Hall A Analysis Software Status & Plans

Ole Hansen

Hall A Collaboration Meeting
January 25, 2018

"Podd" Analyzer: Status

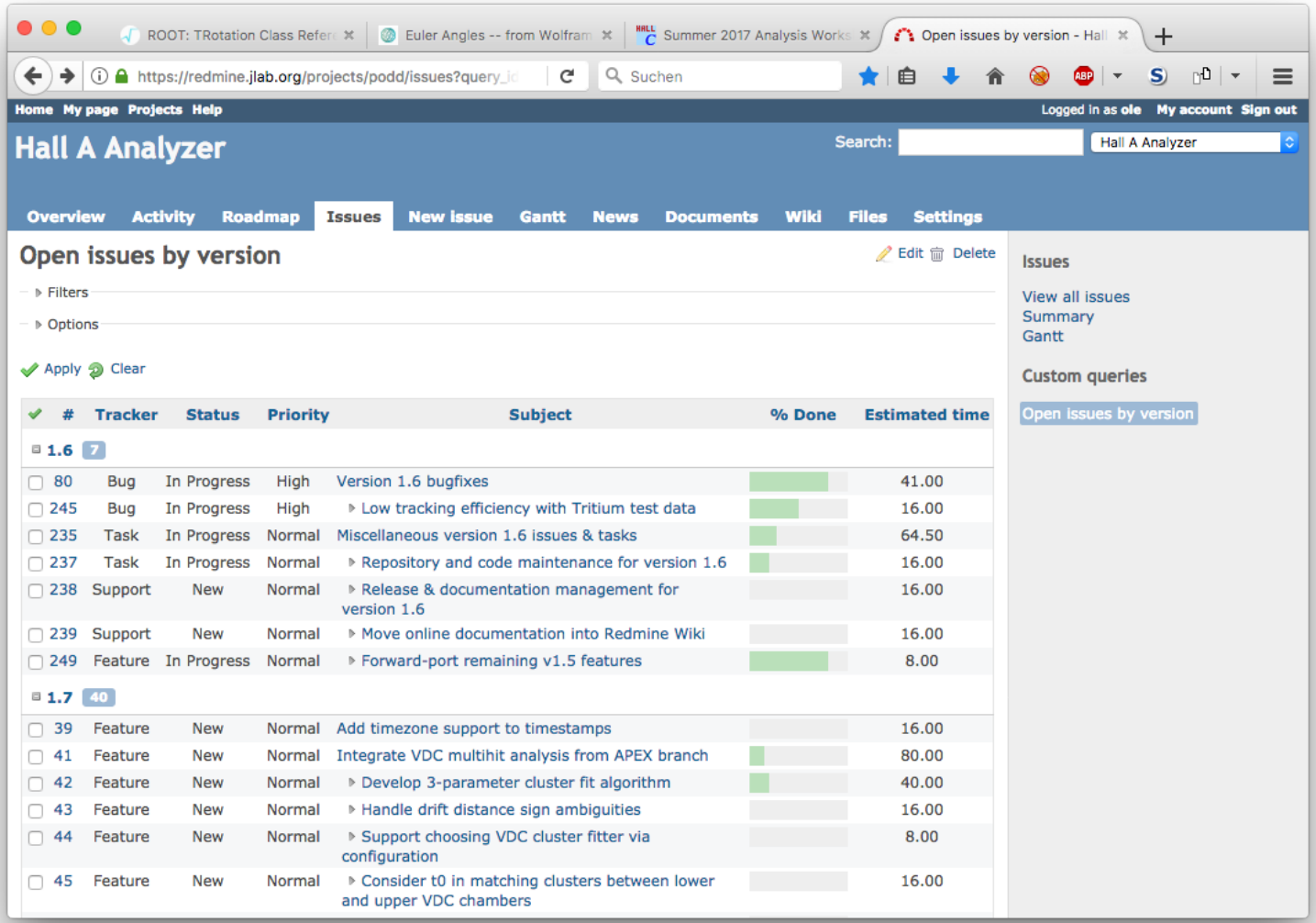
- Latest release: 1.6.0-rc2 (24 Jan 2018) ▸ web
  - Used by Tritium experiments. (Many results shown in this meeting were obtained with analyzer 1.6-beta/rc1.)
  - Feature list frozen. Final testing & documentation in progress. Should be "official" very shortly.
  - Preliminary Release Notes ▸ web
- Legacy version: 1.5.37 (3 Mar 2017) ▸ web
  - Lots of small improvements in late 2016/early 2017
  - Updated VDC tracking algorithm coming in 1.5.38 (ready)
  - Will still be supported for a while (new ROOT versions etc.)
- Future versions will have a much shorter release cycle.

