The LCG Software
and the ROOT Framework

LCG seminar
CERN- 7 November
René Brun

ftp://root.cern.ch/root/lcgtalk.ppt
Plan of talk

- General considerations
- ROOT status & plans
  - what we presented at the Blueprint RTAG
  - required developments
  - modularity issues
- Relations with LCG
  - Where we agree
  - Possible problems
  - Scenario 1, 2, 3
General Considerations

Some time-invariant messages

The perfect program is a rock.
There are no inputs.
There are no outputs.
There are no errors!

"When you watch me, they want me to do it differently."
Some basic rules (1)

When starting a new project, many unexpected obstacles. Must explore many ways before building the final path.
Some basic rules (2)

The development model is crucial.
A “well-designed” system that looks pretty on paper, but not validated early enough by users may end up into a very fat and unusable system.
Some basic rules (3)

Cooperation between projects is difficult.
Different development styles, Team experience.
One more reason for rapid prototyping. Do not wait the end to see the product. Develop the product with users.
Some basic rules (4)

If you have more than $\pi^2$ years experience in managing software projects, multiply by $\pi$ your estimation to complete your next project else multiply by $\pi^2$

Nine women working together have never made a baby in one month.

If $T$ is the time to implement a system and $N$ the number of participants $T = a/N - b*N^2$

Ideal team size 3 to 6 people
Time to develop
ROOT status/plans

The team
Measuring success
The main components
The development plans
Project History

- Jan 95: Thinking/writing/rewriting/???
- November 95: Public seminar, show Root 0.5
- Spring 96: decision to use CINT
- Jan 97: Root version 1.0
- Jan 98: Root version 2.0
- Mar 99: Root version 2.21/08 (1st Root workshop FNAL)
- Feb 00: Root version 2.23/12 (2nd Root workshop CERN)
- Mar 01: Root version 3.00/06
- Jun 01: Root version 3.01/05 (3rd Root workshop FNAL)
- Jan 02: Root version 3.02/07 (LCG project starts: RTAGs)
- Oct 02: Root version 3.03/09 (4th Root workshop CERN)
ROOT Project basic principles

- Born with minimal CERN staff (by constraint)
- Try to involve as many developers as possible from outside CERN => Open Source project
- Try to get as many users asap to validate the ideas. We knew many PAW users.
- Release early, Release often principle
- Target maximum portability (OS & compilers)
- Target maximum functionality and simplicity
- Roottalk newsgroup: vital importance
ROOT Team & Associates

ROOT Team:
- Ilka Antcheva (LCG staff) (since 1st Aug 2002)
- Rene Brun: new SFT group and Alice part time
- Philippe Canal (FNAL/CD) (since 1998)
- Olivier Couet CERN/IT/API (from PAW) (since 1st Jun 2002)
- Gerardo Ganis (LCG/EP/SFT) starting just now
- Masa Goto (Agilent technologies/Japan)
- Valeriy Onuchin (LCG project associate) (since 1st Feb 2002)
- Fons Rademakers: Alice and new SFT group

External Associates
- Bertrand Bellenot (Alcan) Win32gdk (since June 2000)
- Maarten Ballintijn (MIT) PROOF (since Sep 2001)
- Andrei Gheata: (Alice) Geometry package (since Sep 2001)
ROOT Team & Associates (2)

- Now in the LCG
  - Valery Fine (BNL/Atlas) TVirtualX/Qt
  - Victor Perevoztchikov (BNL/Atlas) STL, foreign classes
- More than 50 important contributions from people spending a substantial fraction of their time on the project. See $ROOTSYS/README/CREDITS
- Special thanks to Suzanne Panacek who did a great job with the ROOT Users Guide, tutorials, lectures.
  - Printed copies of the Users Guide in my office.
- Many thanks to FNAL computing Division for the continuous support of the project since 1998.
ROOT ftp Downloads

More than 60,000 downloads of the Users Guide in 2 years

166,000 downloads of binary tar files since 1997

4700 mails to RootTalk since January 2002

Download per platform:

- Unix
- Windows
- Linux
Source & Binary distributions

- Intel x86 Linux for Redhat 7.2 and gcc 3.2, version 3.03/07 (10.9 MB).
- Intel x86 Linux for Redhat 7.2 and gcc 2.96, version 3.03/07 (11.2 MB).
- Intel x86 Linux for Redhat 7.2 and gcc 2.95.3, version 3.03/09 (11.5 MB).
- Intel x86 Linux for Redhat 7.2 and Intel icc 6, version 3.03/07 (16.2 MB).
- Intel x86 Linux for Redhat 6.1 (glibc 2.1) and gcc2.95.2, version 3.03/09 (12.9 MB).
- Intel x86 Linux for Redhat 6.1 (glibc 2.1) and egcs 1.1.2, version 3.03/09 (11.2 MB).
- Intel x86 Linux for Redhat 5.0/5.1/5.2 (glibc) and egcs 1.1.2, version 3.03/09 (10.9 MB).
- Intel Itanium Linux for Redhat 7.1 (glibc 2.2) and gcc 2.96, version 3.02/06 (9.0 MB).
- HP PA-RISC HP-UX 11.20 with aCC (v1.18), version 3.02/09 (16.8 MB).
- HP Itanium HP-UX 11.20 with aCC, version 3.02/07 (16.8 MB).
- Compaq Alpha OSF1 with ccx 4.2, version 3.03/09 (12.1 MB).
- Compaq Alpha OSF1 with egcs 1.1.2, version 3.03/09 (14.2 MB).
- Compaq Alpha Linux with egcs 1.1.2, version 3.02/06 (11.0 MB).
- Compaq iPAQ PocketPC Linux with gcc 2.95, version 3.02/06 (7.0 MB).

