RFC: Collective Reading and Writing of data via one process

M. Scot Breitenfeld
John Mainzer

This RFC proposes an additional transfer property option which, when set for H5Dread, would have one process read the data from disk, and then broadcast the data to the remaining processes in the MPI communicator. When this transfer property is set for H5Dwrite, only one process will write the data. This property should be used only when all the processes are meant to be reading or writing the same data.

1 Rational for the HDF5 parallel IO extension

In general, a typical use case for HDF applications is when users augment the raw data with file format specific metadata (FFSM) as either attributes or small datasets. One category of FFSM is that in which the data is independent of the number of processes reading or writing the data, i.e., the data itself is inherently the same for all the processes. For example, in the case of the CGNS library, it is the metadata information contained in the *base, zone* and *zone-type* descriptions. This means of organizing the data does not cause an issue when reading in serial. For the parallel IO case, however, having all the processes reading or writing all of the same small amounts of data is not scalable. This will be demonstrated in the next few sections. The solution proposed in this RFC is to read the FFSM using one process and then to do a broadcast of the FFSM to the remaining processes in the MPI communicator. For writing of FFSM, one process would write the data. The following investigations used *skybridge* at SNL and *edison* at LBNL.

1.1 Benchmark Description for investigation of all processes reading FFSM

The CGNS benchmark creates 128 bases with 32 zones in each base. The timing reported is during **cgp_open** of an already created CGNS file. The majority of the time in **cgp_open** is spent in **cgi_read**, and the remaining sections of the report breakdown the cascading calls in **cgi_read**. The number and size of the reads during **cgp_open** for this case is as follows:

SIZE OF ARRAY	TOTAL NUMBER OF READS	SIZE PER READ (BYTES)	TOTAL READ SIZE (BYTES)
2X1	128	8	1,024
3X3	4096	72	294,912
10X1	4096	10	40,960
1X1	1	4	4

Total read size is ~329K.

1.2 Benchmark Findings

Within cgp_open, the majority of the time is spent in ADFH_Read_All_Data:

```
if (H5Dread(did, mid, H5S_ALL, H5S_ALL, xfer_prp, data) < 0)

  set_error(ADFH_ERR_DREAD, err);

else

  set_error(NO_ERROR, err);
```

skybridge timing:

1 process	0.00361 s
2048 processes	5.57 s

Thus, for a few FFMS reads, the time would not usually be significate. However, for a large number of FFMS reads, the time can quickly accumulate and make the reading of the FFSM an impediment for large process counts.

1.3 MPI IO only benchmark Investigation

A benchmark was written to simulate the poor read scaling performance using only MPI IO. The benchmark writes an integer array of dimension 524,288 (2MB) and the file is closed. The file is opened, and *MPI_File_read_all* is called 8192 times by all the process reading the same data. A single read consists of a total of 64 bytes, and each read was offset from the previous read by 256 bytes.

The timing shows a similar behavior as to that shown with CGNS on skybridge:

1 process	0.15 s
2048 processes	470 s

Options for reading data:

(1) Read all the metadata on one process, then Bcast the metadata to the all the other processes.
(2) Read one metadata entry on one process, then Bcast that metadata to all the other processes. Repeat until all the metadata has been read (i.e., 8192 times for this case).

Option 0 (All processes read the metadata)	
1 process	0.15 s
2048 processes	474 s

Option 1	
1 process	0.15 s
2048 processes	0.18 s

Option 2	
1 process	0.17 s
2048 processes	0.27 s

Thus, packing all the FFMS into a single MPI_Bcast improves the time over doing an MPI_Bcast for each FFMS. Regardless, both options one and two are significantly faster than having all the processes read the same data.

1.4 CGNS benchmark on GPFS

The benchmark in the main section of the program was run on cetus (mira) at ANL:

The average time spent in ADFH_Read_All_Data in:

ADFH_Read_All_Data

```
  if (H5Dread(did, mid, H5S_ALL, H5S_ALL, xfer_prp, data) < 0)
    set_error(ADFH_ERR_DREAD, err);
  else
    set_error(NO_ERROR, err);
```

cetus (ANL) timing:

1 process	0.56 s
2048 processes**	25.1 s

** job was killed before all the H5Dreads had completed after 10 minutes.

1.5 CGNS benchmark with single process IO option

A new transfer property, H5FD_MPIO_SINGLE_PROC_IO, was added to the HDF5 library to enable reading the data on one process and then broadcasting the data to the other processes. CGNS was changed to use this option when reading data that is less than 2MB.

In **cgi_read,** at line:

```
 for (b=0; b<cg->nbases; b++) if (cgi_read_base(&cg->base[b])) return CG_ERROR;
```

skybridge (SNL) timing:

NUMBER OF PROCESSES	Time (seconds)
1	24.4
2048 (H5FD_MPIO_COLLECTIVE)	781
2048 (H5FD_MPIO_PROC0_BCAST)	53.5
4096 (H5FD_MPIO_PROC0_BCAST)	106.1
8192 (H5FD_MPIO_PROC0_BCAST)	238.3

Edison (LBNL) timing:

NUMBER OF PROCESSES	Time
1	4.9
3072 (H5FD_MPIO_COLLECTIVE)	199.3
3072 (H5FD_MPIO_PROC0_BCAST)	12.9
6144 (H5FD_MPIO_COLLECTIVE)	417.6
6144 (H5FD_MPIO_PROC0_BCAST)	16.6
6144 (H5FD_MPIO_PROC0_BCAST) **	18.7
12288 (H5FD_MPIO_PROC0_BCAST) **	19.1 , 26.6
24576 (H5FD_MPIO_PROC0_BCAST) **	31.1

** with an extra bcast (for read verification/error checking)

1.6 HDF5 small writes of the same values benchmark

1.6.1 HDF5 only, write test

A benchmark was written to compare **writing** a small amount of the same data to a file. An integer array of ten elements (i.e. 40 bytes) is written by having:

1. One process creates the file, writes the array, and closes the file.
2. All the processes write the same data to the same location in the file, this is done independently and collectively.

