ROOT
Status and Current Developments

LCGapp meeting

April 30 2003

René Brun, CERN

(for the ROOT team)

http://root.cern.ch

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)
- Apr 03: Root version 3.05/04

8 years !!

CVS activity

	2000	2001	2002	2003
CVS Changes per month	226	156	159	161
CVS log lines per month	1881	1064	1258	1196

ROOT I/O

Recent Developments in CVS

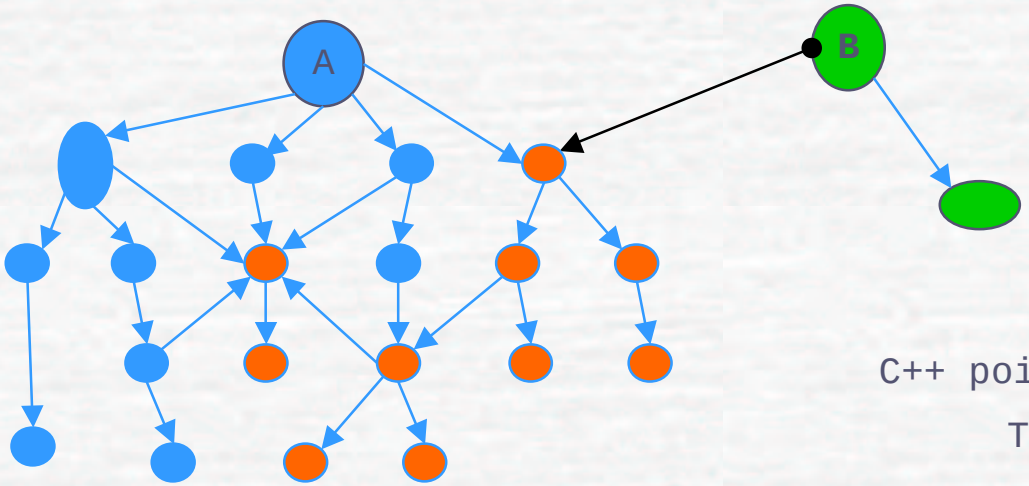
New developments

ROOT I/O Developments

- *Foreign* classes
  - Classes that have not been instrumented for ROOT in any way
  - They can now be stored in ROOT files
  - Just generate a dictionary
  - Very useful for 3rd party libraries
- *Emulated* classes
  - ROOT files are "self-describing"
  - This information can be used to emulate classes for which the actual code is not available while reading

ROOT I/O Developments (2)

- TRef, TRefArray
  - Designed as light weight entities
  - Very fast dereferencing (direct access tables)
  - Polymorphic




```
TBuffer b1;
A.Streamer(b1)
TBuffer b2;
B.Streamer(b2)
```

C++ pointer ⟹ Duplicated

TRef ⟹ Saved only once

CINT / rootcint

- rootcint
  - Improvement in handling of templates in general and STL containers in particular
  - Introduce I/O for foreign classes
  - Enhancement for multiple inheritance
- CINT improvements
  - Very long list of enhancements: http://root.cern.ch/root/Cint.phtml?relnote

TTree improvements

- Creating branches from collection of objects.
- Split algorithm improved
  - For complex cases of inheritance or composition.
- TTreeFormula enhancements
  - Drawing TBits
  - Custom Histogram Size
  - Special variables (Entry #, Iteration #, etc.)
- Automatic file overflow
  - When file reaches user specified maximum
  - Create a new file: with "myfile.root", subsequent files are named "myfile_1.root",

ROOT I/O new developments

- Lifting remaining limitation of foreign class I/O implementation.
- Enhance support for STL container (Victor & Philippe)
  - New I/O mechanism
  - Split vector/list
  - Enable emulation of STL container
- Support for large file (more than 2Gb)
- Support for XML interchange format.
- Automatic class checksum replacing the ClassDef class version number

TTree new developments

- STL list/vector (Victor & Philippe)
  - Split mode
  - Emulated class mode
- TTree::Drawing a user function
  - Will allow the user to provide a function which will be executed for each entry of the TTree in a context such that the branch name can be used to read the values in the branch
  - Branches will be read only if accessed
  - The function can use arbitrary C++
  - The return value of the function will be histogrammed

PROOF

PROOF

- System for the interactive analysis of very large sets of ROOT data files on a cluster of computers
- The main design goals are:
  - ***transparency, scalability, adaptability***
- Allows:
  - parallel analysis of trees or object in a set of files
  - parallel execution of scripts

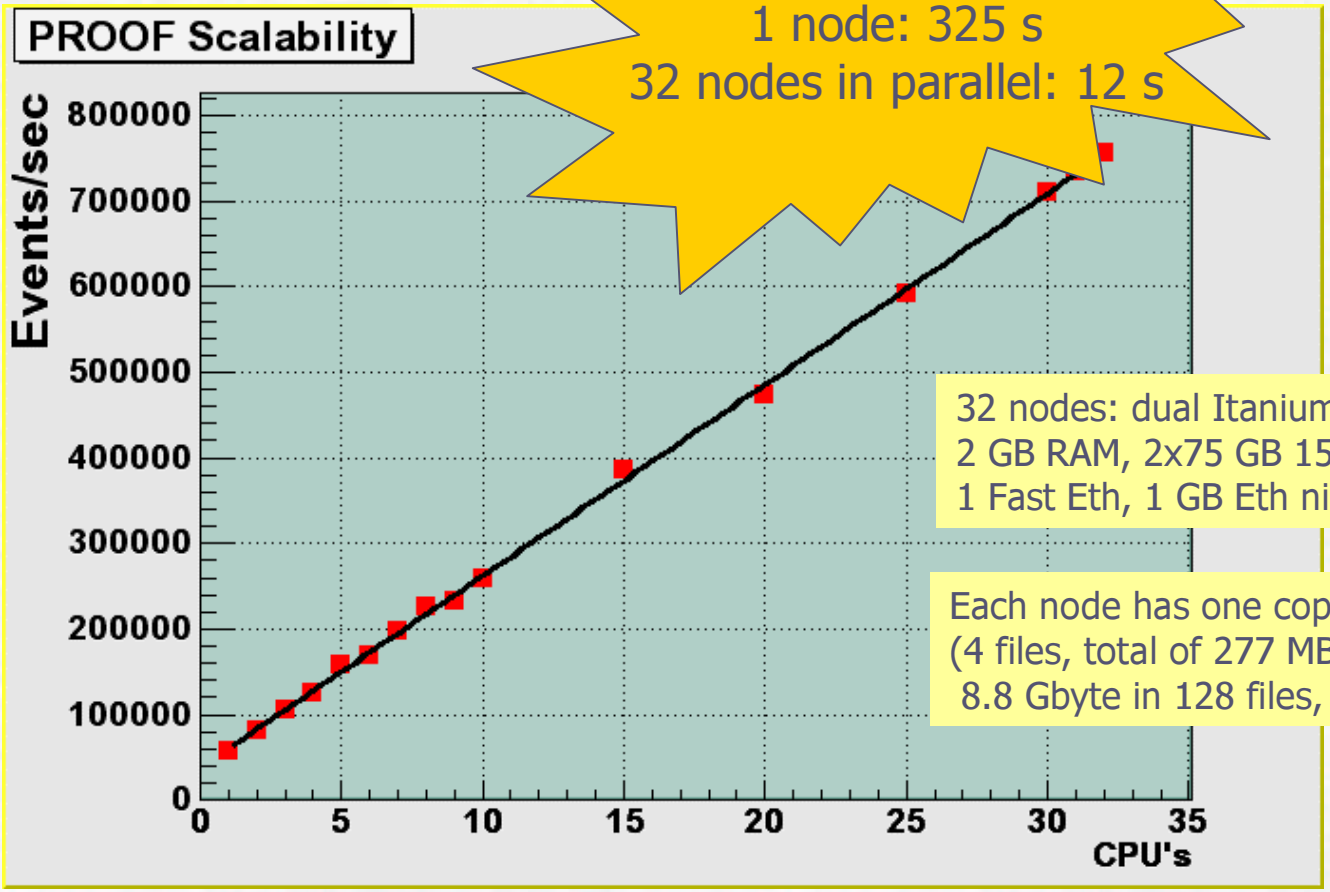
  on a cluster of heterogeneous machines

PROOF - Architecture

- Data Access Strategies
  - Local data first, also **rootd, rfio**, SAN/NAS
- Transparency
  - Input objects copied from client
  - Output objects merged, returned to client
- Scalability and Adaptability
  - Vary packet size (specific workload, slave performance, dynamic load)
  - Heterogeneous Servers
- Migrate to multi site configurations

