Introduction a ROOT

ENVOL 2013 Biarritz
21 Janvier 2013
Rene Brun/CERN
ROOT in a nutshell

- An efficient data storage and access system designed to support structured data sets in very large distributed data bases (Petabytes).
- A query system to extract information from these distributed data sets.
- The query system is able to use transparently parallel systems on the GRID (PROOF).
- A scientific visualisation system with 2-D and 3-D graphics.
- An advanced Graphical User Interface
- A C++ interpreter allowing calls to user defined classes.

An Open Source Project
ROOT: An Open Source Project

- The project is developed as a collaboration between:
  - Full time developers:
    - 8 people full time at CERN
    - 2 developers at FermiLab (Chicago)
  - Many contributors spending a substantial fraction of their time in specific areas (> 50).
  - Key developers in large experiments using ROOT as a framework.
  - Several thousand users given feedback and a very long list of small contributions.
ROOT Application Domains

- Data Analysis & Visualization
- Data Storage: Local, Network

General Framework
ROOT: a Framework and a Library

User classes
- User can define new classes interactively
- Either using calling API or sub-classing API
- These classes can inherit from ROOT classes

Dynamic linking
- Interpreted code can call compiled code
- Compiled code can call interpreted code
- Macros can be dynamically compiled & linked

This is the normal operation mode

Interesting feature for GUIs & event displays

Script Compiler
root > .x file.C++
A Shared Library can be linked dynamically to a running executable module
- either via explicit loading,
- or automatically via plug-in manager

A Shared Library facilitates the development and maintenance phases

---

Dynamic linking from Shared libraries

The "standard" ROOT executable module can dynamically load user's specific code from shared libraries.

```
Root > gSystem->Load("libNA49")
Root > gSystem->Load("libUser")
Root > T49Event event
Root > event.xxxxxxx
```
Plug-in Manager

- User Shared lib
- Exp Shared libs
- General Utility

- Basic Services, GUI, Math...
- Plug-in manager
- I/O manager
- Interpreter

Object Dictionary

21 Janvier 2013
CINT: the C++ interpreter
My first session

root

root 344+76.8
(const double)4.208000000000000010e+002
root float x=89.7;
root float y=567.8;
root x+sqrt(y)
(const double)1.13528550991510710e+002
root float z = x+2*sqrt(y/6);
root z
(float)1.09155929565429690e+002
root .q

See file $HOME/.root_hist

root try up and down arrows
My second session

```
root .x session2.C
for N=100000, sum= 45908.6
root sum
(double)4.59085828512453370e+004
root r.Rndm()
(Double_t)8.29029321670533560e-001
root .q
```

session2.C

```c
{ int N = 100000;
  TRandom r;
  double sum = 0;
  for (int i=0;i<N;i++) {
    sum += sin(r.Rndm());
  }
  printf("for N=%d, sum= %g\n",N,sum);
}
```
My third session

```
void session3 (int N=100000) {
    TRandom r;
    double sum = 0;
    for (int i=0;i<N;i++) {
        sum += sin(r.Rndm());
    }
    printf("for N=%d, sum= %g\n",N,sum);
}
```

Named macro
Normal C++ scope rules
My third session with ACLIC

```c
#include "TRandom.h"

void session4 (int N) {
    TRandom r;
    double sum = 0;
    for (int i=0;i<N;i++) {
        sum += sin(r.Rndm());
    }
    printf("for N=%d, sum= %g\n",N,sum);
}
```

File session4.C
Automatically compiled and linked by the native compiler.
Must be C++ compliant.
**Interpreter & Compiler integration**

- `root > .x script.C`
- `root > DoSomething(...);`
- `root > .x script.C++`
- `root > .x script.C++`

---

- `gROOT->ProcessLine(".L script.C++");`
- `gROOT->ProcessLine("DoSomething(...)");`

**Actions:**
- Execute file `script.C`
- Execute function `DoSomething`
- Compile file `script.C` and execute it
- Compile file `script.C` if file has been modified. Execute it

**Remark:** same from compiled or interpreted code
h.Draw() – local mode

CINT

libCore ------
... I/O
TSystem ...

libX11 ------
... drawline
drawtext ...

libGpad ------
... TPad
TFrame ...

