PERFORMANCE COMPARISON OF COLLECTIVE I/O AND INDEPENDENT I/O WITH DERIVED DATATYPES.

Christian M. Chilan
Kent Yang
2006-6-23

In the current version of HDF5, I/O operations can be carried out in independent or collective mode. These modes correspond to the access types provided by MPI-IO.

In independent I/O access, the requests of each processor are handled individually. On the other hand, collective I/O access combines many requests in a single contiguous I/O operation minimizing the contribution of latency.

For non-interleaved patterns, collective I/O operations should not provide a significant improvement in performance with respect to independent operations. However, when non-interleaved selections contain holes, collective I/O access performs better than independent I/O access in the current version of HDF5. By providing information about the access pattern in the form of derived datatypes, collective access enable a MPI-IO optimization known as data sieving which consists of performing large I/O operations including holes in the selection, instead of many small I/O operations skipping the holes. It is clear that data sieving minimizes the contribution of latency improving the performance significantly.

Since collective I/O mode incurs in some overhead, we believe that independent I/O access can yield better performance provided that it also uses derived datatypes to enable data sieving in collectively way. In order to verify this hypothesis, we perform testing with non-interleaved selections in Bluesky, the NCAR IBM Power4 SP cluster, using two categories.

The first category models the common case in which all the processors of the communicator participate in the I/O operation. In the second category, we want to determine the effect in performance of using only a subset of processors out of 64 processors in the communicator.

Testing with full processor participation

In these tests, we compare the performance of collective I/O and independent I/O access with derived datatypes for the common case in which all the processors participate in the I/O operations. The configuration and geometry are shown in the following Table 1 and Figure 1. The type of each element is a char.

Test	Processors	Buffer_dim	Dset_dim
16p, 1Kx1K	16	1K	16K
16p, 2Kx2K	16	2K	32K
32p, 1Kx1K	32	1K	32K

Table 1 Test parameters

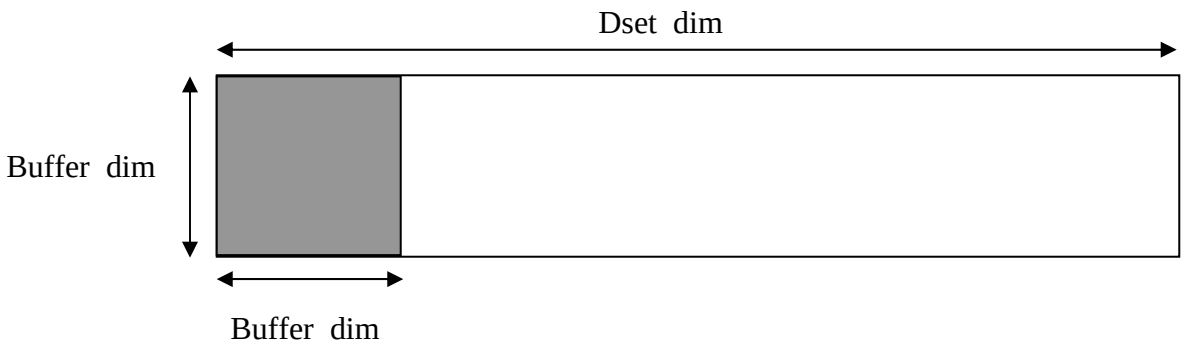

Figure1 Geometry of the selection per processor per I/O operation

The shaded area in Figure 1 represents the processor selection during the first I/O operation. During the subsequent operations, the shaded area shifts to right so that it covers the entire area of the dataset.

The results of our testing are shown in Figure 2. Note that benefit of using independent I/O access with derived datatypes is an improvement in performance in READ operations. However, we see that this advantage decreases as the holes become much larger than the processor selection per I/O operation.

