HDF5
Virtual Dataset

Neil Fortner, Quincey Koziol, Elena Pourmal, Dana Robinson

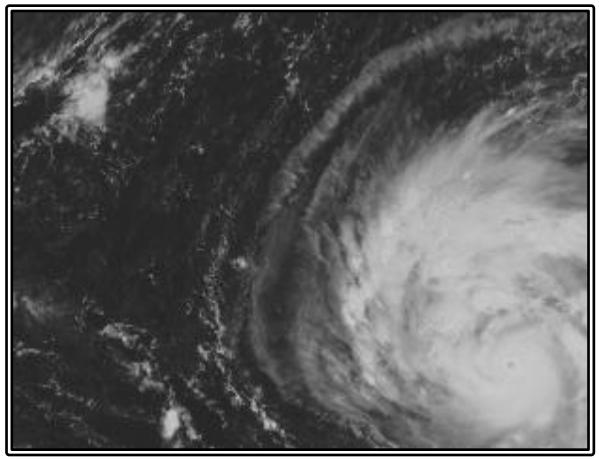
CHALLENGE

Challenge

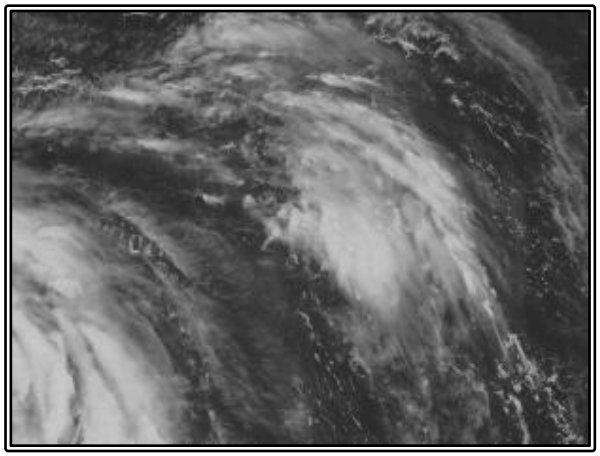
- How to view data stored across the HDF5 files as an HDF5 dataset on which normal operations can be performed?
  - High-level approach
    - Special library that applications like MATLAB and H5Py will need to use
    - Example : THREDDS Data Server based on OPeNDAP http://www.unidata.ucar.edu/software/thredds/current/tds/TDS.html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

TWO SIMPLE USE CASES

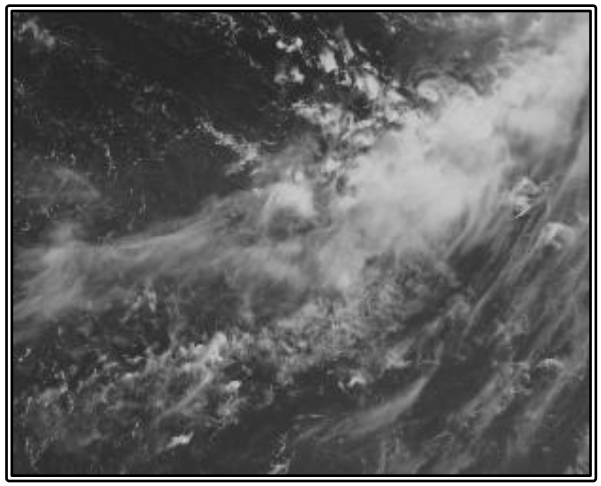
Collect data one way ….