(Plug-in Manager)

libCore

libHist ------
... TH1
TH2 ...

libHistPainter ------
... THistPainter
TPainter3DAlgorithms ...

libHistPainter

pm pm pm pm
The new CERNLIB/MATHLIB

With better and better algorithms

Soft evolution from LHC to CLIC

René Brun, Ingrid 2012 Mumbai
GUI
GUI (Graphical User Interface)
Example of GUI based on ROOT tools
Each element is clickable
GUI Examples

ROOT Shower Monte Carlo
Event Display

Energy loss for each particle

Statistics

<table>
<thead>
<tr>
<th>Entry</th>
<th>Mean</th>
<th>RMS</th>
<th>z-norm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.651e+16</td>
<td>779e+13</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>0.001e+06</td>
<td>0.006e+06</td>
<td>0.96</td>
</tr>
</tbody>
</table>

Particle: gamma, E = 1.000e+02
void bexec(TString &dir, char *macro) {
    if (gROOT->IsBatch()) printf("Processing benchmark: \%s\%s\n", dir.Data(), macro);
    TPaveText *summary = (TPaveText*)bench->GetPrimitive("TPave")
    TText *tmacro = summary->GetLineWith(macro);
    if (tmacro) tmacro->SetTextColor(4);
    bench->Modified(); bench->Update();
    gROOT->Macro(Form("%s\%s", dir.Data(), macro));
    TPaveText *summary2 = (TPaveText*)bench->GetPrimitive("TPave")
    TText *tmacro2 = summary2->GetLineWith(macro);
    if (tmacro2) tmacro2->SetTextColor(2);
    bench->Modified(); bench->Update();
}

void benchmarks() {
    TString dir = gSystem->UnixPathName(gInterpreter->GetCurrentMachiningDir().ReplaceAll("/benchmarks", ""));
    C:\home\bellenot\root\tutorials\benchmarks.C: 59 lines read.
}

GUI Examples II

Can browse a ROOT file on the remote web server
The GUI builder provides GUI tools for developing user interfaces based on the ROOT GUI classes. It includes over 30 advanced widgets and an automatic C++ code generator.
When pressing `ctrl+S` on any widget it is saved as a C++ macro file thanks to the `SavePrimitive` methods implemented in all GUI classes. The generated macro can be edited and then executed via CINT.

Executing the macro restores the complete original GUI as well as all created signal/slot connections in a global way.
Combining UI and GUI

```
for N=100000, sum = 45908.6

(root)(double)4.59085828512453370e+004

(r.Random())8.29029321670533560e-001
```

21 Janvier 2013
Graphics
2-D Graphics

- New functions added at each new release.
- Always new requests for new styles, coordinate systems.
- `ps`, `pdf`, `svg`, `gif`, `jpg`, `png`, `c`, `root`, etc.
A Data Analysis & Visualisation tool

- **Efficiency for Good Vertices**
  - Nent = 22
  - Mean = 1.71
  - RMS = 0.6955
  - Under = 0.06522
  - Over = 0

- **V0c Effective Masses**
  - Nent = 9056
  - Mean = 0.1808
  - RMS = 0.02553
  - Under = 596
  - Over = 16
  - Chi^2 / ndf = 4.503 / 5
  - Goodness = 0.644 7.834
  - Mean = 0.0976 ± 0.00337
  - Sigma = 0.0896 ± 0.00135

- **dEdX vs Pmod**
  - Protons
  - Kaons
  - Pions
  - Electrons

- **Momentum 730-830 MeV/c**
  - Pions
  - Kaons
  - Protons
Graphics : 1,2,3-D functions

