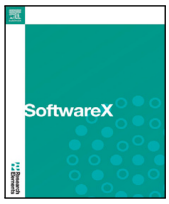
SoftwareX 31 (2025) 102269

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx



Original Software Publication

NetCDFaster: Optimizing NetCDF data querying and geo-visualization using high-performance machine learning

Zhenlei Song [a], Zhe Zhang [a,*], Alan Sussman [b], Yuhang Xie [a], Jorge Brenner [c,d], Jikun Liu [a]

[a] Department of Geography, Texas A&M University, College Station, TX, USA
[b] Department of Computer Science, University of Maryland, College Park, MD, USA
[c] Department of Oceanography, Texas A&M University, College Station, TX, USA
[d] Gulf of America Coastal Ocean Observing System, College Station, TX, USA

ARTICLE INFO

Keywords:
NetCDF
Machine learning
Data visualization
Reading time optimization
Scientific workflow
Cyberinfrastructure

ABSTRACT

Extracting and visualizing multidimensional subsets from large NetCDF datasets is often repetitive, time-consuming, and technically demanding. Existing tools typically trade off between performance and usability, either optimizing data retrieval at the cost of user experience or providing user-friendly interfaces that struggle with performance issues. NetCDFaster bridges this gap by integrating a lightweight CatBoost classifier model into a web-based platform that automatically selects optimal data retrieval strategies. The benchmark results demonstrate that NetCDFaster reduces end-to-end retrieval times by more than 80%, allowing researchers to efficiently explore data subsets and visualize results. This significantly enhances the speed and effectiveness of scientific workflows involving NetCDF data.

Metadata

Current code version	v0.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-25-00210
Permanent link to Reproducible Capsule	Frontend: https://hub.docker.com/r/songzl8/netcdfaster-frontend/tags; Backend: https://hub.docker.com/r/songzl8/netcdfaster-backend/tags
Legal Code License	MIT License
Code versioning system used	git
Software code languages, tools, and services used	Python for backend; node.js and pnpm for frontend
Compilation requirements, operating environments & dependencies	Python3.8, node.js, pnpm, netCDF4, flask, gunicorn
If available Link to developer documentation/manual	Frontend: https://github.com/TAMUCIDI/netCDFaster-frontend/blob/main/README.md; Backend: https://github.com/TAMUCIDI/netCDFaster-backend/blob/main/README.md
Support email for questions	songzl@tamu.edu

1. Motivation and significance

Large multidimensional datasets in climatology, oceanography, and related fields are commonly stored in NetCDF format [1–3], but extracting and visualizing information from these files remains a key scientific challenge [4,5]. Researchers often face repetitive and time-consuming tasks, such as writing custom scripts to extract data subsets and then switching to separate tools for visualization. On the one hand, high-performance I/O libraries (e.g., Parallel NetCDF [6]) and distributed frameworks (e.g., SciHadoop [7]) can accelerate data access in specialized environments. However, such tools require complex setup (parallel file systems, distributed clusters) and typically lack user-friendly interfaces, making them impractical for many scientists [8–10]. While graphical applications for NetCDF visualization exist, they generally utilize default data access methods lacking optimization, which significantly hampers performance on large datasets [11]. There remains a critical gap: the lack of lightweight, web-based tools that integrate

* Corresponding author.
E-mail addresses: songzl@tamu.edu (Z. Song), zhezhang@tamu.edu (Z. Zhang), als@cs.umd.edu (A. Sussman), xieyuhang1997@tamu.edu (Y. Xie), Jorge.brenner@gcoos.org (J. Brenner), jikun@tamu.edu (J. Liu).

https://doi.org/10.1016/j.softx.2025.102269
Received 30 March 2025; Received in revised form 20 June 2025; Accepted 8 July 2025
Available online 9 August 2025

