Mu2e Offline Review: Software, Tools, and Environment

David Brown, LBNL

Talk Outline

- Mu2e Software Development Environment
- Software used by Mu2e
- Mu2e User Environment
- Performance

Mu2e Offline Development Environment

Mu2e Offline Structure

- Core Mu2e packages
  - Separated by sim/reco/analysis, data/algorithm, detector subsystem
    - currently 69 packages
  - src, inc, test, geom, fcl, … subdirectories
  - no README, doc
- External packages
  - art, G4, ROOT, CLHEP, BOOST, PDG, …
    - Mu2e-specific versions and builds at FNAL
    - Distributed through scisoft.fnal.gov
  - BaBar Kalman Fit
  - SuperB FastSim (legacy)
- Offsite development supported
  - Relocatable installation, requires only kerberos ticket
  - Thanks to Lynn Garren

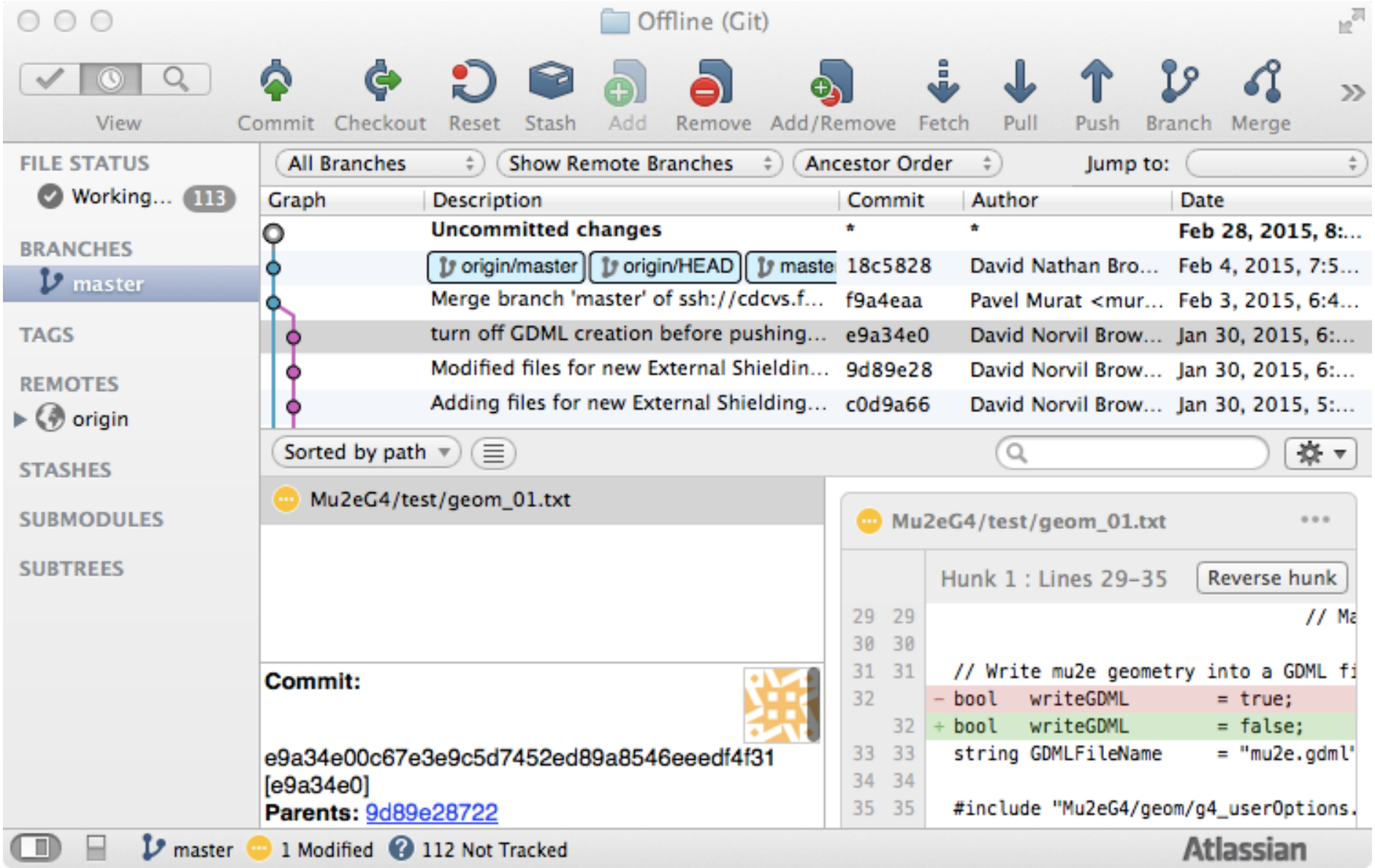
Offline Environment

- Supported on SL5, SL6, (Mavericks)
- Core environment defined by ‘setup mu2e’
  - External package versions managed by UPS
- Version-specific environment defined by setup.sh
  - Part of Offline, tagged with releases
- Command line tools (no IDE)
  - git, emacs, vim, …
- SCONS build system
  - Python-based, multi-platform, open source
  - Automatic dependency generation
    - Follow some but not all schema changes

