Robust Data Type Testing and Dask Support in MetPy

Russell P. Manser[1], Ryan May[2]
[1]Texas Tech University, [2]Unidata

Dask support in ***MetPy*** will enable users to ***distribute calculations*** across ***large meteorological datasets***

- Python package for meteorological calculations and plotting
- Designed and developed for community use and contributions
- Dependent upon the scientific computing stack

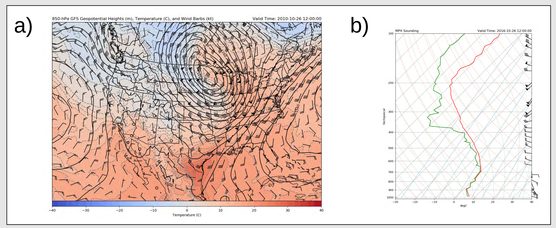

Figure 1: examples of a) a model forecast and b) upper air observations plotted with MetPy (https://unidata.github.io/python-gallery/examples/index.html).

- Python package that enables larger-than-memory and distributed memory computing for large datasets
- Has user API similar to common Python data science tools
- Part of scientific computing stack

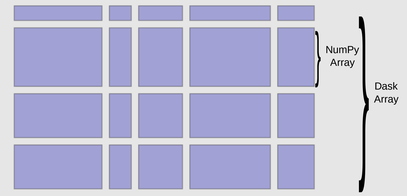

Figure 2: depiction of a Dask Array composed of multiple NumPy arrays that vary in size. Image credit: https://docs.dask.org/en/latest/array.html

1. Introduction

The number of scientists and students using MetPy is growing rapidly. New features continue to be requested to meet the needs ot the exapnding user base. One of these needs is support for efficient calculations on larger-than-memory datasets. Support for Dask Arrays in MetPy will help users derive quantities from inherently large datasets such as climate observations and model output, ensemble model output, and high-resolution cloud model output.

Systematic testing for data types is a hidden need of users as well. MetPy functions are not explicitly tested against supported data types, which results in bugs. An automated and systematic testing suite is required to improve robustness of MetPy functions and add Dask Array support.

2. Objectives

- Integrate Dask support in upstream libraries
- Design automated data type tests
- Identify functions that need special handling or refactoring for full Dask support

3. Challenges

- The type casting hierarchy in the scientific computing stack is not clearly defined
- There are many operations and functions that are difficult to distribute with Dask
- MetPy needs to remain accessible for community use and contributions

4. Dask Integration

The order in which data types are wrapped (type casting hierarchy) and how those wrapped types behave (duck typing) is crucial for expected behavior of each type.

"If it walks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck."

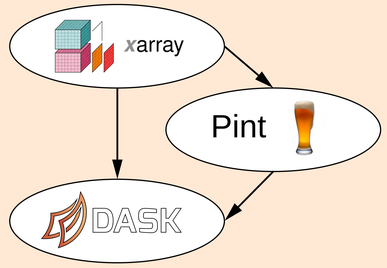

Figure 3: a graph of the type casting hierarchy between Xarray, Dask, and Pint.

- [x] Implement Dask in Pint
- [x] See it on GitHub!
- [ ] Support duck Dask Arrays in Xarray
- [ ] See it on GitHub!
- [ ] Dask array tests in MetPy

5. Data Type Testing in MetPy

MetPy generally supports data types common to the scientific computing stack. However, these types are not always explicitly tested against individual functions. An automated and systematic testing suite is proposed here to a) explicitly test all supported data types and b) to provide the framework for testing of new data types (e.g., Dask Array).

Supported Types

- Scalars
- NaNs
- NumPy array
- NumPy masked array
- Xarray Variable and DataArray

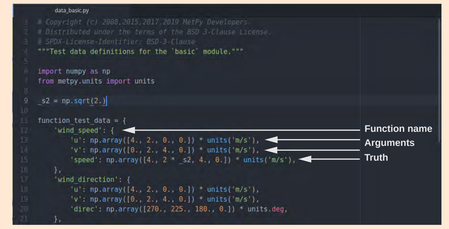

Figure 4: an example of how test data is added for a function in the proposed framework.

6. Future Work

- Implement the proposed testing framework for all calc modules in MetPy
- Address challenges for Dask support in individual MetPy functions

Acknowledgements: Drew Camron and Sean Arms for additional mentorship, Jon Thielen for general encouragement and guidance with implementing the Dask collection interface in Pint, and the maintainers of Pint and Xarray for help with PRs. This work was funded primarily by the National Science Foundation (Grant AGS-1901712).