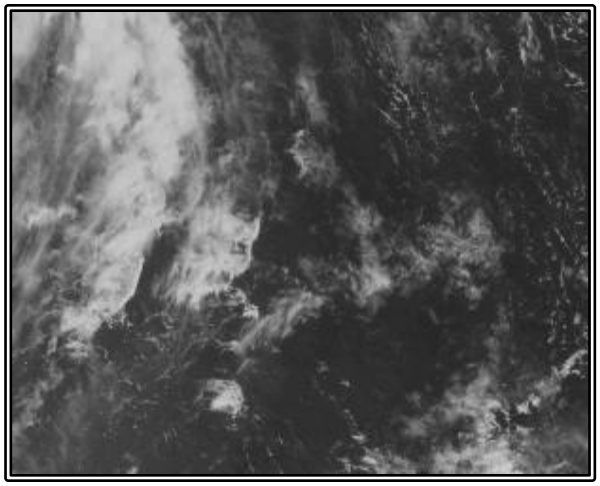
File: a.h5
Dataset /A



File: b.h5
Dataset /B



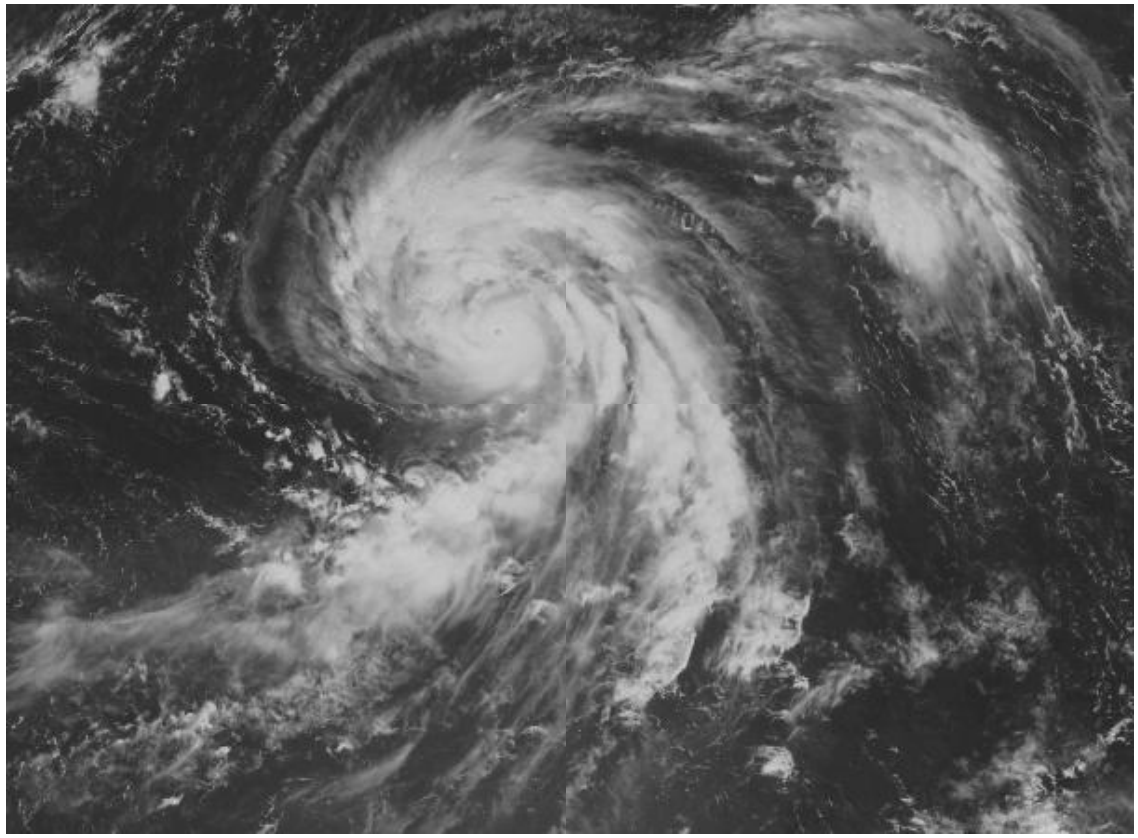
File: c.h5
Dataset /C



File: d.h5
Dataset /D

Present it in a different way…

