
# Use of ROOT in Geant4

A.Dotti, SLAC
I. Hrivnacova, IPN Orsay
W. Pokorski, CERN

ROOT Users Workshop,
11 - 14 March 2013, Saas-Fee

IPN INSTITUT DE PHYSIQUE NUCLÉAIRE ORSAY

SLAC NATIONAL ACCELERATOR LABORATORY

CERN

# Outline

- Analysis tools in Geant4
- Use of Root in Geant4 testing
- Experience with Root in multi-threading programs

# Analysis Tools in Geant4

- AIDA based tools
- New Geant4 analysis tools
- ROOT

# AIDA Based Tools

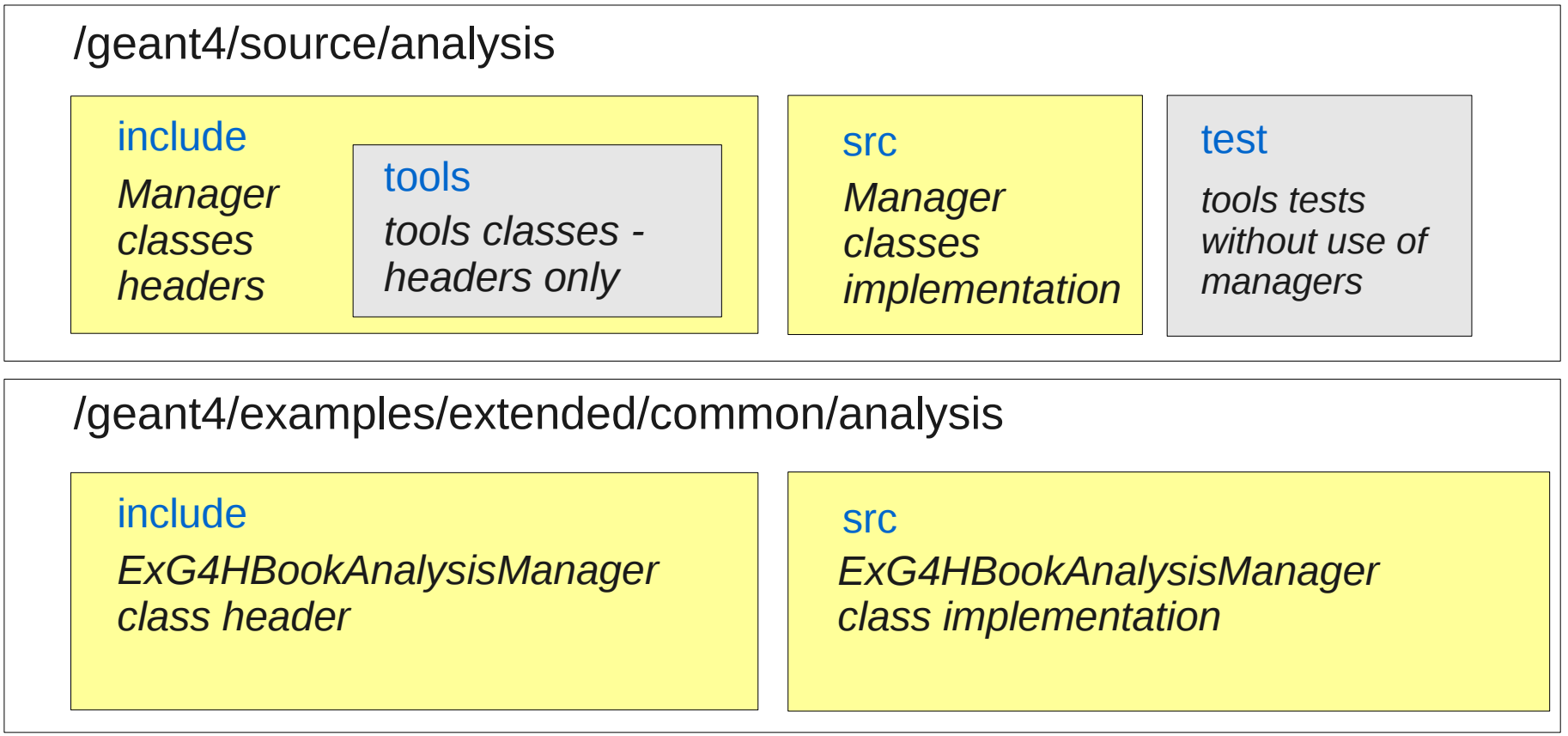
- Historically first analysis tools in Geant4 examples
- Based on AIDA = Abstract Interfaces for Data Analysis
  - Since Geant4 3.0 release (December 2000)
  - First provided within the Geant4 example extended/analysis/AnaEx01 (jas, Lab), then available as external tools
  - The AIDA compliant tools (linked in the Geant4 Guide for Application Developers):
    - JAS, iAIDA, Open Scientist Lab, rAIDA
  - Not all kept maintained, not all implement the AIDA interfaces completely
  - Not always easy to be installed & used
    - See Geant4 user forum, Analysis category
  - Still supported with Geant4 9.6 (November 2012)

