Analysis Software Status Report

Ole Hansen

Jefferson Lab

Hall A Data Analysis Workshop
December 14, 2009

Supported Podd Releases

- 2006 vintage: version 1.4.12 ▸ release notes
    - Stable production code
    - Still used by some older experiments
    - Contains backports of most version 1.5 bugfixes
- 2008 vintage: version 1.5.12 ▸ release notes
    - Stable production code
    - Used by current experiments (2008–)
    - Required for new BigBite tracking software ▸ web
    - Recommended for all new development

Home page: `http://hallaweb.jlab.org/root/`

Supported Podd Releases

- 2006 vintage: version 1.4.12 ▸ release notes
  - Stable production code
  - Still used by some older experiments
  - Contains backports of most version 1.5 bugfixes
- 2008 vintage: version 1.5.12 ▸ release notes
  - Stable production code
  - Used by current experiments (2008–)
  - Required for new BigBite tracking software ▸ web
  - Recommended for all new development

Home page: `http://hallaweb.jlab.org/root/`

Split Runs

- Problem: CODA continuation files do not have run info
- Old solution: Set run date/number etc. explicitly, or copy run object after init → cumbersome
- New solution: Automatically read run info from first segment. Now it “just works” (some limitations):

Split Run Example

```
THaRun* r1 = new THaRun( "/daq/data1/e01001_1000.dat.0" );
THaRun* r2 = new THaRun( "/daq/data2/e01001_1000.dat.1" );
analyzer->Process( r1 );
analyzer->Process( r2 );
```

- Could be futher improved (single object for group of runs)

Split Runs

- Problem: CODA continuation files do not have run info
- Old solution: Set run date/number etc. explicitly, or copy run object after init → cumbersome
- New solution: Automatically read run info from first segment. Now it “just works” (some limitations):

Split Run Example

```
THaRun* r1 = new THaRun( "/daq/data1/e01001_1000.dat.0" );
THaRun* r2 = new THaRun( "/daq/data2/e01001_1000.dat.1" );
analyzer->Process( r1 );
analyzer->Process( r2 );
```

- Could be futher improved (single object for group of runs)

Split Runs

- Problem: CODA continuation files do not have run info
- Old solution: Set run date/number etc. explicitly, or copy run object after init → cumbersome
- New solution: Automatically read run info from first segment. Now it “just works” (some limitations):

Split Run Example

```
THaRun* r1 = new THaRun( "/daq/data1/e01001_1000.dat.0" );
THaRun* r2 = new THaRun( "/daq/data2/e01001_1000.dat.1" );
analyzer->Process( r1 );
analyzer->Process( r2 );
```

- Could be futher improved (single object for group of runs)

Split Runs

- Problem: CODA continuation files do not have run info
- Old solution: Set run date/number etc. explicitly, or copy run object after init → cumbersome
- New solution: Automatically read run info from first segment. Now it “just works” (some limitations):

Split Run Example

```
THaRun* r1 = new THaRun( "/daq/data1/e01001_1000.dat.0" );
THaRun* r2 = new THaRun( "/daq/data2/e01001_1000.dat.1" );
analyzer->Process( r1 );
analyzer->Process( r2 );
```

- Could be futher improved (single object for group of runs)

Crate Map Selection

Select Crate Map File

```
THaAnalyzer* analyzer = new THaAnalyzer;
analyzer->SetCrateMapFileName( "cratemap_L" );  // default "cratemap"
```

This will look for "`db_cratemap_L.dat`" in the appropriate date-format directory under `$DB_DIR`.