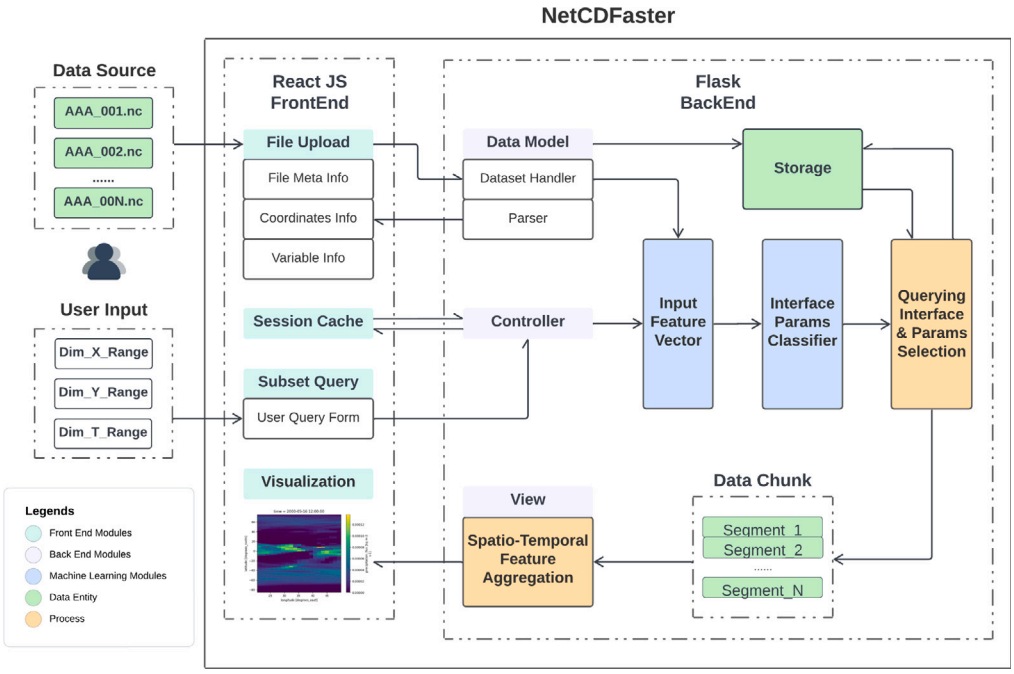


Fig. 1. System-level design overview of the NetCDFaster architecture.

fast, optimized data retrieval with interactive visualization in a unified, seamless workflow.

NetCDFaster fills this gap by offering a streamlined, web-based solution for NetCDF data access and visualization. Its user-friendly interface supports dataset upload, variable subsetting, and real-time plotting, all within a single platform. Behind the scenes, it uses a lightweight machine learning model (CatBoost [12]) to automatically determine the most efficient data reading interface and parameter settings for each query. By automatically tuning chunking strategies, caching behavior, and API selection, NetCDFaster enables fast data access without requiring expert knowledge or additional coding.

This unified approach delivers substantial and measurable performance improvements. Benchmark experiments show that retrieval times for identical NetCDF queries can vary by more than 50 times, depending on the interface and parameter configurations, underscoring the importance of optimized access strategies. By consistently selecting near-optimal settings, NetCDFaster reduces end-to-end data access times by over 80%, achieving near-peak performance with minimal overhead. In practical terms, this enables researchers to move from raw data to visualization much more quickly, accelerating the analysis cycle and supporting more interactive, insight-driven exploration of large-scale geoscientific datasets.

2. Software description

2.1. Software architecture

NetCDFaster's architecture is modular, splitting the front-end and back-end into dedicated components to reduce communication overhead and boost performance, as shown in Fig. 1. NetCDF data files supported in this project include files in NetCDF3, NetCDF4 formats, and in Climate and Forecast (CF), Cooperative Ocean-Atmosphere Research Data Service (COARDS) data conventions [13,14]. The web-based front-end handles all user interactions, built with Next.js [15] and React [16] for a responsive interface. This enables users to upload NetCDF files, inspect metadata (including coordinates and variables), and define subset queries. Heavy data processing is delegated to the Flask [17]-based back-end, which parses uploaded files and compiles the characteristics of the data set alongside the user's specified ranges into an input feature vector for machine learning. A CatBoost [12] classifier model then analyzes this feature vector and predicts the optimal method to execute the query (for example, suggesting whether to use sequential or parallel reads), as shown in Fig. 2. Using these recommendations, the back-end performs NetCDF data extraction (leveraging parallel computation through Dask [18] when appropriate) to quickly retrieve and aggregate the requested data. The resulting data subset is finally returned to the front-end for visualization, allowing users to efficiently explore the queried multidimensional data in real-time. The interface parameter set refers to the specific software interface and configuration used to execute subset querying tasks. The sequential interface represents the default querying method provided by Xarray, which operates in a single-threaded manner. In contrast, the MT_All and MT_XX interfaces correspond to eight distinct parallel multi-threading configurations, each defined by different combinations of Dask-based settings. These multi-threaded strategies are detailed in the Table 2. The technical details are as follows:

1. **Front-end:** The client interface is built with Next.js and React, providing a dynamic web interface for data upload and visualization.
2. **Back-end:** The server side uses Python's Flask framework to handle API requests and coordinate data processing logic. The "model" layer performs efficient NetCDF reading using Dask for parallelized queries.
3. **Machine Learning Module:** A CatBoost classifier is integrated into the back-end to optimize the execution of query tasks. Trained on NetCDF metadata and historical query behavior, the model intelligently recommends optimal query parameters, selects the fastest data access interface, and determines the most suitable Dask multi-threading policy based on the dataset's properties and user-defined inputs.
4. **Deployment:** NetCDFaster is containerized using Docker [19], allowing the application and all its dependencies to be bundled together for consistent deployment across diverse computing environments. The container is deployed on DigitalOcean [20], a software-as-a-service (SaaS) platform, providing scalability and resilience under heavy workloads.