# New Analysis Tools

- New analysis category in Geant4 since Geant4 9.5 (December 2011)
- Based on g4tools from inlib/exlib developed by G. Barrand (LAL):
  - http://inexlib.lal.in2p3.fr/
  - “Pure header code” - all code is inlined
  - Can be installed on iOS, Android, UNIXes, Windows
  - Provides code to write histograms and “flat ntuples” in several formats: ROOT, XML AIDA format, CSV for ntuples. HBOOK
- Complete migration to g4tools in all Geant4 examples in the development plan for 2013

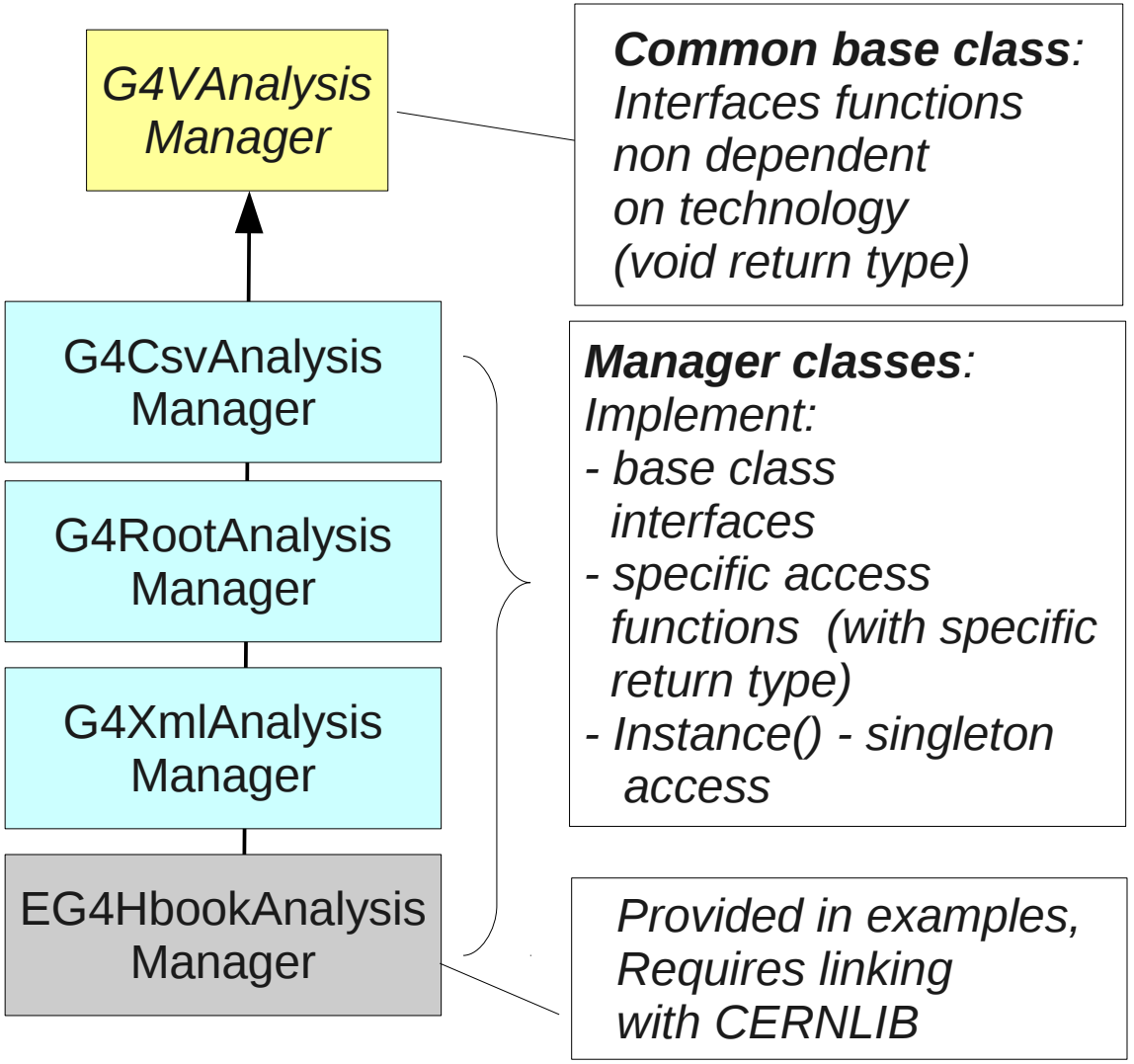
# Analysis Category

- Provides “light” analysis tools
  - Available directly with Geant4 installation
  - No need to link a Geant4 application with an external analysis package



# Analysis Managers

- Uniform interface to g4tools
  - Hide the differences according to a selected technology (Root, XML, HBOOK) from the user
- Higher level management of g4tools objects (file, histograms, ntuples)
  - Memory management
  - Access to histograms, ntuple columns via indexes
- Integration in the Geant4 framework
  - Interactive commands, units, (in)activation of selected histograms



# Example

B4RunAction.cc

```
#include "B4Analysis.hh"

void B4RunAction::BeginOfRunAction(const G4Run* run)
{
  // Get analysis manager
  G4AnalysisManager* man = G4AnalysisManager::Instance();

  // Open an output file
  man->OpenFile("exampleB4");

  // Create histogram(s)
