RFC: Actual I/O Mode

Jacob Gruber

Christian Chilan

Jonathan Kim

Allow a user to determine which type of I/O was performed after the completion of a requested parallel I/O call. This is not necessarily the same as what was requested.

# Introduction

Collective I/O, which is requested by the user via a data transfer property list (DXPL), can perform I/O according to several optimization schemes. The HDF5 library either chooses one based on a user-adjustable parameter, or a user can request an optimization directly.

These optimization schemes may not perform pure collective I/O. Some schemes analyze each chunk in a dataset individually, and may access some chunks collectively and others individually. Thus some independent I/O may still occur even when a collective operation is requested.

Currently, there is no way to check whether collective or independent I/O was actually performed during a dataset access operation. This RFC proposes extensions to the HDF5 library that allow the user to determine the optimization and I/O mode(s) used by each process in an I/O operation, although not at the level of individual chunks. The extensions will also allow the user to determine what caused the HDF5 library to break collective I/O for the local process and among all processes, if that was the case.

# Description

## Description of Optimizations

As this section of HDF5 is being reworked, some of this discussion may be obsolete. However, while details may change, the general thrust should remain intact.

### General Parallel I/O Concerns

Before we discuss specific optimizations, we should note that in certain circumstances, collective I/O will not be attempted at all, even if requested, and HDF5 will perform independent I/O collectively instead. The following conditions[[1]](#footnote-2) bring about this switch:

* Datatype conversions need to be performed
* Data transforms need to be performed
* The file is begin accessed with the MPI-POSIX driver
* One of the dataspaces is neither simple nor scalar
* There are point selections in one of the dataspaces[[2]](#footnote-3)
* The dataset’s storage is neither contiguous nor chunked
* Any filters need to be applied (in the case of chunked dataset storage)

If all of these checks pass, HDF5 chooses a collective I/O optimization scheme. If the dataset’s storage is contiguous, collective I/O proceeds without further consideration and will never switch to independent I/O. However if the dataset’s storage is chunked, a user can set an optimization scheme for choosing collective or independent access on the chunks via the H5Pset\_dxpl\_mpio\_chunk\_opt API call. Refer to the flowchart “Optimizations for Chunk Collective I/O” at the end of this document for the details of this decision process. Also refer to H5Pset\_dxpl\_mpio\_chunk\_opt entry in HDF5 reference manual[[3]](#footnote-4).

## Design of Properties

To track the type of I/O performed, two properties are proposed: actual\_chunk\_opt\_mode, to track the optimization scheme chosen for chunked datasets and actual\_io\_mode, to track whether independent I/O, collective I/O or some mix of both took place during the operation.

Two properties are proposed instead of one composite property because, even though most optimization schemes are limited in what type of I/O they can perform, almost all optimizations have multiple values for the actual I/O mode and most of these modes are shared among several optimizations.

The two properties are described in more detail in the following Reference Manual entries.

# New API Functions RM Entries

## H5Pget\_mpio\_actual\_chunk\_opt\_mode

Signature:

herr\_t H5Pget\_mpio\_actual\_chunk\_opt\_mode(hid\_t dxpl\_id, H5D\_mpio\_actual\_chunk\_opt\_mode\_t \* actual\_chunk\_opt\_mode)

Purpose:

Retrieves the type of chunk optimization that HDF5 actually performed on the last parallel I/O call.

Description:  
H5Pget\_mpio\_actual\_chunk\_opt\_mode retrieves the type of chunk optimization performed when collective I/O was requested. This property is set by H5Pset\_dxpl\_mpio\_chunk\_opt before I/O takes place, and will be set even if I/O fails.

Valid values returned in actual\_chunk\_opt\_mode:

H5D\_MPIO\_NO\_CHUNK\_OPTIMIZATION  
No chunk optimization was performed. Either no collective I/O was attempted or the dataset wasn't chunked. *(Default)*

H5D\_MPIO\_LINK\_CHUNK   
Collective I/O is performed on all chunks together.

Corresponds to the H5FD\_MPIO\_CHUNK\_ONE\_IO mode for H5Pset\_dxpl\_mpio\_chunk\_opt.

