esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Compatibility Between ESDM and NetCDF4

Luciana Pedro Julian Kunkel Benjamin Hodges

Work Package: Work Package 4 Exploitability

Date: April 17, 2020
History October 2019, Version 0.9.
April 2020, Integrated behaviour of the current EDSM development

Contents

1 Introduction

This report is covering the compatibility between ESDM and NetCDF4. It is a supplementary document and not a formal deliverable of the ESiWACE project. We will update this document over the course of the development.

The Earth System Data Middleware (ESDM) provides a high level of abstraction for earth system applications in the presence of storage heterogeneity. This novel middleware aims to include aspects of workflow management capabilities to enable intelligent storage management that not only optimises data locality and performance but also lifts data management to a new level. The architecture utilises scientific metadata to exploit a centric perspective of the data structure while retaining well-established end-user interfaces. For further information on ESDM, check the Git Repository on `https://github.com/ESiWACE/esdm`.

The design goals of the ESDM are:

- Relaxed access semantics, tailored to scientific data generation
- Site-specific (optimised) data layout schemes
- Ease of use and deployment particularly configuration
- Enable a configurable namespace based on scientific metadata

ESDM breaks with the concepts of portability, archivability, and shareability (to some extent) as the aim is to provide a site-specific optimal mapping.

1.1 NetCDF

NetCDF (Network Common Data Form) 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. The project homepage is hosted by the Unidata Program at the University Corporation for Atmospheric Research (UCAR). The format is an open standard. The project started in 1989 and is still actively supported by UCAR. The original NetCDF binary format (released in 1997, now known as ***NetCDF Classic Format****) is still widely used across the world and continues to be fully supported in all NetCDF releases.*

NetCDF is also a community standard for sharing scientific data. The Unidata Program Center supports and maintains NetCDF programming interfaces for C, C++, Java, and Fortran. Programming interfaces are also available for Python, IDL, MATLAB, R, Ruby, and Perl. Version 4.0 (released in 2008) allowed the use of the HDF5 data file format. Version 4.1 (2010) added support for C and Fortran client access to specified subsets of remote data via OPeNDAP. Version 4.3.0 (2012) added a CMake build system for Windows builds. Version 4.7.0 (2019) added support for reading Amazon S3 objects. Further releases are planned to improve performance, add features, and fix bugs. For further information on NetCDF, check

https://www.unidata.ucar.edu/software/netcdf/. [1]

Data in NetCDF format is: [2]

Self-Describing *A NetCDF file includes information about the data it contains.*

Portable *A NetCDF file can be accessed by computers with different ways of storing integers, characters, and floating-point numbers.*

Scalable *Small subsets of large datasets in various formats may be accessed efficiently through NetCDF interfaces, even from remote servers.*

Appendable *Data may be appended to a properly structured NetCDF file without copying the dataset or redefining its structure.*

Sharable *One writer and multiple readers may simultaneously access the same NetCDF file.*

Archivable *Access to all earlier forms of NetCDF data will be supported by current and future versions of the software.*

1.2 Outline

This report is organised as follows:

Chapter 2 presents how ESDM and NetCDF objects are related.

Chapter 3 describes the main NetCDF funcionalities and their coverage by ESDM.

Chapter 4 introduces the C tests provided by NetCDF and their status when running with ESDM.

Chapter 5 demonstrates the compatibility between the current version of ESDM and NetCDF Python.

Chapter 6 analyses the NetCDF Performance Benchmark Tool.

Chapter 7 explores the nccopy utility.

Chapter 8 shows how the automate of parts of software development is available using Jenkins.

Chapter 9 concludes this compatibility report.

[1] `https://en.wikipedia.org/wiki/NetCDF`
[2] `https://www.unidata.ucar.edu/software/netcdf/`

2 ESDM Implementation

2.1 Introduction

ESDM deeply integrates into NetCDF and this chapter describes how ESDM objects are related to NetCDF objects. For each NetCDF file, ESDM creates a container, and for each NetCDF variable, ESDM creates a dataset. Listing 2.1 shows the ESDM structures kept in main memory and Listing 2.2 introduces the ESDM structures kept in disk.

Listing 2.1: Structures kept in memory by ESDM

```
1  typedef struct {
2    int *dimidsp;
3    int fillmode; // remembers if fill mode is on or off, by default fill
       ↪ mode is
4                  // on, actually, ESDM with NetCDF always has a fill mode
       ↪ , uses
5                  // the defaults from NetCDF
6    esdm_dataset_t *dset;
7  } md_var_t;
8
9  typedef struct {
10   int count;
11   md_var_t **var;
12 } md_vars_t;
13
14 typedef struct {
15   int count;
16   uint64_t *size;
17   char **name;
18 } nc_dim_tbl_t;
19
20 typedef struct {
21   int ncid;
22   int fillmode; // remembers if fill mode is on or off, by default fill
       ↪ mode is
23                 // on, actually, ESDM with NetCDF always has a fill mode
       ↪ , uses
24                 // the defaults from NetCDF
25   esdm_container_t *c;
26   // Some attributes provide information about the dataset as a whole
       ↪ and are
27   // called global attributes. These are identified by the attribute
       ↪ name
28   // together with a blank variable name (in CDL) or a special null "
       ↪ global
29   // variable" ID (in C or Fortran).
30   nc_dim_tbl_t dimt;
31   md_vars_t vars;
32   int parallel_mode;
33 #ifdef ESDM_PARALLEL
34   MPI_Comm comm;
35 #endif
36 } nc_esdm_t;
```

Listing 2.2: ESDM Entities

```
1
2  struct esdm_datasets_t {
3    esdm_dataset_t ** dset;
4    int count;
5    int buff_size;
6  };
7
8  struct esdm_container_t {
9    char *name;
10   smd_attr_t *attr;
11   esdm_datasets_t dsets;
12
13   int refcount;
14   esdm_data_status_e status;
15   int mode_flags; // set via esdm_mode_flags_e
16 };
17
18 typedef struct esdmI_hypercubeNeighbourManager_t
     ↪ esdmI_hypercubeNeighbourManager_t;
19 struct esdm_fragments_t {
20   esdm_fragment_t ** frag;
21   esdmI_hypercubeNeighbourManager_t* neighbourManager;
22   int count;
23   int buff_size;
24 };
25
26 typedef struct esdm_fragments_t esdm_fragments_t;
27
28 struct esdm_dataset_t {
29   char *name;
30   char *id;
31   char **dims_dset_id; // array of variable names != NULL if set
32   esdm_container_t *container;
33   esdm_dataspace_t *dataspace;
34   smd_attr_t *fill_value; // use for read of not-written data, if set
35   smd_attr_t *attr;
36   int64_t *actual_size; // used for unlimited dimensions
37   esdm_fragments_t fragments;
38   int refcount;
39   esdm_data_status_e status;
40   int mode_flags; // set via esdm_mode_flags_e
41 };
42
43 struct esdm_fragment_t {
44   char * id;
45   esdm_dataset_t *dataset;
46   esdm_dataspace_t *dataspace;
47   esdm_backend_t *backend;
48   void * backend_md; // backend-specific metadata if set
49   void *buf;
50   size_t elements;
51   size_t bytes;
52   //int direct_io;
53   esdm_data_status_e status;
54 };
55
56 struct esdm_dataspace_t {
57   esdm_type_t type;
58   int64_t dims;
59   int64_t *size;
60   int64_t *offset;
61   int64_t *stride;  //may be NULL, in this case contiguous storage in C
     ↪ order is assumed
62 };
```

To be able to interact with the NetCDF C code, we introduce the file `esdm_dispatch.c`. This file contains the functions that will be used by ESDM that will be called by the dispatch table when NetCDF runs. For each `nc_function`, we have an `ESDM_function` with the same parameters, providing the same outcome, but using this new middleware. Some of the ESDM functions use the flag ESDM_PARALLEL to check if the code is being parallelised and call the respective ESDM parallel functions if that is the case.

2.1.1 ESDM Open

ESDM is first called when `nc_open` is called, which triggers the respective function `ESDM_open` and loads the data from the NetCDF file to memory. Then, the following functions are called to create a representation of the NetCDF file according to the ESDM entities.

esdm_container_open Open the container that contains the NetCDF file.

esdm_container_dataset_count Check the number of datasets in the container.

esdm_container_dataset_from_array For each dataset, load it into memory.

esdm_dataset_ref For each dataset, check whether the dataset was already loaded into memory. If that is not the case, the dataset's metadata is loaded into memory before incrementing the reference count.

esdm_dataset_get_dataspace For each dataset, create a copy of the internal dataspace object.

esdm_dataspace_get_dims For each dataset, read the number of dimensions from the dataspace.

esdm_dataset_get_name_dims For each dimension, read its name.

esdm_dataset_get_size For each dimension, read its size.

add_to_dims_tbl For each dataset and each dimension, insert the dimension into the dimension table, if not already there. The dimension table is constructed in memory to allocate the number of dimensions, sizes, and names.

insert_md For each dataset, insert the information about its metadata.

esdm_container_get_attributes Open the global attributes of the NetCDF file.

add_to_dims_tbl If there exists a dimension that was not previously used in any dataset, insert this dimension into the dimension table.

2.1.2 ESDM Close

When `nc_close` is called, the respective function `ESDM_close` is triggerred to close the NetCDF file. T following functions are called to close the NetCDF file according to the ESDM entities.

ESDM_nc_get_esdm_struct Loads the data from the NetCDF file to memory.

ncesdm_container_commit Insert the information from the dimension table as a global attribute and make the information inside the container persistent.

esdm_container_dataset_count Check the number of datasets in the container.

esdm_container_dataset_from_array For each dataset, load it into memory.

esdm_dataset_close For each dataset, remove its information from storage.

esdm_container_close Remove the container from storage.

3 NetCDF and ESDM Functionalities

This chapter compares the NetCDF functionalities with the current version of ESDM.

3.1 Error Handling

Each netCDF function in the C, Fortran 77, and Fortran 90 APIs returns 0 on success, in the tradition of C. When programming with netCDF in these languages, always check return values of every netCDF API call. The return code can be looked up in netcdf.h (for C programmers) or netcdf.inc (for Fortran programmers), or you can use the strerror function to print out an error message.

In general, if a function returns an error code, you can assume it did not do what you hoped it would. NetCDF functions return a non-zero status codes on error. If the returned status value indicates an error, you may handle it in any way desired, from printing an associated error message and exiting to ignoring the error indication and proceeding (not recommended!).

Occasionally, low-level I/O errors may occur in a layer below the netCDF library. For example, if a write operation causes you to exceed disk quotas or to attempt to write to a device that is no longer available, you may get an error from a layer below the netCDF library, but the resulting write error will still be reflected in the returned status value. [1]

3.1.1 ESDM

NetCDF has an extensive classification for the possible errors that might happen. ESDM does not share this classification, and it is something that their developers are not considering to include in the final version.

