Jon Zarling

Calibration Workflows with Cylc

10/8/2024

Jefferson Lab

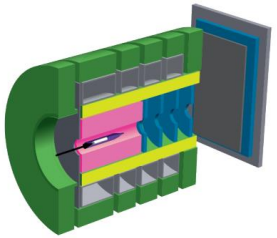
Calibration Motivation

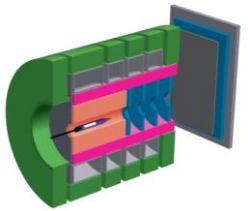
- Often, calibrations need to be done sequentially
- Simple example: tracking → EM calorimeters @ Hall D

CDC
(time-to-distance, dE/dx)

Use preliminary CDC calib. for BCAL

BCAL
(z-position)

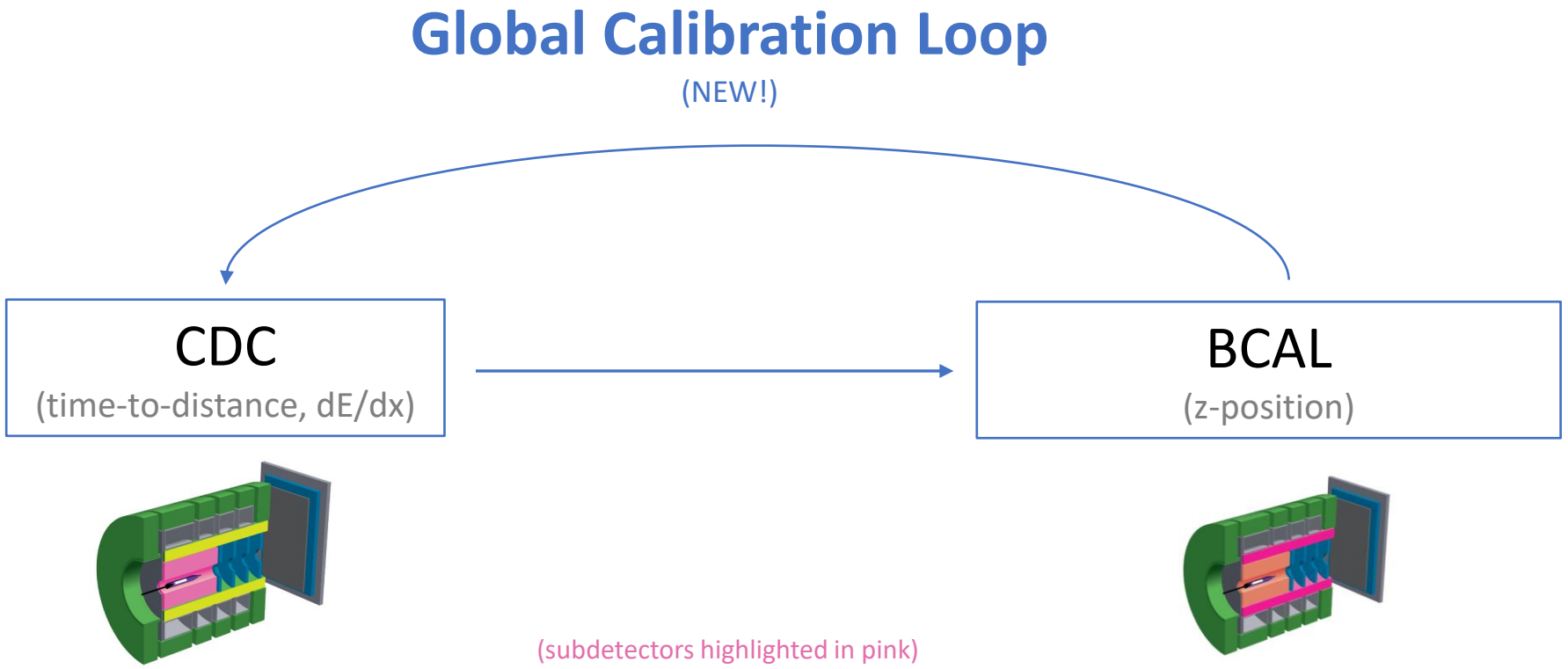




(subdetectors highlighted in pink)

