Abstract

hdf5plugin is a *Python* package (1) providing a set of **HDF5** compression filters (namely: Blosc, Blosc2, BitShuffle, BZip2, FciDecomp, LZ4, SZ, SZ3, ZFP, ZStandard) and (2) enabling their use from the *Python* programming language with **h5py** a thin, pythonic wrapper around `libHDF5`.

This presentation illustrates how to use **hdf5plugin** for reading and writing compressed datasets from *Python* and gives an overview of the different HDF5 compression filters it provides.

In [ ]:
```
# Notebook requirements
# A recent version of Pillow is required!
#%pip install numpy h5py hdf5plugin h5glance rise jupyterlab matplotlib ipympl Pillow
```

In [ ]:
```
%matplotlib inline

# Creates data.h5 used for demos
from matplotlib import pyplot as plt
import h5py
import hdf5plugin
import numpy
from PIL import Image
import urllib.request

url = "https://www.desy.de/e409/e116959/e119238/media/7795/1963_Luftbild_DESY-Gelaend
filename = urllib.request.urlretrieve(url)[0]
image = numpy.array(Image.open(filename).convert("L"))
plt.imshow(image, cmap="gray")

h5file = h5py.File("data.h5", mode="w")
h5file["copyright"] = "DESY"
h5file.attrs["url"] = url
h5file.create_dataset("/data", data=image)
h5file.create_dataset(
    "/compressed_data",
    data=image,
    chunks=image.shape,
    **hdf5plugin.Blosc2('lz4', filters=hdf5plugin.Blosc.BITSHUFFLE)
)
h5file.close()
```

Restart kernel once the file is created!

In [ ]:
```
import os
os._exit(0)  # Makes the kernel restart
```

ESRF | The European Synchrotron

hdf5plugin

hdf5plugin packages a set of HDF5 compression filters and makes them usable from the Python programming language through h5py.

h5py is a thin, pythonic wrapper around HDF5.

Presenter: Thomas VINCENT

European HDF5 User Group Meeting 2023, September 19, 2023

In [1]:
```
from h5glance import H5Glance  # Browsing HDF5 files
H5Glance("data.h5")
```

Out[1]:

⊟ data.h5

compressed_data [📋]: 3744 × 5286 entries, dtype: uint8

copyright [📋]: scalar entries, dtype: UTF-8 string

data [📋]: 3744 × 5286 entries, dtype: uint8

In [2]:
```
import h5py  # Pythonic HDF5 wrapper: https://docs.h5py.org/

h5file = h5py.File("data.h5", mode="r")  # Open HDF5 file in read mode
data = h5file["/data"][()]               # Access HDF5 dataset "/data"
```

In [3]:
```
%matplotlib inline
from matplotlib import pyplot as plt

plt.imshow(data, cmap="gray")
```

Out[3]:
```
<matplotlib.image.AxesImage at 0x7f0896e60760>
```

In [4]:
```
data = h5file["/compressed_data"][()]  # Access compressed dataset
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
Input In [4], in <cell line: 1>()
----> 1 data = h5file["/compressed_data"][()]

File h5py/_objects.pyx:54, in h5py._objects.with_phil.wrapper()

File h5py/_objects.pyx:55, in h5py._objects.with_phil.wrapper()

File ~/venvs/py310/lib/python3.10/site-packages/h5py/_hl/dataset.py:758, in Dataset.__getitem__(self, args, new_dtype)
    756 if self._fast_read_ok and (new_dtype is None):
    757     try:
--> 758         return self._fast_reader.read(args)
    759     except TypeError:
    760         pass  # Fall back to Python read pathway below

File h5py/_selector.pyx:376, in h5py._selector.Reader.read()

OSError: Can't read data (can't open directory: /usr/local/hdf5/lib/plugin)
```

In [ ]:
```
# Check dataset's filters
plist = h5file["/compressed_data"].id.get_create_plist()
plist.get_filter(0)[0::3]
```

`hdf5plugin` usage

Reading compressed datasets

To enable reading compressed datasets not supported by `libHDF5` and `h5py`: Install **hdf5plugin** & import it.

