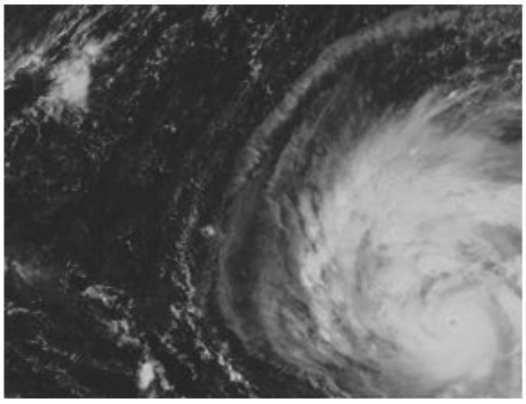
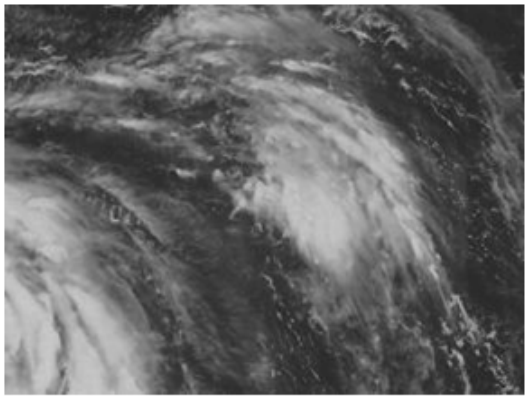
Introduction to the Virtual Dataset - VDS

The HDF5 Virtual Dataset (VDS) feature enables users to access data in a collection of HDF5 files as a single HDF5 dataset and to use the HDF5 APIs to work with that dataset.

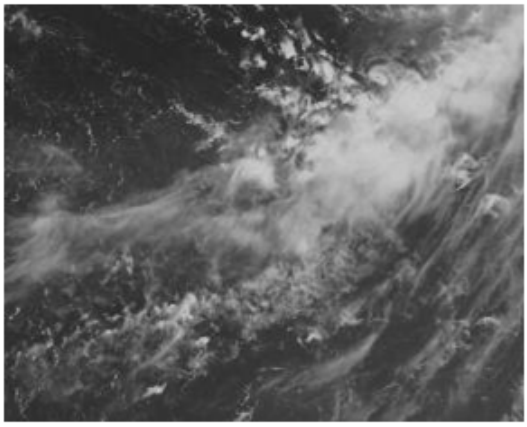
For example, your data may be collected into four files:



