Hall A Analysis Software & Computing Update

Ole Hansen

Jefferson Lab

Hall A Winter Collaboration Meeting
January 17, 2024

Main Hall A Data Analysis Software: Podd (aka "analyzer")

- Key points
  - C++ class library built on top of ROOT. Scripting via ROOT C++ interpreter.
  - Developed in-house. In production since 2003.
  - Sources on GitHub. Documentation & bug tracker in Redmine.
  - Shared development with Hall C since 2012 ("hcana").
- Strengths
  - Highly modular to accommodate frequently changing experimental setups.
  - Intuitively conceptualizes analysis in terms of physical apparatuses (spectrometers, detectors) and physics calculations (kinematics, energy loss corrections, etc.)
  - Light-weight: minimal dependencies (ROOT), small memory footprint.
  - Output & cuts run-time configurable via text files. Flat text file databases.
- Limitations
  - Currently still single-threaded.
  - Designed for one-pass analysis: EVIO raw data $\rightarrow$ n-tuple-like ROOT trees + histograms
- System Requirements
  - Linux or macOS
  - ROOT 6. CMake 3. C++11 compiler. (Podd v1.8+ will require C++17.)

Podd News & Status

- 2023 developments (1.7.5 – 1.7.10)
  - Properly support decoding of "event blocks" (multiple events per CODA readout "block"), primarily for NPS experiment in Hall C.
    - ⋆ Fix various bugs in original implementation.
    - ⋆ Add multiblock support for CAEN 1190 TDC.
    - ⋆ Working well with NPS data.
  - Improved `MultiFileRun` class.
    - ⋆ Honor path list when searching for segment 0 (containing the run date/time).
    - ⋆ Support schema evolution.
  - Fix rare error in database key matching (thanks to Andrew Puckett for reporting).
  - Correct handling of time zones in database time stamps on Linux.
  - Generate `"rootmap"` files along with ROOT dictionaries. Fixes various problems with client apps.
  - Better support for interpreted analysis scripts (avoid global variables as default function arguments).
  - Added Docker support (Anil Panta, see talk at Hall C meeting tomorrow).
  - Integration tests (compares results of small replay against reference data).
- Current release: 1.7.10 (15 Jan 2024)

Integration Test Demonstration

```
% cmake -B build -S .
% cmake --build build -j8
% source ./build/cmake/setup_inbuild.sh
% ./tests/integration/run_tests.sh
Verifying raw data file
File checksum OK
Info in <replay.cxx>: Replaying /var/tmp/g2p_3132.dat.0
Run summary compares OK
Info in <verify.cxx>: Tree loaded OK
Info in <verify.cxx>: Run_Data loaded OK
Info in <verify.cxx>: Run parameters loaded OK
Info in <verify.cxx>: Reference run info loaded OK
Info in <verify.cxx>: Run number matches = 3132
Info in <verify.cxx>: Run date matches = Mon Mar 12 19:21:44 2012
Info in <verify.cxx>: Run data version matches = 2
Info in <verify.cxx>: Number of analyzed events matches = 313476
Info in <verify.cxx>: Beam energy run parameter matches = 2.253340
Info in <verify.cxx>: Beam particle mass run parameter matches = 0.000511
Info in <verify.cxx>: Prescale factors match: 0/0/1/1/0/0/0/0/1/1/1/1
Info in <verify.cxx>: Event length (fEvtHdr.fEvtLen) histogram matches, p = 1.000000
Info in <verify.cxx>: S1 calibrated ADC left PMT paddle 3 (L.s1.la_c[3]) histogram matches, p = 1.000000
...
Info in <verify.cxx>: Vertex z (L.vx.z) histogram matches, p = 1.000000
ROOT file tests OK
% echo $?
0
```

Podd: Work In Progress → v1.8, v2.0

- Event-level parallelization/multi-threading
  - About 50% done. Not completely trivial. Time-consuming.
  - SBS and NPS have been doing well using "split replays" (multiple analyzer instances processing different event ranges). Somewhat error-prone and not quite equivalent.
- Output system upgrade (full set of data types, object variables)
  - Developed in 2016/17. ∼80% complete.
  - Required for multi-threading.
- Test suite (unit & regression tests etc. Essential for long-term maintainability.)
  - Started.
  - Opportunity for summer student project.
- Improved message logging — almost complete.
- Metadata in output file — important for data preservation.