Lorentzian Peak on Quadratic Background

-sin(x^2+y^2+z^2-36)

Examples of Surface options

\(x^2+y^2-x^3-6x^2y+y^4\)

\(x^2+y^2-x^3-8x^2y+y^4\)
Full LaTeX support on screen and postscript

Formula or diagrams can be edited with the mouse

TCurlyArc
TCurlyLine
TWavyLine and other building blocks for Feynmann diagrams
Image processing

ACAT2010 in Jaipur

ROOT reflexion system
3-D Graphics

Detector hits

Mathematical objects

ROOT: design, development, evolution

René Brun, ACAT2010, Jaipur
Graphics (2D-3D)

TGLParametric

“LEGO”

“SURF”
The Geometry Package TGEO

- The **TGEO** classes are now stable (thanks **ALICE**).
- Can work with different simulation engines (G3, G4, Fluka) (See Virtual Monte Carlo)
- **G3-\(\rightarrow\)TGEO, G4-\(\rightarrow\)TGEO, TGEO\(\leftarrow\)GDML**
- Used in online systems and reconstruction programs
- Built-in facilities for alignment

**Impressive gallery of experiments (35 detectors in $\text{ROOTSYS/test/stressGeometry}$)**
OpenGL

see $ROOTSYS/tutorials/geom
Alice 3 million nodes
ROOT 3D Graphics

Atlas
Input/Output
Data Sets types

- Simulated data
  - 10 Mbytes/event

- Raw data
  - 1 Mbyte/event

- Event Summary Data
  - 1 Mbyte/event

- About 10 data sets for 1 raw data set

- Analysis Objects Data
  - 100 Kbytes/event

- A few reconstruction passes

- Several analysis groups
  - Physics oriented

1000 events/data set

About 10 data sets for 1 raw data set
Each experiment will take about \textbf{1 billion events/year}.

1 billion events $\Rightarrow$ \textbf{1 million raw data sets} of 1 Gbyte

$\Rightarrow$ \textbf{10 million data sets} with ESDs and AODs

$\Rightarrow$ \textbf{100 million data sets} with the replica on the GRID

All event data are \textbf{C++ objects} streamed to \textbf{ROOT files}.
Object Persistency (in a nutshell)

- Two I/O modes supported (**Keys** and **Trees**).
- **Key access**: simple object streaming mode. A ROOT file is like a Unix directory tree. Very convenient for objects like histograms, geometries, mag.field, calibrations

**Trees**
- A generalization of ntuples to objects. Designed for storing events
- split and no split modes
- query processor

**Chains**: Collections of files containing Trees

ROOT files are self-describing

Interfaces with RDBMS also available

Access to remote files (RFIO, DCACHE, GRID)
I/O

Object in Memory

Streamer: No need for transient / persistent classes

Buffer

sockets
Net File

http
Web File

XML
XML File

SQL
DataBase

Local

File on disk
A ROOT file `pippa.root` with 2 levels of sub-directories

Objects in directory
`/pippa/DM/CJ`

eg:
`/pippa/DM/CJ/h15`
File types & Access

Local File
X.xml

TFile
TKey/TTree
TStreamerInfo

rootd/xrootd

http

Local File
X.root

hadoop

Chirp

Dcache

Castor

Oracle

MySQL

PgSQL

SapDb

TTreeSQL

user

TSQLServer
TSQLRow
TSQLResult

21 Janvier 2013
Self-describing files

- Dictionary for persistent classes written to the file.
- ROOT files can be read by foreign readers
- Support for **Backward** and **Forward** compatibility
- Files created in 2001 must be readable in 2015
- Classes (data objects) for all objects in a file can be regenerated via **TFile::MakeProject**

```c
Root > TFile f("demo.root");
Root > f.MakeProject("dir","*","new++");
```
TFile::MakeProject

Generate C++ header files and shared lib for the classes in file

```
TFile *falice = TFile::Open("http://root.cern.ch/files/alice_ESDs.root");
falice->MakeProject("alice","*","++");
```