This decision does not affect the performance of ESDM, but it is critical when NetCDF tests are evaluated. NetCDF tests introduce invalid conditions and, because ESDM does not produce the expected error, the test fails. To provide a fair comparison with ESDM, the code in the NetCDF tests that considers invalid parameters as input was removed.

3.2 NetCDF Data Models

*There are two netCDF data models, the **Classic Model** (Section 3.2.1) and the **Common Data Model** (Section 3.2.2) (also called the netCDF-4 data model or enhanced model). The Classic Model is the simpler of the two, and is used for all data stored in classic CDF-1 format, 64-bit offset CDF-2 format, 64-bit data CDF-5 format, or netCDF-4 classic model format. The Common Data Model (sometimes also referred to as the netCDF-4 data model)*

[1] Adapted from `https://www.unidata.ucar.edu/software/netcdf/docs/group__error.html`

is an extension of the Classic Model that adds more powerful forms of data representation and data types at the expense of some additional complexity.

Although data represented with the Classic Model can also be represented using the Common Data Model, datasets that use Common Data Model features, such as user-defined data types, cannot be represented with the Classic Model. Use of the Common Data Model requires storage in the netCDF-4 format. [2]

3.2.1 Classic Data Model

*The **Classic Data Model** consists of variables, dimensions, and attributes. This way of thinking about data was introduced with the very first NetCDF release and is still the core of all NetCDF files.*

Variables *N-dimensional arrays of data. Variables in NetCDF files can be one of six types (char, byte, short, int, float, double).*

Dimensions *describe the axes of the data arrays. A dimension has a name and a length. An unlimited dimension has a length that can be expanded at any time, as more data are written to it. NetCDF files can contain at most one unlimited dimension.*

Attributes *annotate variables or files with small notes or supplementary metadata. Attributes are always scalar values or 1D arrays, which can be associated with either a variable or the file as a whole. Although there is no enforced limit, the user is expected to keep attributes small.*

3.2.2 Common Data Model

*With NetCDF-4, the NetCDF data model has been extended, in a backwards-compatible way. The new data model, which is known as the **Common Data Model**, is part of an effort here at Unidata to find a common engineering language for the development of scientific data solutions. It contains the variables, dimensions, and attributes of the classic data model, but adds:*

Groups *A way of hierarchically organising data, similar to directories in a Unix file system.*

User-defined Types *The user can now define compound types (like C structures), enumeration types, variable-length arrays, and opaque types.*

These features may only be used when working with a NetCDF-4/HDF5 file. Files created in classic formats cannot support groups or user-defined types. [3]

3.2.3 ESDM

NetCDF includes tests with both Classic and Common Data Models. While the ESDM data model basically supports all features of the Classic Model, it does not support the setting of the mode.

For the Common Data Model, ESDM does not support groups and user-defined types.

[2] Adapted from `https://www.unidata.ucar.edu/software/netcdf/docs/faq.html`
[3] Adapted from `https://www.unidata.ucar.edu/software/netcdf/docs/netcdf_data_model.html`

3.3 Data Modes

There are two modes associated with accessing a NetCDF file [4]*:*

Define Mode *In define mode, dimensions, variables, and new attributes can be created, but variable data cannot be read or written.*

Data Mode *In data mode, data can be read or written, and attributes can be changed, but new dimensions, variables, and attributes cannot be created.*

3.3.1 ESDM

The current version of ESDM does not have restrictions regarding the modes. Once the file is open, the user can do any modifications s/he wants. Tables 3.1 and 3.2 compare the options for creating and opening a file using NetCDF and ESDM. ESDM maps the NetCDF flag into an internal flag, if the mode is supported.

FLAG	NetCDF Support	ESDM Support
NC_CLOBBER	Overwrite existing file	ESDM_CLOBBER
NC_NOCLOBBER	Do not overwrite existing file	ESDM_NOCLOBBER
NC_SHARE	Limit write caching - netcdf classic files only	NOT SUPPORTED
NC_64BIT_OFFSET	Create 64-bit offset file	NOT SUPPORTED
NC_64BIT_DATA	Create CDF-5 file (alias NC_CDF5)	NOT SUPPORTED
NC_NETCDF4	Create NetCDF-4/HDF5 file	NOT SUPPORTED
NC_CLASSIC_MODEL	Enforce NetCDF classic mode on NetCDF-4/HDF5 files	NOT SUPPORTED
NC_DISKLESS	Store data in memory	NOT SUPPORTED
NC_PERSIST	Force the NC_DISKLESS data from memory to a file	NOT SUPPORTED

Table 3.1: Modes – Creating a file.

FLAG	NetCDF Support	ESDM Support
NC_NOWRITE	Open the dataset with read-only access	ESDM_MODE_FLAG_READ
NC_WRITE	Open the dataset with read-write access	ESDM_MODE_FLAG_WRITE
NC_SHARE	Share updates, limit caching	NOT SUPPORTED
NC_DISKLESS	Store data in memory	NOT SUPPORTED
NC_PERSIST	Force the NC_DISKLESS data from memory to a file	NOT SUPPORTED

Table 3.2: Modes – Opening a file.

[4] Adapted from `https://northstar-www.dartmouth.edu/doc/idl/html_6.2/NetCDF_Data_Modes.html`

3.4 Data Types

Data in a NetCDF file may be one of the **atomic types** (Section 3.4.1), or may be a **user-defined types** (Section 3.4.3).

3.4.1 Atomic Types

Atomic types are those which can not be further subdivided. All six classic model types (BYTE, CHAR, SHORT, INT, FLOAT, DOUBLE) are atomic, and fully supported in netCDF-4.

The following new atomic types have been added in netCDF-4: UBYTE, USHORT, UINT, INT64, UINT64, STRING. The string type will efficiently store arrays of variable length strings. [5]

Table 3.2 shows the definition for the atomic types supported by the NetCDF interface. [6]

Type	Description
NC_BYTE	8-bit signed integer
NC_UBYTE	8-bit unsigned integer
NC_CHAR	8-bit character
NC_SHORT	16-bit signed integer
NC_USHORT	16-bit unsigned integer
NC_INT (or NC_LONG)	32-bit signed integer
NC_UINT	32-bit unsigned integer
NC_INT64	64-bit signed integer
NC_UINT64	64-bit unsigned integer
NC_FLOAT	32-bit floating-point
NC_DOUBLE	64-bit floating-point
NC_STRING	variable length character string

Table 3.3: NetCDF4 atomic types.

3.4.2 ESDM

ESDM supports all NetCDF4 atomic types but NC_STRING. It is worth mentioning that, althought ESDM has a type SMD_DTYPE_STRING, this type does not work as the NC_STRING type.

Table 3.4 summarizes the available NetCDF data types and the corresponding support from ESDM.

[5] Adapted from `https://www.unidata.ucar.edu/software/netcdf/workshops/2007/nc4features/AtomicTypes.html`

[6] This table is no longer available on Unidata website. To reconstruct it, the information is in `https://www.unidata.ucar.edu/software/netcdf/docs/netcdf_8h.html`, in which the definition is now made using bytes, instead of bits. For example, NC_USHORT is now defined as *unsigned 2-byte int*.

NetCDF Type	Definition	ESDM Type	ESDM Representation
NC_NAT	NAT = Not A Type (c.f. NaN)	SMD_TYPE_AS_EXPECTED	as expected
NC_BYTE	signed 1 byte integer	SMD_DTYPE_INT8	int8_t
NC_CHAR	ISO/ASCII character	SMD_DTYPE_CHAR	char
NC_SHORT	signed 2 byte integer	SMD_DTYPE_INT16	int16_t
NC_INT	signed 4 byte integer	SMD_DTYPE_INT32	int32_t
NC_LONG	deprecated, but required for backward compatibility	SMD_DTYPE_INT32	int32_t
NC_FLOAT	single precision floating-point number	SMD_DTYPE_FLOAT	32 bits
NC_DOUBLE	double precision floating-point number	SMD_DTYPE_DOUBLE	64 bits
NC_UBYTE	unsigned 1 byte int	SMD_DTYPE_UINT8	uint8_t
NC_USHORT	unsigned 2-byte int	SMD_DTYPE_UINT16	uint16_t
NC_UINT	unsigned 4-byte int	SMD_DTYPE_UINT32	uint32_t
NC_INT64	signed 8-byte int	SMD_DTYPE_INT64	int64_t
NC_UINT64	unsigned 8-byte int	SMD_DTYPE_UINT64	uint64_t
NC_STRING	variable length character string	NOT SUPPORTED YET	NOT SUPPORTED YET

Table 3.4: Data Types Compatibility

3.4.3 User-Defined Types

User defined types allow for more complex data structures. NetCDF-4 has added support for four different user defined data types.

Compound Type *Like a C struct, a compound type is a collection of types, including other user defined types, in one package.*

Opaque Type *Used to store ragged arrays.*

Variable Length Array Type *This type has only a size per element, and no other type information.*

Enum Type *Like an enumeration in C, this type lets you assign text values to integer values, and store the integer values.*

Users may construct user defined type with the various `nc_def_*` *functions described in this section. They may learn about user defined types by using the* `nc_inq_` *functions defined in this section.* [7]

3.4.4 ESDM

The current version of ESDM does not support user-defined data types, but the developers intend to support this feature in the final version.

[7] Adapted from `https://www.unidata.ucar.edu/software/netcdf/docs/group__user__types.html`

NetCDF Type	Definition	ESDM Support
NC_VLEN	used internally for vlen types	NOT SUPPORTED YET
NC_OPAQUE	used internally for opaque types	NOT SUPPORTED YET
NC_COMPOUND	used internally for compound types	NOT SUPPORTED YET
NC_ENUM	used internally for enum types	NOT SUPPORTED YET

Table 3.5: User-Defined Types

3.5 Compression

The NetCDF-4 libraries inherit the capability for data compression from the HDF5 storage layer underneath the NetCDF-4 interface. Linking a program that uses NetCDF to a NetCDF-4 library allows the program to read compressed data without changing a single line of the program source code. Writing NetCDF compressed data only requires a few extra statements. And the nccopy utility program supports converting classic NetCDF format data to or from compressed data without any programming. [8]

3.5.1 ESDM

ESDM does not support compression yet. Because of that, all functions and tests related to chunking, deflate, and fletcher will not work when using ESDM.

