Scaling UXarray: Bridging the Gap for High-Performance Unstructured Grid Analysis and Documentation Enhancements

Rachel Tam[1,2], Philip Chmielowiec[1], Orhan Eroglu[1]

[1] NSF National Center for Atmospheric Research, [2] University of Illinois, Urbana-Champaign

Motivation

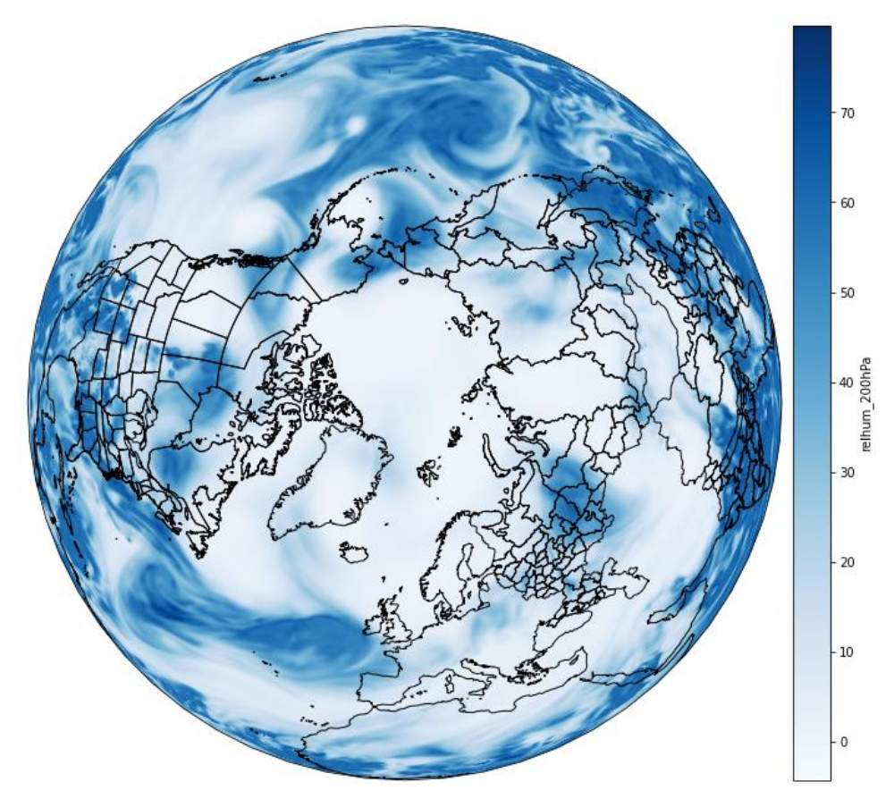
The climate and global weather modeling communities have been transitioning to dynamical cores employing more flexible unstructured grids upon which more complex governing numerical equations of state are solved. The improved computational scalability afforded by unstructured grids allows for higher resolution model runs, producing more realistic simulations of the earth's physical processes, progressing towards the long-term pursuit of producing the world's first global cloud resolving model.

The NSF EarthCube-funded effort, Project Raijin, has spearheaded to create the **UXarray Python Package**. It is a community-owned, sustainable, and scalable tool for analyzing and visualizing unstructured grids. This poster highlights various development efforts that have leveraged tools from the Scientific Python Ecosystem to develop a scalable analysis tool.