In [ ]:
```
%%bash
pip3 install hdf5plugin
# Or:
conda install -c conda-forge hdf5plugin
```

Or on Debian12 and Ubuntu23.04:

In [ ]:
```
%%bash
apt-get install python3-hdf5plugin
```

In [5]:
```
import hdf5plugin
```

In [6]:
```
data = h5file["/compressed_data"][()]  # Access datset
plt.imshow(data, cmap="gray")          # Display data
```

Out[6]:
```
<matplotlib.image.AxesImage at 0x7f089182d480>
```

In [7]:
```
h5file.close()  # Close the HDF5 file
```

Writing compressed datasets

When writing datasets with `h5py`, compression can be specified with: h5py.Group.create_dataset

In [8]:
```
# Create a dataset with h5py without compression
h5file = h5py.File("new_file_uncompressed.h5", mode="w")
h5file.create_dataset("/data", data=data)
h5file.close()
```

In [9]:
```
# Create a compressed dataset
h5file = h5py.File("new_file_blosc2_bitshuffle_lz4.h5", mode="w")
h5file.create_dataset(
    "/compressed_data",
    data=data,
    compression=32026,  # Blosc2 HDF5 filter identifier
    # options: 0, 0, 0, 0, level, filter, compression
    compression_opts=(0, 0, 0, 0, 5, 2, 1)
)
h5file.close()
```

`hdf5plugin` provides some helpers to ease dealing with compression filter and options:

In [10]:
```
h5file = h5py.File("new_file_blosc2_bitshuffle_lz4.h5", mode="w")
h5file.create_dataset(
    "/compressed_data",
    data=data,
    compression=hdf5plugin.Blosc2(
        cname='lz4',
        clevel=5,
        filters=hdf5plugin.Blosc2.BITSHUFFLE),
)
h5file.close()
```

In [ ]:
```
help(hdf5plugin.Blosc2)
```

In [12]:
```
H5Glance("new_file_blosc2_bitshuffle_lz4.h5")
```

Out[12]:

⊟ new_file_blosc2_bitshuffle_lz4.h5

compressed_data [📋]: 3744 × 5286 entries, dtype: uint8

In [13]:
```
h5file = h5py.File("new_file_blosc2_bitshuffle_lz4.h5", mode="r")
plt.imshow(h5file["/compressed_data"][()], cmap="gray")
h5file.close()
```

In [14]:
```
!ls -sh new_file*.h5
```

```
18M new_file_blosc2_bitshuffle_lz4.h5  19M new_file_uncompressed.h5
```

HDF5 compression filters

Available through `h5py`

- Pre-compression filter: Byte-Shuffle provided by `libhdf5`
- Compression filters provided by h5py:
  - Provided by `libhdf5` : **"gzip"** and eventually **"szip"** (optional)
  - Bundled with `h5py` : **"lzf"**

In [15]:
```
h5file = h5py.File("new_file_shuffle_gzip.h5", mode="w")
h5file.create_dataset(
    "/compressed_data_shuffle_gzip", data=data, shuffle=True, compression="gzip")
h5file.close()
```

Provided by `hdf5plugin`

Additional compression filters provided by `hdf5plugin` :

BitShuffle, Blosc, Blosc2, BZip2, FciDecomp, LZ4, SZ, SZ3, ZFP, Zstandard

10 out of the 29 HDF5 registered filter plugins as of September 2023

In [16]:
```
h5file = h5py.File("new_file_bitshuffle_lz4.h5", mode="w")
h5file.create_dataset(
    "/compressed_data_bitshuffle_lz4",
    data=data,
    compression=hdf5plugin.Bitshuffle()
)
h5file.close()
```

General purpose lossless compression

- Bitshuffle(nelems=0, cname='lz4', clevel=3) - ID 32008
  - **Bit**-Shuffle + LZ4, ZStd or no compression
- BZip2(blocksize=9) - ID 307
- LZ4(nbytes=0) - ID 32004
- Zstd(clevel=3) - ID 32015

Specific compression

- FciDecomp() - ID 32018: Based on JPEG-LS:
  - **Optional**: requires C++11
  - Data type: `(u)int8` or `(u)int16`
  - Chunk shape: "Image-like"; 2 or 3 dimensions with at least 16 pixels and at most 65535 rows and columns and at most 4 planes for 3D datasets.

Lossy compression 1/2

SZcompressor family: error-bounded lossy compression

- SZ(absolute=None, relative=None, pointwise_relative=None) - ID 32017
- SZ3(absolute=None, relative=None, norm2=None, peak_signal_to_noise_ratio=None) - ID 32024

Lossy compression 2/2

- ZFP(rate=None, precision=None, accuracy=None, reversible=False, minbits=None, maxbits=None, maxprec=None, minexp=None) - ID 32013:
  - Data type: `float32`, `float64`, `(u)int32`, `(u)int64`
  - Chunk shape: must have at most 4 non-unity dimensions

