CLAS12 Computing Environment, Resources, and Data Processing @ JLab

N. Baltzell

March 24, 2020

CLAS Collaboration Meeting

General Reminders

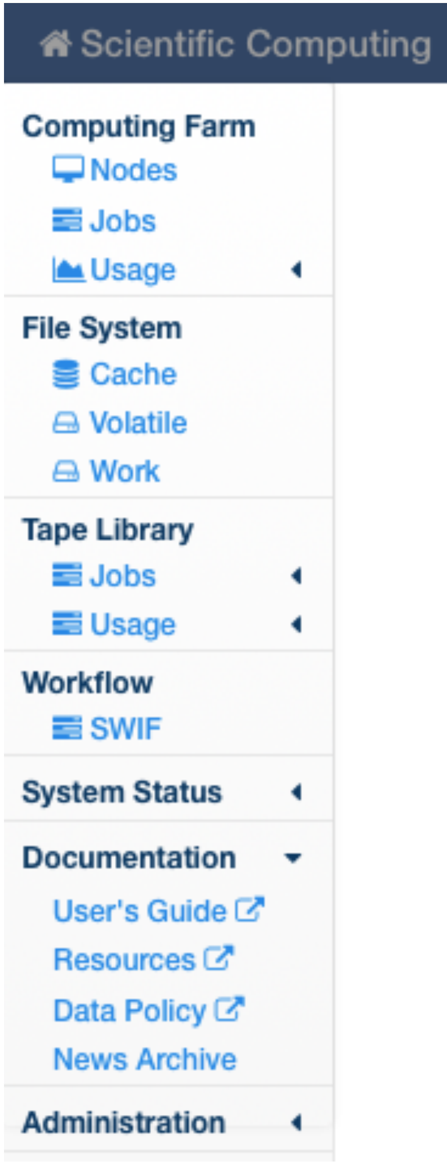
- Pay attention to emails from jlab-scicomp-briefs@jlab.org
    - everyone with a JLab computing account *should* receive them
    - planned/unplanned outages, upgrades, system changes, etc

- Learn and use scicomp's documentation and monitoring web pages for batch jobs, disk quotas, tape access
    - http://scicomp.jlab.org

- You can also monitor non-scicomp quotas (e.g. your /home directory or /group/clas...) at https://cc.jlab.org, after logging in via the link at the top-right

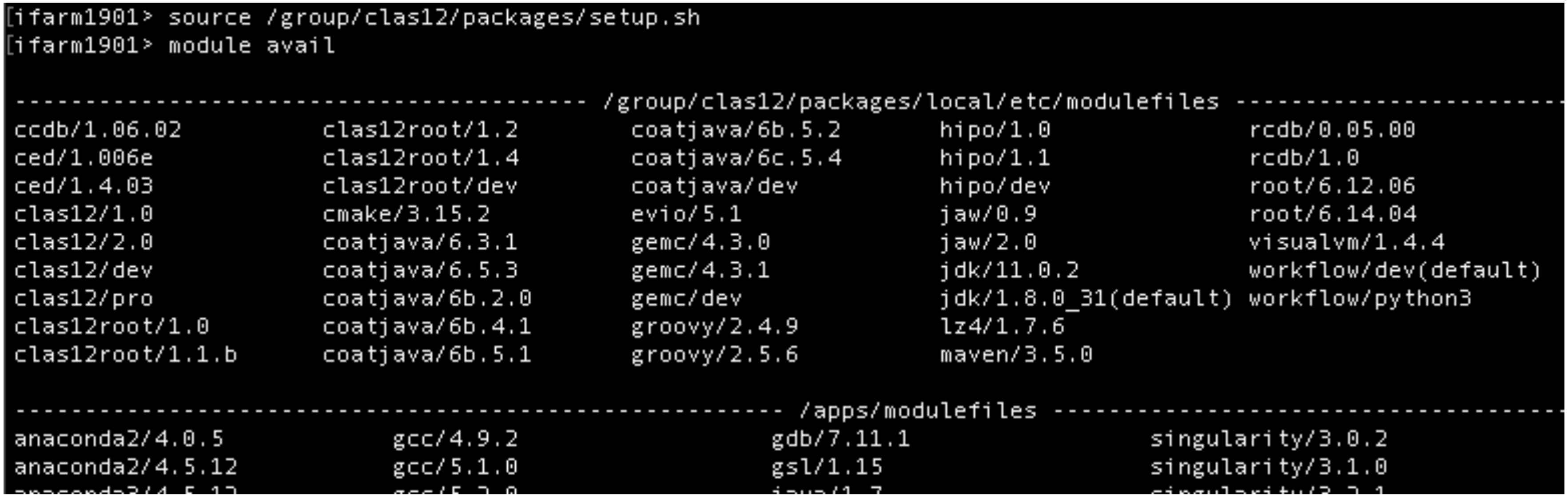
- Announcements regarding general Hall-B computing/software go to:
    - clas12_software@jlab.org (12 GeV era Run Groups)
    - clas_offline@jlab.org (6 GeV era Run Groups)
    - hps/prad/etc@jlab.org

