

¹ pyfive: A pure-Python HDF5 reader

² **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}

⁵ 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](https://doi.org/10.xxxxxx/draft)

Software

- ¹⁰ [Review](#) 
- ¹¹ [Repository](#) 
- ¹² [Archive](#) 

Editor: [Open Journals](#) 

Reviewers:

- ¹³ [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).²¹²²²³

¹⁰ Summary

¹¹ pyfive (<https://pyfive.readthedocs.io/en/latest/>) 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, and to carry out the necessary decompression for the standard 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 (which is a Python shimmy over the HDF5 C-library, and hence 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.¹⁸¹⁹²⁰²¹²²²³

²⁴ Statement of need

²⁵ HDF5¹ ([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 netCDF4² ([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.²⁶²⁷²⁸²⁹³⁰³¹³²³³³⁴³⁵³⁶³⁷³⁸³⁹

⁴⁰ 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>

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

53 Current Status

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

59 Thread safety has become a concern given the wide use of Dask³ in Python based analysis
60 workflows, and this, coupled with a lack of user knowledge about how to efficiently use HDF5,
61 has led to a community perception that HDF5 is not fit for remote access (especially on cloud
62 storage).
63 pyfive addresses thread safety by bypassing the underlying HDF5 c-library and addresses some
64 of the issues with remote access by, firstly optimising access to internal file metadata (and in
65 particular the chunk indexes) and, secondly exposing information about internal file structure
66 which allows informed decisions about how to make data cloud optimised⁴.
67 pyfive now supports several levels of “laziness” for instantiating chunked datasets (variables).
68 The default method preloads internal indices to make parallelism more efficient, but a completely
69 lazy option without index loading is possible. Neither load data until it is requested.
70 To be fully cloud optimised, files need sensible chunking, and variables need contiguous indices.
71 Chunking has been, and is easy to determine. pyfive now also provides simple methods
72 to expose information about file layout - both in API extensions, and via a new p5dump
73 utility packaged with the pyfive library. Either method allows one to determine whether the
74 key internal “b-tree” indices are contiguous in storage. While pyfive itself cannot rewrite
75 files to address chunking or layout, tools such as the HDF5 repack utility, can do this very
76 efficiently(Hassell & Cimadevilla Alvarez, 2025).
77 With the use of pyfive, suitably repacked and rechunked HDF5 data can now be considered
78 “cloud-optimised”, insofar as with lazy loading, improved index handling, and thread-safety,
79 there are no “format-induced” constraints on performance during remote access.

80 Acknowledgements

81 Most of the recent developments outlined have been supported by the UK Met Office and
82 UKRI via 1) UK Excalibur Exascale programme (ExcaliWork), 2) the UKRI Digital Research
83 Infrastructure programme (WacaSoft), and 3) the national capability funding of the UK
84 National Center for Atmospheric Science (NCAS). Ongoing maintenance of pyfive is expected
85 to continue with NCAS national capability funding.

³<https://www.dask.org/>

⁴<https://pyfive.readthedocs.io/cloud.html>

86 References

- 87 Folk, M., Heber, G., Koziol, Q., Pourmal, E., & Robinson, D. (2011). An overview of the HDF5
88 technology suite and its applications. *Proceedings of the EDBT/ICDT 2011 Workshop on*
89 *Array Databases*, 36–47. <https://doi.org/10.1145/1966895.1966900>
- 90 Hassell, D., & Cimadevilla Alvarez, E. (2025). *Cmip7repack: Repack CMIP7 netCDF-4*
91 *datasets*. Zenodo. <https://doi.org/10.5281/zenodo.17550920>
- 92 Rew, R., Hartnett, E., & Caron, J. (2006). NetCDF-4: Software implementing an enhanced
93 data model for the geosciences. *22nd International Conference on Interactive Information*
94 *Processing Systems for Meteorology, Oceanography and Hydrology*.

DRAFT