https://doi.org/10.5194/egusphere-2026-3249
Preprint. Discussion started: 16 June 2026

New insights on the suitability of NetCDF/HDF5 as storage format for climate cloud repositories

Ezequiel Cimadevilla[1], David Hassell[2], and Bryan N. Lawrence[2]

[1]Instituto de Física de Cantabria (IFCA), CSIC-Universidad de Cantabria, Santander, Spain
[2]National Centre for Atmospheric Science, Department of Meteorology, University of Reading, Reading, UK

Correspondence: Ezequiel Cimadevilla (e.cimadevilla@csic.es)

Abstract. Climate data analysis increasingly relies on cloud infrastructures to offer new and efficient methods of accessing the necessary climate data. Cloud repositories allow access to such data in a remote data access basis, which allows users to retrieve and manipulate their data without requiring file downloads, reducing storage costs resulting in more efficient systems. Together, climate cloud repositories and remote data access are evolving fast, due to the necessity of collaboration that brings together diverse communities to address challenges in climate science. In recent years, a prevailing discourse has emerged suggesting that traditional climate data storage formats are inherently unsuitable for remote data access. In this work, we present new insights that challenge this discourse and demonstrate that established storage formats such as NetCDF/HDF5 can continue to operate efficiently in cloud environments when accessed remotely. These findings contrast with the widespread perception that such formats are inherently unsuitable for cloud based workflows. In the context of the onset of CMIP7, these insights have the potential to substantially enhance climate data access and analysis for the broader research community without incurring major maintenance burdens.

1 Introduction

The predominant storage format in climate science has been NetCDF/HDF5, meaning that NetCDF serves as the library interface to files stored in HDF5 (Rew et al., 2006). We abbreviate NetCDF/HDF5 as NetCDF, since we do not consider earlier storage formats of NetCDF in the scope of this work. A NetCDF file may contain several multidimensional arrays, or variables, where each multidimensional array might be a field variable or a support variable (Hassell et al., 2017). Field variables include physical variables of the climate system, such as surface temperature, precipitation or wind speed. Support variables locate field variables in space and time. Field variables account for the majority of data to be stored, while support variables currently represent an almost negligible amount of data relative to field variables. Climate model output consists primarily of *gridded* data, meaning that it comprises field variables that represent physical variable values over cell boundaries and their associated support variables.

For several decades, NetCDF has been the predominant storage format for climate datasets, owing to its efficiency in representing dense climate data (Cimadevilla, 2025). Climate data produced within the framework of Model Intercomparison Projects (MIP) are generated by global or regional climate models. The Earth System Grid Federation (ESGF) has arisen as

https://doi.org/10.5194/egusphere-2026-3249
Preprint. Discussion started: 16 June 2026

the federated solution to share climate datasets worldwide, due to the large amount of data available (Cinquini et al., 2012). The emergence of cloud infrastructures, in contrast to traditional High Performance Computing (HPC) environments that form part of the ESGF, has raised questions regarding the suitability of NetCDF as a storage format for cloud based workflows based on remote data access (Abernathey et al., 2021). The prevailing discourse has suggested that NetCDF is not well suited to the cloud and that alternative formats and workarounds, such as Zarr and Kerchunk, are necessary if climate data are to be effectively stored and accessed in a remote data access basis, either from the ESGF or from cloud repositories.

In this work, we address some factors that have been overlooked and have contributed to the perception that NetCDF is unsuitable as a storage format for remote data access and climate cloud repositories. We provide detailed, low level explanations of the internal data structures of NetCDF files that have led to the conclusion that NetCDF performs poorly for remote data access in cloud environments. Furthermore, we develop and describe the tools and practices necessary to overcome these limitations and render NetCDF performant in the cloud. We then evaluate and analyze the performance of different technologies that are used to achieve efficient remote data access.

2 Background

Climate data analysis has traditionally involved downloading NetCDF files to local workstations or computing infrastructures for subsequent analysis. This approach relies on certain simplifications that must be reconsidered in other modes of data access, as in the context of remote data access. Data access to locally downloaded NetCDF files offers the advantages of relative simplicity and high performance, since latency to randomly access a NetCDF files is limited to low-latency data transfer within a local workstation or local area network (LAN), in contrast to high-latency remote data access that depends on wide area networks (WAN). However, this method presents significant challenges for analysts, as it requires substantial time and effort to manage data acquisition and download. Tasks such as downloading and organizing large volumes of files often divert attention and resources away from core research activities.

Climate cloud repositories leverage cloud optimized storage formats and remote data access mechanisms to enable researchers to query and analyze data directly on remote servers or cloud platforms without transferring entire datasets (Abernathey et al., 2021). This capability, commonly referred to as subsetting, allows specific workflows to retrieve only a limited portion of a dataset, thereby reducing the volume of data transmitted over the network relative to a file download approach. Consequently, spatial subsets of large domains or selected temporal segments of extended time series can be analyzed without downloading entire files that encompass the full spatiotemporal extent of the data. However, to achieve such efficiency, storage formats must be appropriately structured to support remote access and subsetting in a performant manner. The main technique to account for this efficiency is chunking.