- Note, email address links above will take you to the archive and sign-up page for that mailing list



CLAS12 Software Environment

- A shared installation of all standard clas12 software is officially maintained on the /group disk
  - First source one file:
    - `source /group/clas12/packages/setup.csh` (or `setup.sh` for bash)
  - Then use the module command to see what's available and load them into your environment. Note, scicomp/IT has recently been moving more towards modules too, so you'll see non-clas12 options too (e.g. compilers, singularity).
- Documentation! https://clasweb.jlab.org/wiki/index.php/CLAS12_Software_Center#tab=FAQ
- Note the clas12 "uber" modules, which give you everything in one shot.
  - `clas12/pro` is scheduled to be updated to new production versions (coatjava/gemc/clas12root) later this week and will be announced in advance.



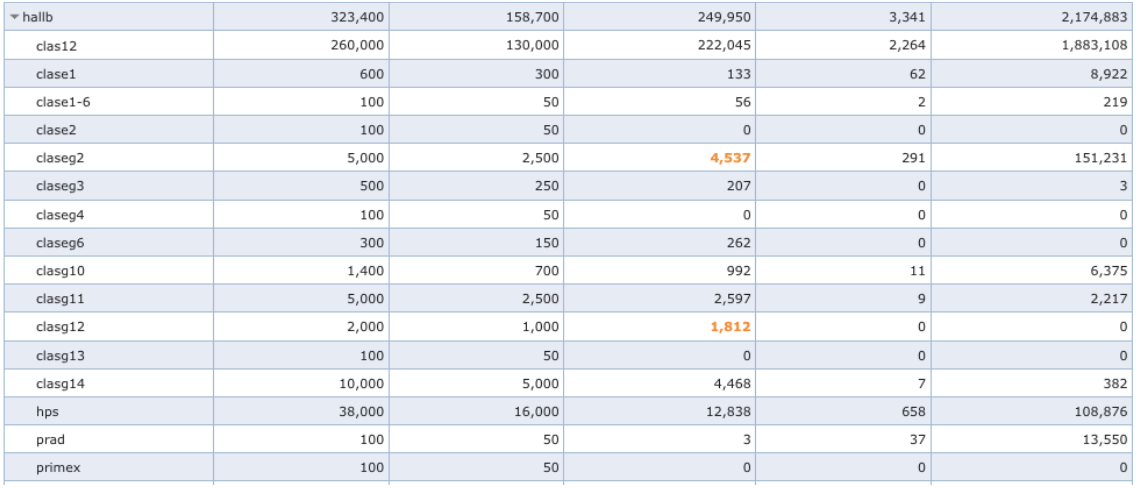
```
[ifarm1901> source /group/clas12/packages/setup.sh
[ifarm1901> module avail

 --------------------------------------- /group/clas12/packages/local/etc/modulefiles -------------------------
 ccdb/1.06.02          clas12root/1.2        coatjava/6b.5.2       hipo/1.0              rcdb/0.05.00
 ced/1.006e            clas12root/1.4        coatjava/6c.5.4       hipo/1.1              rcdb/1.0
 ced/1.4.03            clas12root/dev        coatjava/dev          hipo/dev              root/6.12.06
 clas12/1.0            cmake/3.15.2          evio/5.1              jaw/0.9               root/6.14.04
 clas12/2.0            coatjava/6.3.1        gemc/4.3.0            jaw/2.0               visualvm/1.4.4
 clas12/dev            coatjava/6.5.3        gemc/4.3.1            jdk/11.0.2            workflow/dev(default)
 clas12/pro            coatjava/6b.2.0       gemc/dev              jdk/1.8.0_31(default) workflow/python3
 clas12root/1.0        coatjava/6b.4.1       groovy/2.4.9          lz4/1.7.6
 clas12root/1.1.b      coatjava/6b.5.1       groovy/2.5.6          maven/3.5.0

 ----------------------------------------------------- /apps/modulefiles -------------------------------------
 anaconda2/4.0.5           gcc/4.9.2                 gdb/7.11.1                singularity/3.0.2
 anaconda2/4.5.12          gcc/5.1.0                 gsl/1.15                  singularity/3.1.0
```

