Object-Oriented Python Framework for Operational Graphics

Craig Hartsough[1,2], Christina Holt[1,2], and Brian Jamison[1,3]

[1] National Oceanic and Atmospheric Administration, Global Systems Laboratory (GSL)

[2] Cooperative Institute for Research in Environmental Sciences (CIRES)

[3] Cooperative Institute for Research in the Atmosphere (CIRA)

NOAA GSL Graphics

The Global Systems Laboratory (GSL) at NOAA in Boulder, Colorado, develops and tests many meteorological models and analyses, including the High-Resolution Rapid Refresh (HRRR) model. GSL provides graphical forecasts to the public using a public web interface: https://rapidrefresh.noaa.gov/hrrr/HRRR/Welcome.cgi.

Web graphics have been generated using the NCAR Command Language (NCL) for about 15 years. In 2019, NCAR made the decision to transition to Python for graphic display and discontinued support for NCL. GSL has used this opportunity to also move to Python scripting language to generate its graphics forecasts.

Advantages

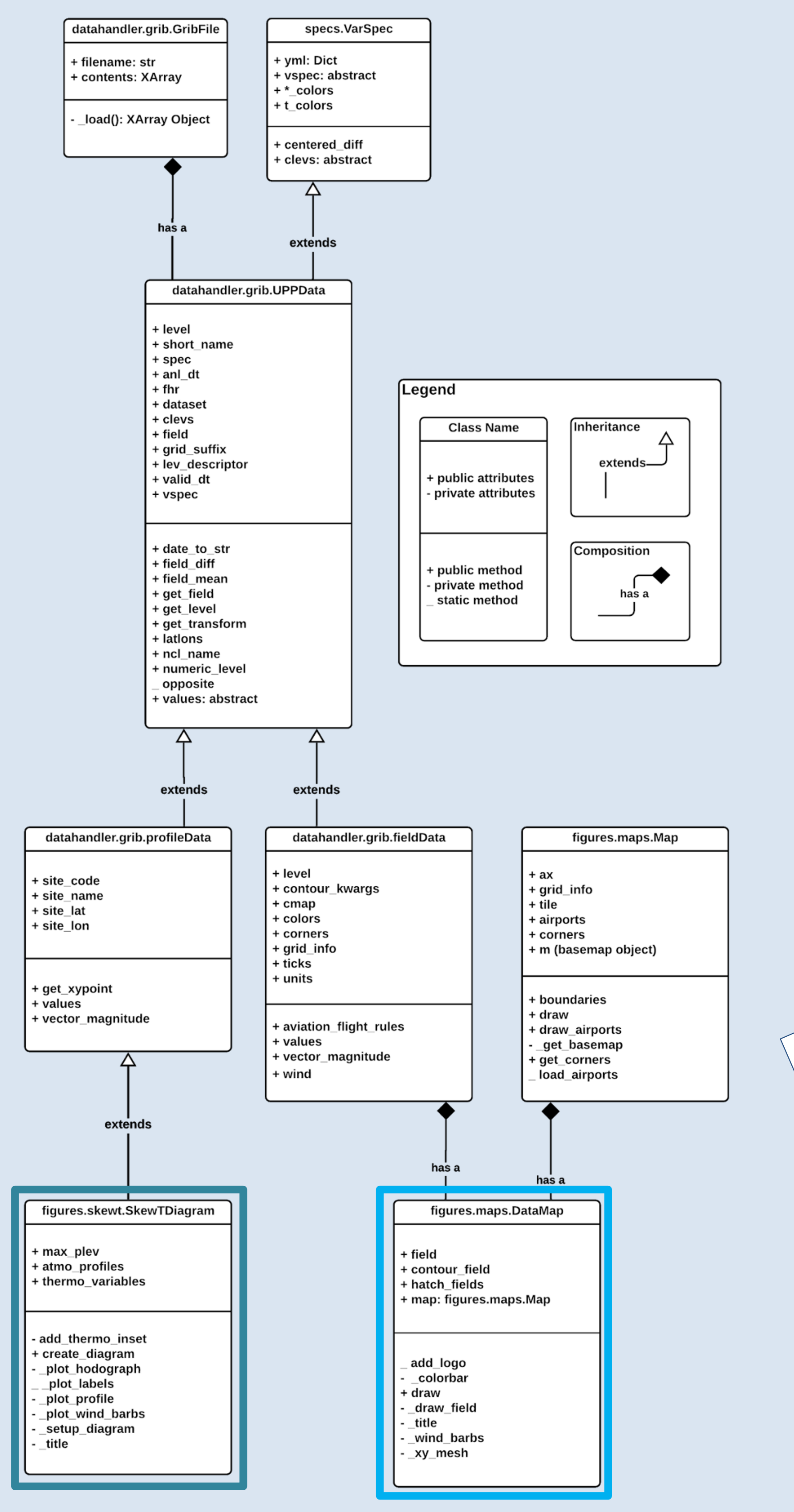
In contrast to our legacy NCL-based graphics system, Python scripting under an object-oriented framework has several advantages. The design allows for **minimal duplication of code** by moving all configuration to YAML files. **Parallelization**, **scripting**, and **graphics generation** can all be done with the **same language**. Python also supports a suite of graphics packages, and the resulting scripts are **extremely portable.**

