GSPDA Software and Computing Boot Camp (Day 2)

“Hands-on” SWIF2 Farm Submission Example

May 23, 2025

Brad Sawatzky

"Scientific Workflow Indefatigable Factotum”

- Or SWIF, to save time...
  - → Swif is a workflow manager designed at JLab
  - → Interfaces intelligently with the JLab tape system and slurm, the job scheduler
    - » will efficiently pre-stage file dependencies
    - » will efficiently ensure output files are placed on tape
  - → Command line tool: swif2
  - → Web monitoring

- SWIF == Workflow manager
  - → provides tools to let you to submit, run, and monitor groups of jobs
    - » ie. a “workflow”
  - → provides information on individual jobs as well as aggregate information
  - → tracks success and failures and allows you to modify and resubmit failed jobs
    - » ie. outlier jobs that need more time, RAM, disk, etc

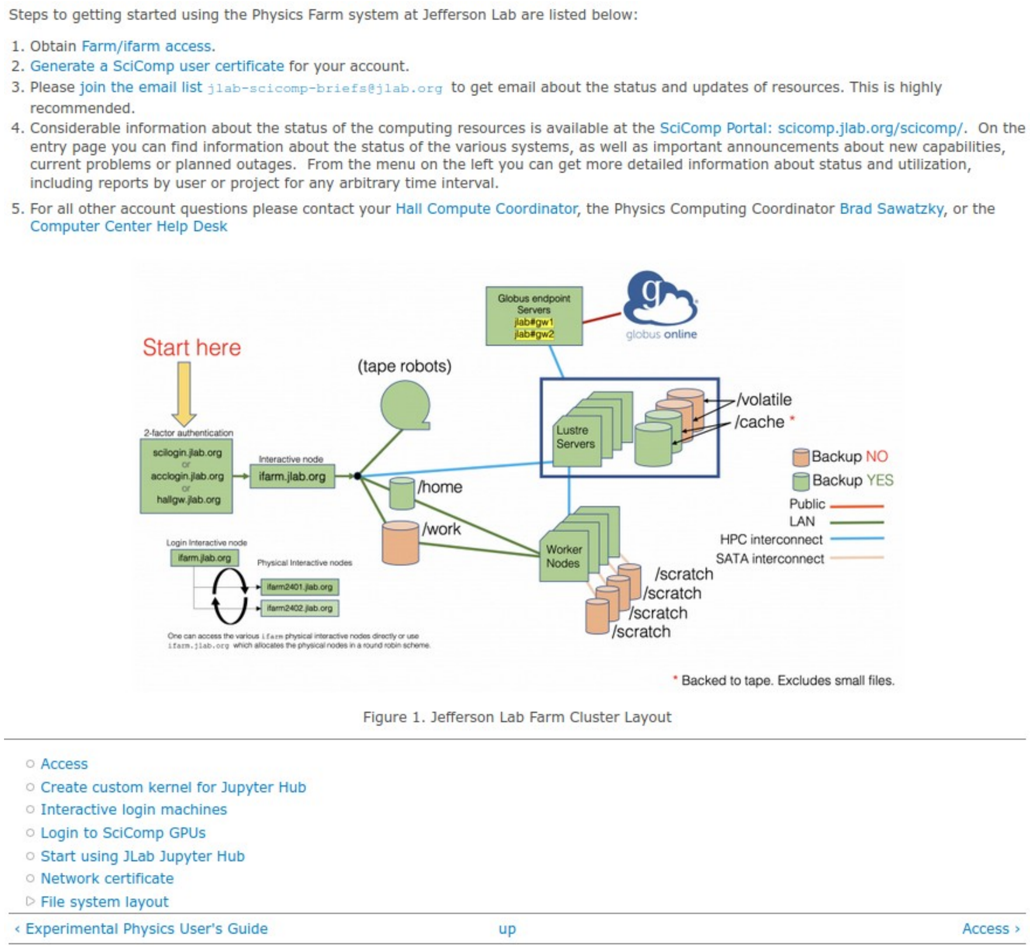
Simple Linux Utility For Resource Management

- Or Slurm, to save time…
  - → Open source compute cluster resource manager and job scheduler
  - → JLab Slurm Docs
    - » sinfo
    - » srun
    - » Useful env-vars
- Jobs run under a particular
  - → Slurm ‘Account’ (ie. group)
    - » Ask helpdesk@jlab.org or Compute Coordinator
  - → Slurm ‘Partition’
    - » usually ‘production’ or ‘priority’

- Cluster/Job Scheduling
  - → takes care of nitty-gritty of actually running jobs on a given compute node
  - → tracks and enforces resource usage and requirements (RAM, disk, time, etc)
  - → enforces a fair balance of allocated farm time across our user base (Division, experiment, Hall, etc)
    - » “Fairshare” allocations are managed by Compute Coordinators, SciComp and Physics Division

General Farm Access

- See “New users start here”
  - → At top of scicomp.jlab.org
  - → See this excellent walkthrough for BRIC by Cameron Clarke too!
- You will need (in general):
  - → JLab CUE account
  - → Get a 2-factor token (ssh access)
  - → Get access to Farm/ifarm
    - » talk to Project lead and/or Compute Coordinator
    - » learn where to put your files (unix group access)
    - » get attached to a slurm group
  - → SciComp user certificate



Steps to getting started using the Physics Farm system at Jefferson Lab are listed below:

1. Obtain Farm/ifarm access.
2. Generate a SciComp user certificate for your account.
3. Please join the email list `jlab-scicomp-briefs@jlab.org` to get email about the status and updates of resources. This is highly recommended.
4. Considerable information about the status of the computing resources is available at the SciComp Portal: scicomp.jlab.org/scicomp/. On the entry page you can find information about the status of the various systems, as well as important announcements about new capabilities, current problems or planned outages. From the menu on the left you can get more detailed information about status and utilization, including reports by user or project for any arbitrary time interval.
5. For all other account questions please contact your Hall Compute Coordinator, the Physics Computing Coordinator Brad Sawatzky, or the Computer Center Help Desk

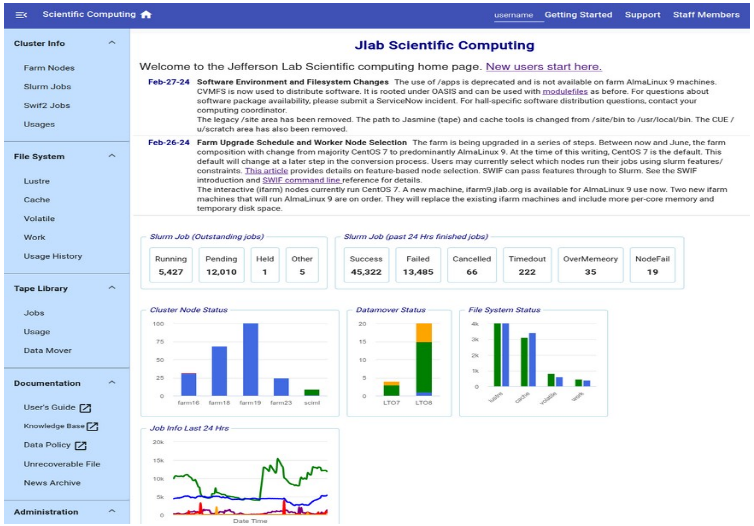
Figure 1. Jefferson Lab Farm Cluster Layout

- ○ Access
- ○ Create custom kernel for Jupyter Hub
- ○ Interactive login machines
- ○ Login to SciComp GPUs
- ○ Start using JLab Jupyter Hub
- ○ Network certificate
- ▷ File system layout

‹ Experimental Physics User's Guide | up | Access ›

Other Information Resources

- scicomp.jlab.org
  - → SciComp web page
- scicomp-briefs
  - → mailing list for JLab Scientific Computing



Jlab Scientific Computing

Welcome to the Jefferson Lab Scientific computing home page. New users start here.

