The Configuration manager of Research and Operational Workflows (CROW) Community Review Report

Developmental Testbed Center (DTC)

Points of Contact: Evan Kalina[1,2,3] and Ligia Bernardet[1,3]

June 2020

[1] Developmental Testbed Center
[2] Cooperative Institute for Research In Environmental Sciences/University of Colorado
[3] NOAA Global Systems Laboratory

EXECUTIVE SUMMARY

- The CROW review was held on April 28, 2020 with the goal of working towards a decision on whether the CROW should replace the existing configuration manager in the Unified Forecast System (UFS) Hurricane Application, commonly referred to as the Hurricane Analysis and Forecast System (HAFS).
- Through reading materials and exercises distributed before the review, presentations, demonstrations, and discussions, the attendees became more familiar with CROW and the following advantages and disadvantages of using CROW in HAFS were identified.
    - Advantages
        - Opportunity for unification across workflow tools used in the UFS.
        - Consolidation around a single language (YAML) as a configuration language for HAFS.
        - More flexibility for specifying configuration files.
        - Small learning curve for HAFS users and developers.
        - CROW documentation is already available.
    - Disadvantages
        - Some situations would require HAFS developers to learn CROW.
        - Community support for CROW still needs to be established.
        - Ease of CROW acceptance by NCEP Central Operations is unknown.
    - Other considerations
        - The integration of CROW and CIME needs to be explored. If HAFS uses its current configuration system or CROW, the CIME CCS would not be used to configure the end-to-end HAFS system.
- Next steps
    - There are funded projects that depend on a decision of whether CROW will be used in HAFS. To advance these projects, a suggested timeline for making this decision is by September 1, 2020, but preferably sooner. Delays in making this decision will impact the projects deliverables and timeline.

1. Introduction

The CROW review was organized by the Developmental Testbed Center (DTC) and held virtually on April 28, 2020. The goal of this review was to work towards a decision on whether the CROW should replace the existing configuration manager in the Unified Forecast System (UFS) Hurricane Application, commonly referred to as the Hurricane Analysis and Forecast System (HAFS).

The CROW review is a deliverable for the project titled *Improve Workflow Usability, Portability, and Testing Capabilities* being executed by the University of Colorado Cooperative Institute for Research in Environmental Sciences (CU/CIRES) at the NOAA Global Systems Laboratory (NOAA/GSL). This project is funded under the *Infrastructure* portfolio of the Fiscal Year 2018 Disaster Related Appropriation Supplemental (DRAS), commonly referred to as the Hurricane Supplemental (HSUP).

This review was motivated by the growth in the diversity of configuration managers and other workflow components being used in the various UFS applications. As an example, HAFS uses a configuration manager inherited from the Hurricane Weather Research and Forecast (HWRF) system, GFS development employes its own Python-based configuration system, developmental versions of the UFS MRW and Subseasonal-to-Seasonal Applications use CROW, and the MRW Application v1.0.0 is publicly distributed with the Community Infrastructure for Modeling the Earth (CIME) Case Control System (CCS). While EMC has been developing and using CROW experimentally for approximately three years, and while CROW was included in the public release of FV3GFS v1, CROW was poorly understood by the community and the programmatic path for CROW adoption in UFS was not clear. This review offered an opportunity for the community to learn more about CROW and make decisions about CROW's path forward. In order to limit the scope of the discussion, the review was focused on the adoption of CROW specifically for HAFS.

Fifty-seven registered attendees were present in the review, representing various sectors of the numerical weather prediction community, including NOAA's National Weather Service/Environmental Modeling Center (NWS/EMC), NOAA GSL, NOAA Atlantic Oceanographic and Meteorological Laboratory, National Center for Atmospheric Research (NCAR), universities, and the private sector. The Google Meet online platform was employed and the audio, video, screen sharing, and chat capability were successfully used to connect the participants.

The review was preceded by two activities. First, to better understand the workflow requirements for HAFS, the DTC collected input from the HAFS and HWRF communities through an online survey and through the issue tracker in the GitHub repository https://github.com/NCAR/ufs_workflows_sandbox. The summary of the four issues contributed and of the input offered by the 27 survey participants (available at https://dtcenter.org/sites/default/files/summary-hwrf-hafs-community-survey-april2020.pdf) was summarized in one of the review talks. Second, DTC distributed read-ahead materials on how to configure HAFS using its current configuration system and with CROW. Participants with access

to NOAA's Hera High-Performance Computing platform had the opportunity to run the two configuration systems before the review.