  man->CreateH1("0","Edep in absorber", 100, 0., 800*MeV);
  man->CreateH1("1","Edep in gap", 100, 0., 100*MeV);
}

void B4RunAction::EndOfRunAction(const G4Run* aRun)
{
  G4AnalysisManager* man = G4AnalysisManager::Instance();
  man->Write();
  man->CloseFile();
}
```

B4Analysis.hh

```
#ifndef B4Analysis_h
#define B4Analysis_h 1

#include "g4root.hh"
//#include "g4xml.hh"
//#include "g4hbook.hh"

#endif
```

*Selection of the output format at a single place*

B4EventAction.cc

```
#include "B4Analysis.hh"

void N4EventAction::EndOfEventAction(const G4Run* aRun)
{
  G4AnalysisManager* man = G4AnalysisManager::Instance();
  man->FillH1(0, fEnergyAbs);
  man->FillH1(1, fEnergyGap);
}
```

*Histogram IDs are attributed automatically*

Fee

# Example (2)

- A set of Geant4 commands which can be used to create histograms or set their properties dynamically
  - Most of commands were defined according to manager classes defined specifically in each example
  - The examples specific manager classes could be then removed or reduced significantly

gammaSpectrum.mac in TestEm5

```
/analysis/setFileName gammaSpectrum
/analysis/h1/set  3 200 0.01 10 MeV          #gamma: energy at vertex
/analysis/h1/set  5 200 0.01 10 MeV log10  #gamma: energy at vertex (log10)
/analysis/h1/set 20 200 0 6 MeV               #gamma: energy at exit
/analysis/h1/set 40 200 0 6 MeV               #gamma: energy at back
```