Code Management

- Offline is in one git repository
  - Online, common will be separate repositories
  - Hosted on FNAL redmine server
    - can use ‘foreign packages’ from github, common repos
  - Still digesting transition from CVS (see workflow)
- Exception: BaBar code
  - now in svn@LBL, moving to github
  - also used by DarkLight
- Releases defined as tags
- Distribution is through git
  - Local binary distribution available via CVMFS

SourceTree git Browser



Code Standards and Validation

- Draft coding guidelines
  - Long list of do/don't
  - No enforcement yet
- Targeted code reviews planned
  - post CD3
- Jenkins test build system operational
  - Builds hourly, nightly, and for release
    - basic email notification of problems
  - Different levels of associated validation
    - compilation, standard scripts, …
  - Automatic content check system design/implementation in progress

http://mu2e.fnal.gov/public/hep/computing/standards/codingGuidelines.shtml

http://mu2e.fnal.gov/atwork/computing/jenkins.shtml

Rules

1. The standard file name extensions for Mu2e code are .hh an
2. Modules may communicate with each other only via the art:: particular, they may not communicate via art::Services, singl module-parallel execution.
3. Only include the header files you need. Do not include extra a particular header, even if the current snapshot does not.
4. Only get data products from the event when you need them.
5. Never code a using directive or using declaration in a heade

```
using std::cout;        // Do not place either of
using namespace std;
```

   This may needlessly slow down compilation. And it may cha changes must also be written carelessly; but we want a defe http://www.parashift.com/c++-faq-lite/coding-standards.html#
   1. A corlollary of this is that you must use fully scoped typ

   You may use using declarations and directives in your .cc fil need them. And remember that any using declaration/directi
6. Always protect header files from multiple inclusion with head
7. There are only a few legimate uses for CPP macros:
   1. #include, header guards, and selection of archtecture s
   2. Invoking DEFINE_ART_MODULE and DEFINE_ART_
   3. Using the macro versions of the mf::Log* clases.
   4. Enabling of debug code.

   For all other cases in which you are tempted to use a CPP m
8. Do not use Exception Specifications.
9. Normally all mu2e classes should be in the mu2e namespac
   1. If you want to define sub-namespaces inside the mu2e project; while a particular choice might make one aspe
10. Do not use variable names with the form, a double or single
11. Do not choose identifiers that can be very easily confused as namespaces make sure that the compiler understands what remove them as targets of opportunity.). We have lots of go
12. All libraries must specify all of their first order dependencies
13. Do not cache handles to data products, GeomHandles or Co
14. You may cache ServiceHandles.
15. Linkage loops are forbidden. If you violate this rule things ma and simple, solution is to create a third library that contains c linkage loops are hidden because they work by accident: if th loop can work by accident.
16. Any code committed to the reposity must compile cleanly usi these exceptions will normally be due to issues that originate
17. All constructors must always leave the object in a safe, well

Development Workflow

- ‘git clone’ to create local work area
  - scons to build
- Local editing and testing
- ‘git commit’ as necessary during development
  - Master branch
  - Alternate branches for shared, major projects
- Update Origin
  - ‘git pull; (git merge;) git push;’
- Diagnostic Tools
  - gdb
  - valgrind, memgrind
  - Simple cyclic dependency checker
  - EventDisplay, root, cout, …

Mu2e Software

art Framework

- Fermilab-supported framework
  - Distilled from CMS framework for smaller experiments
  - Used by NOVA, darkside, microboone, ELBNF, …
- Dedicated configuration language (FHiCL)
  - Intentionally disallows script programming
  - macros (#include, @local, @table, …) allow indirection
- Expanding set of utilities
  - timing, root, messaging, ‘memory checking’
- Also used for mu2e online (artdaq)
- https://web.fnal.gov/project/ArtDoc/Pages/home.aspx

