RFC: Collective Metadata Writes

HDF5 uses a metadata cache internally for fast metadata access. Parallel HDF5 uses the metadata cache in the same way but with a requirement of having the same stream of dirty metadata from all MPI ranks. This means that all operations that modify metadata in the file are required to be collective. The HDF5 metadata cache is a very important component of the library that enables fast access to file metadata instead of issuing multiple small accesses to the file system. Parallel HDF5 uses the same cache on every MPI process with some restrictions. The most important one being that all modifications to the file metadata have to be done collectively on all ranks. For more information about the metadata cache and requirements, consult the metadata cache user guide. Writing metadata in HDF5 results from operations that modify the file structure, creating new objects, extending datasets, etc… Reading metadata results from operations such as opening the file and objects in that file, iterating through the file hierarchy, reading attributes, etc…

A flush of the metadata cache can be triggered either by the user through a call to H5Fflush() or H5Fclose(), or by the library when the amount of dirty metadata crosses a certain threshold. Either way, this operation is collective which means all processes will be participating. Originally, the library wrote out dirty metadata in one of two ways:

1. Process 0 write only strategy, where 1 process (rank 0) wrote the dirty metadata entries 1 entry at a time to the file and bcast the list to the other ranks to clean the flushed entries.
2. Distributed write strategy, where rank 0 sorts and assigns each process to equally write a certain number of entries, independently, 1 entry at a time.

Both algorithms were expensive on parallel file system. A new optimization will be introduced to modify the distributed algorithm with having all processes construct an MPI derived datatype for their assigned dirty entries, then all processes will issue 1 collective write call to write out the metadata to disk.

A new API routine is added to set a file access property that enables collective writing of metadata:

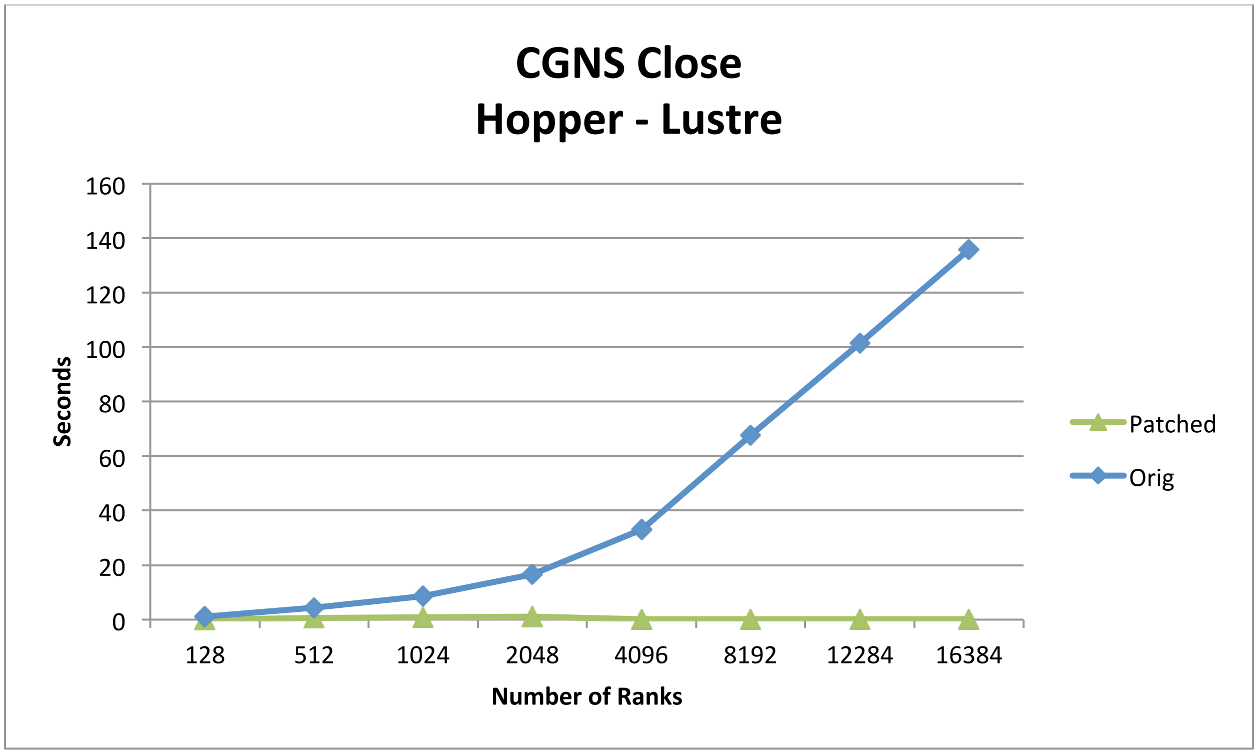
```
herr_t H5Pset_coll_metadata_write(hid_t plist_id,
hbool_t is_collective);
```