H5D\_MPIO\_COLL\_CHUNK\_ATONCE

Each chunk is individually marked with collective or individual based on how many processes are assigned to that chunk. If the fraction is greater than the chunk-ratio threshold, the chunk is marked as collective and collective I/O is performed all at once for all the collective marked chunks. The chunk-ratio threshold can be set using H5Pset\_dxpl\_mpio\_chunk\_opt\_ratio. The default value is 60%.

Corresponds to the H5FD\_MPIO\_COLL\_CHUNK\_ATONCE\_IO mode for H5Pset\_dxpl\_mpio\_chunk\_opt.

H5D\_MPIO\_MULTI\_CHUNK  
Same as the H5D\_MPIO\_COLL\_CHUNK\_ATONCE case, except that collective I/O is performed per chunk which is marked as collective instead of all at once for all the collective chunks.

Corresponds to the H5FD\_MPIO\_CHUNK\_MULTI\_IO mode for H5Pset\_dxpl\_mpio\_chunk\_opt.

H5D\_MPIO\_ALL\_CHUNK\_IND   
Independent I/O is performed on all chunks.

Corresponds to the H5FD\_MPIO\_ALL\_CHUNK\_IND\_IO mode for H5Pset\_dxpl\_mpio\_chunk\_opt.

Parameters:

hid\_t dxpl\_id   
IN: Dataset transfer property list identifier

H5D\_mpio\_actual\_chunk\_opt\_mode\_t \*actual\_chunk\_opt\_mode   
OUT: The type of chunk optimization performed by HDF5.

Returns:

Returns a non-negative value if successful; otherwise returns a negative value.

## H5Pget\_mpio\_actual\_io\_mode

Signature:

herr\_t H5Pget\_mpio\_actual\_io\_mode(hid\_t dxpl\_id,   
H5D\_mpio\_actual\_io\_mode\_t \* actual\_io\_mode)

Purpose:   
Retrieves the type of I/O that HDF5 actually performed on the last parallel I/O call. This is not necessarily the type of I/O requested.

Motivation:   
A user can request collective I/O via a data transfer property list (DXPL) that has been suitably modified with H5Pset\_dxpl\_mpio. However, HDF5 may bypass this request and perform independent I/O instead, if certain conditions[[4]](#footnote-5) are encountered. This property allows the user to see what kind of parallel I/O HDF5 actually performed. Used in conjunction with H5Pget\_mpio\_actual\_chunk\_opt\_mode, this property allows the user to determine exactly HDF5 did when attempting collective I/O.

Description:   
H5Pget\_mpio\_actual\_io\_mode retrieves the type of I/O performed on the selection of the current process. This property is set after all I/O is completed; if I/O fails, it will not be set.

Valid values returned in actual\_io\_mode:

H5D\_MPIO\_NO\_COLLECTIVE\_IO  
No collective I/O was performed. Collective I/O was not requested or collective I/O isn't possible on this dataset. (Default)

H5D\_MPIO\_CHUNK\_INDEPENDENT  
HDF5 performed one of the collective chunk optimization schemes and each chunk was accessed independently.

H5D\_MPIO\_CHUNK\_COLLECTIVE  
HDF5 performed one of the collective chunk optimization schemes and all chunks were accessed collectively.

H5D\_MPIO\_CHUNK\_MIXED  
HDF5 performed one of the collective chunk optimization schemes and some chunks were accessed independently, some collectively.

H5D\_MPIO\_CONTIGUOUS\_COLLECTIVE   
Collective I/O was performed on a contiguous dataset.

**Note:**

All processes need not return the same value. For example, if I/O is being performed using the multi chunk optimization scheme, one process's selection may include only chunks accessed collectively, while another may include only chunks accessed independently and a third may involve both types. In this case, the first process will report H5D\_MPIO\_CHUNK\_COLLECTIVE while the second will report H5D\_MPIO\_CHUNK\_INDEPENDENT and the third H5D\_MPIO\_CHUNK\_MIXED.

Parameters:

hid\_t dxpl\_id  
IN: Dataset transfer property list identifier

H5D\_mpio\_actual\_io\_mode\_t \* actual\_io\_mode  
OUT: The type of I/O performed by this process.

Returns:   
Returns a non-negative value if successful; otherwise returns a negative value.

