Upcoming Software Projects

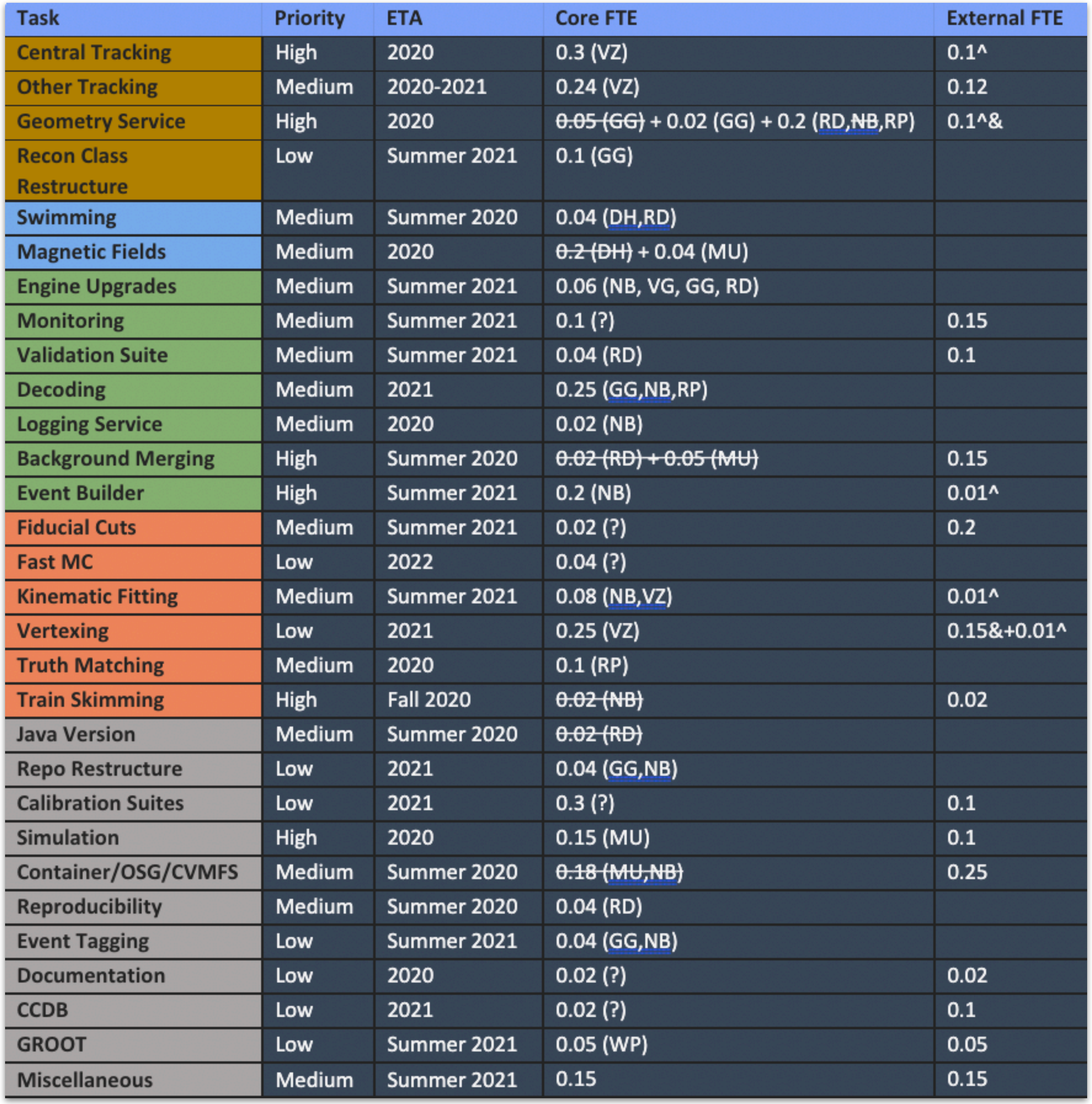
N. Baltzell - March 22, 2023 - CLAS Collaboration Meeting

Introduction

- Hall B initiated task forces ~2.5 years ago to gather projects and set priorities for future CLAS12 developments. One of those task forces was dedicated to software, and ...
  - Many of its projects have actually been completed (believe it or not!), a couple were abandoned, some were low priority and/or incompatible time-wise with other task forces' priorities, and some were later deemed warranting of waiting for pass2 ...
- This presentation is, in some sense, a return to and an extension of that
  - *And also to inform the collaboration, solicit opinions on priorities, and welcome contributions*
- This presentation will not be telling a story but rather a random walk of some prevailing ideas and options on future software infrastructure developments for Hall B and CLAS12, roughly categorized as:

1. Online
2. Simulation
3. High Luminosity
4. Data Processing
5. Physics Analysis
6. Other

2020 Software Task Force Summary Table



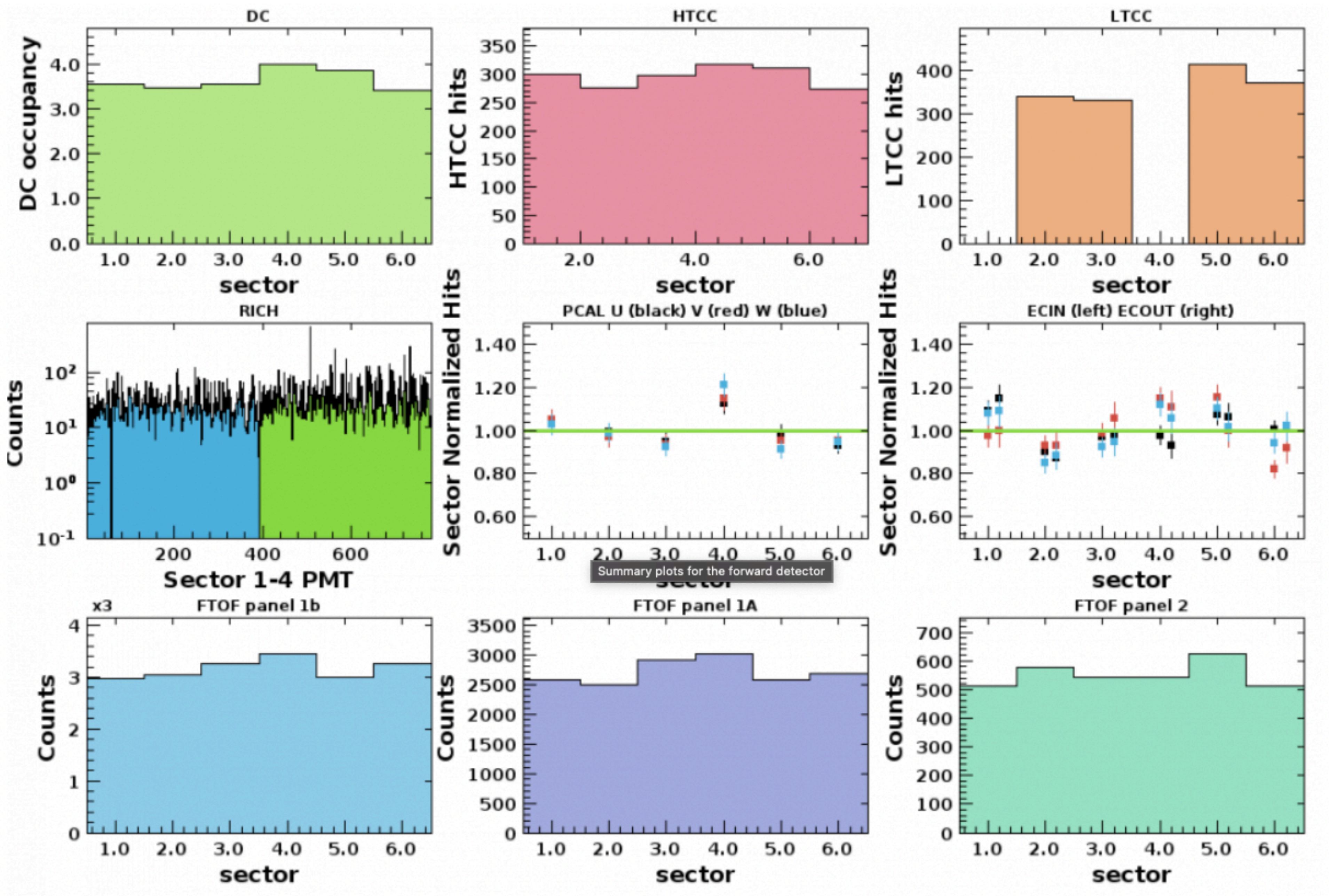
Task	Priority	ETA	Core FTE	External FTE
Central Tracking	High	2020	0.3 (VZ)	0.1^
Other Tracking	Medium	2020-2021	0.24 (VZ)	0.12
Geometry Service	High	2020	~~0.05 (GG)~~ + 0.02 (GG) + 0.2 (RD,~~NB~~,RP)	0.1^&
Recon Class Restructure	Low	Summer 2021	0.1 (GG)	
Swimming	Medium	Summer 2020	0.04 (DH,RD)	
Magnetic Fields	Medium	2020	~~0.2 (DH)~~ + 0.04 (MU)	
Engine Upgrades	Medium	Summer 2021	0.06 (NB, VG, GG, RD)	
Monitoring	Medium	Summer 2021	0.1 (?)	0.15
Validation Suite	Medium	Summer 2021	0.04 (RD)	0.1
Decoding	Medium	2021	0.25 (GG,NB,RP)	
Logging Service	Medium	2020	0.02 (NB)	
Background Merging	High	Summer 2020	~~0.02 (RD) + 0.05 (MU)~~	0.15
Event Builder	High	Summer 2021	0.2 (NB)	0.01^
Fiducial Cuts	Medium	Summer 2021	0.02 (?)	0.2
Fast MC	Low	2022	0.04 (?)	
Kinematic Fitting	Medium	Summer 2021	0.08 (NB,VZ)	0.01^
Vertexing	Low	2021	0.25 (VZ)	0.15&+0.01^
Truth Matching	Medium	2020	0.1 (RP)	
Train Skimming	High	Fall 2020	~~0.02 (NB)~~	0.02
Java Version	Medium	Summer 2020	~~0.02 (RD)~~	
Repo Restructure	Low	2021	0.04 (GG,NB)	
Calibration Suites	Low	2021	0.3 (?)	0.1
Simulation	High	2020	0.15 (MU)	0.1
Container/OSG/CVMFS	Medium	Summer 2020	~~0.18 (MU,NB)~~	0.25
Reproducibility	Medium	Summer 2020	0.04 (RD)	
Event Tagging	Low	Summer 2021	0.04 (GG,NB)	
Documentation	Low	2020	0.02 (?)	0.02
CCDB	Low	2021	0.02 (?)	0.1
GROOT	Low	Summer 2021	0.05 (WP)	0.05
Miscellaneous	Medium	Summer 2021	0.15	0.15

