Feedback ATLAS

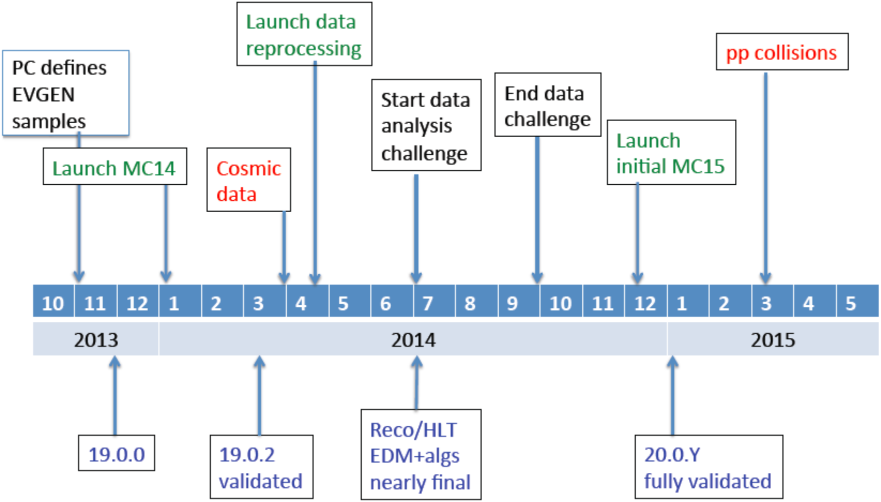
Sébastien Binet

LAL/IN2P3

2013-06-14

LABORATOIRE
DE L'ACCÉLÉRATEUR
LINÉAIRE

ATLAS LS1 Schedule - Releases



C++ version & compilers

- currently: on the road towards release 18.0.0 (Summer)
- GCC−4.6 being validated
  - C++11 support incomplete, binary incompatible w/ non-C++11 code
  - will be retired quickly
- GCC−4.7 will be validated against 4.6 (which will be dropped)
  - C++11 almost fully supported
  - release 18.0.0 with use of C++11 allowed

- release 19.0.0
- test GCC−4.8.x
- capture the (first bits of) integration/migration work with ROOT−6
  - realistically, 19.x.0 will see the first builds with ROOT−6

- on the road towards 19.0.0
- start testing GCC−4.8.x
- start testing CLang−3.2/3.3
- start testing ICC−13

ROOT−6 migration items

Reflex

- Gaudi relies on Reflex::PluginService to declare and load components
  - need a replacement to do anything useful
  - the factory declaration is performed by C-macros so migration should be painless
- dictionaries
  - need to retain 2 ways of creating them (.so and JIT-ing)
  - to understand impact of JIT-ing on VMEM, memory shared thru fork+COW by AthenaMP
  - ATLAS uses a fair number of dictionaries (rootmap entries: 12k classes, 7k factories): need to ensure the CLing db allows that kind of scaling capabilities
  - also: static C-array to hold this db of types ? (would be nice not to have to rebuild ROOT with some special CPP macro value and thus use a dynamic array. . .)

- 15 packages in the whole Athena release use Reflex::Type,Object
- deal with TTree programmatic inspection/description (ie: reading/writing)
  - unified API to deal with a C++ type (class, builtin)
  - mostly located in our POOL layer
- essential: use Reflex::Object as a typed void* to publish and push data around
- need to retain the Reflex way of naming types (no std:: elision!)
  - TClassEdit and friends ease the pain in converting b/w CINT naming and Reflex one, but

ROOT-6 migration items - II

Reflex - needed eventually

- dictionary selection *via* header annotation
- handling of virtual base classes
- handling of multiple virtual base classes
  - thanks to our T/P separation infrastructure, these classes are only needed at the interactive prompt/for PyROOT-based analyses
  - not critical for I/O
- migrate a few PyROOT utility scripts using the Reflex API to its replacement

Reflex - nice to have, not critical

- autoloading of enums
- autoloading of functions

- piggy-back on a LCGCMT-provided nightly with ROOT-6
- start a migration nightly on top of that

Inputs from I/O

- EventStore dev-team in close contact with Philippe
- Most if not all the issues listed below are needed *before* ROOT-6

- Support for 64bit content aware references:
  - mainly to ease merge complications which currently arise when using TTree entry number
  - will also allow to create forward references (to data products which are not yet produced)
- needs ROOT support 64bits int in TTreeFormula (ROOT-5084) and in TTreeIndex (ROOT-5085)
- with the proposed new analysis EDM (AOD/D3PD merger, see next slides) greater challenges will be met on making row- and column-wise access to the same file efficient:
  - following closely work on TTreeCache optimization (ROOT-5080) and basket optimization (ROOT-131)
- also, in connection with multi-cores work: support for multiple TTreeCache (and related extensions, e.g. to TTreePerfStats, ROOT-5180)
- ATLAS could also greatly benefit from the work done by CMS (not yet in ROOT code base) for better cache support of the filtering pattern
  - could be used for our new derivation/reduction analysis framework (see next slide)

Analysis inputs

- currently redesigning our analysis model, EDM, . . . in light of what we learned during Run-I
- first version of the new EDM for `release-19.0.0` (end-2013)
- first version of our new reduction fwk: `release-19.X.0` (before Feb-2014)

- dev-team in close contact w/ Philippe
- issues related to the new EDM are listed here:
  - `ROOT-5035` (Custom converter)
  - `ROOT-5264` (ditto)

Other than the above, and generally speaking, there is nothing holding up physics analysis w/ `ROOT-6`

ROOT−6 contingency

- *Hope for the best, Plan for the worst*
- if we don’t manage to pull out ROOT−6 (and even if we do)
- what is the expected LTS end date for ROOT−5 ?