On April 28, the CROW Review started with four presentations: Arun Chawla (NOAA/NWS/EMC) presented on workflow requirements from the perspective of EMC, Ligia Bernardet (NOAA/GSL and DTC) presented on workflow requirements from the perspective of users and developers of HAFS and the HWRF model, Kate Friedman (NOAA/NWS/EMC) gave an overview of CROW, Bin Liu (IMSG at NOAA/NWS/EMC) provided an overview of the current HAFS Workflow and Configuration System, and Samuel Trahan (CU/CIRES at NOAA/GSL and DTC) described the implementation of CROW in HAFS. Next, Evan Kalina (CU/CIRES at NOAA/GSL and DTC) gave a live demonstration contrasting how to configure HAFS using its current configuration system and using CROW. This was followed by plenary discussion on the topic. Subsequently, Mariana Vertenstein (NCAR) presented on the CIME workflow and tools for hierarchical development. A panel discussion led by Avichal Mehra (NOAA/NWS/EMC), Mariana Vertenstein, Arun Chawla, and Evan Kalina followed. Links to the presentations can be found on the DTC website.

Section 2 in this report summarizes the feedback received about using CROW in HAFS, and Section 3 describes the next steps related to making a decision on whether or not to adopt the CROW for HAFS, the timeline of the decision, and the impact of this decision on various ongoing projects.

2. Summary of Advantages and Disadvantages of CROW

a) Advantages

i) Opportunities for unification

Arun Chawla stated in his presentation about workflow requirements that EMC is currently coalescing around three main workflows – the global, regional, and HAFS – and that common tools for these workflows are being unified. The global workflow in particular will be used for the UFS MRW and S2S applications. A developmental version of the global workflow (in a feature branch that is expected to be transitioned to the main develop branch in the near future) already uses CROW as its configuration manager. If HAFS were also to adopt CROW, it would further reduce the number of configuration managers in use. This reduction in configuration managers would potentially lessen the collective overhead needed to maintain different managers, and allow HAFS to take advantage of new features added to CROW for the global workflow. It would also mean that users and developers familiar with configuring the UFS MRW and S2S Transition-to-Operations workflows would automatically know how to configure the HAFS workflow, since all of these applications would be using CROW. The reduced learning curve for these users and developers would translate into obtaining results from HAFS faster and more easily.

ii) With CROW, Rocoto and ecFlow workflows would be entirely generated from the same set of YAML-based configuration files.

HAFS users and developers at EMC and elsewhere who work on the NOAA RDHPCS machines (i.e., Jet and Hera) are accustomed to using the Rocoto workflow manager to drive the HAFS workflow forward. The existing HAFS configuration manager ingests a template file, called hafs_workflow.xml.in, which is written in XML. The placeholder variables in the XML template are replaced with values from the UNIX-style .INI configuration files in the parm/ directory to produce a complete XML that Rocoto uses to advance the workflow. If HAFS developers want to make substantial modifications to the existing Rocoto workflow (e.g., add a new task to the workflow), they will need to be familiar with the syntax of both UNIX-style configuration files and XML files, since they will likely want to edit both the hafs_workflow.xml.in template file and the configuration files in the parm/ directory. With the CROW-based HAFS configuration system, HAFS developers will no longer need to edit XML files. The template file for Rocoto workflows (parm/hafs_workflow.yaml) and the individual configuration files (i.e., hafs_basic.yaml, hafs.yaml, hafs_input.yaml, and system.yaml) are all written in YAML. The use of YAML provides a common “look and feel” throughout the different components of the configuration system in the CROW-based HAFS workflow, which will lessen the number of tools that a new HAFS developer needs to be familiar with before they can successfully modify the workflow. In addition, the same set of YAML configuration files also would be used to configure the ecFlow-based workflow, which is run by NCEP Central Operations (NCO).

iii) The CROW-based YAML provides more flexibility for specifying configuration files than the UNIX-based .INI format

