RFC: Setting Raw Data Chunk Cache Parameters in HDF5

Neil Fortner
Quincey Koziol

Introduction

Whenever a chunked dataset is read from or written to, the raw data chunk cache (rdcc) can temporarily cache the chunked data as it moves between application memory and the HDF5 file, and can greatly improve the performance of the read and write operations. Because different datasets can have different dimensions, chunk sizes, and access patterns, the optimum configuration for the raw data chunk cache may be different for each dataset. Currently each dataset has a separate rdcc but the rdcc settings are configured on a per file basis. This RFC proposes three new functions to allow individual rdcc configurations for each dataset.

1 Approach

The performance of the rdcc is affected by three parameters, which are currently set on a file access property list (fapl) by the function H5Pset_cache, and retrieved by H5Pget_cache. The fapl is used when opening or creating a file, and the properties in the property lists persist until the file is closed.

We propose three new functions, H5Pset_chunk_cache, H5Pget_chunk_cache, and H5Dget_access_plist to set the same rdcc properties on a dataset access property list (dapl). Dataset access property lists are used whenever a dataset is opened or created, and the dataset retains the property settings until the dataset is closed. The properties set by these new functions will, if implemented, be the first dataset access properties in the HDF5 library.

Under the proposed approach, when a dataset is created or opened, the library will first check whether the chunk caching parameters have been set on the dapl. If they have been set by H5Pset_chunk_cache, then those values will be used. If they have not been set, the values from the file—determined by the fapl used to create or open the file that contains the dataset—will be used. If the user does not want to set a certain parameter, or wants to unset it, a “use default” macro can be passed to H5Pset_chunk_cache for that parameter, as outlined below in section 2.1.

The function H5Pget_chunk_cache queries the raw data chunk cache parameters for a dapl. If a “use default” macro was specified for a parameter when the dapl was set, the value returned for that parameter by H5Pget_chunk_cache will be the default fapl value. Note that the default fapl value may be different from the fapl value for the parameter in a given file. The proposed behavior allows the H5Pget_chunk_cache function to be used to discover the default chunk caching parameters in a manner consistent with that of other properties.

2 Sample Reference Manual Entries

These sample reference manual entries are provided to more clearly illustrate the functionality of the proposed functions, and do not necessarily represent the final wording of the reference manual entries for these functions.

2.1 H5Pset_chunk_cache

Name: H5Pset_chunk_cache

Signature:

herr_t `H5Pset_chunk_cache`(*hid_t* `dapl_id`, *size_t* `rdcc_nslots`, *size_t* `rdcc_nbytes`, *double* `rdcc_w0`)

Purpose:

Sets the raw data chunk cache parameters.

Description:

`H5Pset_chunk_cache` sets the number of elements, the total number of bytes, and the preemption policy value in the raw data chunk cache on a dataset access property list. After calling this function, the values set in the property list will override the values in the file's file access property list (see `H5Pset_cache`).

The raw data chunk cache inserts chunks into the cache by first computing a hash value using the address of a chunk, then using that hash value as the chunk's index into the table of cached chunks. The size of this table (and the number of possible hash values) is determined by the *rdcc_nslots* parameter. If a different chunk in the cache has the same hash value, this causes a collision, which reduces efficiency. If inserting the chunk into cache would cause the cache to be too big, then the cache is pruned according to the *rdcc_w0* parameter.

The *dapl_id* is a dataset access property list.

The number of chunk slots in the raw data chunk cache for this dataset is *rdcc_nslots*. Increasing this value reduces the number of cache collisions, but slightly increases the memory used. Due to the hashing strategy, this value should ideally be a prime number, and not an integer factor of the chunk size. As a rule of thumb, this value should be at least 10 times the number of chunks that can fit in *rdcc_nbytes* bytes, and should ideally be at least 100 times that number of chunks. The default value is 521. If the value passed is H5D_CHUNK_CACHE_NSLOTS_DEFAULT, then the property will not be set on *dapl_id* and the parameter will come from the file.

The total size of the raw data chunk cache for this dataset is *rdcc_nbytes*. In most cases increasing this number will improve performance, as long as you have enough free memory. The default size is 1 MB. If the value passed is H5D_CHUNK_CACHE_NBYTES_DEFAULT, then the property will not be set on *dapl_id* and the parameter will come from the file.

The *rdcc_w0* value is the chunk preemption policy. This should be between 0 and 1 inclusive and indicates the weighting according to which chunks which have been fully read or written are penalized when determining which chunks to flush from cache. A value of zero means fully read or written chunks are treated no differently than other chunks (the preemption is strictly LRU) while a value of one means fully read or written chunks are always preempted before other chunks. If your application only reads or writes data once, this can be safely set to 1. Otherwise, this should be set lower, depending on how often you are re-reading or re-writing the same data. The default value is 0.75. If the value passed is H5D_CHUNK_CACHE_W0_DEFAULT, then the property will not be set on *dapl_id* and the parameter will come from the file.

Parameters:

hid_t `dapl_id`	IN: Identifier of the dataset access property list.
size_t `rdcc_nslots`	IN: Number of chunk slots in the raw data chunk cache.
size_t `rdcc_nbytes`	IN: Total size of the raw data chunk cache, in bytes.
double `rdcc_w0`	IN: Preemption policy.

Returns:

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

2.2 H5Pget_chunk_cache

Name: H5Pget_chunk_cache

Signature:

herr_t `H5Pget_chunk_cache`(*hid_t* dapl_id, *size_t* `*rdcc_nslots`, *size_t* `*rdcc_nbytes`, *double* `*rdcc_w0`)

Purpose:

Queries the raw data chunk cache parameters.