Overview and Highlighted Features

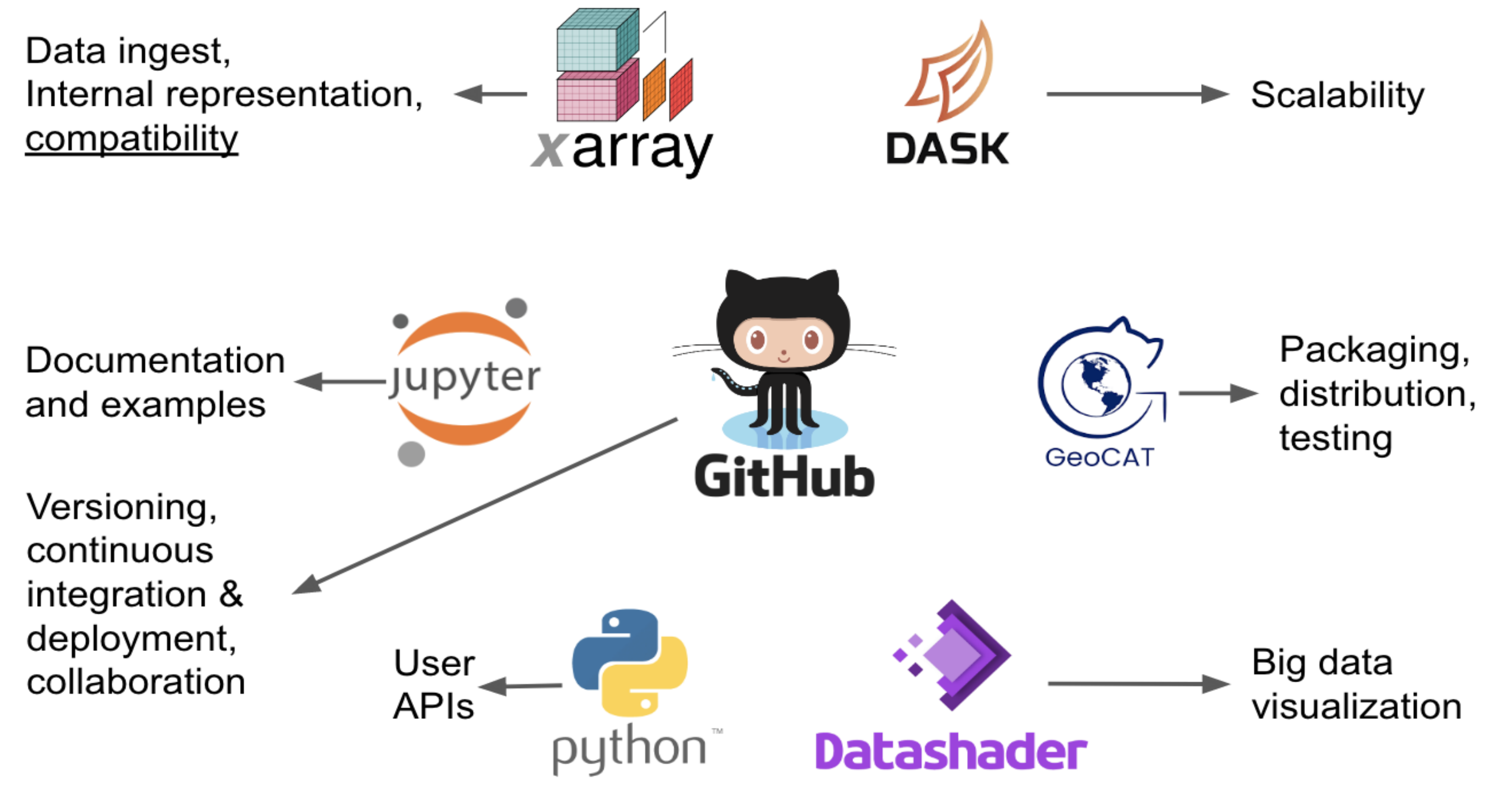
Direct computations on unstructured-grid data reduces overhead and uncertainties from the regridding procedures and avoids duplicating storage consumption. UXarray provides Xarray-styled functionality for better read-in and usage of datasets that follows standard unstructured grid conventions, such as UGRID, MPAS, SCRIP, ESMF, and Exodus. All functionality operators directly on unstructured grids, including the following highlighted features:

- High-resolution Visualization (3.75km)
- Remapping
- Subsetting
- Statistical Operators
- Calculus Operators
- Topological Aggregations



