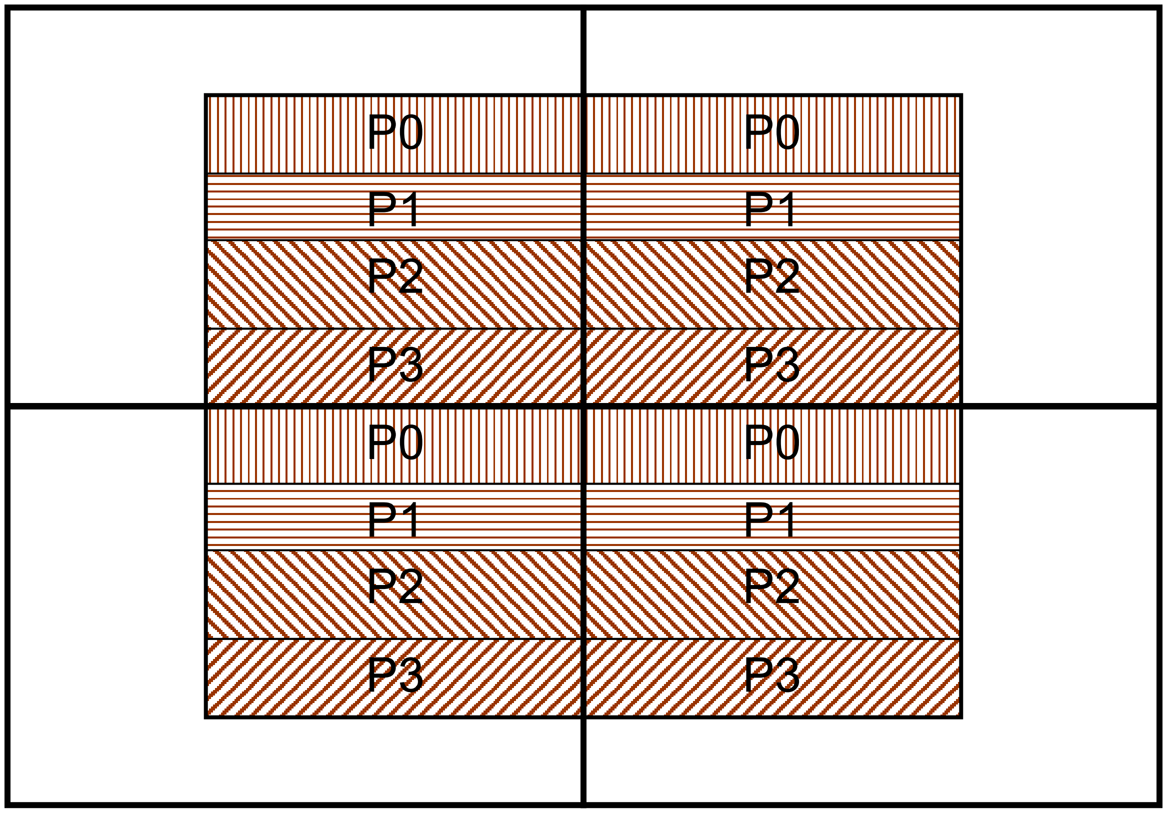
New APIs to enhance the collective IO support in chunking storage

Kent Yang
2006-1-9

I. Background and Purposes

One important aspect of using chunking storage inside HDF5 is to improve performance. On the other hand, collective IO is the key feature provided by MPI-IO to help improve performance. This can be illustrated with the example 1.