Description:

`H5Pget_chunk_cache` retrieves the number of chunk slots in the raw data chunk cache, the maximum possible number of bytes in the raw data chunk cache, and the preemption policy value.

These values are retrieved from a dataset access property list. If the values have not been set on the property list, then values returned will be the same as the corresponding values for a default file access property list.

Any (or all) pointer arguments may be null pointers, in which case the corresponding datum is not returned.

Parameters:

hid_t `plist_id`	IN: Identifier of the dataset access property list.
size_t `*rdcc_nslots`	OUT: Number of chunk slots in the raw data chunk cache.
size_t `*rdcc_nbytes`	OUT: Total size of the raw data chunk cache, in bytes.
double `*rdcc_w0`	OUT: Preemption policy.

Returns:

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

2.3 H5Dget_access_plist

Name: H5Dget_access_plist

Signature:

hid_t `H5Dget_access_plist`(*hid_t* `dataset_id`)

Purpose:

Returns a dataset access property list identifier.

Description:

`H5Dget_access_plist` returns the dataset access property list identifier of the specified dataset.

The chunk cache parameters in the returned property lists will be those used by the dataset. If the properties in the file access property list were used to determine the dataset's chunk cache configuration, then those properties will be present in the returned dataset access property list. If the dataset does not use a chunked layout, then the chunk cache properties will be set to the default. The chunk cache properties in the returned list are considered to be "set", and any use of this list will override the corresponding properties in the file's file access property list.

All link access properties in the returned list will be set to the default values.

See "Dataset Access Properties" in H5P: Property List Interface in this reference manual for additional information and related functions.

Parameters:

hid_t `dataset_id`	IN: Identifier of the dataset to get access property list of.

Returns:

Returns a dataset access property list if successful; otherwise returns a negative value.

3 Future Plans

In the future these functions may be overloaded to accept either a dataset access property list or a file access property list, allowing H5Pset_cache and H5Pget_cache to be deprecated. This will allow a more consistent application of chunk caching parameters, as well as removing the unused parameter in those functions.

The entire chunk cache implementation may be rewritten at some point. In that case, these functions would be deprecated (except for H5Dget_access_plist) and replaced with something similar to H5Pset_mdc_config and H5Pget_mdc_config.

4 Possible enhancements

Should we modify H5Pget_chunk_cache to inform the user whether or not the values have been set? Currently there is no way to tell, and the library will behave differently depending on whether or not the values have been set. This could be done with an extra "flags" parameter, or three extra Boolean parameters.

5 Example

This example shows how to use the new functions, and illustrates how values in the dapl can override those in the fapl. It creates 2 files, with 1 dataset in each. The first file uses non-default values for the fapl, and these values are copied into a dapl using H5Dget_access_plist. This dapl is then used to create a dataset in file 2 using these properties. Finally, the properties in the dapl are modified and the dataset in file 1 is re-opened using this dapl.

```
// Create the file access property list
fapl = H5Pcreate(H5P_FILE_ACCESS);

// Set the chunk caching parameters on the fapl
H5Pset_cache(fapl, 0, nslots_1, nbytes_1, w0_1);

// Create the file using the fapl
file1 = H5Fcreate("file1", H5F_ACC_TRUNC, H5P_DEFAULT, fapl);

// Create dset1 using the default dapl.  This dataset will use the chunk caching
// parameters from the fapl (nelmts_1, nbytes1, w0_1)
dset1   =   H5Dcreate2(file1,   "dset1",   type,   space,   H5P_DEFAULT,   H5P_DEFAULT,
H5P_DEFAULT);

// Retrieve the dataset access property list from dset1.  This dapl will have the
// chunk caching parameters from fapl
dapl = H5Dget_access_plist(dset1);

// Create a new file with the default fapl
file2 = H5Fcreate("file2", H5F_ACC_TRUNC, H5P_DEFAULT, H5P_DEFAULT);

// Create a dataset in file2 with the retrieved dapl.  This dataset will use the
// chunk caching parameters from file1, even though file2 uses the default fapl.
dset2 = H5Dcreate2(file2, "dset2", type, space, H5P_DEFAULT, H5P_DEFAULT, dapl);

// Retrieve the chunk caching parameters from dapl
H5Pget_chunk_cache(dapl, &nslots_temp, &nbytes_temp, &w0_temp);

// Set a different value for nbytes nbytes on dapl.  The properties on this list
// will now be (nslots_1, nbytes_2, w0_1)
nbytes_temp = nbytes_2;
H5Pset_chunk_cache(dapl, nslots_temp, nbytes_temp, w0_temp);

// Re open dset1 using the modified dapl.  Notice we can pass dapl as the lapl
// argument to H5Oopen.
H5Dclose(dset1);
H5Oopen(file1, "dset1", dapl);
```

6 Appendix: Dataset Access Property Lists

While there are currently no dataset access properties in the HDF5 library, dataset access property lists can still be useful. Although not documented at the time of this RFC, dapl's are in fact a superset of link access property lists. This means that link access properties can be set on a dapl and used when opening or creating a dataset to, for example, specify the external link prefix to use while traversing the path. The same can be done with group and datatype access property lists.

The changes proposed here will continue to allow this use of dapl's, and allow both dataset and link access properties to be set on the same dapl. Moreover, a dapl can be passed to functions like H5Oopen instead of a lapl, which will allow the user to specify dataset access properties to use if the target of the H5Oopen call is a dataset.

Revision History

October 30, 2008: Version 1 circulated for comment within The HDF Group.

November 12, 2008: Version 2 circulated for comment within the HDF Group.

December 9, 2008: Version 3 is published and posted for public comment. Comments should be sent to nfortne2@hdfgroup.org or help@hdfgroup.org