For more on Linux on iPAQ see www.handhelds.org.

- IBM AIX 4.3 with xC version 5, version 3.03/09 (13.0 MB, works only on AIX 4.3).
- Sun SPARC Solaris 5.6 with CC4.2, version 3.02/06 (8.7 MB). It cannot be used with Solaris 5.7 or 5.8 even using the same compiler version. You must recompile from the source on these two systems.
- Sun SPARC Solaris 5.7 with CC5.2, version 3.03/09 (13.9 MB). It cannot be used with Solaris 5.6 or 5.8 even using the same compiler version. You must recompile from the source on these two systems.
- Sun SPARC Solaris 5.8 with CC5.2, version 3.03/09 (13.6 MB).
- Sun SPARC Solaris 5.8 with CC5.2, version 3.03/09 (13.6 MB).
- Sun SPARC Solaris 5.8 with CC5.2, version 3.03/09 (13.6 MB).
- Sun SPARC Solaris 5.8 with CC5.2, version 3.03/09 (13.6 MB).
- Solarioid 6.5 with CC, version 3.03/09 (compiled with -m68k) (12.8 MB).
- Solarioid 6.5 with cc++ 2.95.2, version 3.02/09 (14.5 MB).
- Solarioid 6.5 with CC, version 3.03/09 (13.3 MB).
- LinuxPPC (Suse/2.3) gcc 2.95.2, version 3.03/07 (10.5 MB).

Thanks to Darril Baskan for building this version.

MacOS X 10.1 for more info see these pages from Richard Fjeld.

- Windows NT/XP/2000 with VC++ 6.0, version 3.05/08 (good old tar file) **WIN32.GDK** (12.9 MB).

This version is compiled and linked with the GDK driver implemented by Bertrand Sabanet. This is still an experimental version:

- Advantages: Same GUI and look/feel as on Unix
- Disadvantages: cannot use MSDOS shell slower

- Windows NT/XP/2000 with VC++ 6.0, version 3.05/08 (good old tar file) (12.6 MB).
- Windows NT/XP/2000 with VC++ 6.0, version 3.05/08 (good old tar file) (22.3 MB).
- Windows NT/XP/2000 with VC++ 6.0, version 3.05/08 (built with InstallShield) (12.6 MB).
## Cost to develop

### Estimated Value of the main software packages using the SlocCount tool (CoCoMo method)


<table>
<thead>
<tr>
<th>Package</th>
<th>Lines of code</th>
<th>Person Years</th>
<th>Number Years</th>
<th>Number Developers</th>
<th>Total cost $ millions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minuit</td>
<td>5913</td>
<td>1.29</td>
<td>0.59</td>
<td>2.19</td>
<td>0.174</td>
</tr>
<tr>
<td>Hbook</td>
<td>33415</td>
<td>7.96</td>
<td>1.18</td>
<td>6.76</td>
<td>1.075</td>
</tr>
<tr>
<td>CLHEP</td>
<td>34932</td>
<td>8.34</td>
<td>1.21</td>
<td>6.96</td>
<td>1.127</td>
</tr>
<tr>
<td>Zebra</td>
<td>35058</td>
<td>8.38</td>
<td>1.21</td>
<td>6.97</td>
<td>1.135</td>
</tr>
<tr>
<td>Geant3</td>
<td>129727</td>
<td>33.09</td>
<td>2.02</td>
<td>16.34</td>
<td>4.471</td>
</tr>
<tr>
<td>PAW</td>
<td>284277</td>
<td>75.42</td>
<td>2.77</td>
<td>27.24</td>
<td>10.187</td>
</tr>
<tr>
<td>Geant4</td>
<td>339085</td>
<td>90.75</td>
<td>2.97</td>
<td>30.55</td>
<td>12.259</td>
</tr>
<tr>
<td>AliRoot</td>
<td>669419</td>
<td>185.38</td>
<td>3.91</td>
<td>47.57</td>
<td>25.039</td>
</tr>
<tr>
<td>ROOT</td>
<td>849859</td>
<td>238.15</td>
<td>4.29</td>
<td>55.56</td>
<td>32.171</td>
</tr>
</tbody>
</table>
ROOT: Powerful & Light-weight

- Tar ball < 15 Mbytes
- Source < 7 Mbytes
- Install from binary file: 1 minute
- No external dependencies
- Fast start-up (1 second)
- Low memory occupancy (< 15 Mbytes)
- Highly optimized algorithms for histograming, fitting, I/O, Tree queries, etc.
- and yet runs everywhere
The main software areas

