CLAS12 Computing and Data Processing @ JLab

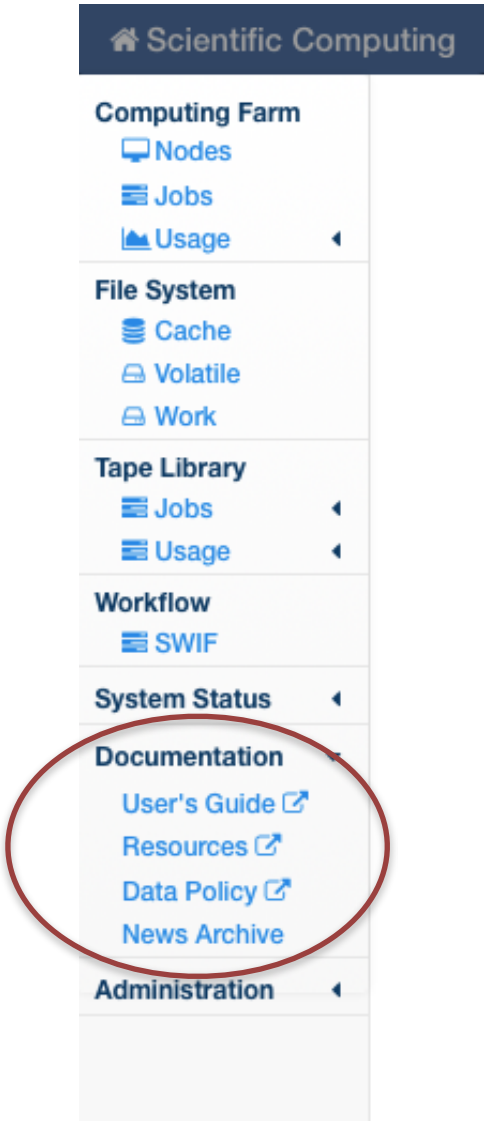
N. Baltzell

June 18, 2019

CLAS Collaboration Meeting

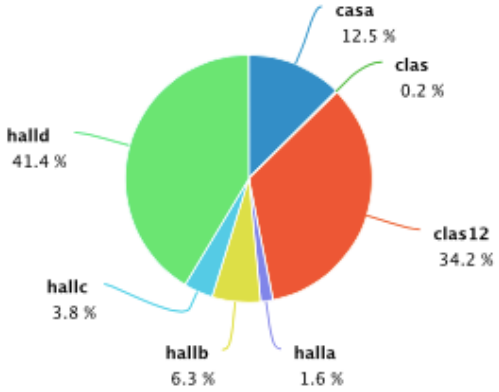
Non-CLAS12 General Reminders

- See Bryan's scicomp talk from earlier today
- SLURM reminders
  - you now have to setup your environment in your farm job, it will no longer source your shell's $HOME directory's login files automatically
  - if you don't specify stdout/stderr job tags, logs now go to
    - `/farm_out/$USER` instead of `$HOME/.farm_out`
    - this avoids quota issues in your `$HOME`
- Pay attention to emails from jlab-scicomp-briefs@jlab.org (click link for archives)
  - everyone with a JLab computing account *should* receive them
  - planned outages, system changes, etc
- Learn and use scicomp's documentation and monitoring web pages for batch jobs, disk quotas, tape access
  - http://scicomp.jlab.org



CLAS12 Processing @ JLab

- CLAS priority accounts (batch farm)
  - For large scale data-processing for CLAS run-groups (not simulations/analysis)
    - these accounts are managed based on resource needs and by agreement with the run groups, Hall computing and software coordinators
  - Currently CLAS's priority is **45%** of batch farm (personal user accounts get much less)
    - we can monitor that at scicomp.jlab.org
    - requires monitoring, learning, studying, to understand the best throughput and how to utilize and optimize our 45%, and this is an ongoing project, e.g. full-node jobs vs 8/16/24-core jobs



- User Accounts
  - you need to be a member of group=clas12 for access to interactive and batch farms at JLab, and clas12 disk space ..... email me if you need it and aren't!
  - but getting the original computer account starts with an application at http://cc.jlab.org

CLAS12 Storage @ JLab

The large fileservers we commonly use:

- `/work/clas12`
  - **175 TB**
  - *manually* managed
  - more traditional fileserver
    - good for lots of small files, small I/O operations
    - *not good for large data and large-scale I/O access (e.g. many simultaneous jobs reading lots of GB from the batch farm)*

- `/volatile/clas12`
  - **50 TB High Quota**
  - **25 TB Guaranteed**
  - *automatically* managed
    - autodeletion possible (certain) if we go over the Guarantee (High Quota)
  - Lustre, distributed system
    - *good for large data I/O from the batch farm*
  - there is some talk with scicomp and the other halls about moving to "hard"-quotas on /volatile, which would make the autodeletion behavior more predictable, and relax the desire to use /work?
  - ***scicomp is ~doubling Lustre soon, we can think about how to best distribute that between /cache and /volatile***

How do we organize our filerserver usage? Let's focus on the usual two ....

