Debugging: Alternative Methods of Running

HWRF Python Scripts Training

College Park, MD

January 22, 2016

Overview

- Usefulness of alternative methods
- Interactive batch jobs and wrappers
- Run ex-scripts from the shell
- Manually run HWRF Python functions

Ways to Run

- Automation system
  - ecFlow
  - Rocoto
- Wrappers
- Interactive batch jobs
- Manually submitting scripts and functions

Usefulness

- When running more than a few cycles of HWRF, it is recommended that some automation capability is used.
- When implementing new capabilities and debugging, the other forms of job submission may be more effective/ efficient in the testing process
- Wrappers can be used to submit the jobs that are supported in the HWRF v3.7a public release. Others would need to be developed as needed.
  - Quickly run one component at a time. (cannot start from the middle)
- Manual execution is ideal for quick turnaround on debugging

Running HWRF with Wrappers

Wrappers

- Each wrapper submits a single component of the system

bufrprep_wrapper	launcher_wrapper
forecast_wrapper	merge_wrapper
gsi_d02_wrapper	post_wrapper
gsi_d03_wrapper	products_wrapper
init_gdas_wrapper	relocate_wrapper
init_gfs_wrapper	unpost_wrapper
init_ocean_wrapper	

- Note that only components supported in the HWRF v3.7a public release have wrappers readily available in the trunk

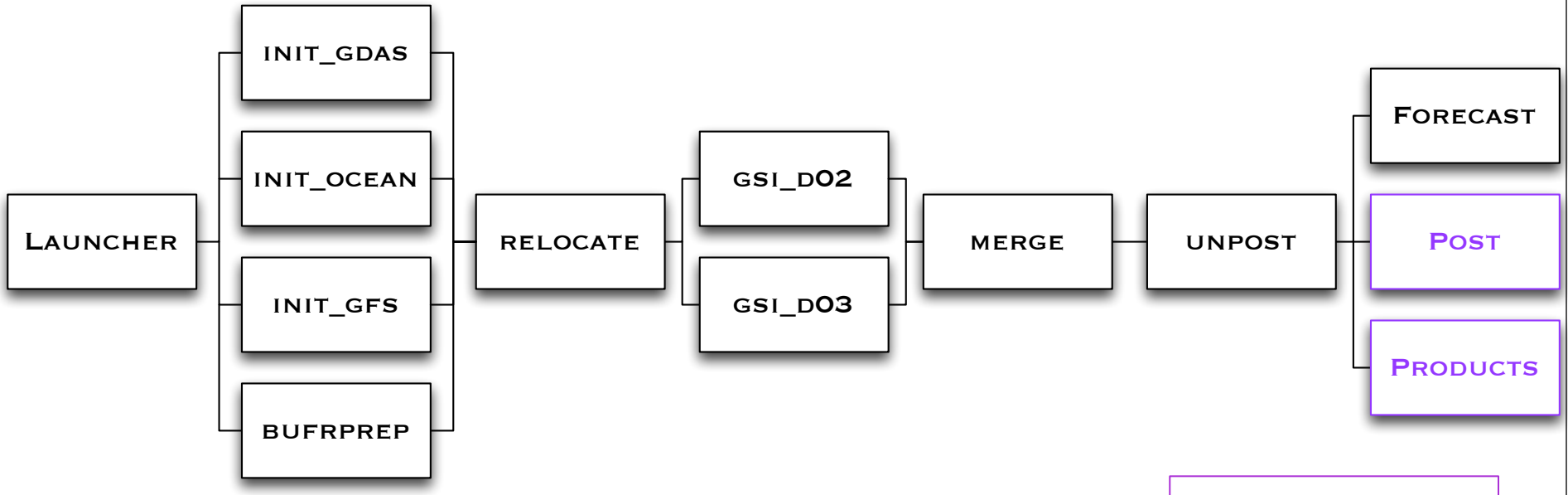
Wrappers: global_vars.ksh

- Each wrapper sources the global_vars.ksh file, which sets a few variables required by each component

```
##### Definition of the Storm ######
export START_TIME=2014101412    # Initial start date
export SID=08L                  # Storm ID
export CASE=HISTORY             # HISTORY OR FORECAST

##### Location of HWRF installation #####
export HOMEhwrf=/PATH/TO/HWRF/INSTALLATION

export EXPT=`echo ${HOMEhwrf} | rev | cut -d/ -f1 | rev`

##### File containing the case-specific variables defined in launcher
#####
export startfile=${HOMEhwrf}/wrappers/$EXPT-${START_TIME}-$SID.start
```

Wrappers

- Wrappers must be submitted in sequence
- Some wrappers may be submitted simultaneously, while others require completion of previous task before submission