PROOF S

8.8GB, 128 files
1 node: 325 s
32 nodes in parallel: 12 s



32 nodes: dual Itanium II 1 GHz CPU's,
2 GB RAM, 2x75 GB 15K SCSI disk,
1 Fast Eth, 1 GB Eth nic (not used)

Each node has one copy of the data set
(4 files, total of 277 MB), 32 nodes:
8.8 Gbyte in 128 files, 9 million events

PROOF and Data Grids

- Many services are a good fit
  - Authentication
  - File Catalog, replication services
  - Resource brokers
  - Monitoring

  → Use abstract interfaces
- Phased integration
  - Static configuration
  - Use of one or multiple Grid services
  - Driven by Grid infrastructure

GRID and ROOT

- TGrid: Abstract interface to GRID services
- TAlien: an implementation of TGrid
- Proof can use
  - Grid Resource Broker
  - Grid File Catalogue
  - Grid Monitoring Services
  - Condor
- Remote File protocol implemented:
  - Chirp (Condor)
  - DCache (Desy, CDF)
  - RFIO (Castor, CERN)

Different PROOF–GRID Scenarios

- Static stand-alone
  - Current version, static config file, pre-installed
- Dynamic, PROOF in control
  - Using grid file catalog and resource broker, pre-installed
- Dynamic, ALiEn in control
  - Idem, but installed and started on the fly by AliEn
- Dynamic, Condor in control
  - Idem, but allowing in addition slave migration in a Condor pool

Authentication Issues

- Both **`rootd`** and **`proofd`** daemons now support wide array of authentication services (Gerri Ganis):
  - Clear password, uid/gid matching, SRP, Kerberos, Globus, ssh
- Ticket forwarding in a PROOF environment not entirely trivial

rootd/proofd: improved authentication features
By Gerri Ganis

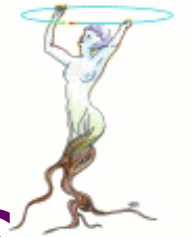
New features:

- Added *New methods*: GLOBUS gssapi, SSH, uid/gid
  (in addition to: Clear, SRP, Kerberos 5)
- Added *Negotiation*:
  - Server sends list of accepted methods (host specific and/or default)
  - Client automatically attempts in turn those locally available
- Added *memory of successful authentications*:
  - faster re-authentication during the same root/proof session

Master-to-slave authentication in PROOF by Gerri Ganis

Possibility to give a *list of mechanisms* to be attempted by the master in negotiation with the slave

- *Credential forwarding* for GLOBUS (*delegated* credentials) and Kerberos 5 (*forwardable tickets*)
- uid/gid can be used very efficiently on *local clusters* or when security is not an issue
- other methods need special files on the master:
  - > Clear/SRP: $HOME/.netrc, $HOME/.rootnetrc
  - > SSH: $HOME/.ssh/identity, id_rsa or id_dsa



Remote File Access Optimizations

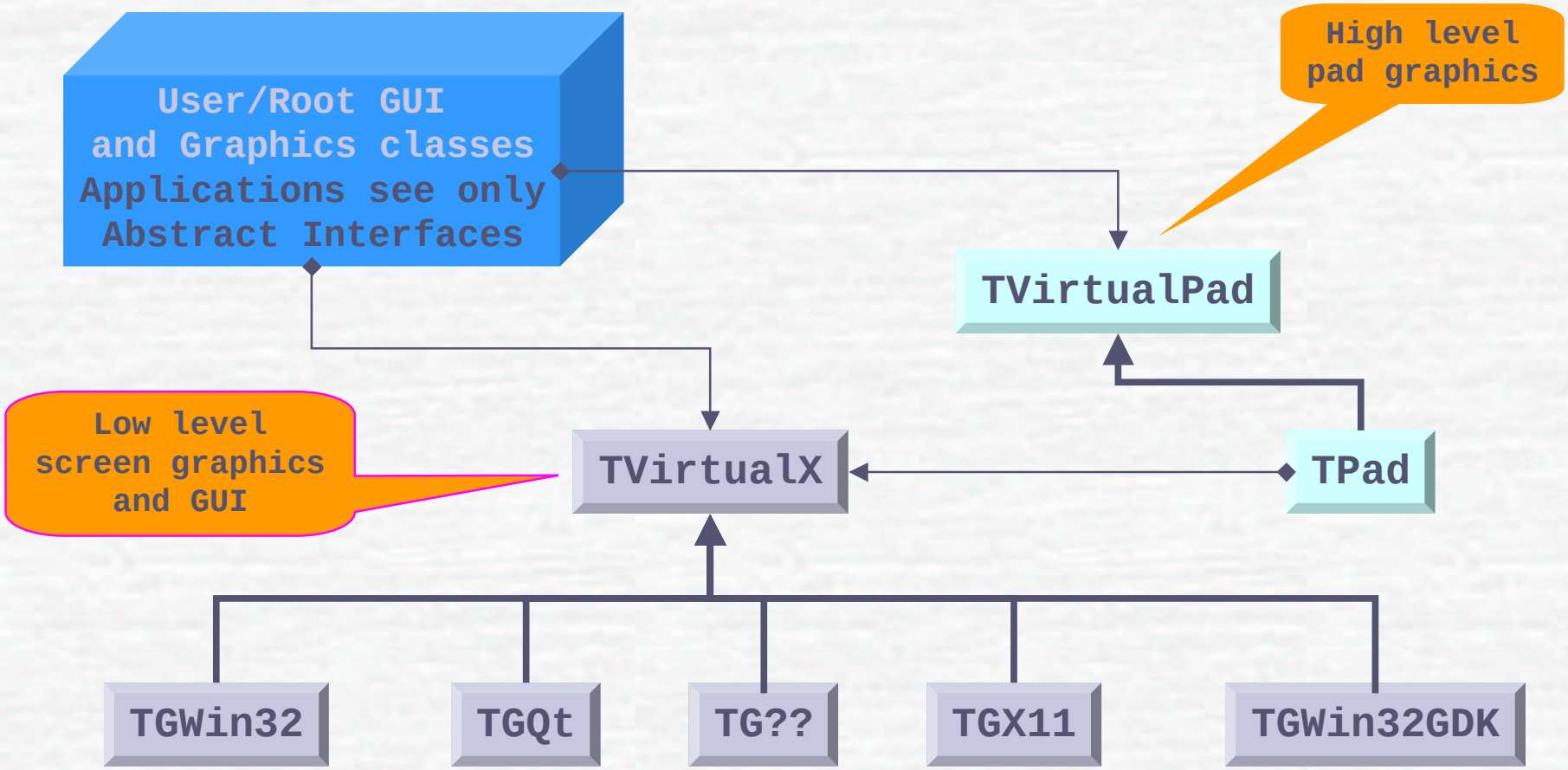
- Improvements in rootd
  - Make multi-threaded
  - Add read thread and cache for read-ahead of Tree buffers
  - Add (optionally) asynchronous write to reduce write latencies
- New TCastorFile that uses Castor HSM API and rootd for file access (bypassing rfio)

