

SWIF and hdswwif

August 11, 2015
Kei Moriya

SWIF

- Scientific Workflow Indefatigable Factotum
- Developed by Chris Larrieu in SciComp group larrieu@jlab.org
- <https://scicomp.jlab.org/docs/swif>
- Advantages:
 - ▶ Better control
 - Auger does not relate tape retrieval and jobs efficiently
 - Auger submission is slow (Java parsing XML)
 - Control over jobs in phases
 - ▶ Info on jobs
 - Success/problems
 - Resource usage

Using SWIF Directly

```
> swif create my_workflow
```

create workflow

swif command

```
> swif add-job -workflow my_workflow -project gluex -track reconstruction -cores 6 -disk 40g  
-ram 5g -time 8h -os centos65  
-input hd_rawdata_002333_000.evio mss:/mss/halld/RunPeriod-2014-10/rawdata/Run002333/  
hd_rawdata_002333_000.evio  
-stdout /home/gxproj1/logfiles/stdout  
-stderr /home/gxproj1/logfiles/stderr  
-name my_job  
/home/gxproj1/script.sh TAGH_online 002333 000 6
```

swif options

script to execute with options

register job

```
> swif run my_workflow
```

run jobs: SWIF will dispatch jobs as it deems fit

```
> swif status my_workflow
```

check status

```
> swif status my_workflow -runs
```

check status for individual jobs

```
> swif cancel my_workflow -delete
```

cancel jobs, option -delete removes workflow

hdswif

- For most part just a wrapper around swif, but also
 - ▶ use config file to set up jobs more easily
 - ▶ organize SWIF output to check job status
- Check out from svn: <https://halldsvn.jlab.org/repos/trunk/scripts/monitoring/hdswif>
- Written in python, uses PyROOT (officially ROOT supported Python extention) for ROOT functionality (histograms, graphics, etc.)
- Offline monitoring code developed by Sean was heavily referenced for development, config file similar to Nathan's Julia scripts

How swif works

- Can run by typing swif in ifarm terminal
- Jobs are grouped into workflows, most natural to have separate workflows for different kinds of jobs
- Can get info with

```
> swif help
```

```
> swif help [command]
```
- List all workflows:

```
> swif list
```
- Show workflow status:

```
> swif status [workflow]
```

Setting Up hdswwif

- Check out source code: <https://halldsvn.jlab.org/repos/trunk/scripts/monitoring/hdswwif>
- Provide valid \$ROOTSYS compiled with PyROOT enabled, have \$ROOTSYS/bin in PATH, \$ROOTSYS/lib in LD_LIBRARY_PATH
- hdswwif will set **PYTHONPATH** to contain \$ROOTSYS/bin internally
- The officially supported ROOT builds on the JLab ifarm CentOS6.5

/group/halld/Software/builds/Linux_CentOS6-x86_64-gcc4.4.7/root/prod

has PyROOT enabled

Wrappers in hdswwif

- All of the following are equivalent:

swif command	hdswwif command	what it does
swif list	hdswwif.py list	Lists all workflows for user
swif create [workflow]	hdswwif.py create [workflow]	Create new workflow
swif status [workflow]	hdswwif.py status [workflow]	Show status of workflow
swif run [workflow]	hdswwif.py run [workflow]	Start submitting jobs
swif cancel [workflow]	hdswwif.py cancel [workflow]	Cancel running jobs

Commands wrapped for convenience, but calling swif directly is faster

Additions in hdswwif.py

hdswwif.py command	result
hdswwif.py fullstatus [workflow] [format]	Show status of all runs, equivalent to swif status [workflow] -runs -summary -display [format] [format] is one of simple, xml, json
hdswwif.py run [workflow] [njobs]	Run only [njobs], equivalent to swif run [workflow] -joblimit
hdswwif.py delete [workflow]	Stop all running jobs and delete workflow. Equivalent to swif cancel [workflow] -delete
hdswwif.py summary [workflow]	Create XML file for all jobs, and summary webpage of results based on this XML file

