Kilometer-scale climate data analysis at global scale: Technical infrastructure for a distributed multi-model hackathon

Florian Ziemen[1], Lukas Kluft[2], Tobias Kölling[2], Mark Muetzelfeldt[3], Sam Greenfield[4], Andrew Gettelman[5], Fabian Wachsmann[1], and Orhan Eroglu[6]

[1]German Climate Computing Centre (DKRZ), Hamburg, Germany
[2]Max Planck Institute for Meteorology, Hamburg, Germany
[3]University of Reading, Reading, UK
[4]Australia
[5]PNNL, ...
[6]NCAR, Boulder, CO, USA

Correspondence: Florian Ziemen (ziemen@dkrz.de)

Abstract. We present the results and technical achievements of a global, multi-model intercomparison hackathon involving over 600 participants across 10 teams. Despite minimal dedicated resources and a largely new technology stack for most teams, we successfully standardized and remapped outputs from a diverse set of regional and global climate models to a common HEALPix grid, storing the data in Zarr format and indexing it with Intake 0.7. This unified approach enabled all participating teams—regardless of background or resources—to access and analyze the same datasets efficiently. A common Python-based stack, deployed via JupyterHub by most teams, facilitated collaborative analysis and reproducibility. Our experience demonstrates the feasibility and benefits of rapid, large-scale, distributed climate data analysis using modern open-source tools and cloud-ready data formats.

1 Introduction

Kilometer-scale global modeling is driving a revolution in climate science. Global models are finally able to explicitly represent deep convection: statistical descriptions of cumulonimbus clouds within a grid cell are no longer needed. Instead, these new models begin to resolve large-scale turbulence in both the atmosphere and ocean. This is why this new model generation is often called *storm-resolving*. The revolution also extends to the sheer amount of data in a single horizontal field. Typical CMIP models operate at a scale of 100 km and usually employ some variant of a latitude/longitude grid. For example, the MPI-ESM High Resolution CMIP6 setup (MPI-ESM-HR) contained $384 \times 192 = 73{,}728$ grid cells. In contrast, the 2.5 km resolution (*R2B10*) ICON setup covers the Earth with an icosahedral grid consisting of $20 \times 4^{10} = 83{,}886{,}080$ grid cells. This factor of 1000 increase in field size, combined with the added complexity of a non-Cartesian (often called *unstructured*) grid, poses significant challenges to established workflows. This change is exemplified by a typical global map for a presentation: with 1500×670 pixels, the switch is

from 14 pixels per grid cell to each pixel representing 84 grid cells (on average) for our two example setups. The explosion in data volume is not limited to climate science; similar issues are observed in bio-imaging and other fields (CITATION NEEDED).

With the increased adoption of kilometer-scale global modeling in the late 2010s came the rise of hackathons in climate modeling. Originating as rapid software prototyping, weekend-style activities around 2000 OpenBSD (2026); Rys (2021); Irani (2015)], the idea of gathering for an intensive period of software development spread in all directions, including civic and corporate hacking events. The climate data analysis variant typically consists of multi-day events where scientists meet and collaborate during normal office hours, learning workflows together. These events tend to focus less on developing a single piece of code and more on using the same or similar datasets for analysis. They have started to replace traditional project meetings, which focused on presentations of past achievements, with collaborative get-togethers that connect the community across research institutions and foster close links between technical and scientific teams. Examples of such events include the ESiWACE/DYAMOND hackathon in June 2019 (Neumann, 2019), which accompanied the first DYAMOND intercomparison of global storm-resolving models. After a break in 2020, enforced by COVID-related restrictions, nextGEMS (nextGEMS consortium), an H2020 project aimed at the development of storm-resolving models, initiated a series of hackathons. In these hackathons, usually tied to new experiment cycles, we iteratively updated data handling and workflows to improve the scientists' experience when working with the data. Major advances in user-friendliness came with the introduction of catalogs for data discovery, the switch to the HEALPix grid, spatiotemporal chunking in storage, Zarr for a unified view of large datasets, and the provision of a resolution hierarchy. The details of these changes will be discussed in later sections, as they were key to the decisions made in preparing for the global hackathon. In parallel to nextGEMS, EERIE (EERIE consortium) conducted hackathons that, while bringing people together in the same place, used datasets spread across multiple European data centers. Both, nextGEMS and EERIE built user workflows on catalogs, shared Python analysis environments and JupyterHub, facilitating the exchange of analysis scripts among scientists.