CLAS12 Disk Storage (1)

- Lustre fileservers, distributed, good for large data and I/O
  - *automatically* managed based on quotas
  - scicomp has been in the progress of tripling Lustre since last time
    - to facilitate this we transitioned off old /volatile and let people copy data to the new one, in order to rebuild the older filesystems and later return them to the pool
  - `/volatile/clas12`
    - ~~50/25~~ 260/130 **TB** High/Guaranteed
    - scicomp recently gave ability to easy adjust quotas
  - `/cache/clas12`
    - 600/250 **TB** High/Guaranteed
    - Only for files staging from/to tape library, and it's write-through to tape
  - we recently cleaned up the quota heirarchy (everything's now inside "hallb")

- `/work/clas12`
  - **150 TB**, manually managed, single fileserver
  - *not good for large data I/O (e.g. access from batch jobs)*
    - *for production data processing, only the smallest final outputs should go to / work*
  - scicomp is working on a higher-reliability/IO replacement for /work!

/volatile

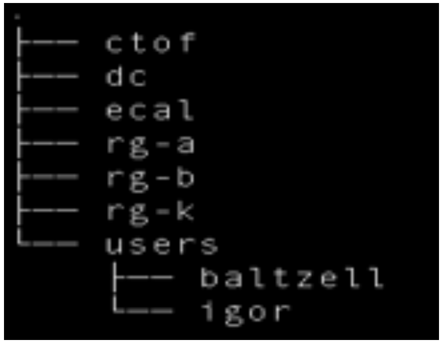


hallb	323,400	158,700	249,950	3,341	2,174,883
clas12	260,000	130,000	222,045	2,264	1,883,108
clase1	600	300	133	62	8,922
clase1-6	100	50	56	2	219
clase2	100	50	0	0	0
claseg2	5,000	2,500	**4,537**	291	151,231
claseg3	500	250	207	0	3
claseg4	100	50	0	0	0
claseg6	300	150	262	0	0
clasg10	1,400	700	992	11	6,375
clasg11	5,000	2,500	2,597	9	2,217
clasg12	2,000	1,000	**1,812**	0	0
clasg13	100	50	0	0	0
clasg14	10,000	5,000	4,468	7	382
hps	38,000	16,000	12,838	658	108,876
prad	100	50	3	37	13,550
primex	100	50	0	0	0

CLAS12 Disk Storage (2)

- Monitoring
  - scicomp.jlab.org
  - And some additional tools we now run, to give finer-grained info for clas12 (linked in the FAQs on our Software Wiki)
    - /work/clas12 usage report, updated weekly
      - http://clasweb.jlab.org/clas12offline/disk/work
    - Auto-deletion queues, updated daily
      - http://clasweb.jlab.org/clas12offline/disk/volatile
      - http://clasweb.jlab.org/clas12offline/disk/cache
- Older clas6 run-groups
  - generally not currently receiving the attention and /volatile disk space increases that ongoing experiments are, and some completely unused /volatile/clas quotas were repurposed
  - /work/clas needs serious cleanup
    - contains data over a decade old!
    - difficult, since run groups are not very active and centralized, lots of inactive accounts own files, need run-group leaders to help
    - and then potentially an increased quota
- Use an appropriate location for your data!
  - e.g. clas6 data in their corresponding run-group locations, not in clas12, and vice-versa
    - This is important to have a quota system that is manageable and appropriate for the needs
  - if an older run-group needs special consideration, more space, make estimates and let me know!

We've talked about moving clas12 to a more organized structure:

- 

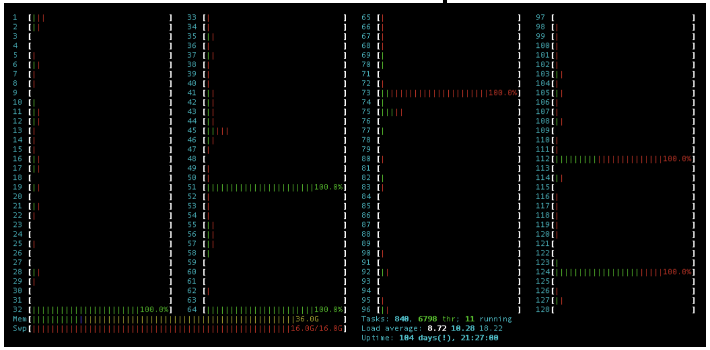
```
.
├── ctof
├── dc
├── ecal
├── rg-a
├── rg-b
├── rg-k
└── users
    ├── baltzell
    └── igor
```

- this will also enable moving towards finer-grained quotas, e.g. per run-group, separately from users easier, if we want to go that route
- To help, move your user directories inside "users", and use your real username for the name of your directory!