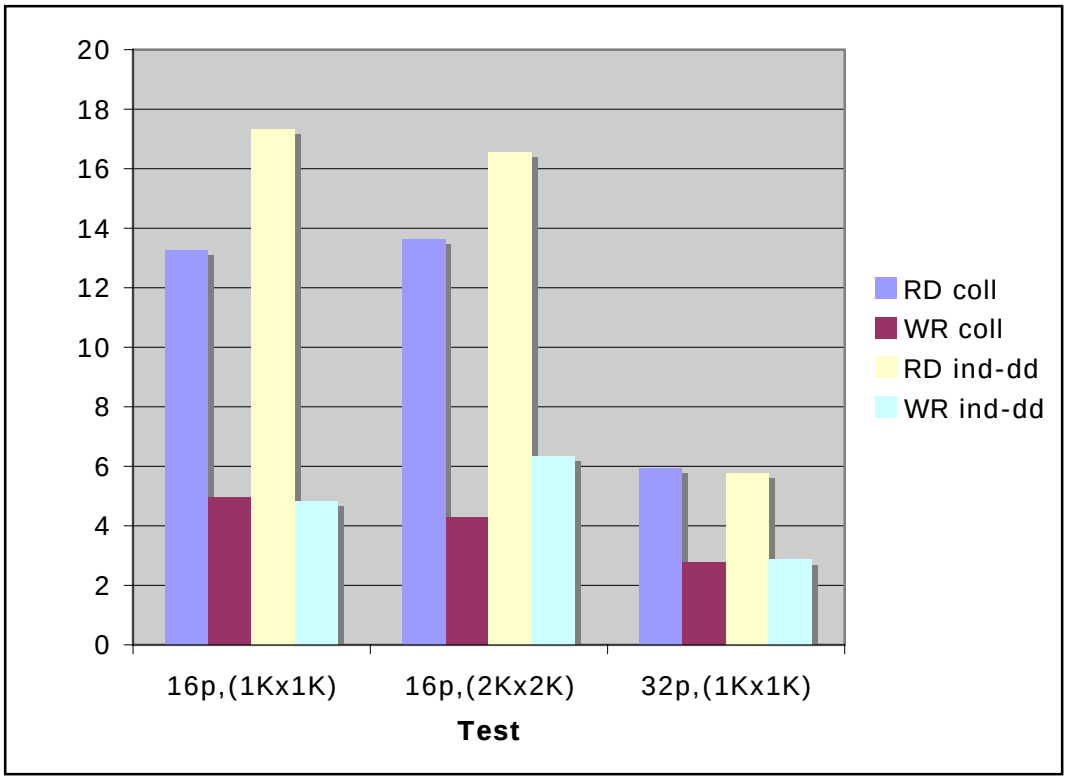

Figure 2 Performance of tests with full processor participation

Testing with subset of processors

In these tests, we wanted to determine the performance impact of using only a small subset of processors to execute I/O operations. The total number of processors is 64 but the actual number of processors that perform I/O is varied.

The geometry of the selection per processor is shown in Figure 3. The type of each element is an integer.

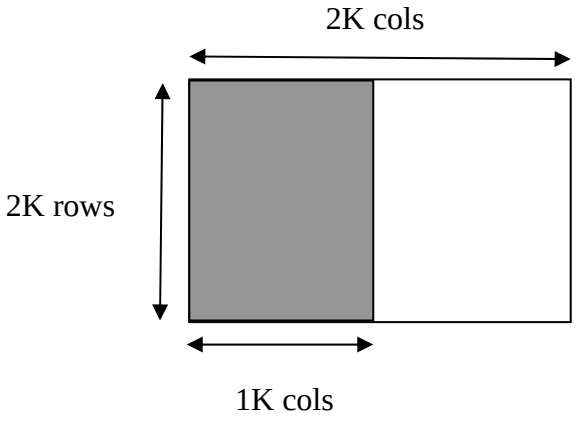

Figure 3 Geometry of the selection per processor

The results of our testing are shown in Figures 4 and 5. As we see, independent access with derived datatypes always provides better performance than collective access. This is more evident when the subset of processors is much smaller than the total number of processors in the communicator.