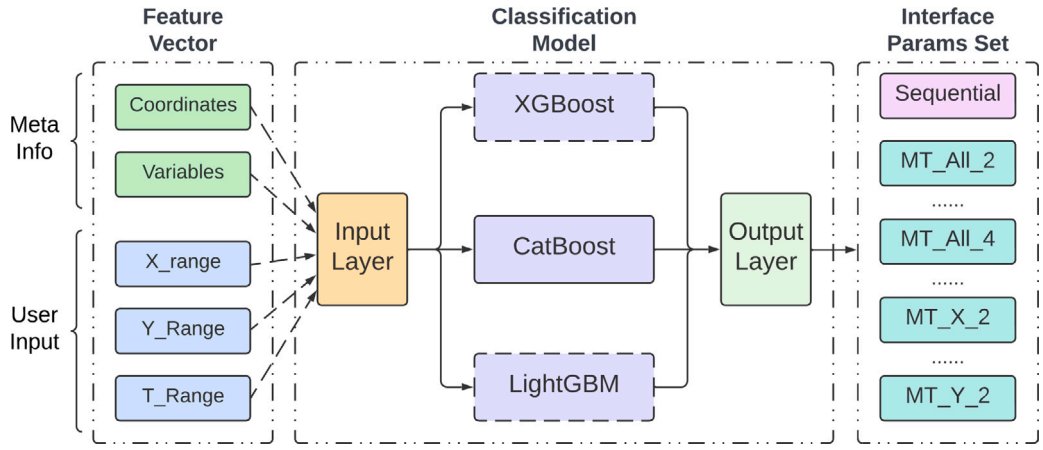


Fig. 2. Illustration of the dynamic interface parameter classifier in NetCDFaster.

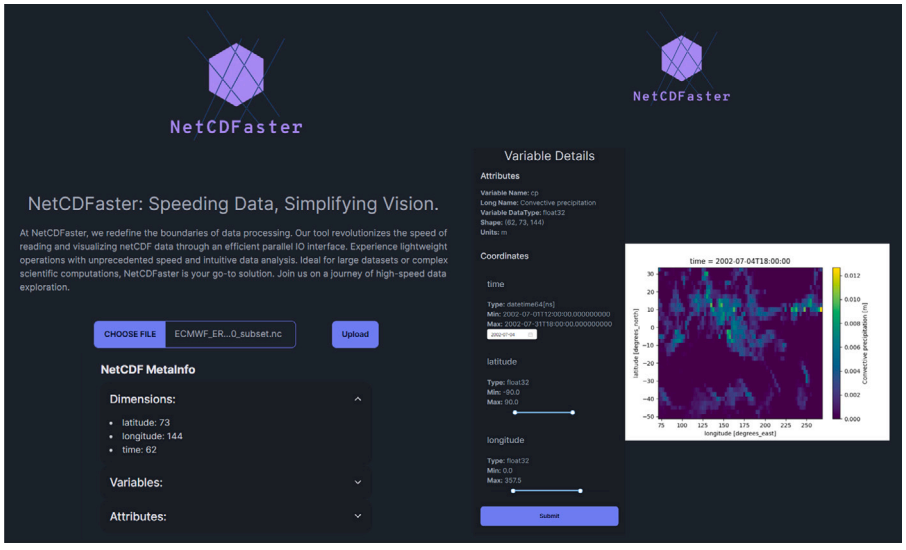


Fig. 3. Illustration of NetCDFaster web application user interface design.

Table 1
Functions NetCDFaster supported.

Functionalities	NetCDFaster
File Upload	✓
Metainfo Preview	✓
Custom Coord Range Query	✓
Dynamic Querying Policy Choice	✓
Online Visualization	✓

Several stages of NetCDFaster's development, such as environment setup, training data collection, and read-time evaluation, were carried out on high-performance computing systems. High-performance computing environments offered key advantages that enhanced the tool's performance and development efficiency. First, the distributed operating and file systems in high-performance computing settings provided additional speedups for multi-threading read operations using Dask within Xarray. Second, the module-based environment management system simplified dependency and version control, which is critical for ensuring the stability and reproducibility of the platform.