Don't submit until forecast job is running

Submitting Jobs

- Each batch system has its own set of requirements for submitting a job
- The following is an example of the resources needed for the GFS Initialization job on NOAA's Jet

```
#!/bin/sh

#PBS -A dtc-hurr
#PBS -l partition=ujet:tjet:sjet:vjet:njet
#PBS -j oe
#PBS -q batch # queue
#PBS -l procs=48
#PBS -l walltime=04:39:00
#PBS -l vmem=40GB
#PBS -N gfsinit

cd $PBS_O_WORKDIR

./init_gfs_wrapper
```

Run ex-scripts Manually

For more detailed information, visit the Doxygen webpage

Running ex-scripts manually

- Pros:
  - Quick way to find simple bugs without waiting on batch queue
  - Potentially a huge time-saver
- Cons:
  - Requires all prior jobs to be run beforehand (database availability), but can be accomplished with automation system
  - Need to create your own wrapper, in essence
    - A bash, ks, or sh script so that you can set the required env variables and load the storm1.holdvars.txt file
  - Each job has different requirements for this method

exhwrf_launch

- Directions are included on the doc webpage for creating your own script to run exhwrf_launch manually
- Suggest using the launcher_wrapper
  - Can be run on a front end node
  - No need to recreate the wheel
- This must be run to set the configuration
  - Each subsequent submission will reset (or change) the configuration files used by all of the HWRF components
- Wrapper creates a start file to make the loading of required env variables easier

Serial and OpenMP Jobs

- For MPI jobs, you must either exit with CTL+C or run it in an interactive batch session
- Larger serial jobs can physically run on the front end nodes, but this is not recommended for resource management reasons
- For jobs that require the env variable `$TOTAL_TASKS` (used to inform the scripts of how many MPI ranks are available), there are two approaches:
  - Check the script for setting up the working directory
    - Use a fake number of MPI tasks, i.e. `$TOTAL_TASKS=1` and exit (CTL+C) script before running the forecast
  - Check the forecast
    - Need to submit from an interactive session

Starting an interactive session

- To start an interactive session, use the –I and –X (X11 forwarding, if needed) options in addition to all the other options needed for your job.

```
$ qsub –I –X <options>
```

- Wait on the request to be granted
- An example for the forecast job:

```
qsub –I –X –A dtc-hurr –l partition=ujet –q
     batch –l procs=1234 –l walltime=06:39:00
     –l vmem=40GB
```

Submitting the job

- Once you have chosen your method for submitting the job, export necessary variables and submit the script

```
cd /path/to/HWRF/scripts
( . /path/to/startfile ; \
    . $COMhwrf/storm1.holdvars.txt \
    TOTAL_TASKS=#### ; \
    $EXhwrf/exhwrf_forecast.py )
```

Depends on choice of FE node (1) or interactive session (1234)

Init & Bdy Jobs

- Require extra environment variables
- Read the `rocoto/tasks/init.ent` and `rocoto/tasks/bdy.ent` files as reference
  - $INIT_MODEL
    - "GDAS" for the FGAT init jobs and
    - "GFS" for the deterministic init.
  - $INIT_FHR
    - 0 for "GFS" or
    - An integer 3, 4, 5, …, 9 for the various parts of the FGAT (INIT_MODEL=GDAS)
  - $INIT_PARTS
    - "3dvar" to process everything needed for the relocation,
    - "bdy" for full forecast length boundary condition processing,
    - "parent" to only run the bare minimum required for a no-init forecast
    - "all" to run everything conceivable.

Archiving & Input Jobs

- Must be run on a node with HPSS access and with sufficient memory (1-3 GB)
- On NOAA machines, the front end nodes and *rdtn*, *transfer*, and *service* queues all suffice

Building your own scripts

- Run the launcher first from wrapper in wrappers directory with desired configuration
- Use start file in wrappers directory and storm1.holdvars.txt in $COMhwrf directory to load required environment variables
- Decide to submit on FE node (not running MPI executables) or in an interactive session (qsub –I <options>)
- Export additional necessary variables using rocoto *.ent files as a reference
- Submit the ex-script

To directly run HWRF Python functions

- Start an interactive session
- Run the launcher
- Start a Python shell

```
cd /path/to/HWRF/scripts
( . /path/to/start/file ; . $COMhwrf/storm1.holdvars.txt \
    export PYTHONPATH=$USHhwrf TOTAL_TASKS=1 ; python )
```

- Initialize the produtil package

```
import produtil.setup
produtil.setup.setup()
```

- Initialize hwrf_expt module

```
import hwrf_expt
hwrf_expt.init_module()
```

To directly run HWRF Python functions

- The entire HWRF system is then accessible via the hwrf_expt module
- An example to get the name of the MOAD domain:

```
print str(hwrf_expt.moad)
```

Prints *moad*

Conceivably, the entire HWRF system could be run this way in a single Python interactive session in an interactive batch job. However, that would be **quite tedious!**