HRS Track Sorting

HRS Track sorting by χ^2

```
THaHRS* HRSL = new THaHRS( "L", "Left HRS" );
HRSL->SetTrSorting(true);  // Default: false (off)
```

- Sort function: `THaTrack::Compare` ▸ doc
- (Re)defines meaning of HRS “golden track”

Thanks to Jin Huang

HRS Track Sorting

HRS Track sorting by χ^2

```
THaHRS* HRSL = new THaHRS( "L", "Left HRS" );
HRSL->SetTrSorting(true);  // Default: false (off)
```

- Sort function: `THaTrack::Compare` doc
- (Re)defines meaning of HRS “golden track”

Thanks to Jin Huang

HRS Track Sorting

HRS Track sorting by χ^2

```
THaHRS* HRSL = new THaHRS( "L", "Left HRS" );
HRSL->SetTrSorting(true);  // Default: false (off)
```

- Sort function: `THaTrack::Compare` ▸ doc
- (Re)defines meaning of HRS "golden track"

Thanks to Jin Huang

Some Other Useful Improvements

- Support for ROOT up to 5.24, `g++` up to 4.4 (Fedora 11)
- Support for large CODA input files ($>$ 2GB)
- Support for JLab 250 MHz Flash ADC (non-pipelined mode)

Some Other Useful Improvements

- Support for ROOT up to 5.24, g++ up to 4.4 (Fedora 11)
- Support for large CODA input files ($>$ 2GB)
- Support for JLab 250 MHz Flash ADC (non-pipelined mode)

Some Other Useful Improvements

- Support for ROOT up to 5.24, `g++` up to 4.4 (Fedora 11)
- Support for large CODA input files ($>$ 2GB)
- Support for JLab 250 MHz Flash ADC (non-pipelined mode)

Plans for Podd 1.6

- “Event Type Handler” plug-ins details
- Consistent use of `LoadDB` doc to read database
- SQL backend for `LoadDB`
- Use time-zone safe `TTimeStamp` doc for date/time
- Output speed improvements
- Extensive tests of 64-bit compatibility

Plans for Podd 1.6

- “Event Type Handler” plug-ins details
- Consistent use of `LoadDB` doc to read database
- SQL backend for `LoadDB`
- Use time-zone safe `TTimeStamp` doc for date/time
- Output speed improvements
- Extensive tests of 64-bit compatibility

Plans for Podd 1.6

- "Event Type Handler" plug-ins details
- Consistent use of `LoadDB` doc to read database
- SQL backend for `LoadDB`
- Use time-zone safe `TTimeStamp` doc for date/time
- Output speed improvements
- Extensive tests of 64-bit compatibility

Plans for Podd 1.6

- "Event Type Handler" plug-ins ▸ details
- Consistent use of `LoadDB` ▸ doc to read database
- SQL backend for `LoadDB`
- Use time-zone safe `TTimeStamp` ▸ doc for date/time
- Output speed improvements
- Extensive tests of 64-bit compatibility

Plans for Podd 1.6

- “Event Type Handler” plug-ins ▸ details
- Consistent use of `LoadDB` ▸ doc to read database
- SQL backend for `LoadDB`
- Use time-zone safe `TTimeStamp` ▸ doc for date/time
- Output speed improvements
- Extensive tests of 64-bit compatibility

Plans for Podd 1.6

- “Event Type Handler” plug-ins details
- Consistent use of `LoadDB` doc to read database
- SQL backend for `LoadDB`
- Use time-zone safe `TTimeStamp` doc for date/time
- Output speed improvements
- Extensive tests of 64-bit compatibility

Event Type Handler Plugins

- Problem: DAQ upgrades → new event types
- Solution: Plug-in modules for `THaAnalyzer` doc, replace `PhysicsAnalysis()`, `ScalerAnalysis()`, etc.
- Modules for standard event types (physics, scalers, EPICS) are provided in core library

Event Type Handler base class

```
class THaEventTypeHandler : public TObject {

public:
  THaEventTypeHandler();
  virtual ~THaEventTypeHandler();

  virtual Int_t  Analyze( const THaEvData& evdata, Int_t prior_status ) = 0;
  ...
};
```

Event Type Handler Plugins

- Problem: DAQ upgrades → new event types
- Solution: Plug-in modules for `THaAnalyzer` ▸ doc, replace `PhysicsAnalysis()`, `ScalerAnalysis()`, etc.
- Modules for standard event types (physics, scalers, EPICS) are provided in core library