2.2. Software functionalities

Table 1 illustrates the major functionalities of NetCDFaster. 1

2.2.1. Data uploading & parsing

NetCDFaster provides an interactive web interface for importing NetCDF datasets, retrieval, and visualization. Users can upload NetCDF files directly through the browser. Upon upload, the system immediately parses the file's metadata, including spatial coordinates, variable definitions, and attributes, without loading the entire dataset into memory.

Unlike conventional metadata viewers that display raw attribute lists, NetCDFaster organizes metadata hierarchically:

- **File-level**: Format version (NetCDF3/4), conventions (CF/COARDS [13,14]), global attributes
- **Variable-level**: Data types, chunking patterns, compression status, valid ranges
- **Coordinates**: Dimension lengths, coordinate values

NetCDFaster's UI/UX designs enabled users to visually verify coordinate ranges and identify target variables by their coordinates, allowing them to pre-configure subset parameters before executing querying tasks.

2.2.2. Query execution & optimization

When executing a data query, NetCDFaster enables users to define spatial (longitude and latitude coordinate) and temporal (time coordinate) ranges, through the interface. Upon query submission, a CatBoost classifier predicts the optimal data retrieval policy from 9

Table 2
Interface parameter set explanations.

Interface Params Name	Parallel Setting	Explanation
`Sequential`	None	Xarray's default querying setting with no parallel chunking
`MT_All_2`	time: 2 lon: 2 lat: 2	multi-threading with 2 chunks on all coordinates
`MT_All_4`	time: 4 lon: 4 lat: 4	multi-threading with 4 chunks on all coordinates
`MT_X_2`	time: – lon: 2 lat: –	multi-threading with 2 chunks only on longitude
`MT_Y_2`	time: – lon: – lat: 2	multi-threading with 2 chunks only on latitude
`MT_T_2`	time: 2 lon: – lat: –	multi-threading with 2 chunks only on time
`MT_XY_2`	time: – lon: 2 lat: 2	multi-threading with 2 chunks on longitude and latitude
`MT_XT_2`	time: 2 lon: 2 lat: –	multi-threading with 2 chunks on longitude and time
`MT_YT_2`	time: 2 lon: – lat: 2	multi-threading with 2 chunks on latitude and time

candidate strategies (1 sequential and 8 multi-threading configurations with specific chunking parameters, as shown in Table 2). This classifier was trained on over 55,000 query tasks across more than 200 NetCDF datasets (including NetCDF3/4 datasets in CF [13] and COARDS [14] conventions), using file attributes, variable metadata, and coordinate ranges as input features to recommend the fastest extraction method. The backend then executes the subset extraction with the selected policy. When the classifier identifies multi-threading as optimal NetCDFaster partitions the task via Dask [18] across CPU cores. Note that policy selection relies on empirical patterns; explicit decision thresholds are unavailable due to model interpretability limits. This adaptive pipeline maximizes efficiency without requiring manual performance tuning.

2.2.3. Visualization & interaction

Once the data subset is retrieved, NetCDFaster visualizes the results using the same web interface — no separate tools or downloads required. The front-end GUI is split into two primary sections: a control panel for user input on the left, and an output display on the right that renders the requested data subset. The extracted subset can be explored through interactive plots that respond in real-time to user interactions. For example, if a user requests a spatio-temporal subset of a climate variable, the application can display an interactive map where one can zoom and pan to examine regional details, or a time-series chart with tool tips to inspect precise values. These built-in visualization capabilities allow users to investigate patterns and anomalies in the data subset on the fly. By integrating data retrieval with immediate graphical exploration, NetCDFaster eliminates the need for external visualization software in the workflow. This lowers the barrier for scientists to analyze their data, aligning with open-science goals of making geospatial data more accessible and readily usable straight from the source.

Although there are many NetCDF data file hosting platforms available [21,22], NetCDFaster offers a different user experience. The differences between its features and those of other widely used data hosting platforms are shown in Table 3.

Table 3
Function comparison between NetCDFaster and other platforms.

	THREDDS [21]	ERDDAP [22]	NetCDFaster
Meta Info Preview	✓	✓	✓
Coordinate Subset Querying	✓	✓	✓
Data Hosting	✓	✓	×
Dynamic Query Interface Parameters	×	×	✓
Online Visualization	×	Only with Griddap	✓

