12A.5 NETCDF-4 PERFORMANCE IMPROVEMENTS OPENING COMPLEX DATA FILES

Edward Hartnett [1,2]

1 CIRES, University of Colorado, Boulder, CO 80309, USA
2 NOAA/ESRL/GSD, Boulder, CO 80305, USA

ABSTRACT

The netCDF-4/HDF5 format offers many advantages for science data users, including built-in compression, a rich data model, and support for parallel I/O. However, several performance bottlenecks have been identified which impact complex files with many variables and attributes defined. These files are especially prevalent in the domain of High Performance Computing (HPC).

The performance issues appear when files include many variables, attributes, or dimensions, and include slow file opens.

A focus on netCDF-4 performance issues has resulted in the elimination of several bottlenecks, including time to open a file, looking up objects in the file, and reading strided arrays. Significant improvement is demonstrated.

Performance trials are conducted on NOAA's Thea platform. Results are presented demonstrating the improved performance with real-world example files.

1 BACKGROUND

NetCDF is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data.

With NetCDF version 4.0 (released in June, 2008), netCDF users can use HDF5 as the storage format, accessed through the netCDF APIs. This enabled the introduction of the enhanced data model, which included groups, user-defined types, and new unsigned and 64-bit integer types. (Folk, 2003, Rew 2006, Rew 2010, Rew 2011).

The netCDF data model includes:

- variables - multi-dimensional arrays of data.
- attributes - metadata which are important to understand the context of the variable data (ex units).

Users can define attributes, but there are some reserved names which the user cannot use. In some cases, these reserved attributes are hidden in the netCDF API. One of the performance enhancements described below will use an already-defined hidden attribute in a new way to improve performance.

2 PERFORMANCE BOTTLENECKS IN NETCDF-4

Some performance bottlenecks in netCDF code stem from the reading of all metadata at file open.

When the user opens a netCDF file (using a version prior to 4.6.2), the contents of the file are scanned, and all metadata are read into memory. This includes all attributes in the file. Subsequent user requests for metadata are fast, because the file is not touched, however the cost of reading all the metadata must be bourne at file open time.

In the classic-based binary formats (classic, 64-bit offset, and CDF5), all file metadata are stored together at the beginning of the file; reading them is fast. However, HDF5 scatters the metadata in the file.

The HDF5 library does not scan the metadata in a file when the file is opened with HDF5. An attribute is only read when and if a user requests it. Consequently, the HDF5 library does not incur a penalty at open time.

For small, simple files, this make little difference, but for the large and complex files often used on HPC systems, there can be a performance penalty opening netCDF-4 files.

3 PERFORMANCE TESTING

To test performance in opening complex files, two new test programs were added to the netCDF C library.

Corresponding author address: Edward J. Hartnett, edward.hartnett@noaa.gov

These programs are only build if the --enable-benchmarks configure option is used.

In order to defeat any hardware buffering which may skew results, each run of the test creates, and then deletes, it's own data file.

To take measurements for this paper, the programs were backported to netCDF versions 4.6.0, 4.6.1, and 4.6.2. The library code after version 4.6.2 is tested first with lazy variable metadata reads, then also includes the use of the coordinates hidden attribute instead of dimscale matching.

3.1 Program tst_attspeft.c

The tst_attspef test creates files with varying number of variables, global and variable attributes..

When testing for variable performance, this program writes files with many variables. Each variable has 10 attributes.

When testing for global attribute performance, the files are created with no variables, and with varying number of global attributes.

When testing for variable attribute performance, the files are created with one variable and varying numbers of attributes attached to that variable. Each attribute is an array of double, length 100.

A timer is used to time a netcdf open/close of the file. HDF5 open/close is also timed (including opening the root group).

The open/close cycle is run 5 times, and the results averaged.

3.2 Program tst_wrf_reads.c

The program tst_wrf_reads.c creates three different types of files, and times the open/close operations. The three file types are:

1. A WRF model chemistry file with > 360 variables.
2. A user-contributed file we call the MERR file, with many global attributes, and variables with attributes.
3. A WRF BDY file. This is a file used in all WRF runs.

For each type of file, the file is created, then closed. Then the file is opened and closed. The creation/close, open, and close operations are timed. Note that no data are written to these files, they only contain metadata.

For each test run, 100 of each file type are created, and the average times used.

3.3 Testing Platforms

Tests were run on two different machines:

- mikado is a dedicated 6-core Intel i7 system, with a solid state drive. When running tests, no other users were on the system, and no other programs were attempting disk access.
- theia is a NOAA 760 Tflop Cray Compute Cluster high performance computing system.

3.4 NetCDF Versions Tested

Testing was done with several netCDF versions, including two intermediate versions after the released 4.6.2, which include code changes that are waiting to be merged into the netCDF master branch.

- 4.6.0 - released January 25, 2018.
- 4.6.1 - released March 19, 2018.
- 4.6.2 - released November 19, 2018.
- lazy var metadata - includes changes to read some variable metadata lazily (i.e. only when a user request for necessary informat triggers the read) See pull request #1260 (https://github.com/Unidata/netcdf-c/pull/1260).
- coordinates att - includes lazy var reads of some metadata, also uses the hidden coordinate attribute to speed opens on newly created netCDF-4 files. Pull request #1262 (https://github.com/Unidata/netcdf-c/pull/1262).

4 ATTRIBUTE PERFORMANCE IMPROVEMENTS

Two improvements have been made to attribute reading, and both of these improvements were part of the netCDF 4.6.2 release.

4.1 Lazy Attribute Read

Starting with version 4.6.2, no attributes are read at file open.

When the user requests a global attribute, then all global attributes (for that group) are read. When the user requests a variable attribute, all attributes of that variable are read.

With the new lazy read code, a file with many attributes can be opened very quickly. If the attributes are never requested by the user, they are never read. If only some variable attributes are requested by the user, then the attributes of all the other variables are never read.

Attributes are read for a variable or the group when the user:

- Does an nc_get_att_*() for any attribute of the variable (or group).
- Does an nc_inq_var() to get the number of attributes for a variable (or nc_inq() for the group attributes).
- Does an nc_del_att()/nc_rename_att()/nc_copy_att() for any attribute of the variable (or group).

4.1.1 Results of Lazy Attribute Performance Testing

Lazy attribute reads result in a significant performance improvement when a file contains many attributes.

The improvement in performance starting with version 4.6.2 is apparent when opening files with one variable with many attributes (see figures 1 and 2).

Even greater improvement is seen when opening a file with many global attributes (see figures 3 and 4).

One surprising result is the extent to which large numbers of variable attributes still cause the file to be slower to open. This is under investigation (See figure 1.).