We will integrate a compression library in the future and support quantification of error tolerance levels for different variables. The Scientific Compression Library (SCIL) it not yet integrated with the current version of ESDM, but it can be found in the following Git Repository:

`https://github.com/JulianKunkel/scil/`

3.6 Endianness

The endianness is defined as the order of bytes in multi-byte numbers: numbers encoded in big-endian have their most significant bytes written first, whereas numbers encoded in little-endian have their least significant bytes first. Little-endian is the native endianness of the IA32 architecture and its derivatives, while big-endian is native to SPARC and PowerPC, among others. The native-endianness procedure returns the native endianness of the machine it runs on. [9]

*NetCDF-4 uses **reader-makes-right** approach, in which:*

- *Writer always uses native representations, so no conversion is necessary on writing*
- *Reader is responsible for detecting what representation is used and applying a conversion, if necessary, to reader's native representation*
- *No conversion is necessary if reader and writer use same representation*

[8] Adapted from `https://www.unidata.ucar.edu/blogs/developer/entry/netcdf_compression`
[9] Adapted from `https://www.gnu.org/software/guile/manual/html_node/Bytevector-Endianness.html`

NetCDF-4 also lets writer control endianness explicitly, if necessary. [10]

3.6.1 ESDM

ESDM only supports native-endianness of the machine it runs on. The developers believe that the native-endianness of the machine is enough for demonstrating the benefits of using ESDM to improve efficiency in the system.

The rationale behind this design choice is that ESDM will be deployed in data centres and will be used to store data optimally in the data centre partitioned across available storage solutions. It is not intended to be stored in a portable fashion. Therefore, data can be imported/exported between, e.g., a NetCDF format and the ESDM native format.

3.7 Groups

NetCDF-4 files can store attributes, variables, and dimensions in hierarchical groups. This allows the user to create a structure much like a Unix file system.

In NetCDF, each group gets an ncid. Opening or creating a file returns the ncid for the root group (which is named "/").

Dimensions are scoped such that they are visible to all child groups. For example, you can define a dimension in the root group, and use its dimension id when defining a variable in a sub-group.

Attributes defined as NC_GLOBAL apply to the group, not the entire file.

The degenerate case, in which only the root group is used, corresponds exactly with the classic data model, before groups were introduced. [11]

3.7.1 ESDM

In general, ESDM does not support groups from NetCDF. When only the root group is used, ESDM can work adequately and assumes the group and the file are the same entity.

The ability to work with groups is a functionality that ESDM developers may implement depending on future requirements.

3.8 Fill Values

Sometimes there are missing values in the data, and some value is needed to represent them. For example, what value do you put in a sea-surface temperature variable for points overland?

In NetCDF, you can create an attribute for the variable (and of the same type as the variable) called _FillValue that contains a value that you have used for missing data. Applications

[10] Reference: `https://www.unidata.ucar.edu/software/netcdf/workshops/2008/netcdf4/ReaderMakesRight.html`

[11] Adapted from `https://www.unidata.ucar.edu/software/netcdf/docs/groups.html`

that read the data file can use this to know how to represent these values. [12]

3.8.1 ESDM

ESDM supports fill values. There are some specific details in the implementation of fill values inside ESDM that is worth noticing.

3.9 Type Conversion

With the new interface, users need not be aware of the external type of numeric variables, since automatic conversion to or from any desired numeric type is now available. You can use this feature to simplify code, by making it independent of external types. The elimination of void pointers provides detection of type errors at compile time that could not be detected with the previous interface. Programs may be made more robust with the new interface, because they need not be changed to accommodate a change to the external type of a variable.*

If conversion to or from an external numeric type is necessary, it is handled by the library. This automatic conversion and separation of external data representation from internal data types will become even more important in netCDF version 4, when new external types will be added for packed data for which there is no natural corresponding internal type (for example, arrays of 11-bit values).

Converting from one numeric type to another may result in an error if the target type is not capable of representing the converted value. For example, a short may not be able to hold data stored externally as an NC_FLOAT (an IEEE floating-point number). When accessing an array of values, an NC_ERANGE error is returned if one or more values are out of the range of representable values, but other values are converted properly.

Note that mere loss of precision in type conversion does not return an error. Thus, if you read double precision values into a long, for example, no error results unless the magnitude of the double precision value exceeds the representable range of longs on your platform. Similarly, if you read a large integer into a float incapable of representing all the bits of the integer in its mantissa, this loss of precision will not result in an error. If you want to avoid such precision loss, check the external types of the variables you access to make sure you use an internal type that has a compatible precision.

The new interface distinguishes arrays of characters intended to represent text strings from arrays of 8-bit bytes intended to represent small integers. The interface supports the internal types text, uchar, and schar, intended for text strings, unsigned byte values, and signed byte values. [13]

3.9.1 ESDM

ESDM supports most of the data conversions but may return a slightly different error.

ESDM deals with type conversion the same way as NetCDF. However, ESDM only accepts conversions for attributes, and not for variables.

[12] Adapted from `https://www.unidata.ucar.edu/software/netcdf/docs/fill_values.html`
[13] Adapted from `https://www.unidata.ucar.edu/software/netcdf/release-notes-3.3.html`

3.10 HDF5 Format

NetCDF-4 allows some interoperability with HDF5. The HDF5 files produced by netCDF-4 are perfectly respectable HDF5 files, and can be read by any HDF5 application.

NetCDF-4 relies on several new features of HDF5, including dimension scales. The HDF5 dimension scales feature adds a bunch of attributes to the HDF5 file to keep track of the dimension information. It is not just wrong, but wrong-headed, to modify these attributes except with the HDF5 dimension scale API. If you do so, then you will deserve what you get, which will be a mess.

Additionally, netCDF stores some extra information for dimensions without dimension scale information. (That is, a dimension without an associated coordinate variable). So HDF5 users should not write data to a netCDF-4 file which extends any unlimited dimension, or change any of the extra attributes used by netCDF to track dimension information.

Also there are some types allowed in HDF5, but not allowed in netCDF-4 (for example the time type). Using any such type in a netCDF-4 file will cause the file to become unreadable to netCDF-4. So do not do it.

NetCDF-4 ignores all HDF5 references. Can not make head nor tail of them. Also netCDF-4 assumes a strictly hierarchical group structure. No looping, you weirdo!

Attributes can be added (they must be one of the netCDF-4 types), modified, or even deleted, in HDF5. [14]

3.10.1 ESDM

ESDM does not support HDF5 format.

[14] Adapted from `https://www.unidata.ucar.edu/software/netcdf/docs/interoperability_hdf5.html`

4 Tests in C

4.1 Introduction

The directory **nc_test4** has originally 105 tests. The tests were classified according to their output using NetCDF. Basically, we expect that all benchmarks run with NetCDF successfully but some tests turned out to be non-functional. There are four categories for the tests:

Not Tested It means the test was not tested. This category usually represents tests that are too long to run or tests that demand a large amount of memory or processing.

Not Building It means it was not possible to compile the test.

Not Running It means the test compiles, but there is something wrong when the test runs and the expected output is not produced.

Working It means the test compiles, runs and produces the expected output.

Tables 4.1 shows the numbers for the final classification of the 105 tests provided by the NetCDF team.

Total Files	Not Building	Not Running	Working
104	11	2	91

Table 4.1: List of nc_test4 files

Tables 4.2, 4.3 and 4.4 presents the tests and their individual classification.

Filename	Not Building	Not Running	Working
bigmeta.c	✓		
bm_file.c	✓		
bm_many_atts.c	✓		
bm_many_objs.c	✓		
bm_netcdf4_recs.c	✓		
cdm_sea_soundings.c			✓
h5testszip.c			✓
openbigmeta.c		✓	
ref_bzip2.c	✓		✓
renamegroup.c			✓
test_filter.c			✓
test_filter_misc.c			✓
test_szip.c			✓
tst_ar4.c	✓		
tst_ar4_3d.c	✓		
tst_ar4_4d.c		✓	
tst_atts-simple.c			✓
tst_atts.c			✓
tst_atts1.c			✓
tst_atts2.c			✓
tst_atts3.c			✓
tst_attsperf.c			✓
tst_atts_mod.c			✓
tst_atts_string_rewrite.c			✓
tst_bug324.c			✓
tst_camrun.c			✓
tst_chunks.c			✓
tst_chunks2.c			✓
tst_chunks3.c			✓
tst_compounds.c			✓
tst_compounds2.c			✓
tst_compounds3.c			✓
tst_converts.c			✓
tst_converts2.c			✓
tst_coords.c			✓
tst_coords2.c			✓
tst_coords3.c			✓
tst_create_files.c			✓
tst_dims.c			✓
tst_dims2.c			✓
tst_dims3.c			✓
tst_elatefill.c			✓
tst_empty_vlen_unlim.c			✓
tst_endian_fill.c			✓
tst_enums.c			✓

Table 4.2: List of nc_test4 files – Part I

Filename	Not Building	Not Running	Working
tst_files.c			✓
tst_files2.c	✓		
tst_files3.c			✓
tst_files4.c			✓
tst_files5.c			✓
tst_files6.c			✓
tst_fill_attr_vanish.c			✓
tst_fillbug.c			✓
tst_fills.c			✓
tst_fills2.c			✓
tst_filterparser.c			✓
tst_grps.c			✓
tst_grps2.c			✓
tst_h5_endians.c			✓
tst_hdf5_file_compat.c			✓
tst_h_many_atts.c	✓		
tst_h_refs.c			✓
tst_h_scalar.c			✓
tst_h_strbug.c			✓
tst_interops.c			✓
tst_interops4.c			✓
tst_interops5.c			✓
tst_interops6.c			✓
tst_knmi.c	✓		
tst_large.c			✓
tst_large2.c			✓
tst_mem.c			✓
tst_mode.c			✓
tst_mpi_parallel.c			✓
tst_nc4perf.c			✓
tst_opaques.c			✓
tst_parallel.c			✓
tst_parallel3.c			✓
tst_parallel4.c			✓
tst_parallel5.c			✓
tst_put_vars.c			✓
tst_put_vars_two_unlim_dim.c			✓
tst_rehash.c			✓
tst_rename.c			✓
tst_rename2.c			✓
tst_simplerw_coll_r.c			✓
tst_strings.c			✓
tst_strings2.c			✓
tst_sync.c			✓
tst_types.c			✓
tst_udf.c			✓

Table 4.3: List of nc_test4 files – Part II

Filename	Not Building	Not Running	Working
tst_unlim_vars.c			✓
tst_utf8.c			✓
tst_utils.c	✓		
tst_v2.c			✓
tst_varms.c			✓
tst_vars.c			✓
tst_vars2.c			✓
tst_vars3.c			✓
tst_vars4.c			✓
tst_vl.c			✓
tst_xplatform.c			✓
tst_xplatform2.c			✓
t_type.c			✓

Table 4.4: List of nc_test4 files – Part III

Table 4.5 introduces the reasons the selected tests are not building.

