Duck array compatibility meeting

Attendees:

- Tom Nicholas (@TomNicholas) - he/him - Columbia University - xarray dev team + dask/pint/pangeo user
- Jon Thielen (@jthielen) - MetPy dev team
- Justus Magin (@keewis) - xarray dev team
- Simon Cross (@hodgestar) - QuTiP dev team
- Simon Heybrock (@simonheybrock) - scipp dev team
- Hameer Abbasi (@hameerabbasi) - PyData/Sparse, uarray/unumpy/udiff
- Ralf Gommers (@rgommers) - Quansight/NumPy/SciPy/PyTorch / array API standard
- Peter Andreas Entschev (@pentschev) - Dask/CuPy
- Benjamin Zaitlen (@quasiben) - Dask/RAPIDS
- Stephan Hoyer (@shoyer) - Xarray/NumPy/JAX
- Nick Becker (@beckernick) - RAPIDS/Dask (interested observer)
- Greg Lucas (@greglucas) - Numpy/MaskedArray, interested observer
- Guido Imperiale (@crusaderky) - xarray, dask dev team
- Jacob Tomlinson (@jacobtomlinson) - he/him - NVIDIA - RAPIDS/Dask
- Leo Fang (@leofang) - CuPy
- John Kirkham (@jakirkham) - Dask/RAPIDS
- Andrew McNichols (@amcnicho) - National Radio Astronomy Observatory
- Jim Pivarski (@jpivarski) - Princeton, IRIS-HEP

Agenda:

- Brief intros:
    - Name / pronouns
    - Institute / company
    - Library(ies) you work on
- Orders of business:
    - 1-hour official meeting, but can stay for discussion afterwards
    - Repo/NEP/etc. for standardizing wrapping order and other future decisions
    - Note-taking in this doc (by Tom) but ideally record the meeting so it can be done afterwards
    - Moderation by Tom when necessary, to keep it focused. Unfinished discussions can continue afterwards or in a dedicated repo
- Definition/minimal API of a duck array
    - Xarray been defining duck array via array protocols (__array_ufunc__ & __array_function__) + possessing dtype + shape
    - Array API Standard: https://data-apis.org/array-api/latest/ and NEP 47: https://numpy.org/neps/nep-0047-array-api-standard.html
        - Not incompatible

        - Already defines minimum subset of API
    - See Also: https://github.com/pydata/xarray/issues/5648#issuecomment-890310954
- Which libraries should wrap which other libraries
    - https://github.com/dask/dask/issues/6635
- Consistency of type deferral (e.g., between array functions, ufuncs, module functions, construction, and binary ops)
    - https://github.com/pydata/xarray/issues/3950
    - Partially: https://github.com/pydata/xarray/issues/5559
- Nested array reprs (both short and full)
    - https://github.com/dask/dask/issues/5329
    - https://github.com/dask/dask/issues/6637
    - https://github.com/pydata/xarray/issues/4324
- Addition / removal of layers in a nested duck array
    - Partially: https://github.com/pydata/xarray/issues/3245
    - Partially: https://github.com/pydata/xarray/pull/5568
- Best practices for "carrying through" type-specific operations to wrapped types
    - https://github.com/dask/dask/issues/6636
    - Partially: https://github.com/dask/dask/issues/6385

Notes:

- Nested Duck Array Definition (Justus Magin):
    - Some properties + Protocols (__array_function__ + __array_ufunc__).
    - Should limit to Array API spec
- Which libraries should wrap which other libraries
    - Jon wrote pint technical commentary
    - Community standard of pair-wise interactions
- Repo for discussions
    - Should live in pydata
    - https://github.com/pydata/duck-array-discussion
- Definition of duck array:
    - Follows Array API standard
    - + any other methods
    - Xarray Variable lives above all these duck arrays
- Which libraries should wrap other libraries?
    - Proposal: xarray -> pint -> dask -> others
    - Point: multiple tops, not just xarray
    - Q: Tree looks fine, but how would a new array type fit in?
        - Can tree be defined independently of actual libraries in it?
        - No: interactions have to be defined pairwise between real libraries
        - But don't have to define all interactions with everything
            - Want to only define operations on similar types, else raise
    - Proposal: possessing dtype, shape, ndim, is a possible hierarchy
    - Need something beyond array_priority to define the hierarchy tree pairwise
    - Further discussions go into https://github.com/pydata/duck-array-discussion

- Step back: outputs we want?
    - Design docs?
        - Live in this project? NEP?
        - https://scientific-python.org/
        - Data-apis.orgs
- How are pairwise interactions defined?
    - Current way: via protocols like __array_ufunc__, __array_function__, etc
    - Explicit vs implicit strategy
        - Implicit is protocols, currently widespread
        - Explicit is NEP37: https://numpy.org/neps/nep-0037-array-module.html
    - Maybe we should have a new library that defines the shared type resolution DAG?
        - That way libraries cannot possibly disagree in their code
        - Implementation?
        - TODO (Jon): make a discussion issue for this
    - Or we could define a slot for “handled types”?
- Consistency of type deferral? (e.g., between array functions, ufuncs, module functions, construction, and binary ops)
    - How much should we trust user to not “break” the DAG through inconsistencies in type deferral? How much should we enforce consistency between different type deferrals?
        - Every library have a way to say “here’s how I defer to another library”
        - How do we support custom (unknown) libraries?
        - TODO (Jon): issue discussion on this?
    - There is consensus that these should be consistent (Pint-like) rather than inconsistent (Dask-like), unless there is opt-out of some kind third-party/custom library...but more discussion needed
- Nested array reprs (both short and full)
    - How do you display information from nested duck arrays in a nice way for user?
    - Xarray has _repr_inline_ function
    - Html repr can defer to type’s html repr
    - Python string repr is harder
    - Suggestion:
        - Just make user explore level by level?
    - Verbosity vs completeness
    - Suggestion: dict of important strings/ints from each library
        - TODO (Hameer): issue fleshing out dict suggestion
            - https://github.com/dask/dask/issues/6637
        - TODO (Tom): meta issue of all the TODOs
- TODO: second meeting possibly in future? Wait for asynchronous discussion to find some sticking points first