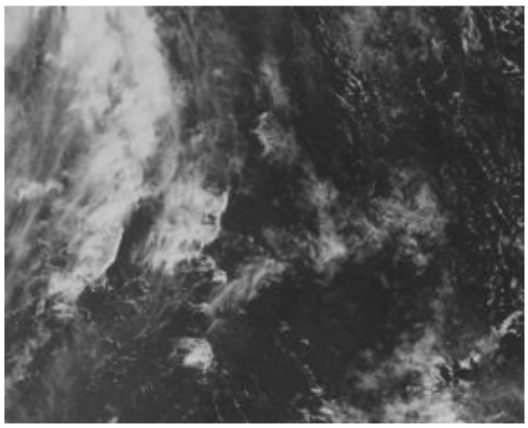
File: a.h5
Dataset /A



File: b.h5
Dataset /B

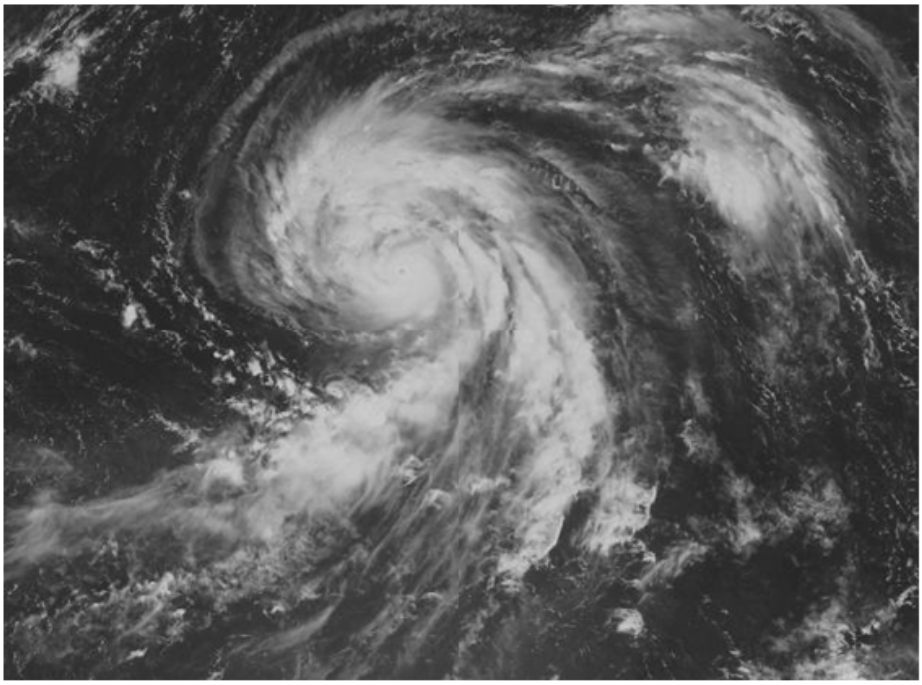


File: c.h5
Dataset /C



File: d.h5
Dataset /D

You can map the datasets in the four files into a single VDS that can be accessed just like any other dataset:



File: F.h5
Dataset /D

The mapping between a VDS and the HDF5 source datasets is persistent and transparent to an application. If a source file is missing the fill value will be displayed.

See the Virtual (VDS) Documentation for complete details regarding the VDS feature.

The VDS feature was implemented using hyperslab selection (H5S_SELECT_HYPERSLAB). See the tutorial on Reading From or Writing to a Subset of a Dataset for more information on selecting hyperslabs.

Programming Model

To create a Virtual Dataset you simply follow the HDF5 programming model and add a few additional API calls to map the source code datasets to the VDS.

Following are the steps for creating a Virtual Dataset:

1. Create the source datasets that will comprise the VDS
2. Create the VDS: Define a datatype and dataspace (can be unlimited)
   Define the dataset creation property list (including fill value)
   (Repeat for each source dataset) Map elements from the source dataset to elements of the VDS:
   - Select elements in the source dataset (source selection)
   - Select elements in the virtual dataset (destination selection)
   - Map destination selections to source selections (see Functions for Working with a VDS)

   Call H5Dcreate using the properties defined above
3. Access the VDS as a regular HDF5 dataset
4. Close the VDS when finished

Functions for Working with a VDS

The H5P_SET_VIRTUAL API sets the mapping between virtual and source datasets. This is a dataset creation property list. Using this API will change the layout of the dataset to H5D_VIRTUAL. As with specifying any dataset creation property list, an instance of the property list is created, modified, passed into the dataset creation call and then closed:

```
dcpl = H5Pcreate (H5P_DATASET_CREATE);

  src_space = H5screate_simple ...
  status = H5Sselect_hyperslab (space, ...
  status = H5Pset_virtual (dcpl, space, SRC_FILE[i], SRC_DATASET[i], src_space);

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

  status = H5Pclose (dcpl);
```

There are several other APIs introduced with Virtual Datasets, including query functions. For details see the complete list of HDF5 library APIs that support Virtual Datasets