Lazy attribute reading also applies to global attributes, and results in significant improvement in performance opening a file. (See figures 3 and 4.)

4.2 Fast Global Attribute Reads

Global attributes apply to the entire data file (or group). In a NetCDF-3 file, there is only one set of global attributes in a file, but a netCDF-4 file can have many groups, and each group may have global attributes.

In the netCDF-4 C library code, global attributes were being read in a different way from variable attributes. The variable attributes used a HDF5 iterator, and the global attributes used a for-loop, in which each attribute is accessed by number.

The iterator is orders of magnitude faster.

The netCDF library code has been changed so that global attributes are now read using the iterator. (The same code is now used to read both global and variable attributes).

This results in much faster read times for global attributes.

4.2.1 Results of Fast Attribute Reads

With lazy attribute reads, the cost of opening a file with global attributes has already been reduced, because the attributes are not read until requested. When any global attribute is read, then all are read and held in the metadata buffer. With the new, faster read code, there is a significant performance improvement in reading global attributes. (For this test the file is opened, then the number of global attributes checked. This triggers a read of all global attributes.) (See figures 5.)

With fast global attribute reads, opening the file and checking global attributes is almost as fast as opening the file in HDF5, which does no metadata reading at all. As can been seen in the graph the performance with and without lazy attribute reading is similar, as expected, because in this test the number of global attributes is inquired, and this triggers a read of all global attributes.

Checking this work on HPC system their, similar results were obtained. The time to open a netCDF-4 file with attributes has improved dramatically. A strong relationship between the number of attributes and the opening time has been eliminated. (See figure 6.)

5 VARIABLE PERFORMANCE IMPROVEMENTS

On file open, metadata is read from the file relating to the HDF5 datasets in the file. These datasets are either netCDF variables and/or dimensions.

The following metadata is read for each dataset:

- scale (see below)
- type
- cache settings

- chunking settings
- fill settings and value

Note that the dimension scale information is read at file open. Dimension scale information can be used to map the dimensions to the variables in the file.

After looping through the objects in the file, the netCDF library matches the dimension scale info of the datasets. Testing has shown that this mapping is taking a significant amount of the file open time.

5.1 Lazy Read of Some Variable Metadata

With lazy read of some variable metadata, much of the variable metadata are not read at file open time.

At file open, we need the type and some of the scale settings. None of the other metadata are immediately needed.

With the lazy read of some variable metadata, these other elements of the metadata are read for the variable only if needed. When the user uses a variable, the additional variable metadata are read.

This results in faster file open times for files with many variables. (See figures 7 and 8.) It also results in improvements in the file open time for the real-world example files (see figures 9, 10, 11, 12, and 13).

5.2 Using Hidden COORDINATES Attribute

One case in which HDF5 dimension scales cannot completely model netCDF dimensions and coordinate variables is the case of a dimension with a multi-dimensional coordinate variable. Dimension scales can only ever be one dimension.

To handle this, a hidden attribute "_Netcdf4Coordinates" is used to hold the list of dimension IDs associated with a multi-dimension coordinate variable. This hidden attribute has been present in all versions of netCDF-4.

By turning this attribute on for all variables, it is possible to use this hidden attribute to map variables to dimensions. This allows the reading of dimscale metadata to be deferred. It also allows the matching of dimension scale information to be skipped. This results in faster file open times. (See figures 7 and 8.) It also results in improvements in the file open time for the real-world example files (see figures 9, 10, 11, 12, and 13).

This performance improvement applies only to newly created netCDF-4 files. Existing netCDF-4 files do not possess the hidden coordinates attribute, Their dimensions must be determined by matching the dimension scale information, which is slower.

6 OTHER PERFORMANCE IMPROVEMENTS FOR INTERNAL METADATA LISTS IN NETCDF-4.4.2

NetCDF release 4.4.2 included some additional performance improvements related to file opens.Testing determined that these changes had a small impact for the tested data compared to the lazy attribute reads.

The additional performance improvements were:

- Hashed Lists
- Reduction of Unnecessary Lookups

The use of hashed lists had a small effect on the test programs. The reductions in unnecessary lookups had little or no effect on the performance in these tests.

7 SUMMARY

The opening of NetCDF-4/HDF5 files by the netCDF C library has been examined for performance issues, and four performance improvements have been developed:

1. Lazy attribute reads
2. Fast global attribute reads
3. Lazy read of some variable metadata
4. Using hidden coordinates attribute for dimension mapping.

Improvements 1 and 2 have been merged with the netCDF code base and were released with version 4.6.2. Improvements 3 and 4 have been submitted as pull requests to the netCDF project. If accepted, they will be in the next released version of netCDF, 4.6.3.

Improvements 1 through 3 will apply to any netCDF-4 file. Improvement 4 will only apply to files created with hidden coordinates attribute, which means only newly created files will benefit from this performance improvement.

8 REFERENCES

M. Folk, R. Rew, M. Yang, Q. Koziol, E Hartnett, R. E. McGrath, NetCDF-4: Combining netCDF and HDF5 Data, December 2003, AGU Fall Meet. Abstr.

Rew R., E. Hartnett, D. Heimbigner,J. Caron, NetCDF-4: Software Implementing an Enhanced Data Model for the Geosciences, January 2006, Conference: 22nd International Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology

Rew R., E. Hartnett, D. Heimbigner,J. Caron, Advances in the NetCDF Data Model, Format, and Software December 2010, University Corporation for Atmospheric Research, Boulder, CO

Russ Rew, Glenn Davis, Steve Emmerson, Harvey Davies, Ed Hartnett, Dennis Heimbigner and Ward Fisher, December 2018: The NetCDF Users Guide, University Corporation for Atmospheric Research, Boulder, CO

