UFS Hierarchical Testing Framework Users Guide

Release v1.0

Oct 05, 2022

CONTENTS

CHAPTER

ONE

GOALS FOR PROTOTYPE HIERARCHICAL TESTING FRAMEWORK

The main goal is to develop a transparent and approachable framework for UFS community testing of the UFS WM and its components consistent with the *HSD* (Hierarchical Systems Development) approach and NOAA baseline operational metrics, that can be easily used by the community.

To achieve the goal above, we adopt Ctest approach. CTest is the part of CMake that handles testing code. Ctest allows for an easy way to run user-defined test cases with various argument and option settings, and then check the results against expected output/baseline. The following are the benefits of using CTest for our prototype HTF:

- Ctest is the CMake test driver. Since CMake-based build system is used for all UFS components and applications, it is suitable to use CTest without any further changes o conduct tests and report results;
- Developers/users working on new codes can easily/quickly follow a few steps to design/add new tests in Ctests with HSD concepts (example1, example2);
- It is easy to maintain repeatability and reproducibility when transfer test cases to other users and whole community using Ctest;
- Ctest approach has potential to integrate with the existing EPIC Jenkins CI/CD pipeline (example)

CHAPTER

TWO

BUILDING AND RUNNING THE UFS-HTF

The Unified Forecast System (*UFS*) Hierarchical Testing Framework (HTF) provides a platform for users to design/add/run their own testing cases. Once the HTF is built, users can follow the instruction/examples to configure and add test cases based on the concept of HIERARCHICAL SYSTEM DEVELOPMENT (*HSD*).

This chapter walks users through how to build and run one of the test cases from Case Studies for the UFS Weather Model.

Attention: Currently The prototype of UFS-HTF is based on the global-workflow, and it has been only tested on Orion. The steps described in this chapter will work only on Orion. This chapter can also serve as a starting point for running the UFS-HTF on other systems (including generic Linux/Mac systems), but the user will need to perform additional troubleshooting (e.g. installation of Spack-Stack, WIP).

2.1 Install the Spack-Stack

Attention: Skip the Spack-Stack installation if working on a Level 1 system (e.g. Orion).

2.1.1 Background

The UFS-HTF draws on over 50 code libraries to run its applications. These libraries range from libraries developed in-house at NOAA (e.g., NCEPLIBS, FMS) to libraries developed by NOAA's partners (e.g., PIO, ESMF) to truly third party libraries (e.g., NETCDF). Individual installation of these libraries is not practical, so the Spack-Stack was developed as a central installation system to ensure that the infrastructure environment across multiple platforms is as similar as possible. Installation of the Spack-Stack is required to run the UFS-HTF.

2.1.2 Instructions

For a detailed description of installation options, see Installing the Spack-Stack.

After completing installation, continue to the next section (Section 2.2: Download the UFS-HTF prototype).

2.2 Download the UFS-HTF prototype

The UFS-HTF prototype source code is publicly available on GitHub. To download the HTF code, clone the `doc` branch of the repository:

```
git clone --recurse-submodules -b doc https://github.com/clouden90/ufs-htf.git
```

2.3 Build the UFS-HTF Using CMake

In the `ufs-htf` directory, create a subdirectory to hold the build's executables:

```
cd ufs-htf
mkdir build
cd build
```

From the build directory, run the following commands to build the testing platform:

```
cmake ..
make -j 2
```

2.4 Example Test Cases

Insipred by Case Studies for the UFS Weather Model, we implemented one of the test cases (2019 Hurricane Barry) for ufs coupled model setup (S2S, S2SW, and S2SWA WIP). Currently the UFS-HTF supports nine tests as shown in Table 2.1. Users who plan to design/add a new test should refer to *Adding test* for details on how to do so. At a minimum, these users will need to add the new test case to the `ufs-htf/test/CMakeLists.txt` script and add the corresponding files in the `ufs-htf/test` folder.