Feb-27-24 **Software Environment and Filesystem Changes** The use of /apps is deprecated and is not available on farm AlmaLinux 9 machines. CVMFS is now used to distribute software. It is rooted under OASIS and can be used with modulefiles as before. For questions about software package availability, please submit a ServiceNow incident. For hall-specific software distribution questions, contact your computing coordinator.
The legacy /site area has been removed. The path to Jasmine (tape) and cache tools is changed from /site/bin to /usr/local/bin. The CUE / u/scratch area has also been removed.

Feb-26-24 **Farm Upgrade Schedule and Worker Node Selection** The farm is being upgraded in a series of steps. Between now and June, the farm composition with change from majority CentOS 7 to predominantly AlmaLinux 9. At the time of this writing, CentOS 7 is the default. This default will change at a later step in the conversion process. Users may currently select which nodes run their jobs using slurm features/ constraints. This article provides details on feature-based node selection. SWIF can pass features through to Slurm. See the SWIF introduction and SWIF command line reference for details.
The interactive (ifarm) nodes currently run CentOS 7. A new machine, ifarm9.jlab.org is available for AlmaLinux 9 use now. Two new ifarm machines that will run AlmaLinux 9 are on order. They will replace the existing ifarm machines and include more per-core memory and temporary disk space.

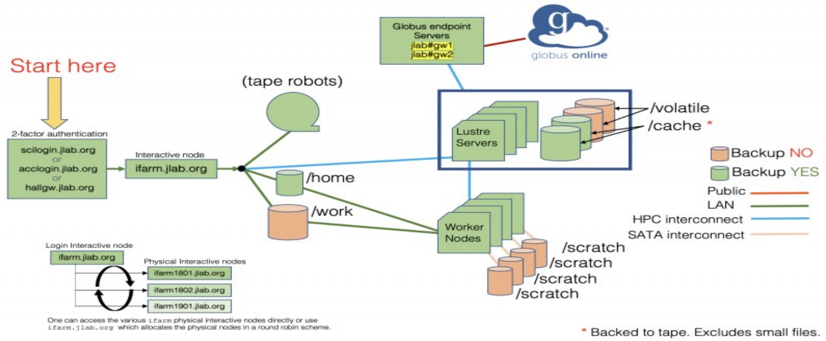
Slurm Job (Outstanding jobs)

Running	Pending	Held	Other
5,427	12,010	1	5

Slurm Job (past 24 Hrs finished jobs)

Success	Failed	Cancelled	Timedout	OverMemeory	NodeFail
45,322	13,485	66	222	35	19

- Documentation links
  - → Getting Started
  - → SciComp Knowledge Base
  - → CST User Portal
  - → JLab Helpdesk
    - » helpdesk@jlab.org
    - » Incident Request



One can access the various `ifarm` physical interactive nodes directly or use `ifarm.jlab.org` which allocates the physical nodes in a round robin scheme.

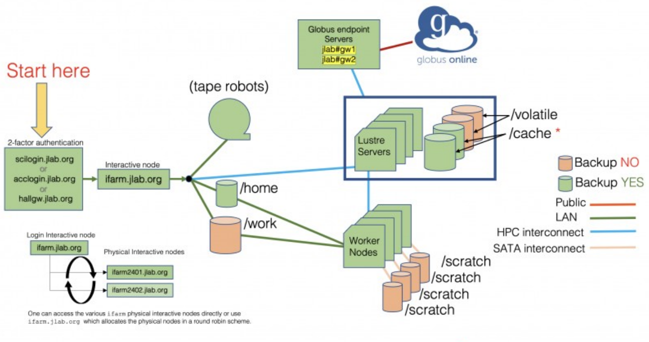
* Backed to tape. Excludes small files.

The Plan (TM)

1) Develop and test an example compute job

2) Develop and test a ‘steering script’ that you will use to
   - → setup your environment
   - → stage your code/exe,
   - → and run your code

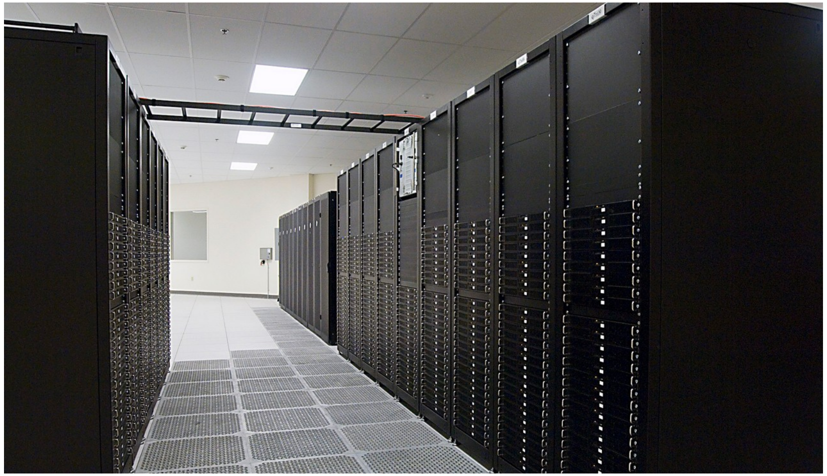
   as a non-interactive farm job



One can access the various `ifarm` physical interactive nodes directly or use `ifarm.jlab.org` which allocates the physical nodes in a round robin scheme.

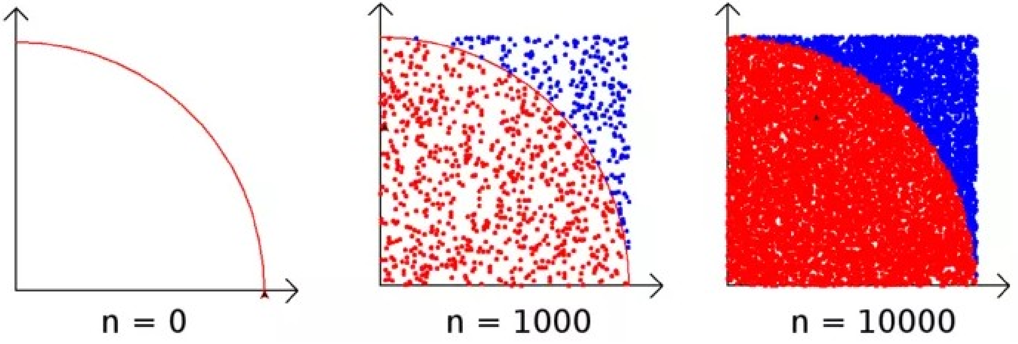
* Backed to tape. Excludes small files.

