Linux memory handling and performance

I am having performance problems. The memory runs down to where it requires swapping, and then the system is very slow. Using "top" I can see that the memory does not get released after an HDF5 file is closed. It stays in memory until the file is deleted.

The memory usage shown by "top" means total memory used by the system, both kernel and users. The Linux OS, contrary to many other Operating Systems, does not impose an upper limit on Kernel memory. If the kernel needs more memory, it grabs as much as there is physical memory. Output to disk is buffered in kernel memory first, before making its way slowly to disk. If a program is output intensive, it can quickly use up all physical memory. At that point, the whole system is memory starved. Not much gets done until the output data is written to the disks and frees up memory. Therefore, a Linux system can become very inefficient by just one write-intensive application.

This problem is not limited to HDF5 programs. Any write-intensive program can exhibit the same phenomenon. For example, keep "top" running in one window and do the following in another window. You will see the same behavior, assuming you have less than 2GB of memory. If you have more memory, increase the value of count=200 to exceed it.

```
% dd if=/dev/zero of=junk bs=10MB count=200  #generate a 2000MB file
% # by now top will show not much free memory left.
% rm junk
```

You will then see a big jump of free memory because the data in the dirty buffers is no longer valid and the memory is free for other use.

I believe it's a system problem, not the HDF5 library. I'm just looking for a manual flush that would alleviate my problem.

"Flushing" data means an application is very paranoid of the data integrity and is willing to initiate a real disk write and WAIT until it is finished. It is a requirement of data integrity at the cost of response speed. You can keep telling the system to flush the data out but the data can only move at the limit of the I/O channel. If you want your memory to be freed up sooner, you would have to employ one or more of the following:

1. faster disks and faster I/O channel;
2. more memory;
3. more disks and use stripping;

When a system write call is made, linux writes to buffer which marks the pages as dirty. There is a kernel parameter called the dirty background ratio which is a percentage of total system memory. When the system hits this number, a gang of processes are started at write the files to disk. This system exists to maximize IO without blocking IO bound processes.

In our case, we're saving ~125MB/s to disk which all goes into the buffer. When the buffer hits the limit, it slows the entire system down (we have pseudo real time elements which degrade in performance as a result).

We theorize that using something like fasdvise or fsync would maybe let us control the behavior of this buffer.

Currently, I am experimenting with getting the unix file descriptor using H5Fget_vfd_handle() and calling fsync().

I actually have it working with fsync(). We're getting some really good numbers.