Weather Forecasting at Home

Open-Source Chemical Transport Modeling

Speaker: Sebastian Val

Track: Applied Science

Repo: https://github.com/sval-dev/wrf-docker

Presentation: https://sval-dev.github.io/wrf-docker

Everyday encounters

- How does the weather app on your phone know it’s going to rain at 3:00 PM?
- How do environmental agencies predict wildfire smoke plumes days in advance?
- How do air quality or pollen alerts get generated?
- Massive open-source fluid dynamics and chemistry engines running on Linux clusters.

What is a Chemical Transport Model (CTM)?

- A CTM is a numerical simulation that predicts how gases and aerosols behave in the atmosphere.
- It solves complex, non-linear equations for fluid motion, thermodynamics, and chemical kinetics.
- This gives us information such as predicted temperature, relative humidity, dew point, and chance of percipitation (rain or snow).
- Also allows us to forecast pollutants like ground-level Ozone (O3) and Fine / Coarse Particulate Matter (PM_2.5 / PM_10).

The Open-Source Science Ecosystem

- Atmospheric modeling is heavily reliant on open-source software (OSS) and Linux.
- The majority of modern numerical weather prediction (NWP) models are built using Fortran and C.
- They are designed explicitly to be compiled and executed on Linux-based High-Performance Computing (HPC) systems.

The "Dependency Hell" Problem

- Compiling legacy Fortran code from scratch is notoriously difficult.
- It requires compatibility between specific compiler versions (GNU `gfortran`, Intel `ifort`), MPI libraries, and specific builds of NetCDF and HDF5.
- A single missing shared object library will instantly halt the build.
- A single conflicting / mismatched shared object library can cause crashes at runtime.

Reproducibility

- Reproducibility
- https://www.intel.com/content/dam/develop/external/us/en/documents/pdf/fp-consistency-121918.pdf

Our OSS Solution Today

- We bypass dependency hell using **Linux Containers / Linux Namespaces**.
- We'll use Docker to isolate the OS, compilers, libraries, and model binaries into a single portable image.
- Alows running a high-resolution weather and air quality forecast over Pasadena, California, or any other location, entirely on our local machines.

Global Forcing: The Atmosphere Has No Borders

- To run a regional model over California, we can't simulate in a vacuum.
- The physics equations require starting values (initial conditions) and information about conditions on the edges of our map (boundary conditions).
- We rely on open-data global models to feed our regional model.

The Operational Standard: UFS

- When the public sees a forecast from the National Weather Service, it is often driven by the Unified Forecast System (UFS).
- UFS is NOAA’s next-generation, community-based Earth system model.
- It utilizes the Finite-Volume Cubed-Sphere (FV3) dynamical core to solve global fluid dynamics.
- https://github.com/ufs-community/ufs-weather-model

Our Engine Today: NASA GEOS-5

- While UFS is the operational weather standard, we are using NASA's GEOS-5 (Forward Processing) global forecast in this model.
- It provides us with a high-fidelity global distribution of aerosols to feed our regional boundaries.
- It also uses FV3
- Data Products: https://portal.nccs.nasa.gov/datashare/gmao/geos-fp/forecast/
- Underlying CTM / GCM Software: https://github.com/GEOS-ESM/GEOSgcm

Downscaling with WRF-Chem

- Global models like GEOS-5 give us the macro-scale circulation (typically ~25km or 12.5km resolution; 0.25 deg or 0.125 deg).
- To get street-level, high-precision forecasts, we use the **Weather Research and Forecasting model coupled with Chemistry (WRF-Chem)**.
- geos2wrf takes the global data and dynamically "downscales" it to localized, high-resolution grids.
- https://modelingguru.nasa.gov/docs/DOC-2242

Offline vs. Online Coupling

- Historically, air quality was run "offline": A weather model runs, writes its output, and then a chemistry model reads that output to process kinetics.
- **Problem:** No physical feedback.
- WRF-Chem is "online-coupled." The weather and chemistry are solved simultaneously on the exact same computational grid.

The Power of Online Feedback

- If a massive wildfire emits a dense smoke plume, WRF-Chem calculates the physical presence of the particulates.
- The model's radiation physics see the smoke, calculate how it blocks sunlight, cools the surface, and changes the local wind patterns.
- The changing wind then alters the path of the smoke.

Pasadena, CA example

- We are configuring our model to focus on Pasadena, California (here!).
- Interesting and complex coastal-mountain topography + pollution dynamics.
- Pasadena is bordered by the Pacific Ocean to the west and "walled in" by the steep San Gabriel Mountains to the north and San Bernidino Mountains east.
- Local weather pattern creates thermal inversions (warm air above trapping cooler air below) causing polluted air at the surface.