PROOF: Near Future

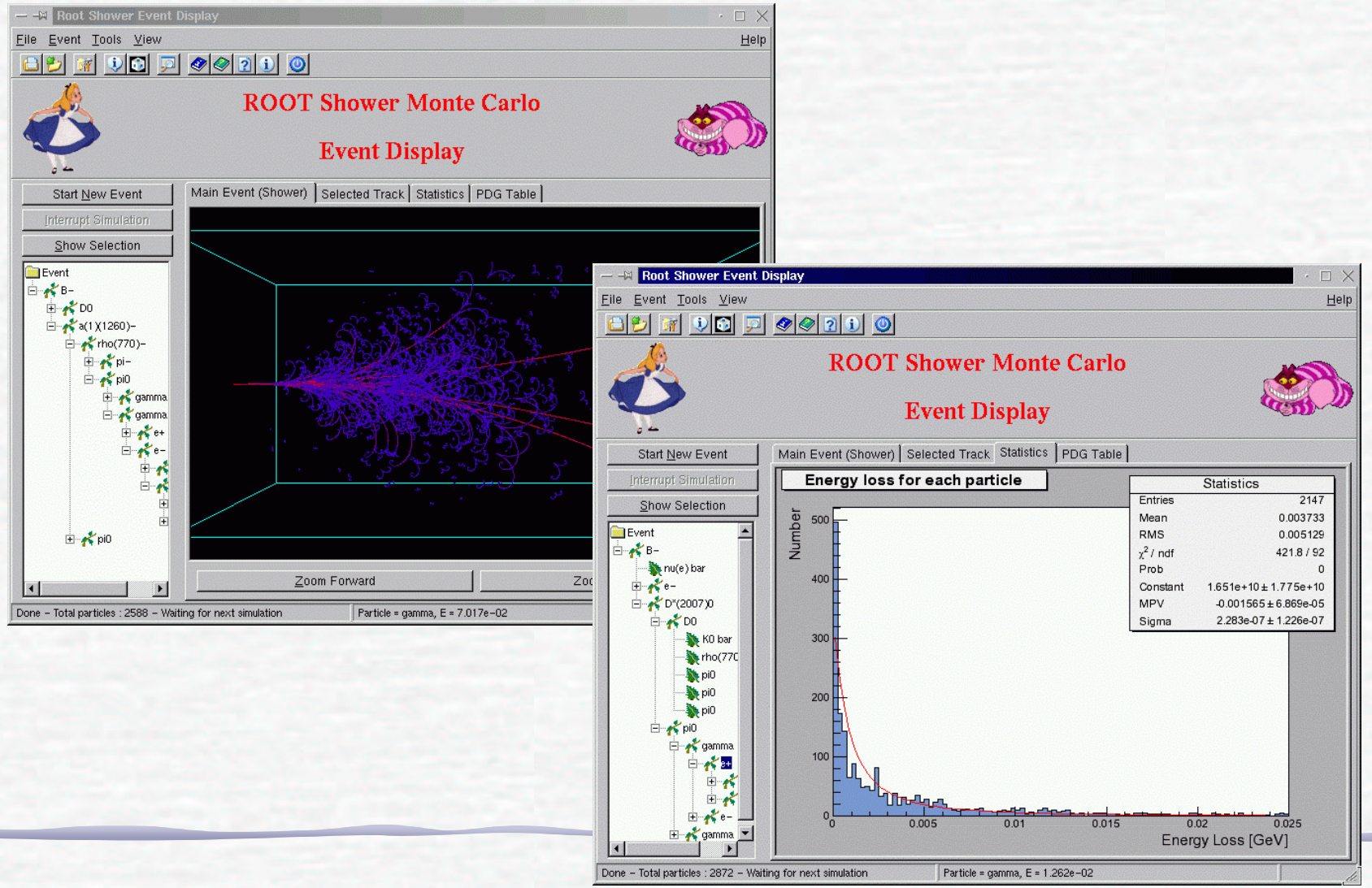
- Working with some early users
- Ironing out of several remaining issues
- Writing install and user guide
- Still to do for analysis:
  - Support for: event lists, friend trees
- Still to do for grid:
  - Interfacing to file catalogs, resource broker
- Multi site PROOF sessions