3) Submit a few example batch jobs to the Farm using SWIF2 using a ‘swif-submission.sh’ script
   - → monitor those jobs with some swif and slurm commands
   - → modify and re-run any failed jobs



The Plan (TM)

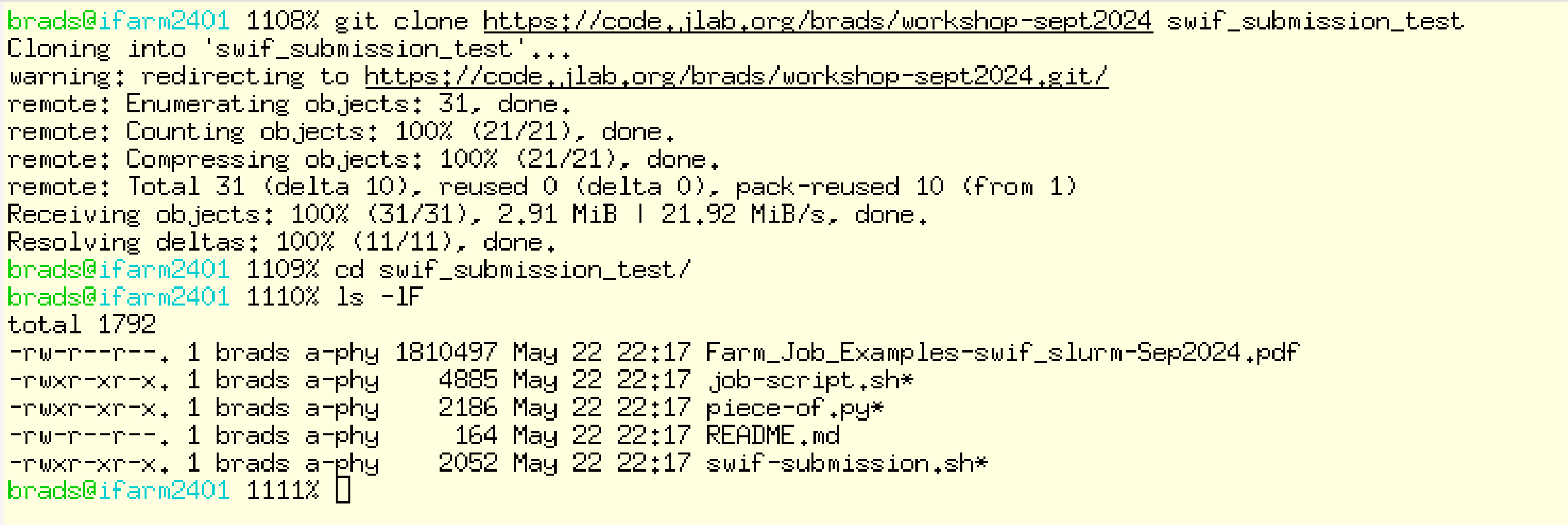
- We are going to compute pi using a simple (but not super efficient) Monte Carlo program
  - → Pick random points in a 1x1 square area and count the number that land within a radius of 1 from the origin
  - → Area of the square is 1x1
  - → Area of the ¼ circle is ¼ π 1^2
  - → So the ratio of the areas is ¼ π
  - → So lets randomly throw N_{sample} points at the square and see what fraction of them are also in the circle



- A_{circle} / A_{square} = ¼ π
  - → π = 4 x $hits_{circle} / hits_{square}$
  - → π = 4 x $hits_{circle} / N_{samples}$
- More samples
  - → better coverage
  - → better accuracy (but takes longer)

Grab the Example Code

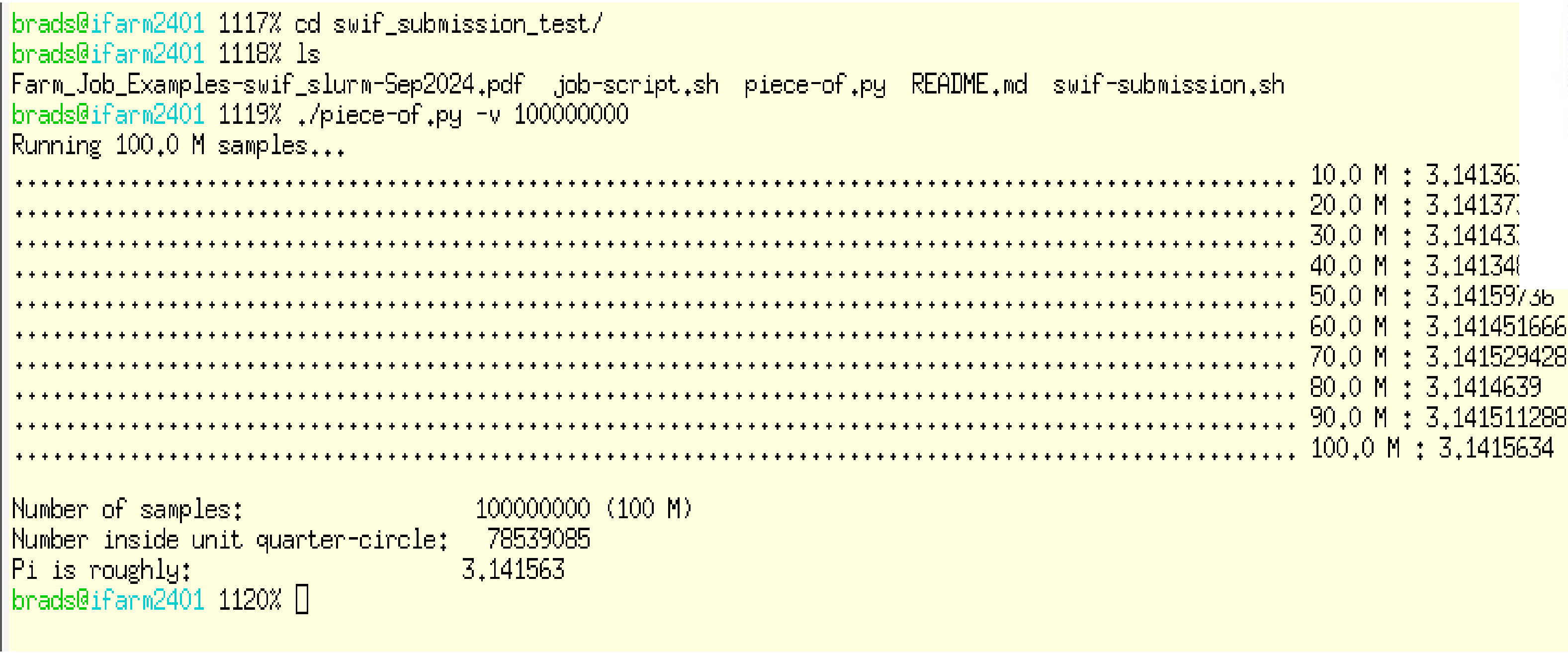
- ssh <you>@login.jlab.org
- ssh ifarm
- git clone https://code.jlab.org/brads/workshop-sept2024 swif_submission_test