NetCDF variables make use of chunking to allow compression of data and more efficient data access and storage. Because storage media are inherently single dimensional, whereas climate data are multidimensional in nature, multidimensional arrays that contain climate data must be transformed from their multidimensional spatiotemporal representation into a single dimensional storage space. This transformation entails a critical trade off, as the relationship between data access patterns and storage

https://doi.org/10.5194/egusphere-2026-3249
Preprint. Discussion started: 16 June 2026

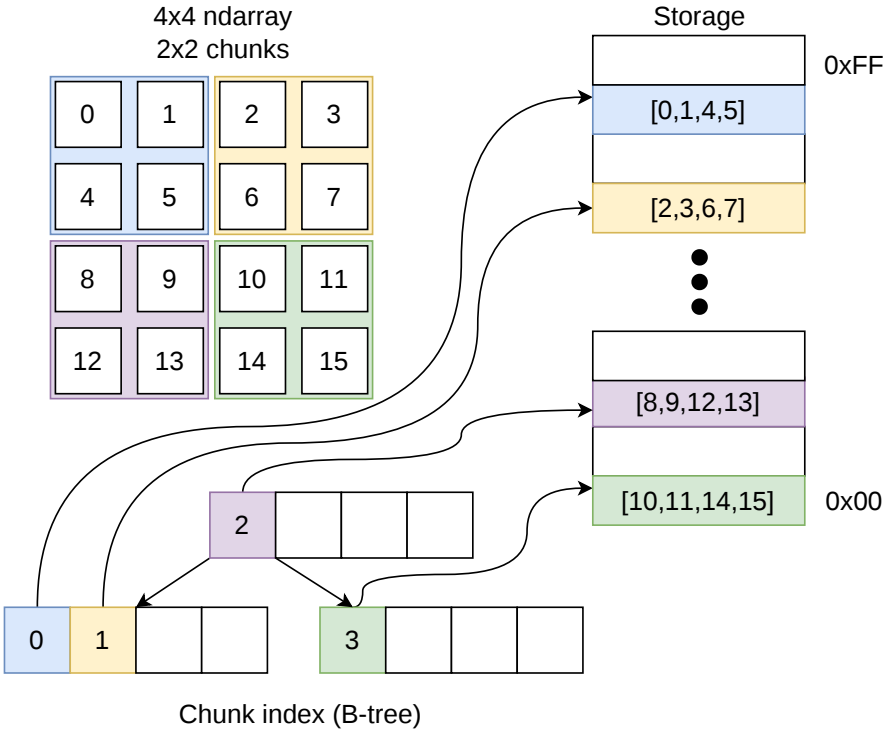

Figure 1. Illustration of chunk indexing. A chunk index is a data structure that maps each chunk identifier to its position in linear storage. In the context of this work, a chunk index is implemented by a B-tree, whose nodes may be scattered throughout the linear space of the NetCDF file. Arrows in the B-tree represent either node pointers to other nodes or pointers to raw chunks, both located arbitrarily in the space of the file.

layout directly determines data retrieval latency. The resulting performance differences can span several orders of magnitude. As will be shown, chunking involves not only the size of the chunks, but also the data structures used to index their location within a climate data storage format. Figure 1 provides visual conceptualization of chunking and chunk indexing.

To accommodate the discontiguous storage of chunks within the linear address space of a NetCDF file, a mechanism is required to index the chunk locations. This is required because the data might be written incrementally in no specific order and we don't want to reserve empty space in advance. B-trees provide a flexible storage solution for objects that grow in a way that leads to discontiguous storage of chunks. B-trees are self-balancing search trees that maintain sorted data and allow searches, sequential access, insertions, and deletions in logarithmic time (Bayer and McCreight, 1972). They are widely used in databases and file systems due to their efficiency in handling large amounts of data (Comer, 1979). In order to understand why NetCDF has come to be regarded as unsuitable for storing climate data in the cloud for remote access, it is essential to examine the internal data structures of a NetCDF file, particularly those used to index chunk locations. We will provide the evidence that, by organizing these data structures appropriately, NetCDF can be considered cloud optimized and effectively employed as a storage format in climate cloud repositories.

https://doi.org/10.5194/egusphere-2026-3249
Preprint. Discussion started: 16 June 2026

3 Methods

This work is based on the hypothesis that it is possible to achieve performance comparable to that of cloud-native storage formats when using NetCDF. It further examines the factors that must be considered in order to make NetCDF performant in cloud environments. We selected a small set of NetCDF files produced in the context of CMIP that represent different file size scales that correspond to datasets with varying temporal resolutions. This selection serves to illustrate how differences in temporal resolution influence the internal data structures of NetCDF files and, consequently, have a significant impact on performance. Three files were chosen, with monthly, daily, and three hourly temporal resolutions. Table 1 summarizes the original NetCDF files, as downloaded from the ESGF, that are considered in this study.

NetCDF file name	File size	Chunks	Metadata
uas_Amon_IPSL-CM6A-LR_piControl_r1i1p1f1_gr_185001-234912.nc	381 MB	6000	877KB
uas_day_IPSL-CM6A-LR_piControl_r1i1p1f1_gr_185001-234912.nc	12 GB	182621	25,6MB
uas_3hr_IPSL-CM6A-LR_piControl_r1i1p1f1_gr_187001010300-197001010000.nc	19 GB	292192	41MB

Table 1. Relation of the original NetCDF files as obtained from the ESGF. The selected files sample a range of temporal resolutions in order to account for differences in the size of the internal data structures of HDF5. It can be observed from the metadata column that the size of the internal data structures increases rapidly as a greater number of chunks must be indexed for the NetCDF variables. Note that the field variable and *time* coordinate of these files are chunked with one spatial latitude/longitude chunk per time step. The chunks column refers to the number of chunks of the field variable only (*uas*).

