MicroBooNE Goals, Status, Needs

Larsoft CI Workshop
June 17, 2014

H. Greenlee

Outline

- Status.
- Testing.

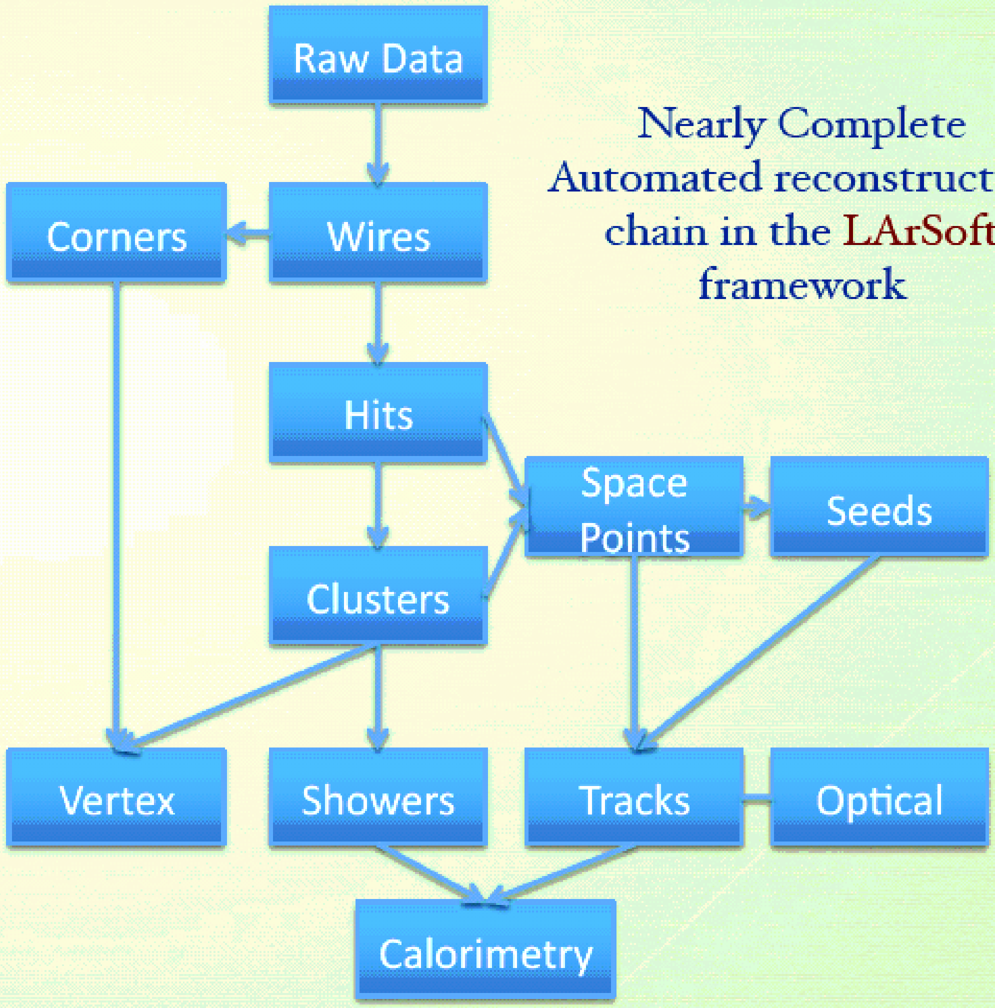
Nightly Build

- Nightly builds in place for uboonecode package.
  - Uses script in laradmin/nightly/nightly.sh.
  - Wrote our own cron wrapper (4am). Uses uboone shared keytab.
  - No systematic error checking.

Tingjun Yang
Users Meeting

Analysis Tools

* Analysis Tools sub-groups in place and developing tools since May, 2013
* Regular MC challenges.
* Progress on Reconstruction/ simulation/software tools.
* Reconstruction workshop at Yale in March, 2014.
* >90% efficiency to reconstruct cosmic muons.



Nearly Complete Automated reconstruction chain in the LArSoft framework

Algorithms Status

- There has been major recent progress in the following areas.
  - Simulation.
    - Electronics simulation (including noise).
    - Field response simulation.
    - Optical and trigger simulation.
  - Time service.
  - Reconstruction.
    - Filtering (deconvolution) Wire regions of interest.
    - Clustering (cluster crawler and fuzzy cluster).
    - Tracking (including momentum determination).
    - Shower reconstruction.
    - Vertex reconstruction.

My Perspective on Testing

- My perspective is largely colored by my experience with software testing in D0 since the fortran to c++ transition (~17 years).
- Modern testing terminology (as I understand it).
  - Unit test – pass/fail test run on the smallest testable unit of code.
  - Regression test – Seeks to track evolution of software performance from version to version (not necessarily pass/fail).

Release Testing in D0

- In D0, we had two kinds of pass/fail tests (aka unit tests).
  - Component tests.
  - Integrated tests.
- Component tests.
  - Mandatory, one component test per source file.
    - The build system would annoy you with a warning if you didn't provide a component test.
  - Invocation – "make ctest."
  - Interface.
    - Developer was expected to supply a c++ main program that would call the class or function being tested and return success or failure.
    - Test factory was available for testing D0 equivalent of modules without firing up the full framework.
      - A test factory for modules, services, algorithms would be useful.

Release Testing in D0 II

- Integrated tests.
  - Optional.
  - Invocation – “make itest.”
  - Interface.
    - Arbitrary script, which would typically run a framework program (but could in principle do anything).
      - Standard test data files were available, or developer could supply his/her own test data.
      - Since integrated tests ran during build, they couldn’t be something that took a long time (hours).
    - Script decides success or failure.
      - To my knowledge, we never had an integrated test that did anything as sophisticated as comparing histograms.
      - Parsing log files – yes.

Release Testing in D0 III

- Component and integrated tests were available to be run, and were run, by individual developers and in official release builds.
- Errors generated in official builds triggered e-mails to developers.

Regression Testing in D0

- RecoCert program.
  - In larsoft terms, RecoCert was a framework program that would be run on reconstructed data, which included an analysis module for every reconstruction module.
    - Whenever a new reconstruction module was added in reconstruction program, a new analysis module was added in RecoCert.
    - Result was a histogram file contained O(1000) 1D and 2D histograms (about 50 pages when printed).
    - RecoCert histograms could be overlaid between two releases run on the same input data.
      - Performance judged by humans (not automatic, not pass/fail).
    - RecoCert histogram output was archived in sam (on disk and on tape) for every version of the reconstruction program.
    - RecoCert output was also displayed online overlaid against known good reference data.

Lessons Learned

- Unit tests should be integrated into the build system.
- Errors in official builds should trigger e-mails or some other action.
- Testing interfaces should be flexible.
  - It will be useful to be able to test classes/functions below the level of modules, services, and algorithms.
  - It will be useful to have test factories that will allow testing of modules, services, and algorithms outside the art framework.
- We (larsoft/microboone) need some kind of regression testing framework. Probably this should be separate from build system.
  - A regression testing framework would probably have some aspects common among experiments, as well as some experiment-specific.

Summary

- Nightly build status.
- Overview of algorithm progress.
- Unit testing and regression testing in D0.