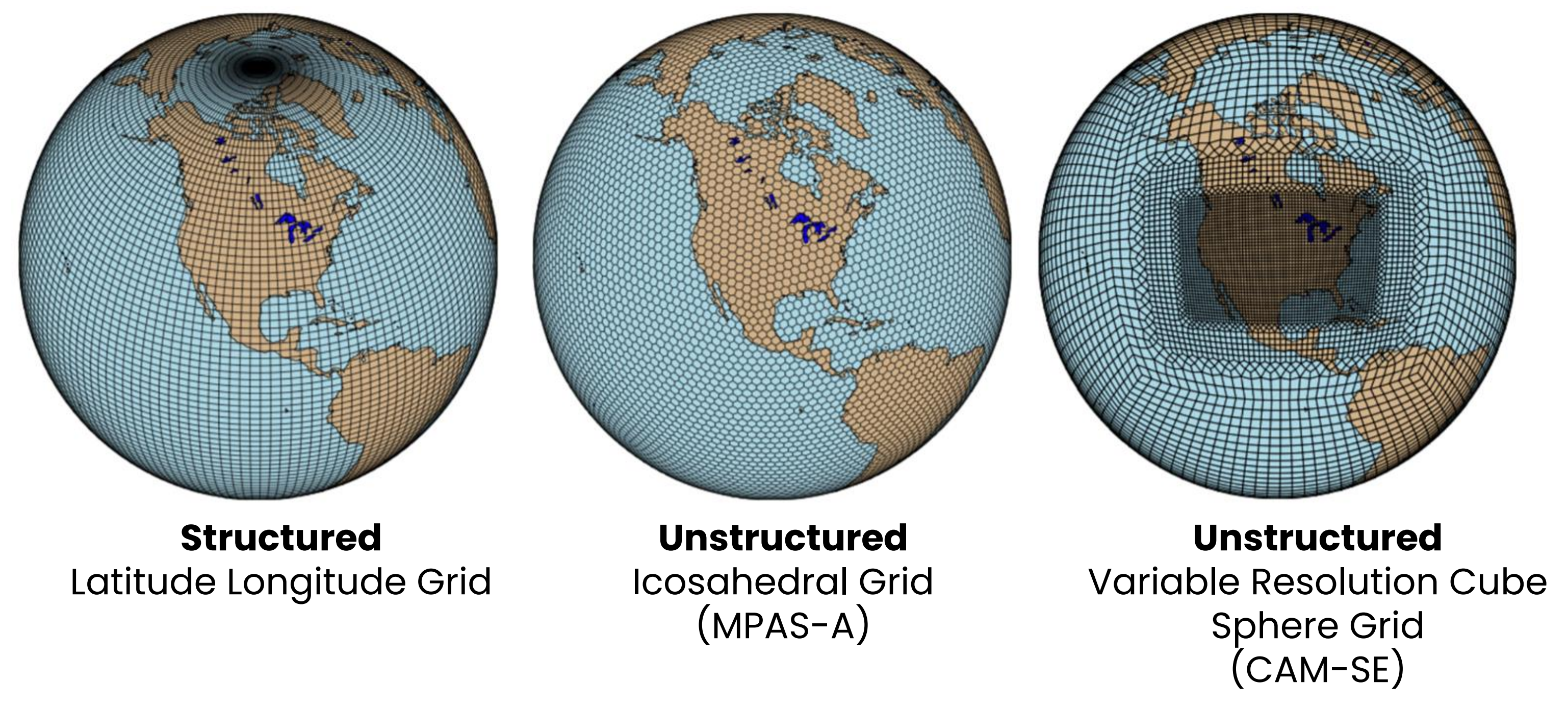
MPAS Point Rasters at 3.75km
42 million faces and 84mil nodes