These original files are generated within HPC infrastructures and are intended for use in a low-latency file downloads approach rather than a high-latency remote data access setting. As a result, their internal data structures and B-trees are positioned arbitrarily within the NetCDF file. This arrangement does not pose a problem when analyzing data on a local workstation, where the latency associated with accessing B-tree fragments is negligible. However, in a remote data access context, the latency cost of retrieving many small byte segments from disparate locations becomes substantial. This factor has been largely overlooked in the literature when accounting for the high cost of accessing NetCDF files from cloud repositories. Figure 2 illustrates the locations of B-tree fragments in the original NetCDF files as downloaded from the ESGF.

Table 1 highlights several factors that contribute to the inefficiencies associated with accessing these NetCDF files when they are uploaded directly to the cloud and accessed through remote data access. First, the daily and three hourly files contain a substantial volume of metadata, which in this context refers to internal HDF5 structures and not user attributes. Although this metadata is relatively small compared with the total size of the NetCDF file, it is sufficiently large to generate inefficiencies in remote access scenarios. The daily file contains approximately 25 MB of metadata, which increases to more than 40 MB in the three hourly file. However, the principal source of inefficiency is not merely the size of the metadata, which could be retrieved within a few seconds if it were stored contiguously. Rather, the key factor is that these metadata bytes are scattered throughout the whole linear space of the NetCDF file, as displayed in Figure 2. Consequently, retrieving the metadata requires thousands of individual requests, which, in high-latency environments such as remote data access, renders data access effectively impractical.

https://doi.org/10.5194/egusphere-2026-3249
Preprint. Discussion started: 16 June 2026

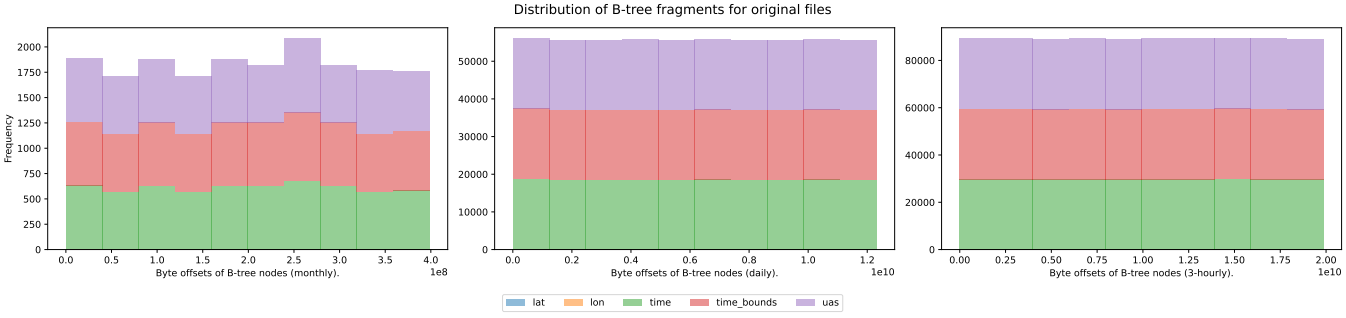

Figure 2. Stacked histogram of the locations of all B-tree fragments in the original CMIP6 files, given by B-tree offsets in bytes from the start of the file, showing that for the uas and time, and time_bounds variables they are scattered throughout the whole NetCDF files. The X-axis shows that B-tree fragments occupy the full space of each original NetCDF file.

This issue also affects smaller files, such as the monthly NetCDF file from ESGF shown in Table 1. Although it contains a relatively small amount of metadata, the fact that these metadata are scattered throughout the file makes retrieving subsets practically intractable.

If these metadata bytes were stored in a predictable location within the linear storage space of the NetCDF file, clients could exploit this organization to retrieve all metadata in a single read operation. This would avoid the inefficiencies that currently render such files impractical to analyze when stored in the cloud. We refer to files whose metadata bytes are scattered throughout the file as *fragmented*, in analogy to traditional file system fragmentation that occurs in conventional computer systems. We refer to the process of modifying NetCDF files with the aim of reallocating these bytes to appropriate locations as *repacking*. Files that have undergone this procedure are described as having *consolidated metadata* or being *defragmented*, meaning that their internal data structures are placed at the beginning of the file, where clients can retrieve this information in a single read operation. However, it is important to notice that climate data producers may produce NetCDF with consolidated metadata without the need for repacking, given that they configure their tools properly. We will see that to fully leverage modern WAN networks and maximize bandwidth utilization, clients must not only understand the internal data structure of a NetCDF file for efficient remote data access, but also implement appropriate concurrency mechanisms to retrieve data chunks in parallel.

4 Validation

Under the current assumptions, the available mechanisms for efficient remote access to climate data involve either transforming the data into Zarr format or generating a *Kerchunk* file for the NetCDF dataset, which effectively means creating an extended copy of the NetCDF B-tree index. Kerchunk provides a Zarr interface to existing NetCDF files, enabling efficient data access without duplicating the full dataset, as it stores only the locations and sizes of the chunks within the NetCDF files. As a result, a Kerchunk file represents only a small fraction of the original NetCDF dataset, although, as discussed below, this size is not negligible under current chunking practices within the climate data community. In this sense, Kerchunk consolidates metadata

https://doi.org/10.5194/egusphere-2026-3249
Preprint. Discussion started: 16 June 2026

without requiring reorganization of the original files, albeit at the expense of managing the resulting Kerchunk files. This cost might be far from insignificant as we shall see. Table 2 shows the corresponding Kerchunk files for the original NetCDF files obtained from the ESGF.