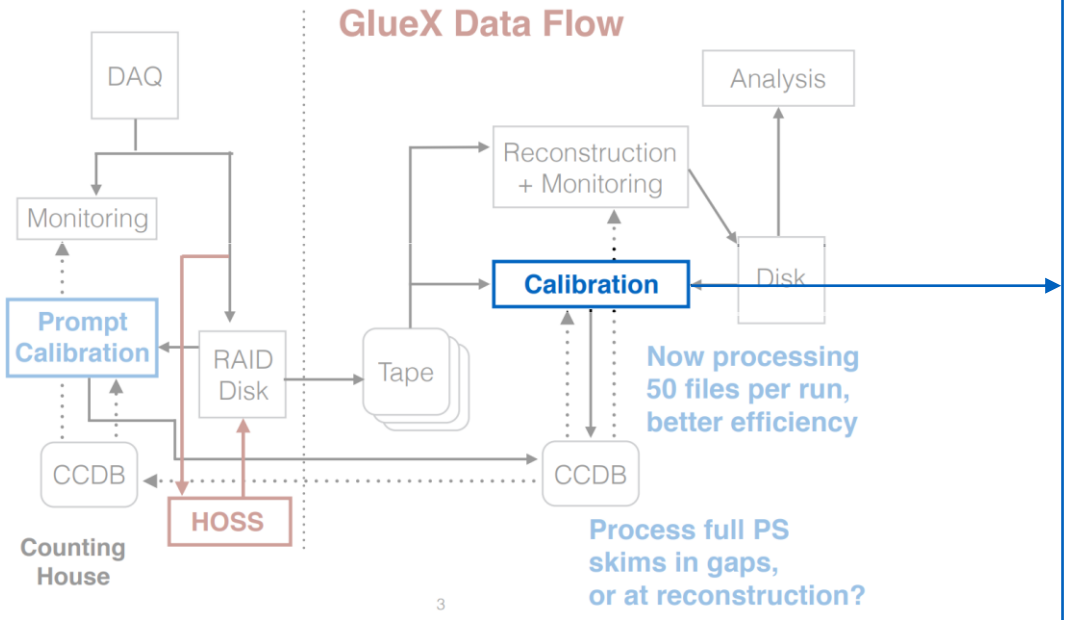
Calibration Motivation

Will iterating over multiple subsystems improve calibrations?



Calibration Motivation, cont.

Usually situation is more complex.
Motivation to use cylc (pronounced “silk”)



Run-dependent Calibrations

Calibration	Person
Overall Timing (first pass)	Sean Dobbs
Overall Timing (post-ST updates)	Sean Dobbs
CDC wire gains	Naomi Jarvis
CDC overall gains	Naomi Jarvis
CDC dE/dx	Naomi Jarvis
CDC time-to-distance	Naomi Jarvis
PS/PSC Timing	Olga Cortes
TAGH timewalks	Olga Cortes
TAGM timewalks	Sean Dobbs/Richard Jones
TOF	Beni Zihlmann
DIRC	Justin Stevens

