Rocoto-Based HWRF Automation

Sam Trahan

October 16, 2014

Overview

- Other HWRF automation systems
- Overview of the Rocoto + pyHWRF system
- How to configure.
- How to run.
- Configuring by Command Line
- Where is everything?
- Troubleshooting

HWRF Automation Systems

NCO	EMC
Human Operator	HHS
ecFlow	kick_scripts
J-Jobs	J-Jobs
ex-Scripts	ex-Scripts
ush	ush

Inter-cycle dependencies,
Failure recovery

Inter-job dependences
within one forecast cycle

Set up environment for NCO
Does nothing in EMC workflow

High-level logic

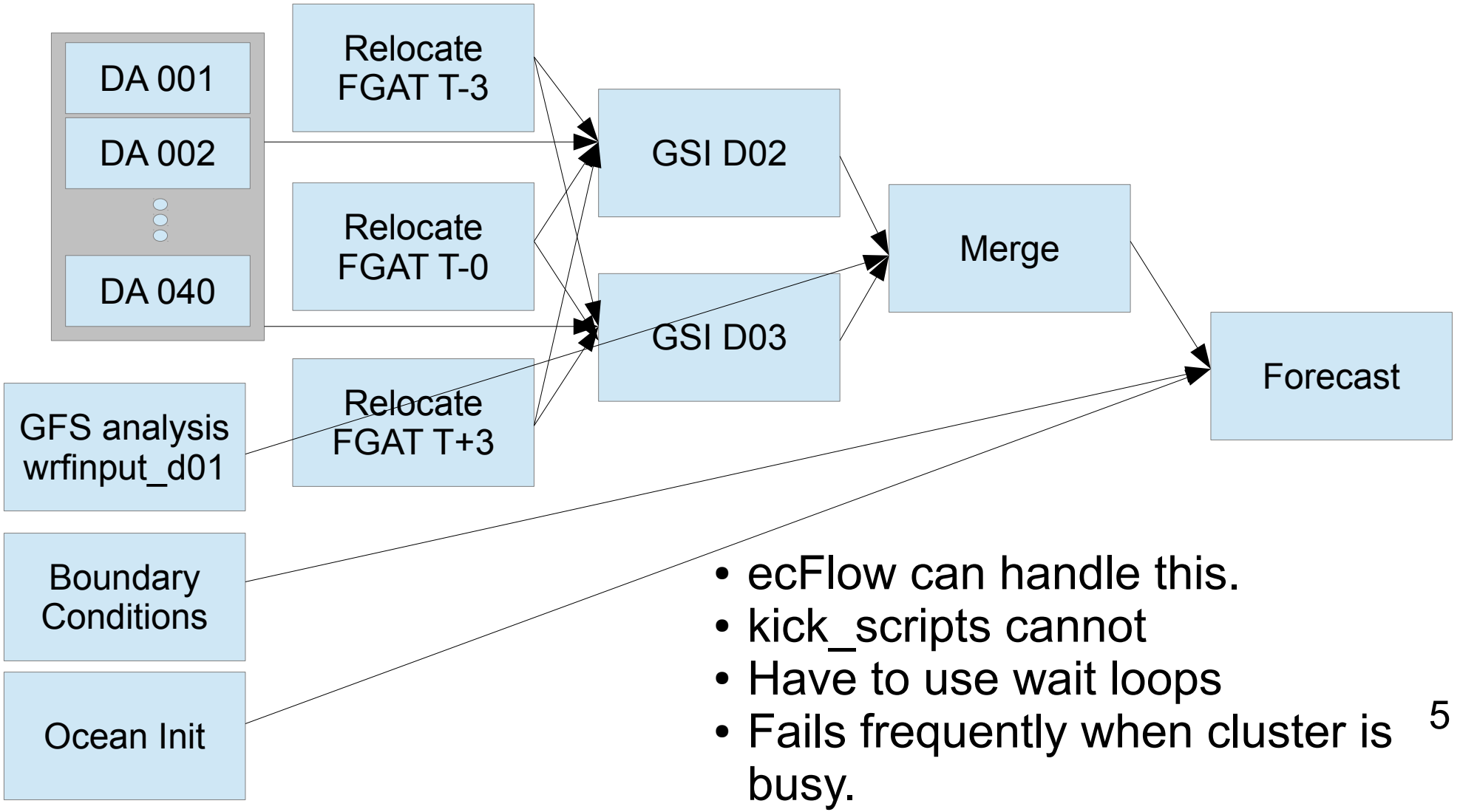
Low-level logic

HWRF Automation Systems

	ecFlow	kick_scripts+HHS
Job submission	**A central server submits jobs. Makes ecFlow unusable to us.**	Within a cycle: **each job submits the next**. HHS submits first job in each cycle.
Dependency tracking	Central server tracks dependencies.	**Have to use wait loops. Limited due to wallclock.**
Multi-year tests	Not designed for this.	Excellent, well-tested.
Failure detection	Based on exit status, files and runtime.	Based on file availability.
Failure reporting	Graphical and logging	Logging, web-based, email-based.
Failure recovery	**24/7 human operators press buttons to resubmit failed jobs.**	**Completely manual. No 24/7 operators, so things break in real-time.**
Configurability	**None.**	Easy: modify hwrf_driver