Edison (LBNL)

NUMBER OF PROCESSES	TIME (SECONDS)
1	0.052

3072 (INDEPENDENT)	98.5
3072 (COLLECTIVE)	0.647
6144 (INDEPENDENT)	171.2
6144 (COLLECTIVE)	1.54
12288 (INDEPENDENT)	** timed out after 10 minutes **
12288 (COLLECTIVE)	1.81

SkyBridge (SNL)

NUMBER OF PROCESSES	**TIME (SECONDS)**
1	0.012 - 0.020
2048 (INDEPENDENT)	1.5
2048 (COLLECTIVE)	0.17
4096 (INDEPENDENT)	4.7
4096 (COLLECTIVE)	0.49
8192 (INDEPENDENT)	10.1
8192 (COLLECTIVE)	2.4

1.6.2 CGNS test – Only process zero writes

The option of write only with process zero was add to both HDF5 and CGNS for writes of the same data that are less than 4MB.

Edison (LBNL)

NUMBER OF PROCESSES	**TIME (SECONDS)**
1536 (COLLECTIVE)	2.82
1536 (PROC 0)	2.73
3072 (COLLECTIVE)	3.71
3072 (PROC 0)	3.36

2 H5Dread/H5Dwrite parallel extension proposal

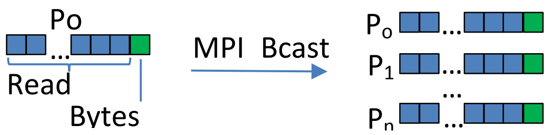
2.1 Overall Implementation

The lowest process in the MPI communicator reads the data and then broadcasts the data to the remaining processes in the MPI communicator. There are no checks:

1. That all the processors are reading the same data.
2. That the amount of data being read and broadcasted is of a reasonable size.

This implementation relies on the user, with no checking in the HDF5 library, to explicitly use this option only when all the processors are reading the same value. This option will always return a read value that is the same for all the processes, regardless of whether this was the actual read usage from the application. Thus, it is possible for a read to "succeed" but return the wrong data.

The main advantage of this option is it has the highest potential in performance since there is a minimum number of broadcasts: (1) the read data, and (2) a one element value being broadcasted. The later (2) is used to determine if the actual number of bytes requested is greater than the bytes read, and if so, the buffer gives zeroes beyond the end of the physical MPI file. This last broadcast (2) can be combined with the read data broadcast (1) to eliminate the need for the extra broadcast.



The write case does not involve an MPI_Bcast.

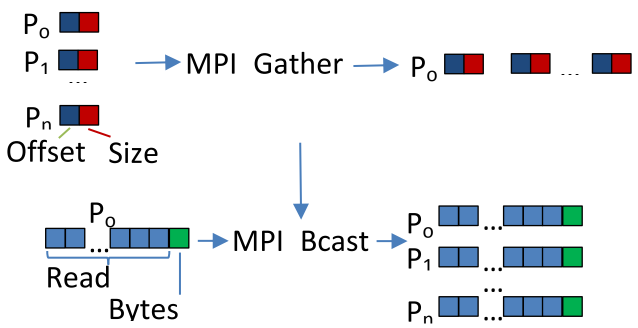
2.1.1 Detailed Implementation

A new transfer property, H5FD_MPIO_SINGLE_PROC_IO, will be added. The MPI_BCAST (i.e., "BCAST") makes all APIs using this transfer property collective. An alternative name, H5FD_MPIO_SINGLE_PROC_COLL_IO would reemphasize this fact. Another option would be to require H5FD_MPIO_COLLECTIVE and then set H5FD_MPIO_SINGLE_PROC_IO additionally. This ends the extent of the additions needed that would be visible to the end user.

--- LOW LEVEL HDF5 DETAILS ---

2.2 Safe Guard Implementation Extension

This extension adds additional checking to the implementation by checking that the buffer's offset and size are all the same on all the processors in the MPI communicator. A boolean property, H5_CHECK_SINGLE_PROC_IO, would allow the user to control whether or not this check occurs. The default would be that this check always happens. Overall, this check would add an MPI_Gather.



If the check is enabled, this option ensures that either the correct data is read or written by all ranks, or that the read or write fails.

--- LOW LEVEL HDF5 DETAILS ---

References

1. "HDF Group Reference Manual", https://support.hdfgroup.org/HDF5/doc/RM/RM_H5Front.html

Acknowledgements

This work is supported by the TriLabs project.

Revision History

Oct 8, 2018: Version 1 circulated for comment within The HDF Group.

Comments should be sent to brtnfld@hdfgroup.org or help@hdfgroup.org