Recently added/completed in 1.6

- Better geometry handling
  - Can define arbitrary detector misalignments and rotations
  - Consistent coordinate systems
  - Working detector "active area" test
- Build system
  - All components can be built with `SCons`, including SDK & utilities
  - Better detection/auto installation of EVIO
  - Support for `SCons` 3/Python 3
- `OldVDC` library
  - Makes v1.5 VDC algorithm available for tracking comparisons
  - Reads new-style database files
- Output and cut definition files support `#include` directives
  - Much easier management of definitions per detector/per spectrometer
  - Heavily used by Hall C (Eric Pooser's talk Monday)
- Bugfixes, tweaks

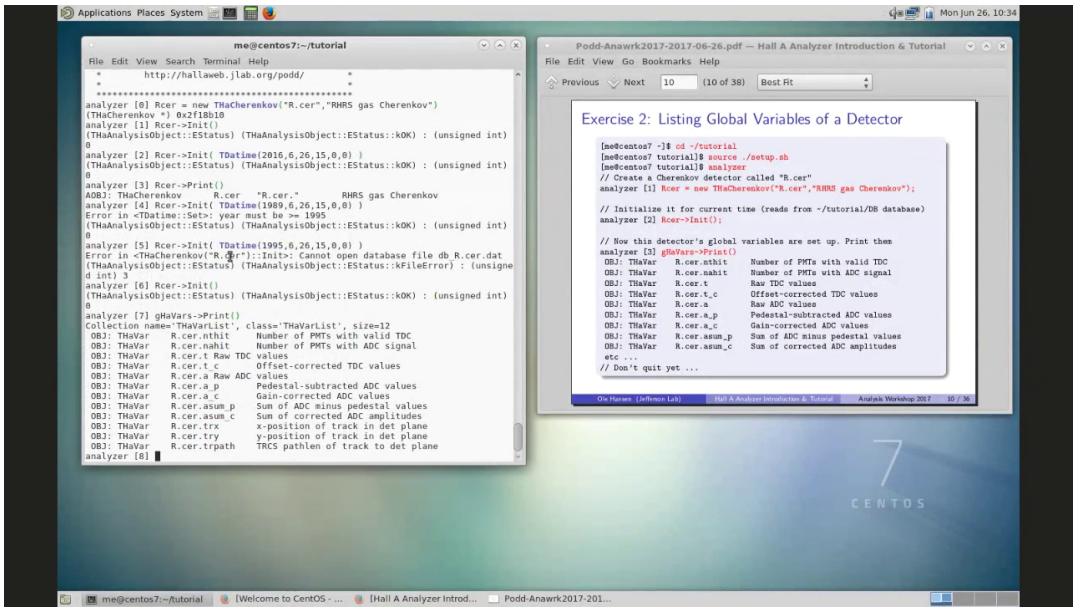
Redmine Tracker

- May not be the best project tool, but (sort of) supported at JLab
- Feature-rich Wiki. Planning to move Podd documentation there



Analysis Workshop 2017

- Successful joint Hall A & C analysis workshop June 26–27, 2017
- `http://hallaweb.jlab.org/data_reduc/AnaWork2017/`
- Live hands-on tutorials, using preconfigured virtual machine environment (still available from the link above)
- Simulation, calibration, on- & offline data analysis, ROOT tips & tricks
- Video recordings available at `https://bluejeans.com/s/IEjGT` (quality varies, recommend to download clips)
- Some 50 local participants, 10–20 remote, many students
- Plan to repeat in summer 2018



Resources

- Web site ▸ home page
  - Documentation
  - Release Notes
  - Software Development Kit (SDK)
  - Source code downloads
  - Archived tutorials & example replays
- Bug trackers ▸ GitHub ▸ JLab Redmine
- Mailing list: `halla_software@jlab.org`. Subscribe on ▸ mailman
- Analysis Workshop archive ▸ archive (includes tutorials)
- (Weekly meetings currently on hold)