GUI and Graphics

Gui/Graphics strategy

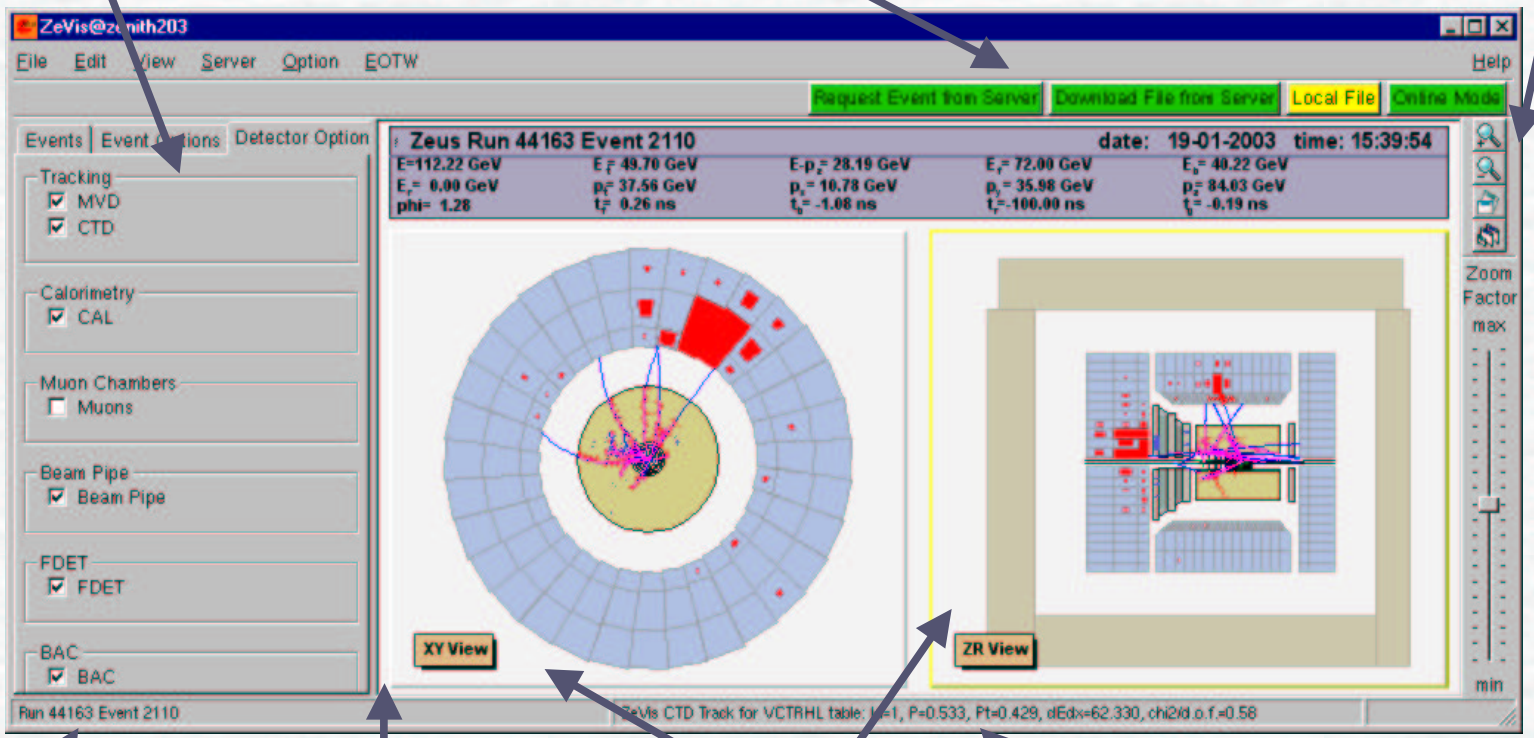


GUI Examples

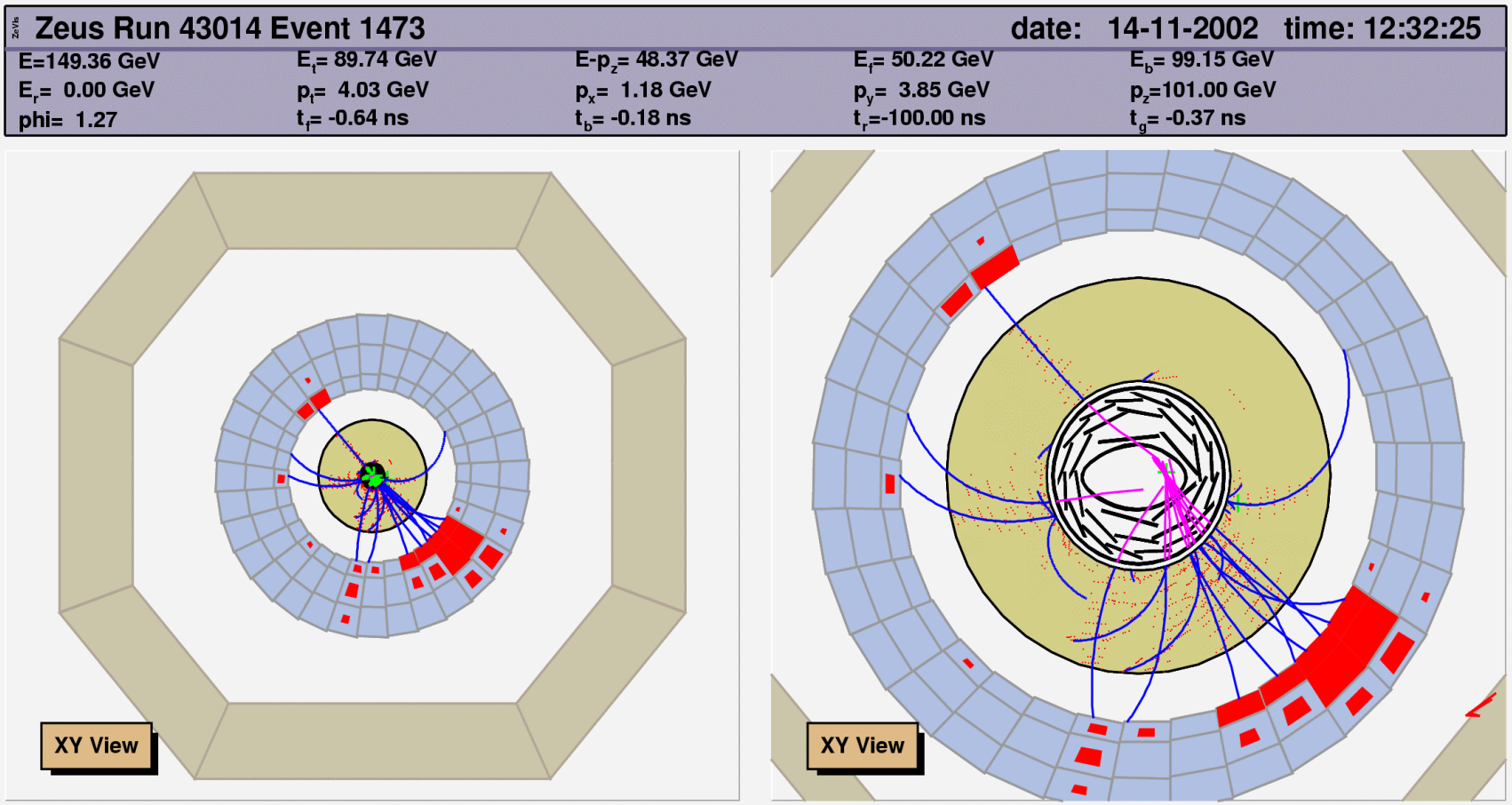


Graphical User Interface

Ex: ZEUS event display



Fish-eye View



GUI on Windows

- Old TGWin32 by Valeri Fine
  - OK for graphics in canvas
  - Not OK for the GUI
  - A maintenance burden
- TGWin32gdk by Bertrand Bellenot
  - TVirtualX compliant (see next slide)
- TGQt by Valeri Fine
  - Should be TVirtualX compliant
  - Hope to have it soon in CVS

TGWin32gdk status on Windows

Current Status

Pro:

- Gui fully working
- Same look and feel as other platforms
- Fully compatible with other platforms
- OpenGL is working

Cons:

- slower
- No Sockets (pending)
- No memory mapped files

To-do list

Short term:

- Improve speed
- Implement Sockets

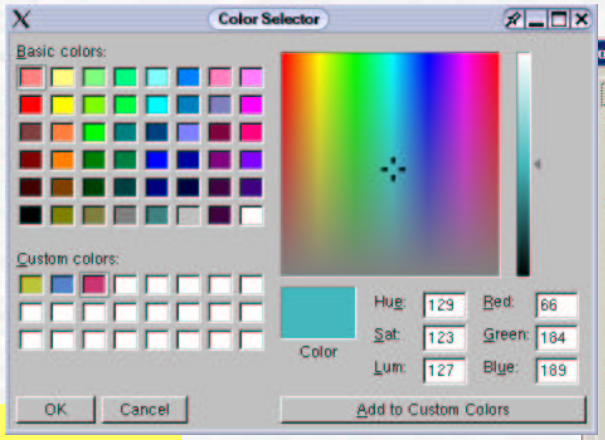
Middle term:

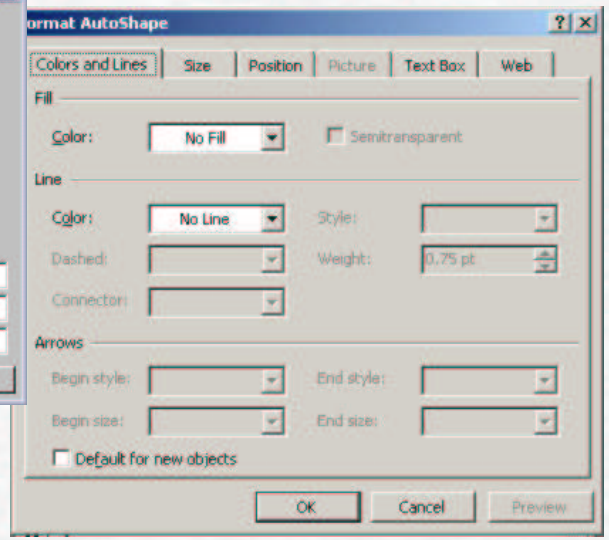
- Implement X3D
- Implement TMapFile

Long term:

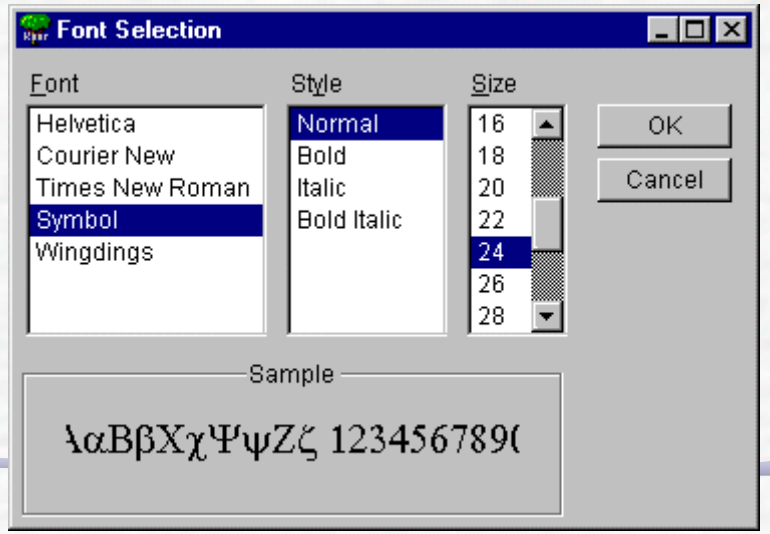
- "win32 native" version
- rootd as a windows service

New graphical interface



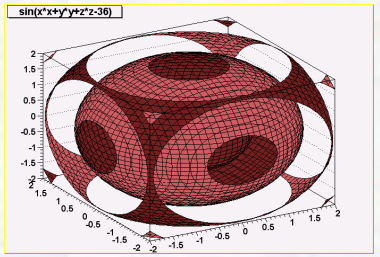


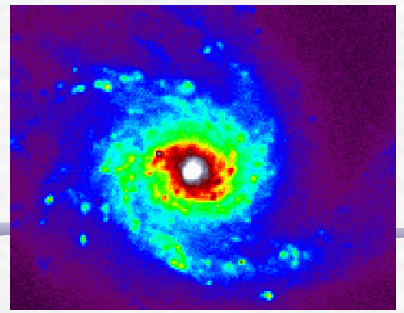
- Interface Redesign
  - core panels and tools to be more in line with modern interfaces
- GUI Builder
  - First step (done): implement the ability to generate the code for an existing GUI.
  - Second step: implement a GUI builder



Graphics Improvements

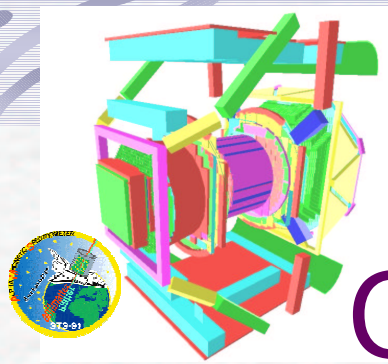
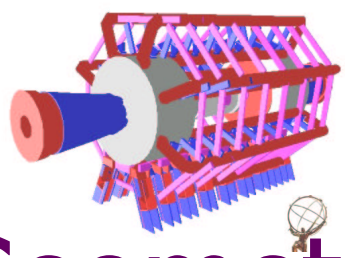
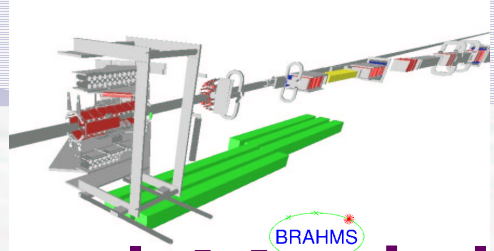
- **New class TGraphSmooth (Christian Stratowa)**
  - This class is a helper class to smooth TGraph, TGraphErrors or interpolate a graph at a set of given points. See new tutorial motorcycle.C
- **New class TSVG**
  - TSVG may be used like TPostScript to produce a Scalable Vector Graphics file instead of a postscript file.
  - Viewers like Internet Explorer can view directly the SVG files.
- **Drawing TF3**
- **TASImage**
  - Based on AfterImage
- **freetype 2 font support**
  - Distributed by default

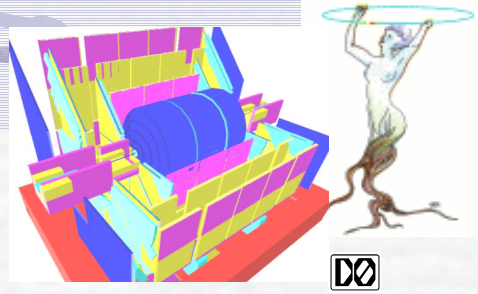




New 3-D Graphics Interface
by Olivier Couet

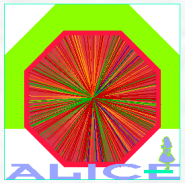
- Abstract class TVirtualView3D
- Implementations
  - TViewPad3D
  - TViewX3D
  - TViewOpenGL
  - TViewOpenInventor

