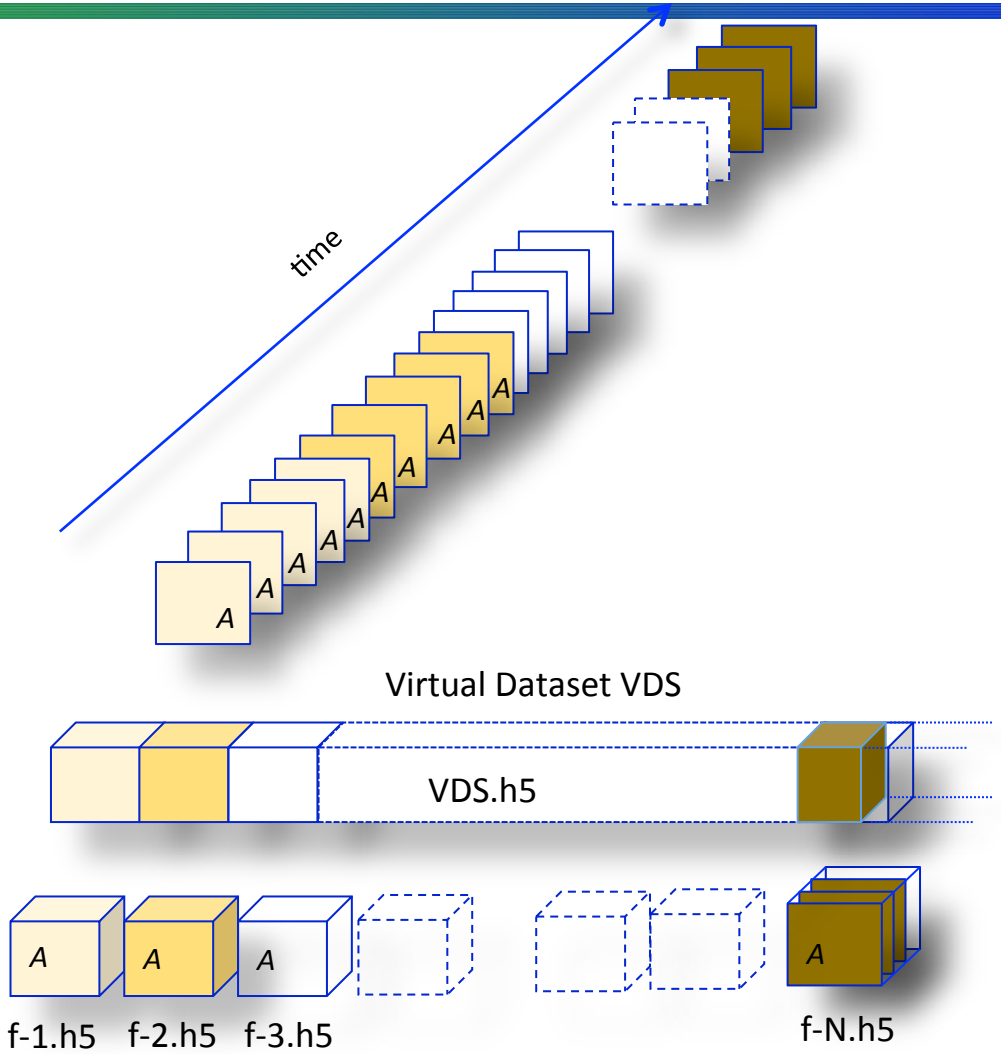


Dataset E

e.h5

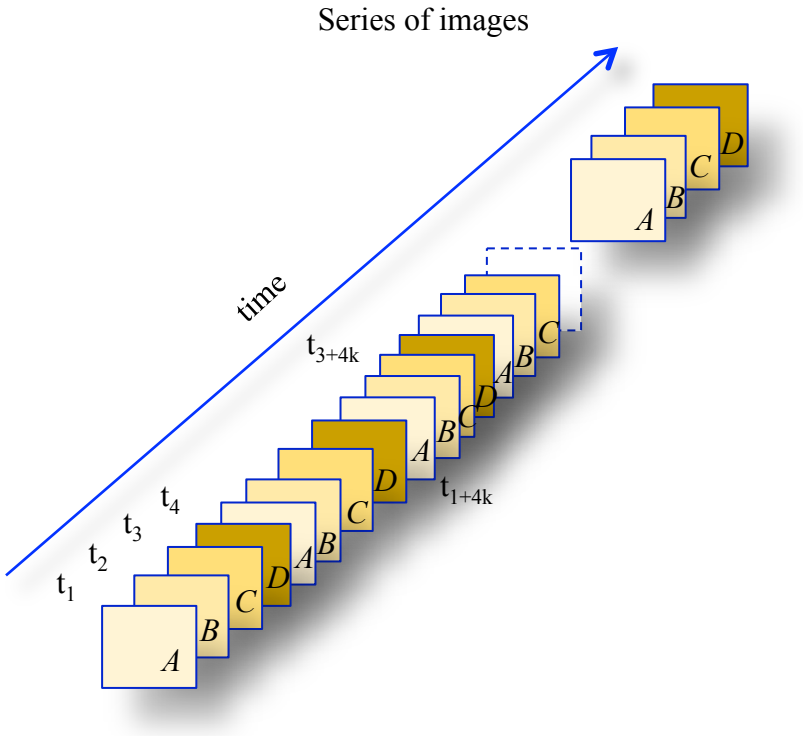


“Printf-type” Source Generation

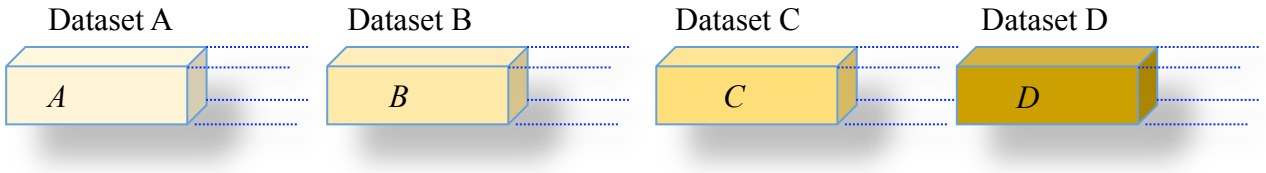


File names are generated by the “printf” capability

Use Case with Interleave Planes



Virtual Dataset VDS has images *A*, *B*, *C and D* interleaved



a.h5 b.h5 c.h5 d.h5

High-Level Requirements

- No change in the programming model for VDS I/O
- Mapping between VDS and HDF5source datasets is persistent and transparent to application