8 FIGURES

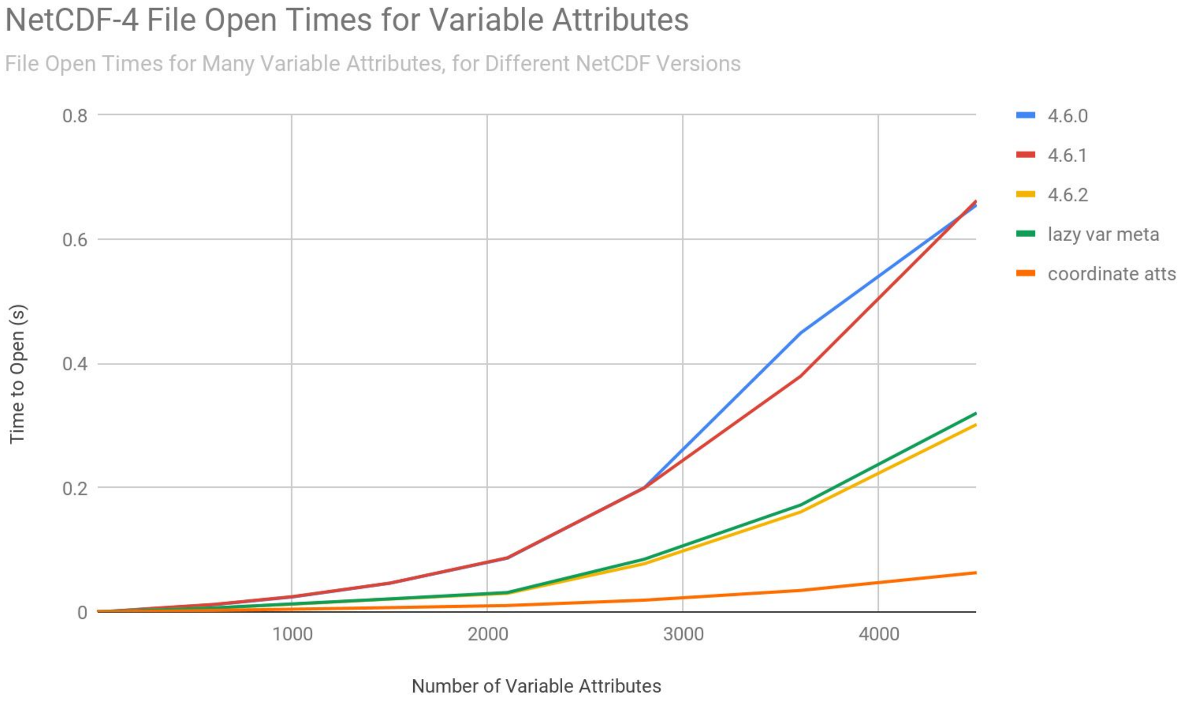

Figure 1: Time to open file with many variable attributes. With lazy attribute reads (introduced in 4.6.2), files with many attributes open more quickly. Using the hidden coordinates attribute further speeds file opens. Tests were run on workstation mikado.

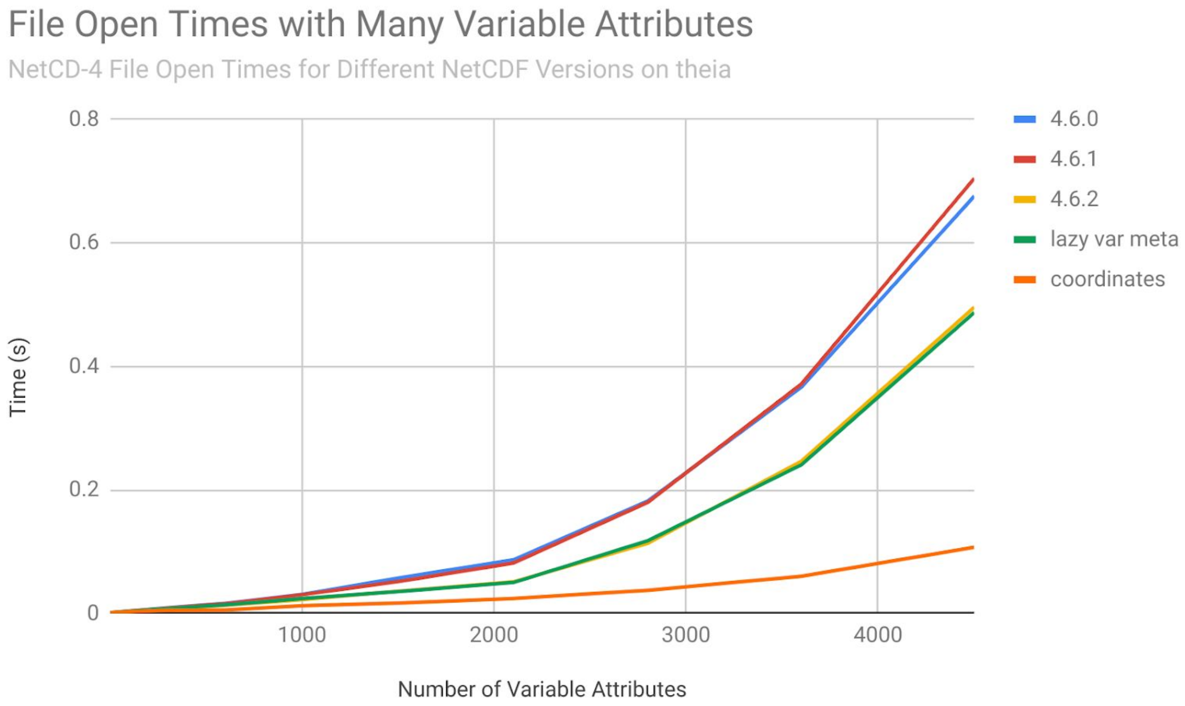

Figure 2: In this chart the time to open a netCDF-4 file with many variables is shown on theia. As on the development workstation mikado, version 4.6.2 improves performance. Use of hidden coordinates attribute for dimension mapping adds additional performance improvements. Test run on theia.