Table 2.1: Test Cases

Test Number	Test Name	Test Description
Test #1	build_ufs	build ufs model and its utilities
Test #2	get_ufs_fix_data	stage model input data from AWS S3 bucket
Test #3	ATM_c96_Barry	C96 grid atm only Hurricane Barry run
Test #4	S2S_c96_Barry	C96 grid atm-ocn-ice Hurricane Barry run
Test #5	S2SW_c96_Barry	C96 grid atm-ocn-ice-wav Hurricane Barry run
Test #6	S2SWA_c96_Barry	C96 grid atm-ocn-ice-wav-aerosal Hurricane Barry run
Test #7	Barry_track_err	Hurricane Barry track error check
Test #8	model_vrfy	Comparsion between fcst tmp2m/tmpsfc and reanalysis
Test #9	fcst_only_S2S_c96_Barry	Same as Test #4 without using Rocoto Workflow Manager

2.5 Running Test Cases

Following Section 2.3, you can first check the list of test cases with:

```
cd <build-directory>/test
ctest -N
```

This will list all available tests in the test suite:

```
Test #1: build_ufs
Test #2: get_ufs_fix_data
Test #3: ATM_c96_Barry
Test #4: S2S_c96_Barry
Test #5: S2SW_c96_Barry
Test #6: S2SWA_c96_Barry
Test #7: Barry_track_err
Test #8: model_vrfy
Test #9: fcst_only_S2S_c96_Barry
```

Then you can run tests with:

```
ctest
```

This will run all tests in the test suite. This can take a while so be patient. When the tests are complete, ctest will print out a summary. For example:

```
Test project /work2/noaa/epic-ps/ycteng/case/20220828/ufs-htf/build/test
    Start 1: build_ufs
1/9 Test #1: build_ufs .......................   Passed  907.80 sec
    Start 2: get_ufs_fix_data
```

(continues on next page)

(continued from previous page)

```
2/9 Test #2: get_ufs_fix_data .................   Passed    0.08 sec
    Start 3: ATM_c96_Barry
3/9 Test #3: ATM_c96_Barry ....................   Passed  1226.49 sec
    Start 4: S2S_c96_Barry
4/9 Test #4: S2S_c96_Barry ....................   Passed  1273.41 sec
    Start 5: S2SW_c96_Barry
5/9 Test #5: S2SW_c96_Barry ...................   Passed  1329.12 sec
    Start 6: S2SWA_c96_Barry
6/9 Test #6: S2SWA_c96_Barry ..................   Passed  1771.19 sec
    Start 7: Barry_track_err
7/9 Test #7: Barry_track_err ..................   Passed   28.91 sec
    Start 8: model_vrfy
8/9 Test #8: model_vrfy .......................   Passed   66.50 sec
    Start 9: fcst_only_S2S_c96_Barry
9/9 Test #9: fcst_only_S2S_c96_Barry ..........   Passed  821.40 sec

100% tests passed, 0 tests failed out of 9

Total Test time (real) = 7424.93 sec
```

If you want to run a single test or a subset of tests, you can do this with the `-R` option, for example:

```
ctest -R S2S_c96_Barry # run a single test
ctest -R S2S* # run a subset of tests
```

The output from these tests (stdout) will be printed to the screen but, to allow for greater scrutiny, it will also be written to the file **LastTest.log** in the directory `<build-directory>/Testing/Temporary`. In that same directory you will also find a file called **LastTestsFailed.log** that lists the last tests that failed. This may be from the last time you ran ctest or, if all those tests passed, it may be from a previous invocation.

If you're not happy with the information in LastTest.log and you want to know more, you can ask ctest to be **verbose**

```
ctest -V -R S2S_c96_Barry
```

. . . or even **extra-verbose** (hypercaffeinated mode):

```
ctest -VV -R S2S_c96_Barry
```

The `-V` and even `-VV` display the output messages on the screen in addition to writing them to the LastTest.log file. However, sometimes the amount of information written to LastTest.log isn't much different than if you were to run ctest without these options, particularly if all the tests pass.