- GRID middleware
- DAQ Online
- Object persistency
- Event Models Folders
- Object Dictionary
- Event Display
- System services
- Ntuple analysis
- GUI Toolkits
- Interpreters
- Math Libs Statistics
- Detector Simulation
- Detector Geometry
- Event Generators
- ETC...
ROOT libs

ls -l $ROOTSYS/lib

203856 libASImage.so
1273308 libCint.so
5658143 libCore.so
419481 libEG.so
152912 libEGPythia.so
160874 libEGPythia6.so
162181 libEGVenus.so
326000 libGX11.so
183065 libGX11TTF.so
2306421 libGeom.so
158895 libGeomPainter.so
1019977 libGpad.so
1602106 libGraf.so
1028762 libGraf3d.so
3669409 libGui.so
1605344 libHbook.so
1940222 libHist.so
332268 libHistPainter.so
114970 libHtml.so
167670 libMC.so
580851 libMatrix.so
319945 libMinuit.so
268321 libMySQL.so
21981 libNew.so
88438 libPgSQL.so
336736 libPhysics.so
196318 libPostscript.so
576691 libProof.so
681086 libRFIO.so
2017467 libRGL.so
177657 libRint.so
35410 libSRPAuth.so
1120731 libTable.so
312785 libThread.so
1067715 libTree.so
356186 libTreePlayer.so
409350 libTreeViewer.so
155664 libX3d.so

ls -l /cern/pro/lib

1434404 libgrafX11.a
1046944 libgraflib.a
4981896 libkernlib.a
2002460 libmathlib.a
11849762 libpacklib.a
4350440 libpawlib.a

TOTAL = 25.2 MBytes

TOTAL = 30.7 MBytes
The naive component model

PROs: In principle easy to add or replace a component because of weak coupling

In reality, you simply postpone the integration problem if the number of components N is big eg N> 50
Any box connected to many boxes

- GRID middleware
- RDBMS run/file catalogs
- Object persistence
- System services
- Histograming Fitting
- Math Libs Statistics
- Interpreters
- 2-d, 3-d graphics
- GUI Toolkits
- Detector Simulation
- Event Display
- Event Models Folders
- Ntuple analysis
- DAQ Online
Any box connected to many boxes

GRID middleware
RDBMS run/file catalogs
DAQ Online
Objects persistency
Event Models Folders
Event Display
Detector Simulation
System services
Ntuple analysis
Interpreters
Math Libs Statistics
2-d, 3-d graphics
GUI Toolkits
Event Generators
Detector Geometry
Any box connected to many boxes

- DAQ Online
- Event Models
- Folders
- Object persistency
- Event Display
- System services
- Ntuple analysis
- GUI Toolkits
- 2-d, 3-d graphics
- Detector Geometry
- Dectector Simulation
- Event Generators
- Histograming
- Fitting
- Math Libs
- Statistics
- Interpreters
- RDBMS run/file catalogs
- GRID middleware
Framework with Object bus

Object bus: Object dictionary

Data Interface (I/O): Functional Interface
Framework: Basic components

<table>
<thead>
<tr>
<th>Lib1</th>
<th>RTTI: Objects Dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input/Output</td>
</tr>
<tr>
<td></td>
<td>Stream</td>
</tr>
<tr>
<td></td>
<td>full</td>
</tr>
<tr>
<td></td>
<td>object</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Shared Memory</td>
</tr>
<tr>
<td></td>
<td>Sockets</td>
</tr>
<tr>
<td></td>
<td>Local data base</td>
</tr>
<tr>
<td></td>
<td>Remote data base</td>
</tr>
<tr>
<td></td>
<td>Gateways (Corba, RMI...)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>High Level Components</td>
</tr>
<tr>
<td></td>
<td>2-D/3-D graphics, Viewers, Browsers, Inspectors, Doc tools</td>
</tr>
<tr>
<td></td>
<td>Histogramming, Minimisation, Ntuples, Trees</td>
</tr>
<tr>
<td></td>
<td>Containers, Event iterators, Selectors</td>
</tr>
</tbody>
</table>
Object Persistency (in a nutshell)

- Two I/O modes supported (Key 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)
ROOT I/O : An Example

Program Writing

```cpp
TFile f("example.root","new");
TH1F h("h","My histogram",100,-3,3);
h.FillRandom("gaus",5000);
h.Write();
```

Program Reading

```cpp
TFile f("example.root");
TH1F *h = (TH1F*)f.Get("h");
h->Draw();
f.Map();
```
Memory <---> Tree
Each Node is a branch in the Tree
Tree Friends
Object Persistency

- Collaboration with FNAL and BNL
- Object persistence
- Collaboration with LCG ???
- Collaboration with BNL
- Grid middleware
- RDBMS run/file catalogs
- Math Statistics Interpreters
- History Fitting
- Event Factory Services

Continue current developments in ROOT I/O

- Foreign classes
- Support for STL
- Improved Trees
- TLongRefs
- Interface to Catalogs

Large files > 2 Gbytes

Implement in TStreamerInfo in interpreted mode
what is currently generated by Rootcint for STL containers
GRIDS Run/file catalogs

GRID middleware
RDBMS run/file catalogs
DAQ
Object persistence
System service
Interpreters
Analyzers
Interfaces with RDBMS
Oracle, MySQL, Postgres, etc
TSQL, TSQLResult
Interface with Grid middleware
Interface with Globus
Interface with Alien PROOF (GRID oriented)
Expecting close relationship with LCG
DAQ Online

- GRID middleware
- Multi-Thread support
  - Shared memory
  - Sockets/Monitors
  - Client/Server
  - Network classes
  - System interface/Signals
    - Interpreter
    - Histograming
    - Event Display
    - Browsers/Inspectors
    - Persistency

- Electronic logbook at FNAL
  - Well integrated with ROOT

- JavaRoot interface
  - Possibly interesting
    - Sample analysis

- Math Libs
- Statistics

LCG seminar 7 November Rene Brun

ROOT Framework
Object Dictionary(ies)