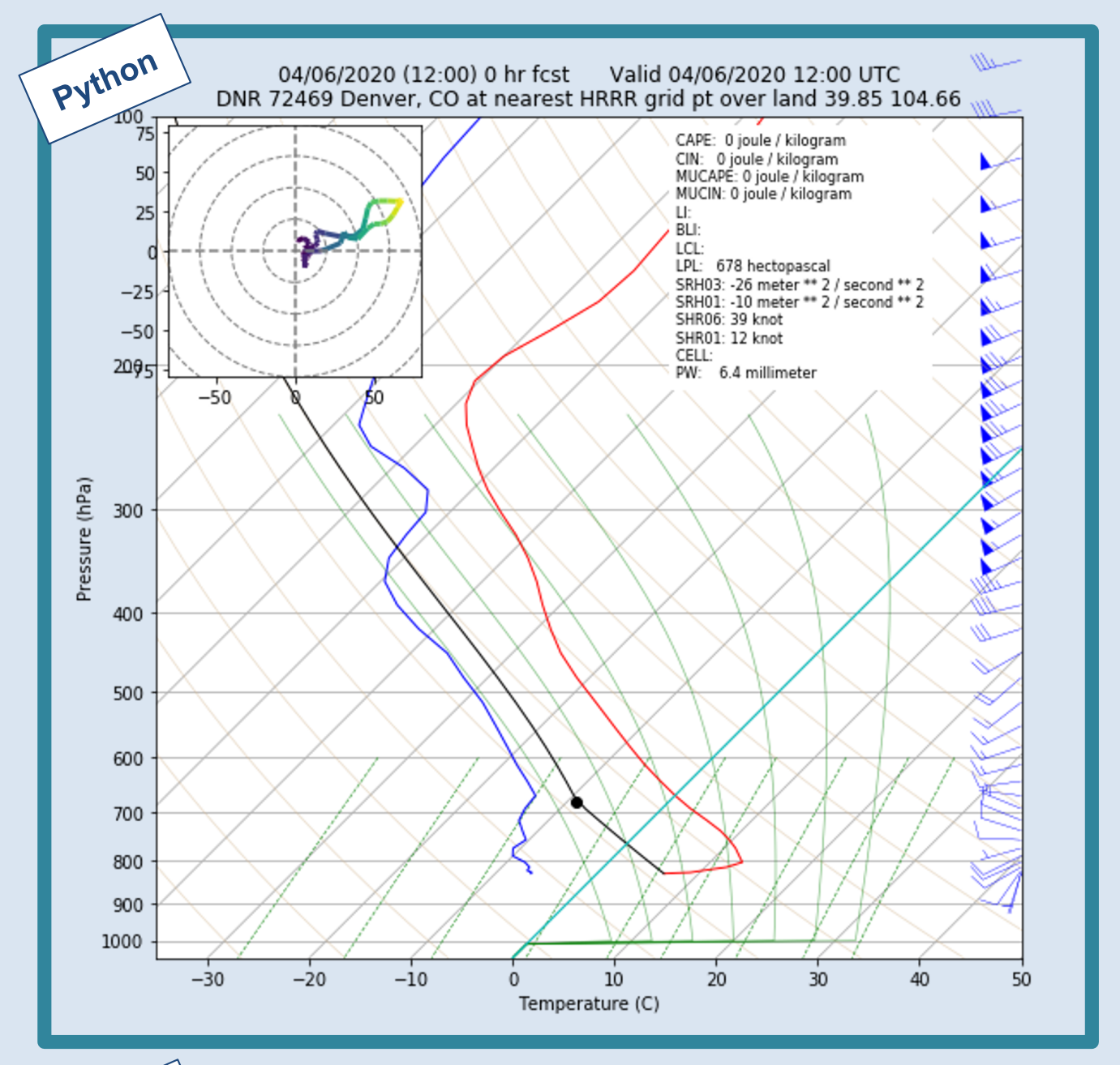
Capabilities and Code

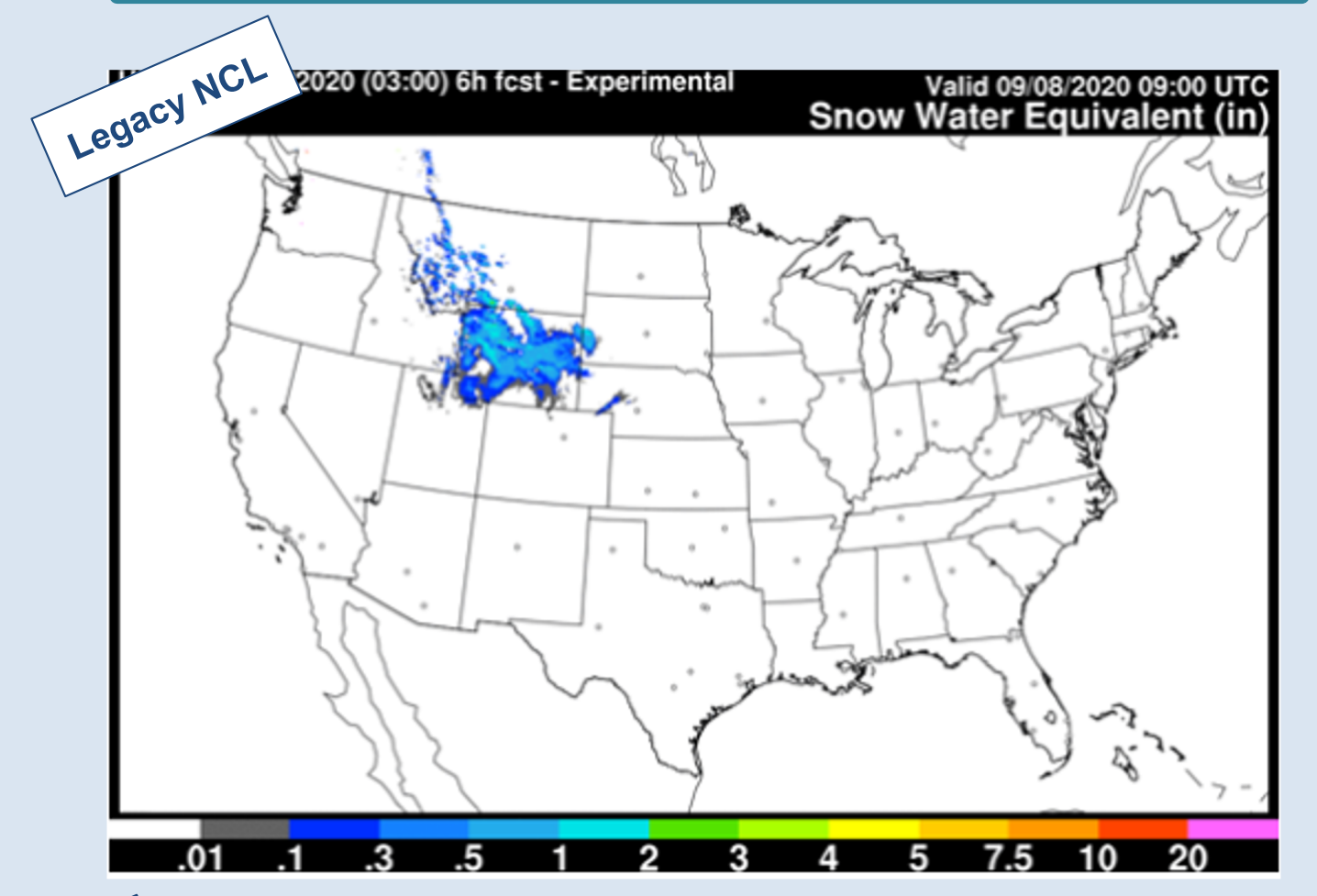
- Configuration-defined colors, contour levels, and variable transformations
- Configuration lists for choosing which maps to plot
- Plotting UPP output from RAP, HRRR, RTMA
- Plot "plan view" maps and SkewT diagrams at specified lat/lon
- Python driver for running map creation per forecast cycle
- Parallelization using Python Subprocess (single node)

