**Data Filtering in HDF5**

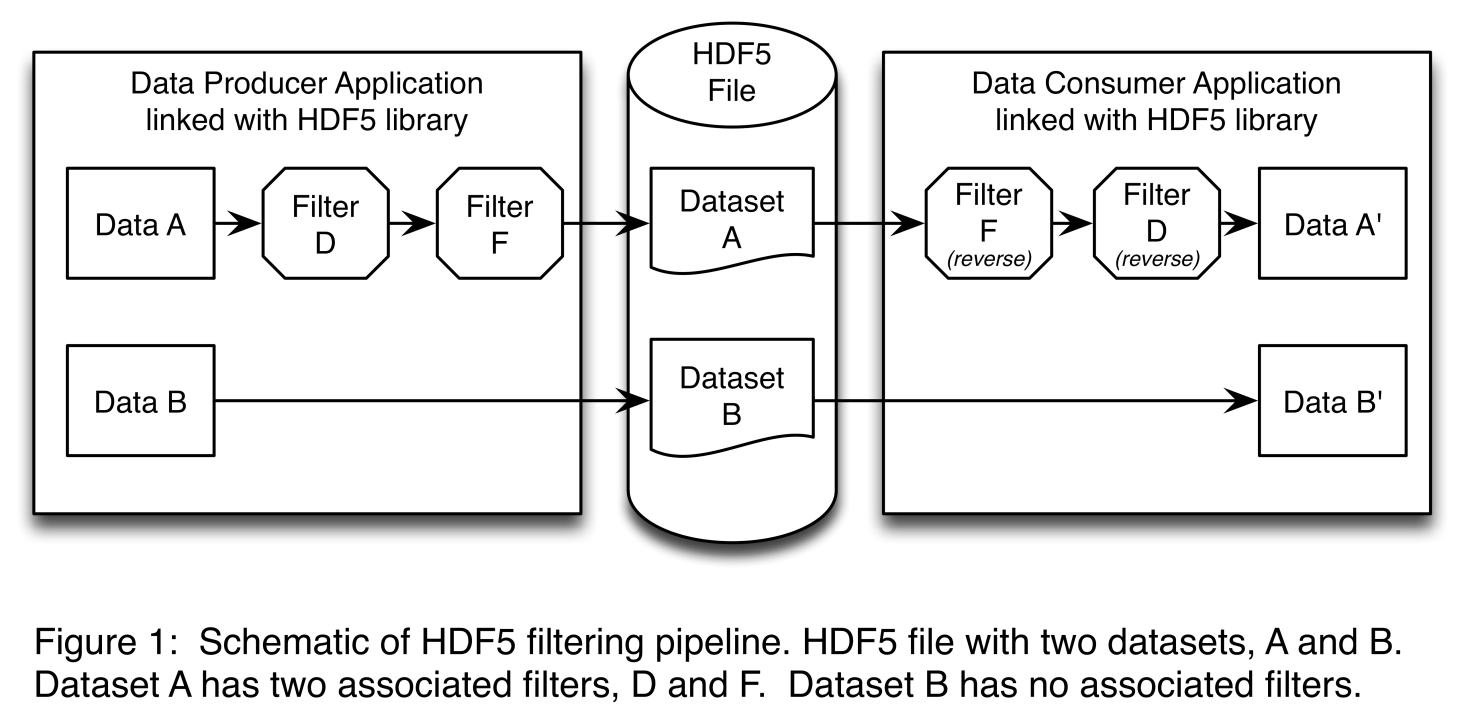
Ruth Aydt (aydt@hdfgroup.org)

July 30, 2008

The HDF5 library supports a *data filtering* pipeline that allows standard and customized processing of raw data during I/O operations. A small number of predefined filters for compression and error checking are included with the HDF5 distribution. Predefined filters are available to all HDF5 users.