HWRF Automation Systems

- Jobs submitting jobs does not work:



- ecFlow can handle this.
- kick_scripts cannot
- Have to use wait loops
- Fails frequently when cluster is busy.

HWRF Automation Systems

- Need automated failure recovery
  - ecFlow – requires human intervention
    - But you get to press buttons!
    - There are sad-face icons when jobs fail!
  - kick_scripts+HHS – requires human intervention
    - Scripts instead of buttons.
    - Text files instead of sad-face icons.
  - Major problem for real-time workflows and large tests.

Rocoto+pyHWRF

- What is Rocoto?
  - Formerly the NOAA Workflow Manager
  - Lead dev = Chris Harrop in NOAA ESRL
  - Runs most of the HFIP forecasting and retrospective systems over the past few years.
- XML workflow description
- Run “rocotorun” over and over to track jobs and dependencies.

Rocoto+pyHWRF System

NCO	Old	New
Human Operator	HHS	**Rocoto**
ecFlow	kick_scripts	
ksh J-Jobs	ksh J-Jobs	**XML**
Python ex-Scripts	**Python** ex-Scripts	**Python** ex-Scripts
Python ush	**Python** ush	**Python** ush

Rocoto+pyHWRF System

- Advantages:
  - Rocoto and ecFlow have similar workflow management capabilities
  - Automated failure recovery
  - EMC, DTC, NCO have an identical system below the J-Jobs layer
- XML can be easily translated to J-Jobs for NCO.

NCO	New
Human Operator	Rocoto
ecFlow	
J-Jobs	XML
Python ex-Scripts	Python ex-Scripts
Python ush	Python ush

Rocoto+pyHWRF System

	ecFlow	Rocoto
Job submission	A **central server** submits jobs. Makes ecFlow unusable to us.	A **central database** tracks jobs. **Repeat a command** every ~2-10 minutes to submit jobs (usually via CRON).
Dependency tracking	**Central server** tracks dependencies.	**Repeat a command** every ~2-10 minutes to check dependencies.
Multi-year tests	Not designed for this.	Excellent, well-tested.
Failure detection	Based on exit status, files and runtime.	Based on exit status, files and runtime.
Failure reporting	Graphical and logging	**Manual and logging**
Failure recovery	**24/7 human operators press buttons to resubmit failed jobs.**	**Automated resubmission. Can also manually intervene.**
Configurability	**None.**	**Add arguments to run_hwrf.py**

How to Configure
Simple Config

- Check out and compile:
  - Check out from Subversion
    - GSI is no longer checked out automatically.
  - Load modules, set $PNETCDF, etc.
  - cd sorc ; make ; make install
- Link fix files.
- Set up parm/system.conf
  - Replaces most of hwrf_driver.sh configuration
  - Templates in system.conf.jet, etc.
  - Fairly self-explanatory
- Run run_hwrf.py with arguments every 2-10 minutes.

How to Configure
Physics Schemes, GSI, etc.

- parm/hwrf.conf, parm/hwrf_basic.conf
  - Nearly everything is configured from this file.
  - Only exception is POM, which is not configurable.
- Example:

  [namelist_3km]

  …

  physics.cu_physics=0
- Change to 84 to enable SAS in inner domain.
  - Will set cu_physics=84 in &physics for all WRF runs.

How to Configure
Paths

- parm/system.conf
  - Most paths are set here.
  - [config] CDSCRUB – base scrub directory
  - [config] CDNOSCRUB – where to put tracks
  - [config] CDSAVE – where to find scripts
  - [config] input_catalog – which input source to use.
  - [config] archive – how and where to archive results
- parm/hwrf_input.conf
  - Where to find input data.
  - system.conf's [config] input_catalog decides which set of input sources to use (jet_fcst, zeus_hist, etc.)