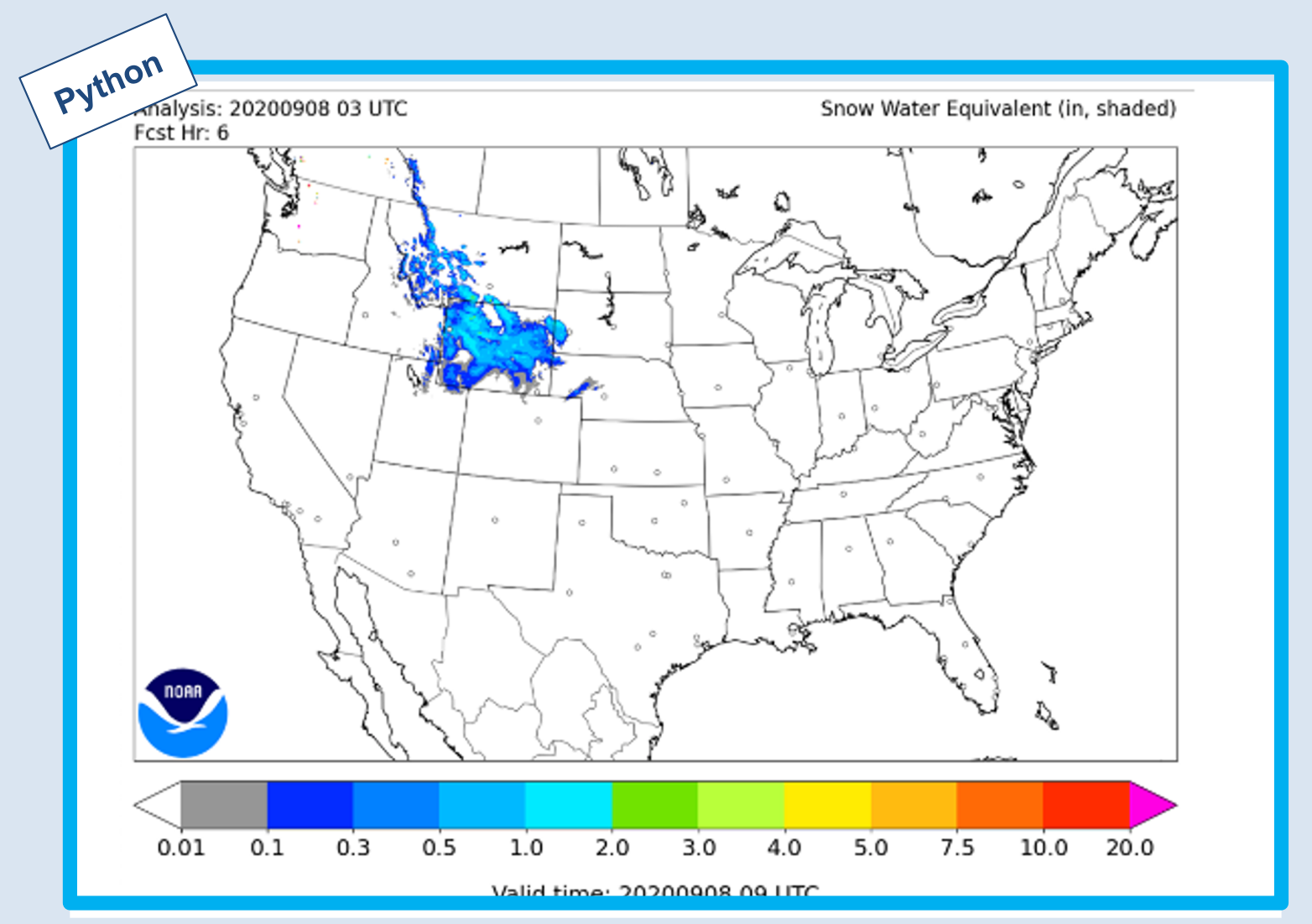
Try the code from the Official NOAA-GSL GitHub Page:
https://github.com/NOAA-GSL/pygraf