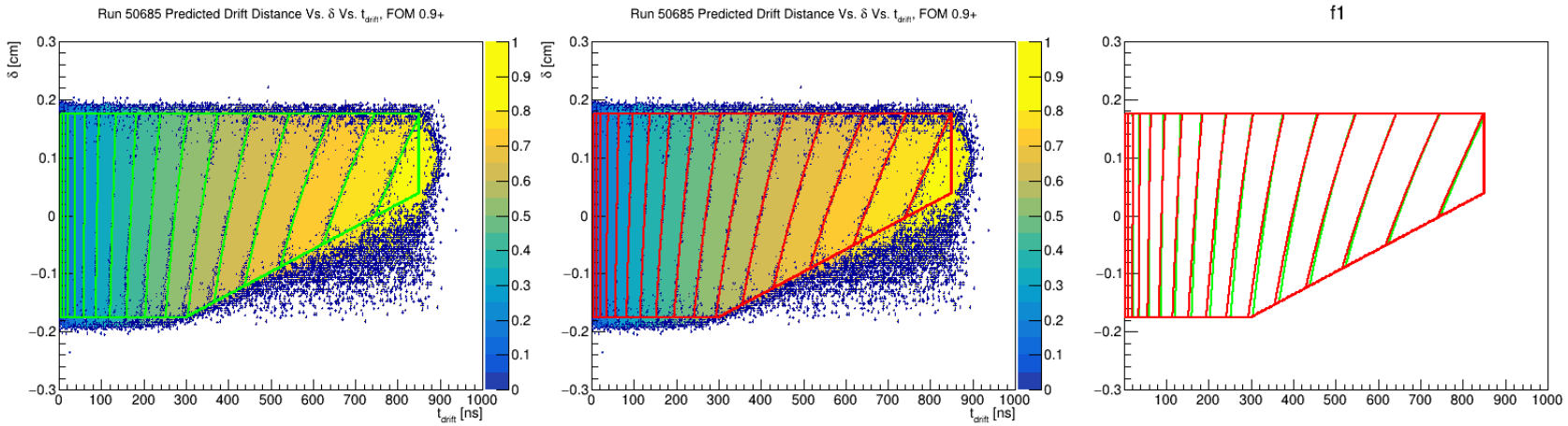
Overall Calibrations

Calibration	Person	Status
BCAL channel timing	Mark Dalton	
BCAL attenuation length/gain ratio	Mark Dalton	
BCAL z-position	Mark Dalton	
BCAL gains/non-linearities	Karthik Suresh	
FCAL gains/non-linearities	Colin Gleason	
FCAL timing	Colin Gleason	
SC Timewalks	Rupesh Dotel	Done
SC Propagation Time	Rupesh Dotel	

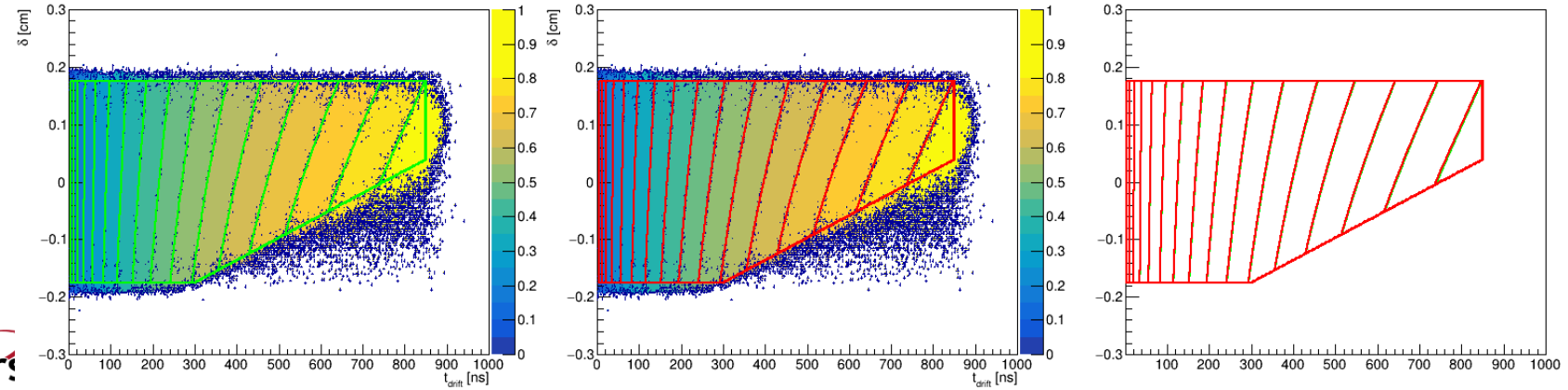
Case Study: CDC Time-to-Distance (TTOD)