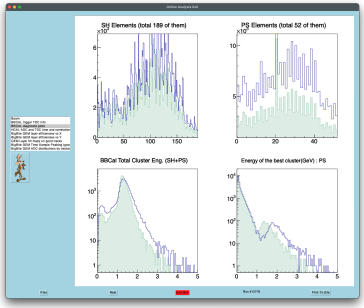
Unit Tests Demonstration

```
% git checkout hansenjo/testing
% cmake -B build -S . -DPODD_ENABLE_TESTS=YES
% cmake --build build -j8
% ctest --test-dir build/tests
Internal ctest changing into directory: /Users/ole/Develop/analyzer/build/tests
Test project /Users/ole/Develop/analyzer/build/tests
Start 1: Global Variables of Arrays
1/3 Test #1: Global Variables of Arrays .............   Passed    1.00 sec
Start 2: Formulas of Global Scalar Variables
2/3 Test #2: Formulas of Global Scalar Variables ...   Passed    0.37 sec
Start 3: Formulas of Global Vector Variables
3/3 Test #3: Formulas of Global Vector Variables ...   Passed    0.36 sec

100% tests passed, 0 tests failed out of 3

Total Test time (real) =   1.74 sec

# Alternative
% build/tests/Catch2Tests_t
Randomness seeded to: 2417342843
===============================================================================
All tests passed (373 assertions in 3 test cases)
```

Run `Catch2Tests -s` to see each individual check (long output).

Panguin (Online GUI)

- Default data visualization tool (next to plain ROOT).
- Merging Hall C onlineGUI with Hall A panguin (nearly complete).
- Will support image file output for Hydra.
- See README.md for full documentation.



Panguin 2.6 Command Line Options

```
$ panguin --version
Panguin version 2.6 (12-Dec-2023)
$ panguin --help
panguin: configurable ROOT data visualization tool
Usage: panguin [OPTIONS]

Options:
-h,--help                     Print this help message and exit
-f,--config-file <file name> [default.cfg] Job configuration file
-r,--run <run number>         Run number
-R,--root-file <file name>    ROOT file to process
-G,--goldenroot-file <file name> Reference ROOT file
-P,-b,--batch                 No GUI. Save plots to summary file(s)
-E,--plot-format <fmt>        Plot format (pdf, png, jpg ...)
-C,--config-path,--config-dir <path>
                              Search path for configuration files & macros
--root-dir <path>             ROOT files search path (":"-separated)
-O,--plots-dir <dir>          Output directory for summary plots
-I,--images                   Save individual plots as images (implies -P)
-F,--image-format <fmt>       Image file format (png, jpg ...)
-H,--images-dir <dir>         Output directory for individual images
                              (default: plots-dir)
-v,--verbosity <level>        Set verbosity level (>=0)
-V,--version                  Display program version information and exit
```

Preinstalled Hall A (and C) Software

CentOS 7 farm/ifarm (works in Counting House, too)

```
% module use /group/halla/modulefiles
% module avail
-------- /group/halla/modulefiles --------
analyzer/1.7.0  boost/1.84      gcc/12.3.0     group.apps      python/3.11.0  root/6.26.08          tbb/2021.10.0
analyzer/1.7.4  clhep/2.4.6.4   gdb/13.2       hcana/0.96      python/3.11.4  root/6.26.10          xerces-c/3.2.4
analyzer/1.7.6  evio/5.3        geant4/10.7.4  hcana/0.99      qt/5.15.10     root/6.26.10.pythia6
analyzer/1.7.7  evio/5.3_gcc48  geant4/11.0.4  hcana/1.0.0     root/6.22.06   root/6.28.04
analyzer/1.7.8  gcc/12.2.0      geant4/11.1.2  pythia6/6.4.28  root/6.26.06   tbb/2020.3
% module load analyzer
Loading analyzer/1.7.8
 Loading requirement: group.apps gcc/12.3.0 python/3.11.4 root/6.26.10 evio/5.3
% analyzer --version
Podd 1.7.8 git@Release-178-0-g24e3330 3 Nov 2023
Built for CentOS-7 using gcc-12.3.0, ROOT 6.26/10
```