3. Illustrative examples

In this section, we demonstrate how to perform a multidimensional range query on a variable from a sample NetCDF file and visualize the resulting subset using the web application. First, we launch the front-end and back-end containers on the deployment server. Then, we upload the example NetCDF file to NetCDFaster. The test NetCDF file [23] contains the surface data for July 2002 from the ECMWF Reanalysis v5 (ERA5) [24] 40 Years Re-Analysis, daily fields. NetCDFaster file parsing module will parse the meta information first and return variable attributes and coordinate ranges back to the front-end (as shown in the left subplot of Fig. 3), marked as feature set 1. Then users can specify their interested variable and the corresponding coordinate ranges (as shown in the widgets of the right subplot in Fig. 3) to query and visualize, marked as feature set 2. The front-end passes an HTTP POST request (composed of feature sets 1 and 2) to the back-end querying module. The extracted features are passed to the machine learning module, which predicts the optimal querying interface along with the corresponding parameters. The querying module then uses this recommendation to retrieve the specified data subset, enhancing read efficiency. Finally, the queried data is returned to the front-end for visualization, as illustrated in the right subplot of Fig. 3.

4. Impact

NetCDFaster fundamentally transforms multidimensional scientific data workflows by dramatically accelerating end-to-end processing. To quantify its advantages, we benchmarked NetCDFaster against two standard approaches: the serial-only NetCDF4 Python API [25] and the multi-interface Xarray library. Our evaluation leveraged more than 200 variables from ECMWF Reanalysis v5 (ERA5) [24] and Global Ocean Data Assimilation System (GODAS) [26] datasets (2 MB–2.5 GB), filtered to retain only three-dimensional variables (time/longitude/latitude). For each variable, we generated 50 randomized subset queries spanning diverse coordinate ranges, executing more than 10,000 tasks through three methods: (1) NetCDF4's baseline APIs [25]; (2) 9 Xarray's interfaces (1 sequential and 8 parallel configurations with distinct chunking strategies, detailed in Table 2), and (3) NetCDFaster dynamically selects the optimal Xarray interface using a CatBoost classifier.

The results showed promising performance by comparing with NetCDF4's serial approach. For large datasets (larger than 1 GB), NetCDFaster has demonstrated median speedups of 6.96 times, as illustrated in the Fig. 5. For smaller datasets (smaller than 500MB), NetCDFaster performed close to NetCDF4, as shown in Fig. 4. These findings highlight NetCDFaster's superior parallel acceleration capabilities in processing large datasets, which are not supported by NetCDF4.

When compared to Xarray's full suite of interfaces, NetCDFaster's end-to-end time t^{e2e} (comprising model prediction time t^{p}, and optimized read time $\tilde{t^{r}}$ approached the theoretical optimum t^{min} (the

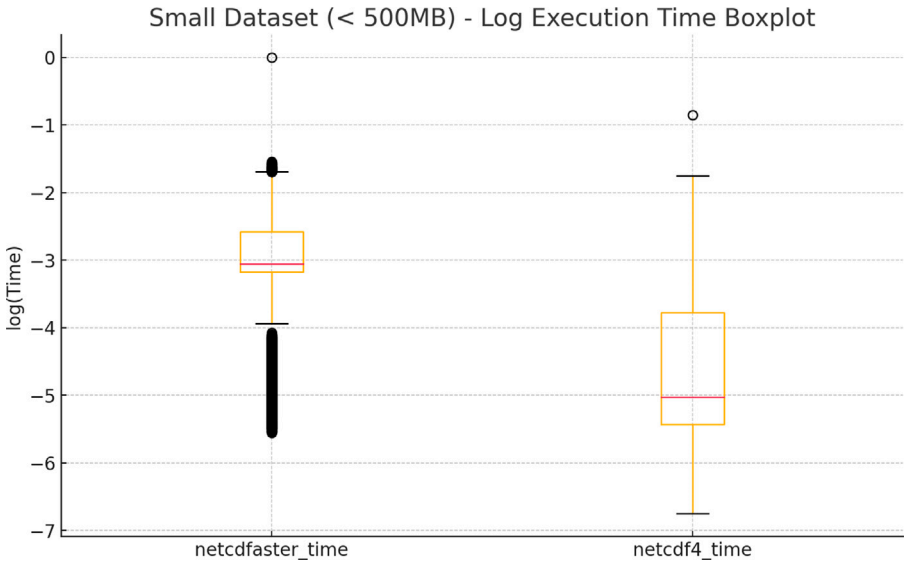


Fig. 4. Comparison of time cost (log scale) between NetCDFaster and NetCDF4 Python APIs on small datasets (≤ 500MB).