While there is a long tradition of sharing data across research centers (e.g., ESGF, GLOBUS), most of these approaches were built around creating a copy of (a subset of) the dataset at the destination site, and then analyzing this copy. In the case of petabyte-scale datasets, this becomes problematic. In contrast, the current trend of cloud-native datasets suggests loading data on demand for analysis. This led to the development of the eerie.cloud server, providing data hosted on the DKRZ parallel file system to users outside the center. (-> Fabi)

The open catalogs and data are more related to intercomparison projects than traditional research projects that tend to keep the data private until the first publications are written. They were therefore very well suited for the global hackathon and the connected DYAMOND phase 3 intercomparison. This intercomparison is the third in a row of intercomparison projects that focus on comparing storm-resolving models. In contrast to the coupled model intercomparison project (CMIP), where data is heavily standardized (CMOR), the first two phases of DYAMOND focused on making contributing easy by not requiring any standardization of the output. In the first phase, DYAMOND

Summer (Stevens et al., 2019) dealing with data formats, model grids and variables was left to the scienists analyzing the data. In the second phase, DYAMOND Winter (doi:10.1186/s40645-024-00668-1) contributions still were come-as-you are, but DKRZ dedicated resources to some minimal standardization of the data, making it easier to analyze the datasets. In the third phase (Takasuka et al., 2024), more standardization was requested, and all datasets were remapped to the HEALPix grid for the global hackathon. This grid is in conflict to the data request paper, where lat/lon was requested, and very much representative of the dynamic nature of the DYAMOND intercomparisons.

The Global hackathon was run with self-organized nodes to minimize overhead resulting from the different ways of running activities in different countries, but with a shared technical stack to maximize opportunities for scientific collaboration. In the run-up to the hackathon, regular video conferences and a shared chat system helped harmonize planning, while technical teams collaborated closely to provide datasets with similar structure and handling. All datasets were remapped to the HEALPix grid, provided as big, coherent zarr stores, and listed in a shared catalog that supplemented locally available copies with online datasets. A coherent python-based software stack enabled code sharing across locations. Cross-node science topics, such as MCS tracking (Team, b), connected participants across locations and fostered collaborative analysis. In addition, cross-cutting technical activities like AI representations of the model output allowed the participants to push the envelope of technical possibilities.

The technical side of the hackathon was made possible by an ad-hoc technical team that at some nodes simply consisted of the scientific PI, and at other nodes of technical staff that was pulled in from related activities or staff resources. Due to the global and ad-hoc nature of this hackathon, there were no in-person meetings prior to the hackathon, but the coordination happened in a series of video conferences, and using Mattermost as a chat system and github for code sharing. Some scientists and technical staff joined other nodes during the hackathon to foster exchange. In the following, we will describe the run-up and technical aspects of the execution of the global hackathon in greater detail.

2 The technical stack

2.1 The catalog

A central objective of the technical stack was to streamline the analysis and comparison of datasets. For most participants, the data catalog (Team, a) along with its machine-readable *intake* representation served as the primary entry point. Key requirements for the machine-readable catalog included ease of data loading at any node, support for dataset variants (such as different spatial and temporal aggregations), and the ability to handle multi-file datasets, as not all data had been consolidated into single Zarr stores for each aggregation due to time constraints. The catalog also needed to be straightforward to create and maintain.