# Examples With ROOT

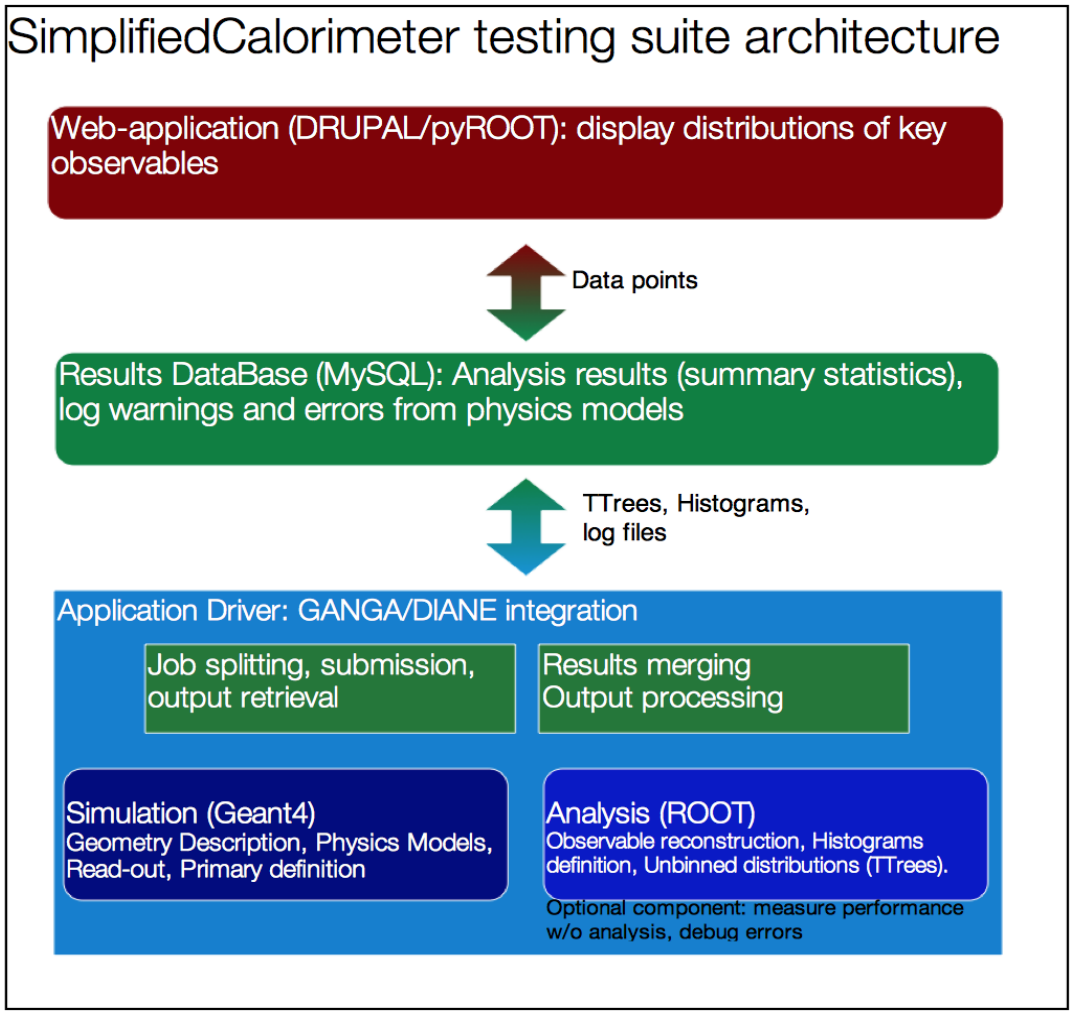
- The Geant4 applications with use of ROOT classes are demonstrated in extended examples:
  - analysis/AnaEx02 – demonstration of use of Root histograms and ntuples
    - AnaEx01 – same with g4tools; AnaEx03 – same with AIDA
    - http://geant4.web.cern.ch/geant4/UserDocumentation/Doxygen/examples_doc/html/Examples_analysis.html (link)
  - persistency/P01, P02
    - Root I/O examples for storing and retrieving calorimeter hits (P01) and geometry objects (P02)
    - Storing objects in a file using the 'reflection' technique for persistency provided by the Reflex tool
    - *The generation of the Reflex dictionary fails for Geant4 geometry classes using c-array with dynamic size declared via a variable of size_t type (as Reflex requires int) and therefore saving the Geant4 geometry with ROOT I/O is currently not possible*
    - Reviewing these examples with changes for Geant4 MT

# Use of Root in Geant4 testing

# Simplified Calorimeter Application

- SimplifiedCalorimeter application:
  - Simplified versions of HEP calorimeters implemented with Geant4
  - All LHC calorimeter materials and technologies
  - The most important variables for calorimetric measurements (response, resolution and shower shapes) are reconstructed and recorded for analysis.
  - This application is used to test and verify physics improvements and new developments.
- Geant4 testing:
  - A limited sample of about 10 millions events every month with each internal Geant4 development tag; and a sample at least 10 times larger for the June (beta) and the November (production) releases
  - Scattered in about 2k - 10k jobs each producing 2 ROOT files

# Use of ROOT in Testing Suite




1) Application level: the application produces histograms and trees that are saved in ROOT files
2) Python (pyROOT) program: performs merging and analysis of the produced ROOT files and saves the result in a MySQL DB
3) Web application (DRUPAL module + 2nd pyROOT program) to produce plots comparing, for a given quantity, different versions of Geant4

# Use of ROOT in Testing Suite (2)

- Strength points:
  - We are pretty satisfied with it and we never had big issues.
  - Since our GRID system is python based, we appreciate the possibility to integrate ROOT via pyROOT (any strengthen of python and ROOT integration is welcome).
- Weak points:
  - For the web-application part since we want to integrate with a DRUPALweb-site prototype *we need to use php*, the integration with the pyROOT script that reads the DB and produces TGraphs is a bit cumbersome and complex.
  - We may need to review this part and evaluate other histogramming facilities that are more web-friendly. Any idea?

