ROOT
A Database and Analysis Tool

Dr. Fons Rademakers
ROOT Project Leader
CERN

FDT2, Frascati, 29-Nov-2011.

Introduction

- The ROOT data analysis framework
- Distributed analysis of very large data bases
- Summary

The ROOT Data Analysis Framework

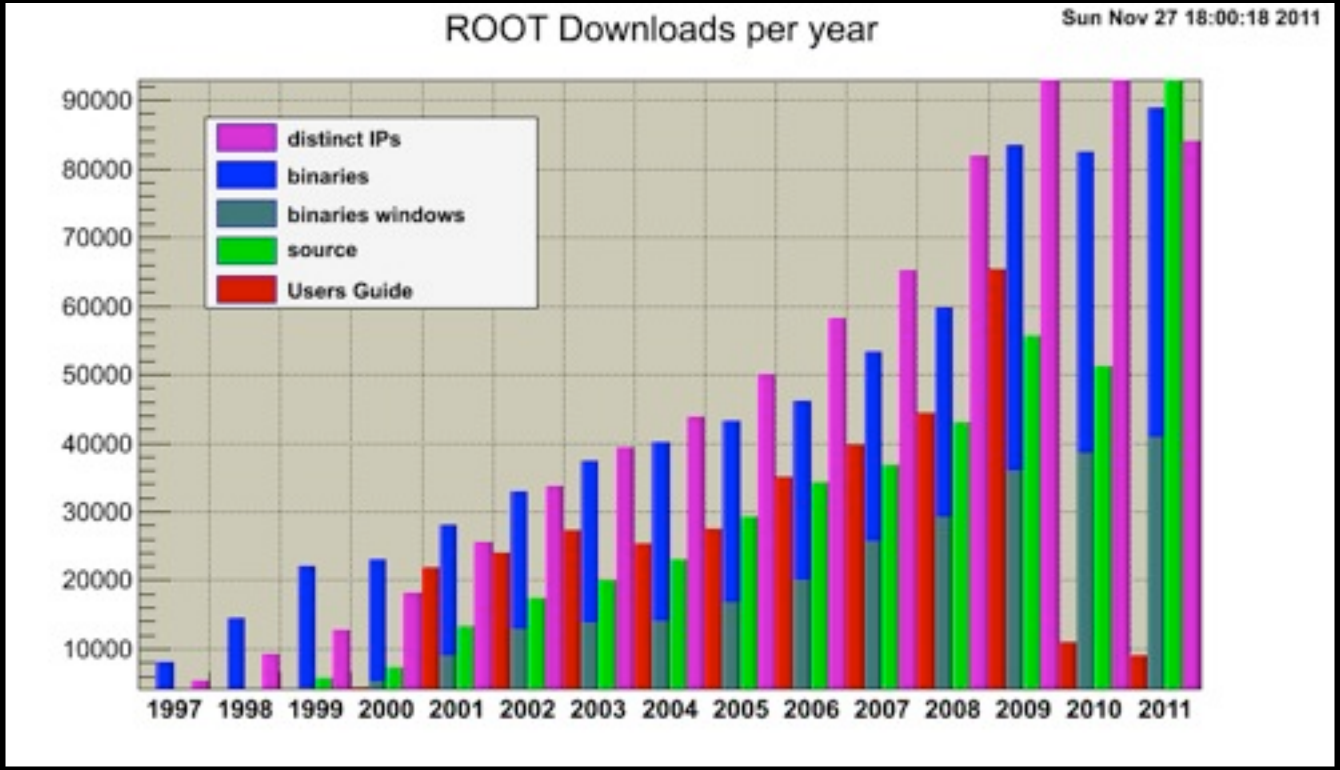
- ROOT is a extensive data handling and analysis framework
  - Efficient object data store scaling from KB's to PB's
  - C++ interpreter
  - Extensive 2D+3D scientific data visualization capabilities
  - Extensive set of data fitting, modeling and analysis methods
  - Complete set of GUI widgets
  - Classes for threading, shared memory, networking, etc.
  - Parallel version of analysis engine runs on clusters and multi-core
  - Fully cross platform, Unix/Linux, Mac OS X and Windows
  - 3.1 million lines of C++, building into more than 100 shared libs
  - Licensed under the LGPL
- Used by all HEP experiments in the world
- Used in many other scientific fields and in commercial world

A Little ROOT History

- Development started in Jan 1995
- First two years two developers
- First presentation and release Nov 1995
- Against the will of CERN management
  - Commercial solutions by professional software companies was the management line
- First usage in a small CERN heavy ion experiment NA49
  - Good precursor for our final target the LHC
- Followed by the Frascati experiment Finuda
- Followed by the Fermilab experiments CDF and D0
  - Fermilab assigned two FTE's to ROOT
- Followed by BNL, SLAC and DESY
- And finally followed by the CERN LHC experiments

ROOT Usage Stats

- ROOT binaries have been downloaded more than 800000 times since 1997, and currently more than 90000 binaries per year
- There are about 5800 people registered on the RootTalk forum
- An estimated user community of about 25000 people



The Importance of the C++ Interpreter

- CINT is the core of ROOT for:
  - Parsing and interpreting code in macros and on command line
  - Providing class reflection information
  - Generating function/method calling stubs
- We are working on moving from CINT to LLVM as new interpreter technology

```
bash$ root
root [0] TH1F *hpx = new TH1F("hpx","This is the px distribution",100,-1,1);
root [1] for (Int_t i = 0; i < 25000; i++) hpx->Fill(gRandom->Rndm());
root [2] hpx->Draw();
```

```
bash$ cat script.C
{
   TH1F *hpx = new TH1F("hpx","This is the px distribution",100,-1,1);
   for (Int_t i = 0; i < 25000; i++) hpx->Fill(gRandom->Rndm());
   hpx->Draw();
}
bash$ root
root [0] .x script.C
```

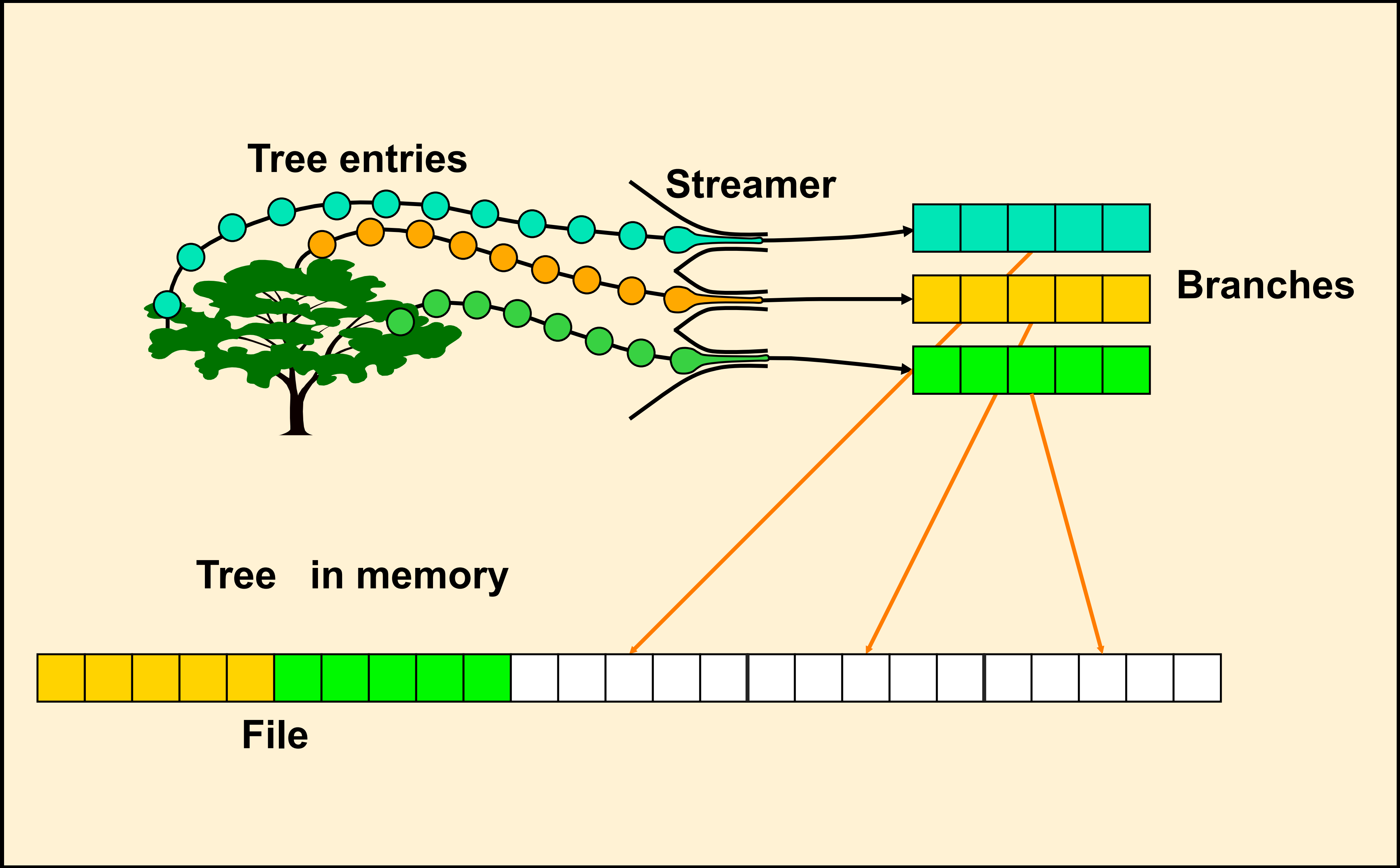
ROOT Object Persistency

- Scalable, efficient, machine independent format
- Orthogonal to object model
  - Persistency does not dictate object model
- Based on object serialization to a buffer
- Automatic schema evolution (backward and forward compatibility)
- Object versioning
- Compression
- Easily tunable granularity and clustering
- Remote access
- Self describing file format (stores reflection information)
- ROOT I/O is used to store all LHC data (actually all HEP data)