The main options considered for the catalog were Intake and STAC. While STAC has become the de facto standard for satellite and other observational data, it currently lacks convenient support for dataset variants and for loading multi-file datasets as a single object. Intake exists in two major versions: Intake 2.0, which allows for describing dataset transformations during loading but has a complex description language and lacks support for dataset variants; and Intake 0.7, which supports dataset variants and multi-file datasets but is technically incompatible with Intake 2.0. For this hackathon, Intake 0.7 was selected due to its robust support for variants and multi-file datasets. For details on the catalog creation see below in section ... The choice of Intake 0.7 enabled participants to obtain an xarray dataset with a simple four-line Python script:

```
import intake
url = "https://digital-earths-global-hackathon.github.io/catalog/catalog.yaml"
cat = intake.open_catalog(url)["online"]
ds = cat["icon_d3hp003"](zoom=7).to_dask()
```

For future hackathons, a transition towards STAC would likely be advantageous. The requirement for cleaner datasets should be achievable as workflows improve with experience. One possible solution for managing dataset variants is to store the different variants in separate Zarr groups within a single dataset.

2.2 Grid

High-resolution climate models employ a variety of computational grids, which complicates direct comparison between model outputs. To facilitate intercomparison during the global hackathon, all data were remapped to a common grid. The chosen grid needed to support efficient data access and storage, which translated into requirements for hierarchical structure, spatially coherent storage, direct conversion between geographic coordinates and cell indices, and equal-area cells to avoid over- or undersampling.

While latitude/longitude grids offer intuitive access and spatial chunking, they suffer from significant oversampling near the poles, resulting in up to 50% more data storage compared to an equal-area grid with the same equatorial resolution. Most non-latitude/longitude grids lack space-filling curve cell ordering or direct computational relationships between geographic coordinates and model cell indices, making efficient subsetting challenging.

Recently, the HEALPix grid (Górski et al., 2005) has gained traction in the geoscience community. Originally developed for astronomy, HEALPix has proven effective for representing climate model output in projects such as nextGEMS and DestinE. HEALPix decomposes the sphere into 12 equal-area diamond cells, which are recursively subdivided to create a hierarchy of grids with 12×4^n cells. This structure supports nested cell ordering for spatially coherent data storage and provides direct computations for converting between geographic coordinates and cell indices.

For the hackathon, the HEALPix grid was selected as the common target for remapping. For global models, this choice is straightforward; for regional models there is the quostion of how to deal with areas outside the model domain. The simple solution is to fill them with Not-a-Number (NaN) values. Compressed storage efficiently handles these NaN regions, as standard compression algorithms reduce their storage footprint to nearly zero, and Zarr can even skip writing chunks that contain only NaNs. Horizontal data chunking and careful processing remains advisable to avoid loading NaN regions into memory. An alternative approach stores only non-NaN grid cells alongside their HEALPix indices as an index vector, enabling Python syntax such as `dataset.sel(cell=12345)` to access specific HEALPix cells. Both approaches integrate with the nearest-neighbor interpolation functionality in `easygems.healpix_show`.

Since the hackathon, the specification for HEALPix grids has been added (Kölling et al.) to the CF conventions in version 1.13 (ChrisBarker-NOAA et al.), enabling data providers and tools to adopt a standardized approach for describing and interpreting HEALPix data. Further tooling support, including for differential operators, is anticipated as HEALPix continues to gain adoption within the climate modeling community.

2.3 Data format

The choice of data format was critical for enabling efficient access and analysis of large, high-resolution datasets across distributed nodes. Key requirements included the ability to open large datasets quickly, support for chunking in all dimensions, and reliable performance over network connections. Zarr natively supports all required features and was selected as the primary format for the hackathon. Several other formats were considered. In principle, HDF5 and netCDF also support multi-file datasets, but as we have never seen this used in real-life, and thus had no prior experience on performance robustness of this approach, we refrained from using this at the scale of the hackathon. When combined with Kerchunk, netCDF can be used as pseudo-Zarr in python, providing the fast loading of full datasets, but this is only supported in Python. As km-scale datasets can grow into the PB-range, even a dataset wit (multi-) GB files would consist of thousands of files, and just opening the full dataset in `xarray` would require opening all of those and then combining the contentents across variables and time. GRIB, even with Kerchunk, has all the limitations of NetCDF and additionally suffers from poor chunking performance. For some of the IFS data we still used this in combination with DKRZ's xpublish-based eerie.cloud server, providing the IFS data as zarr to the outside world. The global GRIB data chunks however led to severe performance penalties for this approach.

