Summary of UFS Workflows workshop follow-up meeting (29-30 June 2021, virtual)

Editors:

Mrinal Biswas, Ben Cash and Michael Ek

Participants:

Alper Altuntas (NCAR)
Arun Chawla (NOAA/EMC)
Benjamin A Cash (GMU/COLA)
Christina Holt (CU/CIRES at NOAA/GSL)
Christopher Harrop (CU/CIRES at NOAA/GSL)
Evan Kalina (CU/CIRES at NOAA/GSL, DTC)
Hendrik Tolman (NOAA/STI and UFS-SC)
Jeff Beck (CSU/CIRA at NOAA/GSL, DTC)
Kate Friedman (NOAA/EMC)
Ligia Bernardet (NOAA/GSL, DTC)
Louisa Nance (NCAR, DTC)
Mariana Vertenstein (NCAR)--invited, though couldn't attend
Michael Ek (NCAR, DTC)
Mrinal Biswas (NCAR, DTC)
Rahul Mahajan (NOAA/EMC)
Rocky Dunlap (NCAR)
Stylianos Flampouris (NOAA/STI, Present Affiliation: Raytheon)
Walter Kolczynski (IMSG at NOAA/EMC)

Introduction

This is a report on the Unified Forecast System (UFS) Workflows meeting held 29-30 June 2021, which was a follow-up to the UFS Workflows workshop held virtually 29-30 April 2020. The June 2021 meeting was spread over a two-day period (each 2-hour duration). The organizing committee included UFS collaborators from the Developmental Testbed Center (DTC), George Mason University (GMU), the National Oceanic and Atmospheric Administration (NOAA), and the National Center for Atmospheric Research (NCAR). The logistics of the virtual meeting were provided by the DTC. The aim of the workshop was to continue communications across the modeling groups and translate communication into concrete steps to improve workflows across the UFS. The meeting was attended by 17 participants, which included participation from Environmental Modeling Center (EMC)'s management and UFS leads.

This report provides an overview of the presentations and a summary of the extensive discussions that followed.

Requirements of a workflow

Arun Chawla (NOAA/NWS/EMC) presented "Workflow requirements" on the first day of the workshop. His presentation revisited and emphasized the issues caused by having numerous independent workflows within the UFS applications. There was general agreement that future development should aim to develop a flexible array of tools that can be used to configure, execute and process a variety of jobs as defined by an application suite. It was emphasized that the aim is not to find a single workflow for all applications but to build a system that is sustainable and works as a composable toolset. All the workflow components were discussed at length and the requirements list was modified accordingly. Below are the arrived-at requirements of a workflow system.

(a) Skeleton of a workflow system:

1. Modular and Configurable:
    - Broken down to sub-tasks that allow for efficient use of resources, e.g. minimize total time to completion (or makespan) and maximize allocated CPU usage.
    - Easily allows a task to be reconfigured, switched on, or off.
    - Common repeatable parts are separated out as generalized functions/methods.
2. Documentation is critical:
    - Detailed guides on how to use and port system
    - Every function/script should have inline documentation on -- a) purpose; b) input/output; c) use case.

3. Portability:
    - Workflow systems should be easily portable to multiple platforms (HPC, non-HPC, Cloud).
    - Should use language easily available on all platforms (Python 3 for example).
    - System-specific information should be set via easily accessible configuration files, as opposed to buried in scripts.
    - All paths specified through configuration files (no paths in any script unless relative).
    - Should contain no source code within the workflow framework.
    - Utility/aux codes needed should be easily built and ported (e.g. CMAKE + HPC-Stack).
    - Libraries/packages needed should be easily deployable (use third-party libraries as needed but judiciously).
4. Workflow Engine:
    - Should be interfaced generically with workflow engines used by operations and community (ecflow/cylc/rocoto)
    - Should work without a scheduler (e.g. slurm) on a workstation (have the ability to mix scheduled and non-scheduled jobs)
    - Should be able to add new or other workflow engines e.g. airflow, Amazon SWF
5. Configurable for multiple applications:
    - UFS-based applications that cover a range of use case conditions:
        - Fully-coupled global system ((d)atm+waves+ocean+ice+chemistry) with multiple options.
        - A regional system with multiple options.
        - Data Assimilation (DA) (var+ens.), forecast only, or DA+forecast (Note: DA itself will have multiple options).
        - A hurricane modeling system.
        - Deterministic/ensemble mode.
        - Single column/hierarchical testing.
    - Observation Ingest and Processing:
        - Decoding, tanking (storing), dumping (concatenating or grouping) and filtering, etc.
    - Post-processing, product generation, validation, and verification.
6. Configuration management system:
    - A workflow system will be driven by a range of configuration files + namelist options. Configuration files will be YAML or equivalent.
    - A GUI-based tool with an easy to use interface to build the configuration files and perform basic consistency checking based on choices made by the user.
    - Once created, these files can be shared/changed to run the workflow systems (i.e. the management system is not needed to run the workflow).

(b) Implementation:

Two options for how workflow(s) can be used or shared by many applications were presented for discussion. The first was to build a single workflow that satisfies all requirements that is hosted in a single repository and works for a range of environments. The second approach presented was to build a set of generic and abstract tools that can be used and implemented with different applications. In this second approach, the majority of the tasks would be performed by common libraries and packages, with the configuration separating the workflows for different applications.