VDC Algorithm Improvements

Version 1.5.38

- Disallow UV ambiguities (configurable)
- UV fiducial cut
- Proper lower-upper matching cut
- Disallow cluster sharing

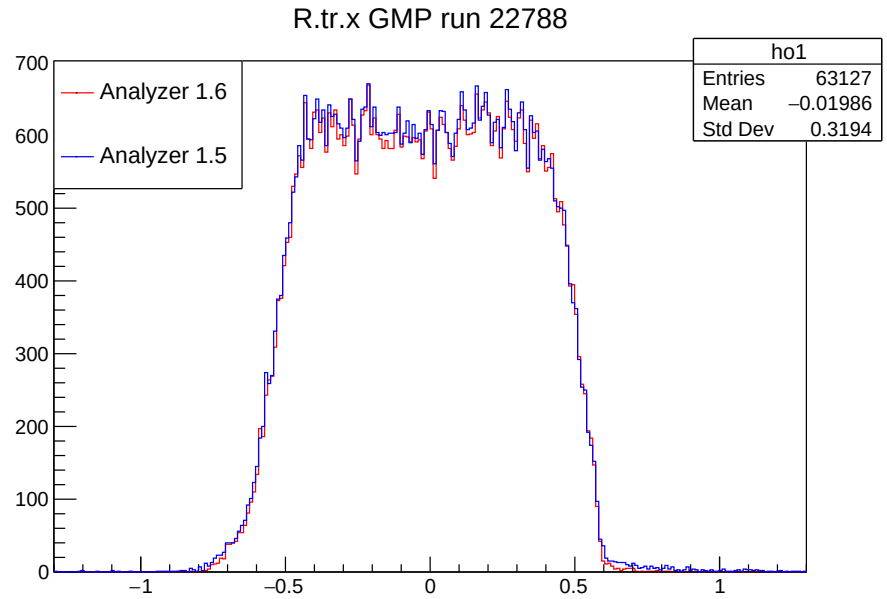
→ Guarantees clean single track at expense of slightly lower tracking efficiency.
Multi-cluster analysis optional.