Whole image



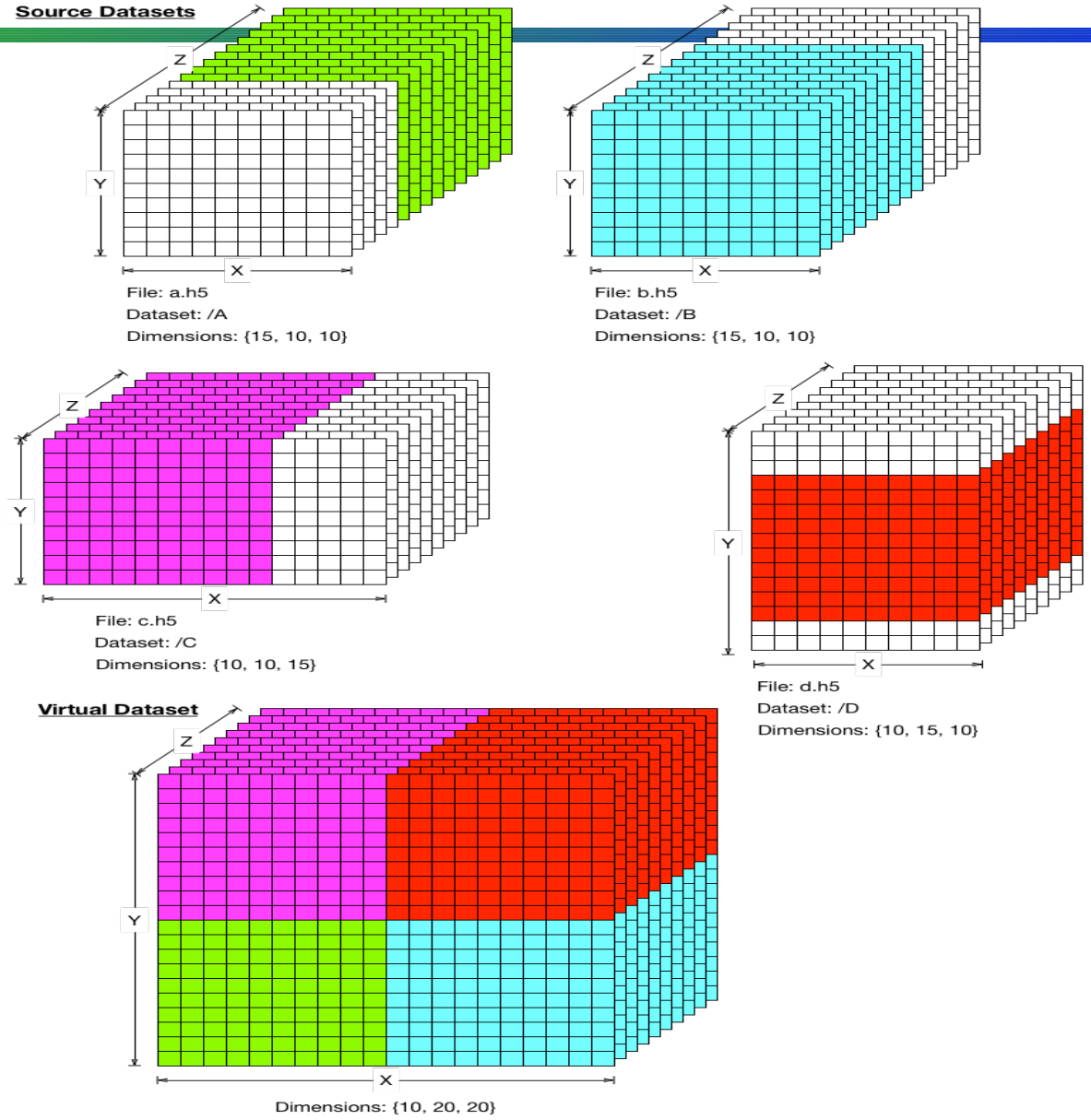
File: F.h5
Dataset /D

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

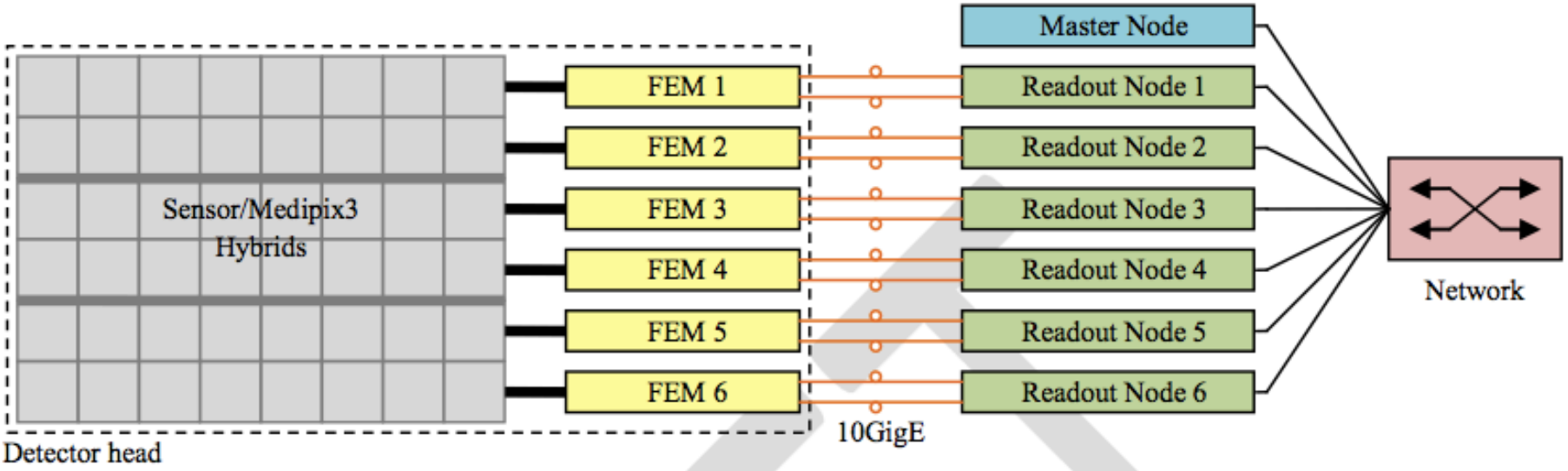


Figure 1: Excalibur hardware architecture.