Framework (UML Diagram)







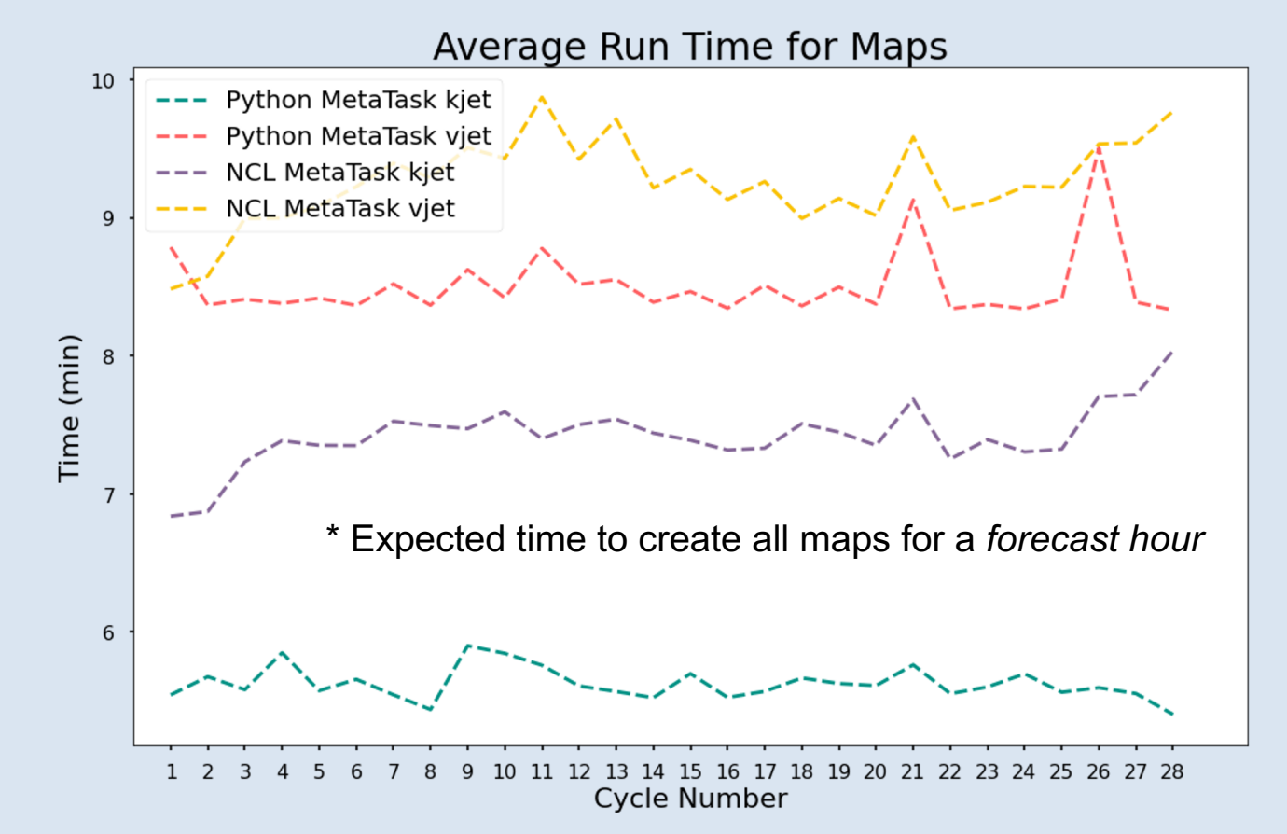