G4 Simulation

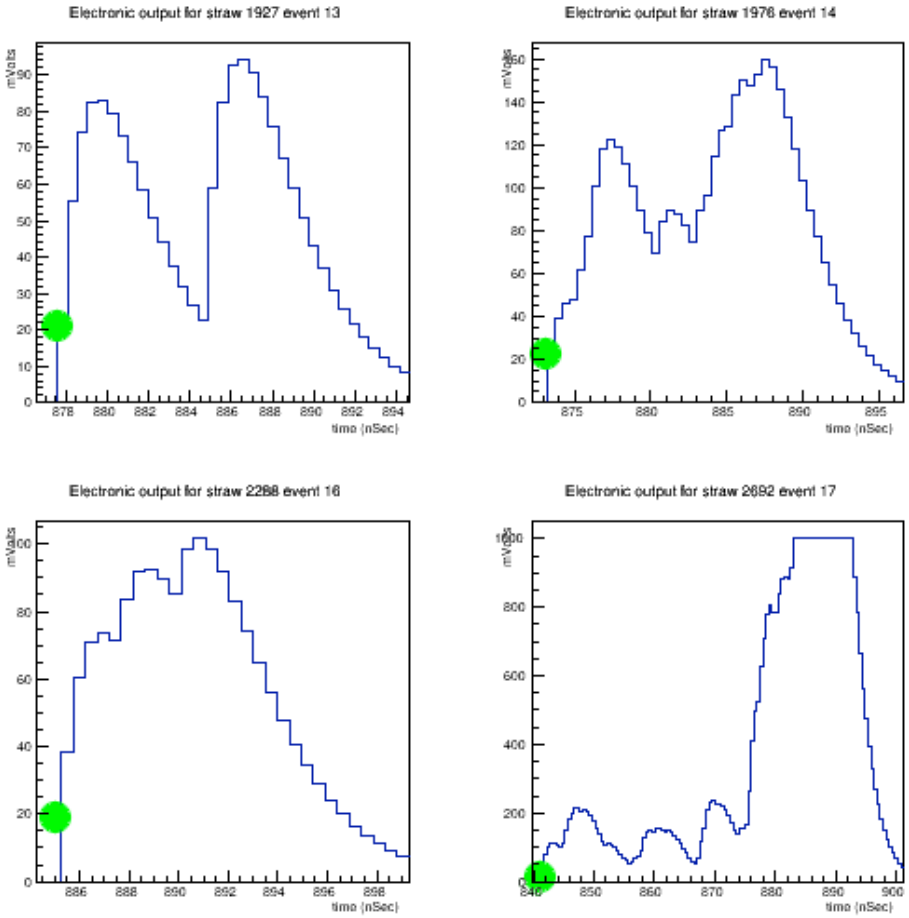
- Low momentum physics, muon capture are important to Mu2e
  - We must follow developments closely to balance LHC
- Recent Geant4 developments benefiting Mu2e:
  - Corrections and improvements to the bound muon capture and decay rates (9.6.p02)
  - Labeling of the muon capture products (10.1)
  - ShieldingM physics list (10.1)
    - Created by Mu2e, now an official part of G4
  - Correction to the muon capture products timing (9.6.p04, 10.1.p01)
  - Addressing rare cases of infinite loop in muon capture process (9.6.p04, 10.1.p01)

Simulation (non G4)

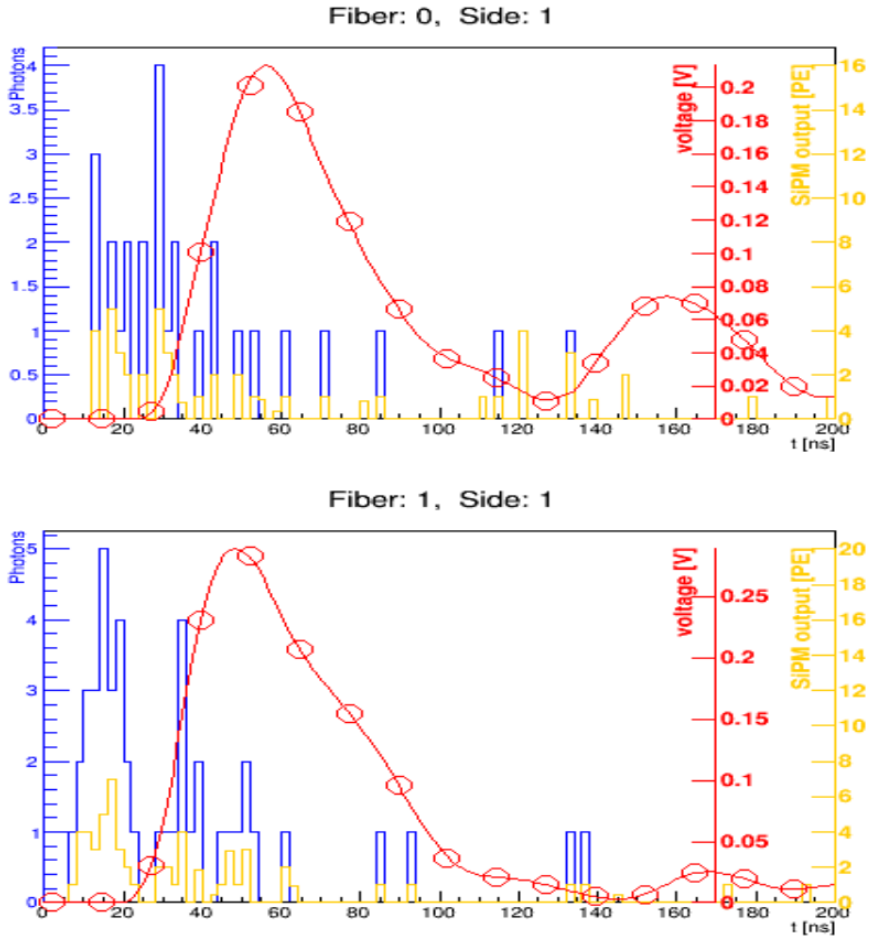
- Generators
  - Daya Bay cosmic ray generator
  - Specialty generators for muon capture products
    - Generally we do not rely on G4 for this
    - Tuned to most recent measurements and calculations
    - Weighted and unweighted (optional) outputs
- Detailed detector response simulation
  - Translate G4 energy deposits to ‘digis’ (raw data)
  - Model detector physics + electronics response
    - Use G4 studies, lab bench measurements to set parameters
- Full Background frame overlay
  - From multiple sources
  - Analog signal summing

