OMPS Level 2 Ozone Products in CSPP LEO

Geoff Cureton

Community Satellite Processing Package for LEO

Cooperative Institute for Meteorological Satellite Studies
University of Wisconsin - Madison

Introduction

- The CSPP OMPS Level 2 DB Package
- The OMPS Version 8 Ozone Total Column & Nadir Profile CCAPs
- From CCAP to CSPP package
- Trying out some new stuff
- Comparing CSPP DB and NOAA prod outputs
- Future Work

The CSPP OMPS Level 2 DB Package

- **The objectives of the this CSPP package are:**
  - Enable the user to run the NOAA Version 8 Total Column (V8TOz) Ozone, and the Nadir Profile (V8Pro) CCAPS in DB context.
  - Ease of installation (extract tarball, set a couple of environment variables)
  - Support S/NPP, NOAA-20, and NOAA-21 SDR HDF5 inputs.
  - Support aggregated inputs
  - Easy to use!

The OMPS Version 8 Ozone Total Column & Nadir Profile CCAPs

- **The V8TOz and V8Pro Cloud Containerized Algorithm Packages "CCAP"s generally contain (from the README docs):**
  - Source code, and relevant makefiles and/or build scripts
  - Statically compiled executables
  - Python wrapper scripts and configuration files
  - Dockerfile, docker image, and libraries/utilities needed to build the docker image
  - Sample test data, including inputs, reference outputs
  - Reference documents including the Delivery Memo, README, and the Production Rules

From CCAP to CSPP Package

- **Translating CCAPs into Direct Broadcast packages involves making some decisions about what to keep, and what to reimplement:**
  - Keep the compiled executable, and any static ancillary files (generally a hard dependency)
  - Don't use YAML files right now.
  - CSPP python scripting implements a CLI to accept input L1b files and any runtime options (which may be CSPP or CCAP relevant)
  - If production rules are simple, they are generally implemented in CSPP python code, which assembles units of work and writes PCF files or sets env vars
  - If production rules are complex, a thin CSPP wrapper for any CCAP scripting which implements the rules.

From CCAP to CSPP Package

- **The CSPP scripting tends to handle, at a minimum:**
  - Accepting input file/dir paths and globs, and cataloging the found files (and handling deaggregation)
  - Filtering inputs to construct available processing candidates
  - Mapping processing candidates into invocations of the CCAP exe (or CCAP wrapper script), as the base unit of work
  - Setting up parallel or asynchronous processing of work units as appropriate
  - Logging, cleanup

From CCAP to CSPP Package

- **Our Python based workflow requires a python runtime:**
  - For many years we used an in-house bespoke Python distribution "ShellB3", developed by Ray Garcia, containing the required libs, which was bundled in the top-level CSPP package tarball.
  - Maintaining ShellB3 was a lot of work, but just wrapping up a conda environment had portability issues, and some users didn't want to use the bundled OpenSSL files.
  - The advent of `conda-pack` devolved the work of assembling a runtime to the individual CSPP package dev (a good thing).
  - But what if we did something else…

Trying Some New Stuff

- **It's nice to get statically compiled CCAP binaries, so what if we did that for everything? :**
  - The original CSPP OMPS L2 package dev, Scott Mindock, replaced the CSPP python runtime with a statically compiled Rust binary
  - Rust has the high-level ergonomics of Python (via a rich package ecosystem), but with strict typing, mature CLI building, logging, and error handling
  - For CPU-bound tasks (like running CCAP binaries), parallel processing is almost trivial (using the Rayon crate).
  - Rather than dragging around a python runtime which can break if the user moves it, all the same functionality is in a single, statically compiled binary

Trying Some New Stuff

- **Aside from using Rust for the CSPP logic, there are some other new things:**
  - Much like the uv package has become very popular for python development, Pixi (`https://pixi.prefix.dev/`) is also being used in the Rust context.
  - Pixi is a "build manager" which seems like uv, but supports various compiled and scripted langs (C/C++, Rust, Fortran, Python, R…).
  - Rust's own Cargo package manager is leveraged to manage the Rust module relationships, while Pixi can be used to manage *how* binaries and libs are built (compiler and linker options…)
  - Intent is to eventually use Pixi to manage the CCAP exe, CSPP Rust exe, integration/packaging, testing, doc generation… in a single place.
  - So… does it work?