- SWMR access to VDS
- Other
  - HDF5 selection mechanism handles “unlimited selections”
  - Source file names can be generated automatically

Status

- The feature is implemented except SWMR access
- Source code

https://svn.hdfgroup.org/hdf5/features/vds/

- Acceptance test suite

https://svn.hdfgroup.org/hdf5_vds_use_cases/

- Documentation

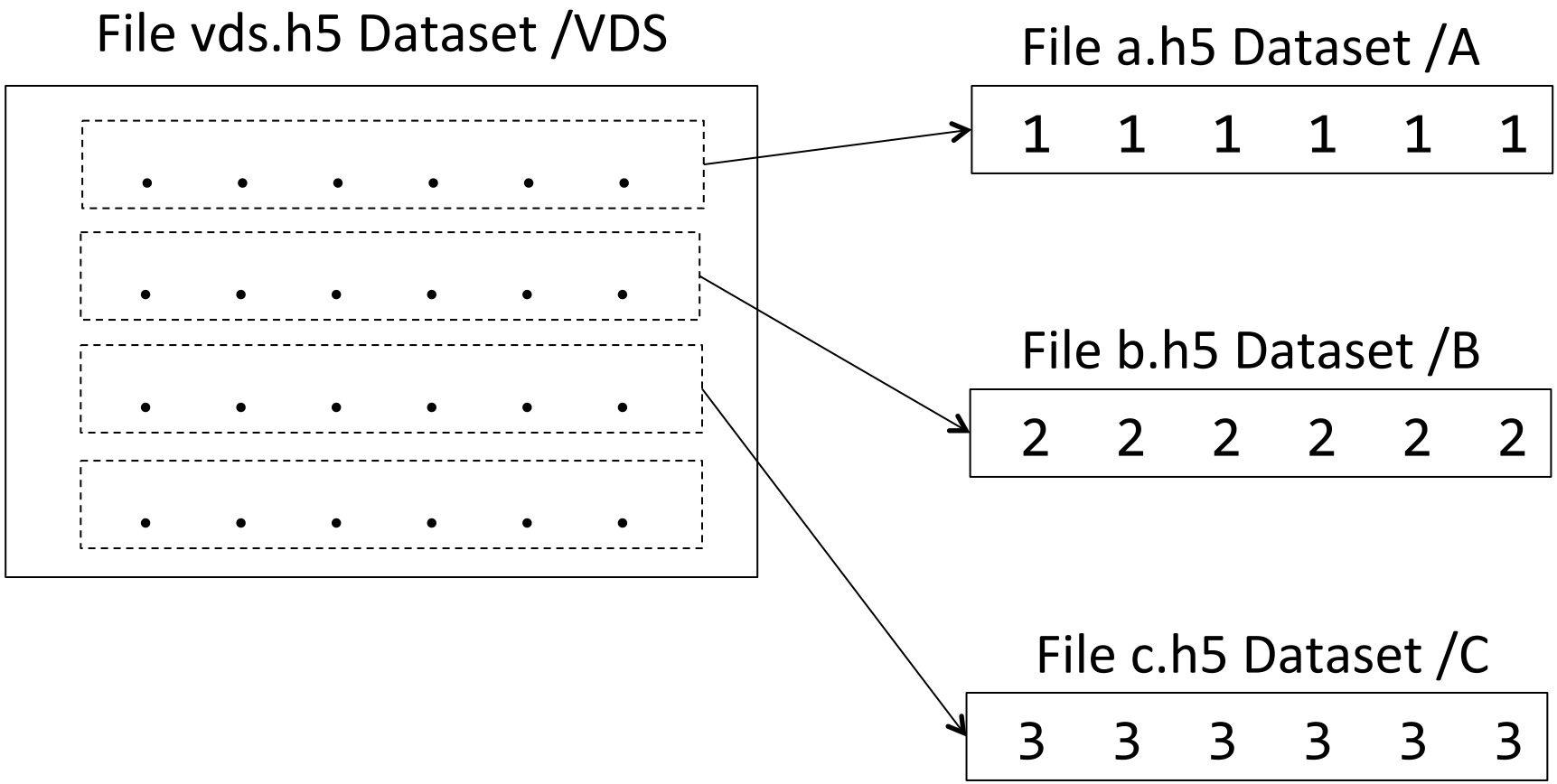
http://www.bigdata.org/HDF5/docNewFeatures/NewFeaturesVirtualDatasetDocs.html