Link to full O365 document with details (expires in a couple months): Software Task Force - V2.docx

Online

Reconstruction & Monitoring

- We have unused computing power available in the Hall B counting house, e.g. for better online monitoring
  - With current, full reconstruction, enough for 100 Hz on a single node, 400 Hz if distributed, or more if we choose a subset of reconstruction
- *But currently we only look at occupancies!!* Limited and made cumbersome by missing software infrastructure and some weak points, for example ...
  - single-threaded/slow EVIO→HIPO decoding
  - no inline EVIO→HIPO decoding, no HIPO ring for distribution
  - standalone mon12 application
- We need to
  - Implement the missing functionalities (lots already exist in some form)
  - Distribute the existing ET-ring to a HIPO ring, implement multi-threaded processing to leverage the existing computing resources
  - Move histogram generation to a server-side process, with clients that don't need to be manually clicked/operated but instead automatic
- Remember the EPSCI group is supporting Hydra for online, automated fault detection, but we need to label more images ...
  - See Torri Jeske's recent presentation at a Calcom meeting: https://clasweb.jlab.org/wiki/images/3/35/HYDRA_CALCOM_3102023_-_2023-03-09_14.37.29.pdf



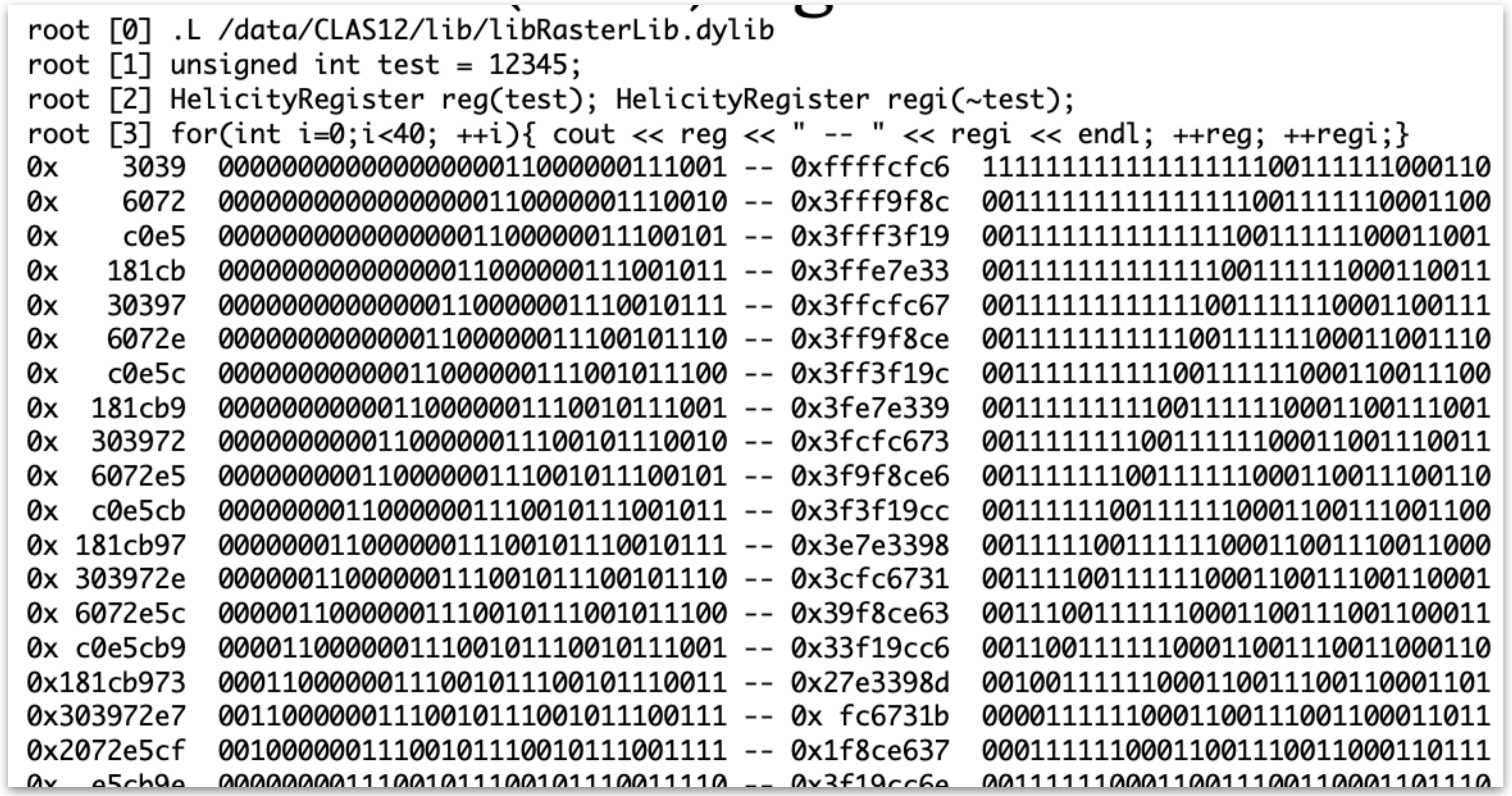
This has various synergies, e.g.,

- **Opens the possibility to run any combination of our reconstruction services online, with monitoring**
- **Allows to leverage the CPU horsepower already available in the Hall B counting house**
- **Reduces the cumbersomeness of shift workers having to run a standalone application just to look at occupancies**
- **Allows to leverage currently offline algorithms, e.g. AI/ML, in the L3 trigger**
- **Allows to test offline algorithms online, in real time**

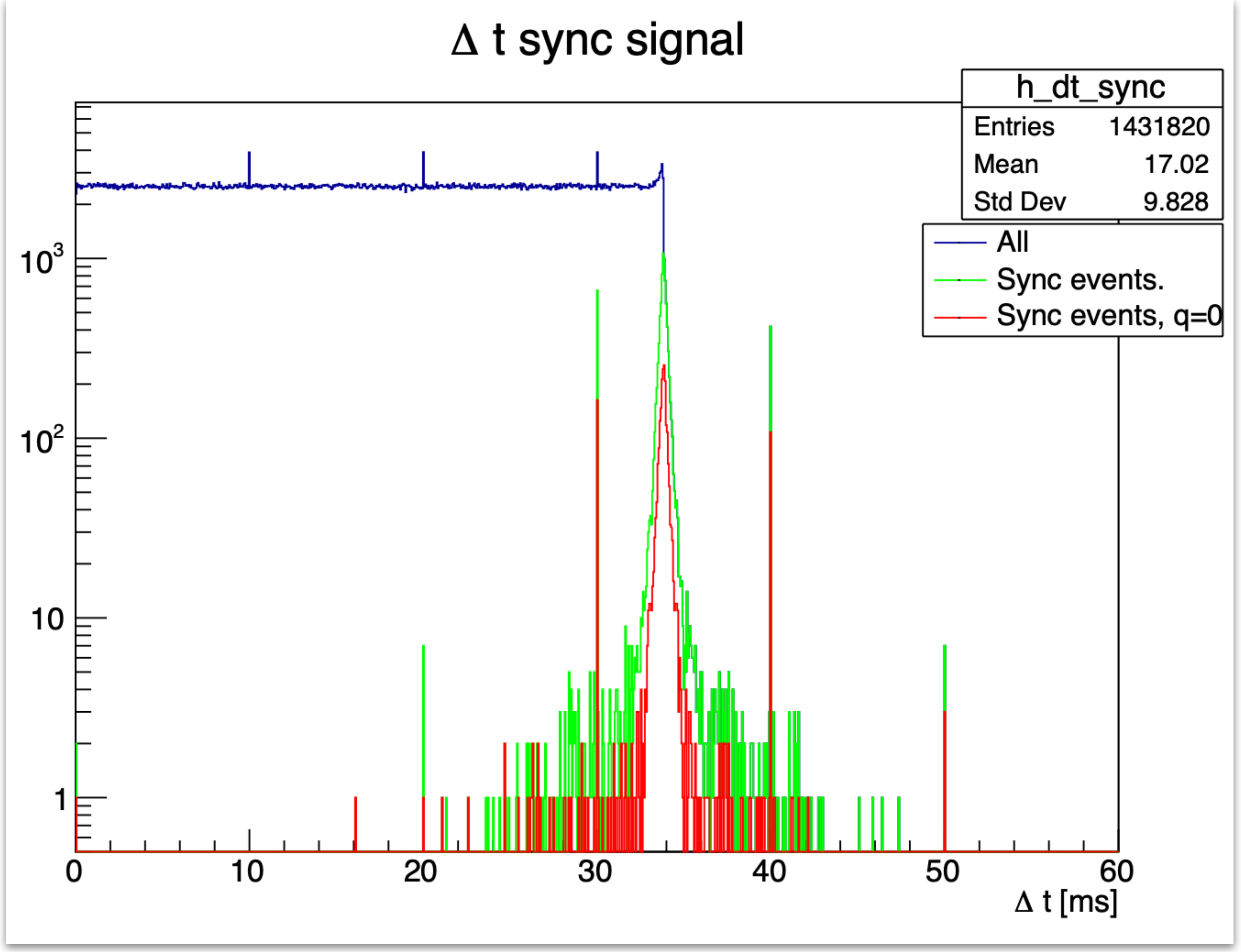
Online

Helicity Decoder Board

- Currently we analyze the sequence and correct for the delay in 2 independent places:
    - Online in L3, based on projecting from the pseudorandom sequence once seeded
    - Offline based on both pseudorandom and empirical sequence (due to the luxury of mapping the sequence before looking at physics events), allowing stronger validity checks and significantly higher efficiency
    - *Both require serial event access to initialize the sequence*
- A new board to be used by all halls is under testing
    - Provides the previous 30+ helicity states on every readout, and various debugging information
        - So every event can be analyzed independently, like it should always have been!
    - Critical for going to the kHz+ helicity clocks needed by parity experiments in other halls in coming years, where DAQ deadtime will necessarily cause much more frequent missing state changes
    - Maurik Holtrop did some initial testing and heavy lifting during RG-C, and we need to finish and incorporate it in CLAS12 software
    - *Not a terribly big project, but a critical one*