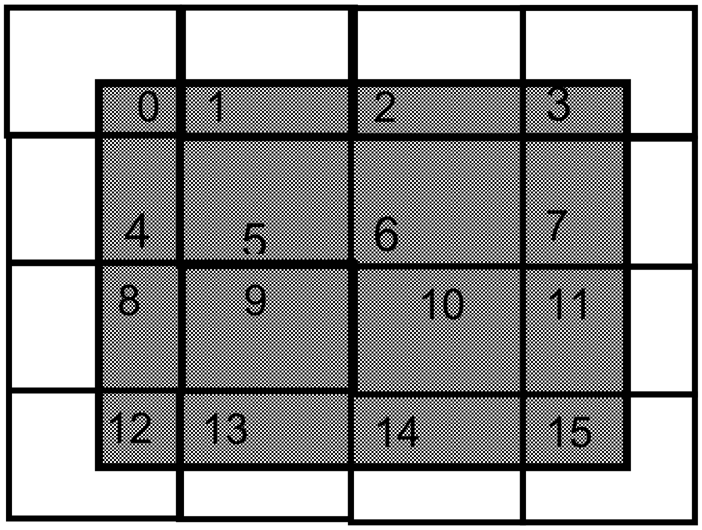
Example 1: multi-chunk collective IOs with selections from all processes in each chunk, totally there are four chunks in this case



The collective IO will outperform independent IO significantly. Four independent IO accesses are done compared with one collective IO access inside one chunk.

However, when using collective IO with chunking storage without any tuning, performance may become worse under some circumstances. Example 2 and example 3 show this scenario.

Example 2: many small chunks with the selection in the center, totally there are 16 chunks

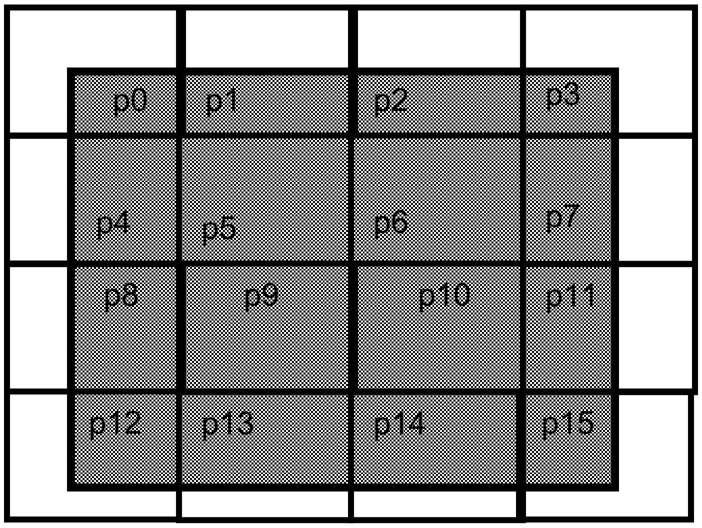


The number on the chart is the chunk number. We assume every process is doing IO at every chunk. Sixteen accesses to the disk are needed to do collective IO. The performance can be improved with the following tuning inside HDF5:

1. Create an MPI derived datatype that links all chunks
2. Do collective IO with the created MPI derived datatype

Only one access to the disk is needed for each process with this tuning. We will refer this IO pattern to be "linked-chunk IO" hereafter. Unfortunately this kind of optimization may not work well for all cases. For example, if the size of MPI derived data type is too large, some MPI-IO packages may fail based on our experiences with various MPI-IO packages. Under this case, HDF5 has to do IO per chunk. So we propose to provide an API for users to participate in the decision-making process in choosing the usage of one linked-chunk IO. We will describe more detailed in section II.

Example 3: multi-chunk collective IOs with the selection from only one process in each chunk



'pn' represents <process n> in the above chart. With the use of collective IO, only one process does IO for each chunk; other processes are required to participate without providing any contributions. Sixteen IO accesses are needed with the overhead caused by communications among processes to participate collectively. With the use of independent IO, each process does IO individually. 16 IO accesses are also needed, but without communications involved. The performance in collective IO will be worse than the performance in independent IO for this case.

Although doing collective IO generates overheads due to communications among processors, it is still possible that the performance will be greatly improved as shown in example 1. Four independent IO accesses are done compared with one collective IO access inside one chunk.

The real application may be between example 1 and example 3. So inside HDF5 library, we would like to provide some criteria to optimize the way to do collective IO. One simple approach is to provide a threshold of something the library can use to compare with. One obvious parameter is to use the number of processes participating IOs in this chunk to check whether it is worth doing collective IOs or simply doing independent IOs. With the consideration of differences among applications, parallel platforms and MPI packages, the threshold may be different to obtain the optimal performance or to avoid unexpected bad performance. Again, we would like to give the application a choice to participate the decision-making process. That will also be addressed in section II.