Adding Jobs

- In swif:

swif command

```
> swif add-job -workflow my_workflow -project gluex -track reconstruction -cores 6 -disk 40g  
-ram 5g -time 8h -os centos65  
-input hd_rawdata_002333_000.evio mss:/mss/halld/RunPeriod-2014-10/rawdata/Run002333/  
hd_rawdata_002333_000.evio  
-stdout /home/gxproj1/logfiles/stdout  
-stderr /home/gxproj1/logfiles/stderr  
-name my_job  
/home/gxproj1/script.sh TAGH_online 002333 000 6
```

swif options

script to execute with options

- In hdswhif:

```
> hdswhif.py add my_workflow -c myconfig.txt
```

all options handled with config file

Config File Example

- Included example input.config:

PROJECT	gluex
TRACK	reconstruction
OS	centos65
NCORES	2
DISK	40
RAM	5
TIMELIMIT	8
JOBNAMEBASE	myjobs
RUNPERIOD	2015-03
VERSION	99
OUTPUT_TOPDIR	/volatile/halld/test/RunPeriod-[RUNPERIOD]/ver[VERSION]
SCRIPTFILE	/home/gxproj5/halld/hdswif/script.sh
ENVFILE	/home/gxproj5/halld/hdswif/setup_jlab-2015-03.csh

Example of other variables included in variable.

Must specify full path

- These will overwrite defaults - safer to specify them in config file

Config File Variables

- Variables should be obvious for the most part
- RUNPERIOD is used to look for file in /mss/halld/RunPeriod-[RUNPERIOD]/rawdata/ for evio data files
- VERSION is a variable for offline monitoring - do not need to use
- Config variables can depend on each other:

OUTPUT_TOPDIR	/volatile/halld/test/RunPeriod-[RUNPERIOD]/ver[VERSION]
---------------	---

- Replacements will be made (do not put in cyclic dependencies)

User Tags

- To easily access information about jobs, SWIF allows user-defined tags
- Specify with -tag [tag name] [tag value] when using SWIF
- hdswwif gives each job tags user_run and user_file for the run and file specified for parsing output

Specifying Runs/Files

- Specify runs, files with -r [RUN] and -f [FILE]
- If option -r is not specified, will run over ALL runs in RUNPERIOD
- Same for files
- Can use UNIX-style wildcards

```
> hdsrif.py add my_workflow
```

Run over all runs, files in RUNPERIOD

```
> hdsrif.py add my_workflow -r 3185
```

Run over all files in run 003185

```
> hdsrif.py add my_workflow -r 3185 -f 1
```

Run over file 001 in run 003185

```
> hdsrif.py add my_workflow -r '318[05]' -f '01[0-9]'
```

Run over files 010-019 in run 003180, 003185

Re-Submitting Jobs

- In swif:

swif command

modification

problem type

```
> swif modify-jobs -workflow my_workflow -ram add 2gb -problems AUGER-OVER_RLIMIT
```

- In hdswhif:

```
> hdswhif.py resubmit my_workflow 3
```

Additional option for adding memory
(+2GB is default)

Currently only supports AUGER-OVER_RLIMIT, can add more resolutions to different types of problems as they show up

Job Summary

- In hdswwif:

```
> hdswwif.py summary my_workflow
```

- Creates html page summary_swif_output_[workflow].html

Note:
Need CSS file mystyle.css for
html formatting, included in svn

summary of jobs

offline_monitoring_RunPeriod2015_03_ver11_hd_rawdata

Status

Info retrieved: 2015-08-10 07:30:19.0

Job Limit: 0

Total attempts: 4314

Undispatched	Dispatched	Succeeded	Failed	Problems	Canceled	Total
0	0	4237	0	77	0	4314