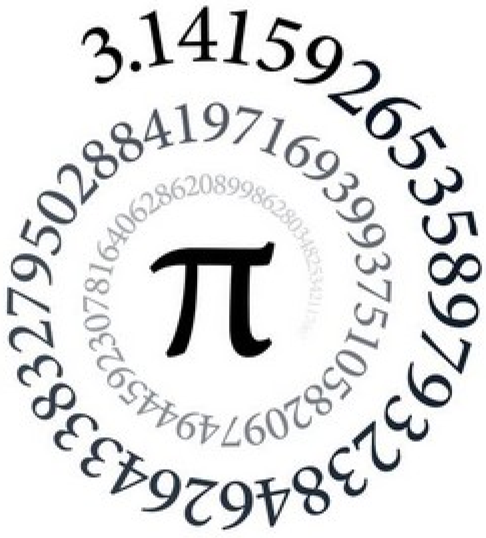
```
brads@ifarm2401 1108% git clone https://code.jlab.org/brads/workshop-sept2024 swif_submission_test
Cloning into 'swif_submission_test'...
warning: redirecting to https://code.jlab.org/brads/workshop-sept2024.git/
remote: Enumerating objects: 31, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 31 (delta 10), reused 0 (delta 0), pack-reused 10 (from 1)
Receiving objects: 100% (31/31), 2.91 MiB | 21.92 MiB/s, done.
Resolving deltas: 100% (11/11), done.
brads@ifarm2401 1109% cd swif_submission_test/
brads@ifarm2401 1110% ls -lF
total 1792
-rw-r--r--. 1 brads a-phy 1810497 May 22 22:17 Farm_Job_Examples-swif_slurm-Sep2024.pdf
-rwxr-xr-x. 1 brads a-phy    4885 May 22 22:17 job-script.sh*
-rwxr-xr-x. 1 brads a-phy    2186 May 22 22:17 piece-of.py*
-rw-r--r--. 1 brads a-phy     164 May 22 22:17 README.md
-rwxr-xr-x. 1 brads a-phy    2052 May 22 22:17 swif-submission.sh*
brads@ifarm2401 1111% 
```

Dev Process: Test your executable!

- Test your code on the ifarm
  - → piece-of.py
    - » this does the actual calculation

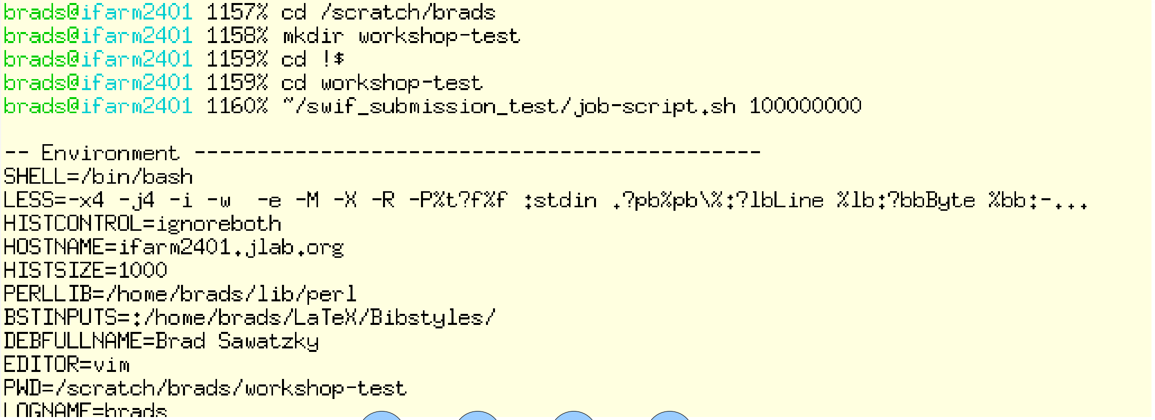


```
brads@ifarm2401 1117% cd swif_submission_test/
brads@ifarm2401 1118% ls
Farm_Job_Examples-swif_slurm-Sep2024.pdf  job-script.sh  piece-of.py  README.md  swif-submission.sh
brads@ifarm2401 1119% ./piece-of.py -v 100000000
Running 100.0 M samples...
.................................................................................................... 10.0 M : 3.14136
.................................................................................................... 20.0 M : 3.14137
.................................................................................................... 30.0 M : 3.14143
.................................................................................................... 40.0 M : 3.14134
.................................................................................................... 50.0 M : 3.14159736
.................................................................................................... 60.0 M : 3.1414516666666668
.................................................................................................... 70.0 M : 3.1415294285714284
.................................................................................................... 80.0 M : 3.1414639
.................................................................................................... 90.0 M : 3.1415112888888888
.................................................................................................... 100.0 M : 3.1415634

Number of samples:                   100000000 (100 M)
Number inside unit quarter-circle:    78539085
Pi is roughly:                      3.141563
brads@ifarm2401 1120% 
```

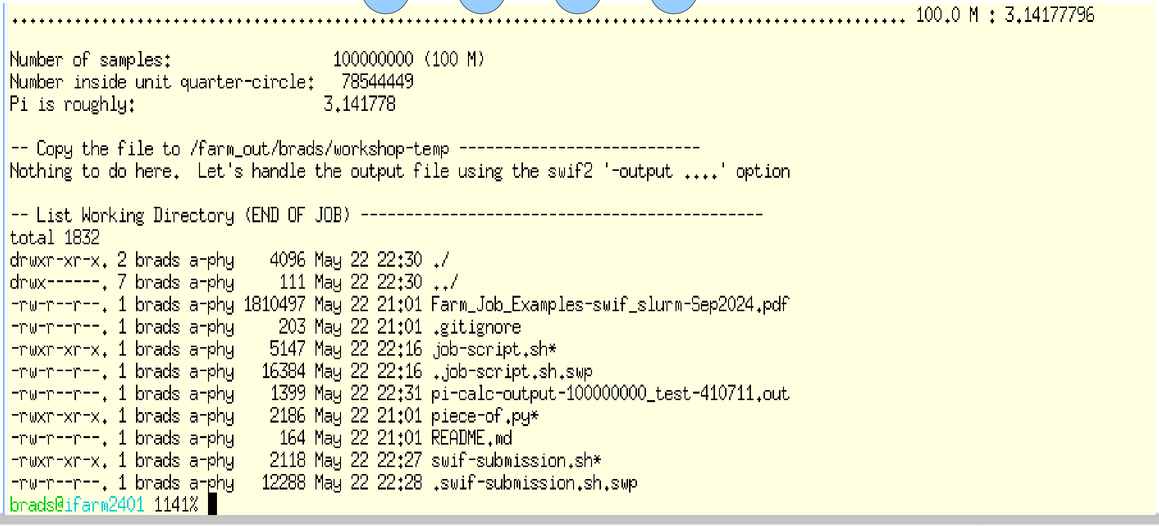


Dev Process: Test your job-script!