CSPP OMPS L2 Usage

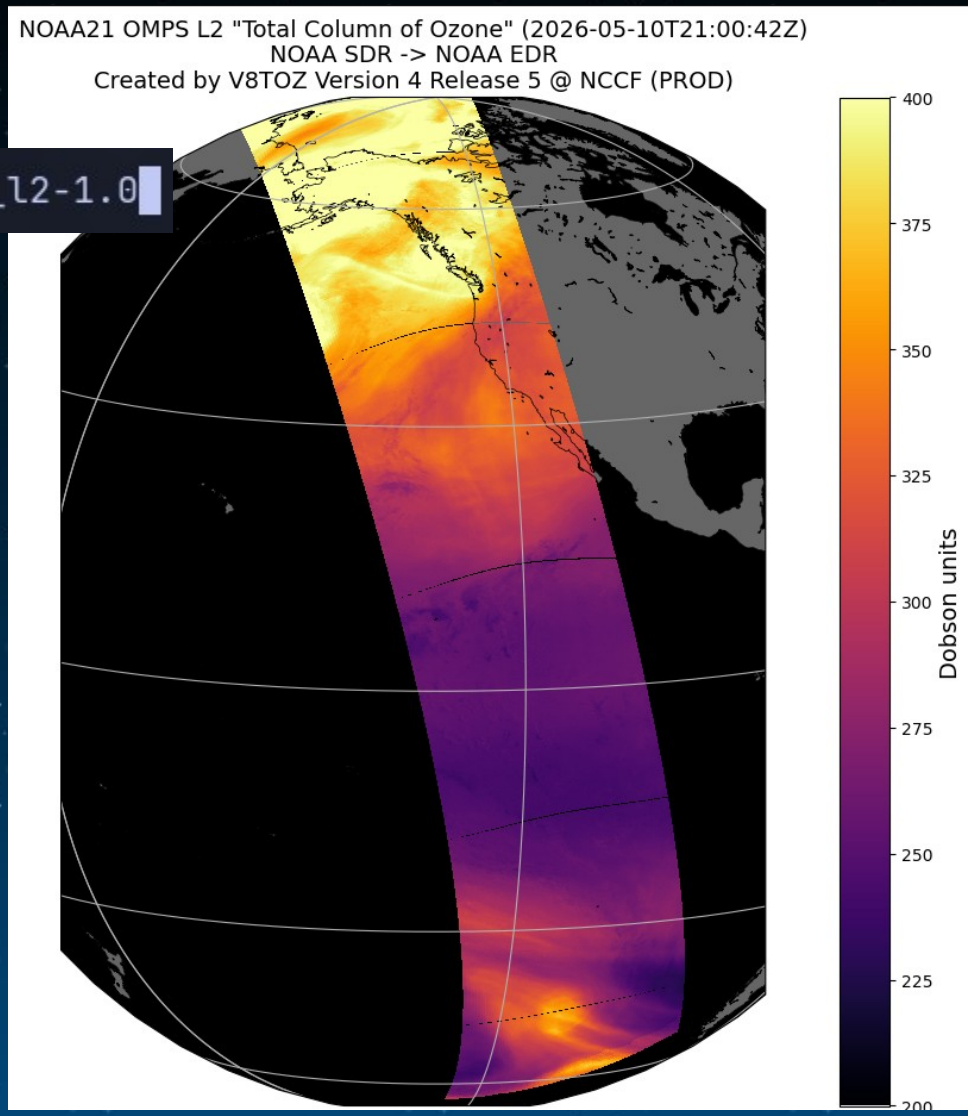
- Installation:

```
) tar xzvf cspp_omps_l2-1.0.tar.gz
) export CSPP_OMPS_L2_HOME=$PWD/cspp_omps_l2-1.0
) . $CSPP_OMPS_L2_HOME/omps_l2_env.sh
```