You can also display the output messages only for the failed tests by using `--output-on-failure`

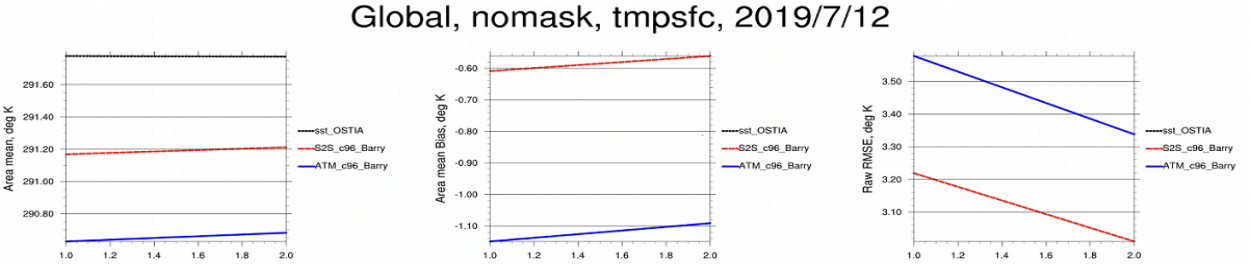
```
ctest --output-on-failure
```

Then You can check error messages for the failed tests, figure out the issue and fix it.

ctest also has an option to only re-run the tests that failed last time:

```
ctest --rerun-failed
```

You can also check *<build-directory>/test/*.png* to examine the modeled results from test cases. Here is one of the example plots (surface temperature comparsion: atm-only vs s2s vs OSTIA) generated from test #7:



Surface temperature timeseries comparison

CHAPTER

THREE

MORE ON TESTING

The UFS-HTF comes with a suite of ctests. These tests are designed for proof of concept of *HSD*. Developers working on a new test case could follow the below high-level instructions, and should ensure that all existing tests pass before adding a new test.

3.1 Adding a test to HTF

When new test cases are added to UFS-HTF, this test (and corresponding files) should be added into the standard ctest set. This ensures this test could be potentially used in the CI/CD pipeline in the future (WIP).

3.1.1 Example - Add C384 ATM-only case

All the ctesting in UFS-HTF is controlled through `ufs-htf/test/CMakeLists.txt`. A ctest could be either a unit test, or an integration test that executes an application (e.g. UFS-SRW, UFS-WM, or UFS-MRW). Benchmark results are provided accompanying the ctests. A unit ctestcontains results in a reference log file based on analytical solutions or accurate numerical studies. For application ctest, one could compare to associated reference based on a previous execution of the same test. To determine the pass or failure for a ctest, the actual output could be compared against the reference or obs data (WIP). The reader is referred to `ufs-htf/test/CMakeLists.txt`, where numerous examples exist for both. For example, one can modify `ufs-htf/test/CMakeLists.txt` to add c384 atm-only test case:

```
add_test ( NAME ATM_c384_Barry
           COMMAND bash run_ctest.sh --app=ATM --grid=384 --case=Barry --ctest -v
           WORKING DIRECTORY ${CMAKE_BINARY_DIR}/test )
set_tests_properties(ATM_c384_Barry PROPERTIES TIMEOUT 10800)
```

This will add a C384 atm-only test case for Hurricane Barry. Then you will have to rebuild under your <*build-directory*> folder. Then you will find a new test has been added in the test set:

```
cd  <build-directory>/test
ctest -N
Test  #1: build_ufs
```

(continues on next page)

(continued from previous page)

```
Test  #2: get_ufs_fix_data
Test  #3: ATM_c96_Barry
Test  #4: S2S_c96_Barry
Test  #5: S2SW_c96_Barry
Test  #6: S2SWA_c96_Barry
Test  #7: Barry_track_err
Test  #8: model_vrfy
Test  #9: fcst_only_S2S_c96_Barry
Test #10: ATM_c384_Barry
```

Keep in mind the developer will have to prepare the associated input files such as model ICs. Once can check *ufs-htf/test/prep.sh* for more details. To get model gird/fix input files, you can simply use the following command:

```
cd
./prep.sh -a 384 -o 025
```

Then the script will get model input files from AWS S3. Initial condition (IC) files for FV3 (created from GFS operational dataset) can be downloaded from below link.

```
Download initial condition files: 2019071200.c384.tar.gz
```

Since the C384 case requires more computational resources, users may have to modify a few env parameters, which is located in *<build-directory>/test/case/Barry.env*. If you are on Orion, please modify this file from:

```
export _QUEUE="debug"
export _PARTITION_BATCH="debug"
export _wtime_fcst_gfs="00:30:00"
```

to

```
export _QUEUE="batch"
export _PARTITION_BATCH="orion"
export _wtime_fcst_gfs="04:00:00"
```

Then you can run the new test:

```
ctest -VV -R ATM_c384_Barry
```

CHAPTER

FOUR

GLOSSARY

advect
To transport substances in the atmostphere by *advection*.

advection
According to the American Meteorological Society (AMS) definition, advection is "The process of transport of an atmospheric property solely by the mass motion (velocity field) of the atmosphere." In common parlance, advection is movement of atmospheric substances that are carried around by the wind.