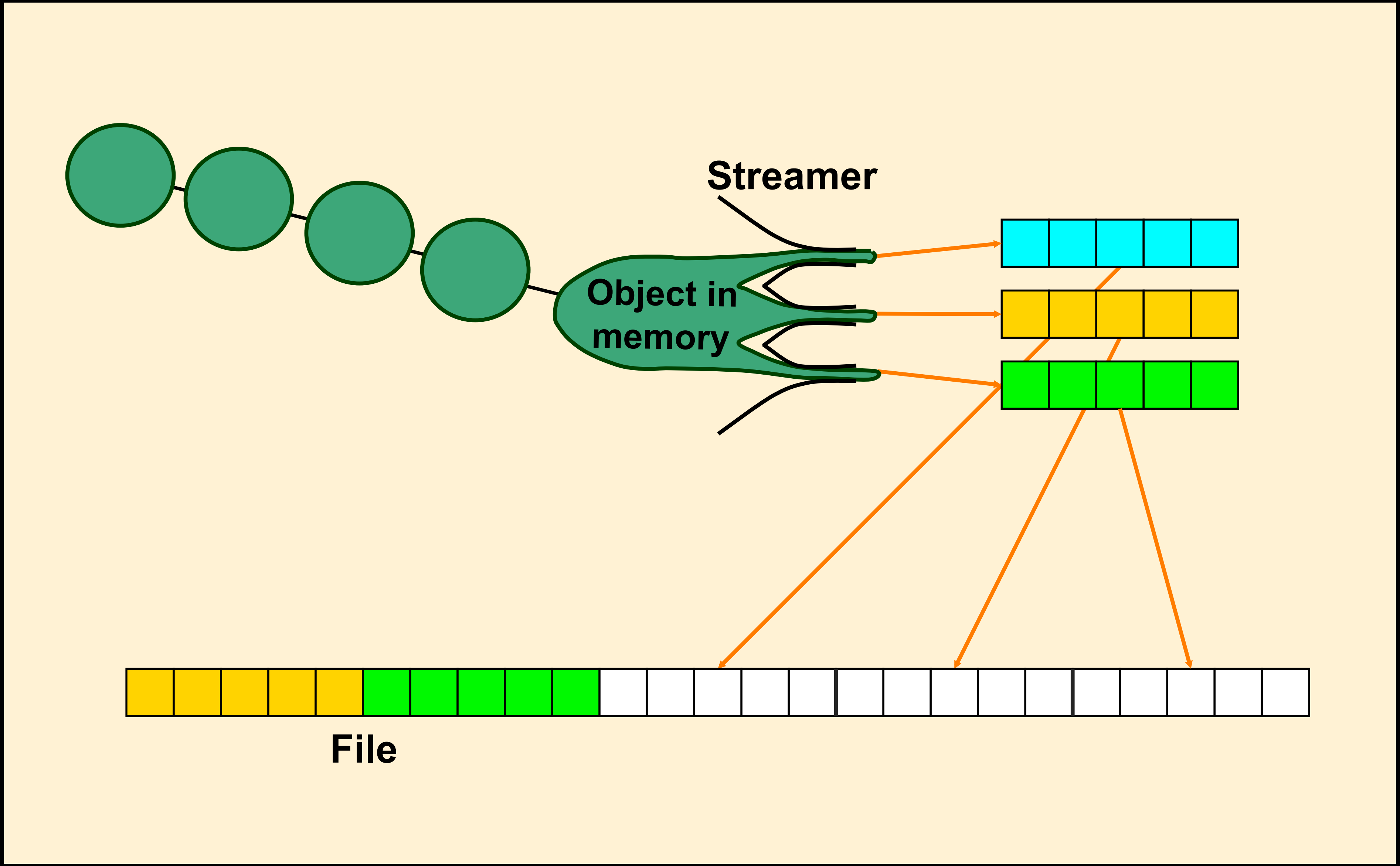
Object Containers - TTree's

- Special container for very large number of objects of the same type (events)
- Minimum amount of overhead per entry
- Objects can be clustered per sub object or even per single attribute (clusters are called branches)
- Each branch can be read individually
- Industry calls this "Vertical Data Storage"

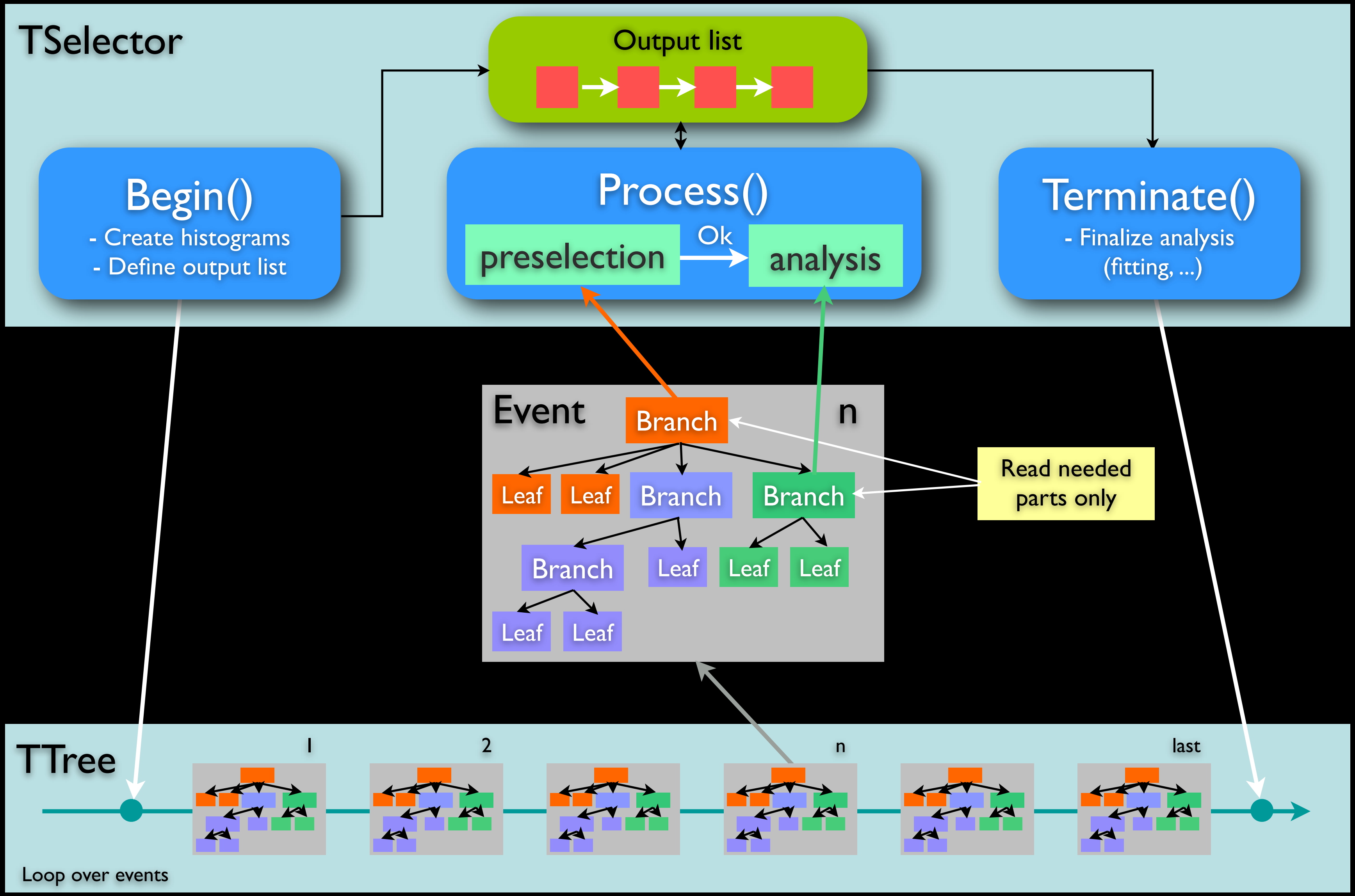
TTree - Clustering per Object



TTree - Clustering per Attribute



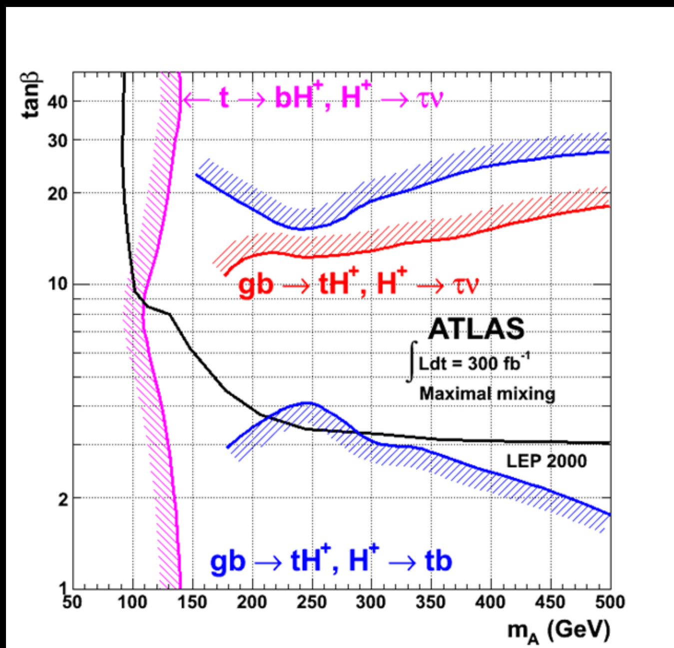
Processing a TTree

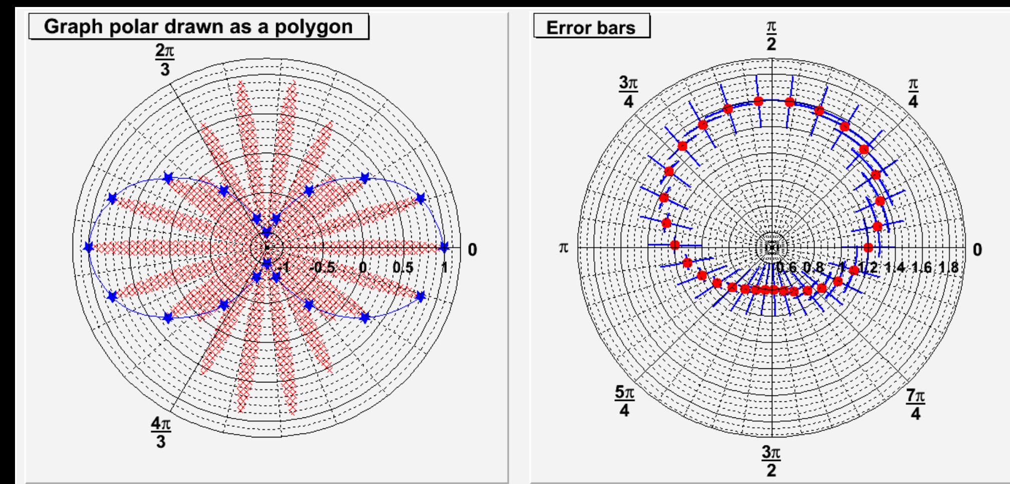


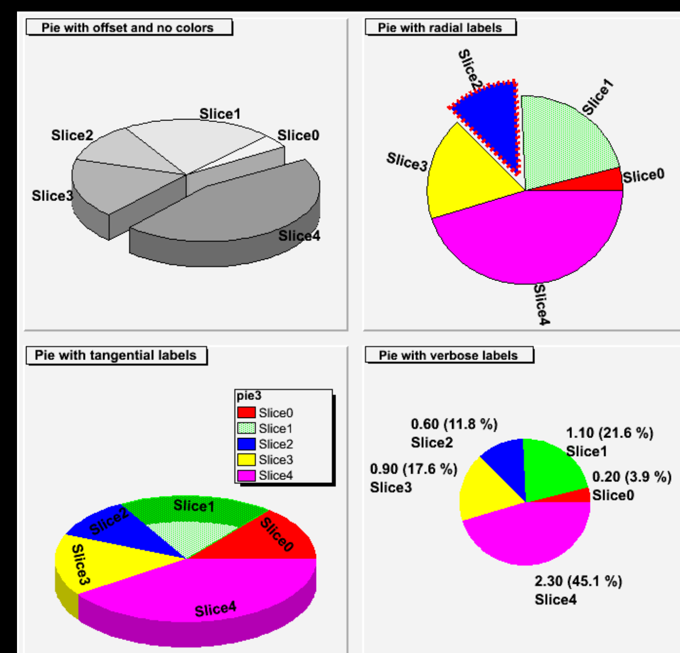
TSelector::Process()

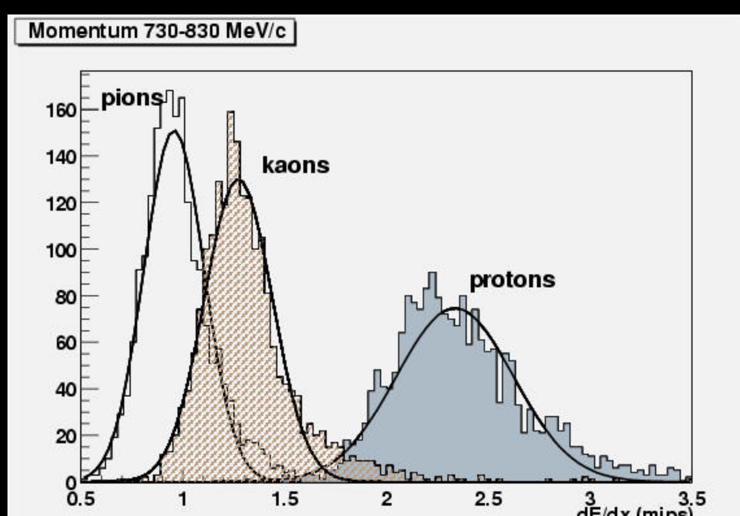
```
...
...
// select event
b_nlhk->GetEntry(entry);          if (nlhk[ik] <= 0.1)    return kFALSE;
b_nlhpi->GetEntry(entry);         if (nlhpi[ipi] <= 0.1)  return kFALSE;
b_ipis->GetEntry(entry); ipis--; if (nlhpi[ipis] <= 0.1) return kFALSE;
b_njets->GetEntry(entry);         if (njets < 1)          return kFALSE;

// selection made, now analyze event
b_dm_d->GetEntry(entry);          //read branch holding dm_d
b_rpd0_t->GetEntry(entry);        //read branch holding rpd0_t
b_ptd0_d->GetEntry(entry);        //read branch holding ptd0_d

//fill some histograms
hdmd->Fill(dm_d);
h2->Fill(dm_d,rpd0_t/0.029979*1.8646/ptd0_d);
...
...
```