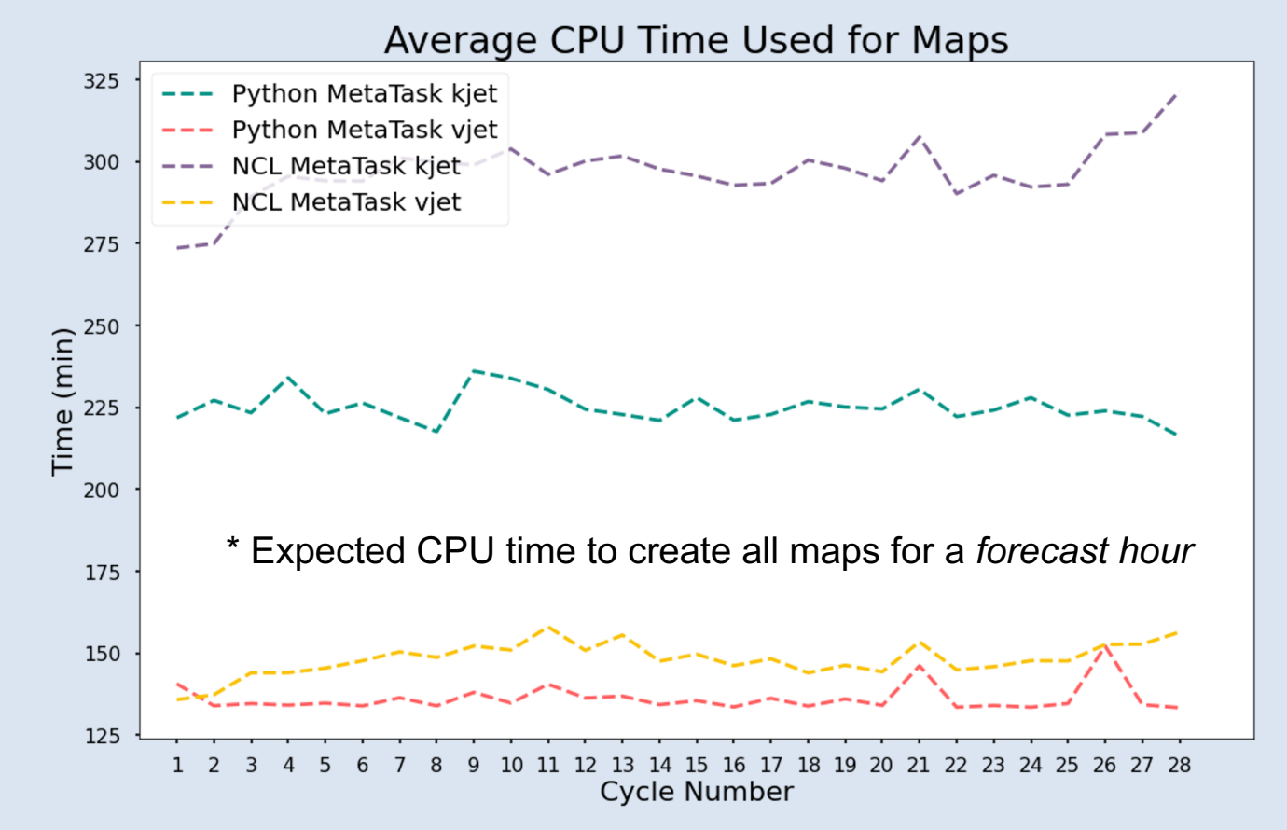
Ease of Use

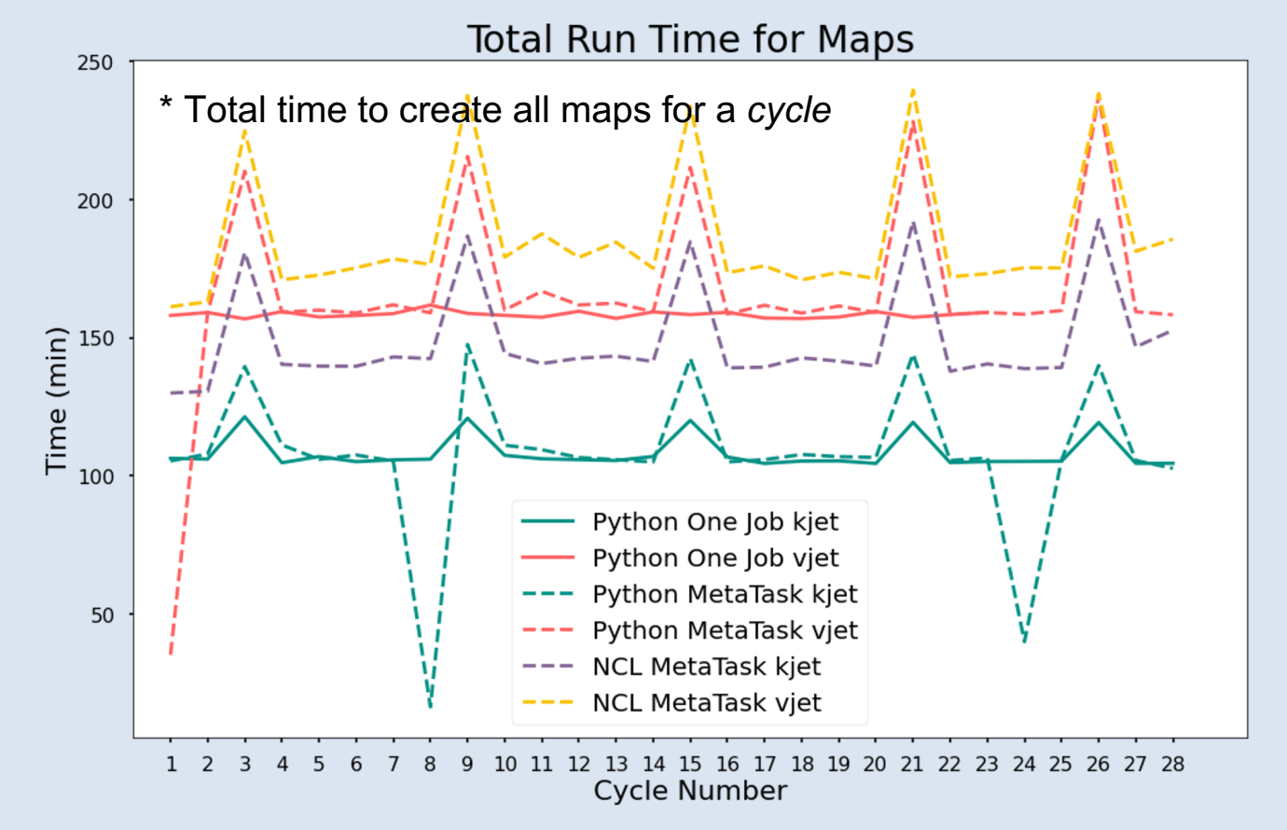
The new Python-based graphics package provides several advantages for our weather-modeling scientists