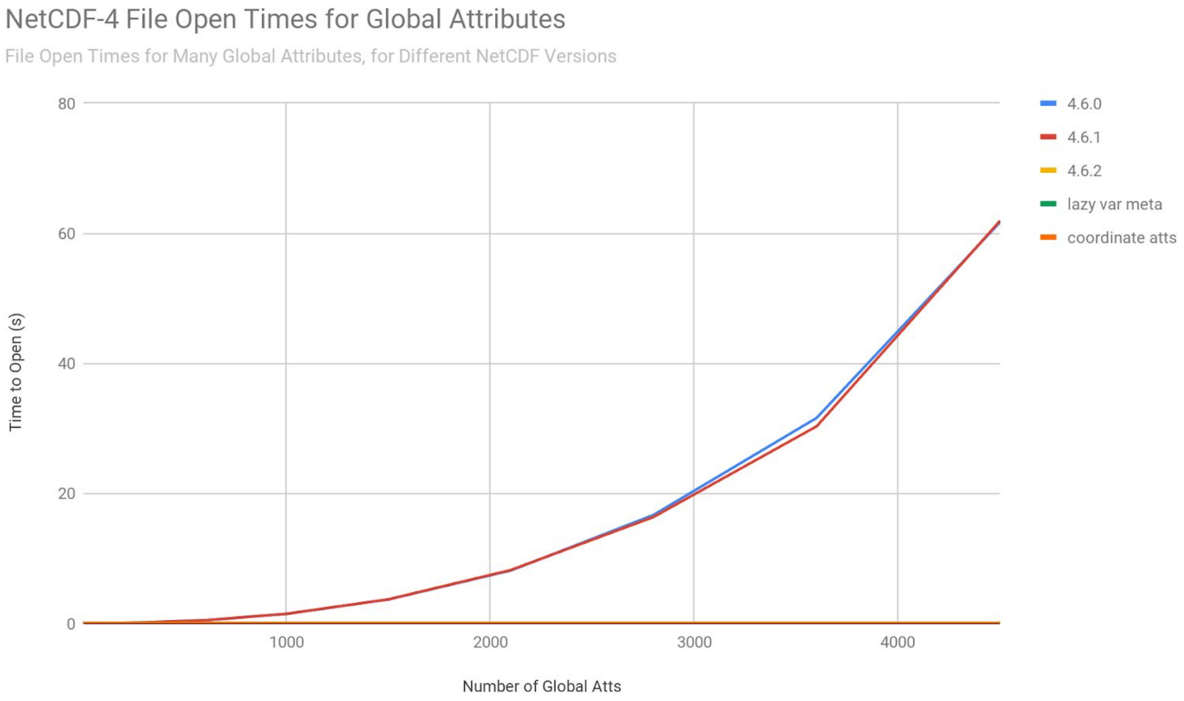

Figure 3: Time to Open NetCDF-4 File with Global Attributes. Using lazy attribute read (introduced in version 4.6.2), global attributes are not read at file open. All versions of netCDF-4 after 4.6.2 show almost zero open time global attributes. Tests were run on workstation mikado.

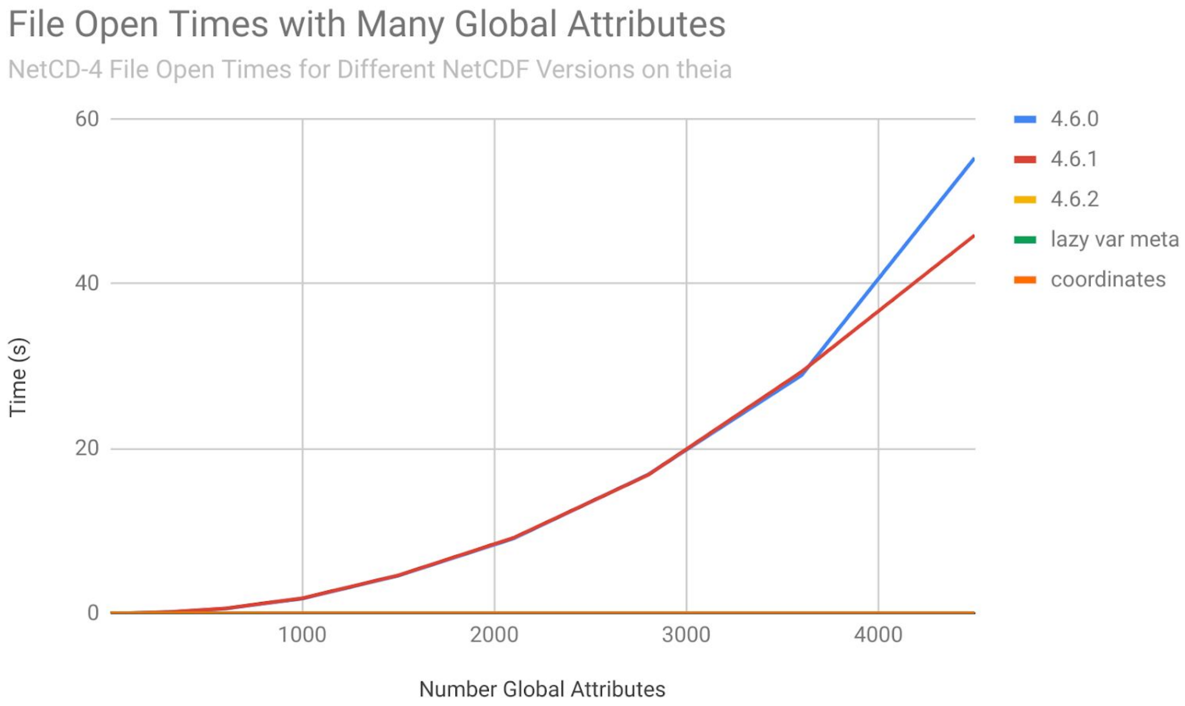

Figure 4: This chart shows the time taken by different versions of netCDF to open a netCDF-4 file with many global attributes on theia. Due to the introduction of lazy attribute reading, the time to open the file is greatly reduced starting with version 4.6.2. This matches the results on the development workstation. Tests were run on theia.