ROOT Scientific Graphics

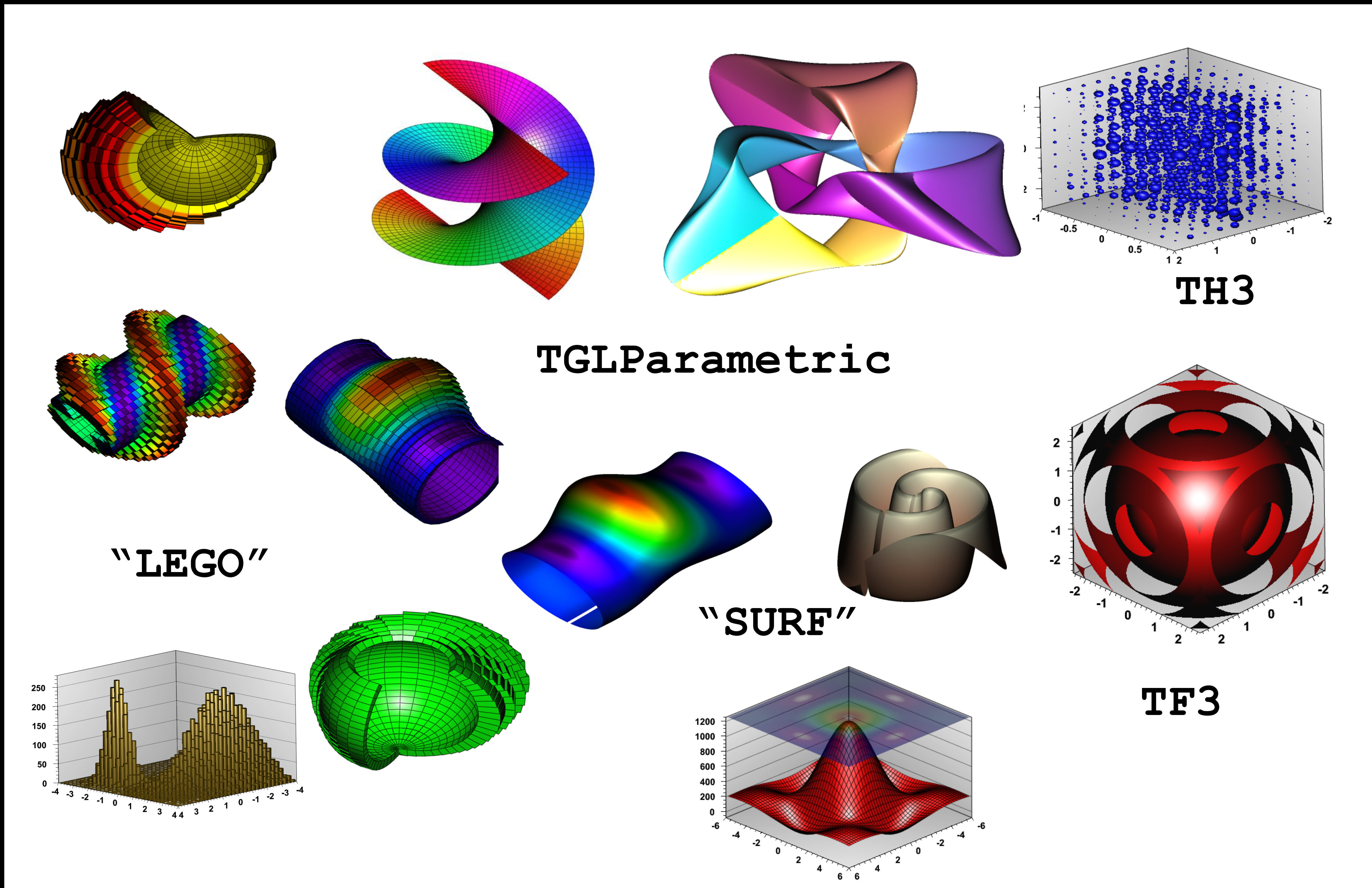




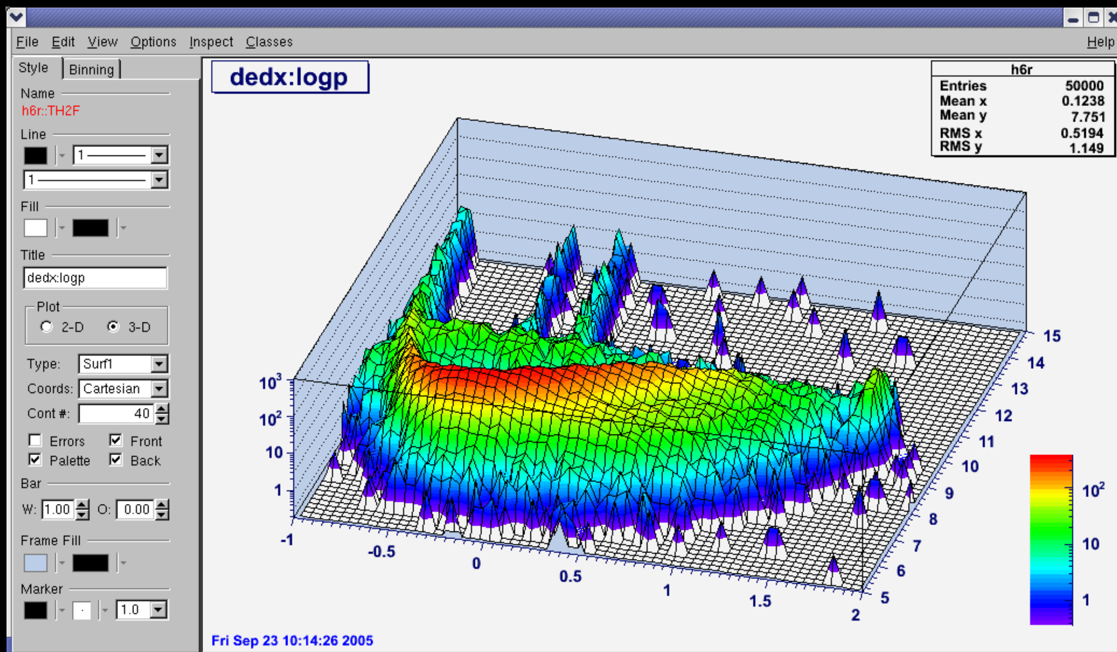


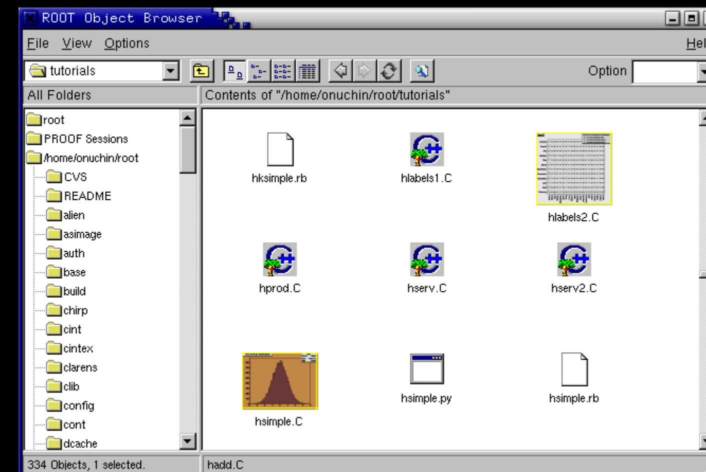


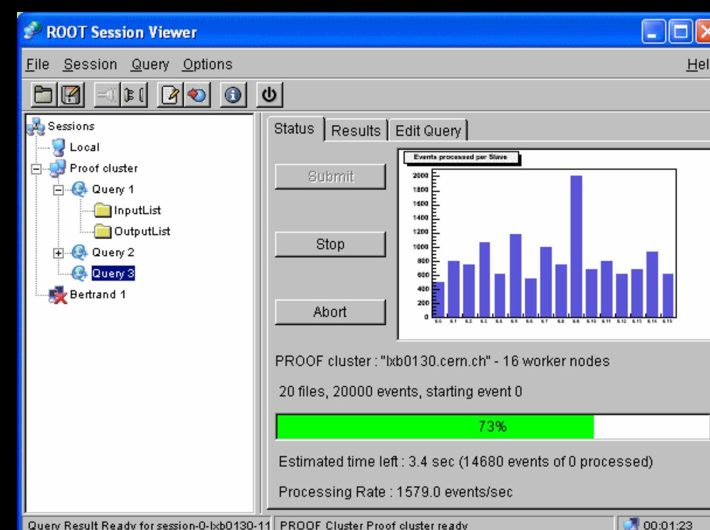
More Graphics

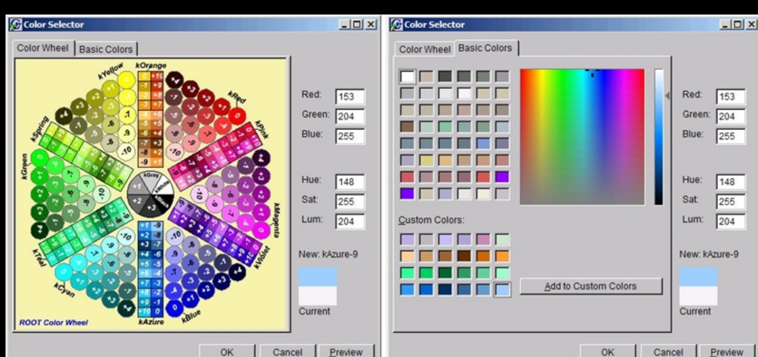


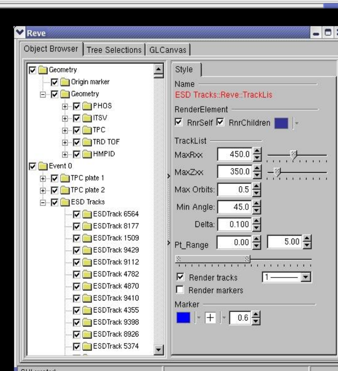
ROOT Cross Platform GUI



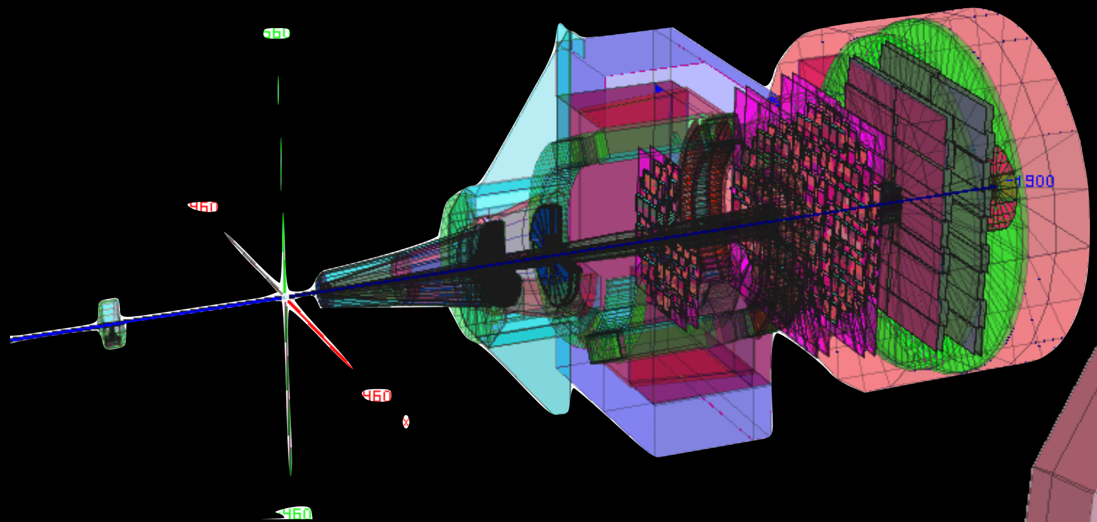


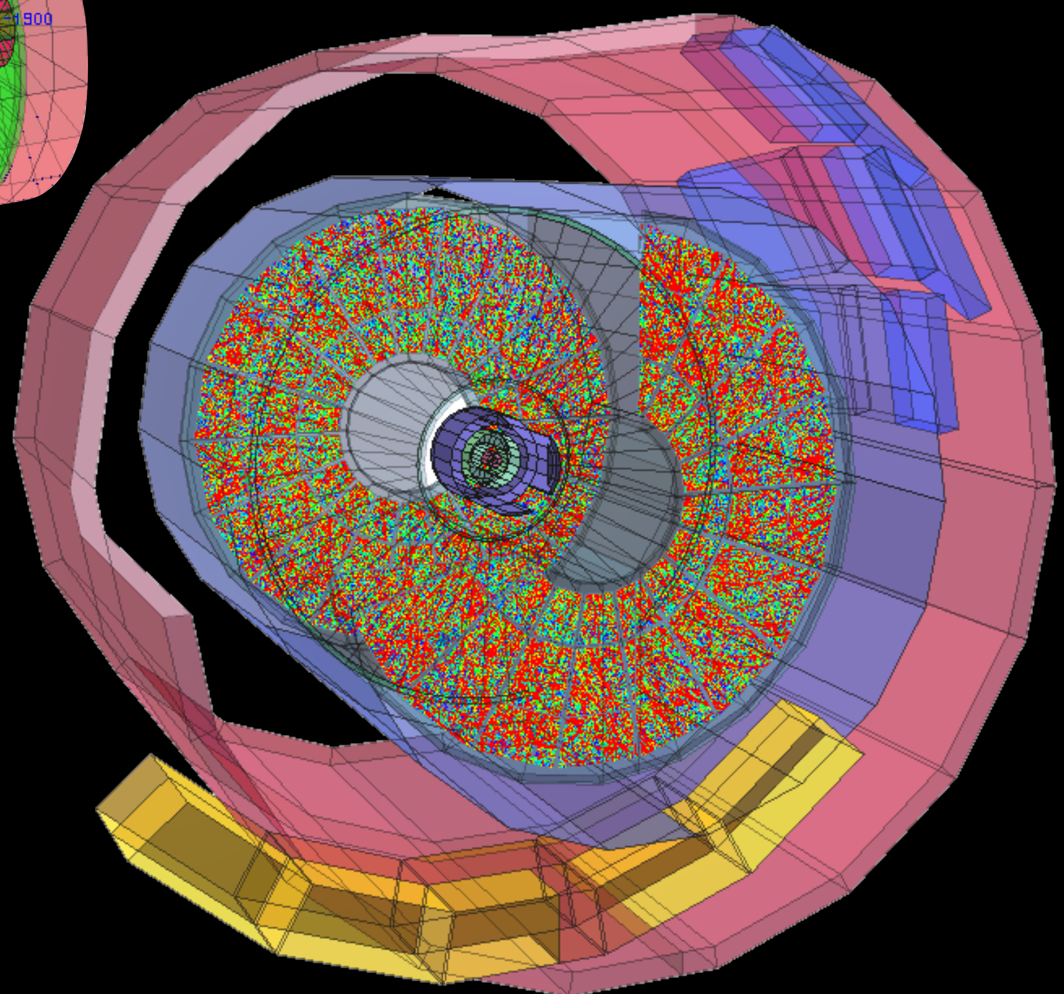




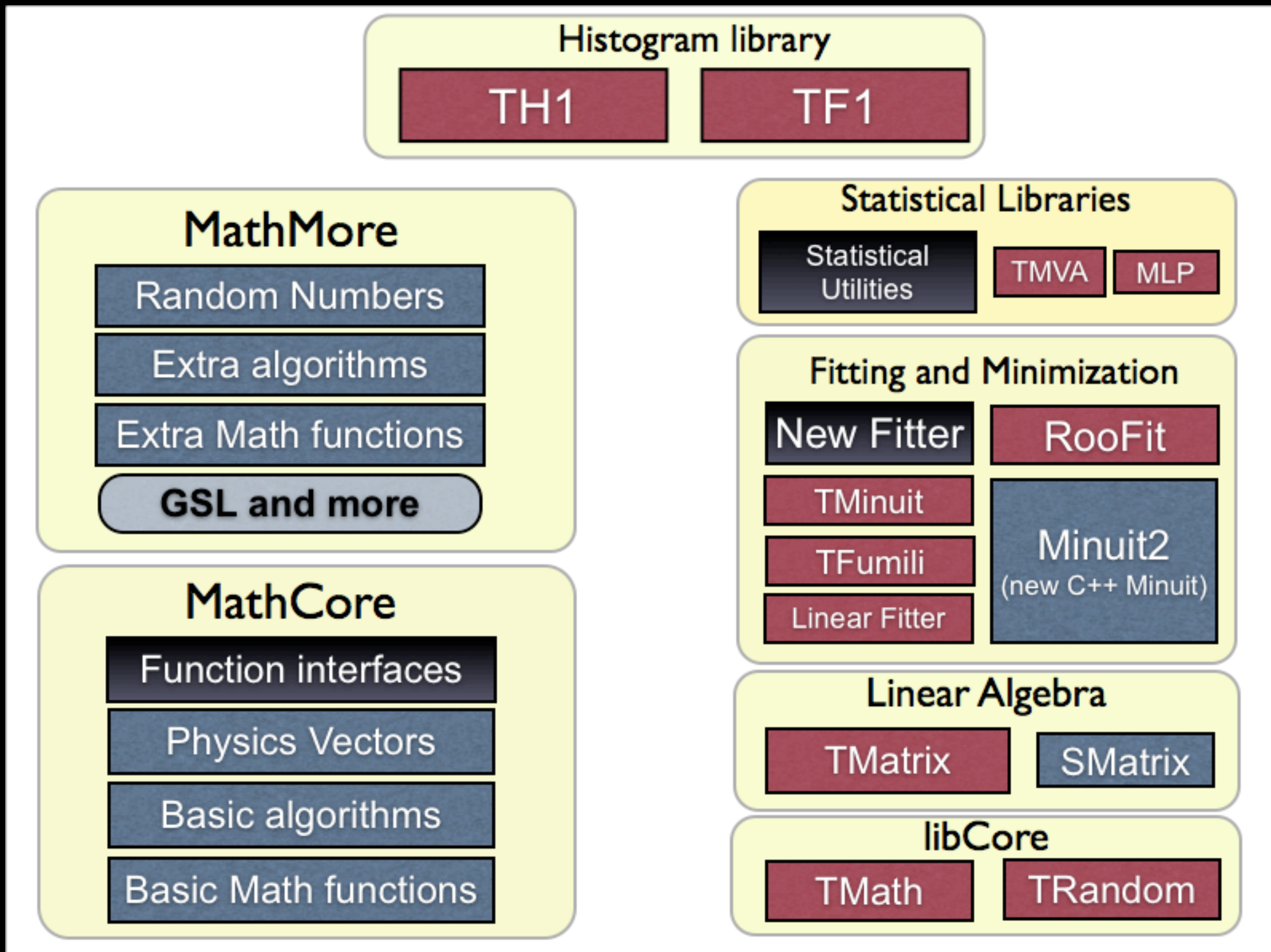


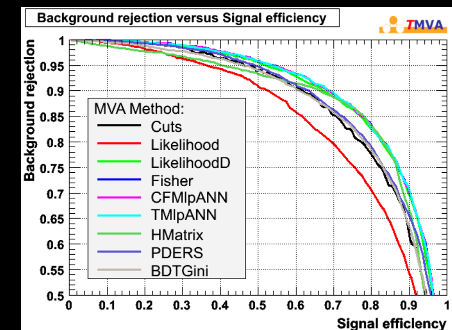
Complex Geometries

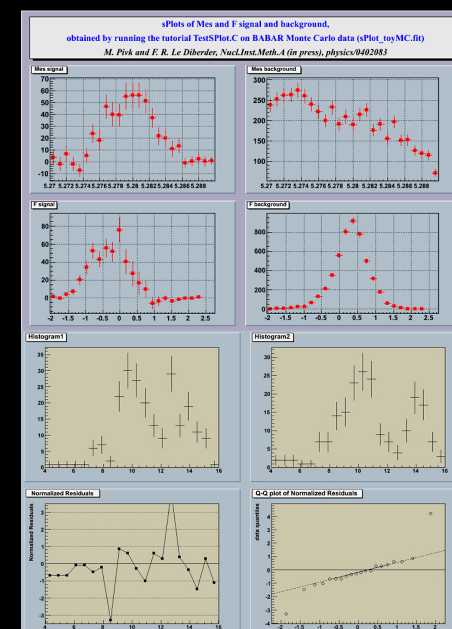


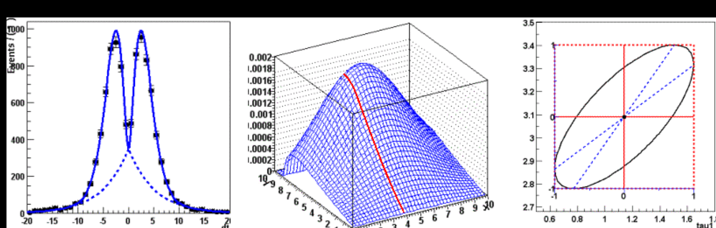


ROOT Math/Stat Libraries









RooFit/RooStats

- Framework for statistical calculations
  - Works on arbitrary models and datasets
  - Implements most accepted techniques (frequentists, Bayesian and likelihood based methods)
- Common purposes:
  - Point estimation: determine the best estimate of a parameter
  - Estimation of confidence (credible) intervals: lower/higher limit or multi-dimensional contours
  - Hypothesis tests: evaluation of p-values (e.g discovery significance)
  - Goodness-of-fit: how well a model describes the data