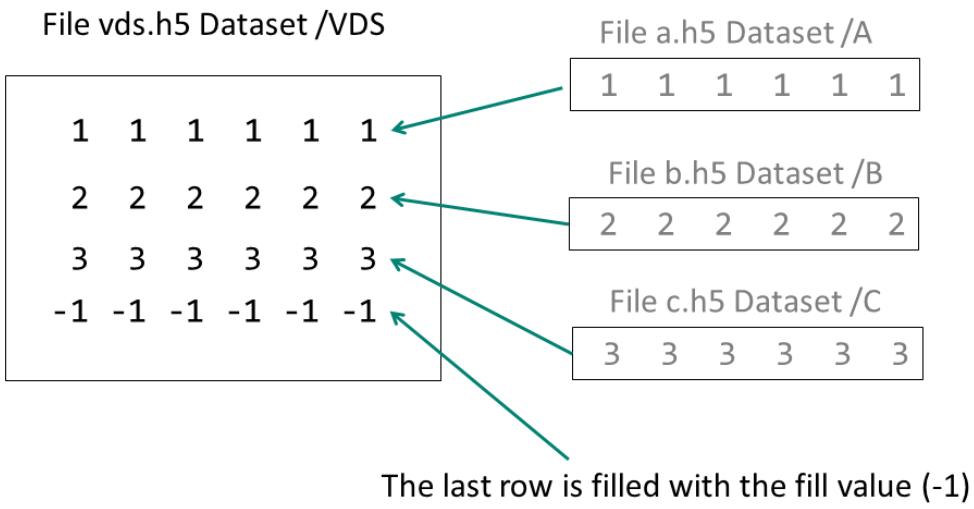
Limitations

This feature requires HDF5-1.10.
The number of source datasets is unlimited. However, there is a limit on the size of each source dataset.

Programming Examples

Example 1

This example creates three HDF5 files, each with a one-dimensional dataset of 6 elements. The datasets in these files are the source datasets that are then used to create a 4 x 6 Virtual Dataset with a fill value of -1. The first three rows of the VDS are mapped to the data from the three source datasets as shown below:



In this example the three source datasets are mapped to the VDS with this code:

```
src_space = H5Screate_simple (RANK1, dims, NULL);
    for (i = 0; i < 3; i++) {
 start[0] = (hsize_t)i;
 /* Select i-th row in the virtual dataset; selection in the source datasets is the same. */
 status = H5Sselect_hyperslab (space, H5S_SELECT_SET, start, NULL, count, block);
 status = H5Pset_virtual (dcpl, space, SRC_FILE[i], SRC_DATASET[i], src_space);
    }
```

After the VDS is created and closed, it is reopened. The property list is then queried to determine the layout of the dataset and its mappings, and the data in the VDS is read and printed.

This example is in the HDF5 source code and can be obtained from here:

C Example

For details on compiling an HDF5 application: [Compiling HDF5 Applications]

Example 2

This example shows how to use a C-style printf statement for specifying multiple source datasets as one virtual dataset. Only one mapping is required. In other words only one H5P_SET_VIRTUAL call is needed to map multiple datasets. It creates a 2-dimensional unlimited VDS. Then it re-opens the file, makes queries, and reads the virtual dataset.

The source datasets are specified as A-0, A-1, A-2, and A-3. These are mapped to the virtual dataset with one call:

```
status = H5Pset_virtual (dcpl, vspace, SRCFILE, "/A-%b", src_space);
```

The %b indicates that the block count of the selection in the dimension should be used.

C Example

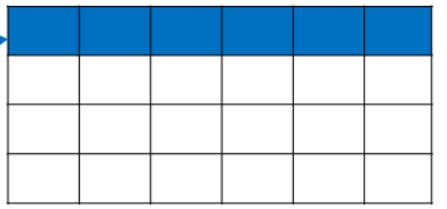
For details on compiling an HDF5 application: [Compiling HDF5 Applications]

Using h5dump with a VDS

The h5dump utility can be used to view a VDS. The h5dump output for a VDS looks exactly like that for any other dataset. If h5dump cannot find a source dataset then the fill value will be displayed.

You can determine that a dataset is a VDS by looking at its properties with h5dump -p. It will display each source dataset mapping, beginning with *Mapping 0*. Below is an excerpt of the output of h5dump -p on the vds.h5 file created in Example 1.You can see that the entire source file a.h5 is mapped to the first row of the /VDS dataset:

```
$ 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
            }
         }
```



Entire dataset is selected