The datasets were provided in Zarr version 2, as support for Zarr version 3 was not yet available in all relevant libraries, such as `libnetcdf`. Zarr's chunk-based storage model offers advantages for both compression and data access, but requires careful balancing between file system constraints (e.g., inode quotas and IOPS) and the desire for small chunks to minimize unnecessary data loading. When chosing the chunk shape, one is faced with a trade-off between optimizing for map-based access and pointwise timeseries (Rew, 2013). For the hackathon, chunk sizes were chosen to balance efficient access for both time series and map-based analyses, shifting the focus towards map-based access as we recognized that many processing tools still operate primarily on map data. However, Zarr version 2 can

generate a large number of files, which can be challenging for file systems, especially on HPC platforms with inode quotas.

Looking forward, adoption of Zarr version 3 is anticipated, as its sharding capability will significantly reduce the number of files on disk. Improved support for Zarr 3 in NetCDF and other zarr reading libraries will further enhance interoperability and performance for future large-scale climate data analyses. However, the current situation for the US science funding makes it unclear how fast this is to be expected to be available.

2.4 Documentation

To document the planning and the workflows for the hackathon we created and used a set of web-pages. Digital Earths Global Hackathon (Team, c) was the main page created in the planning of the hackathon, and contained information for the hosting teams. The information for the particpants was collected in a separate site (Team, b). Each node created its own registration and local planning pages. Finally a GitHub repository (digital-earths-global hackathon) contained readme files for the topical teams that went beyond the information provided via the HK25 site (Team, b).

For the data processing, we pointed the participants to project pythia (Rose et al., 2023), *the education working group for Pangeo*, and easy.gems (DYAMONDs, 2021), a collection of examples for analysis codes showing access and plot methods, and instructions for creating own scripts for converting data to HEALPix / Zarr run by MPI-M and DKRZ.

2.5 Python environment

To facilitate seamless collaboration across distributed nodes, we established a standardized Python software stack capable of accessing all datasets in the catalog and performing standard analysis tasks.

The primary options for Python package management consist of the conda-forge ecosystem and the Python Package Index (PyPI). Each system contains packages unavailable in the other, necessitating a hybrid approach. We selected conda-forge as the primary package manager due to its compatibility with additional PyPI packages via `pip` installation. Mamba was recommended as the package installer due to its established reliability and performance advantages over the standard conda installer, although as of release 23.10.0 conda has used the libmamba solver (Leidel et al.). The environment specification was maintained via GitHub, with automated daily testing of package installation, data access functionality, and basic plotting capabilities using GitHub Actions. This continuous integration approach enabled rapid detection of compatibility issues arising from package updates while maintaining currency with the evolving ecosystem.

For future hackathons, pixi (pix) represents a promising alternative installer for conda-forge environments, offering significant performance improvements. Similarly, uv (, creators of Ruff) would be well-suited for PyPI-only environments due to its exceptional installation speed.

2.6 HPC access

The computational infrastructure requirements for data analysis encompassed three primary considerations: sufficient memory resources for large dataset processing, internet connectivity for remote data access, and user-friendly scripting environments.

Most participating sites deployed their Python environments alongside JupyterHub installations, providing browser-based access to HPC resources. Integration with HPC schedulers such as SLURM enabled efficient user distribution across compute nodes while maintaining process isolation to prevent resource contention between concurrent analyses. Alternative approaches included remote-capable editors such as Visual Studio Code combined with SLURM job submission for Python script execution. Experience at DKRZ indicates that hackathons supporting approximately 100 participants require dedicated allocation of 10 high-memory nodes (512 GB RAM each) for interactive analysis workloads.

Computational requirements vary significantly based on research objectives and participant technical proficiency. Effective utilization of the resolution hierarchy, appropriate chunking strategies, and intermediate result caching enable many analyses to execute on modest hardware platforms, including standard laptops. Training programs and technical consulting can substantially reduce resource requirements in many cases. However, some analyses necessitate processing of complete high-resolution datasets, requiring dedicated HPC infrastructure.