- *xml
- *idl

**CINT**
- Data Dictionary in memory
- Functions Dictionary in memory

You can do I/O
You can call functions from the CINT command line or scripts
Object Dictionary

- Follow the new C++ standard XTI proposal in the area of introspection (eXtended Type Information)
- Too much emphasis so far on the transient class dictionary. *(may be we can have one in common!!)*
- The real difficulty is the support for automatic schema evolution (persistent views and relationship with current transient classes).
- We did not discuss enough this essential area.
- Remove as much as possible automatic generated code in favor of dynamic interpretation. *Reading files without the original classes.*
Interpreters

- It was surprising to see that 3 of the 4 architects did not express any interest in the CINT command line. I may have a distorted perception of the user requirements. Please speak now on this important subject.
  - Python (like Perl) is a nice scripting language, but is inferior to CINT for what we want to do! A scripting language that can be compiled by a native compiler is essential.
    - In PAW it was not possible to compile a 5000 lines kumac.
  - It will happen to Python what we have seen with Iris Explorer a few years ago. It can eat a lot of manpower and users silently ignoring it.
  - RootPython is an interesting tool to extend ROOT to services already interfaced with Python.
  - JavaRoot could be very interesting for Java-based projects that need an interface to ROOT services.
CINT, Python, Java, C#

**CINT**: Smooth transition from interpreted code to compiled code dynamically unlinked/linked:

```
root > .x script.C (interpreted)
root > .x script.C++ (compiled)
```

Facilitate automatic interfaces to Python and Java

Current implementations are slow

Root dictionary could be better exploited to improve run time.
RootPython (Pere Mato)
Example

```python
C:\> python
...
>>> from rootmodule import *
>>> f1 = TF1('func1','sin(x)/x',0,
>>> f1.Eval(3)
0.047040002686622402
>>> f1.Derivative(3)
-0.34567505667199266
>>> f1.Integral(0,3)
1.8486525279994681
>>> f1.Draw()
<TCanvas::MakeDefCanvas>: created default TCanvas with name cl
```

- No much difference between CINT and Python!  
  P. Mato

```cint
TF1 f1("func1","sin(x)/x",0,10)
f1.Eval(3)
f1.Derivative(3)
f1.Integral(0,3)
f1.Draw()
```
Extend Java with Root libraries

- Java gets a matured Histograming API, Fitting and Physics analysis classes, a HEP specific Socket programming API
- Root reaches an even wider audience, finds a number of interpreter and scripting environments that are (re)implemented in Java (Jython, BeanShell etc.)
CINT vs Java/Python/Jython

CINT

```c
{ 
  gROOT->Reset();
  c1 = new TCanvas("c1", "A graph with error bars",
                   200,10,700,500);

  c1->SetFillColor(42);
  c1->SetGrid();
  c1->SetFrame();
  c1->SetFillColor(21);
  c1->SetBorderSize(12);

  Int_t n = 10;
  Float_t x[n] = {-0.22, 0.05, 0.25, 0.35, 0.5,
                   0.61, 0.7, 0.85, 0.89, 0.95};

  Float_t y[n] = {1.0, 2.9, 5.6, 7.4, 9.0,
                   9.6, 8.7, 6.3, 4.5, 1.0};

  Float_t ex[n] = {.05, .10, .07, .07, .04,
                   .05, .06, .07, .08, .05};

  Float_t ey[n] = {.8, .7, .6, .5, .4,
                   .4, .5, .6, .7, .8};
  gr = new TGraphErrors(n, x, y, ex, ey);
  gr->SetTitle("TGraphErrors Example");
  gr->SetMarkerColor(4);
  gr->SetMarkerStyle(21);
  gr->Draw("ALP");
  c1->Update();
}
```

Jython

```python
class errorsTest:
  def __init__(self):
    c1 = gpad.TCanvas("c1",
                      "A Graph with error bars",
                      200,10,700,500)
    c1.SetFillColor(42)
    c1.SetGrid()
    c1.GetFrame().SetFillColor(21)
    c1.GetFrame().SetBorderSize(12)

  n = 10
  x = [-0.22, 0.05, 0.25, 0.35, 0.5,
       0.61, 0.7, 0.85, 0.89, 0.95]
  y = [1.0, 2.9, 5.6, 7.4, 9.0,
       9.6, 8.7, 6.3, 4.5, 1.0]
  ex = [0.05, 0.10, 0.07, 0.07, 0.04,
        0.05, 0.06, 0.07, 0.08, 0.05]
  ey = [0.8, 0.7, 0.6, 0.5, 0.4,
        0.4, 0.5, 0.6, 0.7, 0.8]
  gr = gpad.TGraphErrors(n, x, y, ex, ey)
  gr.SetTitle("TGraphErrors Example")
  gr.SetMarkerColor(4)
  gr.SetMarkerStyle(21)
  gr.Draw("ALP")
  c1.Update()
```

LCG seminar 7 November  Rene Brun
ROOT Framework

CINT: 280 Rootmarks
JAVA: 105 Rootmarks
JYTHON: 52 Rootmarks
Gui/Graphics strategy

User/Root GUI and Graphics classes
Applications see only Abstract Interfaces

High level pad graphics

Low level screen graphics and GUI

TVirtualX

TPad

TGWin32
TGQt
TG??
TGX11
TGWin32GDK

ROOT Framework
GUI Toolkits

The ROOT event loop has proven to work with all known graphics systems: X11, Xt, Motif, Qt, Open Inventor, etc.