Simulation Examples

Straw analog waveforms with threshold crossing



CRV counter photons, photo-electrons, and SIPM waveforms



Reconstruction

- Complete chain from simulated raw data to high level objects used in analysis
  - Tracks, calorimeter clusters, CRV co-incidences
- Mu2e-specific pat. rec. and noise filtering
  - tracker and calorimeter
- BaBar material model and Kalman fit
  - Battle tested
  - Full fit not persisted!
- Track and Calorimeter integration
  - Track-cluster matching
  - Cluster seeded tracking

Special Cases

- Extinction Monitor
  - Independent subsystem tied to primary (proton) beam, not muon beam
  - Code in Offline, but not executed in ‘standard’ job
  - Data access will be through ‘conditions’ interface
    - synchronized, but not per-event
- Online Filter (Software ‘trigger’)
  - Same code as Offline
    - specialized configuration
    - #pragma if necessary
  - Vendor compiler required to achieve performance
  - Will have separate tags, builds, …

Persistence

- Currently using root 5
  - Schema generated by genreflex
  - No ‘custom streamers’
- Data classes defined in dedicated packages
  - Reco package separate from Sim
  - Sim-Reco matching done through associations
    - Explicit or implicit (indexing)
- Automatic schema evolution partially supported
  - Schema additions/subtractions processed by genreflex
  - New production, major changes → new schema
  - Wait till commissioning (2020) before preserving schema

Mu2e User Environment

Runtime Configuration

- SimpleConfig
  - legacy system from earlier projects
  - Controls G4 geometry, generators
    - < 100 hierarchically-structured *.txt files
  - Will eventually be phased out
- art configuration (FHiCL)
  - Module instantiation, runtime path
  - Algorithm parameters and options
  - Also used to set ‘constants’ (pedestals, gain, …)
    - Access through Conditions Service
  - Policy: parameter defaults set in scripts, not code
    - except when default value is unambiguous
- BaBar code constants
  - Material properties ‘database’ (text file)

Example User Job

```
> mu2e --config Analyses/test/TracksFromStepPoints.fcl --source-list tdr.beam.g4s4.conversion.1812b.16763106/files.txt --nevts 100
```

- Static sequence of modules (not ‘on-demand’)

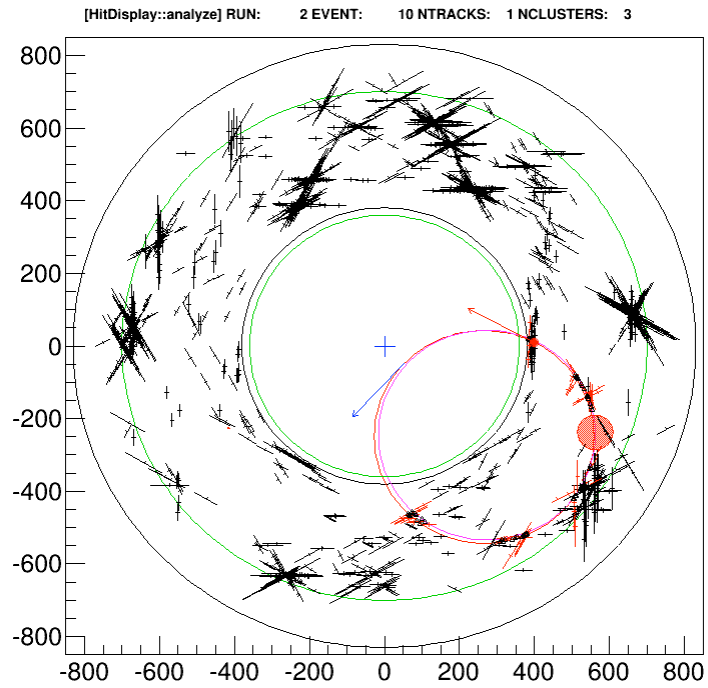
```
#
#  This script runs hit and track reconstruction starting from MCStepPoints.
# for example, the command
#  > mu2e --config KalTests/test/TracksFromStepPoints.fcl --source-list /mu2e/data/tdr/beam/g4s4p7/fi
# will process 100 events from the TDR sample of pure conversion electrons
#
# David Brown, 19 Feb 2015
#
#include "minimalMessageService.fcl"
#include "standardProducers.fcl"
#include "standardServices.fcl"

process_name : TracksFromStepPoints

source :
{
  module_type : RootInput
  maxEvents: 100
}

services :
{
  messageService : @local::default_message
  TFileService : { fileName : "TracksFromStepPoints.root" }
  RandomNumberGenerator: { }
  user :
  {
    GeometryService        : { inputFile : "Mu2eG4/test/geom_01.txt" }
    ConditionsService : { conditionsfile : "Mu2eG4/test/conditions_01.txt" }
    GlobalConstantsService : { inputFile : "Mu2eG4/test/globalConstants_01.txt" }
    SeedService: @local::automaticSeeds
  }
}

physics :
{
  producers :
  {
    protonTimeMap: { module_type : GenerateProtonTimes }
    muonTimeMap: { module_type : GenerateMuonLife }
    makeSD: @local::makeSD
    makeSH: @local::makeSHfromSD
    FSHPreStereo: @local::FSHPreStereo
    MakeStereoHits: @local::MakeStereoHits
    FlagStrawHits: @local::FlagStrawHits
    FlagBkgHits: @local::FlagBkgHits
# track downstream eminus (ie conversion candidates)
    TPRDownstreameMinus : @local::TrkPatRecDownstreameMinus
  }
```