Sometimes procedure need to iterate/converge, too

Converged when red totally lies on top of green



Red: 0th iteration (pre-calib)
Green: 1st iteration

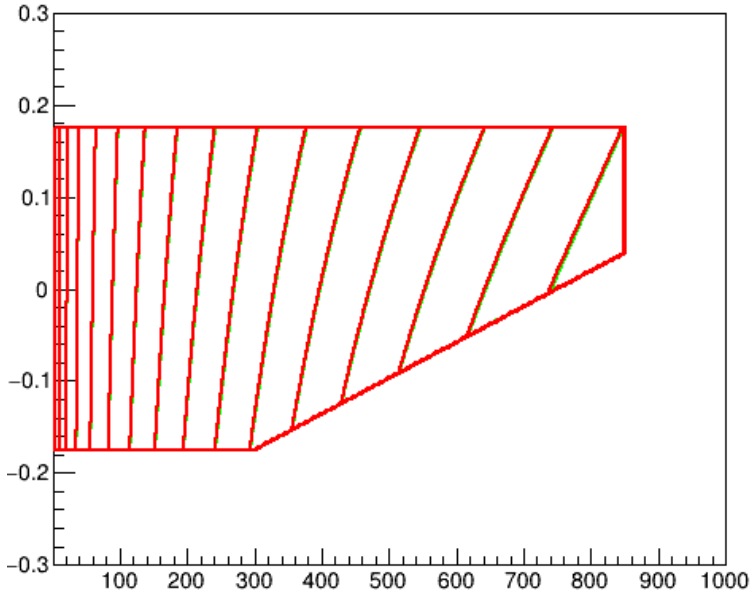


Red: 1st iteration
Green: 2nd iteration

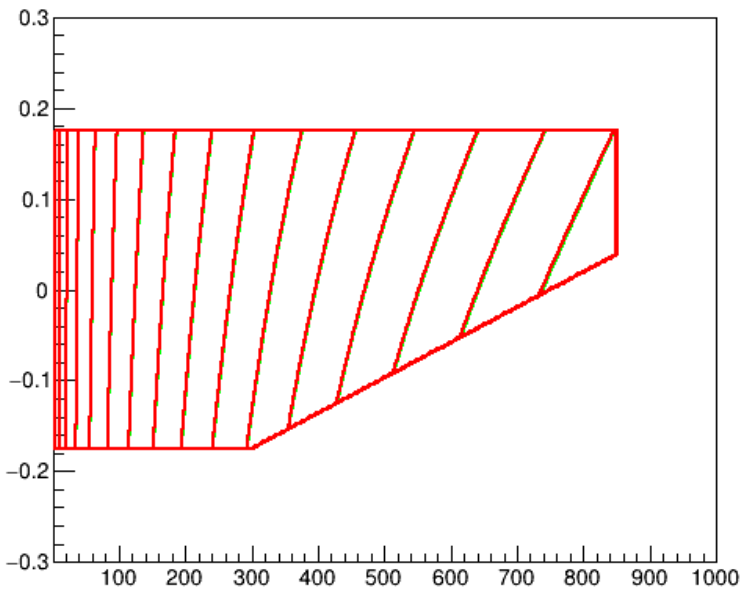
Case Study: CDC Time-to-Distance (TTOD)

Here: converged when red totally lies on top of green

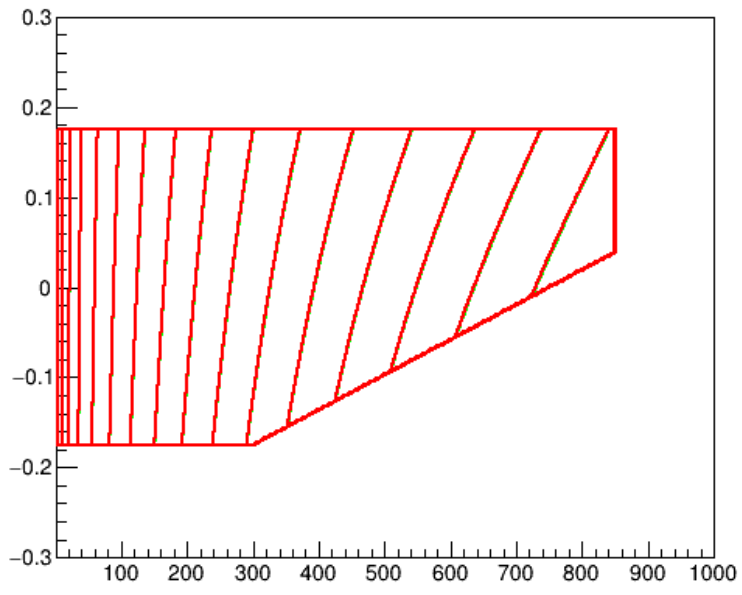
Red: 1st iteration
Green: 2nd iteration



Red: 2nd iteration
Green: 3rd iteration



Red: 7th iteration
Green: 8th iteration



Resolution reduced by ~ 7%
(1st to 8th iteration)