Courtesy DLS

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

Excalibur Chip Layout and Gap detail

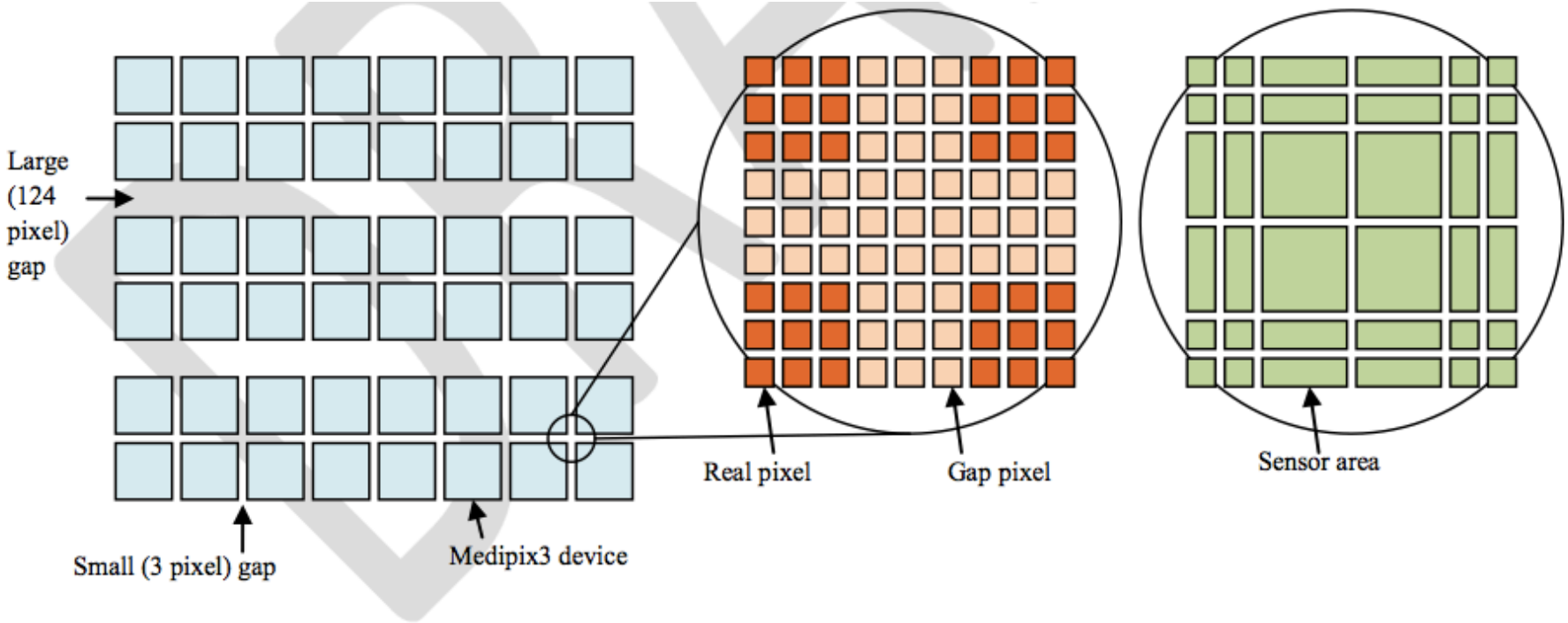
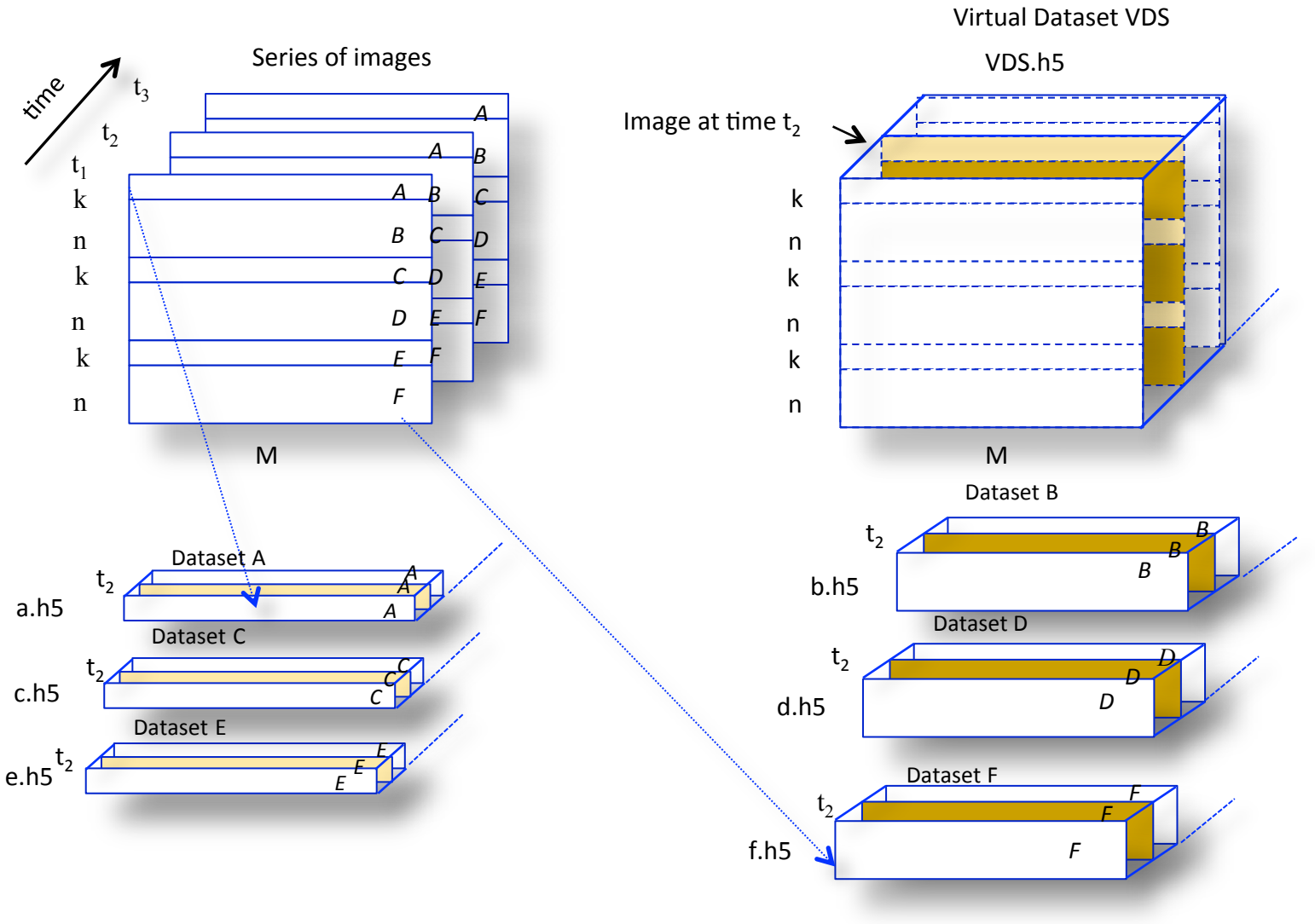


Figure 2. Excalibur Medipix3 chip layout and gap details.

Courtesy DLS

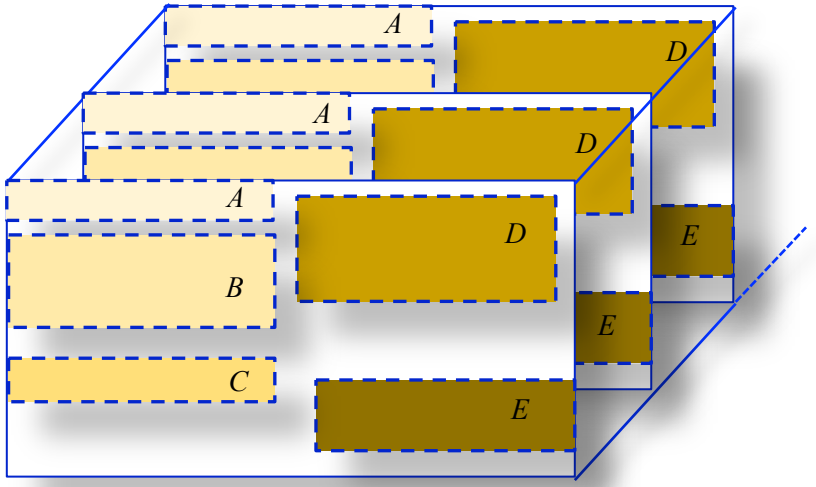
See Confluence - DLS – Virtual Dataset Phase 0 for the document

Unlimited Use Case



Use Case with Gaps

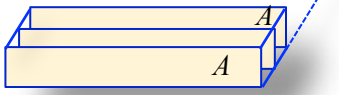
Virtual Dataset VDS with “gaps”