Configuration files written in YAML in the CROW-based HAFS workflow offer multiple advantages over the UNIX configuration files in the existing HAFS workflow. First, the implementation of YAML in CROW allows the user to perform calculations in the configuration files using the !calc function. For example, the following block of code in sites/xjet.yaml sets the mpi_ranks variable to 360 for a regional HAFS run with a processor layout of 12x12 (layout_x times layout_y) for the forecast plus three groups of 72 cores dedicated to output from the FV3 write component:

```
all.FORECAST_RESOURCES=="regional_12x12io3x72_omp2"
      take: !JobRequest
        - <<: *forecast_base
          mpi_ranks: !calc 30*12
```

The last line is equivalent to `mpi_ranks: 360`, but in this case, it is helpful for the user to be able to infer that the 360 cores are distributed across 30 nodes, using 12 cores per node. One could imagine other situations in which being able to perform mathematical calculations in the configuration files might be essential rather than simply helpful. For instance, a user might want

to set the write component output frequency to some factor of the cycling interval to supply files at an appropriate cadence to the next cycle's data assimilation step.

Another advantage of the YAML-based CROW configuration files is that case statements can be included in them to implement logic within the configuration files. For example, consider the following portion of sites/xjet.yaml:

```
  forecast: !FirstTrue
    - when: !calc
all.FORECAST_RESOURCES=="regional_12x12io3x36_omp2"
      take: !JobRequest
        - <<: *forecast_base
          mpi_ranks: !calc 21*12
    - when: !calc
all.FORECAST_RESOURCES=="regional_12x12io3x48_omp2"
      take: !JobRequest
        - <<: *forecast_base
          mpi_ranks: !calc 24*12
    - when: !calc
all.FORECAST_RESOURCES=="regional_12x12io3x72_omp2"
      take: !JobRequest
        - <<: *forecast_base
          mpi_ranks: !calc 30*12
…
    - otherwise: !error "Don't know FORECAST_RESOURCES for
{all.FORECAST_RESOURCES} on xjet"
```

This case statement instructs the configuration system to match the value of FORECAST_RESOURCES to a specific character string, and then set the value of mpi_ranks based on this value. If a match isn't found, an error will be supplied to the user that informs them that the configuration system does not know how to configure the forecast job for the setting of FORECAST_RESOURCES that they supplied. Unlike the CROW-based system, the existing HAFS configuration system does not allow for the direct use of conditionals in the configuration files or for sanity checking to be embedded within them.

Finally, YAML offers more flexibility in the structuring of configuration files than the .INI format. .INI files are limited to a single section that contains key-value pairs within it. For example,

```
[config]
run_vortexinit=no
run_gsi=no
run_ocean=no
```

However, configuration files written in YAML can contain sections with multiple nested subsections that each contain their own unique set of key-value pairs. For example,

```
resources:
  forecast_base: &forecast_base
    walltime: !timedelta '06:00:00'
    OMP_NUM_THREADS: 2
    exclusive: true
    max_ppn: 12
  forecast: !FirstTrue
    - when: !calc
all.FORECAST_RESOURCES=="FORECAST_RESOURCES_regional_12x12io3x36_omp2
"
      take: !JobRequest
        - <<: *forecast_base
          mpi_ranks: 252
          OMP_NUM_THREADS: 2
          max_ppn: 12
    - when: !calc
all.FORECAST_RESOURCES=="FORECAST_RESOURCES_regional_12x12io3x48_omp2
"
      take: !JobRequest
        - <<: *forecast_base
          mpi_ranks: 288
          OMP_NUM_THREADS: 2
          max_ppn: 12
```

Here, the top-level section is called “resources,” and it contains two subsections (“forecast_base” and “forecast”). The forecast_base subsection is included in multiple places within the case statement in the forecast subsection via `- <<: *forecast_base` without needing to repeat the contents of that subsection each time. The use of subsections enhances the organization of the configuration files and is not possible in .INI files. The use of `- <<: *forecast_base` performs the same function as `@inc=forecast_base` would in the .INI files.

