RFC: Splitter_VFD

John Mainzer
Binh-Minh Ribler

Functionally, the objective of the Splitter VFD is to relay all VFD calls to an underlying VFD, and also send all write calls to a second VFD.

The splitter VFD is an essential component of the rsync / mirror VFD prototype being developed under the EOD contract. However, it is a sufficiently useful general-purpose component that it deserves its own RFC.

1 Introduction

The immediate motivation for developing the splitter VFD follows from the following architectural diagram from the rsync / mirror VFD sketch design:

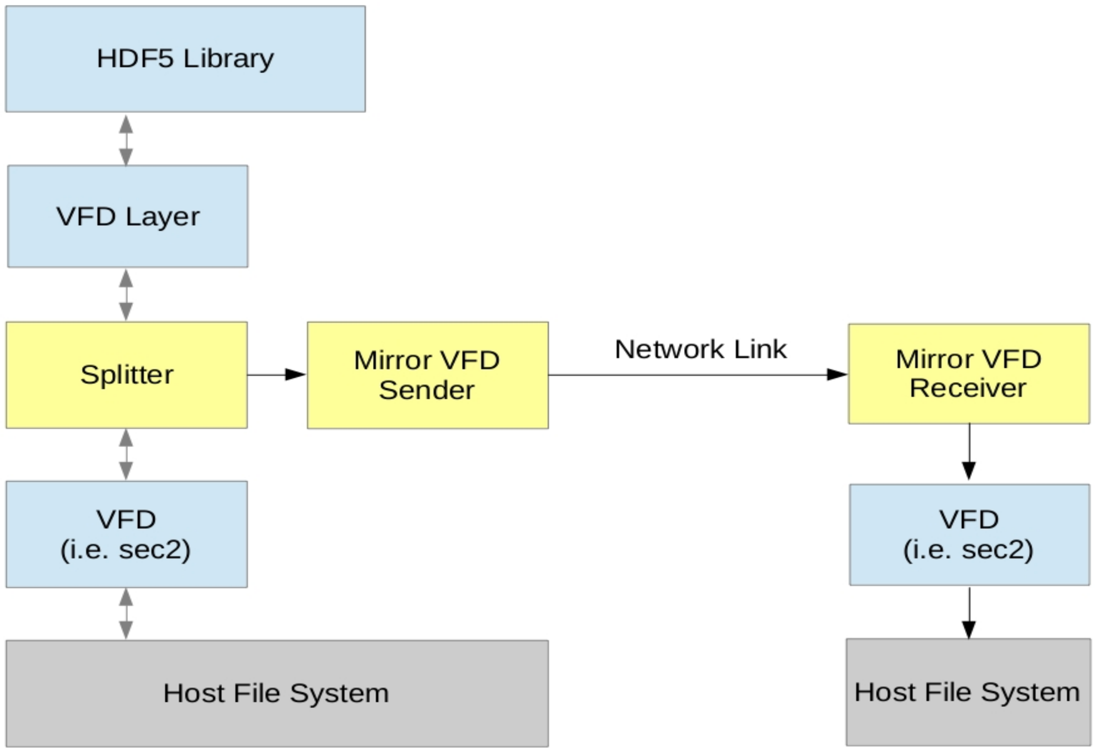

Figure 1 Block diagram illustrating the Mirror VFD design. Blue boxes represent existing code, and yellow boxes code that must be designed and implemented.

As can be seen from the above diagram, the splitter VFD has two channels to underlying VFDs. Call the channel to the VFD that accesses the local file system the R/W channel, and the channel that relays writes to the Mirror VFD the W/O channel. With this terminology in hand, we can summarize the functional requirements of the of the splitter VFD as follows:

- Pass all VFD calls to the R/W channel, and pass the results back up to the caller.
- Pass the open, write, truncate, close, set_eoa, and alloc calls to the W/O channel as well.

The remainder of this RFC expands upon the above requirements, and the issues involved in their implementation.