Modeling Smog

- Daily solar heating generates a strong onshore sea breeze.
- This wind sweeps vehicular and industrial emissions from the dense Los Angeles core directly inland.
- The mountain ranges act as a barricade, causing Volatile Organic Compounds (VOCs) and Nitrogen Oxides (NOx) to get trapped and heat in sunlight to form Ozone (O3).

Feeding the Model: Input Requirements

Before WRF-Chem can simulate the future, it needs:

- **1. Dynamic Data:** 3D meteorological (Temp / RH / Precip) and aerosol boundary conditions.
- **2. Static Data:** High-resolution geographical mapping to define the physical surface of the earth.

GEOS-FP Chemical Variables

As an enhancement, we could add additional GEOS-FP aerosol information:

- SO2 (Sulfur Dioxide)
- SO4 (Sulfate aerosols)
- BC (Black Carbon)
- OC (Organic Carbon)
- Dust and Sea Salt

NetCDF: The Scientific Standard

- NetCDF files are self-describing.
- They contain internal metadata specifying map projections, variable descriptions, and scientific units (e.g. ug/kg or ppm)
- CF-conventions: https://cfconventions.org/
- Binary files but lots of readers

Static Terrestrial Datasets

- The model must know how the ground behaves to calculate friction, heat retention, and moisture evaporation.
- We use datasets from the USGS and the MODIS satellite to map terrain elevation, soil categories, and land-use classifications down to 500-meter resolutions.
- Crucial for resolving Pasadena's varying microclimates.
- https://www2.mmm.ucar.edu/wrf/users/download/get_sources_wps_geog.html

The WRF Preprocessing System (WPS)

- How do we combine NASA global data and satellite terrain data together?
- We use a pipeline of programs called the WRF Preprocessing System (WPS).
- They are all controlled by a single Fortran namelist file: `namelist.wps`.

Nested Grid Architectures

- Regional models use nested grids to balance high resolution with computational cost.
- **Production Setup:** A 36km domain for the Western US, a 12km domain for California, and a 4km domain over Los Angeles.
- WRF Domain Wizard: https://jiririchter.github.io/WRFDomainWizard/

Example files: The "Toy" Grid

- We will use a simplified "toy" configuration as our example.
- **Outer Domain:** 12km grid.
- **Inner Domain:** 4km grid tightly bounding Pasadena.
- Center Coords: 34.144, -118.144
- Computationally cheap, but demonstrates the nested architecture mechanics

WPS Step 1: `geogrid.exe`

- `geogrid.exe` defines the physical geometry of our simulation domain.
- It reads `namelist.wps` for our map projection, center coordinates, and grid spacing (dx, dy).
- It extracts the static MODIS/USGS data and interpolates it onto our specific mesh.

WPS Step 2: `geos2wrf`

- Global models like GEOS-5 package their data in netCDF.
- `runGeos2Wrf.sh` unpacks the netCDF fields into that metgrid format.

WPS Step 3: `metgrid.exe`

- The final preprocessing step bridges the static map with the dynamic weather.
- `metgrid.exe` ingests the static grids from `geogrid` and the time-stamped weather from `geos2wrf`.
- It horizontally interpolates the coarse global data perfectly onto our high-resolution 12km/4km nested grids.

Initializing the Atmosphere

- Moving from WPS to WRF-Chem begins with `real.exe`.
- `real.exe` takes our horizontal data and interpolates it vertically across desired number of hydrostatic pressure layers (or levels) up to the stratosphere.
- It generates our final `wrfinput` (initial state) and `wrfbdy` (boundary conditions) files.

The Control Panel: `namelist.input`

- The entire numerical execution of WRF-Chem is dictated by a massive configuration file called `namelist.input`.
- It controls time integration steps, spatial dimensions, physics parameterizations, and chemical kinetics.
- https://github.com/wrf-model/WRF/blob/master/run/README.namelist

Configuring Chemistry

- For air quality, the `&chem` record is vital. The `chem_opt` parameter determines our chemical mechanism.
- `chem_opt = 0`: Standard weather forecasting (chemistry disabled).
- `chem_opt = 300`: GOCART only (bulk aerosols, computationally fast).
- Goddard Chemistry Aerosol Radiation and Transport (GOCART)

MOZART + GOCART for Ozone

- For our Pasadena example, we utilize a pairing of MOZART and GOCART (`chem_opt = 112`).
- Combines GOCART with Model for Ozone and Related chemical Tracers (MOZART)

Containerizing the Workflow

- Executing this natively requires compiling many packages of Fortran and C code, linking compatible libraries, and choosing MPI paths.