Calibration Workflows with cylc

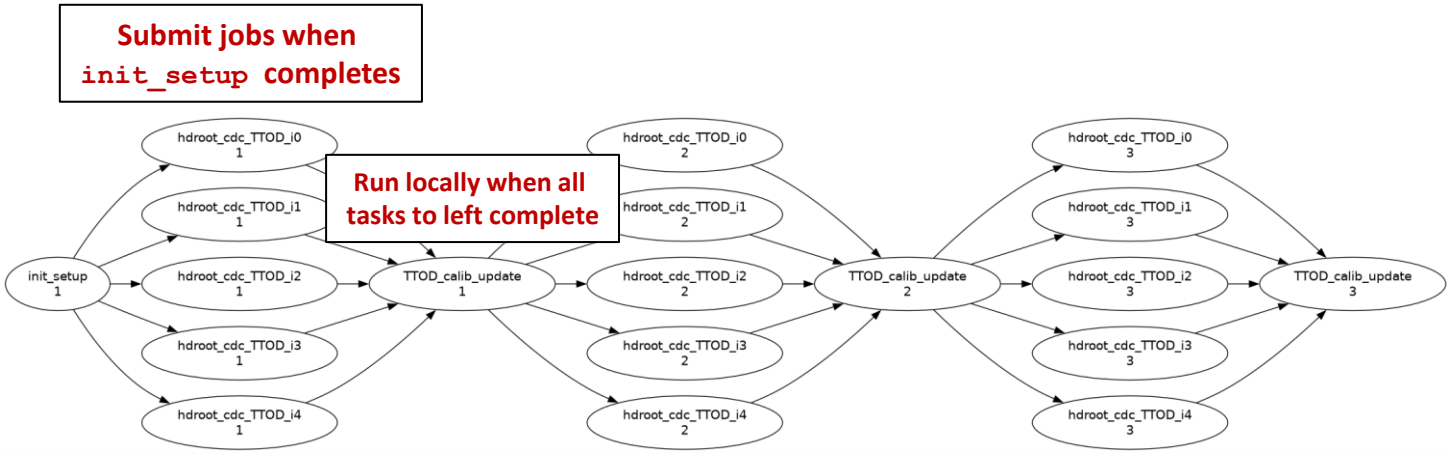
(pronounced “silk”)

Punchline: I did all this from a single terminal command!

```
> cylc vip gx_ttod
```

- Runs 20 jobs per iteration
- Then repeat ×8 iterations
- Read/write ccdb (local copy)
- Automatically resubmits failed jobs
  (but only if I asked it to)

Workflows can be written as graphs



(developed, primarily used for weather & climate science)

Jefferson Lab

Defining Workflows

Workflows described in file called flow.cylc

Gives graph of tasks to run

```
[scheduling]
    cycling mode = integer
    initial cycle point = 1
    final cycle point = 8
    [[graph]]
        # First cycle also needs initial setup
        R1 = "init_setup => hdroot_cdc_TTOD<i> => TTOD_calib_update" # R1: run once (at start)
        # Subsequent cycles begin after `TTOD_calib_update` from last cycle finishes
        P1 = "TTOD_calib_update[-P1] => hdroot_cdc_TTOD<i> => TTOD_calib_update" # P1= run every
        "cycle point", P2=every other, P3=every third, ...
        R1/P0 = "TTOD_calib_update[-P1] => hdroot_cdc_TTOD<i> => TTOD_calib_update => commit_ccdb"
        # Run once at final cycle point (denoted P0)
```

Defining Workflows, cont.

