Building an MPAS App with the Unified Workflow Tools

Janet Derrico, Emily Carpenter, Christina Holt

The Rapid Refresh Forecast System (RRFS) development team leads at NOAA EMC, NOAA GSL, and NOAA NSSL recently published a recommendation that the next version of the RRFS transition to using the MPAS dynamic core[1]. They showed that the FV3 dynamic core is likely the source of large biases in storm intensity and rainfall and that transitioning to MPAS will address this.

As a result, researchers at NOAA GSL are beginning thorough testing of the MPAS model, including retrospective case studies and running real-time experiments for NOAA Testbeds. To support this rigorous, fast-moving research, the UFS Unified Workflow (UW) team worked closely with NOAA GSL to build an MPAS Application. We leveraged the experience of those who came before us at NCAR, NSSL, and GSL to include a portable build system that utilizes spack-stack-installed modules for the libraries and packages for building the developmental codes. The MPAS App installs Miniconda to support the Python environments needed for codes that configure and run a workflow with Rocoto.

While other applications are in the early stages of longer-term transition plans for incorporating the tools into their existing code bases, this application marks the first fully functional Unified-Workflow-enabled system. At its core, the GSL MPAS App configures a set of UW Drivers to run the components needed to produce an MPAS regional cold-start forecast from GFS initial and lateral boundary conditions. A driver's responsibility is to provision a run directory that upholds the contract required by the software it's running and then executes that software. The application is responsible for tracking common sets of configuration settings that work for the users of that application. While the drivers can still be configured manually to do things the application may not have considered, the default behavior of any of the application's supported workflows will be version-controlled. These defaults may include physics choices, meshes, forecast length, etc. The configuration of the application will also check to make sure that the choices made by the user are internally consistent with what we know to be scientifically valid.

Because several components need to be run in a specific order on a batch system, we include the Rocoto workflow manager to help automate larger experiments. The uwtools package provides a YAML interface tool that further unifies the workflow generation configuration language for the MPAS App. Again, the user has full control to create a workflow composed of any tasks they prefer, but the default behavior of the basic cold start configuration requires little human manipulation to run. In fact, we attempt to limit the required settings to start and end dates, which platform the experiment is running on, and the user's account information.

By preparing this App as a short-term solution, we hope that the NOAA GSL research staff working with MPAS will have a comfortable, usable environment in which to progress their science while many other UFS-related logistics are decided. The ultimate goal is to include the MPAS dycore in the ufs-weather-model and to have MPAS as an optional choice in the UFS Short Range Weather App. Due to the unified nature of the tools and framework, the experience we gain from the GSL MPAS App using a standalone MPAS model will be directly applicable in the UFS SRW App since both will (by that time) use the same configuration language and framework.

[1]Carley, Jacob, et al. "Mitigation Efforts to Address Rapid Refresh Forecast System (RRFS) v1 Dynamical Core Performance Issues and Recommendations for RRFS v2", 2023, https://doi.org/10.25923/ccgj-7140