Kerchunk file name	Size	References
uas_Amon_IPSL-CM6A-LR_piControl_r1i1p1f1_gr_185001-234912.nc.json	1.4 MB	18017
uas_day_IPSL-CM6A-LR_piControl_r1i1p1f1_gr_185001-234912.nc.json	44 MB	547880
uas_3hr_IPSL-CM6A-LR_piControl_r1i1p1f1_gr_187001010300-197001010000.nc.json	71 MB	876593

Table 2. File size and number of references contained within the corresponding Kerchunk files generated from NetCDF datasets obtained from the ESGF. The large size and number of references result from historical suboptimal chunking practices within the climate data community. Alternative Kerchunk formats, such as Parquet, aim to reduce the size of the resulting files but do not address the underlying issue, namely the presence of an excessively large and unnecessary number of references.

We will provide evidence that properly repacked NetCDF files, when accessed with suitably optimized clients, can support efficient remote data access without the need for Kerchunk files. Moreover, we illustrate how repacking favors smaller and more manageable Kerchunk representations by reducing their file size while improving access performance, thereby enhancing their usefulness when generating aggregations from multiple NetCDF source datasets. Thus, other uses of Kerchunk such as generating virtual aggregations of multiple NetCDF files would greatly benefit from repacked NetCDF files.

4.1 Repacking

We performed repacking of the original files obtained from the ESGF using *cmip7repack* (Hassell and Cimadevilla, 2025). This tool is designed to organize the internal data structures of an HDF5 file so that they are placed at known and predictable locations at the beginning of a NetCDF file. After repacking, the metadata bytes are both reduced in size and relocated to the beginning of the file. Although the reduction in size has a positive impact on performance, the relocation of metadata is a crucial factor, but not the only one, in enabling NetCDF to perform efficiently in the cloud, as clients can then retrieve these metadata bytes in a single read operation. Furthermore, using larger chunks and consequently fewer chunks, substantially reduced the size of internal data structures by decreasing the number of chunks and, consequently, the size of the B-trees used for indexing. Table 3 illustrates the benefits of repacking the original NetCDF files while specifying a chunk size of 4 MB (before compression) for the field variable (*uas*).

It is also important to note that support variables that were originally stored with suboptimal chunking are now stored in a single larger chunk, thereby eliminating the associated B-tree fragments previously required to index the chunks of those variables (*time* and *time_bounds*). The most striking example of the impact of rechunking support variables into a single chunk can be observed in the three hourly file, which originally required 292192 chunks for each of the *uas*, *time*, and *time_bounds* variables. This configuration accounted for the majority of the 40 MB of metadata stored in the file. This also has the effect of substantially reducing the size of the generated Kerchunk files for these NetCDF datasets, which may still be desirable for

https://doi.org/10.5194/egusphere-2026-3249
Preprint. Discussion started: 16 June 2026

NetCDF file name	Chunks	Metadata
uas_Amon_IPSL-CM6A-LR_piControl_r1i1p1f1_gr_185001-234912.nc_cmip7repack4mb	120	32KB
uas_day_IPSL-CM6A-LR_piControl_r1i1p1f1_gr_185001-234912.nc_cmip7repack4mb	3653	227KB
uas_3hr_IPSL-CM6A-LR_piControl_r1i1p1f1_gr_187001010300-197001010000.nc_cmip7repack4mb	5844	355KB

Table 3. Relation of the repacked NetCDF files obtained after applying *cmip7repack* to the original files. Repacking was performed by specifying chunk sizes of 4 MB for the climate variable (*uas*). It can be observed that the size of the internal HDF5 data structures decreases significantly as the chunk size increases.

constructing aggregated datasets from multiple NetCDF files. Table 4 shows the information for Kerchunk files of repacked datasets.

Kerchunk file name	Size	References
uas_Amon_IPSL-CM6A-LR_piControl_r1i1p1f1_gr_185001-234912.nc_cmip7repack4mb.json	28 KB	139
uas_day_IPSL-CM6A-LR_piControl_r1i1p1f1_gr_185001-234912.nc_cmip7repack4mb.json	592 KB	3672
uas_3hr_IPSL-CM6A-LR_piControl_r1i1p1f1_gr_187001010300-197001010000.nc_cmip7repack4mb.json	992 KB	5863

Table 4. File size and number of references contained within the corresponding Kerchunk files generated from NetCDF datasets that have repacked with *cmip7repack*. By using larger chunks and storing support variables within a single chunk, both the file sizes and the number of references are substantially reduced in comparison with the original Kerchunk files.

By storing the support variables in a single chunk and using a chunk of size $(50, 143, 144)$ instead of the original $(1, 143, 144)$ for the *uas* variable, the number of B-tree nodes is reduced substantially. Consequently, for the 3-hourly file the volume of metadata decreases from the original 40 MB to approximately 355 KB. More importantly, this metadata is now located at the beginning of the file rather than being scattered throughout the entire file. Figure 3 illustrates the distribution of B-tree fragments when 4 MB chunks are used.