- Create and test a ‘job-script’ wrapper on ifarm
  - → job-script.sh <arguments>
  - → this ‘wrapper’ will be invoked for every farm job
  - → its purpose is to:
    - » *definitively* set-up your working environment
    - » load software dependencies
    - » sanity check directories, etc
    - » configure and run your executable for a single job in your workflow
      - – update run number
      - – update config files
      - – update event count
      - – etc…
- Test this script in a clean ‘working dir’ to replicate farm node environment
  - → we will use /scratch/$USER/workshop-test/



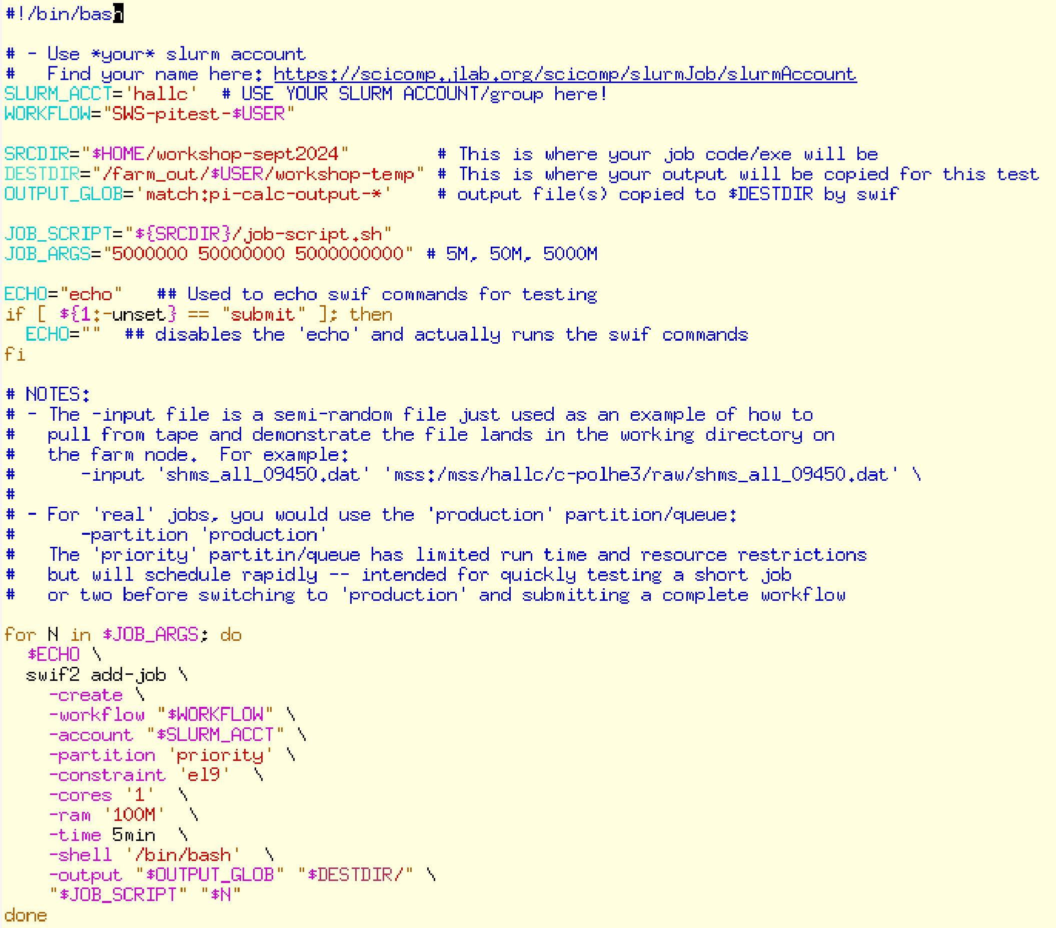
```
brads@ifarm2401 1157% cd /scratch/brads
brads@ifarm2401 1158% mkdir workshop-test
brads@ifarm2401 1159% cd !$
brads@ifarm2401 1159% cd workshop-test
brads@ifarm2401 1160% ~/swif_submission_test/job-script.sh 100000000

-- Environment ------------------------------------------
SHELL=/bin/bash
LESS=-x4 -j4 -i -w  -e -M -X -R -P%t?f%f :stdin .?pb%pb\%:?lbLine %lb:?bbByte %bb:-...
HISTCONTROL=ignoreboth
HOSTNAME=ifarm2401.jlab.org
HISTSIZE=1000
PERLLIB=/home/brads/lib/perl
BSTINPUTS=:/home/brads/LaTeX/Bibstyles/
DEBFULLNAME=Brad Sawatzky
EDITOR=vim
PWD=/scratch/brads/workshop-test
LOGNAME=brads
```



```
............................................................................................... 100.0 M : 3.14177796

Number of samples:                  100000000 (100 M)
Number inside unit quarter-circle:   78544449
Pi is roughly:                      3.141778

-- Copy the file to /farm_out/brads/workshop-temp --------------------------
Nothing to do here.  Let's handle the output file using the swif2 '-output ....' option

-- List Working Directory (END OF JOB) ------------------------------------------
total 1832
drwxr-xr-x. 2 brads a-phy    4096 May 22 22:30 ./
drwx------. 7 brads a-phy     111 May 22 22:30 ../
-rw-r--r--. 1 brads a-phy 1810497 May 22 21:01 Farm_Job_Examples-swif_slurm-Sep2024.pdf
-rw-r--r--. 1 brads a-phy     203 May 22 21:01 .gitignore
-rwxr-xr-x. 1 brads a-phy    5147 May 22 22:16 job-script.sh*
-rw-r--r--. 1 brads a-phy   16384 May 22 22:16 .job-script.sh.swp
-rw-r--r--. 1 brads a-phy    1399 May 22 22:31 pi-calc-output-100000000_test-410711.out
-rwxr-xr-x. 1 brads a-phy    2186 May 22 21:01 piece-of.py*
-rw-r--r--. 1 brads a-phy     164 May 22 21:01 README.md
-rwxr-xr-x. 1 brads a-phy    2118 May 22 22:27 swif-submission.sh*
-rw-r--r--. 1 brads a-phy   12288 May 22 22:28 .swif-submission.sh.swp
brads@ifarm2401 1141%
```

Dev Process: Test your ‘swif submission’ script

- Develop and test a script that can generate the ‘swif2 add-job …’ command for each job in your workflow
    - » swif-submission.sh
    - » Update **SLURM_ACCT**
  - → This can/will loop over runs / run_numbers (from directory or file), etc and create a swif job submission for each case.
      - – Declare (tape) -outputs
      - – Declare (tape) -inputs
    - » **NOTE**: mss:// (tape) files will be copied to working dir!
      - – Access them as *./<filename>* in your analysis script *not* */cache/…./<filename>*
  - → Also sets ‘defaults’ like:
    - » slurm partition/queue
    - » slurm ‘account’
    - » RAM, disk space, CPU needs



```
#!/bin/bash

# - Use *your* slurm account
#   Find your name here: https://scicomp.jlab.org/scicomp/slurmJob/slurmAccount
SLURM_ACCT='hallc'  # USE YOUR SLURM ACCOUNT/group here!
WORKFLOW="SWS-pitest-$USER"

SRCDIR="$HOME/workshop-sept2024"         # This is where your job code/exe will be
DESTDIR="/farm_out/$USER/workshop-temp" # This is where your output will be copied for this test
OUTPUT_GLOB='match:pi-calc-output-*'     # output file(s) copied to $DESTDIR by swif

JOB_SCRIPT="${SRCDIR}/job-script.sh"
JOB_ARGS="5000000 50000000 5000000000" # 5M, 50M, 5000M

ECHO="echo"   ## Used to echo swif commands for testing
if [ ${1:-unset} == "submit" ]; then
  ECHO=""   ## disables the 'echo' and actually runs the swif commands
fi

# NOTES:
# - The -input file is a semi-random file just used as an example of how to
#   pull from tape and demonstrate the file lands in the working directory on
#   the farm node.  For example:
#      -input 'shms_all_09450.dat' 'mss:/mss/hallc/c-polhe3/raw/shms_all_09450.dat' \
#
# - For 'real' jobs, you would use the 'production' partition/queue:
#      -partition 'production'
#   The 'priority' partitin/queue has limited run time and resource restrictions
#   but will schedule rapidly -- intended for quickly testing a short job
#   or two before switching to 'production' and submitting a complete workflow

for N in $JOB_ARGS; do
  $ECHO \
  swif2 add-job \
    -create \
    -workflow "$WORKFLOW" \
    -account "$SLURM_ACCT" \
    -partition 'priority' \
    -constraint 'el9'  \
    -cores '1'  \
    -ram '100M'  \
    -time 5min  \
    -shell '/bin/bash'  \
    -output "$OUTPUT_GLOB" "$DESTDIR/" \
    "$JOB_SCRIPT" "$N"
done
```