In closing, note that there is no requirement that the VFD on the W/O channel be the Mirror VFD Sender – for testing purposes, we will use another sec2 VFD, and one could imagine using a object store based VFD to archive the HDF5 file off site as it is generated.

2 Cycle of Operation

At the gross level, the cycle of operation of the splitter VFD can be broken into the following phases:

- Configuration of the splitter VFD and the underlying VFDS
- File Open
- File I/O
- File Close

These phases are discussed in the following sections.

2.1 Configuration

Configuring the splitter VFD is mostly a matter of telling the HDF5 library how to setup the splitter VFD when opening the file, and telling the splitter VFD how to setup the underlying VFDs. This will be done via a new splitter driver info property, which will specify:

1. R/W channel VFD selection and configuration in the form of a FAPL with the appropriate .
2. W/O channel VFD selection and configuration in the form of a FAPL.
3. File name to pass to the VFD on the W/O channel on file open.
4. Error reporting configuration for the W/O channel
5. Any splitter VFD specific configuration.

2.2 File Open

The file open process for the splitter VFD is complicated by the fact that HDF5 will frequently open a file tentatively, close it, and then re-open it for real.

This leaves the splitter VFD with two options:

1) Setup the W/O channel on open – and thus close and re-open it again in many cases.
2) Delay setting up the W/O channel until the first actual I/O call is received by the splitter VFD.

While one can make an argument for either option depending on the application, for simplicity in the initial implementation, we will go with the first option for now – with the proviso that we may re-visit this decision.

The file open procedure starts with a query operation on the VFD. Using the splitter VFD FAPL entry, determine the VFD on the R/W path, pass the query to that VFD, and return the results.

This should be followed by an open call, which should be handled as follows:

1) When the open VFD call is received, first check the flags to see if the file is being opened R/O – if so, fail.
2) Use the splitter VFD FAPL entry in the supplied FAPL to determine the R/W channel VFD.
3) Relay the open call to the VFD configured for the R/W channel, substituting the R/W channel FAPL ID for the supplied value. On success, store the returned pointer to H5FD_t as the R/W channel VFD. On failure, report failure and exit.
4) Use the splitter VFD FAPL entry in the supplied FAPL to determine the W/O channel VFD.
5) Relay the open call to the VFD configured for the W/O channel, substituting the file name configured for the W/O channel for the name parameter, and the W/O channel FAPL ID for the supplied values. On success, store the returned pointer to H5FD_t as the W/O VFD. On error, ignore it, log it and ignore it, or fail as configured.
6) Allocate, initialize, and return the instance of H5FD_splitter_t containing the data required to operate the splitter VFD. As with other VFDs, the first field in this structure is an instance of H5FD_t.

Note that the above does not address cleanup of errors – which should be handled as cleanly as possible.

2.3 File I/O

The file I/O phase is relatively straightforward – all VFD API calls must be relayed to the R/W channel, and the results reported to the caller. In addition, write, truncate, set_eoa, and alloc[1] calls must be relayed to the W/O VFD as well. Any errors on the W/O channel are ignored, logged and ignored, or returned via the usual error stack depending on configuration.

2.4 File Close

The file close phase differs from the previous phases in that the VFD close call must be relayed to the VFDs on both channels, regardless of the results. Proceed as follows:

1) Relay the VFD close call to the VFD on the R/W channel, and make note of the results.
2) If the VFD on the W/O channel exists, relay the close call to it. Ignore, ignore and log, or report any errors in this operation as configured.

[1] The set_eoa and alloc calls are necessary, as they set the EOA. The EOA is needed on the W/O channel as the truncate call does not take a file length, but instead truncates the file to the current EOA.

3) Free the instance of H5FD_splitter_t.
4) Return the results of the close of the VFD on the R/W channel.