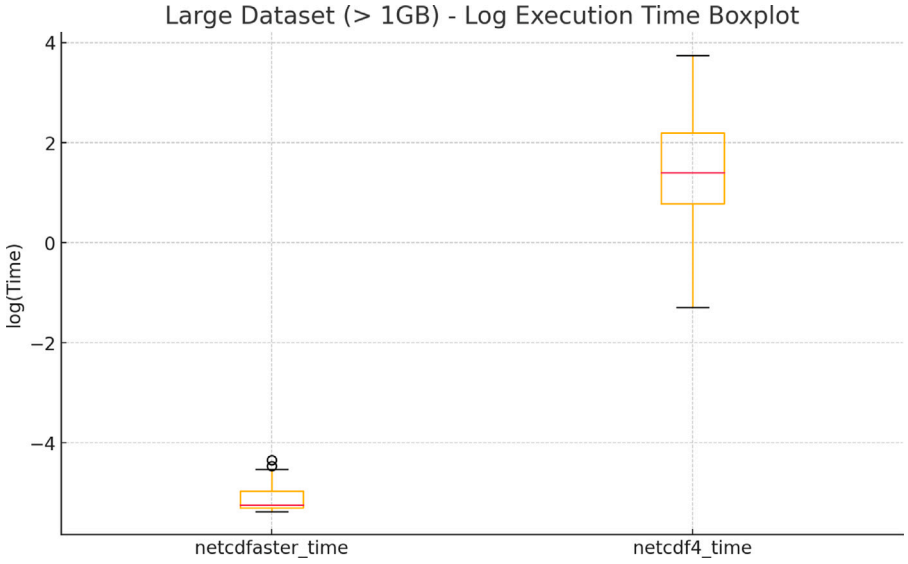


Fig. 5. Comparison of time cost (log scale) between NetCDFaster and NetCDF4 Python APIs on large datasets (≥ 1 GB).

fastest of Xarray's nine strategies), yielding a median ratio t^{e2e}/t^{min} of 1.08. Crucially, NetCDFaster outperformed Xarray's average performance t^{mean} by over 10% in 90% of all queries and by up to 80% in worst-case scenarios, as shown in Fig. 6, while adding negligible prediction overhead (0.2–6 ms).

$$\begin{aligned} t^{e2e} &= \tilde{t}^r + t^p \\ t^{min} &= \min_{i\in[0,8]} t_i^r \\ t^{max} &= \max_{i\in[0,8]} t_i^r \\ t^{mean} &= \sum_{i=0}^{8} t_i^r/9 \end{aligned} \tag{1}$$

Where, t_i^r is the reading time cost using the ith Xarray's interface in Table 2; t^p is the prediction time cost using the optimal model; $\tilde{t}^r$ is the reading time using the predicted class.

NetCDFaster enables researchers to iterate on data exploration six to nine times faster than with conventional tools. By leveraging machine learning to automate interface selection, it eliminates the need for manual tuning while delivering near-peak theoretical performance. More than just a speed enhancement, NetCDFaster offers a web-based platform that abstracts complex I/O optimizations, integrates accelerated data retrieval, and supports real-time visualization—positioning it as a practical tool for daily scientific workflows. Crucially, NetCDFaster is built to exploit high-performance computing (HPC) environments: it utilizes distributed file systems for Dask-based multi-threading, ensures version compatibility through robust module dependency management, and supports large-scale training and evaluation via Slurm job scheduling. This work has been validated across full HPC workflows, from processing to model deployment, which enables integration into the modern scientific cyberinfrastructure (CI). This extensibility empowers collaborative, data-intensive research, allowing scientists to focus on discovery over data-handling bottlenecks while taking advantage of scalable compute resources.

5. Conclusions

Previous approaches, such as Parallel NetCDF[6], NetCDF4 APIs [25], in multidimensional data range indexing are often constrained