Dev Process: Test your ‘swif add-job …’ script

- Comments / Notes
  -cores <N>
  - → know what your job can actually use

  -ram <size>
  - → shoot for < 2GB/core if you can

  -disk <size>
  - → disk space needed on the farm node itself
  - → If set, then <size> == size for both
    - » ‘-input’ files, *AND*
    - » ‘-output’ files on the local disk
  - → If unset, then Swif will use
    - » SizeOf(input files) * 1.1

  -disk-scratch <size>
  - → <size> has to account for only new output files on local disk (space for input files is automatically taken into account)
    - » *This is a new, better, option!*

- “Gotchas” to remember:
  - → Do *not* modify your executable or job-script.sh until your workflow is fully completed
    - » no recompiling; no ‘tweaks’
    - » (It could change what runs on the farm mid-workflow!)
  - → Remember many jobs will run at the exact same time!
    - » If you run your job from within a network mounted space (ie. /group, /work) then you *must* ensure input and output files have unique names, ‘temp’ files don’t step on each other, etc.
  - → Access ‘-input mss://….’ files from job’s working directory (*not from /cache*)

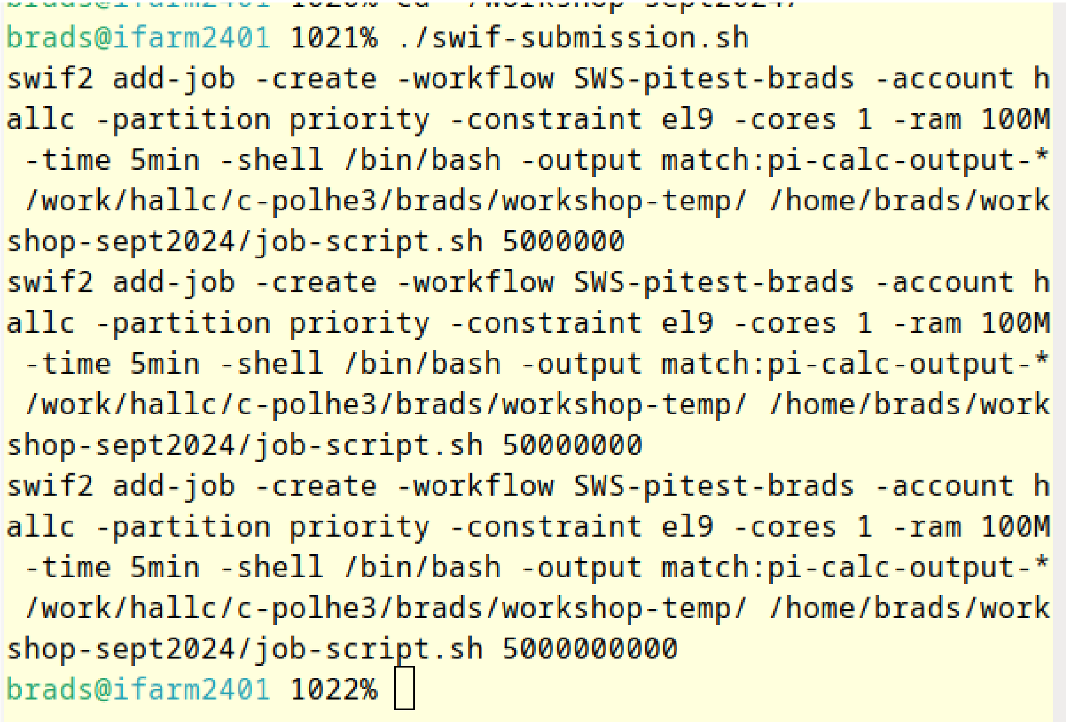
Submit your Jobs to SWIF

- Check lines by running

  swif-submission.sh

  → review the ‘debug’ output

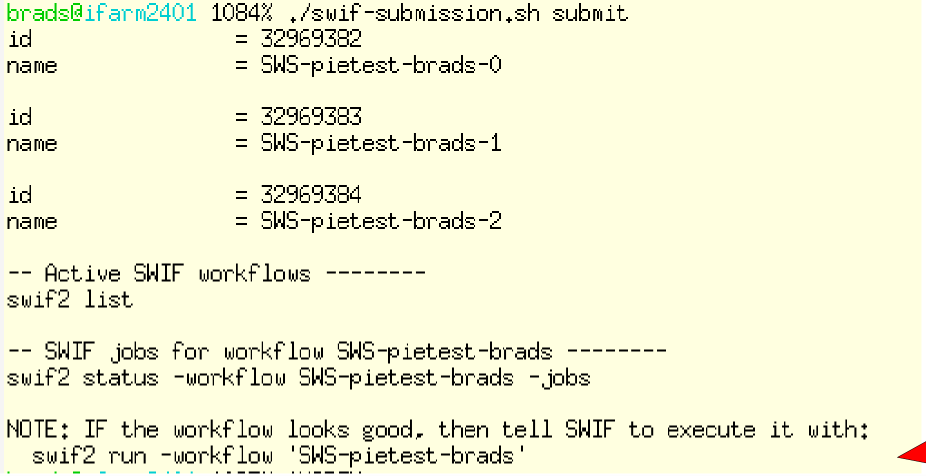
  ~ Only Then ~



```
brads@ifarm2401 1021% ./swif-submission.sh
swif2 add-job -create -workflow SWS-pitest-brads -account h
allc -partition priority -constraint el9 -cores 1 -ram 100M
 -time 5min -shell /bin/bash -output match:pi-calc-output-*
 /work/hallc/c-polhe3/brads/workshop-temp/ /home/brads/work
shop-sept2024/job-script.sh 5000000
swif2 add-job -create -workflow SWS-pitest-brads -account h
allc -partition priority -constraint el9 -cores 1 -ram 100M
 -time 5min -shell /bin/bash -output match:pi-calc-output-*
 /work/hallc/c-polhe3/brads/workshop-temp/ /home/brads/work
shop-sept2024/job-script.sh 50000000
swif2 add-job -create -workflow SWS-pitest-brads -account h
allc -partition priority -constraint el9 -cores 1 -ram 100M
 -time 5min -shell /bin/bash -output match:pi-calc-output-*
 /work/hallc/c-polhe3/brads/workshop-temp/ /home/brads/work
shop-sept2024/job-script.sh 5000000000
brads@ifarm2401 1022% 
```

- Run

  swif-submission.sh submit



```
brads@ifarm2401 1084% ./swif-submission.sh submit
id                     = 32969382
name                   = SWS-pietest-brads-0

id                     = 32969383
name                   = SWS-pietest-brads-1

id                     = 32969384
name                   = SWS-pietest-brads-2

-- Active SWIF workflows --------
swif2 list

-- SWIF jobs for workflow SWS-pietest-brads --------
swif2 status -workflow SWS-pietest-brads -jobs

NOTE: IF the workflow looks good, then tell SWIF to execute it with:
  swif2 run -workflow 'SWS-pietest-brads'
```

Don’t forget this step!!