- Analysis combination:
  - Provide utilities to build a combined model
    - Full information available to treat correlations
- Digital publishing and sharing of results

RooFit

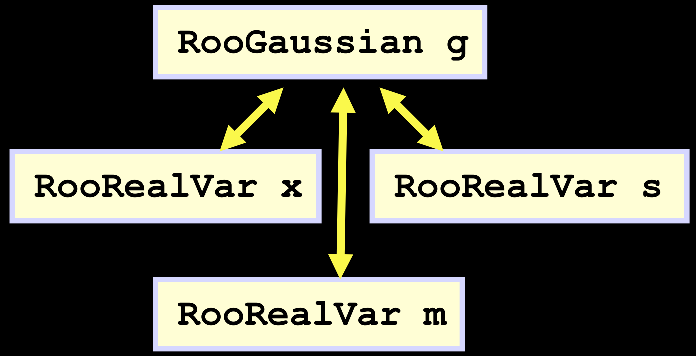
- Toolkit for data modeling (by W. Verkerke and D. Kirkby)
  - Model probability density function (pdf):
    - P(x;p,q)
      x: observables, p,q: parameters
- Functionality for building the pdf's
  - Complex model building from standard components
  - Composition with addition, product and convolution
- All models (pdf) provide the functionality for
  - Fitting of models to data sets
  - Toy data sets Monte Carlo generation
  - Visualization of models and data with ROOT graphics

RooFit Modeling

Mathematical concepts are represented as C++ objects

Mathematical concept		RooFit class
variable	x	`RooRealVar`
function	$f(x)$	`RooAbsReal`
PDF	$f(x)$	`RooAbsPdf`
space point	$\vec{x}$	`RooArgSet`
integral	$\int_{x_{\min}}^{x_{\max}} f(x)dx$	`RooRealIntegral`
list of space points		`RooAbsData`

$Gaus(x,m,s)$




```
RooRealVar x("x","x",2,-10,10)
RooRealVar s("s","s",3) ;
RooRealVar m("m","m",0) ;
RooGaussian g("g","g",x,m,s)
```