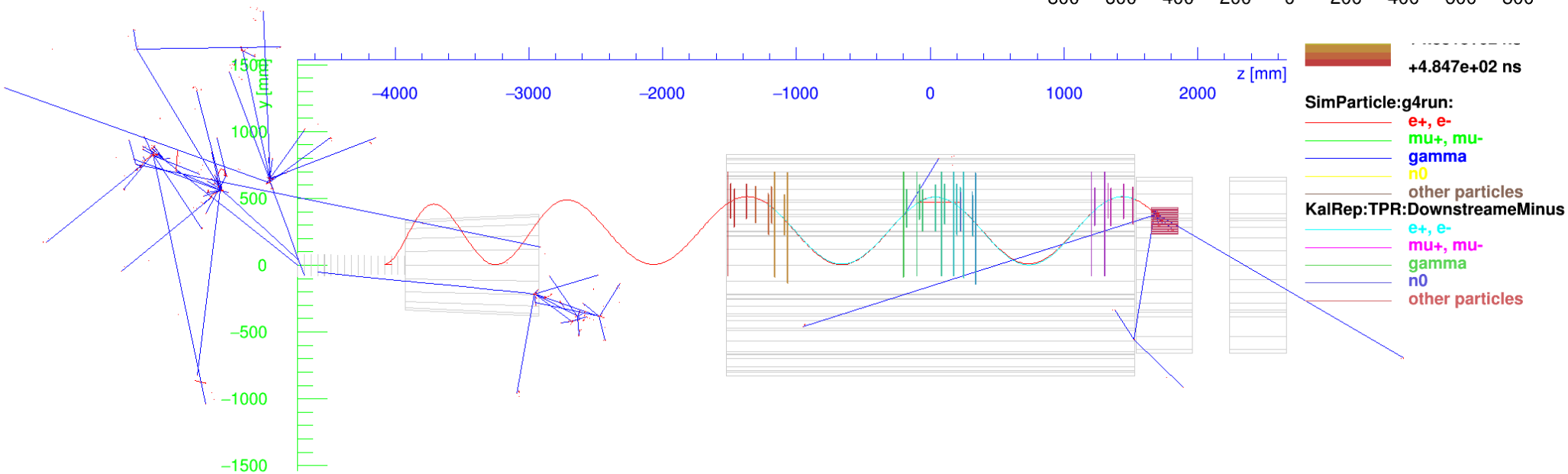
```
  analyzers: {
# count generated events
      genCountLogger: { module_type: GenEventCountReader }
# read Kalman fit DownstreameMinus tracks
      RKFDownstreameMinus: {
         module_type: ReadKalFits
         fitterModuleLabel: TPRDownstreameMinus
         KalDiag : {
          MCPtrLabel : "makeSH"
          MCStepsLabel : "detectorFilter"
          SimParticleLabel : "detectorFilter"
          SimParticleInstance : "s0"
          StrawHitMCLabel : "makeSH"
          TimeOffsets :  { inputs : [ "protonTimeMap", "muonTimeMap" ] }

          # track-level diagnostics.  For detailed (hit-level) diagnostics use diagL
          diagLevel : 1
        }
      }
   }
# trigger path
  p1 : [protonTimeMap, muonTimeMap, makeSD, makeSH, FSHPreStereo, MakeStereoHits, Fl
  TPRDownstreameMinus ]
# end path
  e1 : [RKFDownstreameMinus]

  trigger_paths  : [p1]
  end_paths      : [e1]
}

# specify the time offset maps when building straw digis
physics.producers.makeSD.TimeOffsets.inputs : [ "protonTimeMap", "muonTimeMap" ]
```

Use of @table, @sequence will further simplify this!