- /work/clas12 and /volatile/clas12
  - we have some obvious subdirs:
    - run group names (e.g. rg-a, rg-b)
    - user names (e.g. avakian, zwzhao)
    - detector names (e.g. rich, ltcc)
  - and then some different organization ideas (richcap, jnp, BST_alignment)
    - which makes finding out who owns/needs what a bit cumbersome
  - this is presumably years of buildup, ***and I think we might should do some big cleanup***
    - this would also enable moving towards finer-grained quotas, e.g. per run-group, separately from users easier, if we want to go that route
- Also, these disks are not meant for permanent or backed-up data storage, that's what tape is for!
- ***And some cleanup is always needed!***

CLAS12 Software Installs @ JLab

- Previously we haven't done a great job of supporting shared installations of *all* CLAS12 software
  - some of that might come from more use of personal computers, or the java mentality of compile-once, run-everywhere, or IDE usage, etc ....
  - but that doesn't mean you should have to download/install anything just to run at JLab!
  - we'd like to move towards a more standard environment for CLAS12
- A standard installation scheme will now be maintained on the /group disk
  - To use it, you need to use "modulefiles" environment setup, but it's easy:
  - Source one file:
    - `source /group/clas12/packages/setup.csh` (or `setup.sh` if you use bash shell)
  - Then use the module command to see what's available, load them into your environment, see screenshot below.
  - There's still some missing items, mainly the C++ HIPO libraries and Clas12Tool, to be added ASAP!
    - *I just remembered before this talk and I guess none of the beta-testers apparently used those, sorry:(*

```
ifarm1401> source /group/clas12/packages/setup.csh
ifarm1401> module avail

--------------------------- /group/clas12/packages/local/etc/modulefiles ---------------------------
ccdb/1.07       clas12/pro      coatjava/6b.1.1 gemc/4.3.0      jaw/0.9         lz4/1.7.6
ced/1.006e      coatjava/5.9.0  coatjava/6b.2.0 gemc/4.3.1      jdk/11.0.2      maven/3.5.0
clas12/dev      coatjava/6b.0.0 evio/5.1        groovy/2.4.9    jdk/1.8.0_31    rcdb/1.0
ifarm1401> module load coatjava/6b.2.0
>>>
>>>  Environment set for:
>>>      coatjava-6b.2.0
>>>      clara-4.3.10
>>>      grapes-2.1
>>>
ifarm1401> which hipo-utils
/group/clas12/packages/coatjava/6b.2.0/bin/hipo-utils
ifarm1401> echo $CLARA_HOME
/group/clas12/packages/clara/4.3.10_6b.2.0
ifarm1401>
```

Expect a demo in the workshop this week ...

What is "Swif"?

- It's a JLab product: https://scicomp.jlab.org/docs/swif
  - the "scientific workflow indefatigable factotum"

- Swif allows to interact with JLab batch farm *programmatically*
  - Write jobs and retrieve job status in a standard, structured format (xml/json)
  - Retry failed jobs
  - Modify jobs, e.g.
    - increase resource requirements
    - abandon jobs
  - ***Organize jobs into a workflow***
    - group jobs into serial "phases"
    - explicit job-job dependencies, "precedents"

- For some/many objectives you might be right to not care, e.g.
  - your failed jobs can be ignored and/or just run another independent job
  - your jobs have no inter-dependencies
  - you'd rather spend time manually monitoring your jobs, doing bookkeeping, independent error/integrity checking, writing and monitoring cleanup scripts

- *Meanwhile, Swif is also to be the JLab conduit to offsite resources (e.g. NERSC, OSG, etc)*

CLAS12 Data Processing for Run Groups

We're leveraging Swif workflows for managing large-scale data processing

- For decoding
  - settled on a 3-stage, Rolling workflow as one option
    - essentially 3 three-phase workflows interleaved, to give some reasonable optimization between tape usage and disk usage, including reliability considerations on disk space, and keeping runs from getting mixed up on tape
    - decode, merge, move+delete
  - file integrity checks during the jobs, with status reporting to batch system
  - utilize Swif's job tags (e.g. output directory, run/file numbers, coatjava version) for later automated bookkeeping, with named jobs and log files to facilitate debugging if necessary
  - automatically retry jobs due to system failures and adjusts job resource reqs if necessary