Consolidate the TVirtualX interface
Complete TVirtualX/Qt implementation
TVirtualX/Win32-GDK (free on Windows)
Export script from a running GUI
Build GUI from a given script
GUI editor/builder
Math Libs & Statistics

In ROOT today
- TVector2,3
- TLorentzRotation
- TLorentzVector
- TRandom,2,3
- TMatrix
- TMath
- TFeldmanCousins
- TPrincipal
- TMultidimFit
- TConfidenceLevel
- TFractionFitter

Can generate random numbers from basic distributions; gaus, poisson, etc from parametric analytic functions 1,2,3-d from histograms, 1,2,3-d

Matrix package maintained by E. Offermann (Rentec)

A collection of many algorithms CERNLIB, Numerical Recipes in C/C++
Event Display

Many algorithms classes developed by a huge user community See recent FNAL meeting and effort organized within ACAT

Would like to see an interface to GSL to Numerical Recipes in C++
Collaboration with
- Fred James
- Louis Lyons
- Sherry Towers
Histogramming & Fitting

Much more than HBOOK/PAW
Fix & var bin size for 1-d, 2-d 3-D
Profile 1, 2 & 3D
All kinds of projections, slices
Errors for all dims
Filling with strings (auto sort)
time axis
associated fitting
Random n. generation
+ auto binning
+ auto addition
support for parallelism

Histograming services
Fitting
Fairly complete no requests for extensions expected
Interpreters
Fitting
new ideas
collaborators

LG seminar 7 November Rene Brun
ROOT Framework
RooFit
A general purpose tool kit for data modeling

Wouter Verkerke (UC Santa Barbara)
David Kirkby (UC Irvine)

Development and Use of RooFit in BaBar

Development

- RooFit started as RooFitTools (presented at ROOT2001) in late 1999
  - Original design was rapidly stretched to its limits
- Started comprehensive redesign early 2001
  - New design was released to BaBar users in Oct 2001 as RooFit
  - Extensive testing & tuning of user interface in the past year
- RooFit released on SourceForge in Sep 2002

Current use

- Almost all BaBar analysis requiring a non-trivial fit now use RooFit or are in the process of switching to RooFit, e.g.
  - CP violation and mixing in hadronic decays (‘$\sin 2\beta$’)
  - B-Mixing in di-lepton events, $D^{*\ell\nu}$ events
  - Measurement of $\sin 2\alpha_{\text{eff}}$ from $B \to \rho \pi$, $B \to \pi \pi$
  - Searches for rare decays ($B \to \phi K_{S}, \eta' K_{S}, ...$)

- Typical fit complexity
  - 30 – 70 floating parameters
  - 4-8 dimensions
  - PDF consists of 1000-10000 objects
  - Dataset of 500-100000 events
Ntuples & Trees analysis

PAW-like queries on attributes

```
  tree.Draw("varx","sqrt(x*y) < z")
  tree.Draw("event.tracks.GetPt()")
```

+ Tree browser and viewer
  + MakeClass
  (generation of skeleton analysis code)
  + MakeSelector
  same as MakeClass for PROOF

Histogramming
Fitting

Math Libs
Statistics

Ntuple analysis

Collaboration with FNAL

LCG seminar 7 November Rene Brun

ROOT Framework

Collaboration with MIT
Parallel ROOT Facility

The PROOF system allows:
- parallel analysis of trees in a set of files
- parallel analysis of objects in a set of files
- parallel execution of scripts

on clusters of heterogeneous machines

Its design goals are:
- transparency, scalability, adaptability

Prototype developed in 1997 as proof of concept, full version nearing completion now

Collaboration between core ROOT group at CERN and MIT Heavy Ion Group
Running a PROOF Job

// Analyze TChains in parallel

gROOT->Proof();
TChain *chain = new TChain("AOD");
chain->Add("lfn://alien.cern.ch/alice/prod2002/P2001*");
...
chain->Process("myselector.C");

// Analyze generic data sets in parallel

gROOT->Proof();
TDSet *objset = new TDSet("MyEvent", ",", ",/events");
objset->Add("lfn://alien.cern.ch/alice/prod2002/file1");
...
objset->Add(set2003);
objset->Process("myselector.C++");
PROOF Demo at ROOT workshop

- **Client machine**
  - PIII @ 1GHz / 512 MB
  - Standard IDE disk

- **Cluster with 15 nodes at CERN**
  - Dual PIII @ 800 MHz / 384 MB
  - Standard IDE disk

- **Cluster with 4 nodes at MIT**
  - Dual AthlonMP @ 1.4GHz / 1GB
  - Standard IDE disk
Detector geometry

A very important element
Request number 1 at ROOT FNAL workshop 2001
Work in progress with ALICE
Huge interest in many experiments
I have a complete talk on this

Modeling
Visualization
Interactivity
Where am I?
Distance to boundary
Closest boundary
Persistency