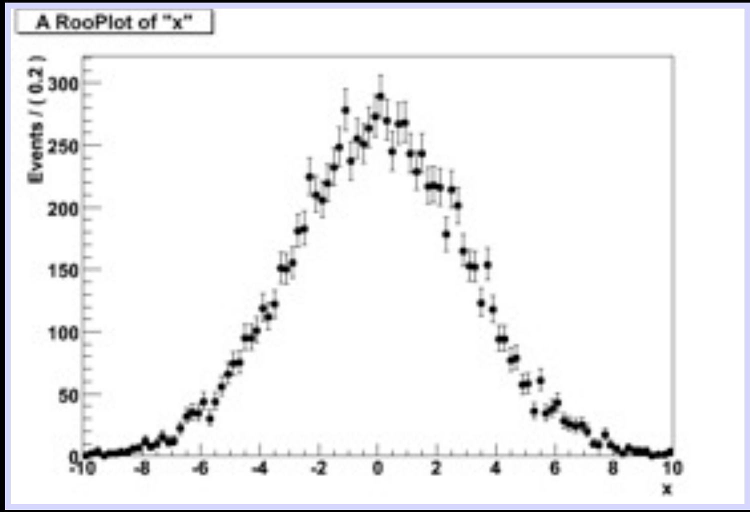
Provides a factory to auto-generates objects from a math-like language

```
RooWorkspace w;
w.factory("Gaussian::g(x[2,-10,10],m[0],s[3])")
```

RooFit Functionality

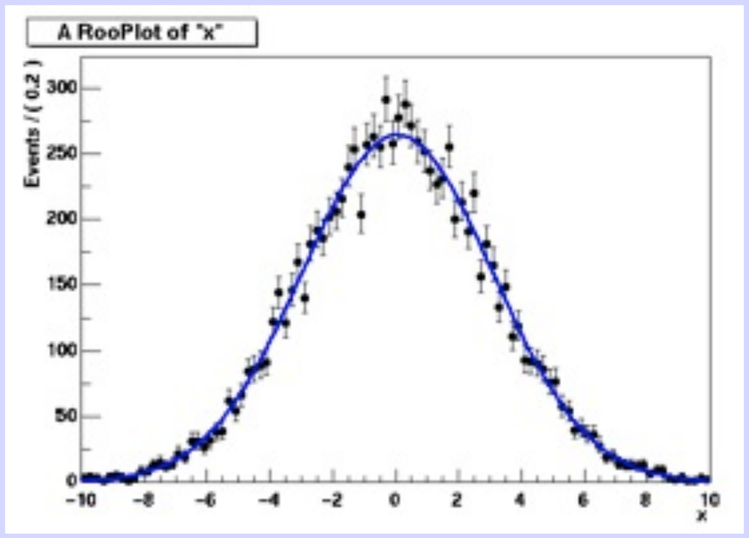
- Toy MC generation from any pdf

```
RooAbsPdf *pdf = w.pdf("g");
RooRealVar *x = w.var("x");
RooDataSet *data = pdf->generate(*x,10000);
```



- Fit of model to data
  - Maximum likelihood or least square fit
  - Different algorithms for minimization available

```
pdf = pdf->fitTo(data);
//parameters will have now fitted values
w->var("m")->Print();
w->var("s")->Print();
```

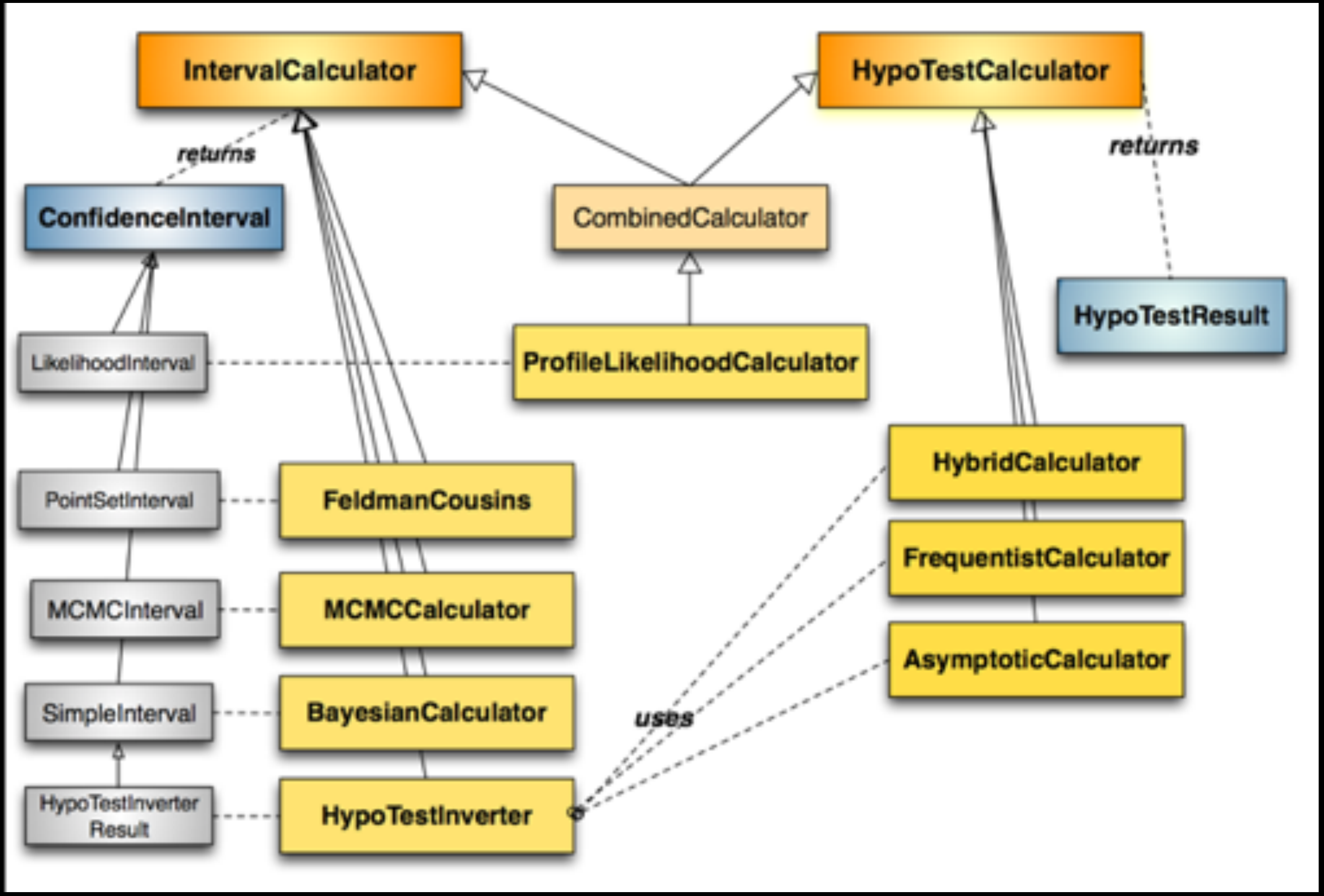


- Data and pdf visualization

```
RooAbsPdf *pdf = w.pdf("g");
RooPlot *xframe = x->frame();
data->plotOn(xframe);
pdf->plotOn(xframe);
xframe->Draw();
```

RooStats

- Framework for statistical calculations built on top of RooFit
  (by K. Cranmer, L. Moneta, G. Schott and W. Verkerke + many other contributors)
- C++ interfaces and classes mapping to real statistical concepts
  - Interval estimation or hypothesis tests



RooStats Calculator Classes

- Profile Likelihood calculator
  - Interval estimation and hypothesis testing using asymptotic properties of the likelihood function
- FeldmanCousins and Neyman construction
  - Frequentist interval calculator based on generation of toy data
- Bayesian calculators
  - Interval estimation using Bayes theorem
    `BayesianCalculator` (analytical or adaptive numerical integration)
    `MCMCCalculator` (Markov-Chain Monte Carlo)
- HybridCalculator and FrequentistCalculator
  - Frequentist hypothesis test calculators using toy data
    - Difference in treatment of nuisance parameters
- HypoTestInverter
  - Invert hypothesis test (e.g. from Hybrid or FrequentistCalculator) to estimate an interval

Example: Bayesian Analysis

- RooStats provides classes for
  - Marginalize posterior and estimate credible interval

likelihood function — prior probability — nuisance parameters marginalization

posterior probability

$$P(\mu|x) = \frac{\int L(x|\mu,\nu)\Pi(\mu,\nu)d\nu}{\underbrace{\int\int L(x|\mu,\nu)\Pi(\mu,\nu)d\mu d\nu}_{\text{normalisation term}}}$$

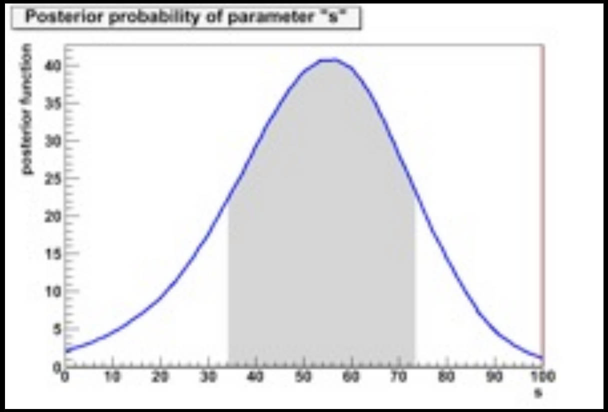
POI — data

Bayesian Theorem

  - Support for different integration algorithms:
    - Adaptive (numerical), MC integration or Markov-Chain
    - Can work with models with many parameters (e.g few hundreds)

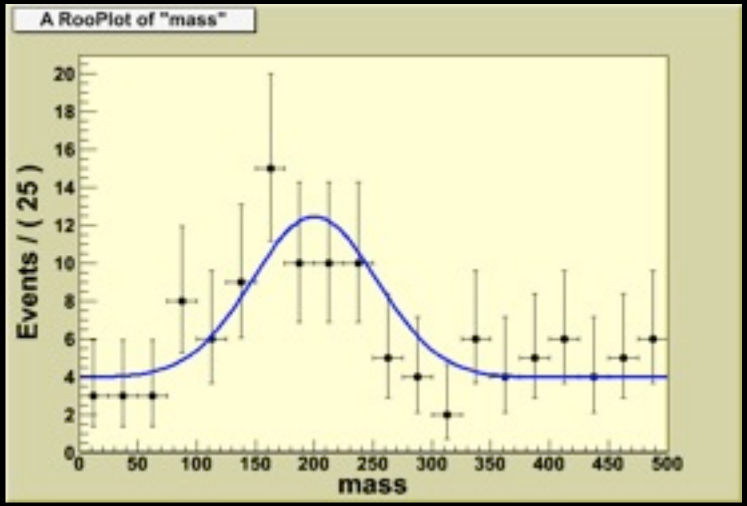
Example:
95% CL interval

```
BayesianCalculator bc(data, model);
bc.SetConfidenceLevel(0.683);
SimpleInterval *cint = bc.GetInterval();
double upperLimit = cint->UpperLimit();
RooPlot * pl = bc.GetPosteriorPlot();
pl->Draw();
```

