Analysis Software Update

Ole Hansen

Jefferson Lab

Hall A Collaboration Meeting
December 9, 2014
Hall A Analysis Framework (“C++ Analyzer”, “Podd”)

- Class library on top of ROOT
- Highly modular: “Everything is a plug-in”
- Software for non-standard equipment typically provided by users
- In production use since 2003
- Code for all 6 GeV-era equipment exists and is well tested
- Currently being adopted by Hall C
Current Version: Podd 1.5.27

Version 1.5 is in maintenance mode, i.e. we only apply backward-compatible bugfixes & small improvements

- Updates in 2014
  - Updated CAEN 1190 decoder (Brad Sawatzky)
  - Tweaks for Hall C (Steve Wood)
  - Updated SDK
  - Minor bugfixes
  - Support for latest ROOT, Linux and compiler versions (ROOT 6 not yet supported)
  - 1.5.28: New raster decoding (Luke Myers)

- Download/documentation: http://hallaweb.jlab.org/podd/
- Source code repository: https://github.com/JeffersonLab/analyzer.git
- Bug tracking using GitHub's issue tracker
Open Software Issues

Core Analyzer

- Decoders for pipelined readout modules: missing
- Parallel processing on multi-core systems: not directly supported
- Scalability to very large data sets: to be tested

Specific experiments

- $G^p_M$: FPP tracker plane (optional, see yesterday’s $G^p_M$ talk)
- APEX: High-rate VDC track reconstruction
- SBS: GEM tracker & calorimeter reconstruction
- SBS GEp(5): Recoil polarimetry, kinematic correlation analysis
- SBS SIDIS: RICH analysis & PID
- Møller, SoLID: to be determined
## Reconstruction Software Status & Tasks

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Base Software</th>
<th>Required Extensions</th>
<th>Status / Required By</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMp</td>
<td>C++ Analyzer</td>
<td>(FPP tracker integration)</td>
<td>Spring 2015</td>
</tr>
<tr>
<td>DVCS</td>
<td>C++ Analyzer</td>
<td>Photon detector analysis</td>
<td>Done</td>
</tr>
<tr>
<td>$^3$H/$^3$He</td>
<td>C++ Analyzer</td>
<td>BigBite MWDC track reconstruction</td>
<td>Done</td>
</tr>
<tr>
<td>$A_1^n$</td>
<td>C++ Analyzer</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APEX</td>
<td>C++ Analyzer</td>
<td>High-rate VDC track reconstruction</td>
<td>Spring 2016</td>
</tr>
<tr>
<td>PREX/CREX</td>
<td>PAN &amp; C++ Analyzer</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
| SBS general | C++ Analyzer  | • Analyzer parallelization  
• Pipelined electronics decoder  
• GEM track reconstruction  
• BigBite GEM/MWDC tracking  
• Calorimeter cluster reconstruction | ≥ Fall 2017 |
| SBS GEp(5) | C++ Analyzer  | • Recoil polarimetry  
• Kinematic correlation analysis | ≥ 2018? |
| SBS SIDIS  | C++ Analyzer  | • RICH analysis & PID | ≥ late 2018? |

**Red:** not yet written  
**Purple:** exists, but incomplete and/or not yet fully tested/integrated
Decoders

- Object-oriented decoder framework (Bob Michaels)
  - Rewrite of existing THaEvData, THaCodaDecoder
  - Will allow extending decoding capabilities via plug-ins
  - Also, will allow flexible (non-hardcoded) handling of DAQ event types
  - Essentially finished, under testing. ETA Spring 2015.

- Pipelined electronics support
  - Under development: Bob Michaels, Dasuni Adikaram (new Hall A postdoc), Brad Sawatzky
  - Experimental decoders exist (e.g. FADC250)
  - Event “unblocking” to be implemented. Testing/development underway (Bryan Moffit, Dasuni)
  - ETA: sometime in 2015
Analyzer Parallelization

- Goal: provide automatic event-level parallelization of any replay
- Ideally, should be as transparent as possible to user code, i.e. no or only minimal code modifications necessary
- Initial code review shows that
  - Multi-threading of existing THaAnalyzer is possible and relatively easy
  - Required user code modifications:
    - Replace globals like gHaVars with per-thread static members like THaAnalysisObject::fgVars
    - Provide some form of virtual copy method (to be specified)
  - Most of the work will have to be done in the output module THaOutput. Could become a bottleneck due to slowness of ROOT file output.
- Significant project. Need about 3 months of (preferably uninterrupted) time
- ETA: hopefully in 2015
VDC Tracking Improvements I: Multi-Cluster Events

- At **low rates**, multiple clusters with \( t_0 \approx 0 \) may occur, often in close proximity to each other.
- Long-standing problem. Probably caused by delta electrons.
- Causes \( u-\nu \) matching ambiguity.
- Cannot be fully resolved in software only, using only VDC data.
- 3rd wire direction (expensive) or 3rd tracker plane (less effective) may help.
  → upcoming \( G_M^p \) experiment will test.
- Fortunately, typically less than 10% of events are affected.
Accidentals occur at high singles rates (MHz)

Cause multiple cluster topology as in previous case

Can be largely resolved in software only by performing non-linear 3-parameter fit to cluster TDC values to extract track time offset $t_0$

$\approx \times 10$–20 background rejection with APEX test data

Could likely be further improved with 3rd tracker plane, as in previous case

Prototype analysis code written in 2010, improved version in 2014. To be tested & integrated.
A thorough code review in late 2013 revealed several bugs in the handling of multi-cluster VDC events. We concluded:

The current VDC tracking algorithm is definitely broken for events with multiple clusters in more than one plane. Such events should be rejected in any analysis using the present code.