## H5Pget\_mpio\_no\_collective\_cause

Signature:

herr\_t H5Pget\_mpio\_no\_collective\_cause(hid\_t dxpl\_id,   
uint32\_t \* local\_no\_collective\_cause,   
uint32\_t \* global\_no\_collective\_cause)

Purpose:   
Retrieves local and global causes that broke collective I/O on the last parallel I/O call.

Motivation:   
A user can request collective I/O via a data transfer property list (DXPL) that has been suitably modified with H5Pset\_dxpl\_mpio. However, there are conditions that can cause HDF5 to forgo collective I/O and perform independent I/O. Such causes can be different across the processes of a parallel application. This function allows the user to determine what caused the HDF5 library to skip collective I/O locally, in the local process, and globally, across all processes.

Description:   
H5Pget\_mpio\_no\_collective\_cause serves two purposes. It can be used to determine whether collective I/O was used for the last preceding parallel I/O call. If collective I/O was not used, it retrieves the causes that broke collective I/O on that parallel I/O call. The properties retrieved by this function are set before I/O takes place and are retained even when I/O fails.

Valid values returned on the property are as follows; the numbers on the right are bitmask values:

H5D\_MPIO\_COLLECTIVE = 00000000  
Collective I/O was performed successfully. (Default)

H5D\_MPIO\_SET\_INDEPENDENT = 00000001  
Collective I/O was not performed because independent I/O was requested.

H5D\_MPIO\_DATATYPE\_CONVERSION = 00000010  
Collective I/O was not performed because datatype conversions were required.

H5D\_MPIO\_DATA\_TRANSFORMS = 00000100  
Collective I/O was not performed because data transforms needed to be applied.

H5D\_MPIO\_SET\_MPIPOSIX = 00001000  
Collective I/O was not performed because the selected file driver was MPI-POSIX.

H5D\_MPIO\_NOT\_SIMPLE\_OR\_SCALAR\_DATASPACES = 00010000  
Collective I/O was not performed because one of the dataspaces was neither simple nor scalar.

H5D\_MPIO\_POINT\_SELECTIONS = 00100000  
Collective I/O was not performed because there were point selections in one of the dataspaces.

H5D\_MPIO\_NOT\_CONTIGUOUS\_OR\_CHUNKED\_DATASET = 01000000  
Collective I/O was not performed because the dataset was neither contiguous nor chunked.

H5D\_MPIO\_FILTERS = 10000000  
Collective I/O was not performed because filters needed to be applied.

The above name/value pairs are members of the H5D\_mpio\_no\_collective\_cause\_t enumeration.

Each process determines whether it can perform collective I/O and broadcasts the result. Those results are combined to make a collective decision; collective I/O will be performed only if all processes can perform collective I/O.

If collective I/O was not used, the causes that prevented it are reported by individual process by means of an enumerated set. The causes may differ among processes, so H5Pget\_mpio\_no\_collective\_cause returns two property values. The first value is the one produced by the local process to report local causes. This local information is encoded in an enumeration, the H5D\_mpio\_no\_collective\_cause\_t described above, with all individual causes combined into a single value by means of a bitwise OR operation. The second value reports global causes; this global value is the result of a bitwise-OR operation across the values from all the processes.

Parameters:

hid\_t dxpl\_id  
IN: Dataset transfer property list identifier

uint32\_t \* local\_no\_collective\_cause  
OUT: A enumerated set value indicating the causes that prevented collective I/O in the local process.

uint32\_t \* global\_no\_collective\_cause  
OUT: An enumerated set value indicating the causes across all processes that prevented collective I/O.

Returns:   
Returns a non-negative value if successful; otherwise returns a negative value.

# Notes

In a collective operation, the values available to actual\_io\_mode are dependent on the value of actual\_chunk\_opt\_mode.

The actual\_chunk\_opt\_mode and actual\_io\_mode properties are not strictly paired nor all combinations of the properties are possible.