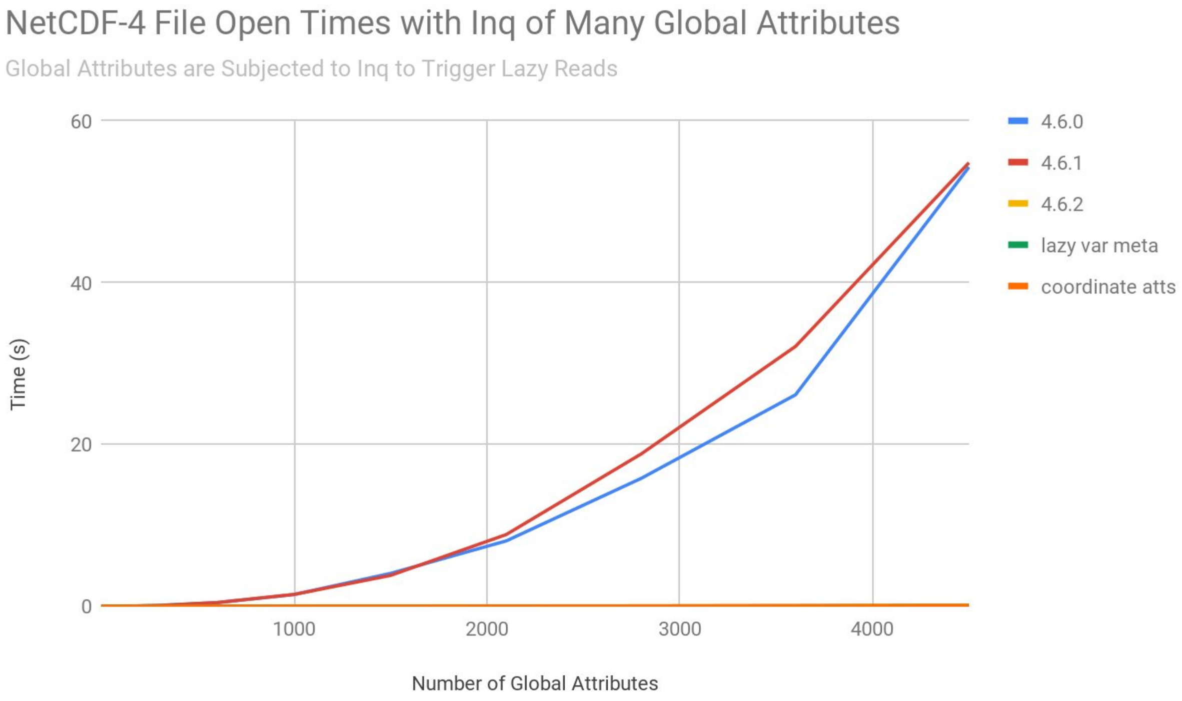

Figure 5: Time to Open NetCDF-4 File and Check Global Attributes: Use of fast global attribute read results in performance improvement for files that use global attributes. In this test, the file is opened, and nc_inq() is called, which triggers a read of the global attributes. Starting with version 4.6.2, reads of the global attributes are much faster. Tests were run on workstation mikado.

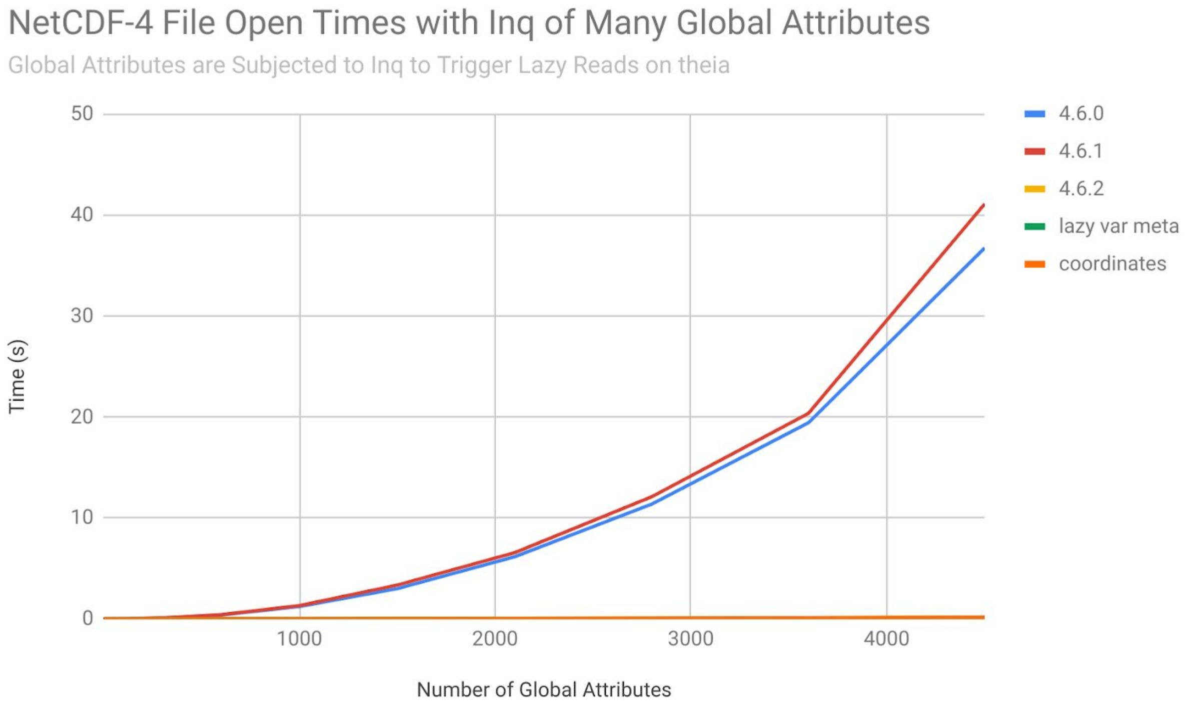

Figure 6: This chart shows the time to open a netCDF-4 file with many global attributes, and perform an nc_inq(), for different netCDF versions, on theia. The call of nc_inq() defeats the benefits of lazy attribute reads. The global attributes are being read in all cases. But starting with version 4.6.2, fast read of global attributes has removed the former performance penalty of global attributes. Tests were run on theia.

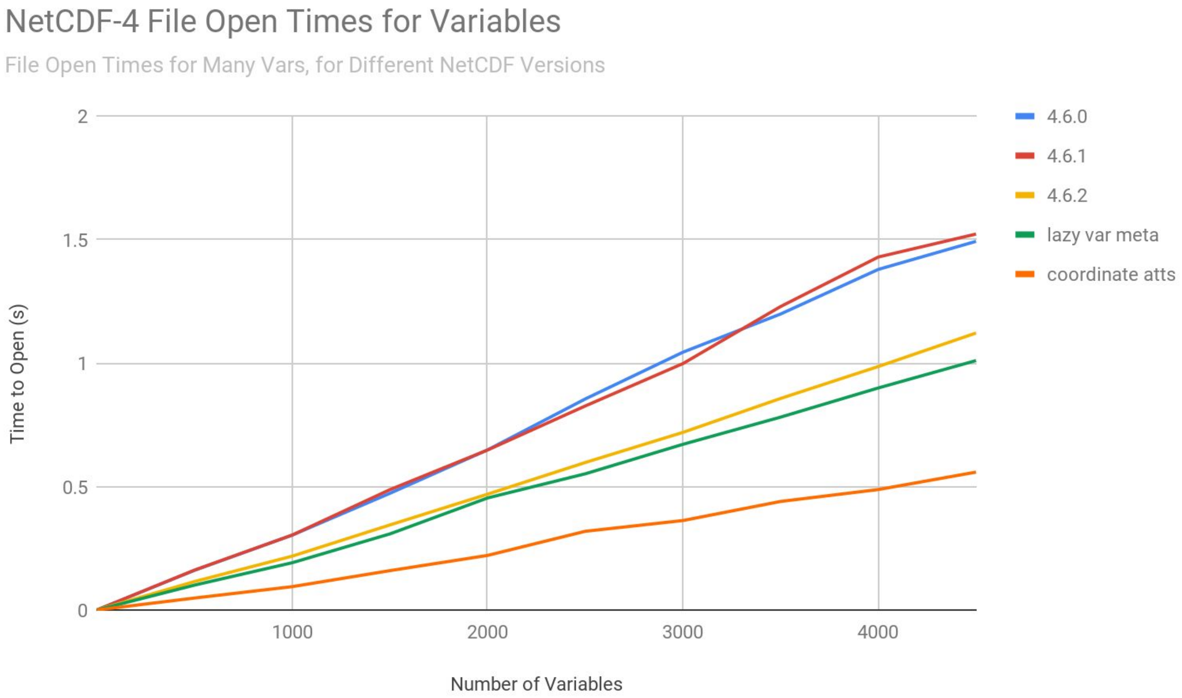

Figure 7: Time to Open NetCDF-4 File with Many Variables. Times are shown for recently release versions of netCDF, as well as the current (December 2018) master branch, with lazy variable metadata reads, and with use of hidden coordinates attributes. Each variable also has 10 attributes. Version 4.6.2 introduced lazy attribute reads, leading to improvement in performance. Lazy variable metadata reads adds some performance improvement. The use of the hidden coordinates attribute for dimension mapping adds further performance improvement. Tests were run on workstation mikado.