VDS.h5

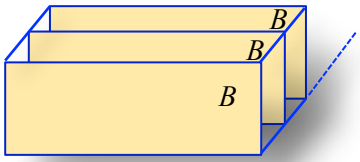
Dataset A

a.h5



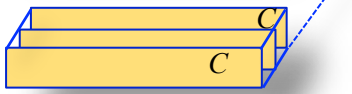
Dataset B

b.h5



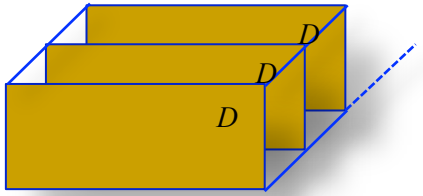
Dataset C

c.h5



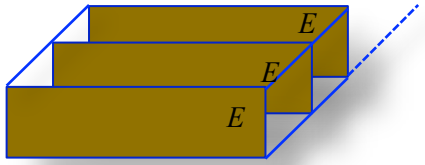
Dataset D

d.h5

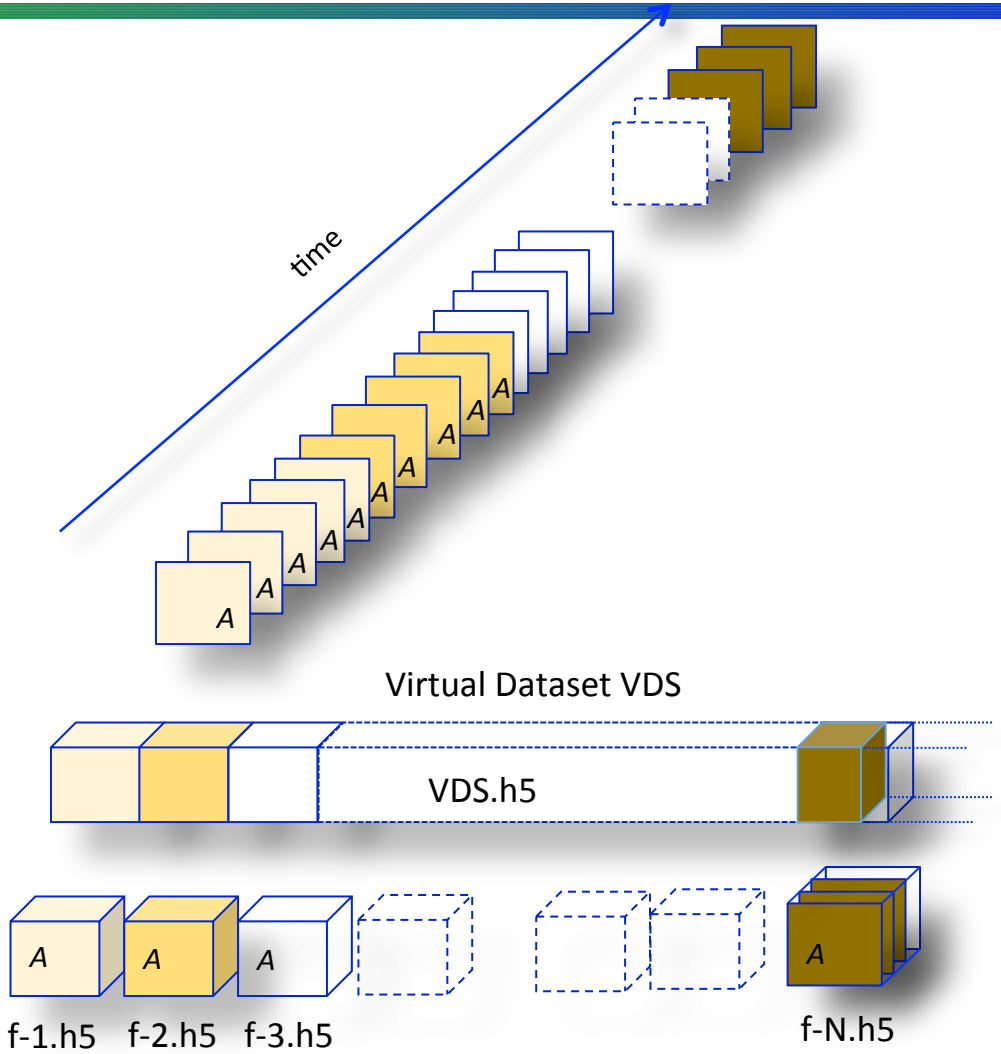


Dataset E

e.h5

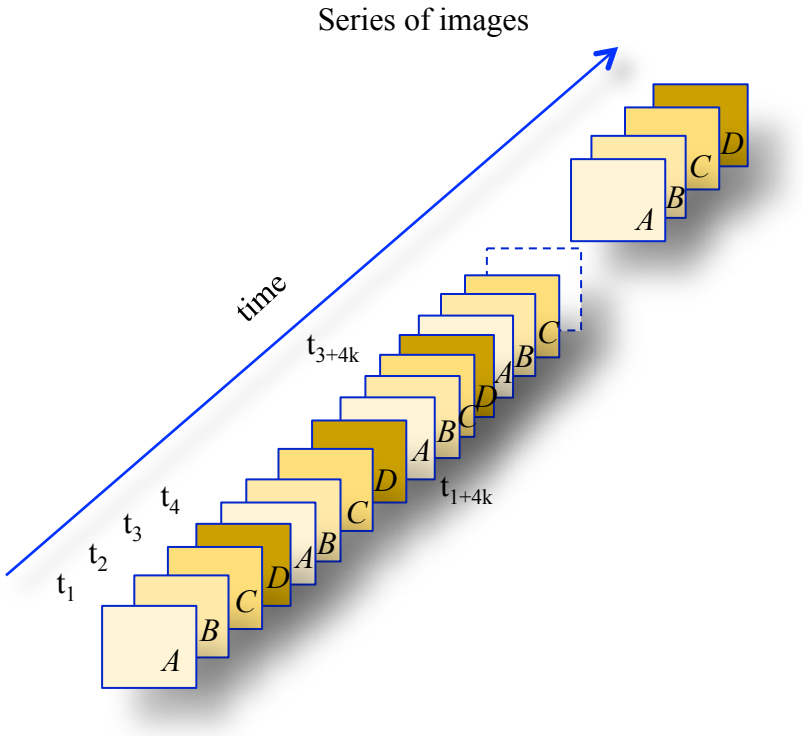


“Printf-type” Source Generation