CAPE
Convective Available Potential Energy.

CCPA
Climatology-Calibrated Precipitation Analysis (CCPA) data. This data is required for METplus precipitation verification tasks within the SRW App. The most recent 8 days worth of data are publicly available and can be accessed here.

CCPP
The Common Community Physics Package is a forecast-model agnostic, vetted collection of code containing atmospheric physical parameterizations and suites of parameterizations for use in Numerical Weather Prediction (NWP) along with a framework that connects the physics to the host forecast model.

chgres_cube
The preprocessing software used to create initial and boundary condition files to "coldstart" the forecast model.

CIN
Convective Inhibition.

cron
crontab
cron table
Cron is a job scheduler accessed through the command-line on UNIX-like operating systems. It is useful for automating tasks such as the `rocotorun` command, which launches each workflow task in the SRW App. Cron periodically checks a cron table (aka crontab) to see if any tasks are are ready to execute. If so, it runs them.

CRTM
Community Radiative Transfer Model. CRTM is a fast and accurate radiative transfer model

developed at the Joint Center for Satellite Data Assimilation (JCSDA) in the United States. It is a sensor-based radiative transfer model and supports more than 100 sensors, including sensors on most meteorological satellites and some from other remote sensing satellites.

Component
A software element that has a clear function and interface. In Earth system models, components are often single portions of the Earth system (e.g. atmosphere, ocean, or land surface) that are assembled to form a whole.

Component Repository
A *repository* that contains, at a minimum, source code for a single component.

Container
Docker describes a container as "a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another."

CONUS
Continental United States

CAM
convection-allowing models
Convection-allowing models (CAMs) are models that run on high-resolution grids (usually with grid spacing at 4km or less). They are able to resolve the effects of small-scale convective processes. They typically run several times a day to provide frequent forecasts (e.g., hourly or subhourly).

cycle
cycles
An hour of the day on which a forecast is started.

cycle-dependent
Describes a workflow task that needs to be run at the start of each *cycle* of an experiment.

cycle-independent
Describes a workflow task that only needs to be run once per experiment, regardless of the number of cycles in the experiment.

dycore
dynamical core
Global atmospheric model based on fluid dynamics principles, including Euler's equations of motion.

echo top
The radar-indicated top of an area of precipitation. Specifically, it contains the height of the 18 dBZ reflectivity value.

EMC
The Environmental Modeling Center.

EPIC
The Earth Prediction Innovation Center seeks to accelerate scientific research and modeling contributions through continuous and sustained community engagement in order to produce the most accurate and reliable operational modeling system in the world.

ESG
: Extended Schmidt Gnomonic (ESG) grid. The ESG grid uses the map projection developed by Jim Purser of NOAA *EMC* (**:cite:t:`Purser_2020`**).

ESMF
: Earth System Modeling Framework. The ESMF defines itself as "a suite of software tools for developing high-performance, multi-component Earth science modeling applications."

FV3
: The Finite-Volume Cubed-Sphere *dynamical core* (dycore). Developed at NOAA's Geophysical Fluid Dynamics Laboratory (GFDL), it is a scalable and flexible dycore capable of both hydrostatic and non-hydrostatic atmospheric simulations. It is the dycore used in the UFS Weather Model.

FVCOM
: Finite Volume Community Ocean Model. FVCOM is used in modeling work for the Great Lakes Coastal Forecasting System (next-gen FVCOM) conducted by the Great Lakes Environmental Research Laboratory.

GFS
: Global Forecast System. The GFS is a National Centers for Environmental Prediction (NCEP) weather forecast model that generates data for dozens of atmospheric and land-soil variables, including temperatures, winds, precipitation, soil moisture, and atmospheric ozone concentration. The system couples four separate models (atmosphere, ocean, land/soil, and sea ice) that work together to accurately depict weather conditions.