PROGRAMMING MODEL AND EXAMPLES OF MAPPING

VDS Programming Model

- Create datasets that comprise the VDS (the source datasets) (optional)
- Create the VDS
  - Define a datatype and dataspace (can be unlimited)
  - Define the dataset creation property list (including fill value)
  - Map elements from the source datasets to the elements of the VDS
    - Iterate over the source datasets:
      - Select elements in the source dataset (source selection)
      - Select elements in the virtual dataset (destination selection)
      - Map destination selections to source selections
    - End iteration
  - Call H5Dcreate using the properties defined above
- Access the VDS as a regular HDF5 dataset
- Close the VDS when finished

My First VDS Example

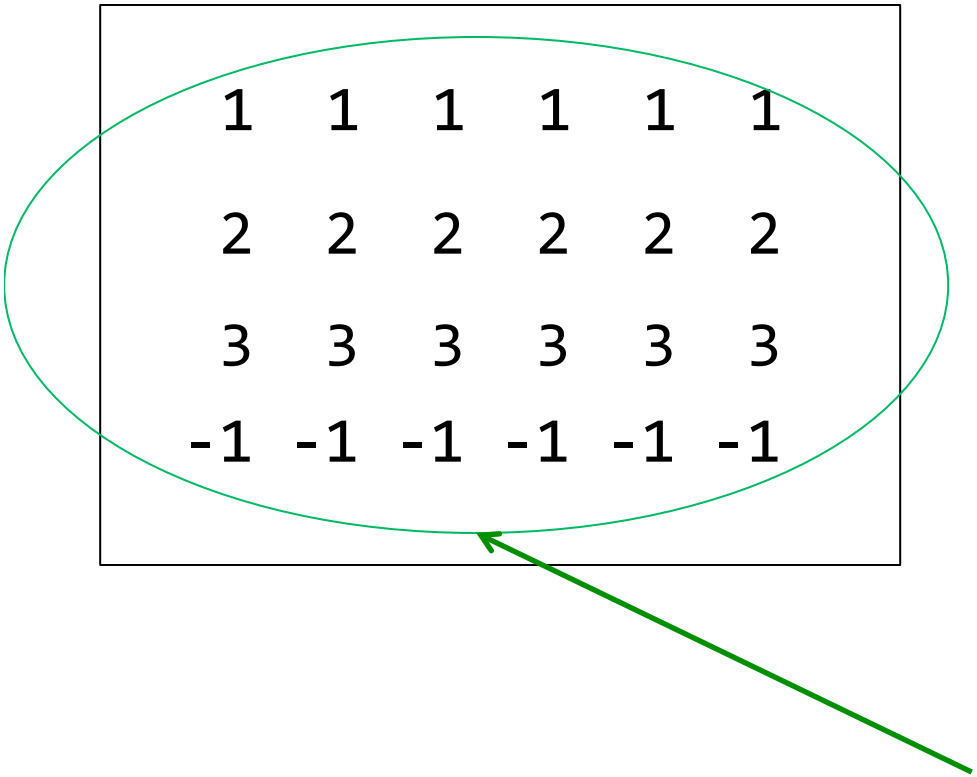


Defining Mapping

```
src_space = H5Screate_simple (RANK1, dims, NULL);
for (i = 0; i < 3; i++) {

start[0] = (hsize_t)i;
status = H5Sselect_hyperslab(space, ..., start,...);
status = H5Pset_virtual(dcpl, space, SRC_F[i], SRC_D[i],
                        src_space);

}

dset = H5Dcreate2 (file, DATASET, H5T_NATIVE_INT, space,
                   H5P_DEFAULT, dcpl, H5P_DEFAULT);
```