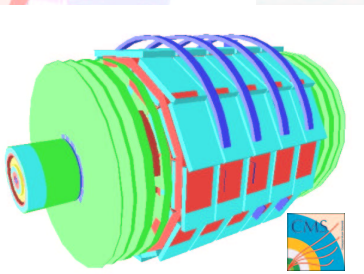


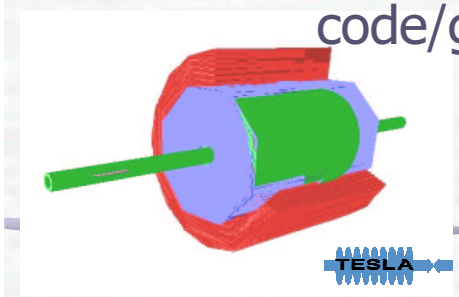
Geometrical Modeler

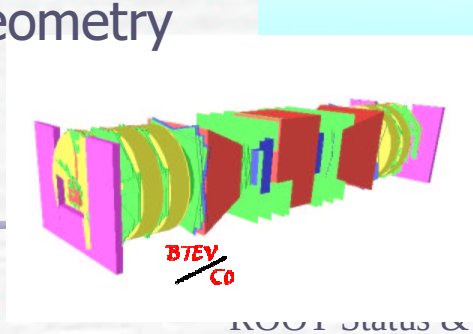
Andrei Gheata

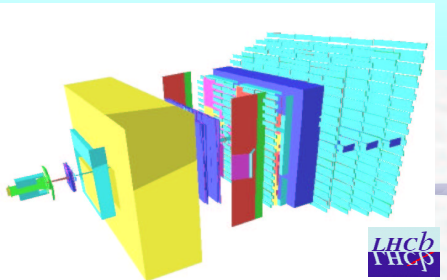


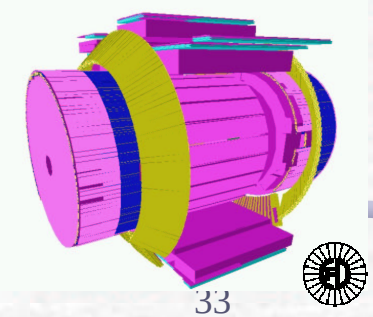
- Able to represent & navigate geometries of most HEP experiments
- Faster than Geant3 : 20% (ATLAS) up to 800% (CDF)
- Includes an interactive geometry checker able to detect/visualize most geometry definition errors (none of the tested geometries is error-free)
- To be soon integrated in a general VMC scheme able to run several simulation MC's with same user code/geometry











Geometrical Modeler Features

- Geometry browsing + interactive visualization based on context menus
- Easy-to-use API/user interface
- Extensible set of 16 shapes, Boolean composite shapes & parameterized
- Able to automatically map any G3 geometry
- Provides navigation functionality ("where am I?", distances computation, "safety radius", stepping, …)
- I/O:

	N objects	RZ size (G3)	.root size (fully voxelized)	Time to load P IV/2.4Ghz
ATLAS	29.046.966	9632 kBytes	4238 kBytes	~ 1 second

Geometrical Modeler Features

- Several interactive checking methods
- Geometry overlap checker
  - providing a sorted list of overlaps/extrusions & taking as input a minimum value for the overlapping distance
- Extensively benchmarked against G3
  - able to optimize even non-optimizable G3 geometries
  - big gain in speed

The Virtual MonteCarlo

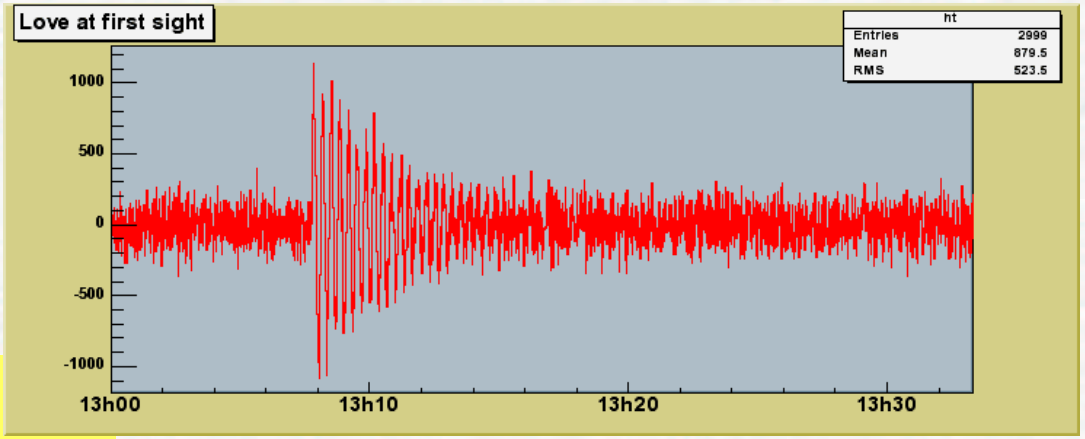
- The abstract interface TVirtualMC is being redesigned to accommodate the geometry package.
  - Geant3 operational with new geometry
  - Fluka: work in progress
  - Geant4: discussion in June?

Histograms

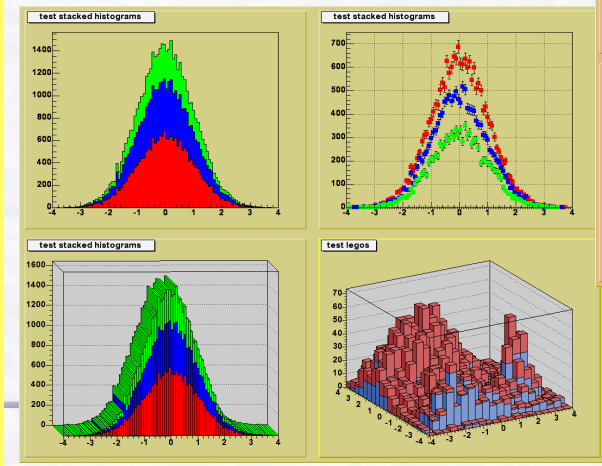
Fitting

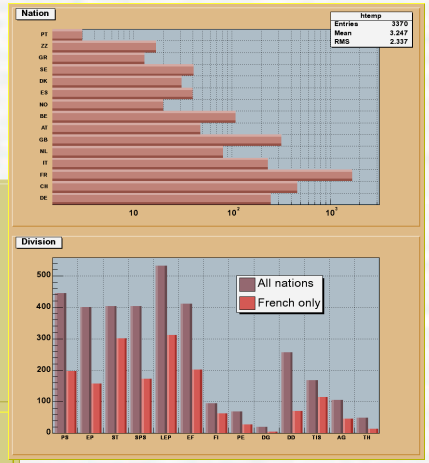
Statistics

Histograms



- New class THStack
- Long list of new functions in TH1
- Plenty of new drawing options
- Filling with string variables
- TH1::Merge(TCollection *list)