GRIB2
: The second version of the World Meterological Organization's (WMO) standard for distributing gridded data.

halo
: A strip of cells on the edge of the regional grid. The wide halo surrounds the regional grid and is used to feed the lateral boundary conditions into the grid. The HALO_BLEND parameter refers to a strip of cells *inside* the boundary of the native grid. This halo smooths out mismatches between the external and internal solutions.

HPC
HPCs
: High-Performance Computing.

HPC-Stack
: The HPC-Stack is a repository that provides a unified, shell script-based build system for building the software stack required for numerical weather prediction (NWP) tools such as the Unified Forecast System (UFS) and the Joint Effort for Data assimilation Integration (JEDI) framework.

HPSS
: High Performance Storage System (HPSS).

HRRR
: High Resolution Rapid Refresh. The HRRR is a NOAA real-time 3-km resolution, hourly updated, cloud-resolving, convection-allowing atmospheric model initialized by 3km grids with 3km radar assimilation. Radar data is assimilated in the HRRR every 15 min over a 1-h

period adding further detail to that provided by the hourly data assimilation from the 13km radar-enhanced Rapid Refresh.

HSD
: Hierarchical system development refers to the ability to engage in development and testing at multiple levels of complex prediction software such as the Unified Forecast System (UFS).

IC/LBC
IC/LBCs
: Initial conditions/lateral boundary conditions

IC
ICs
: Initial conditions

LAM
: Limited Area Model (grid type), formerly known as the "Stand-Alone Regional Model," or SAR. LAM grids use a regional (rather than global) configuration of the *FV3 dynamical core*.

LBC
LBCs
: Lateral boundary conditions

MERRA2
: The Modern-Era Retrospective analysis for Research and Applications, Version 2 provides satellite observation data back to 1980. According to NASA, "It was introduced to replace the original MERRA dataset because of the advances made in the assimilation system that enable assimilation of modern hyperspectral radiance and microwave observations, along with GPS-Radio Occultation datasets. It also uses NASA's ozone profile observations that began in late 2004. Additional advances in both the GEOS model and the GSI assimilation system are included in MERRA-2. Spatial resolution remains about the same (about 50 km in the latitudinal direction) as in MERRA."

MPI
: MPI stands for Message Passing Interface. An MPI is a standardized communication system used in parallel programming. It establishes portable and efficient syntax for the exchange of messages and data between multiple processors that are used by a single computer program. An MPI is required for high-performance computing (HPC).

MRMS
: Multi-Radar/Multi-Sensor (MRMS) System Analysis data. This data is required for METplus composite reflectivity or *echo top* verification tasks within the SRW App. A two-day archive of precipitation, radar, and aviation and severe weather fields is publicly available and can be accessed here.

NAM
: North American Mesoscale Forecast System. NAM generates multiple grids (or domains) of weather forecasts over the North American continent at various horizontal resolutions. Each grid contains data for dozens of weather parameters, including temperature, precipitation, lightning, and turbulent kinetic energy. NAM uses additional numerical weather models to generate high-resolution forecasts over fixed regions, and occasionally to follow significant weather events like hurricanes.

namelist
A namelist defines a group of variables or arrays. Namelists are an I/O feature for format-free input and output of variables by key-value assignments in FORTRAN compilers. Fortran variables can be read from and written to plain-text files in a standardised format, usually with a `.nml` file ending.

NCAR
The National Center for Atmospheric Research.

NCEP
National Centers for Environmental Prediction (NCEP) is an arm of the National Weather Service consisting of nine centers. More information can be found at https://www.ncep.noaa.gov.

NCEPLIBS
The software libraries created and maintained by *NCEP* that are required for running *chgres_cube*, the UFS Weather Model, and *UPP*. They are included in the HPC-Stack.

NCEPLIBS-external
A collection of third-party libraries required to build *NCEPLIBS*, *chgres_cube*, the UFS Weather Model, and *UPP*. They are included in the *HPC-Stack*.

NCL
An interpreted programming language designed specifically for scientific data analysis and visualization. Stands for NCAR Command Language. More information can be found at https://www.ncl.ucar.edu.

NDAS
NAM Data Assimilation System (NDAS) data. This data is required for METplus surface and upper-air verification tasks within the SRW App. The most recent 1-2 days worth of data are publicly available in PrepBufr format and can be accessed here. The most recent 8 days of data can be accessed here.