Core Technologies



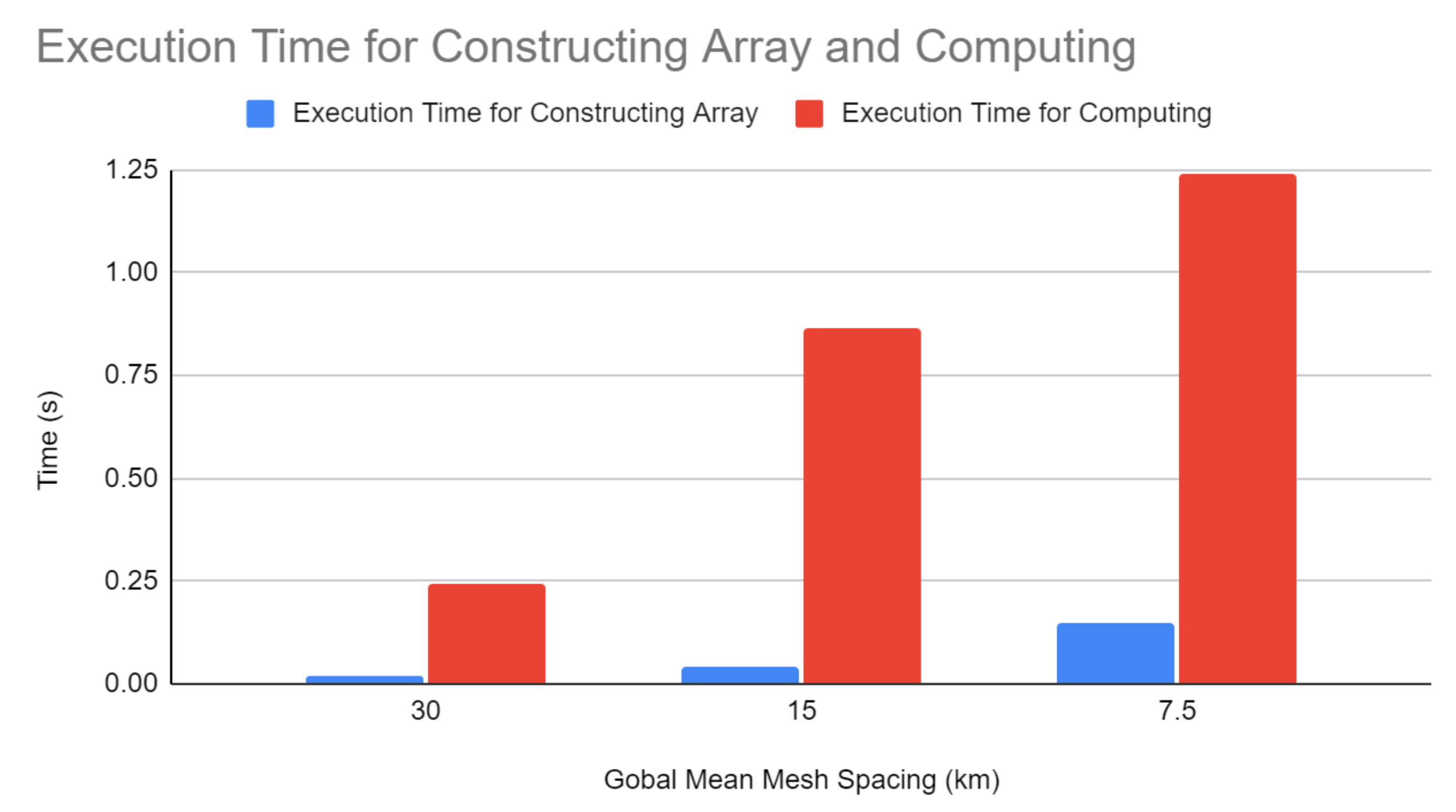
Optimized Weighted Mean

Data variables are typically mapped to geometries that are either nodes, edges, or faces of an unstructured grid. With the spherical spatial structure of the Earth and the highly flexible configurations permitted by unstructured grid, data at each geometry may not be equally weighted.



One of the most common operations is taking average over global data. By modifying the built-in mean function, UXarray takes the geometries as weights and would scale the datapoints by face, edge or node and return the global mean.

Benchmarking is conducted to evaluate the function's performance for grids at varying resolutions. 1-year MPAS simulations for DYAMOND, the global storm resolving models intercomparison project (Stevens et al., 2019) at horizontal resolutions of 7.5, 15 and 30km are used. Chunking is done automatically for the time and vertical levels, while the geometry dimension remains unchunked for the calculations.

