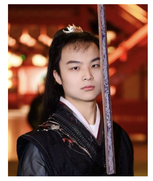
C++ Acceleration of Thermodynamics Module in Metpy & Jupiter 3D Visualization

Linfeng Li[1,2]; Ryan May[1]; Drew Camron[1]; Huazhi Ge[3]; Yuan Ho[1]; Sean Arms[1];
[1]University Corporation for Atmospheric Research (UCAR), NSF Unidata Program Center, Boulder, USA
[2]Climate and Space Sciences and Engineering, University of Michigan, Ann Arbor, USA
[3]Division of Geological and Planetary Sciences, California Institute of Technology, Pasadena, USA



Summary
Waiting on MetPy to process large datasets? We're accelerating key thermodynamic calculations by swapping the Python core with high-performance C++, all without changing the familiar user interface.

What is Metpy?

MetPy[1] is a key **Python** library for analyzing and visualizing **atmosphere** and **weather** data. It meshes well with the scientific Python ecosystem, including the Numpy, Scipy, and Matplotlib, adding functionality specific to **meteorology**.

The Challenge: While user-friendly, Python struggles with the heavy computation required for large, multi-dimensional weather datasets in MetPy.

Our Goal: Accelerate key thermodynamic calculations in MetPy, targeting CAPE (Convective Available Potential Energy) to make analysis faster.

Methods

1) Keep the Python interface, but replace the computational engine with C++.

Table 1. Duty of C++ and Python

Units	Python preprocess
Input check	Python; C++ in special cases
Scalar/Array input	Pybind11; C++ vectorization
Dimension Broadcast	Python
Built-in constants	C++ retrieving from Python
Computing	C++
Assembling	Python

2) Create Python bindings with pybind11, allowing Python to call the compiled C++ code seamlessly.

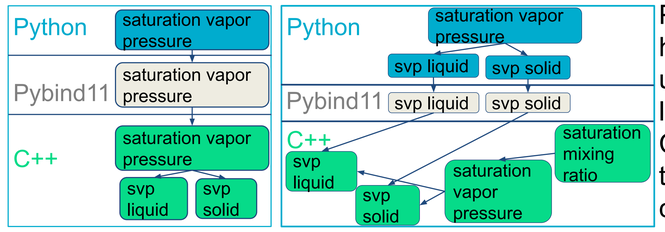
```cpp
#include <pybind11/pybind11.h>

int add(int i, int j) {
    return i + j;
}

PYBIND11_MODULE(_calc_mod, m) {
    m.doc() = "accelerator module docstring";
    m.def("add", &add, "Add two numbers");
}
```

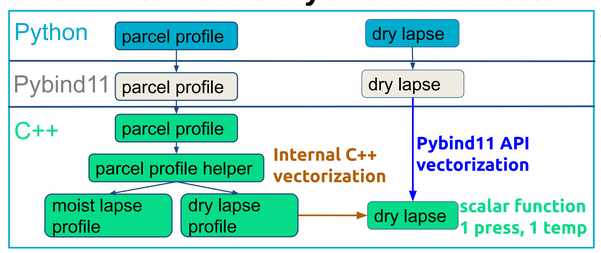
Bridging Python and C++: Design Patterns

a. Case Switch stays in Python



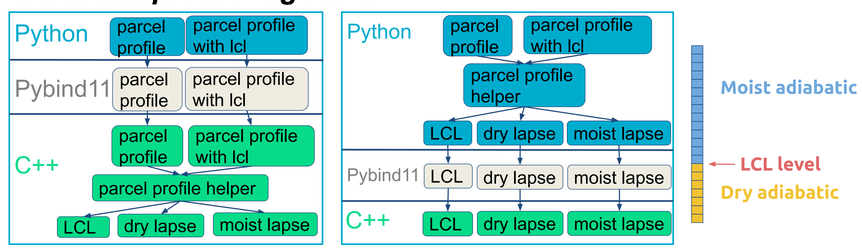
Python layer handles the user-facing logic, while the C++ layer does the underlying computing work

b. C++ Internal .vs. Pybind11 API vectorization



The C++ function is vectorized for external Python calling and internal C++ calling.

c. Parcel lifting profile — piece LCL level, dry, moist adiabatic profile together.



Results: Performance Benchmarking

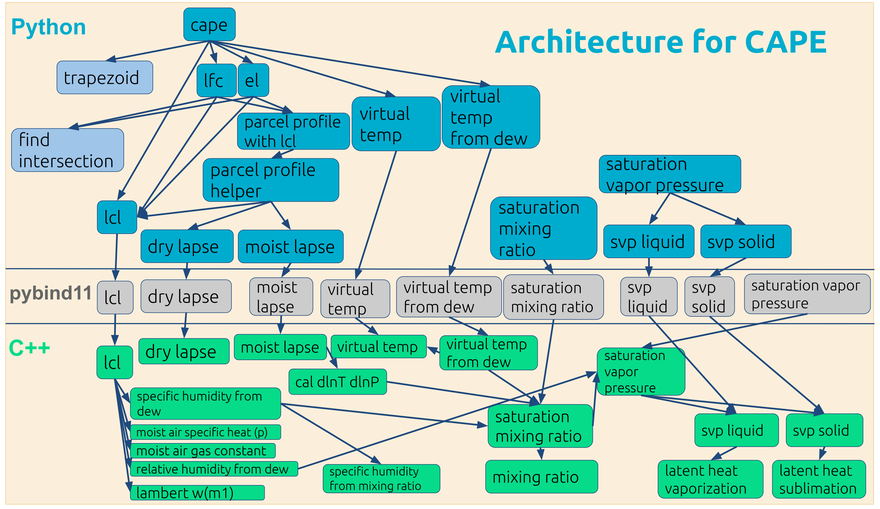
airspeed velocity (asv) is a tool that can rigorously compare the performance of the new C++ architecture.against the original Python code.