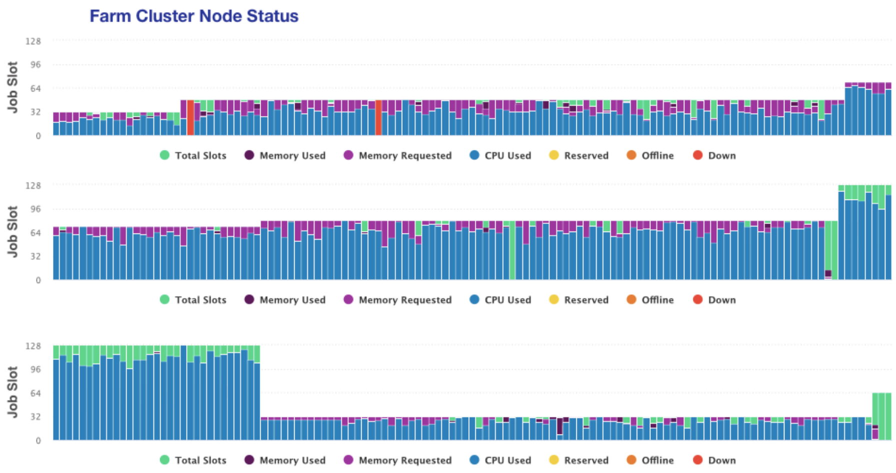
JLab Computing Resources

- Batch and Interactive nodes
  - now all centos7.7
  - interactive
    - ifarm1401 was removed recently
    - ifarm1801/2 now supplemented by ifarm1901
  - batch
    - a mixture of years of purchases
      - farm18XXX and farm19XXX flavors are the same as the corresponding interactive nodes
    - the oldest, least efficient, qcd nodes will be decommissioned this year
  - scicomp supports more options for batch nodes than in previous years
    - interactive jobs and GPUs, some nodes reserved for jupyterhub usage
    - see documentation at https://scicomp.jlab.org

ifarm1901 cpus



```
1  [|||                         ]  33 [|                           ]  65 [|                           ]  97 [                            ]
2  [||                          ]  34 [|                           ]  66 [|                           ]  98 [|                           ]
3  [                            ]  35 [||                          ]  67 [|                           ]  99 [|                           ]
4  [                            ]  36 [|                           ]  68 [|                           ]  100[|                           ]
5  [|                           ]  37 [||                          ]  69 [|                           ]  101[|                           ]
6  [||                          ]  38 [|                           ]  70 [|                           ]  102[|                           ]
7  [|                           ]  39 [|                           ]  71 [                            ]  103[||                          ]
8  [|                           ]  40 [|                           ]  72 [|                           ]  104[|                           ]
9  [                            ]  41 [||                          ]  73 [||||||||||||||||||||||100.0%]  105[||                          ]
10 [|                           ]  42 [||                          ]  74 [|                           ]  106[|                           ]
11 [||                          ]  43 [||                          ]  75 [|||||                       ]  107[|                           ]
12 [||                          ]  44 [||                          ]  76 [                            ]  108[||                          ]
13 [|                           ]  45 [|||||                       ]  77 [|                           ]  109[                            ]
14 [|                           ]  46 [||                          ]  78 [                            ]  110[|                           ]
15 [|                           ]  47 [|                           ]  79 [                            ]  111[|                           ]
16 [||                          ]  48 [                            ]  80 [|                           ]  112[||||||||||||||||||||||100.0%]
17 [||                          ]  49 [|                           ]  81 [                            ]  113[                            ]
18 [                            ]  50 [|                           ]  82 [|                           ]  114[||                          ]
19 [||                          ]  51 [||||||||||||||||||||||100.0%]  83 [|                           ]  115[                            ]
20 [                            ]  52 [|                           ]  84 [                            ]  116[|                           ]
21 [||                          ]  53 [|                           ]  85 [                            ]  117[|                           ]
22 [|                           ]  54 [|                           ]  86 [                            ]  118[|                           ]
23 [                            ]  55 [||                          ]  87 [                            ]  119[|                           ]
24 [                            ]  56 [||                          ]  88 [                            ]  120[|                           ]
25 [|                           ]  57 [||                          ]  89 [                            ]  121[                            ]
26 [                            ]  58 [|                           ]  90 [|                           ]  122[                            ]
27 [                            ]  59 [                            ]  91 [                            ]  123[|                           ]
28 [||                          ]  60 [                            ]  92 [||                          ]  124[||||||||||||||||||||||100.0%]
29 [|                           ]  61 [                            ]  93 [                            ]  125[                            ]
30 [                            ]  62 [|                           ]  94 [                            ]  126[|                           ]
31 [                            ]  63 [                            ]  95 [|                           ]  127[||                          ]
32 [||||||||||||||||||||||100.0%]  64 [||||||||||||||||||||||100.0%]  96 [||                          ]  128[                            ]
Mem[|||||||||||||||||||||||||||||||||||||||||||||||||||||||36.0G      ]  Tasks: 840, 6798 thr; 11 running
Swp[|||||||||||||||||||||||||||||||||||||||||||||||||||||||16.0G/16.0G]  Load average: 8.72 10.28 18.22
                                                                         Uptime: 104 days(!), 21:27:00
```