My First VDS Example

File vds.h5 Dataset /VDS



File a.h5 Dataset /A

1	1	1	1	1	1

File b.h5 Dataset /B

2	2	2	2	2	2

File c.h5 Dataset /C

3	3	3	3	3	3

Data the application will see when reading /VDS dataset from file vds.h5
The last row is filled with the fill value

Discovering Mappings

- H5Pget_virtual_count
- H5Pget_virtual_vspace
- H5Pget_virtual_srcspace
- H5Pget_virtual_filename
- H5Pget_virtaul_dsetname

h5dump –p vds.h5

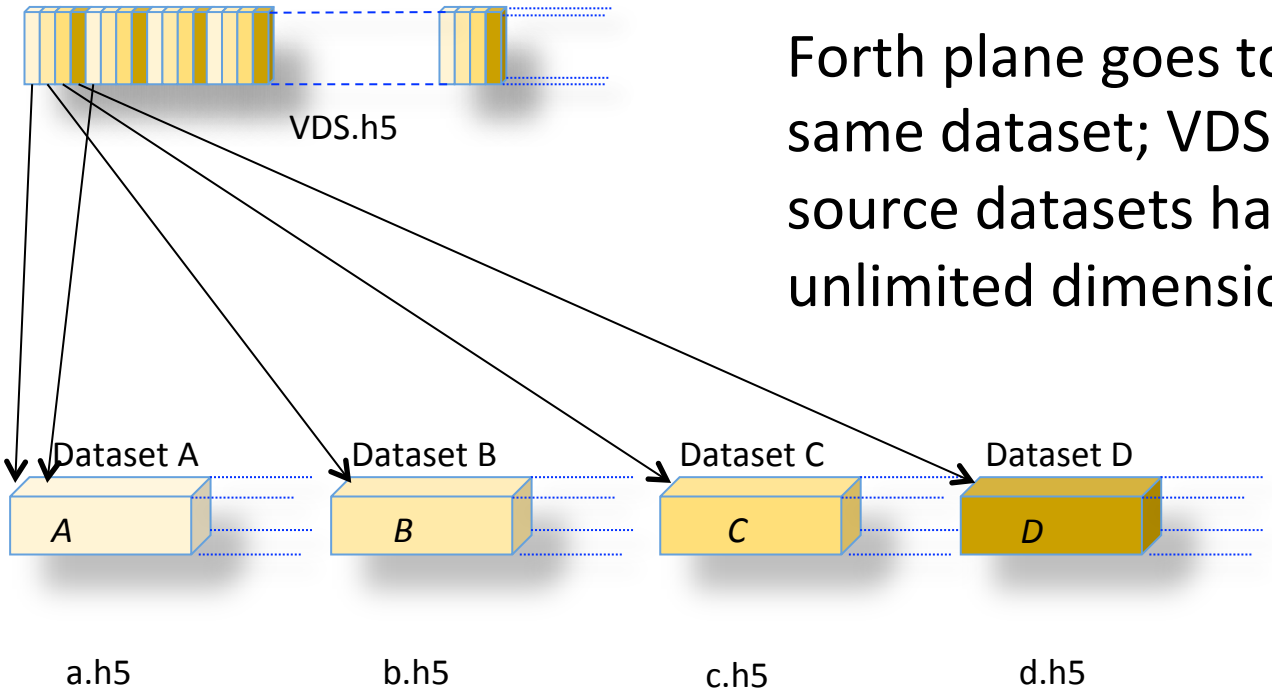
```
HDF5 "vds.h5" {
GROUP "/" {
   DATASET "VDS" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SIMPLE { ( 4, 6 ) / ( 4, 6 ) }
      STORAGE_LAYOUT {
         MAPPING 0 {
            VIRTUAL {
               SELECTION REGULAR_HYPERSLAB {
                  START (0,0)
                  STRIDE (1,1)
                  COUNT (1,1)
                  BLOCK (1,6)
               }
            }
            SOURCE {
               FILE "a.h5"
               DATASET "A"
               SELECTION ALL
            }
         }
```

DEMO

https://svn.hdfgroup.org/hdf5/features/vds/examples/h5_vds.c

Use Case with Interleaved Planes

Virtual Dataset VDS has images *A*, *B, C and D* interleaved



Forth plane goes to the same dataset; VDS and source datasets have unlimited dimension