results

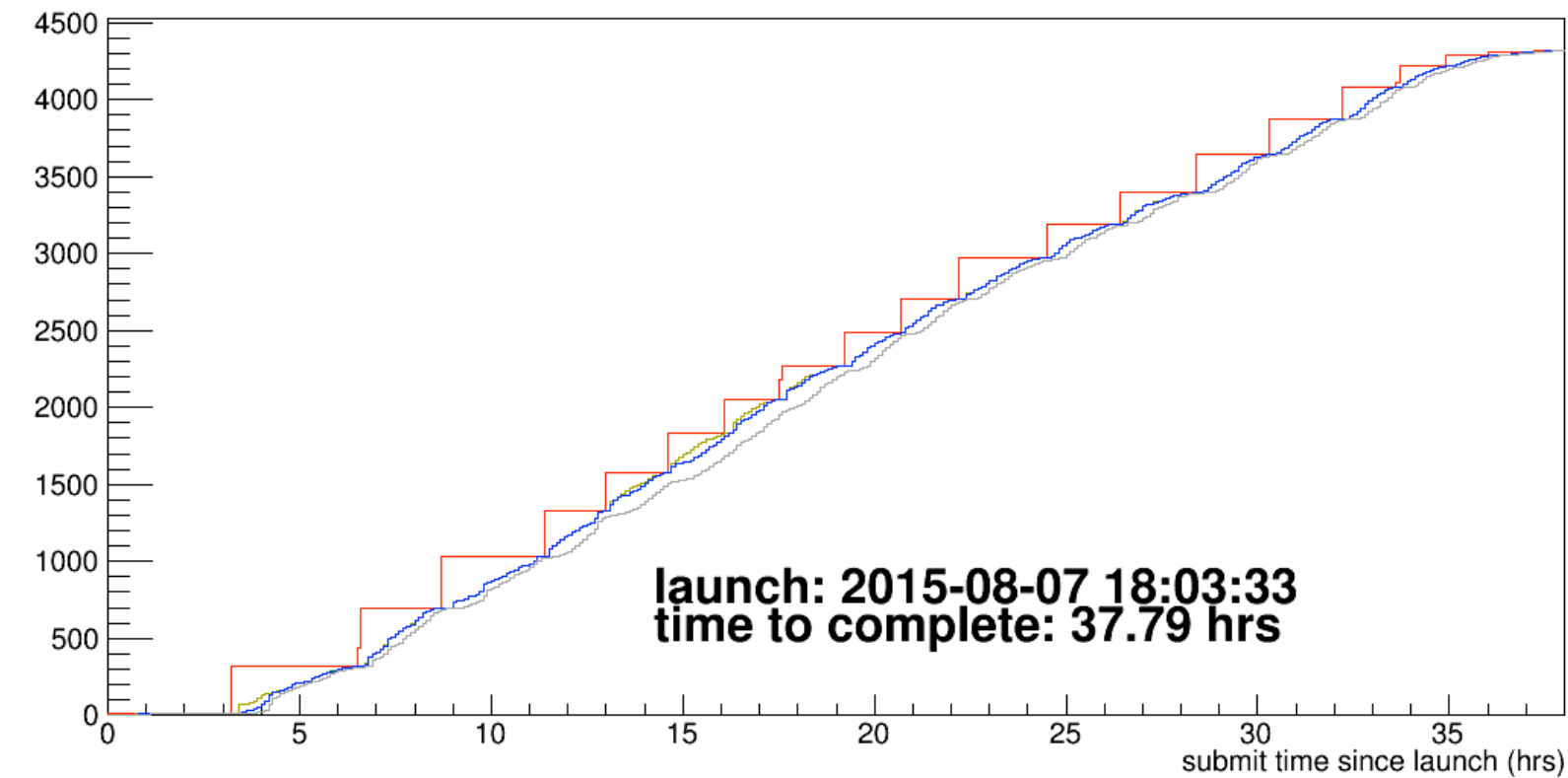
Auger Status

Dependency	Pending	Staging In	Active	Staging Out	Finishing	Dispatched Total
0	0	0	0	0	0	0