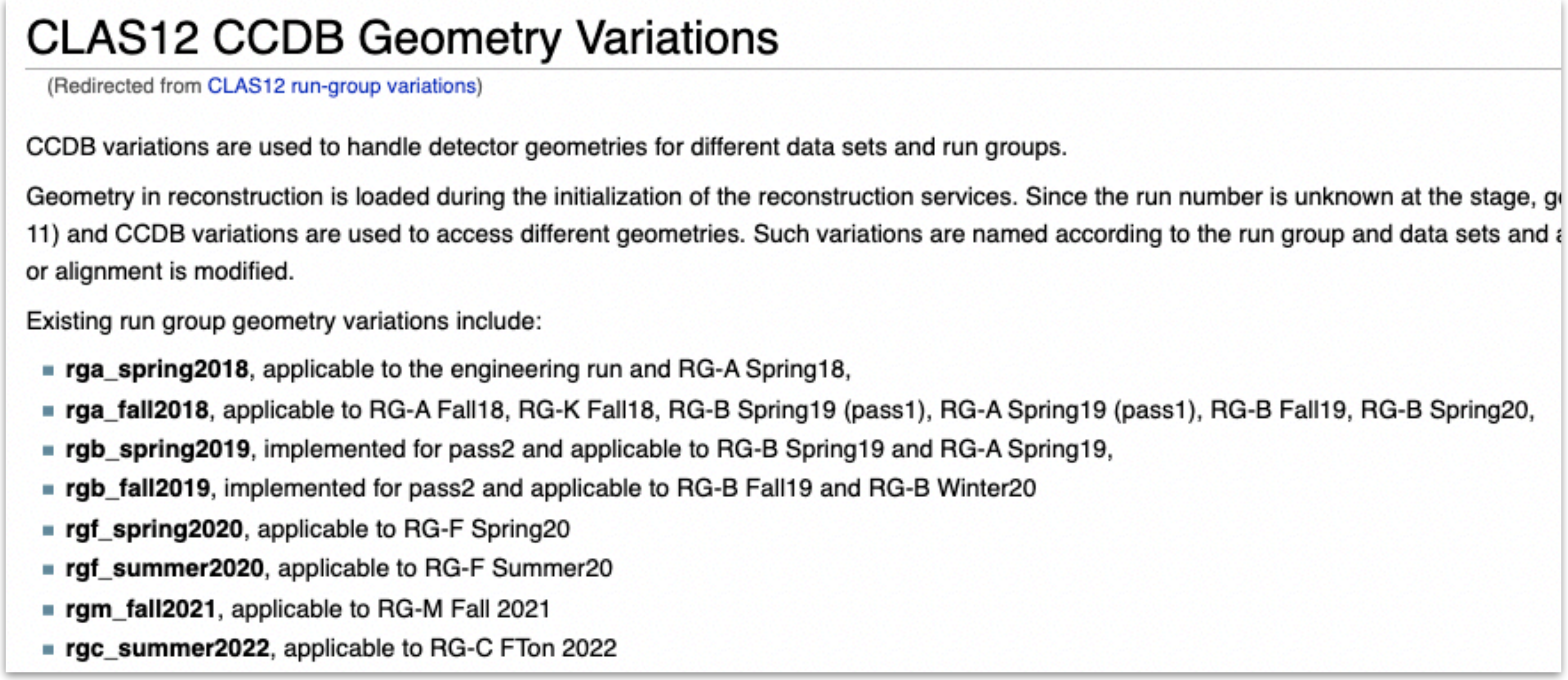
```
root [0] .L /data/CLAS12/lib/libRasterLib.dylib
root [1] unsigned int test = 12345;
root [2] HelicityRegister reg(test); HelicityRegister regi(~test);
root [3] for(int i=0;i<40; ++i){ cout << reg << " -- " << regi << endl; ++reg; ++regi;}
0x    3039  00000000000000000011000000111001 -- 0xffffcfc6  11111111111111111100111111000110
0x    6072  00000000000000000110000001110010 -- 0x3fff9f8c  00111111111111111001111110001100
0x    c0e5  00000000000000001100000011100101 -- 0x3fff3f19  00111111111111110011111100011001
0x   181cb  00000000000000011000000111001011 -- 0x3ffe7e33  00111111111111100111111000110011
0x   30397  00000000000000110000001110010111 -- 0x3ffcfc67  00111111111111001111110001100111
0x   6072e  00000000000001100000011100101110 -- 0x3ff9f8ce  00111111111110011111100011001110
0x   c0e5c  00000000000011000000111001011100 -- 0x3ff3f19c  00111111111100111111000110011100
0x  181cb9  00000000000110000001110010111001 -- 0x3fe7e339  00111111111001111110001100111001
0x  303972  00000000001100000011100101110010 -- 0x3fcfc673  00111111110011111100011001110011
0x  6072e5  00000000011000000111001011100101 -- 0x3f9f8ce6  00111111100111111000110011100110
0x  c0e5cb  00000000110000001110010111001011 -- 0x3f3f19cc  00111111001111110001100111001100
0x 181cb97  00000001100000011100101110010111 -- 0x3e7e3398  00111110011111100011001110011000
0x 303972e  00000011000000111001011100101110 -- 0x3cfc6731  00111100111111000110011100110001
0x 6072e5c  00000110000001110010111001011100 -- 0x39f8ce63  00111001111110001100111001100011
0x c0e5cb9  00001100000011100101110010111001 -- 0x33f19cc6  00110011111100011001110011000110
0x181cb973  00011000000111001011100101110011 -- 0x27e3398d  00100111111000110011100110001101
0x303972e7  00110000001110010111001011100111 -- 0x fc6731b  00001111110001100111001100011011
0x2072e5cf  00100000011100101110010111001111 -- 0x1f8ce637  00011111100011001110011000110111
0x  e5cb9e  00000000111001011100101110011110 -- 0x3f19cc6e  00111111000110011100110001101110
```



Simulation

Real Run Numbers

- We went down the path of using CCDB variations for geometry parameters
    - Maybe in part due to the history of using (and really liking) run number 11!
    - https://clasweb.jlab.org/wiki/index.php/CLAS12_CCDB_Geometry_Variations
- But this has gotten too cumbersome, difficult to maintain, and using run numbers for geometry is just the right way to go anyway
    - Not a tremendous effort required, but definitely non-trivial, is it worth it?
- Note, using run numbers will allow to sample run conditions programmatically, particularly luminosity
    - For example, "I want to simulate run numbers 4000-4100 and 4110 and 4112"
        - then we retrieve beam currents and event counts from RCDB, and sample background merging files accordingly
        - could also do beam energy automatically, or #events, based on RCDB
- Also, we can move to a unified geometry manager, while currently every service/detector that wants another detector's geometry has to go it alone



CLAS12 CCDB Geometry Variations

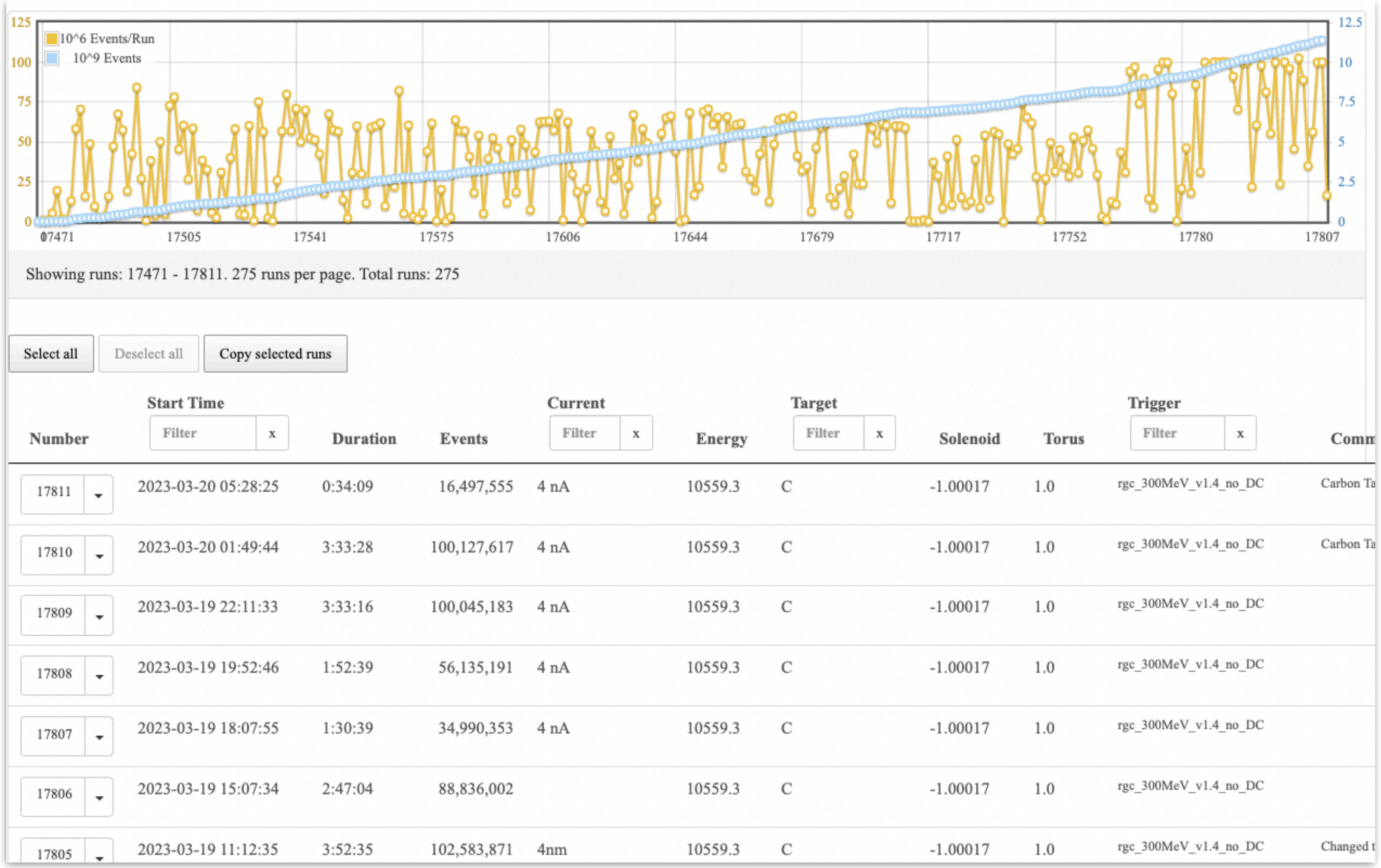
(Redirected from CLAS12 run-group variations)

CCDB variations are used to handle detector geometries for different data sets and run groups.

Geometry in reconstruction is loaded during the initialization of the reconstruction services. Since the run number is unknown at the stage, g
11) and CCDB variations are used to access different geometries. Such variations are named according to the run group and data sets and
or alignment is modified.