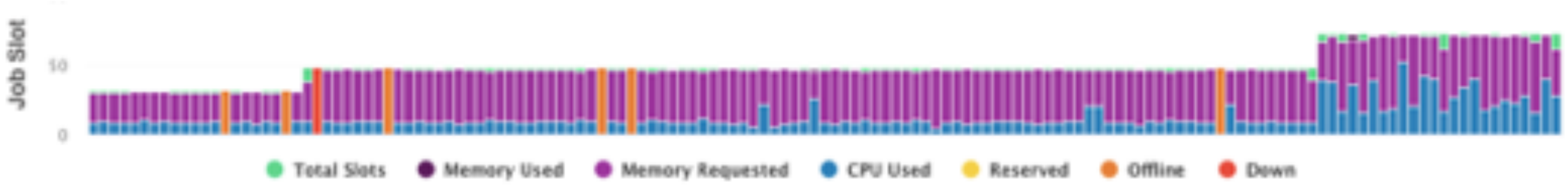


Batch Farm Resource Requests (1)

Optimize your requests according to what your jobs really need!

- **memory/cores**
  - Over-requesting can prevent the farm from running at 100%, screenshot below is a particularly bad example. Memory is the usual culprit, but sometimes people request multiple cores while using only use one.
- **time**
  - Allows the scheduler to optimize backfilling, e.g. inserting shorter, lower priority jobs opportunistically while maintaining the fairshare targets.
- **disk**
  - Under-requesting can cause local node filesystems to max out and cause everyone's jobs on that node to crash.

And use the right "project" for your jobs, for proper accounting and fairshare.



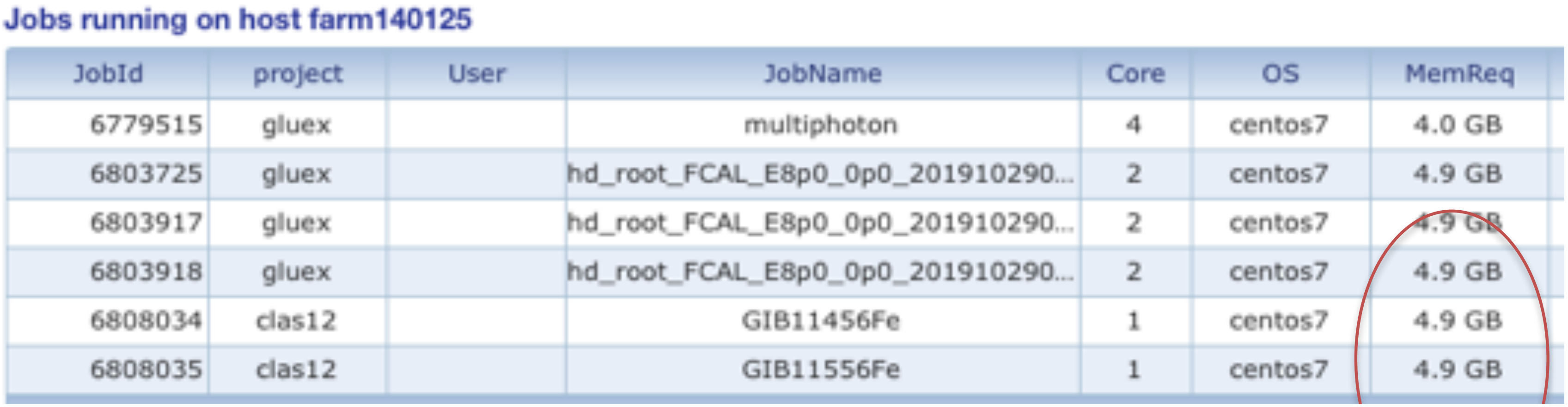
Batch Farm Resource Requests (2)