Workflows described in file called flow.cylc

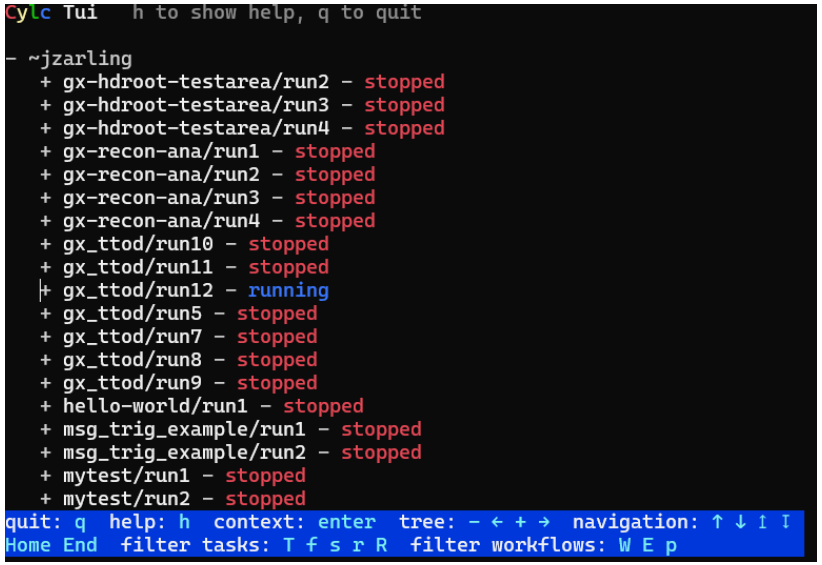
Add `inherit = gx_recontask` to run as batch job
(otherwise runs locally)

```
[[hdroot_cdc_TTOD<i>]]
    inherit = gx_recontask
    script = """
        source gx_env.sh
        hd_root --config=${jana_config} ${evio_folder}/hd_rawdata_0${run}_${fnum}.evio
        mv hd_root.root ${ttod_topdir}/root/hd_root_TTOD_${run}_${fnum}_CP$
        {CYLC_TASK_CYCLE_POINT}.root
    """
```

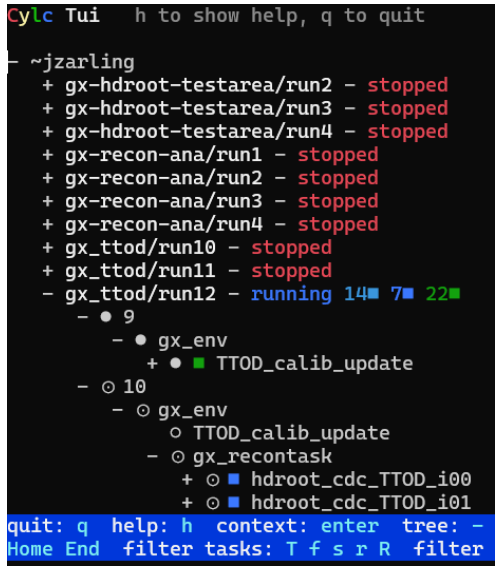
Calibration Workflows with cylc

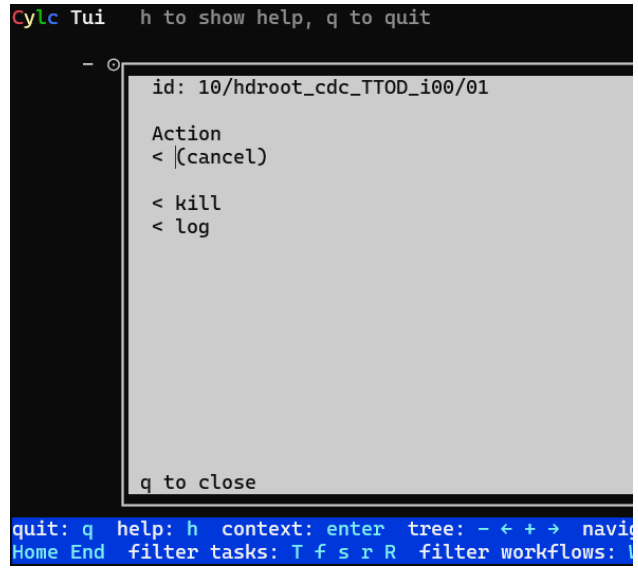
Terminal interface (interactive!):

```
> cylc tui
```