Existing run group geometry variations include:

- **rga_spring2018**, applicable to the engineering run and RG-A Spring18,
- **rga_fall2018**, applicable to RG-A Fall18, RG-K Fall18, RG-B Spring19 (pass1), RG-A Spring19 (pass1), RG-B Fall19, RG-B Spring20,
- **rgb_spring2019**, implemented for pass2 and applicable to RG-B Spring19 and RG-A Spring19,
- **rgb_fall2019**, implemented for pass2 and applicable to RG-B Fall19 and RG-B Winter20
- **rgf_spring2020**, applicable to RG-F Spring20
- **rgf_summer2020**, applicable to RG-F Summer20
- **rgm_fall2021**, applicable to RG-M Fall 2021
- **rgc_summer2022**, applicable to RG-C FTon 2022



Showing runs: 17471 - 17811. 275 runs per page. Total runs: 275

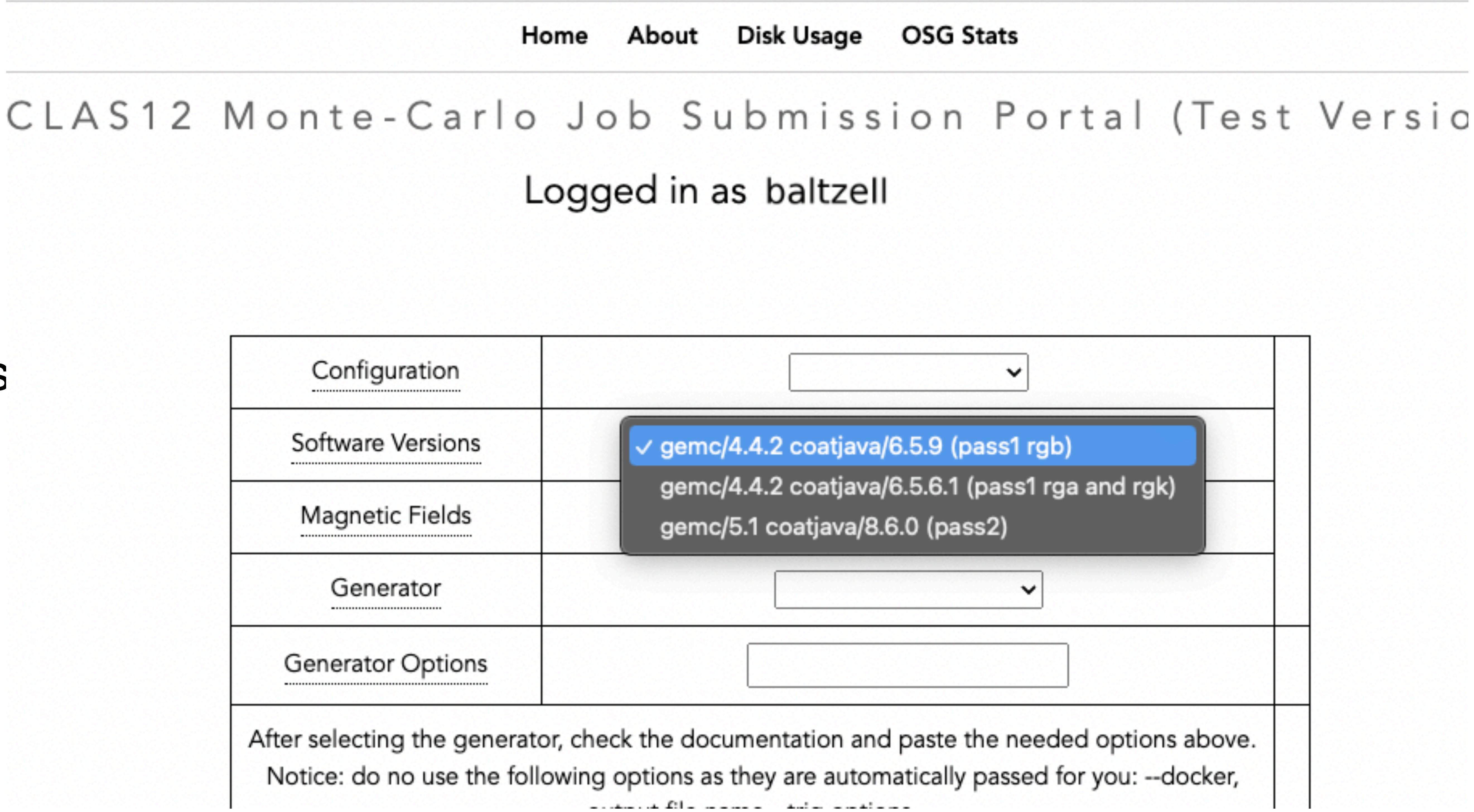
Number	Start Time	Duration	Events	Current	Energy	Target	Solenoid	Torus	Trigger	Comm
17811	2023-03-20 05:28:25	0:34:09	16,497,555	4 nA	10559.3	C	-1.00017	1.0	rgc_300MeV_v1.4_no_DC	Carbon Ta
17810	2023-03-20 01:49:44	3:33:28	100,127,617	4 nA	10559.3	C	-1.00017	1.0	rgc_300MeV_v1.4_no_DC	Carbon Ta
17809	2023-03-19 22:11:33	3:33:16	100,045,183	4 nA	10559.3	C	-1.00017	1.0	rgc_300MeV_v1.4_no_DC	
17808	2023-03-19 19:52:46	1:52:39	56,135,191	4 nA	10559.3	C	-1.00017	1.0	rgc_300MeV_v1.4_no_DC	
17807	2023-03-19 18:07:55	1:30:39	34,990,353	4 nA	10559.3	C	-1.00017	1.0	rgc_300MeV_v1.4_no_DC	
17806	2023-03-19 15:07:34	2:47:04	88,836,002		10559.3	C	-1.00017	1.0	rgc_300MeV_v1.4_no_DC	
17805	2023-03-19 11:12:35	3:52:35	102,583,871	4nm	10559.3	C	-1.00017	1.0	rgc_300MeV_v1.4_no_DC	Changed t

Simulation

OSG Features

- Generalize web submission to support specifying software versions
  - Necessary soon and for the distant future; currently it's just what was needed for pass1s
- Remove software builds from the container and rely on CVMFS
  - Easier maintenance, avoid container bloat
- Those are all in progress and will be in use and available in the coming weeks

- Also, down the road, add support for previous slide's run number selection



CLAS12 Monte-Carlo Job Submission Portal (Test Versio

Logged in as baltzell

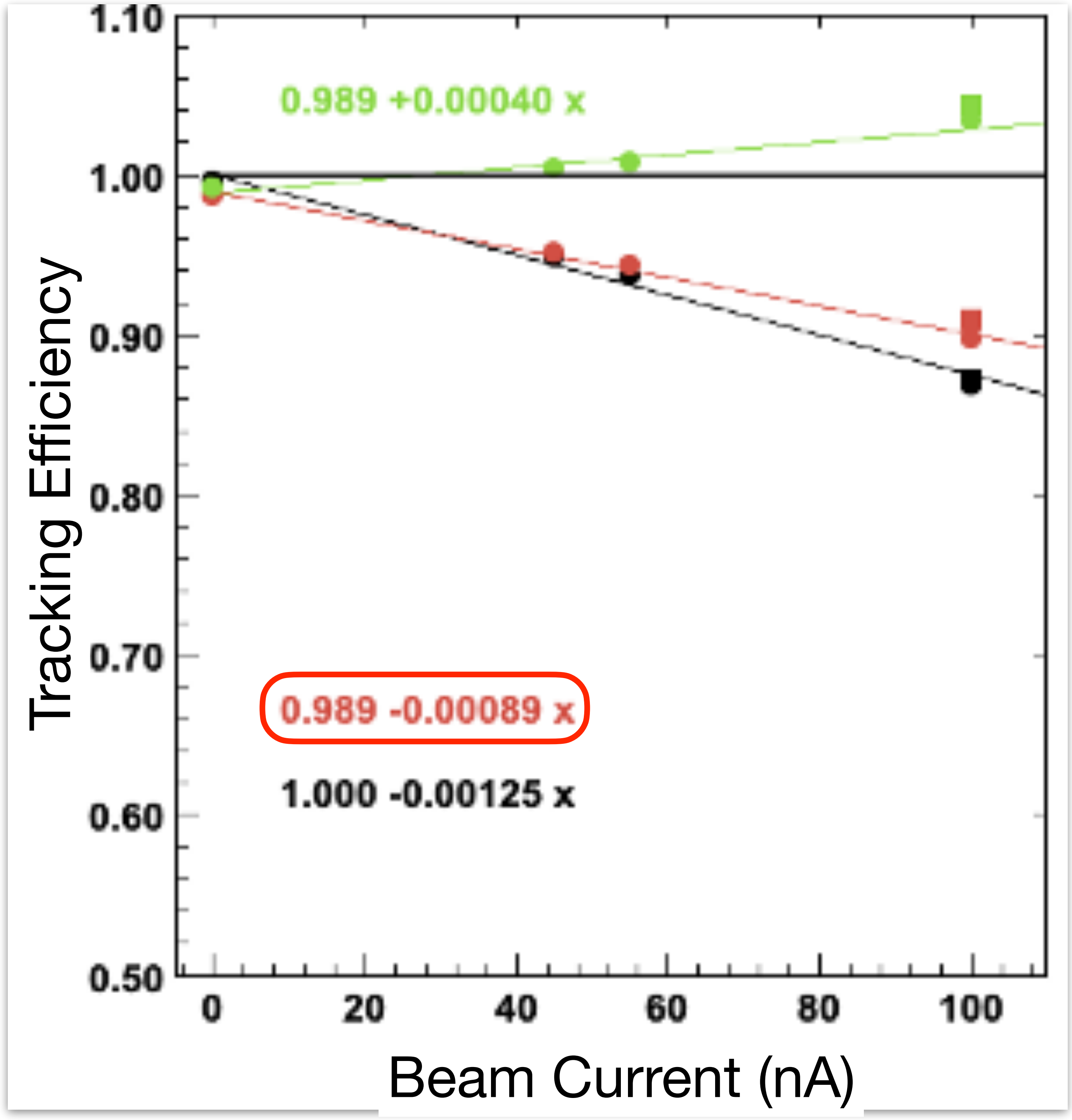
Configuration	
Software Versions	✓ gemc/4.4.2 coatjava/6.5.9 (pass1 rgb) gemc/4.4.2 coatjava/6.5.6.1 (pass1 rga and rgk) gemc/5.1 coatjava/8.6.0 (pass2)
Magnetic Fields	
Generator	
Generator Options	