The inclusion of additional features within the CROW-based configuration files makes them more complex. It is important to realize that none of the additional features available in the CROW-based YAML configuration files, including !calc, sanity checking, and nested subsections, need to be employed in HAFS. They are simply features that can be utilized for their benefits if the HAFS community chooses to embrace them. A set of coding standards can be dictated that excludes the use of features that are deemed undesirable.

iv) The inclusion of CROW in HAFS does not require users or developers to relearn the system

From a user perspective, the process of configuring HAFS with and without CROW is essentially the same. The user starts by defining some user- and system-specific characteristics, like their CPU project code and their system's directory structure in either system.yaml (CROW) or system.conf (non-CROW). They also make any necessary changes to the other configuration files (i.e., hafs_basic, hafs, and hafs_input, either *.yaml or *.conf) to specify their desired experiment configuration. Finally, they prepare a cron script in the rocoto/ directory that defines the case they want to run. The format and content of the cron script are the same regardless of whether CROW is used. Therefore, from a user perspective, the only difference between the existing HAFS configuration system and the CROW-based system is that the latter requires the user to have a working knowledge of YAML. In practice, only a small amount of YAML knowledge is needed, since there are many examples in the configuration files and most users would only change the values of a few specific options. Therefore, from a user perspective, we can say that HAFS is largely unchanged by the inclusion of CROW.

In most cases (see section 2bi for an exception), developers will be similarly unaffected by the use of CROW in HAFS. Since CROW is a configuration system and the first task in the HAFS workflow (i.e., the launcher) accomplishes the configuration, only the content of the ush/hafs/launcher.py script has changed substantially, but the coding style remains the same. To modify the workflow template to, for example, add a new task to the workflow or add a new dependency to a task, developers would edit parm/hafs_workflow.yaml (CROW) instead of rocoto/hafs_workflow.xml.in (non-CROW), which amounts to writing instructions in YAML rather than XML. As one HAFS developer remarked at the CROW review, "CROW can be treated as a black box" in HAFS, as users and most developers are not expected to have a reason to interact with it directly.

v) CROW documentation is already available

EMC/EIB staff have already prepared documentation that explains the design philosophy behind CROW, how to use CROW, and a brief description of how to port CROW to a new platform. The documentation is available through GitHub. While HAFS users should not need to refer to the documentation to understand how to configure HAFS (since the HAFS configuration process is largely unchanged), it is still helpful to have documentation available in the event that the HAFS community wants to learn more about the software. Additional documentation on CROW will be added by EMC/EIB staff in the future.

b) Disadvantages

i) Some situations would require HAFS developers to have knowledge of CROW

A few review attendees have commented that if CROW were included in HAFS, it would be yet another software system for HAFS developers to learn, thereby bringing additional

complexity to developing HAFS. Although most HAFS users and developers would not require much new knowledge to configure HAFS with CROW, there are scenarios in which at least some CROW knowledge would be necessary, particularly if troubleshooting were required. One scenario in which CROW knowledge would be necessary is if new XML tags (e.g., custom dependency tags) were added to Rocoto, and a HAFS developer wanted to add support for them in CROW so that they could be used in HAFS. Instead of modifying the Rocoto XML template as they would have done in the past, the developer would need to understand the CROW system and make modifications to the underlying CROW code to implement support for the new tag. Another scenario in which some CROW knowledge would likely be required is if certain changes were made to the NOAA RDHPCS machines, such as adopting a new job scheduler or if a new RDHPCS machine were brought online. Porting CROW to a new computer platform should mainly involve preparing an appropriate platform.yaml file that describes the resources needed for each task in the workflow. In regards to software requirements on a new platform, Python 3.6+ and its standard libraries would be sufficient to run the CROW-based HAFS launcher and the rest of the HAFS scripts.

ii) Community support for CROW still needs to be established

The above situations do not necessarily mean that a HAFS developer would be left on their own to perform the changes themselves. Rather, EMC/Engineering and Implementation Branch (EIB) staff are interested in standing up community support for CROW, and this support would ostensibly be available to the community to resolve issues like the ones mentioned above. However, this community support is not yet available, and EMC/EIB staff have stated that to have a successful support system, more resources will be needed than the current single staff member who is tasked with developing and supporting CROW.

iii) Ease of CROW acceptance by NCO is unknown

Any configuration system used in HAFS will need to be transitioned to operations, since there are storm-specific and cycle-specific entries in multiple configuration files (e.g., storm1.yaml, storm1.holdvars.txt) that are not known prior to the formation of the storm or to the start of a cycle. NCO has yet to perform a review of the CROW software, and therefore, it is unknown whether NCO would accept it “as is” or require modifications to it. If modifications were required, EMC/EIB staff would likely be available to assist with this work.

c) Other considerations

i) Use of CROW precludes the use of the full CIME CCS to configure the overarching HAFS workflow