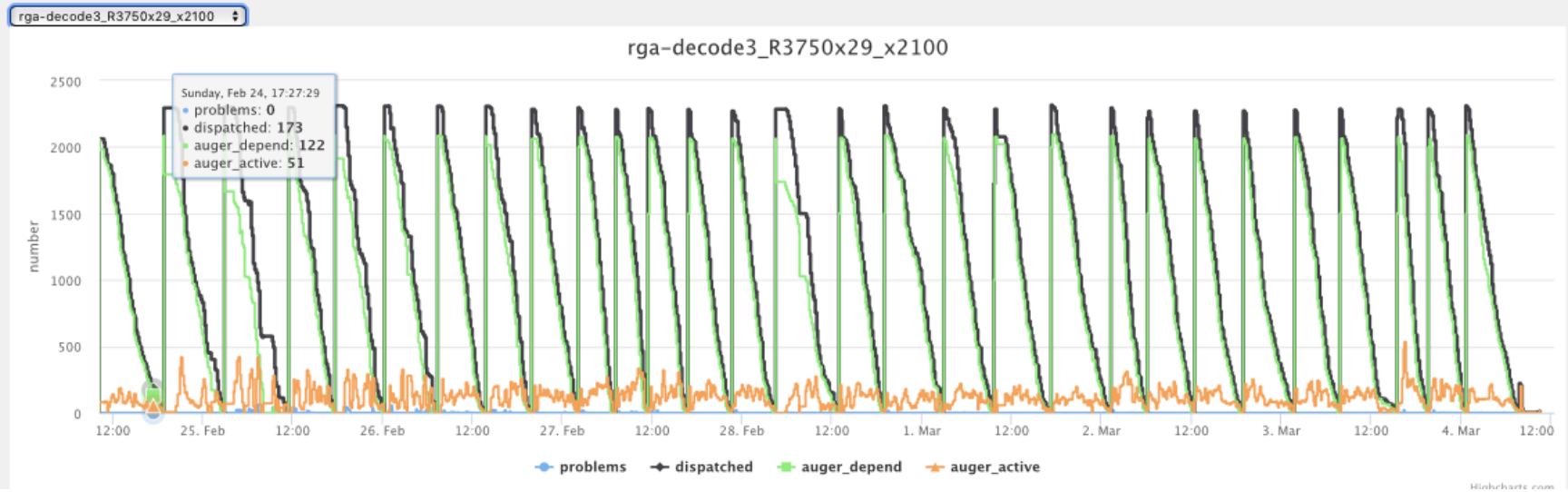
- The software currently lives here: https://github.com/baltzell/clas12-workflow
  - written to be somewhat generic to accommodate more tasks

- ***Incorporating rest of CLAS12 data processing chain in progress ...***
  - manage single-threaded decoding dependencies into multi-threaded CLARA jobs automatically
  - then tack standard calibration filtering onto the workflow too
  - goal to use one standard, easy interface, no chef-scripting, no file list generation, no extra scripts/filelists lying around, for all CLAS12 run-groups

CLAS12 Decoding Workflow Example

Pushes to clas12mon for long-term monitoring, via standard Swif JSON format, and a cronjob that cleans up and issues automatic retries for failed jobs, for a largely hands-free operation. We processed now 600K jobs for RG-A with only a handful of system glitches (fixed via CCPR), and only 6 corrupt files, but otherwise no human intervention ... plus what the RG-A/B chefs have been doing ...

workflow_name	jobs	succeeded	success	attempts	phase	dispatched	depend	active	update_date	current_date
rga-decode9_R3135x178_x1300	62672	57482	91.7%	74868	161	83	83	0	Jun 18,2019 16:38	Jun 18,2019 16:45
rga-decode8_R3191x65_x1300	65648	65642	100.0%	88013	67	0	0	0	May 11,2019 11:03	May 11,2019 15:44
rga-decode7_R3249x43_x2100	67203	67200	100.0%	72843	45	0	0	0	Apr 21,2019 02:28	Apr 21,2019 10:40
rga-decode6_R3466x34_x2100	67597	67597	100.0%	83442	36	0	0	0	Apr 10,2019 17:35	Apr 10,2019 17:40
rga-decode5_R3355x30_x2100	65162	65162	100.0%	68044	32	0	0	0	Mar 25,2019 09:16	Mar 25,2019 09:20
rga-decode4_R3501x30_x2100	67620	67620	100.0%	68232	32	0	0	0	Mar 13,2019 14:06	Mar 13,2019 14:10
rga-decode3_R3750x29_x2100	66583	66583	100.0%	67803	31	0	0	0	Mar 4,2019 10:32	Mar 4,2019 10:40
rga-decode2_R3464x28_x1300	66415	66415	100.0%	66998	58	0	0	0	Feb 24,2019 04:54	Feb 24,2019 05:00
rga-decode1_R3432x25_x1300	64076	64074	100.0%	79290	52	0	0	0	Feb 11,2019 15:03	Feb 11,2019 17:42



Highlights

- Promoting CLAS12 awareness about our computing resources at JLab
  - For example:
    - Anticipate some push to cleanup /work/clas12 and /volatile/clas12 and enforce a more standard top-level directory structure
      - we're thinking about a finer-grained quota (e.g. per run-group).
      - what mailing list should these types of updates go to?
  - Our current usage of /work is not ideal
    - work is better for small files, software, small I/O, and not lots of batch farm jobs with large data.
    - this will be somewhat alleviated by the new Lustre fileserver doubling size, if we move more to /volatile (or /cache)

- Standard builds of all CLAS12 software on JLab computers will be maintained for everyone to use
  - already existed for gemc, now for the rest of offline software, unification in the future

- The existing decoding workflows for CLAS12, originally for Spring 2018, are proven to be very reliable, leveraging Swif to achieve ultimately 100% hands-free success rate.
  - RG-B (and now RG-A) has been using the same workflow tools and given good feedback
  - We'll use this experience for the future, and are currently incorporating CLARA (and then filtering/trains) as part of the standard workflow for CLAS12 rungroup data processing