Meta-compressor: Blosc family

- Blosc(cname='lz4', clevel=5, shuffle=1) - ID 32001:
  - Based on c-blosc: A blocking, shuffling and lossless compression library
  - Pre-compression shuffle: None, **Byte**-Shuffle, **Bit**-Shuffle
  - Compression: `blosclz`, `lz4`, `lz4hc`, `snappy` (**optional**, requires C++11), `zlib`, `zstd`
- Blosc2(cname='blosclz', clevel=5, filters=1) - ID 32026:
  - Based on c-blosc2: A high performance compressor optimized for binary data
  - Pre-compression filters: None, **Byte**-Shuffle, **Bit**-Shuffle, and more
  - Compression: `blosclz`, `lz4`, `lz4hc`, `zlib`, `zstd`
  - More filters and compressions can be supported: `zfp`, `htj2k`

Equivalent filters

`Blosc` and `Blosc2` includes some pre-compression filters and algorithms provided by other HDF5 compression filters:

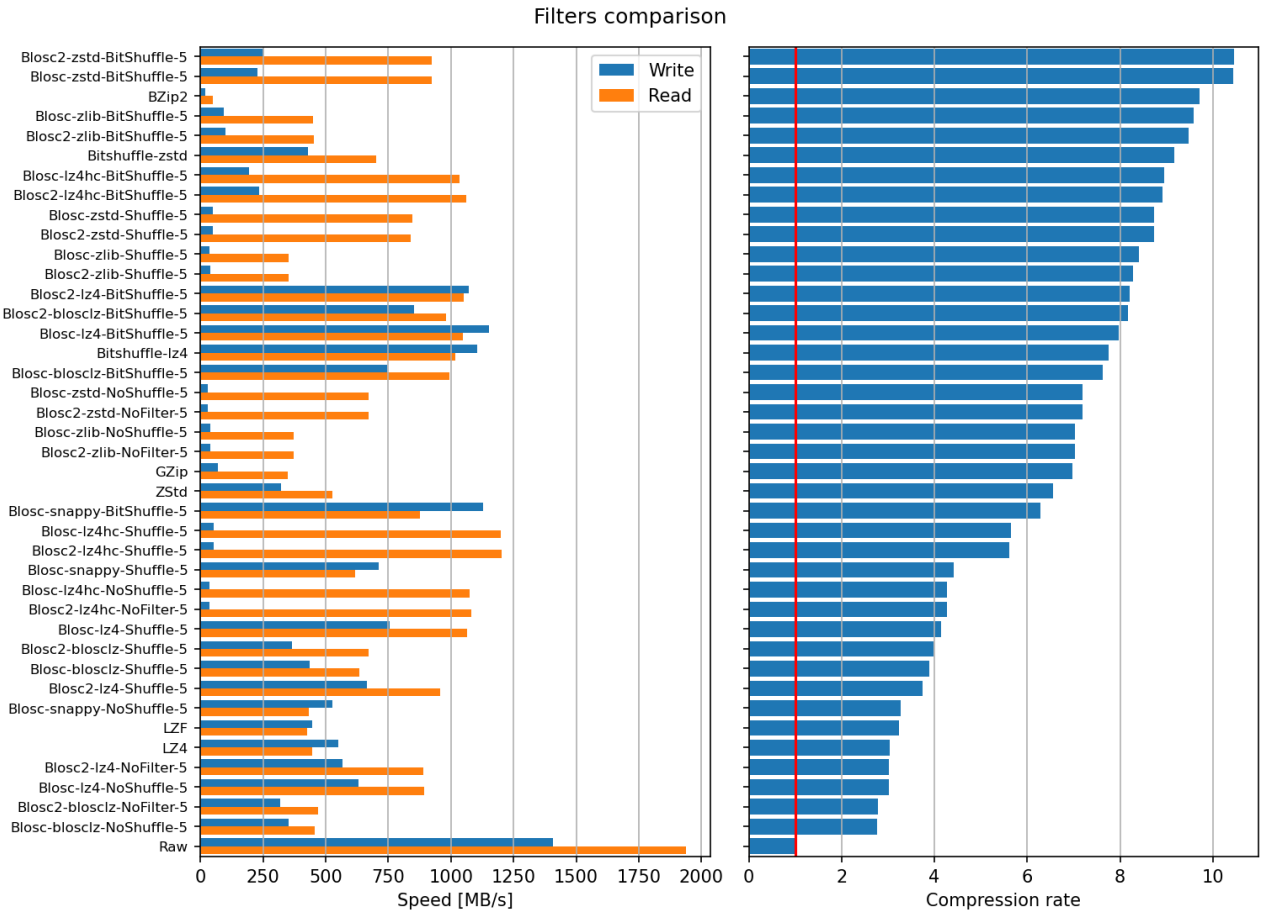
- HDF5 shuffle => `Blosc2(..., filters=Blosc2.SHUFFLE)`
- `Bitshuffle()` => `Blosc2("lz4" or "zstd", 5, Blosc2.BITSHUFFLE)`
- `LZ4()` => `Blosc2("lz4", 9)`
- `Zstd()` => `Blosc2("zstd", 2)`