C++ classes
MySQL

Geometry package

Simulation program
Geant3-based
Geant4-based
Fluka-based

Reconstruction program

Detector Simulation
Event Display

System services
Physics, d, 3-d graphics
Interplay, analysis

GUI Toolkits

Event Generators
Some detectors in ROOT geometry
## TGeo performance vs Geant3

<table>
<thead>
<tr>
<th></th>
<th>Number nodes</th>
<th>gtmedi physics</th>
<th>Root physics</th>
<th>Geant3/Root</th>
<th>gtmedi random</th>
<th>Root random</th>
<th>Geant3/Root</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gexam1</td>
<td>425</td>
<td>3.08</td>
<td>1.84</td>
<td>1.67</td>
<td>6.60</td>
<td>4.39</td>
<td>1.50</td>
</tr>
<tr>
<td>Gexam3</td>
<td>86</td>
<td>2.87</td>
<td>2.15</td>
<td>1.33</td>
<td>3.47</td>
<td>2.50</td>
<td>1.38</td>
</tr>
<tr>
<td>Gexam4</td>
<td>12781</td>
<td>2.51</td>
<td>2.20</td>
<td>1.14</td>
<td>12.09</td>
<td>11.18</td>
<td>1.08</td>
</tr>
<tr>
<td>Brahms</td>
<td>2649</td>
<td>5.82</td>
<td>3.04</td>
<td>1.91</td>
<td>4.17</td>
<td>1.93</td>
<td>2.16</td>
</tr>
<tr>
<td>Tesla</td>
<td>15370</td>
<td>6.56</td>
<td>5.58</td>
<td>1.17</td>
<td>12.95</td>
<td>7.15</td>
<td>1.81</td>
</tr>
<tr>
<td>CDF</td>
<td>24422</td>
<td>14.81</td>
<td>4.31</td>
<td>3.43</td>
<td>20.94</td>
<td>5.85</td>
<td>3.57</td>
</tr>
<tr>
<td>Minos_near</td>
<td>30988</td>
<td>30.93</td>
<td>20.99</td>
<td>1.47</td>
<td>21.57</td>
<td>13.70</td>
<td>1.57</td>
</tr>
<tr>
<td>BTeV_Ecal</td>
<td>52</td>
<td>1.57</td>
<td>1.08</td>
<td>1.45</td>
<td>1.78</td>
<td>0.73</td>
<td>2.43</td>
</tr>
<tr>
<td>BTeV</td>
<td>295310</td>
<td>45.27</td>
<td>25.88</td>
<td>1.75</td>
<td>197.06</td>
<td>26.83</td>
<td>7.34</td>
</tr>
<tr>
<td>CMS_Ecal</td>
<td>251713</td>
<td>5.60</td>
<td>1.81</td>
<td>3.09</td>
<td>5.69</td>
<td>1.74</td>
<td>3.27</td>
</tr>
<tr>
<td>CMS</td>
<td>1166310</td>
<td>33.57</td>
<td>8.76</td>
<td>3.83</td>
<td>39.09</td>
<td>24.98</td>
<td>1.56</td>
</tr>
<tr>
<td>LHCb</td>
<td>1533488</td>
<td>7.98</td>
<td>6.75</td>
<td>1.18</td>
<td>12.58</td>
<td>2.89</td>
<td>4.35</td>
</tr>
<tr>
<td>Alice</td>
<td>3080198</td>
<td>11.50</td>
<td>8.63</td>
<td>1.33</td>
<td>11.45</td>
<td>7.28</td>
<td>1.57</td>
</tr>
<tr>
<td>Atlas</td>
<td>29046966</td>
<td>8.90</td>
<td>9.94</td>
<td>0.89</td>
<td>32.48</td>
<td>23.39</td>
<td>1.38</td>
</tr>
</tbody>
</table>
Detector Simulation

This strategy facilitates migration or comparisons with a common input and a common output.

ROOT can provide a solid base for: geometry, visualization, interactivity, interpreter and persistency.

Kinematics
- Histogramming
- Fitting
- Math Libs
- Statistics

Geometry
- TVirtualMC
- Analysis
- Interpreters

Geant3
- Geant4
- Fluka

Event Display
- 2-d, 3-d graphics
- GUI
- Toolkit

Geant3.tar.gz includes an upgraded Geant3 with a C++ interface to geometry.

Geant4_mc.tar.gz includes the TVirtualMC <---> Geant4 interface classes.

LCG seminar 7 November  Rene Brun

ROOT Framework
Modularity

Ignominy plots
Decoupling components
The Plug-in Manager
## Package Metrics (Ignominy)