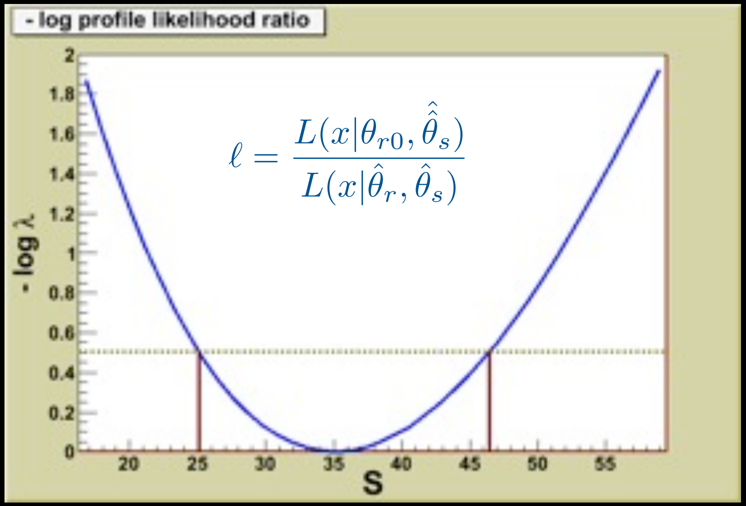


RooStats Example

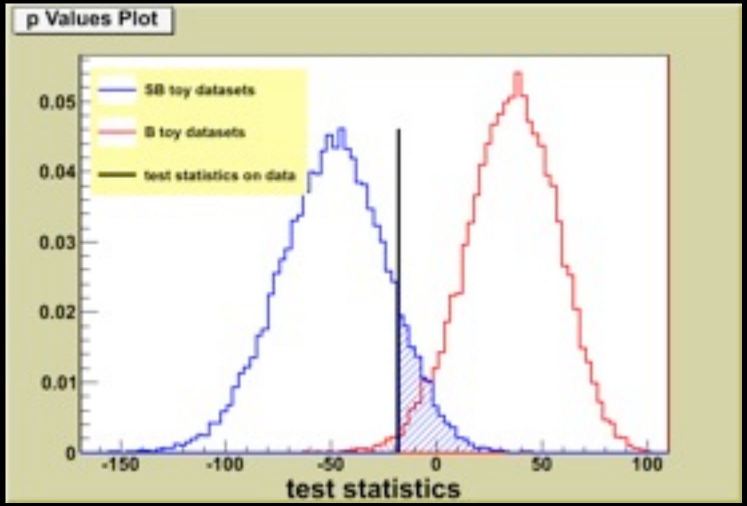
Gaussian peak over a flat background



model fit to observed data

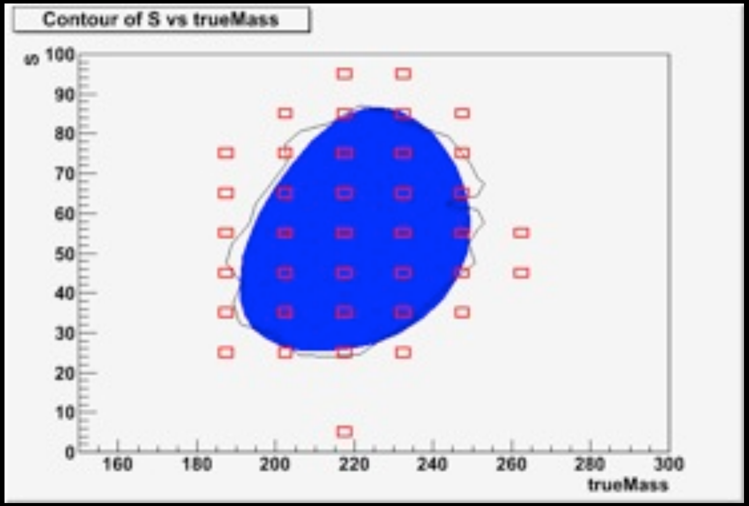


1 σ interval
from likelihood function



2d interval estimation
mass vs signal rate
result of 3 methods:

- Likelihood
- Bayes (MCMC)
- FeldmanCousins



Result on Signal Significance from hybrid calculator

Distributed Analysis

End-User Analysis Activities

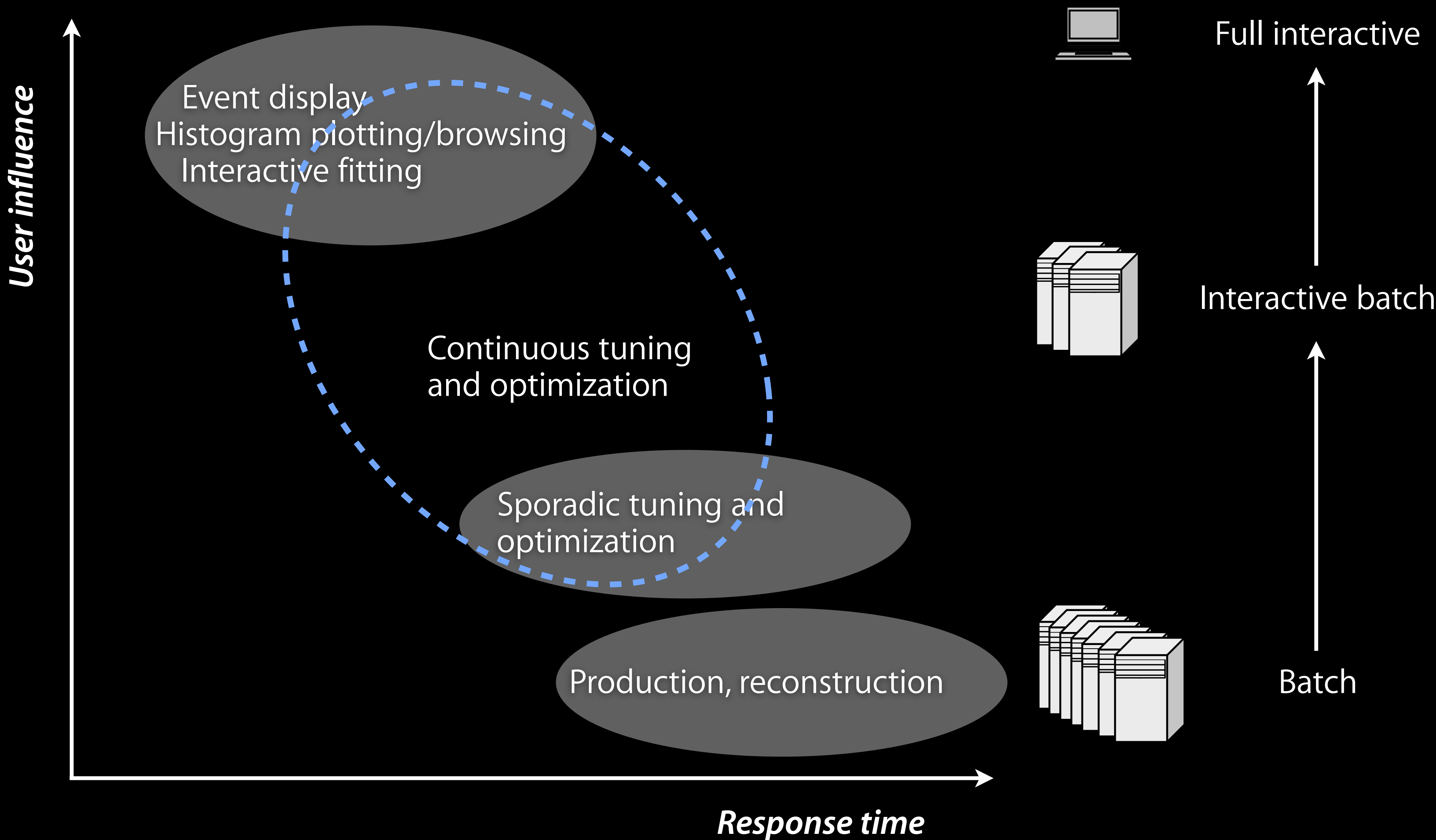
- Interactive tasks: desktop/laptop
  - Browsing output, final fits, visualization
- I/O bound tasks: data mining
  - O(1~10TB) data effectively read
  - O(10h~100h) @ ~25 MB/s (typical I/O rate)
- CPU bound tasks:
  - Complex combinatorial analysis
  - Fast “private” simulations
  - Toy Monte-Carlo’s for systematic studies

End-User Analysis Activities

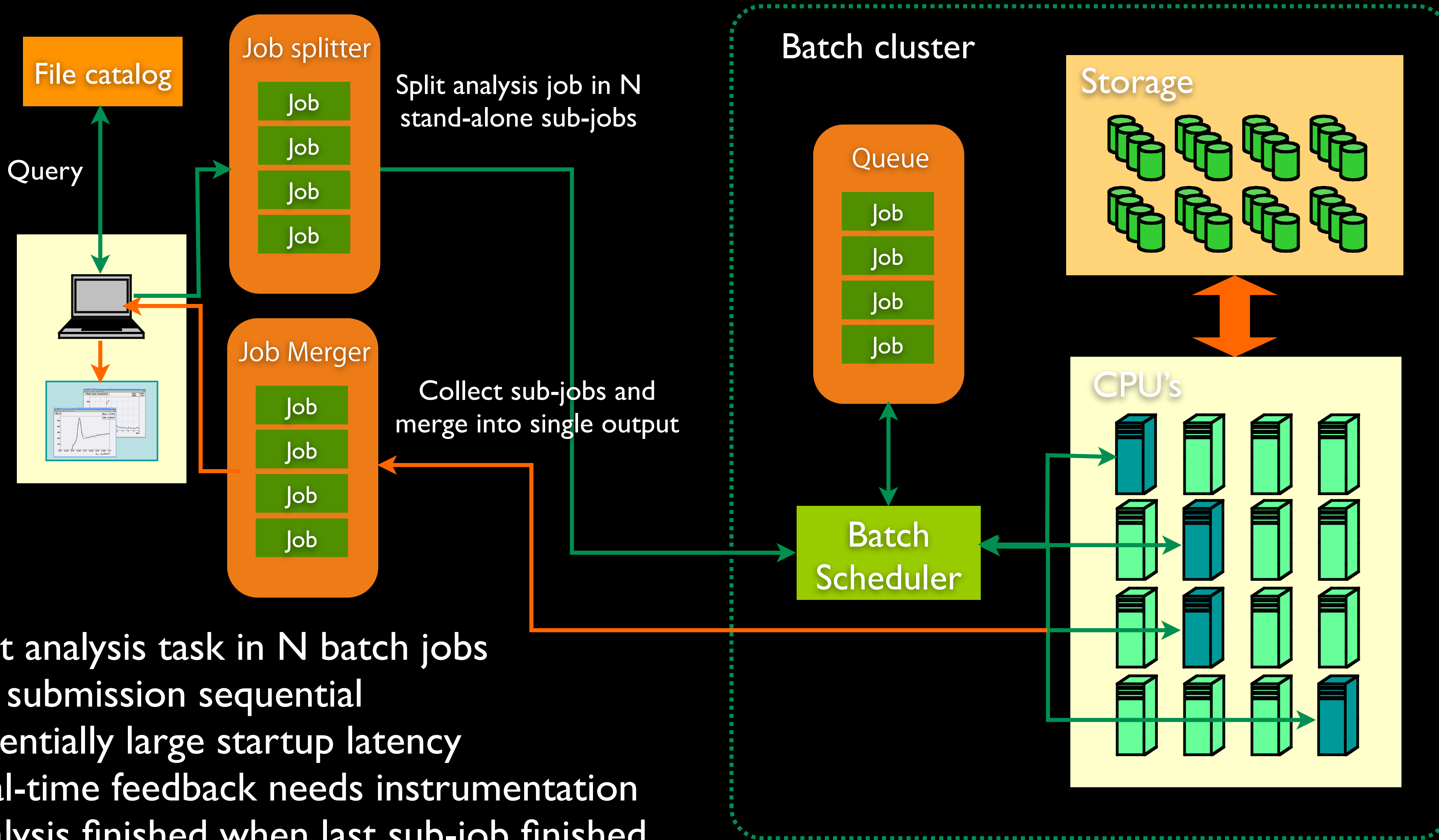
- Interactive tasks: desktop/laptop
  - Browsing output, final fits, visualization
- I/O bound tasks: data mining
  - O(1~10TB) data effectively read
  - O(10h~100h) @ ~25 MB/s (typical I/O rate)
- CPU bound tasks:
  - Complex combinatorial analysis
  - Fast "private" simulations
  - Toy Monte-Carlo's for systematic studies

Typically embarrassingly parallel tasks: just split job to get ideal parallel speedup