# Experience with ROOT in Multi-threading programs

- Simplified Calorimeter Application
- Geant4 VMC

# Geant4 MT

- Geant4MT aims to reduce the memory footprint by sharing the largest data structures in Geant4
- Key requirements for Geant4MT
  - Bit-level compatibility of results with the sequential version - given the same starting state of a pseudo Random Number Generator (pRNG) for each event
  - Simple porting of applications
  - Efficient use of multi-core and many-core hardware though good scaling of performance.
- Geant4 MT prototype
  - Public releases: 31 October 2011 based on 9.4.p01 and 13 August 2012 based on 9.5.p01
- To be provided with standard Geant4 distribution since the next Geant4 release 10.0 (December 2013)

# Simplified Calorimeter MT Application

- SimplifiedCalorimeter application uses ROOT for histograms, N-tuples and linear algebra calculations (matrix diagonalization)
  1) Start of Run:
     - Book of histograms and TTrees (simple variables, C-array and std::vector<double>)
  2) Event Loop:
     - TH::Fill(), TTree::Fill() ; creation, filling and manipulation of temporary histograms
  3) End of Run:
     - Run summary and analysis,TFile::Write()

# Simplified Calorimeter MT Application (2)

- Multi-threaded application
  - Each G4 thread simulates a subset of the total number of events.
  - Each thread also performs analysis: event reconstruction, filling of histograms and of two TTree
- Since the goal is to have a testing application to keep the code as simple as possible each thread is independent (no shared data):
  - Each object has its own instances of each object (TH*, TTree, TFile) needed for the analysis
  - Each thread will write out a separate file (unique name contains thread-id)

# Simplified Calorimeter MT Application (3)

- Problems with concurrent access to (hidden) shared resources make several steps non thread-safe:
  - new TH* , new TTree(), TTree::SetBranchAddress() ,TFile::Write
  - *ROOT's TH3 replaced with g4tools::histo object that is thread safe*
  - *For the TTree methods a ROOT's forum entry discusses a possible work-around: to be tested*
- TTree::Fill is thread-safe only if no std::vectors are used
  - *The work-around for the previous item may also solve this problem*
- Added explicit lock via global mutex around critical sections of the code (same method used as in a ROOT's tutorials of TThread)
- A stand-alone (no Geant4) application that shows these problems is available

# Geant4 VMC MT

- An independent experience with Geant4 MT prototype and ROOT IO also from tests with Geant4 VMC
  - The ROOT IO – an integral part of the VMC application
- Geant4 VMC MT - the same approach as in Geant4 MT
- Singleton objects -> singletons per thread
  - Both TGeant4 and UserMCApplication are instantiated per each thread
- Added a new function in VMC (available since Root v5.34/00)
  - TVirtualMC::InitMT(Int_t threadRank)

# Geant4 VMC MT + ROOT IO

- The same approach as in SimplifiedCalorimeter test:
  - Each thread opens and writes its own ROOT file
  - No need for final merge: user analysis can chain files
- Geant4 MT + ROOT IO:
  - Example ParN02Root in Geant4 MT branch: adds Root IO to ParN02
  - Added classes for ROOT IO management and locking: RootManagerMT, RootMutex
  - Use of TThread,
  - *Locking Root IO needed till first TTree::Fill in each thread*
- Geant4 VMC MT + ROOT IO
  - TMCRootManager* classes in E02 (but not specific to E02)

# Conclusions

- While the Geant4 framework itself is independent from any choice of analysis tool, the Geant4 user developing his own application can include ROOT analysis in several ways:
  - Via g4tools, direct use of ROOT classes or via AIDA compliant tools
  - All these options are demonstrated in the Geant4 examples.
- ROOT is successfully used in Geant4 testing
  - As the test application analysis tools, via pyROOT programs and finally also in a Web application
- The integration of ROOT in multi-threading applications is not straightforward
  - More effort required to solve remaining problems