2.7 Communication platform

Effective communication infrastructure was essential for both preparation phases and hackathon execution, requiring a platform supporting universal account creation and intuitive operation. While commercial solutions such as Slack impose limitations on free-tier usage, MPI-M's self-hosted Mattermost installation provided an established solution already deployed across multiple European projects, eliminating setup and maintenance overhead. Mattermost offers comparable functionality to commercial alternatives while providing institutional control over data and user management. A critical feature of this installation is automated email notification for user mentions, ensuring message delivery to participants who do not actively monitor the platform, maintaining engagement throughout extended collaboration periods.

Equivalent functionality could be achieved through alternative self-hosted solutions, with Matrix offering additional federation capabilities. The selection was primarily driven by operational convenience among functionally equivalent options.

2.8 Code sharing

The code sharing infrastructure aimed to facilitate cross-site collaboration by enabling scientists to share analysis implementations for application to different datasets, while establishing a persistent repository of analysis methods

for post-hackathon utilization. Key requirements included accessible account management for all participants and structured review processes for quality control.

GitHub was selected over self-hosted GitLab or alternative providers due to widespread existing account ownership among participants and straightforward account creation procedures. A centralized repository was established with mandatory pull request reviews for main branch contributions, balancing minimal participant burden with essential quality assurance and error detection capabilities. However, post-hackathon evaluation identified code sharing as a significant challenge requiring enhanced approaches for future events.

A primary technical issue involved Jupyter notebook file size inflation due to embedded plot outputs. This could potentially be addressed by standardizing on percent-format notebooks (plain Python files with `# %%` cell delimiters) combined with continuous integration systems executing notebooks against online datasets. However, such automation would require notebooks to operate within constrained computational resources and might encounter compatibility issues with specialized analysis packages.

Overall effectiveness could be improved through pre-hackathon training programs covering version control systems, collaborative development practices, and efficient analysis techniques. However, such initiatives would require significant resource allocation and planning commitment in preparation phases.

2.9 Custom Python packages

To optimize analysis performance on the HEALPix grid, several specialized Python packages were developed and deployed. These tools build upon established foundational libraries: healpix (Tessore et al.) and healpy (healpy developers), which provide core HEALPix grid functionality. The choice between these libraries involves licensing considerations, with `healpy` distributed under the GPL license while `healpix` employs the more permissive BSD 3-clause license.

The easygems package (Kluft et al.) provides lightweight utility functions optimized for hackathon workflows, including the `healpix_show` function for rapid visualization of high-resolution HEALPix data as demonstrated on the easy.gems documentation portal (DYAMONDs, 2021). Additional functionality includes optimized routines for data transformation to and from HEALPix representations.

XDGGS (xarray-contrib/xdggs contributors) extends the xarray ecosystem with specialized functionality for Discrete Global Grid Systems (DGGS), including HEALPix. By focusing on the subset of unstructured grids with inherent spatial structure, XDGGS achieves superior computational efficiency compared to generic unstructured grid approaches.

UXarray (developers) adopts a comprehensive approach to unstructured grid support, providing generalized functionality for diverse grid types including specialized readers for HEALPix and other structured variants.

3 Preparation

Preparatory activities for the hackathon commenced in late 2023 with initial concept development and team identification. Throughout 2024, the fundamental framework was disseminated to prospective participants and institutional nodes were established. Formal technical preparations began in Fall 2024 with a two-day virtual workshop featuring presentations on core technical concepts and implementation strategies. Presentation materials were archived on the hackathon website (Team, d) with recordings distributed to participating teams for reference.

The data specification was collaboratively developed through a shared GitHub repository integrated with the hackathon documentation system. This specification underwent iterative refinement through community discussion to achieve an optimal balance between dataset comprehensiveness and manageable data volumes.

To facilitate seamless intercomparison, all datasets required standardization to HEALPix grid format and Zarr storage, necessitating substantial preprocessing efforts. Among participating models, only ICON and IFS possess native HEALPix output capability; all other model outputs required grid transformation. Additionally, with the exception of ICON, no models produced Zarr-format outputs, requiring format conversion for all datasets. Technical documentation (MPI-M and DKRZ) detailing conversion procedures was provided through the easy.gems platform. The combination of late-arriving simulation outputs and limited dedicated technical personnel at some nodes created significant challenges in completing multi-terabyte dataset conversions within hackathon timelines. In several cases, the data preprocessing effort exceeded the effort involved in the original model simulations using established workflows. The heterogeneous nature of model output formats precluded development of universal conversion tools, requiring dataset-specific processing approaches. Integration of pre-existing simulation data introduced additional complexity, as perfect harmonization of output variables and temporal frequencies across all models was not achievable, though efforts were made to maximize dataset coherence.