Event Type Handler base class

```
class THaEventTypeHandler : public TObject {

public:
  THaEventTypeHandler();
  virtual ~THaEventTypeHandler();

  virtual Int_t  Analyze( const THaEvData& evdata, Int_t prior_status ) = 0;
  ...
};
```

Event Type Handler Plugins

- Problem: DAQ upgrades → new event types
- Solution: Plug-in modules for `THaAnalyzer` ▸ doc, replace `PhysicsAnalysis()`, `ScalerAnalysis()`, etc.
- Modules for standard event types (physics, scalers, EPICS) are provided in core library

Event Type Handler base class

```
class THaEventTypeHandler : public TObject {

public:
  THaEventTypeHandler();
  virtual ~THaEventTypeHandler();

  virtual Int_t  Analyze( const THaEvData& evdata, Int_t prior_status ) = 0;
  ...
};
```

Event Type Handler Plugins

- Problem: DAQ upgrades → new event types
- Solution: Plug-in modules for `THaAnalyzer` ▸ doc, replace `PhysicsAnalysis()`, `ScalerAnalysis()`, etc.
- Modules for standard event types (physics, scalers, EPICS) are provided in core library

Event Type Handler base class

```
class THaEventTypeHandler : public TObject {

public:
  THaEventTypeHandler();
  virtual ~THaEventTypeHandler();

  virtual Int_t  Analyze( const THaEvData& evdata, Int_t prior_status ) = 0;
  ...
};
```

Limitations: Multi-threading

- Takes advantage of multi-core systems
- Would require significant re-write
- Will not necessarily improve speed. ROOT output bottleneck
- May just run multiple jobs, one per run

Limitations: Multi-threading

- Takes advantage of multi-core systems
- Would require significant re-write
- Will not necessarily improve speed. ROOT output bottleneck
- May just run multiple jobs, one per run

Limitations: Multi-threading

- Takes advantage of multi-core systems
- Would require significant re-write
- Will not necessarily improve speed. ROOT output bottleneck
- May just run multiple jobs, one per run

Limitations: Multi-threading

- Takes advantage of multi-core systems
- Would require significant re-write
- Will not necessarily improve speed. ROOT output bottleneck
- May just run multiple jobs, one per run

Limitations: Event re-assembly

- With pipelined front-ends, CODA stream will contain chunks of 10s – 100s of events (triggers)
- Chunks from different ROCs must be re-assembled into individual events
- No support announced from CODA group
- May be done by custom decoder
  $\longrightarrow$ framework exists in Podd, but big job

Limitations: Event re-assembly

- With pipelined front-ends, CODA stream will contain chunks of 10s – 100s of events (triggers)
- Chunks from different ROCs must be re-assembled into individual events
- No support announced from CODA group
- May be done by custom decoder
  $\longrightarrow$ framework exists in Podd, but big job

Limitations: Event re-assembly

- With pipelined front-ends, CODA stream will contain chunks of 10s – 100s of events (triggers)
- Chunks from different ROCs must be re-assembled into individual events
- No support announced from CODA group
- May be done by custom decoder
  $\longrightarrow$ framework exists in Podd, but big job

Limitations: Event re-assembly

- With pipelined front-ends, CODA stream will contain chunks of 10s – 100s of events (triggers)
- Chunks from different ROCs must be re-assembled into individual events
- No support announced from CODA group
- May be done by custom decoder
  $\longrightarrow$ framework exists in Podd, but big job

Plans for 2010

- Podd 1.6
- Hall C adoption (?)
- Re-visit BigBite Tracking code (?) $\rightarrow$ see later talks

Plans for 2010

- Podd 1.6
- Hall C adoption (?)
- Re-visit BigBite Tracking code (?) $\rightarrow$ see later talks

Plans for 2010

- Podd 1.6
- Hall C adoption (?)
- Re-visit BigBite Tracking code (?) $\rightarrow$ see later talks