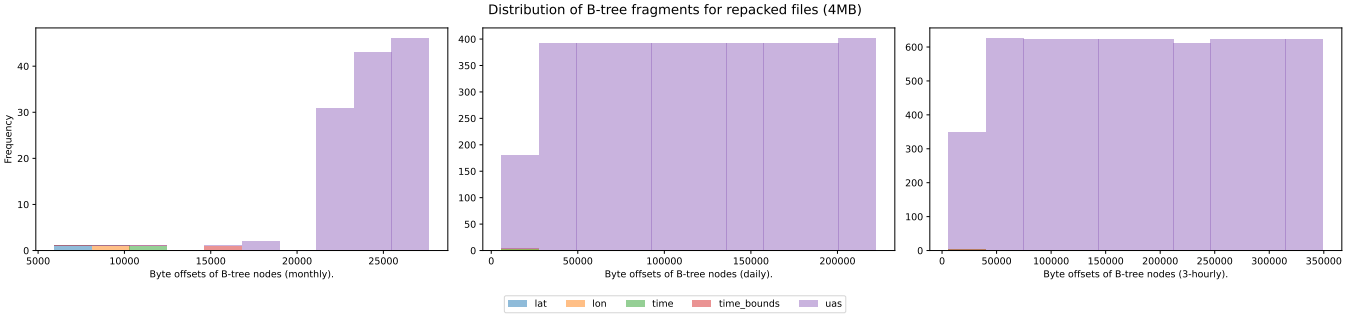

Figure 3. Locations of all of the B-tree fragments in the repacked CMIP6 files using 4MB chunks for the climate variable (*uas*). Locations are given by B-tree offsets in bytes from the start of the file. In contrast to the original ESGF files, coordinate and support variables are stored in a single chunk after repacking, and all B-tree data is stored at the beginning of the file.

https://doi.org/10.5194/egusphere-2026-3249
Preprint. Discussion started: 16 June 2026

4.2 Performance

To assess the impact of repacking, we conducted a performance experiment aimed at evaluating the implications of properly formatted NetCDF files when accessed remotely over a WAN network, where a cloud repository located in the UK stores the data and these are accessed from a virtual machine client located in a private cloud provider in Spain. We aim to demonstrate that defragmented NetCDF files meet the criteria to be cloud efficient without the need for Kerchunk sidecar files. Despite the importance of repacking, it is important to note that repacking is a necessary but not sufficient condition to guaranty optimal performance in a remote data access context. To achieve full performance, software clients must satisfy at least two conditions: they must be able to exploit the structure of repacked NetCDF files, and they must be capable of reading chunks concurrently.

The first condition can be satisfied by instructing clients to request an appropriate amount of data from the beginning of the file, where the consolidated metadata are located. The second condition is more challenging to achieve. In this case, clients must employ parallelism techniques, such as asynchronous execution, multithreading, and multiprocessing, to enable parallel reading of chunks so that the available bandwidth can be effectively utilized. For the evaluation of performance of repacked NetCDF files we will utilize both Kerchunk/Zarr and Pyfive (Lawrence et al., 2026), a recent pure Python HDF5 reader developed to enhance access to climate data.

We designed the performance experiment in two parts. First, we demonstrate that the lack of repacking accounts for the poor performance often attributed to NetCDF in cloud environments. Second, we show that once a file has an appropriate structure—that is, when the B-trees are located at well-defined positions—overall performance depends on the ability of clients to implement concurrent chunk retrieval to saturate the available bandwidth. We compare the results obtained from both *opening* and *loading* the data using different configurations and clients. Figure 4 presents the time required to open the different files and stores under different formats and configurations.

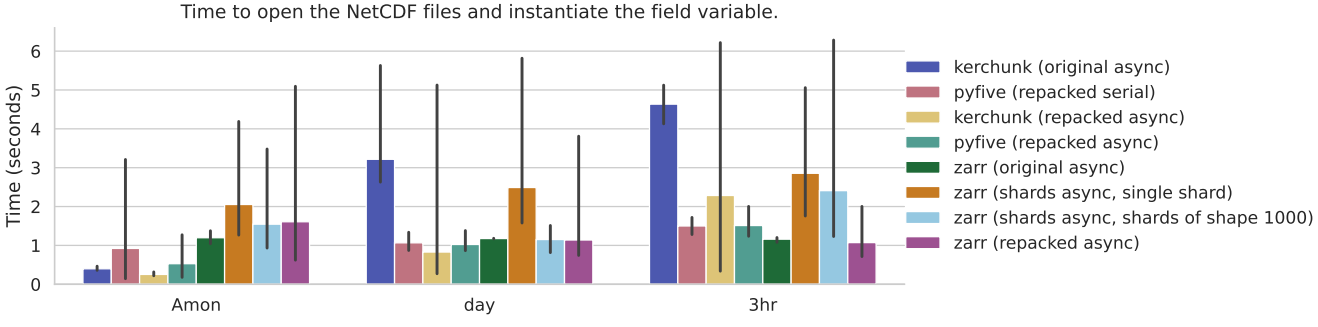

Figure 4. Measured time required to open different NetCDF files from an object store using various clients and configurations. Original NetCDF files are accessed only via Kerchunk, which compensates for the fact that the B-trees in the original files are dispersed throughout the file space, albeit at the cost of significantly larger Kerchunk files. Concurrent access to chunks is marked as *async*. Error bars display minimum and maximum times available in the sample.

https://doi.org/10.5194/egusphere-2026-3249
Preprint. Discussion started: 16 June 2026

The first observation in Figure 4 is the absence of results for opening an original (non-repacked) NetCDF file using Pyfive. In this context, we define *opening* a file as the process of reading its basic metadata, including the B-trees of the chunked variables. Because the B-tree fragments in the original files are dispersed throughout the entire file space, as shown in Table 1, attempting to open such files becomes impractical. This limitation has contributed to the perception that NetCDF is inherently unsuitable for cloud storage. However, we report the time required to open the original NetCDF files using Kerchunk. This illustrates the usage of Kerchunk, which is to provide a duplicated cache of the B-trees for the chunked variables. However, the time required to open the file is considerably higher than for repacked files in relative terms. This behavior is driven by the substantial size of Kerchunk files generated for the original NetCDF data and the large number of references they contain, as shown in Table 2.