3.1 Remapping

Multiple remapping approaches were available for transforming native model grids to the target HEALPix configuration, with modeling teams selecting methods based on their specific requirements and technical constraints. Nearest-neighbor remapping represents the most straightforward approach, typically implemented using K-D tree algorithms. This method preserves non-linear relationships between variables by directly transferring values from native grid points, though prior vertical interpolation to pressure levels may have already compromised such relationships. The primary limitation of nearest-neighbor remapping is spatial displacement of data points, which degrades gradient calculations and related differential operators.

Delaunay triangulation-based interpolation offers an alternative that better maintains spatial gradients through linear interpolation between native grid vertices. However, this approach tends to smooth extreme values and may

compromise non-linear variable relationships. Conservative remapping preserves area-averaged quantities but similarly introduces smoothing effects.

Within the HEALPix framework, standard interpolation algorithms assume great circle boundaries for grid cells, introducing theoretical geometric errors. However, at the 10 km spatial scales characteristic of this study, such approximation errors remain practically negligible. Following remapping to the finest HEALPix resolution level, coarser resolution levels were generated through hierarchical averaging. Four spatially adjacent cells (consecutive in the nested ordering scheme) were combined into single coarser-level cells. For datasets containing missing values, appropriate weighting based on valid data fractions was implemented to maintain conservation properties throughout the resolution hierarchy. As discussed in Grid above, regional models represent out-of-domain values as NaNs, meaning that it is particularly important to handle the weighting of missing values for these datasets in order to ensure consistent domain-wide statistics across the resolution hierarchy.

Several established tools support these remapping operations. The ICON model employs its YAC coupler to remap from native icosahedral grids to HEALPix through the hiopy output server. YAC's remapping algorithms are also accessible via the Climate Data Operators (CDO) suite. Nearest-neighbor interpolation and Delaunay triangulation implementations are readily available in Python and documented through the easy.gems platform.

3.2 Data format conversion

Dataset conversion to Zarr format was typically integrated with the remapping workflow. While most remapping approaches operate optimally on complete 2D fields, efficient access to both time series and spatial maps requires multidimensional chunks spanning all dataset dimensions, targeting sizes of a few megabytes.

This chunking strategy introduces computational challenges, as large 4D data cubes must be maintained in memory for efficient reorganization. For the IFS 2.8 km simulation, global GRIB fields were initially divided into smaller regional chunks and written to intermediate storage before final assembly into multidimensional chunks. This approach proved more efficient than direct processing, even on supercomputer nodes with 1 TB of main memory.

Some IFS datasets were retained in their original format and accessed through Kerchunk indexing, providing virtual Zarr access via Python libraries or DKRZ's xpublish-based eerie.cloud server. However, the global extent of GRIB data chunks resulted in significant performance penalties for remote access patterns.

3.3 Data distribution and access

The majority of datasets were made available as online-accessible Zarr stores distributed across multiple institutions. DKRZ and JASMIN provided datasets through their object storage systems, while the NICAM team operated a dedicated web server. DKRZ additionally served IFS datasets via their eerie.cloud infrastructure.

Given the high frequency of common data subset analysis by multiple users, implementing reverse proxy caching at hackathon sites would likely yield substantial performance improvements. However, establishing such infrastructure for multi-terabyte datasets with 50 MB chunk sizes presents significant technical challenges and was not implemented for this event. This capability should be prioritized in future iterations.

Pre-hackathon data replication across nodes helped minimize transfer requirements during the event. The rclone utility proved effective for copying data from object stores and web servers, while NERSC provided datasets through the Globus file transfer system. The replication approach faces limitations when datasets arrive close to the hackathon deadline, as frequently occurs with dedicated simulations. Additionally, complete dataset replication requires substantial local storage resources and may include variables unnecessary for specific user communities. Consequently, selective replication through caching seems preferable for high-resolution dataset variants.