MakeProject has generated 26 classes in alice

Shared lib alice/alice.so has been generated

Shared lib alice/alice.so has been dynamically loaded

```
ls alice
```

```
AliESDCaloCluster.h
AliESDZDC.h
AliESDTrigger.h
AliESDcascade.h
AliESDEvent.h
AliESDfriend.h
AliESDFMD.h
AliESDfriendTrack.h
AliESDHeader.h
AliESDkink.h
AliESDMuonTrack.h
AliESDtrack.h
AliESDpmdTrack.h
AliESDv0.h
AliESDRun.h
AliExternalTrackParam.h
AliESDTZERO.h
AliFMDFloatMap.h
AliESDTrdTrack.h
AliFMDMap.h
AliESDVZERO.h
AliMultiplicity.h
AliESDVertex.h
AliRawDataErrorLog.h
```

Rene Brun: Introduction to ROOT
ROOT Trees
The bulk data containers
Memory $\leftrightarrow$ Tree

Each Node is a branch in the Tree
Browsing a TTree with TBrowser

- 8 leaves of branch Electrons

- 8 branches of T

- A double click to histogram the leaf

- Image shows a ROOT Object Browser with TTree navigation.
A TFile typically contains 1 TTree

A TChain is a collection of TTrees or/and TChains

A TChain is typically the result of a query to the file catalogue
Queries to the data base

- via the GUI (TBrowser of TTreeViewer)
- via a CINT command or a script
  ```
  tree.Draw("x", "y<0 && sqrt(z)>6");
  tree.Process("myscript.C");
  ```
- via compiled code
  ```
  chain.Process("myselector.C+");
  ```
- in parallel with PROOF
PROOF and GRID(s) interface
Systems in 2030?

- **End user Analysis software**: 1 MLOC
- **Experiment Software**: 50 MLOC
- **Frameworks like ROOT, Geant5**: 20 MLOC
- **OS & compilers**: 100 MLOC
- **Multi-level parallel machines 10000x1000x1000**

**Hardware**
- **RAM**: 10 TB
- **Disks**: 1000 PB
- **Networks**: 10 Tbit/s
- **CLOUDS on demand**
- **GRIDS**
Keyword: parallelism
Grid(s) evolution (maybe)
The PROOF Approach

- Cluster perceived as extension of local PC
- *Same macro and syntax* as in local session
- More *dynamic* use of resources
- Real-time feedback
- Automatic *splitting* and *merging*
Current Developments
ROOT User Workshop
Saas-Fee, Switzerland

Topics
ROOT roadmap
The new cling C++11 interpreter
New persistency features
Concurrency and ROOT
Distributed data analysis with PROOF and PoD
ROOT on mobile
ROOT in Javascript
Latest on math tools and techniques
Moving ROOT to C++11
User feedback

Organising Committee
Philippe Canal, FNAL
Federico Carminati, CERN
Axel Naumann, CERN
Edmond Offermann
Tania Pardo, CERN
Fons Rademakers, CERN

Featuring ROOT 6
The Next Generation

Since almost two decades, ROOT has established itself as the framework for HENP data processing and analysis. The LHC upgrade program and the new experiments being designed at CERN and elsewhere will pose even more formidable challenges in terms of data complexity and size. The new parallel and heterogeneous computing architectures that are either announced or already available will call for a deep rethinking of the code and the data structures to be exploited efficiently.

This workshop, following from a successful series of such events, will help shaping the future evolution of ROOT.
The LLVM system

- We are having big hopes with LLVM (open source project with people from Apple, Adobe, Google, etc).

- LLVM is a gcc compatible compiler, parser and JIT (Just In Time) compiler.