To investigate your jobs' actual resource usage:

- Run interactively and monitor with the usual linux utilities (e.g. top/htop/du)

- Or, for previous batch jobs, see the"Jobs"link at the top-left of https://scicomp.jlab.org, and then the "Job Query" link in the top-middle
  - For a command-line version of the same info, see slurm-status.py in our workfow module

We've been occasionally checking out Hall B batch jobs, and emailing users on improving their resource requests, to get the most throughput for everyone.

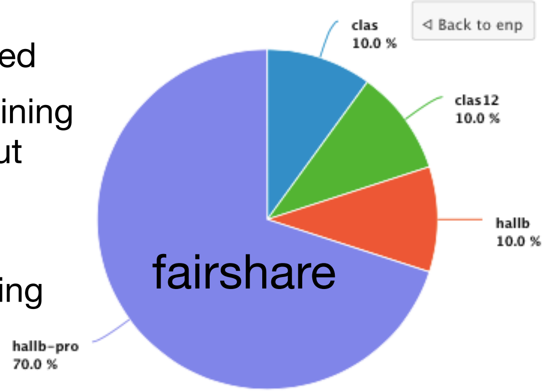


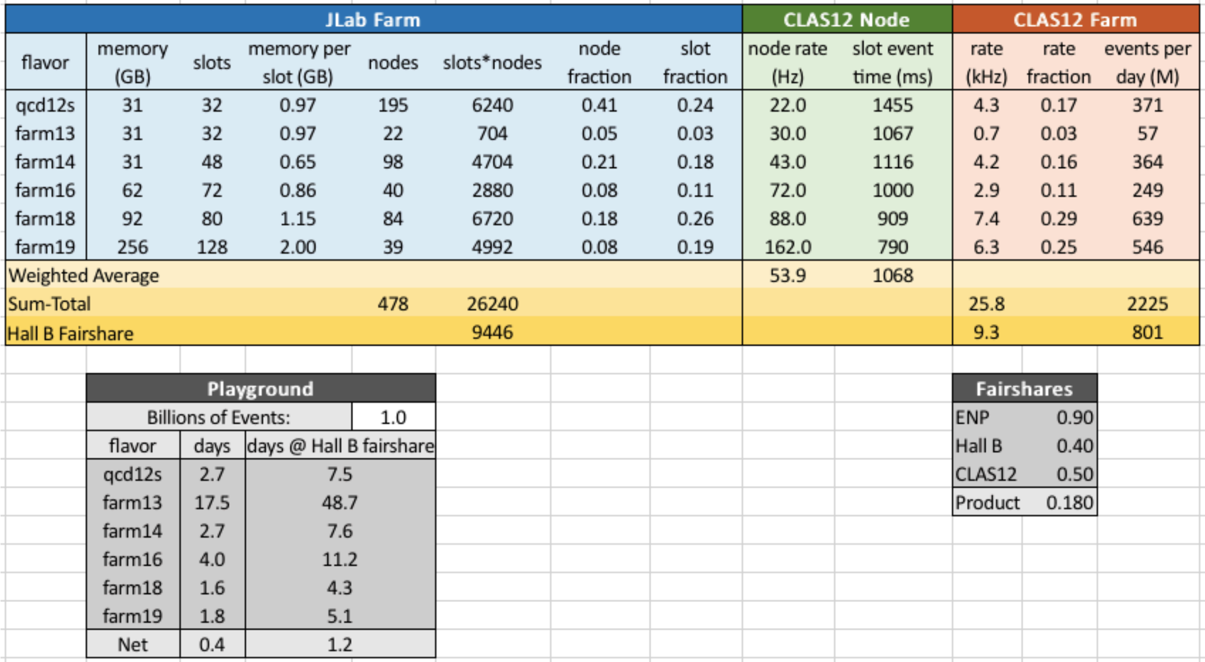
Jobs running on host farm140125

JobId	project	User	JobName	Core	OS	MemReq
6779515	gluex		multiphoton	4	centos7	4.0 GB
6803725	gluex		hd_root_FCAL_E8p0_0p0_201910290...	2	centos7	4.9 GB
6803917	gluex		hd_root_FCAL_E8p0_0p0_201910290...	2	centos7	4.9 GB
6803918	gluex		hd_root_FCAL_E8p0_0p0_201910290...	2	centos7	4.9 GB
6808034	clas12		GIB11456Fe	1	centos7	4.9 GB
6808035	clas12		GIB11556Fe	1	centos7	4.9 GB