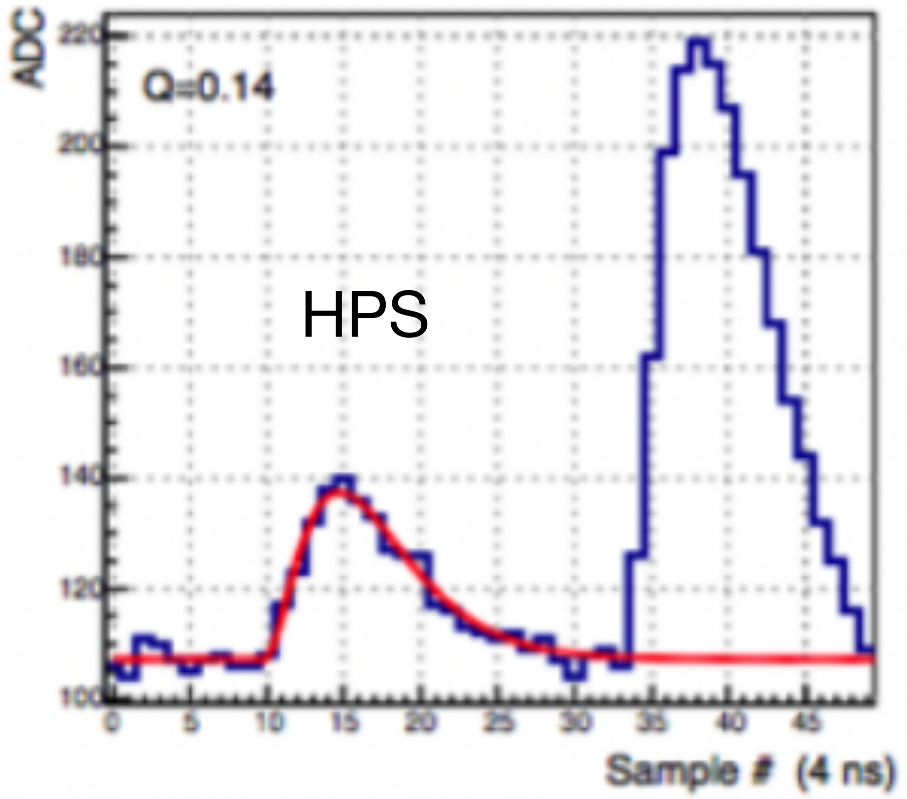
After selecting the generator, check the documentation and paste the needed options above.
Notice: do no use the following options as they are automatically passed for you: --docker,

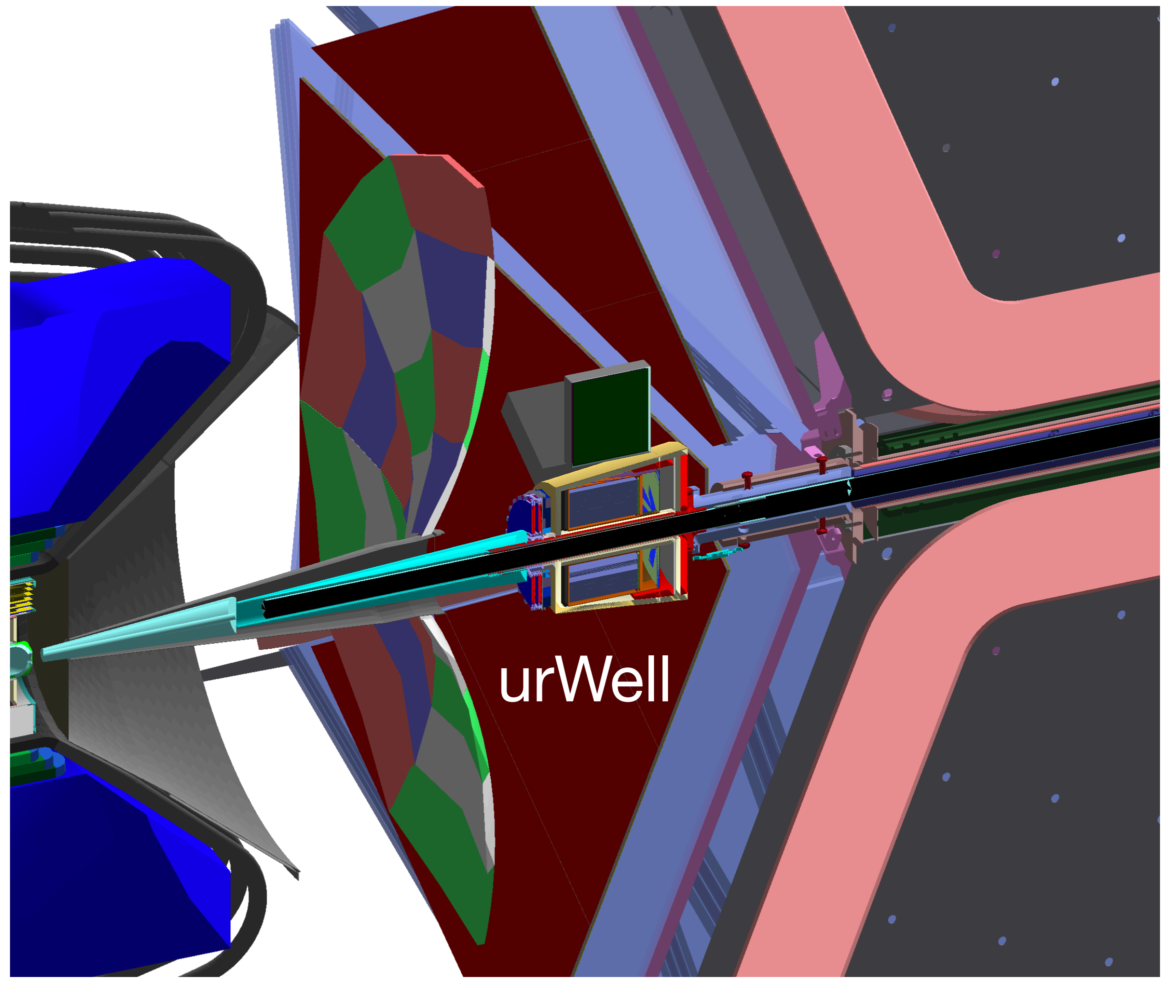
High Luminosity

uRWell Tracking, Multi-hit FADC

- Down to < 0.1% per nA inefficiency with uRWell and DC-denoising
    - Already achieves the current DC-only efficiency but at double the luminosity just by requiring geometric matching with uRWell
    - Work progressing on many fronts: CED, GEMC, Kalman filtering, ...
    - First version of full tracking with uRWell+DC has been developed by Tongtong, including refactoring to make it fit
    - Will be exploring new conventional track seeding options, AI track finding with uRWell, smoothing and energy loss, more generic tracking tools
- Multi-Hit FADC
    - We readout in "Mode-1" with 4 ns samples from the FADC250 boards, with bitpacking to reduce data volume
    - But currently we then extract only a single pulse offline using the FADC250's "Mode-7" algorithm
        - the first threshold crossing in the readout window, with an interpolation across half-height for time (to reduce time-walk) and a simple arithmetic sum for the integral
    - For high-luminosity, this is a no-go for some detectors and we should anticipate needing to accommodate pileup
- Also, AI trigger in L3, but that's in Gagik's talk