II. Proposed APIs

Based on the above descriptions, we would like to provide two parameters for applications to choose the way to do collective chunk IO. These two parameters are **NOT REQUIRED** to be set by applications. HDF5 will set default values instead.

In common sense, the one linked-chunk IO should outperform multi-chunk IOs. However, due to immaturity of MPI-IO packages and parallel file systems, it is better to give the user an option in case unexpected output occurs. Parameter 1 should describe a threshold to use one-linked chunk IO. The threshold we propose is a certain value of the average number of chunks each process holds. If the average number of chunk that covers the dataspace is greater than the threshold, the HDF5 library will try to do one-linked chunk IO. In general, the bigger the dataset size is, the safer it is to provide a higher number of chunks. If the threshold is 0, the library will try to use one-linked chunk IO. The threshold also depends on the system hardware, the parallel file system, and the robustness of MPI-IO packages and the IO pattern of the application.

If the application has to do IO for each chunk, parameter 2 determines whether to do collective IO for this chunk.
The threshold is the certain value of the percentage of number of processes participating IOs in this chunk. Only if the percentage of number of processes participating IOs in this chunk is greater than the threshold, collective IO is done. If the percentage is 0, the library will always do collective IO. If the percentage is 100, the library will only do collective IO if all processes participate for the current chunk.

1. Adding two new APIs

1) One-linked chunk IO or multiple chunk IOs

Name: H5Pset_dxpl_mpio_chunk_opt_num

Signature:
```
herr_t     H5Pset_dxpl_mpio_chunk_opt_num     (hid_t     dxpl_id,     unsigned
num_chunk_per_proc)
```

Purpose:
To set a threshold for doing one-linked chunk IO

Description:
The library will calculate the average number of chunk selected by each process. If the number is greater than the threshold set by the user, the library will do one-linked chunk IO; otherwise, IO will be done for every chunk.

Parameters:

hid_t dxpl_id	in: Data transfer property list identifier
unsigned num_proc_per_chunk	in: the threshold of the average number of chunks selected by each process

Returns:
Returns a non-negative value if successful. Otherwise returns a negative value.

2) For multiple collective chunk IO, collective or independent

Name: H5Pset_dxpl_mpio_chunk_opt_ratio

Signature:

```
herr_t H5Pset_dxpl_mpio_chunk_opt_ratio
       (hid_t dxpl_id, unsigned percent_num_proc_per_chunk)
```

Purpose:

To set a threshold for doing collective IO for each chunk

Description:

The library will calculate the percentage of the number of process holding selections at each chunk. If that percentage of number of process in the individual chunk is greater than the threshold set by the user, the library will do collective chunk IO for this chunk; otherwise, independent IO will be done for this chunk.

Parameters:

hid_t dxpl_id
in: Data transfer property list identifier

unsigned percent_num_proc_per_chunk
in: the threshold of the percentage of the number of process holding selections per chunk

Returns:

Returns a non-negative value if successful. Otherwise returns a negative value.

2. Adding one new API

Name: H5Pset_dxpl_mpio_chunk_opt

Signature:

```
herr_t H5Pset_dxpl_mpio_chunk_opt
(hid_t dxpl_id, unsigned num_chunk_per_proc, unsigned percent_num_proc_per_chunk)
```

Purpose:

(very much like the previous one, will fill in later if decided to be used)

Description:

(Will fill in later)

Returns:

Returns a non-negative value if successful. Otherwise returns a negative value.

3. Changing the current API

Change the original

```
herr_t H5Pset_dxpl_mpio (hid_t dxpl_id, H5FD_mpio_xfer_t xfer_mode)
```

to

```
herr_t  H5Pset_dxpl_mpio(hid_t  dxpl_id,  H5FD_mpio_xfer_t  xfer_mode,
unsigned num_proc_per_chunk, unsigned num_chunk_per_proc)
```

III. Backward and forward Compatibility

The added APIs are only used for performance tuning for collective chunk IO in chunking storage. These APIs are optional. The output from the application won't be affected without using the added APIs. It won't involve any backward and forward compatibility or file format issues.