In addition to the predefined filters, applications can create, register, and share *custom data filters*. Custom filters are not distributed as part of the HDF5 library, but are linked separately and made available to the library at run-time when an application registers them via the *H5Zregister()* function. Data that is written with a custom filter applied can only be read by applications that have linked to and registered the same custom filter.

An HDF5 file can contain multiple datasets. Data filters are associated with a particular dataset when the dataset is initially defined (created), and multiple filters can be associated with a single dataset. It is possible to create datasets that do not have any filters associated with them. Every time raw data is written to a dataset, the filter(s) associated with the dataset will be applied automatically by the HDF5 library before the data is stored in the file. Any time data is read from a dataset, the filters(s) associated with the dataset are applied automatically as the data is retrieved from the file, reversing the effects of the filter operations performed when the data was stored. Figure 1 illustrates an HDF5 data filtering pipeline.



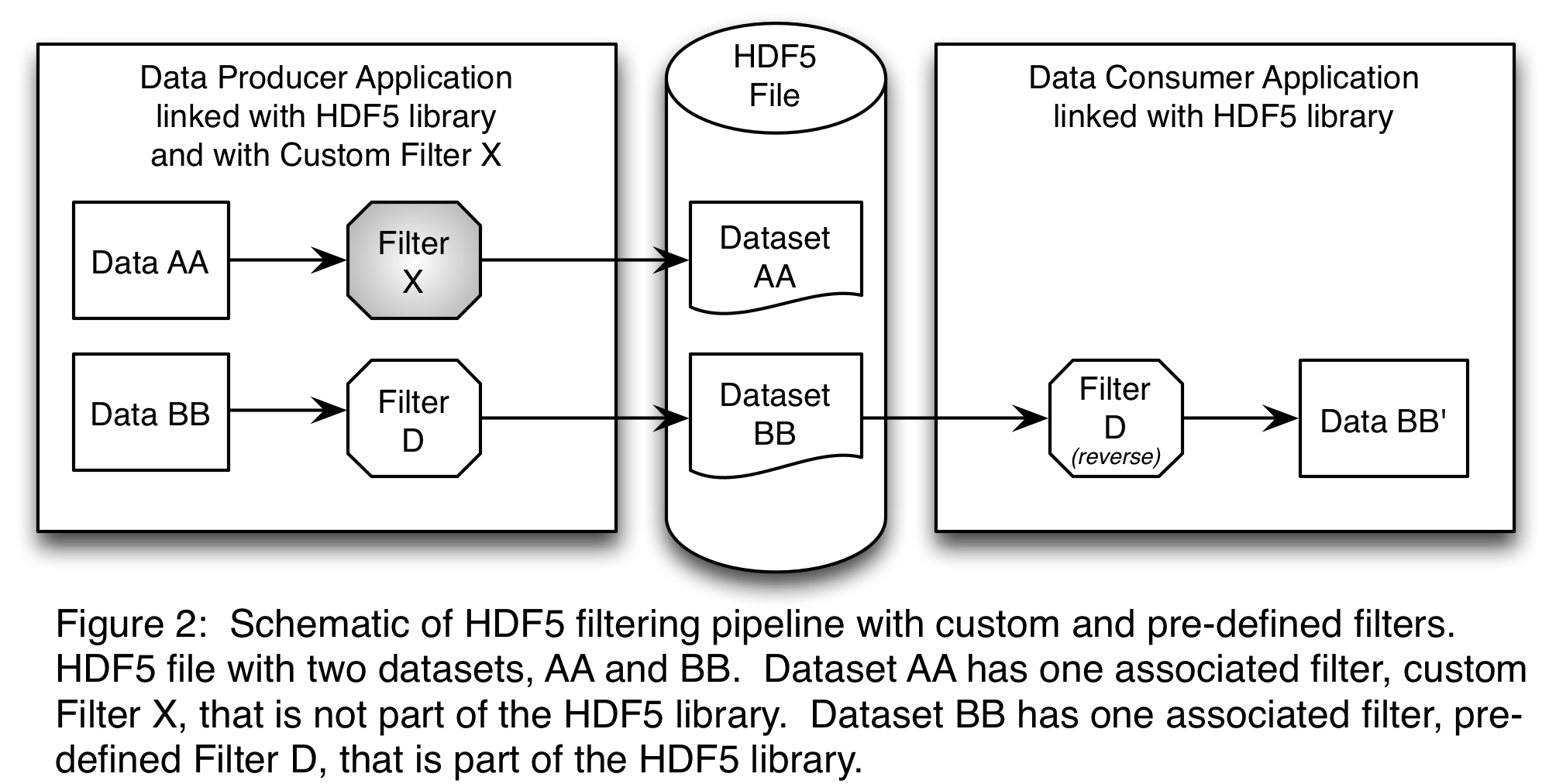
In the example shown in Figure 1, the Data Producer Application writes to an HDF5 file with two datasets, A and B. Dataset A had two filters, D and F, associated with it when it was defined (created). Let’s assume Filter D is the pre-defined DEFLATE (<http://tools.ietf.org/html/rfc1951>) compression filter and Filter F is the pre-defined Fletcher-32 (<http://tools.ietf.org/html/rfc1146>) checksum filter for error checking. Dataset B has no associated filters.