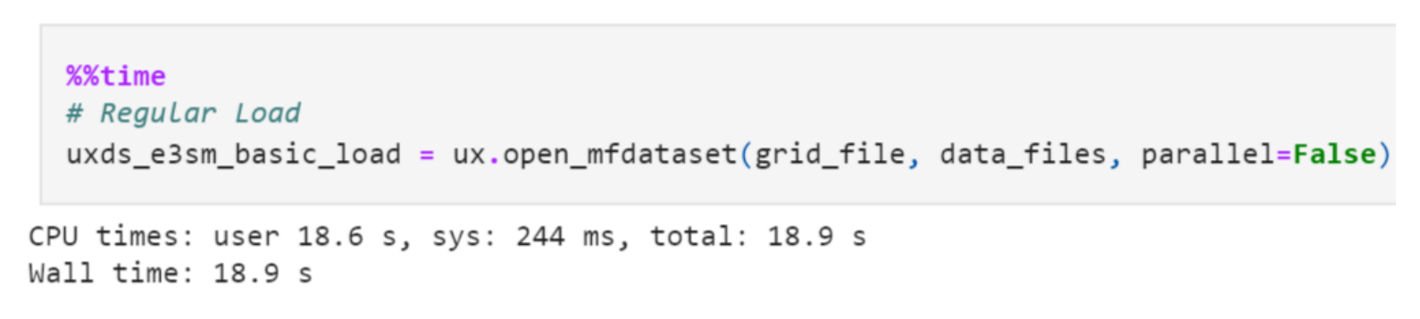


Bar chart of execution time for constructing Dask array and conducting the calculations for taking weighted mean on MPAS datasets at varying resolution.

Additional benchmarking of the weighted mean function with varying configuration of the local cluster is also recorded as supplementary data.

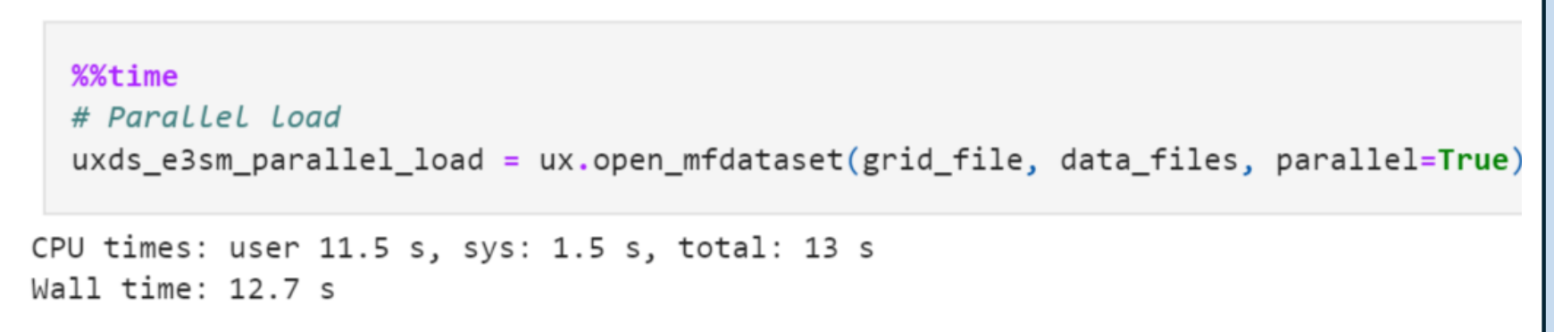
Documentation Enhancements

A new step-by-step user guide page is dedicated to demonstrate how to use UXarray with Dask for scalable workflows. Given the scale of high resolution model output, using Dask for parallel loading and out-of-memory operations is necessary.



```
%%time
# Regular Load
uxds_e3sm_basic_load = ux.open_mfdataset(grid_file, data_files, parallel=False)
```

```
CPU times: user 18.6 s, sys: 244 ms, total: 18.9 s
Wall time: 18.9 s
```