The first option was unanimously deemed to be impractical. The second option was the preferred one because it unifies workflows while providing independence for different applications to address their concerns. It will make porting the workflow simpler on different platforms. It will follow agile software development practices in implementing components instead of replacing the whole workflow system. This will be flexible enough to create new workflows in the future.

(c) Project Management:

The proposed work will be executed by a single project manager. All of the development will be in the public repositories (e.g. Github). A lead software engineer will direct solutions and build generic toolsets in consultation with representatives from different applications who should contribute to testing for agile development. A CI system with coding standards, linters (automated code checking), and testing protocols should be in place from the beginning. There will be Live Documentation and regular interactions to share accomplishments, ideas, and concerns.

Perspective from UFS management - Hendrik Tolman

Hendrik Tolman's presentation stressed the need for a unified system (not unitary) with focused resources allowing diversity. Requirements for fast cycling systems (reduce latency by running DA and the model together in-core), and slow cycling systems (apply appropriate resources to DA and model separately, hence running DA and model sequentially) are not compatible. Therefore a single workflow does not seem feasible. One of the key points was to design systems starting from requirements (software and system engineering) and not from solutions. Software engineering perspective includes the hierarchy of running the codes of each step of a workflow (e.g. obs processing, DA etc) (UFS code), scripts to run the tasks ("ex" scripts), managing file I/O, configure and running of the code, functional scheduler (job run sequence) and finally the system scheduler (ECflow). The system engineering includes (1) the elements of a workflow

e.g. input data processing, model initialization, producing forecast and post processing of output; (2) the workflow should be flexible, modular, and efficient for both researchers and operations; and (3) workflow should have elements that can be shared by different applications. The above requirements-based assessment shows that a “unitary” workflow is not likely to be feasible, but that instead a library of systems engineering and software engineering-based tools could be the foundation for a Unified approach to customizable workflows.

Working with JEDI tools - Benefits and challenges

Joint Effort for Data assimilation Integration (JEDI)-based workflow consists of EWOK (Experiments and Workflows Orchestration Kit) that can provide generic tools to describe and create application suites and create appropriate tasks. Examples of application suites are end-to-end cycled systems, forecast only (deterministic or ensemble forecasts), observation and post-processing (inline or offline), etc. Also, there are tools within JEDI for file handling that can be useful for UFS. There is an ongoing effort to evaluate the use of EWOK in whole or in part for the UFS workflow system, which has identified a number of potential issues that will need to be explored further. For example, several module environments needed for JEDI are in a private repository. NOAA is moving away from private repos, so that will be a non-starter for community workflows. There were also questions about whether the JEDI workflow will take into consideration the UFS workflow requirements as well as concerns raised about its use and development pathlines as to whether it would be useful for the UFS community.

Workflow System Architecture

The intent is to have a single workflow architecture that is capable of supporting all UFS applications. The workflow working group’s role is to identify and document the requirements of the workflow system, as well as articulating and communicating those requirements to management, developers, and stakeholders. The system architecture should have the flexibility to address multiple applications end-to-end (e.g. including pre-and post-processors).

GUI for CESM simpler models framework

Alper Altuntas (NCAR) presented “A GUI for CESM simpler model framework,” which is a GUI-based tool for users to create Community Earth System Model (CESM) cases, choose compsets and grids. The goal is to streamline modeling within CESM and to enable hierarchical testing so users can adjust model complexities. This required development of new relational metadata in Common Infrastructure for Modeling the Earth (CIME) and providing a graphical interface using Jupyter lab. The project adds

new metadata called compliance metadata. The presentation provided an overview of CIME, which allows the user to create a case, configure it, build the model, and submit jobs to the batch system. The creation of a case requires a compset that includes a complete set of components integrated together to run a specific case (e.g. linking atmosphere, ocean, initialization time, resolution etc). The user can use an already existing configured compset and grids or create a new one for their needs. To accomplish this, there is new meta data in CIME. The purpose is to express interdependencies and incompatibilities between different options (OCN_GRID, COMP_ATM, DATM_MODE, etc). A live demo was also presented to show the capability. The general consensus during the discussion was the sanity check is worth exploring with or without the availability of GUI.

Working with NCO - moving forward

The attendees agreed that the job of the National Centers for Environmental Prediction (NCEP) Central Operations (NCO) is to provide seamless forecasts to end users. So, any failures during runtime will jeopardize the mission. Hence, NCO should be able to debug any issue as quickly as possible. The bash scripts (e.g. set -x) provide line-by-line tracking of commands executed along with errors if any. Developers should be prepared to educate NCO to debug codes easily. This shift will require regular communications with NCO. A biweekly tag-up between EMC and NCO will ensure continued communication. The expectations and responsibilities of both parties should be laid out formally as part of an “Environmental Equivalence” phase 3 (EE3).

Summary

The meeting provided an opportunity to review and further refine the requirements, implementation, and project management of a workflow system. It was a consensus that the workflow system should be modular and be able to diversify, so a unified system should be built instead of a unitary system. The UFS management and the participants agreed that all of the repositories should be available to the community. One of the critical aspects of the workflow system is working closely with NCO. Communicating with them regularly and addressing their needs for running the model in operations will be critical to making it successful.