The possible combinations between the two APIs are:

|  |  |  |
| --- | --- | --- |
|  | **actual\_chunk\_opt\_mode** | **actual\_io\_mode** |
|  | H5D\_MPIO\_NO\_CHUNK\_OPTIMIZATION | H5D\_MPIO\_NO\_COLLECTIVE H5D\_MPIO\_CONTIGUOUS\_COLLECTIVE |
|  | H5D\_MPIO\_LINK\_CHUNK | H5D\_MPIO\_CHUNK\_COLLECTIVE |
|  | H5D\_MPIO\_COLL\_CHUNK\_ATONCE | H5D\_MPIO\_NO\_COLLECTIVE H5D\_MPIO\_CHUNK\_INDEPENDENT H5D\_MPIO\_CHUNK\_COLLECTIVE H5D\_MPIO\_CHUNK\_MIXED |
|  | H5D\_MPIO\_MULTI\_CHUNK | H5D\_MPIO\_NO\_COLLECTIVE H5D\_MPIO\_CHUNK\_INDEPENDENT H5D\_MPIO\_CHUNK\_COLLECTIVE H5D\_MPIO\_CHUNK\_MIXED |
|  | H5D\_MPIO\_ALL\_CHUNK\_IND | H5D\_MPIO\_CHUNK\_INDEPENDENT |

Also, at the present time, there is no way of telling whether a specific chunk was read collectively or independently.

# Usage

If a user is experiencing difficulties with parallel I/O, support personnel could use these properties to get extra diagnostic information. Additionally, a user could use these functions to ensure that a specific optimization is chosen to prevent unexpected slowdown of parallel applications.

# Example

The following pseudo code illustrates the use of the actual I/O mode properties in determining whether a process performed collective I/O, independent I/O or both in an application with three processes. In this example Process 0 will report collective I/O, Process 1 will report both collective and independent I/O and Process 2 will report independent I/O. This example is contrived, but it isn’t too hard to imagine that if the processes’ selections were determined by a computation or user input, a similar scenario might arise.

H5D\_mpio\_actual\_chunk\_opt\_mode\_t actual\_chunk\_opt\_mode;

H5D\_mpio\_actual\_io\_mode\_t actual\_io\_mode;

<set up mpi\_rank and mpi\_size>

<open file collectively>

<create space>

<create dataset with three chunks>

<create file and memory spaces>

if (mpi\_rank == 0) {

<select hyperslab in Chunk 0>

} else if (mpi\_rank == 1) {

<select hyperlab in Chunk 0 and Chunk 1>

} else if (mpi\_rank == 2) {

<select hyperslab in Chunk 2>

}

dxpl = H5Pcreate(H5P\_DATASET\_XFER);

H5Pset\_dxpl\_mpio(dxpl, H5FD\_MPIO\_COLLECTIVE);

/\* Set chunk optimization mode that can utilize ratio threshold \*/

H5Pset\_dxpl\_mpio\_chunk\_opt(dxpl,H5FD\_MPIO\_COLL\_CHUNK\_ATONCE\_IO);

/\* Set the threshold fraction of processes per chunk for  
 \* collective I/O. Here, collective I/O will only occur   
 \* if a process is selected by at least 40% of processes.  
 \*/

H5Pset\_dxpl\_mpio\_chunk\_opt\_ratio(dxpl, 40);

H5Dwrite(dataset, data\_type, mem\_space, file\_space, dxpl, buffer);

H5Pget\_mpio\_actual\_io\_mode(dxpl, &actual\_io\_mode);

H5Pget\_mpio\_actual\_chunk\_opt\_mode(dxpl, &actual\_chunk\_opt\_mode);

/\* Check properties against expected values \*/

assert(actual\_chunk\_opt\_mode == H5D\_MPIO\_MULTI\_CHUNK);

if (mpi\_rank == 0) {

assert(actual\_io\_mode == H5D\_MPIO\_CHUNK\_COLLECTIVE);

} else if (mpi\_rank == 1) {

assert(actual\_io\_mode == H5D\_MPIO\_CHUNK\_MIXED);

} else if (mpi\_rank == 2) {

assert(actual\_io\_mode == H5D\_MPIO\_CHUNK\_INDEPENDENT);

}

The next example illustrates the use of the no-collective-cause property in determining why collective I/O was interrupted. In this case, a file is opened using the MPI-POSIX driver and a collective write operation is requested. The returned property value indicates that collective I/O could not be performed because of the MPI-POSIX driver is in use.