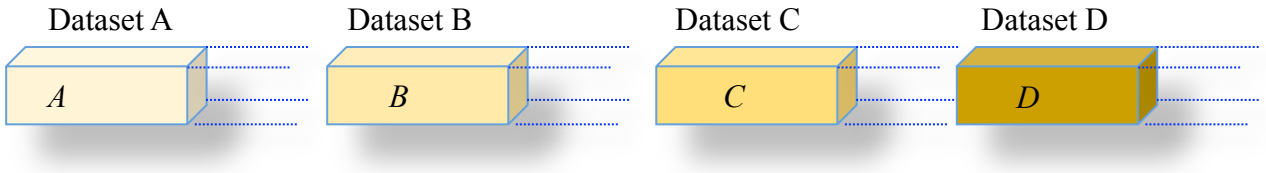


File names are generated by the “printf” capability

Use Case with Interleave Planes



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



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

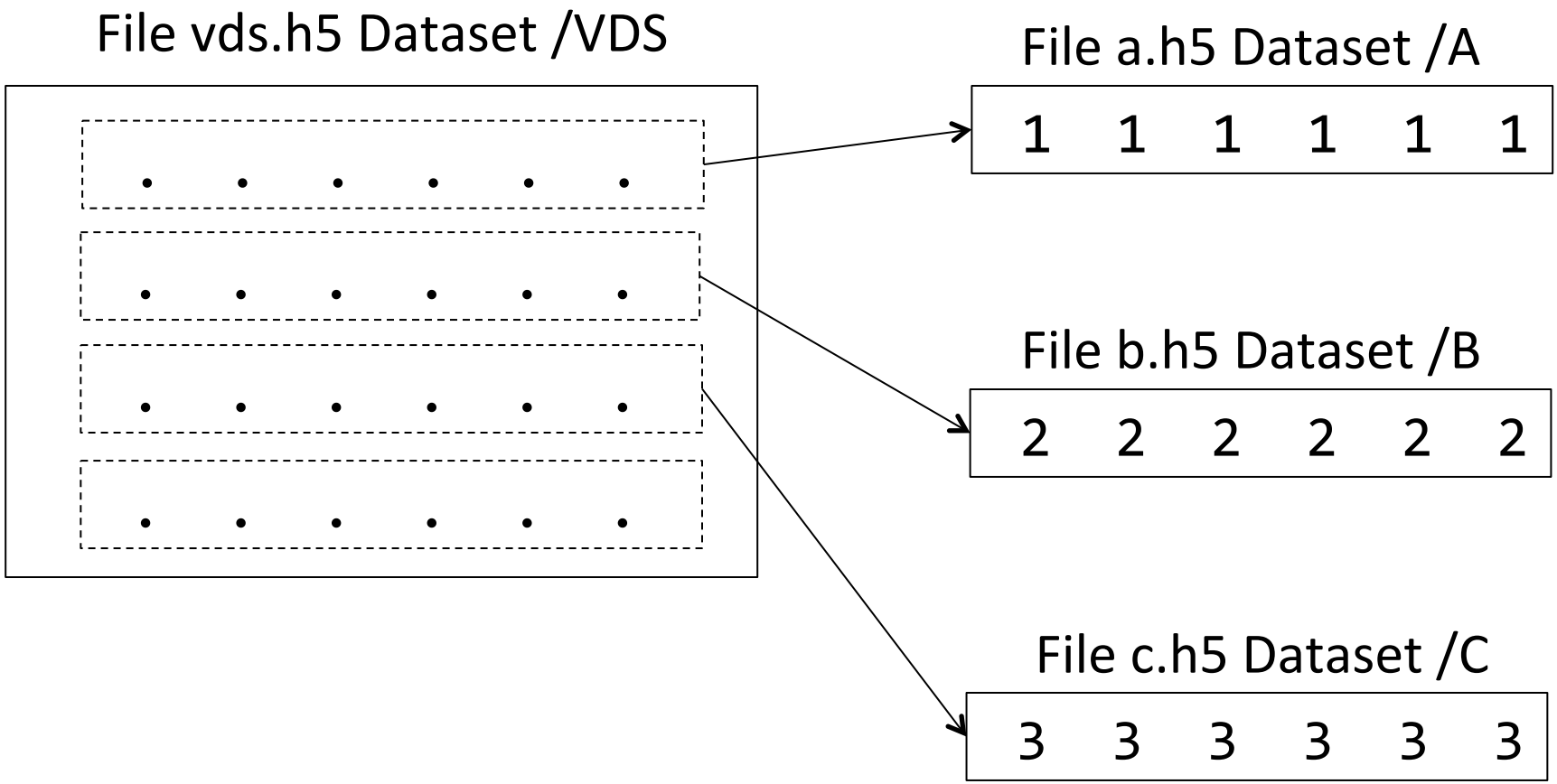
http://www.bigdata.org/HDF5/docNewFeatures/NewFeaturesVirtualDatasetDocs.html