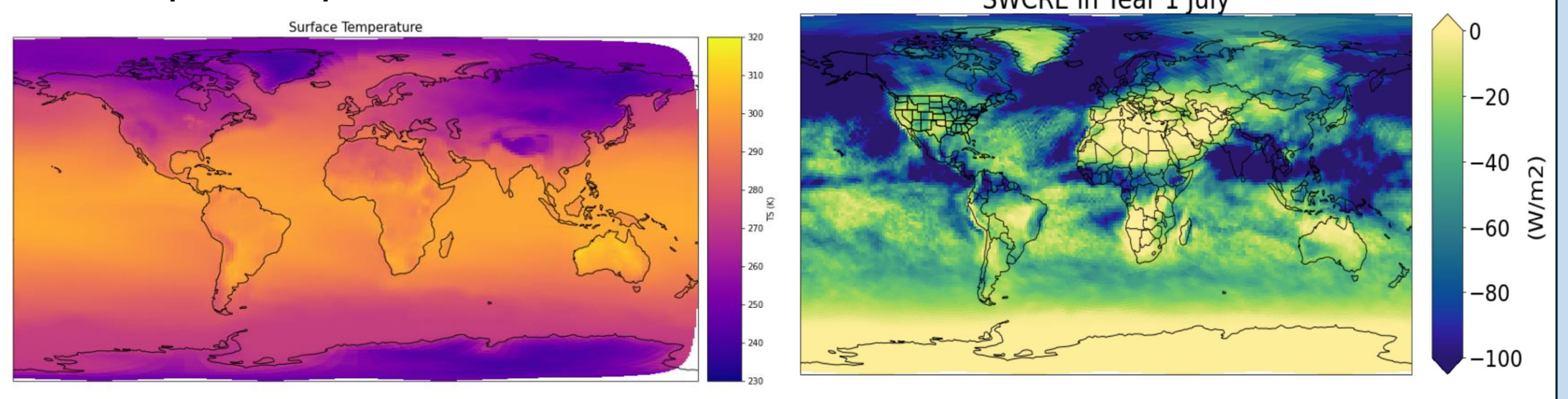


```
%%time
# Parallel Load
uxds_e3sm_parallel_load = ux.open_mfdataset(grid_file, data_files, parallel=True)
```

```
CPU times: user 11.5 s, sys: 1.5 s, total: 13 s
Wall time: 12.7 s
```

Comparison between Regular and Parallel Loading Performance for E3SM-v2 output at 1 degree resolution (72 timesteps, 21600 faces, 72 vertical levels and 471 variables). Local cluster is set up with 128 workers and 2 threads per worker.

Additional usage examples for UXarray are also populated, showcasing the analysis and visualization of E3SM-v2 model output, serving both the functions of presenting package functionalities including unstructured grid arithmetic operations, and testing the package's performance on such large dataset and different unstructured grid convention (relative to MPAS) in the package development process.



E3SMv2 simulation with Present Day Control Forcings at 1 Degree Resolution. Left panel shows Year 1 January Surface Temperature. Right panel shows shortwave cloud radiative effect (SWCRE) by doing arithmetic operations with climate output variables on native unstructured grids like net shortwave radiation (FSNT) and outgoing longwave radiation (FLUT).

Future Work

With majority of the package written in Numpy, other existing functionalities like topological aggregations will be translated to Dask for better scalability and memory performances.

Additional benchmarking on the NSF-NCAR supercomputer, Derecho, will be conducted to evaluate the performances for all functionalities in Uxarray with Airspeed Velocity (ASV).

User Guide

GitHub

Acknowledgements

This work part of Project Raijin, funded by the National Science Foundation under Grant No. 2126458. A great thank you to my mentors Philip and Orhan for their guidance, support, patience and understanding on the project and beyond. Thank you NCAR, CISL and SIParCS, particularly Virginia, Jessica and Eva, for all resources and facilitation throughout the SIParCS internship that made this project all possible.