3.4 Data catalog development

Effective data discovery required a comprehensive catalog system accessible to users across all participating nodes. The catalog needed to support dataset variants (different spatial and temporal aggregations), multi-file datasets, and straightforward maintenance procedures.

Two primary catalog standards were evaluated: SpatioTemporal Asset Catalogs (STAC) and Intake. While STAC has emerged as the standard for satellite and observational data, its design emphasizes cataloging numerous individual objects with explicit geometries, reflecting its origins in satellite imagery. This architecture suits datasets with millions of discrete assets but less effectively serves climate model outputs, which typically comprise unified global datasets available at multiple resolutions where spatiotemporal metadata is less critical.

Intake version 0.7 was selected for its robust support of parameterized datasets, enabling syntax such as `icon_d3hp003(zoom=7, time="PT1H", time_method="mean")` to specify dataset variants. This approach clearly separates the model realization (`icon_d3hp003`) from spatiotemporal aggregation parameters. Intake 0.7 also supports multi-file dataset definitions, reducing preprocessing requirements during data preparation phases.

The catalog was maintained through a GitHub repository with YAML-based dataset descriptions, largely adapted from existing nextGEMS and EERIE catalog entries by manual editing (manageable for the 44 datasets involved). A Python script executed via GitHub CI combined individual entries into a unified catalog (Team, a). In this process, locally available datasests were supplemented with those available online where needed. A second script generated the human-readable landing page (Team, a) from the embedded descriptions. For online datasets, links to a JavaScript-based web viewer (gridlook developers) enabled interactive data inspection.

4 Lessons learned

Overall, the hackathon was a great success. The close collaboration between the technical and science teams during the hackathon proved extremely useful. The scientists made good use of the examples and instructions provided and picked up the new workflows extremely fast. At the same time, the technical team could observe their ways of interacting with the data and find and solve key pain points. These rapid development cycles minimized dead-end developments. The shared coding environment allowed for collaboration across nodes. Distributing the data before the hackathon reduced the bandwidth requirements for cross-global access but also required the simulations to be provided in advance, which did not work in all cases. In retrospect, it would have been useful to have an in-person meeting with the technical team before the hackathon, as direct communication is much more effective than remote. This has proven useful in the nextGEMS hackathons, where most of the technical staff was located in Germany anyways, but would have been more complicated here given the global context of this project, and the lack of dedicated technical staff at some of the nodes. Also, language barriers still exist and complicate international collaboration, however, we expect these to be reduced by international exchange and large language models in the future.

References

pixi: fast conda-forge environment installer, https://pixi.prefix.dev, last retrieved April 1, 2026.

ChrisBarker-NOAA, castelao, cofinoa, Kölling, T., Gregory, J., Hassell, D., pvanlaake, and TomLav: CF-1.13 (CF Conventions Release), https://github.com/cf-convention/cf-conventions/releases/tag/v1.13.0, last retrieved April 1, 2026.

(creators of Ruff), A.: uv: An extremely fast Python package and project manager, written in Rust, https://docs.astral.sh/uv/, last retrieved April 1, 2026.

developers, U.: UXarray Documentation: Xarray extension for unstructured climate and global weather data analysis, https://uxarray.readthedocs.io/, last retrieved April 1, 2026.

digital-earths-global hackathon: hk25-teams repository, https://github.com/digital-earths-global-hackathon/hk25-teams, last retrieved April 1, 2026.

DYAMONDs, T.: easy.gems: User-driven site for documenting high-resolution climate simulation output and analysis, https://easy.gems.dkrz.de/, last retrieved April 1, 2026, 2021.

EERIE consortium: EERIE project website, https://eerie-project.eu/, last retrieved April 1, 2026.

Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., and Bartelmann, M.: HEALPix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere, The Astrophysical Journal, 622, 759–771, https://doi.org/10.1086/427976, 2005.

gridlook developers: gridlook web viewer, https://gridlook.pages.dev/, last retrieved April 1, 2026.

healpy developers: healpy documentation, https://healpy.readthedocs.io/en/latest/, last retrieved April 1, 2026.