3 API Additions

The only new API calls needed for the splitter VFD are the get and set routines for the splitter VFD FAPL entry. As the set of parameters is fairly large and will likely change as the splitter VFD is optimized[2], we will package them in the following structure:

```
/*******************************************************************************
 *
 * struct H5F_splitter_vfd_config_t
 *
 * Instances of H5F_splitter_vfd_config_t are used to configure the splitter
 * VFD.  Note that the configuration of the underlying VFDs on both the R/W and
 * W/O channels is specified by inserting the appropriate FAPL entries in
 * the R/W and W/O FAPLs respectively.
 *
 * The fields of H5F_splitter_vfd_config_t are discussed below:
 *
 * version: Integer field indicating the version of the H5F_splitter_vfd_config_t
 *         structure used.  This field must always be set to a known version
 *         number.  The most recent version of the structure will always be
 *         H5F__CURR_SPLITTER_VFD_CONFIG_VERSION.
 *
 * rw_fapl_id: ID of the FAPL containing the VFD selection and configuration data
 *         for the R/W channel of the splitter VFD.  This FAPL ID is substituted
 *         for the supplied FAPL ID when relaying the open command to the R/W
 *         channel.  It is also used to identify the desired VFD for the R/W
 *         channel.
 *
 * wo_fapl_id: ID of the FAPL containing the VFD selection and configuration data
 *         for the W/O channel of the splitter VFD.  This FAPL ID is substituted
 *         for the supplied FAPL ID when relaying the open command to the W/O
 *         channel.  It is also used to identify the desired VFD for the W/O
 *         channel.
 *
 * wo_path: Array of char of length MAX_PATH + 1.  It is used to specify the
 *         file name with which to replace the name parameter when passing the
 *         vfd open call to the W/O channel VFD.
 *
 * log_file_path Array of char of length MAX_PATH+! containing the path of the
 *         file in which to record any errors reported by the W/O path.  An empty
 *         log file path indicates that no log file is desired.  Note that this
 *         field is ignored if ignore_wo_errs is false, as in that case the
 *         usual error reporting mechanism is use.
 *
 * ignore_wo_errs: Boolean flag.  If it is set to TRUE, errors on the W/O
 *         channel are ignored (and logged if the log file is defined().
 *
 *******************************************************************************/

#define H5F__CURR_SPLITTER_VFD_CONFIG_VERSION 1
```

[2] For example, in later versions we may find it useful to run the R/W and W/O channels on separate threads to minimize I/O delay.

```
typedef struct H5F_splitter_vfd_config_t  {

    int32 version;
    hid_t rw_fapl_id;
    hid_t wo_fapl_id;
    char wo_path[MAX_PATH + 1];
    char log_file_path[MAX+PATH + 1];
    hbool_t ignore_wo_errs;


} H5F_splitter_vfd_config_t;
```

This definition in hand, the signatures of the new FAPL management routines may be give below:

```
herr_t
H5Pset_fapl_splitter(hid_t plist_id, H5F_splitter_vfd_config_t *config_ptr);

herr_t
H5Pget_fapl_splitter(hid_t plist_id, H5F_splitter_vfd_config_t *config_ptr);
```

4 Implementation Details

While section 2 discussed the cycle of operation of the splitter VFD in some detail, it did so on a largely conceptual level. This section records details of the actual implementation that are difficult or time consuming to extract from the source.

Note that the objective for the initial implementation is simply to develop a working prototype for inclusion in the rsync / Mirror VFD prototype. As such, we will not be overly concerned optimizing performance at this time.

Remainder of this section TBD.

5 Testing

Initial tests can be performed by configuring the splitter VFD to use the sec2 driver on both the R/W and R/O channels, writing a file, and comparing the resulting files, which should be identical.

While this should make a useful smoke check, more complete unit tests are desirable – update the RFC to suggest a suitable selection.

6 Recommendation

Implement a simple version of the splitter VFD to support a prototype of the mirror VFD.

Acknowledgements

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Contract Number DE-AC02-05CH11231.

Revision History

AUG 18, 2018:	Version 1 circulated for comment.