Filename	Comment
bigmeta.c	fatal error: config.h: No such file or directory
bm_file.c	fatal error: nc_tests.h: No such file or directory
bm_many_atts.c	fatal error: config.h: No such file or directory
bm_many_objs.c	fatal error: config.h: No such file or directory
bm_netcdf4_recs.c	fatal error: config.h: No such file or directory
tst_ar4.c	undefined reference to nc4_timeval_subtract
tst_ar4_3d.c	undefined reference to nc4_timeval_subtract
tst_files2.c	undefined reference to nc4_timeval_subtract
tst_h_many_atts.c	undefined reference to nc4_timeval_subtract
tst_knmi.c	undefined reference to nc4_timeval_subtract
tst_utils.c	undefined reference to main

Table 4.5: Tests that are not building

Table 4.6 introduces the reasons the selected tests are not running.

Filename	Comment
openbigmeta.c	Missing file bigmeta.nc
tst_ar4_4d.c	Missing NetCDF file

Table 4.6: Tests that are not running

4.2 Current Status

According to the classification presented in Tables 4.2, 4.3 and 4.4, the 91 **Working** tests were used to demonstrate ESDM functionalities. There are now three categories in which each test was classified according to their status:

Success It means the original test is successful when using ESDM.

Partial Success It means the original test is successful when using ESDM, but some parts of the code of the test had to be commented out because ESDM does not support that specific feature.

Failure It means the original test is not successful when using ESDM. Some of the missing features should be available in a medium-term and others are not expected to work with ESDM at all.

Table 4.7 summarises the results and Tables 4.8 and 4.9 presents the current status of each test.

Total Files	Success	Partial Success	Failure
91	27	25	39

Table 4.7: Current status of all tests

Filename	Success	Partial Success	Failure
cdm_sea_soundings.c			✓
h5testszip.c	✓		
ref_bzip2.c	✓		
renamegroup			✓
test_filter.c			✓
test_filter_misc.c			✓
test_szip.c	✓		
tst_atts-simple.c	✓		
tst_atts.c		✓	
tst_atts1.c		✓	
tst_atts2.c		✓	
tst_atts3.c	✓		
tst_attsperf.c	✓		
tst_atts_mod.c			✓
tst_atts_string_rewrite.c	✓		
tst_bug324.c		✓	
tst_camrun.c	✓		
tst_chunks.c			✓
tst_chunks2.c		✓	
tst_chunks3.c	✓		
tst_compounds.c			✓
tst_compounds2.c			✓
tst_compounds3.c			✓
tst_converts.c			✓
tst_converts2.c			✓
tst_coords.c		✓	
tst_coords2.c		✓	
tst_coords3.c		✓	
tst_create_files.c	✓		
tst_dims.c		✓	
tst_dims2.c		✓	
tst_dims3.c			✓
tst_elatefill.c			✓
tst_empty_vlen_unlim.c			✓
tst_endian_fill.c		✓	
tst_enums.c			✓
tst_files.c		✓	
tst_files3.c		✓	
tst_files4.c			✓
tst_files5.c	✓		
tst_files6.c		✓	
tst_fill_attr_vanish.c		✓	
tst_fillbug.c		✓	
tst_fills.c		✓	
tst_fills2.c			✓

Table 4.8: Current status of each individual test – Table I

Filename	Success	Partial Success	Failure
tst_filterparser.c	✓		
tst_grps.c			✓
tst_grps2.c			✓
tst_h5_endians.c			✓
tst_hdf5_file_compat.c			✓
tst_h_refs.c			✓
tst_h_scalar.c			✓
tst_h_strbug.c			✓
tst_interops.c			✓
tst_interops4.c			✓
tst_interops5.c			✓
tst_interops6.c			✓
tst_large.c	✓		
tst_large2.c	✓		
tst_mem.c	✓		
tst_mode.c		✓	
tst_mpi_parallel.c	✓		
tst_nc4perf.c			✓
tst_opaques.c			✓
tst_parallel.c	✓		
tst_parallel3.c		✓	
tst_parallel4.c	✓		
tst_parallel5.c			✓
tst_put_vars.c	✓		
tst_tst_put_vars_two_unlim_dim.c	✓		
tst_rehash.c	✓		
tst_rename.c		✓	
tst_rename2.c	✓		
tst_simplerw_coll_r.c	✓		
tst_strings.c			✓
tst_strings2.c			✓
tst_sync.c		✓	
tst_types.c			✓
tst_udf.c			✓
tst_unlim_vars.c	✓		
tst_utf8.c			✓
tst_v2.c	✓		
tst_varms.c	✓		
tst_vars.c		✓	
tst_vars2.c		✓	
tst_vars3.c		✓	
tst_vars4.c		✓	
tst_vl.c			✓
tst_xplatform.c			✓
tst_xplatform2.c			✓
t_type.c	✓		

Table 4.9: Current status of each individual test – Table II

4.3 Specific Information For The Tests

This section describes the current behaviour of the tests provided by NetCDF in its documentation. For each test, we have four descriptors:

Description: The short description provided by the original test.

Output: The current output for the test using ESDM.

Comments: What was modified in the original test, if any.

Status: The current status for the test.

4.3.1 Test cdm_sea_soundings.c

Description: The cdm tests confirm compliance with the Common Data Model. This file creates some sample data structures to hold sea soundings.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/cdm_sea_soundings.c, line: 60

Comments: ESDM does not support user-defined datatypes from NetCDF.

Status: FAILURE.

4.3.2 Test h5testszip.c

Description: Example illustrates the use of SZIP compression in HDF5.

Output: ***PASS

Comments: No comments.

Status: SUCCESS.

4.3.3 Test ref_bzip2.c

Description: No description.

Output: No output.

Comments: No comments.

Status: SUCCESS.

4.3.4 Test renamegroup.c

Description: Utility to rename a group.

Output: No arguments:

usage: renamegroup <filename> <old group path name> <new name>

Comments: ESDM does not support groups from NetCDF.

Status: FAILURE.

4.3.5 Test test_filter.c

Description: Example program for write then read of a variable using bzip2 compression.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/test_filter.c, line: 303

Comments: ESDM does not support filters.

Status: FAILURE.

4.3.6 Test test_filter_misc.c

Description: No description.

Output: fail: line=152 id mismatch

Comments: ESDM does not support filters.

Status: FAILURE.

4.3.7 Test test_szip.c

Description: Example illustrates the use of SZIP compression in NetCDF5.

Output: *** Tests successful!

Comments: No comments.

Status: SUCCESS.

4.3.8 Test tst_atts-simple.c

Description: Test the netCDF-4 attribute code.

Output: *** Tests successful!

Comments: No comments.

Status: SUCCESS.

4.3.9 Test tst_atts.c

Description: Test the NetCDF-4 attribute code.

Output: *** Tests successful!

Comments: Remove lines 78-101. ESDM does not support different access mode.

Remove lines 115-118, 148, 180-186, 230-268, 279-286, 327-334, 340-342, 352-354, 364-366, 373. Code testing invalid parameters as input.

Remove lines 203-208. The test works with ESDM.

Remove lines 288-316. ESDM does not support reserved words.

Status: PARTIAL SUCCESS.

4.3.10 Test tst_atts1.c

Description: Test attributes.

Output: *** Tests successful!

Comments: Remove lines 191-196, 205-207, 209. Expected error code: NC_ERANGE.

Remove lines 274-619. ESDM does not support conversion for attributes.

Remove lines 703-742, 771-784. Code testing invalid parameters as input.

Status: PARTIAL SUCCESS.

4.3.11 Test tst_atts2.c

Description: Test copy of attributes.

Output: *** Tests successful!

Comments: Remove lines 22-83, 157-259. ESDM does not support user-defined datatypes from NetCDF.

Remove lines 101-105. Code testing invalid parameters as input.

Lines 108, 115-119. Not clear what is the problem. It should be working.

Status: PARTIAL SUCCESS.

4.3.12 Test tst_atts3.c

Description: This is a very simple example which writes a NetCDF file with Unicode names encoded with UTF-8. It is the NETCDF3 equivalent of tst_unicode.c.

Output: *** Tests successful!

Comments: No comments.

Status: SUCCESS.

4.3.13 Test tst_attsperf.c

Description: Test the NetCDF-4 attribute code.

Output: *** Tests successful!

Comments: Relatively long to run.

Status: SUCCESS.

4.3.14 Test tst_atts_mod.c

Description: Test the netCDF-4 attribute code.

Output: *** testing attribute renaming for a global attribute...Sorry! Unexpected result, ../../libsrcesdm_test/tst_atts_mod.c, line: 361

Comments: Remove lines 309-310, 357-358. Code testing invalid parameters as input.

Remove line 361. Renaming a global attribute. It should be working.

Status: FAILURE.

4.3.15 Test tst_atts_string_rewrite.c

Description: This test was provided by Jeff Whitaker as an example of a bug, specifically, a segfault when re-writing an NC_CHAR attribute as an NC_STRING attribute.

Output: *** Tests successful!

Comments: No comments.

Status: SUCCESS.

4.3.16 Test tst_bug324.c

Description: No description.

Output: *** Tests successful!

Comments: Remove line 71. ESDM does not support NC_FORMAT_CLASSIC

Status: PARTIAL SUCCESS.

4.3.17 Test tst_camrun.c

Description: This program writes a data file from the CAM model run.

Output: *** Tests successful!

Comments: No comments.

Status: SUCCESS.

4.3.18 Test tst_chunks.c

Description: Test netcdf-4 variables.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_chunks.c, line: 66

Comments: ESDM does not support compression.

Status: FAILURE.

4.3.19 Test tst_chunks2.c

Description: Test netcdf-4 chunking.

Output: *** Tests successful!

Comments: Remove lines 139, 175, 211-213, 375, 414, 453. ESDM does not support compression.

Status: PARTIAL SUCCESS.

4.3.20 Test tst_chunks3.c

Description: Runs benchmarks on different chunking sizes.

Output:

```
contiguous write   1 256 256                0.048 sec
chunked    write   1 256 256   32  32  32    0.047 sec     1 x faster
compressed write   1 256 256   32  32  32    0.044 sec   1.1 x faster

contiguous write 256   1 256                0.045 sec
chunked    write 256   1 256   32  32  32     0.05 sec   1.1 x slower
compressed write 256   1 256   32  32  32    0.059 sec   1.3 x slower

contiguous write 256 256   1                0.058 sec
chunked    write 256 256   1   32  32  32    0.051 sec   1.1 x faster
compressed write 256 256   1   32  32  32    0.061 sec     1 x slower

contiguous  read   1 256 256                 0.16 sec
chunked     read   1 256 256   32  32  32     0.13 sec   1.2 x faster
compressed  read   1 256 256   32  32  32    0.061 sec   2.6 x faster
```

```
contiguous  read 256   1 256                 0.13 sec
chunked     read 256   1 256   32  32  32    0.12 sec   1.1 x faster
compressed  read 256   1 256   32  32  32    0.11 sec   1.2 x faster

contiguous  read 256 256   1                 0.16 sec
chunked     read 256 256   1   32  32  32    0.17 sec     1 x slower
compressed  read 256 256   1   32  32  32    0.12 sec   1.4 x faster
```

Comments: ESDM does not support compression.