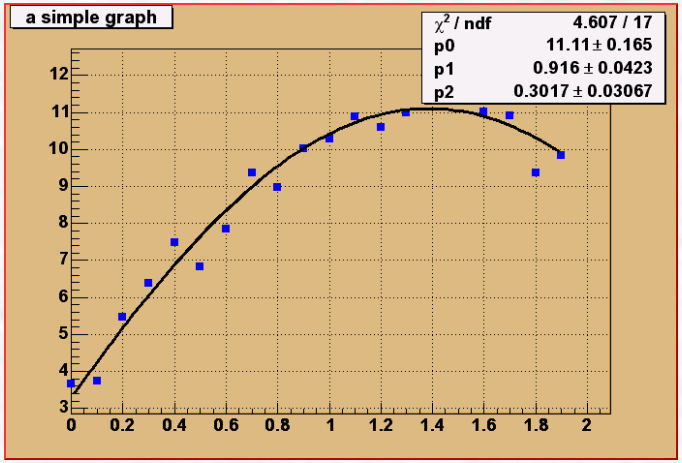


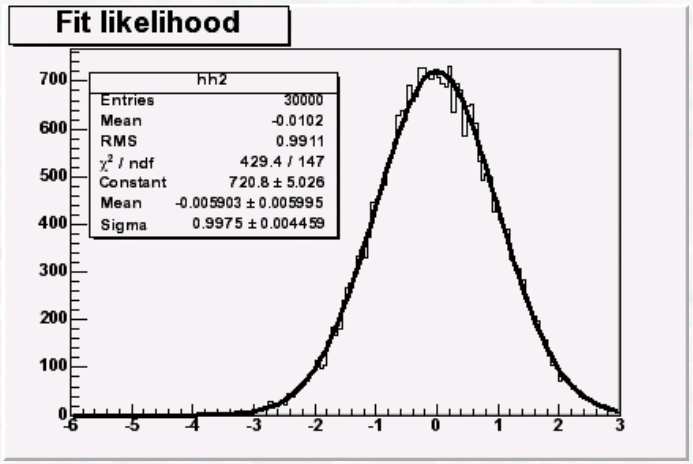
New Fitter based on Fumili

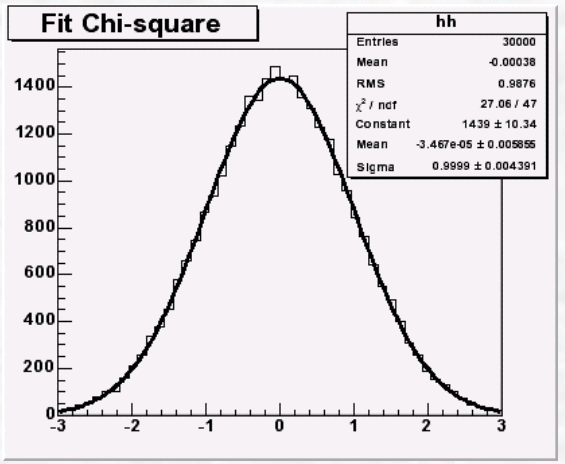
by Stanislav Nesterov

TFumili

```
                                                  10 CALLS OF FCN
                                                      STEP          FIRST
 NO.    NAME          VALUE          ERROR           SIZE          DERIVATIVE
  1 Constant      1.43868e+03   1.03381e+01   7.03174e-02  -3.01604e-03
  2 Mean         -3.46702e-05   5.85516e-03   1.27473e-05  -3.70873e-01
  3 Sigma         9.99909e-01   4.39103e-03   9.85699e-05  -9.33517e+00
FCN=429.416 FROM FUMILI  STATUS=CONVERGED        23 CALLS OF FCN
EDM=-0.000153733
 EXT PARAMETER                                    STEP          FIRST
 NO.    NAME          VALUE          ERROR           SIZE          DERIVATIVE
  1 Constant      7.20812e+02   5.02644e+00   2.45219e-02   1.45077e-04
  2 Mean         -5.90300e-03   5.99549e-03  -6.84163e-05   1.62894e+00
  3 Sigma         9.97522e-01   4.45945e-03  -3.69350e-05   1.24123e+00
FCN=4.60736 FROM FUMILI  STATUS=CONVERGED        41 CALLS OF FCN
EDM=-1.86562e-12
 EXT PARAMETER                                    STEP          FIRST
 NO.    NAME          VALUE          ERROR           SIZE          DERIVATIVE
  1 p0            1.11054e+01   3.07514e-01   2.98072e-07  -4.84978e-06
  2 p1            9.15990e-01   7.88300e-02  -1.79370e-08  -3.45869e-06
  3 p2            3.01665e-01   5.71617e-02   2.15786e-08  -2.23404e-05
```



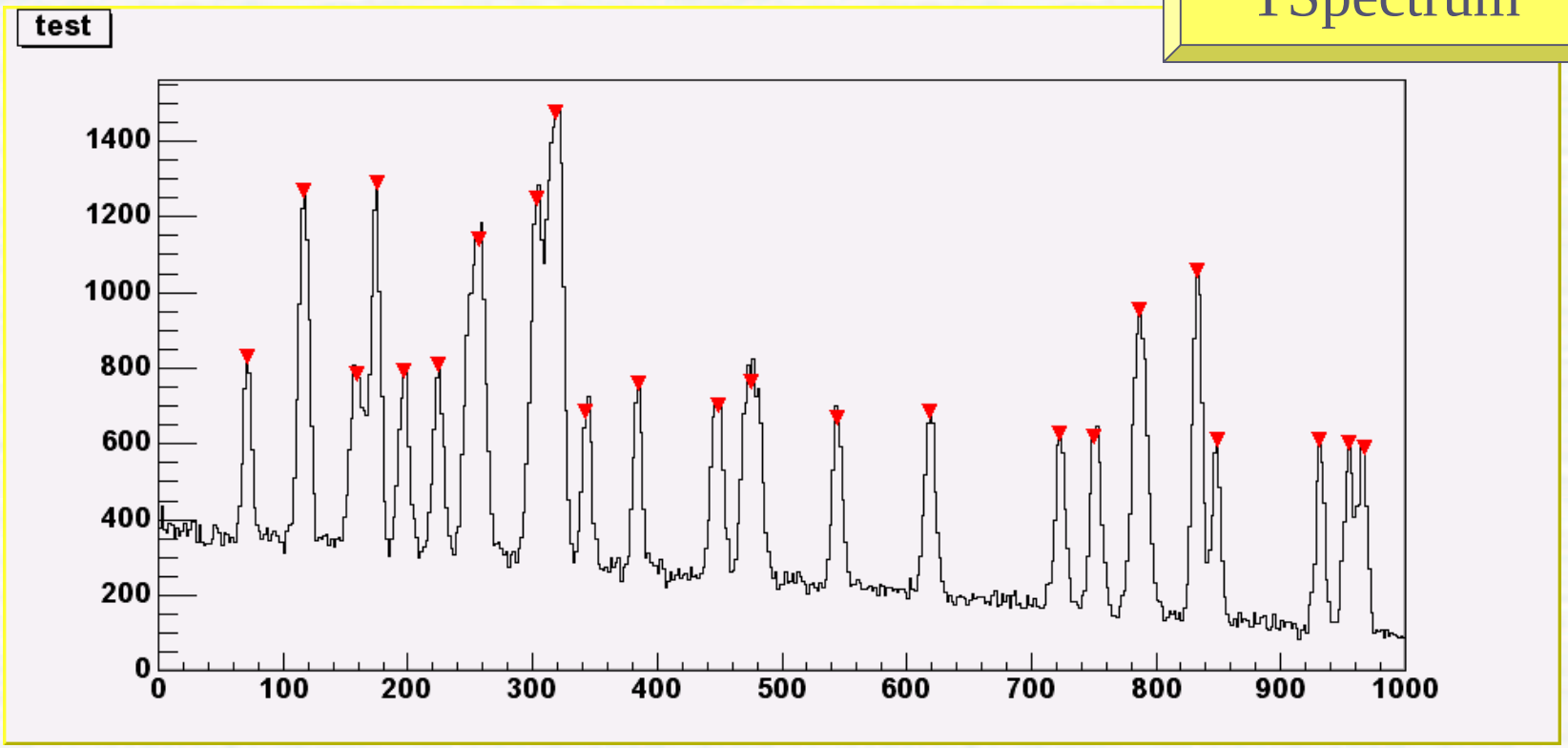




Peak Finder + Deconvolutions

by Miroslav Morach

TSpectrum



New Features and Classes

- TMatrix
  - Many enhancements by Eddy Offermann (Renaissance Technologies)
- TMath
  - New fundamental mathematical and physical constants (Tony Colley).
  - New functions: Voigt(), BreitWigner(), Bessel and Struve
  - Add two new sorting functions (Adrian Bevan)
  - Add functions TMath::IsInside

TMath (1)