Benchmark	Time Changes
cape_cin	−6.8% (-91.615µs)
el	−54.9% (-1.667ms)
lfc	−55.3% (-1.694ms)
lcl (1, 1, 200)	−22.7% (-58.797µs)
lcl (50, 1, 1)	−16.2% (-33.618µs)
lcl (50, 200, 200)	−39.5% (-263.985ms)
lcl (50, 800, 800)	−45.8% (-5.502s)

Benchmark	Time Changes
Broadcast drylapse_1D	−16.7% (-18.000µs)
Broadcast drylapse_2D	+17.6% (+22.614µs)
Broadcast drylapse_3D	+1487.3% (+4.423ms)
Hard Code drylapse_1D	−16.5% (-17.716µs)
Hard Code drylapse_2D	−1.3% (-1.848µs)
Hard Code drylapse_3D	+46.3% (+1.737ms)
moist lapse_1D	−91.3% (-2.685ms)
parcel profile	−65.7% (-1.814ms)

The broadcasted array can be processed faster by Python than C++ pybind11 in the simple dry lapse calculation.

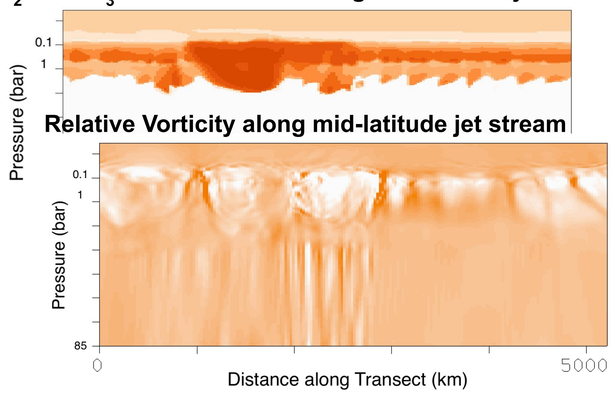
Table 2. C++ function performance



Jupiter 3D Visualization in Integrated Data Viewer

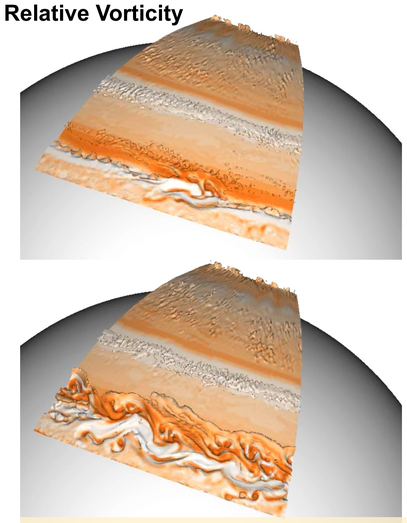
We created 3D visualization for Jupiter in the Unidata Integrated Data Viewer[2] with a simulation output[3] from the nonhydrostatic cloud and convection resolving model, SNAP[4].

H_2O & NH_3 cloud at 0.1 bar along mid-latitude jet stream



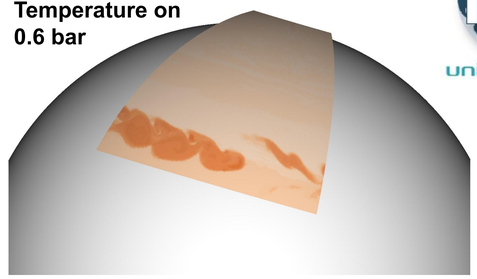
Shallow layer vortices extend from 0.1 bar down to 85 bar pressure level.

Relative Vorticity



Jovian band and zone structure, eastward plume and vortices displayed by H_2O relative humidity isosurface colored by relative vorticity.

Temperature on 0.6 bar



Powered By IDV unidata IDV

References

1. NSF Unidata, (2025): Metpy[software]. Boulder, CO: UCAR/NSF Unidata Program Center. https://doi.org/10.5065/D6WW7G29
2. NSF Unidata, (2025): Integrated Data Viewer (IDV). Boulder, CO: UCAR/NSF Unidata Program Center. https://doi.org/10.5065/D6RN35XM
3. Jupiter data in courtesy of Huazhi Ge, California Institute of Technology, Large-Scale Atmospheric Dynamics in Jupiter's Mid-latitude — Compositional Structure, Thermal Structure, Large-Scale Dynamics, and Forward-Modelling Compared to Radio Observation — Simulated by the Nonhydrostatic Model.
4. Li, Cheng, and Xi Chen. 2019. "Simulating Nonhydrostatic Atmospheres on Planets (SNAP): Formulation, Validation, and Application to the Jovian Atmosphere." The Astrophysical Journal Supplement Series 240(2):37. doi:10.3847/1538-4365/aafdaa.

This material is based upon work supported by NSF Unidata under awards #2103682 and #2403649 from the U. S. National Science Foundation. Any opinions, findings and conclusions or recommendations expressed in this material do not necessarily reflect the views of the NSF.