Status: SUCCESS.

4.3.21 Test tst_compounds.c

Description: Test netcdf-4 compound type feature.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_compounds.c, line: 53

Comments: ESDM does not support user-defined datatypes from NetCDF.

Status: FAILURE.

4.3.22 Test tst_compounds2.c

Description: Test netcdf-4 compound type feature.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_compounds2.c, line: 68

Comments: ESDM does not support user-defined datatypes from NetCDF.

Status: FAILURE.

4.3.23 Test tst_compounds3.c

Description: Test netcdf-4 compound type feature, even more.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_compounds3.c, line: 78

Comments: ESDM does not support user-defined datatypes from NetCDF.

Status: FAILURE.

4.3.24 Test tst_converts.c

Description: Test data conversions and fill value handling.

Output:

```
0 = 3
Sorry! Unexpected result, ../../libsrcesdm_test/tst_converts.c, line: 111
Sorry! Unexpected result, ../../libsrcesdm_test/tst_converts.c, line: 149
2 failures
*** Testing conversion in netCDF 64-bit offset files... 0 = 3
Sorry! Unexpected result, ../../libsrcesdm_test/tst_converts.c, line: 111
Sorry! Unexpected result, ../../libsrcesdm_test/tst_converts.c, line: 149
2 failures
*** Testing conversion in netCDF netCDF-4 files... 0 = 3
Sorry! Unexpected result, ../../libsrcesdm_test/tst_converts.c, line: 111
Sorry! Unexpected result, ../../libsrcesdm_test/tst_converts.c, line: 149
2 failures
*** Testing conversion in netCDF netCDF-4 classic model files... 0 = 3
Sorry! Unexpected result, ../../libsrcesdm_test/tst_converts.c, line: 109
Sorry! Unexpected result, ../../libsrcesdm_test/tst_converts.c, line: 147
2 failures
*** Testing conversion in netCDF CDF5 files... 0 = 3
Sorry! Unexpected result, ../../libsrcesdm_test/tst_converts.c, line: 109
Sorry! Unexpected result, ../../libsrcesdm_test/tst_converts.c, line: 147
2 failures
10 errors detected! Sorry!
```

Comments: Not clear what is the problem. It used to be working.

Status: FAILURE.

4.3.25 Test tst_converts2.c

Description: Test even more data conversions.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_converts2.c, line: 46

Comments: ESDM does not support conversion for variables.

Status: FAILURE.

4.3.26 Test tst_coords.c

Description: Test netcdf-4 coordinate variables and dimensions.

Output: *** Tests successful!

Comments: Remove lines 92-93, 488, 490, 492, 494, 537, 539, 785-786, 789, 791. ESDM does not keep the same order for the dimensions.

Remove line 582-602. ESDM does not support groups from NetCDF.

Remove lines 627, 633, 643-644. ESDM does not support two variables with the same name.

Remove line 780. Not clear what is the problem. It should be working.

Status: PARTIAL SUCCESS.

4.3.27 Test tst_coords2.c

Description: Test netcdf-4 coordinate variables and dimensions.

Output: *** Tests successful!

Comments: Remove lines 118-119, 125, 127, 129, 131, 133. ESDM does not keep the same order for the dimensions.

Remove lines 162-163, 167, 175-179. ESDM does not support groups from NetCDF.

Status: PARTIAL SUCCESS.

4.3.28 Test tst_coords3.c

Description: Test netcdf-4 coordinate variables and dimensions with an example from the CF conventions.

Output: *** Tests successful!

Comments: Remove lines 136, 138, 140, 144-145, 155-156, 177-178, 186-187, 195-196. ESDM does not keep the same order for the dimensions.

Status: PARTIAL SUCCESS.

4.3.29 Test tst_create_files.c

Description: This program creates a test file.

Output: *** Tests successful!

Comments: No comments.

Status: SUCCESS.

4.3.30 Test tst_dims.c

Description: Test netcdf-4 dimensions.

Output: *** Tests successful!

Comments: Remove lines 104-106, 116, 119-120, 136-138, 141, 150, 164, 217, 226, 261-263, 466. Code testing invalid parameters as input.

Remove lines 1102, 1165-1168. ESDM does not support conversion for variables.

Remove lines 1257-1258. ESDM does not keep the same order for the dimensions.

Remove lines 1315-1357. Creates a new file name (file_in) and tries to open it. ESDM does not understand the new name.

Status: PARTIAL SUCCESS.

4.3.31 Test tst_dims2.c

Description: Test netcdf-4 dimensions some more.

Output: *** Tests successful!

Comments: Remove lines 58-59. ESDM does not keep the same order for the dimensions.

Remove lines 325-332. ESDM does not support the conversion of vectors.

Remove lines 440-441. Code testing invalid parameters as input.

Status: PARTIAL SUCCESS.

4.3.32 Test tst_dims3.c

Description: Test netcdf-4 dimensions inheritance.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_dims3.c, line: 52

Comments: ESDM does not support groups from NetCDF.

Status: FAILURE.

4.3.33 Test tst_elatefill.c

Description: Test proper elatefill return when fill value is assigned outside of the initial define.

Output: line 41 expecting NC_ELATEFILL but got 0

Comments: Different error code. It is not clear if it works or not.

Status: FAILURE.

4.3.34 Test tst_empty_vlen_unlim.c

Description: This program exersizes HDF5 variable length array code.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_empty_vlen_unlim.c, line: 67

Comments: ESDM does not support user-defined datatypes from NetCDF.

Status: FAILURE.

4.3.35 Test tst_endian_fill.c

Description: Create a test file with fill values for a variable of specified endianness.

Output: *** Tests successful!

Comments: Remove lines 84-85. Expected error code: NC_ERANGE.

Remove lines 86-87. Fill value not set properly. It should be working.

Status: PARTIAL SUCCESS.

4.3.36 Test tst_enums.c

Description: Test netcdf-4 enum types.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_enums.c, line: 56

Comments: ESDM does not support user-defined datatypes from NetCDF.

Status: FAILURE.

4.3.37 Test tst_files.c

Description: Test netcdf-4 file code.

Output: *** Tests successful!

Comments: Remove line 189. Expected error code: NC_ENOTINDEFINE.

Remove lines 279, 303. ESDM does not support NC_FORMAT_CLASSIC.

Remove lines 285, 309. ESDM does not support NC_FORMAT_64BIT_OFFSET.

Remove lines 291, 479-481. ESDM does support with NC_FORMAT_NETCDF4.

Remove lines 410-417. ESDM does not work with specific formats and flags.

Remove lines 517-527. Not clear what is the problem. It might be: format model or expected error.

Remove line 534. Expected error code: NC_EPERM.

Remove lines 561-565, 601-605. Not clear what is the problem. It might be: conversion, classic model or expected error.

Remove line 569. Not clear what is the problem. It should be working because nc_inq works.

Remove line 597. Expected error code: NC_ERANGE. Not clear if this is the only problem.

Remove line 607. Not clear what is the problem. Probably, conversion.

Status: PARTIAL SUCCESS.

4.3.38 Test tst_files3.c

Description: This is a benchmark program which tests file writes with compressed data.

Output: *** Tests successful!

Comments: Remove lines 87-88, 215. ESDM does not support compression.

Status: PARTIAL SUCCESS.

4.3.39 Test tst_files4.c

Description: Test netcdf-4 file from user-reported error. This code based on an ncgen output.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_files4.c, line: 69

Comments: ESDM does not support groups from NetCDF.

Status: FAILURE.

4.3.40 Test tst_files5.c

Description: Test netcdf files a bit.

Output: *** Tests successful!

Comments: No comments.

Status: SUCCESS.

4.3.41 Test tst_files6.c

Description: Test netcdf files a bit.

Output: *** Tests successful!

Comments: Remove line 86. Expected error code: NC_EHDFERR.

Status: PARTIAL SUCCESS.

4.3.42 Test tst_fill_attr_vanish.c

Description: Based on tst_fillbug.c.

Output: *** Tests successful!

Comments: Remove lines 91-97. Expected error code: NC_ELATEFILL.

Status: PARTIAL SUCCESS.

4.3.43 Test tst_fillbug.c

Description: Test for a bug that Russ found testing fill values.

Output: *** Tests successful!

Comments: Remove line 83. ESDM does not keep the same order for the dimensions.

Status: PARTIAL SUCCESS.

4.3.44 Test tst_fills.c

Description: Create a test file with default fill values for variables of each type.

Output: *** Tests successful!

Comments: Remove lines 24-86. ESDM does not support datatype NC_STRING.

Status: PARTIAL SUCCESS.

4.3.45 Test tst_fills2.c

Description: Create a test file with default fill values for variables of each type.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_fills2.c, line: 48

Comments: ESDM does not support datatype NC_STRING.

Status: FAILURE.

4.3.46 Test tst_filterparser.c

Description: No description.

Output: SUCCESS!!

Comments: No comments.

Status: SUCCESS.

4.3.47 Test tst_grps.c

Description: Test netcdf-4 group code.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_grps.c, line: 51

Comments: ESDM does not support groups from NetCDF.

Status: FAILURE.

4.3.48 Test tst_grps2.c

Description: Test netcdf-4 group code some more.

Output: [ESDM NC] WARN ESDM_def_grp():1787. ESDM does not support groups from NetCDF.

Sorry! Unexpected result, ../../libsrcesdm_test/tst_grps2.c, line: 57

Comments: ESDM does not support groups from NetCDF.

Status: FAILURE.

4.3.49 Test tst_h5_endians.c

Description: No description.

Output:

```
** Checking test files.
*** tst_h5_endians.nc
Little-Endian Float... failed
Big-Endian Float... failed
Little-Endian Int... failed
Big-Endian Int... failed
Little-Endian Double... failed
Big-Endian Double... failed
** Failures Returned: [6]
```

Comments: ESDM does not support HDF5 format. ESDM only supports native endianness.

Status: FAILURE.

4.3.50 Test tst_hdf5_file_compat.c

Description: Tests library ability to open files generated by a netcdf instance linked against libhdf5 1.10.0.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_hdf5_file_compat.c, line: 42

Comments: ESDM does not support HDF5 format.

Status: FAILURE.

4.3.51 Test tst_h_refs.c

Description: This program tests fixes for reading NetCDF-4 files that contain datasets with reference data types. The NetCDF-4 library should ignore the datasets and attributes that have reference data types and allow the rest of the file to be accessed.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_h_refs.c, line: 85

Comments: ESDM does not support HDF5 format.