End-User Analysis Scenarios

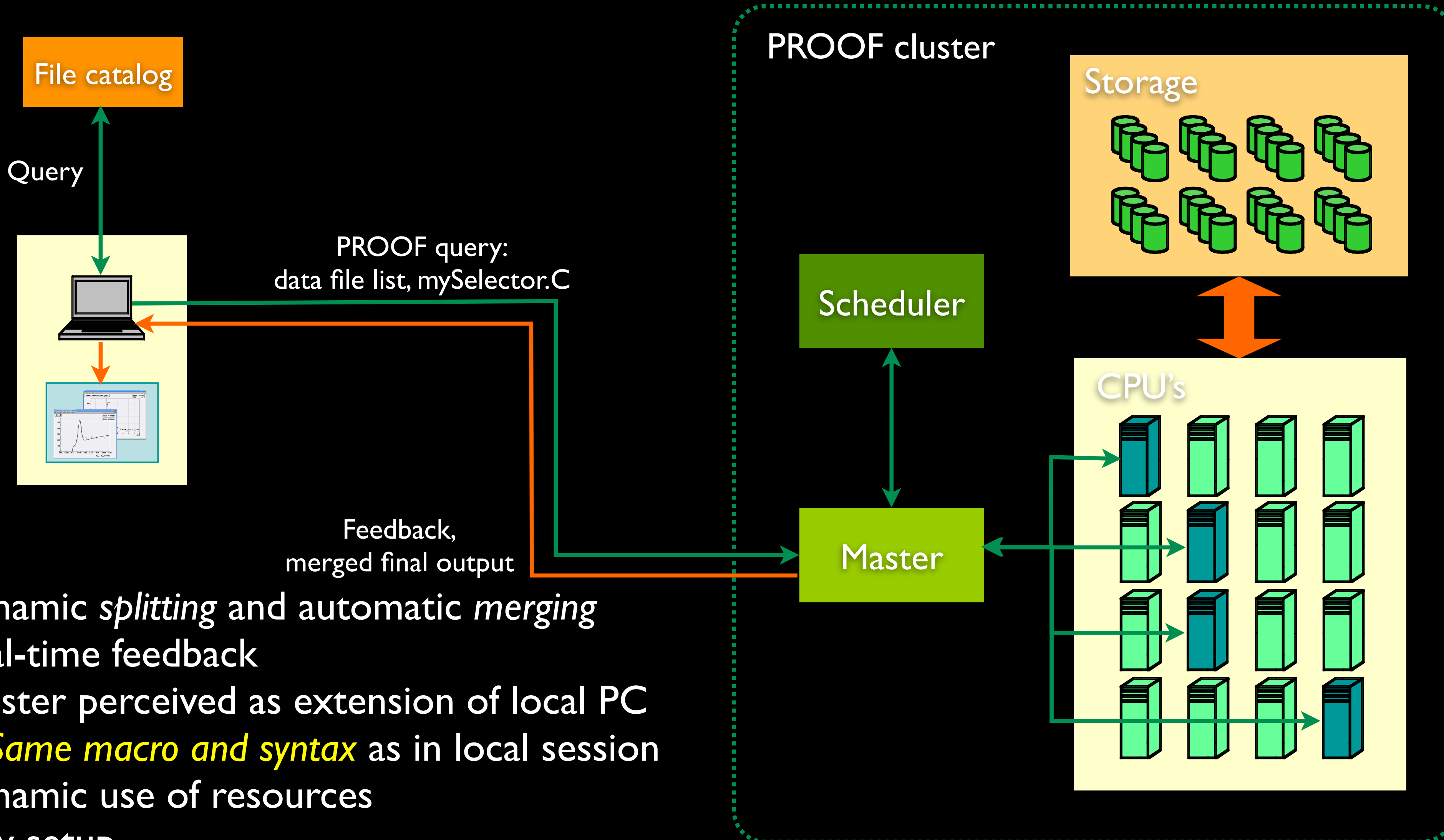


The Traditional Batch Approach



- Split analysis task in N batch jobs
- Job submission sequential
- Potentially large startup latency
- Real-time feedback needs instrumentation
- Analysis finished when last sub-job finished

The PROOF Approach

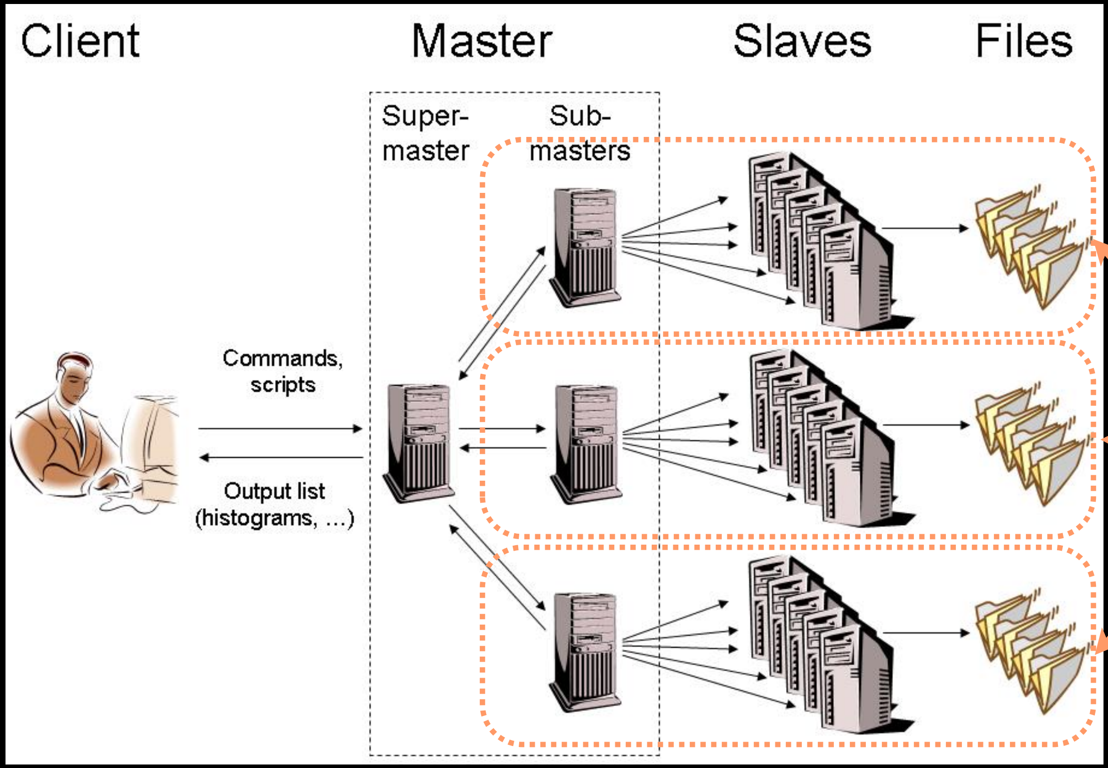


- Dynamic *splitting* and automatic *merging*
- Real-time feedback
- Cluster perceived as extension of local PC
  - *Same macro and syntax* as in local session
- Dynamic use of resources
- Easy setup

PROOF - Parallel ROOT Facility

- Parallel coordination of distributed ROOT sessions
  - Transparent: extension of the local ROOT prompt
  - Scalable: small serial overhead
- Multi process parallelism
  - Easy adaptation to broad range of setups
  - Less requirements on user code
- Process the data from the local disk, if possible
  - Output much smaller than input
  - Minimize data transfers, network overhead
- Dynamic load balancing
  - Pull architecture
  - Minimize amount of wasted cycles
- Real-time feedback, interactive
- Reduces the time to completion

Multi-Tier Architecture



Adapts to wide area *virtual* clusters

Geographically separated domains, heterogeneous machines

Network performance

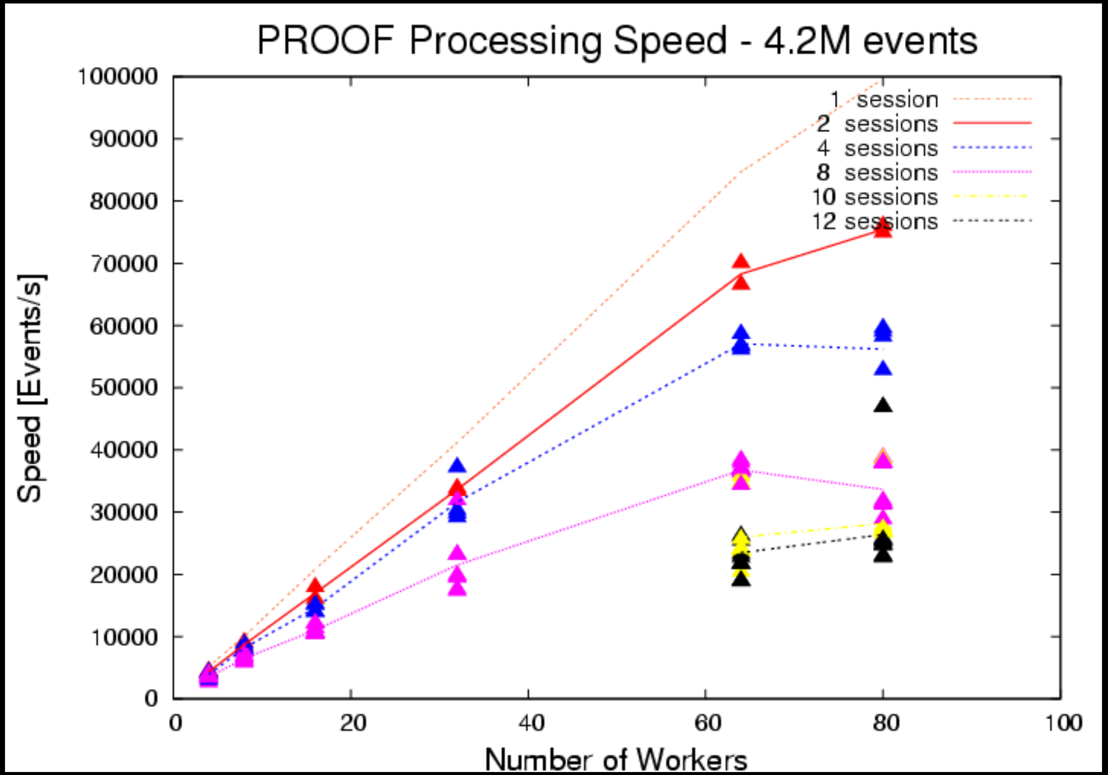
Less important → VERY important

Optimize for data locality or high bandwidth data server access

Performance - ATLAS Analysis

- Higgs 4-lepton analysis
- 50 nodes, AMD 64bit quad-core, 4 GB RAM
- 4.5 M events, 68 GB
- 845 files
- Analysis include fit

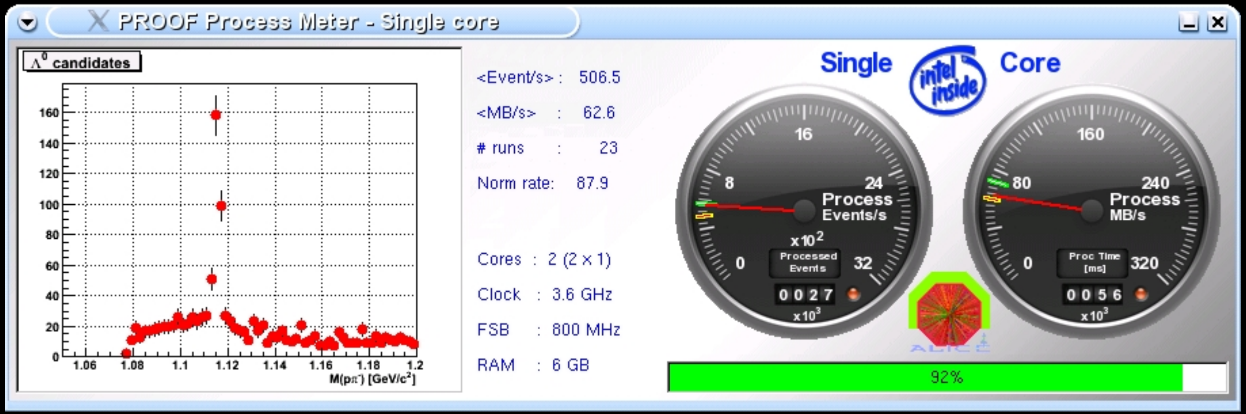
- Single session
  - 1.5 kEvt/s @ ~50 min
- PROOF 1 user (80 workers)
  - 100 kEvt/s @ ~1 min
- PROOF 8 users (64 workers)
  - 40 kEvt/s @ ~ 2.5 min