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

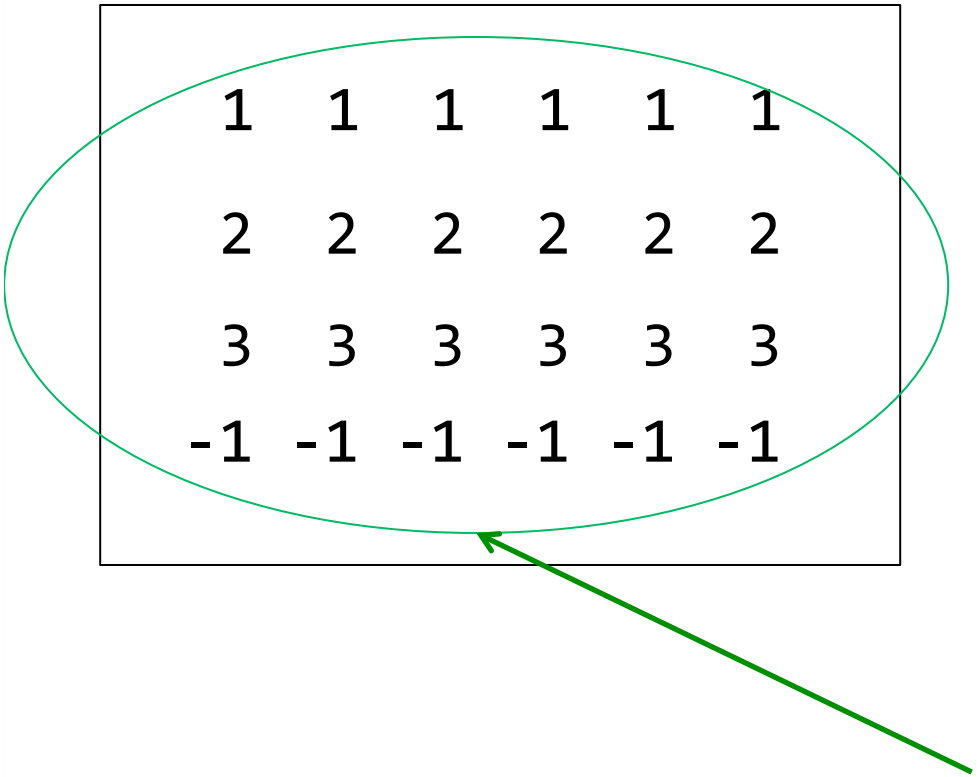


Defining Mapping

```
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

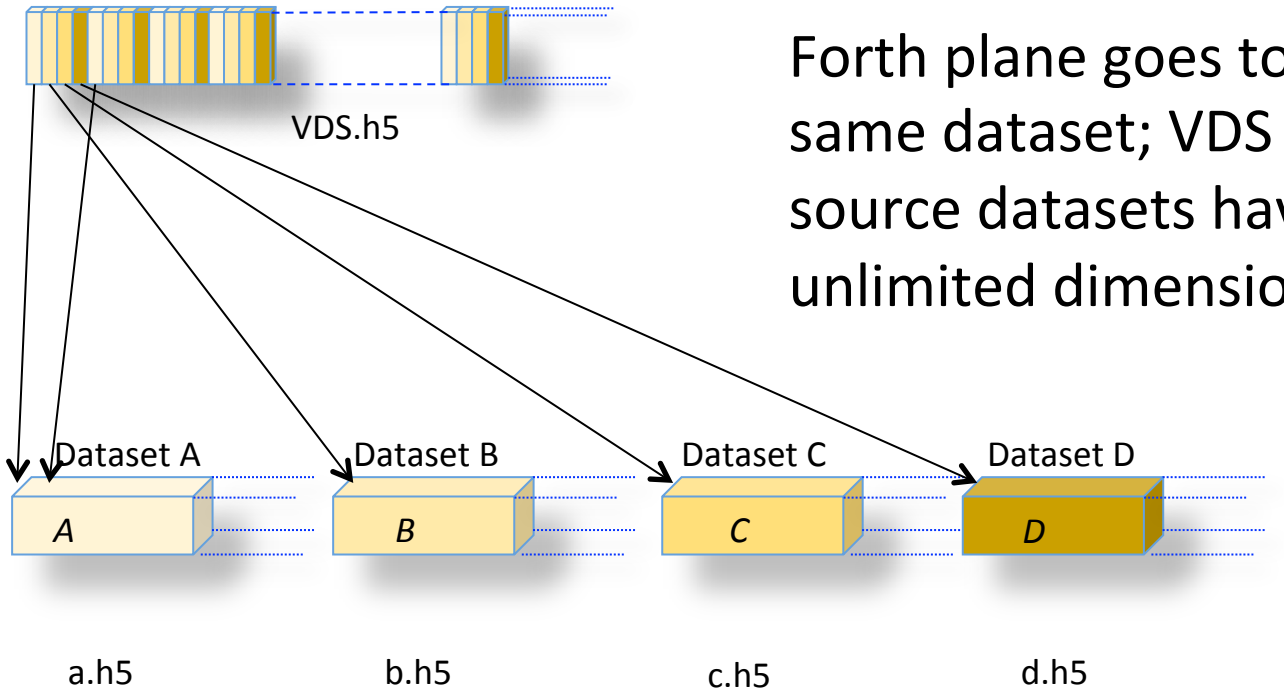
```
HDF5 "vds.h5" {
GROUP "/" {
   DATASET "VDS" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SIMPLE { ( 4, 6 ) / ( 4, 6 ) }
      STORAGE_LAYOUT {
         MAPPING 0 {
            VIRTUAL {
               SELECTION REGULAR_HYPERSLAB {
                  START (0,0)
                  STRIDE (1,1)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