Since then

- Bugs have been fixed in development branch. To be included in next analyzer release
- Extensive additional VDC analysis code has been developed (improved cluster fitting, cluster splitting, global optimization)
- About 80% finished. To be tested.
Generic Simulation Decoder Framework

- Derived decoder class **Podd::SimDecoder**, designed for simulation data
- Provides generic access to **MC truth data** (tracks, hits), directly and via global variables
- Allows detailed studies of reconstruction performance, reasons for reconstruction failures, background contamination of measured signals etc.
- Doesn’t come for free: user must implement actual interface to simulation data & fill data structures. Reconstruction code usually needs to be modified as well.
Database API

- Retain 1.5 API in v1.6+ for backward compatibility
- Only minimal code changes required (see code snippets)
- v1.6+ API allows different backends, e.g.
  - Hall A-style flat files
  - Hall C-style parameter file
  - MySQL server
- Backend can be set and/or configured from replay script

Podd 1.5 Database Access

```cpp
UserDetector::ReadDatabase( const TDateTime& date ) {
  FILE* file = OpenFile( date );
  DBRequest request[] = {
    { "planeconfig", &planeconfig, kString },
    { "MCdata", &mc_data, kInt, 0, 1 },
    { 0 }
  };
  Int_t err = LoadDB( file, date, request, fPrefix );
  fclose(file);
};
```

Podd 1.6+ Database Access

```cpp
THaDatabase.C:
THaDB* gHaDB = new THaFileDB( DB_DIR ); // Default DB

UserDetector::ReadDatabase( const TTimeStamp& date ) {
  DBRequest request[] = {
    { "planeconfig", &planeconfig, kString },
    { "MCdata", &mc_data, kInt, 0, 1 },
    { 0 }
  };
  Int_t err = LoadDB( file, date, request, fPrefix );
  fclose(file);
};
```
Assorted Improvements

std::vector global variables

UserDetector.h:

```cpp
std::vector<float> fData;
```

```cpp
UserDetector::DefineVariables( EMode mode ) {
    RVarDef vars[] = {
        { "data", "Data values", "fData" }, // var-size array of floats
        { 0 }
    };
    return DefineVarsFromList( vars, mode );
};
```

Cuts defined on variable-size arrays

replay.cuts:
Block: RawDecode

<table>
<thead>
<tr>
<th>Condition</th>
<th>FullTrack</th>
<th>RawDecode_master</th>
</tr>
</thead>
</table>
| FullTrack                        | MC.tr.n==1&&MC.tr.planebits[0]==0xFF
| RawDecode_master                 | FullTrack   |
Next Release: Podd 1.6

- Object-oriented decoder ✓
- VDC bugfixes ✓
- Simulation decoder framework ✓
- Improved tests & formulas ✓
- `std::vector` global variables ✓
- Rewritten THaDecData & VDCEff modules ✓
- Abstract database interface ✓
- Time-zone safe `TTimeStamp` for time stamps
- Test & validation procedures ✓
- EVIO unbundled ✓
- Code split into core and hall-specific libraries

✓ done, ✓ partly done

ETA: Spring 2015
SBS Software

- Beginning to take shape
- Reconstruction software development started
  - Track reconstruction based on neural network algorithm & Kalman filter (INFN)
  - Calorimeter cluster finding (UConn)
  - RICH analysis (UConn)
- Simulations work ongoing (Seamus Riordan coordinating)
  - Meetings every two weeks
  - Active GitHub repository
  - Need to develop common interfaces, database
  - Digitization to be done
Collaboration with Hall C

- Hall C migrating to C++ analyzer based on Hall A framework
- Found that Hall C code seamlessly integrates into framework (only very minor changes needed)
- Common developments
  - SCons build system
  - Database interface (in progress)
  - Object-oriented decoder (in progress)
  - Code validation tools (in progress)
- Regular meeting every two weeks
- Collaboration on code via GitHub
January 2015 Analysis Workshop


- More detailed presentation of many topics of this talk as well as Hall C analyzer development

- Contributions welcome, including work on current experiments
Conclusions

- Core Hall A analysis software needs upgrades for post-2015 12 GeV experiments
- Work has been ongoing throughout 2014
- Additional manpower very desirable to keep schedule!
- Collaboration with Hall C has already proven to be very productive. Looking forward to continuing the good work.
- Release 1.6 planned for spring 2015