Useful SWIF2 Commands

- swif2 list
  - → see all of your workflows
- swif2 status -workflow $WORKFLOW
  - → see summary details for $WORKFLOW
- swif2 status -workflow "$WORKFLOW" -jobs | less
  - → see a list of jobs associated with $WORKFLOW (can be big list!)
  - → find SWIF job_ids, (*etc*)
- watch -d -n30 -- “swif2 status $WORKFLOW”
  - → if you’re impatient like me…
- swif2 show-job -jid <swif_job_id>
  - → get info on a particular SWIF2 job
    - » job_status, slurm_id, etc
- swif cancel “$WORKFLOW” -delete
  - → Stop and delete a workflow
  - → NOTE: Deleting a workflow frees the workflow name for future (re-)use, BUT it also means you can not query/debug the old jobs.

Useful Slurm Commands

NOTE: Slurm commands take slurm job-ids, not SWIF job-ids!

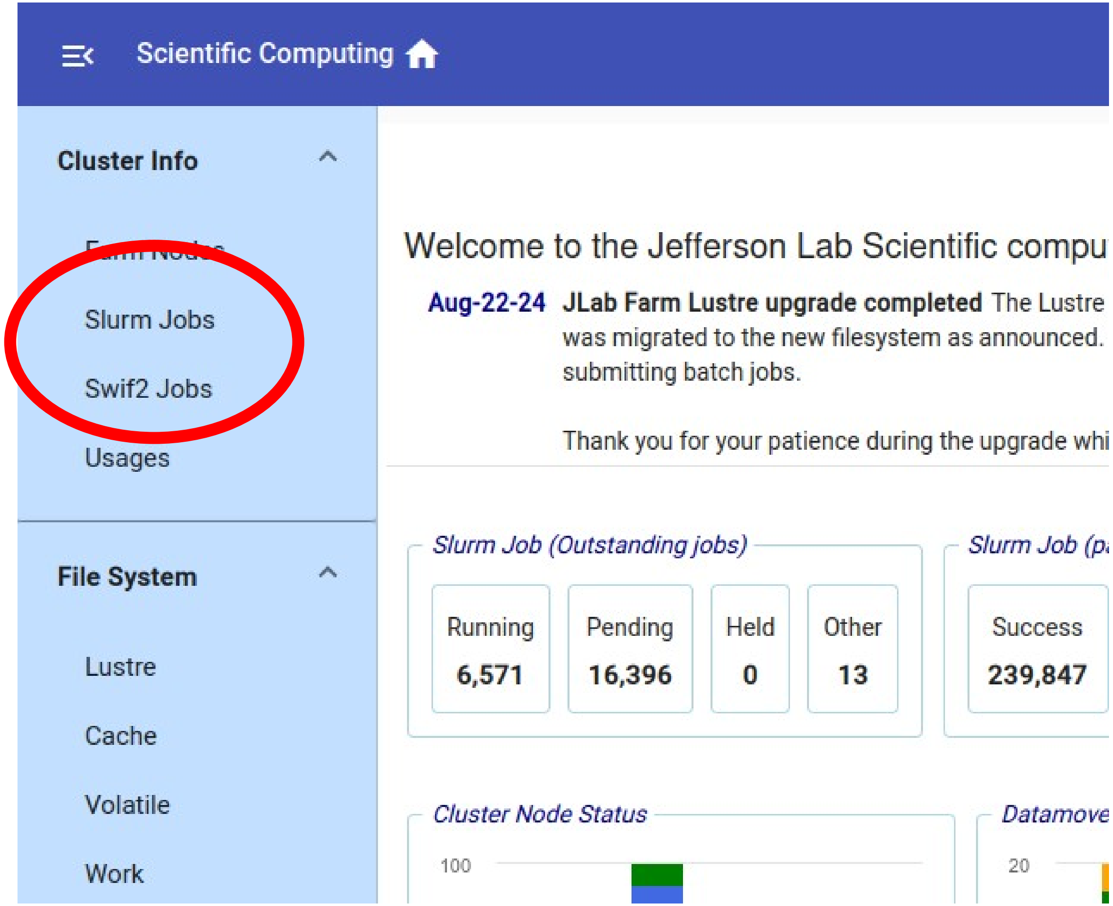
- sacct
  → queries *all the things* (ie. slurm DB access)
  → https://slurm.schedmd.com/sacct.html
- sacct --long -j <slurm_id>
- sacct -- env-vars -j <slurm_id>
  → env vars set during a slurm job
  → SWIF_JOB_ID, SWIF_JOB_ATTEMPT_ID ← map back to SWIF job!
- seff <slurm_id>
  → wrapper around sacct that reports job efficiency numbers
  → RAM, CPU usage

Don’t forget about /farm_out/$USER/swif/...

- cd /farm_out/$USER/swif/$WORKFLOW

- ls
  - → Should see a list of stdout and stderr for jobs that have run (or are currently running!)
- less <job-name-id>.out
  - → shows everything you dumped in your job-script.sh and what your executable wrote to stdout
- less <job-name-id>.err
  - → shows text written to stderr (usually ‘errors’) and/or debug info
  - → in this test you should see the software version output from the job-script.sh

Web-based SWIF/Slurm Reporting



Welcome to the Jefferson Lab Scientific compu

Aug-22-24 **JLab Farm Lustre upgrade completed** The Lustre was migrated to the new filesystem as announced. submitting batch jobs.

Thank you for your patience during the upgrade whi

Running	Pending	Held	Other	Success
6,571	16,396	0	13	239,847

- Most of the command-line info provided by SWIF and slurm can also be found on scicomp.jlab.org web page
  - → See Swif2 Jobs
  - → See Slurm Jobs

Web-based SWIF2 Reporting

- SWIF2 Jobs
  - → Workflow Summary
    - » aggregate job info
    - » failed jobs at bottom
  - → Active (Dormant) Workflows
    - » What is running/waiting
    - » Filter to find your workflow and select it for job list
      - – click on job for job info

Workflow Summary

Choose a user: brads

Choose a workflow: SWS-pitest-brads

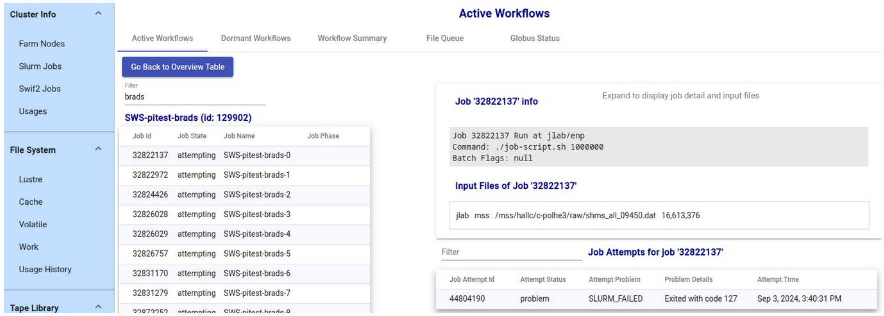
Workflow Summary

Workflow ID: 129902
Workflow Name: SWS-pitest-brads
Workflow Site: jlab/enp
Max Concurrent: 500
Undispatched: 0
Output Generated: 0 MB
Create Time: Sep 3, 2024, 2:01:18 PM
Last Update Time: Sep 4, 2024, 12:23:11 AM