When the Producer Application calls *H5Dwrite()* for dataset A, the raw data in the Data A buffer will be compressed by Filter D and then a checksum will be computed by Filter F. The resulting compressed data will be stored in the HDF5 file along with the computed checksum. When the Consumer Application calls *H5Dread()* for dataset A, Filter F will compute the checksum for the stored (compressed) data and compare the computed value to the checksum that was saved with the data. If there is a mismatch, an error will be returned. If the checksums agree, Filter D will decompress the data and the raw data values will be available to the Consumer Application in the Data A’ buffer.

When the Producer Application calls *H5Dwrite()* for dataset B, the raw data in the Data B buffer will be stored in the HDF5 file. When the Consumer Application calls *H5Dread()* for dataset B, the stored data will be available to the Consumer Application in the Data B’ buffer. *Note that from the applications’ perspective the H5Dwrite() and H5Dread() calls are the same for datasets A and B. The filters for dataset A are automatically applied by the HDF5 library, having been associated with the dataset when it was created.*

The HDF5 data filtering pipeline processes data in equally-sized *chunks* that are stored and retrieved separately. The chunk size for a given dataset is specified when the dataset is defined (created). The subdivision of data into chunks provides very efficient access to subsets of the stored data, and reduces overall memory requirements for large datasets. *From the applications’ perspective the H5Dwrite() and H5Dread() calls are unchanged when chunked data storage is used.*

Within a given HDF5 file, different datasets may have different filters associated with them. If one dataset in the file has a custom filter associated with it that is not linked with an application reading the file, the application will not be able to read the raw data in that dataset. The application will, however, be able to read datasets in the file that were created without filters, as well as datasets whose associated filters are available to the application. This scenario is illustrated in Figure 2.



In the example shown in Figure 2, the Data Producer Application is linked with the Custom Filter X, and registers it with the HDF5 library using *H5Zregister()*. When the Producer Application calls *H5Dwrite()* for dataset AA, the raw data in the Data AA buffer will be processed by Custom Filter X before being stored in the HDF5 file. When the Producer Application calls *H5Dwrite()* for dataset BB, the pre-defined DEFLATE filter discussed earlier will be applied to the raw data in the Data BB buffer, and the resulting compressed data will be stored in the HDF5 file.

When the Consumer Application calls *H5Dread()* for dataset BB, Filter D will decompress the data and the raw data values will be available to the Consumer Application in the Data BB’ buffer. The Consumer Application in this example was not linked with the Custom Filter X, and therefore will be unable to read the data stored in Dataset AA.