CLAS12 Data Processing (1)

- Hall B's net fairshare is ~**36%** of batch farm, see the "Usage"link at scicomp.jlab.org
  - scicomp is now using SLURM's Tree Faishare Algorithm, with better load-balancing between production jobs and everything else, keeping farm loaded
- We've developed tools to aid in studying our throughput by analyzing and combining info from our jobs' log files, SWIF and SLURM database queries, checking output locations
  - This allowed really tracking progress, implementing fixes, optimizing Clara/SLURM job configurations, memory usage, I/O logistics ... and understanding how our empirical throughput compares to fairshare and benchmarks

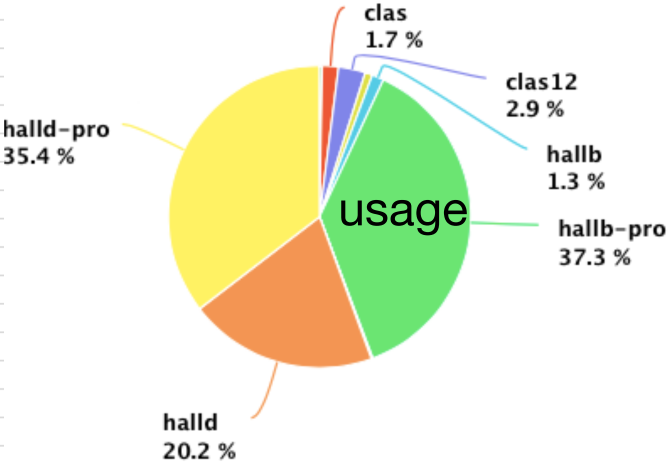




JLab Farm								CLAS12 Node		CLAS12 Farm		
flavor	memory (GB)	slots	memory per slot (GB)	nodes	slots*nodes	node fraction	slot fraction	node rate (Hz)	slot event time (ms)	rate (kHz)	rate fraction	events per day (M)
qcd12s	31	32	0.97	195	6240	0.41	0.24	22.0	1455	4.3	0.17	371
farm13	31	32	0.97	22	704	0.05	0.03	30.0	1067	0.7	0.03	57
farm14	31	48	0.65	98	4704	0.21	0.18	43.0	1116	4.2	0.16	364
farm16	62	72	0.86	40	2880	0.08	0.11	72.0	1000	2.9	0.11	249
farm18	92	80	1.15	84	6720	0.18	0.26	88.0	909	7.4	0.29	639
farm19	256	128	2.00	39	4992	0.08	0.19	162.0	790	6.3	0.25	546
Weighted Average								53.9	1068			
Sum-Total				478	26240					25.8		2225
Hall B Fairshare					9446					9.3		801

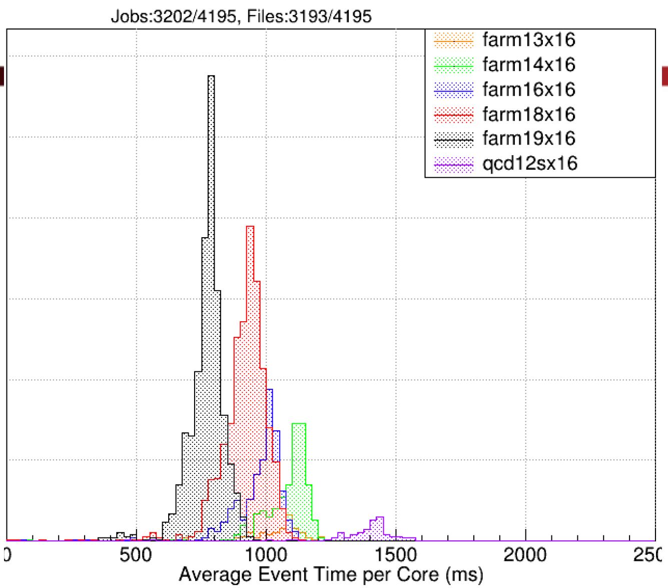
Playground		
Billions of Events:		1.0
flavor	days	days @ Hall B fairshare
qcd12s	2.7	7.5
farm13	17.5	48.7
farm14	2.7	7.6
farm16	4.0	11.2
farm18	1.6	4.3
farm19	1.8	5.1
Net	0.4	1.2