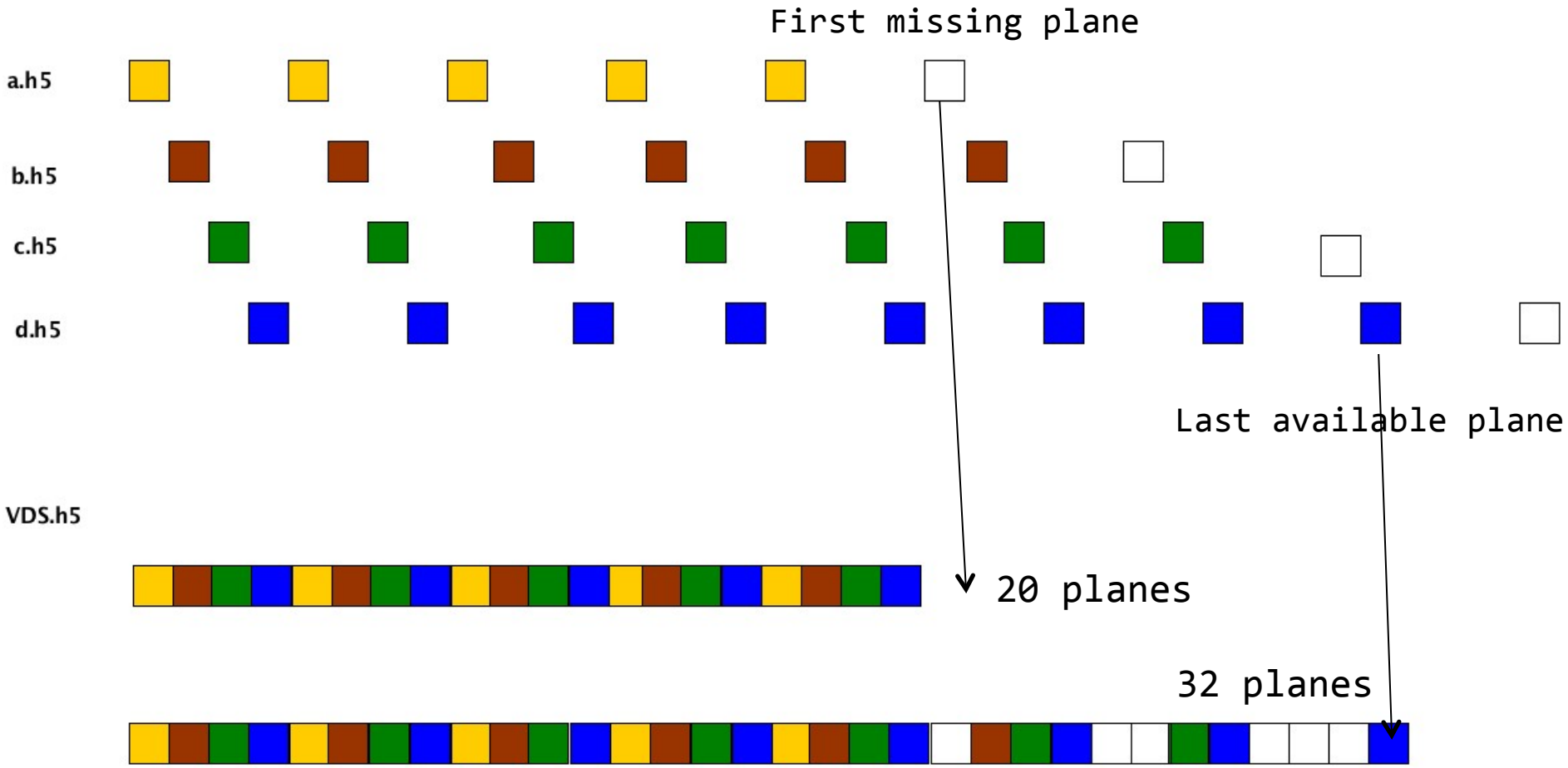


Forth plane goes to the same dataset; VDS and source datasets have unlimited dimension

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]++;
}
```

How to deal with missing data?



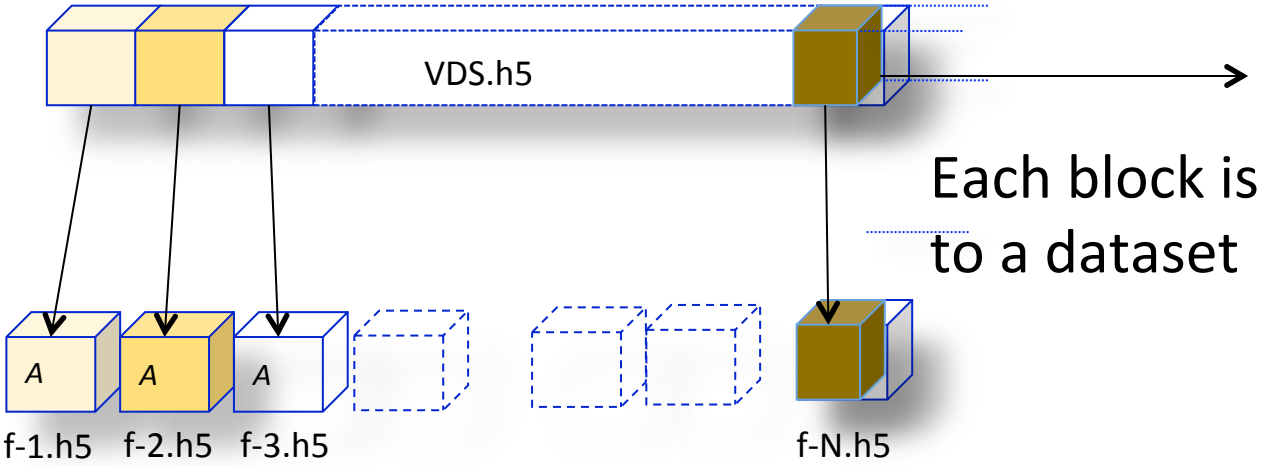
H5Pset_virtual_view sets extent to the position of the first missing plane or the last available. Missing planes will have fill values.

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);
```

Thank you!

?