Figure 4 clearly demonstrates the benefits of NetCDF files with consolidated metadata from the repacking process. Because the B-trees are located at the beginning of the NetCDF files, all clients that support the basic functionality of loading the initial bytes of a remote file can take advantage of this, enabling files to be opened within a few seconds. This latency is typical and expected when accessing remote data over WAN networks. Additionally, due to the rechunking of the original files into larger chunks, Kerchunk files become significantly smaller, allowing them to be loaded much more quickly. Thus, contrary to common belief, NetCDF can be natively performant in the cloud, provided that the internal structures of HDF5 are properly understood and configured to meet the requirements of remote data access over WAN networks. However, the performance of NetCDF cannot be assessed solely by the time required to open a file; it is also necessary to consider the latency and throughput when loading data from its variables. Figure 5 shows the time required to load the *uas* variable under different configurations and across various clients.

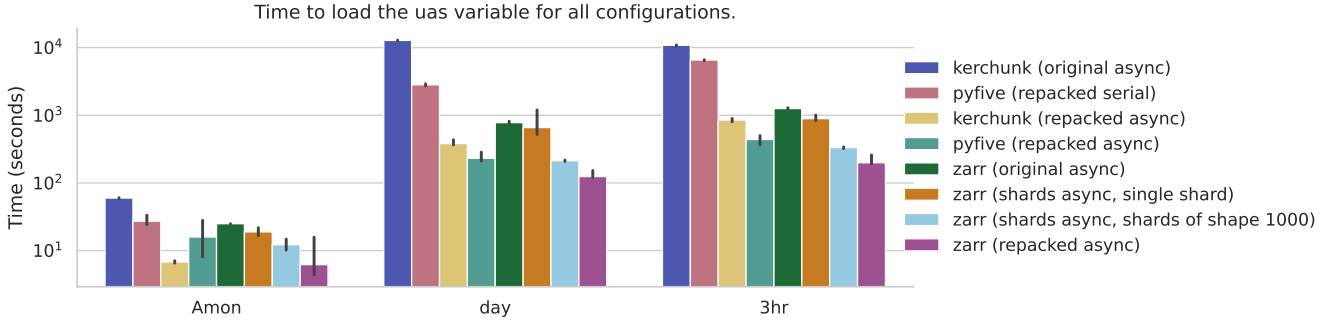

Figure 5. Measured time required to load the *uas* variable from different NetCDF files stored in an object store using various clients and configurations. The variable is accessed in its entirety, without a specific data access pattern, to illustrate the throughput capabilities of each configuration and client. Error bars display minimun and maximun times available in the sample.

The results highlight two key factors, beyond the previously discussed issue of fragmented B-trees, that have led the climate community to assume that NetCDF is unsuitable for cloud storage. First, the small chunk sizes defined in the original NetCDF files impose a major limitation on throughput. Chunks on the order of kilobytes incur significant latency costs, whereas larger chunks on the order of megabytes better utilize the network and can saturate the available bandwidth. Second, the lack of

https://doi.org/10.5194/egusphere-2026-3249
Preprint. Discussion started: 16 June 2026

clients with the capability to fetch chunks concurrently represents another major limitation, as it prevents full utilization of the available bandwidth.

Figure 5 illustrates these caveats by using Kerchunk to access the original NetCDF files as published in the ESGF and by using Pyfive in serial mode, i.e., without concurrency mechanisms. While Kerchunk relies on the concurrency features provided by asynchronous HTTP requests, the small chunk sizes still prevent it from achieving optimal performance. Thus, assumptions that Kerchunk resolves all performance issues caused by poorly formatted or fragmented NetCDF files should be treated with caution.

To further assess the performance of formats other than NetCDF, we converted the NetCDF files into corresponding Zarr stores that matched the chunking configuration of the original datasets (i.e., identical chunk shapes, compression settings, and filter configurations). In evaluating Zarr, we considered stores that: (a) matched the configuration of the original NetCDF files, (b) matched the configuration of the repacked NetCDF files, and (c) employed different sharding configurations to account for performance differences between full HTTP requests and partial HTTP range requests. We observe that our storage system of choice delivers higher performance when Zarr stores chunks independently, whereas performance is reduced when sharding is used, as this requires HTTP range requests to access individual chunks.

The results demonstrate that remote data-access performance is not limited to a particular library or format; rather, it arises from more complex interactions between the characteristics of specific formats and the capabilities of the libraries used to access them. This is further illustrated by using Pyfive to access repacked NetCDF files without using the now default concurrency support, which still fails to achieve optimal performance.

