

A new sub-chunking strategy for fast netCDF-4 access in local, remote and cloud infrastructures.

Cédric PENARD ¹ cedric.penard@thalesgroup.com Flavien GOUILLON ² flavien.gouillon@cnes.fr Xavier DELAUNAY ³ xavier.delaunay@eobytes.com Sylvain HERLÉDAN ⁴ sylvain.herledan@oceandatalab.com Pierre marie BRUNET ² pierre-marie.brunet@cnes.fr
¹Thales Services Numériques, Labège France ; ²CNES DTN/CD/AR, Toulouse, France ; ³eobytes, Ramonville-Saint-Agnes France ; ⁴OceanDataLab, Locmaria-Plouzané, France

INTRODUCTION

With the predominance of cloud data storage, evaluating NetCDF performance on cloud infrastructures is essential.

In this work, we propose a novel approach that was designed to improve the access to time series from native NetCDF files in the cloud.

The advantage of our approach is that it keeps existing data as they are **without requiring any reformatting**. The idea is to **reduce the amount of data read** from the NetCDF file when accessing a time series. To do this, our method creates **virtual sub-chunks** that can be read independently.

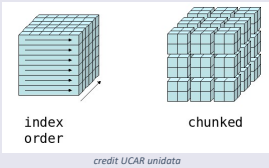
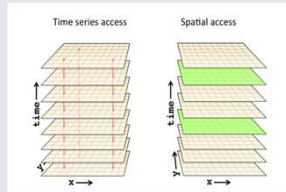
This novel approach called **chunkindex** involves indexing data within **compressed NetCDF** chunks, enabling **extraction of smaller compressed data portions** without reading the entire chunk. This feature is very valuable for accessing time series or for extracting small amounts of data from datasets with large chunks. It also saves reading time, particularly in scenarios of poor network connection such as those encountered onboard research vessels.

The objectives of this study are:

- To evaluate performance of NetCDF format on Posix and S3 file system through different use case.
- To improve performance of NetCDF via chunk indexing using chunkindex.

MATERIAL AND METHOD

In a NetCDF4 (and HDF5) file each variable is chunked. Chunking is necessary to apply data compression with the deflate algorithm. Each chunk is read or written as a single operation. To access to part of data lying within a chunk, the chunk have to be entirely loaded and uncompressed. If a file has big chunks this operation could take time, especially in some situation, like in time series access.



Chunkindex directly accesses data inside a chunk by using an index. The chunk is not entirely read and uncompressed, this saves time and reduces the amount of data transferred. The counter part is the creation of the index which increases the total size of the data, and add some reading time.

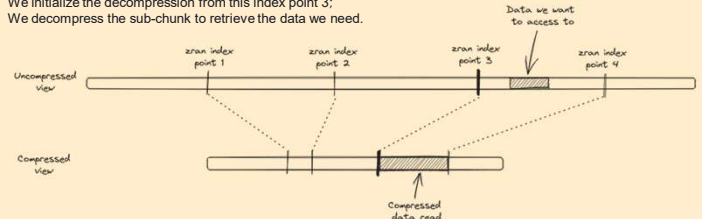
DEFINITIONS

Chunk: a portion of data in the netCDF file that can be read or written as a single I/O operation. The data compression is applied to full chunks.

Sub-chunk: a smaller portion of data that can be read using chunkindex approach.

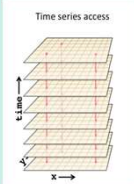
How do we do that ?

1. We build an index that contains 32KB windows of compressed data at some points in the chunk (eg zran index points 1 to 4). The 32KB windows are the decompression context for the deflate algorithm. They are necessary to start the decompression from these intermediate points;
2. We read from the index the 32KB window of the index point that lies just before the data we want to extract (eg. point 3);
3. We initialize the decompression from this index point 3;
4. We decompress the sub-chunk to retrieve the data we need.

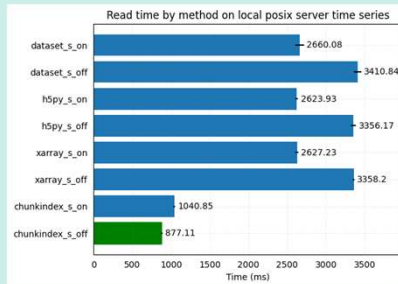


On Posix

Time series



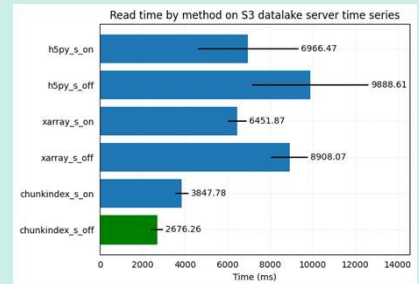
The chunkindex library enables a reduction in the amount of data read, thereby decreasing the reading time.



s_on : shuffle filter on
s_off : shuffle filter off

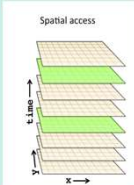
On S3

Time series

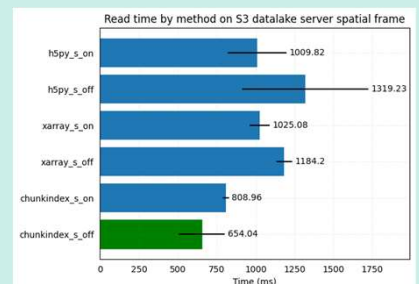
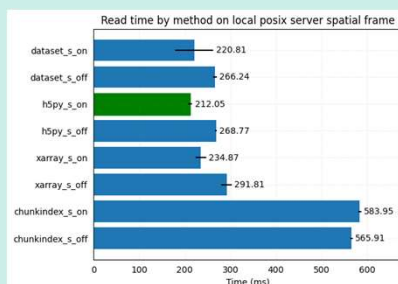


The chunkindex library works well with the fs_s3 library to directly access data stored on S3.

Spatial frame



Reading the index file incurs a cost in terms of reading time with the chunkindex library.



With chunkindex library, reducing the amount of data read saves time on network requests, despite the reading of the index file.

Abstract:



Contact:



CONCLUSION AND PERSPECTIVES

The reading speed of a NetCDF file essentially depends on the use case and the structure of the file read. Chunkindex method works well when getting time series from data with chunks divided by time steps. Results are mitigated when getting all data from a spatial frame.

In remote situation the best way to read a NetCDF file is to get only necessary data. In network degraded situation the H5py library is a good solution to read NetCDF files, but chunkindex could be a suitable alternative to limit the amount of data downloaded. The universality, portability and performance of the NetCDF format, even in cloud environments, mean that this format still has a bright future ahead.

A paper is in progress on the subject, and the next step is to explore the possibility of integrating the chunkindex library into the Kerchunk library. Additionally, we would like to investigate the potential offered by the new ncZarr format developed by UCAR.