- In Hall A counting house do `module use /adaqfs/apps/modulefiles`.
- analyzer 1.7.10, latest ROOT, panguin, Pythia 8, etc., will be installed shortly.
- Need to have separate build for each supported OS (currently only CentOS 7).
- Mirroring to CVMFS for offsite access (OSG etc.) planned.
  On-site under `/u/scigroup/cvmfs/halla`.

Hall A Online Computing

- Will upgrade counting house to RHEL/AlmaLinux 9 in Feb/Mar 2024 (all systems).
- Successfully tested, except for DAQ[1]
- Minor host reorganization:
  - `aonl1`: Main analysis system (128 threads, 512 GB RAM).
  - `aonl2-4`: Legacy/fallback systems (3×32 threads, 3×64 GB RAM).
- Additional:
  - `compton`: New hardware.
  - `avideo1`: New system, for Hall cameras.
- DAQ resources: 240 threads, 1 TB RAM, 170 TB disk, 40 Gbps Ethernet.
- Analysis resources: 240 threads, 736 GB RAM, 196 TB disk, 40/10 Gbps Ethernet.
- Exceeds online computing requirements through the MOLLER experimental run (2028/29).

[1] The DAQ is Java-based, thus platform-independent, what can go wrong?

Scientific Computing

- Farm/ifarm will migrate to AlmaLinux 9 during 2024.
- Push towards increasing containerization underway. See tomorrow morning's talk by Anil Panta.
- Farm resources (details)
  - More CPU! 25 farm23 systems @ 256 threads each.
    Total $\sim$ 26,000 threads $\approx$ 240 Skylake (2018) M-core-hours/year.
  - 7 nodes with Nvidia TitanRTX, T4, or A100 GPUs dedicated for ML ("gpu" partition).
  - Disk: Lustre: 4.1 PB, Work: 1.4 PB.
  - Lustre upgrade planned (several PB). Work disk to follow (double capacity).
- Mass storage system (as of Jan 2024) (details)
  - Capacity $\approx$ 150 PB (LTO-8, uncompressed), $\approx$ 113.1 PB used
    (27.7 production, 32.4 raw, 31.9 rawdup, 17.3 HPC, 3.8 user data).
  - Throughput $\approx$ 10 GB/s (24 LTO-8 drives, uncompressed, theoretical)

Recent Tutorials: Software & Computing Workshop May 2023

- https://indico.jlab.org/event/683/
- Very nice 3-day hands-on introduction to C++ by experts from CERN.
- 2 additional days with tutorials on JLab-specific topics, including Hall A & Hall C analysis.
- Video recordings linked from conference page.

Main Page
Workshop Registration
Overview
Timetable
Contribution List
Poster
My Conference
My Contributions

Video Recordings

Day 1 Language Basics
Day 2 Object orientation & Core modern C++
Day 3 Tools & Object orientation Part 1
Tools & Object orientation Part 2
Day 4 JLab oriented I
Day 5 JLab oriented II

Outlook

- In Hall A, Podd is expected to be used throughout the SBS program and for MOLLER counting mode measurements.
- Similarly, upcoming Hall C experiments will use Podd/hcana for the foreseeable future, *i.e.* through the 2020s.
- Continuing maintenance as well as significant improvements and modernization (multithreading etc.) ongoing.
- Recent focus on containerization and automatic archiving of software versions will go a long way towards long-term analysis preservation and reproducibility.