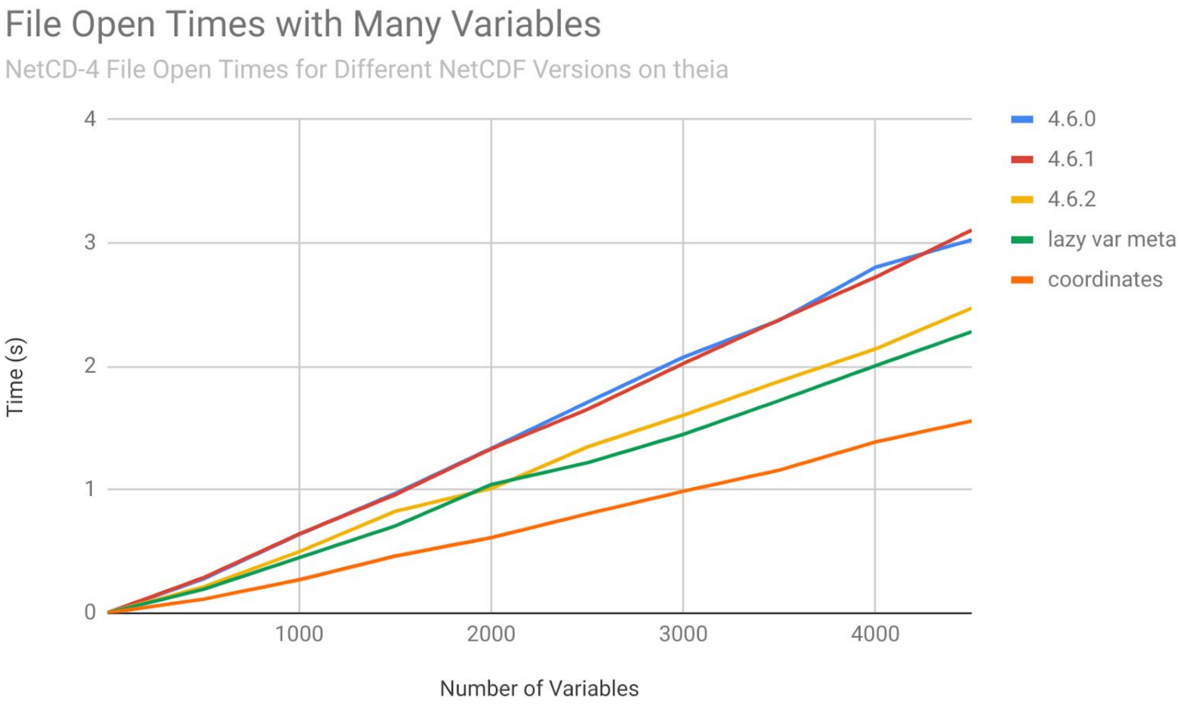

Figure 8: This chart shows the time to open a netCDF-4 file with may variables in different versions of netCDF on theia. Each variable in the test file has 10 attributes. The lazy attribute reads of version 4.6.2 improve performance. Further small improvement is obtained from doing a lazy read of some of the variable metadata. Further performance improvement is seen using the hidden coordinates attributes to map dimensions. Test were run on theia.

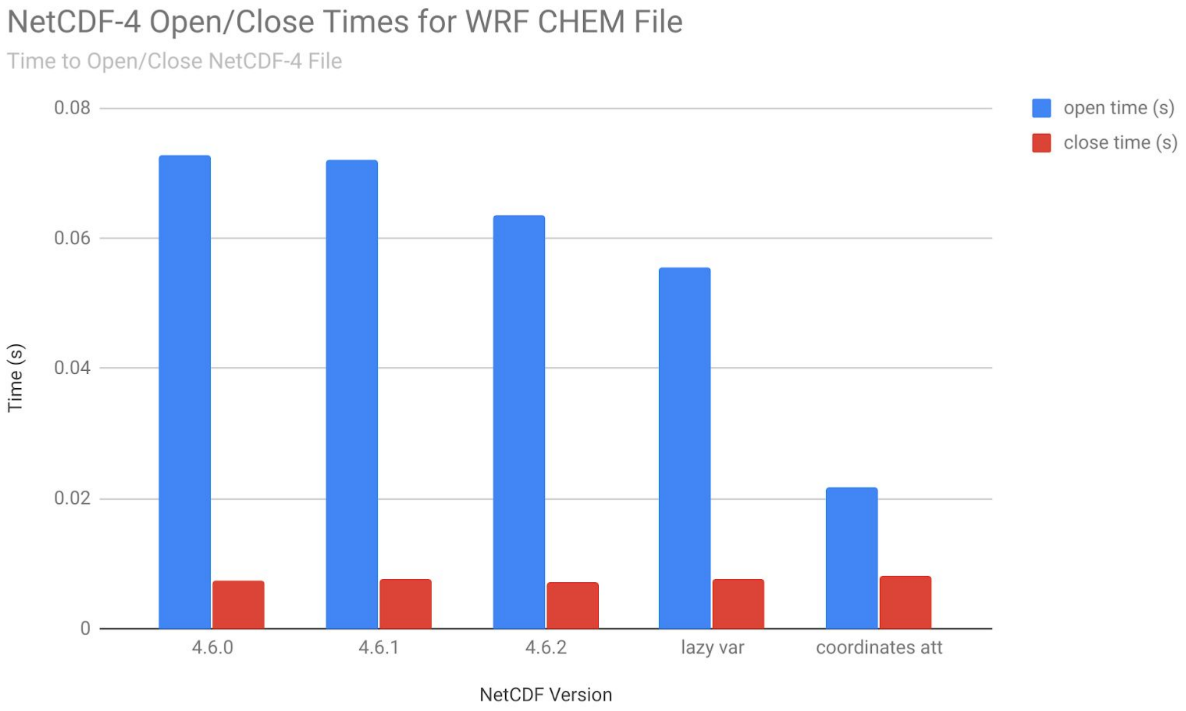

Figure 9: Open Times for NetCDF-4 WRF CHEM file. The data file is a real-world netCDF file from the WRF model. This chart shows the times for different versions of netCDF to open and close the sample data file. In version 4.6.2 there is an improvement due to lazy attribute reads and other performance improvements. When lazy var metadata reads are used, there is a further improvement. When hidden coordinates attributes are used for dimension mapping, there is a larger improvement. Tests were run on workstation mikado.