Status: FAILURE.

4.3.52 Test tst_h_scalar.c

Description: This program tests reading HDF5 files that contain scalar attributes and variables, of both string and numeric datatypes. The NetCDF-4 library should allow access to all of these.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_h_scalar.c, line: 272

Comments: ESDM does not support HDF5 format.

Status: FAILURE.

4.3.53 Test tst_h_strbug.c

Description: This program tests fixes for bugs reported with accessing fixed-length scalar string variables and variable-length scalar string attributes from HDF5 files through the NetCDF-4 API.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_h_strbug.c, line: 96

Comments: ESDM does not support HDF5 format.

Status: FAILURE.

4.3.54 Test tst_interops.c

Description: Test that HDF5 and NetCDF-4 can read and write the same file.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_interops.c, line: 126

Comments: ESDM does not support HDF5 format.

Status: FAILURE.

4.3.55 Test tst_interops4.c

Description: Test NetCDF-4 files with lots of attributes on big vs. little endian platforms.

Output: testing with file ref_tst_interops4.nc...

Sorry! Unexpected result, ../../libsrcesdm_test/tst_interops4.c, line: 154

Comments: It seems file ref_tst_interops4.nc should be generated by the test itself. Not clear what is the problem.

Status: FAILURE.

4.3.56 Test tst_interops5.c

Description: Test that HDF5 and NetCDF-4 can read and write the same file.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_interops5.c, line: 164

Comments: ESDM does not support HDF5 format.

Status: FAILURE.

4.3.57 Test tst_interops6.c

Description: Test that HDF5 and NetCDF-4 can read and write the same file.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_interops6.c, line: 159

Comments: ESDM does not support HDF5 format.

Status: FAILURE.

4.3.58 Test tst_large.c

Description: Test netcdf-4 large file fill values.

Output: *** Tests successful!

Comments: No comments.

Status: SUCCESS.

4.3.59 Test tst_large2.c

Description: Test large file problems reported by user.

Output: *** Tests successful!

Comments: Relatively long to run.

Status: SUCCESS.

4.3.60 Test tst_mem.c

Description: Test internal netcdf-4 file code.

Output: *** Tests successful!

Comments: No comments.

Status: SUCCESS.

4.3.61 Test tst_mode.c

Description: Test some illegal mode combinations.

Output: *** Tests successful!

Comments: Remove lines 27, 34. Expected error code: NC_EINVAL.

Status: PARTIAL SUCCESS.

4.3.62 Test tst_mpi_parallel.c

Description: This just exercises MPI file I/O to make sure everything's working properly. If this does not work, netcdf/HDF5 parallel I/O also won't work.

Output: *** Tests successful!

Comments: It has to be tested with more nodes.

Status: SUCCESS.

4.3.63 Test tst_nc4perf.c

Description: This program tests netcdf-4 parallel I/O. These tests are based on the needs of the NASA GMAO model, and are based on some test code from Dennis Nadeau.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_nc4perf.c, line: 97

Sorry! Unexpected result, ../../libsrcesdm_test/tst_nc4perf.c, line: 294

Comments: Problem with function nc_put_vara_float. It is not expected to work now. I do not recall why.

Status: FAILURE.

4.3.64 Test tst_opaques.c

Description: Test netcdf-4 opaque types.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_opaques.c, line: 52

Comments: ESDM does not support user-defined datatypes from NetCDF.

Status: FAILURE.

4.3.65 Test tst_parallel.c

Description: This program tests netcdf-4 parallel I/O.

Output: No output.

Comments: It has to be tested with more nodes.

Status: SUCCESS.

4.3.66 Test tst_parallel3.c

Description: This test of netCDF-4 parallel I/O was contributed by the HDF5 team.

Output: Tests successful!

Comments: It has to be tested with more nodes.

Remove line 274: Expected error code: NC_ECANTEXTEND.

Status: PARTIAL SUCCESS.

4.3.67 Test tst_parallel4.c

Description: This is a benchmarking program for netCDF-4 parallel I/O.

Output: No output.

Comments: It has to be tested with more nodes.

Status: SUCCESS.

4.3.68 Test tst_parallel5.c

Description: This program tests netcdf-4 parallel I/O. In this test I write data on one task, while writing 0 items on others.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_parallel5.c, line: 121

Comments: ESDM does not support user-defined datatypes from NetCDF.

Status: FAILURE.

4.3.69 Test tst_put_vars.c

Description: No description.

Output: *** SUCCESS writing example file tst_put_vars.nc.

Comments: No comments.

Status: SUCCESS.

4.3.70 Test tst_put_vars_two_unlim_dim.c

Description: Test contributed in support of netCDF issue `https://github.com/Unidata/netcdf-c/issues/160`.

Output: No output.

Comments: No comments.

Status: SUCCESS.

4.3.71 Test tst_rehash.c

Description: Tests to see if the hashmap is being properly updated.

Output: Tests successful!

Comments: No comments.

Status: SUCCESS.

4.3.72 Test tst_rename.c

Description: Test renames of vars and dims.

Output: *** Tests successful!

Comments: Remove lines 480-556. ESDM does not support groups from NetCDF.

Status: PARTIAL SUCCESS.

4.3.73 Test tst_rename2.c

Description: Test more renames of vars and dims.

Output: *** Tests successful!

Status: SUCCESS.

4.3.74 Test tst_simplerw_coll_r.c

Description: This test is for parallel IO and the collective access of metadata with HDF5.

Output: *** Tests successful!

Comments: No comments.

Status: SUCCESS.

4.3.75 Test tst_strings.c

Description: Test netcdf-4 string types.

Output: Segmentation fault (core dumped)

Comments: ESDM does not support datatype NC_STRING. It is important to have a flag in ESDM code saying it does not support NC_STRING and stopping the program before reaching segmentation fault.

Status: FAILURE.

4.3.76 Test tst_strings2.c

Description: Test netcdf-4 string types.

Output: Segmentation fault (core dumped)

Comments: ESDM does not support datatype NC_STRING.

Status: FAILURE.

4.3.77 Test tst_sync.c

Description: Test netcdf-4 syncs.

Output: *** Tests successful!

Comments: Remove line 128. Expected error code: NC_EINDEFINE.

Status: PARTIAL SUCCESS.

4.3.78 Test tst_types.c

Description: Test netcdf-4 types.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_types.c, line: 105

Comments: ESDM does not support user-defined datatypes from NetCDF.

Status: FAILURE.

4.3.79 Test tst_udf.c

Description: Test user-defined formats.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_udf.c, line: 233

Comments: ESDM does not support user-defined datatypes from NetCDF.

Status: FAILURE.

4.3.80 Test tst_unlim_vars.c

Description: Test netcdf-4 variables with unlimited dimensions.

Output: *** Tests successful!

Comments: No comments.

Status: SUCCESS.

4.3.81 Test tst_utf8.c

Description: This is a very simple example which writes a NetCDF file with Unicode names encoded with UTF-8. It is the NETCDF3 equivalent of tst_unicode.c.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_utf8.c, line: 189

Comments: ESDM does not support Unicode names encoded with UTF-8.

Status: FAILURE.

4.3.82 Test tst_v2.c

Description: Test internal netcdf-4 file code.

Output: *** Tests successful!

Comments: No comments.

Status: SUCCESS.

4.3.83 Test tst_varms.c

Description: Test netcdf-4 mapped var operations.

Output: *** Tests successful!

Comments: No comments.

Status: SUCCESS.

4.3.84 Test tst_vars.c

Description: Test netcdf-4 variables.

Output: *** Tests successful!

Comments: Remove lines 354-369, 859-860. Code testing invalid parameters as input.

Remove lines 454-477. Functions nc_inq_type, nc_inq_typeid and nc_inq_typeids to be implemented. It should be working.

Remove lines 527-604, 731-808. ESDM does not support conversion for variables.

Remove lines 850-851, 874-875, 932-933, 955-956, 1017, 1019, 1041-1042, 1044, 1089, 1131. ESDM does not support compression.

Remove line 1294. ESDM does not support NC_FORMAT_CLASSIC.

Remove line 1299. ESDM does not support NC_FORMAT_64BIT_OFFSET.

Remove line 1304, 1309. ESDM does support with NC_FORMAT_NETCDF4.

Status: PARTIAL SUCCESS.

4.3.85 Test tst_vars2.c

Description: Test netcdf-4 variables.

Output: *** Tests successful!

Comments: Remove lines 94-95. Expected error code: NC_ELATEFILL.

Remove line 148, 744-745. Expected error code: NC_EPERM.

Remove lines 434-439. ESDM does not support NC_FORMAT_CLASSIC.

Remove lines 524, 530-535, 541, 544-551. Code testing invalid parameters as input.

Remove lines 638-639, 644-646, 652. Expected error code: NC_EBADNAME.

Remove line 676. Not clear what is the problem. It should be working.

Remove lines 720-725, 823-828, 841-846, 881-885, 902-907. Expected error code: NC_ENOTVAR.

Remove lines 733, 758. ESDM only supports native endianness.

Remove lines 821-822, 839-840, 878-879, 900-901. Expected error code: NC_EBADID.

Remove lines 829-830, 917, 1346. Expected error code: NC_EBADCHUNK.

Remove lines 847-850, 1407-1410, 1416, 1419. Expected error code: NC_EINVAL.

Remove lines 867-868, 890-891, 1360-1361, 1379-1380. Not clear what is the problem. It should be working.

Remove lines 888, 894. Not clear what is the problem. It should be working.

Remove lines 909. Expected error code: NC_ELATEDEF.

Remove lines 918-1031. Not clear what is the problem. If nc_def_var_chunking_ints does not work, it is pointless to continue with this part of the code.

Remove lines 1060-1062, 1079-1080, 1107-1108. Expected error code: NC_ENOTNC4.

Remove lines 1159, 1180. Expected error code: NC_CONTIGUOUS.

Remove lines 1242-1243, 1274-1275, 1303-1304. Expected error code: NC_ENAMEINUSE.

Remove lines 1287, 1314. Functions nc_inq_type, nc_inq_typeid and nc_inq_typeids to be implemented. It should be working.

Remove lines 1457-1458. Expected error code: NC_EBADDIM.

Remove lines 1470, 1473, 1478, 1484, 1490, 1499, 1532, 1535, 1573, 1574. ESDM does not support compression.

Status: PARTIAL SUCCESS.

4.3.86 Test tst_vars3.c

Description: Test netcdf-4 variables.

Output: *** Tests successful!

Comments: Remove lines 141, 143, 147-148, 150-151, 283. ESDM does not keep the same order for the dimensions.

Remove lines 159-197. ESDM only supports native endianness. ESDM does not support user-defined datatypes from NetCDF.

