HDF5 Update

High-Data Rate MX Meeting
NSLS-II at BNL
May 26, 2016
epourmal@hdfgroup.org
Outline

- HDF5 features requested and sponsored by Synchrotron Community (2012 -2016)
  - Direct chunk I/O (PSI, Dectris)
  - Dynamically loaded filters (DESY)
  - Single Write/ Multiple Reader or SWMR (DLS, ESRF, DESY)
  - Virtual Dataset (VDS) (DLS, DESY, XFEL)
- New requirements and features
- HDF5 roadmap for 2016
Direct chunk write: H5DOwrite_chunk
Performance results for H5DOwrite_chunk

Test result on Linux 2.6, x86_64
Each dataset contained 100 chunks, written by chunks

<table>
<thead>
<tr>
<th>Dataset size (MB)</th>
<th>95.37</th>
<th>762.94</th>
<th>2288.82</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size after compression (MB)</td>
<td>64.14</td>
<td>512.94</td>
<td>1538.81</td>
</tr>
<tr>
<td>Dataset dimensionality</td>
<td>100x1000x250</td>
<td>100x2000x1000</td>
<td>100x2000x3000</td>
</tr>
<tr>
<td>Chunk dimensionality</td>
<td>1000x250</td>
<td>2000x1000</td>
<td>2000x3000</td>
</tr>
<tr>
<td>Datatype</td>
<td>4-byte integer</td>
<td>4-byte integer</td>
<td>4-byte integer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>H5Dwrite writes without compression filter</th>
<th>speed(^1)</th>
<th>time(^2)</th>
<th>speed</th>
<th>time</th>
<th>speed</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>77.27</td>
<td>1.23</td>
<td>97.02</td>
<td>7.86</td>
<td>91.77</td>
<td>24.94</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>H5DOwrite_chunk writes uncompressed data</th>
<th>speed(^1)</th>
<th>time(^2)</th>
<th>speed</th>
<th>time</th>
<th>speed</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>79</td>
<td>1.21</td>
<td>95.71</td>
<td>7.97</td>
<td>89.17</td>
<td>25.67</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>H5Dwrite writes with compression filter</th>
<th>speed(^1)</th>
<th>time(^2)</th>
<th>speed</th>
<th>time</th>
<th>speed</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.68</td>
<td>35.59</td>
<td>2.67</td>
<td>285.75</td>
<td>2.67</td>
<td>857.24</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>H5DOwrite_chunk writes compressed data</th>
<th>speed(^1)</th>
<th>time(^2)</th>
<th>speed</th>
<th>time</th>
<th>speed</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>77.19</td>
<td>0.83</td>
<td>78.56</td>
<td>6.53</td>
<td>96.28</td>
<td>15.98</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Unix writes compressed data to Unix file</th>
<th>speed(^1)</th>
<th>time(^2)</th>
<th>speed</th>
<th>time</th>
<th>speed</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>76.49</td>
<td>0.84</td>
<td>95</td>
<td>5.4</td>
<td>98.59</td>
<td>15.61</td>
</tr>
</tbody>
</table>

\(^1\) Speed in MB/s  
\(^2\) Time in seconds
Dynamically loaded filters

• Problems with using custom filters
  • “Off the shelf” tools do not work with the third-party filters
    • h5dump, MATLAB and IDL, etc.
  • HDF5 tools cannot read file created by
    • h5py, PyTables, etc.

• Solution
  • Modify HDF5 source with your code and distribute it
    • And what will happen if a user wants filters from the different distributions???? Oh…. No…..
  • Use a 1.8.11 and later
  • Provide maintained library of HDF5 compression filters
Approach

• There are predefined default locations where the HDF5 library searches the shared libraries or DLLs with the HDF5 filter functions.
• The default location may be overwritten by an environment variable.
• Encoding: Standard programming model to register compression filter on write operation
• Decoding: Transparent on read; no user action required.
• Works great, but....
User’s headache and possible solutions

• Where to find HDF5 plugins?

• No official distribution site
  
  • Example of HDF5 filters distributions on Github
    
    ➢ [https://github.com/dectris/HDF5Plugin](https://github.com/dectris/HDF5Plugin)
    ➢ [https://github.com/Blosc/hdf5-blosc](https://github.com/Blosc/hdf5-blosc)
    ➢ [https://github.com/nexusformat/HDF5-External-Filter-Plugins](https://github.com/nexusformat/HDF5-External-Filter-Plugins)
    ➢ [https://svn.hdfgroup.org/hdf5_plugins/](https://svn.hdfgroup.org/hdf5_plugins/)

• The HDF Group can help with
  
  • Maintaining common repository
  • Testing plugin with the releases
  • Distributing source and binaries for Linux, Mac and Windows
    
    • Free and licensed (for fee) binary distributions
SWMR: Data access to file being written

New data elements...

...are added to a dataset in the file...

HDF5 File

...which can be read by a reader...

with no IPC necessary.

New data elements can be read by a reader with no IPC necessary.
- Released in HDF5 1.10.0
- Restricted to append-data only scenario
- SWMR doesn’t work on NFS
- Files are not compatible with HDF5 1.8.* libraries
- Use h5format_convert tool
  - Converts HDF5 metadata in place
  - No raw data is rewritten
• Data stored in multiple files and datasets can be accessed via one dataset (VDS) using standard HDF5 read/write.
VDS Example

Series of images

time

M

Dataset A

Dataset C

Dataset E

Dataset B

Dataset D

Dataset F

VDS.h5

Image at time $t_2$

$M$

$k$

$n$

$k$

$n$

$k$

$n$

$a.h5$

$b.h5$

$c.h5$

$d.h5$

$e.h5$

$f.h5$
• VDS works with SWMR
• File with VDS cannot be accessed by HDF5 1.8.* libraries
• Use h5repack tool to rewrite data (1.10.0-patch1)
New requirements and features?

• Tell us your needs:
  • Multi-threaded compression filters
  • H5DOread_chunk function
  • Full SWMR implementation
  • Performance
  • Backward/forward compatibility

• Other requests?
HDF5 Roadmap for 2016

• May 31 - HDF5 1.10.0-patch1
  • H5repack, Windows builds, Fortran issues on HPC systems

• Mid-summer  HDF5 1.10.1
  • Some internal issues found for 1.10.0

• December
  • HPC features that didn’t make it into 1.10.0 release
Questions?

Thank you!