TMath.h

File Edit Search Preferences Shell Macro Windows Help

/home/brun/root/base/inc/TMath.h 19391 bytes

```
// Fundamental constants
static Double_t Pi()       { return 3.14159265358979323846; }
static Double_t TwoPi()    { return 2.0 * Pi(); }
static Double_t PiOver2()  { return Pi() / 2.0; }
static Double_t PiOver4()  { return Pi() / 4.0; }
static Double_t InvPi()    { return 1.0 / Pi(); }
static Double_t RadToDeg() { return 180.0 / Pi(); }
static Double_t DegToRad() { return Pi() / 180.0; }

// e (base of natural log)
static Double_t E()        { return 2.71828182845904523536; }

// natural log of 10 (to convert log to ln)
static Double_t Ln10()     { return 2.30258509299404568402; }

// base-10 log of e  (to convert ln to log)
static Double_t LogE()     { return 0.43429448190325182765; }

// velocity of light
static Double_t C()        { return 2.99792458e8; }        // m s^-1
static Double_t Ccgs()     { return 100.0 * C(); }         // cm s^-1
static Double_t CUncertainty() { return 0.0; }             // exact

// gravitational constant
static Double_t G()        { return 6.673e-11; }           // m^3 kg^-1 s^-2
static Double_t Gcgs()     { return G() / 1000.0; }        // cm^3 g^-1 s^-2
static Double_t GUncertainty() { return 0.010e-11; }

// G over h-bar C
static Double_t GhbarC()   { return 6.707e-39; }           // (GeV/c^2)^-2
static Double_t GhbarCUncertainty() { return 0.010e-39; }

// standard acceleration of gravity
static Double_t Gn()       { return 9.80665; }             // m s^-2
static Double_t GnUncertainty() { return 0.0; }            // exact

// Planck's constant
static Double_t H()        { return 6.62606876e-34; }      // J s
static Double_t Hcgs()     { return 1.0e7 * H(); }         // erg s
static Double_t HUncertainty() { return 0.00000052e-34; }

// h-bar (h over 2 pi)
static Double_t Hbar()     { return 1.054571596e-34; }     // J s
static Double_t Hbarcgs()  { return 1.0e7 * Hbar(); }      // erg s
static Double_t HbarUncertainty() { return 0.000000082e-34; }

// hc (h * c)
static Double_t HC()       { return H() * C(); }           // J m
static Double_t HCcgs()    { return Hcgs() * Ccgs(); }     // erg cm

// Boltzmann's constant
static Double_t K()        { return 1.3806503e-23; }       // J K^-1
static Double_t Kcgs()     { return 1.0e7 * K(); }         // erg K^-1
static Double_t KUncertainty() { return 0.0000024e-23; }

// Stefan-Boltzmann constant
```

TMath.h

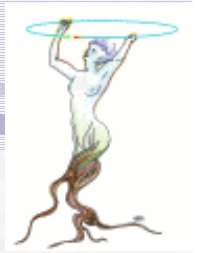
File Edit Search Preferences Shell Macro Windows Help

/home/brun/root/base/inc/TMath.h 19391 bytes

```
// Trigo
static Double_t Sin(Double_t);
static Double_t Cos(Double_t);
static Double_t Tan(Double_t);
static Double_t SinH(Double_t);
static Double_t CosH(Double_t);
static Double_t TanH(Double_t);
static Double_t ASin(Double_t);
static Double_t ACos(Double_t);
static Double_t ATan(Double_t);
static Double_t ATan2(Double_t, Double_t);
static Double_t ASinH(Double_t);
static Double_t ACosH(Double_t);
static Double_t ATanH(Double_t);
static Double_t Hypot(Double_t x, Double_t y);

// Misc
static Double_t Sqrt(Double_t x);
static Double_t Ceil(Double_t x);
static Double_t Floor(Double_t x);
static Double_t Exp(Double_t);
static Double_t Factorial(Int_t);
static Double_t Power(Double_t x, Double_t y);
static Double_t Log(Double_t x);
static Double_t Log2(Double_t x);
static Double_t Log10(Double_t x);
static Int_t    Nint(Float_t x);
static Int_t    Nint(Double_t x);
static Int_t    Finite(Double_t x);
static Int_t    IsNaN(Double_t x);
```

TMath (2)



TMath.h

File Edit Search Preferences Shell Macro Windows Help

/home/brun/root/base/inc/TMath.h 19391 bytes

```
// Binary search
static Int_t BinarySearch(Int_t n, const Short_t *array, Short_t value);
static Int_t BinarySearch(Int_t n, const Short_t **array, Short_t value);
static Int_t BinarySearch(Int_t n, const Int_t *array, Int_t value);
static Int_t BinarySearch(Int_t n, const Int_t **array, Int_t value);
static Int_t BinarySearch(Int_t n, const Float_t *array, Float_t value);
static Int_t BinarySearch(Int_t n, const Float_t **array, Float_t value);
static Int_t BinarySearch(Int_t n, const Double_t *array,
static Int_t BinarySearch(Int_t n, const Double_t **array,
static Int_t BinarySearch(Int_t n, const Long_t *array, Lo
static Int_t BinarySearch(Int_t n, const Long_t **array, L

// Hashing
static ULong_t Hash(const void *txt, Int_t ntxt);
static ULong_t Hash(const char *str);

// IsInside
static Bool_t IsInside(Double_t xp, Double_t yp, Int_t np,
static Bool_t IsInside(Float_t xp, Float_t yp, Int_t np, F
static Bool_t IsInside(Int_t xp, Int_t yp, Int_t np, Int_t

// Sorting
static void Sort(Int_t n, const Short_t *a,  Int_t *index,
static void Sort(Int_t n, const Int_t *a,    Int_t *index,
static void Sort(Int_t n, const Float_t *a,  Int_t *index,
static void Sort(Int_t n, const Double_t *a, Int_t *index,
static void Sort(Int_t n, const Long_t *a,   Int_t *index,
static void BubbleHigh(Int_t Narr, Double_t *arr1, Int_t *
static void BubbleLow (Int_t Narr, Double_t *arr1, Int_t *
```

TMath.h

File Edit Search Preferences Shell Macro Windows Help

/home/brun/root/base/inc/TMath.h 19391 bytes

```
// Advanced
static Float_t *Cross(Float_t v1[3],Float_t v2[3],Float_t out[3]);     // Calculate the Cross
static Float_t  Normalize(Float_t v[3]);                              // Normalize a vector
static Float_t  NormCross(Float_t v1[3],Float_t v2[3],Float_t out[3]); // Calculate the Norma
static Float_t *Normal2Plane(Float_t v1[3],Float_t v2[3],Float_t v3[3], Float_t normal[3]); /

static Double_t *Cross(Double_t v1[3],Double_t v2[3],Double_t out[3]);// Calculate the Cross
static Double_t  Erf(Double_t x);
static Double_t  Erfc(Double_t x);
static Double_t  Freq(Double_t x);
static Double_t  Gamma(Double_t z);
static Double_t  Gamma(Double_t a,Double_t x);
static Double_t  BreitWigner(Double_t x, Double_t mean=0, Double_t gamma=1);
static Double_t  Gaus(Double_t x, Double_t mean=0, Double_t sigma=1);
static Double_t  Landau(Double_t x, Double_t mean=0, Double_t sigma=1);
static Double_t  LnGamma(Double_t z);
static Double_t  Normalize(Double_t v[3]);                            // Normalize a vector
static Double_t  NormCross(Double_t v1[3],Double_t v2[3],Double_t out[3]); // Calculate the N
static Double_t *Normal2Plane(Double_t v1[3],Double_t v2[3],Double_t v3[3], Double_t normal[3
static Double_t  Poisson(Double_t x, Double_t par);
static Double_t  Prob(Double_t chi2,Int_t ndf);
static Double_t  KolmogorovProb(Double_t z);
static Double_t  Voigt(Double_t x, Double_t sigma, Double_t lg, Int_t R = 4);

// Bessel functions
static Double_t BesselI(Int_t n,Double_t x);      // integer order modified Bessel function I
static Double_t BesselK(Int_t n,Double_t x);      // integer order modified Bessel function K
static Double_t BesselI0(Double_t x);             // modified Bessel function I_0(x)
static Double_t BesselK0(Double_t x);             // modified Bessel function K_0(x)
static Double_t BesselI1(Double_t x);             // modified Bessel function I_1(x)
static Double_t BesselK1(Double_t x);             // modified Bessel function K_1(x)
static Double_t BesselJ0(Double_t x);             // Bessel function J0(x) for any real x
static Double_t BesselJ1(Double_t x);             // Bessel function J1(x) for any real x
static Double_t BesselY0(Double_t x);             // Bessel function Y0(x) for positive x
static Double_t BesselY1(Double_t x);             // Bessel function Y1(x) for positive x
static Double_t StruveH0(Double_t x);             // Struve functions of order 0
static Double_t StruveH1(Double_t x);             // Struve functions of order 1
static Double_t StruveL0(Double_t x);             // Modified Struve functions of order 0
static Double_t StruveL1(Double_t x);             // Modified Struve functions of order 1
```