EventDisplays

- eve-based art display
  - Realtime, OpenGL
- root-based projection displays
- Need to integrate
  - graphics expert needed





Analysis Output

- Persistent objects not ‘user-friendly’
  - Data objects are concise, have limited interfaces
  - root interface to art data difficult to navigate
  - ‘interesting’ quantities not direct data members
- Analysis-specific post processing is required
- Some standard tools for supporting analysis exist
  - Example: KalDiag
    - Computes track hit residuals, momentum resolution, chisquared, …
    - Provides output at 3 levels: struct, TBranch, TTree
    - Used successfully for several TDR analyses
- Ongoing discussion of general strategy
  - goals: user friendly access within resource limitations

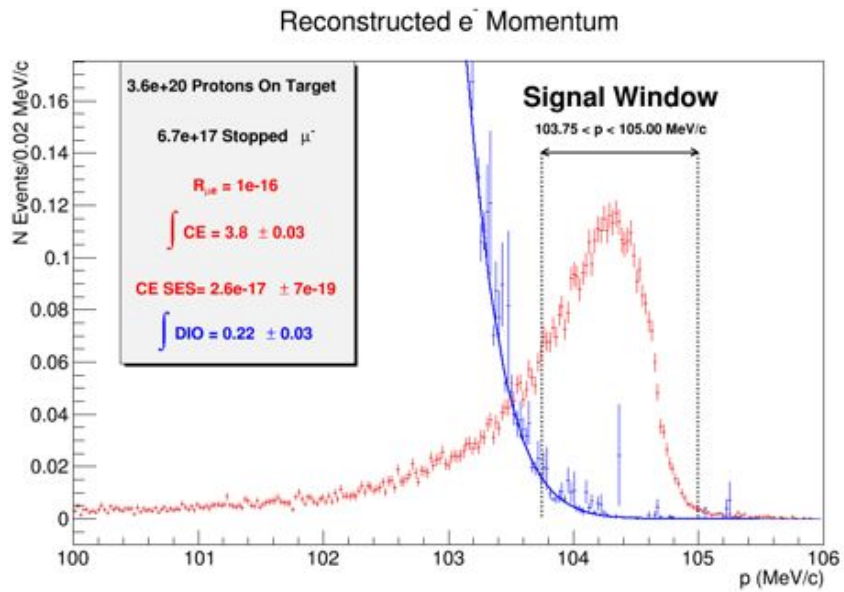
Documentation

- ‘GettingStarted’
  - http://mu2e.fnal.gov/public/hep/computing/gettingstarted.shtml
  - Covers core issues relevant for using + developing
  - Formatted as a long list
  - Search not supported
- Introductory examples
- Data content and algorithms
  - Distributed among doc-db (talks, documents), header files, …
  - Dynamic situation
- Production data access
  - see Andrei’s talk

Performance

Mu2e TDR Results

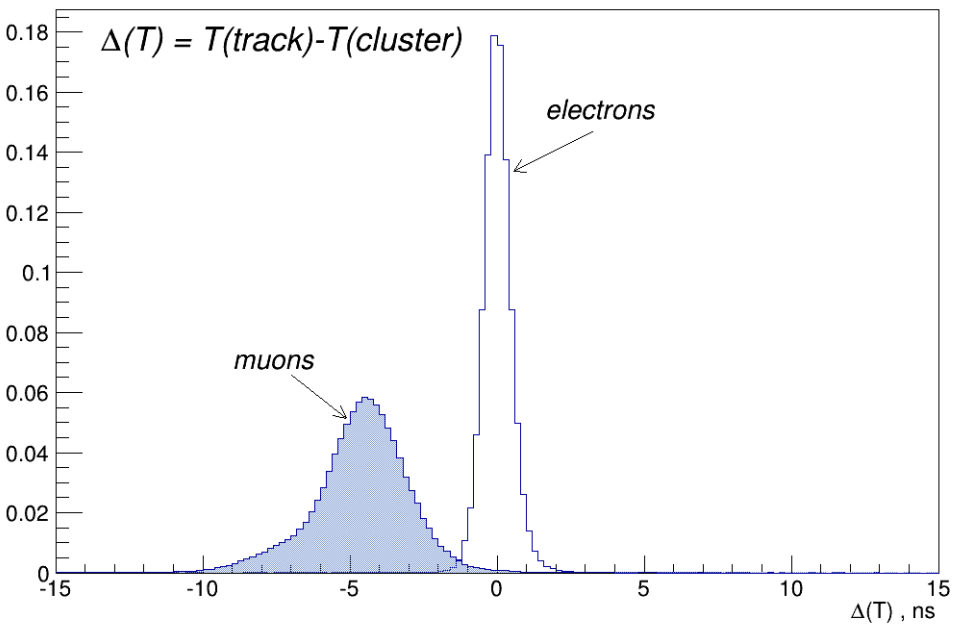
- Complete chain from G4 to analysis in Offline
  - start: protons on target
  - end: analysis TTrees
- Many studies and results
  - Experimental sensitivity
  - Accidental hit rates
  - Detector parameter optimization
    - in terms of CE sensitivity
  - Systematic error propagation
    - Cross-sections and detector effects