Mouse click in terminal to expand



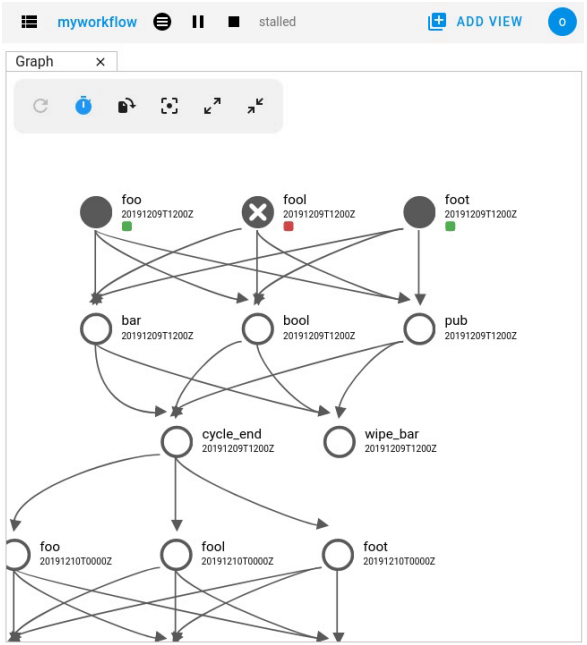


Compare to swif2

(also has cool GUI version)

Maybe One Day?

Web GUI via Jupyter Hub setup:



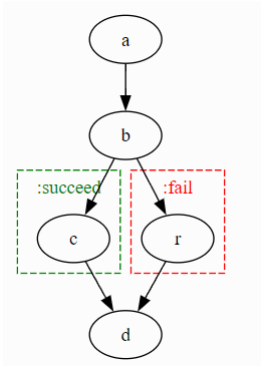
An online dashboard for calibrations in process

- Multiple users can access, monitor, modify, etc.
- Different users can have different privilege levels
  - Person A: global control
  - Person B: can start/stop/modify jobs related to their subdetector
  - Person C: can look at monitoring plots

Further inspiration

Other Nifty Features

- Supports job submission to remote machines
- Workflows can support:
  - Trigger startup: on regular clock cycle
  - Trigger startup: on file appearance
  - Branching logic
  - Workflows nested in other workflows