When the two criteria, appropriate chunking and effective concurrency mechanisms, are met by both the storage format and the clients, in addition to properly repacked NetCDF files, full performance can be achieved. This is illustrated in Figure 5, where Kerchunk is used to access repacked NetCDF files and Pyfive uses the concurrency support which we recently added (and which is now used by default for remote access). When Kerchunk accesses a properly repacked NetCDF file, it can take advantage of larger chunks, thereby saturating the available bandwidth and improving performance accordingly—something that could not be achieved with Kerchunk applied to the original NetCDF files. In the case of Pyfive, the use of concurrency mechanisms enables high performance as well, which was not possible under a serial, non-concurrent approach, even when the NetCDF files were properly repacked. We will look now at the times required to access the data but only for those configurations that make sense in an operational setting, leaving out those that were obtained for experimental reference only. These results are displayed in Figure 6.

Figure 6 shows that the performance of Kerchunk based on repacked NetCDF files is significantly worse than that of Pyfive or Zarr, even though these Kerchunk references point to properly repacked NetCDF files rather than the original files. This makes Kerchunk far from optimal for large NetCDF datasets. Consequently, the common assumption within the climate community that Kerchunk can fully offset the limitations of fragmented NetCDF files should be treated with caution.

We also present the performance of Pyfive, which accesses repacked NetCDF files directly, and compare it against native Zarr stores. The results indicate that our storage infrastructure penalizes HTTP range requests, thereby reducing the performance of both Pyfive and sharded Zarr configurations. For this particular storage system, unsharded Zarr, in which each chunk is stored

https://doi.org/10.5194/egusphere-2026-3249
Preprint. Discussion started: 16 June 2026

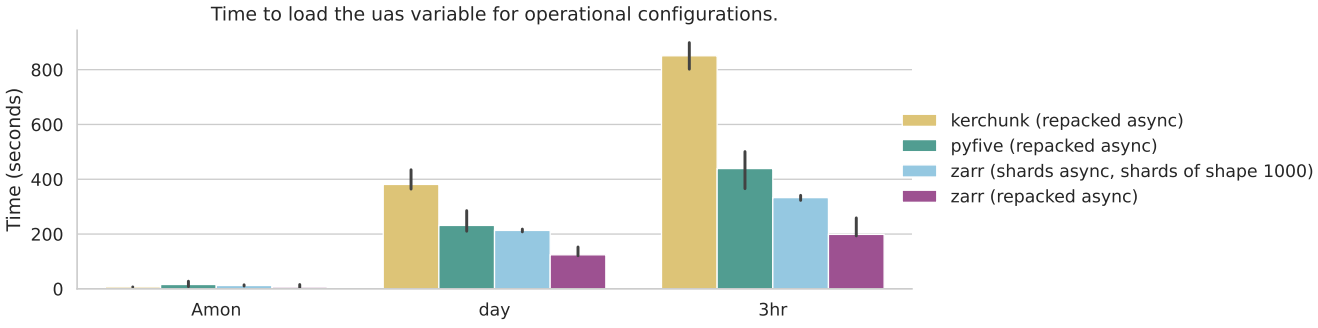

Figure 6. Measured time required to load the *uas* variable from different NetCDF files stored in an object store using various clients and configurations. The variable is accessed in its entirety, without a specific data access pattern, to illustrate the throughput capabilities of each configuration and client. Error bars display minimun and maximun times available in the sample.

as an individual object, achieves the best performance, albeit at a considerable cost in terms of object/file maintenance once every chunk is a file/object (and detrimental performance on parallel file systems). Overall, these results illustrate that native NetCDF can in fact provide strong performance when properly configured and accessed, although achieving this required the development of a new library, Pyfive, capable of exploiting these features effectively.

5 Conclusions

Climate data access is undergoing rapid transformation in response to emerging requirements driven by increasingly collaborative research environments and the challenges associated with Big Data. Traditionally, climate data distribution has relied on the generation of NetCDF files at HPC centers where climate models are executed. Subsequent data analysis has typically been conducted either at these same centers or at external institutions that obtain the data by downloading NetCDF files to their local infrastructures. Within this framework, the ESGF has served as the primary federated infrastructure for the distribution of such datasets. More recently, cloud-based climate data repositories have emerged and demonstrated their effectiveness in enabling remote access to large-scale climate datasets. These platforms facilitate more efficient data sharing and significantly enhance collaboration among climate researchers and research initiatives.

Throughout these developments, NetCDF has increasingly been perceived as an inefficient format for remote data access. Consequently, prevailing narratives have favored alternative approaches, such as emerging formats like Zarr, or workarounds including Kerchunk sidecar files accompanying NetCDF datasets, as necessary to achieve acceptable performance in cloud-based repositories and remote access workflows. While these concerns are valid, the role of NetCDF's internal data structures in contributing to these performance limitations has been largely overlooked in the literature. In this work, we have addressed this gap by providing a detailed examination of the internal organization of NetCDF files and identifying the structural factors that cause performance issues. Furthermore, we have introduced tools designed to mitigate these limitations and have demonstrated that NetCDF can achieve competitive performance in both cloud-based environments and remote data access scenarios.

https://doi.org/10.5194/egusphere-2026-3249
Preprint. Discussion started: 16 June 2026

With these new insights in mind, we believe that future operational decisions will no longer depend on the specific storage format (i.e., NetCDF or Zarr), as both can achieve comparable performance when properly configured. Instead, future operational management will focus on identifying and applying the appropriate configuration. Our results show that performance is strongly influenced by the number of underlying objects within a dataset, leading to a trade-off between the number of NetCDF files or Zarr objects/shards and the characteristics of the storage system. Object storage systems appear to benefit from a larger number of individual objects, whereas parallel file systems tend to favor configurations that enable efficient random byte-range reads. This trade-off in the number of objects—whether files, shards, or chunks—is also a critical factor for the performance of aggregation technologies such as Kerchunk. Consequently, object granularity emerges as an additional consideration alongside the more traditional trade-off associated with chunk shape and size.