- Our idea is that the interpreter will be the compiler itself (C++0x compliant) and offering many additional possibilities like quality and speed for important classes like TFormula, TTreeFormula, etc

- We have a working prototype today, but a fully operational use of LLVM in ROOT is probably 1 year away.
Graphics News

- Full LaTeX support on screen, pdf, gif, png, etc
- Direct interface with Cocoa on MAC (instead of X11)
- Support for new devices (IOS, Android)
- ROOT files browsed by any web browser via javascript.
\[ \Pi_{j \geq 0} \left( \sum_{k \geq 0} a_{jk} z^k \right) = \sum_{n \geq 0} z^n \left( \sum_{k_0, k_1, \ldots \geq 0} a_{0k_0} a_{1k_1} \cdots \right) \]

\[ \mathcal{W}_{\delta_1 \rho_1 \sigma_2}^{3\beta} = U_{\delta_1 \rho_1 \sigma_2}^{3\beta} + \frac{1}{8\pi^2} \int_{a_1}^{a_2} d\alpha' \left[ \frac{U_{\delta_1 \rho_1}^{2\beta} - a'_2 U_{\rho_1 \sigma_2}^{1\beta}}{U_{\rho_1 \sigma_2}^{0\beta}} \right] \]

\[ d\Gamma = \frac{1}{2m_A} \left( \prod_f \frac{d^3 p_f}{(2\pi)^3} \frac{1}{2E_f} \right) |\mathcal{M}(m_A - \{p_f\})|^2 (2\pi)^4 \delta^{(4)}(p_A - \sum p_f) \]

\[ 4 \text{Re}\left\{ \frac{2}{1-\Delta a} \chi(s) \left[ \hat{g}_V^e \hat{g}_V^f (1 + \cos^2 \theta) + \hat{g}_a^e \hat{g}_a^f \cos \theta \right] \right\} \]

\[ p(n) = \frac{1}{n \sqrt{2}} \sum_{k=1}^{\infty} \sqrt{k A_k(n)} \frac{d}{dn} \frac{\sinh \left\{ \frac{\pi}{k \sqrt{3}} \sqrt{n - \frac{1}{24}} \right\}}{\sqrt{n - \frac{1}{24}}} \]

\[ \frac{(\ell + 1) C_{\ell}^{TE}}{2\pi} \quad \mathbb{N} \subset \mathbb{R} \quad \text{RHIC スピン物理 Нью-Йорк} \]
ROOT objects in any web browser

Read a ROOT file with Javascript

Select a ROOT file to read, or enter a url (*):
*: Other URLs might not work because of cross site scripting protection, see e.g. http://developer.mozilla.org/en/http_access_control on how to avoid it.

hsimple.root

Load  Reset

JSROOTIO.RootFile.js version: 1.6 2012/02/24
load: hsimple.root

This is the px distribution

Entries = 24001
Mean = 0.005094
RMS = 1.003

Histogram from a remote ROOT file shown in the browser
Trends in GUIs

IPAD style GUI In ROOT-dev on IPAD
I/O

- Support for parallelism (one file written by multiple threads)
GRIDs and Parallelism

The GRIDs are beautiful parallel systems running conventional non-parallelized jobs.

However the current way is far to be optimal because GRID centers run typically one job per core instead of one job per node and each job produces one or more output files.
GRIDs and Parallelism

It would be more efficient (memory and I/O) to run one job per node using all nodes and producing one file.

This requires changes (thread-safety) in the application and the use of the ROOT buffer merger.

Move file to DS
Parallelism everywhere

What about this picture in the very near future?

- **Local cluster**
  - 1000x16 cores

- **Remote clusters**

- **Cache**
  - 1 PByte
  - 1 TByte

- **File catalogue**
  - 8 cores

Rene Brun: Introduction to ROOT
Summary

- The ROOT system is the result of 18 years of cooperation between the development team and thousands of heterogeneous users.

- ROOT is not only a file format, but mainly a general object I/O storage and management designed for rapid data analysis of very large shared data sets.

- It allows concurrent access and supports parallelism in a set of analysis clusters (LAN and WAN).