

# <sup>1</sup> pyfive: A pure-Python HDF5 reader

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

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

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- <sup>10</sup> [Review](#) 
- <sup>11</sup> [Repository](#) 
- <sup>12</sup> [Archive](#) 

---

Editor: [Open Journals](#) 

## Reviewers:

- <sup>13</sup> [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

Authors of papers retain copyright  
<sup>22</sup> and release the work under a  
Creative Commons Attribution 4.0  
International License ([CC BY 4.0](#))<sup>23</sup>

## <sup>10</sup> Summary

<sup>11</sup> 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. 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.

## <sup>14</sup> Statement of need

<sup>15</sup> HDF5<sup>1</sup> ([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<sup>2</sup> ([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.

<sup>16</sup> The issues of the dependency on a complex code maintained by one private company in the context of maintaining data access (over decades, and potentially centuries), can only be

<sup>1</sup><https://www.hdfgroup.org/solutions/hdf5/>

<sup>2</sup><https://www.unidata.ucar.edu/software/netcdf>

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).

## 52 Current Status

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

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

65 To improve internal metadata access,pyfive now supports several levels of “laziness” for  
66 instantiating chunked datasets (variables). The default method preloads internal indices to  
67 make parallelism more efficient, but a completely lazy option without index loading is possible.  
68 Neither load data until it is requested.

69 To be fully cloud optimised, files needs sensible chunking, and variables need contiguous indices.  
70 Chunking has been, and is easy to determine. pyfive now also provides simple methods to  
71 expose information about internal file layout - both in API extensions, and via a new p5dump  
72 utility packaged with the pyfive library<sup>4</sup>. Either method allows one to determine whether  
73 the key internal “b-tree” indices are contiguous in storage, and to determine the parameters  
74 necessary to rewrite the data with contiguous indices. 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.

<sup>3</sup><https://www.dask.org/>

<sup>4</sup><https://pyfive.readthedocs.io/>

## 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