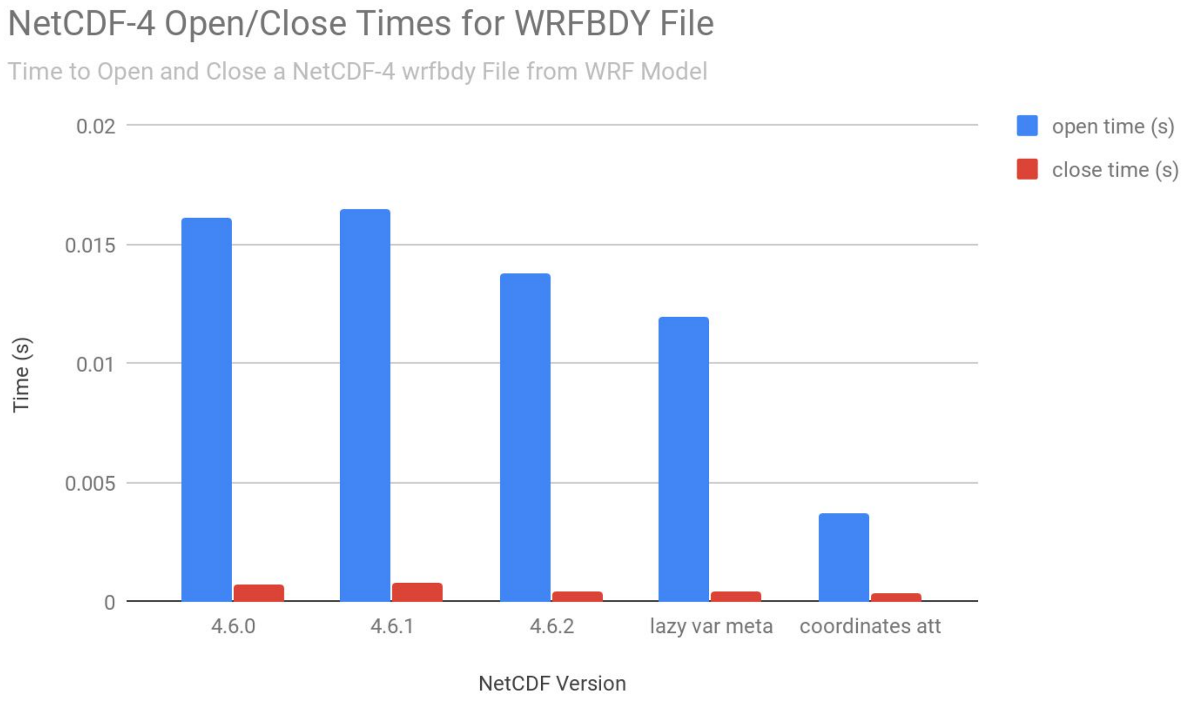

Figure 10: This chart shows the times to open and close a WRFBDY file by different netCDF versions. Version 4.6.2 introduced lazy attribute reads, and this is apparent from the improvement in performance. Using lazy read for some of the variable metadata further improves performance. Use of the hidden coordinates attribute to do dimension mapping adds a more substantial performance improvement. Tests were run on workstation mikado.

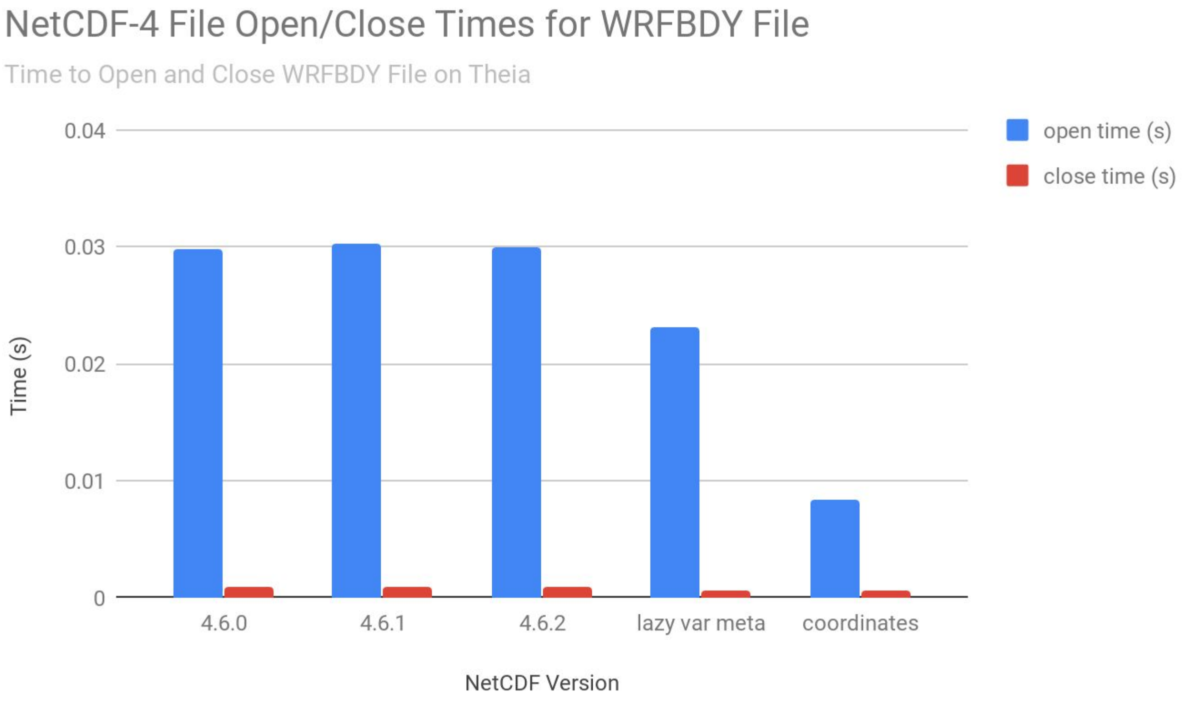

Figure 11: Times to open/close the WRFBDY file on theia. The use on lazy attribute reads starting in version 4.6.2 has only a small impact on the time to open the file. Doing lazy read of some variable metadata further improves performance. Using the hidden coordinates attribute yields a further significant improvement. Tests were run on theia.