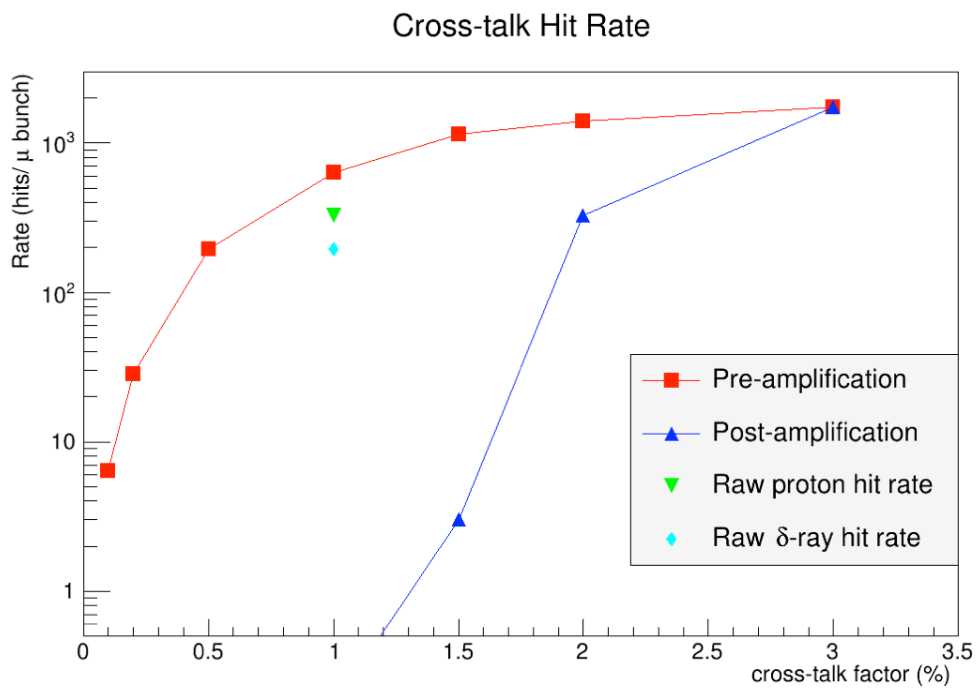
Effect	Uncertainty in DIO background yield	Uncertainty in CE single-event-sensitivity ($\times 10^{-17}$)
MC Statistics	±0.02	±0.07
Theoretical Uncertainty	±0.04	-
Tracker Acceptance	±0.002	±0.03
Reconstruction Efficiency	±0.01	±0.15
Momentum Scale	+0.09, -0.06	±0.07
μ-bunch Intensity Variation	±0.007	±0.1
Beam Flash Uncertainty	±0.011	±0.17
μ-capture Proton Uncertainty	±0.01	±0.016
μ-capture Neutron Uncertainty	±0.006	±0.093
μ-capture Photon Uncertainty	±0.002	±0.028
Out-Of-Target μ Stops	±0.004	±0.055
Degraded Tracker	-0.013	+0.191
Total (in quadrature)	+0.10, -0.08	+0.35, -0.29

Analysis Examples

Separation of muons from tracks using TOF between tracker and calorimeter



Tracker accidental hit rate vs electronics cross-talk coupling



Computational Performance

- TDR reconstruction plus analysis processing
  - Tracker and calorimeter
  - grid worker node (average)
  - Signal only: ~25 Hz
  - Background mix: ~0.3 Hz
    - Track and calorimeter reconstruction
    - Pre-mixed: ~1 Hz
- Online filter
  - Intel compiler, 16-core Xeon processor
  - Tracking only, accidental hits only (no signal)
  - Preliminary performance results within reach of requirements

Conclusions

- Mu2e Offline environment is fully functional
- Successfully used to produce the TDR
  - All relevant science issues covered
  - Detailed simulations
- CD3 goals are achievable without major changes

Backup

Known Missing Pieces

- Reconstruction geometry model
  - steal from LHC/PANDA/BaBar/… ?
- Constants database
- Partial build system (+ binary release)
- Code reviews
- Coordinated documentation
- Persistent connection between event/run data
- Integration of CRV with tracking

Medium Term Projects (pre/post CD3)