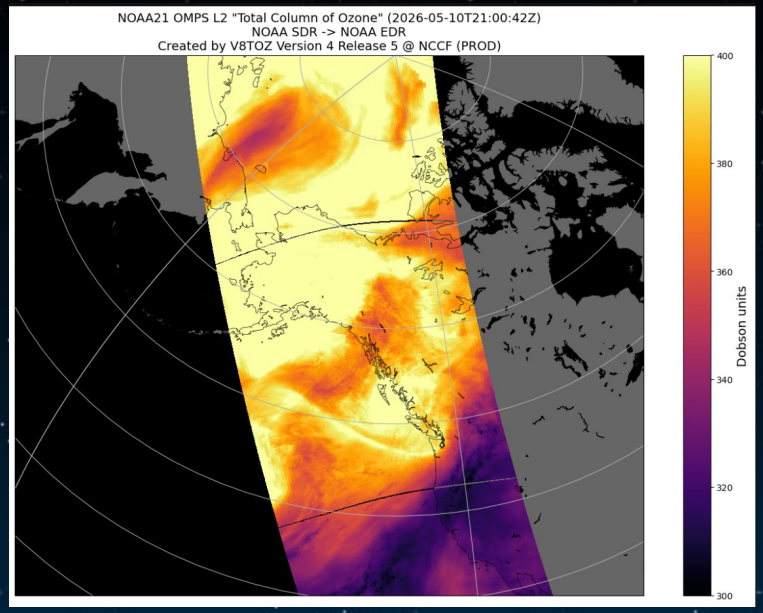
- Simplest CLI invocation:

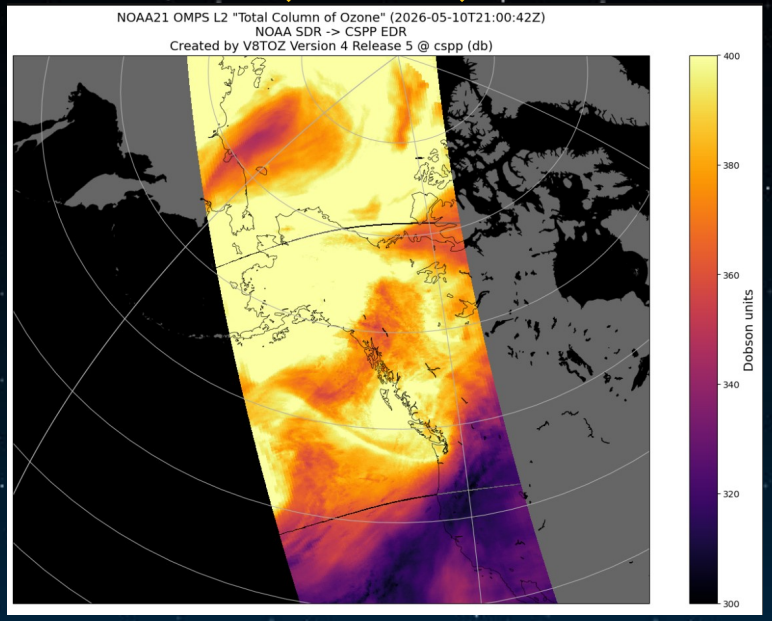
```
) omps_l2.sh -W work_dir inputs_dir
```

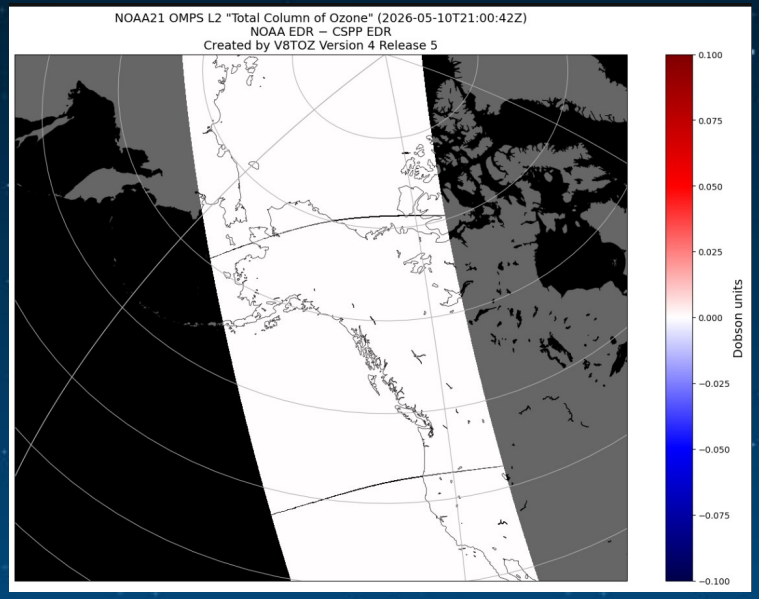
- Production Rules:
  - OMPS GEO and SDR files for both TC and NP must be in the same input dir
  - **V8TOz** requires TC GEO+SDR
  - **V8Pro** requires NP GEO+SDR, and bracketing TC GEO+SDR
  - Approx 4.5mins for 80 37.5s granules