Current developments within CMIP and the ESGF increasingly promote the use of Kerchunk to facilitate cloud-based access to NetCDF datasets. However, the findings of this work demonstrate that the mere generation of a Kerchunk file fragmented NetCDF datasets is not sufficient to ensure efficient remote data access. Instead, performance depends critically on the internal structure of the NetCDF file, including factors such as the organization of B-tree nodes and the chosen chunking strategy. Only when these elements are appropriately configured can Kerchunk deliver proper efficient data access. At the same time, the maintenance of Kerchunk files introduces an additional layer of complexity within the data federation. For this reason, we advocate for a more sustainable approach based on tools that allow repacking or directly generate metadata consolidated NetCDF files, which organize the internal structure of NetCDF files to enable efficient remote access without the need for supplementary artifacts. This strategy confines maintenance overhead to the one-time process of restructuring files prior to publication. Collectively, these findings provide a basis for more informed decision-making within the climate research community regarding data storage practices, with the ultimate goal of enhancing accessibility and usability for end users.

Code and data availability. Code and data have been archived at Zenodo and are available at 10.5281/zenodo.19677471 (Cimadevilla, 2026). Kerchunk files are also available in the repository. The original NetCDF files have been obtained from the ESGF. Their repacked versions where obtained by applying *cmip7repack* (Hassell and Cimadevilla, 2025).

Author contributions. **Ezequiel Cimadevilla:** Investigation; methodology; software; visualization; writing – original draft; writing – review and editing.

Author contributions. **David Hassell:** Investigation; methodology; software; visualization; writing – original draft; writing – review and editing.

https://doi.org/10.5194/egusphere-2026-3249
Preprint. Discussion started: 16 June 2026

Author contributions. **Bryan N. Lawrence:** Investigation; methodology; software; visualization; writing – original draft; writing – review and editing.

Competing interests. The authors declare that they have no competing interests.

Acknowledgements. The article processing charges for this open-access publication were covered by the CSIC Open Access Publication Support Initiative through its Unit of Information Resources for Research (URICI).

https://doi.org/10.5194/egusphere-2026-3249
Preprint. Discussion started: 16 June 2026

References

Abernathey, R. P., Augspurger, T., Banihirwe, A., Blackmon-Luca, C. C., Crone, T. J., Gentemann, C. L., Hamman, J. J., Henderson, N., Lepore, C., McCaie, T. A., Robinson, N. H., and Signell, R. P.: Cloud-Native Repositories for Big Scientific Data, Computing in Science & Engineering, 23, 26–35, https://doi.org/10.1109/MCSE.2021.3059437, 2021.

Bayer, R. and McCreight, E. M.: Organization and maintenance of large ordered indexes, Acta Informatica, 1, 173–189, https://doi.org/10.1007/BF00288683, 1972.

Cimadevilla, E.: Why the relational data model matters for climate data management, Computers & Geosciences, 201, 105 931, https://doi.org/10.1016/j.cageo.2025.105931, 2025.

Cimadevilla, E.: New insights NetCDF (data and software), https://doi.org/10.5281/ZENODO.19677471, 2026.

Cinquini, L., Crichton, D., Mattmann, C., Harney, J., Shipman, G., Wang, F., Ananthakrishnan, R., Miller, N., Denvil, S., Morgan, M., Pobre, Z., Bell, G. M., Drach, B., Williams, D., Kershaw, P., Pascoe, S., Gonzalez, E., Fiore, S., and Schweitzer, R.: The Earth System Grid Federation: An open infrastructure for access to distributed geospatial data, in: 2012 IEEE 8th International Conference on E-Science, pp. 1–10, IEEE, Chicago, IL, USA, ISBN 978-1-4673-4466-1 978-1-4673-4467-8 978-1-4673-4465-4, https://doi.org/10.1109/eScience.2012.6404471, 2012.

Comer, D.: Ubiquitous B-Tree, ACM Computing Surveys, 11, 121–137, https://doi.org/10.1145/356770.356776, 1979.

Hassell, D. and Cimadevilla, E.: cmip7repack: Repack CMIP7 netCDF-4 datasets, https://doi.org/10.5281/ZENODO.17550919, language: en, 2025.

Hassell, D., Gregory, J., Blower, J., Lawrence, B. N., and Taylor, K. E.: A data model of the Climate and Forecast metadata conventions (CF-1.6) with a software implementation (cf-python v2.1), Geoscientific Model Development, 10, 4619–4646, https://doi.org/10.5194/gmd-10-4619-2017, 2017.

Lawrence, B. N., Cimadevilla, E., Nolf, W. D., Hassell, D., Helmus, J., Hodel, B., Maranville, B., Mühlbauer, K., and Predoi, V.: pyfive: A pure-Python HDF5 reader, Journal of Open Source Software, 11, 9688, https://doi.org/10.21105/joss.09688, 2026.

Rew, R., Hartnett, E., and Caron, J.: NetCDF-4: Software implementing an enhanced data model for the geosciences, in: 86th AMS Annual Meeting, https://www.scopus.com/inward/record.uri?eid=2-s2.0-77949308478&partnerID=40&md5=0903d47dc1ecf01f2472095fa3191055, type: Conference paper, 2006.