Remove lines 443-459. ESDM does not support filters.

Status: PARTIAL SUCCESS.

4.3.87 Test tst_vars4.c

Description: Test netcdf-4 variables.

Output: *** Tests successful!

Remove line 76, 82, 98, 104. Not clear what is the problem. It is related to compression. It is probably related to the order of parameters too.

Status: PARTIAL SUCCESS.

4.3.88 Test tst_vl.c

Description: Test netcdf-4 variable length code.

Output: Sorry. Unexpected result, ../../libsrcesdm_test/tst_vl.c, line: 56

Comments: ESDM does not support user-defined datatypes from NetCDF.

Status: FAILURE.

4.3.89 Test tst_xplatform.c

Description: Test netcdf-4 cross platform compound type.

Output: Sorry! Unexpected result, ../../libsrcesdm_test/tst_xplatform.c, line: 53

Comments: ESDM does not support user-defined datatypes from NetCDF.

Status: FAILURE.

4.3.90 Test tst_xplatform2.c

Description: Test netcdf-4 cross platform compound type.

Output: tst_xplatform2: /home/lucy/esiwace/esdm/deps/smd/src/smd-core.c:1348:
smd_attr_copy_val_to_internal: Assertion '0 && SMD cannot copy unknown type' failed.

Comments: ESDM does not support user-defined datatypes from NetCDF.

Status: FAILURE.

4.3.91 Test t_type.c

Description: This test program is only built if NetCDF-4 is disabled. It tests the NetCDF-3 version of NC_inq_type().

Output: *** Tests successful!

Comments: No comments.

Status: SUCCESS.

5 Tests in Python

5.1 Introduction

There is a Python module for accessing NetCDF files. We provide a patch (Section 5.2) to this module to support ESDM. The patch and instructions to run the tests are available in the directory `esdm-netcdf/dev`. The procedures to run the tests are following.

Run the build script:

```
$ ./build.sh
```

Python tests are created now inside the directory `dev/netcdf4-python/test`. Go to that directory.

```
$ cd netcdf4-python/test
```

Now, copy or link the `_esdm.conf` file to this directory. This can be done using a copy from the directory `esdm-netcdf/dev`.

5.2 Patch

The changings in the original patch to include ESDM can be seen next. Basically, we add the flags HAS_ESDM_SUPPORT, NC_FORMATX_ESDM, NC_ESDM and fmt == 'ESDM' (ESDM format).

```
diff --git a/include/netCDF4.pxi b/include/netCDF4.pxi
index 625752a..7bcd019 100644
--- a/include/netCDF4.pxi
+++ b/include/netCDF4.pxi
@@ -49,6 +49,7 @@ cdef extern from "netcdf.h":
          NC_WRITE # read & write
          # Use these 'mode' flags for nc_create.
          NC_CLOBBER
+         NC_ESDM # ESDM support for Python
          NC_NOCLOBBER # Don't destroy existing file on create
          NC_64BIT_OFFSET # Use large (64-bit) file offsets
          NC_64BIT_DATA # Use cdf-5 format
@@ -122,6 +123,7 @@ cdef extern from "netcdf.h":
          NC_FORMAT_NC3
          NC_FORMAT_NC_HDF4
          NC_FORMAT_NC_HDF5
+         NC_FORMATX_ESDM
          NC_FORMAT_DAP2
          NC_FORMAT_DAP4
          NC_FORMAT_PNETCDF
@@ -704,7 +706,7 @@ IF HAS_NC_CREATE_MEM:
              int flags
          int nc_close_memio(int ncid, NC_memio* info);
```

```
24
25 -IF HAS_PARALLEL4_SUPPORT or HAS_PNETCDF_SUPPORT:
26 +IF HAS_PARALLEL4_SUPPORT or HAS_PNETCDF_SUPPORT or HAS_ESDM_SUPPORT:
27      cdef extern from "mpi-compat.h": pass
28      cdef extern from "netcdf_par.h":
29          ctypedef int MPI_Comm
30 diff --git a/netCDF4/_netCDF4.pyx b/netCDF4/_netCDF4.pyx
31 index b693354..58d02e8 100644
32 --- a/netCDF4/_netCDF4.pyx
33 +++ b/netCDF4/_netCDF4.pyx
34 @@ -1264,7 +1264,7 @@ import_array()
35  include "constants.pyx"
36  include "membuf.pyx"
37  include "netCDF4.pxi"
38 -IF HAS_PARALLEL4_SUPPORT or HAS_PNETCDF_SUPPORT:
39 +IF HAS_PARALLEL4_SUPPORT or HAS_PNETCDF_SUPPORT or HAS_ESDM_SUPPORT:
40      cimport mpi4py.MPI as MPI
41      from mpi4py.libmpi cimport MPI_Comm, MPI_Info, MPI_Comm_dup,
    ↪ MPI_Info_dup, \
42                                 MPI_Comm_free, MPI_Info_free, MPI_INFO_NULL
    ↪ ,\
43 @@ -1344,11 +1344,13 @@ _intnptonctype  = {'i1' : NC_BYTE,
44  # create dictionary mapping string identifiers to netcdf format codes
45  _format_dict  = {'NETCDF3_CLASSIC' : NC_FORMAT_CLASSIC,
46                   'NETCDF4_CLASSIC' : NC_FORMAT_NETCDF4_CLASSIC,
47 -                 'NETCDF4'         : NC_FORMAT_NETCDF4}
48 +                 'NETCDF4'         : NC_FORMAT_NETCDF4,
49 +                 'ESDM'            : NC_FORMATX_ESDM}
50  # create dictionary mapping string identifiers to netcdf create format
    ↪ codes
51  _cmode_dict  = {'NETCDF3_CLASSIC' : NC_CLASSIC_MODEL,
52                  'NETCDF4_CLASSIC' : NC_CLASSIC_MODEL | NC_NETCDF4,
53 -                'NETCDF4'         : NC_NETCDF4}
54 +                'NETCDF4'         : NC_NETCDF4,
55 +                'ESDM'            : NC_ESDM}
56  IF HAS_CDF5_FORMAT:
57      # NETCDF3_64BIT deprecated, saved for compatibility.
58      # use NETCDF3_64BIT_OFFSET instead.
59 @@ -1547,6 +1549,8 @@ cdef _get_full_format(int grpid):
60              return 'DAP2'
61          elif formatp == NC_FORMAT_DAP4:
62              return 'DAP4'
63 +        elif formatp == NC_FORMATX_ESDM:
64 +            return 'ESDM'
65          elif formatp == NC_FORMAT_UNDEFINED:
66              return 'UNDEFINED'
67      ELSE:
68 @@ -1578,7 +1582,7 @@ be raised in the next release."""
69                  PyMem_Free(string_ptrs)
70          else:
71              # don't allow string array attributes in NETCDF3 files.
72 -            if is_netcdf3 and N > 1:
73 +            if is_netcdf3 and N > 1 and fmt != 'ESDM':
74                  msg='array string attributes can only be written with
    ↪ NETCDF4'
75                  raise IOError(msg)
76              if not value_arr.shape:
77 @@ -1593,7 +1597,7 @@ be raised in the next release."""
78      # if array is 64 bit integers or
79      # if 64-bit datatype not supported, cast to 32 bit integers.
80      fmt = _get_format(grp._grpid)
81 -    is_netcdf3 = fmt.startswith('NETCDF3') or fmt == 'NETCDF4_CLASSIC'
82 +    is_netcdf3 = fmt.startswith('NETCDF3') or fmt == 'NETCDF4_CLASSIC' or
    ↪ fmt == 'ESDM'
83      if value_arr.dtype.str[1:] == 'i8' and ('i8' not in _supportedtypes or
    ↪ \
84          (is_netcdf3 and fmt != 'NETCDF3_64BIT_DATA')):
85          value_arr = value_arr.astype('i4')
86 @@ -2030,7 +2034,7 @@ strings.
87      __pdoc__['Dataset.data_model']=\
88      """`data_model` describes the netCDF
89      data model version, one of `NETCDF3_CLASSIC`, `NETCDF4`,
90 -    `NETCDF4_CLASSIC`, `NETCDF3_64BIT_OFFSET` or `NETCDF3_64BIT_DATA`."""
91 +    `NETCDF4_CLASSIC`, `NETCDF3_64BIT_OFFSET` or `NETCDF3_64BIT_DATA` or
    ↪ ESDM."""
92      __pdoc__['Dataset.file_format']=\
93      """same as `data_model`, retained for backwards compatibility."""
94      __pdoc__['Dataset.disk_format']=\
95 @@ -2084,7 +2088,7 @@ strings.
96
97          **`format`**: underlying file format (one of `'NETCDF4',
```

```
98          'NETCDF4_CLASSIC', 'NETCDF3_CLASSIC'`, `'NETCDF3_64BIT_OFFSET'` or
99  -       `'NETCDF3_64BIT_DATA'`.
100 +       `'NETCDF3_64BIT_DATA'` or ESDM.
101         Only relevant if `mode = 'w'` (if `mode = 'r','a'` or `'r+'` the
    ↪ file format
102         is automatically detected). Default `'NETCDF4'`, which means the
    ↪ data is
103         stored in an HDF5 file, using netCDF 4 API features.  Setting
104 @@ -2156,7 +2160,7 @@ strings.
105         cdef char *path
106         cdef char namstring[NC_MAX_NAME+1]
107         cdef int cmode
108 -       IF HAS_PARALLEL4_SUPPORT or HAS_PNETCDF_SUPPORT:
109 +       IF HAS_PARALLEL4_SUPPORT or HAS_PNETCDF_SUPPORT or
    ↪ HAS_ESDM_SUPPORT:
110             cdef MPI_Comm mpicomm
111             cdef MPI_Info mpiinfo
112
113 @@ -2183,7 +2187,7 @@ strings.
114                 msg='parallel mode requires MPI enabled netcdf-c'
115                 raise ValueError(msg)
116             ELSE:
117 -               parallel_formats = []
118 +               parallel_formats = ['ESDM']
119                 IF HAS_PARALLEL4_SUPPORT:
120                     parallel_formats += ['NETCDF4','NETCDF4_CLASSIC']
121                 IF HAS_PNETCDF_SUPPORT:
122 @@ -2222,7 +2226,7 @@ strings.
123             else:
124                 if clobber:
125                     if parallel:
126 -                       IF HAS_PARALLEL4_SUPPORT or HAS_PNETCDF_SUPPORT:
127 +                       IF HAS_PARALLEL4_SUPPORT or HAS_PNETCDF_SUPPORT or
    ↪  HAS_ESDM_SUPPORT:
128                             ierr = nc_create_par(path, NC_CLOBBER | cmode,
    ↪  \
129                                                  mpicomm, mpiinfo, &grpid)
130                         ELSE:
131 @@ -2272,7 +2276,7 @@ strings.
132         version 4.4.1 or higher of the netcdf C lib, and rebuild netcdf4-
    ↪ python."""
133                     raise ValueError(msg)
134             elif parallel:
135 -               IF HAS_PARALLEL4_SUPPORT or HAS_PNETCDF_SUPPORT:
136 +               IF HAS_PARALLEL4_SUPPORT or HAS_PNETCDF_SUPPORT or
    ↪ HAS_ESDM_SUPPORT:
137                     ierr = nc_open_par(path, NC_NOWRITE | NC_MPIIO, \
138                                        mpicomm, mpiinfo, &grpid)
139                 ELSE:
140 @@ -2853,7 +2857,7 @@ Use if you need to ensure that a netCDF attribute is
    ↪ created with type
141  `NC_STRING` if the file format is `NETCDF4`."""
142         cdef nc_type xtype
143         xtype=-99
144 -       if self.data_model != 'NETCDF4':
145 +       if self.data_model != 'NETCDF4' and self.data_model != 'ESDM':
146             msg='file format does not support NC_STRING attributes'
147             raise IOError(msg)
148         _set_att(self, NC_GLOBAL, name, value, xtype=xtype, force_ncstring
    ↪ =True)
149 @@ -3435,7 +3435,7 @@ return the group that this `netCDF4.Dimension` is a
    ↪ member of."""
150  returns `True` if the `netCDF4.Dimension` instance is unlimited, `False`
    ↪ otherwise."""
151         cdef int ierr, n, numunlimdims, ndims, nvars, ngatts, xdimid
152         cdef int *unlimdimids
153 -       if self._data_model == 'NETCDF4':
154 +       if self._data_model == 'NETCDF4' or self._data_model == "ESDM":
155             ierr = nc_inq_unlimdims(self._grpid, &numunlimdims, NULL)
156             _ensure_nc_success(ierr)
157             if numunlimdims == 0:
158 @@ -4138,7 +4138,7 @@ Use if you need to ensure that a netCDF attribute is
    ↪ created with type
159  Use if you need to set an attribute to an array of variable-length strings
    ↪ ."""
160         cdef nc_type xtype
161         xtype=-99
162 -       if self._grp.data_model != 'NETCDF4':
163 +       if self._grp.data_model != 'NETCDF4' and self._grp.data_model != '
    ↪ ESDM':
164             msg='file format does not support NC_STRING attributes'
165             raise IOError(msg)
```

```
166           _set_att(self._grp, self._varid, name, value, xtype=xtype,
      ↪ force_ncstring=True)
167 diff --git a/setup.py b/setup.py
168 index febc020..d7ba091 100644
169 --- a/setup.py
170 +++ b/setup.py
171 @@ -58,6 +58,7 @@ def check_api(inc_dirs):
172       has_nc_create_mem = False
173       has_parallel4_support = False
174       has_pnetcdf_support = False
175 +     has_esdm_support = False
176
177       for d in inc_dirs:
178           try:
179 @@ -564,6 +565,7 @@ if 'sdist' not in sys.argv[1:] and 'clean' not in sys.
      ↪ argv[1:]:
180       else:
181           sys.stdout.write('netcdf lib does not have pnetcdf parallel
      ↪ functions\n')
182           f.write('DEF HAS_PNETCDF_SUPPORT = 0\n')
183 +     f.write('DEF HAS_ESDM_SUPPORT = 1\n') # TODO Fixme
184
185       f.close()
```