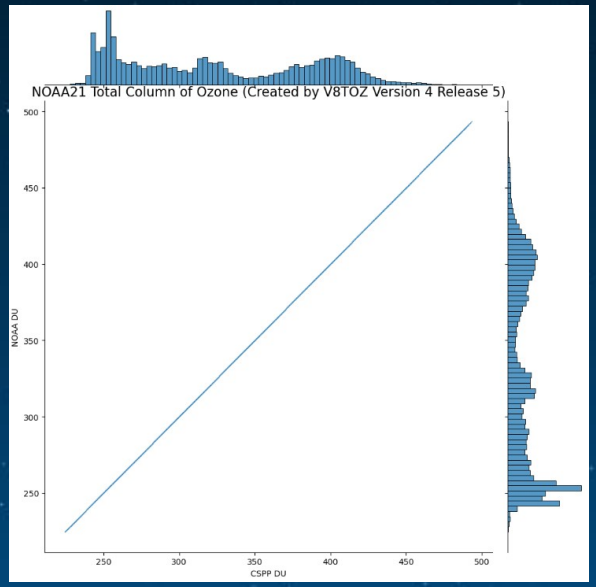


NOAA & CSPP V8TOz (N21)



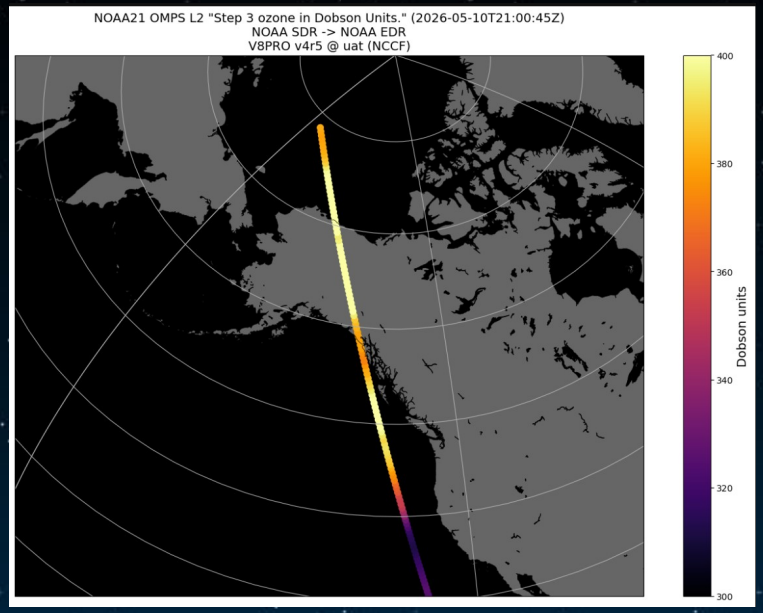


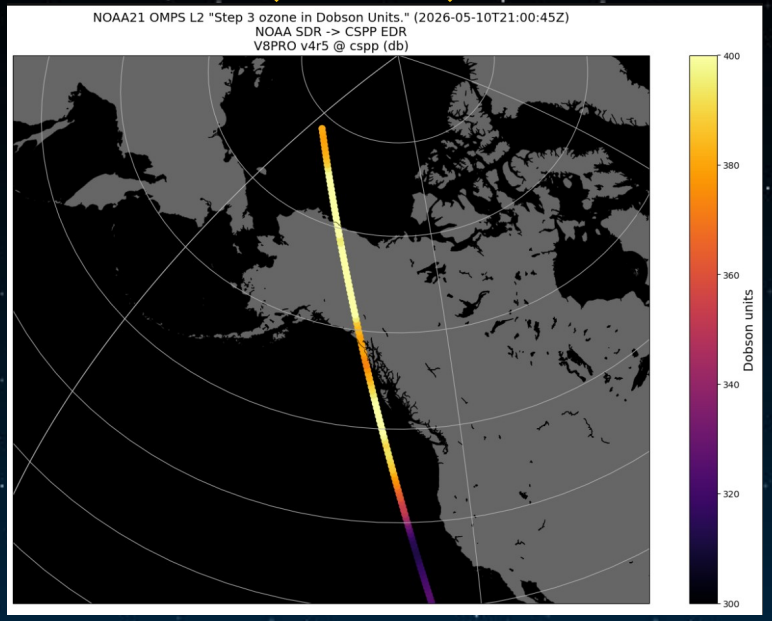


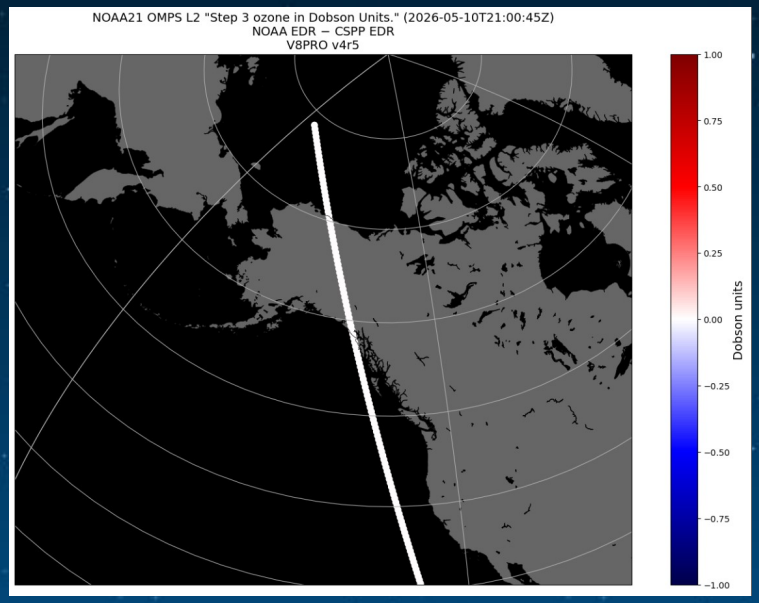


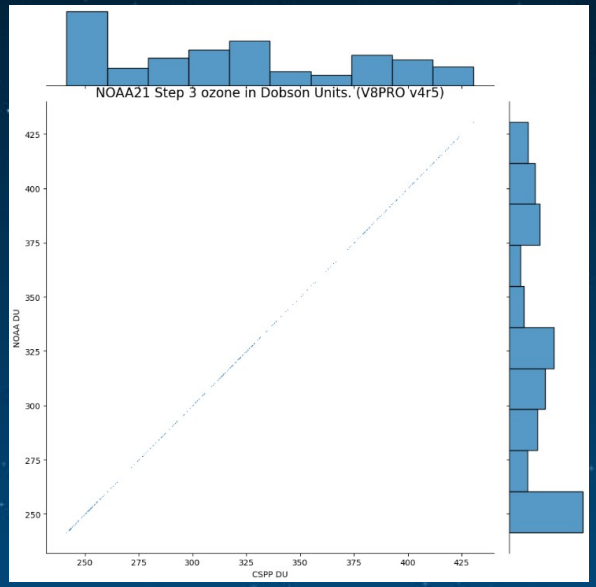
Alg version: v4r5

NOAA & CSPP V8Pro (N21)









Alg version: v4r5

Future Work

- CSPP OMPS L2 package supporting Version 8 Total Ozone and Nadir Profile CCAPS is currently in release-candidate evaluation
- Version 1.0 release expected June/July 2026
- SO_2 Corrected Version 8 Total Ozone (V8TOS): Q4, 2026

Contacts / Resources

- Geoff Cureton (geoff.cureton@ssec.wisc.edu)
- https://cimss.ssec.wisc.edu/cspp/
- https://forums.ssec.wisc.edu