Why are my files sizes different, if I open an HDF5 file more than once rather than writing the data out in one call?

The size discrepencies can be related to the way small metadata and raw data gets allocated in the file.

Currently, all metadata below a certain threshold size (2KB by default) will cause the library to allocate a block of that threshold size (i.e. 2KB) to store the metadata in, anticipating that more metadata will be added to the file soon and could be sub-allocated from that block. A program which doesn't add more metadata to the block will cause the rest of that block to be wasted in the file because the library doesn't currently remember the free space in the file from one file open to the next.

The threshold block size in the library can be changed with a call to H5Pset_meta_block_size (and H5Pset_small_data_block_size, in libraries which have it - should be in the 1.4.4 release) like so:

```
fapl_id=H5Pcreate(H5P_FILE_ACCESS);
 printf ("H5Pcreate returns: %i\n", fapl_id);

 status = H5Pset_meta_block_size (fapl_id,0);
 printf ("H5Pset_meta_block_size returns: %i\n", status);

#ifdef WHEN_ITS_AVAILABLE
 status = H5Pset_small_data_block_size (fapl_id,0);
 printf ("H5Pset_small_data_block_size returns: %i\n", status);
#endif /* WHEN_ITS_AVAILABLE */
```

Setting the block size to zero should really only be used when small amounts of metadata are being added each time the file is opened. Setting the block size to zero will intermix the raw data blocks allocated in the file with the metadata information in the file and cause the overall number of I/O operations on the file to increase (reducing performance), because the library cannot cache as much metadata in memory.

Performance-wise, it would be better to hold the file open as long as possible and not to adjust the block size, but users will have to decide whether file size or I/O performance is their overall goal.