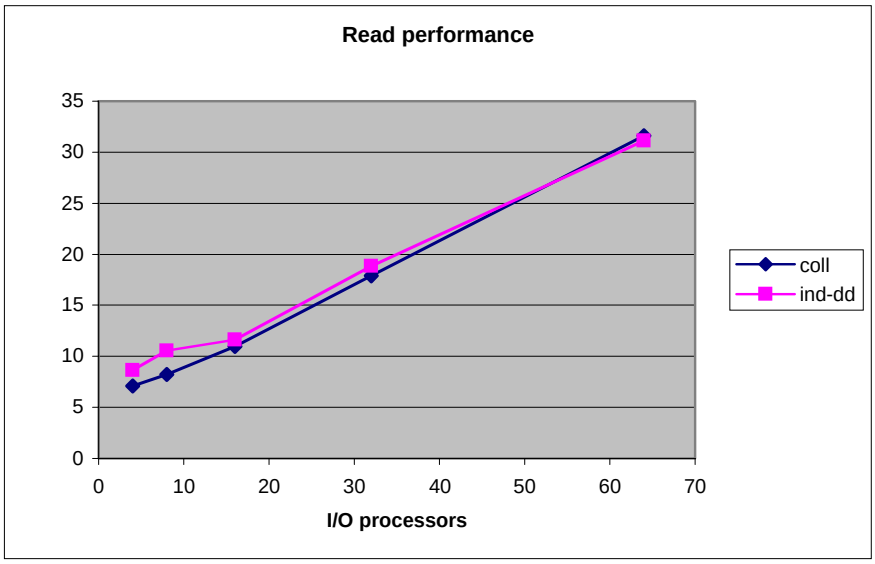

Figure 4 Read performance using a subset of processors for I/O

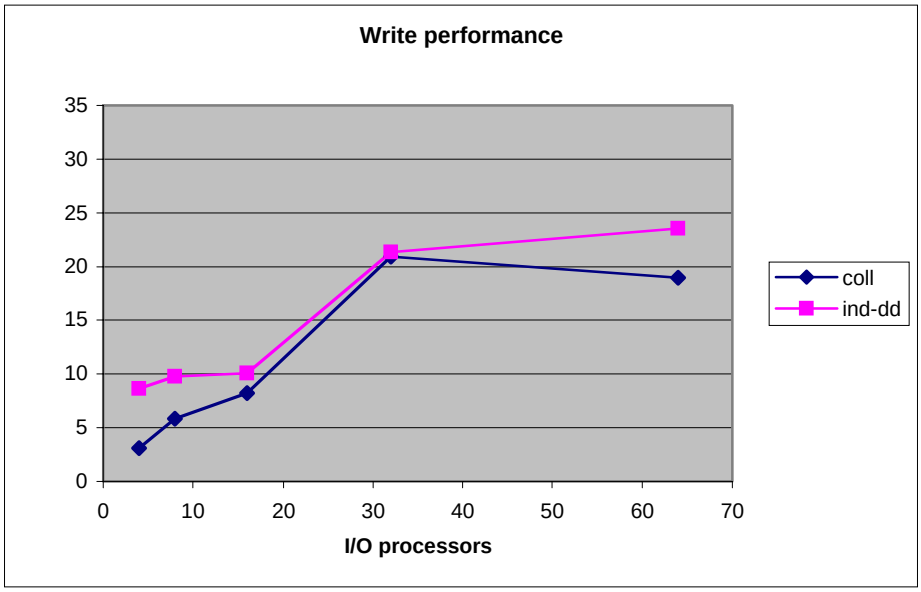

Figure 5 Write performance using a subset of processors for I/O

Conclusions

Since we did not find a case in which independent I/O access with derived datatypes reduces the performance significantly with respect to collective I/O operations, we believe that it is a valid option to include in HDF5. The condition to use independent I/O access with derived datatypes inside HDF5 is the same as the condition to use collective I/O inside HDF5.

The magnitude of the performance improvement of independent access with derived datatypes depends on the size of the selection and the holes per I/O operation.

Appendix:

The new API for doing independent IO with DDT

Name: H5Pset_dxpl_mpio_collective_opt

Signature:

```
herr_t H5Pset_dxpl_mpio_collective_opt
       (hid_t dxpl_id, H5FD_mpio_collective_opt_t opt_mode)
```

Purpose:

Applications that set data transfer property list to `H5FD_MPIO_COLLECTIVE` can set a flag in this API to use MPI-IO independent I/O functions inside HDF5. This API allows controlling the low-level type of I/O while maintaining the same collective interface at the application level.

Description:

This API is an optional API. It **should only be used** when **`H5FD_MPIO_COLLECTIVE`** is set through data transfer API `H5Pset_dxpl_mpio`. When the application sets the flag to `H5FD_MPIO_INDIVIDUAL_IO, the library will use low-level MPI independent I/O functions. Otherwise, collective I/O functions are used. The library will do collective I/O if this API is not called.`

Valid flags are as follows:

`H5FD_MPIO_COLLECTIVE_IO`
: Use collective I/O access(default)

`H5FD_MPIO_INDIVIDUAL_IO`
: Use independent I/O access

Parameters:

hid_t dxpl_id in: Data transfer property list identifier

H5FD_mpio_collective_opt_t opt_mode

in: The flag to determine the usage of collective I/O or independent I/O.

Returns:

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