- WPS Container
- WPS with geos2wrf Container
- WRF-Chem Container
- Dev environment Container

Executing with Docker Bind Mounts

- Containers are isolated. How do we get our GEOS-FP input files in and our output netCDF files out?
- We launch Docker using Volume Bind Mounts:
  `-v /local/host/data:/wrf/data`
- The container reads/writes directly to the host's volume, ensuring our output persists after the container exits.

Launching the Container

```
# Launching the WPS environment
docker run --rm -it --name wps_pasadena -v $(pwd)/data:/wrf

# Launching the WRF-Chem environment
docker run --rm -it --name wrf_pasadena -v $(pwd)/data:/wrf
```

High-Performance Computing: MPI

- Simulating fluid dynamics and complex photochemistry simultaneously is computationally heavy.
- WRF-Chem achieves speed through domain decomposition.
- The map is sliced into rectangular tiles (patches). Each tile is assigned to an individual CPU core.

Patch Flow

- The atmosphere is a continuous fluid so tiles must constantly talk to each other.
- At every time step, WRF-Chem exchanges wind, heat, and chemical fluxes across the outer boundaries of each patch.

Executing the Simulation

```
# Executing WRF-Chem across 8 CPU cores using MPI
cd /wrf/data
mpirun -np 8 /WPS-run/geogrid.exe
mpirun -np 8 /WPS-run/metgrid.exe
/geos2wrf_merra2wrf/scripts/G5.7.2/runGeos2Wrf.sh
mpirun -np 8 /usr/local/bin/real.exe
mpirun -np 8 /usr/local/bin/wrf.exe
```

The Output: NetCDF

- WRF-Chem writes its results in the Network Common Data Form (**netCDF**) format.
- It generates files containing 4-Dimensional arrays:
- Longitude (x)
- Latitude (y)
- Vertical Pressure Levels (z)
- Time (t)

Rapid Prototyping: `ncview`

- `ncview` is a lightweight, X11-based visual browser.
- https://cirrus.ucsd.edu/ncview/

```
# Launching ncview on our completed forecast file
ncview wrfout_d02_2026-03-06_18:00:00
```

High-Fidelity Mapping: NASA Panoply

- **Panoply** is a cross-platform Java GUI developed by NASA Goddard Institute for Space Studies (GISS).
- Panoply is good at slicing 3D/4D arrays into 2D contours and overlaying high-resolution continent outlines and state borders.
- https://www.giss.nasa.gov/tools/panoply/download/

Programmatic Analysis: Python

- For actual analytics, we can use Python.
- netCDF4 module in PyPI
- `xarray` provides convenient manipulation for multi-dimensional netCDF arrays.
- `wrf-python` provides higher level routines for extracting particular pressure levels and calculating derived meteorological variables.
- https://wrf-python.readthedocs.io/en/latest/

Democratizing Atmospheric Science

- Historically, this level of modeling was restricted to massive institutional supercomputers.
- Today, through the convergence of Linux, open-source Fortran/C code, and containerization via Docker, the barrier to entry has been substantially lowered.

Reproducible research

- By standardizing inputs (GEOS-5), containerizing the environment (Docker), and scripting the outputs (Python/xarray), your research becomes more reproducible.
- You can push your docker image and inputs to a peer, and they can replicate your atmospheric forecast.

Summary

1. **Inputs:** GEOS-5 Global Data + MODIS Terrestrial Maps
2. **Pre-Processing:** WPS (`geogrid`, `geos2wrf`, `metgrid`)
3. **Simulation:** WRF-Chem (`real.exe` and `wrf.exe`)
4. **Analysis:** NetCDF via `ncview`, Panoply, and Python

The future: AI Foundation Models for Weather & Climate

From a NOAA News Release:
The new suite of AI weather models includes three distinct applications:

- **AIGFS** (Artificial Intelligence Global Forecast System): A weather forecast model that implements AI to deliver improved weather forecasts more quickly and efficiently (using up to 99.7% less computing resources) than its traditional counterpart.
- **AIGEFS** (Artificial Intelligence Global Ensemble Forecast System): An AI-based ensemble system that provides a range of probable forecast outcomes to meteorologists and decision-makers. Early results show improved performance over the traditional GEFS, extending forecast skill by an additional 18 to 24 hours.
- **HGEFS** (Hybrid-GEFS): A pioneering, hybrid "grand ensemble" that combines the new AI-based AIGEFS (above) with NOAA’s flagship ensemble model, the Global Ensemble Forecast System. Initial testing shows that this model, a first-of-its kind approach for an operational weather center, consistently outperforms both the AI-only and physics-only ensemble systems.

Get Involved

- The software is free. The global satellite data is free.
- Run it on your laptop, push it to AWS, or deploy it on an HPC cluster.
- Start forecasting the air quality in your own backyard!

Questions?

- **GitHub:** github.com/sval-dev/wrf-docker
- Presentation slides and Dockerfiles are available in the repo.