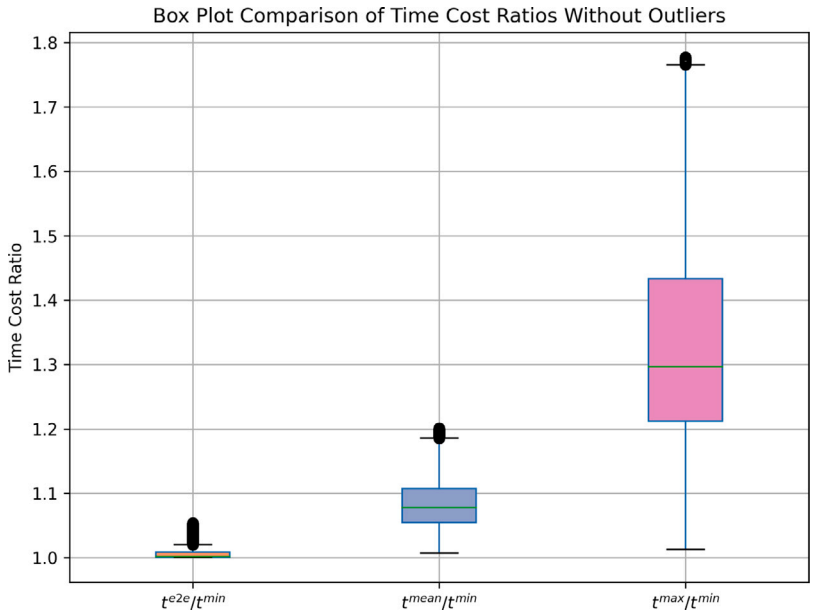


Fig. 6. Box plot comparison of time cost ratios: Total time cost to minimum time cost (t^{e2e}/t^{min}), average time cost to minimum time cost (t^{mean}/t^{min}), and maximum time cost to minimum time cost (t^{max}/t^{min}) using the NetCDFaster optimal classifier.

by a key limitation: they are typically optimized for a narrow class of tasks, leading to inefficiencies when applied across diverse configurations. NetCDFaster addresses this challenge by offering a CI-integrated, web-based platform that accepts flexible task configurations and user inputs. At the core of NetCDFaster's design is a CatBoost classification model that predicts the optimal data access interface and corresponding parameter settings, significantly reducing read times across diverse multidimensional indexing scenarios. This approach introduces a novel, machine learning–driven methodology for optimizing read-time performance in complex data access workflows. By automating interface selection and tuning, NetCDFaster streamlines the analysis pipeline, enabling faster and more efficient data exploration and visualization across a broad range of scientific domains.

The current version of NetCDFaster is optimized for rectilinear grids with 1D longitude and latitude coordinates. This design choice reflects the reality that most publicly available NetCDF datasets, particularly those following CF or COARDS conventions, use simple 1D coordinate systems at moderate spatial resolutions. Given that our primary users work with large-scale, regularly gridded atmospheric, oceanographic, and geospatial data, efficient 1D slicing was prioritized for this initial release. We plan to incorporate full curvilinear support in a future version, including front-end visualization enhancements and backend slicing logic tailored to 2D coordinate structures.

Finally, Dask is employed for multi-threading querying when the CatBoost classification model identifies it as the optimal interface for a given task. However, there is currently no explicit threshold or rule for when Dask is chosen, as the current CatBoost classifier model cannot provide interpretable boundaries for the interface selection. In the future, we plan to enhance the model's interpretability to offer more transparent decision-making.

CRediT authorship contribution statement

Zhenlei Song: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Zhe Zhang:** Writing – review & editing, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization. **Alan Sussman:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Yuhang Xie:** Writing – review & editing. **Jorge Brenner:** Writing – review & editing. **Jikun Liu:** Writing – review & editing, Investigation.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Zhe Zhang reports financial support was provided by National Science Foundation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This material is based on work funded by the National Science Foundation under Grant No. #2137684 #2019129 #2321069 #2339174 #2519476 #2526748.

References

[1] Rew R, Davis G. NetCDF: an interface for scientific data access. IEEE Comput Graph Appl 1990;10:76–82. http://dx.doi.org/10.1109/38.56302.

[2] Gordov E, Shiklomanov A, Okladnikov I, Prusevich A, Titov A. Development of Distributed Research Center for analysis of regional climatic and environmental changes. IOP Conf Ser: Earth Environ Sci 2016;48:012033. http://dx.doi.org/10.1088/1755-1315/48/1/012033.

[3] Guo Q, Zhang Y, He Z, Min Y. Web-based data integration and interoperability for a massive spatial-temporal dataset of the Heihe river basin escience framework. Adv Meteorol 2015;1:1–13. http://dx.doi.org/10.1155/2015/982062.

[4] Rieder B. Studying facebook via data extraction. In: Proceedings of the 5th annual ACM web science conference. 2013, http://dx.doi.org/10.1145/2464464.2464475, URL https://dl.acm.org/citation.cfm?id=2464475.

[5] Singh SS, Kumawate B, Vaishnav A, Kumar L, Bushra S, Teitsana DS. Exploring data visualization techniques for large datasets. In: 2024 1st international conference on advances in computing, com- munication and networking. IEEE; 2024, p. 1318–23. http://dx.doi.org/10.1109/ICAC2N63387.2024.10895382.

[6] Li J-W, Liao W-k, Choudhary A, Ross R, Thakur R, Gropp W, Latham R, Siegel AF, Gallagher B, Zingale M. Parallel netCDF. In: SC '03: Proceedings of the 2003 ACM/IEEE conference on supercomputing (11 2003). 2003, http://dx.doi.org/10.1145/1048935.1050189.