Irani, L.: Hackathons and the Making of Entrepreneurial Citizenship, Science, Technology, & Human Values, 40, 799–824, https://doi.org/10.1177/0162243915578486, 2015.

Kluft, L., Kölling, T., and Franke, H.: easygems: Utilities for earth system model analysis, https://github.com/mpimet/easygems, last retrieved April 1, 2026.

Kölling, T., Horn, C., Hassell, D., Magin, J., Okada, T., and Gregory, J.: Healpix Pull Request 605 (CF Conventions Pull Request), https://github.com/cf-convention/cf-conventions/pull/605, last retrieved April 1, 2026.

Leidel, J., Hoth, D., jaimergp, Hoffmann, T., and Odegard, K.: Conda Release 23.10.0, https://github.com/conda/conda/releases/tag/23.10.0, last retrieved April 3, 2026.

MPI-M and DKRZ: easy.gems technical documentation for HEALPix processing, https://easy.gems.dkrz.de/Processing/healpix/index.html, last retrieved April 1, 2026.

Neumann, P.: ESiWACE Newsletter 07/2019, https://doi.org/10.5281/ZENODO.3342475, 2019.

nextGEMS consortium: nextGEMS project website, https://nextgems-h2020.eu/, last retrieved April 1, 2026.

OpenBSD: OpenBSD Hackathons, https://www.openbsd.org/hackathons.html, last retrieved April 1, 2026, 2026.

Rew, R.: Chunking Data: Why it Matters, https://www.unidata.ucar.edu/blogs/developer/entry/chunking_data_why_it_matters, blog post, Jan 29, 2013, 2013.

Rose, B. E. J., Clyne, J., May, R., Munroe, J., Snyder, A., Eroglu, O., and Tyle, K.: Project Pythia: An education and training hub for the geoscientific Python community, Tech. rep., Zenodo, https://doi.org/10.5281/ZENODO.8184298, 2023.

Rys, M.: Invention Development. The Hackathon Method, Knowledge Management Research & Practice, 21, 499–511, https://doi.org/10.1080/14778238.2021.1911607, 2021.

Stevens, B., Satoh, M., Auber, L., Biercamp, J., Bretherton, C. S., Chen, X., Düben, P., Judt, F., Khairoutdinov, M., Klocke, D., et al.: DYAMOND: the DYnamics of the Atmospheric general circulation Modeled On Non-hydrostatic Domains, Progress in Earth and Planetary Science, 6, 1–17, https://doi.org/10.1186/s40645-019-0304-z, 2019.

Takasuka, D., Satoh, M., Miyakawa, T., Kodama, C., Klocke, D., Stevens, B., Vidale, P. L., and Terai, C. R.: A protocol and analysis of year-long simulations of global storm-resolving models and beyond, Progress in Earth and Planetary Science, 11, https://doi.org/10.1186/s40645-024-00668-1, 2024.

Team, D. E. G. H.: Digital Earths Global Hackathon Data Catalog, https://digital-earths-global-hackathon.github.io/catalog/, last retrieved April 1, 2026, a.

Team, D. E. G. H.: Digital Earths – Global Hackathon: Science Plan and Tutorials, https://digital-earths-global-hackathon.github.io/hk25, last retrieved April 1, 2026, b.

Team, D. E. G. H.: Digital Earths – Global Hackathon, https://digital-earths-global-hackathon.github.io/, last retrieved April 1, 2026, c.

Team, D. E. G. H.: Digital Earths – Global Hackathon: Talks, https://digital-earths-global-hackathon.github.io/talks/, last retrieved April 1, 2026, d.

Tessore, N., Roet, S., and dependabot[bot]: healpix: Python and C package for HEALPix discretisation of the sphere, https://github.com/ntessore/healpix, last retrieved April 1, 2026.

xarray-contrib/xdggs contributors: xdggs: Discrete global grid systems with xarray, https://github.com/xarray-contrib/xdggs, last retrieved April 1, 2026.