### from Lassi Tuura

<table>
<thead>
<tr>
<th>Project</th>
<th>Release</th>
<th>Packages</th>
<th>Average # of direct dependencies</th>
<th>Cycles (Packages Involved)</th>
<th># of levels</th>
<th>ACD</th>
<th>CCD</th>
<th>NCCD</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anaphe</td>
<td>3.6.1</td>
<td>31</td>
<td>2.6</td>
<td>--</td>
<td>8</td>
<td>5.4</td>
<td>167</td>
<td>1.3</td>
<td>630/170k</td>
</tr>
<tr>
<td>ATLAS</td>
<td>1.3.2</td>
<td>230</td>
<td>6.3</td>
<td>2 (92)</td>
<td>96</td>
<td>70</td>
<td>16211</td>
<td>10</td>
<td>1350k</td>
</tr>
<tr>
<td></td>
<td>1.3.7</td>
<td>236</td>
<td>7.0</td>
<td>2 (92)</td>
<td>97</td>
<td>77</td>
<td>18263</td>
<td>11</td>
<td>1350k</td>
</tr>
<tr>
<td>CMS/ORCA</td>
<td>4.6.0</td>
<td>199</td>
<td>7.4</td>
<td>7 (22)</td>
<td>35</td>
<td>24</td>
<td>4815</td>
<td>3.6</td>
<td>420k</td>
</tr>
<tr>
<td></td>
<td>6.1.0</td>
<td>385</td>
<td>10.1</td>
<td>4 (9)</td>
<td>29</td>
<td>37</td>
<td>14286</td>
<td>4.9</td>
<td>580k</td>
</tr>
<tr>
<td>CMS/COBRA</td>
<td>5.2.0</td>
<td>87</td>
<td>6.7</td>
<td>4 (10)</td>
<td>19</td>
<td>15</td>
<td>1312</td>
<td>2.7</td>
<td>180k</td>
</tr>
<tr>
<td></td>
<td>6.1.0</td>
<td>99</td>
<td>7.0</td>
<td>4 (8)</td>
<td>20</td>
<td>17</td>
<td>1646</td>
<td>2.9</td>
<td>200k</td>
</tr>
<tr>
<td>CMS/IGUANA</td>
<td>2.4.2</td>
<td>35</td>
<td>3.9</td>
<td>--</td>
<td>6</td>
<td>5.0</td>
<td>174</td>
<td>1.2</td>
<td>150/38k</td>
</tr>
<tr>
<td></td>
<td>3.1.0</td>
<td>45</td>
<td>3.3</td>
<td>1 (2)</td>
<td>8</td>
<td>6.1</td>
<td>275</td>
<td>1.3</td>
<td>150/60k</td>
</tr>
<tr>
<td>Geant4</td>
<td>4.3.2</td>
<td>108</td>
<td>7.0</td>
<td>3 (12)</td>
<td>21</td>
<td>16</td>
<td>1765</td>
<td>2.8</td>
<td>680k</td>
</tr>
<tr>
<td>ROOT</td>
<td>2.25/05</td>
<td>30</td>
<td>6.4</td>
<td>1 (19)</td>
<td>22</td>
<td>19</td>
<td>580</td>
<td>4.7</td>
<td>660k</td>
</tr>
</tbody>
</table>

*) John Lakos, Large-Scale C++ Programming

- **Size** = total amount of source code (roughly—not normalised across projects!)
- **ACD** = average component dependency (~ libraries linked in)
- **CCD** = sum of single-package component dependencies over whole release: test cost
- **NCCD** = Measure of CCD compared to a balanced binary tree
  - < 1.0: structure is flatter than a binary tree (= independent packages)
  - > 1.0: structure is more strongly coupled (vertical or cyclic)
- **Aim**: Minimise NCCD for given software/functionality (good toolkit: ~ 1.0)
Plug-in Manager

- Where are plug-ins used?
  ```
  TFile *rf = TFile::Open("rfio://castor.cern.ch/alice/aap.root")
  TFile *df = TFile::Open("dcache://main.desy.de/h1/run2001.root")
  ```

- Previously dependent on "magic strings" in source, e.g. in TFile.cxx:

- Adding case or changing strings requires code change and recompilation. Not user customizable.

- Currently 29 plug-ins are defined for 20 different (abstract) base classes

- No magic strings in code anymore

The Plug-in Manager is expected to solve most problems reported by Lassi
Relations with LCG

Where we agree
Possible problems
Wishes
The LCG Project

- Must be a success
- Because success is not guaranteed
  - we better start on solid grounds
  - time is very very very critical
- The LCG has an opportunity to capitalize on the success of ROOT.
- But, there is a potential danger to see
  - parallel developments and conflicts
  - software “balkanisation” and “saucissonage”
  - wrong balance in manpower between projects and experiments
Proposal to Blueprint RTAG in June

The existing set of ROOT libs is the starting core of the LCG software. Because the system is already widely distributed and used, it guarantees the initial acceptance of a wide community.

We invite architects and key developers to review the current organisation of libraries and to propose an evolution if it proves necessary.

This proposal was rejected at the level of the architects (3:1) in fact, it was never discussed on pure technical grounds.
The ROOT data analysis framework is widely used in HENP and beyond, and is being heavily used by the LHC experiments and the LCG. We see the LCG software as a user of ROOT; a user with a very close relationship with the ROOT team. While the ROOT team is highly attuned and responsive to the needs of the LHC experiments, it also supports a large and diverse non-LHC community (including many major HENP experiments) with its own requirements, not least the stability of ROOT itself.

It is impractical for LCG software architecture and development to be tightly coupled to ROOT and vice versa. We expect the user-provider relationship to work much better. The ROOT team has an excellent record of responsiveness to users. So while ROOT will be used at the core of much LCG software for the foreseeable future, there will always be a 'line' with ROOT proper on one side and LCG software on the other.

ROOT itself will grow and change over time. Decisions on making use of ROOT in the implementation of LCG software components should be made on a case by case basis, driven by the circumstances. Despite the user-provider relationship, LCG software may nonetheless place architectural, organizational or other demands on ROOT. For example, the library organization and factorization of ROOT will impact component interdependencies in LCG software employing ROOT implementations and may drive changes in the organization and/or factorization.
Blueprint RTAG impressions

- We had useful meetings, sometime hot meetings.
- An opportunity to discuss many topics (first time since many years!). Dialog is better than wars.
- There is nothing wrong if there are some divergences. Uniformity is a bad sign.
- We had input from architects and some experts. I hope that we will get feedback now from end users.
- Torre, as application area coordinator, has done a great job in just a few months.
- There are a few but important points where I positively disagree with him (next slides).
Software Structure

Applications

Simulation Framework
Reconstruction Framework
Visualization Framework
Other Frameworks

Basic Framework

Foundation Libraries
Optional Libraries

Blueprint RTAG 2002

Picture from 1995
Domain Decomposition

EvtGen
Event Generation

Engine
Detector Simulation

Algorithms
Reconstruction

Modeler
Geometry

Fitter
Analysis

NTuple
Event Model

Scripting
Interactive Services

GUI
Calibration

FileCatalog
Persistency

StoreMgr

Dictionary
Core Services

PluginMgr

Scheduler
Grid Services

Monitor

Foundation and Utility Libraries

ROOT
GEANT4
FLUKA
MySQL
DataGrid
Python
Qt
...