[7] Buck JB, Watkins N, LeFevre J, Ioannidou K, Maltzahn C, Polyzotis N, Brandt S. SciHadoop. In: SC '11: Proceedings of 2011 international conference for high performance computing, networking, storage and analysis (11 2011). 2011, http://dx.doi.org/10.1145/2063384.2063473.

[8] Feng K, Sun X, Yang X, Zhou S. SciDP: Support HPC and big data applications via integrated scientific data processing. In: 2018 IEEE International Conference on Cluster Computing (CLUSTER). 2018, p. 114–23. http://dx.doi.org/10.1109/CLUSTER.2018.00023.

[9] Wilson B, Palamuttam R, Whitehall K, Mattmann C, Goodman A, Boustani M, Shah S, Zimdars P, Ramirez P. SciSpark: Highly interactive inmemory science data analytics. In: 2016 IEEE International Conference on Big Data (Big Data). 2016, p. 2964–73. http://dx.doi.org/10.1109/BigData.2016.7840948.

[10] Biookaghazadeh S, Zhou S, Zhao M. Kaleido: Enabling efficient scientific data processing on bigdata systems. In: 2017 International Conference on Networking, Architecture, and Storage (NAS). 2017, p. 1–10. http://dx.doi.org/10.1109/NAS.2017.8026864.

[11] Su Y, Agrawal G, Woodring J. 2012 41st international conference on parallel processing. In: Indexing and parallel query processing support for visualizing climate datasets. 2012, p. 249–58. http://dx.doi.org/10.1109/ICPP.2012.33.

[12] Prokhorenkova LO, Gusev G, Vorobev A, Dorogush AV, Gulin A. CatBoost: unbiased boosting with categorical features. 2017, http://dx.doi.org/10.48550/arxiv.1706.09516, ArXiv (Cornell University).

[13] Eaton B, Gregory J, Drach B, Taylor K, Hankin S, Caron J. NetCDF climate and forecast (CF) metadata conventions. CF Community 2024. http://dx.doi.org/10.5281/zenodo.14275599, URL https://zenodo.org/records/14275599.

[14] Laboratory PME. COARDS NetCDF conventions | science data integration group - ferret support. 2024, URL https://ferret.pmel.noaa.gov/Ferret/documentation/coards-netcdf-conventions.

[15] Vercel. Next.js by vercel - the react framework. 2024, URL https://nextjs.org/.

[16] React. React – a JavaScript library for building user interfaces. 2022, URL https://reactjs.org/.

[17] Flask. Welcome to flask — Flask documentation (2.0.x). 2024, URL https://flask.palletsprojects.com.

[18] core developers D. Dask | scale the python tools you love. 2024, URL https://www.dask.org/.

[19] Merkel D. Docker: lightweight linux containers for consistent development and deployment. Linux J 2014;2014(239).

[20] DigitalOcean. API overview | DigitalOcean documentation. 2025, URL https://docs.digitalocean.com/reference/api/.

[21] UCAR. Unidata | THREDDS data server (TDS). 2023, URL https://www.unidata.ucar.edu/software/tds/.

[22] Wilson C, Robinson D, Simons RA. Erddap: Providing easy access to remote sensing data for scientists and students. In: IGARSS 2020- 2020 IEEE international geoscience and remote sensing symposium. 2020, http://dx.doi.org/10.1109/igarss39084.2020.9323962.

[23] UNIDATA. Example netcdf files. 2023, URL https://www.unidata.ucar.edu/software/netcdf/examples/files.html.

[24] Hersbach H, Bell B, Berrisford P, Hirahara S, Horányi A, Muñoz-Sabater J, Nicolas J, Peubey C, Radu R, Schepers D, Simmons A, Soci C, Abdalla S, Abellan X, Balsamo G, Bechtold P, Biavati G, Bidlot J, Bonavita M, Chiara G, Dahlgren P, Dee D, Diamantakis M, Dragani R, Flemming J, Forbes R, Fuentes M, Geer A, Haimberger L, Healy S, Hogan RJ, Hólm E, Janisková M, Keeley S, Laloyaux P, Lopez P, Lupu C, Radnoti G, Rosnay P, Rozum I, Vamborg F, Villaume S, Thépaut J. The ERA5 global reanalysis. Q J R Meteorol Soc 2020;146. http://dx.doi.org/10.1002/qj.3803.

[25] UCAR. netCDF4 API documentation. 2019, URL https://unidata.github.io/netcdf4-python/.

[26] Behringer DW, Ji M, Leetmaa A. An improved coupled model for ENSO prediction and implications for ocean initialization. Part I: The ocean data assimilation system. Mon Weather Rev 1998;126:1013–21. http://dx.doi.org/10.1175/1520-0493(1998)126<1013:aicmfe>2.0.co;2.