How to Configure
Job Cards, Throttling

- rocoto/sites/*.ent – site-specific (uJet, vJet, Zeus) configuration
- rocoto/tasks/*.ent – configure specific tasks
- rocoto/hwrf_workflow.xml.in – configure throttling
  - CYCLE_THROTTLE – maximum number of active cycles
  - COM_SCRUB_TIME – seconds after hwrf_output is done at which to scrub COM. Will also wait until COM is no longer needed.
  - WORK_SCRUB_TIME – seconds after hwrf_output is done at which to scrub work areas. Will also wait until all other jobs are done except com scrubber and special “completion” job.
  - taskthrottle=”20” – maximum number of jobs to run at a time.
- rocoto/cycling_condition.ent – inter-cycle relocate job dependency
- rocoto/env_vars.ent – environment variables to set for all jobs on all platforms.

How To Run
Initial Start

- First, set up the database and run rocoto once by command line:
  - cd /path/to/myHWRF/rocoto
  - ./run_hwrf.py -w 19w2014.xml -d 19w2014.db 2014 19w HISTORY config.EXPT=myHWRF
- If all goes well, first job starts.

How To Run
Continue the Workflow

- After first run_hwrf.py succeeds, rerun with -f added, every five minutes or so:
  - ./run_hwrf.py **-f** -w 19w2014.xml -d 19w2014.db 2014 19w HISTORY config.EXPT=myHWRF
- The -f tells run_hwrf that you are continuing an existing workflow.
- Do this in a CRON job.
  - there are other ways, but that is the safest

How To Run

Arguments to run_hwrf.py

- ./run_hwrf.py -w 19w2014.xml -d 19w2014.db 2014 19w HISTORY config.EXPT=myHWRF
- -w 19w2014.xml
  - Rocoto workflow description AND cycle list
- -d 19w2014.db
  - Used by Rocoto for bookkeeping
- 2014
  - Cycles to run

How To Run
Arguments to run_hwrf.py

- Instead of “2014”
  - 2014100100-2014100718 – range of cycles
  - 2014100500 – a single cycle
  - 2014100500 2014100518 – two specific cycles
- Can also specify forecast ensemble members:
  - 01-20 – run for all GEFS members from 1 to 20
  - 03 05 07 09 – run for these four
  - 01-07 13-15 – run for these ten
  - Must be before storm ID argument.
  - Not merged to trunk yet. Soon...

How To Run
Arguments to run_hwrf.py

- ./run_hwrf.py -w 19w2014.xml -d 19w2014.db 2014 19w HISTORY config.EXPT=myHWRF
- 19w
  - Storm to run
- HISTORY
  - HISTORY=retrospective, FORECAST=real-time
  - Affects input sources, whether we wait for data
- config.EXPT=myHWRF
  - Sets many paths. Must be name of parent of rocoto, ush, parm, etc.

Configuring By Command Line

- Any configuration option can be overridden by the command line when running run_hwrf.py:
  - ./run_hwrf.py -w 19w2014.xml -d 19w2014.db 2014 19w HISTORY config.EXPT=myHWRF **config.run_gsi=no archive=disk:/arch/out.tar.gz**
- No need to edit config files!
- Just make sure you run the same arguments each time. (Changing mid-storm is bad.)

Configuring By Command Line

- You can also send a config file:
  - ./run_hwrf.py -w 19w2014.xml -d 19w2014.db 2014 19w HISTORY config.EXPT=myHWRF **../parm/hwrf_wpac_2013.conf**
- Puts many configuration changes in one file
- Easy way to have many configurations in one branch.

Where is Everything?

- EMC, DTC and others had different locations for various components.
- Standardized locations were chosen.
- Original EMC locations were nonsensical.

Where is Everything?

- $WORKhwrf
  - runwrf – forecast execution directory
  - gsi_d0* – GSI
  - tracker.* – trackers
  - fgat.(date) – fgat initialization
  - gdas.(date) – merge
  - gfsinit – init based on GFS
  - ensda – DA ensemble based on GFS ENKF
  - regribber – GRIB processing
  - intercom – more on this later

Where is Everything?

- inside fgat.(date) and gfsinit:
  - wps – run directory for analysis time WPS
  - prep – prep_hybrid
  - realinit – initialization-length real_nmm
  - wrfanl – creation of wrfanl files
  - wrfghost – creation of wrfghost files
  - realfcst – creation of wrfbdy file
  - wpsfcst – forecast-length WPS, if enabled

Where is Everything?

- intercom
  - For communicating between HWRF components
  - EMC ksh system had components copying files into and out of each other's directories
    - caused many bugs, especially with init and GSI jobs
  - Instead: if another job needs it, put it in intercom
    - except for runwrf directory
    - forecast job cannot run Python script in parallel for file delivery

Where is Everything?