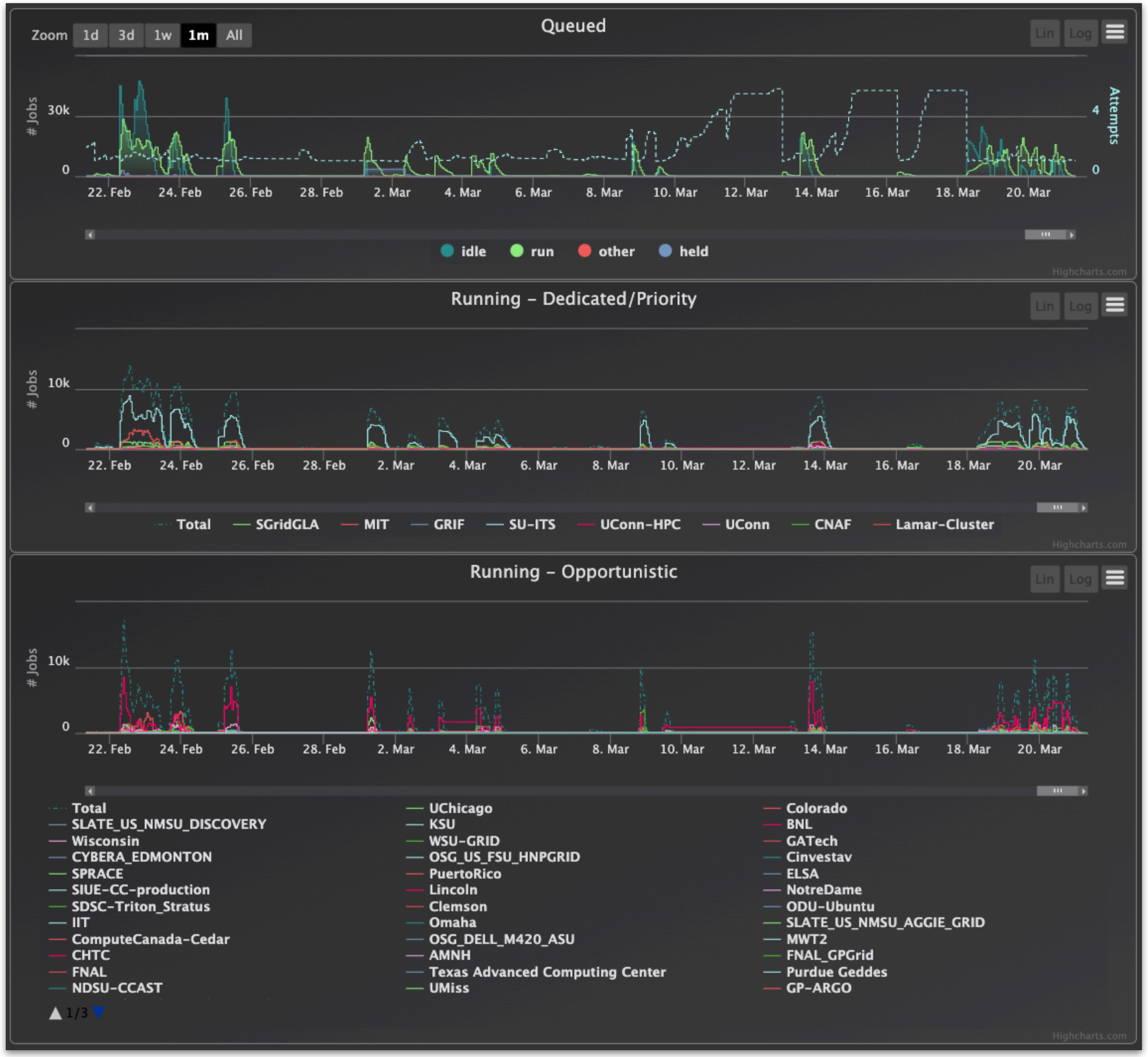




Data Processing

Non-JLab Resources

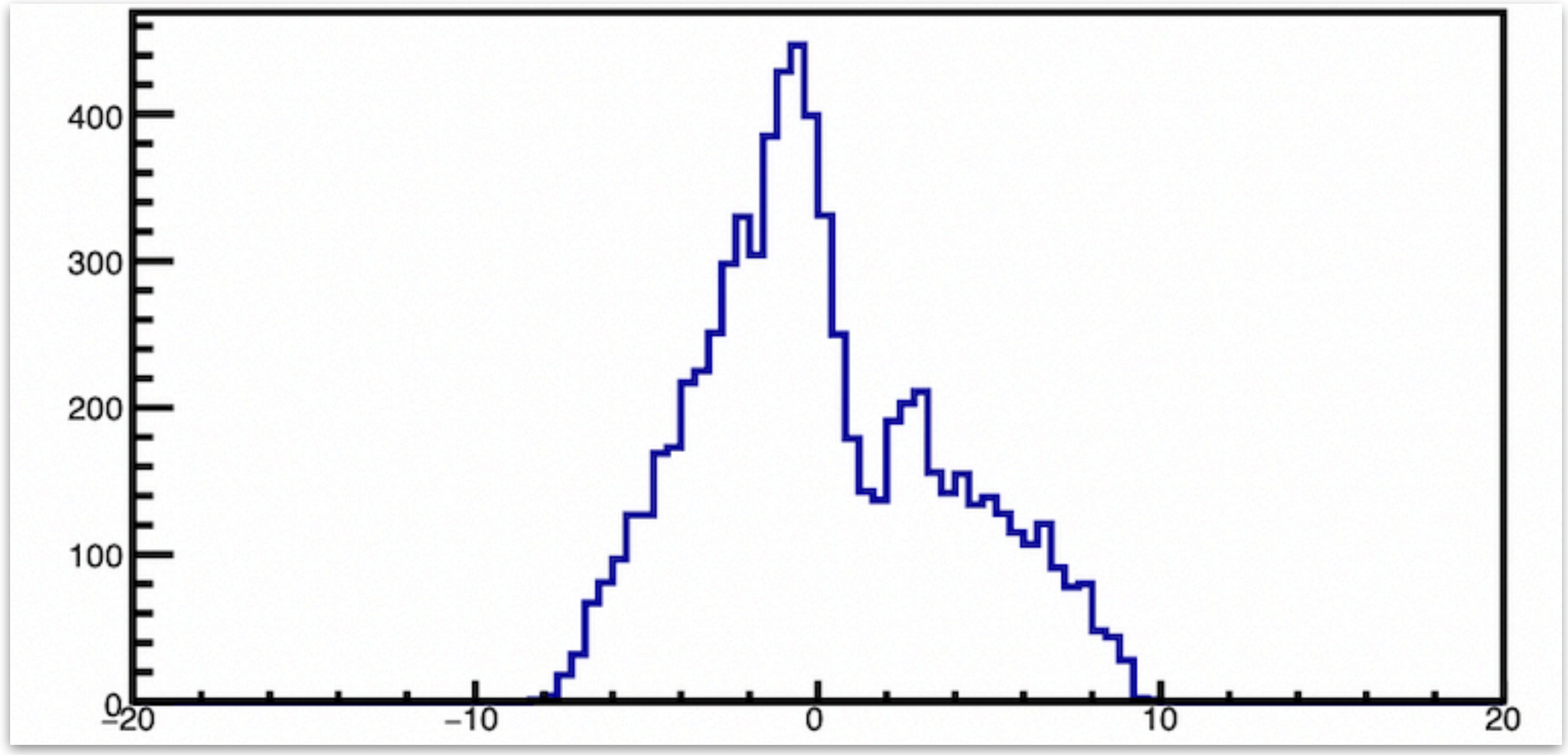
- OSG
  - We're using it for simulations, pretty effectively, certainly keeping up with the submissions (CLAS12 was the largest OSG consumer last year, but queues often empty lately!)
  - Nothing really new here other than the previously mentioned web submission changes, and soliciting what improvement users want?
  - But opportunistic resources are not a great fit for reconstruction of real data, at least not yet, lots of bookkeeping required, although there's been a broader discussion with JLab's scicomp and other halls on this
- NERSC
  - Last year we did some basic tests, including scaling, with Nick Tyler (graduate student with CLAS, now a postdoc at NERSC)
  - We applied for and were granted an allocation for this calendar year. We'll start using it with the coming pass2s and should exhaust it within a couple months
  - With a fully vetted and utilized workflow, we'll apply for a significantly larger allocation next year



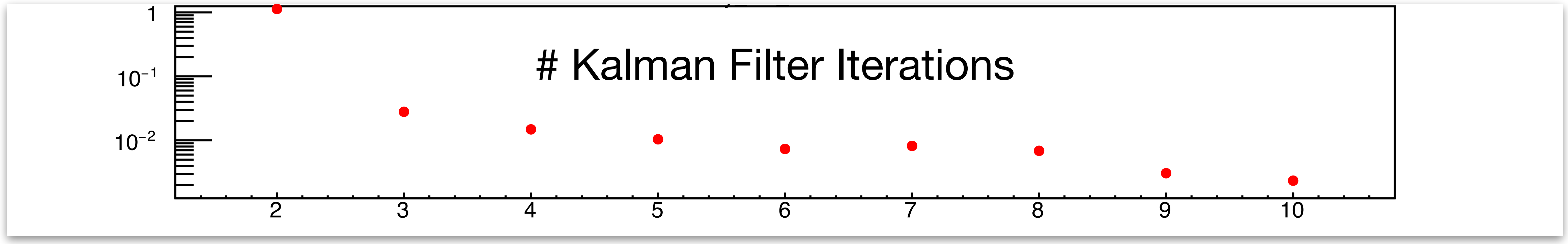
Data Processing

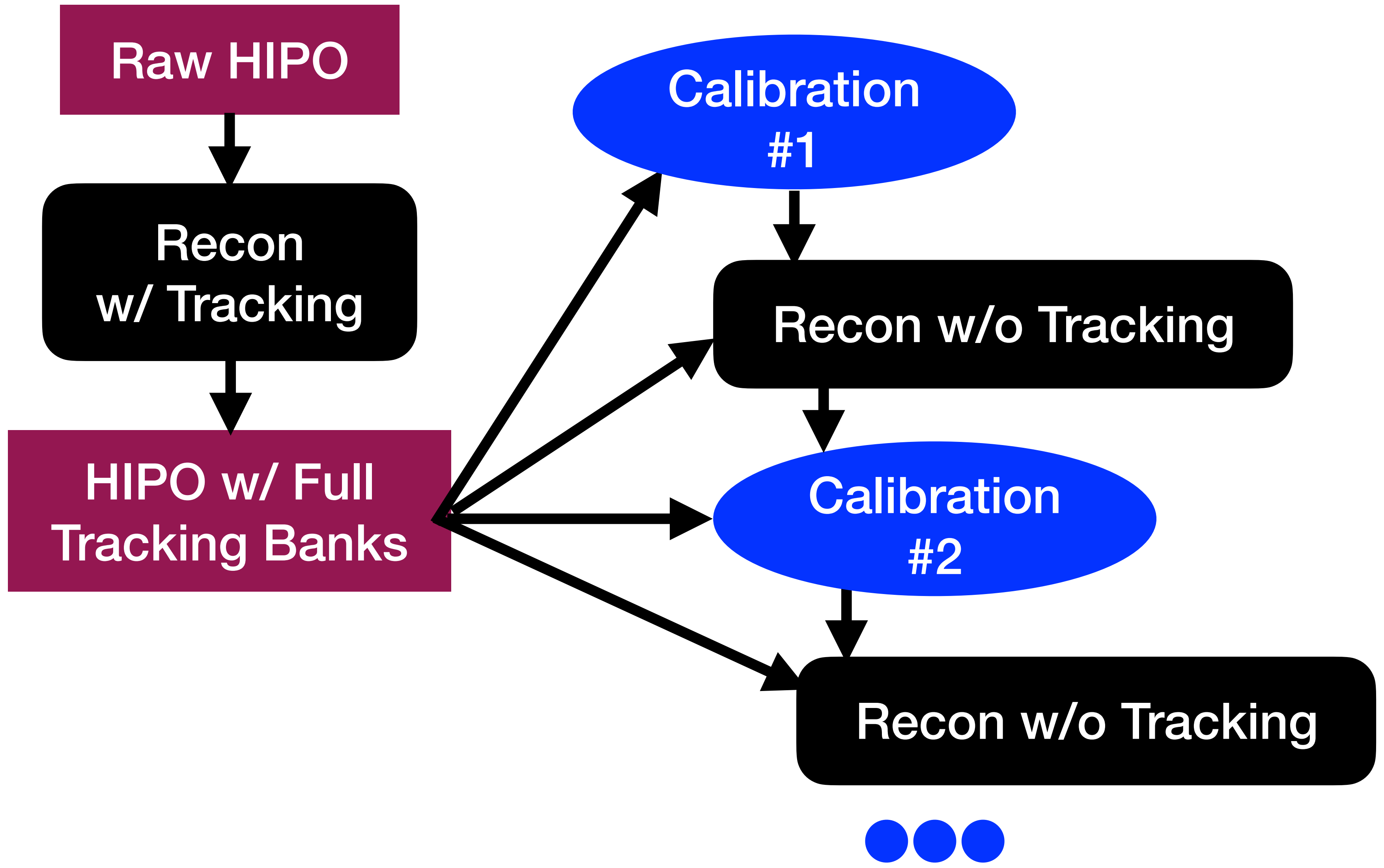
Software Related Speedups

- There may be improvements possible in tracking algorithms that will give significant speed improvements (and tracking is of course the large majority of CPU time)
    - Tongtong has been studying, for example, optimizing factors that contribute to the number of Kalman filter iterations in forward tracking, including adding and using more truth information from simulation for rigorous studies andvalidation
        - Seeding accuracy and convergence criteria, already achieved ~20% speedup
- Better leveraging of service-oriented architecture
    - We may be rerunning tracking many times during calibration phases, even though tracking doesn't change during many of them
- See also the AI/ML presentation



Seeding Accuracy (GeV)