Succeeded Job Usage Distribution

Problem Job Attempts

swif Job Id	Slurm Job Id	Attempt Id	Job Name	Problem Code	Node	Resolution	Complete Time
32904344	27238167	44827150	SWS-pitest-brads-20	SLURM_TIMEOUT	farm180245	modify	Sep 3, 2024, 11:56:42 PM
32904344	27238907	44827150	SWS-pitest-brads-20	SLURM_TIMEOUT	farm180243		Sep 4, 2024, 12:23:11 AM



Active Workflows

Filter: brads

SWS-pitest-brads (id: 129902)

Job Id	Job State	Job Name	Job Phase
32822137	attempting	SWS-pitest-brads-0	
32822972	attempting	SWS-pitest-brads-1	
32824426	attempting	SWS-pitest-brads-2	
32826028	attempting	SWS-pitest-brads-3	
32826029	attempting	SWS-pitest-brads-4	
32826757	attempting	SWS-pitest-brads-5	
32831170	attempting	SWS-pitest-brads-6	
32831279	attempting	SWS-pitest-brads-7	

Job '32822137' info

Expand to display job detail and input files

```
Job 32822137 Run at jlab/enp
Command: ./job-script.sh 1000000
Batch Flags: null
```

Input Files of Job '32822137'

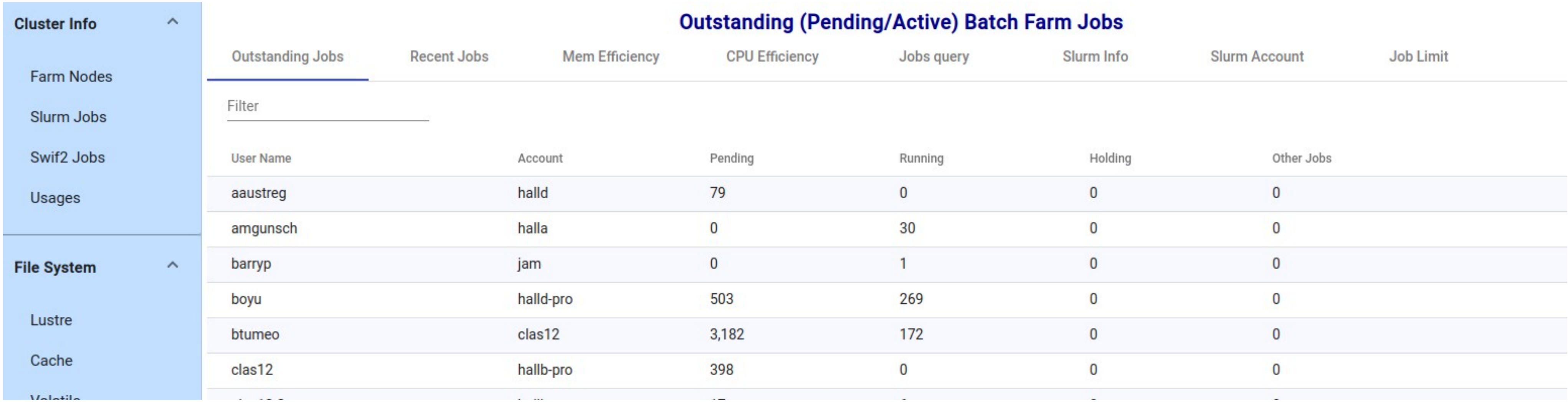
jlab mss /mss/hallc/c-polhe3/raw/shms_all_09450.dat 16,613,376

Filter

Job Attempts for job '32822137'

Job Attempt Id	Attempt Status	Attempt Problem	Problem Details	Attempt Time
44804190	problem	SLURM_FAILED	Exited with code 127	Sep 3, 2024, 3:40:31 PM

Web-based Slurm Reporting



Cluster Info

Farm Nodes

Slurm Jobs

Swif2 Jobs

Usages

File System

Lustre

Cache

Outstanding (Pending/Active) Batch Farm Jobs

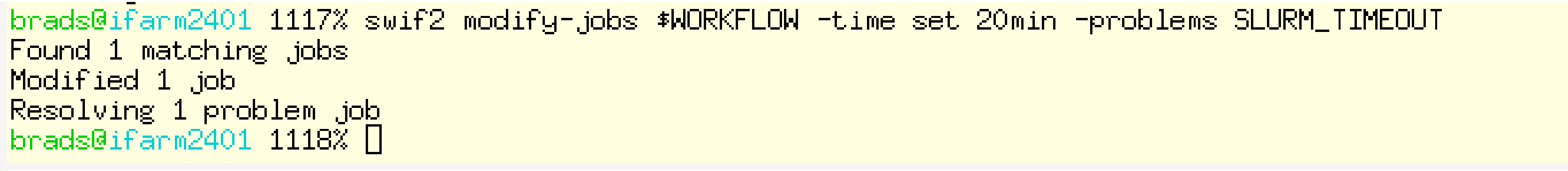
User Name	Account	Pending	Running	Holding	Other Jobs
aaustreg	halld	79	0	0	0
amgunsch	halla	0	30	0	0
barryp	jam	0	1	0	0
boyu	halld-pro	503	269	0	0
btumeo	clas12	3,182	172	0	0
clas12	hallb-pro	398	0	0	0

- Inefficient Job information
  - → Mem / CPU efficiency tabs
  - → You will get an email if you land on this list
- Jobs Query
  - → look up slurm job information
- “Slurm info” tab
  - → Find ‘feature’ tags for use with job ‘constraints’
  - → See a list of partitions (job queues)
  - → Slurm Account
    - » Find your name; use the correct slurm account for your job!

Fix and Resubmit failed job(s) with Swif2

- If we get this far and you are monitoring your jobs, you will find that 1 of the 3 jobs we submitted failed:
  - → SLURM_TIMEOUT error
  - → The 500M event job took longer than the 5min we declared in swif

- Fix this and resubmit:
  - → swif2 modify-jobs
$WORKFLOW
-time 20min
-problems SLURM_TIMEOUT



```
brads@ifarm2401 1117% swif2 modify-jobs $WORKFLOW -time set 20min -problems SLURM_TIMEOUT
Found 1 matching jobs
Modified 1 job
Resolving 1 problem job
brads@ifarm2401 1118% 
```

Any Wisdom/Hints for Others?

- What issues have you run into?

  → Raise them here so others can learn from your experience!

- Any quirks that need better documentation?

- Let me know so we have a shot at fixing the problems/issues!

New(’ish) feature:

- ‘swif2 notify’ now allows for email notification if there are no more jobs capable of running
  -
    ```
    swif2 notify <workflow> \
    -email you@jlab.org \
    -when done
    ```
  -
    ```
    ‘swif2 notify <workflow> \
    -email you@jlab.org \
    -when stalled
    ```

  → I’d run both commands so you are notified when your workflow has stopped for either reason