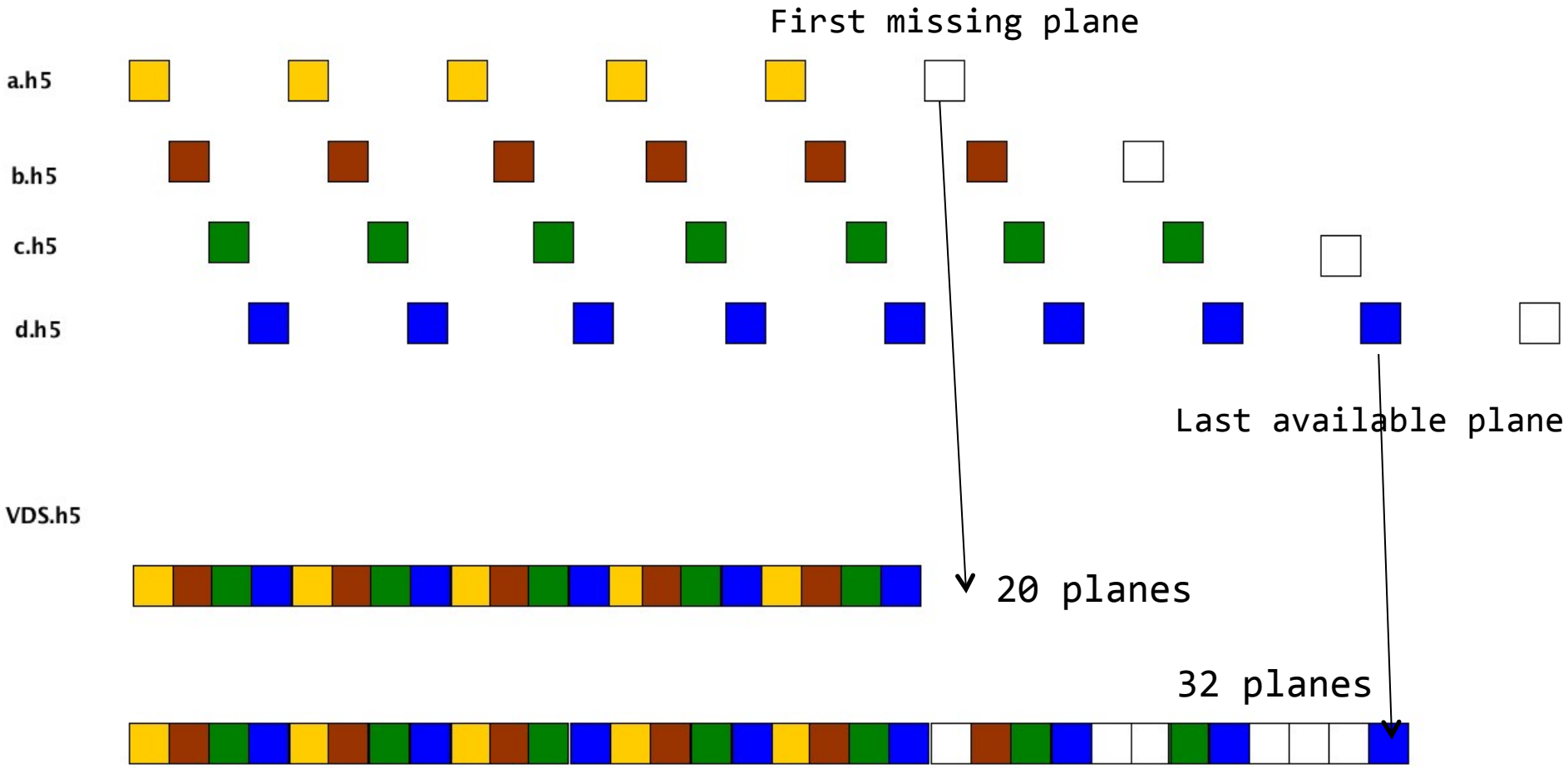
Defining Mapping

```
stride[0] = PLANE_STRIDE; stride[1] = 1; stride[2] = 1;
count[0] = H5S_UNLIMITED; count[1] = 1; count[2] = 1;
src_count[0] = H5S_UNLIMITED; src_count[1] = 1;
src_count[2] = 1;

status = H5Sselect_hyperslab (src_space, H5S_SELECT_SET,
start, NULL, src_count, block);
for (i=0; i < PLANE_STRIDE; i++) {
status = H5Sselect_hyperslab (vspace, H5S_SELECT_SET,
                    start, stride, count, block);
status = H5Pset_virtual (dcpl, vspace, SRC_FILE[i],
                    SRC_DATASET[i], src_space);
start[0]++;
}
```

Strided unlimited mapping

```
../hdf5/bin/h5dump -pH vds-percival-unlim.h5
HDF5 "vds-percival-unlim.h5" {
GROUP "/" {
   DATASET "VDS-Percival-unlim" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SIMPLE { ( 80, 10, 10 ) / ( H5S_UNLIMITED, 10, 10 ) }
      STORAGE_LAYOUT {
         MAPPING 0 {
            VIRTUAL {
               SELECTION REGULAR_HYPERSLAB {
                  START (0,0,0)
                  STRIDE (4,1,1)
                  COUNT (H5S_UNLIMITED,1,1)
                  BLOCK (1,10,10)
               }
            }
            SOURCE {
               FILE "a.h5"
               DATASET "A"
               SELECTION REGULAR_HYPERSLAB {
                  START (0,0,0)
                  STRIDE (1,1,1)
                  COUNT (H5S_UNLIMITED,1,1)
                  BLOCK (1,10,10)
```

How to deal with missing data?