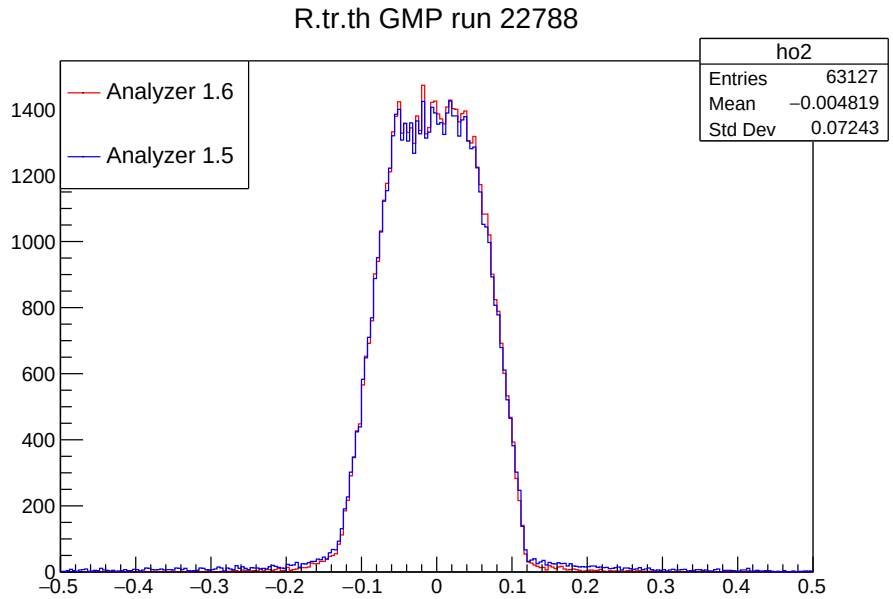
Version 1.6

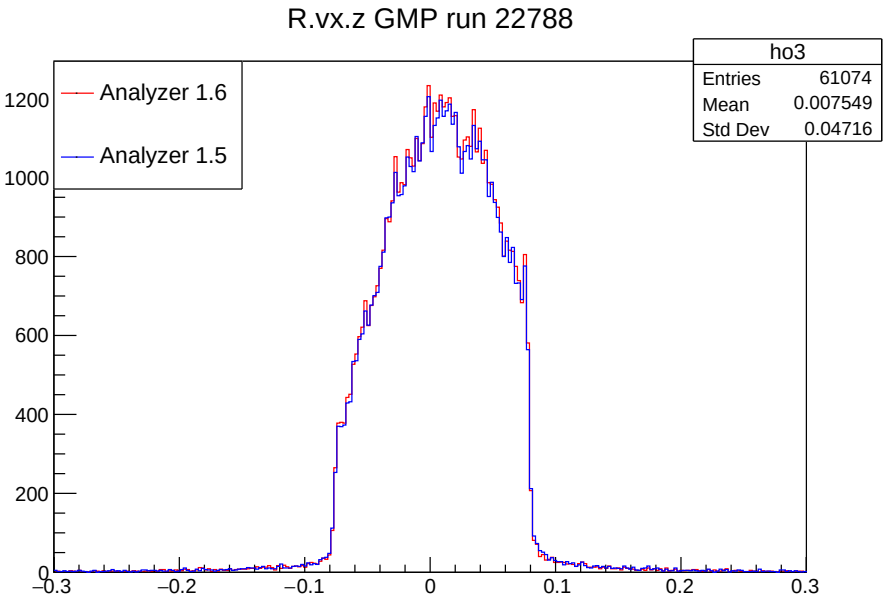
- Cluster shape analysis
- Overlapping cluster splitting
- UV fiducial cut
- Proper lower-upper matching cut
- Disallow cluster sharing
- Old VDC code for reference

→ Allows multiple tracks, improves tracking efficiency, high-rate capable

VDC Tracking Comparison v1.5.37 vs. v1.6.0-rc2







Improved algorithm:

- Fewer reconstruction errors: smaller tails, larger signal
- No need for explicit cuts on “clean” events
- Should be particularly helpful with high-rate/high-noise data like APEX’s (to be tested)

VDC Algorithm Improvement (II)

Significantly better ghost track suppression while maintaining efficiency

v1.5.37 vs. v1.6.0-rc2 statistics

```
Cut summary:
Name              Def                                Called     Passed
------------------------------------------------------------------------
BLOCK: Reconstruct
MultiClusterOld   R0.vdc.u1.nclust>1||
                  R0.vdc.v1.nclust>1||
                  R0.vdc.u2.nclust>1||
                  R0.vdc.v2.nclust>1                 891754     43979   (4.93%)
MultiClusterNew   R.vdc.u1.nclust>1||
                  R.vdc.v1.nclust>1||
                  R.vdc.u2.nclust>1||
                  R.vdc.v2.nclust>1                  891754     40245   (4.51%)

MultiTrackOld     R0.tr.n>1                          891754     4816    (0.54%)
MultiTrackNew     R.tr.n>1                           891754     1663    (0.186%)
...
HaveTrackOld      R0.tr.n>0&&GoodVertexOld           891754     274613 (30.8%)
HaveTrackNew      R.tr.n>0&&GoodVertexNew            891754     275169 (30.9%)
```

Possible Upgrade Plans

- Many ideas for improvements
  - APEX tracking
  - Database interface
  - Output system
  - Code reorganization
  - Multithreading
- Just see the Redmine task list for version 1.7 for a subset

Output Improvement

Currently:

- Only D data type
- Multiple, redundant array size variables with parallel arrays