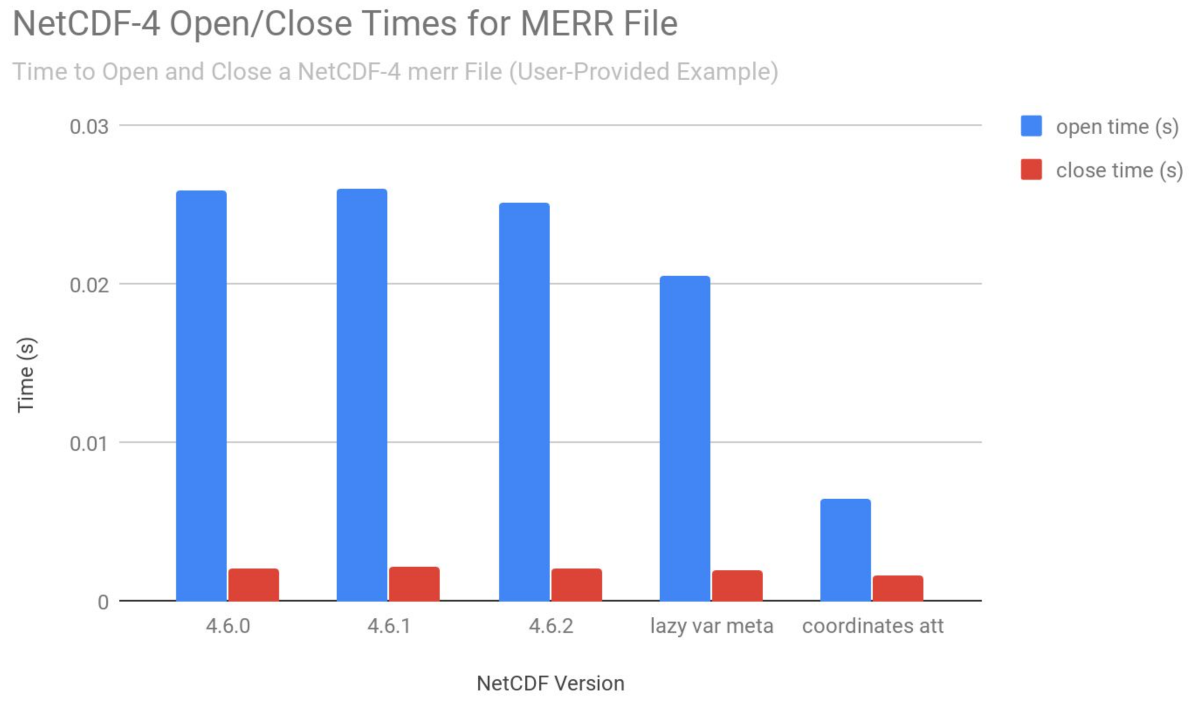

Figure 12: This chart shows the time to open and close the GMAO sample MERR data file for different versions of netCDF. Version 4.6.2 introduced lazy reads of attributes, which shows minor improvement. The lazy read of some variable metadata yields further improvement in performance opening the file. Using the hidden coordinates attribute to map dimension yields a greater improvement in performance. Tests were run on workstation mikado.

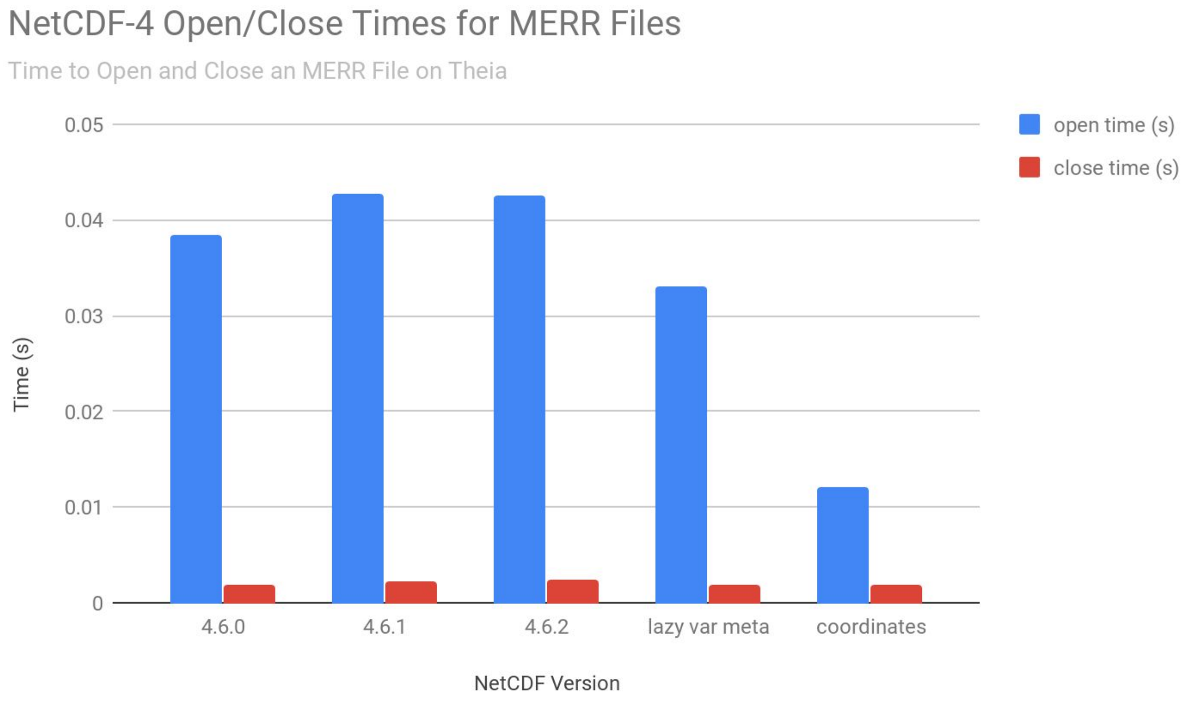

Figure 13: This chart shows the time to open and close the MERR sample file for various netCDF versions on theia. It is not clear why, in this case, 4.6.0 ran more quickly, but there are other users on theia when tests are being run, and these may interfere with results. The performance improvement on 4.6.2 is not evident in this chart, but the performance improvements of lazy variable metadata reads, and of the use of the hidden coordinates attribute, at clearly seen. Tests were run on theia.