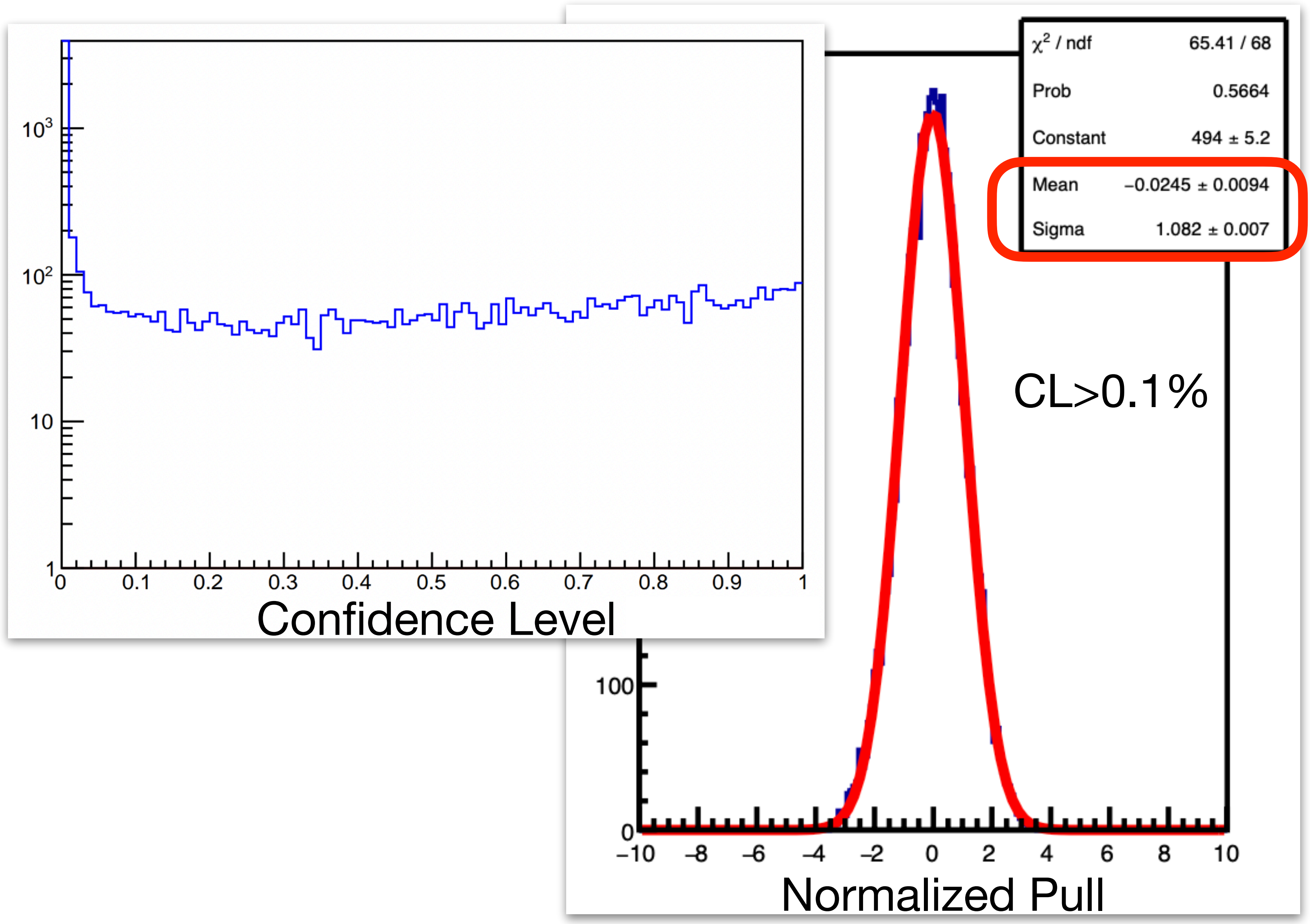


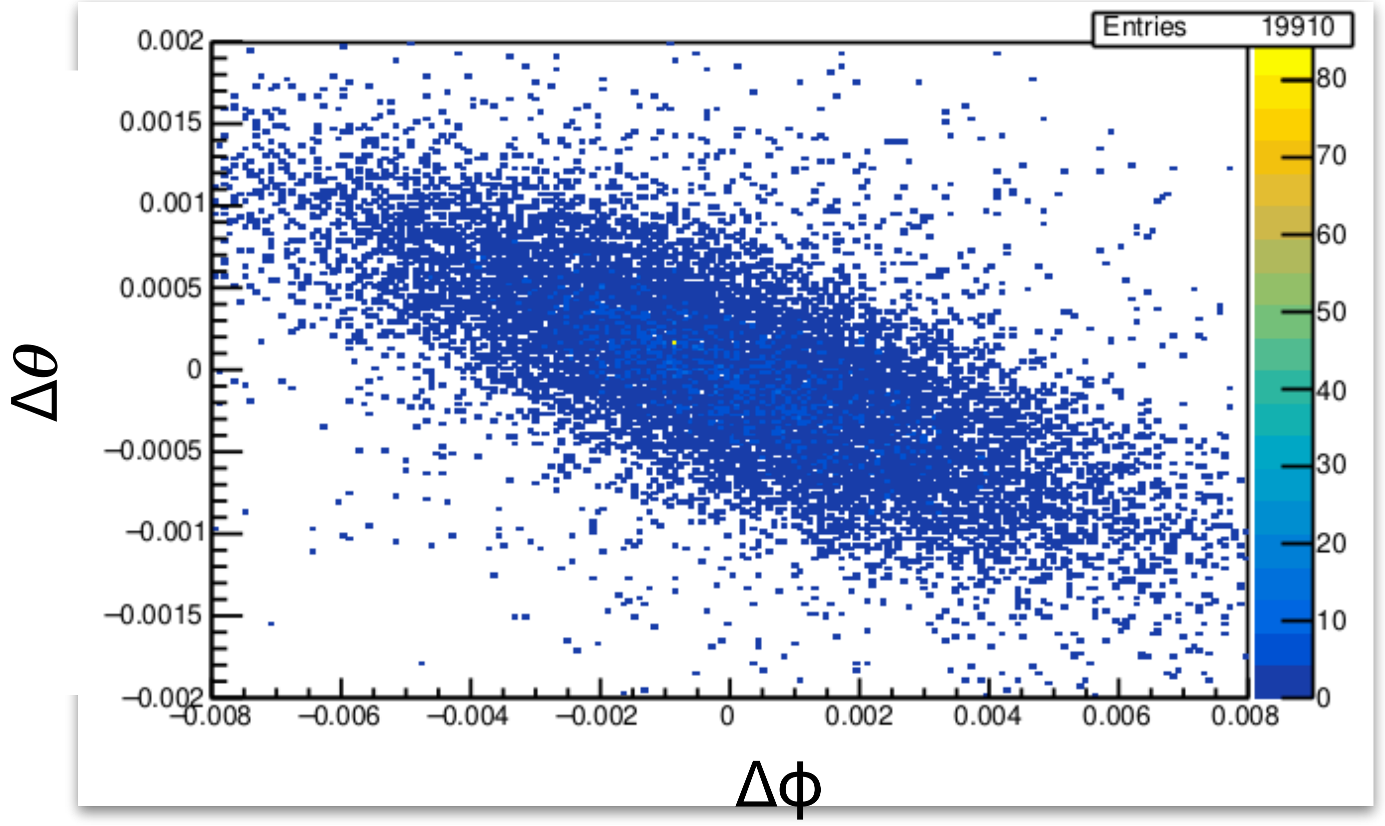


Physics Analysis

Kinematic Fitting

- Did a little survey of the available implementations, and picked one to start with, from Frank Cao, previously used for CLAS
    - minimal dependencies, very standalone, not attached to a larger framework, interface is just 4-vectors, only a few-hundred lines of C++, uses standard matrix/linear-algebra libraries
    - if we need to rewrite it in another language later, not a big deal
- We want it to be runnable standalone, ultimately HIPO in and HIPO out and pluggable inside standard workflows. Started by adding and testing the various types of constraints needed for CLAS12 analyses and testing with toy MC.
- For pass2 software the DC covariance matrix is very far from usable
    - But we're currently seeing if we can model the covariance matrix with simulation, and then massage it to work with real data
    - Still in early phases, working through some unexpected correlations, but looks promising
- Trevor Reed and Pierre Chatagnon have been working on this





AI/ML

See Gagik's and Will's Presentations!

- Meanwhile, here's some applications under consideration ...
    - L3 trigger (speed, high luminosity, flexibility)
    - CVT track finding (speed, high luminosity)
    - Kalman filter initialization (speed)
    - RICH (complexity)
    - Calorimeter simulation (speed)
    - etc ...

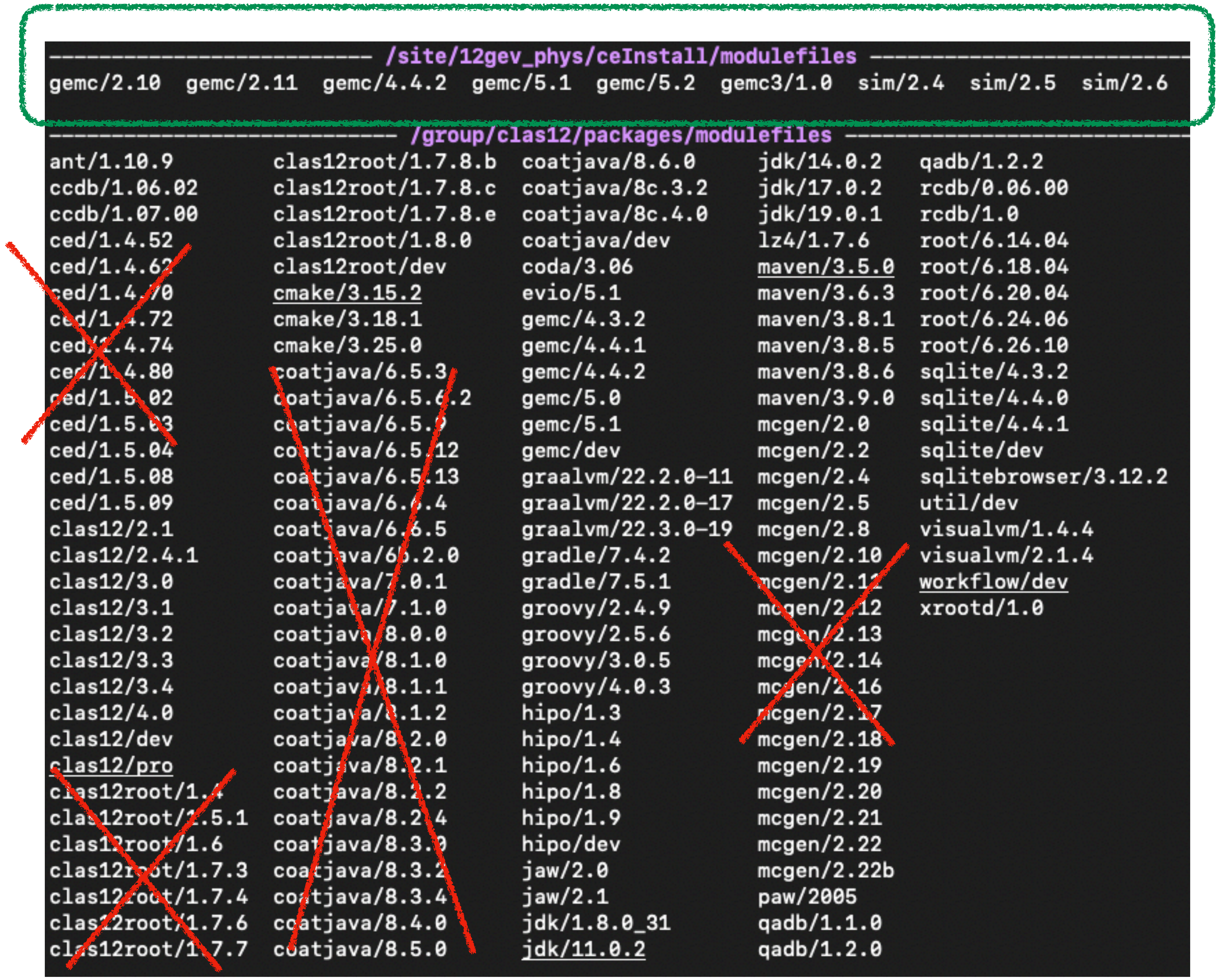
Finally, some administrativy stuff ...