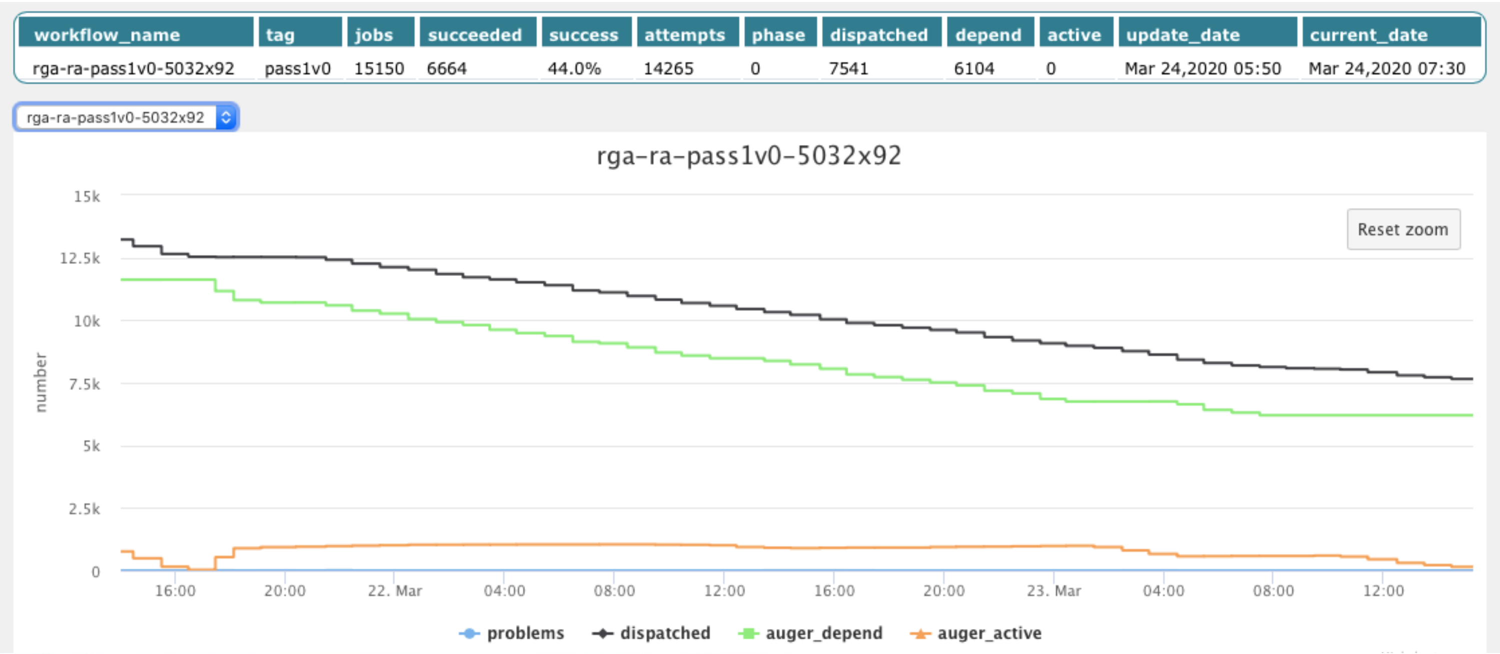
Fairshares	
ENP	0.90
Hall B	0.40
CLAS12	0.50
Product	0.180



CLAS12 Data Processing (2)

- We use JLab's SWIF workflow tools
    - to combine all data processing stages, multi- and single-core, into one workflow, using job-job dependencies to automatically trigger downstream jobs when ready
    - to ultimately get ~100% hands-free success rate for chefs, when combining job optimization from previous slide, and automatic SWIF job retries
- With a single, easy interface for chefs; no one-off scripts needed, no file-list generation required.
- Python-based and written with extension to other experiments in mind
- Plus shipping periodic SWIF snapshots to clas12mon for better monitoring (need to see if scicomp would just support that instead)





workflow_name	tag	jobs	succeeded	success	attempts	phase	dispatched	depend	active	update_date	current_date
rga-ra-pass1v0-5032x92	pass1v0	15150	6664	44.0%	14265	0	7541	6104	0	Mar 24,2020 05:50	Mar 24,2020 07:30

Done