H5D\_mpi\_no\_collective\_cause\_t local\_no\_collective\_cause;

H5D\_mpi\_no\_collective\_cause\_t global\_no\_collective\_cause;

<set up mpi\_rank and mpi\_size>

fapl = H5Pcreate(H5P\_FILE\_ACCESS);

H5Pset\_fapl\_mpiposix(fapl, MPI\_COMM\_WORLD, 0);

<open file collectively>

<create space>

<create contiguous dataset>

<create file and memory spaces>

<hyperslab selection divides dataset equally among processes>

dxpl = H5Pcreate(H5P\_DATASET\_XFER);

H5Pset\_dxpl\_mpio(dxpl, H5FD\_MPIO\_COLLECTIVE);

H5Dwrite(dataset, data\_type, mem\_space, file\_space, dxpl, buffer);

H5Pget\_mpi\_no\_collective\_cause(dxpl, &local\_no\_collective\_cause, &global\_no\_collective\_cause);

/\* check property against expected value \*/

assert(local\_no\_collective\_cause == H5D\_MPIO\_SET\_MPIPOSIX);

assert(global\_no\_collective\_cause == H5D\_MPIO\_SET\_MPIPOSIX);

# Recommendation

The HDF5 API extensions proposed in this RFC have been implemented, but the parallel I/O code is changing. Thus the details of this RFC and the associated code will probably need to be revisited.

# Optimizations and I/O operations Flowchart



Brief descriptions of the optimization modes for H5Pset\_dxpl\_mpio\_chunk\_opt follow:

|  |  |
| --- | --- |
| Optimization modes | Description |
| H5FD\_MPIO\_CHUNK\_ONE\_IO | Do collective I/O all at once for all the selected chunks.  This mode will not switch to independent I/O. |
| H5FD\_MPIO\_COLL\_CHUNK\_ATONCE\_IO | Do collective I/O all at once for all the selected chunks that marked as collective. Do individual I/O for the rest chunks.  Thus, this mode will switch between collective and independent I/O. |
| H5FD\_MPIO\_CHUNK\_MULTI\_IO | Do collective I/O per chunk for the selected chunks that marked as collective. Do individual I/O for the rest chunks.  Thus, this mode will switch between collective and independent I/O. |
| H5FD\_MPIO\_ALL\_CHUNK\_IND\_IO | Do independent I/O for all the selected chunks.  This mode will not switch to collective I/O. |

# Flowchart to determine whether collective I/O can be performed or not

**CollectiveIO6.pdf**

# **RFC Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| August 04, 2011 | | Version 1 posted for public comment. Comments should be sent to gruber1@hdfgroup.org | |
| August 22, 2011 | | Minor tweaks after comments from Quincey. | |
| September 6, 2012 | | Minor update for H5Pget\_mpio\_no\_collective\_cause section. (Property name changes, local cause change.) | |
| November 6, 2012 | | Update according to the removing of the broken ‘multi-chunk IO without opt’ feature. | |
| January 9, 2013 | | Update for refracting framework and add an improved optimization mode ‘H5FD\_MPIO\_COLL\_CHUNK\_ATONCE\_IO’ based on the ‘H5FD\_MPIO\_CHUNK\_MULTI\_IO’ mode.  Also added ‘H5FD\_MPIO\_ALL\_CHUNK\_IND\_IO’ mode as opposite of ‘H5FD\_MPIO\_CHUNK\_ONE\_IO’.  The update is from HDFFV-8244 task. | |
| February 12,2013 | | Some updates after comments from Quincey. (HDFFV-8244) | |

1. Some of these conditions are pretty opaque to me, and my descriptions are little more than educated guesses. The decision process is illustrated in a flowchart in Section 9. If you need more detail, look at H5Dmpio.c, specifically in H5D\_mpio\_opt\_possible. [↑](#footnote-ref-2)
2. Allowing collective I/O on point selections is actively being worked on and should be supported soon. [↑](#footnote-ref-3)
3. <http://www.hdfgroup.org/HDF5/doc/RM/RM_H5P.html#Property-SetDxplMpioChunkOpt> [↑](#footnote-ref-4)
4. These conditions can be referred from section 9 or from the list under “General Parallel I/O Concerns” in section2. If you need more detail, look at H5Dmpio.c, specifically in H5D\_mpio\_opt\_possible. [↑](#footnote-ref-5)