Products mentioned are examples; not a comprehensive list
Domain Decomposition
What ROOT covers in red

Event Generation
EvtGen

Detector Simulation
Engine

Reconstruction
Algorithms

Analysis
Fitter

Interactive Services
Scripting

Event Model
Modeler

Calibration
NTuple

Geometry

Core Services
Dictionary

Scheduler

Grid Services
PluginMgr

Monitor

Foundation and Utility Libraries

ROOT

GEANT4

FLUKA

MySQL

DataGrid

Python

Qt

...
Domain Decomposition
LCG proposed projects
Domain Decomposition
LCG proposed projects

- EvtGen
  - Event Generation
- Engine
  - Detector Simulation
- Algorithms
  - Reconstruction
- Modeler
  - Geometry
- FileCatalog
  - Persistency
- Dictionary
- PluginMgr
- Whiteboard
- CTS
  - Core Tools & Services
- P & I
  - Physics Interface
  - Interactive Services

- Fitting
- NTuplizer
  - Analysis
- Event Modeler
  - Calibration

ROOT
GEANT4
FLUKA
MySQL
DataGrid
Python
Qt
LCG proposed projects

- Scenario 1
  Two extreme views on the LCG proposal

- Scenario 2

- Scenario 3
  A simple alternative model

We more or less agree on the domain decomposition.

We agree that RTAGS are necessary and useful.

The mapping between domains and responsibilities is wrong.
LCG - ROOT: Scenario 1

My understanding of the LCG proposal

Layer on top of ROOT
some complementary additions

Why 2 interfaces?
Who is doing what?

Complex management, many heads
Plenty of ambiguities on goals
Committee-driven layer
LCG - ROOT : Scenario 1 comments

- Assuming that S1 is the way to go, we will provide the best possible service (as provider).
- We anticipate a committee-driven system that will not be in the interest of the users.
- It will be difficult to maintain a coherent view between ROOT and the other projects working in the same domains.
- Cannot be a motivating environment because you cannot be creative in this context.
- Motivation will come from the large ROOT users base (LHC being a fraction only).
LCG - ROOT: Scenario 2
The conditions for a failure

Layer on top of ROOT in 2002
becoming an alternative in 2005

Why 2 systems?
Who is doing what?
Why it is so complex?
Why it does not work?
HELP!!!!!
LCG - ROOT : Scenario 2
Comments

- The risk is that Scenario 1 may drift gradually to Scenario 2.
- User-Provider relation at the beginning
- Parts of ROOT copied to the new project
- Two parallel projects
- ROOT: seen as a pain in the neck
- We better stop ROOT now in the LCG context if the "secret idea" is to follow this scenario.
### Personnel resources

<table>
<thead>
<tr>
<th>Approximate expected resources:</th>
</tr>
</thead>
<tbody>
<tr>
<td>20  LCG (17 now)</td>
</tr>
<tr>
<td>3   CERN IT DB</td>
</tr>
<tr>
<td>5   CERN IT API G4</td>
</tr>
<tr>
<td>5   CERN IT API other</td>
</tr>
<tr>
<td>5   CERN EP non-expt</td>
</tr>
<tr>
<td>10  ATLAS</td>
</tr>
<tr>
<td>0   ALICE (currently forseen contributions of 4 FTEs come through ROOT)</td>
</tr>
<tr>
<td>10  CMS</td>
</tr>
<tr>
<td>3   LHCb</td>
</tr>
<tr>
<td><strong>61</strong> Total</td>
</tr>
</tbody>
</table>

These are contributions to LCG software. ROOT contributions are not shown here; the LCG software will be a user of ROOT.

These numbers are all very approximate.
LCG - ROOT : Scenario 3
The conditions for success

GRID Software

TOOR

Simulation Software

less projects
less people
less overhead

Applications area manager

More people in experiments
more feedback to projects
more results on time

LCG seminar 7 November  Rene Brun

ROOT Framework
LCG - ROOT : Scenario 3 comments

- More realistic mapping of existing responsibilities
- Clear path for ROOT and users
- Minimize possible divergences
- Facilitate the task of the applications coordinator, but gives him more responsibility.
- Easier to monitor progress
- Increase useful manpower in experiments
Summary 1

- We have developed a simple and powerful framework now in use by thousands of people in most HEP labs.
- We have many outside contributors.
- As already expressed to the Blueprint architects, we are willing to discuss the evolution of ROOT in close cooperation with the LHC experiments.
- Scenario 1 & 2 will end-up into fights and fat systems. It is our duty to ring the bell!
- We invite end-users in LHC experiments to reassess the situation in view of the realistic scenario 3.
Summary 2

- Software is not a technical problem. It is mainly a sociological problem.
- It is easy to write many lines of code. It is more difficult to give a momentum to a system.
- Our goal should not be to write software per se, but to make sure we deliver a simple, robust, coherent framework in time for the LHC.
- We must create the conditions to remove all possible ambiguities on the path to follow.
- **Users must give feedback now, not in 2005.**