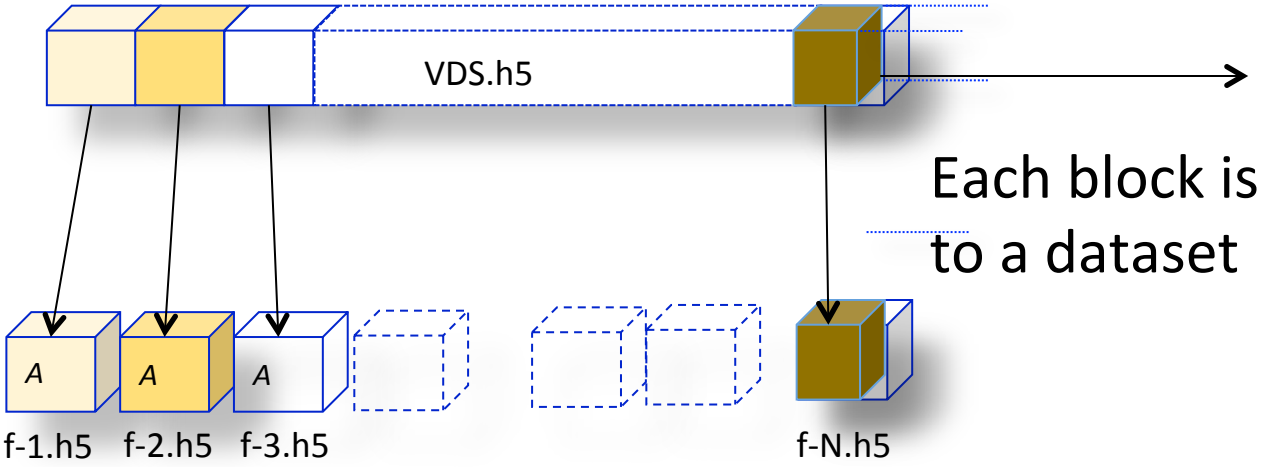
H5Pset_virtual_view sets extent to the position of the first missing plane or the last available. Missing planes will have fill values.

DEMO

https://svn.hdfgroup.org/hdf5/features/vds/examples/h5_vds-percival-unlim-maxmin.c

Unlimited Use Case – Infinite Block Count

VDS with unlimited dimension



Each block is mapped to a dataset

Source files;

Names are generated by the “printf” capability

Defining Mapping

```
start[0] = 0; start[1] = 0; start[2] = 0;
stride[0] = DIM0; stride[1] = 1; stride[2] = 1;
count[0] = H5S_UNLIMITED; count[1] = 1; count[2] = 1;
block[0] = DIM0;
block[1] = DIM1;
block[2] = DIM2;

status = H5Sselect_hyperslab (vspace, H5S_SELECT_SET,
                              start, stride, count, block);
status = H5Pset_virtual (dcpl, vspace, "f-%b.h5", "/A",
                         src_space);
```