CROW and the CIME CCS are both fundamentally capable of configuring the full HAFS workflow, although further development of the CIME CCS would be necessary to support the realtime, cycling, and pre- and post-processing steps in HAFS. Therefore, there is an

opportunity cost associated with selecting CROW, since using CROW would preclude using the CIME CCS to configure the full HAFS workflow. However, the CIME CCS could be used to configure and build the HAFS coupled forecast executable if CROW is selected to configure the overall HAFS system, which the DTC Infrastructure DRAS/HSUP project team is exploring with NCAR/Climate and Global Dynamics Laboratory (CGD) staff. In addition, other CIME features that exist outside of the CCS (e.g., data models, tools for validating HAFS ports to new machines) could still be used.

3. Next Steps

It is important to realize that as of the writing of this report, no decision has been made on whether to use CROW in HAFS. The report has been made available to members of the EMC Hurricane Team, the UFS Hurricane application team, and EMC/EIB. We hope that the information contained within the report will help the EMC Hurricane Team determine whether the scripting and configuration changes made by DTC staff to connect CROW to HAFS (fully contained within the feature/crow branch of the HAFS community repository) will be accepted into the master. A suggested timeline for making this decision is by September 1, 2020, but preferably sooner.

There are negative consequences to waiting to make a decision on whether to use CROW. If the decision process is prolonged, there is more opportunity for the master to diverge from the feature/crow branch, which will increase the burden on DTC staff to keep the branch in sync with the latest changes in master. Resources to continue this synchronization at the DTC are already stretched thin. In addition, the next milestone in the DRAS/HSUP project on improving the HAFS workflow usability, portability, and testing capabilities is:

“Demonstrate that CROW or a CROW alternative can interact with CIME for building and running simple forecast model configurations.”

This milestone is a joint milestone shared by DTC staff at CIRES/NOAA/GSL and NCAR/CGD staff working on a related DRAS/HSUP project to include CIME in the HAFS workflow. The CIRES/NOAA/GSL milestone is currently due by June 30, 2020 (though a delay until September 30, 2020 is anticipated so that the CIRES/NOAA/GSL team can synchronize with NCAR/CGD’s schedule). Because of the short timeline, staff are currently planning to demonstrate that CIME and the existing (i.e., non-CROW) HAFS configuration system can interact, rather than CROW, since we do not know whether CROW will be included in HAFS. This amounts to demonstrating that CIME and a CROW alternative can interact. However, if it were known prior to the end of June that CROW would be used in HAFS, staff could focus on exploring a CIME-CROW connection earlier in the process. This early knowledge would be extremely helpful, since resources are limited and the entire project, which involves demonstrating a prototype version of the full HAFS workflow, must be finished by June 30, 2021.

4. Acknowledgments

We thank Samuel Trahan for performing the technical work to convert the HAFS Python scripts from Python2 to Python3 and to implement CROW in HAFS. The conversion from Python2 to Python3 was a prerequisite for being able to use CROW in HAFS, and has already been included in the master of the HAFS community repository. We also thank Mariana Vertenstein and Rocky Dunlap (NCAR) for many helpful discussions about how CROW and CIME could potentially interact within the HAFS workflow, as well as conversations with Bin Liu, Henry Winterbottom (IMSG at EMC), and James Frimel (CIRA at NOAA/GSL) about their perceptions of using CROW in HAFS. We are grateful for the opportunity to conduct the CROW Review in conjunction with the UFS Workflows Workshop, made possible by the UFS Workflows Workshop organizing committee, which included Benjamin Cash (GMU), Jennifer Bolton (NCAR), Arun Chawla, Michael Ek (NCAR, DTC), Louisa Nance (NCAR, DTC), and Mariana Vertenstein. We also appreciate the advice and feedback provided by the UFS Workflows Workshop committee on the format and content of the CROW Review. We are indebted to the CROW Review speakers, including Arun Chawla, Kate Friedman, Bin Liu, Samuel Trahan, Mariana Vertenstein, and Avichal Mehra, who delivered compelling, informative presentations and panel discussions despite the COVID-19 pandemic and the challenges associated with virtual meetings. Finally, we greatly appreciate the 57 members of the community who spent their day contributing to the CROW Review, as well as their willingness to prepare for the Review in advance by completing the read-ahead exercises and taking the survey to define HAFS workflow requirements.