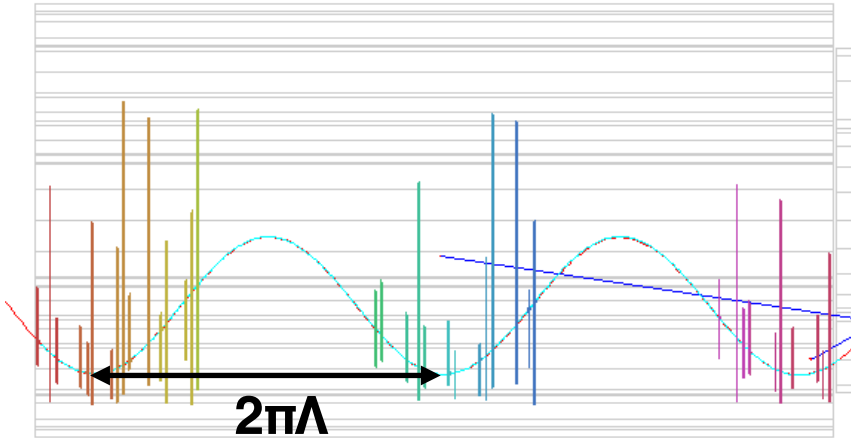
- Modernize BaBar code
- Root 6
- CLHEP linear algebra -> Eigen
  - following recent ATLAS migration
- Configuration simplification (@table, @sequence)
- Common analysis output format

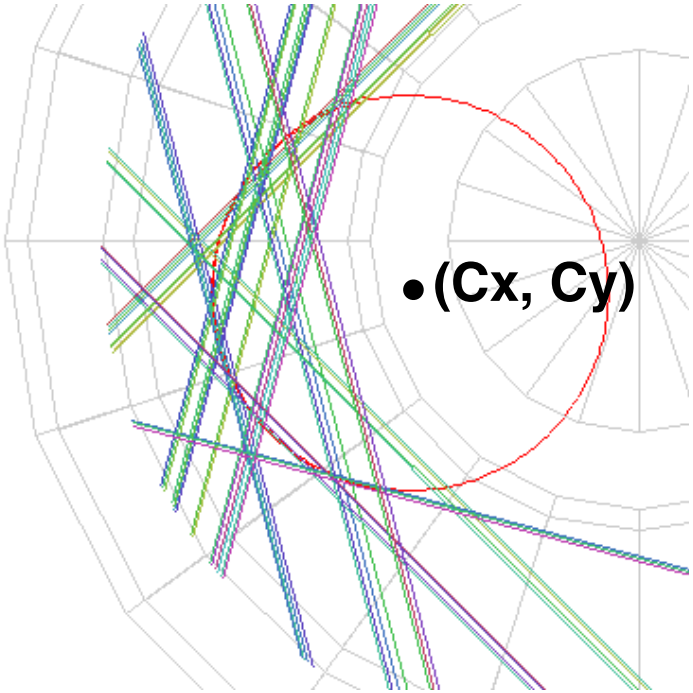
Example TDR Job

- Background Mixing is the most complicated part
- 530 lines (fully expanded)

R, Λ, t Helix Parameterization

- **R** = transverse radius
  - sign(**R**) ≡ sign(dΦ/dt) = -sign(qB_z)
- **Λ** = longitudinal wavelength
  - **Λ** ≡ dz/dϕ
  - sign(**Λ**) = helicity
- **C_x,C_y** = helix axis transverse position
- **t_0** = time when particle passes z=0
- **ϕ_0** = momentum azimuth when z=0
  - **ϕ_0** ≡ atan2(P_y,P_x), t=**t_0**
- **Q** = $cqB_z(0,0,0)$
  - c = speed of light
  - B(x) = magnetic field
- **m** = particle mass



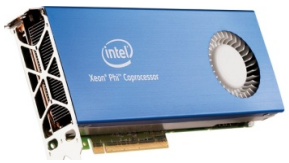


Online Filter Performance

- ‘Triggerless’ DAQ
  - data streamed to a 36-server processor farm
  - @200 KHz raw event rate
- Need factor ~100 filtering to meet storage limits
- Select using full track reconstruction
  - ~4 msec/event
  - meets requirements

16 cores

120 cores



	XEON E5-2687v2	XEON PHI 5510P
Stereo Hits		
0) reference code (gcc compiler)	83.6 msec	-
1) algorithmic improvements (gcc compiler)	4.3 msec	-
2) Intel compiler, loop vectorization	**1.4 msec**	**4.8 msec**
Background Hits		
0) reference code (gcc compiler)	9.0 msec	-
1) Intel compiler	5.1 msec	123.0 msec
2) refactoring	3.4 msec	38.1 msec
3) double → single precision	**2.1 msec**	**23.9 msec**
Overhead		
0) reference code (gcc compiler)	0.9 msec	-
1) Intel compiler (estimated)	**0.3 msec**	**2.0 msec**
total processing time	3.8 msec	30.7 msec
events/sec (single core)	260	32
number of cores (36 servers)	720	4,320
events/sec (36 servers)	**187,000**	**138,000**

190KHz **140KHz**