The Journal of Open Source Software

pyfive: A pure-Python HDF5 reader

Bryan N. Lawrence [1,2], **Ezequiel Cimadevilla** [3], **Wout De Nolf** [4], **David Hassell** [1,2], **Jonathan Helmus**[5], **Benjamin Hodel**[8], **Brian Maranville** [6], **Kai Mühlbauer** [7], **and Valeriu Predoi** [1,2]

1 National Center for Atmospheric Science (NCAS), United Kingdom. **2** Department of Meteorology, University of Reading, Reading, United Kingdom. **3** Instituto de Física de Cantabria (IFCA), CSIC-Universidad de Cantabria, Santander, Spain. **4** European Synchrotron Radiation Facility (ESRF), Grenoble, France. **5** Astral Software Inc **6** NIST Center for Neutron Research **7** Institute of Geosciences, Meteorology Section, University of Bonn, Germany. **8** No affiliation.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

pyfive is an open-source thread-safe pure Python package for reading data stored in HDF5. While it is not a complete implementation of all the specifications and capabilities of HDF5, it includes all the core functionality necessary to read gridded datasets, whether stored contiguously or with chunks (with or without standard compression options). All data access is fully lazy, the data is only read from storage when the numpy data arrays are manipulated. Originally developed some years ago, the package has recently been upgraded to support lazy access, and to add missing features necessary for handling all the environmental data known to the authors. It is now a realistic option for production data access in environmental science and more widely. The API is based on that of h5py (https://github.com/h5py/h5py, a Python shimmy over the HDF5 C-library which itself is not thread-safe), with some API extensions to help optimise remote access. With these extensions, coupled with thread safety, many of the limitations precluding the efficient use of HDF5 (and netCDF4) on cloud storage have been removed.

Statement of need

HDF5[1] (Folk et al., 2011) is probably the most important data format in physical science. It is particularly important in environmental science given the fact that netCDF4[2] (Rew et al., 2006) is HDF5 under the hood. From satellite missions, to climate models and radar systems, the default binary format has been HDF5 for decades. While newer formats are starting to get mindshare, there are petabytes, if not exabytes, of existing HDF5, and there are still many good use-cases for creating new data in HDF5. However, despite the history, there are few libraries for reading HDF5 file data that do not depend on the official HDF5 library maintained by the HDF Group, and in particular, apart from pyfive, in Python there are none that cover the needs of environmental science. While the HDF5 c-library is reliable and performant, and battle-tested over decades, there are some caveats to depending upon it: Firstly, it is not thread-safe, secondly, the code is large and complex, and should anything happen to the financial stability of the HDF5 Group, it is not obvious the C-code could be maintained. Finally, the code complexity also meant that it is not suitable for developing bespoke code for data recovery in the case of partially corrupt data. From a long-term curation perspective both of these last two constraints are a concern.

The issues of the dependency on a complex code maintained by one private company in the

[1] https://www.hdfgroup.org/solutions/hdf5/
[2] https://www.unidata.ucar.edu/software/netcdf

context of maintaining data access (over decades, and potentially centuries), can only be mitigated by ensuring that the data format is well documented, that data writers use only the documented features, and that public code exists which can be relatively easily maintained. The HDF5group have provided good documentation for HDF5, but while there is a community of developers beyond the HDF5 group, recent events suggest that given most of those developers and their existing funding are US based, some spreading of risk would be desirable. To that end, a pure Python code covering the core HDF5 features of interest to the target scientific community, which is relatively small and maintained by an international constituency provides some assurance that the community can maintain HDF5 access for the foreseeable future. A pure Python code also makes it easier to develop scripts which can work around data and metadata damage should they occur, and has the additional advantage of being able to be deployed in resource or operating-system constrained environments (such as mobile).

Current Status of pyfive

The original implementation of `pyfive` (by JH), which included all the low-level functionality to deal with the internals of an HDF5 file was developed with POSIX access in mind. The recent upgrades were developed with the use-cases of performant remote access to curated data as the primary motivation - including full support for lazy loading only parts of chunked datasets as they are needed.

Thread safety has become a concern given the wide use of Dask[3] in Python based analysis workflows, and this, coupled with a lack of user knowledge about how to efficiently use HDF5, has led to a community perception that HDF5 is not fit for remote access (especially on cloud storage). `pyfive` addresses thread safety by bypassing the underlying HDF5 c-library and addresses some of the issues with remote access by optimising access to internal file metadata (in particular, the chunk indexes) and by supporting the determination of whether or not a given file is cloud optimised.

To improve internal metadata access, `pyfive` now supports several levels of "laziness" for instantating chunked datasets (variables). The default method preloads internal indices to make parallellism more efficient, but a completely lazy option without index loading is possible. Neither load data until it is requested.

To be fully cloud optimised, files needs sensible chunking, and variables need contiguous indices. Chunking has been, and is easy to determine. `pyfive` now also provides simple methods to expose information about internal file layout - both in API extensions, and via a new `p5dump` utility packaged with the `pyfive` library[4]. Either method allows one to determine whether the key internal "b-tree" indices are contiguous in storage, and to determine the parameters necessary to rewrite the data with contiguous indices. While `pyfive` itself cannot rewrite files to address chunking or layout, tools such as the HDF5 repack utility, can do this very efficiently(Hassell & Cimadevilla Alvarez, 2025).

With the use of `pyfive`, suitably repacked and rechunked HDF5 data can now be considered "cloud-optimised", insofar as with lazy loading, improved index handling, and thread-safety, there are no "format-induced" constraints on performance during remote access.

Acknowledgements

Most of the recent developments outlined have been supported by the UK Met Office and UKRI via 1) UK Excalibur Exascale programme (ExcaliWork), 2) the UKRI Digital Research Infrastructure programme (WacaSoft), and 3) the national capability funding of the UK

[3] https://www.dask.org/
[4] https://pyfive.readthedocs.io/

The Journal of Open Source Software

National Center for Atmospheric Science (NCAS). Ongoing maintenance of `pyfive` is expected to continue with NCAS national capability funding.

References

Folk, M., Heber, G., Koziol, Q., Pourmal, E., & Robinson, D. (2011). An overview of the HDF5 technology suite and its applications. *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, 36–47. https://doi.org/10.1145/1966895.1966900

Hassell, D., & Cimadevilla Alvarez, E. (2025). *Cmip7repack: Repack CMIP7 netCDF-4 datasets*. Zenodo. https://doi.org/10.5281/zenodo.17550920

Rew, R., Hartnett, E., & Caron, J. (2006). NetCDF-4: Software implementing an enhanced data model for the geosciences. *22nd Inernational Conference on Interactive Information Processing Systems for Meteorology, Oceanography and Hydrology*.