active jobs

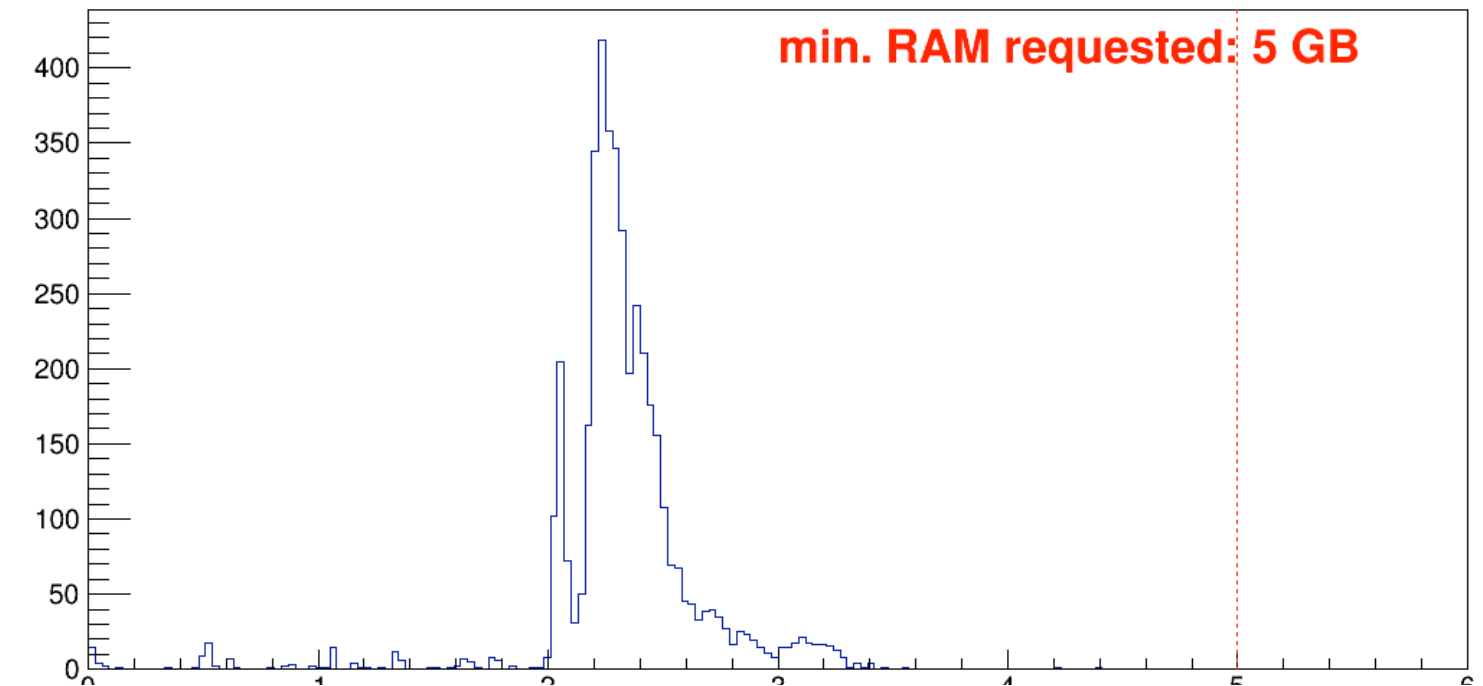
Job Summary Figures

- All included in same generated webpage

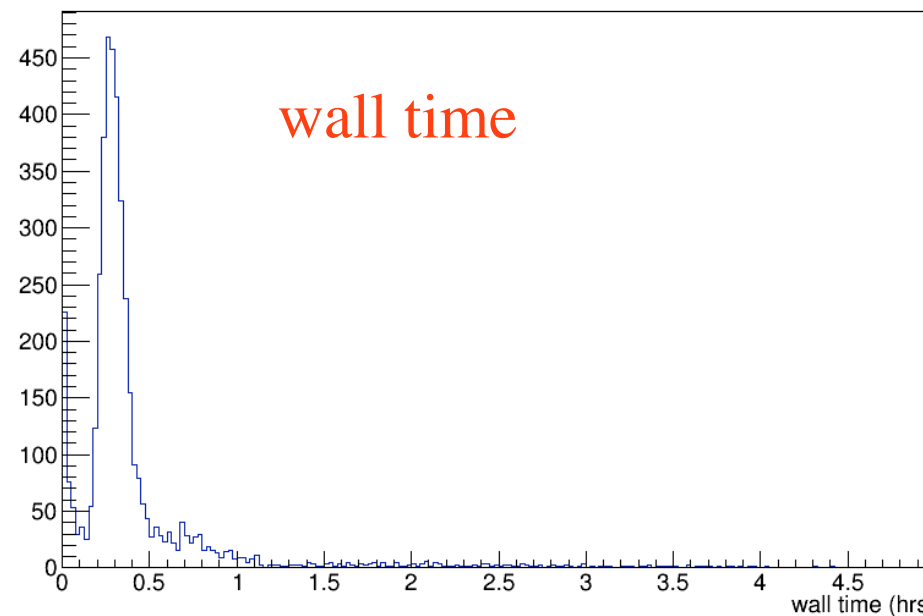


of jobs reaching each stage of processing against time

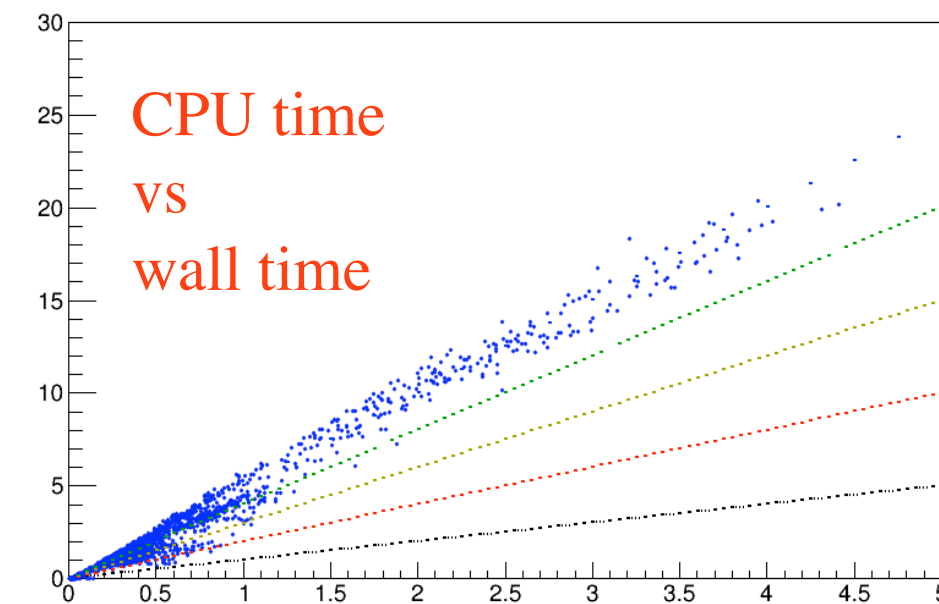
RAM usage



wall time



CPU time
vs
wall time



hdswwif Cheatsheet

hdswwif command	what it does
hdswwif.py create [workflow]	Create workflow
hdswwif.py status [workflow] (format)	Check status
hdswwif.py add [workflow] -c (config file) -r (runs) -f (files)	Add jobs to workflow
hdswwif.py run [workflow]	Run workflow
hdswwif.py run [workflow] (n)	Only run n jobs
hdswwif.py resubmit [workflow] (n)	Resubmit jobs with nGB of additional RAM
hdswwif.py cancel [workflow]	Cancel all dispatched jobs
hdswwif.py delete [workflow]	Delete workflow

TO DO

- Working with Chris on quicker batch submission
- Better control of resubmitting failed jobs
- Freeze option to stop modifications to workflow
- Questions, comments, requests are very welcome