- jlogfile – combined logging from all cycles of all storms
  - /ptmp/$USER/myHWRF/2014100500/19W/
  - /ptmp/$USER/myHWRF/log/jlogfile
- Greatly improved over prior years' jlogfile logging.
- Same purpose and contents of NCO jlogfile

Troubleshooting
Overview

- The jlogfile
- Querying the Rocoto Workflow
  - rocotostat
  - rocotocheck
  - “DEAD” jobs
- rocotorewind: resubmit jobs
- storm1.conf: change a cycle's configuration
- manually run parts of the workflow

Troubleshooting
jlogfile

- Workflow status questions?
  - Why did my job fail?
  - Was this cycle a cold start?
  - When did I last run Rocoto?
  - Did I start 19W yet?
  - Has my 08L forecast reached hour 48 yet?
- LOOK IN THE JLOGFILE
  - All significant events are logged.

Troubleshooting
rocotostat

- Has Rocoto given up on any jobs yet?
- What jobs and cycles have completed?
  - rocotostat -w 19w2014.xml -d 19w2014.db -c ALL
  - Jobs that failed too many times will be listed as “DEAD”
  - Jobs that have completed successfully are listed as “SUCCESSFUL”
  - The special “completion” job marks the end of the workflow for that cycle

Troubleshooting
rocotocheck

- Why is Rocoto not submitting this job?
  - rocotocheck -w 19w2014.xml -d 19w2014.db -c 201410041800 -t gsi_d02_E99
  - NOTE: 12 digit cycle with minutes=00
  - Look in rocotostat output for task names (ie.: gsi_d02_E99)

Troubleshooting
Resubmitting Jobs: rocotorewind

- Rocoto, please resubmit these jobs.
  - rocotorewind -w 19w2014.xml -d 19w2014.db -c 201410041800 -c 201410050000 -t atmos_forecast_E99 -t unpost_E99 -t post_E99 -t post_helper_E99
  - NOTE: 12 digit cycle (minutes=00)
  - You can specify multiple tasks and cycles.
  - DISABLE YOUR CRON JOB before doing this if you are going to run rocotorewind multiple times, or it may submit jobs in between.

Troubleshooting
Where are the rocoto* commands?

- Don't use the Jet/Zeus default Rocoto!!
  - We have our own Rocoto for now.
- Look in ush/hwrf_pre_job.ksh.inc.
  - varies based on the system
  - may change in the future
- Add them to your $PATH:
  - sh: export PATH=/path/to/rocoto/bin:$PATH
  - csh: setenv PATH /path/to/rocoto/bin:$PATH

Troubleshooting
Changing a Cycle's Configuration

- Changing parm/*.conf and run_hwrf.py arguments does not affect started cycles.
- $COMhwrf/storm1.conf
  - Contains the cycle's configuration.
  - Result of merging many conf files into one during exhwrf_launch.py job.

Troubleshooting
Manually Running Parts of the Workflow

- Useful for debugging.
- Step 1: get an interactive job:
  - qsub -I -q debug -l partition=ujet:tjet:sjet:vjet:njet -A hwrfv3 -l procs=32 -d . -l walltime=00:30:00
- Step 2: set up environment:
  - bash
  - . /path/to/com/2014100418/19W/storm1.holdvars.txt
  - export TOTAL_TASKS=32 (or whatever you requested)
  - export PYTHONPATH=$USHhwrf

Troubleshooting
Manually Running Parts of the Workflow

- Step 3:
  - python
- Step 4:
  - import produtil.setup, hwrf_expt
  - produtil.setup.setup()
  - hwrf_expt.init_module()
- Step 5: run whatever you were going to run.
- Example: run the tracker:
  - hwrf_expt.tracker.run()

Troubleshooting
Manually Running Parts of the Workflow

- Step 3:
  - python
- **Alternate** Step 4:
  - import produtil.setup, hwrf_expt**, logging**
  - produtil.setup.setup(**level=logging.DEBUG**)
  - hwrf_expt.init_module()
- Step 5: run whatever you were going to run.
- Example: run the tracker:
  - hwrf_expt.tracker.run()

Troubleshooting
Manually Running Parts of the Workflow

- Alternate Step 3-5: run an ex-script instead of running python:
  - export TOTAL_TASKS=9
  - $EXhwrf/exhwrf_ocean_init.py
- Some ex-scripts expect environment variables that specify the forecast hour, domain or input model. See $HOMEhwrf/rocoto/tasks/*.ent.

Future Developments

- check_hwrf.py – summarize status of HWRF by examining logs and rocotostat
- Add wildcards to rocotorewind

- still taking feature requests