`Blosc2` filter could also provides `ZFP` (not included yet in `hdf5plugin`)

A look at performances on a single use case

- Machine: Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz (40 cores)
- Filesystem: /dev/shm

- `hdf5plugin` built from source
- Running on a single core
- Diffraction tomography dataset: 100 frames from http://www.silx.org/pub/pyFAI/pyFAI_UM_2020/data_ID13/kevlar.h5
- Dataset: 100x2167x2070, uint16, chunk: 2167x2070



Multithreaded filter execution

Some filters can use multithreading:

- Blosc/Blosc2:
  - Using a pool of threads
  - Disabled by default for Blosc1
  - Configurable with `BLOSC_NTHREADS` environment variable
- Bitshuffle, Fcidecomp, SZ, SZ3, ZFP:
  - Using OpenMP
  - Enabled at compilation time
  - If enabled, configurable with `OMP_NUM_THREADS` environment variable

Performance do not increase linearly with the number of CPU cores used.

Summary

Having different pre-compression filters and compression algorithms at hand offers different read/write speed versus compression rate (and eventually error rate) trade-offs.

Also to keep in mind availability/compatibility: Since `"gzip"` is included in `libHDF5` it is the most compatible one (and also `"lzf"` as included in `h5py`).

Using `hdf5plugin` filters with other applications

Set the `HDF5_PLUGIN_PATH` environment variable to: `hdf5plugin.PLUGINS_PATH`

In [ ]:
```
%%bash
export HDF5_PLUGIN_PATH=`python3 -c "
import hdf5plugin; print(hdf5plugin.PLUGINS_PATH)"`
echo "HDF5_PLUGIN_PATH=${HDF5_PLUGIN_PATH}"
ls ${HDF5_PLUGIN_PATH}
```

Note: Only works for reading compressed datasets, **not for writing!**

A word about `hdf5plugin` license

The source code of `hdf5plugin` itself is licensed under the MIT license...

It also embeds the source code of the provided compression filters and libraries which are licensed under different open-source licenses (Apache, BSD-2, BSD-3, MIT, Zlib...) and copyrights.

Limitations

- Only "gzip" available by default:
  - Many compression filters provided by `hdf5plugin` are included in c-blosc2
- A central repository for filters source code: https://github.com/HDFGroup/hdf5_plugins?

- Need to link filters with libhdf5:
  - `hdf5plugin` relies on a "hack" to avoid linking with libhdf5

- Compressed data accessed by "chunks" even if compressor uses smaller blocks
- Multi-threaded access support
- When reading compressed data, some memory copy could be spared:
  - Direct chunk access offers a way to improve performance/flexibility

Avoid memory copies

Compression filters allocates a memory buffer to store decompressed data = memory copies.

Allowing the user to pass a memory buffer through h5py->libhdf5->compression_filter would prevent it.

An example with **h5py** and **Blosc2** (bitshuffle+lz4) for a 8.5MB chunk on 1 core (± ~300 µs):

- Standard access `dataset[()]` : 8.9 ms
- `read_direct()` to existing array: 5.2 ms
- `read_direct_chunk()` & decompression with `blosc2` : 3.7 ms

Credits

To hdf5plugin contributors: Armando Sole, @orioltinto, @mkitti , @Florian-toll, Jerome Kieffer, @fpwg, @mobiusklein , @junyuewang, @Anthchirp, and

to all contributors of embedded libraries

Partially funded by the LEAPS-INNOV and PaNOSC EU-project.

 This project has received funding from the European Union´s Horizon 2020 research and innovation programme under grant agreement no. 101004728 and 823852.

Conlusion

`hdf5plugin` provides additional HDF5 compression filters (namely: BitShuffle, Blosc, Blosc2, BZip2, FciDecomp, LZ4, SZ, SZ3, ZFP, Zstandard) mainly for use with h5py.

- Packaged for **pip** and **conda**
- Documentation: http://www.silx.org/doc/hdf5plugin/latest/
- Source code repository: https://github.com/silx-kit/hdf5plugin

Thanks for your attention! Questions?