Not-so Interesting Stuff ...

CLAS12 software builds/environment at JLab

- This week's reorganization, and upgrade of `clas12/pro`, was postponed to coincide with the COATJAVA pass2 release, next week🤞
    - See announcement and details on forum and mailing list:
    - https://clas12.discourse.group/t/prep-for-new-uber-release-envionrment-module-cleanup-and-reorg/640/2
    - https://mailman.jlab.org/pipermail/clas12_software/2023-March/002826.html

1. The default `clas12/pro` uber module will point to latest versions of everything, currently usable/testable at `clas12/dev`
2. The available list of tons of old versions will be minimized. Their software builds will still be there, just not accessible via modules, but that can easily and quickly be reverted on demand.
3. The modules for GEMC will appear under `/site/12gev_phys` and maintained by Mauri, same as the GEMC software and all its dependencies. Transparent to the user, except appearances.



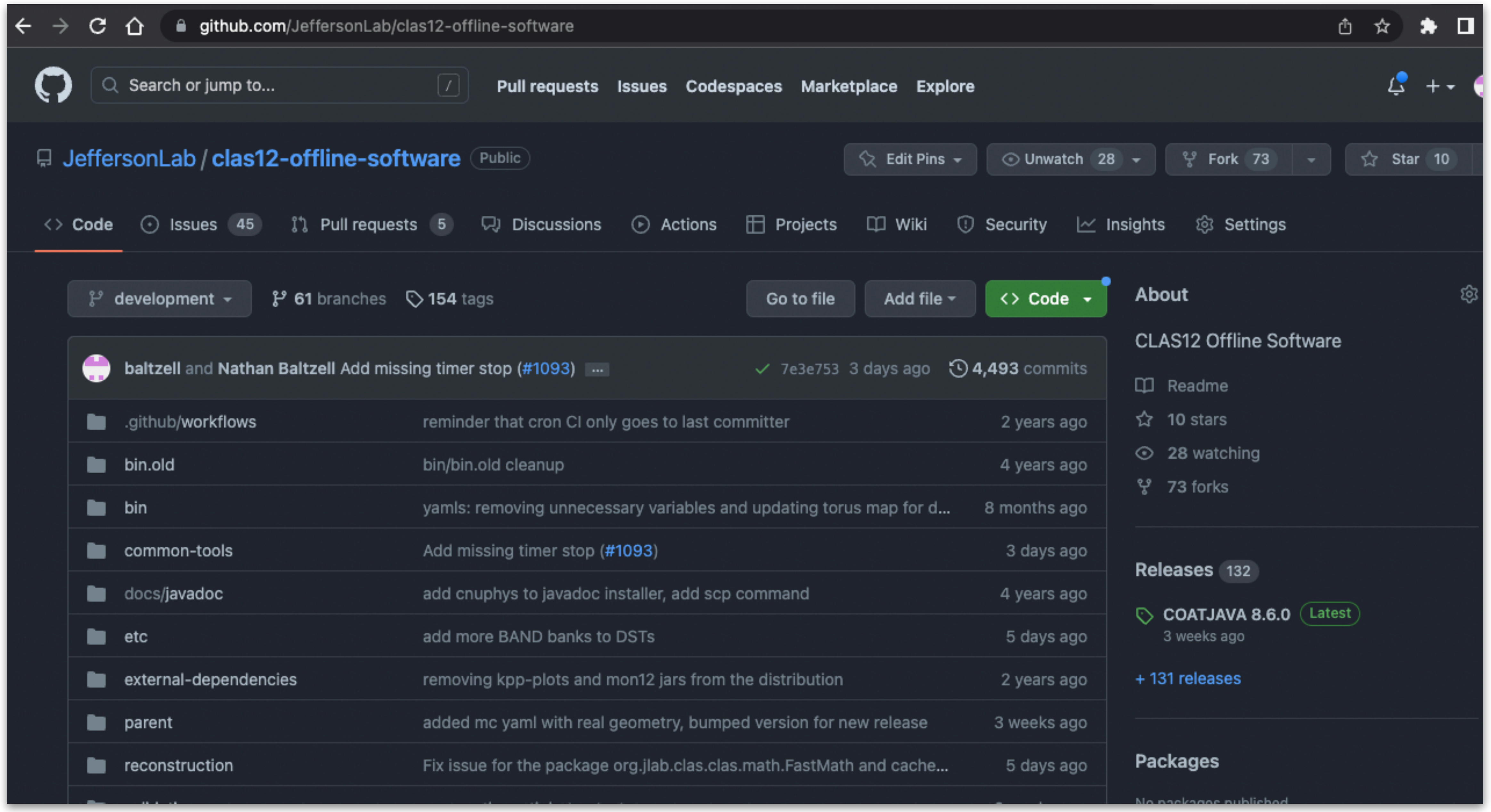
```
---------------------------- /site/12gev_phys/ceInstall/modulefiles ----------------------------
gemc/2.10  gemc/2.11  gemc/4.4.2  gemc/5.1  gemc/5.2  gemc3/1.0  sim/2.4  sim/2.5  sim/2.6

----------------------------- /group/clas12/packages/modulefiles -----------------------------
ant/1.10.9         clas12root/1.7.8.b  coatjava/8.6.0       jdk/14.0.2   qadb/1.2.2
ccdb/1.06.02       clas12root/1.7.8.c  coatjava/8c.3.2      jdk/17.0.2   rcdb/0.06.00
ccdb/1.07.00       clas12root/1.7.8.e  coatjava/8c.4.0      jdk/19.0.1   rcdb/1.0
ced/1.4.52         clas12root/1.8.0    coatjava/dev         lz4/1.7.6    root/6.14.04
ced/1.4.62         clas12root/dev      coda/3.06            maven/3.5.0  root/6.18.04
ced/1.4.70         cmake/3.15.2        evio/5.1             maven/3.6.3  root/6.20.04
ced/1.4.72         cmake/3.18.1        gemc/4.3.2           maven/3.8.1  root/6.24.06
ced/1.4.74         cmake/3.25.0        gemc/4.4.1           maven/3.8.5  root/6.26.10
ced/1.4.80         coatjava/6.5.3      gemc/4.4.2           maven/3.8.6  sqlite/4.3.2
ced/1.5.02         coatjava/6.5.6.2    gemc/5.0             maven/3.9.0  sqlite/4.4.0
ced/1.5.03         coatjava/6.5.9      gemc/5.1             mcgen/2.0    sqlite/4.4.1
ced/1.5.04         coatjava/6.5.12     gemc/dev             mcgen/2.2    sqlite/dev
ced/1.5.08         coatjava/6.5.13     graalvm/22.2.0-11    mcgen/2.4    sqlitebrowser/3.12.2
ced/1.5.09         coatjava/6.6.4      graalvm/22.2.0-17    mcgen/2.5    util/dev
clas12/2.1         coatjava/6.6.5      graalvm/22.3.0-19    mcgen/2.8    visualvm/1.4.4
clas12/2.4.1       coatjava/6b.2.0     gradle/7.4.2         mcgen/2.10   visualvm/2.1.4
clas12/3.0         coatjava/7.0.1      gradle/7.5.1         mcgen/2.11   workflow/dev
clas12/3.1         coatjava/7.1.0      groovy/2.4.9         mcgen/2.12   xrootd/1.0
clas12/3.2         coatjava/8.0.0      groovy/2.5.6         mcgen/2.13
clas12/3.3         coatjava/8.1.0      groovy/3.0.5         mcgen/2.14
clas12/3.4         coatjava/8.1.1      groovy/4.0.3         mcgen/2.16
clas12/4.0         coatjava/8.1.2      hipo/1.3             mcgen/2.17
clas12/dev         coatjava/8.2.0      hipo/1.4             mcgen/2.18
clas12/pro         coatjava/8.2.1      hipo/1.6             mcgen/2.19
clas12root/1.4     coatjava/8.2.2      hipo/1.8             mcgen/2.20
clas12root/1.5.1   coatjava/8.2.4      hipo/1.9             mcgen/2.21
clas12root/1.6     coatjava/8.3.0      hipo/dev             mcgen/2.22
clas12root/1.7.3   coatjava/8.3.2      jaw/2.0              mcgen/2.22b
clas12root/1.7.4   coatjava/8.3.4      jaw/2.1              paw/2005
clas12root/1.7.6   coatjava/8.4.0      jdk/1.8.0_31         qadb/1.1.0
clas12root/1.7.7   coatjava/8.5.0      jdk/11.0.2           qadb/1.2.0
```

Not-so Interesting Stuff ...

COATJAVA Repository Cleanup

- This one:
    - https://github.com/JeffersonLab/clas12-offline-software
- Proposed a couple months ago, waiting on pass2 software release, next week!🤞
    - https://clas12.discourse.group/t/coatjava-git-repo-cleanup-proposal/633

1. Mark it "archival". Means read-only, no commits allowed, no pull requests accepted, just leave it there forever.
2. Switch to a new, cleaned up one, ~10x faster to clone/copy, and with a more intuitive name, "coatjava"!
    - *Big caveat: outstanding forks and local copies of the old repository will become unmergeable via standard git commands and have to be done manually. So get your changes in on a branch now!*
    - That's because the cleanup is rewriting the history, removing stuff that should never have been committed to the main repository, e.g. big data files and old, large commits that are no longer reachable in any branch.



baltzell and Nathan Baltzell Add missing timer stop (#1093) ... ✓ 7e3e753 3 days ago 4,493 commits

.github/workflows	reminder that cron CI only goes to last committer	2 years ago
bin.old	bin/bin.old cleanup	4 years ago
bin	yamls: removing unnecessary variables and updating torus map for d...	8 months ago
common-tools	Add missing timer stop (#1093)	3 days ago
docs/javadoc	add cnuphys to javadoc installer, add scp command	4 years ago
etc	add more BAND banks to DSTs	5 days ago
external-dependencies	removing kpp-plots and mon12 jars from the distribution	2 years ago
parent	added mc yaml with real geometry, bumped version for new release	3 weeks ago
reconstruction	Fix issue for the package org.jlab.clas.clas.math.FastMath and cache...	5 days ago

Summary

- Lots of software projects worth pursuing that would benefit the CLAS12 physics program, more than there was time to mention
- Prioritization will be important, opinions and contributions welcome
- Didn't mention much on end-user analysis in this presentation, in large part because ...
  - *Hall B is in the process of hiring a staff scientist position with a focus on CLAS collaboration physics analysis, supporting, standardizing*