NEMS
The NOAA Environmental Modeling System is a common modeling framework whose purpose is to streamline components of operational modeling suites at *NCEP*.

NEMSIO
A binary format for atmospheric model output from *NCEP*'s Global Forecast System (*GFS*).

netCDF
NetCDF (Network Common Data Form) is a file format and community standard for storing multidimensional scientific data. It includes a set of software libraries and machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data.

NUOPC
The National Unified Operational Prediction Capability Layer "defines conventions and a set of generic components for building coupled models using the Earth System Modeling Framework (*ESMF*)."

NWP
Numerical Weather Prediction (NWP) takes current observations of weather and processes

them with computer models to forecast the future state of the weather.

Orography
The branch of physical geography dealing with mountains.

Parameterization
Parameterizations
Simplified functions that approximate the effects of small-scale processes (e.g., microphysics, gravity wave drag) that cannot be explicitly resolved by a model grid's representation of the earth.

RAP
Rapid Refresh. The continental-scale NOAA hourly-updated assimilation/modeling system operational at NCEP. RAP covers North America and is comprised primarily of a numerical forecast model and an analysis/assimilation system to initialize that model. RAP is complemented by the higher-resolution 3km High-Resolution Rapid Refresh (*HRRR*) model.

Repository
A central location in which files (e.g., data, code, documentation) are stored and managed.

SDF
Suite Definition File. An external file containing information about the construction of a physics suite. It describes the schemes that are called, in which order they are called, whether they are subcycled, and whether they are assembled into groups to be called together.

spack-stack
The spack-stack is a collaborative effort between the NOAA Environmental Modeling Center (EMC), the UCAR Joint Center for Satellite Data Assimilation (JCSDA), and the Earth Prediction Innovation Center (EPIC). *spack-stack* is a repository that provides a Spack-based method for building the software stack required for numerical weather prediction (NWP) tools such as the Unified Forecast System (UFS) and the Joint Effort for Data assimilation Integration (JEDI) framework. spack-stack uses the Spack package manager along with custom Spack configuration files and Python scripts to simplify installation of the libraries required to run various applications. The *spack-stack* can be installed on a range of platforms and comes pre-configured for many systems. Users can install the necessary packages for a particular application and later add the missing packages for another application without having to rebuild the entire stack.

tracer
tracers
According to the American Meteorological Society (AMS) definition, a tracer is "Any substance in the atmosphere that can be used to track the history [i.e., movement] of an air mass." Tracers are carried around by the motion of the atmosphere (i.e., by *advection*). These substances are usually gases (e.g., water vapor, CO2), but they can also be non-gaseous (e.g., rain drops in microphysics parameterizations). In weather models, temperature (or potential temperature), absolute humidity, and radioactivity are also usually treated as tracers. According to AMS, "The main requirement for a tracer is that its lifetime be substantially longer than the transport process under study."

UFS
The Unified Forecast System is a community-based, coupled, comprehensive Earth modeling system consisting of several applications (apps). These apps span regional to global domains

and sub-hourly to seasonal time scales. The UFS is designed to support the *Weather Enterprise* and to be the source system for NOAA's operational numerical weather prediction applications. For more information, visit https://ufscommunity.org/.

UFS_UTILS
A collection of code used by multiple *UFS* apps (e.g., the UFS Short-Range Weather App, the UFS Medium-Range Weather App). The grid, orography, surface climatology, and initial and boundary condition generation codes used by the UFS Short-Range Weather App are all part of this collection.

Umbrella repository
A repository that houses external code, or "externals," from additional repositories.

UPP
The Unified Post Processor is software developed at *NCEP* and used operationally to post-process raw output from a variety of *NCEP*'s *NWP* models, including the *FV3*.

Weather Enterprise
Individuals and organizations from public, private, and academic sectors that contribute to the research, development, and production of weather forecast products; primary consumers of these weather forecast products.

Weather Model
A prognostic model that can be used for short- and medium-range research and operational forecasts. It can be an atmosphere-only model or an atmospheric model coupled with one or more additional components, such as a wave or ocean model. The SRW App uses the fully-coupled UFS Weather Model.

Workflow
The sequence of steps required to run an experiment from start to finish.

INDEX

O

P

R

S

T

U

W