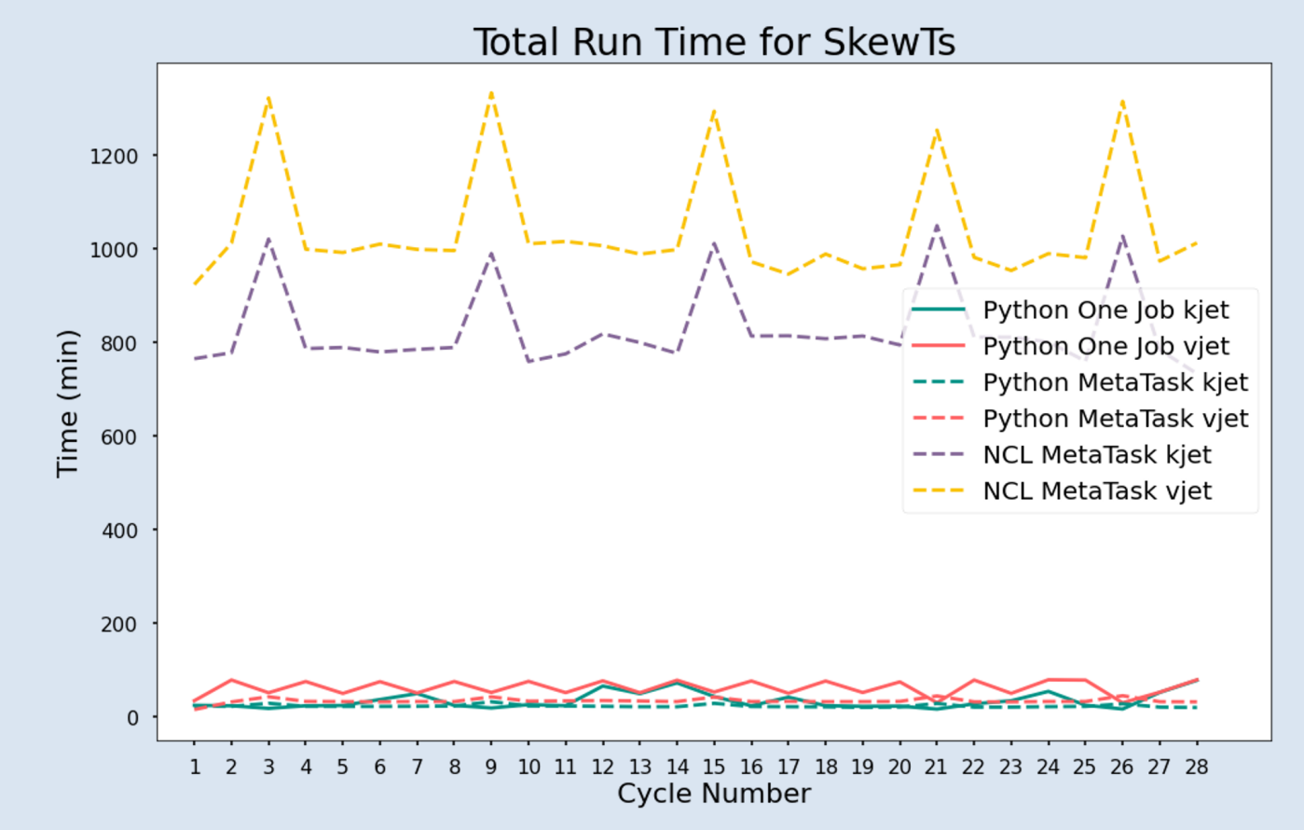
- machine-level support
- suitable for both research and real-time activities
- identical graphics for any UPP output
- parallelization at the workflow level, or job level to minimize resource usage
- industry standard approaches

Resource Usage









Strategies

- Two run strategies:
  - "**One Job**" (Python only) is a single batch job and the run script waits on new Grib2 files
  - **Metatask** (Rocoto construct) submits 1 batch job per forecast hour when Grib2 is available. AKA workflow parallelization.
- Both parallelize map creation using multiprocessing on a single node.
- Only Python uses multiprocessing for Skew T, serial in NCL with some workflow

	Cores/node	Mem/Core
kjet	40	2.4 Gb
vjet	16	4 Gb

Usage Experimental Setup:

- Real-time runs using 1-minute recurring cron with Rocoto
- 28 hourly cycles on Jet
- **420 Maps** per forecast hour -- 20 field maps for 21 tiles
- **91 SkewTs** per forecast hour
- 24 hour forecasts at SYNOP, 18 hr otherwise
- parallelization.

Contact

Craig Hartsough
CIRES at NOAA/GSL
Craig.Hartsough@noaa.gov

Christina Holt
CIRES at NOAA/GSL
Christina.Holt@noaa.gov

Brian Jamison
CIRA at NOAA/GSL
Brian.D.Jamison@noaa.gov