5.3 Tests

Similarly to what happened with the tests in C, the tests in Python mix functionalities that are not yet available in ESDM. In this case, we opt to just present the results using the original NetCDF tests without any modification. As expected, not many tests are sucessful using ESDM directly, for the reasons mentioned in Chapter 4.

Hopefully, the positive results are enough to prove the compatibility between the current version of ESDM and NetCDF Python.

Filename	Success	Failure
tst_atts.py		✓
tst_cdf5.py		✓
tst_compound_alignment.py		✓
tst_compoundatt.py		✓
tst_compoundvar.py		✓
tst_compression.py		✓
tst_create_mem.py	✓	
tst_dap.py		✓
tst_dims.py		✓
tst_diskless.py		✓
tst_endian.py		✓
tst_enum.py		✓
tst_fancyslicing.py		✓
tst_filepath.py		✓
tst_get_variables_by_attributes.py	✓	
tst_grps2.py		✓
tst_grps.py		✓
tst_issue908.py	✓	
tst_masked2.py		✓
tst_masked3.py		✓
tst_masked4.py		✓
tst_masked5.py		✓
tst_masked6.py		✓
tst_masked.py		✓
tst_multifile2.py		✓
tst_multifile.py		✓
tst_open_mem.py		✓
tst_refcount.py		✓
tst_rename.py		✓
tst_scalarvar.py		✓
tst_scaled.py		✓
tst_shape.py		✓
tst_slicing.py		✓
tst_stringarr.py	✓	
tst_types.py		✓
tst_unicode3.py		✓
tst_unicodeatt.py		✓
tst_unicode.py		✓
tst_unlimdim.py		✓
tst_Unsigned.py	✓	
tst_utils.py	✓	
tst_vars.py		✓
tst_vlen.py		✓
Total Files: 43	6	37

Table 5.1: Current status of all tests

6 NetCDF Benchmark

6.1 Introduction

So far, we tested individual features of ESDM. As part of this section, we discuss the usage of a benchmark application that uses NetCDF (and hence, ESDM) similarly to a real application.

NetCDF Performance Benchmark Tool (NetCDF-Bench) was developed to measure NetCDF performance on devices ranging from notebooks to large HPC systems. It mimics the typical I/O behaviour of scientific climate applications and captures the performance on each node/process. In the end, it aggregates the data to human-readable summary.

The data layout is inspired by simulation where a 3D object changes its shape over time. Therefore, it creates a 3-dimensional space and several time steps. Furthermore, we assume that a scientific application is executed on several processes on multiple nodes, and processes the time steps in sequential order.

NetCDF-Bench is parallel bechmark. It supports independent I/O, collective I/O and chunked I/O modes. If necessary, it can pre-fill the variables with some value. [1]

Scripts to set up the NetCDF Benchmark with ESDM can be found in the directory

```
esiwace/esdm-netcdf/libsrcesdm_test/netcdf-bench
```

The procedures to run the tests are following.

Run the prepare script:

```
./prepare.sh
```

Before starting the tests, we need to copy or link the file `_esdm.conf` to the directory `esdm-netcdf/libsrcesdm_test/netcdf-bench`. This can be done using a copy from the directory `esdm-netcdf/dev`.

Run the `mkfs.esdm` utility with the following parameters.

```
$ mkfs.esdm --create --remove --ignore-errors -g -c _esdm.conf
```

The benchmark file is called `benchtool` and it can be run with several parameters. For more information about this benchmark, run the command

[1] `https://github.com/joobog/netcdf-bench`

```
./benchtool –help
```

7 Tests with nccopy

7.1 Introduction

The nccopy command-line utility copies and optionally compresses and chunks netCDF data.

The nccopy has options to specify what kind of output to generate and optionally what level of compression to use and how to chunk the output. [1]

The nccopy utility can be found in the directory

```
esdm-netcdf/build/ncdump/nccopy
```

and the simplest way to call is

```
nccopy input_file output_file
```

As we want to test if the files generated by ESDM are compatible with NetCDF, we are going to run four different tests, described in the next sections. To call ESDM, just insert `esdm:\\` before the file.

Before starting the tests, we need to copy or link the file `_esdm.conf` to the directory `esdm-netcdf/build/ncdump/nccopy`. This can be done using a copy from the directory `esdm-netcdf/dev`. It is also good to run the `mkfs.esdm` utility prior to the tests with the following parameters.

```
$ mkfs.esdm --create --remove --ignore-errors -g -c _esdm.conf
```

7.2 Files

Table 7.1 introduces information about two originally NetCDF files[2] that will be used for testing. For simplicity, these files are renamed using their sizes as reference (column Nickname).

[1] `https://www.unidata.ucar.edu/software/netcdf/workshops/2011/utilities/Nccopy.html`
[2] `https://www.unidata.ucar.edu/software/netcdf/examples/files.html`

NetCDF File	Nickname	Size	Description
`sresa1b_ncar_ccsm3-example.nc`	`small.nc`	2.8 MB	From the Community Climate System Model (CCSM), one time step of precipitation flux, air temperature, and eastward wind.
`test_echam_spectral.nc`	`big.nc`	281.4 MB	Example model output from the ECHAM general circulation model. Almost CF, but not quite. Has a spectral coordinate for variables such as temperature (st) and vorticity (svo).

Table 7.1: Sample files following CF conventions

7.3 Tests

Copy the original NetCDF file to ESDM:

```
nccopy small.nc esdm:\\small-out.nc
```

Copy the ESDM file to NetCDF format:

```
nccopy esdm:\\small-out.nc small-final.nc
```

Compare the initial and final files:

```
diff small.nc small-final.nc
```

The absense of output using the command `diff` comproves the files are the same. The procedures are the same considering the file `big.nc` which is also perfectly copied.

We can also check the metadata of the ESDM file, which is available inside the directory `_metadummy/`.

8 Jenkins

8.1 Introduction

Jenkins is a free and open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat. It supports version control tools, including AccuRev, CVS, Subversion, Git, Mercurial, Perforce, ClearCase and RTC, and can execute Apache Ant, Apache Maven and sbt based projects as well as arbitrary shell scripts and Windows batch commands. The creator of Jenkins is Kohsuke Kawaguchi.[4] Released under the MIT License, Jenkins is free software.[5] [1]

8.2 Tests in Python

The tests in Python are inside the directory `dev/netcdf4-python/test`. Go to that directory.

All Python tests start with the prefix `tst_`. This string has to change to enable us to run `pytest`. One option to do that is using the `mmv` utility. Note that the names of the tests need to be changed, but the tests files (`.nc` files) have to remain unchanged.

The tool can be installed and used in Debian-based distributions as follows:

```
$ sudo apt-get install mmv
```

Now, run the `mmv` in the tests.

```
$ mmv tst.py test#1.py
```

and run the `mkfs.esdm` utility with the following parameters.

```
$ mkfs.esdm --create --remove --ignore-errors -g -c _esdm.conf
```

Finally, install the `pytest` utility

[1] `https://en.wikipedia.org/wiki/Jenkins_(software)`

```
$ sudo apt-get install python3-pytest
```

and run it

```
$ pytest --junitxml results.xml
```

The file `results.xml` can now be uploaded by Jenkins which provides the results in the website.

9 Conclusion

This has to be done in the end, with the current status of what is already working and what we can expect to work in the future.