Printf name generation

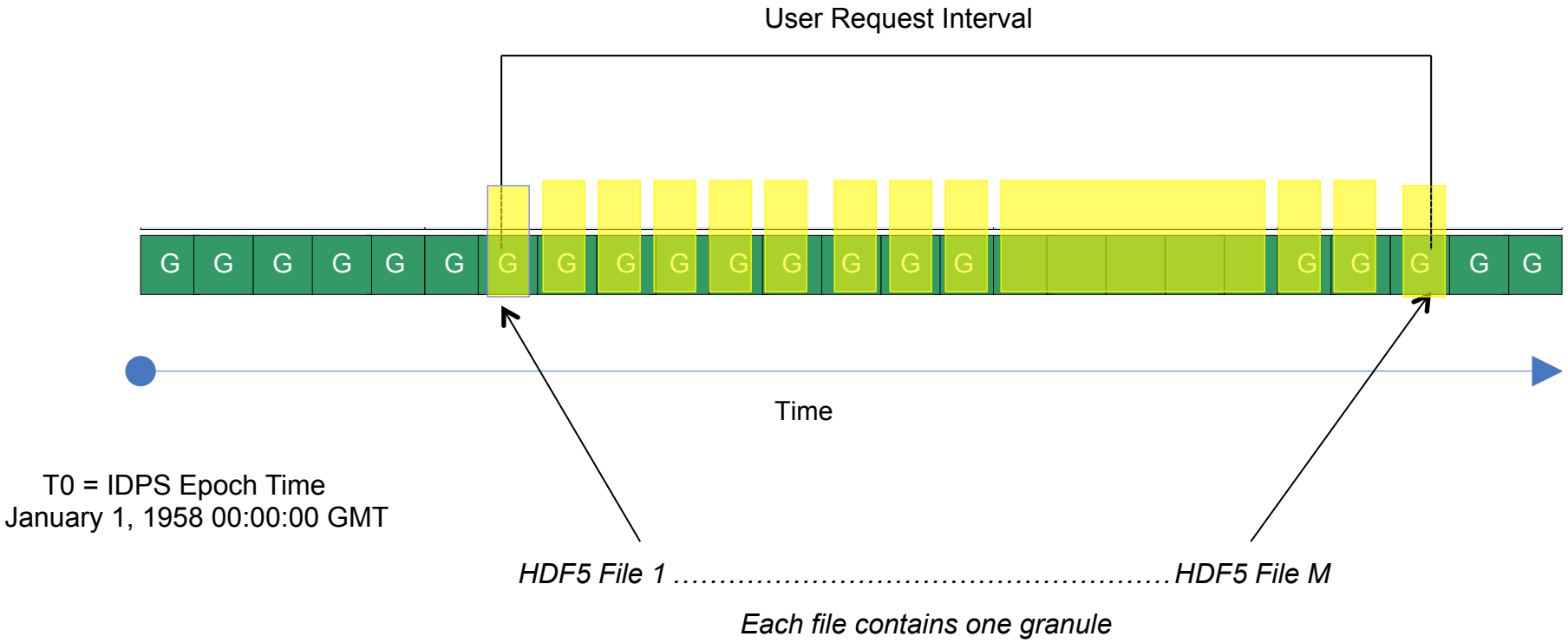
```
../hdf5/bin/h5dump -pH vds-eiger.h5
HDF5 "vds-eiger.h5" {
GROUP "/" {
   DATASET "VDS-Eiger" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SIMPLE { ( 0, 10, 10 ) / ( H5S_UNLIMITED, 10, 10 ) }
      STORAGE_LAYOUT {
         MAPPING 0 {
            VIRTUAL {
               SELECTION REGULAR_HYPERSLAB {
                  START (0,0,0)
                  STRIDE (5,1,1)
                  COUNT (H5S_UNLIMITED,1,1)
                  BLOCK (5,10,10)
               }
            }
            SOURCE {
               FILE "f-%b.h5"
               DATASET "/A"
               SELECTION ALL
            }
```