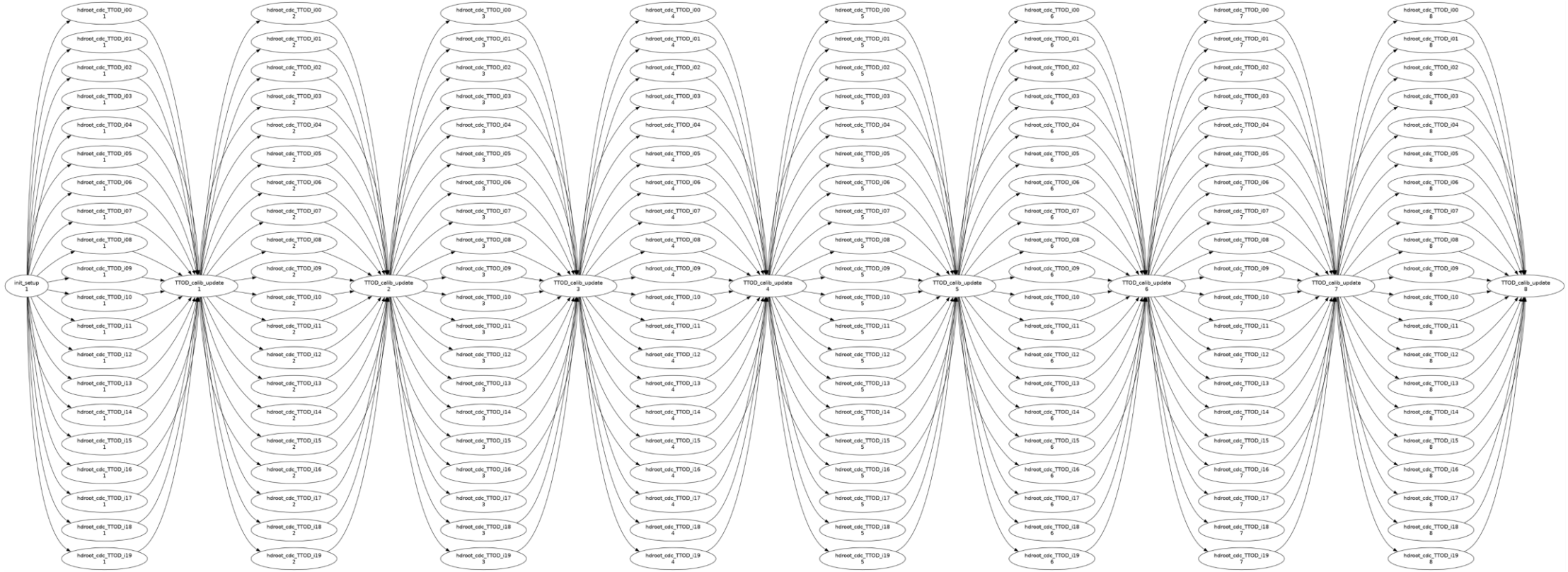


- Local database to store/query workflow info maintained

Summary

- Cylc seems like a great multipurpose tool https://cylc.github.io/
  - Functionality of swif2 (+more)
- Here: use for calibration workflows
  - A few more details here: https://wiki.jlab.org/epsciwiki/index.php/File:Jz_gluex_calib_10.2.24.pdf
- Potential for other use cases
  - MCWrapper offloading from OSG?
  - Hydra?
- Plan to put examples on wiki, Github
  - Let me know if there's interest in starting sooner

Backup: Full Workflow Graph



Jefferson Lab

Backup: Defining Workflows

Workflows described in file called flow.cylc

Gives graph of tasks to run

```
[scheduling]
    cycling mode = integer
    initial cycle point = 1
    final cycle point = 8
    [[graph]]
        # First cycle also needs initial setup
        R1 = "init_setup => hdroot_cdc_TTOD<i> => TTOD_calib_update" # R1: run once (at start)
        # Subsequent cycles begin after `TTOD_calib_update` from last cycle finishes
        P1 = "TTOD_calib_update[-P1] => hdroot_cdc_TTOD<i> => TTOD_calib_update" # P1= run every
        "cycle point", P2=every other, P3=every third, ...
        R1/P0 = "TTOD_calib_update[-P1] => hdroot_cdc_TTOD<i> => TTOD_calib_update => commit_ccdb"
        # Run once at final cycle point (denoted P0)
```

Backup: Defining Workflows, cont.

Workflows described in file called flow.cylc

A single “task”

```
57      [[init_setup]]
58          inherit = gx_env
59          script = """
60              source gx_env.sh
61              ccdb_LocalUpdate.sh
62              cd ${CYLC_WORKFLOW_SHARE_DIR}
63              mkdir -p ccdb/ccdb_add ccdb/ccdb_dump ${ttod_topdir}/root
64          """
```

Backup: Defining Workflows, cont.

Workflows described in file called flow.cylc

Defining how to run batch jobs

```
# Reconstruction default job (MANY-THREADED!)
[[gx_recontask]]
    inherit = gx_env
    platform = jlab_slurm # Defined in my global.cylc file
    execution retry delays = 3*PT10S # I allow 3 timeout retries before declaring "failed"
    # Submit to the host system
    # job requires 5000MB of RAM and TMP disk, 16 CPUs
    [[[directives]]]
        --mem = 5000 # MB
        --ntasks = 16 # Threads
        --tmp = 5000 # MB
        --time = 420 # walltime (minutes)
        --nodes = 1-1 # -N, --nodes=N, number of nodes on which to run  (N = min[-max])
```

Backup: Defining Workflows, cont.

Workflows described in file called flow.cylc

Add `inherit = gx_recontask` to run as batch job
(otherwise runs locally)

```
[[hdroot_cdc_TTOD<i>]]
    inherit = gx_recontask
    script = """
        source gx_env.sh
        hd_root --config=${jana_config} ${evio_folder}/hd_rawdata_0${run}_${fnum}.evio
        mv hd_root.root ${ttod_topdir}/root/hd_root_TTOD_${run}_${fnum}_CP$
        {CYLC_TASK_CYCLE_POINT}.root
    """
```