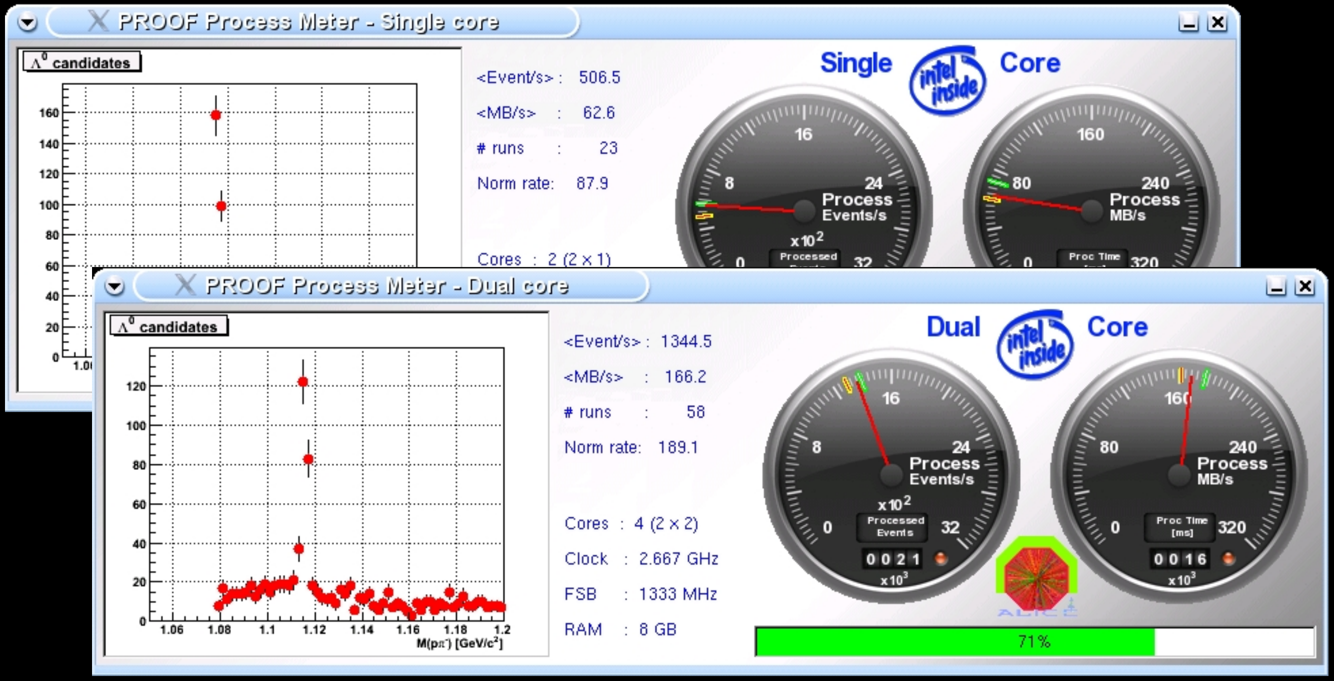
Courtesy of G.C. Montoya, Wisconsin.

PROOF Scalability on Multi-Core Machines

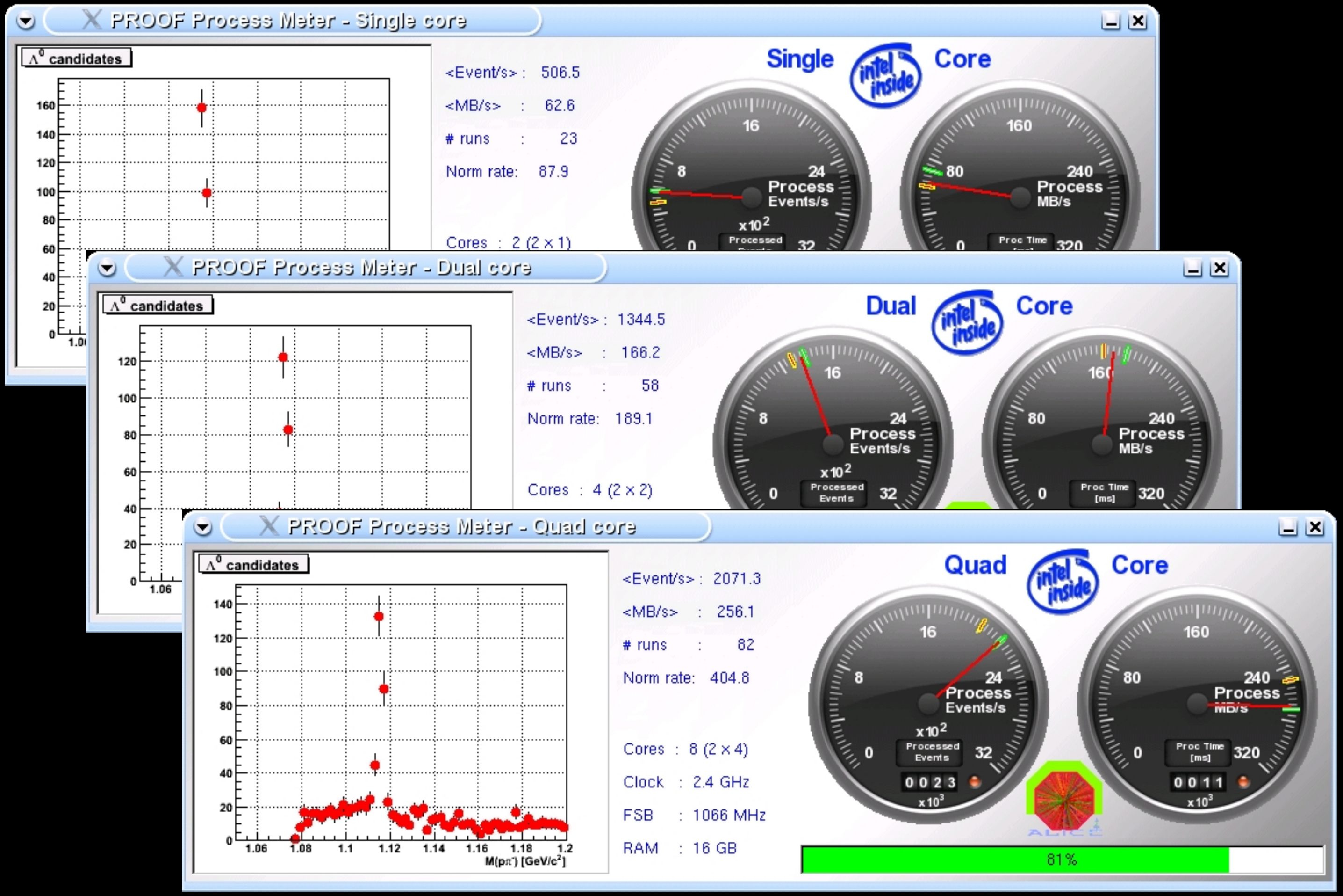
PROOF Scalability on Multi-Core Machines



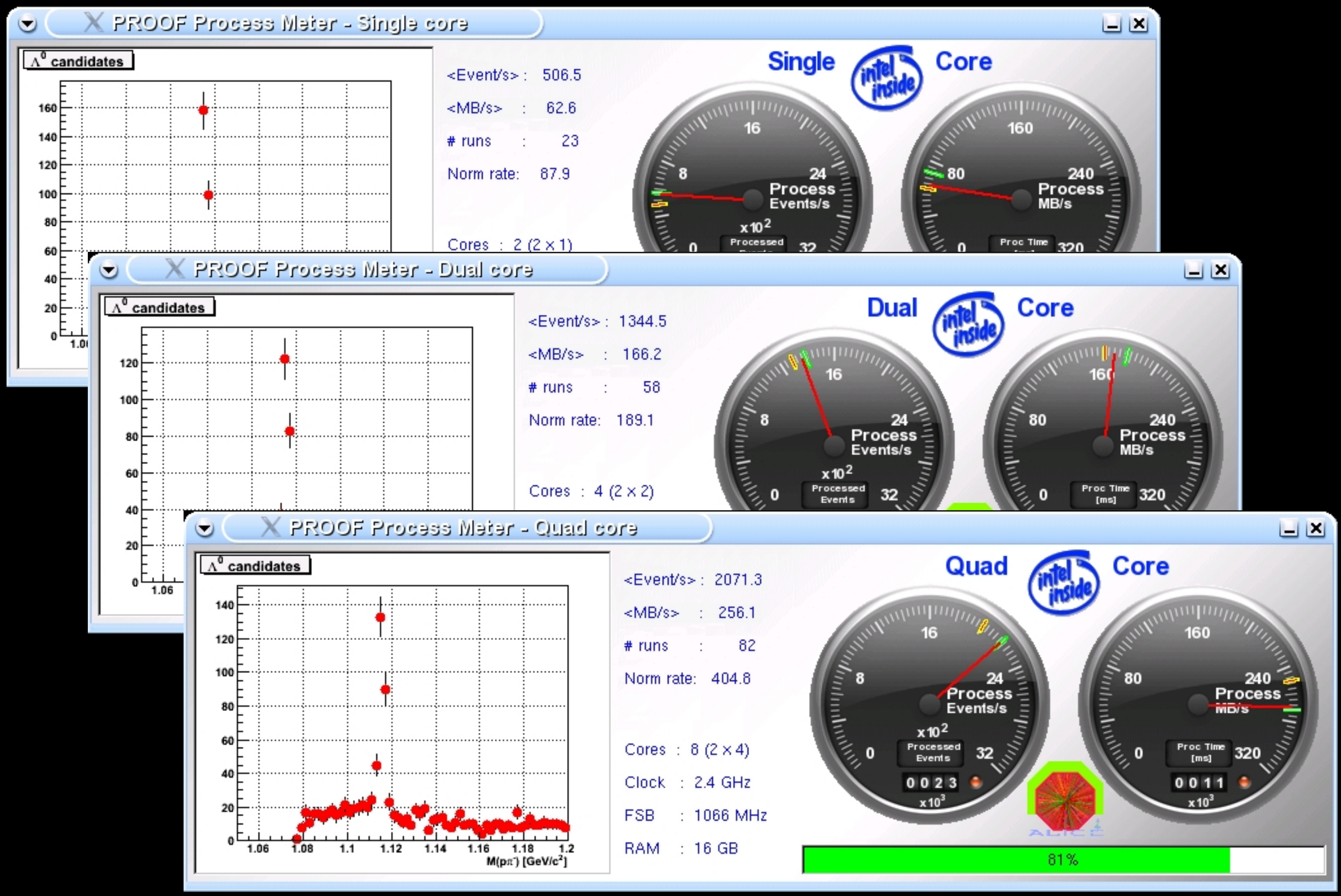
PROOF Scalability on Multi-Core Machines



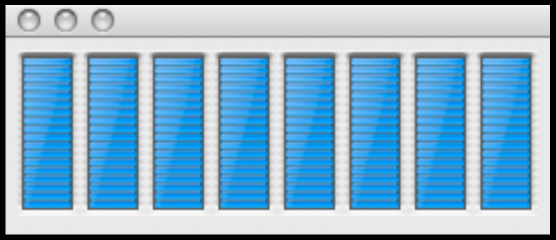
PROOF Scalability on Multi-Core Machines



PROOF Scalability on Multi-Core Machines



Running on MacPro with
dual Quad Core CPU's.



New Developments

- New Cling/LLVM based C++11 compliant just in time interpreter
- New browser based Javascript data access and display classes
- New iOS (iPad, iPhone, iPod) support
- New MacOS X native backend

Summary

- The ROOT system provides the common LHC data storage and analysis software infrastructure
- ROOT pioneered and provides an optimized vertical data store
- ROOT provides a set of first class statistical tools
- ROOT provides PROOF, a parallel and distributed analysis engine
- A whole bunch of new exciting developments in the pipeline