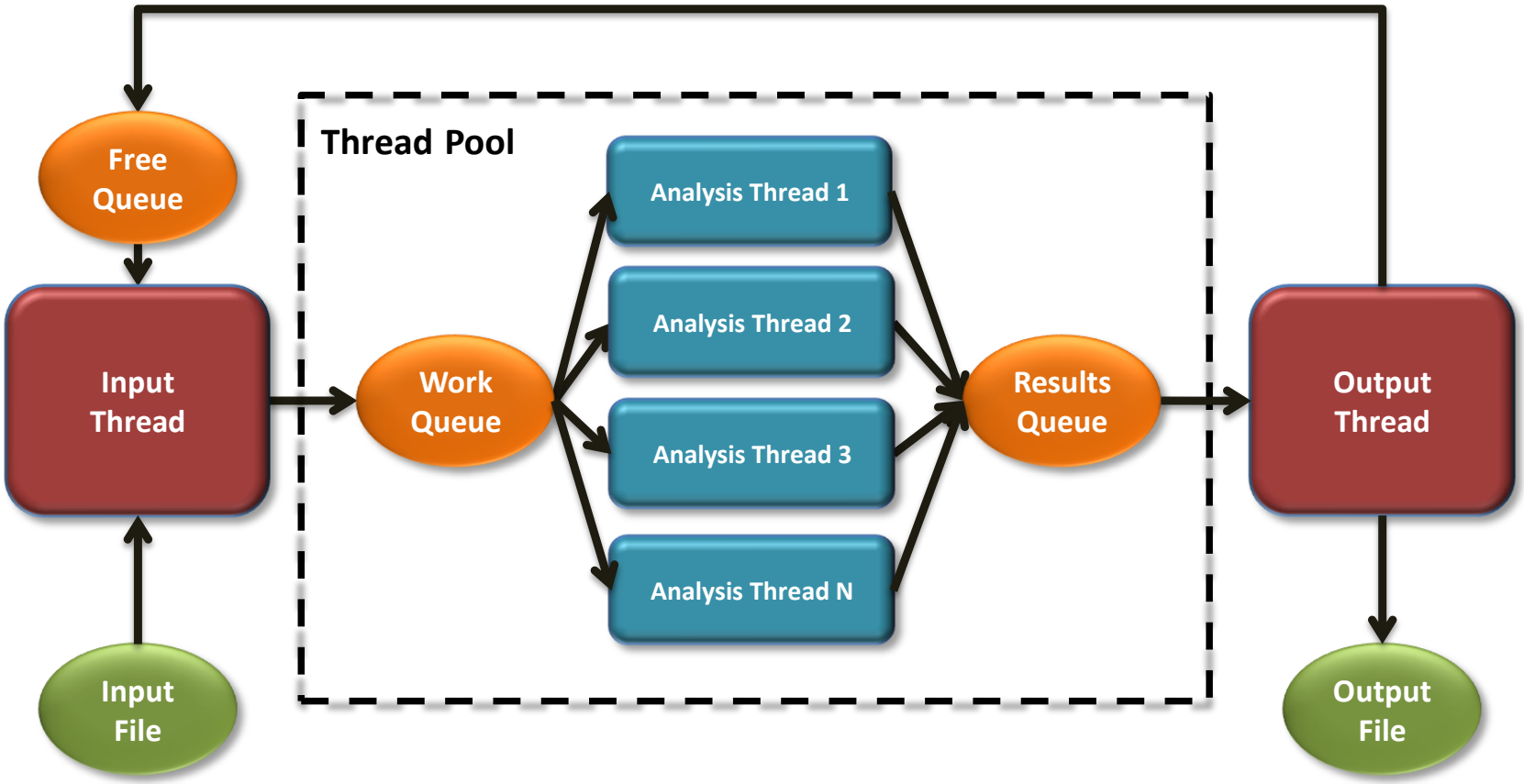
```
*Br   15 :L.vdc.u1.time : data[Ndata.L.vdc.u1.time]/D                   *
*Entries :   265460 : Total  Size=   11970559 bytes  File Size  =    3662638 *
*Baskets :       81 : Basket Size=    2683904 bytes  Compression=   3.27     *
*.....*
*Br   16 :L.vdc.u1.trdist : data[Ndata.L.vdc.u1.trdist]/D               *
*Entries :   265460 : Total  Size=   11970731 bytes  File Size  =   10832974 *
*Baskets :       81 : Basket Size=    2683904 bytes  Compression=   1.10     *
*.....*
*Br   17 :L.vdc.u1.wire : data[Ndata.L.vdc.u1.wire]/D                   *
*Entries :   265460 : Total  Size=   11970559 bytes  File Size  =    2295168 *
*Baskets :       81 : Basket Size=    2683904 bytes  Compression=   5.21     *
*.....*
*Br   18 :L.vdc.u1.nclust : L.vdc.u1.nclust/D                           *
*Entries :   265460 : Total  Size=    2136991 bytes  File Size  =      45063 *
*Baskets :      131 : Basket Size=      64512 bytes  Compression=  47.36     *
*.....*
*Br   19 :L.vdc.u1.nhit : L.vdc.u1.nhit/D                               *
*Entries :   265460 : Total  Size=    2136721 bytes  File Size  =     184378 *
*Baskets :      131 : Basket Size=      64512 bytes  Compression=  11.57     *
*.....*
analyzer [2]
```