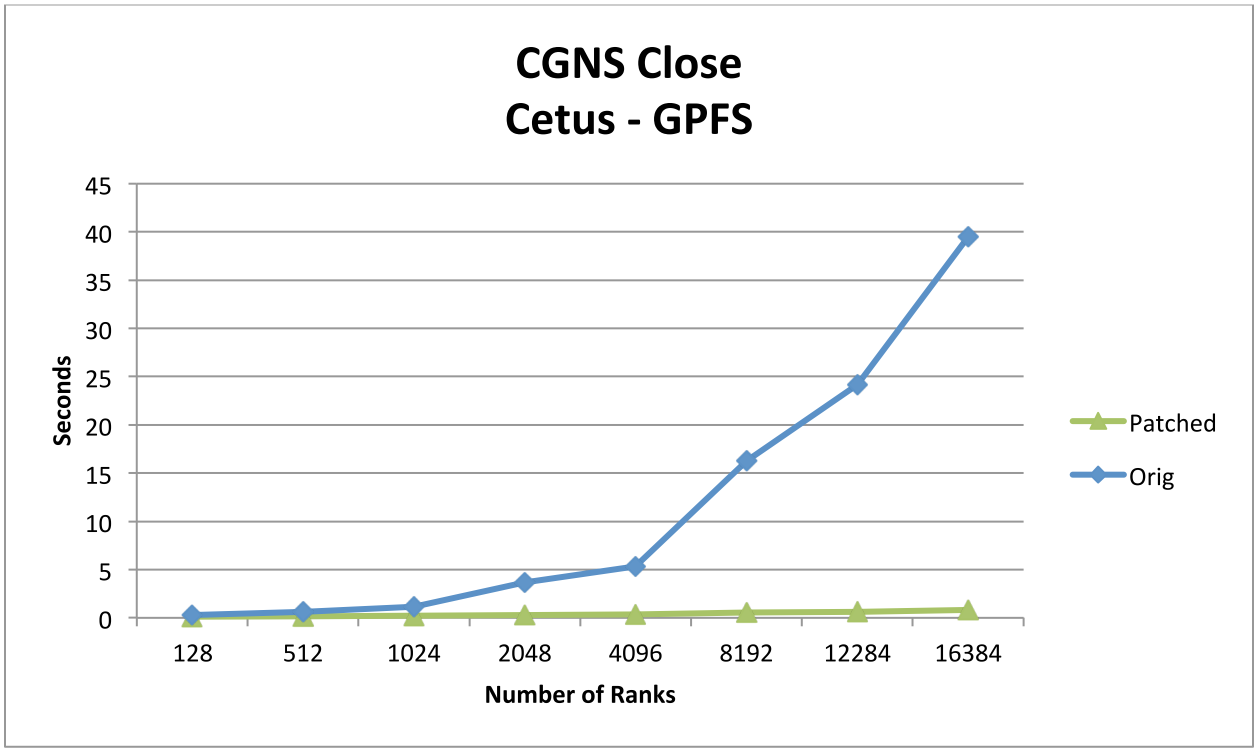
where `is_collective` indicates whether collective metadata writes will be enabled (true) or disabled (false). The default setting retains the old behavior which is false.

Another routines is added to retrieve the collective metadata write setting:

```
H5_DLL herr_t H5Pget_coll_metadata_write(hid_t
plist_id, hbool_t *is_collective);
```

Preliminary results benchmarking the CGNS close operation that writes out some HDF5 metadata it creates in the file show a huge speedup with the new optimization on LUSTRE and GPFS:




CGNS Close
Cetus - GPFS
Seconds
45
40
35
30
25
20
15
10
5
0
128
512
1024
2048
4096
8192
12284
16384
Number of Ranks
Patched
Orig