TMath.h

File Edit Search Preferences Shell Macro Windows

/home/brun/root/base/inc/TMath.h 19391 bytes

```
// Min
static Short_t  Min(Short_t a, Short_t b);
static UShort_t Min(UShort_t a, UShort_t b);
static Int_t    Min(Int_t a, Int_t b);
static UInt_t   Min(UInt_t a, UInt_t b);
static Long_t   Min(Long_t a, Long_t b);
static ULong_t  Min(ULong_t a, ULong_t b);
static Float_t  Min(Float_t a, Float_t b);
static Double_t Min(Double_t a, Double_t b);

// Max
static Short_t  Max(Short_t a, Short_t b);
static UShort_t Max(UShort_t a, UShort_t b);
static Int_t    Max(Int_t a, Int_t b);
static UInt_t   Max(UInt_t a, UInt_t b);
static Long_t   Max(Long_t a, Long_t b);
static ULong_t  Max(ULong_t a, ULong_t b);
static Float_t  Max(Float_t a, Float_t b);
static Double_t Max(Double_t a, Double_t b);

// Locate Min, Max
static Int_t  LocMin(Int_t n, const Short_t *
static Int_t  LocMin(Int_t n, const Int_t *a)
static Int_t  LocMin(Int_t n, const Float_t *
static Int_t  LocMin(Int_t n, const Double_t
static Int_t  LocMin(Int_t n, const Long_t *a
static Int_t  LocMax(Int_t n, const Short_t *a);
static Int_t  LocMax(Int_t n, const Int_t *a);
static Int_t  LocMax(Int_t n, const Float_t *a);
static Int_t  LocMax(Int_t n, const Double_t *a);
static Int_t  LocMax(Int_t n, const Long_t *a);
```

New Features and Classes

- TFractionFitter (Frank Filthaut)
  - Fits MC fractions to data histogram
- Confidence Level classes (Christophe Delaere)
  - TLimit, TLimitDataSource and TConfidenceLevel
- TFeldmanCousins for confidence level (Adrian Bevan)

TerraFerMA
A Suite of Multivariate Analysis Tools

Sherry Towers
SUNY-SB
smjt@fnal.gov

Version 1.0 has been released
useable by anyone with access to the Root libraries

www-d0.fnal.gov/~smjt/multiv.html

TerraFerMA=Fermilab Multivariate Analysis (aka "FerMA")

TerraFerMA is, foremost, a convenient interface to various disparate multivariate analysis packages (ex: MLPfit, Jetnet, PDE/GEM, Fisher discriminant, binned likelihood, etc)

User first fills signal and background (and data) "Samples", which are then used as input to TerraFerMA methods. A Sample consists of variables filled for many different events.

Using a multivariate package chosen by user (ie; NN's, PDE's, Fisher Discriminants, etc), TerraFerMA methods yield probability that a data event is signal or background.

TerraFerMA also includes useful statistics tools (means, RMS's, and correlations between the variables in a Sample), and a method to detect outliers.

TStatUtils: Statistics
by Christian Stratowa

Proceedings of DSC 2003 in Vienna 25-29 March

Novel high-throughput technologies such as DNA microarray analyses are allowing biologists to generate sets of data in the terabyte realm. Many of these data will be deposited in the public domain, necessitating a common standard. Currently available database systems are not appropriate for these intentions.

In this paper, I will introduce ROOT (http://root.cern.ch), an object oriented framework that has been developed at CERN for distributed data warehousing and data mining of particle data in the petabyte range. Data are stored as sets of objects in machine-independent files, and specialized methods are used to get direct access to separate attributes of selected data objects. ROOT has been designed in such a way that it can query its databases in parallel on SMP/MPP machines, on clusters of PC's, or using common GRID services.

TStatUtils: Statistics
by Christian Stratowa

In order to demonstrate the applicability of ROOT to microarray data, I will present a functional prototype system, called XPS - eXpression Profiling System, which can be considered to be an alternative to the Bioconductor project. The current implementation handles the storage of GeneChip schemes and data, and the pre-processing, normalization and filtering of GeneChip data.
Based on this system, I will propose a novel standard for the distributed storage of microarray data.
Finally, I will emphasize the similarities between R and ROOT, and show how R could be easily extended to access ROOT from within R.

Proceedings of DSC 2003

System Infrastructure

System Enhancement

- Plugin Manager
  - Allow easy extension of abstract interfaces

```
# base class    regexp     plugin class   plugin lib   ctor or factory
Plugin.TFile:   ^rfio:     TRFIOFile      RFIO         "TRFIOFile(const char*,Option_t*,const char*,Int_t)"
+Plugin.TFile: ^dcache:    TDCacheFile    DCache       "TDCacheFile(const char*,Option_t*,const char*,Int_t)"
```

- ACLiC
  - Full dependency check
  - Allow selection of debug or optimized compilation
  - Building of read-only scripts directories
  - Ability to use CINT's pragma anywhere
  - Root > .x script.C (interpreted by CINT)
  - Root > .x script.C++ (by the compiler)

Port to new platforms

- Port to MacOS
  - Ben Cowan, Keisuke Fujii, George Irwin, etc.
- Port to Windows with VisualC++7.NET
  - Matt Langston
- Port to Windows with cygwin and gcc 3.2
  - Axel Naumann
- Port to Intel Compiler v7
  - Fons Rademakers
- Port to Itanium 64
  - Fons Rademakers

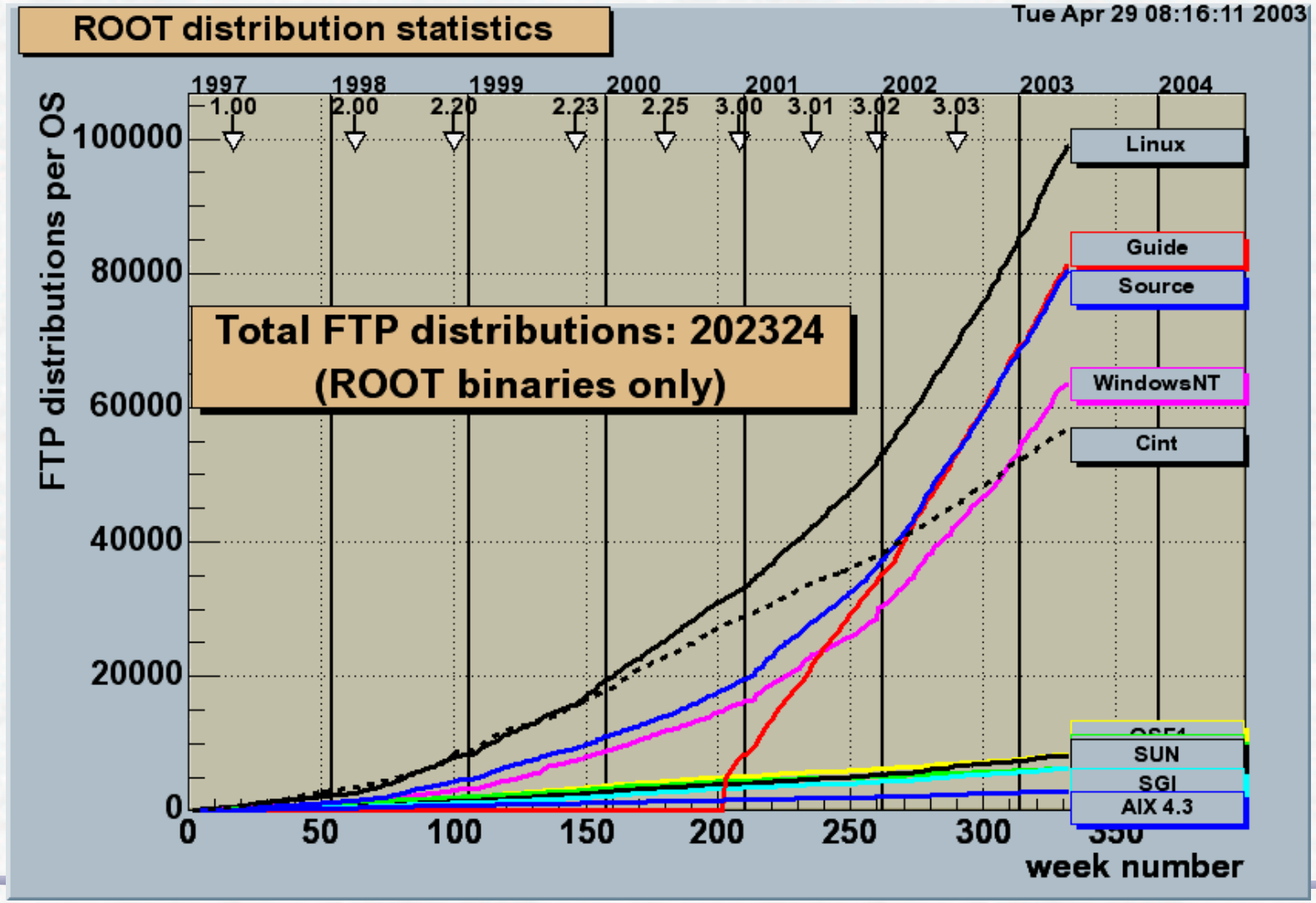
Others future developments

- New online help system
  - Will generate a compact file containing the documentation
  - An interface will allow easy lookup in this file
  - Information will be available in any ROOT process

Documentation: Users Guide by Ilka Antcheva

- New version of the Users Guide
  - Improved style
  - Improved Index
  - New chapters;
    - GUI
    - Geometry Package

Downloads Total



Monthly Downloads