DEMO

https://svn.hdfgroup.org/hdf5/features/vds/examples/h5_vds-eiger.c

USING VDS FOR DATA AGGREGATION IN NAGG

nagg: Aggregation Example

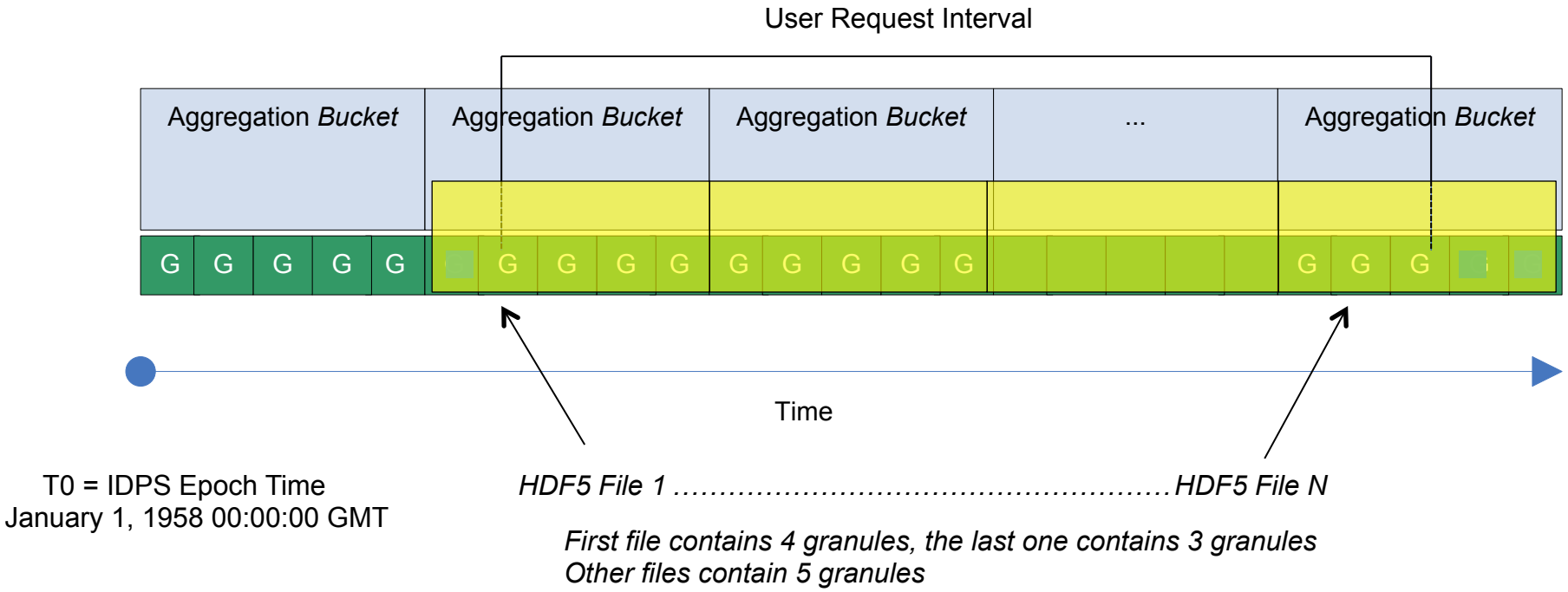


- User requests data from the IDPS system for a specific time interval
- Granules and products are packaged in the HDF5 files according to the request
- This example shows one granule per file for one product

nagg: Aggregation Example

Example: nagg –n 5 –t SATMS SATMS_npp_d2012040*.h5

Nagg copies data to the newly generated file(s).

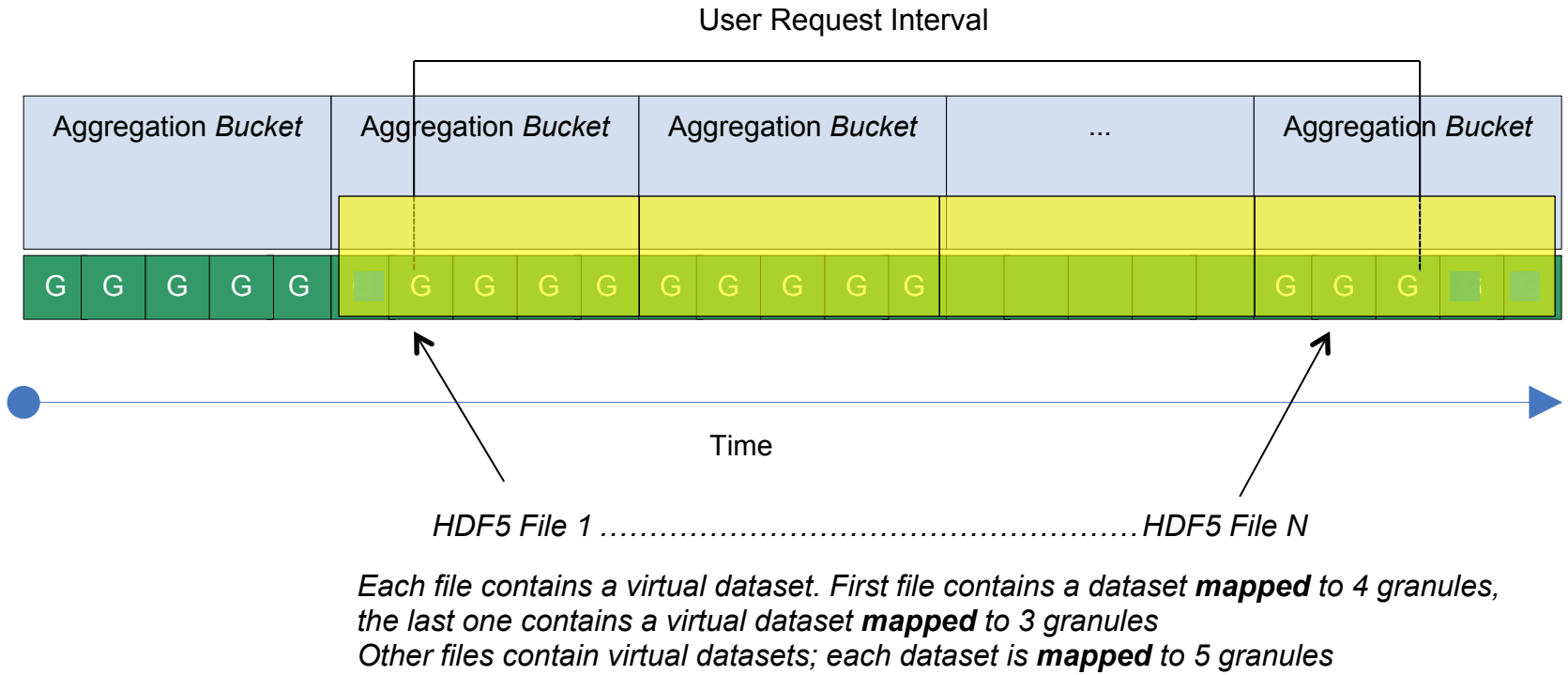


- Produced files co-align with the aggregation bucket start
- HDF5 files are 'full' aggregations (full, relative to the aggregation period)
- Geolocation granules are aggregated and packaged; see –g option for more control

Possible enhancement

Example: nagg –n 5 **–v** –t SATMS SATMS_npp_d2012040*.h5

Nagg with –v option doesn't copy data to the newly generated file(s).



- NO RAW DATA IS REWRITTEN
- Space savings
- No I/O performed on raw data

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!

?