Improved:

- Full range of data types
- Automatic detection of parallel arrays, only one size variable (with limitations)
- Automatic basket size adjustment (overriding ROOT default)

```
analyzer [6] T->Print("L.vdc.*")
******************************************************************************
*Tree    :T         : Hall A Analyzer Output DST                           *
*Entries :   280231 : Total =       325369054 bytes  File  Size =  186808850 *
*       :          : Tree compression factor =   1.74                     *
******************************************************************************
*Br    0 :L.vdc.u1.nhit : L.vdc.u1.nhit/I                                 *
*Entries :   280231 : Total  Size=    1122414 bytes  File Size  =     160443 *
*Baskets :       12 : Basket Size=     537088 bytes  Compression=   6.99     *
*.....*
*Br    1 :L.vdc.u1.wire : L.vdc.u1.wire[Ndata.L.vdc.u1.wire]/I             *
*Entries :   280231 : Total  Size=    7420715 bytes  File Size  =    2064016 *
*Baskets :       49 : Basket Size=    1556480 bytes  Compression=   3.59     *
*.....*
*Br    2 :L.vdc.u1.rawtime : L.vdc.u1.rawtime[Ndata.L.vdc.u1.wire]/I       *
*Entries :   280231 : Total  Size=    7420874 bytes  File Size  =    3614951 *
*Baskets :       49 : Basket Size=    1556992 bytes  Compression=   2.05     *
*.....*
*Br    3 :L.vdc.u1.time : L.vdc.u1.time[Ndata.L.vdc.u1.wire]/D             *
*Entries :   280231 : Total  Size=   13718165 bytes  File Size  =    4116377 *
*Baskets :       80 : Basket Size=    2575872 bytes  Compression=   3.33     *
*.....*
*Br    4 :L.vdc.u1.dist : L.vdc.u1.dist[Ndata.L.vdc.u1.wire]/D             *
*Entries :   280231 : Total  Size=   13718165 bytes  File Size  =   12032644 *
*Baskets :       80 : Basket Size=    2575872 bytes  Compression=   1.14     *
*.....*
```

Parallelization



- Thread Pool with three thread-safe queues
- Queues hold working sets: event object, analysis chain & modules
- Option to sync event stream at certain events (*e.g.* scaler events, run boundaries)
- Option to preserve strict event ordering (at a performance penalty)

Discussion: Determining Needs

- Podd has design limitations: single-pass analysis, no object output, difficult to parallelize ...
- Other collaborations (*e.g.* FAIR, ALICE) have thrown away similar designs in favor of more modern approaches (distributed computing). But they have more demanding requirements.
- Is continuing active development of Podd generic features (infrastructure aspects) a wise use of manpower?
- Which features (infrastructure and/or algorithms) are really needed?
  $\rightarrow$ user feedback would help
- Which capabilities will future projects need?
  - Remaining HRS experiments will be OK
  - Hall C—probably OK with current Podd
  - SBS (seem to like Podd)
  - Møller (may lean toward rolling their own)
  - SoLID (open)
  - Overall lab strategy? EIC?

Discussion: Future Hall A Software Direction

Options

1. Shelf infrastructure upgrades. Focus on algorithms as needed (SBS, SoLID).
2. Make the best of Podd. It may even handle SoLID.
3. Roll something new ourselves. Unwise.
4. Join the club: JANA (GlueX). Seems to enjoy management support.
5. Look elsewhere: *e.g. art* (FNAL). Technically superior, wide adoption, *e.g.* DUNE

Caveats

- We have very little manpower. We can only do *one* of the above properly
- Software development timescales are long; should plan now for 10 years out